GNU RADIO BASED DIGITAL BEAMFORMING SYSTEM: BER AND COMPUTATIONAL PERFORMANCE ANALYSIS

Sarankumar Balakrishnan, Lay Teen Ong

Temasek Laboratories, National University of Singapore, Singapore

ABSTRACT

The rapid growth in computational capacity of general purpose processors (GPPs) has allowed for an alternative to traditional implementation of digital signal processing systems. Signal processing algorithms that were once implemented in dedicated field programmable gate arrays (FPGAs) and embedded digital signal processors are now being increasingly implemented using softwares. This paper presents the development of a GPP based digital beamforming system using GNU Radio -an Open Source software development platform for signal processing applications to be used with software defined radio systems. The developed beamforming system is based on minimum variance distortionless response (MVDR) algorithm. We study the Bit Error Rate (BER) performance of the beamforming system. We provide the experimental BER results to highlight the signal recovery capabilities of the beamformer. This paper also addresses the challenges of real-time implementation and analyses the computational complexity of the GPP based digital beamforming system.

Index Terms— GNU Radio, software defined radio, digital beamforming

1. INTRODUCTION

Digital beamforming is a signal processing technique to control the reception pattern of the antenna array such that, nulls are placed in the direction of the interference signals while maintaining appropriate gain in the direction of the desired signal. Antenna array based digital beamforming systems exploit the spatial diversity to achieve interference mitigation.

Traditional beamforming systems are based on FPGA technology which is a cost effective solution but offers very little flexibility in terms of design and quick prototyping. They are associated with high development costs and long time-to-market and are customized for a specific application. The exponential increase in the computational capabilities of modern day General Purpose Processors (GPPs) offers alternative design to the traditional design using FPGAs. An alternative approach is to use GPP based Software Defined Radio (SDR) [1]. SDR based signal processing systems have made significant progress over the years. In SDR, the signal processing algorithm is typically implemented in a software framework rather than being embedded in a chip. This offers flexibility in quick prototyping of signal processing applications. Some of the widely used SDR systems that uses GPPs are GNU Radio [2], OSSIE [3], and Microsoft's Sora [4]. Some of the practical examples of a GPP based SDR systems are the works in [5] and [6] that uses SDR concept to demonstrate IEEE 802.11a/g/p OFDM receiver system and an interference canceller respectively.

Despite the flexibility offered by the GPP based SDR systems, several limitations remains to be seen. Notably, the resource allocation and real-time capabilities of general purpose processors. Unlike FPGAs in which the available resources are optimized for the application, GPPs are designed for running several applications simultaneously thus competing for available resources. Moreover, real world signal processing systems are fundamentally real-time. The system must complete processing the incoming data segment before the next one arrives. These limitations places stringent requirements on the resource utilization of the signal processing systems.

In this paper we discuss the performance and computational complexity of software based digital beamforming system. The contribution of this work is the implementation and performance analysis of GNU Radio based digital beamforming system for real-time operation. This paper is organized as follows. Section II discusses the architecture of the software based digital beamforming system, the signal model and presents the MVDR beamforming algorithm. Section III provides performance analysis in terms of BER. The real-time capabilities and computational resources of the GNU Radio software based digital beamforming system are also discussed. Section IV offers conclusion.

2. DIGITAL BEAMFORMING ARCHITECTURE

Figure 1 gives an overview of the software based digital beamforming architecture together with the BPSK receiver. The architecture has a N-channel RF front end to downconvert and digitize the signal received by the N element antenna array and a software part where the signal processing algorithms are implemented. A host driver at the GPP known as UHD (Universal Hardware Driver) enables communication between the RF front end and the software part in the GPP using Gigabit Ethernet (GigE). On the hardware

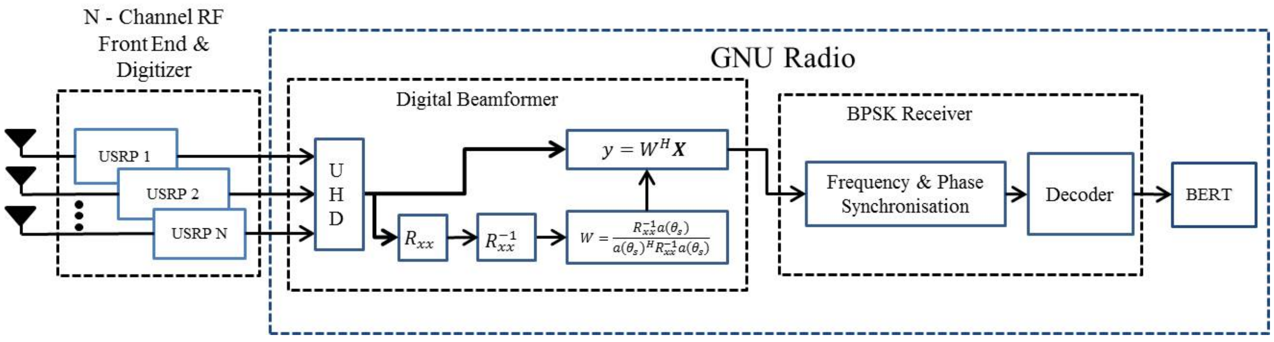
N - Channel RF Front End & Digitizer
USRP 1
USRP 2
USRP N
GNU Radio
Digital Beamformer
U H D
$y = W^H X$
R_{xx}
R_{xx}^{-1}
$W = \frac{R_{xx}^{-1}a(\theta_s)}{a(\theta_s)^H R_{xx}^{-1} a(\theta_s)}$
BPSK Receiver
Frequency & Phase Synchronisation
Decoder
BERT

Fig. 1. Overview of the digital beamforming system and the BPSK receiver.

side, there are several commercial RF front ends that are available to interface with the software architecture. For instance, the Universal Software Radio Peripheral (USRP) [7] is a software defined radio equipped with RF daughterboards that can tune to different frequencies, downconvert and send digitized samples to the host processing system. With the USRP supporting wide range of RF front ends targeted for various applications, the development work is entirely offloaded to the software domain. The software part of the digital beamforming system is based on GNU Radio, an open source software framework for implementing software radio applications. GNU Radio provides libraries of primitive C++ signal processing blocks for implementing various signal processing and communication applications. In GNU Radio, signal processing applications are written as one-directional graph of DSP blocks known as flowgraphs. The output of one DSP block serves as input to other DSP blocks. GNU Radio uses thread-per-block scheduler. Each DSP block in a GNU Radio flowgraph is executed by its own thread. The OS scheduler automatically distributes these threads in a multicore processor. Flow of data between DSP blocks is through shared memory. The first DSP block in the flowgraph writes data to the shared memory and the second DSP block reads the data from the shared memory. The performance of the software part is crucial since it has to process the incoming digitized samples from the RF front end in real-time. To achieve the required computational speed imposed by the sample rate of the RF front end, the signal processing algorithms are implemented using C++ and Armadillo [8], an open source and optimized C++ linear algebra library. The software architecture is also modular. The modular structure makes it easy to add new beamforming algorithms and study the performances. For the study presented in this paper, we have implemented an 8-channel Minimum Variance Distortionless Response (MVDR) beamforming algorithm [9, 10] which will be discussed in the subsequent sections. Also, to study the BER performance of the beamforming system, a BPSK receiver with phase and frequency synchronization is designed and implemented using GNU Radio.

2.1. Antenna Geometry

A uniform circular array (UCA) with N elements is considered. Let (θ, ϕ) denote the elevation angle and azimuth angle of the signal impinging on the antenna array. The array factor $AF(\theta, \phi)$ for this antenna array is given by [11]

$$AF_{UCA}(\theta, \phi) = \sum_{n=1}^{N} \alpha_n e^{j\beta a \sin(\theta)\cos(\phi - \phi_n)} \qquad (1)$$

where α_n is the complex antenna excitation of the n^{th} element, a is the radius of the UCA, θ is the elevation angle and ϕ is the azimuth angle. $\phi_n = \frac{2\pi n}{(N-1)}$ is the angular position of the n^{th} element. $\beta = \frac{2\pi}{\lambda}$ defines the wave number and λ is the wavelength.

Assuming *M* signals are impinging at the *N* element uniform circular array, the received signal $\mathbf{x}(k)$ at sample instance *k* can be expressed as

$$\mathbf{x}(k) = \mathbf{A}(\theta, \phi)\mathbf{s}(k) + \mathbf{n}(k) \qquad (2)$$

where $\mathbf{x}(k) = [x_1(k), x_2(k), ..., x_N(k)]$, $\mathbf{n}(k)$= $[n_1(k), n_2(k), ..., n_N(k)] \in \mathbb{C}^{N\times 1}$ is complex weight vector whose components corresponds to the weights of the beamformer, $(\boldsymbol{.})^T$ denotes the transpose operator and $(\boldsymbol{.})^H$ denotes the Hermitian transpose operator. The weights are computed by the MVDR algorithm which is described in the following section.

2.2. Minimum Variance Distortionless Response

The optimal weight vector $\mathbf{W}$ is obtained by a classical minimum variance distortionless response (MVDR) algorithm. MVDR algorithm minimizes the total output power $\mathbb{E}\left\{\left\|\mathbf{W}^H\mathbf{X}\right\|^2\right\}$, where $\mathbf{X}$ is the signal received at N elements, at the output of the antenna array while keeping unit gain in the look direction of the desired signal. i.e., it ensures distortionless response of the beamformer in the direction of the desired signal. The corresponding optimization problem is

$$\min_{\mathbf{W}} \mathbf{W}^H \mathbf{R}_{xx} \mathbf{W} \quad s.t. \quad \mathbf{W}^H \mathbf{a}(\theta_s) = 1 \qquad (3)$$

where $\mathbf{R}_{xx}$ is the signal covariance matrix. The solution to the optimization problem is given as

$$\mathbf{W}_{MVDR} = \frac{\mathbf{R}^{-1}\mathbf{a}(\theta_s)}{\mathbf{a}(\theta_s)^H\mathbf{R}^{-1}\mathbf{a}(\theta_s)} \tag{4}$$

where $(.)^{-1}$ denotes the inverse of the positive definite square matrix and $\mathbf{a}(\theta_s)$ is the steering vector of the desired signal. In practice,the interference-plus-noise covariance matrix is not known a priori and therefore it is substituted with an estimate of the sample covariance matrix of the received signal

$$\hat{\mathbf{R}} \triangleq \frac{1}{k}\sum_{k=1}^{K}\mathbf{X}(k)\mathbf{X}^H(k) \tag{5}$$

where k is the number of data snapshots available.

2.3. Digital Beamformer GNU Radio Blocks

The key GNU Radio signal processing blocks in the MVDR digital beamformer architecture shown in Figure 1 is tabulated in Table 1. The blocks in Table 1 are custom developed using C++ and Armadillo. In addition to the main blocks mentioned in Table 1, there is a frequency & phase synchronization block and a decoder block which are part of the BPSK receiver, a BERT block to compute the BER and an UHD block which is used to interface the GNU Radio to the USRP hardware. These blocks are available within the GNU Radio framework. The functions of the individual blocks in the digital beamformer are discussed in the following:

- *UHD*: UHD is a block available within the GNU Radio. This block acts as an interface to the USRP hardware. It receives data samples from the USRP receivers and forwards them to the blocks in the downstream.
- *sample covariance matrix*: this block receives the data samples from the *UHD* block and calculates the sample covariance matrix R_{xx} given in (5). The output of this block is provided to the downstream block *inverse covariance matrix.*
- *inverse covariance matrix*: it calculates the inverse of an 8×8 covariance matrix.
- *beamformer weight*: this block computes the beamformer weight vector given in (4).
- *beamformer sum*: this block receives the computed beamformer weight vector from the *beamformer weight* block and the data samples from the *UHD* block as its input. The output of this block is the beamformed signal.
- *frequency & phase synchronisation*: this block performs the frequency and phase synchronisation on the beamformed signal. The output of this block is sent to the downstream block *decoder*.

GNU Radio Block	Notation
Sample Covariance Matrix	$\mathbf{R}_{xx}$
Inverse Covariance Matrix	$\mathbf{R}_{xx}^{-1}$
Beamformer Weight	$\mathbf{W} = \frac{\mathbf{R}^{-1}\mathbf{a}(\theta_s)}{\mathbf{a}(\theta_s)^H\mathbf{R}^{-1}\mathbf{a}(\theta_s)}$
Beamformer Sum	$y = \mathbf{W}^H\mathbf{X}$

Table 1. Signal Processing Blocks in the Digital Beamformer.

- *decoder*: this block performs BPSK decoding on the data arriving at its input.
- *bert*: the BER is computed by this block.

3. SIMULATION

In this section, the performance of the digital beamforming system in terms of BER, and computational complexity are discussed. The simulations were carried over an AWGN channel. An eight elements circular antenna array with an element spacing of 0.5λ and centre frequency $f = 1.575$ GHz is considered. 3000 data snapshots were considered to calculate the array covariance matrix. The signal sampling rate is 5 MSps (Mega Samples per second).

3.1. Bit Error Rate

The performance of the GNU Radio based 8-channel beamforming system was measured in terms of the BER of a BPSK signal arriving at the azimuth and elevation angle $(\theta, \phi) = (0°, 0°)$. To show the BER gain achieved with beamforming, the BER performance of an 8-channel MVDR beamformer with BPSK receiver is compared with the BER of a single channel BPSK receiver. The single channel BER of a differentially encoded BPSK signal in an AWGN channel is given as [12]

$$BER = \frac{1}{2}erfc\left(\sqrt{\frac{E_b}{N_0}}\right)\left(1 - erfc\left(\sqrt{\frac{E_b}{N_0}}\right)\right) \tag{6}$$

where erfc$(.)$ is the complementary error function. The theoretical BER gain of an N channel beamforming system when compared to a single channel system is given as $10\log(N)$. For an 8-channel beamformer, the theoretical BER gain is 9 dB. Figure 2 shows the probability of bit error for a single channel BPSK receiver using (6) and its experimental results matches closely. The experimental BER of the 8-channel MVDR beamformer under AWGN conditions is also included in Figure 2. It is observed from Figure 2, the probability of bit error for an 8 channel MVDR beamformer has a BER gain of 9 dB in comparison to the single channel BPSK system.

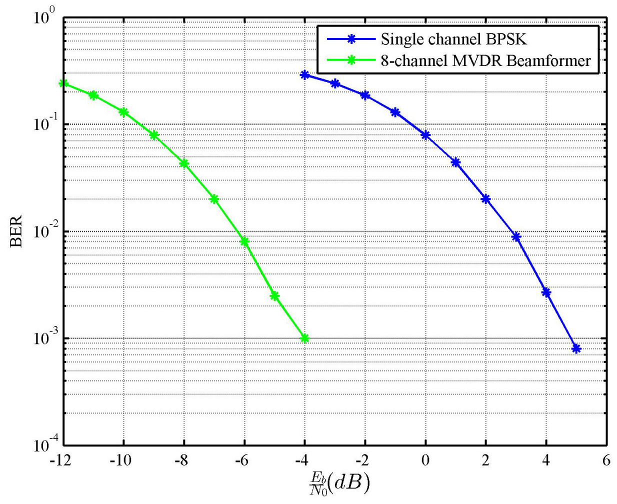

Fig. 2. BER performance of MVDR algorithm compared with conventional BPSK system.

3.2. Computational Analysis

This section introduces the performance tools and performance metrics used to analyse the MVDR beamforming system. This study is important since it shows that, the beamforming system in its current state does not reach the performance limits of the computer system and thus can perform beamforming computations without limitations. Otherwise, the beamforming system would not be able to cope with the sample rate of the incoming data stream, resulting in data samples drop-offs. This is referred to as near-real-time or low latency operation. To achieve low latency operation of the MVDR digital beamforming system, it is important to know the resource utilization of each block in the beamforming system. Knowing the computational requirements and performance of each block in the digital beamforming system helps us to identify the part of the system that utilizes heavy computational resources and hence requiring resource optimization. Unlike FPGA's in which the timings are controlled by a common clock that can guarantee specific timing requirements, the software radios which run on GPP's with various operating systems (OS) like Linux, have uncertain execution times due to the shared scheduling nature of the operating system (soft real-time) [13] and other external processes. The problem is compounded especially if the application is multi-threaded. Because of these limitations, exact timing analysis of the MVDR beamforming system is complicated. Hence, we limit our analysis to the metrics that can be analysed with the tools available within GNU Radio platform. The analysed performance metric includes the average utilization of the buffer and the average runtime of various blocks of the beamforming system. The simulations were carried on a PC with an Intel i7-3770 processor and 16 GB of RAM, running Ubuntu 12.04. The algebraic operations mentioned in Table 2 were implemented using C++ and Armadillo linear algebra library. The computational cost in Table 2 is referred from [14]. Figure 3 represents the buffer fullness and runtime performances of the beamforming system in the form of a flow graph. The node size in the flow graph is proportional to the runtime and the edge thickness is proportional to the buffer fullness of the beamforming system. This buffer utilization investigation provides insight into where the samples might be queued resulting in the dropping of samples and affecting the performance of the system. This result gives an indication that the buffer from the source to the spatial auto-correlation matrix is near full. This is because, the spatial auto-correlation matrix needs to store and process K snapshots of data at any given time. However, we can see that the buffers of the path involving inverse auto-correlation block and MVDR weight block are near empty, indicating that all the samples are almost immediately consumed. Figure 4 shows the average buffer utilization of the individual blocks namely 'signal source', 'spatial correlation', 'inverse correlation' and 'MVDR weight' in the beamforming system. From the results, it can be seen that the source block (signal source) utilizes maximum buffer. The buffer utilization of rest of the blocks in the system is negligible, indicating there is no queuing up of samples that would result in data overflow. Another important performance metric considered for low latency capabilities is average runtime of the algorithm. Figure 5 shows the average runtime of each of the blocks in the beamforming system. It is observed that, about approximately 65% of the computational time is spent in performing the spatial auto-correlation matrix. It is reasonable given the algorithmic complexity in processing K data snapshots at any given instant. The inverse of the spatial auto-correlation matrix and beamforming weight computation processes takes about 28% and 5% respectively. These results are in conformance with the complexity of various operations given in Table 2.

Computation	Size	Cost
$\mathbf{R}_{xx}$	$N \times K \times K \times N$	$\mathcal{O}(N^2K)$
$\mathbf{R}_{xx}^{-1}$	$N \times N$	$\mathcal{O}(N^3)$
$\mathbf{W} = \frac{\mathbf{R}^{-1}\mathbf{a}(\theta_s)}{\mathbf{a}(\theta_s)^H\mathbf{R}^{-1}\mathbf{a}(\theta_s)}$	$N \times N \times N \times 1$	$\mathcal{O}(2N^2 + 3N)$
$y = \mathbf{W}^H\mathbf{X}$	$1 \times N \times N \times K$	$\mathcal{O}(NK)$

Table 2. Computational Complexity of MVDR Algorithm.

4. CONCLUSIONS

In this paper, we presented a GNU Radio based MVDR digital beamforming system. The performance of the beamforming system in recovering the desired signal was studied in terms of the BER achieved. The BER simulations match the theoretical BER gain. Understanding the computational complex-

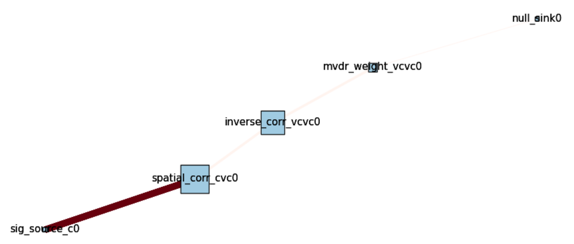

Fig. 3. Flowgraph representing the buffer utilization and runtime performance of the beamforming system.

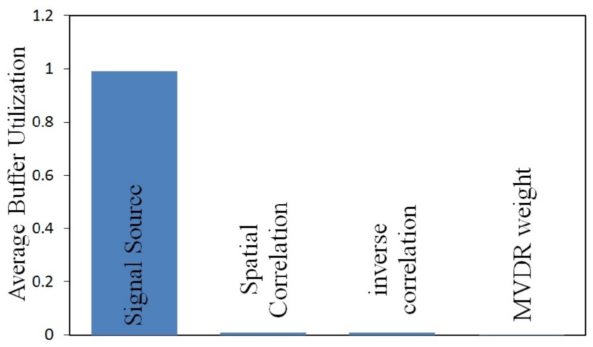

Fig. 4. Average buffer utilization of individual blocks of the beamforming system.

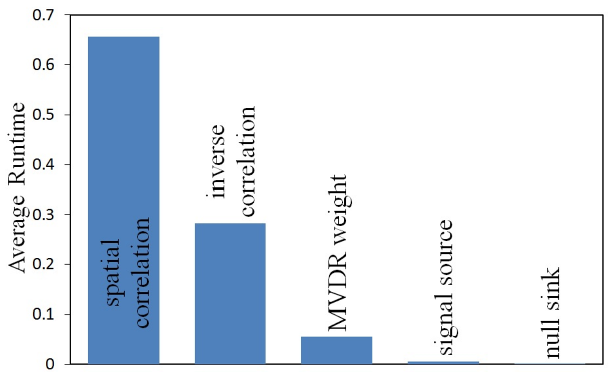

Fig. 5. Total runtime consumed by individual blocks of the beamforming system.

ity of the beamforming system is critical to achieve real-time capability. To that end, our work also investigates the buffer utilization and runtime complexity of the beamforming system. Performance measurements confirms the implemented software beamforming system was able to achieve real-time performance.

5. REFERENCES

[1] J. Mitola, "The software radio architecture," *Communications Magazine, IEEE*, vol. 33, no. 5, pp. 26–38, May 1995.

[2] GNU Radio Website, accessed June 2014. [Online]. Available: http://www.gnuradio.org

[3] [Online]. Available: http://ossie.wireless.vt.edu

[4] K. Tan, H. Liu, J. Zhang, Y. Zhang, J. Fang, and G. M. Voelker, "Sora: high-performance software radio using general-purpose multi-core processors," *Communications of the ACM*, vol. 54, no. 1, pp. 99–107, 2011.

[5] B. Bloessl, M. Segata, C. Sommer, and F. Dressler, "An IEEE 802.11a/g/p OFDM Receiver for GNU Radio," in *ACM SIGCOMM 2013, 2nd ACM SIGCOMM Workshop of Software Radio Implementation Forum (SRIF 2013)*. Hong Kong, China: ACM, August 2013, pp. 9–16.

[6] L. T. Ong, "An usrp-based interference canceller," in *Communication Systems (ICCS), 2012 IEEE International Conference on*. Singapore: IEEE, 2012, pp. 95–99.

[7] Ettus research website. [Online]. Available: http://www.ettus.com

[8] C. Sanderson, "Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments," NICTA, Tech. Rep., 2010.

[9] S. Haykin, *Adaptive Filter Theory (3rd Ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1996.

[10] J. Capon, "High-resolution frequency-wavenumber spectrum analysis," *Proceedings of the IEEE*, vol. 57, no. 8, pp. 1408–1418, Aug 1969.

[11] C. A. Balanis, *Antenna Theory: Analysis and Design*. Wiley-Interscience, 2005.

[12] J. G. Proakis, *Digital communications, 1995*. McGraw-Hill, New York.

[13] T. W. Rondeau, T. O'Shea, and N. Goergen, "Inspecting gnu radio applications with controlport and performance counters," in *Proceedings of the second workshop on Software radio implementation forum*. ACM, 2013, pp. 65–70.

[14] P. C. Javier Arribas, Carles FernndezPrades, "Multi-antenna techniques for interference mitigation in GNSS signal acquisition," *EURASIP Journal on Advances in Signal Processing 2013*, **2013**:143.