UWB MICROWAVE IMAGING VIA MODIFIED BEAMFORMING FOR EARLY DETECTION OF BREAST CANCER AND FPGA IMPLEMENTATION OF THE SIGNAL PROCESSING ALGORITHM

By

Siam Umar Hussain (0506014)
Mrinal Kanti Paul (0506022)
Md. Abdul Kader Sagar (0506023)

Supervised by

Dr. A. B. M. Harun-ur Rashid

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF SCIENCE AT
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING, BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY DHAKA, BANGLADESH JANUARY, 2011
The thesis entitled “Uwb Microwave Imaging Via Modified Beamforming For Early Detection Of Breast Cancer And Fpga Implementation Of The Signal Processing Algorithm” by Siam Umar Hussain (0506014), Mrinal Kanti Paul (0506022), Md. Abdul Kader Sagar (0506023) has been accepted in partial fulfillment of the requirement for the degree of bachelor science on 26 January, 2011.

Supervisor

Dr. A. B. M. Harun-ur Rashid
It is hereby declared that neither this thesis nor any part thereof has been submitted elsewhere for the award of any degree or diploma.

Author
Siam Umar Hussain

Author
Mrinal Kanti Paul

Author
Md. Abdul Kader Sagar

Supervisor
Dr. A. B. M. Harun-ur Rashid
TO OUR PARENTS
# Table of Contents

Table of Contents .......................... v
List of Illustrations ......................... vii
Abstract .................................. ix
Acknowledgements ........................ x

## Chapter 1: Introduction ................. 1
1.1 Background ........................... 1
1.2 Motivation ............................. 2
1.3 Overview ............................. 3

## Chapter 2: System modeling .......... 6
2.1 Dielectric contrast .................... 6
2.2 Debye model .......................... 8
2.3 FIT analysis .......................... 8
2.4 Breast model designing ............... 9

## Chapter 3: Antenna Modeling ....... 13
3.1 Ultrawideband (UWB) ................. 13
3.2 UWB Antenna ........................ 14
3.3 Rounded Bowtie Antenna ............. 15
3.4 Antenna Designing .................. 17
3.5 Cross Polarized Antenna Elements and Their Positioning .................. 19

## Chapter 4: Signal processing and image formation 21
4.1 Transmitted and received signal .... 22
4.2 Artifact removal ..................... 23
4.3 Beamforming ........................ 25
4.4 Image reconstruction ............... 29
4.4 Image reconstruction ............... 31
Chapter 5: Hardware Description  32
   5.1 FPGA and Its Architecture  32
   5.2 Programming the FPGA  33
   5.3 Configuration of the DE2 Board  34
   5.4 JTAG Programming  36
   5.5 Memory Operation  36
   5.6 IP Core  38

Chapter 6: Implementation Details  39
   6.1 Working with High Precision  39
   6.2 Circuit Description  42
      6.2.1 Artifact Remover  42
      6.2.2 Beamformer  43
   6.3 Results  47

Chapter 7: Conclusion  48
   7.1 Achievements  48
   7.2 Future Goals  49

Appendix A: MATLAB Codes  50

Appendix B: Schematics & Verilog Codes for FPGA Implementation  64
# List of Illustrations

## Chapter 2

**2.1** Summary of measured dielectric constant (a) and conductivity (b) data for normal and malignant breast tissue at radio and microwave frequencies. Four-term Cole–Cole parametric dispersion models for infiltrated fat and muscle are used to illustrate the extrapolation of measured data to higher frequencies  

**2.2** Patient orientation for the (a) planar and (b) cylindrical systems  

**2.3** Cross sectional view of the experimental 3-D model  

**2.4** Cross sectional view of the system from another angle

## Chapter 3

**3.1** Different types of UWB antennas- (a) Horn Antenna, (b) Vivaldi Antenna, (c) Dipole Antenna, (d) Bow-Tie Antenna  

**3.2** Rounded Bow-Tie Antenna  

**3.3** FIT model of the UWB antenna pair used in imaging, receiver and transmitter both consisting of a rounded-edge bow-tie element  

**3.4** Radiation pattern of the rounded bowtie elements at centre frequency, $f_c$ (in both polar and linear form)  

**3.5** Input reflection coefficient, $S_{11}$ of the rounded bowtie elements in free space  

**3.6** Different antenna positions

## Chapter 4

**4.1** Flowchart of the signal processing algorithm  

**4.2** Block diagram illustrating the artifact removal and beamforming technique for scan location $r_0$ in the breast
4.3 Artifact free tumor response of antennas No. 1-5 of the middle row

4.4 Diagram illustrating height and spherical coordinate distance

4.5 various layers in breast model in cross sectional view and different parameters extracted from it

4.6 Intensity for an image pixel calculation in beamformer

4.7 Block diagram illustrating the artifact removal and beamforming technique for scan location r0 in the breast

4.8 Pixel by pixel image formation of the intensity matrix

4.9 Color image of backscattered energy for the simulated breast model for xy-plane at zf = 20 mm, with delay-and-product beamforming (a) and traditional delay-and-sum beamforming (b). In both cases, tumor diameter is 1 mm and is placed at x = 56 mm, y = 26 mm

Chapter 5

5.1 Simplified Architecture of FPGA

5.2 A logic cell of FPGA

5.3 The block diagram of the DE2 board

5.4 JTAG configuration setup

5.5 SRAM Schematics

5.6 SRAM Pin Description

Chapter 6

6.1 Block diagram of the circuitry to store the product in mantissa and exponent format with radix 2

6.2 comparison of image matrix formed by FPGA and MATLAB

6.3 Artifact Remover

6.4 Beamformer

6.5 Operation of the CONTROL block of the beamformer

6.6 Image formed by (a) MATLAB, (b) FPGA
Abstract

Ultra-wideband (UWB) microwave imaging is a promising technique for detecting early stage breast cancer, which exploits the significant contrast in dielectric properties between normal and malignant breast tissues. In this research, we have proposed a new modified compensation method and beamforming technique for microwave imaging and implemented the system in FPGA. We used a three dimensional (3-D) Finite Integration Technique (FIT) based breast model, with normal breast tissue, supported on a layer of chest muscle and covered by a thin layer of skin. A small sized (1 mm diameter) tumor is placed within the breast tissue layer. A pair of rounded-edge bow-tie antennas at crossed position is used for transmitting and receiving microwave signals. This antenna pair is then placed at different positions over the breast surface and the incident and backscattered signal at each position are stored. Backscatters are then processed to eliminate artifacts. Finally they are passed through the beamformer and an image is formed. The beamformer is designed with adaptive weighting to compensate both propagation attenuation and lossy medium effect. Despite using the traditional delay-and-sum approach, new delay-and-product technique is used in beamforming. This modified beamforming approach is shown to outperform its previous counterparts in terms of resolution and sensitivity. For image formation by FPGA the backscattered signals are stored in memory after sampling and quantization. The total system is composed of two basic blocks: artifact remover and beamformer. Besides these, some extra circuitry is needed to store the image matrix in mantissa and exponent format with radix of 2 to have a high precision level as required by our algorithm. The image reconstructed by FPGA shows good compatibility with the simulated data.
Acknowledgement

We would like to express our sincere gratitude to our respected thesis supervisor Dr. A. B. M. Harun-ur Rashid for his constant guidance and helpful suggestion which have helped us to embark on this endeavor. We would never achieve what we have achieved if it was not for his tutelage.

We would also like to earnest thanks Mr. Upal mahbub, lecturer, Department of EEE, BUET for his cordial help regarding hardware implementation.

The laboratory official of VLSI lab and Microwave lab of department of EEE, BUET have been very helpful. So we would also like to thank the laboratory officials.

We express our earnest thanks to our friends for helping us at various stages of the thesis.

Of course, we are very much grateful towards our parents who have helped us at every stage of our life. Without their love, patience and belief on us we would never have succeeded in completion of this task.

BUET, Bangladesh

The Authors

January, 2011
Chapter 1

Introduction

1.1 Background

In recent years, breast cancer has been a great threat to women all over the world. Worldwide, breast cancer comprises 10.4% of all cancer incidences among women, making it the most common type of non-skin cancer in women and the fifth most common cause of cancer death [1]. According to American Cancer Society, about 1.3 million women are diagnosed with breast cancer annually and an alarming portion of them (about 465,000) meet their death [2]. For the third world countries the scenario is more severe and the mortality rate is much higher than the developed countries. Like, in the case of Bangladesh, about 70% of the diagnosed patients result in death.

With proper and timely screening the mortality rate can be reduced to great extent. Hence, a lot of work has been done to search for a more sensitive and robust system of diagnosis, specially a system that can detect malignancy even in the earliest stage. Alongside the typical clinical or self breast exams, different Computer Aide Diagnosis (CAD) systems (like mammography, ultrasound and magnetic resonance imaging) have been employed.

A clinical or self breast exam involves feeling the breast for lumps or other abnormalities. Research evidence does not support the effectiveness of either type of breast exam, because by the time a lump is large enough to be found it is likely to have been growing for several years and will soon be large enough to be found without an exam [3]. Mammographic screening for breast cancer uses x-rays to examine the breast for any uncharacteristic masses or lumps. The Task Force points out that in addition to unnecessary surgery and anxiety, the risks of more frequent mammograms include a small but significant increase in breast cancer induced by radiation [4].

Again, mammography is not generally considered as an effective screening technique for women less than 50 years old. A systematic review by the American College of Physicians concluded that, for women 40 to 49 years of age, the risks of mammography outweighed the benefits, and the US Preventive Services Task Force says that the
evidence in favor of routine screening of women under the age of 50, with radiographically dense breasts, is "weak"[5].

Medical ultrasonography is another diagnostic aid to mammography [6]. But because of its lower resolution and poor sensitivity mammography still remains to be the more preferred system.

Magnetic resonance imaging (MRI), on the other hand, has been shown to detect cancers not visible on mammograms[6]. A negative MRI can rule out the presence of cancer to a high degree of certainty, making it an excellent tool for screening in patients at high genetic risk or radiographically dense breasts, and for pre-treatment staging where the extent of disease is difficult to determine on mammography and ultrasound.

However, breast MRI has long been regarded to have disadvantages. For example, although it is 27–36% more sensitive, it has been claimed to be less specific than mammography [7]. As a result, MRI studies may have more false positives (up to 30%), which may have undesirable financial and psychological costs. It is also a relatively expensive procedure, and one which requires the intravenous injection of gadolinium, which has been implicated in a rare reaction called nephrogenic systemic fibrosis. Further, an MRI may not be used for screening patients with a pacemaker or breast reconstruction patients with a tissue expander due to the presence of metal.[9]

Consequently, the search goes on for a more preferred diagnosis system.

1.2 Motivation

Drawbacks of the existing diagnosis methods give motivation for finding new non-invasive, non-radioactive, low cost, yet comfortable and more sensitive methods. Hence, as an alternative to the existing methods comes the application of microwave in biological imaging, especially in the case of breast imaging. Actually some interesting properties of the breast tissue at microwave frequencies have drawn attention to this field.

Property 1: Microwaves interact with biological tissues primarily according to the tissue water content. This is a different interaction mechanism than for X-rays. The relevant physical properties contrast between malignant tumors and normal breast tissues is significantly greater for microwaves (5:1 in dielectric constant and 6:1 in conductivity) than for either X-rays or ultrasound, approaching an order of magnitude.
This large dielectric contrast causes malignant tumors to have significantly greater microwave scattering cross sections than normal tissues of comparable geometry.

Property 2: Microwave attenuation in normal breast tissue is less than 4 dB/cm up to 10 GHz[10]. This may permit existing microwave equipment having standard sensitivity and dynamic range to detect tumors located up to about 5 cm beneath the skin. The microwave attenuation and phase characteristic of normal breast tissue is such that constructive addition is possible for wide-bandwidth backscattered returns using UWB radar array techniques. These techniques suppress returns from spurious scatterers such as a vein interposed between the tumor and the surface of the breast.

Property 3: Unlike X-ray, the contrast between normal and malignant breast tissues in microwave does not change much with age. Even in the earliest stage the contrast is significantly high. This aids to the problem of early detection of tumors [11]-[12]. Moreover, microwave imaging technique would cause zero ionizing radiation exposure. It could be relatively comfortable since it would require access to only one side of the breast. It is also a comparatively low cost system. These safety and comfort features at a lower cost might facilitate the use of this technology for both frequent screening of the public and frequent monitoring of the progress of the treatment protocol for an individual patient. Hence comes the motivation to work with a microwave imaging technique that implies Ultrawideband (UWB) microwave pulses to detect early stage breast tumors.

Again, the statistics shown earlier for the third world countries motivate us to develop a low cost diagnosis system that can be made available for the mass use for regular screening and checkup. This requires hardware level implementation of the system. For the availability, low cost and robustness of the commercial FPGA chips, it offers a good scope for the hardware implementation of the algorithm.

1.3 Overview

Inspired from these facts, we set our research goal to develop a compact, low-cost diagnosis unit that detects tumor with good sensitivity. In our thesis we worked with developing a new way of microwave imaging that is simplified enough to implement in hardware level same time will be robust enough to detect tumors with good sensitivity. Then we worked with implementing the algorithm in hardware level (currently FPGA).
So, our work actually consisted of three basic parts. Firstly, we developed a model of the breast with malignancy (discussed in chapter 2) and designed an antenna configuration (discussed in chapter 3) for transmitting and receiving UWB microwave pulses. Secondly, we simulated the system in software level, collected the incident and received data and developed a modified tumor detection algorithm (discussed in chapter 3-4). And finally, we implemented the system with FPGA and checked the reliability of the hardware by comparing with the simulation (discussed in chapter 5-6).

While modeling, despite using the traditional analysis method of Finite Difference Time Domain (FDTD), we used Finite Integration Technique (FIT) in our simulation. Being a better estimation method for modeling complicated and irregular shapes (like human body), FIT was proven to be a better choice.

The antenna set used for transmitting and receiving is also improved. In spite of using the conventional UWB antennas, we used the rounded-edge bow-tie antenna [13]. It is compact like bow-tie, but being free from unwanted scattering at the corners it produces a smoother radiation pattern and better return loss. Again, the crossed arrangement of the transmitter-receiver elements is used to avoid unwanted scattering from the planar surfaces, while allowing cross-polarized reflections.

While imaging, despite looking for the dielectric distribution throughout the breast, as in the case of microwave tomography[14] or hybrid microwave-induced acoustic imaging[15]-[17], in this paper we concentrate on locating the malignant breast tumor using UWB radar technique. In this method, a set of antennas are placed at different locations on the breast and the backscattered signals are observed. Due to significant contrast between malignant and normal breast tissue, malignant tumor produces localized region of relatively large backscatter signals. The backscattered signals are then analyzed. The artifacts due to backscatters from skin and muscle regions are removed and the resultant signals are then beamformed to produce an image.

Previously, data-independent beamforming methods like Delay-And-Sum (DAS) [18] and Microwave Imaging via Space Time (MIST) beamforming [19] have been used. In this paper, we introduce a modified beamforming technique with data adaptive weighting. Here the weighting is made adaptive to compensate for the attenuation of the tumor response due to both propagation and lossy medium effect. Instead of using delay-and-sum approach in beamforming we used a delay-and-product approach. This
simplifies the post-beamforming power estimation technique and reduces the image noise level.

This simplified algorithm was then implemented in FPGA (in our case Altera Cyclone II) and tested for accuracy of the hardware system. The antenna response was sampled, quantized and stored in memory chip to be used by the FPGA to form an image of the breast plane. To meet the requirement of high precision with limited number of bits was a great challenge which we dealt with successfully. Our designed hardware gives result that is fairly close to the simulated result.
Chapter 2

System modeling

The main goal of our endeavor was to develop a novel system design for breast cancer detection, development of a novel signal processing algorithm which would be readily implementable in a hardware system. But before this, we have to design a system which correctly represents the original breast condition and its ambient material. We first aim to develop the algorithm based on simulation and then plan to implement it in hardware level. For modeling, we have exploited the dielectric property of the breast materials. As our frequency range is constrained to specific value, we know the dielectric behavior in that frequency range. For the simulation purpose we have used CST microwave studio [20] software. This software allows the user to assign user-defined material of particular dielectric constant and conductivity. In the subsequent sections we explain the detail of the method of system modeling, dielectric constant assigning and the model itself. First we go through the motive behind the use UWB in brief.

2.1 Dielectric contrast

As discussed earlier, the basis for microwave imaging was significant dielectric contrast between the malignant cancer cell and the fatty breast tissue. UWB radar techniques do not attempt to reconstruct the dielectric-properties profile, but instead seek to identify the presence and location of significant scatterers in the breast. Chaudhury et al[21] and Surowiec et al[22] showed the contrast in dielectric constant and conductivity. The enhanced dielectric properties of breast carcinomas appear to arise in part from increased protein hydration. The contrast is further enhanced by the vascularization of malignant tumors. As a result, malignant tumors have large microwave scattering cross-sections relative to comparably sized heterogeneity in normal breast tissue. Malignant breast tissues exhibit considerable increase in bound water content compared to the normal tissues and hence a high value of permittivity. When exposed to microwaves, the high water content of malignant breast tissues cause significant microwave scattering than normal fatty breast tissues that have low water content.
Fig. 2.1: Summary of measured dielectric constant (a) and conductivity (b) data for normal and malignant breast tissue at radio and microwave frequencies. Four-term Cole–Cole parametric dispersion models for infiltrated fat and muscle are used to illustrate the extrapolation of measured data to higher frequencies [2]-[3].

The dielectric contrast is depicted in figure 2.1. This figure is based on cole-cole dispersion model. The Cole-Cole equation is a dielectric relaxation model that constitutes a special case of Havriliak-Negami relaxation when the symmetry parameter ($\beta$) is equal to 1 - that is, when the relaxation peaks are symmetric:

$$\varepsilon'\omega - \varepsilon_{\infty} = \frac{\varepsilon_s - \varepsilon_{\infty}}{1 + (j\omega \tau)^{1-\alpha}}$$

(2.1)

Where $\varepsilon'$ is the complex dielectric constant, $\varepsilon_s$ and $\varepsilon_{\infty}$ are the "static" and "infinite frequency" dielectric constants, $\omega$ is the angular frequency and $\tau$ is a relaxation constant. The parameter $\alpha$, which takes a value between 0 and 1, is an experimentally
determined correction factor. When $\alpha = 0$, the Cole-Cole model reduces to the Debye model.

2.2 Debye model

While defining the material, the dispersive properties of the human tissues are introduced in the model. The single pole Debye model of the following form is used to calculate the frequency dependence of the dielectric constant $\varepsilon_r(\omega)$ and the conductivity $\sigma(\omega)$.

$$\varepsilon_r(\omega) - j \frac{\sigma(\epsilon)}{\omega \varepsilon_0} = \varepsilon_{\infty} + \frac{\varepsilon_s - \varepsilon_{\infty}}{1 + j \omega \tau} - j \frac{\sigma_s}{\omega \varepsilon_0} \cdots \cdots \cdots (2.2)$$

The parameter $\varepsilon_s$ is the static dielectric constant. $\varepsilon_{\infty}$ is the dielectric constant at infinity and $\sigma_s$ is the static conductivity and $\tau$ is the time constant. Debye relaxation is the dielectric relaxation response of an ideal, non-interacting population of dipoles to an alternating external electric field. It is usually expressed in the complex permittivity $\varepsilon$ of a medium as a function of the field's frequency $\omega$. The single-pole Mariya et al [23] have shown that Cole-Cole models provided an excellent fit to experimental data over the entire measurement frequency range. However, due to the computational complexity of embedding a Cole–Cole dispersion model into FIT, we were motivated to investigate the accuracy of simpler Debye models for capturing the frequency-dependent dielectric properties of breast tissue. Mariya et al have demonstrated that each two-pole Debye model is in excellent agreement with the corresponding Cole–Cole model from 0.5 to 20 GHz, and that each one-pole Debye model offers an excellent fit to the one-pole Cole–Cole data from 3.1 to 10.6 GHz. So we decipher that one pole Debye model perfectly matches our FCC allocated frequency range. This is why our decision to go with single pole Debye model was expedient.

2.3 FIT analysis

Instead of using Finite Difference Time Domain (FDTD) method, we have used Finite Integration Technique for our analysis. The finite integration technique (FIT) is a spatial discretization scheme to numerically solve electromagnetic field problems in
time and frequency domain. It preserves basic topological properties of the continuous equations such as conservation of charge and energy. FIT was proposed in 1977 by Thomas Weiland and has been enhanced continually over the years. [24] This method covers the full range of electromagnetics (from static up to high frequency) and optic applications and is the basis for commercial simulation tools [25].

The basic idea of this approach is to apply the Maxwell equations in integral form to a set of staggered grids. Maxwell’s equation in integral form,

\[ \phi \mathbf{E} \cdot d\mathbf{A} = \frac{a}{\varepsilon_0} = 4\pi k q \]  

(2.3)

This method stands out due to high flexibility in geometric modeling and boundary handling as well as incorporation of arbitrary material distributions and material properties such as anisotropy, non-linearity and dispersion. Furthermore, the use of a consistent dual orthogonal grid (e.g. Cartesian grid) in conjunction with an explicit time integration scheme (e.g. leap-frog-scheme) leads to compute and memory-efficient algorithms, which are especially adapted for transient field analysis in radio frequency (RF) applications

### 2.4 Breast model designing

Due to the vast variation in breast anatomy from patient to patient, it is hard to work with a realistic yet simplified model. To ease the problem, we consider a case when the patient is lying in supine position with her breasts naturally flat. And hence flat rectangular model of breast is used in our simulations. There are two possible physical orientations for test conditions which are shown in figure 2.2. As a more pragmatic approach we go for planar orientation or the supine position. The figure evinces the validity of our consideration of supine positioning for planar configuration.

Initially we used the 3-D model is shown in figure 2.3 The model consist of a skin layer of thickness 1 mm, a layer of fatty breast tissue of thickness 40 mm and then a muscle layer of thickness 3 mm. A spherical tumor of radius 0.5 mm is placed at location x = 56 mm, y = 26 mm, z = 24 mm. The dimension of the model is 150 mm × 40 mm.
Next, a pair of UWB transmitter-receiver antenna is placed at different locations over the breast surface. These locations are chosen in 3 rows, each with 10 equidistant points (placed 10 mm apart). These rows are separated by 10 mm from each other. The points in the middle row are 5 mm shifted from the corresponding points of the other two rows. This variation in alignment provides us with the benefits of random array radar technique. Thus, the attenuation of backscatters, coming at the null angle of one antenna, is compensated by antennas in adjacent rows.

As we are using UWB microwave imaging as we have described earlier, we will need proper dielectric constant apropos to the frequency. The best way to model the breast would be to assign proper dielectric behavior which will be enacted upon illuminating the breast with that frequency signal. As already discussed we have chosen the female patient to be in supine position and the breast will be almost rectangular in shape, a rectangular model will almost qualitatively produce the same

**Fig. 2.2: Patient orientation for the (a) planar and (b) cylindrical systems.**
result. As a generalization we have considered three individual layers for the model for simplification. Figure 2.3 illustrates the breast model with all the layers. The first layer if considered sequentially from outside of the body is the skin layer. The second layer and which is also the largest layer is the breast tissue. And the last layer is the muscle layer. And at the aforementioned described location the 0.5 mm tumor is placed which is also clearly shown figure 2.3 and 2.4. Now our task is to assign appropriate Debye parameters. The Debye parameters chosen for fatty breast tissue are \( \varepsilon_s = 10,\ varepsilon_n = 7,\ \sigma_s = 0.15 \text{ s/m},\ \tau = 7 \text{ ps} \). For tumor \( \varepsilon_s = 54,\ varepsilon_n = 4,\ \sigma_s = 0.7 \text{ s/m},\ \tau = 7 \text{ ps} \) and for skin \( \varepsilon_s = 37,\ varepsilon_n = 4,\ \sigma_s = 1.1 \text{ s/m},\ \tau = 7 \text{ ps} \). For muscles these parameters are \( \varepsilon_r = 12 \) and \( \sigma = 0.95 \text{ s/m} \). In Fig. 2(a)-(b) the frequency dependency of \( \varepsilon_r(\omega) \) and \( \sigma(\omega) \) are shown for different tissues [2]-[3]. The antennas used in our setup were designed for a central frequency of 6.65 GHz. At this central frequency \( \varepsilon_r \) for normal breast tissue is 9.76 and for tumor is 50.59, which represents a 5:1 contrast in dielectric constant. Due to this significant contrast, malignant tumors produce localized regions of relatively large backscatter energy, which after passing through the beamformer helps us locate the tumor.

![Figure 2.3: Cross sectional view of the experimental 3-D model](image-url)
Fig 2.4: cross sectional view of the system from another angle
Chapter 3

Antenna Modeling

3.1 Ultrawideband (UWB)

Ultra-wideband is a radio technology that can be used at very low energy levels for short-range high-bandwidth communications by using a large portion of the radio spectrum. It is a technology for transmitting information [26] spread over a large bandwidth (>500 MHz) that should, in theory and under the right circumstances, be able to share spectrum with other users. Regulatory settings of Federal Communications Commission (FCC) in United States are intended to provide an efficient use of scarce radio bandwidth while enabling both high data rate Personal Area Network (PAN) wireless connectivity and longer-range [26], low data rate applications as well as radar and imaging systems. UWB is also used in see-through-the-wall precision radar imaging technology [27], precision locating and tracking (using distance measurements between radios), and precision time-of-arrival-based localization approaches [28].

A significant difference between traditional radio transmissions and UWB radio transmissions is that traditional systems transmit information by varying the power level, frequency, and/or phase of a sinusoidal wave. UWB transmissions transmit information by generating radio energy at specific time instants and occupying large bandwidth thus enabling a pulse-position or time-modulation. The information can also be imparted (modulated) on UWB signals (pulses) by encoding the polarity of the pulse, the amplitude of the pulse, and/or by using orthogonal pulses. UWB pulses can be sent sporadically at relatively low pulse rates to support time/position modulation, but can also be sent at rates up to the inverse of the UWB pulse bandwidth.

One of the valuable aspects of UWB radio technology is the ability for a UWB radio system to determine "time of flight" of the direct path of the radio transmission between the transmitter and receiver at various frequencies. This helps to overcome multi path
propagation, as at least some of the frequencies pass on radio line of sight. With a cooperative symmetric two-way metering technique distances can be measured to high resolution as well as to high accuracy by compensating for local clock drifts and stochastic inaccuracies.

Another valuable aspect of pulse-based UWB is that the pulses are very short in space, so most signal reflections do not overlap the original pulse, and thus the traditional multipath fading of narrow band signals does not exist. However, there still is multipath propagation and inter-pulse interference for fast pulse systems which have to be mitigated by coding techniques.

Again multiple antenna schemes such as MIMO have been used to increase the system throughput and the reception reliability. Since UWB has almost impulse-like channel response, the combination with multiple antenna techniques is preferable as well. Coupling MIMO spatial multiplexing with UWB’s already high throughput gives the possibility of short-range networks with multi-gigabit rates.

### 3.2 UWB Antenna

Ultrawideband (UWB) technology, positioned as the cutting edge of research and development, paves the way to meet the emerging demands set by broadband wireless applications, such as high-speed data transmission, medical imaging, short-range radars, electromagnetic testing, etc. This breathtaking resource builds upon the basics of UWB technology to provide a complete compilation of figures of merit along with a vital state-of-the-art of the different antenna alternatives that are to be employed according to the specific application.

For UWB radar based approaches to breast imaging, several antenna structures have been introduced. Initially, resistively loaded bowtie and dipole antennas were introduced. Resistively loaded Vee dipoles [30] have also been proposed. Alternatives such as modified ridged horn antenna [31], patch antenna, vivaldi antenna etc. have also been introduced.
In our work, we used a modified form of the bowtie antenna, the rounded bowtie. Bowtie configuration provides compactness and the rounded structure provides an improved radiation pattern.

![Diagram of different types of UWB antennas](image)

**Figure 3.1:** Different types of UWB antennas- (a) Horn Antenna, (b) Vivaldi Antenna, (c) Dipole Antenna, (d) Bow-Tie Antenna

### 3.3 Rounded Bowtie Antenna

Modified dipole shapes are often used to obtain wide-band operation without increasing the complexity of the antenna. The rounded bow-tie antenna represents a fairly simple dipole variation, and provides good wide-band performance in spite of its simplicity.

This antenna is popular for frequencies ranging from UHF up to the millimeter wave range, and has also found application in arrays. The rounded bow-tie is closely related to the conventional (triangular) bow-tie; the rounding results in an impedance frequency response that is flatter than that of the regular bow-tie. For transient applications (i.e. when short duration pulses are used), bow-tie antennas with rounded
edges demonstrate better performance as reflections from the ends occur at the same time instant. Pulse radiation can be further improved by resistive loading. The rounded bow-tie antenna performance is not sensitive to small parameter variations, improving robustness to manufacturing tolerances.

The rounded bow-tie antenna can be seen as a modified dipole. A thin-wire dipole’s radiation is essentially formed by the superposition of the direct radiation of the feed (incident field) and strong diffractions from the wire ends. The magnitude and phase relationship between the incident and diffracted waves determine the pattern and impedance performance of the antenna. Due to the strong phase coherence between the various fields in a thin wire dipole, its performance is highly frequency dependent. By modifying the thin wire dipole into a bow-tie, extra sources of diffraction (the bow-tie edge and corners) that have weaker phase coherence are introduced, resulting in a broader operating bandwidth. By rounding the edge of the bow-tie, the path length from the feed to the edge of the bow-tie is the same in all directions, suppressing some higher order modes; this results in a flatter frequency response than that of the conventional (triangular) bow-tie.

The rounded bow-tie has good impedance behavior over a wide band. Due to the taper of the bow-tie, the VSWR does not degrade significantly as the frequency is increased, but the radiation characteristics at high frequencies may be unattractive. At the low end of the frequency spectrum, performance is chiefly limited by input impedance. At higher frequencies, the VSWR shows significant ripple, but remains below a certain value. VSWR is degraded by the use of thicker and higher permittivity substrates; behavior is seen that is qualitatively the same as for thin substrates, but the VSWR values at the peaks are higher. Thicker, higher permittivity substrates do, however, enable a slight reduction in the size of the antenna.

![Figure 3.2: Rounded Bow-Tie Antenna.](image)
3.4 Antenna Designing

In our work, we chose the rounded bowtie antenna elements for their compactness. Unlike the traditional co-polarized antennas we used a cross-polarized pair of the rounded bowtie elements. These elements are resistively loaded with 250Ω matched impedance. In figure 3.3 the top view of the 3-D FIT model of our antenna geometry is shown. Both bow-ties are designed to operate at the centre frequency of 6.65 GHz. Both have flare angle of 76.45° and arm length of 22.981 mm. The feed gap and feed width are set as 0.6781 mm and 0.7138 mm. Finally, the antenna is fabricated on the substrate Arlon Di-870 (\(\varepsilon_r = 2.33, \tan \delta = 0.0013\)) with thickness = 1.4276 mm.

![FIT model of the UWB antenna pair used in imaging, receiver and transmitter both consisting of a rounded-edge bow-tie element.](image)

A Gaussian Monocycle Pulse (GMP) is used as the input signal. The operating frequency range is chosen from 3.1-10.6 GHz (the FCC allocated frequency band for medical imaging) and the center frequency, \(f_c\), is set at 6.65 GHz. The antenna parameters mentioned above are all optimized for this centre frequency. The antenna designing tool Antenna Magus v1.0.2 was used for designing and optimizing antenna parameters.
In the following figures, two important features of the used bowtie elements are shown (in free space). In figure 3.4 the radiation pattern is shown at the centre frequency $f_c$ and in figure 3.5 the input reflection coefficient, $S_{11}$ is shown. Both show quiet good behavior for serving our purpose.

**Figure 3.4:** Radiation pattern of the rounded bowtie elements at centre frequency, $f_c$ (in both polar and linear form).
3.5 Cross Polarized Antenna Elements and Their Positioning

By exciting one antenna of a pair of perpendicular antenna elements forming a Maltese cross and receiving on the other antenna, the cross-polarized backscattered return from a tumor can be obtained. The cross-polarized backscatter of an axially symmetric tumor such as the spherical tumor positioned directly below the antenna feed point is exactly zero. However, when the antenna pair is positioned such that the tumor is off the central perpendicular axis of the two antennas there is a nonzero cross-polarized component of the tumor backscatter. The cross-polarized component is nonzero for any axially asymmetric tumor.

One of the key advantages of using the Maltese cross configuration is the reduction in backscatter from planar structures such as chest wall, while permitting observation of reflections from axially asymmetric tumors. These properties of the cross-polarized arrangement of the antenna pair encouraged us to use the rounded bowtie elements in cross orientation.

The cross pair is then positioned at different positions over the breast surface. Considering the length of the antennas, instead of using multiple antennas for tumor
localization, we used the same crossed pair at different positions. The positions are distributed in three equidistant rows with 10 equidistant locations each. The rows are spaced 10mm apart and antennas in each row are also spaced 10mm apart. The middle row is 5mm shifted to the right, to bring in some randomness in the positioning. This is done to exploit the benefits of the random array radar technique. Again, this configuration helps us to remove the null points in the model due to the limited shape of the antenna radiation pattern. Because with this configuration the null point of one antenna is observed by the antennas of the adjacent rows.

Figure 3.6: Different antenna positions.
Chapter 4
Signal processing and image formation

In the preceding chapters we have gone through various aspects of breast cancer detection system, antenna design and existing algorithms. It was previously mentioned that our goal is to develop an algorithm which not only is simple and offer better performance but also it has to implementable on low coast hardware system (for our case, FPGA). The existing algorithm [19] uses DFT and IDFT which are intricate to be implemented on a single chip. And if implemented, logic element will increase and subsequently the cost of the hardware system will escalate. So our goal was to develop an algorithm which is readily implementable on hardware circuitry and which is fast. We had to avoid complex mathematical calculation for the sake of simplicity. Therefore we have proposed a novel beamforming algorithm which not only offers simplicity but also better performance compared to other beamforming algorithm for breast cancer detection.

For obtaining the backscattered signal antenna pair was placed at different places. FIT domain analysis was performed on the model. For modeling and FIT domain analysis, the professional software CST Microwave Studio, 2009 was used in our experiment.

In figure 4.1 the whole signal processing and image reconstruction algorithm are described in brief. The whole process is described step by step. At first data acquisition is done. Then the data are averaged to remove artifact from the antenna data. The adaptive delay calculation is done for a particular antenna data. Then windowing is done based on the delay calculated. Intensity for a particular position is calculated and the image matrix is formed. We first discuss the sample transmitted and received signal.
4.1 Transmitted and received signal

We have used a Gaussian Signal of frequency range 3.1 -10.6 GHz in the antenna feed to illuminate the breast model. The backscattered signal is received in the other antenna of the crossed bowtie configuration. This received signal contains information about the tumor location as well as unwanted response from skin surface and other heterogeneities. The sample transmitted and received signal is shown in figure 4.2. As evident from the figure, the backscattered signal is time delayed and other responses are subsumed.
These signals are then stored and analyzed, artifacts are removed, compensations are made and finally the compensated tumor responses are subjected to the beamformer to produce an image.

### 4.2 Artifact removal

The dielectric constant and conductivity of the skin layer and chest wall muscle show good contrast with the fatty breast tissue. As a result the received signals contain backscatter from both these layers in their early and late time frames. As the skin layer is closest to the antenna pairs, its effect on the backscatter is the largest. There are also the reflections from the substrate.

Consider an array of antennas. Each received signal is converted to a sampled waveform, containing contributions from the skin-breast interface, clutter due to
heterogeneity in the breast, backscatter from possible lesions, and noise. The response from the skin-breast interface is orders of magnitude larger than the response from all other contributions and may persist in time beyond the time at which the lesion response occurs. Thus the skin-breast response must be removed prior to performing tumor detection. The skin-breast artifacts in the channels are similar but not identical due to local variations in skin thickness and breast heterogeneity. If the skin-breast artifact for all channels were identical, it could be removed by subtracting the average of the skin artifact across the channels from each channel. We have generalized this idea to compensate for channel-to-channel variation by estimating the skin-breast artifact in each channel as a filtered combination of the signal in all other channels. So to gain insight on the actual tumor response, these artifacts must be removed. As we are considering a flat breast model, at all locations, the antenna pair will face almost identical breast structures. And hence, these artifacts are considered to be almost equal.

Consider an array of \( M \) rows with \( N \) antenna locations in each row and denote the backscatter from the \( i^{th} \) antenna of the \( j^{th} \) row as \( \text{ant}_{ij}(t) \). Each backscattered signal is converted to a sampled waveform \( \text{ant}_{ij}(n) \). A reference signal \( \text{avg}_{ij}(n) \) is formed by averaging the other \( (N-1) \) antenna backscatters of the same row and then it is subtracted from \( \text{ant}_{ij}(n) \) to get the tumor response \( y_{ij}(n) \) signals.

\[
\text{avg}_{ij}(n) = \frac{1}{N-1} \sum_{k=1, k \neq i}^{N} \text{ant}_{kj}(n) \tag{4.1}
\]

\[
y_{ij}(n) = \text{ant}_{ij}(n) - \text{avg}_{ij}(n) \tag{4.2}
\]

Decomposing the received data at each location into two parts, tumor response \( t_{ij}(n) \) and artifacts \( s_{ij}(n) \),

\[
y_{ij}(n) = s_{ij}(n) + t_{ij}(n) - \text{avg}_{s_{ij}}(n) - \text{avg}_{t_{ij}}(n) \tag{4.3}
\]

\[
s_{ij}(n) \approx \text{avg}_{s_{ij}}(n) \tag{4.4}
\]

\[
y_{ij}(n) \approx t_{ij}(n) - \text{avg}_{t_{ij}}(n) \tag{4.5}
\]

Thus the tumor response is distorted by the \( \text{avg}_{t_{ij}}(n) \) term. This term represents the average of the tumor response over the other \( (N-1) \) channels. As these responses are not time aligned, their average is very small and hence can be ignored. Thus the tumor...
response becomes \( y_{ij}(n) \approx t_{ij}(n) \). These artifact free backscattered signals (of 1\textsuperscript{st} to 5\textsuperscript{th} antenna of the middle row) are shown in figure 4.3.

![Image showing artifact free tumor response](image)

**Fig. 4.3: Artifact free tumor response of antennas No. 1-5 of the middle row**

In this diagram the read mark shows the authenticity of our algorithm and experimental setup. The data set given here, when taken, the tumor was nearest to the ant23 position. In this diagram it is evident that, in the ant23 position the antenna receives large backscatter in the earliest time. And as the antenna position is moved away from the tumor, the large backscatter is received at a delayed time.

### 4.3 Beamforming

A microwave image of the breast model is formed by passing the compensated backscattered signal through the beamformer. Backscattered energy for each scan locations \( r_0 \) is obtained and hence a corresponding image is formed. For simplicity, 2-D images are formed at different heights \( z_0 \) from the antenna plane. For each antenna location \( r_{ij} \), \((r_0 - r_{ij})\) is calculated. By converting \((r_0 - r_{ij})\) to spherical co-ordinates the
distance \( d_{ij}(r_0) \) and angular orientation \( \phi_{ij}(r_0) \) and \( \theta_{ij}(r_0) \) of point \( r_0 \) with respect to each antenna location \( r_{ij} \) is observed.

![Diagram illustrating height and spherical coordinate distance](image)

**Fig 4.4: Diagram illustrating height and spherical coordinate distance**

Here \( d_{ij}(r_0) = |r_0 - r_{ij}| \). Dividing this distance term by the propagation velocity, the time delay between the antennas and the \( r_0 \) location can be obtained. In our algorithm instead of using a constant velocity term, we consider the variation of propagation speed in different layers. Hence the total delay,

\[
\tau_{ij}(r_o) = \tau_{a_{ij}}(r_o) + \tau_{s_{ij}}(r_o) + \tau_{f_{ij}}(r_o) \tag{4.6}
\]

\[
\tau_{ij}(r_o) = \frac{2}{c} \left[ d_{a_{ij}}(r_0) \sqrt{\varepsilon_a(f_c)} + d_{s_{ij}}(r_0) \sqrt{\varepsilon_s(f_c)} + d_{f_{ij}}(r_0) \sqrt{\varepsilon_f(f_c)} \right] \tag{4.7}
\]

Where, \( \tau_{a_{ij}}(r_o) \), \( \tau_{s_{ij}}(r_o) \) and \( \tau_{f_{ij}}(r_o) \) are the corresponding time delays in air, skin and fatty breast tissue layer. \( d_{a_{ij}}(r_0) \), \( d_{s_{ij}}(r_0) \) and \( d_{f_{ij}}(r_0) \) are the corresponding distance terms and \( \varepsilon_a(\omega_c) \), \( \varepsilon_s(\omega_c) \) and \( \varepsilon_f(\omega_c) \) are the corresponding dielectric constants.

Applying basic geometry,

\[
\frac{d_{ij}(r_0)}{z} = \frac{d_{a_{ij}}(r_0)}{z_a} = \frac{d_{s_{ij}}(r_0)}{z_s} = \frac{d_{f_{ij}}(r_0)}{z_f(z)} \tag{4.8}
\]

Where, \( z_a, z_s, z_f(r_0) \) are the average air-gap, skin thickness and depth of the scan location \( r_0 \) from the skin layer. Thus, if \( \Delta t \) is the sampling interval, the discrete time delay terms will be,

\[
n_{ij}(r_o) = \frac{\tau_{ij}(r_o)}{\Delta t} = \frac{2d_{ij}(r_0)}{z c \Delta t} \left[ z_a \sqrt{\varepsilon_a(\omega_c)} + z_s \sqrt{\varepsilon_s(\omega_c)} + z_f(z) \sqrt{\varepsilon_f(\omega_c)} \right] \tag{4.9}
\]
Artifact free backscatters from each antenna $y_{ij}(n)$ is then time gated at the intervals $n_{ij}(r_0)$ and from each signal a strip of $n_a$ samples is cut at the given interval. Where $n_a$ is the approximate duration of the transmitted Gaussian signal. This data strips $x_{ij}(r_0,k)$ are then compensated for propagation attenuation and attenuation due to lossy medium effect.

For the $ij$-th location of the antennas, the compensation factors are given by $K_{ij}(r_0) = |r_0 - r_{ij}|^2 \times |r_0 - r_{ij}|^2 = |r_0 - r_{ij}|^4$ and $S_{ij}(r_0) = \text{dir}(\phi_{ij}(r_0), \theta_{ij}(r_0))$, where $\text{dir}(\phi_{ij}(r_0), \theta_{ij}(r_0))$ is the simulated normalized directivity of the antenna pair subject to an average breast model at the direction $\phi_{ij}(r_0), \theta_{ij}(r_0)$.

The compensated signals are calculated as,

$$\tilde{x}_{ij}(r_0, k) = \frac{x_{ij}(r_0,k)K_{ij}(r_0)}{S_{ij}(r_0)} \hspace{2cm} \text{(4.10)}$$

Next, sample to sample product of these compensated data strips are taken,

$$P(r_0,k) = \prod_{i=1}^{M} \tilde{x}_{ij}(r_0, k) \hspace{2cm} \text{(4.11)}$$

When the tumor responses within the time gated signals are in phase, most of the samples of $P(r_0,k)$ are positive. The more out of phase these signal components are,
the more will be the number of negative samples in $P(r_0,k)$. Hence, as $r_0$ approaches actual tumor location, these time gated signals get more in phase and $P(r_0,k)$ becomes more positive, whereas, for points away from the tumor location $P(r_0,k)$ gets more and more negative.

For each location $r_0$, sum of the samples of $P(r_0,k)$ is calculated. This sum term correspond to the intensity of the backscatters from that point. Thus, by varying $r_0$, an intensity matrix $I(r_0)$ is formed.

$$I(r_0) = I(x_0, y_0, z_0) = \sum_{k=1}^{n_a} P(r_0,k) \quad \text{................................. (4.12)}$$

The whole beamforming process along with the artifact removal algorithm is illustrated in figure 4.222. This figure shown how the signal were averaged to get artifact free signal. How adaptive delay was calculated and windowing was done. The weightage addition and subsequent multiplication are done to get image intensity for a particular point in the image.
4.4 Image reconstruction

The intensity matrix is then compared with a previously stored average intensity matrix of the tumor free cases. Next, the negative values in \( I(r_0) \) are ignored and hence for each \( z_0 \) locations, a modified intensity matrix \( I'(r_0) \) or \( I'(x_0, y_0) \) is obtained. Variation in this intensity matrix is then converted into a RGB image. And thus an artifact free microwave image of the breast is obtained. In the next chapter we discuss and show the simulated result using our proposed algorithm and compare it with the existing algorithms.

Fig. 4.7: Block diagram illustrating the artifact removal and beamforming technique for scan location \( r_0 \) in the breast.
The final image formed by our modified beamforming technique is given in Figure 4.9(a). In this image, the point with the highest intensity is at $x = 56$ mm, $y = 27$ mm, where the actual tumor location is at $x = 56$ mm, $y = 26$ mm. We can see that the difference between the simulated and actual location is quite insignificant. To show the contrast between our method and the traditional DAS beamforming technique, we also applied the traditional approach on our simulated data. The resultant image is shown in Figure 4.9(b). Also from this figure, we can predict the tumor location. But the precision is much lower, artifact noise is higher and information about the tumor size is hard to get.

This result is comparatively better than the existing methods such as MIST and DAS method. Moreover, the above mentioned method use frequency domain transformation, inverse frequency domain transformation adaptive filtering. These processes are comparatively easy to be implemented on the software level. But to be practically implementable in a low cost device we have to have a system that fulfills the demand of simplicity and not being hardware intensive. Our algorithm satisfied this criterion.
perfectly and we also see that it has offered way better resolution than traditional DAS algorithm. It not only offers better resolution but also offers better precision.

Figure 4.9: Color image of backscattered energy for the simulated breast model for xy-plane at $z_t = 20$ mm, with delay-and-product beamforming (a) and traditional delay-and-sum beamforming (b). In both cases, tumor diameter is 1 mm and is placed at $x = 56$ mm, $y = 26$ mm.
Beamforming algorithm described in the previous chapter is implemented in an FPGA (Field Programmable Gate Array) chip. Altera’s DE2 board containing a cyclone II FPGA chip is used for this purpose. The SRAM on the DE2 board is used to store the response from the antennas and the intermediate as well as final results. The circuit is defined using Verilog HDL (Hardware Description Language) and FPGA is configured using Quartus II software.

5.1 FPGA and Its Architecture

FPGA is an integrated circuit that can be programmed by the end user. It can be used to implement any logical function that an application-specific integrated circuit (ASIC) could perform, but the ability to update the functionality in the field offers advantages for many applications. ASIC designs can also be prototyped in FPGA for hardware verification and early software development.
Figure 5.1 shows a simplified architecture of an FPGA chip. Its basic elements are programmable logic cells that can perform simple logic functions. These cells are interconnected by a set of routing wires and switching matrices. Generally, all the routing channels have the same width (number of wires). The I/O pads are used to feed binary input into the chip or monitor its output. Multiple I/O pads may fit into the height of one row or the width of one column in the array [32].

![Figure 5.2: A logic cell of FPGA](image)

Logic cells can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. A cell generally comprises of a Look Up Table (LUT), a Flip-Flop and a Mux as shown in figure 5.2. Each cell supports 3 to 10 binary inputs and 1 or 2 outputs. The LUT performs the logical operation as specified by the user. The Mux selects either clocked or simple combinational logic at the output.

**5.2 Programming the FPGA**

The FPGA chip is configured to perform the desired operation of the user. It is done by specifying the simple logic functions for each cell and selectively closing the switches in the matrix. Arrays of logic cells and switches form blocks of logic circuits. Complex designs are created by combining these blocks. The steps of configuration are described below.

1. The circuit to perform the desired operation is described using a HDL or schematic design. HDL is a computer language for formal description and design of electronic circuits, and most-commonly, digital logic. It can describe the circuit's operation, its design and organization, and tests to verify its operation by means of simulation. This form is easier to work with when
handling large structures as it is possible to just specify them numerically rather than having to draw every piece by hand. There are different HDLs like VHDL, Verilog or AHDL. Verilog is used in this design. On the other hand schematic entry allows easier visualization of design. Often a combination of these two is used to describe the complete operation as is done in our design.

2. The design is simulated to check the functionality. Generally several stages of simulation and redesign are needed to ensure desired functionality.

3. For the purpose of going from HDL or schematic to actual configuration, the source files are fed to a software suit where different compilation steps generate a file that can be fitted to the actual FPGA architecture. Here QuartusII version 9.1 is used for simulation and compilation. The generated file is transferred to the FPGA via a serial interface, JTAG in this case.

5.3 Configuration of the DE2 Board

Altera’s [33] DE2 board is used to implement the algorithm. The DE2 board has many features that allow the user to implement a wide range of designed circuits, from simple circuits to various multimedia projects. Figure 5.3 gives the block diagram of the DE2 board. The heart of the board is the Cyclone II FPGA chip. To provide maximum flexibility for the user, all connections are made through the FPGA. Thus, the user can configure the FPGA to implement any system design.

Following is more detailed information about the blocks used in our design [34]:

Cyclone II 2C35 FPGA

- 33,216 LEs
- 105 M4K RAM blocks
- 483,840 total RAM bits
- 35 embedded multipliers
- 4 PLLs
- 475 user I/O pins
- Fine Line BGA 672-pin package
Serial Configuration device and USB Blaster circuit

- Altera’s EPCS16 Serial Configuration device
- On-board USB Blaster for programming and user API control
- JTAG and AS programming modes are supported

SRAM

- 512-Kbyte Static RAM memory chip
- Organized as 256K x 16 bits
- Accessible as memory for the Nios II processor and by the DE2 Control Panel

Pushbutton switches

- 4 pushbutton switches
- Debounced by a Schmitt trigger circuit
- Normally high; generates one active-low pulse when the switch is pressed
Toggle switches

- 18 toggle switches for user inputs
- A switch causes logic 0 when in the DOWN (closest to the edge of the DE2 board) position and logic 1 when in the UP position

Clock inputs

- 50-MHz oscillator
- 27-MHz oscillator
- SMA external clock input

5.4 JTAG Programming

JTAG refers to the IEEE standards Joint Test Action Group. Figure 5.4 illustrates the JTAG configuration setup. In JTAG programming the configuration bit stream for the desired circuit is downloaded directly into the Cyclone II FPGA. DE2 board is connected to the host computer via a USB cable. FPGA will retain the configuration as long as power is applied to the board; the configuration is lost when the power is turned off [2].

Figure 5.4: JTAG configuration setup [2]

5.5 Memory Operation

The 512kB SRAM on the DE2 board is used to store the digitized antenna response, intermediate data and the final image matrix. The memory is organized as 256k x 16 bits. So each sample is stored as a 16 bit binary number. Figure 5.5 shows the schematic
diagram of the SRAM and table 5.1 gives the description of the pins shown on the figure [2].

![SRAM Schematics](image)

**Figure 5.5: SRAM Schematics [2]**

<table>
<thead>
<tr>
<th>Signal Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRAM_ADDR[17:0]</td>
<td>SRAM Address Bus[17:0]</td>
</tr>
<tr>
<td>SRAM_DQ[15:0]</td>
<td>SRAM Data Bus[15:0] input/output</td>
</tr>
<tr>
<td>SRAM_WE_N</td>
<td>SRAM Write Enable (Active Low)</td>
</tr>
<tr>
<td>SRAM_OE_N</td>
<td>SRAM Output Enable (Active Low)</td>
</tr>
<tr>
<td>SRAM_UB_N</td>
<td>SRAM High-byte Data Mask (Active Low)</td>
</tr>
<tr>
<td>SRAM_LB_N</td>
<td>SRAM Low-byte Data Mask (Active Low)</td>
</tr>
<tr>
<td>SRAM_CE_N</td>
<td>SRAM Chip Enable (Active Low)</td>
</tr>
</tbody>
</table>

**Table 5.1: SRAM Pin Description [2]**

Before any read/write operation the desired address is given on the address bus. Both read and write operation is done through the data bus. The write enable and output enable signals defines the direction of data flow. The write enable signal has a higher preference over the output enable signal [35]. After any write operation the data bus has to be buffered with both write enable and output enable signals inactive before any read operation. Figure 5.6 shows a sample memory operation read-write-read.
In the figure x denotes ‘don’t care’ and z denotes ‘high impedance’. The memory circuit operates in the positive edge while other circuits operate in the negative edge. This ensures stability of data during write operation as well as arithmetic and logical operations. The Verilog code for the memory interface is given in Appendix B.

5.6 IP Core

To simplify the design of complex systems in FPGAs there exist libraries of predefined complex functions and circuits that have been tested and optimized for best performance. These are called IP (Intellectual Property) cores. Altera delivers all IP cores, including the NiosII embedded processor, as an IP library that is built into the QuartusII software download file and installation process. The IP downloaded with QuartusII software includes Altera’s MegaCore library of licensed cores. The portfolio includes IP for digital signal processing (DSP), protocol, memory interface, and embedded processors and related peripherals [36]. In this design the ‘Square Root’ function from the MegaCore library is used to calculate the delay (Eq 4.9) of our delay and product beamforming algorithm. The Verilog code for the ‘Square Root’ function is given in Appendix B.
Chapter 6

Implementation Details

In chapter 3 we depicted that an array of 30 antennas are used to transmit UWB microwave Gaussian pulses through breast tissue and receive the reflected signals. These reflected signals are then sampled, quantized with 16 bits and stored in the SRAM of the DE2 board. The stored data is fetched to the FPGA chip via SRAM Interface to form the desired image by beamforming algorithm.

6.1 Working with High Precision

In chapter 5 it was shown that the amplitude of the received signal is of the order of $10^{-4}$ and the maximum value of the image matrix is of the order of $10^{-14}$. So to form the image we need a precision level of at least $10^{-10}$. With 16 bit samples we have a precision level of $2^{-16}$ or about $10^{-4}$. If we look from a different angle, to store the product of thirty 16 bit numbers 480 bits are needed, which is absurd in practical sense. Here the product of thirty signals is formed in thirty steps. Each time a product of two 16 bit numbers is formed, only the upper 16 bits of the 32 bit product is stored and the lower 16 bits are discarded. This is equivalent to dividing the product by $2^{16}$. As a result the product becomes trivial after five or six multiplication steps. To deal with this problem we need to add some extra circuitry to our basic design.

The main idea behind these circuitry is that in two’s complement format if the number of bits needed to hold a value is lower than the number of bits used to store it, the higher bits hold copies of the sign bit. So we can discard the redundant bits at left and pad zeros at right so that the product does not get lost. We also need to store the number of zeros padded as it will be the negative exponent with base 2 of the product. The product is now saved in the mantissa and exponent format [37] with radix 2. Figure 6.1 shows the block diagram of these circuitry for one pixel in the image matrix.

In the figure $P_i$ is the product formed at the $i$-th multiplication step and $P_{ij}$ is its $j$-th sample. Each time a product $P_{ij}$ is formed, it is XORed with its 1 bit left shifted.
Figure 6.1: Block diagram of the circuitry to store the product in mantissa and exponent format with radix 2 version. If two consecutive bits of $P_{ij}$ is same then their XOR will be 0. The position of the first 1 represents the effective MSB of $P_{ij}$ which is denoted as $F_{ij}$. The comparator then finds the maximum of these $F_{ij}$s which is denoted as $C_{ati}$. After the multiplication of all samples at the $i$-th step are complete $C_{ati}$ is fed to the Booster which fetches the product samples $P_{ij}$s from memory, right shifts them by $C_{ati}$ and stores again in the memory.

In this way, for the first step we have,

$$P_{ij} = K \times \text{dev}_{1j}$$ .................................................................................................................. (6.1),

where $\text{dev}_{1}$ is the windowed ARAD for the first antenna as described in chapter 5, $j$ is the sample index and $K$ is a constant. Then

$$P_{1j} = P'_{1j} \times 2^{-C_{at1}}$$ .................................................................................................................. (6.2)

$P'_{1j}$ is stored in the memory. For later steps,

$$P_{(i+1)j} = P'_{ij} \times \text{dev}_{(i+1)j}; i=1,2,.....29, j=1,2,........,N$$ ..............................................(6.3)

where $\text{dev}_i$ is the windowed ARAD for the $i$-th antenna and there are thirty antennas and $N$ is its length.

$$P_{(i+1)j} = P'_{(i+1)j} \times 2^{-C_{ati+1}}$$ ................................................................. (6.5)
As Cat is the exponent, it is summed after each multiplication step in the summer. The final output of the summer is

\[ \text{Exp} = \sum_{i=1}^{30} \text{Cat}_i \] \hspace{1cm} (6.6)

The final product after thirty multiplication steps is

\[ P_{\text{Fin},j} = \text{P}_{30,j} \times 2^{-\text{Exp}} \] \hspace{1cm} (6.7)

In the end, P\text{'}_{30,j} and Exp are the only two values that are needed to hold the products. As discussed in chapter 5, we need to sum up all the samples to have one pixel in the image matrix. The sum is

\[ S = \sum_{j=1}^{N} P_{\text{Fin},j} = 2^{-\text{Exp}} \times \sum_{j=1}^{N} \text{P}_{30,j} = 2^{-\text{Exp}} \times S' \] \hspace{1cm} (6.8)

\[ S' = \sum_{j=1}^{N} \text{P}_{30,j} \] \hspace{1cm} (6.9)

Finally only S\text{'} (mantissa) and Exp (exponent) are stored in the memory requiring 32 bits for each pixel with a very small data loss. These steps are repeated for every pixel on the image matrix. In figure 6.2 a portion of the straightened image matrix formed with FPGA is compared with the one formed with MATLAB. We can see that the shapes of both matrices are almost identical which implies a very small data loss.

![Comparison of image matrix formed by FPGA and MATLAB](image.png)

**Figure 6.2: comparison of image matrix formed by FPGA and MATLAB**

The above circuitry is spread over three blocks of the beamformer as described in the next section.
6.2 Circuit Description

The complete operation from the backscattered signals to the formation of image comprises of two basic blocks, the artifact remover and the beamformer. The first part extracts the tumor response from the received signals and the second part forms the image from this response. The Verilog [32], [38]-[39] code for the circuit is given in Appendix B.

6.2.1 Artifact Remover

A closer look at the block diagram of the artifact remover in figure 5.x reveals that it needs only two basic blocks – an ADDER/SUBTRACTOR and a SUMMER to perform the operations. Figure 6.3 shows these blocks in details.

![Figure 6.3: Artifact Remover](image)

The ADDRESS REGISTERS on each block hold the read and write addresses as well as the start and end addresses of different datasets. The CTR signal from the SYNCHRONIZER controls the update of the address registers on both blocks after each operation to decide the set of antenna data being processed.

With simple manipulation Eq(4.2) can be expressed respectively as

\[
\begin{align*}
    y_{ij} &= a_{nt_{ij}} - a_{vg_{ij}} \quad \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cdots \cd - 42 -
\[ \text{dev}_{ij} = y_{ij} + s\text{dev}_{ij} \]
\[ \Rightarrow \text{dev}_{ij} = y_{ij} + \frac{1}{N-1} \sum_{k=1}^{N} y_{kj} \]
\[ \Rightarrow \text{dev}_{ij} = y_{ij} + \frac{1}{N-1} \sum_{k=1}^{N} y_{kj} - \frac{1}{N-1} y_{ij} \]
\[ \Rightarrow \text{dev}_{ij} = \frac{N-2}{N-1} y_{ij} + \frac{1}{N-1} \sum_{k=1}^{N} y_{kj} \] .............................. (6.10)

It shows that instead of using different average values for different values of only one average can be used if the input signal is multiplied by a constant gain. The ADDER/SUBTRACTOR block incorporates a gain unit which changes its gain value and polarity during different operations under the control of the CTR signal. The SUMMER block calculates the sum of the input data and multiplies it by a constant gain to find the average. Each block informs the SYNCHRONIZER about the completion of an operation through their STG signals and the SYNCHRONIZER enables the blocks accordingly.

6.2.2 Beamformer

The beamformer implements our proposed Delay and Product algorithm. This part takes the Artifact Removed Antenna Data (ARAD) as input and forms images for a plain at a particular depth in the breast tissue with the Energy Concentration (EC) of each point on the plain as pixels. The ECs are stored in mantissa and exponent format with a radix of two e.g. mantissa*2^{exponent}. Besides the basic blocks - MULTIPLIER, SUMMER and CONTROLLER, there is a BOOSTER block that facilitates the beamformer to have a high precision level. Figure 6.4 shows these blocks.

6.2.2.1 Controller

The CONTROLLER block synchronizes the operations of all the blocks and calculates the delay (Eq 4.9) for our proposed Delay-and-Product beamforming. At first it initiates the operation for one point on the breast plain by asserting the INIT signal. The delay for any multiplication operation is calculated within this block and provided to the multiplication block. The MULTIPLIER block, the BOOSTER block and the SUMMER block sends signal to this CONTROLLER block after their operation is finished and CONTROLLER block starts the next operation according to the flow.
Figure 6.4: Beamformer

diagram shown at Figure 6.5. As all the blocks are connected to the buses the CONTROLLER block ensures that only one block is enabled at a time.

6.2.2.2 Multiplier

The MULTIPLIER block forms the products of the windowed tumor responses (Eq…). The delay value from the DELAY UNIT is fed to the ADDRESS REGISTERS of the MULTIPLIER block. Here the delay is added to the start and end addresses so that the window is shifted to the right. At the start of the operation, for every point on the breast plains SYNCHRONIZER sets the INIT signal and the MULTIPLIER initializes the product array with a constant value. The PRODUCT unit has two sixteen bit registers on the input side and one thirty two bit register on the output side to hold the product. The product is subject to the MSB-FIND unit which finds its most significant bit. The MAX unit finds the highest value of MSB of all the products of the array for one multiplication stage. The difference between this MSBmax and thirty two is output from the MAX unit to be fed to the BOOSTER and SUMMER blocks. Only higher sixteen bits of the thirty-two bit product are stored.
Figure 6.5: Operation of the CONTROL block of the beamformer
6.2.2.3 Booster

The BOOSTER block receives the CAT value from the MULTIPLIER block. It fetches the product stored by the MULTIPLIER from the SRAM, right shifts them CAT times and store again at the same address in the memory.

6.2.2.4 Summer

The SUMMER block incorporates two summing units. The first unit calculates the sum of the MSBmax values from the MULTIPLIER block. The STGM signal acts as the clock to this unit. This sum is the negative exponent of the EC for the corresponding point. The mantissa of the EC is the sum of the product array stored in the SRAM (Eq. ). The second unit operates with the system clock to find this sum. The SUMMER block is the last stage of calculating the energy concentration for one point in the breast plains. The STGS signal from this block initiates the operation for the next point.

6.2.2.5 Resource Usage

Table 6.1 shows the resource usage summary for the design

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Total logic elements</td>
<td>2778</td>
</tr>
<tr>
<td>Total registers</td>
<td>904</td>
</tr>
<tr>
<td>Total pins</td>
<td>42</td>
</tr>
<tr>
<td>Embedded Multiplier 9-bit elements</td>
<td>10</td>
</tr>
<tr>
<td>Maximum Frequency</td>
<td>41 MHz</td>
</tr>
<tr>
<td>Minimum time</td>
<td>6 s</td>
</tr>
<tr>
<td>Memory</td>
<td>25.15kB</td>
</tr>
</tbody>
</table>

Table 6.1: Resource Usage
6.3 Results

Figure 6.2 shows that there is very small difference between the image matrix formed by FPGA and that formed by MATLAB simulation. This small difference is hardly visible when the image is plotted. So the final image looks similar to the one showed in figure 4.9. Here the images formed by FPGA and by MATLAB are displayed in a different style.

![Image](image.png)

Figure 6.6: Image formed by (a) MATLAB, (b) FPGA

In the figure the white portion have pixels with higher values than the remaining area. This means that the reflected signal from that portion have higher energy. So this is the location of the tumor in the respective plane of the breast tissue. This white portion is slightly smaller in the image formed by FPGA. This is due to the small data loss for finite precision. Moreover, the image formed by FPGA has lower resolution as we down sampled the data before storing in memory to reduce both memory requirement and operation time. Still this image shows the location of the tumor successfully.
In this research, we have proposed and demonstrated a new beamforming technique for detection of early stage breast tumors. Our results show that this method is exceptionally robust and noise immune. We designed sophisticated breast models and antennas to simulate the response of breast tissue to UWB microwave Gaussian pulses. When fed to our proposed beamforming algorithm the response generates an image of the breast plane showing the position of the tumor cell. The calculated position has negligibly small difference with the actual position. We also implemented our algorithm in an FPGA chip and found results that have excellent match with the simulated results.

7.1 Achievements

A number of achievements of our research can be noted. First, we have introduced new ideas like Delay and Product beamforming, adaptive delay calculation and adaptive weighting. The Delay and Product beamforming is shown to have better performance over traditional Delay and Sum technique. The delay is calculated adaptively with the location of the breast tissue under consideration. Propagation attenuation and lossy medium effect both have been considered for compensation by adaptive weighting, providing better sensitivity.

In addition to providing better performance, this method is simpler than most of the existing methods. The artifact removal and power estimation algorithms both have been kept simple to avoid unnecessary complexities. It needs no Fourier transformation or adaptive filtering as needed in DAS or MIST. This makes the technique more feasible for hardware level implementation.

We also improved the antenna design and positioning. We used cross arrangement of transmitter- receiver pair and a rounded Bow-Tie antenna. It is shown to produce smoother radiation pattern and better return loss. The middle row of antennas is shifted
to exploit benefits of random array radar technique and remove null points in the breast model.

Due to our modified beamforming and improved compensation techniques, this method generates a high resolution image of the breast plane. Tumors up to 1 mm diameter can significantly be detected on the image.

Finally, we implemented our proposed algorithm in an FPGA chip. Our design provides similar result as generated by simulation. This proves feasibility of our algorithm to be used in real world opening the chance of portable and cheaper breast cancer diagnostic units. We successfully dealt with the requirement of high precision with limited number of bits in the digital domain.

### 7.2 Future Goals

We want to work on our beamforming algorithm for further improvement in accuracy and resolution, while still being simple enough for practical implementation. There is also scope for improving the breast model to make it more realistic, by adding other tissue layers. And obviously, our ultimate goal is to apply the algorithm in real life.

Our designed chip in FPGA needs to be optimized for power and speed. The final design will be implemented in ASIC for mass production. Finally, we want to implement the whole system within a portable & low cost unit containing antenna, LNA, ADC, memory, the ASIC chip and a display to show the image. We wish to design LNA for this specific application. The design of ADCs for the high frequency signals used here is also a challenging task.
Appendix A

MATLAB Codes

A1. MATLAB Code for Simulation

Final13.m

clear all
close all
clc
run([pwd,'\data5.m'])
run([pwd,'\directivity.m'])
load c

len=150;
wid=40;
hight=40;
hskin=1;
hmuscle=1;
hair=3;

tfeed=.5929;%ns  .7628-.1699
fact=0;
ttol=tfeed*fact;
xant=[30:10:120;35:10:125;30:10:120];
yant=[30 20 10];
zant=0;
ht=20;
x=0:len;
y=0:wid;
z=hair+hskin+ht;

tedd=interp1(fed(:,1),fed(:,2),time(min(find(time>=.1699)):max(find(time<=.7628)),1));
im=zeros(length(x),length(y));

for i=20:length(x)-20
    for j=10:length(y)-10
        rx=x(i);ry=y(j);rz=z;
        for l=1:3
            for k=1:length(xant)
                rx0=xant(1,k);ry0=yant(l);rz0=zant;
                [fi,thta,dist] = cart2sph(rx-rx0,ry-ry0,rz-rz0);
                dist=dist*2;
                %dist2(k,l)=dist;
                fi=fi/pi*180;
                if(fi<0)
                    fi=360+fi;
                end
                if(mod(fi,1)>=.5)
                    fi=fi-mod(fi,1)+1;
                else
                    fi=fi-mod(fi,1);
                end
            end
        end
    end
end
thta=90-thta/pi*180;
if (mod(thta,1)>=.5)
    thta=thta-mod(thta,1)+1;
else
    thta=thta-mod(thta,1);
end

drctvt=nptot(fi*181+thta+1);
tt=dist/c(l,k);%%%/(9+ht*10^.5);%%%c(l,k);
tt1=tt-ttol;
tt2=tt+tfeed+ttol;
if(k==1 && l==1)
    nn=max(find(time<=tt2))-min(find(time>=tt1));
    sss=ones(1,nn+1);
end

sdev=dev(min(find(time>=tt1)):min(find(time>=tt1))+nn,l,k)'/drctvt;
    sss=sss.*sdev;
end

sumsss=sum(sss);
if(sumsss>0)
    im(i,j)=sumsss;
end
end

figure(1)
contourf(im')
figure(2)
imshow(im/ max(max(im)))

% data5.m

clear all
clc
close all

run([pwd, '\feed.m'])
run([pwd, '\none.m'])

run([pwd, '\a1.m'])
run([pwd, '\b1.m'])
run([pwd, '\c1.m'])
run([pwd, '\d1.m'])
run([pwd, '\e1.m'])
run([pwd, '\f1.m'])
run([pwd, '\g1.m'])
run([pwd, '\h1.m'])
run([pwd, '\i1.m'])
run([pwd, '\j1.m'])

trr1=min([ant1(end,1) ant2(end,1) ant3(end,1) ant4(end,1) ant5(end,1)
ant6(end,1) ant7(end,1) ant8(end,1) ant9(end,1) ant10(end,1)]);
tll1=[ant1(end,1) ant2(end,1) ant3(end,1) ant4(end,1) ant5(end,1)
ant6(end,1) ant7(end,1) ant8(end,1) ant9(end,1) ant10(end,1)];
tloc1=find(tll1==trr1);
tl=eval(['ant' int2str(tloc1(1)) '(:,1)']);

run([pwd, '\a2.m'])
run([pwd, '\b2.m'])
run([pwd,'\c2.m'])
run([pwd,'\d2.m'])
run([pwd,'\e2.m'])
run([pwd,'\f2.m'])
run([pwd,'\g2.m'])
run([pwd,'\h2.m'])
run([pwd,'\i2.m'])
run([pwd,'\j2.m'])

trr2=min([ant21(end,1) ant22(end,1) ant23(end,1) ant24(end,1) ant25(end,1) ant26(end,1) ant27(end,1) ant28(end,1) ant29(end,1) ant210(end,1)]);
tll2=[ant21(end,1) ant22(end,1) ant23(end,1) ant24(end,1) ant25(end,1) ant26(end,1) ant27(end,1) ant28(end,1) ant29(end,1) ant210(end,1)];
tloc2=find(tll2==trr2);
t2=eval(['ant2' int2str(tloc2(1)) '(:,1)']);

run([pwd,'\a3.m'])
run([pwd,'\b3.m'])
run([pwd,'\c3.m'])
run([pwd,'\d3.m'])
run([pwd,'\e3.m'])
run([pwd,'\f3.m'])
run([pwd,'\g3.m'])
run([pwd,'\h3.m'])
run([pwd,'\i3.m'])
run([pwd,'\j3.m'])

trr3=min([ant31(end,1) ant32(end,1) ant33(end,1) ant34(end,1) ant35(end,1) ant36(end,1) ant37(end,1) ant38(end,1) ant39(end,1) ant310(end,1)]);
tll3=[ant31(end,1) ant32(end,1) ant33(end,1) ant34(end,1) ant35(end,1) ant36(end,1) ant37(end,1) ant38(end,1) ant39(end,1) ant310(end,1)];
tloc3=find(tll3==trr3);
t3=eval(['ant3' int2str(tloc3(1)) '(:,1)']);

trrf=min([t1(end) t2(end) t3(end)]);
tllf=[t1(end) t2(end) t3(end)];
tlocf=find(tllf==trrf);
time=eval(['t' int2str(tlocf(1))]);

non=interp1(non(:,1),non(:,2),time(:,1));
ant1=interp1(non(:,1),ant1(:,2),time(:,1))-non;
ant2=interp1(non(:,1),ant2(:,2),time(:,1))-non;
ant3=interp1(non(:,1),ant3(:,2),time(:,1))-non;
ant4=interp1(non(:,1),ant4(:,2),time(:,1))-non;
ant5=interp1(non(:,1),ant5(:,2),time(:,1))-non;
ant6=interp1(non(:,1),ant6(:,2),time(:,1))-non;
ant7=interp1(non(:,1),ant7(:,2),time(:,1))-non;
ant8=interp1(non(:,1),ant8(:,2),time(:,1))-non;
ant9=interp1(non(:,1),ant9(:,2),time(:,1))-non;
ant10=interp1(non(:,1),ant10(:,2),time(:,1))-non;
sum1=ant1+ant2+ant3+ant4+ant5+ant6+ant7+ant8+ant9+ant10;
avg1=sum1/10;%average

dev(:,1,1)= ant1-(sum1-ant1)/9;%integral
dev(:,1,2)= ant2-(sum1-ant2)/9;
\[ \text{dev}(:,1,3) = \text{ant3} - \left( \text{sum1} - \text{ant3} \right) / 9; \]
\[ \text{dev}(:,1,4) = \text{ant4} - \left( \text{sum1} - \text{ant4} \right) / 9; \]
\[ \text{dev}(:,1,5) = \text{ant5} - \left( \text{sum1} - \text{ant5} \right) / 9; \]
\[ \text{dev}(:,1,6) = \text{ant6} - \left( \text{sum1} - \text{ant6} \right) / 9; \]
\[ \text{dev}(:,1,7) = \text{ant7} - \left( \text{sum1} - \text{ant7} \right) / 9; \]
\[ \text{dev}(:,1,8) = \text{ant8} - \left( \text{sum1} - \text{ant8} \right) / 9; \]
\[ \text{dev}(:,1,9) = \text{ant9} - \left( \text{sum1} - \text{ant9} \right) / 9; \]
\[ \text{dev}(:,1,10) = \text{ant10} - \left( \text{sum1} - \text{ant10} \right) / 9; \]
\[ \text{sdev1} = \text{dev}(:,1,1) + \text{dev}(:,1,2) + \text{dev}(:,1,3) + \text{dev}(:,1,4) + \text{dev}(:,1,5) + \text{dev}(:,1,6) + \text{dev}(:,1,7) + \text{dev}(:,1,8) + \text{dev}(:,1,9) + \text{dev}(:,1,10); \]
\[ \text{dev}(:,1,1) = \text{dev}(:,1,1) + \left( \text{sdev1} - \text{dev}(:,1,1) \right) / 9; \]
\[ \text{dev}(:,1,2) = \text{dev}(:,1,2) + \left( \text{sdev1} - \text{dev}(:,1,2) \right) / 9; \]
\[ \text{dev}(:,1,3) = \text{dev}(:,1,3) + \left( \text{sdev1} - \text{dev}(:,1,3) \right) / 9; \]
\[ \text{dev}(:,1,4) = \text{dev}(:,1,4) + \left( \text{sdev1} - \text{dev}(:,1,4) \right) / 9; \]
\[ \text{dev}(:,1,5) = \text{dev}(:,1,5) + \left( \text{sdev1} - \text{dev}(:,1,5) \right) / 9; \]
\[ \text{dev}(:,1,6) = \text{dev}(:,1,6) + \left( \text{sdev1} - \text{dev}(:,1,6) \right) / 9; \]
\[ \text{dev}(:,1,7) = \text{dev}(:,1,7) + \left( \text{sdev1} - \text{dev}(:,1,7) \right) / 9; \]
\[ \text{dev}(:,1,8) = \text{dev}(:,1,8) + \left( \text{sdev1} - \text{dev}(:,1,8) \right) / 9; \]
\[ \text{dev}(:,1,9) = \text{dev}(:,1,9) + \left( \text{sdev1} - \text{dev}(:,1,9) \right) / 9; \]
\[ \text{dev}(:,1,10) = \text{dev}(:,1,10) + \left( \text{sdev1} - \text{dev}(:,1,10) \right) / 9; \]

ant21 = interp1(ant21(:,1), ant21(:,2), time(:,1)) - non;
ant22 = interp1(ant22(:,1), ant22(:,2), time(:,1)) - non;
ant23 = interp1(ant23(:,1), ant23(:,2), time(:,1)) - non;
ant24 = interp1(ant24(:,1), ant24(:,2), time(:,1)) - non;
ant25 = interp1(ant25(:,1), ant25(:,2), time(:,1)) - non;
ant26 = interp1(ant26(:,1), ant26(:,2), time(:,1)) - non;
ant27 = interp1(ant27(:,1), ant27(:,2), time(:,1)) - non;
ant28 = interp1(ant28(:,1), ant28(:,2), time(:,1)) - non;
ant29 = interp1(ant29(:,1), ant29(:,2), time(:,1)) - non;
ant210 = interp1(ant210(:,1), ant210(:,2), time(:,1)) - non;
\[ \text{sum2} = \text{ant21} + \text{ant22} + \text{ant23} + \text{ant24} + \text{ant25} + \text{ant26} + \text{ant27} + \text{ant28} + \text{ant29} + \text{ant210}; \]
\[ \text{avg2} = \text{sum2} / 10; \]
\[ \text{dev}(:,2,1) = \text{ant21} - \left( \text{sum2} - \text{ant21} \right) / 9; \]
\[ \text{dev}(:,2,2) = \text{ant22} - \left( \text{sum2} - \text{ant22} \right) / 9; \]
\[ \text{dev}(:,2,3) = \text{ant23} - \left( \text{sum2} - \text{ant23} \right) / 9; \]
\[ \text{dev}(:,2,4) = \text{ant24} - \left( \text{sum2} - \text{ant24} \right) / 9; \]
\[ \text{dev}(:,2,5) = \text{ant25} - \left( \text{sum2} - \text{ant25} \right) / 9; \]
\[ \text{dev}(:,2,6) = \text{ant26} - \left( \text{sum2} - \text{ant26} \right) / 9; \]
\[ \text{dev}(:,2,7) = \text{ant27} - \left( \text{sum2} - \text{ant27} \right) / 9; \]
\[ \text{dev}(:,2,8) = \text{ant28} - \left( \text{sum2} - \text{ant28} \right) / 9; \]
\[ \text{dev}(:,2,9) = \text{ant29} - \left( \text{sum2} - \text{ant29} \right) / 9; \]
\[ \text{dev}(:,2,10) = \text{ant210} - \left( \text{sum2} - \text{ant210} \right) / 9; \]
\[ \text{sdev2} = \text{dev}(:,2,1) + \text{dev}(:,2,2) + \text{dev}(:,2,3) + \text{dev}(:,2,4) + \text{dev}(:,2,5) + \text{dev}(:,2,6) + \text{dev}(:,2,7) + \text{dev}(:,2,8) + \text{dev}(:,2,9) + \text{dev}(:,2,10); \]
\[ \text{dev}(:,2,1) = \text{dev}(:,2,1) + \left( \text{sdev2} - \text{dev}(:,2,1) \right) / 9; \]
\[ \text{dev}(:,2,2) = \text{dev}(:,2,2) + \left( \text{sdev2} - \text{dev}(:,2,2) \right) / 9; \]
\[ \text{dev}(:,2,3) = \text{dev}(:,2,3) + \left( \text{sdev2} - \text{dev}(:,2,3) \right) / 9; \]
\[ \text{dev}(:,2,4) = \text{dev}(:,2,4) + \left( \text{sdev2} - \text{dev}(:,2,4) \right) / 9; \]
\[ \text{dev}(:,2,5) = \text{dev}(:,2,5) + \left( \text{sdev2} - \text{dev}(:,2,5) \right) / 9; \]
\[ \text{dev}(:,2,6) = \text{dev}(:,2,6) + \left( \text{sdev2} - \text{dev}(:,2,6) \right) / 9; \]
\[ \text{dev}(:,2,7) = \text{dev}(:,2,7) + \left( \text{sdev2} - \text{dev}(:,2,7) \right) / 9; \]
\[ \text{dev}(:,2,8) = \text{dev}(:,2,8) + \left( \text{sdev2} - \text{dev}(:,2,8) \right) / 9; \]
ant31 = interp1(ant31(:,1), ant31(:,2), time(:,1)) - non;
ant32 = interp1(ant32(:,1), ant32(:,2), time(:,1)) - non;
ant33 = interp1(ant33(:,1), ant33(:,2), time(:,1)) - non;
ant34 = interp1(ant34(:,1), ant34(:,2), time(:,1)) - non;
ant35 = interp1(ant35(:,1), ant35(:,2), time(:,1)) - non;
ant36 = interp1(ant36(:,1), ant36(:,2), time(:,1)) - non;
ant37 = interp1(ant37(:,1), ant37(:,2), time(:,1)) - non;
ant38 = interp1(ant38(:,1), ant38(:,2), time(:,1)) - non;
ant39 = interp1(ant39(:,1), ant39(:,2), time(:,1)) - non;
ant310 = interp1(ant310(:,1), ant310(:,2), time(:,1)) - non;
sum3 = ant31 + ant32 + ant33 + ant34 + ant35 + ant36 + ant37 + ant38 + ant39 + ant310;
avg3 = sum3 / 10;
dev(:,3,1) = ant31 - (sum3 - ant31) / 9;
dev(:,3,2) = ant32 - (sum3 - ant32) / 9;
dev(:,3,3) = ant33 - (sum3 - ant33) / 9;
dev(:,3,4) = ant34 - (sum3 - ant34) / 9;
dev(:,3,5) = ant35 - (sum3 - ant35) / 9;
dev(:,3,6) = ant36 - (sum3 - ant36) / 9;
dev(:,3,7) = ant37 - (sum3 - ant37) / 9;
dev(:,3,8) = ant38 - (sum3 - ant38) / 9;
dev(:,3,9) = ant39 - (sum3 - ant39) / 9;
dev(:,3,10) = ant310 - (sum3 - ant310) / 9;

sdev3 = dev(:,3,1) + dev(:,3,2) + dev(:,3,3) + dev(:,3,4) + dev(:,3,5) + dev(:,3,6) + dev(:,3,7) + dev(:,3,8) + dev(:,3,9) + dev(:,3,10);

dev(:,3,1) = dev(:,3,1) + (sdev3 - dev(:,3,1)) / 9;
dev(:,3,2) = dev(:,3,2) + (sdev3 - dev(:,3,2)) / 9;
dev(:,3,3) = dev(:,3,3) + (sdev3 - dev(:,3,3)) / 9;
dev(:,3,4) = dev(:,3,4) + (sdev3 - dev(:,3,4)) / 9;
dev(:,3,5) = dev(:,3,5) + (sdev3 - dev(:,3,5)) / 9;
dev(:,3,6) = dev(:,3,6) + (sdev3 - dev(:,3,6)) / 9;
dev(:,3,7) = dev(:,3,7) + (sdev3 - dev(:,3,7)) / 9;
dev(:,3,8) = dev(:,3,8) + (sdev3 - dev(:,3,8)) / 9;
dev(:,3,9) = dev(:,3,9) + (sdev3 - dev(:,3,9)) / 9;
dev(:,3,10) = dev(:,3,10) + (sdev3 - dev(:,3,10)) / 9;

mmm = 3;
devv = zeros(length(dev) * mmm, 3, 10);
dev(1:length(dev), :, :) = dev;
devv = devv;
time = [time(:, 1)';
    linspace(2 * time(end, 1) - time(end - 1, 1), mmm * time(end, 1), (mmm - 1) * length(time(:, 1)))';
    fedd = interp1(fed(:, 1), fed(:, 2), time(min(find(time >= .1699)):max(find(time <= .7628)), 1));

dev(:, 2, 9) = dev(:, 2, 9) + (sdev2 - dev(:, 2, 9)) / 9;
dev(:, 2, 10) = dev(:, 2, 10) + (sdev2 - dev(:, 2, 10)) / 9;
directivity.m

% // CST Farfield Source File
% % // Version:
% % 1.1
% % // Radiated Power
% % 1.804103e-001
% % // Accepted Power
% % 4.736657e-001
% % // Stimulated Power
% % 5.000000e-001
% % // Frequency
% % 6.650000e+009
% % // >> Total #phi samples, total #theta samples
% % 361 181
% % // >> Phi, Theta, Re(E_Theta), Im(E_Theta), Re(E_Phi), Im(E_Phi)
load directivity
phi=directivity(:,1);
theta=directivity(:,2);
ephi=(directivity(:,5).^2+directivity(:,6).^2);
etheta=(directivity(:,3).^2+directivity(:,4).^2);
etot=ephi+etheta;
etot=etot/2/(1/6.7)^.5;
ptot=4*pi*etot/1.804103e-001;
nptot=ptot/max(ptot);

A2. MATLAB making and checking HEX Files

sampling.m

function s=sampling(x,t)
Ts=t(2)-t(1);
n1=0:10*Ts:t(end);
n=round(n1/(Ts));
s=zeros(1,length(n));
for i=1:length(n)
    s(i)=x(n(i)+1);
end

make_hex.m

% This code quantizes non and makes hex file
non=devall;
non(2650)=0;
non(5300)=0;
non(7950)=0;
[in,Qn]=quantiz(non,(linspace(-max(abs(non)),max(abs(non)),2^16))*1.1 ,linspace(-2^15,2^15,2^16+1));
%Qn=non;
fid=fopen('dev_load.hex','w');
for i=1:length(non)
    temp=str2double(dec2mvl(Qn(i),16));
    temp=num2str(temp);
    temp=bin2dec(temp);
    temp=dec2hex(temp,4);
    fwrite(fid,temp);
end
fclose(fid);

combine_hex.m

all=[];

fid=fopen('non.hex','r');
a=fscanf(fid,'%s');
a=['FFFF',a];
all=[all,a];
close(fid);

for i=1:3
    for j=1:10
        fid=fopen(['ant',num2str(i),num2str(j),'.hex'],'r');
        a=fscanf(fid,'%s');
        a=['FFFF',a];
        all=[all,a];
        fclose(fid);
    end
end

fid=fopen('antallwithnon.hex','w');
fwrite(fid,all);
close(fid);

ant_q.m

% This code quantizes ant, non, fed and makes hex files
clc
clear all

run(['pwd','\a1.m'])
run(['pwd','\b1.m'])
run(['pwd','\c1.m'])
run(['pwd','\d1.m'])
run(['pwd','\e1.m'])
run(['pwd','\f1.m'])
run(['pwd','\g1.m'])
run(['pwd','\h1.m'])
run(['pwd','\i1.m'])
run(['pwd','\j1.m'])
trr1=min([ant1(end,1) ant2(end,1) ant3(end,1) ant4(end,1) ant5(end,1)
ant6(end,1) ant7(end,1) ant8(end,1) ant9(end,1) ant10(end,1)]);
tll1=[ant1(end,1) ant2(end,1) ant3(end,1) ant4(end,1) ant5(end,1)
ant6(end,1) ant7(end,1) ant8(end,1) ant9(end,1) ant10(end,1)];
tloc1 = find(tll1 == trr1);
t1 = eval(['ant' int2str(tloc1) '(:,1)']);

run([pwd, '\a2.m'])
run([pwd, '\b2.m'])
run([pwd, '\c2.m'])
run([pwd, '\d2.m'])
run([pwd, '\e2.m'])
run([pwd, '\f2.m'])
run([pwd, '\g2.m'])
run([pwd, '\h2.m'])
run([pwd, '\i2.m'])
run([pwd, '\j2.m'])

trr2 = min([ant21(end,1) ant22(end,1) ant23(end,1) ant24(end,1) ant25(end,1) ant26(end,1) ant27(end,1) ant28(end,1) ant29(end,1) ant210(end,1)]);
tll2 = [ant21(end,1) ant22(end,1) ant23(end,1) ant24(end,1) ant25(end,1) ant26(end,1) ant27(end,1) ant28(end,1) ant29(end,1) ant210(end,1)];
tloc2 = find(tll2 == trr2);
t2 = eval(['ant2' int2str(tloc2) '(:,1)']);

run([pwd, '\a3.m'])
run([pwd, '\b3.m'])
run([pwd, '\c3.m'])
run([pwd, '\d3.m'])
run([pwd, '\e3.m'])
run([pwd, '\f3.m'])
run([pwd, '\g3.m'])
run([pwd, '\h3.m'])
run([pwd, '\i3.m'])
run([pwd, '\j3.m'])

trr3 = min([ant31(end,1) ant32(end,1) ant33(end,1) ant34(end,1) ant35(end,1) ant36(end,1) ant37(end,1) ant38(end,1) ant39(end,1) ant310(end,1)]);
tll3 = [ant31(end,1) ant32(end,1) ant33(end,1) ant34(end,1) ant35(end,1) ant36(end,1) ant37(end,1) ant38(end,1) ant39(end,1) ant310(end,1)];
tloc3 = find(tll3 == trr3);
t3 = eval(['ant3' int2str(tloc3) '(:,1)']);

%%%%%%
trrf = min([t1(end) t2(end) t3(end)]);
tllf = [t1(end) t2(end) t3(end)];
tlocf = find(tllf == trrf);
time = eval(['t' int2str(tlocf)]);

anta = zeros(length(time), 3, 10);
for i = 1:10
anta(:,:,i) = interp1(eval(['ant', int2str(i), '(:,1)']), eval(['ant', int2str(i), '(:,2)']), time(:,1));
end
for j = 2:3
for i=1:10
anta(:,j,i)=interp1(eval(['ant',int2str(j),int2str(i),'(:,1)']),eval(['ant',int2str(j),int2str(i),'(:,2)']),time(:,1));
end

ant=zeros(length(anta)/10,3,10);
for i=1:3
    for j=1:10
        ant(:,i,j)=sampling(anta(:,i,j),time(:,1));
    end
end

run([pwd,'none.m'])
non=interp1(non(:,1),non(:,2),time(:,1));
%Mn=min([min(non),min(min(min(ant)))])*10/9;
Mx=max([max(abs(non)),max(max(max(abs(ant))))])*10/9;
Q=zeros(size(ant));
for i=1:3
    for j=1:10
        [in,Q(:,i,j)]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));
    end
end

for j=1:3
    for k=1:10
        fid=fopen(['ant',int2str(j),int2str(k),'.hex'],'w');
        for i=1:length(ant(:,1))
            temp=str2double(dec2mvl(Q(i,j,k),16));
            temp=num2str(temp);
            temp=bin2dec(temp);
            temp=dec2hex(temp,4);
            fwrite(fid,temp);
        end
        fclose(fid);
    end
end

non=sampling(non,time(:,1));
[in,Qn]=quantiz(non,linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));
fid=fopen('non.hex','w');
for i=1:length(ant(:,1))
    temp=str2double(dec2mvl(Qn(i),16));
    temp=num2str(temp);
    temp=bin2dec(temp);
    temp=dec2hex(temp,4);
    fwrite(fid,temp);
end
fclose(fid);
run([pwd,'/feed.m']);
fed=interp1(fed(:,1),fed(:,2),time(:,1));
fed=sampling(fed,time(:,1));
[in,Qn]=quantiz(fed,linspace(min(fed),max(fed),2^16),linspace(-2^15+1,2^15+1,2^16+1));
 fid=fopen('fed.hex','w');
for i=1:length(fed)
    temp=str2double(dec2mvl(Qn(i),16));
    temp=num2str(temp);
    temp=bin2dec(temp);
    temp=dec2hex(temp,4);
    fwrite(fid,temp);
end
fclose(fid);

ant_q1.m

% this code quantizes and samples ants, non, fed and saves the matrices
clc
clear all

run([pwd,'/a1.m'])
run([pwd,'/b1.m'])
run([pwd,'/c1.m'])
run([pwd,'/d1.m'])
run([pwd,'/e1.m'])
run([pwd,'/f1.m'])
run([pwd,'/g1.m'])
run([pwd,'/h1.m'])
run([pwd,'/i1.m'])
run([pwd,'/j1.m'])

trr1=min([ant1(end,1) ant2(end,1) ant3(end,1) ant4(end,1) ant5(end,1) ant6(end,1) ant7(end,1) ant8(end,1) ant9(end,1) ant10(end,1)]);
 tll1=[ant1(end,1) ant2(end,1) ant3(end,1) ant4(end,1) ant5(end,1) ant6(end,1) ant7(end,1) ant8(end,1) ant9(end,1) ant10(end,1)];
 tloc1=find(tll1==trr1);
 t1=eval(['ant' int2str(tloc1(1)) '(:,1)']);

run([pwd,'/a2.m'])
run([pwd,'/b2.m'])
run([pwd,'/c2.m'])
run([pwd,'/d2.m'])
run([pwd,'/e2.m'])
run([pwd,'/f2.m'])
run([pwd,'/g2.m'])
run([pwd,'/h2.m'])
run([pwd,'/i2.m'])
run([pwd,'/j2.m'])

trr2=min([ant21(end,1) ant22(end,1) ant23(end,1) ant24(end,1) ant25(end,1) ant26(end,1) ant27(end,1) ant28(end,1) ant29(end,1) ant210(end,1)]);
 tll2=[ant21(end,1) ant22(end,1) ant23(end,1) ant24(end,1) ant25(end,1) ant26(end,1) ant27(end,1) ant28(end,1) ant29(end,1) ant210(end,1)];
 tloc2=find(tll2==trr2);
 t2=eval(['ant2' int2str(tloc2(1)) '(:,1)']);

run([pwd,'/a3.m'])
run([pwd,'\b3.m'])
run([pwd,'\c3.m'])
run([pwd,'\d3.m'])
run([pwd,'\e3.m'])
run([pwd,'\f3.m'])
run([pwd,'\g3.m'])
run([pwd,'\h3.m'])
run([pwd,'\i3.m'])
run([pwd,'\j3.m'])

trr3=min([\text{ant31}(\text{end},1) \text{ant32}(\text{end},1) \text{ant33}(\text{end},1) \text{ant34}(\text{end},1)
\text{ant35}(\text{end},1) \text{ant36}(\text{end},1) \text{ant37}(\text{end},1) \text{ant38}(\text{end},1) \text{ant39}(\text{end},1)
\text{ant310}(\text{end},1)])
\text{tll3}=[\text{ant31}(\text{end},1) \text{ant32}(\text{end},1) \text{ant33}(\text{end},1) \text{ant34}(\text{end},1) \text{ant35}(\text{end},1)
\text{ant36}(\text{end},1) \text{ant37}(\text{end},1) \text{ant38}(\text{end},1) \text{ant39}(\text{end},1) \text{ant310}(\text{end},1)]
\text{tloc3}=Math\text{find}(\text{tll3}==\text{trr3})
\text{t3}=\text{eval}(['\text{ant3}' int2str(tloc3(1)) '(:,1)'])

%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%
trrf=min([\text{t1}(\text{end}) \text{t2}(\text{end}) \text{t3}(\text{end})])
\text{tllf}=[\text{t1}(\text{end}) \text{t2}(\text{end}) \text{t3}(\text{end})]
\text{tlocf}=Math\text{find}(\text{tllf}==\text{trrf})
\text{time}=\text{eval}(['\text{t}' int2str(tlocf(1))])

\text{anta}=\text{zeros}(\text{length}(\text{time}),3,10);
\text{for} \text{i}=1:10
\text{anta}(:,:,\text{i})=\text{interp1}(\text{eval}(['\text{an}' int2str(i) '(:,1)']),\text{eval(['\text{ant}' int2str(i) '(:,2)'])},\text{time}(:,:,1));
\text{end}
\text{for} \text{j}=2:3
\text{for} \text{i}=1:10
\text{anta}(:,:,\text{i},\text{j})=\text{interp1}(\text{eval(['\text{an}' int2str(i) '(:,1)']),\text{eval(['\text{ant}' int2str(j) '(:,1)'])},\text{eval(['\text{ant}' int2str(j) '(:,2)'])},\text{time}(:,:,1));
\text{end}
\text{end}
\text{ant}=\text{zeros}(\text{length}(\text{anta})/10,3,10);
\text{for} \text{i}=1:3
\text{for} \text{j}=1:10
\text{anta}(\text{i},\text{j},\text{i})=\text{interp1}(\text{anta}(\text{i},\text{j}),\text{time}(:,:,1));
\text{end}
\text{end}
run([pwd,'\none.m'])
\text{non}=\text{interp1}(\text{non}(:,:,1),\text{non}(:,:,2),\text{time}(:,:,1));
\text{Mn}=\text{min}([\text{min}(\text{non}),\text{min}([\text{min}(\text{\text{min}(\text{anta}))}))])*10/9;
\text{Mx}=\text{max}([\text{max}(\text{abs}(\text{non})),\text{max}([\text{max}(\text{\text{max}(\text{abs}(\text{\\text{anta})}))])])*10/9;
\text{non}=\text{sampling}(\text{non},\text{time}(:,:,1));
\text{[in,non]}=\text{quantiz}(\text{non},\text{linespace}(-\text{Mx},\text{Mx},2^{16}),\text{linespace}(-2^{15}+1,2^{15}+1,2^{16}+1));
save ant ant
clear ant*
load ant
i=1; j=1;
[in,ant1]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant2]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant3]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant4]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant5]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant6]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant7]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant8]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant9]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant10]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
i=1;j=1;
[in,ant21]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant22]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant23]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant24]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant25]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant26]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant27]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant28]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant29]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant30]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
i=3;j=1;
[in,ant31]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant32]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant33]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant34]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant35]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant36]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
[in,ant37]=quantiz(ant(:,i,j),linspace(-Mx,Mx,2^16),linspace(-2^15+1,2^15+1,2^16+1));j=j+1;
\[ \text{in,ant38} = \text{quantiz(ant(:,i,j),linspace(-Mx,Mx,2^{16}),linspace(-2^{15}+1,2^{15}+1))}; j=j+1; \]
\[ \text{in,ant39} = \text{quantiz(ant(:,i,j),linspace(-Mx,Mx,2^{16}),linspace(-2^{15}+1,2^{15}+1))}; j=j+1; \]
\[ \text{in,ant310} = \text{quantiz(ant(:,i,j),linspace(-Mx,Mx,2^{16}),linspace(-2^{15}+1,2^{15}+1))}; j=j+1; \]

run([pwd,'/feed.m']);
fed=interp1(fed(:,1),fed(:,2),time(:,1));
fed=sampling(fed,time(:,1));
\[ \text{in,fed} = \text{quantiz(fed,linspace(min(fed),max(fed),2^{16}),linspace(-2^{15}+1,2^{15}+1,2^{16}+1))}; fedd=fed(30:128); \]

\textbf{check\_hex\_dev.m}

\% this code reads the hex files, converts to matrices and compares to the 
\% real matrix

\texttt{clear devall}
\texttt{fid=fopen('read\_dev.hex','r'); \% hex file name}
\texttt{ANT1=fscanf(fid,'%s');}
\texttt{j=1;}
\texttt{for i=1:4:length(ANT1)-3}
\texttt{\quad temp=ANT1(i:i+3);}
\texttt{\quad temp=hex2dec(temp);}
\texttt{\quad temp=dec2bin(temp,16);}
\texttt{\quad temp=mvl2dec(temp,1);}
\texttt{\quad devall(j)=temp;}
\texttt{\quad j=j+1;}
\texttt{end}
\texttt{plot(devall)}

\textbf{check\_image.m}

\texttt{clear ima imex}
\texttt{fid=fopen('im.hex','r'); \% hex file name from 3017 length 1236}
\texttt{ANT1=fscanf(fid,'%s');}
\texttt{j=1;}
\texttt{for i=1:4:length(ANT1)-3}
\texttt{\quad temp=ANT1(i:i+3);}
\texttt{\quad temp=hex2dec(temp);}
\texttt{\quad temp=dec2bin(temp,16);}
\texttt{\quad temp=mvl2dec(temp,1);}
\texttt{\quad ima(j)=temp;}
\texttt{\quad j=j+1;}
\texttt{end}
\texttt{fclose(fid);}

\texttt{fid=fopen('imex.hex','r'); \% hex file name from 4000 length 1236}
\texttt{ANT1=fscanf(fid,'%s');}
\texttt{j=1;}
\texttt{for i=1:4:length(ANT1)-3}
\texttt{\quad temp=ANT1(i:i+3);}
\texttt{\quad temp=hex2dec(temp);}
\texttt{\quad temp=dec2bin(temp,16);}
\texttt{\quad temp=mvl2dec(temp,1);}
\texttt{\quad ima(j)=temp;}
\texttt{\quad j=j+1;}
\texttt{end}
temp=dec2bin(temp,16);
temp=mvl2dec(temp,1);
imex(j)=temp;
j=j+1;
end
fclose(fid);
for i=1:length(ima)
    q(i)=ima(i)*2^(-imex(i));
end

figure(3)
imq=zeros(111,21);
for i=1:length(q)/21
    imq(i,:)=q(21*(i-1)+1:21*i);
end

imshow(imq/max(max(abs(imq))))
Appendix B
Schematics & Verilog Codes for FPGA Implementation

B1. Artifact Remover
B1.1 Schematics
dev_builder.bdf
average.bdf
B1.2 Verilog Codes

**clk_divider.v**

module clk_divider(iCLK,oCLK,oe);
    input iCLK,oe;
    output oCLK;
    reg oCLK;
    reg [27:0] count;

    initial begin
        oCLK=1;
    end

    always@(posedgeiCLK)
    begin
        if (oe)
            begin
                count=count+1'b1;
                if (count==500)
                    begin
                        oCLK=!oCLK;
                        count=0;
                    end
            end
        else
            begin
                oCLK = 1'b1;
                count = 0;
            end
    end
endmodule

**hex_7seg.v**

module hex_7seg(hex_digit,seg);
    input [3:0] hex_digit;
    output [6:0] seg;
    reg [6:0] seg;
    // seg = {g,f,e,d,c,b,a};
    // 0 is on and 1 is off

    always @(hex_digit)
    case (hex_digit)
        4'h0: seg = 7'b1000000;
        4'h1: seg = 7'b1111001;  // ---a---
        4'h2: seg = 7'b0100100;  // |      |
        4'h3: seg = 7'b0110000;  // f      b
        4'h4: seg = 7'b0011001;  // |      |
module hex_7seg(hex_digit, seg);
input [3:0] hex_digit;
output [6:0] seg;
reg [6:0] seg;
// seg = {g,f,e,d,c,b,a};
// 0 is on and 1 is off
always @(hex_digit)
case (hex_digit)
  4'h0: seg = 7'b1000000; // ---a-----
  4'h1: seg = 7'b1111001; // |   |
  4'h2: seg = 7'b0100100; // |   |  
  4'h3: seg = 7'b0110000; // f   b  
  4'h4: seg = 7'b0011001; // |   |  
  4'h5: seg = 7'b0010010; // ---g-----
  4'h6: seg = 7'b0000010; // |   |  
  4'h7: seg = 7'b1111000; // e   c  
  4'h8: seg = 7'b0000000; // |   |  
  4'h9: seg = 7'b0011000; // ---d-----
  4'ha: seg = 7'b0001000;
  4'hb: seg = 7'b0000011;
  4'hc: seg = 7'b1000110;
  4'hd: seg = 7'b0100001;
  4'he: seg = 7'b0000110;
  4'hf: seg = 7'b0001110;
endcase
endmodule

op
module op_avg
input [3:0] hex_digit
input [6:0] seg;
reg [6:0] seg;
// seg = {g,f,e,d,c,b,a};
// 0 is on and 1 is off
always @(hex_digit)
case (hex_digit)
  4'h0: seg = 7'b1000000; // ---a-----
  4'h1: seg = 7'b1111001; // |   |
  4'h2: seg = 7'b0100100; // |   |  
  4'h3: seg = 7'b0110000; // f   b  
  4'h4: seg = 7'b0011001; // |   |  
  4'h5: seg = 7'b0010010; // ---g-----
  4'h6: seg = 7'b0000010; // |   |  
  4'h7: seg = 7'b1111000; // e   c  
  4'h8: seg = 7'b0000000; // |   |  
  4'h9: seg = 7'b0011000; // ---d-----
  4'ha: seg = 7'b0001000;
  4'hb: seg = 7'b0000011;
  4'hc: seg = 7'b1000110;
  4'hd: seg = 7'b0100001;
  4'he: seg = 7'b0000110;
  4'hf: seg = 7'b0001110;
endcase
endmodule

op_avg.v
module op_avg
  input iCLK;
  input [3:0] stop;
  input [6:0] oADDR;
  input [6:0] oDATA;
  input [31:0] iCLK, count, iDATA,ctr
);
input [3:0] count,ctr;
input signed [15:0] iDATA;

output stop;
output [17:0] oADDR;
output signed [15:0] oDATA;

reg [17:0] oADDR,rADDR,rADDR1,sADDR,wADDR;
reg stop;
reg signed [19:0] sum;
reg signed [15:0] oDATA;

initial begin
    rADDR = 18'b000000000100001001;//109
    rADDR1 = 18'b000000000100001001;//109
    sADDR = 18'b000000001000010010;//212
    wADDR = 18'b00000100000000010111;//2017
    stop = 1;
end

always @(negedgeiCLK) begin
    stop = 1;
    if (count == 0) begin
        oADDR = rADDR;
        rADDR = rADDR +
        18'b000000000100001001;
    end
    else if (count >=1 && count <=9) begin
        oADDR = rADDR;
        rADDR = rADDR +
        18'b000000000100001001;
        sum = sum + iDATA;
    end
    else if (count =10) begin
        oADDR = wADDR;
        wADDR = wADDR +
        18'b000000000000000010111;
        sum = sum + iDATA;
    end
    else if (count ==11) begin
        oDATA = sum/6;
        sum = 0;
    end
end
else if (count == 13)
begin
  rADDR1 = rADDR1 + 18'b000000000000000001;
  rADDR = rADDR1;
  if (rADDR1 == sADDR)
  begin
    if (ctr == 1)
    begin
      rADDR = 18'b000000000100001001;
      rADDR1 = 18'b000000000100001001;
      sADDR = 18'b000000001000010010;
      wADDR = 18'b000010000000010111; //2017
    end
    else if ((ctr == 3) || (ctr == 6))
    begin
      rADDR = 18'b000000101101100011; //B63
      rADDR1 = 18'b000000101101100011; //B63
      sADDR = 18'b000000110011011001; //C6C
      wADDR = 18'b000010000100100000; //2120
    end
    else if ((ctr == 8) || (ctr == 11))
    begin
      rADDR = 18'b000001010110111101; //15BD
      rADDR1 = 18'b000001010110111101; //15BD
      sADDR = 18'b000001011011000110; //16C6
      wADDR = 18'b000010001000101001; //2229
    end
  end
  stop = 0;
end
end

endmodule

SRAM_interface.v

module SRAM_interface
(
  oDATA,
  c1, c2,
SRAM_DQ,SRAM_ADDR,
SRAM_Ub_N,SRAM_Lb_N,SRAM_WE_N,SRAM_CE_N,
SRAM_OE_N,
//GPIO_0, GPIO_1,
HEX0,HEX1,HEX2,HEX3,
LEDG,
iCLK,OE,
dDATA,iADDR
);

input iCLK,OE;
input [15:0] dDATA;
input [17:0] iADDR;
output [15:0] oDATA;
output c1,c2;
inout [15:0] SRAM_DQ;
output [17:0] SRAM_ADDR;
output
N,SRAM_OE_N;

//inout [35:0] GPIO_0, GPIO_1;
output [6:0] HEX0,HEX1,HEX2,HEX3;
wire [6:0] HEX0ss,HEX1ss,HEX2ss,HEX3ss;
output [1:0] LEDG;

// set all inout ports to tri-state
//assign GPIO_0 = 36'hzzzzzzzzz;
//assign GPIO_1 = 36'hzzzzzzzzz;
assign LEDG[1:0] = {SRAM_WE_N,SRAM_OE_N};
assign HEX0 = HEX0ss;
assign HEX1 = HEX1ss;
assign HEX2 = HEX2ss;
assign HEX3 = HEX3ss;
assign SRAM_CE_N = OE? 1'b0 : 1'bz;
assign SRAM_Ub_N = OE? 1'b0 : 1'bz;
assign SRAM_Lb_N = OE? 1'b0 : 1'bz;

assign SRAM_ADDR = OE? iADDR:
18'bzzzzzzzzzzzzzzzzz;
reg [15:0] oDATA;
reg c1,c2;
reg [15:0] SRAM_DQ;
//reg [17:0] SRAM_ADDR;
reg SRAM_WE_N,
SRAM_OE_N;

always @(posedgeiCLK )
begin
  case({c1,c2})
2'b00:
begin //read
  SRAM_WE_N = OE? !c2 :
  l'bz;
  SRAM_OE_N = OE? c2 :
  l'bz;
  oDATA = OE?
(SRAM_OE_N ? 16'h1111:SRAM_DQ) :
16'hzzzz;
  c1 = 1'b1;
end
2'b10:
begin //read
  SRAM_WE_N = OE? !c2 :
  l'bz;
  SRAM_OE_N = OE? c2 :
  l'bz;
  oDATA = OE?
(SRAM_OE_N ? 16'h1111:SRAM_DQ) :
16'hzzzz;
  c2 = 1'b1;
end
2'b11:
begin//write
  SRAM_WE_N = OE? !c2 :
  l'bz;
  SRAM_DQ = OE?
(SRAM_WE_N ? 16'hzzzz:dDATA) : 16'hzzzz;
  c1 = 1'b0;
end
2'b01:
begin
  SRAM_WE_N = OE? 1'b1 :
  l'bz;
  SRAM_DQ = 16'hzzzz;
  c2 = 1'b0;
end
endcase

end
hex_7seg dsp0(.hex_digit(iADDR[3:0]),.seg(HEX0ss));
hex_7seg dsp1(.hex_digit(iADDR[7:4]),.seg(HEX1ss));
hex_7seg dsp2(.hex_digit(iADDR[11:8]),.seg(HEX2ss));
hex_7seg dsp3(.hex_digit(iADDR[15:12]),.seg(HEX3ss));
endmodule
module op_avg

(  
  stop,  
oADDR,  
oDATA,  
iCLK, count,  
iDATA,ctr  
);

input  iCLK;
input [3:0] count,ctr;
input signed [15:0] iDATA;

output  stop;
output [17:0] oADDR;
output signed [15:0] oDATA;

reg [17:0] oADDR,rADDR,rADDR1,sADDR,wADDR;
reg stop;
reg signed [19:0] sum;
reg signed [15:0] oDATA;

initial begin
  rADDR = 18'b000000000010001001;//109
  rADDR1 = 18'b00000000010001001;//109
  sADDR = 18'b00000000100001000100010010;
  //212
  wADDR = 18'b000010000000010111;//2017
  stop = 1;
end

always @(negedgeiCLK)
begin

  stop = 1;
  if (count == 0)
  begin
    oADDR = rADDR;
    rADDR = rADDR + 18'b000000000010001001;  
  end
  else if (count >=1 && count <=9)
  begin
    oADDR = rADDR;
    rADDR = rADDR + 18'b000000000010001001;  
  end
end
sum = sum + iDATA;
end
else if (count == 10)
begin
    oADDR = wADDR;
    wADDR = wADDR + 18'b000000000000000001;
    sum = sum + iDATA;
end
else if (count == 11)
begin
    oDATA = sum/6;
    sum = 0;
end
else if (count == 13)
begin
    rADDR1 = rADDR1 + 18'b000000000100001001;
    rADDR = rADDR1;
    if (rADDR1==sADDR)
    begin
        if (ctr == 1)
        begin
            rADDR = 18'b000000000100001001;
            rADDR1 = 18'b000000000100001001;
            sADDR = 18'b000000001000010010;
            wADDR = 18'b000010000000010111; //2017
        end
        else if ((ctr == 3) || (ctr == 6))
        begin
            rADDR = 18'b000000010110110111; //B63
            rADDR1 = 18'b000000010110110111; //B63
            sADDR = 18'b000000011000110110; //C6C
            wADDR = 18'b00001000100000101111; //2120
        end
        else if ((ctr == 8) || (ctr == 11))
        begin
            rADDR = 18'b00000001010110111011101; //15BD
controller

module controller
    (OE1,OE2,fin, ctr, GPIO_0,GPIO_1, stop1,stop2
    );

input stop1,stop2;
output OE1,OE2,fin;
output [3:0] ctr;
inout [35:0] GPIO_0,GPIO_1;

reg [3:0] ctr1,ctr2;

assign GPIO_0 = 36'hzzzzzzzz;
assign GPIO_1 = 36'hzzzzzzzz;

assign ctr = ctr1 + ctr2;
assign fin = ((ctr==15)?1'b1:1'b0 ;
assign OE1 = ((ctr==0) || (ctr==2) || (ctr==4) || (ctr==5) || (ctr==7) || (ctr==9) || (ctr==10) || (ctr==12) || (ctr==14))? 1'b1:1'b0 ;
assign OE2 = ((ctr==1) || (ctr==3) || (ctr==6) || (ctr==8) || (ctr==11) || (ctr==13))?1'b1:1'b0 ;
always  @ (negedge stop1)
begin
    ctr1 = ctr1 + 4'b0001;
end

always  @ (negedge stop2)
begin
    ctr2 = ctr2 + 4'b0001;
end

dendmodule
B.2 Beamformer
B.2.1 Schematics
mul.bdf
B2.2 Verilog Codes

op_mul.v

module op_mul

( oDATA,oADDR, 
 iCLK, count,mflag, 
 iDATA,cat,str1,n1
);

input 
 iCLK,str1;
input 
 [2:0] count;
input 
 signed [15:0] iDATA;
input 
 [17:0] n1;
output 
 mflag;
output 
 [17:0] oADDR;
output 
 signed [15:0] oDATA;
output 
 [4:0] cat;

reg 
 [17:0] oADDR,wADDR,aADDR,nADDR,n2;
//reg
reg 
 stg;
reg 
 signed [31:0] temp2;
reg 
 signed [15:0] oDATA;
reg 
 [4:0] msbmax;
reg 
 en,mflag;
wire 
 [4:0] msb;

initial
begin

mflag= 1;

nADDR = 18'b000000000000000000;
aADDR = 18'b000000000000000000;
temp2 = 32'h00000001;
msbmax = 5'b01111;
en = 1'b0;

end
msb_find
assign 
 msb1 (msb,en,temp2); 
 always @(negedgeiCLK)
begin

//stg = 1'b1;
case (count)
3'b000://read
begin
if(!mflag)
begin
msbmax <= 5'b01111;

end
```
en <= 1'b0;
oADDR <= nADDR;
temp2 <= 32'h00000001;
if (nADDR==18'h00000)
begin
    n2 <= n1;
end
end

3'b001:// read and mul
begin
    oADDR <= aADDR+nADDR+n2;
temp2 <= str1? 32'h01000000 : temp2*iDATA;
end

3'b010://mul and write
begin
    mflag <= 1'b1;
oADDR <= nADDR;
temp2 <= str1? 32'h01000000 : temp2*iDATA;
end

3'b011:// write
begin
    en <= 1'b1;
oDATA <= str1? 16'h00ff : (((n2+nADDR)>=18'h00108) ? 16'h0000 : temp2[31:16]);
end

3'b100:
begin
    if (msb>msbmax)
    begin
        msbmax <= msb;
    end
    nADDR <= nADDR + 18'b0000000000000000000000001;
end

3'b101:
begin
if
(nADDR==18'b0000000000000000)//d100
begin
 nADDR <= 18'h0000000000000000;
aADDR <= aADDR + 18'b0000000000000001; //109
if (aADDR==18'h01f0e)
begin
 aADDR <= 18'h00000;
end
// n2 <= n1;
mflag <= 1'b0;
end
endcase
end
endmodule

msb_find.v

module msb_find(msb,en,x,comp);

input en;
input [31:0] x;
output [4:0] msb;
reg [4:0] msb;
reg flag;
output [30:0] comp;
integer k;

oxor xor0 (comp[0],x[1],x[0]);
oxor xor1 (comp[1],x[2],x[1]);
oxor xor2 (comp[2],x[3],x[2]);
oxor xor3 (comp[3],x[4],x[3]);
oxor xor4 (comp[4],x[5],x[4]);
oxor xor5 (comp[5],x[6],x[5]);
oxor xor6 (comp[6],x[7],x[6]);
oxor xor7 (comp[7],x[8],x[7]);
oxor xor8 (comp[8],x[9],x[8]);
oxor xor9 (comp[9],x[10],x[9]);
oxor xor10 (comp[10],x[11],x[10]);
oxor xor11 (comp[11],x[12],x[11]);
oxor xor12 (comp[12],x[13],x[12]);
oxor xor13 (comp[13],x[14],x[13]);
xor xor14 (comp[14],x[15],x[14]);
xor xor15 (comp[15],x[16],x[15]);
xor xor16 (comp[16],x[17],x[16]);
xor xor17 (comp[17],x[18],x[17]);
xor xor18 (comp[18],x[19],x[18]);
xor xor19 (comp[19],x[20],x[19]);
xor xor20 (comp[20],x[21],x[20]);
xor xor21 (comp[21],x[22],x[21]);
xor xor22 (comp[22],x[23],x[22]);
xor xor23 (comp[23],x[24],x[23]);
xor xor24 (comp[24],x[25],x[24]);
xor xor25 (comp[25],x[26],x[25]);
xor xor26 (comp[26],x[27],x[26]);
xor xor27 (comp[27],x[28],x[27]);
xor xor28 (comp[28],x[29],x[28]);
xor xor29 (comp[29],x[30],x[29]);
xor xor30 (comp[30],x[31],x[30]);

initial
begin
  msb = 15;
end

always @(posedge en)
begind
  flag = 1'b1;
  msb = 5'b01111;
  for(k=30;k>15;k=k-1)
    begin
      if (comp[k]==1'b1)
        begin
          if (flag)
            begin
              msb = k+1;
            end
          flag = 1'b0;
        end
    end
end
endmodule

SRAM_Interface_mul.v

module SRAM_interface_mul (
  oDATA,
  count,
  SRAM_DQ,SRAM_ADDR,
input iCLK,OE,OEm;
input [15:0] dDATA;
input [17:0] iADDR;

output [15:0] oDATA;
output [2:0] count;
inout [15:0] SRAM_DQ;
output [17:0] SRAM_ADDR;

// set all inout ports to tri-state
assign GPIO_0 = 36'hzzzzzzzzzzzzzzzzzz;
assign GPIO_1 = 36'hzzzzzzzzzzzzzzzzzz;
assign HEX0 = HEX0ss;
assign HEX1 = HEX1ss;
assign HEX2 = HEX2ss;
assign HEX3 = HEX3ss;
assign SRAM_CE_N = OE? 1'b0 : 1'bz;
assign SRAM_UB_N = OE? 1'b0 : 1'bz;
assign SRAM_LB_N = OE? 1'b0 : 1'bz;
assign SRAM_ADDR = OEm? (OE? iADDR : 18'bzzzzzzzzzzzzzzzzzz) : 18'bzzzzzzzzzzzzzzzzzz;

reg [15:0] oDATA;
reg [2:0] count;
reg [15:0] SRAM_DQ;
//reg [17:0] SRAM_ADDR;
reg SRAM_WE_N,
reg SRAM_OE_N;

initial begin
  count = 3'b000;
end
always @(posedge iCLK)
begin

case(count)
    3'b000: begin //read
        SRAM_WE_N = OE?
        1'b1 : SRAM_OE_N = OE?
        1'b0 : SRAM_OE_N = OE?
        1'bz;
        oDATA = OE?
        (SRAM_OE_N ? 16'h1111:SRAM_DQ) : 16'hzzzz;
        count = count +
    end
    3'b001: begin //read
        SRAM_WE_N = OE?
        1'b1 : SRAM_OE_N = OE?
        1'b0 : SRAM_OE_N = OE?
        1'bz;
        oDATA = OE?
        (SRAM_OE_N ? 16'h1111:SRAM_DQ) : 16'hzzzz;
        count = count +
    end
    3'b010: begin
        count = count +
    end
    3'b011: begin //write
        SRAM_WE_N = OE?
        1'b0 : SRAM_DQ = OE?
        (SRAM_WE_N ? 16'hzzzz:dDATA) : 16'hzzzz;
        count = count +
    end
    3'b001: begin
    end
    3'b100: begin //zzzz
end
SRAM_WE_N = OE?
1'b1 : 1'bz;
SRAM_DQ = 16'hzzzz;
count = count + 3'b001;
end
3'b101:
begin
    count = 3'b000;
end
endcase
end

hex_7seg dsp0(.hex_digit(iADDR[3:0]),.seg(HEX0ss));
hex_7seg dsp1(.hex_digit(iADDR[7:4]),.seg(HEX1ss));
hex_7seg dsp2(.hex_digit(iADDR[11:8]),.seg(HEX2ss));
hex_7seg dsp3(.hex_digit(iADDR[15:12]),.seg(HEX3ss));

endmodule

booster.v

module booster
endmodule

booster.v

module booster
endmodule
always @ (negedgeiCLK)
beg

begin case (count) 3'b001: begin oADDR = aADDR; mflag = 1'b1; end 3'b010: begin temp = iDATA; end 3'b011: begin oDATA = temp << cat; end 3'b100: begin aADDR = aADDR + 18'h00001; end 3'b101: begin if (aADDR == 18'h00064) begin aADDR = 18'h00000; mflag = 1'b0; end end endcase end endmodule

**op_sum.v**

module op_sum

( stgs,
oADDR,
oDATA,
iCLK, count,
iDATA,
stgm,str1,cat
);

output stgs;
output [17:0] oADDR;
output      signed [15:0] oDATA;
//output      signed [23:0] sum;

input      iCLK, stgm, str1;
input      [6:0] count;
input      [4:0] cat;
input      signed [15:0] iDATA;

reg      [17:0] oADDR, rADDR, wADDR;
reg      stgs;
reg      signed [23:0] sum;
reg      signed [15:0] oDATA, exponent;

initial begin

rADDR = 18'b000000000000000000;
wADDR = 18'b000011000000010111://3017
stgs = 1;
sum = 24'h000000;
exponent = 16'h0000;

end

always @(negedgestgm) begin

exponent <= str1? 16'h0000 : (exponent + cat);

end

always @(negedgeiCLK) begin

stgs = 1;
if (count == 0) begin

  oADDR = rADDR;
rADDR = rADDR + 18'b000000000000000001;
end
else if (count >=1 && count <=99) begin

  oADDR = rADDR;
rADDR = rADDR + 18'b000000000000000001;
sum = sum + iDATA;
end
else if (count==100) begin

end

}
oADDR = wADDR;
sum = sum + iDATA;
end
else if (count == 101)
begin
  oDATA = sum[23:8];
oADDR = wADDR;
end
else if (count == 102)
begin
  oDATA = exponent;
oADDR = wADDR + 18'h00FE9;
end
else if (count == 104)
begin
  sum = 24'h000000;
rADDR = 18'b000000000000000000;
wADDR = wADDR + 18'b000000000000000001;
stgs = 0;
end
end
endmodule

SRAM_Interface_sum.v

module SRAM_interface_sum
(
  oDATA,
  count,
  SRAM_DQ,SRAM_ADDR,
  SRAM_UB_N,SRAM_LB_N,SRAM_WE_N,SRAM_CE_N,SRAM_OE_N,
  //GPIO_0, GPIO_1,
  HEX4,HEX5,HEX6,HEX7,
  LEDG,
iCLK,OE,
dDATA,iADDR
);

input iCLK,OE;
input [15:0] dDATA;
input [17:0] iADDR;
output [15:0] oDATA;
output [6:0] count;
inout [15:0] SRAM_DQ;
output [17:0] SRAM_ADDR;
output

SRAM_UB_N, SRAM_LB_N, SRAM_WE_N, SRAM_CE_N, SRAM_OE_N;

//inout [35:0] GPIO_0, GPIO_1;
output [6:0] HEX4, HEX5, HEX6, HEX7;
wire [6:0] HEX0ss, HEX1ss, HEX2ss, HEX3ss;
output [1:0] LEDG;

assign LEDG[1:0] = {SRAM_WE_N, SRAM_OE_N};
assign HEX4 = HEX0ss;
assign HEX5 = HEX1ss;
assign HEX6 = HEX2ss;
assign HEX7 = HEX3ss;
assign SRAM_CE_N = OE? 1'b0 : 1'bz;
assign SRAM_UB_N = OE? 1'b0 : 1'bz;
assign SRAM_LB_N = OE? 1'b0 : 1'bz;
assign SRAM_ADDR = OE? iADDR : 18'bz
reg [15:0] oDATA;
reg [6:0] count;
reg [15:0] SRAM_DQ;
//reg [17:0] SRAM_ADDR;
reg SRAM_WE_N,
    SRAM_OE_N;
reg flag;

always @(posedgeiCLK)
begin
if ((count <= 99) && (count>= 0))
begin//read
    SRAM_WE_N = OE? 1'b1 :
    1'b0;
    SRAM_OE_N = OE? 1'b0 :
    1'b0;
    oDATA = OE?
    (SRAM_OE_N ? 16'h1111:SRAM_DQ) :
16'hzzzz;
end
else if(count==101)
begin//write
    SRAM_WE_N = OE? 1'b0 :
1'b0;

SRAM_DQ = OE?
(SRAM_WE_N ? 16'hzzzz:dDATA) : 16'hzzzz;
end
else if(count==102)
begin
//write
  SRAM_WE_N = OE? 1'b0 :
  1'bz;
  SRAM_DQ = OE?
(SRAM_WE_N ? 16'hzzzz:dDATA) :
  16'hzzzz;
end
else if(count==103)
begin
//zzzz
  SRAM_WE_N = OE? 1'b1 :
  1'bz;
  SRAM_DQ = 16'hzzzz;
end
else if(count==105)
begin
  count = 7'b0000000;
  flag = 1;
end
if (!flag)
begin
  count = count + 7'b0000001;
end
flag = 0;
end

hex_7seg dsp0(.hex_digit(iADDR[3:0]),.seg(HEX0ss));
hex_7seg dsp1(.hex_digit(iADDR[7:4]),.seg(HEX1ss));
hex_7seg dsp2(.hex_digit(iADDR[11:8]),.seg(HEX2ss));
hex_7seg dsp3(.hex_digit(iADDR[15:12]),.seg(HEX3ss));
endmodule
References


[9] http://caonline.amcancersoc.org/cgi/content/full/57/2/75


[25] CST Studio Suite developed by Computer Simulation Technology (CST AG)


[27] www.ieee802.org/15/


[34] DE2 Development and Education Board User Manual, Altera Corporation


[38] Bucknell Handbook on Verilog HDL, Dr. Daniel C. Hyde

[39] The Verilog Language, Prof. Stephen A. Edwards