# IMPLEMENTATION OF A PROPOSED 2D DOA ESTIMATION ALGORITHM ON AN FPGA PLATFORM

[1]Sachin Khedekar, [2]Rachna Kumari, [3]Sitakanta Maharatha, [4]Dr. Mainak Mukhopadhyay
[1,2,3]Research Scholars,Birla Institute of Technology Mesra Deogarh Campus, Jharkhand, India

HOD, ECE Dept., Research Scholars,Birla Institute of Technology Mesra Deogarh Campus, Jharkhand, India
[1]Sac_khedekar@rediffmail.com

***Abstract*— Direction of arrival estimation is an important signal parameter in smart antenna which can be used for source localization or source tracking by determining the desired signal location. A new type of estimation algorithm is proposed for the 2-D azimuth and elevation angle estimation problem. FPGA implementation of algorithm can be done using HDL coder of MATLAB.**

***Index Terms*— DOA, GPS, FPGA**

## I. OVERVIEW OF DOA ESTIMATION ALGORITHMS

Smart antenna is one of the dynamic research areas in wireless communication systems. The demand for smart antenna increases drastically when dealing with multiuser communication system, which needs to be adaptive, especially in time varying scenarios. Direction of Arrival (DOA) estimation is considered as an important task in smart antennas. It is an important signal parameter which can be used for source localization or source tracking by determining the desired signal location.

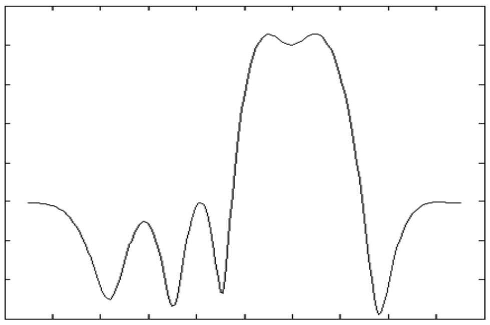
Also, it plays a key role in enhancing the performance of adaptive antenna arrays for wireless communication system and other numerous applications in the field of radar and sonar. Therefore, research has been accomplished about DOA estimation during last recent decades. Various DOA estimation methods have been proposed. These methods differ in technique, speed, computational complexity, accuracy and their dependency on the array structure. Different methods have been suggested to enhance the performance of available algorithms including the increase in the accuracy and resolution of DOA estimation algorithms.

According to the underlying methodology, the array signal processing algorithms can be categorized into two classes. The first class is called the non-parametric approach in which the source locations (or the DOA) are estimated by choosing the strongest output power of a spatial filter after sweeping over the space of interest. The advantage of this approach is that no assumption has to be made on the studied signal. The second class is called the parametric approach, in which a nominated model is assumed for the array observations. Once the model is determined, the quantities of interest in the array problem can be determined by choosing the best parameters that fit the model under some optimality criteria.

In general, the direction-of-arrival (DOA) estimation techniques can be broadly classified into conventional beamforming techniques, subspace-based techniques, and maximum likelihood techniques.

### *A. Classical beamforming*

The conventional beamformer works described earlier on the premise that pointing the strongest beam in a particular direction yields the peak power arriving in that direction. In other words, all the degrees of freedom available to the array were used in forming a beam in the required look direction. This works well when there is only one incoming signal present. Two peaks can be seen at $10^0$ and $30^0$, but the $30^0$ peak is not obvious and are somewhat averaged with the peak at $10^0$. In other words, the spread of each peak is large, and if two impinging angles are close to each other, the two peaks may be "blurred" into one pair. In a more general term, although it is simple to implement, the width of the beam associated with a peak and the height of the side lobes, as seen in figure are relatively large; they limit the method's effectiveness when signals arriving from multiple directions and/or sources are present. This



-100 -80 -60 -40 -20 0 20 40 60 80 100

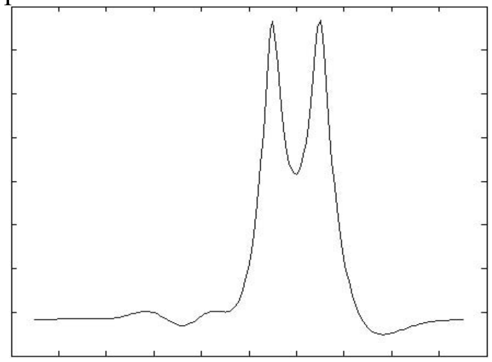
Angle in degree

Fig 1: DOA estimation with Classical Beamformer

technique has poor resolution. Although it is possible to increase the resolution by adding more array elements, it leads to the increase in the numbers of receivers and the amount of storage required for the data.

*B. Capon's Beamformer:*

The conventional beamformer works well when there is only one incoming signal present. But when there is more than one signal present, the array output power contains signal contributions from the desired angle as well as from the undesired angles. Capon's method overcomes this problem by using the degrees of freedom to form a beam in the look direction and at the same time the nulls in other directions in order to reject other signals. In terms of the array output power, forming nulls in the directions from which other signals arrive can be accomplished by constraining a beam (or at least maintaining unity gain) in the look direction. Thus, for a particular look direction, Capon's method uses all but one of the degrees of the freedom to minimize the array output power while using the remaining degrees of freedom to constrain the gain in the look direction to be unity and at the same time the nulls in other directions in order to reject other signals. In terms of the array output power, forming nulls in the directions from which other signals arrive can be accomplished by constraining a beam (or at least maintaining unity gain) in the look direction. Thus, for a particular look direction, Capon's method uses all but one of the degrees of the freedom to minimize the array output power while using the remaining degrees of freedom to constrain the gain in the look direction to be unity. It can be seen that in comparison with figure, the peaks at 10° and 30° are much sharper and better separated compared to that of the conventional beamformer. The side peaks or lobes at other angles are also reduced, making them less likely to confuse the interpretation of the output power. The best resolution achieved was 10°. However, this increased resolution comes at the cost of increased computing time or power.



-100 -80 -60 -40 -20 0 20 40 60 80 100

Angle in degree

Fig 2: DOA estimation with Capon's Beamformer

*C. Maximum Likelihood Techniques*

Maximum likelihood (ML) techniques were some of the first techniques investigated for DOA estimation. Since ML techniques were computationally intensive, they are less.

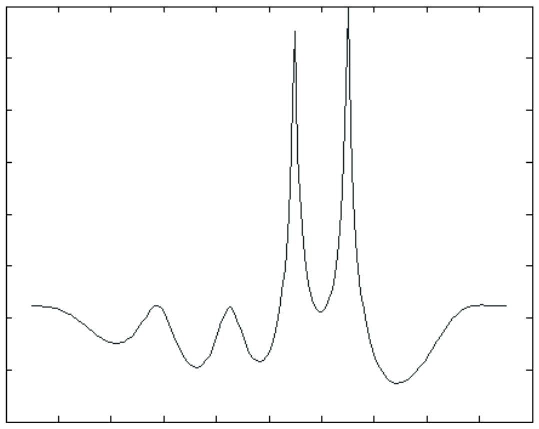


Fig. 3: DOA estimation with the linear prediction; the signal impinges at 10° and $30^0$

popular than other techniques. However, in terms of performance, they are superior to other estimators, especially at low SNR

*D. MUSIC*

MUSIC (Multiple Signal Classification) is one of the earliest proposed and a very popular method for super-resolution direction finding. The DOAs of the multiple incident signals can be estimated by locating the peaks. The *d* largest peaks in the MUSIC spectrum above correspond to the DOAs of the signals impinging on the array.

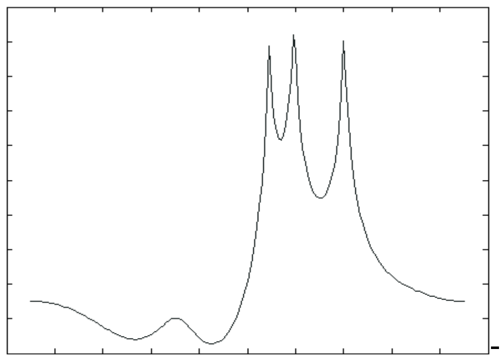


Fig. 4: DOA estimation with MUSIC; the radio signals impinges at 10°, 20°, and 40°.

*E. ESPRIT*

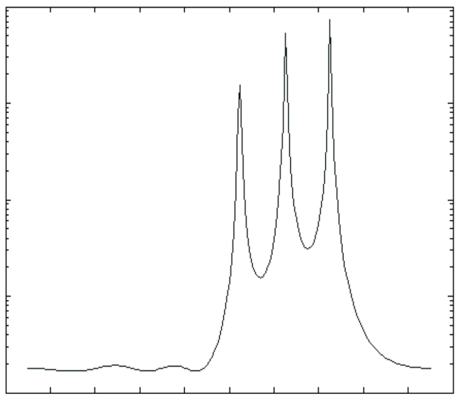


Fig 5: DOA estimation with MUSIC; the radio signals impinges at 10°, 20°, and 40°.

Due to its simplicity and high resolution capability, ESPRIT has become one of the most popular signals subspace-based DOA estimating schemes. ESPRIT is applicable to array geometries that are composed of two identical sub arrays and is restricted to use with array geometries that exhibit invariances. This requirement, however, is not very prohibitive in practical applications since many of the common array geometries used in practice exhibit these invariances. There are three primary steps in any ESPRIT based DOA estimation algorithm:

1. Signal subspace estimation: Computation of a basis matrix for the estimated signal subspace.
2. Solution of the invariance equation: Solution of an (in general) over determined system of equations, the invariance equation, derived from the basis matrix.
3. DOA estimation: Computation of the eigenvalues of the solution of the invariance equation formed in step 2.

## II. PROPOSED METHOD OF DIRECTION OF ARRIVAL ESTIMATION BY ROTATING ADAPTIVE ARRAY ANTENNA PLANE MECHANICALLY

A new technique for 2 D DOA estimation of signals impinging on the array, using mechanical rotation of the array plane by small angle (Azimuth & Elevation) has been proposed for further analysis and discussion.

For an adaptive antenna system, if p users transmit signals from different locations, and each user's signal arrives at the array through multiple paths.
Let LMi denote the number of multipath components of i-th user. We have $\sum_{i=1}^{p} LM_i = p$.
Let's further assume that all of the multi path components for a particular user arrive within a time window which is much less than the channel symbol period for that user, then the input data vector could be expressed as-

$$x(\mathrm{t}) = \sum_{i=1}^{p}\sum_{k=1}^{LMi} \alpha_{i,k} a(\theta_{i,k}) s_i(\mathrm{t}) + n(\mathrm{t})$$
(1)

Or we can write $$x(\mathrm{t}) = \sum_{i=1}^{p} G_i s_i(t) + \mathrm{n(t)}$$
(2)

where $\theta_{i;k}$ is the DOA of the k-th multi path component for the i-th user, $a(\theta_{i;k})$ is the steering vector corresponding to $\theta_{i;k}$, $\alpha_{i;k}$ is the complex amplitude of the k-th multipath component for the i-th user, and $G_i$ is the spatial signature for the i-th user and is given by

$$G_i = \sum_{k=1}^{LMi} \alpha_{i,k} a(\theta_{i,k})$$
(3)

The signal component arriving on *n*th antenna element at a particular instance of time is given by

$$X_n = A\exp(\mathrm{j}2\pi\,\mathrm{ndsin}\,\theta\cos\phi/\lambda)$$
(4)
$$Y_n = A\exp(\mathrm{j}2\pi\,\mathrm{ndsin}\,\theta\sin\phi/\lambda)$$
(5)

Where A= complex amplitude of the signal, φ = Direction of Arrival (DOA) of the signal (Azimuth Angle) (unknown), θ = Direction of Arrival (DOA) of the signal (Elevation Angle) (unknown), d= spacing between antenna elements and λ = wavelength.
Now one can view (4) & (5) as-

$$X_n = A\exp[\mathrm{j}2\pi f(\mathrm{dsin}\,\theta\cos\phi/\mathrm{c})]$$
(6)
$$Y_n = A\exp[\mathrm{j}2\pi f(\mathrm{dsin}\,\theta\sin\phi/\mathrm{c})]$$
(7)

Where f= frequency of the signal and c= velocity of wave.
Now if we mechanically steer the antenna plane by δφ & δθ, then (6) & (7) becomes –

$$X^1{}_n = A\exp[\mathrm{j}2\pi f(\mathrm{dsin}\,\theta\cos(\phi+\delta\phi)/\mathrm{c})]$$
(8)
$$Y^1{}_n = A\exp[\mathrm{j}2\pi f(\mathrm{dsin}\,\theta\sin(\phi+\delta\phi)/\mathrm{c})]$$
(9)
$$X^2{}_n = A\exp[\mathrm{j}2\pi f(\mathrm{dsin}(\theta+\delta\theta)\cos\phi/\mathrm{c})]$$
(10)
$$Y^2{}_n = A\exp[\mathrm{j}2\pi f(\mathrm{dsin}(\theta+\delta\theta)\sin\phi/\mathrm{c})]$$
(11)

Now taking the frequencies (which can be known by seeing the spectra of the signal) of the signal from (6) and (8), and taking their ratio one could get-

$$\frac{frequency \to X_n}{frequency \to X^1{}_n} = \frac{\cos\phi}{\cos(\phi+\delta\varphi)} = \frac{1}{k}$$ (k is known)

Hence $$\phi = \tan^{-1}[\frac{\cos\delta\phi - k}{\sin\delta\phi}]$$
(12)

And from (7) & (11), we could get

$$\frac{frequency \to Y_n}{frequency \to Y^2{}_n} = \frac{\sin\theta}{\sin(\theta+\delta\theta)} = \frac{1}{k}$$

$$\theta = \cot^{-1}[\frac{k - \sin\theta}{\cos\delta\theta}]$$
(13)

Now using the simple relation given in (12) & (13) one can determine the unknown DOA (θ & φ) of all incoming signal impinging on the array with suitable algorithm based on (6), (7), (8), (9), (10), (11), (12) and (13).

## III. FPGA Implementation Procedure

Basically, an FPGA is a large-scale integrated circuit containing programmable logic blocks, programmable interconnect and programmable input-output blocks. The programmable logic blocks can be programmed to duplicate the functionality of basic logic gates such as AND, OR, XOR, NOT or more complex combinatorial functions such as flip-flops, memory elements, decoders or simple mathematical functions. The programmable input-output blocks at the periphery of the devices provide programmable input and output capabilities. By programming the hierarchy of programmable interconnects, the programmable logic blocks and programmable input-output blocks can be interconnected to perform whatever logical functions and input-output connections are required. During the past decade, FPGAs have experienced extensive architecture innovations. Many advanced technologies have been applied to FPGA devices that enable the development of higher density and much more powerful devices. Now most FPGA devices also have block RAMs, hardware multipliers and embedded microprocessors besides traditional logic blocks and interconnects. Therefore FPGA devices become extremely well suited to the high-performance real-time signal processing. Defining the behavior of an FPGA chip can be done using a Hardware Description Language (HDL) such as VHDL and Verilog to describe the functions directly. The handwritten code can be guaranteed as optimal by the designer in the sense that one can be sure what is got as an output. However, the optimality of the design is highly related to the experience of the designer which makes the HDL design method difficult for inexperienced designers. Alternatively, defining the behavior of an FPGA can be done using a schematic based design tool, such as the System Generator we mentioned above. The System Generator provides blocks of pre-defined functions, which can be arranged through a graphical user interface. Therefore, the System Generator is easy for designers, especially for persons unexperienced with HDL design method. After defining the behavior using either the HDL method or the schematic method, a technology-mapped net list is generated using an electronic design automation tool. The net list can then be fitted to the actual FPGA architecture using a process called place and route, usually performed by the FPGA Company's proprietary place-and-route software. The user will validate the map, place and route results via timing analysis, simulation, and other verification methodologies. Once the design and validation process is complete, the binary file can be generated (also using the FPGA company's proprietary software) and downloaded to (re)configure the FPGA device. To simplify the design of complex systems in FPGAs, there exist libraries of predefined complex functions and circuits that have been tested and optimized to speed up the design process. These predefined circuits are commonly called IP cores, such as the CORDIC cores and are available from FPGA vendors and third-party IP suppliers. The FPGA device vendors also provide related software to support their chips, such as the Xilinx Integrated Software Environment (ISE). With assistance of these software tools and IP cores, FPGA design is simpler now.

### A. MATLAB to FPGA using HDL Coder (TM):

FPGAs provide a good compromise between general purpose processors (GPPs) and application specific integrated circuits (ASICs). GPPs are fully programmable but are less efficient in terms of power and performance; ASICs implement dedicated functionality and show the best power and performance characteristics, but require extremely expensive design validation and implementation cycles. FPGAs are also used for prototyping in ASIC workflows for hardware verification and early software development.

Due to the order of magnitude performance improvement when running high-throughput, high-performance applications, algorithm designers are increasingly using FPGAs to prototype and validate their innovations instead of using traditional processors. However, many of the algorithms are implemented in MATLAB due to the simple-to-use programming model and rich analysis and visualization capabilities. When targeting FPGAs or ASICs, these MATLAB algorithms have to be manually translated to HDL.

For many algorithm developers who are well-versed with software programming paradigms, mastering the FPGA design workflow is a challenge. Unlike software algorithm development, hardware development requires them to ***think parallel***. Other obstacles include: learning the VHDL or Verilog language, mastering IDEs from FPGA vendors, and understanding esoteric terms like "multi-cycle path" and "delay balancing".

We will see how we can automatically generate HDL code from MATLAB algorithm, implement the HDL code on an FPGA, and use MATLAB to verify HDL code.

### B. MATLAB to Hardware Workflow

The process of translating MATLAB designs to hardware consists of the following steps:

-Model algorithm in MATLAB - use MATLAB to simulate, debug, and iteratively test and optimize the design.

-Generate HDL code - automatically create HDL code for FPGA prototyping.
-Verify HDL code - reuse our MATLAB test bench to verify the generated HDL code.
-Create and verify FPGA prototype - implement and verify design on FPGAs. There are some unique challenges in translating MATLAB to hardware. MATLAB code is procedural and can be highly abstract; it can use floating-point data and has no notion of time. Complex loops can be inferred from matrix operations and toolbox functions. Implementing MATLAB code in hardware involves:
-Converting floating-point MATLAB code to fixed-point MATLAB code with optimized bit widths suitable for efficient hardware generation.
-Identifying and mapping procedural constructs to concurrent area- and speed-optimized hardware operations.
-Introducing the concept of time by adding clocks and clock rates to schedule the operations in hardware.
-Creating resource-shared architectures to implement expensive operators like multipliers and for-loop bodies.
-Mapping large persistent arrays to block RAM in hardware
HDL Coder™ simplifies the above tasks though workflow automation.
Let's look at each workflow step in detail.

### 1) *Fixed-Point Conversion*

Signal processing applications are typically implemented using floating-point operations in MATLAB. However, for power, cost, and performance reasons, these algorithms need to be converted to use fixed-point operations when targeting hardware. Fixed-point conversion can be very challenging and time-consuming, typically demanding 25 to 50 percent of the total design and implementation time. The automatic floating-point to fixed-point conversion workflow in HDL Coder™ can greatly simplify and accelerate this conversion process. The floating-point to fixed-point conversion workflow consists of the following steps:
-Verify that the floating-point design is compatible with code generation.
-Propose fixed-point types based on computed ranges, either through the simulation of the testbench or through static analysis that propagates design ranges to compute derived ranges for all the variables.
-Generate fixed-point MATLAB code by applying proposed fixed-point types.
-Verify the generated fixed-point code and compare the numerical accuracy of the generated fixed-point code with the original floating point code. Note that this step is optional. We can skip this step if MATLAB design is already implemented in fixed-point.

### 2) *HDL Code Generation*

The HDL Code Generation step generates HDL code from the fixed-point MATLAB code. We can generate either VHDL or Verilog code that implements MATLAB design. In addition to generating synthesizable HDL code, HDL Coder™ also generates various reports, including a traceability report that helps we navigate between MATLAB code and the generated HDL code, and a resource utilization report that shows, at the algorithm level, approximately what hardware resources are needed to implement the design, in terms of adders, multipliers, and RAMs. During code generation, we can specify various optimization options to explore the design space without having to modify our algorithm. In the Design Space Exploration and Optimization Options section below, we can see how we can modify code generation options and optimize your design for speed or area.

### 3) *HDL Verification*

Standalone HDL test bench generation:
HDL Coder™ generates VHDL and Verilog test benches from MATLAB scripts for rapid verification of generated HDL code. We can customize an HDL test bench using a variety of options that apply stimuli to the HDL code. We can also generate script files to automate the process of compiling and simulating your code in HDL simulators. These steps help to ensure the results of MATLAB simulation match the results of HDL simulation. HDL Coder™ also works with HDL Verifier to automatically generate two types of cosimulation testbenches:
-HDL cosimulation-based verification works with Mentor Graphics® ModelSim® and QuestaSim®, where MATLAB and HDL simulation happen in lockstep.
-FPGA-in-the-Loop simulation allows running a MATLAB simulation with an FPGA board in

strict synchronization. We can use MATLAB to feed real world data into our design on the FPGA, and ensure that the algorithm will behave as expected when implemented in hardware.

*4) HDL Synthesis*

Apart from the language-related challenges, programming for FPGAs requires the use of complex EDA tools. Generating a bit stream from the HDL design and programming the FPGA can be daunting tasks. HDL Coder™ provides automation here, by creating project files for Xilinx® and Altera® that are configured with the generated HDL code. We can use the workflow steps to synthesize the HDL code within the MATLAB environment, see the results of synthesis, and iterate on the MATLAB design to improve synthesis results.

*a) Design Space Exploration and Optimization Options*

HDL Coder™ provides the following optimizations to help we explore the design space trade-offs between area and speed. We can use these options to explore various architectures and trade-offs without having to manually rewrite our algorithm.

*b) Speed Optimizations*

Pipelining: To improve the design's clock frequency, HDL Coder enables us to insert pipeline registers in various locations within our design. For example, we can insert registers at the design inputs and outputs, and also at the output of a given MATLAB variable in our algorithm.

Distributed Pipelining: HDL Coder also provides an optimization based on retiming to automatically move pipeline registers, we have inserted to maximize clock frequency, by minimizing the delay through combinational paths in our design.

Area Optimizations:

RAM mapping: HDL Coder™ maps matrices to wires or registers in hardware. If persistent matrix variables are mapped to registers, they can take up a large amount of FPGA area. HDL Coder™ automatically maps persistent matrices to block RAM to improve area efficiency. The challenge in mapping MATLAB matrices to block RAM is that block RAM in hardware typically has a limited set of read and writes ports. HDL Coder™ solves this problem by automatically partitioning and scheduling the matrix reads and writes to honor the block RAM's port constraints, while still honoring the other control- and data-dependencies in the design.

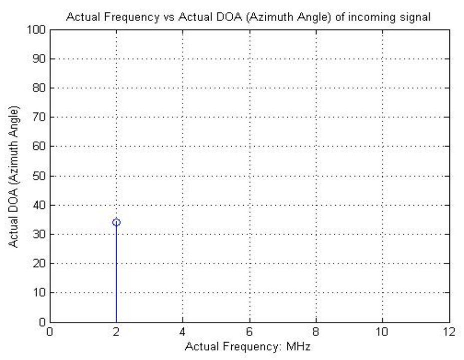
Resource sharing: This optimization identifies functionally equivalent multiplier operations in MATLAB code and shares them. We can control the amount of multiplier sharing in the design.

Loop streaming: A MATLAB for-loop creates a FOR_GENERATE loop in VHDL. The body of the loop is replicated as many times in hardware as the number of loop iterations. This results in an inefficient use of area. The loop streaming optimization creates a single hardware instance of the loop body that is time-multiplexed across loop iterations.

Constant multiplier optimization: This design level optimization converts constant multipliers into shift and add operations using canonical signed digit (CSD) techniques.

## IV. SIMULATIONS

### *A. Actual & estimated signals DOA and frequencies*



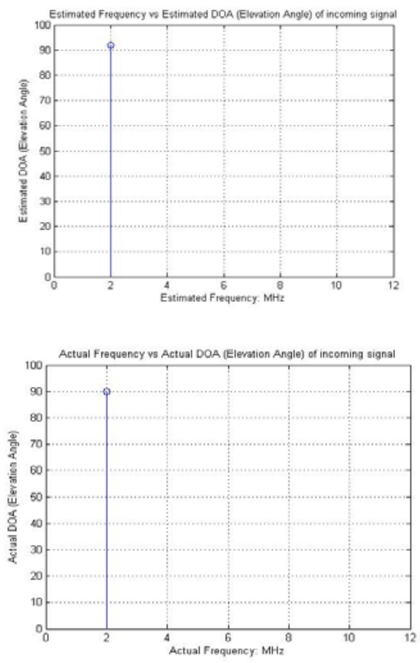
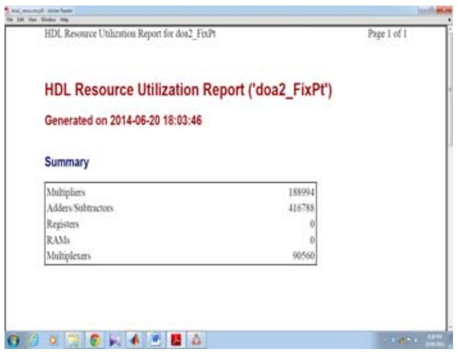


Fig.6-9: Estimated & Actual signals DOA and frequencies

***B.****HDL Resource Utilization Report*



HDL Resource Utilization Report for doa2_FixPt Page 1 of 1

HDL Resource Utilization Report ('doa2_FixPt')

Generated on 2014-06-20 18:03:46

Summary

| | |
|---|---|
| Multipliers | 188994 |
| Adders/Subtractors | 416788 |
| Registers | 0 |
| RAMs | 0 |
| Multiplexers | 90560 |

In these simulations **δφ = 1$^0$.** Estimated Frequencies and Estimated DOAs are not with the same order as signals are sensed by the array, but after estimating the entire signal space, their plots almost identical as exhibited in fig. (6) to (9).

## CONCLUSIONS

FPGA implementation of a proposed 2D DOA estimation algorithm is presented on MATLAB platform. The design employs CORDIC-based processing (array boundary cell) which is well matched to the computational resources of an FPGA. This work points out that spatial processing techniques provide new perspectives in applications related with GPS. The use of 2 D DOA algorithm leads to good solutions where the interfering and multipath signals need to be canceled. Others scenarios, that made a better representation of GPS problem will be established in order to test the structure. Future studies will work on in the way of have DOA estimators with lower computational burden with 3 dimensional geometry. Also the System Generator programming environment enables the rapid development of heterogeneous systems (processors and FPGAs) while insulating programmers from the frequently complex and error prone programming associated with hardware software partitions. The results indicate successful real time implementation of the proposed and the existing methods.

## REFERENCES

[1] Parkinson, B. W., Spilker, J., *Global Positioning System: Theory and Applications*, vol. 1&2, AIAA, 1996.

[2] Zhizhang Chen, Gopal Gokeda, Yiqiang Yu, *Introduction to Direction of Arrival Estimation*, Artech House pub.

[3] Kaplan, E. D., *Understanding GPS Principles and Application*, Arthech House, 1996.

[4] Compton, R. T., *Adaptive Antennas: Concepts and Performance*", Prentice Hall, Englewood Cliffs, NJ, 1988.

[5] Junqueira, C., Ribeiro, M., Romano, J.M.T. *Adaptive Techniques for GPS Systems Enhancement*, 13Th International Technical Meeting of The Satellite Division of the Institute of Navigation set. 2000.

[6] Balanis, *Antenna Theory: Analysis & design*, Wiley publication

[7] T. K. Sarkar, M. C. Wicks, M. Salazar-Palma, R. J. Bonneu, *Smart Antennas*, John Wiley & Sons.

[8] Harry L. Van Trees, *Optimum Array Processing*, John Wiley & Sons.

[9] B. Widrow, P.E. Mantey, L. J. Griffiths, B. B. Goode, *Adaptive Antenna Systems*, IEEE Proc. Vol. 55, No.12, pp. 2143 – 2159, December 1967.

[10] Zhang, Y., Z. Ye, X. Xu, and J. Cui, "Estimation of two-dimensional direction-of-arrival for uncorrelated and coherent signals with low complexity", IET Radar, Sonar & Navigation, Vol. 4, No. 4, 507-509, 2010.

[11] Ichige, K. and H. Arai, "Implementation of FPGA based fast DOA estimator using unitary MUSIC algorithm [cellular wireless base station applications]," IEEE 58th Vehicular Technology Conference, VTC2003-Fall, Vol. 1, 213-217, 2003.
[12] Ichige, K. and H. Arai, "Real-time smart antenna system incorporating FPGA-based fast DOA estimator," IEEE 60th Vehicular Technology Conference, VTC2004-Fall, Vol. 1, 160-164, 2004.
[13] Wang, H. and M. Glesner, "Hardware implementation of smart antenna systems," Adv. Radio Sci., Vol. 4, 185-188, 2006.
[14] www.mathworks.com

**S. R. Khedekar -** He is a research scholar in Birla Institute of Technology, Mesra (Deogarh campus), Jharkhand, INDIA. He received bachelor in Electronics & Telecommunication engineering in 2000 from NMU &master in Electronics & Telecommunication engineering in 2008 from Shivaji University, Maharashtra. He has more than 11 years of teaching experience. His area of research is signal processing & antenna wave theory.

**Sitakanta Maharatha -** He is a research scholar in Birla Institute of Technology, Mesra (Deogarh campus), Jharkhand, INDIA. He received bachelor in engineering from Utkal University& M.Tech. from KIIT University, Odisha. He has more than 18 years of teaching experience. His area of research is signal processing & VLSI.

**Rachna Kumari -** She is a research scholar in Birla Institute of Technology, Mesra (Deogarh campus), Jharkhand, INDIA. She received M.Tech from Dr. B.C.Roy Engineering College, Durgapur, WB. She has more than 7 years of teaching experience. Her area of interest isccommunication system, data communication, advance electric circuit, switching &pulse theory, computer network, digital communication, basic electronics, telecommunication and switching networks

**Dr. Mainak Mukhopadhyay** – He is working as Head of ECE department in Birla Institute of Technology, Mesra (Deogarh campus), Jharkhand, INDIA.He received PhD in E&ECE from, IIT Kharagpur, M.Tech. in Microwave Engineering from University of Burdwan. He has total teaching, R&D and industrial experience of 11 years. His specialization & areas of interest are digital processing architecture, microwave communication, embedded system & VLSI, control engineering & genetic algorithm.