REAL-TIME IMPLEMENTATION OF AN ADAPTIVE BAYESIAN BEAMFORMER

Scott D. Briles*, Joseph Arrowood Jr.*, Thierry Cases°, Dakx Turcotte°, & Etienne Fiset°

*Los Alamos National Laboratory, Los Alamos, New Mexico, USA
°LYRTech Signal Processing, Quebec (Qc), Canada

ABSTRACT

An implementation of the adaptive Bayesian beamformer is examined for execution on field-programmable-gate-array (FPGA) devices. Using multiple sensor inputs, the Bayesian beamformer can estimate the direction-of-arrival (DOA) of a low-power signal in an environment that is simultaneously populated by high-power interference of limited DOA knowledge. A weighted sum of a discrete set of beamformers with known associated DOAs forms the Bayesian beamformer. Previously observed data provides the basis for the calculation of the *a posteriori* probability distribution function that renders the sum weights. This paper incorporates further approximations to the derivations to allow for its implementation on FPGA devices, in particular those with lesser gate counts. The feasibility of an all-FGPA implementation versus a heterogeneous implementation is explored.

1. INTRODUCTION

The real-time execution of an adaptive Bayesian beamformer is explored for implementation on field programmable gate array (FPGA) hardware. Previous work has demonstrated that a robust adaptive beamformer can be developed using a Bayesian approach[3]. Heuristic modification to the analytical algorithm [1] allows for greater efficiency in the execution of the algorithm. We further modify the algorithm so that it can be exercised on hardware intended for deployment in real-world conditions and requiring real-time execution. The use of a programming tool for the FPGA hardware was governed by an algorithmic-centric approach where all RTL (register transfer level) code was generated automatically from a graphical programming environment[1].

The adaptive Bayesian beamformer presented by K. L. Bell [1] has been demonstrated for applications where the DOA (direction of arrival) of a sought after source is uncertain and the signal environment varies greatly. The algorithm performs robustly in the presence of interference signals and over a wide range of signal-to-noise ratios. The beamformer is a weighted sum of minimum variance distortionless response (MVDR) beamformers where each is associated with a particular DOA. An *a posteriori* probability distribution function, calculated from previously observed data, is used to determine the contribution of each of the MVDR beamformers. Previous work has examined how much previous data needs to be examined [5] and modifications to the algorithm that make use of the Fast Fourier Transform (FFT) [2]. The determination of this *a posteriori* probability distribution function, while computationally intensive, is amenable to implementation in FPGA technologies, which is the focus of this paper.

We present several modifications to the algorithm for hardware implementation and evaluate them with a test case used previously [1]. The test case involves a desired signal with a signal-to-noise ratio (SNR) of 0 dB, two interfering signals with interference-to-noise ratios of 20 dB and an array of 10 sensors. Both interferers have DOAs outside of the six MVDR beamformers that compose the Bayesian beamformer.

2. BACKGROUND

The system model used is a set of M narrowband plane-wave signals with known center frequency ω_o, impinging on a uniform linear array of N sensors where $M < N$. Complex envelope representations of the narrowband, bandpass signals and narrowband, bandpass noise are produce from quadrature frequency down conversion. For additive noise this yields a received vector of N x 1 lowpass complex-amplitude signals, $\mathbf{x}(t)$, given by

$$\mathbf{x}(t_k) = \sum_{m=0}^{M-1} \mathbf{a}(\theta_m) s_m(t_k) + \mathbf{n}(t_k), \quad k = 1,2,... \qquad (1)$$

where $s_m(t_k)$, $m = 0,\ldots M\text{-}1;\ k = 1,2,\ldots$ are the source signals at time instance, $\mathbf{a}(\theta_m)$ is the N x 1 array response vector in direction θ_m, and $\mathbf{n}(t_k)$, k = 1,2,…are the N x 1 vectors of additive white noise. The assumption is made that source and noise signals are sample functions of zero-mean random processes. For successive time instances of

[1] Programming software tools used included, Simulink™ v6.1 by the MathWorks® Inc., SYSTEM GENERATOR™ 6.3 and ISE™ 6.3i by Xlinx® and FPGALink™ by LYRTECH Signal Processing.

both the source signals and noise signal, statistical independence is assumed.

One of the source signals is designated as the desired signal and the remaining source signals as interference. When the desired signal, $s_0(t)$, is uncorrelated with the noise and interference, the intra-array covariance matrix can be expressed as

$$\mathbf{R}_x(\theta_0) = \sigma_s^2 \mathbf{a}(\theta_0)\mathbf{a}(\theta_0)^H + \mathbf{R}_n \quad (2)$$

where σ^2_s is the power of the desired signal, $\mathbf{R}_n$ is the interference-plus-noise covariance matrix, and $\mathbf{a}(\theta_0)$ is the complex-valued response vector in the direction of the desired signal.

The beamformer operation uses a vector of N complex weights multiplied with the output of the sensors in the array to yield an estimate of the desired signal given by

$$y(t_k) = \hat{s}_0(t_k) = \mathbf{w}^H \mathbf{x}(t_k). \quad (3)$$

If the weights are chosen via the minimum power distortionless response criterion, the beamformer weights can be determined by

$$\mathbf{w}_{MV} = \frac{\mathbf{R}_\mathbf{x}^{-1}\mathbf{a}(\theta_0)}{\mathbf{a}(\theta_0)^H \mathbf{R}_\mathbf{x}^{-1}\mathbf{a}(\theta_0)}. \quad (4)$$

The weights are usually referred to as minimum variance distortionless response (MVDR) weights. An estimate of the intra-array covariance matrix can be obtained via time averaging of matrices that under stationary and ergodic assumptions converges to the true covariance matrix. Thus, an estimate of $\mathbf{R}_\mathbf{x}$ is

$$\hat{\mathbf{R}}_K = \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}(t_k)\mathbf{x}(t_k)^H. \quad (5)$$

With the estimate of (5), the equation for an estimate of the weights of the MVDR beamformer can be given as

$$\hat{\mathbf{w}}_{MV} = \frac{\hat{\mathbf{R}}_K^{-1}\mathbf{a}(\theta_0)}{\mathbf{a}(\theta_0)^H \hat{\mathbf{R}}_K^{-1}\mathbf{a}(\theta_0)}. \quad (6)$$

Adaptive beamforming can be achieved through time-progressive estimates of the covariance matrix given by (6) of the MVDR beamformer weights. After every K samples for the array outputs, a new covariance matrix estimate can be determined. This new estimate contains the most recent information regarding unknown interferers and fluctuations in noise and thus is adaptive to a changing environment.

3. BAYSESIAN BEAMFORMER

The Bayesian beamformer starts with the condition that the direction of arrival (DOA) of the desired signal is unknown, but assumed to exist in a range of interest [θ_a , θ_b]. A discrete set of MVDR beamformers, each with a unique direction component inside the range of interest, is combined in a weighed-sum fashion to yield the Bayesian beamformer. In this case each element of the Bayesian *a priori* probability distribution function is associated with each MVDR beamformer. Thus the DOA is a discrete random variable defined on a set of L points, $\Theta = \{\theta_1 \ldots \theta_L\}$ with associated probability distribution function $p(\theta)$. The weights are determined via the *a posteriori* probability for each pointing direction. Assumptions that the source and noise signals are sampled functions of stationary, zero-mean Gaussian random processes, allows for an expression to be determined for the *a posteriori* probability density function and for the MVDR beamformers to be considered spatial Wiener filters. The Bayesian beamformer can be expressed as

$$\mathbf{w}_B = \sum_{i=1}^{L} p(\theta_i \mid \mathbf{X})\mathbf{w}_{MV}(\theta_i), \quad (7)$$

where $\mathbf{X}$ is the collection of K time instances of acquired data vectors, $\mathbf{x}(t)$.

The derivation and expression for the *a posteriori* probability distribution function is not presented in this paper; instead conceptual insights are discussed. First, the *a priori* $p(\theta)$ probability distribution function is assumed known. In an adaptive situation, this knowledge could be derived from previous *a posteriori* probability distribution functions. With no prior knowledge, a maximum entropy approach of a uniform probability distribution function provides the prudent starting point.

Two extreme-result beamformers can be produced by the Bayesian beamformer. First, with high SNR, highly-accurate estimation of the covariance matrix and an exact match between the bearing of the desired signal and a bearing present in Θ, the *a posteriori* probability of the true DOA will approach one and other DOAs will approach zero. Under these conditions the Bayesian beamformer will reduce to a single MVDR beamformer. The other extreme is when the covariance matrix is weakly estimated and/or the SNR is low, then the Bayesian beamformer will average the individual MVDR beamformers according to the *a priori* probability distribution function.

Apart from the two extreme cases, the Bayesian beamformer combines the individual MVDR beamformers in a manner that balances the knowledge acquired via observations of the data with the *a priori* probability distribution function. Even in the case where the desired signal has a high SNR and one is able to perform accurate covariance matrix estimation, if the true DOA of the desired signal is not exactly present in the finite set Θ, then a combination of MVDR beamformers form the Bayesian beamformer. It should be apparent that there is a tradeoff that can be made between

computational effort and output precision with the number of discrete DOAs, *L*, in Θ.

3.1. Approximations

Approximations are made to calculate the *a posteriori* probability distribution function in a manner that is suitable for implementation. First the constraint is made that no interference signal has a bearing that lies in the range of interest [θ_a , θ_b]. This constraint allows for the approximation of a constant noise term, which thus allows for a constant, γ, that is a function of SNR, to be a design parameter. Without knowledge of the noise covariance matrix, the estimate of the data covariance matrix, which was also determined for the MVDR beamformers, can be used to determine the *a posteriori* probability distribution function. Thus the MV spatial power spectrum estimate, $\hat{P}_{MV}(\theta)$, can be used in the estimate of the *a posteriori* probability distribution function, $\hat{p}(\theta_i \mid \mathbf{X})$, which was shown by the authors in [1]. This is given as

$$\hat{p}(\theta_i \mid \mathbf{X}) = c\,p(\theta_i)\exp\left(K\gamma \hat{P}_{MV}(\theta_i)\right) \tag{8}$$

where

$$\hat{P}_{MV}(\theta_i) = \left(\mathbf{a}(\theta_i)^H \mathbf{R}_K^{-1}\mathbf{a}(\theta_i)\right)^{-1} \tag{9}$$

is a non-negative scalar for a given bearing.

The estimate of the data covariance matrix is revised with the use of diagonal loading. It has been shown that diagonal loading can be of great benefit to system performance for beamformers [6] when the covariance matrix is inadequately estimated. For the adaptive Bayesian beamformer, diagonal loading is also applied to the data covariance estimate of the spatial power spectrum estimate. It was noted in [1] that diagonal loading in the *a posteriori* probability distribution function allowed for fewer MVDR beamformers to be needed in the adaptive beamformer. Thus the estimate of data covariance matrix of (5) becomes

$$\hat{\mathbf{R}}_{K,DL} = \hat{\mathbf{R}}_K + \sigma^2\mathbf{I}\,. \tag{10}$$

3.2. Algorithm & Design Parameters

The differences of spatial power spectrum estimates of Eq. (9) are nonlinearly amplified by Eq. (8). The constants, *K* and γ, produce the slope of the amplification in dB. With greater averaging and larger K, differences in the spatial power spectrum estimate are more enhanced along with the quality of the data covariance estimate. The parameter γ is a design parameter that is related to SNR. It determines the level of gain in the exponential function. Dominance among the MVDR beamformers is governed by this value. This value can also be used to compensate for overall signal levels.

Examining Eqs (6) and (8), both the calculation of the MVDR beamformer and the *a posteriori* probability distribution function depend on the spatial power spectrum estimate. The algorithm presented by K. L. Bell[1] that exploits this fact is replicated below:

1) $\hat{\mathbf{R}}_{K,DL}^{-1} = \left(\left(1/K\right)\sum_{k=1}^{K}\mathbf{x}(t_k)\mathbf{x}(t_k)^H + \sigma^2\mathbf{I}\right)^{-1}$

2) $i = 1,\ldots L$

$$\mathbf{v}_i = \hat{\mathbf{R}}_{K,DL}^{-1}\mathbf{a}(\theta_i)$$

$$\hat{\mathbf{P}}_{MV,DL}(\theta_i) = \left(\mathbf{a}(\theta_i)^H\,\mathbf{v}_i\right)^{-1}$$

$$\hat{\mathbf{w}}_{MV,DL}(\theta_i) = \hat{P}_{MV,DL}(\theta_i)\mathbf{v}_i$$

$$\hat{p}(\theta_i \mid X) = p(\theta_i)\exp\left\{K\gamma\hat{P}_{MV,DL}(\theta_i)\right\}$$

3) $c = \left\{\sum_{i=1}^{L}\hat{p}(\theta_i \mid \mathbf{X})\right\}^{-1}$

4) $\mathbf{w}_B = c\sum_{i=1}^{L}\hat{p}(\theta_i \mid \mathbf{X})\hat{w}_{MV,DL}(\theta_i)\,.$

5. FPGA IMPLEMENATION

Using a graphical user interface (GUI) based programming environment, the above algorithm [1], was modified for implementation on real-time FPGA hardware[2]. Current software development tools allow not only the simulation of the hardware but also hardware-in-loop testing of the algorithm. The performance of the adaptive Bayesian beamformer intended for fixed-precision, real-time execution can be compared to the analytical algorithm. The major focus for this work was to determine where modifications were needed for a deployed implementation and what effects these modifications have on overall algorithmic performance.

Embedding the analytical algorithm in hardware involved several algorithm modifications primarily relating to fixed-point arithmetic, diagonal loading, nonlinear mathematical operations, and matrix inversion. A governing principle for the current phase of this work was to make every attempt to limit any deviations from the original algorithm. In order to develop these modifications to the algorithm, performance was heuristically evaluated against the analytical solution of Example 1 formulated by K.L. Bell[1]. This test-case example involved two strong interferes (INR = 20dB) outside bearings of interest at -0.5 and 0.6 radians, a weaker signal of interest (SNR = 0dB) located at 0.14 radians, ten sensors (N=10) in a standard uniform linear array, and a diagonal loading of 10 dB. The design

[2] VHS-ADC board by LYRTECH Signal Processing using a Virtex-II™ XC2V6000 FPGA from Xilinx®.

parameters for the example were, equal probability within the set Θ = {-0.167, -0.1, -0.033, 0.033, 0.1 0.167} that consisted of 6 bearing angles, γ=0.3 and 30 data covariance matrix averages.

5.1 Amplitude Scaling

Amplitude scaling plays a significant roll in the performance of the algorithm. If the digitized input signal, i.e., signal post analog-to-digital conversion, does not fill a specific dynamic range, then the nonlinear exponential operation does not adequately splay the *a posteriori* probabilities. Compensation for amplitude variation can be implemented in two fashions. First the input signal itself can be evaluated and then have the appropriate scaling applied to it. Thus an automatic gain control in digital hardware is implemented. Alternatively, γ, which controls the amplification of $\hat{P}_{MV}(\theta)$ can be made dynamic with a simple linear scaling related to the dynamic range of the input signal.

Input signal scaling was chosen since it can be manipulated simply in binary format and it alters the original algorithm the least. The test-case example signal amplitudes were referenced to a noise power level of 0 dB. For a real-world instrument, the input single amplitude levels are usually constrained to fill a specific amplitude range. For this example a range of ±0.5 V was chosen. In relation to the test case example, the best compensation for this constraint would be to scale the input signal by a factor of 88. However, scaling by a factor of 64 relates to a movement of the binary point by 6 bits for the digitized samples. On its own this modification does cause less separation of the *a posteriori* probabilities, but only to a minor extent in the test-case example.

5.2. Diagonal Loading

A further modification was dealing with a lack of knowledge of noise and interference power levels needed for diagonal loading calculations in a real-time, real-world deployment. It is desirable to set the diagonal loading to a level that is between the noise and interference levels. In the analytical algorithm this knowledge is used in diagonal loading the covariance matrix to achieve greater efficiency in the algorithm [1], [3]. Without knowledge of the noise, interference or desired-signal power levels, the estimate of the covariance matrix can be used to determine the diagonal loading level. Averaging the diagonal elements of the covariance matrix estimate gives an estimate of total power of all sources plus noise. Setting the diagonal loading to some fraction of total power estimation provides a means for setting the diagonal loading. For this paper the diagonal loading was set at a level 10 dB below (0.1 scaling) the variance-diagonal average from the covariance matrix.

This modification, both on its own and combined with the amplitude-scaling modification resulted in very little degradation of performance. Examination of the *a posteriori* probabilities revealed greater contribution in the Bayesian beamformer of MVDR beamformers that were non-adjacent to the desired signal's bearing. However a strong center peak for the desired signal was produced along with deep nulls for the interfering signals.

5.2. Exponential Function

The exponential operation presented in the algorithm is critical for the proper splaying of the *a posteriori* probabilities and thus performance degradation does materialize in the approximation needed for FPGA implementation. Two straightforward approaches exist for the approximating the exponential in hardware. The first method is to make use of a lookup able and convert from the logarithm domain. A second approach is to use a Taylor series approximation for the exponential function. Both approaches can be implemented to yield similar precision. The lookup table approach has its precision defined by memory depth and the Taylor series approach is defined by largest power used.

This paper's implementation of the exponential function was a version that used both a Taylor series and a lookup table. The Taylor series converges most quickly when the exponent has an absolute value of less than 0.5. For the non-negative exponent in the *a posteriori* probability distribution function calculation, it can be expressed as a sum of the portion greater than 0.5, y_{LT}, and less than 0.5, y_{TS}. Combining with substitution with yields

$$\exp\left(K\gamma\hat{P}_{MV}(\theta_i)\right)=\left[\left(2^{y_{LT}}\right)^{\frac{1}{\ln 2}}\right]\exp(y_{TS}). \qquad (11)$$

Using the Taylor series the following substitution was used in the Bayesian beamformer algorithm

$$\exp(y_{TS})\cong 1+y_{TS}+0.5\times {y_{TS}}^2+0.1.667\times {y_{TS}}^3. \quad (12)$$

with y_{TS} at a precision of 17 bits, binary point 16 (17.16b) and the result at a precision of (18.16b). The y_{LT} term, intended for lookup table implementation, makes use of the approximation ln(2) ≅ 2/3. This approximation allows for the following exponential approximation

$$\exp(y_{LT})\cong 2^{\frac{3}{2}y_{LT}}. \qquad (13)$$

Eq. (13) can be further refined for FPGA implementation by scaling, $y'_{LT}=2y_{LT}$, an then forcing the result to be an integer, represented by the FIX function, as

$$\exp(y_{LT}) \cong 2^{FIX\left(\frac{3}{4}y'_{LT}\right)}. \quad (14)$$

For a constant input signal level, Eq. (14) can be offset by the minimum value encountered. This is equivalent to normalizing the smallest y_{LT} to be zero and has no affect on the splaying of the probabilities. Once the offset is discovered from simulations, the power of two associated with part of the exponential function result can be given as

$$B = FIX\left(\frac{3}{4}y'_{LT}\right) - Offset\,, \quad (15)$$

where B is a non-negative integer. Multiplying by a power of two is equivalent to a binary-shifting operation, and thus the approximation to the exponential function is obtaining by shifting the binary point of the result of Eq. (12) by B positions to the right.

Examination of the test-case example performance, when this substitution was combined with the amplitude and diagonal loading modifications showed somewhat less splaying of the *a posteriori* probability distribution function. However, the level of this degradation was not sufficient to diminish the overall results of the algorithm.

5.1. Inverse Covariance Matrix

The greatest challenge to the FPGA implementation of the adaptive Bayesian beamformer was the inversion of the covariance matrix. Other adaptive methods, e.g., least-mean-squares, could have been used to adjust the beamformer weights. However, in an effort to change the algorithm as little as possible the matrix inversion was attempted in FPGA hardware. Since the size of the matrix is determined by the number of sensors in the array, the matrix dimensions are not large, i.e., 10 by 10, and are fixed. With the covariance matrix being suitable for Cholesky factorization, a "hardwired" matrix inversion can be implemented in the FPGA hardware.

Although requiring a great deal of effort to program, there is little modification of the algorithm compared to the analytical solution except for fixed-precision arithmetic. The FPGA implementation used more or less a standard Cholesky matrix-inversion algorithm. Other approaches are being explored, including recursive matrix inversions [7].

6. EXAMPLE COMPARISON

Using Example 1 from previous work [1] of others, a comparison is made between the analytical floating-point solution and the FPGA implementation. Figure 1 shows a comparison between the two typical *a posteriori* probability distribution functions produced by the two implementations. It is clear from these two plots that for the case of the FGPA implementation the separation between the members of the candidate MVDR beamformers is less pronounced that than of the analytical solution. However, the FPGA implementation still maintains the characteristic of accentuating the two MVDR beamformers that surround the true DOA of the desired signal. Fig. 2 shows two typical beampatterns for the two implementations. The two are similar in profile, but the level of attenuation of the interfering signals is not as strong for the FPGA implementation.

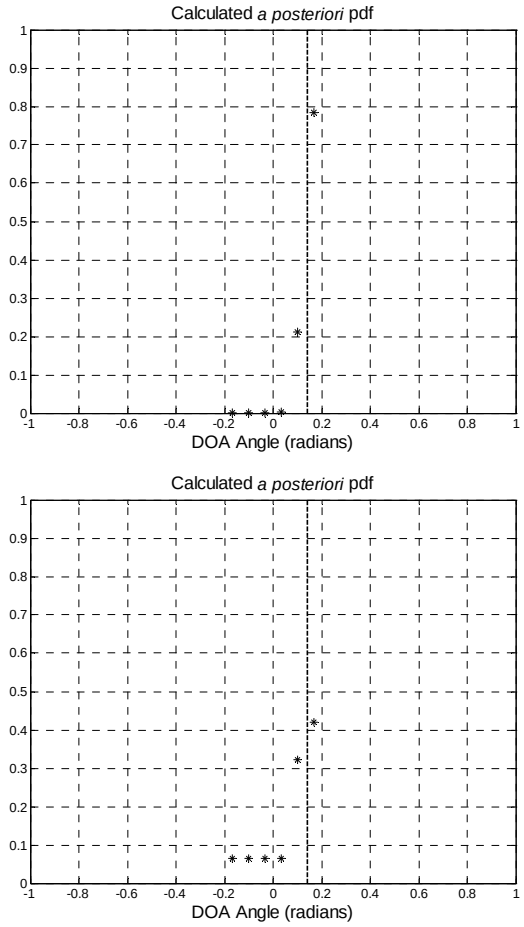

Fig. 1. Typical calculated *a posteriori* pdfs for the analytical implementation (top) and the FPGA implementation (bottom).

7. SUMMARY

The adaptive Bayesian beamformer algorithm can be implemented in FPGA hardware with the intent that it performs in a real-world environment and under real-time (high sampling rate) conditions. Though modifications were necessary for the fixed-precision implementation preferred on FPGA hardware, the analytical algorithm was approximated with enough accuracy that the results were showed only minor-to-moderate degradation. Interference signals were acceptably attenuated (though not completely nulled as in the floating-point

implementation) and the beam was properly steered for the desired signal.

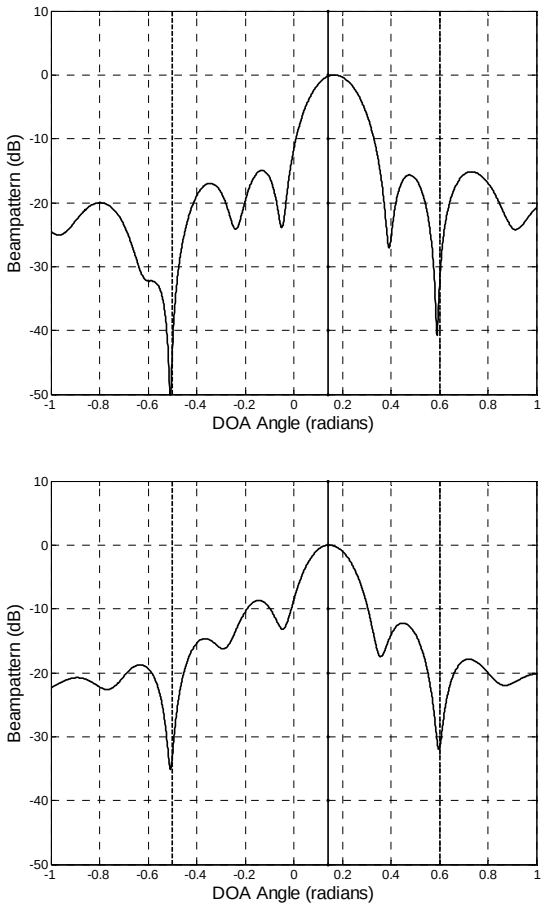

Fig. 2. Typical calculated beampatterns for the analytical implementation (top) and the FPGA implementation (bottom).

While the FPGA implementation performs adequately for most remote sensing applications, hardware implementation performance can be increased via the use of heterogeneous hardware architectures where DSP microprocessors could be used in tandem with the FPGA processors. This would allow the algorithm to perform low bandwidth calculations (such as the matrix inverse needed for weighting coefficient updates) in a floating-point environment on the DSPs. This would result in an increase in the accuracy of some of the critical intermediate results and hopefully allow the overall hardware implementation to more closely mimic the analytical performance.

9. RELEASE & ACKNOWLEDGEMENTS

This paper is released under Los Alamos National Laboratory certification LA-UR-05-4151.

The authors would like to acknowledge U.S. Department of Energy, NN-22 for sponsoring this work. Also we would like to acknowledge Kristine Bell for her input.

10. REFERENCES

[1] K. L. Bell, Y. Ephraim, and H. L. Van Trees, "A Bayesian approach to robust adaptive beamforming," *IEEE Trans. Signal Processing*, vol. 48, pp. 386-398, February 2000.

[2] Lam, C.J.; Singer, *A.C.;* **"**Fast adaptive Bayesian beamforming using the FFT Statistical," Signal Processing, 2003 IEEE Workshop on, pp. 413 – 416, 28 Sept.-1 Oct. 2003

[3] Bell, K.L.; Ephraim, Y.; Van Trees, H.L.*;* **"**Robust adaptive beamforming using data dependent constraints," Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on, Volume: 5, pp. 3513 - 3516 vol.5, 21-24 April 1997

[4] H. L. Van Trees, *Optimum Array Processing*. New York: Wiley Interscience, 2002.

[5] C. J. Lam and A. C. Singer, "Performance Analysis of the Bayesian Beamformer," Proceedings of ICASSP 2004, Montreal, Quebec, Canada, v2, pp. II197-II200, May 17-21, 2004.

[6] B.D Carlson; "Covariance matrix estimation errors and diagonal loading in adaptive arrays," IEEE Transactions on Aerospace and Electronic Systems, vol. 24, Issue 4, pp.397 – 401, July 1988.

[7] N. Gaarder, "A recursive relation for the inverse of the covariance matrix in optimal pattern classifiers (Corresp)," IEEE Transactions on Information Theory, Vol. 11, Issue 1, pp.151 – 152, Jan 1965.