Indian Journal of Science and Technology, *Vol 8(19), DOI: 10.17485/ijst/2015/v8i19/77051, August 2015*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

VLSI Architecture for Broadband MVDR Beamformer

N. Hema[1*], Jayaraj U. Kidav[2] and B. Lakshmi[1]

[1]SENSE Department, VIT University, Chennai - 600127, Tamil Nadu, India;
1990.hema@gmail.com, lakshmi.b@vit.ac.in
[2]Department of VLSI Design, NIELIT, Calicut University, Calicut - 673635, Kerala, India;
jayaraj@calicut.nielit.in

Abstract

Background/Objectives: The main objective of this proposed work is to investigate the significance of adaptive beamforming technique and to develop an efficient VLSI architecture for broadband MVDR beamformer in the field of medical ultrasound imaging. **Methods/Statistical Analysis**: The proposed algorithm is Minimum Variance Distortionless Response (MVDR) for near field beamforming of broadband data which gives better contrast and resolution compared to conventional Delay and Sum (DAS) beamformers and is implemented in frequency domain. MVDR beamformer minimizes the output power by allowing the desired signal to pass undistorted with unity gain. The solution for this optimization problem, involves correlation matrix inversion, which is the challenging part of MVDR algorithm. **Findings:** MVDR algorithm is applied by finding the inverse of correlation matrix which is a complex matrix. Here four elements are used for simplicity. Calculation of inverse complex matrix is the challenging part of MVDR algorithm where different methods are being used. Here QR decomposition is used which follows givens rotation algorithm. The paper demonstrates the formal verification of the proposed work. Final result is compared with the golden reference model which is designed using FIELD II scanner in Matlab. From the results it's seen that, MVDR beamformer gives a pencil like beamform which shows high resolution and better contrast when compared to DAS. The timing constraints and device utilization parameters are obtained from synthesis report of final architecture designed in FPGA. **Conclusion/Improvements:** MVDR algorithm gives a pencil like beamformer output with reduced main lobe width and reduced side lobe level. By upgrading number of elements from 4 to 64, it can be made real time.

Keywords: Broadband, Correlation Matrix, DAS, MVDR Beamformer

1. Introduction

Ultrasound imaging plays a vital role in diagnostic purposes for decades. Ultrasound imaging is one of the safest methods to capture the internal images of human body as it does not use any ionizing radiation. Also it has added advantages like it is relatively portable, in-expensive and causes less discomfort for the patient. It has significant role in the fetal scan during pregnancy. It is carried out by emitting high frequency acoustic wave fields in to the patient's body. This frequency is in the range of MHz and is out of human's audible range. When these acoustic waves encounter any change in the density of tissues while comparing to normal ones, then these waves get reflected back at different depths within the human body. These reflected waves are measured by transducers which has piezo electric crystals in it. By delaying and summing these signals using array signal techniques, the final ultrasound image is formed. These signals are conventionally processed using DAS beamformers[1]. In DAS beamformers, apodization weights are predefined and are independent of the input data. Since there exists a compromise between side lobe level and main lobe level, smoothing apodization function such as Hanning is used. It reduces side lobe level at the expense of increase in main lobe width[2].

The Capon or MVDR beamformer is an adaptive beamformer which updates its apodization weights

*Author for correspondence

in such a way that the variance of the bamformer is minimized under the condition, the desired signal passes without any distortion[3]. The main objective of this paper is to design a VLSI architecture that effectively performs the function of MVDR beamformer. The ultrasound wavefields[4] are near field and broadband and so the same is applied here.

Section 2 deals with the MVDR beamformer, with beamforming explained in detail.

2. MVDR Beamformer

MVDR beamformer is one of the innovative ideas being applied in the field of medical ultrasound imaging as it requires near field broadband data as the beam to be applied to the phantom, the body mimicking system.

2.1 Beamforming

In MVDR beamformer, initially an input signal was applied to an array transducer, calculated the delay and then MVDR algorithm is applied in Field II[2], an ultrasound image scanner[5]. The objective of MVDR algorithm is to minimize the variance by transferring the desired signal without any distortion with unity gain. This is the main constraint followed in MVDR algorithm and can be expressed as,

$\min W^H R_y W$
W
subject to $W^H e = 1$
where power,
$P = W^H R_y W$

w is the adaptive apodization complex weight, {.}H is Hermitian transpose, R_y is correlation matrix[6].

This correlation matrix[4] can be found by,

$$Ry = E\{X.XT\}$$

Optimization problem for this can be given by,

$$w = \frac{R_y^{-1} e}{e^H R_y^{-1} e}$$

where e is the steering vector in desired direction[7].

2.2 QR Decomposition

To find the complex matrix inversion of correlation matrix, QR Decomposition technique is used[8]. It is done using Givens rotation. This is the most hectic task while using MVDR algorithm. Matrix inversion utilizing QR decomposition obliges various rotations to invalidate the undesirable values. Givens Rotation (GR), a standard rotation algorithm is utilized for that. Givens Rotations (GR) are kind of unitary changes that are known to be altogether less sensitive to round-off errors and to show a high level of numerical soundness[9]. They work specifically on the data information instead of methods in view of the correlation matrix which are numerically more discriminating and have bigger word length necessities.

Let A be a square matrix, whose inverse is expressed as A-1. As per the properties of matrix, it can be written as,

$$A \times A{-}1 = I$$

Where I is identity matrix.

QR decomposition is a simple operation, which deteriorates a matrix into an orthogonal and a triangular matrix[7]. Square matrix A is undergone QR decomposition to give A = QR, where Q is an orthogonal matrix which by property gives, $Q^T \times Q = I$ and R is an upper triangular matrix. The crumbling of m × n matrix (with m>= n) of full rank is the aftereffect of a m × n orthogonal matrix and a n × n upper triangular matrix[5].

QR decomposition can be transformed with a movement of Givens rotations. Each turn zeros an element in the sub inclining of the matrix, confining the R matrix. The concatenation of all Givens rotations outlines the orthogonal matrix, Q[10].

3. Matlab Modelling

In ultrasound field system, the images are simulated using an ultrasound image scanner Field-II. The rfdata given as input to this in MATLAB is delayed and summed to get conventional beamformer output, while same rfdata is delayed and then MVDR algorithm is applied to get adaptive beamformer output. In the former, the apodization is data-independent while in later one, the complex apodization weight is data dependent. This output graphs which clearly shows the comparison between DAS and MVDR beamformer is the golden reference model which is later compared with the RTL implementation output. The same rfdata used in fieldII is saved as text file and is loaded in DPRAM as coe file, which is given as input to FFT IP core, so that the broadband signals are passed through narrow band filters and finally applied to MVDR beamformer. Formal verification is the procedure adopted in this work. The result obtained from RTL

implementation is compared with the golden reference model of MATLAB. Flow diagram is shown in Figure 1.

3.1 Field-II

FIELD-II is an ultrasound image scanner being used in ultrasound imaging field for simulation purposes. FIELD-II is a product system utilized for making simulations of a wide range of linear therapeutic ultrasound systems. The spatial impulse response and linear acoustic models are utilized by the medical ultrasound system for generating simulations of the transducer pressure and the backscattered field from a limited number of point scatterers. It gives a precise and effective methodology of computing a wide range of fields, for example, discharged and pulsed with all conceivable transducer geometries. The project is fit for making simulated results of all picture sorts, for example, simulated human phantoms, cyst phantoms what's more, stream imaging phantoms[5].

In this paper, the input data to MVDR beamformer is loaded from Field-II and finally comparing with the simulation obtained in MATLAB.

4. Architecture Design

In VLSI architecture design of array signal processor, bottom-up approach is adopted. Initially sub architectures are designed and finally all of them are integrated to give adaptive beamformer output. FPGA implementation is done with reference to the algorithm that have been developed and verified in the simulation model of MATLAB. VLSI architecture consists of processor blocks, memory blocks and controller. Processor block consists of FFT processor, bin processor and adaptive weight computation processor. Dual Port RAM is a memory which has dual ports, one port used to write the data into memory and other port used to read the data from memory. Memory block consists of a memory to store input to be given to FFT, real and imaginary DPRAMs to store output from FFT memory and from which input is given to correlation matrix, other set of DPRAMs to store output from correlation matrix and from which input is given to adaptive weight computation processor, again other set of DPRAMs to store output from adaptive weight computation processor and from which input is given to output power computation memory and finally a memory to store the beamformer output. Controller generates control signals for the synchronization of all these processors and memories.

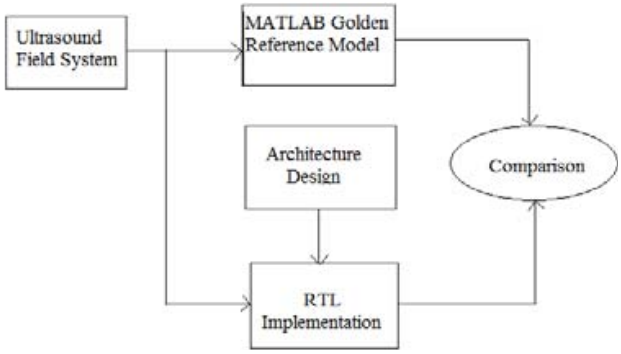

Figure 1. Flow diagram.

4.1 Design and Constraints

Figure 2 shows the components to be designed that include, sensors, FFT processors, bin processors, correlation matrix processor and finally beamformer output along the steered angle.

4.2 High Level Architecture

High level architecture of the design is shown in Figure 3. It has a system controller that controls the whole processing of the system to which main clock, reset and start signal is given. Finally controller outs the done and end signals.

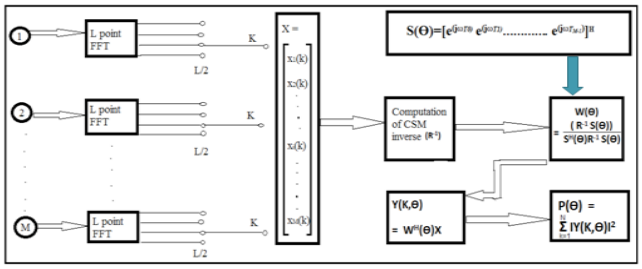

Figure 2. Design component.

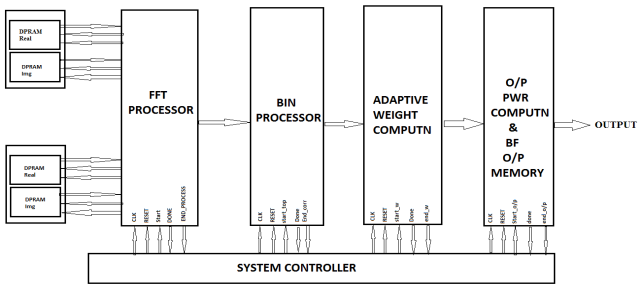

Figure 3. System architecture.

From Figure 3 it is clear that it has four main processors called FFT processor, bin processor, adaptive weight computation processor and output computing engine. Initially some samples from four sensors were loaded in DPRAM (dual port RAM). Input signal was given in MATLAB. In MATLAB beamformer output was obtained using Field II ultrasound scanner by applying MVDR algorithm. The graph obtained by MVDR algorithm is compared with the graph obtained by conventional delay and sum beaformer. From MATLAB the samples were taken from the signal in a text format, which later converted to coe file and are loaded to DPRAM. Here 256 samples from four signals given by four sensors are taken in to consideration. These values are stored in 1.15 formats. These samples from DPRAM are given to FFT processor, where IP core of FFT processor is used. Here 256 point FFT is used. FFT out is given to bin processor, where out of 256 samples, only 127 samples were taken. In bin processor, correlation matrix and its inverse is found, where QR Decomposition was applied for finding complex matrix inverse. This is applied to adaptive weight computation processor, where an external input called steering vector was given, to make the desired signal to pass undistorted as per (Minimum Variance Distortion less Response) MVDR algorithm. Thus optimization problem is done according to this algorithm where complex division also came in to picture. Finally power was computed by taking the summation of absolute value of beamformer output and stored in a DPRAM. Finally, the beamformer output is compared with the output obtained from MATLAB, which is called as golden reference model.

4.3 Low Level Architecture

Low level architectures consist of four sub architectures. They are FFT processor, bin processor, adaptive weight computation processor and beamformer output processor. Among which bin architecture is again divided into correlation matrix computation engine and correlation matrix inversion engine.

4.3.1 Sub Architecture I-FFT Processor

Figure 4 shows the sub architecture of FFT processor. Samples loaded from MATLAB are of 1.15 formats with each sample having 16 bit hexadecimal values. DPRAM with 16 bit width and 256 memory locations are taken. Samples are loaded in as coe file to DPRAM. These values are then given as input into the FFT IP core. IP core

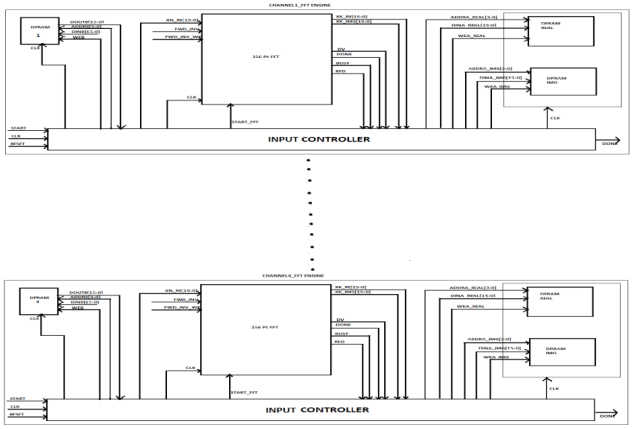

Figure 4. FFT processor.

is set as radix 2, 256 point FFT. Since four sensors are considered, four FFTs are used each having different controller. Finally all four are topped to a single module and the FFT out is stored in another DPRAM. Since FFT has real and imaginary values, each channel required two different DPRAMs. From 256 samples only 127 values are taken at the output, since after (N/2+1) values, FFT will repeat the output.

Following are the states in the controller FSM of FFT processor shown in Figure 5.

S0, S1, S2, S3 and S4 are Initial state, Read, Data2FFT, Write and Stop respectively.

4.3.2 Sub Architecture II-Bin Processor

Figure 6 and Figure 7 shows the computation engine of correlation matrix and its controller. Here the data is fetched from the DPRAM, which bears the out of FFT. This data is taken as input vector, X for MVDR algorithm which has 16 bit values in 127 memory locations.

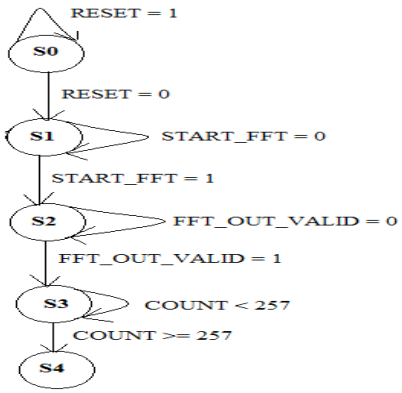

Figure 5. FFT processor controller FSM.

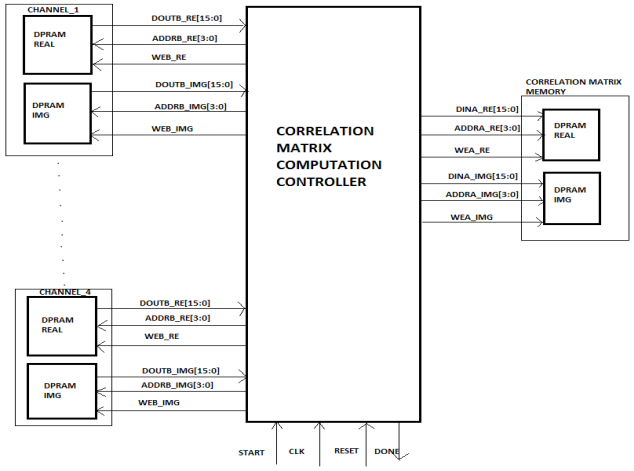

Figure 6. Correlation matrix computation engine.

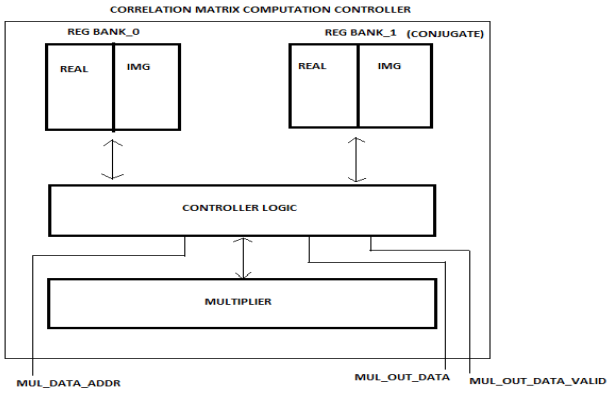

Figure 7. Controller of correlation matrix computation engine.

Now, for finding correlation matrix, the transpose of X, represented as X^T was found. Thus correlation matrix, R is computed as

$$R = E\{X.X^T\}.$$

Since it's the multiplication between two complex matrices, multiplication logic is applied in the controller for finding R. Now, correlation matrix, R has 32 bit values, which is then directly given to next memory called, 'A' memory in matrix inversion block, to find the inverse of correlation matrix, R^{-1}.

Figure 8 shows the controller FSM of correlation processor where the states are S0, S1, S2, S3 and S4 are initial_state, read, save2reg, restart and stop respectively. S2 state in this FSM controls the controller of correlation matrix inversion processor, whose controller FSM is also shown in Figure 8 and the states in it are S0, S1, S2, S3

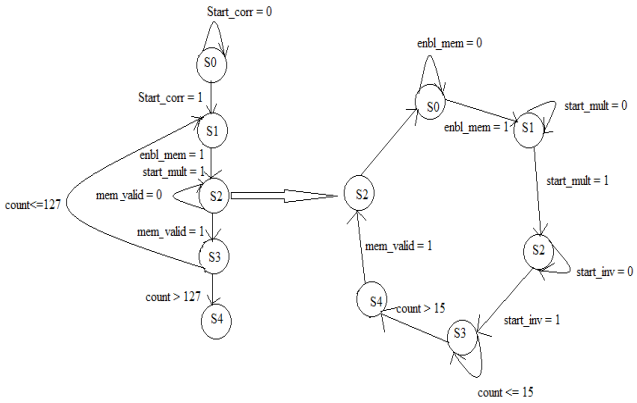

Figure 8. Controller FSM of Correlation matrix and its inversion.

and S4 are initial_state, save2reg, cmplx_mult, write and mem_out respectively.

Figure 9 shows the correlation matrix inversion. Thus the output from correlation matrix computation engine is given as input to the A memory in the correlation matrix inversion engine. It also shows how the information is streaming between distinctive modules. At first the input matrix is given to A1 memory directly. From the A1 memory, the controller will read the X and Y values and is given to the processor. The yield of processor is written into the G memory as indicated by the address generation programs written in controller. At that point, controller will read the A1 memory and G memory and provide it to the multiplier block. The yield of the multiplier is written into A2 memory. At that point, controller read A2

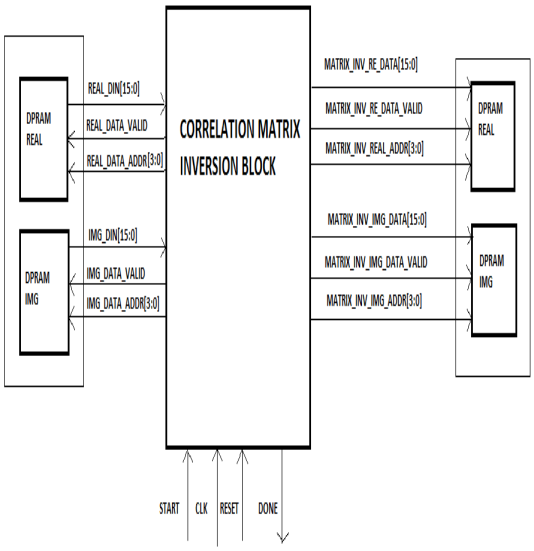

Figure 9. Correlation matrix inversion engine.

memory and writes it into A1 memory. And then the G memory is again read and the transpose of values in G memory is written into T memory. After this, controller read the Q1 memory and T memory and gives the information to matrix multiplier. The yield of matrix multiplier is written into Q2 memory. At that point, controller read Q2 and sends in to Q1 memory. In the wake of finishing the loop, A1 memory will contain the triangular matrix R and Q1 memory will contain the orthogonal matrix Q. After that, controller read A1 memory and R memory and find the inverse of upper triangular matrix utilizing back substitution strategy. The outcome is overwritten in to the R memory, during each and every iteration. At that point, controller read Q1 memory and finds its transpose and the outcome is saved into T memory. At long last, controller read both R memory and T memory for matrix multiplication and the final result will be stored into the output memory Ainv. In Figure 10, its architecture is shown.

Now in Ainv memory, matrix inversion of real value is stored. This Ainv is multiplied to the values in imaginary matrix and the output is saved in a DPRAM, called as r0. This value in r0 is again multiplied with the values in imaginary matrix and the output is added with the values in real matrix using adder logic. This output is again given to matrix inversion block from which final real part of complex matrix inversion is computed.

Now, this real part value is multiplied with the value in r0 memory and then takes its negative, so that final imaginary part of complex matrix inversion was obtained. Thus real and imaginary part of complex matrix inversion was computed. This process is shown in Figure 11. This is processed according to a particular algorithm

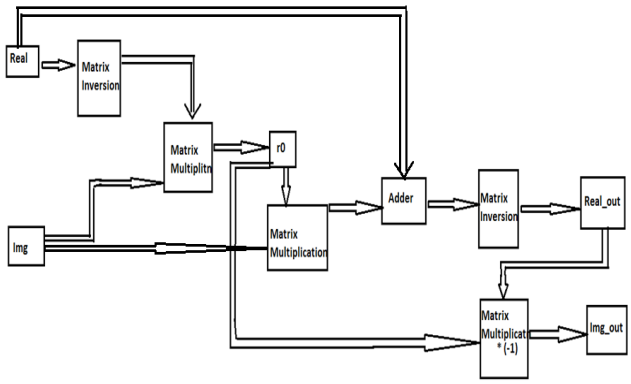

Figure 11. Complex matrix inversion block.

which helps to find the complex matrix inversion from normal matrix inversion. This inverse correlation matrix, R^{-1} is given as input to adaptive weight computation processor.

4.3.3 *Sub Architecture III- Adaptive Weight Computation Engine*

In adaptive weight computation processor, the correlation matrix inverse, R^{-1} stored in DPRAM is fetched to the controller. Along with that, a steering vector is also given in accordance with the MVDR algorithm. The objective of MVDR algorithm is to minimize the power and allow the beam to pass in the desired direction without distortion by keeping the product of adaptive weight transpose and steering vector in unity gain. This calculated complex adaptive apodization weight is stored in DPRAM for finding the beamformer output and its power. It's shown in Figure 12.

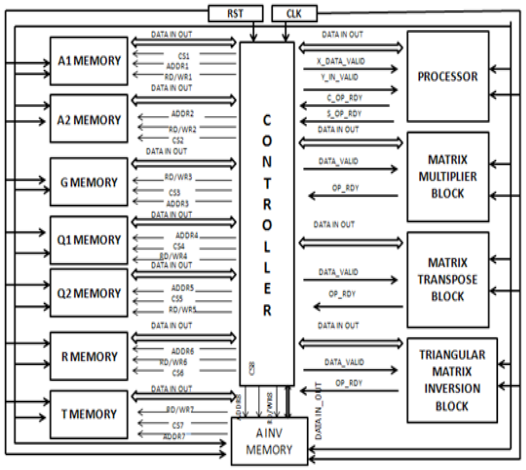

Figure 10. Matrix inversion block.

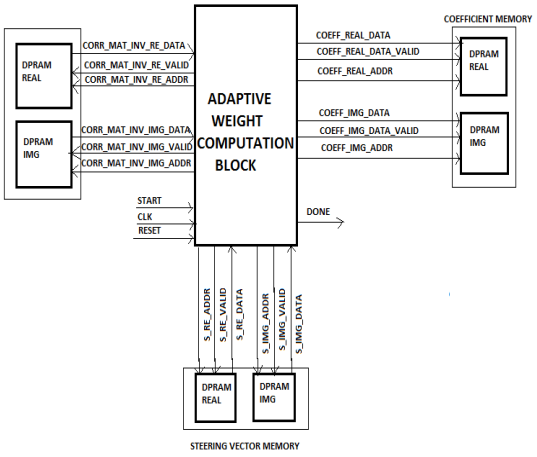

Figure 12. Adaptive weight computation processor.

4.3.4 *Sub Architecture IV-Beamformer Output Processor*

Figure 13 shows the beamformer output sub architecture. Thus finally the beamformer output is found and stored in memory. For comparing it with MATLAB output, the output power is taken and normalized. The data representation is shown in Table 1.

5. Results and Discussion

In conventional delay and sum beamformer, the input is given with a time pulse. The delay of each wave is calculated first, then delay of all signals are made equal, by applying new delays to each signal, so that the resultant will be a beamformed output. This is shown in following graph, Figure 14. In first case, signals are given normally and the corresponding output is shown. In second case, delay of each signal was calculated and applied it to each signal, such that the result of it super imposes the former one and the final output after summation gives beamformed output.

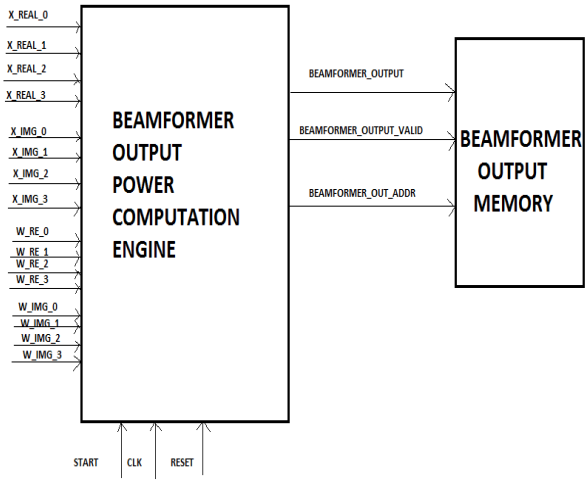

Figure 13. Beamformer output.

Table 1. Data representation

Processor	Input	Output
FFT Processor	1.15	1.15
Bin Processor	8.8	16.16
Adaptive weight computation Processor	16.16	16.16
Beamformer Output memory	16.16	16.16

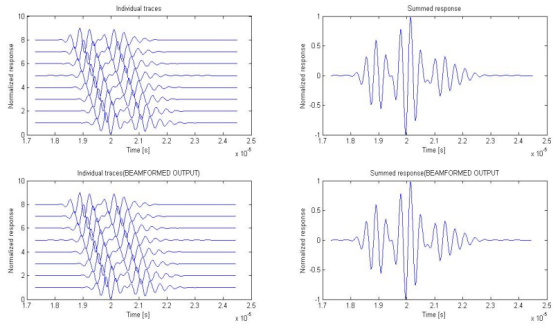

Figure 14. Comparison with summed response.

Figure 15, describes the array distribution. Here the ultrasound beams are steered in both directions which is indicated as red color. In this graph, 32 sensors are taken in to consideration. From graph it is clear that the output beam and the steered beam are in phase, which is indicated by dark red circles in the graph. An array of blue colour circles is seen in the graph, which represents the delay of signal from each sensor. From graph it can see that, delay is more at the centre. Reflected beam has more delayed signal at the centre.

In the graph, Figure 16, the look direction is taken as 0 degree. Since its seen in the previous graph, that

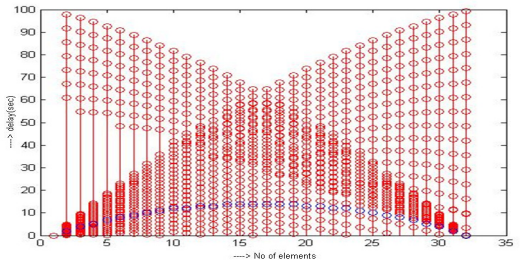

Figure 15. Delay estimation graph.

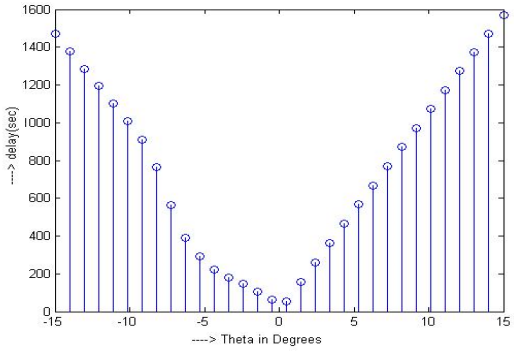

Figure 16. Delay estimation graph at look direction 0 degree.

the reflected beam has more delay at the centre, so while applying delay to signals, only minimum delay is needed to be applied at the centre, so that when sum of all signals are taken, beamformed output is resulted. Hence only minimum delay is applied at the centre, which is clear from graph. Thus delay estimation is done here.

In the Figure 17, direct beamformed output by fieldII, ultrasound image scanner and beamformed output using MVDR algorithm is compared with the help of phantom. Phantom is a body mimicking system, which acts exactly like our body. Here flat image is got, because here only four sensors are taken in to consideration for the simplicity of project work. From Figure 17, it is clear that the beamformed output obtained by applying MVDR algorithm gives more resolution.

The output of FFT processor is shown in the simulation result shown, Figure 18. The output values are seen in xk_re and xk_im with corresponding address, where real values are stored in xk_re and imaginary values are stored in xk_im. When write enable, wea goes high, the values in xk_re and xk_im are written to Dual Port RAM with the increment of their addresses. The values are valid when data valid of FFT IP core goes high. Also it is seen in simulation result that the fft_out is valid, which indicates that the controller is ready for next operation. Here the values are 16 bit and 256 point FFT is taken. The input is given to FFT IP core in 1.15 formats. From the FFT out only 127 samples out of 256 samples is taken, since after 127th sample it will be simply the repetition from the start. FFT IP core takes in samples when ready for data signal (rfd) goes high and gives out valid samples when data valid (dv) goes high. Here all four channel outputs of FFT processor are shown.

FFT processor output is stored in different register banks as real and imaginary values. Also conjugate of imaginary values is stored in another register for calculating correlation matrix, R, since $R = E\{X.X^T\}$. These values are compared with the values loaded from the MATLAB as coe file to the DPRAM. Only first channel values in memory are shown here. From Figure it's clear that the output values got here is same as that got in MATLAB.

Real and imaginary values of correlation matrix are stored in different memories as shown in the simulation result shown, Figure 19. Each value from each sensor is taken at a time to make a 4×4 matrix. Corresponding to each address a 4×4 matrix is formed. So each register of correlation matrix will have 16 values at a time. This is done till 128th address is reached in DPRAM. Then the enable signal, start signal and write signal will become low. Finally from this processor a ready signal is sent to next engine.

Inverse correlation matrix is obtained by performing the complex matrix inversion algorithm. This is the most challenging part of MVDR algorithm. For finding matrix inversion input is converted to 8.8 formats. Then this is applied to adaptive weight computation engine to compute the adaptive weight and finally taking hermitian transpose of it and multiplying with input vector gives the beamformer output. This beamformer output is compared

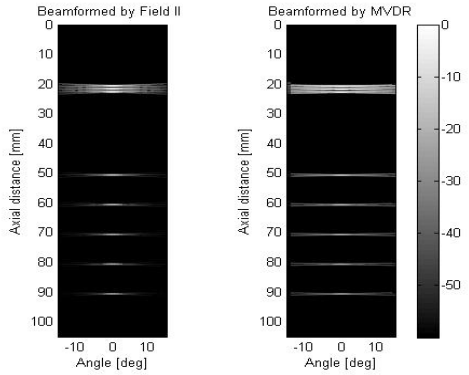

Figure 17. Beamformed output in FieldII: comparison.

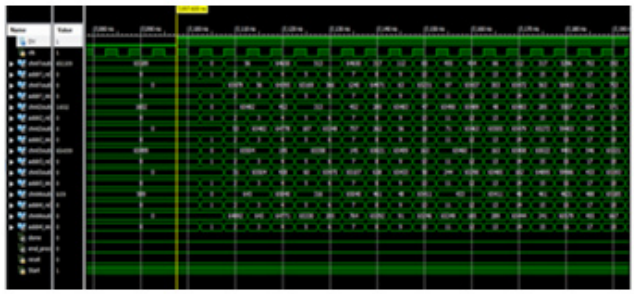

Figure 18. FFT processor output.

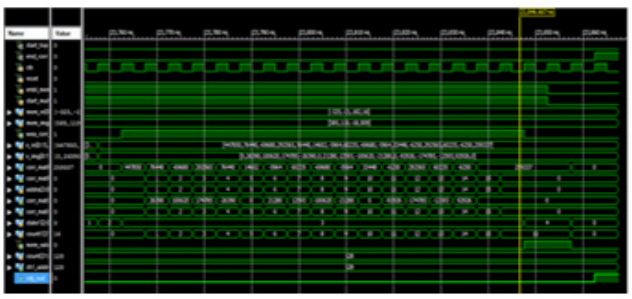

Figure 19. Correlation matrix.

with the golden reference model obtained in MATLAB simulation.

From graph, Figure 20, it is clear that, MVDR output is narrower and gives maximum resolution when compared to delay sum beamformer. While applying MVDR, a pencil like beam is obtained with reduced mainlobe width and reduced sidelobe level. Figure 21 shows the nirmalized beam profile of MVDR beamformer in which the look direction is taken at 40 degree.

5.1 Timing Constraints

Minimum period = 5.503ns
Maximum Frequency = 181.719MHz
Minimum input arrival time before clock = 1.107ns
Maximum output required time after clock = 1.939ns
Maximum combinational path delay = No path found

Table 2 shows the device utilization summary.

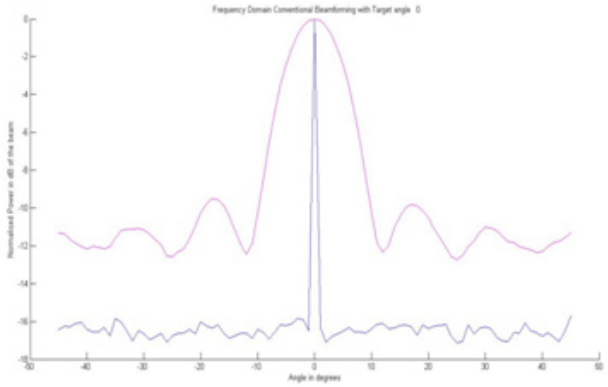

Figure 20. MVDR beamformer output Vs Delay and sum beamformer output.

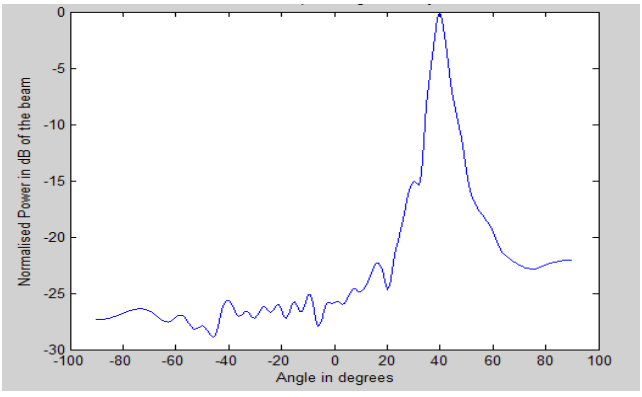

Figure 21. Normalized power Vs angle at look direction 40 degree.

Table 2. Device utilization summary

Logic Utilization	Used	Available	Utilization
No. of slice registers	12449	301440	4%
No. of slice LUTs	11080	150720	7%
No. of fully used LUT-FFT pairs	9032	14497	62%
No. of bonded IOBs	4	600	0%
No. of Block RAM/ FIFO	27	416	6%
No. of BUFG/ BUFGCTRLs	16	32	50%
No. of DSP48E1s	132	768	17%

6. Conclusion

Initially MVDR algorithm is applied in MATLAB design and compared the results obtained for delay and sum beamformer and MVDR beamformer. FPGA implementation is done by studying various architectural designs. QR decomposition is used for finding matrix inversion. Both beamformer outputs were compared. By applying MVDR algorithm a pencil like beam is obtained as beamformer output with reduced main lobe width and reduced side lobe level. Future work includes, by upgrading the number of elements from 4 to 64, this can be made real time applicable algorithm. For upgraded version of the proposed work, DCD algorithm can be preferred for complex matrix inversion, as it is multiplication and division free algorithm.

7. Acknowledgement

I would like to express my heart filled gratitude to Dr. V. S. Kanchana Bhaskaran, Dean of SENSE department and Dr. A. Ravi Shankar, Program Chair of VLSI Design for granting me the permission to do Internship at NIELIT as my final project and the continuous support they have given all the way through. I would like to bring on record the NIELIT for giving me an opportunity to pursue my project as a part of Internship in R & D.

8. References

1. Ranganathan K, Walker WF. A novel beamformer design method for medical ultrasound. Part I: Theory. IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control. 2003 Jan; 50(1):15–24.

2. Hu CH, Zhang L, Cannata JM, Yen J, Shung K. Development of a 64 channel ultrasonic high frequency linear array imaging system. Elsevier. 2011 May; 51(8):953–9.
3. Synnevåg J-F, Austeng A, Holm S. Adaptive beamforming applied to medical ultrasound imaging. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control. 2007 Aug; 54(8):1606–13.
4. Synnevåg J-F, Austeng A, Holm S. Minimum variance adaptive beamforming applied to medical ultrasound imaging. Proc IEEE Ultrasonics Symposium; 2005 Sep 18-21; 2:1199–202.
5. Vorobyov SA. Principles of minimum variance robust adaptive beamforming design. Department of Electrical and Computer Engineering, University of Alberta, Alberta, Canada; Elsevier 2013 Dec; 93(12):3264–77.
6. Holfort IK, Gran F, Jensen JA. Broadband minimum variance beamforming for ultrasound imaging. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control. 2009 Feb; 56(2):314–25.
7. Synnevåg J-F, Austeng A, Holm S. High frame-rate and high resolution medical imaging using adaptive beamforming. Proc IEEE Ultrasonics Symposium; 2006 Oct. p. 2164–67.
8. Synnevåg J-F, Austeng A, Holm S. Benefits of minimum variance beamforming in medical ultrasound imaging. IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control. 2009 Sep; 56(9):1868–79.
9. Chisty NA. Matrix inversion using qr decomposition by parabolic synthesis [Master's thesis]. Lund, Sweden; Department of Electrical and Information Technology Faculty of Engineering, LTH, Lund University 2012. p. 3–4.
10. Walke R, Lam J, McAllister J. FPGA acceleration of an adaptive beamformer within GEDAE. Copyright QinetiQ Ltd; 2003. p. 3–4.