ACOUSTIC BEAMFORMING ON WIRELESS SENSOR NODES

By

Stephen William Collings

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

May, 2008

Nashville, Tennessee

To Melissa

ACKNOWLEDGEMENTS

This work would not have been possible without the financial support of the National Science Foundation, in the form of a MURI grant on "Heterogeneous Sensor Webs for Automated Target Recognition and Tracking in Urban Terrain". Thanks are also due to Dr. Akos Ledeczi for providing me with the protected academic time to pursue my degree. The advice of Dr. Gabor Karsai has been an invaluable help, as has that of Peter Volgyesi, Isaac Amundson, and Manish Kushwaha. I would especially like to thank Jon Fouss, Dr. Greg Nordstrom, Charlie Adkins, my parents, and most of all my fiancée Melissa for being so personally supportive as I completed this project.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

Beamforming a signal processing technique whereby the directionality of an array of transducers, either transmitting or receiving, may be controlled electronically [1]. This is frequently achieved by means of a phased transducer array. Electromagnetic phased arrays were first developed during World War II for use in ground-controlled approach radar systems. Similar systems have since been integrated into many fields, including AM and FM broadcasting stations, modern naval combat systems, and the communication system aboard NASA's MESSENGER space probe to Mercury. Phased ultrasonic speaker arrays are used in medical imaging, materials testing, and range finding [2], [3]. The mathematics and theory of acoustic beamforming are treated in some depth in [4]. The phased array principle may be applied equivalently to both receiving and transmitting arrays. Here we will primarily consider reception.

For the case of multiple ideal receiving transducers occupying different spatial locations, and disregarding the signal weakening over distance, the signals reaching each transducer are identical except for a phase shift caused by the differing distance between each transducer and the signal source. Thus, if the signals received by each transducer are phase shifted by values which cancel the phase shift caused by the differing transducer locations, the signals will regain phase cohesion. Summing the phase shifted signals will then result in constructive interference. Since signals from other directions will have different phase offsets, signals from the direction correlating with the selected phase shift values will have greater intensity than signals from any other direction.

The resulting directional preference may be swept across a range of directions by changing the transducers' phase offsets to match each angle being tested, creating the effect of a single rotating directional sensor. As the directional preference changes, constructive interference in the summed signal will tend to manifest most in the direction of the sound stimuli, while in other directions destructive interference occurs. Thus, if all other variables are held constant, the resulting comparatively high signal intensities will tend to correlate with the directions of nearby audible stimuli. This delay and sum operation is the basic principle of the time-domain beamformer.

Online beamforming obviously requires continual signal storage and processing. Because of this, deploying it on a wireless sensor node can be problematic due to power consumption and battery life issues. Implementing beamforming in hardware on an FPGA instead of in software on a standard microprocessor can help address this problem. Further, the utility of a beamformer may be significantly reduced due to noise in the region of operation. Implementing a frequency filter in conjunction with the beamformer can address this issue, by allowing the beamformer to only focus on sound frequencies of interest.

Chapter I has presented a general introduction to the material. Chapter II goes into more detail, giving background information needed to more fully understand the problem and proposed solution. Chapter III describes the hardware used for the project. Chapter IV gives the algorithm used by the hardware to perform beamforming and frequency filtering. Chapter V describes the implementation of this algorithm on the FPGA. Chapter VI presents details of the experimental setup, the experiments performed, and of the results of those experiments. In Chapter VII we give our conclusions drawn from this experimental data. Chapter VII discusses possible directions for future research.

CHAPTER II

BACKGROUND

Our purpose in implementing a beamformer is to obtain directional information about sound stimuli in the region around a wireless sensor node. To accomplish this, at regular intervals the node will perform beamforming on a set of evenly spaced angles around it. The collection of the signal intensities for each angle is known as the beamform. Beamforms are commonly presented as polar plots, to visually convey the relative directional intensities.

Just as the beamform will have maximal intensity at angles where the phase shifted stimuli are most nearly in phase, the beamform will only have minimum energy in the directions at which two or more phase shifted signals most nearly cancel each other. As the phase shifts change with the angle being tested, the constructive interference will decrease until the signals reach this point of maximum cancellation. These minima thus depend on the frequency content of the signal being received and the physical layout of the microphones. If there is more than one minimum, the beamform will be composed of multiple lobes, as the phase shifts move beyond the point of least constructive interference and begin increasing in signal cohesion again. Typically, each of these other lobes possesses some fraction of the intensity of the primary. Polar plots of various beamforms, some of which display this behavior, may be seen in Figure 19-22.

The nature of the secondary lobes varies with the stimuli presented to the beamformer. For low frequency signals, a phase shift of some small fraction of one period reduces the phase coherence of the incoming signals by a relatively small amount. However, at higher frequencies a phase shift of the same time length would cause a much greater decline in the phase coherence of the incoming signals, and a more rapid crossover from destructive interference back to constructive. Thus, for constant sampling rates and phase shifts, higher frequency stimuli will typically result in a greater number of minima in the beamform, and thus a greater number of narrower lobes.

The utility of a beamformer may be reduced in a noisy environment, as multiple sources may be difficult to distinguish in the final beamform, and only some sources may be of interest. Frequency filtering may therefore be useful in attenuating the noise level in a given environment. As the delayed and summed signal is equivalent to the original signal, the frequency filter may be applied after the delay and sum operation, consuming fewer resources than filtering each channel individually.

One simple way to perform selective frequency filtering on a discrete signal is to perform a fast Fourier transform (FFT). The FFT is a computationally efficient method for transforming a discrete time-domain signal into its frequency-domain equivalent, with the most common implementation being the Cooley-Tukey algorithm first presented in [5]. An N-point FFT divides a signal into N frequency bins of equal width, with the maximum frequency represented being the Nyquist frequency of the signal, or half the sampling rate. Manipulating the values in the frequency domain allows us to filter the signal's frequency content as desired. Since only the intensity of the signal is of interest, and since the overall energy of the signal is the same in the time and frequency domains, no inverse transform is required.

Many other acoustic beamformers have been implemented. However, these beamformers are frequently not well suited to independent deployment as part of a wireless sensor network due to size and power constraints. Wireless networks of any size must be able to run for useful periods of time without intervention. Because of this, power is a primary concern in wireless sensor nodes. Even implementations designed for low power consumption such as in [6] are typically implemented in software on a general-purpose microprocessor.

A more energy efficient solution would be to use a reprogrammable hardware logic device, such as a field-programmable gate array (FPGA). These semiconductor devices contain user-configurable logic blocks, and can be programmed using VHDL (VHSIC Hardware Description Language) to implement any logic function within the FPGA's resource limits [7]. The individual logic blocks, frequently called slices, are typically composed of one or more lookup tables, one or more flip-flops, control signals, multiplexers, and other assorted logic. Since FPGAs can perform many operations in parallel, they can perform an equivalent number of computations in fewer clock cycles than a single-core general purpose microprocessor would require. Thus, an FPGA does not require as fast a system clock as a microprocessor, and is likely to consume significantly less power.

To encourage code reuse and thus save development time, many blocks of VHDL, known as cores, are available for public use. Some cores are open-source, others are released under commercial licenses by private companies. Appropriate use of these cores can greatly speed a development project.

Many other hardware-based beamformers have been implemented, both for general use and specific purposes including radar processing and sonar processing [8],

[9], [10]. Hardware acoustic beamformers have also been implemented, such as [11], using a CPLD. However, an FPGA-based acoustic beamformer has not to our knowledge been previously designed for use in a wireless sensor network.

CHAPTER III

HARDWARE

The system is implemented in VHDL on a Xilinx Spartan XC3S1000. As specified in [12], this FPGA is composed of 1,920 four-slice configurable logic blocks and 24 block RAMs. Each block RAM is capable of storing sixteen kilobits of data and two kilobits of parity information, with word widths configurable at synthesis. The system clock runs at 20 MHz.

The FPGA is mounted on a sensor board (Figure 1), which was designed and built for [13]. This board provides the FPGA with a JTAG interface, a UART serial interface, connections for four microphones, and an I^2C interface to a MicaZ mote. The JTAG and UART interfaces provide programming and run-time control of the FPGA via a PC. The microphones collect the sound samples for beamforming and route them to a 1 MHz A/D converter, and from there to the FPGA. The mote uses its 802.15.4/ZigBee compliant radio to relay the FPGA's results to the rest of the sensor network, which can include a PC base station. The mote's radio is specified in [14] as being capable of transmitting 250 kbps, and as having an outdoor range of 75-100 meters and an indoor range of 20-30 meters. Other peripherals such as Bluetooth are available on the sensor board, but are not used in this project.

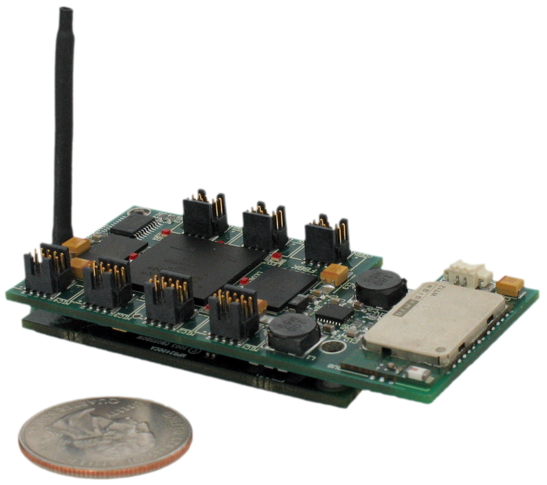


Figure 1: Sensor board and MicaZ mote

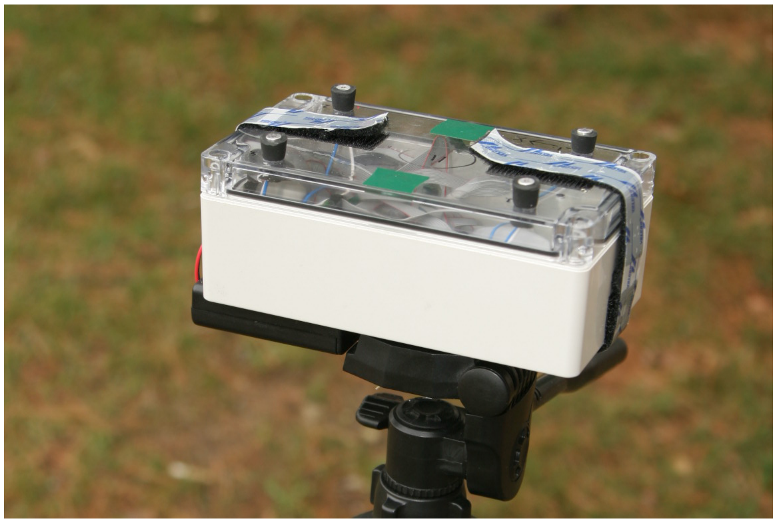


Figure 2: Tripod-mounted project box

The FPGA has 256 eight-bit externally accessible registers, which are accessible both via the sensor board's serial interface and by TinyOS programs running on the mote. These registers are used to specify parameters for the operation of the FPGA, and to store results to send to the PC base station. An intermediary device connected to the board's serial port provides a standard terminal interface to any computer with USB host capability and the appropriate drivers.

The hardware is contained in a plastic project box, approximately 16 cm x 9 cm x 6 cm (Figure 2). Four Panasonic WM-64PN microphones protrude approximately 1 cm from the top of the box in a rectangular array, approximately 10.5 cm x 7 cm. The frequency response of the WM-64PN as given in [15] is approximately flat for all frequencies of interest, as can be seen in Figure 3. The microphones are assumed to be

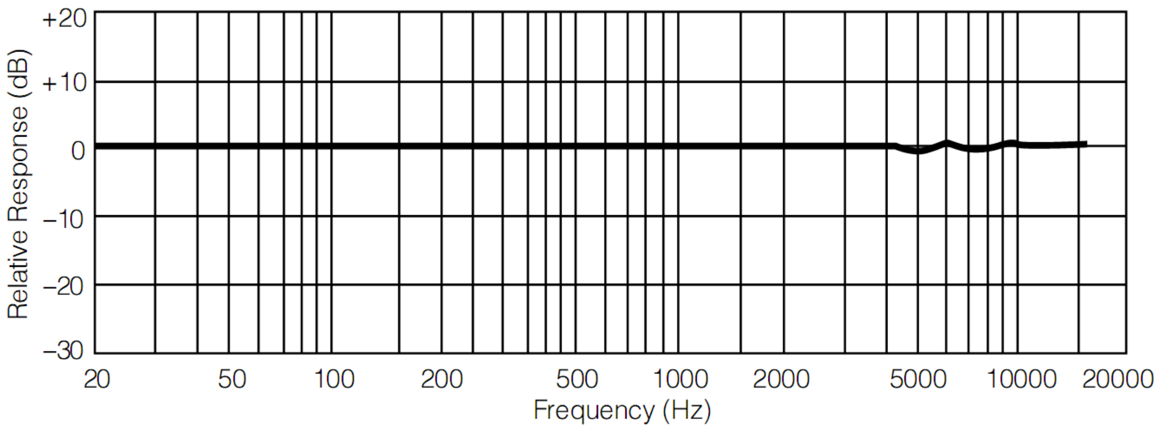


Figure 3: WM-64PN microphone frequency response

omnidirectional, as specified in [15]. It is also assumed that any variation in the response of individual microphones is insignificant.

The system can be powered by batteries mounted on the mote, via the serial port from a PC's USB port, or from a battery pack mounted outside the plastic box. The external battery pack holds four AA batteries and a 3.3 volt regulator. The box assembly is tripod-mountable, allowing for easy and stable placement of the sensor node.

CHAPTER IV

BEAMFORMER AND FILTER ALGORITHM

The following pseudocode describes the beamformer and filter algorithm:

```
on arrival of samples
   store samples in circular input buffers
   if number of samples since last beamforming is sufficient
      for each angle to be tested
         find the distance between each mic and the source at that angle
         compute the offsets to cancel the phase shifts for each mic
         select and sum samples from each buffer using phase offsets
         store sum in active filter buffer for angle
         if active filter buffer set is full
            switch active and secondary filter buffer sets
            for each angle to be transformed
               serially load appropriate filter buffer into FFT core
               on transform completion
                  sum output values, excepting filtered frequencies
                  exponentially average sum with previous output value
                  output new value
```

Incoming samples are stored in circular buffers, one buffer for each microphone. Since beamforming may not be completed in the time between two incoming samples, some downsampling of the incoming signal may be required, effected by performing delay and sum operations at a lower frequency than that of the incoming samples. The basic method of determining the decimation factor *n* is to use the Nyquist criterion to identify the sampling rate which passes all frequencies of interest and filters out all others. Dividing the input sample frequency by this Nyquist frequency gives the maximum acceptable decimation factor, *n*. We downsample the signal by taking one of every n samples to form our new signal with a lower sampling rate.

When beamforming is initiated, the system iterates through the angles to be tested. For each angle, the samples stored for each microphone are phase shifted by an appropriate value for the microphone and sound source location, and the phase shifted samples summed. The phase shift calculation is based around the signal sampling rate, as well as the distance between the microphone in question and the sound source location being tested. To compute this distance, a distance between the sound source and the coordinate system origin must be assumed. This radius value is essentially an arbitrary constant for the purposes of this algorithm. Since all directions are tested at the same distance, it serves only as a scaling factor, useful for keeping all phase offsets within a given range while maintaining as much precision as possible.

Once the beamform energies for each angle are computed, they are stored in buffers, one buffer for each angle. When enough beamforms have been computed to fill the buffers, a discrete Fourier transform is performed on each buffer in series. Since the energy of a signal is the same in the time and frequency domains, the sum of the Fourier transform of the beamform energies is equal to the sum of the original beamform energies. Zeroing out a given frequency component of the Fourier transform before performing this sum reduces the computed energy at the angle in question, in proportion to how much that frequency contributed to the overall energy of the sound source.

The new output energy is computed by performing an exponentially weighted average of the filtered sum and the previous output energy for the angle in question.

This helps smooth the output results between transforms, reducing output jitter at minimal resource expense.

CHAPTER V

BEAMFORMER AND FILTER IMPLEMENTATION

This project was implemented in VHDL on the FPGA as modules in a pre-existing framework of code. The code dealing with direct hardware interfaces was written for [13], and as such those modules will not be addressed in detail here. Three modules were implemented for this project: the FFT core, the FFT filter block, and the beamformer module. The interfaces between these modules are displayed in Figure 4. The only other modules of immediate relevance are the register module and the A/D converter interface.

The beamformer module accepts eight-bit samples from the four A/D converters in parallel, at a rate of 1 MHz. These samples are stored in four circular buffers, each buffer implemented as a single block RAM storing 2048 samples. On power-up, the buffers are first preloaded to ensure beamforming operations are only performed on

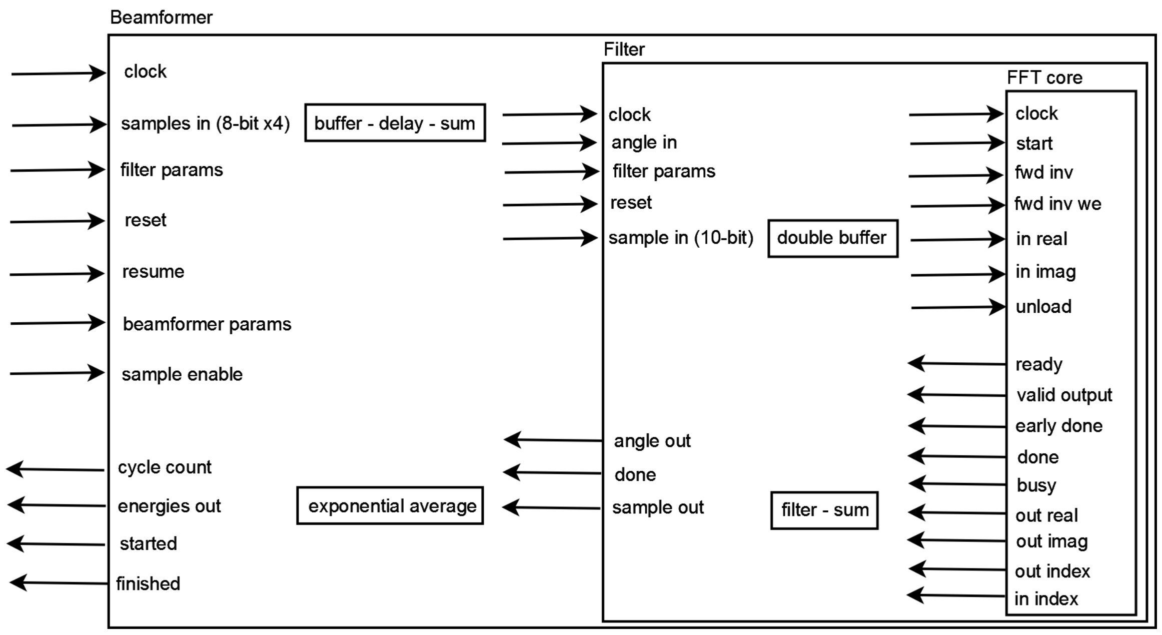


Figure 4: VHDL blocks

valid data. Once the buffers are full, beamforming commences at 25 kHz, on the arrival of every fortieth sample, effectively downsampling the incoming signal.

During beamforming, the module iterates through the 36 angles, retrieving the appropriate offset value for each microphone for that angle from a lookup table. This lookup table is implemented in a block RAM and contains sixteen-bit precomputed delay values for each combination of microphone and angle. These delay values are computed on a PC by a Python script prior to VHDL synthesis. The table is needed due to the FPGA's inability to perform floating point arithmetic, and thus the necessary trigonometry. The arbitrary radius value in the LUT calculations is set as .49 meters, which is the largest value such that all offset values can be stored in sixteen bits.

Once the phase offset values are retrieved for a given angle, those values are subtracted from the addresses of the newest samples in the circular buffers at the time beamforming began. This gives the addresses of four samples, one from each channel, that would have been generated at the same time had the sound source been in the direction currently being analyzed. Those four values are summed, giving the beamform energy for the current angle. This energy and the angle are fed to the FFT filter block.

The FFT filter block is composed of 36 double-buffers, one double-buffer for each angle. These buffers are implemented using 18 block RAMs, with each block RAM comprising four buffers. Each buffer stores 256 ten-bit samples from the beamformer module. As there are only 24 block RAMs on the FPGA, the filter block is the largest user of those resources, and is thus strictly limited as to the number of angles it is capable of handling at once. The buffers are divided into active and secondary banks. Incoming samples are stored in the active buffer bank, until that buffer bank is full. The secondary bank becomes the active bank and continues storing incoming samples, while the previously active bank begins feeding one buffer at a time serially into the FFT core.

The FFT core is a serial load 256-point Xilinx LogiCORE^{TM} IP FFT core, described in [16]. This core uses the Cooley-Tukey FFT algorithm, first presented in [5]. As configured for this project, it accepts and outputs complex values, with eleven-bit real components and twenty-bit imaginary components. The core is configured to use distributed RAM instead of block RAM, to save system resources.

When the FFT core completes a transform, it serially outputs the results. As each result bin is output, the magnitude of each complex component is taken, and the sum of these magnitudes taken. If the frequency bin is selected in the user parameters to be passed by the filter, this sum is added to an overall sum for the transform. Since the filter's sampling rate is 25 kHz, its Nyquist frequency is 12.5 kHz. Dividing this among 256 frequency bins shows us that each bin has a frequency width of 48.3 Hz. When all passed frequency bins have been summed, this sum is output to the beamformer block, along with the angle with which it is associated.

The beamformer block accepts this sum and exponentially averages it with 256 times the previous output energy for that angle, to generate the new output energy. Due again to the lack of an FPU, the divisor used in the exponential averaging must be a power of two so that the division may be performed by simple bit shifting. The user can control the number of bits to be shifted via a parameter register.

Parameter registers are available to control various beamformer and filter attributes at runtime (Table 1). Num_beams is mainly a holdover from earlier designs in which the number of angles checked was variable. However, in the present implementation it might still be useful if one wished to analyze lower angles at a higher

Table 1: Beamformer Run-Time Parameters

Name	Description	Default Value
Beamform_rate	Decimation factor, number of samples between beamforming operations	40
Report_rate	Number of cycles between result reports	250,000
Smooth_factor	Number of bits right shifted for exponential average	5
Mic_positions	Associations for which microphone is in which physical location on top of the box	1,2, 3, 4
Num_beams	Number of beams to check (0-36)	36
One_shot	If true, beamformer reports results once and stops	'0' (boolean)
Wait_time	Number of cycles after power-on before beamformer begins operations	8
Low_bit	Bit of the FFT core output selected as LSB of output value	2
Filter_choice	256 bits, one for each FFT bin; '1' if that frequency is passed, '0' if it is blocked	All '1'

data rate. One_shot and wait_time are primarily for debugging purposes. Low_bit is variable since, due to the summing of the four input signals, it is theoretically possible for the exponentially averaged output signal to be a ten bit number. If necessary, this parameter may be modified for a given application to ensure that the greatest precision possible is output, while still avoiding overflow of the output registers.

Once all the buffers have been transformed, the smoothed sum results are stored in the output registers on the FPGA. At a specified frequency, the FPGA signals the mote to read those registers, which then transmits the results to another mote connected to a PC. This PC may then correlate results from multiple sensor nodes to generate a fuller picture of their environment. After the FPGA signals the mote, the FPGA ceases to write to the output registers until signaled by the mote that the results have been read. This is to ensure that the output registers do not change during reading, and thus that the mote reads a single consistent data set. If no signal is received by the mote within 10,000,000 cycles (.5 seconds), the FPGA resumes operations.

This implementation uses 5020 of the FPGA's 7680 logic slices, for 65% utilization. It uses 3697 of the FPGA's 15360 slice registers, for 24% utilization. The beamformer module requires four block RAMs for buffers and one for the offset lookup tables, for a total of five. The FFT filter module uses eighteen block RAMs for the double-buffers. The remaining block RAM is used by the pre-existing serial interface.

When provided with 2.4 volts via the mote's battery pack, the entire assembly consumes 110±1 milliamps of current, or 260±2.4 milliwatts of power. At this power consumption rate, a pair of standard 2400 milliamp-hour NiMH AA batteries could power a constantly transmitting sensor node for approximately 42 hours. When provided with 3.3 volts from the project box's external battery pack, the system consumes 134±1 milliamps of current, or 442±3 milliwatts of power. At this rate, four 2400 milliamp-hour batteries could power the node for 72 hours.

Pre-loading the sample buffers requires the arrival of 2048 samples, which at the sampling rate of 1 MHz takes 2.048 mS. Performing an individual sum and delay operation takes four cycles (200 nS). The time from the first sample arriving to the final FFT buffer is filled is 204,026 cycles (10.2 mS). It takes 256 cycles (12.8 μS) to input one set of data into the FFT core, and the FFT itself lasts 1114 cycles (55.7 μS). Summing and exponentially averaging the results of one transform takes 263 cycles (13.15 μS). It thus takes 58,788 cycles (2.9 mS) to transform an entire bank of 36 buffers. In total, from first sample to a complete result set, a single beamform run takes

262,814 cycles (13.1 mS).   Because of double buffering in the FFT filter, beamforming completes every 204,026 cycles (10.2 ms). Reporting results at any faster rate would provide no new information.

CHAPTER VI

EVALUATION

Experimental Setup

The tests were performed at Vanderbilt Medical Center in the Bill Wilkerson Center's Anechoic Chamber Laboratory. This is a room measuring 4.6m x 6.4m x 6.7m, with all six surfaces covered by large fiberglass wedges. The chamber is designed to absorb sounds, effectively eliminating multipath effects and uncontrolled stimuli from consideration in acoustic experiments. The Center reports that the chamber has a measured cutoff frequency of 100 Hz. A wire mesh is suspended above the floor to provide a walking surface with minimal acoustic reflectivity.

Inside the room is an inward-facing ring of 64 evenly spaced (5.625 degree separation) speakers approximately 3.4 meters in diameter, suspended approximately 1.5 meters above the floor. These speakers are controlled from an external control room by three computers interfaced to Tucker-Davis System 2 and System 3 signal acquisition and processing devices. The 64 speakers are divided into two alternating groups, odds and evens. A group must play a single sound, but within the group what speakers play that sound are selectable from the control room. Due to this, experiments where one might normally set up two different sounds directly opposite each other are not possible. The resulting 5.625 degree offset is considered acceptable, given that the beamformer itself has only 10 degree resolution.

The beamformer unit was placed on a tripod in the center of the speaker circle, with the zero degree line pointing towards the speaker designated speaker 1. The top

Table 2: Frequencies Tested

Pure Tones	Band Limited Samples
500 Hz	500-600 Hz
1 kHz	1-1.1 kHz
1.5 kHz	1.5-1.6 kHz
2 kHz	2-2.1 kHz
3 kHz	3-3.1 kHz
4 kHz	4-4.1 kHz
6 kHz	6-6.1 kHz
8 kHz	8-8.1 kHz
10 kHz	10-10.1 kHz

of the unit was approximately 25 cm below the bottom of the speaker array. Experiments were performed with three different types of sound samples: white noise, band limited white noise, and pure tones. All band limited samples used had a frequency width of 100 Hz. Three different types of experiments were performed with these three stimulus types.

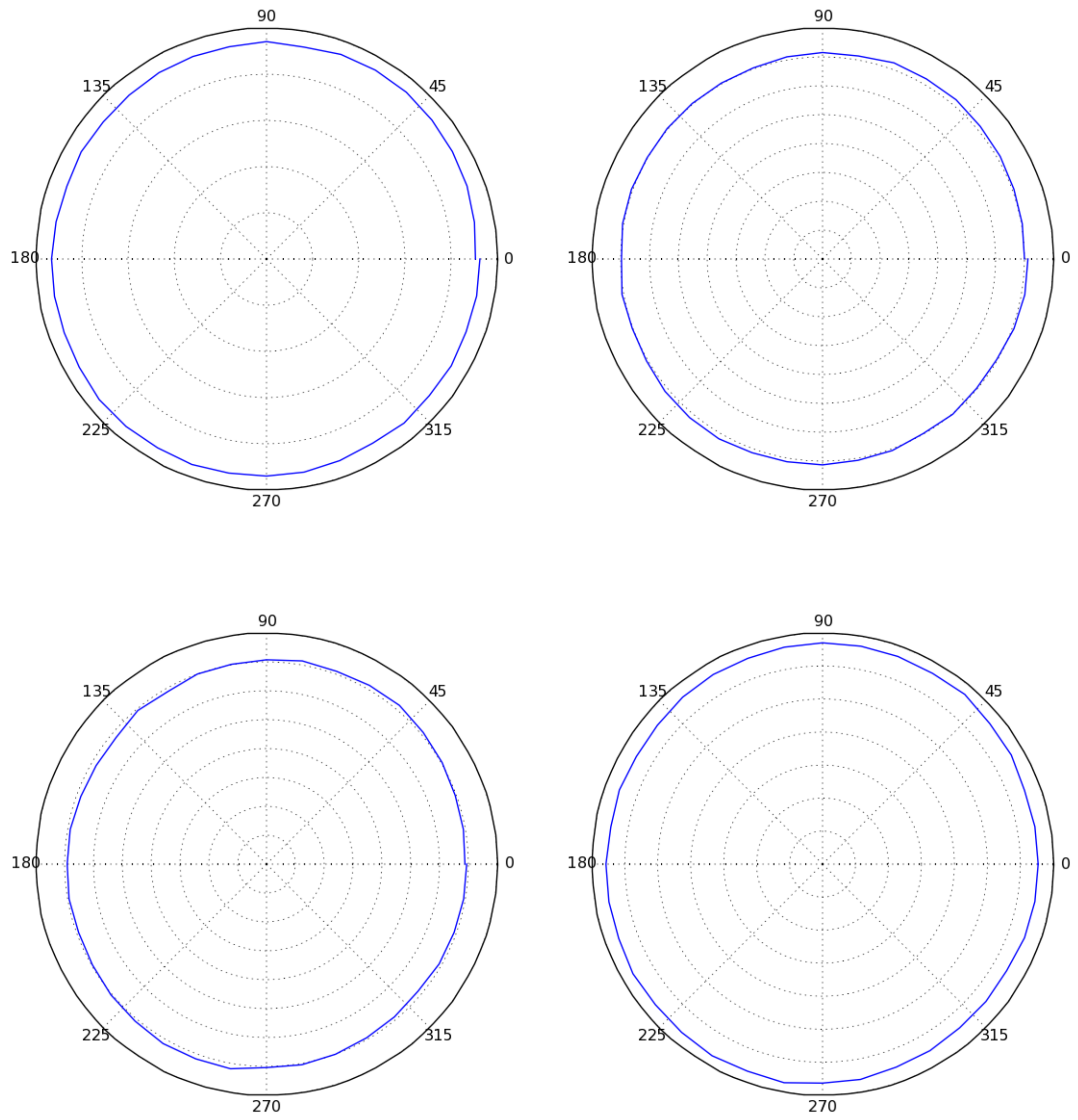
1. Single-source accuracy and symmetry. Each speaker was successively programmed to emit a given sound at a consistent intensity. For pure tone runs, the sound used was 1 kHz. For band limited runs, the sound used was 1-1.1 kHz. The beamforms recorded demonstrate both whether the beamformer constantly indicates the direction of the sound source, and whether it's response is approximately similar in shape and intensity regardless of the angle of the sound source.
2. Single-source frequency response. Speaker 1 was successively programmed to emit sound samples at varying frequencies, listed in Table 2, both pure tones and band-limited noise. The beamforms recorded demonstrate how the beamformer responds to different frequencies and frequency bands.
3. Filter response. Speaker 1 (0 degrees) was programmed to emit a consistent 1 kHz pure tone or 1-1.1 kHz band limited sound, while another speaker at a chosen angle was programmed to emit a sound of the same type at 2 kHz. The register bits controlling the frequency filter were incrementally cleared, giving a

highpass filter with a cutoff frequency increasing over time. The resulting beamforms, recorded after each cleared bit, demonstrate the changes to the beamformer's frequency response caused by the filter. Separation angles of 84.4 and 129.4 degrees were tested, using speakers 16 and 24. Care was taken to ensure that each time a filter bit was cleared, enough time went by before any results were recorded for the exponential averaging to catch up with the filtered energy values.

Consistency of sound intensity was maintained by manually adjusting the volume of each channel so that a meter measuring the intensity of the signal sent to the speakers remained relatively consistent within a given set of experiments. Perfect accuracy was unattainable due to reading fluctuations, especially with white noise and band limited stimuli, but the same displayed intensity was typically maintained within 1 dbV.

Results

One hundred data sets were recorded for each experiment run, each data set being composed of a single 36-angle beamform. For the first two experiment types, each of these data sets was linearly scaled such that the minimum beam value translated to zero, and the maximum translated to 1. These scaled sets were then summed beam by beam across the run, and this sum itself scaled, giving a summed scaled beamform (SSB) for each run. Since all values in the SSB are between 0 and 1, level variations between data sets are discarded, giving a more informative picture of the beamformer's typical directional response to a given stimulus set. Due to this, each plot's scale is independent of any other; comparisons of energy levels are only meaningful within a given plot, and not between plots.



For reference, data was recorded in the chamber with no sounds playing through the speakers. The beamformer was rotated to 0, 90, 180, and 270 degrees, and 300 data sets recorded at each angle. At each angle of rotation, a very low energy was reported in all directions, less than 10% of a typical experimental value. The average beamforms were very nearly circular, as may be seen in Figure 5-8; the range between

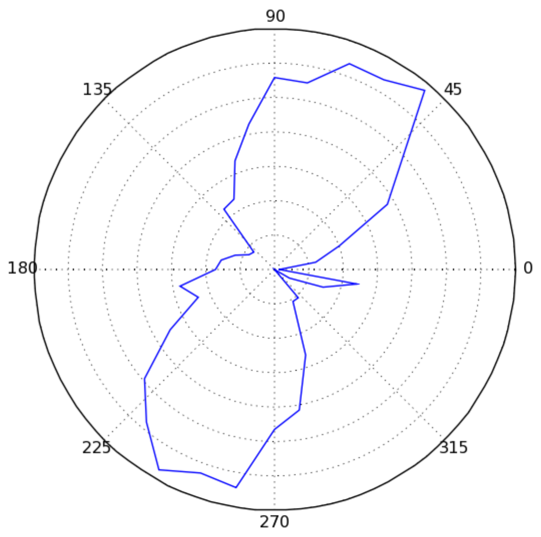


Figure 9: Ambient stimuli SSB

the greatest and least energy was at most 5.6% of the least energy. Within this small variation, however, the beamformer displays a slight but definite bias. The SSB of all 1200 of these data sets taken together is displayed in Figure 9. It is apparent that the beamformer consistently indicates the presence of stimuli at approximately 60 and 240 degrees, regardless of the direction the beamformer is oriented. This would indicate that this result is in some way a function of the beamformer itself, and not of any unidentified stimulus in the chamber.

It is possible that an error in the beamforming code is responsible for this, though it is difficult to imagine just what sort of error would cause such a result. It is also possible that the beamformer hardware emits a low intensity sound, which is picked up by the microphones, or that an asymmetry in the speaker array is somehow responsible. In any case, this result is unlikely to have any great impact, as the average variation in the returned beamforms is less than 1% of the energy in any experiment involving audible stimuli. The effect is only noticeable after scaling the data to between 0 and 1. As subtracting this average variation from other experimental beamforms produced no visually notable effect, we will assume this to be negligible for the purposes of our experiments.

For the accuracy and symmetry experiments, the sound samples from all 64 speakers resulted in visually similar beamforms within an experiment type. Examples can be seen in Figure 10-12. Further, the maximum intensity of each beamform varied

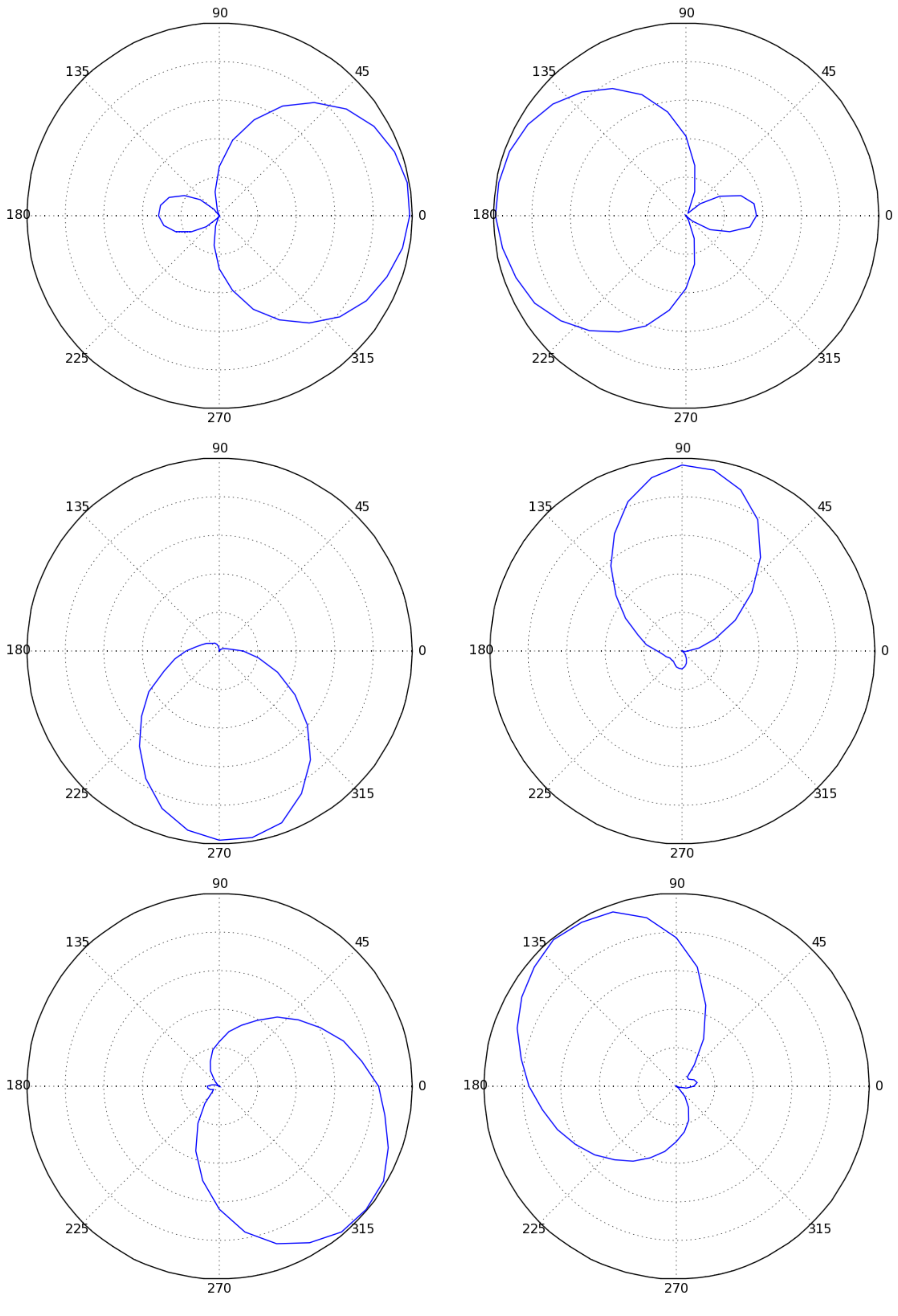


Figure 10: Clockwise from upper left: 0, 180, 90, 135, 315, and 270 degree band limited white noise beamforms

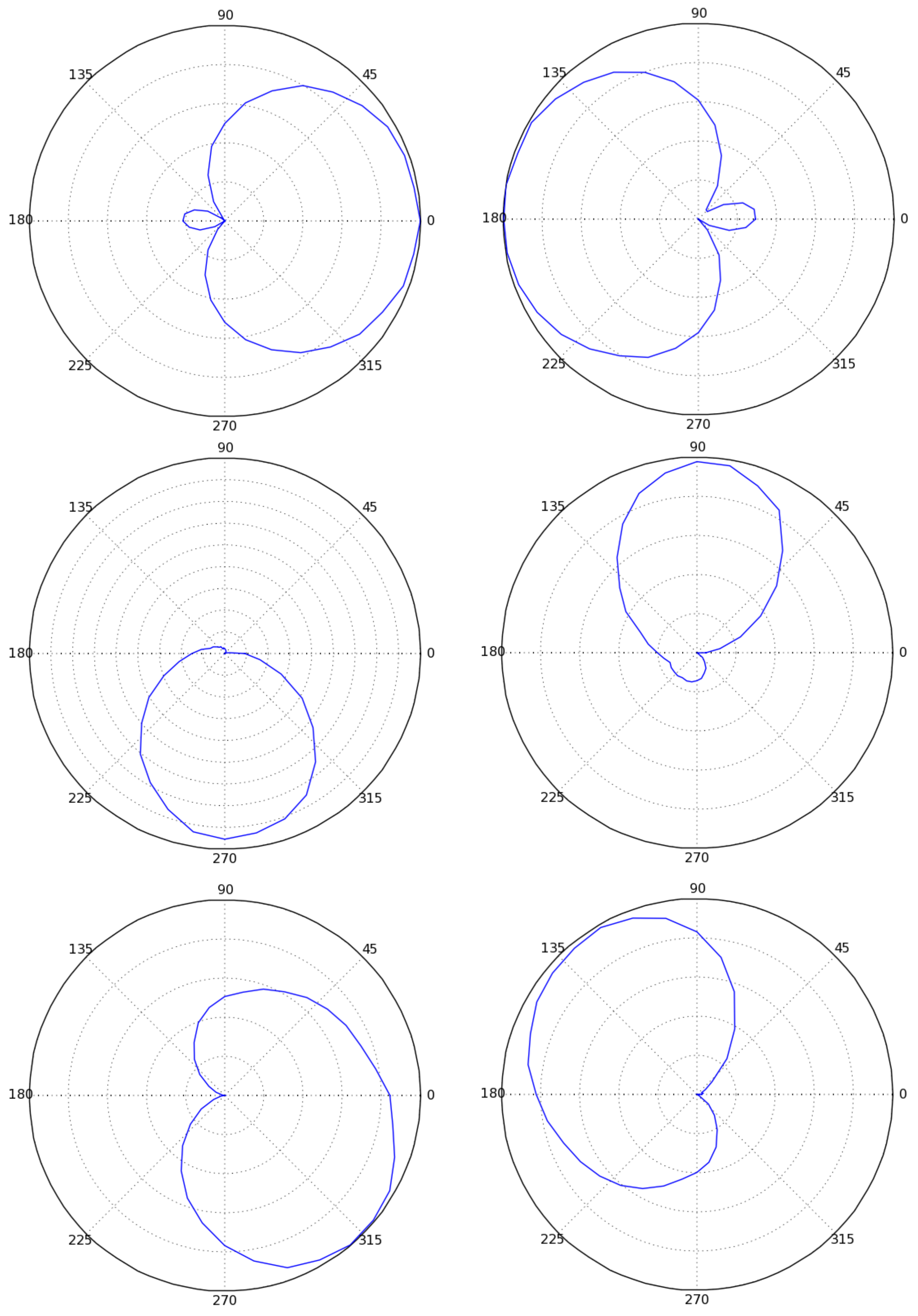


Figure 11: Clockwise from upper left: 0, 180, 90, 135, 315, and 270 degree pure tone beamforms

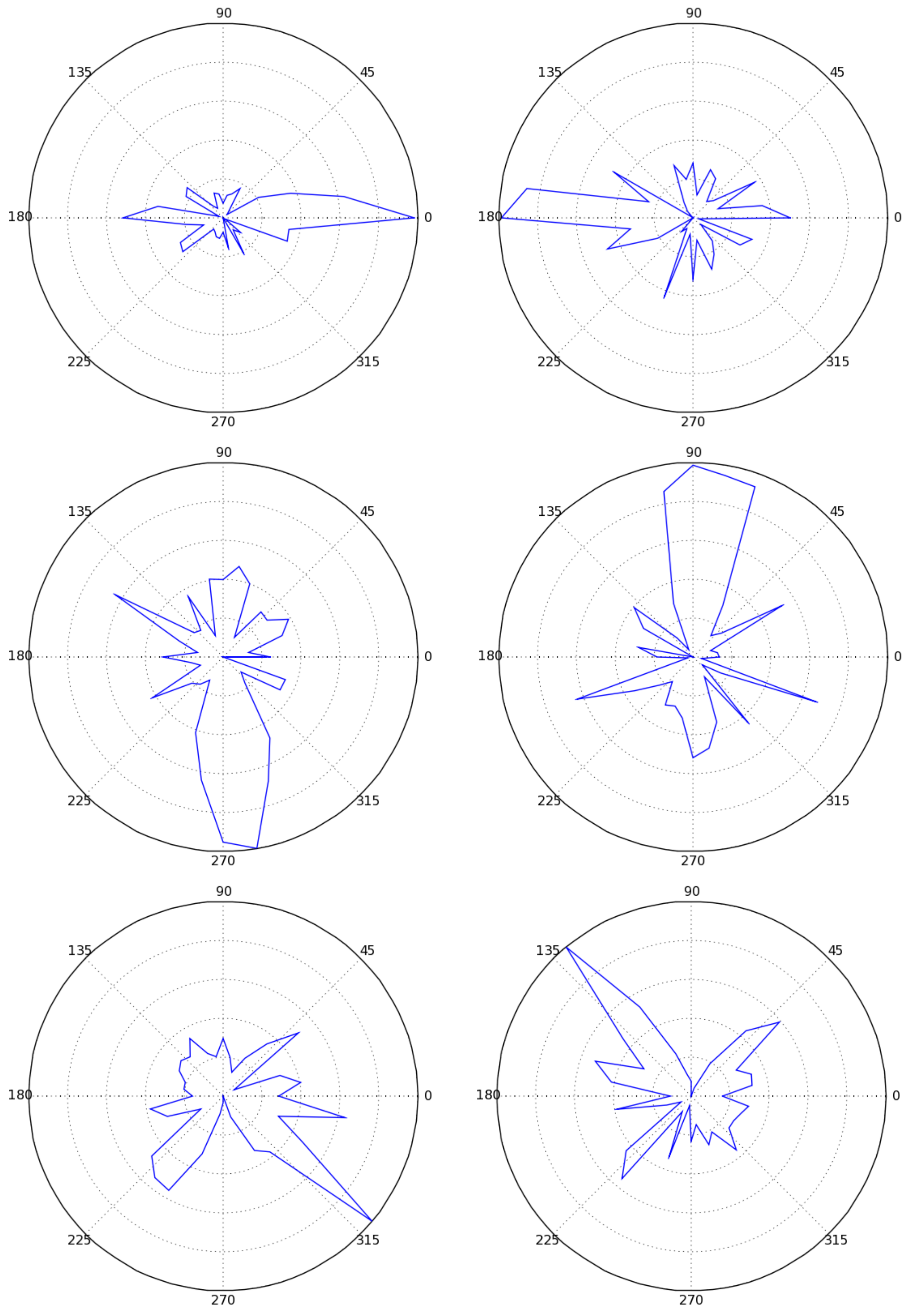


Figure 12: Clockwise from upper left: 0, 180, 90, 135, 315, and 270 degree white noise beamforms

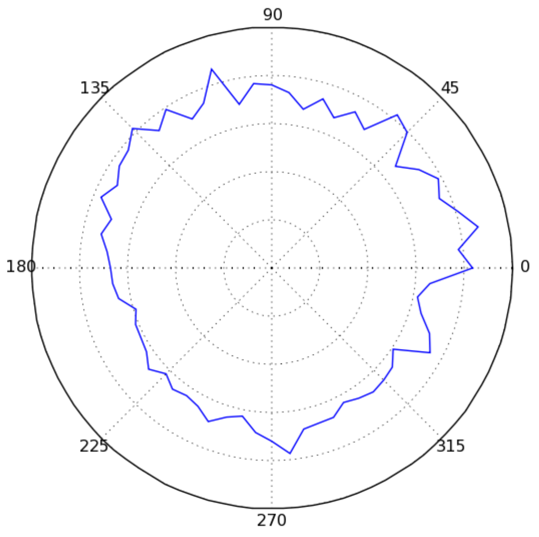


Figure 13: Band-limited maximal intensities

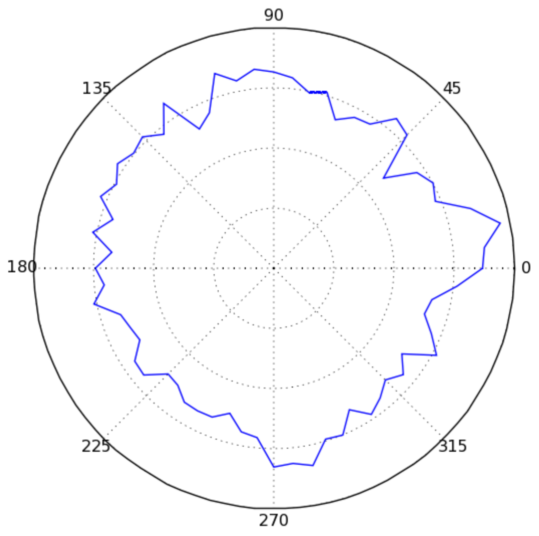


Figure 14: Pure tone maximal intensities

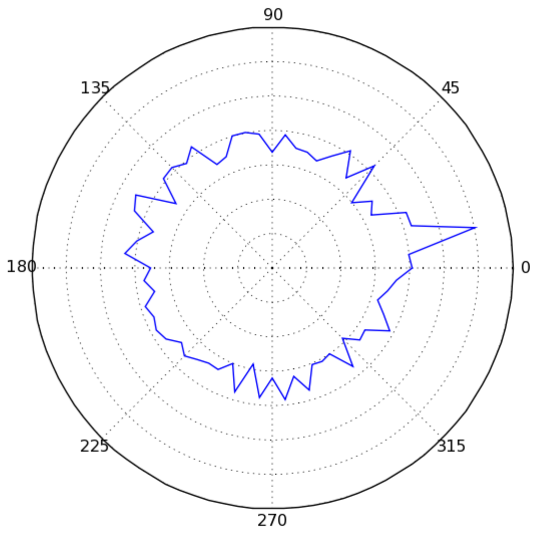


Figure 15: White noise maximal intensities

little within an experiment set, as can be seen in Figure 13-15. This indicates that the beamformer operates in an unbiased fashion, not favoring any one direction over the others. The beamformer's much sharper response to white noise, as compared to band-limited and pure tone stimuli, is due to the wide band of frequency components making accidental constructive interference far more unlikely. Still, many smaller lobes are present at different angles, due to the beamformer's response to the high-frequency components of the stimulus.

The accuracy of the beamformer is judged by analyzing the errors in its results. Error is defined as the difference between the direction the beamformer indicates that the sound is coming from and the direction from which the stimulus is actually

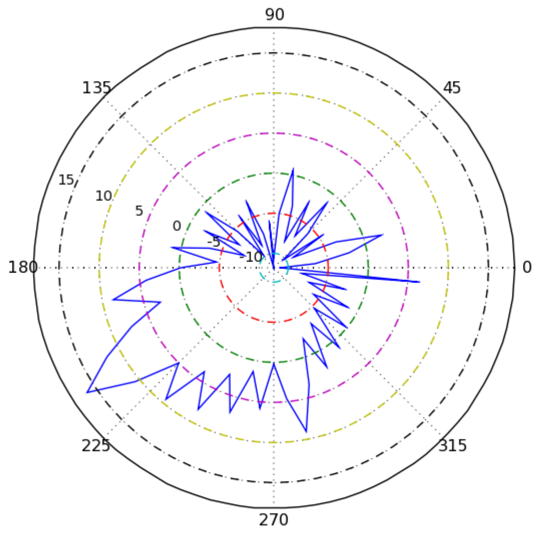


Figure 16: 1-1.1 kHz band limited directional error

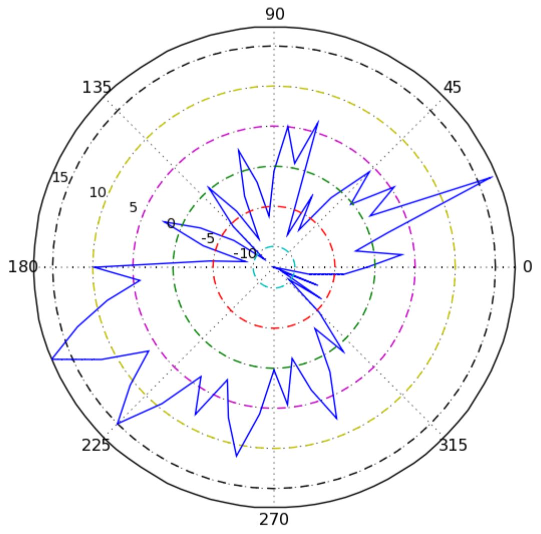


Figure 17: 1 kHz pure tone directional error

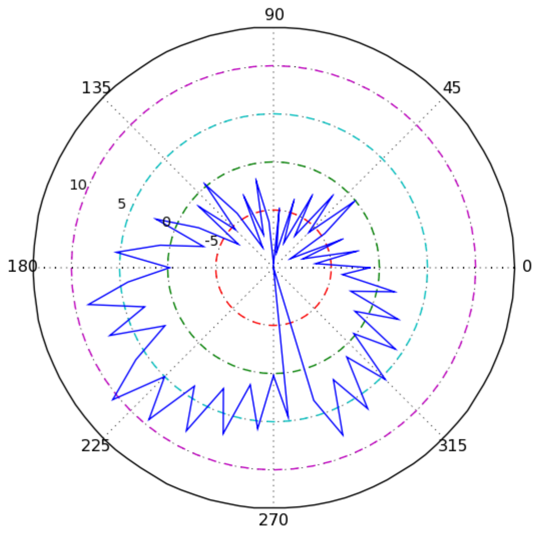


Figure 18: White noise directional error

originating. The direction indicated by the beamformer is taken to be the direction in which the device reports the greatest energy. Other metrics are conceivable, but here this simple method will suffice. Due to the beamformer having only 10 degree resolution, some error is unavoidable. Optimally, error will be constrained to within five degrees of the correct direction.

The error for the SSB for each angle tested was computed for each stimulus set. Plots of these errors are displayed in Figure 16-18. As can be seen, white noise gives the most accurate results, while pure tones give the least accurate. This is likely related to the sharpness of the white noise beamforms. As can be seen in Table 3, in no case is an error of more than 18.4 degrees reported, the mean error is always between -0.3 and 1.9 degrees, and the RMS of the error is never greater than 7.3 degrees. This

Table 3: Beamformer Error Statistics

	Mean	RMS	Max	Min
Pure tone	1.9°	7.3°	18.4°	-11.6°
Band limited	-0.3°	6.4°	17.2°	-10.8°
White noise	0.6°	5.4°	11.6°	-10.0°

indicates that the beamformer gives largely accurate results, for a reasonable variety of stimulus types.

It appears from Figure 16-18 that there is a tendency for the beamformer to report larger positive errors for sound sources in the 180-270 degree quadrant, regardless of the stimulus type. It is likely that this is due to some small asymmetry in the test system. The microphones themselves are likely to differ slightly due to manufacturing tolerances. As the project boxes are hand modified, the microphone locations and orientations may not be precisely identical, leading to asymmetries such as this one. It is also possible that since the placement and orientation of the box was done by hand, they may both be slightly off of true. It may even be that the minor asymmetrical variance detected in the ambient noise tests is responsible, though the magnitude of that effect makes this unlikely.

For frequency response tests (Figure 19-22), it is apparent that the beamformer results become more ambiguous with higher frequency stimuli. As frequency increases, the number of lobes in the beamform also increases, and the lobes present narrow, as expected. The greatest energy still tends to be in the direction of the actual sound source, but at higher frequencies the other lobes' energy can almost match that of the primary, reducing the reliability of the beamform. The band limited beamforms tend to be somewhat cleaner than the pure tone beamforms, with fewer lobes and stronger emphasis in the direction of the stimulus.

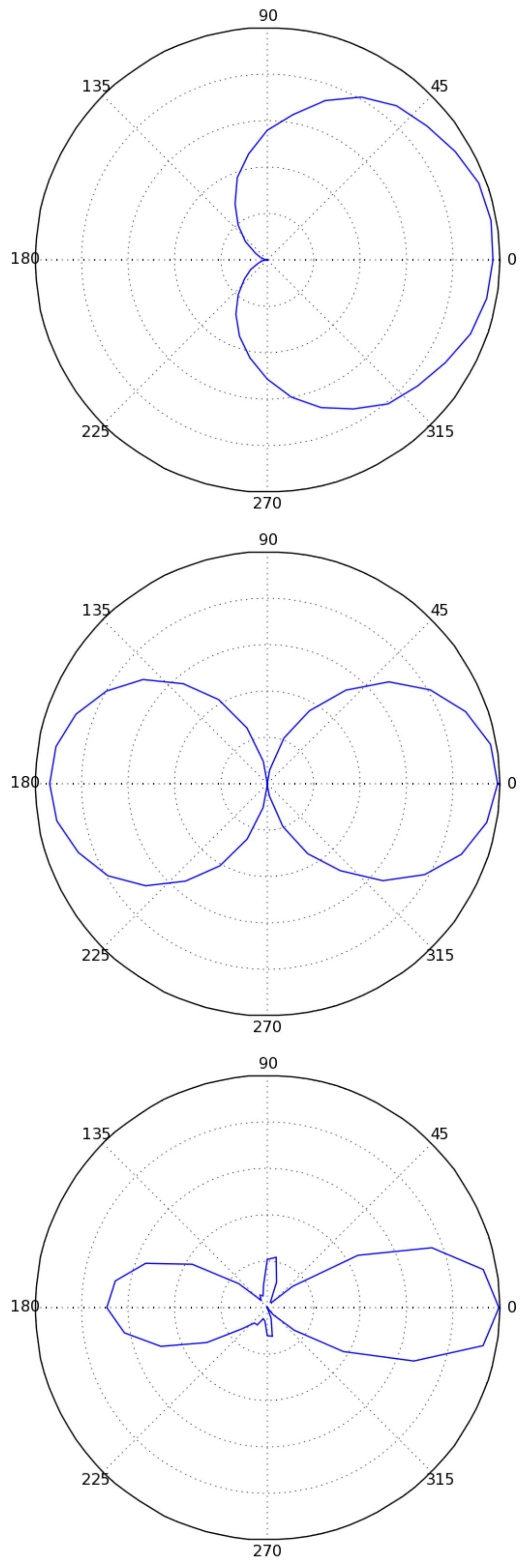


Figure 19: Clockwise from upper left: .5 kHz, 1 kHz, 2 kHz, 4 kHz, 3 kHz, and 1.5 kHz band limited beamforms

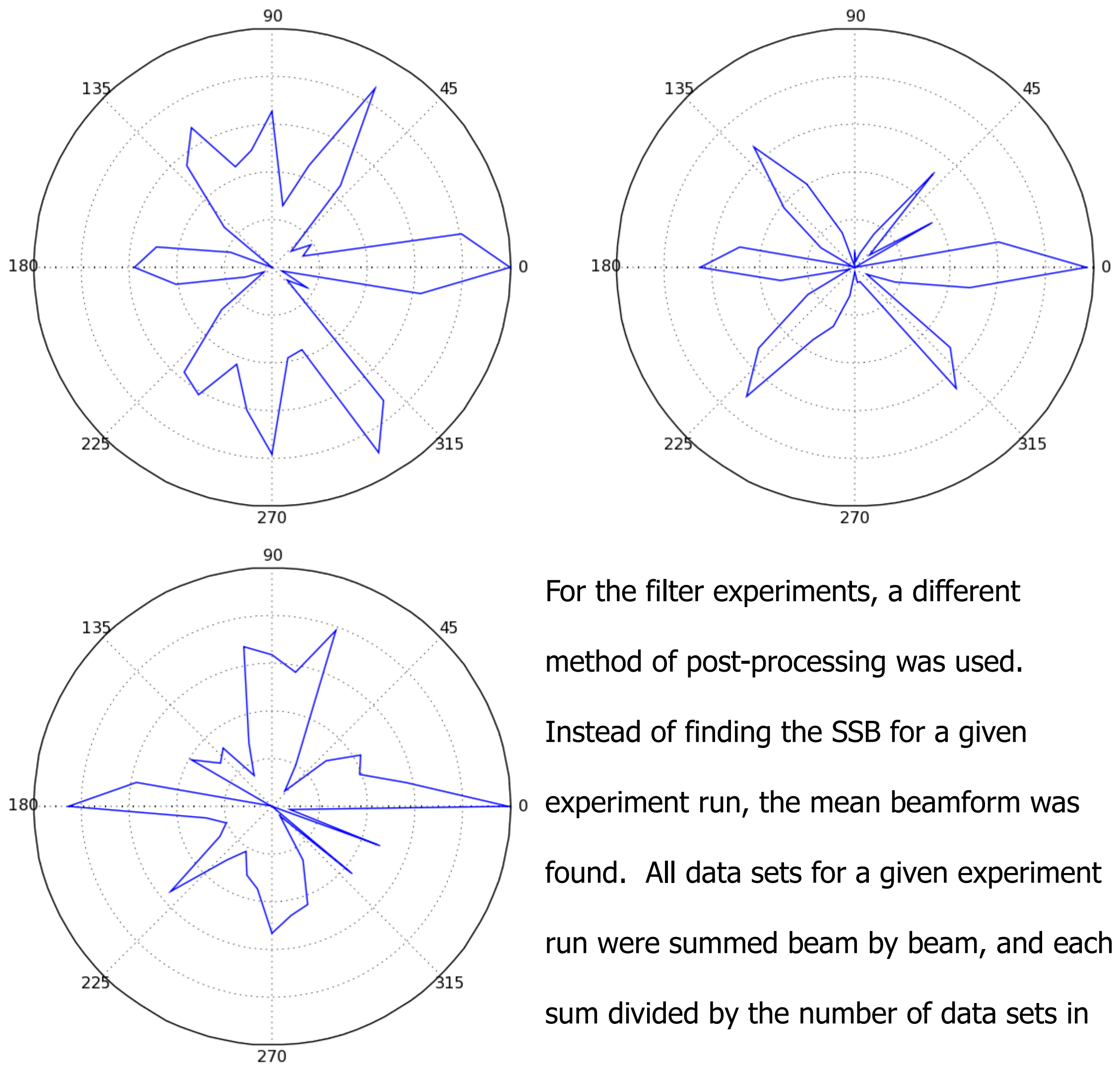


Figure 20: Clockwise from upper left: 6 kHz, 8 kHz, and 10 kHz band-limited beamforms

For the filter experiments, a different method of post-processing was used. Instead of finding the SSB for a given experiment run, the mean beamform was found. All data sets for a given experiment run were summed beam by beam, and each sum divided by the number of data sets in the run, giving the mean beamform for the run. The mean beamform allows us to directly compare the average energies of each beam in different experiment runs, which is necessary when quantifying the effects of the filter.

For each stimulus set, the mean beamform was first calculated with no filter in place. The mean beamform was then calculated for each experiment run in which the filter was active. The difference between the filtered and unfiltered mean beamforms gives us the profile of the filter, which, when scaled, is greatest in the

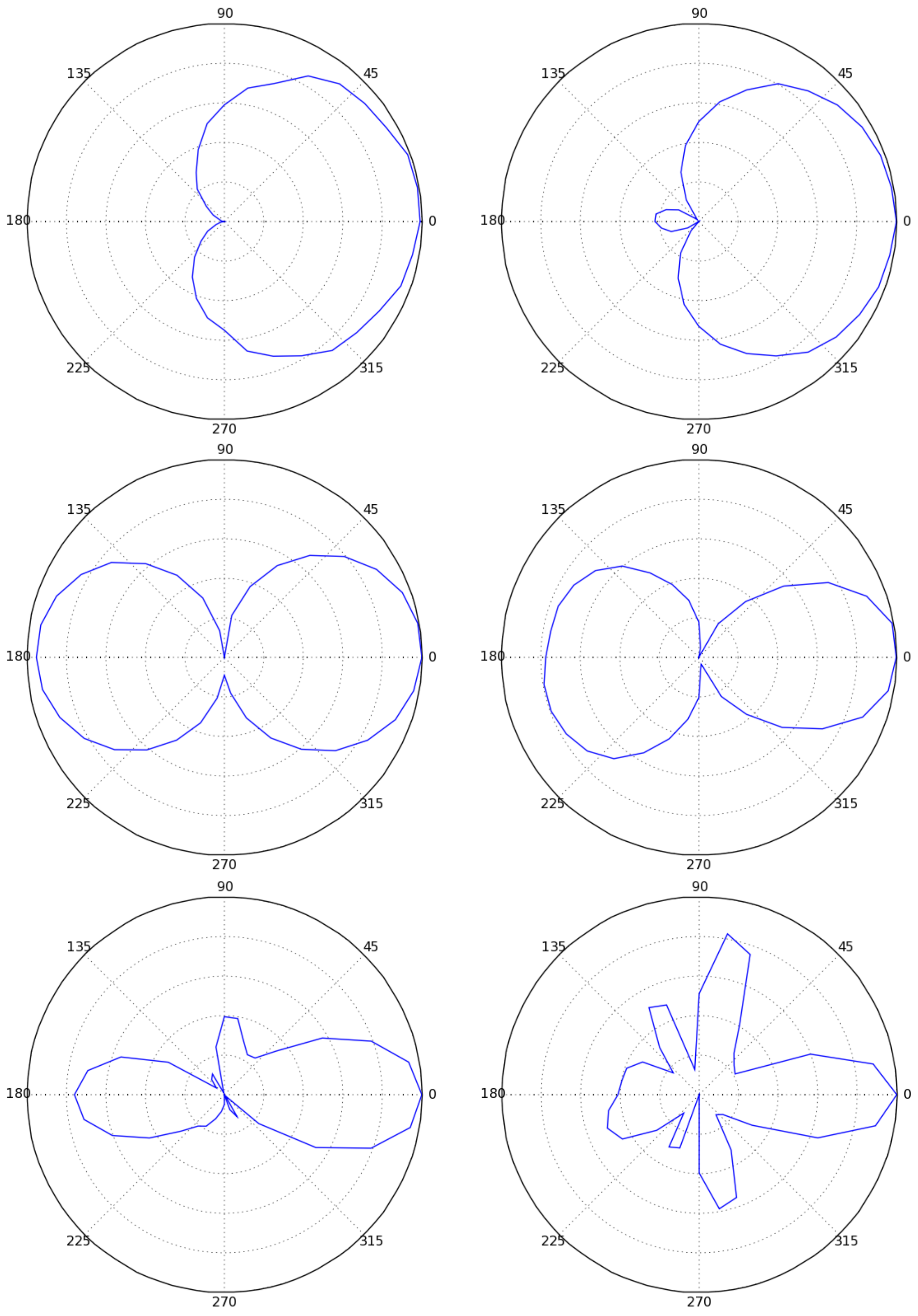
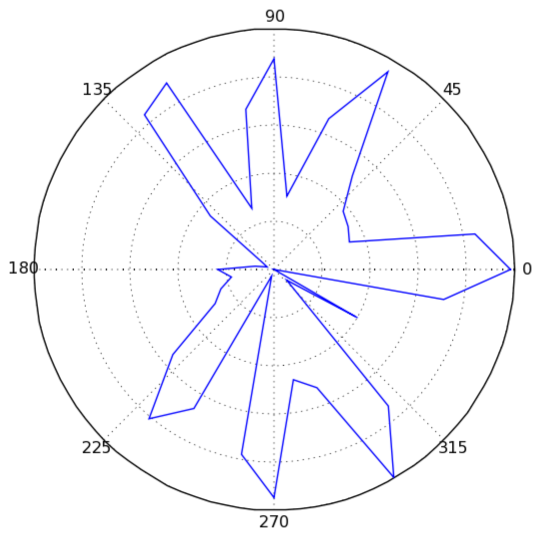
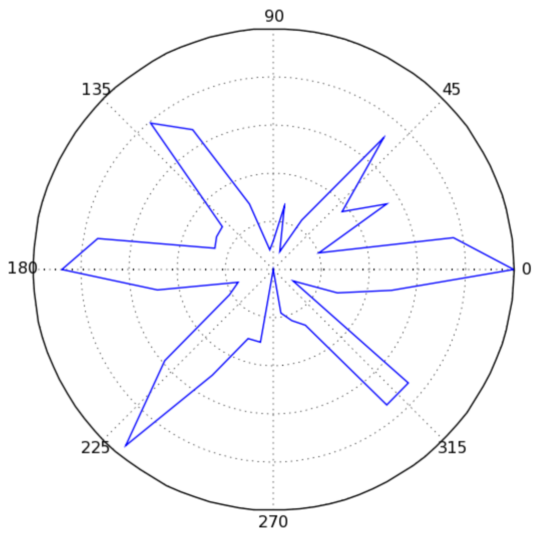


Figure 21: Clockwise from upper left: .5 kHz, 1 kHz, 2 kHz, 4 kHz, 3 kHz, and 1.5 kHz pure tone beamforms





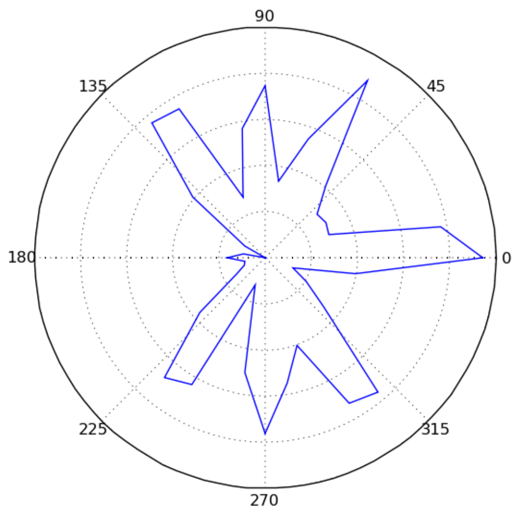


Figure 22: Clockwise from upper left: 6 kHz, 8 kHz, and 10 kHz pure tone beamforms

direction which is least effected by the filter. For a lowpass filter in a two source experiment, this scaled differential mean beamform (SDMB) should point in the direction of the higher frequency sound.

Figure 23-30 show the results of four sets of filter experiments. In each experiment, a 1 kHz stimulus was positioned at 0 degrees, and a 2 kHz stimulus was positioned at a different angle. When a 40-bit highpass filter (equivalent to a cutoff frequency of 1953 Hz) is applied, we would expect to see the SDMB point clearly in the direction of the higher frequency 2 kHz stimulus. In each case, this is clearly the result. Based on these results, we can safely conclude that the FFT filter block functions as desired, and is capable of emphasizing one distinct stimulus over another. As can be seen in Figure 31 and 32, however, the visual differences between the unfiltered and filtered beamforms

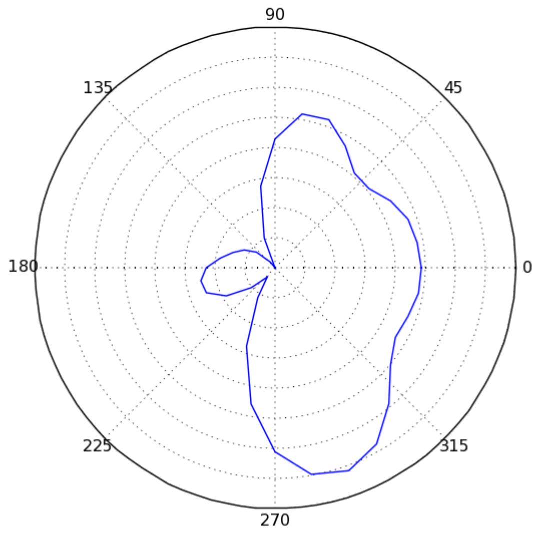


Figure 23: Unfiltered SSB of two band limited sources at 0 and 270 degrees

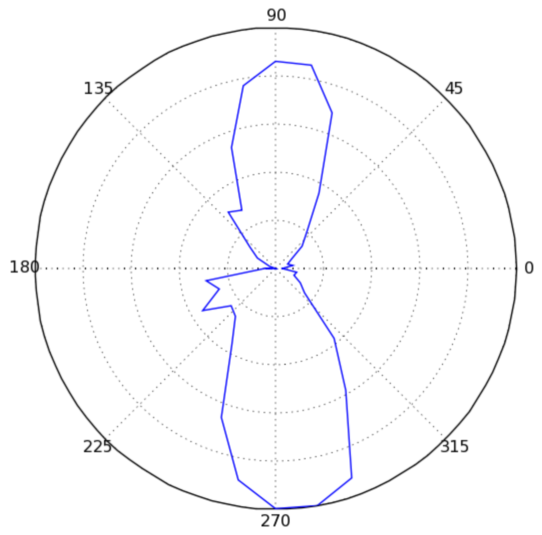


Figure 24: 40-bit highpass filtered SDMB of two band limited sources at 0 and 270 degrees

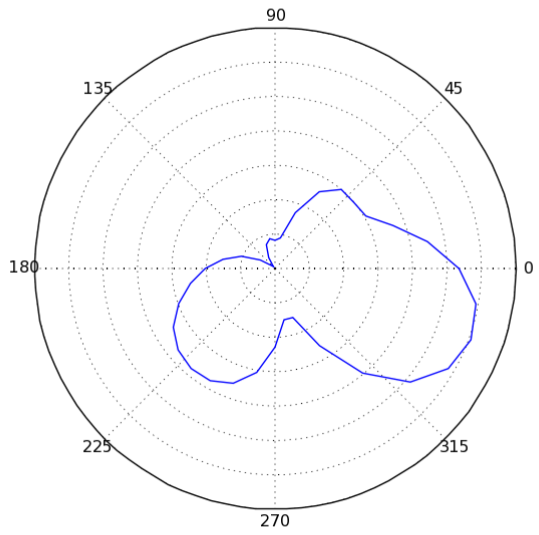


Figure 25: Unfiltered SSB of two band limited sources at 0 and 225 degrees

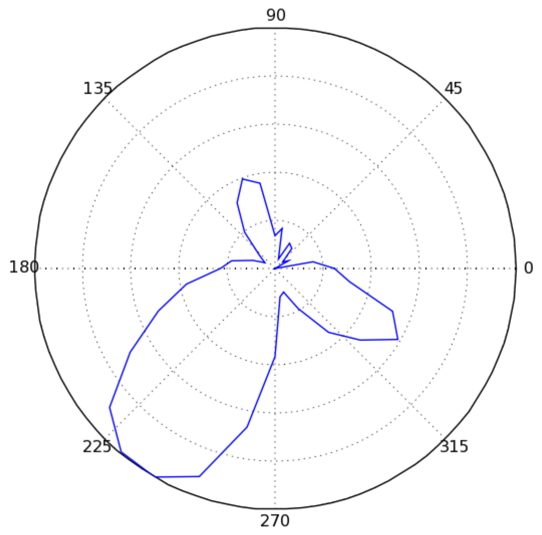


Figure 26: 40-bit highpass filtered SDMB of two band limited sources at 0 and 225 degrees

are minimal. Post-processing is necessary in this case for demonstration of the filter's capabilities.

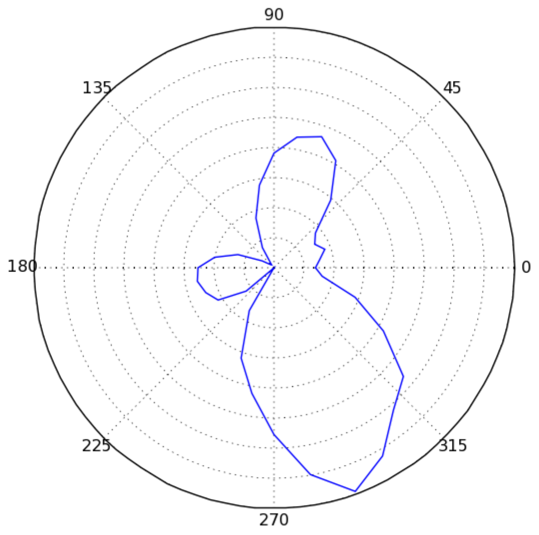


Figure 27: Unfiltered SSB of two pure tone sources at 0 and 270 degrees

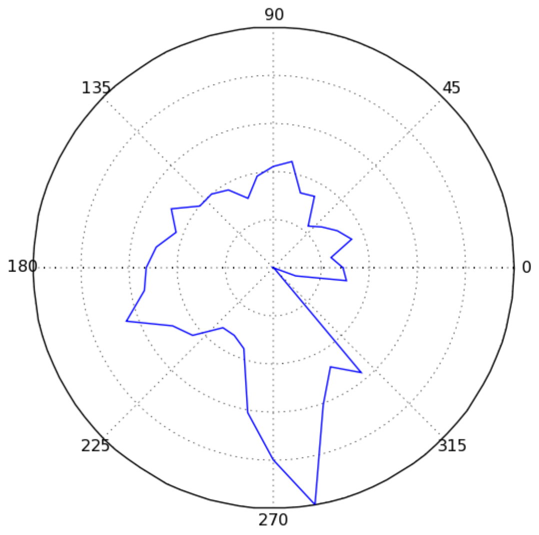


Figure 28: 40-bit highpass filtered SDMB of two pure tone sources at 0 and 270 degrees

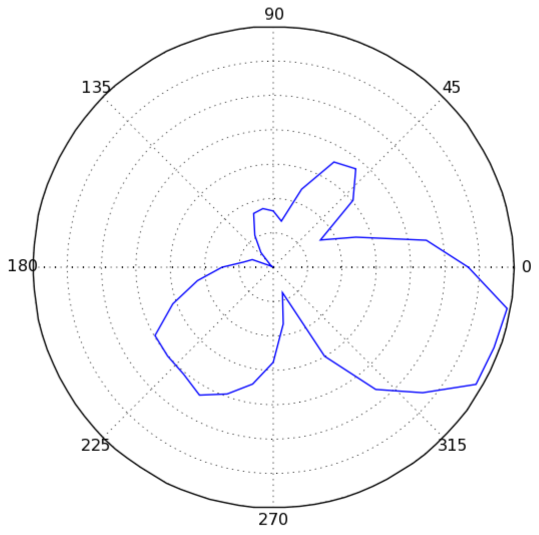


Figure 29: Unfiltered SSB of two pure tone sources at 0 and 225 degrees

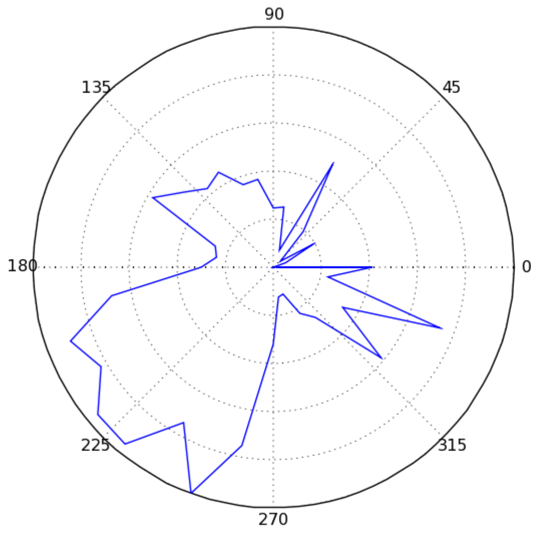


Figure 30: 40-bit highpass filtered SDMB of two pure tone sources at 0 and 225 degrees

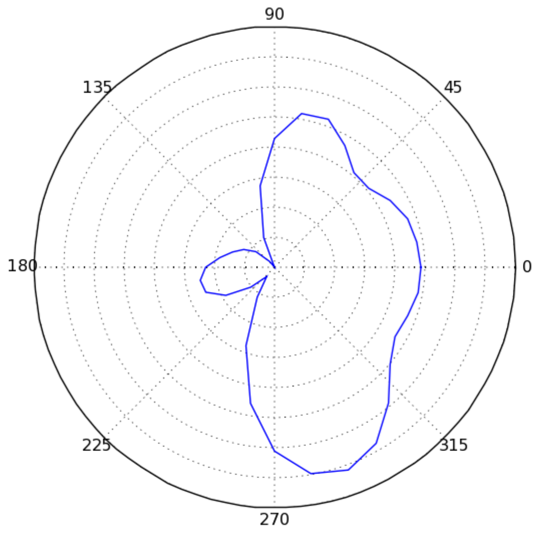


Figure 31: Unfiltered SSB, two sources, 90 degree separation

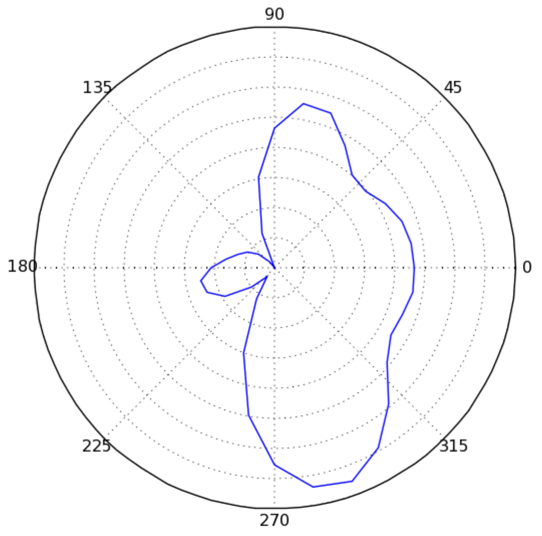


Figure 32: 40-bit highpass filtered SSB, two sources, 90 degree separation

CHAPTER VII

CONCLUSION

We have successfully implemented an accurate, low-power acoustic beamformer suitable for use as a node in a wireless sensor network. Maximum observed directional error is 18.4°, and maximum RMS directional error is 7.3°. We have also integrated this beamformer with an FFT-based runtime-configurable frequency filter, allowing frequency selection. This system is implemented in VHDL on a commercial FPGA. The node's power consumption is very low when compared with other power-efficient beamforming implementations, and the unit is capable of battery-powered operation for a useful length of time as part of a wireless sensor network.

When tested at frequencies of .5 and 1 kHz, the beamformer consistently delivers results of reasonable accuracy without significant bias. As frequency increases, the results become steadily more ambiguous, with multiple high-intensity side lobes arising as the stimulus frequency approaches 12.5 kHz, the Nyquist maximum frequency the beamformer can process. However, the direction of highest energy reported still remains largely accurate. The frequency filter integrated with the beamformer operates correctly, attenuating frequency spectrum of the reported beamforms in the desired manner and improving the beamformer's ability to identify frequencies of interest. However, the results reported by the beamformer may be of little use if only frequency-filtered beamforms are taken into account. The difference between filtered and unfiltered results is necessary to obtain useful information.

CHAPTER VIII

FUTURE WORK

The battery life of the node could be extended significantly by adding a power-saving sleep mode. In sleep mode, the beamformer would greatly reduce its calculation and transmission activity in the absence of any interesting stimuli. With such a mode, the node could potentially last for weeks without intervention.

The block RAM used to store the beamformer's offset lookup table is largely empty. This space could be put to effective use in a number of ways. If one was willing to forgo the frequency filter it would be possible for the beamformer to test up to 256 angles. This would be useful in situations requiring greater precision of the reported angle where no frequency filtering is needed. Similarly, reducing the FFT from 256 to 128 points would allow it to process 72 angles instead of 36, at the cost of greater frequency granularity.

As the output beamforms are likely to remain relatively consistent between transform runs, double-buffering the filter block is not strictly necessary. The delay and sum operations could be paused while the transforms are performed, and resumed once the filter buffers are cleared. Samples would be lost, but the change would likely be of minimal impact. At most, 58,788 cycles (2.9 mS) would pass before beamforming could resume, resulting in 74 lost beamform samples. A sound source is unlikely to change significantly in this time frame. Similarly, it would be possible to alternate between banks of angles, beamforming and filtering different sets of directions in turn, though more data would be lost using this method.

Perhaps most obviously, it would be possible to reimplement this system on a more powerful FPGA. As this implementation is pushing the limits of the XC3S1000, especially in memory usage, future expansion without sacrificing existing capabilities would likely require hardware with more resources.

As seen in Figure 31 and 32, the immediately visible difference between filtered and unfiltered beamforms is typically minimal. The high frequency source can still be drowned out by the low frequency source even with the filter active. However, the direction of the desirable stimulus is very easily picked out by performing a comparison between the filtered and unfiltered results. For useful real-world frequency selection it might be of value to alter the code such that the unit continuously alternates between filtered and unfiltered beamforming and performs the post-processing itself, giving a much more useful frequency-selective direction. This could be used in conjunction with the previous suggestions of increasing the resolution of the beamformer by alternating the filter block between banks of angles.

BIBLIOGRAPHY

[1] V. D. Van Veen and K. M. Buckley, "Beamforming: A Versatile Approach to Spatial Filtering," *ASSP Magazine, IEEE*, vol. 5, no. 2, pp. 4-24, 1988.

[2] H. E. Karrer, J. F. Larson, R. D. Pering, S. H. Maslak, and D. A. Wilson, "A Phased Array Acoustic Imaging System for Medical Use," in *IEEE Ultrasonics Symposium*, 1980, pp. 957-962

[3] R. Suoranta, "Novel Ultrasonic Beamforming Method Based On Nonlinear Filtering," presented at IEEE Ultrasonics Symposium, Tuscon, Arizona, 1992

[4] J. C. Chen, K. Yao, and R. E. Hudson, "Acoustic Source Localization and Beamforming: Theory and Practice," *EURASIP Journal on Applied Signal Processing*, vol. 2003, no. 4, pp. 359-370, 2003.

[5] J. Cooley and J. Tukey, "An Algorithm for the Machine Calculation of Complex Fourier Series," *Mathematics of Computation*, vol. 19, no. 90, pp. 297-301, 1965.

[6] R. Riley, B. Schott, J. Czarnaski, and S. Thakkar, "Power-Aware Acoustic Processing," in *proceedings of The 2nd International Workshop on Information Processing in Sensor Networks*, 2003, pp. 566-581

[7] S. Brown and J. Rose, "FPGA and CPLD Architectures: A Tutorial," *Design & Test of Computers*, vol. 13, no. 2, pp. 42-57, 1996.

[8] J. Liu, B. Weaver, Y. Zakhariv, and G. White, "An FPGA-based MVDR Beamformer Using Dichotomus Coordinate Descent Iterations," in *proceedings of the IEEE International Conference on Communications*, 2007, pp. 2551-2556

[9] R. Stapleton, K. Merranko, C. Parris, and J. Alter, "The Use of Field Programmable Gate Arrays in High Performance Radar Signal Processing Applications ," in *The Record of the IEEE 2000 International Radar Conference*, 2000, pp. 850-855

[10] P. Graham and B. Nelson, "FPGA-Based Sonar Processing," in *proceedings of the 1998 ACM/SIGDA Sixth International Symposium on Field Programmable Gate Arrays*, 1998, pp. 201-208

[11] E. Weinstein, K. Steele, A. Agarwal, and J. Glass, "LOUD: A 1020-Node Microphone Array and Acoustic Beamformer," presented at The 14th International Conference on Sound and Vibration, Cairns, Australia, 2007

[12] Xilinx technical staff, *Spartan-3 1.2V FPGA Family: Complete Data Sheet*, Xilinx, 2003.

[13] P. Volgyesi, G. Balogh, A. Nadas, C. Nash, and A. Ledeczi, "Shooter Localization and Weapon Classification Wtih Soldier-Wearable Networked Sensors," in *proceedings of the 5th International Conference on Mobile Systems, Applications and Services*, 2007, pp. 113-126

[14] Crossbow technical staff, *MicaZ Wireless Measurement System*, Crossbow, 2006.

[15] Panasonic technical staff, *Series WM-64PN Omnidirectional Back Electret Condenser Microphone Cartridge*, Panasonic, 2004.

[16] Xilinx technical staff, *Fast Fourier Transform 5.0*, Xilinx, 2007.