International Journal of Research in Engineering, Science and Management
Volume 6, Issue 4, April 2023
https://www.ijresm.com | ISSN (Online): 2581-5792

Adaptive Beamforming using Minimum Variance Distortionless Response

R. Nandakrishnan[1*], S. Arjun[2], C. Navanit Nandakumar[3], Saurav Sajesh[4], K. G. Harikrishanan[5], P. S. Aparna Devi[6]

[1,2,3,4,5]UG Student, Department of Electronics and Communication Engineering, Government Model Engineering College, Kochi, India
[6]Assistant Professor, Department of Electronics and Communication Engineering, Government Model Engineering College, Kochi, India

***Abstract*: This paper proposes a system to implement adaptive beamforming technique using minimum variance distortionless response (MVDR) with enhanced nulling level control. In this paper, the key motive is to maximize sensitivity in one direction by minimizing output power of beamforming.**

***Keywords*: Minimum Variance Distortionless Response (MVDR), Hardware Description Language (HDL), Digital Signal Processing (DSP).**

1. Introduction

Adaptive beamforming using MVDR (Minimum Variance Distortionless Response) improves the quality of signals received from a certain direction by eliminating noise and interference from other directions. On hard-ware platforms like FPGAs and ASICs, tools like HDL Coder and System Generator make it easier to create and implement digital signal processing algorithms. To increase the speed and effectiveness of signal processing systems, there has been an increase in interest in adapting adaptive beamforming techniques to hardware platforms in recent years. The goal of this research has been to create high-performance architectures with low power consumption that can handle real-time signal processing jobs [1]. This article shows that adaptive beam-forming with MVDR may be implemented on hardware platforms like FPGAs. To enhance the quality of signals received in noisy environments, the suggested system can be employed in a variety of applications, such as wireless communication systems, radar systems, and biomedical signal processing systems.

2. Simulink

A graphical programming environment called Simulink is used to model, simulate, and analyse dynamic systems and is extensively employed in sectors including robotics, automotive, and aerospace. Many add-ons that enhance Simulink's functionality are supported. System Generator and HDL Coder are two examples of these add-ons.

Simulink users can develop and simulate digital signal processing (DSP) systems using the System Generator add-on. For FPGA and ASIC implementation, it enables designers to produce excellent, synthesizable VHDL and Verilog code. Building complicated DSP systems can be done using the System Generator block library, which has blocks for arithmetic, logic, signal processing, memory, and communication functions.

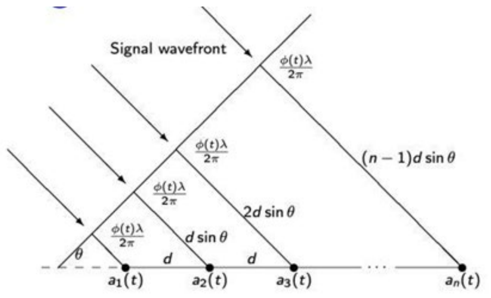


Fig. 1. m samples, n antenna elements, m ≫ n. m by n data matrix A. a(t) is a n by 1 column vector. $a(t)^H$ form the rows of A [14]

Users can create synthesizable VHDL and Verilog code from Simulink models with the add-on HDL Coder. It can automatically create test benches and documentation for the created code, as well as a variety of FPGA and ASIC targets. Many Simulink blocks, including those from the DSP System Toolbox, Communications Toolbox, and other Simulink add-ons, are supported by HDL Coder.

Digital systems, particularly those using DSP, can be designed and implemented more quickly using both System Generator and HDL Coder. Compared to conventional hardware description languages, they enable designers to work at a higher degree of abstraction while still producing excellent, effective hardware implementations.

3. Minimum Variance Distortionless Response

This is a method used in signal processing to deter-mine a signal's properties after it has been captured by an array of sensors.

The MVDR method maintains a distortion-free response to the required signal while minimizing the variance of the predicted signal. In applications like beamforming, where several sensors are combined to separate a desired signal from a noisy background, this technique is especially helpful. In

*Corresponding author: nandakrishnan.rajeev@gmail.com order to create an estimate of the desired signal, the signals received by the sensors are combined using the weight vector w. A high signal-to-noise ratio can be provided by MVDR, improving performance in noisy conditions by reducing the variation of the estimated signal [2], [3].

$$w = R^{-1} * p / (p^{H} * R^{-1} * p) \tag{1}$$

where:
w is the weight vector
R is the covariance matrix of the received signal
p is the vector representing the desired signal
(H) denotes the Hermitian transpose operator

A. *Singular Value Decomposition (SVD)*

The objective of MVDR is to estimate a signal's direction of arrival while reducing interference from other directions. This is accomplished by utilising an array of sensors to produce a beam in the desired direction and varying the weights of the sensors according to the covariance matrix of the data received. By resolving a linear system of equations involving the covariance matrix and an output power restrict ion, the weights are produced [4]. One typical method is to calculate the inverse of the covariance matrix using the singular value decomposition (SVD). The SVD factors the matrix into a sum of three unitary, diagonal, and additional unitary matrices. The singular values of the matrix are contained in the diagonal matrix, which can be utilised to calculate the inverse. Then, using a linear combination of the sensor data, the resulting beamforming weights are calculated.

$$\text{Covariance matrix solve: } (A^{H}A)x=b \tag{2}$$

$$\text{MVDR weight vector: } w=x/(b^{H}x) \tag{3}$$

$$\text{MVDR response: } y=w^{H}a(t) \tag{4}$$

B. *QR and Cholesky Decomposition*

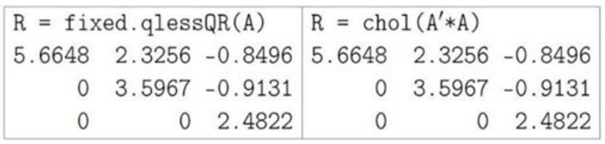
1) *Cholesky Factorization: Covariance Matrix*

Solve: $(A^{H}A)x=b$
MATLAB Code: $R=\text{chol}(A' * A)$

Cholesky requires symmetric positive definite input, which $A^{H}A$ is $R=\text{chol}(A'A)$, here R is upper right triangle and $R' * R = A'A$ [5], [6].

2) *QR vs Cholesky*

Computing $A'A$ squares the condition number of A, but may be better for faster updates in some hardware applications.



R = fixed.qlessQR(A)	R = chol(A'*A)
5.6648 2.3256 -0.8496	5.6648 2.3256 -0.8496
0 3.5967 -0.9131	0 3.5967 -0.9131
0 0 2.4822	0 0 2.4822

Fig. 2. QR vs. Cholesky [14]

4. HDL Coder Implementation

Implementing MVDR using HDL Coder can be useful for applications that require real-time beamforming and processing of RF signals in hardware. This approach can provide a number of benefits over software-based implementations, including:

High processing speed: By implementing MVDR in hardware using HDL Coder, you can achieve much higher processing speeds than would be possible with a software-based implementation. This is because the HDL code is optimized for hardware execution, and can take advantage of parallel processing capabilities of FPGAs and ASICs [7].

Low power consumption: Hardware-based implementations of MVDR using HDL Coder can be designed to be highly power-efficient, which can be important in battery-powered or portable devices. This is because the hardware implementation can be optimized to minimize power consumption, and can avoid the overhead asso-ciated with running a full software stack.

Small form factor: Hardware-based implementations of MVDR using HDL Coder can be designed to be very compact, which can be important in applications where size and weight are critical factors. This is because the HDL code can be optimized for a specific hardware platform, and can be designed to minimize the use of hardware resources.

Real-time operation: Hardware-based implementations of MVDR using HDL Coder can provide real-time processing of RF signals, which can be important in applications such as radar, wireless communications, and medical imaging. This is because the HDL code can be optimized to provide low-latency processing of signals, which is critical for real-time applications [8].

A. *RF input signal*

The RF input signal is generated with the help of simulink blocks such as Random number, QPSK, Square root raised cosine filter, cubic polynomial and amplifier. Block diagram of input generator is shown below. Function of each block is listed below:

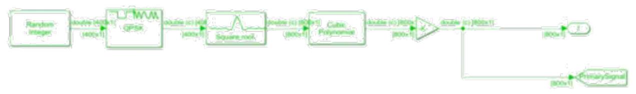


Fig. 3. Block diagram of input generator

Random number: Random number block provides random output of integers, the range of integers can be configured, here the range is set from 0 to 3 such that stream of integers from 0 to 3 will be generated randomly. This stream of integers is given to the QPSK block.

QPSK: Here the stream of integers from 0 to 3 is modulated by QPSK. Quaternary signaling schemes are all examples of quadrature-carrier multiplexing system, as they combine two independent signals to occupy the same transmission bandwidth, and yet it allows for the separation of the two signals at the receiver.

Square root raised cosine filter: The output from QPSK is pulse shaped by the filter to avoid inter symbol interference (ISI).

Cubic Polynomial: To include the irregularities of the filter and amplifier cubic polynomial is used. amplifier: To amplify the RF signal [9].

B. Interfering signal

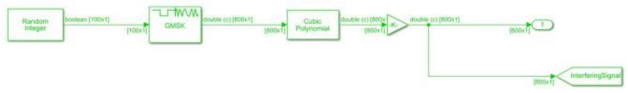


Fig. 4. Block diagram of interference generator

The RF interference signal is generated with the help of simulink blocks such as Random number, GMSK (Gaussian Minimum Shift Keying), cubic polynomial and amplifier. Function of each block is listed below.

Random number: Random number block provides ran-dom output of integers, the range of integers can be con-figured, here the range is set from 0 to 1 (boolean) such that stream of intergers from 0 to 1 will be generated randomly. This stream of integers is given to the GMSK block.

GMSK: Here the stream of integers from 0 to 1 is modulated by GMSK.

Cubic Polynomial: To include the irregularities of the filter and amplifier cubic polynomial is used. amplifier: To amplify the interference signal.

C. HDL Coder Block Diagram of MVDR

A linear array of sensors collects the input signals, which are then fed into a bank of adaptive weights that the MVDR algorithm computes. A beamformed output is created by multiplying the adaptive weights by the received signals and summing the resulting products.

The MVDR method reduces the beamformer's output power while keeping a cap on how quickly it may react to signals coming from a particular direction. A vector that represents the desired response of the beamformer is multiplied by the inverse of the covariance matrix of the received signals to achieve this.

The beamformed output is then calculated using the resulting vector of weights. Compared to traditional algorithms, the MVDR algorithm offers superior spatial resolution [10].

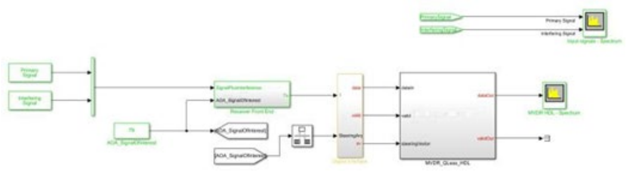


Fig. 5. HDL coder block diagram of MVDR

5. Zynq Ultrascale MPSoC

Xilinx created a family of multi-core, highly integrated system-on-chip (SoC) devices called the Zynq UltraScale+ MPSoC. They produce a hybrid computing platform that is highly suited for a variety of applications, including aerospace and defence, automotive, industrial, and wireless infrastructure by combining high-performance processing with programmable logic [11].

The Zynq UltraScale+ MPSoC devices have programmable logic in the form of Xilinx's FPGA technology and up to six computing cores, including quad-core and dual-core ARM Cortex-A53 processors. Together with high-speed serial transceivers and a variety of on-chip peripherals and interfaces, including as Ethernet, USB, PCIe, and DDR memory controllers. In order to deliver high performance, low power consumption, and flexibility in a single device, the Zynq UltraScale+ MP-SoC family was created. As a result, it is well-suited for a variety of applications that call for real-time processing, low latency, and high-bandwidth data transfer [12], [13].

The Zynq Ultrascale MPSoC includes a variety of features that are specifically designed for RF applications, such as high-speed ADC and DAC interfaces, RF data converters, and RF front-end modules. It also includes a range of high-speed interfaces, including PCIe, USB 3.0, and Gigabit Ethernet, which can be used to interface with other RF components, such as antennas, transceivers, and amplifiers.

Fig. 6. Zynq Ultrascale MpSoC

6. Results

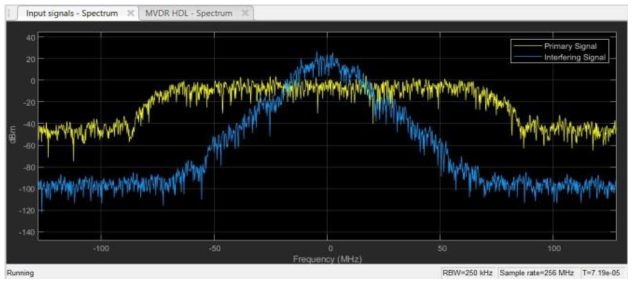


Fig. 7. Input & interference simulation output

Fig. 7 shows the QPSK modulated input signal and GMSK modulated interference signal. The combination of both these signals is given to the MVDR block along with the steering vector.

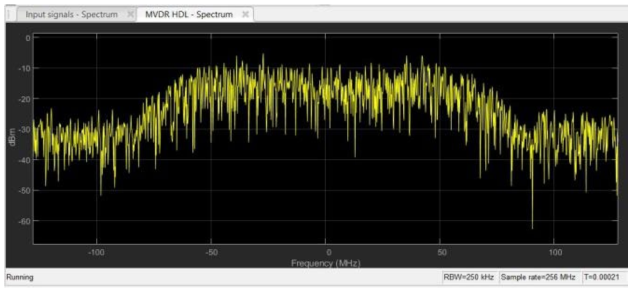


Fig. 8. HDL coder output of MVDR

Fig. 8 shows the MVDR response of the signal plus interference signal, which closely resembles the input signal. The sharp downward peak in the MVDR response shows the enhanced nulling of the interference.

7. Conclusion

This paper presented a study on adaptive beamforming using minimum variance distortionless response.

References

[1] S. Haykin, Adaptive Filter Theory, 2nd Ed. Englewood Cliffs, NJ: Prentice-Hall, 1991

[2] H. Robbins and S. Monro, "A stochastic approximation method," Ann. Math. Statist., vol. 22, pp. 400 –407, 1951.

[3] J. Cioffi, The fast adaptive rotor's RLS) algorithm. IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-38. 631453,1990.

[4] F. L. Liu, J. K. Wang, C. Y. Sun, and R. Y. Du, "Robust MVDR beam-former for nulling level control via multi-parametric quadratic programming," Progress in Electromagnetics Research C, vol. 20, 239–254, 2011.

[5] Benesty, J., Chen, J., & Huang, Y. (2008). Microphone Array Signal Processing. Springer-Verlag Berlin Heidelberg.

[6] Van Trees, H. L. (2002). Optimum Array Processing (Part IV of Detection, Estimation, and Modulation Theory). Wiley-Interscience.

[7] Van Veen, B. D., & Buckley, K. M. (1988). Beamforming: a versatile approach to spatial filtering. IEEE ASSP Magazine, 5(2), 4-24.

[8] Johnson, D. H., & Dudgeon, D. E. (1993). Array Signal Processing: Concepts and Techniques. Prentice-Hall.

[9] Mestre, X., & Lagunas, M. (1997). Adaptive filtering and equalization using unconstrained optimization. Kluwer Academic Publishers.

[10] Cox, H., & Zeskind, R. M. (1987). Robust adaptive beamforming. IEEE Transactions on Acoustics, Speech, and Signal Processing, 35(10), 1365-1376.

[11] Frost, O. L. (1972). An algorithm for linearly constrained adaptive array processing. Proceedings of the IEEE, 60(8), 926-935.

[12] pg105-cordic.pdf, Viewer, AMD Adaptive Computing Documentation Portal (xilinx.com)

[13] https://docs.xilinx.com/r/en-US/ds925-zynq-ultrascale-plus

[14] Tom Mealey (2023). FPGA, https://github.com/mathworks/FPGA-Adaptive-Beamforming-and-Radar-Examples/releases/tag/1.1.0.3, GitHub. Retrieved April 9, 2023.