FPGA Implementation of DOA Estimation Method Based on Polarization Sensitive Array

Lutao Liu
Harbin Engineering University
Ying Cao
Harbin Engineering University
Muran Guo (guomuran@hrbeu.edu.cn)
Harbin Engineering University https://orcid.org/0000-0001-7013-9612

Research

Keywords: DOA estimation, FPGA, MUSIC algorithm, Polarization sensitive array

Posted Date: June 1st, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1701368/v1

RESEARCH

FPGA Implementation of DOA Estimation Method Based on Polarization Sensitive Array

Lutao Liu[1,2]
, Ying Cao[1,2]
and Muran Guo[1,2*]

*Correspondence:
guomuran@hrbeu.edu.cn
[1]The College of Information and Communication Engineering, Harbin Engineering University, Harbin, China
Full list of author information is available at the end of the article

Abstract

In this paper, the Field Programmable Gate Arrays (FPGAs) implementation of multiple signal classification (MUSIC) algorithm based on polarization sensitive array is discussed for the first time. Directly embedding the conventional complex-valued MUSIC in FPGA will waste a lot of hardware resources and involve high computational burden. Thus, we propose a real-valued direction of arrival (DOA) estimation method based on polarization sensitive arrays for FPGA implementation in this paper. The proposed method introduces a linear transformation on the received signal, thus simplifying the subsequent calculations of the polarization MUSIC algorithm. In addition, the whole FPGA implementation scheme using the proposed real-valued MUSIC based on distributed polarization-sensitive arrays is also developed in this paper. The proposed scheme releases the computational burden via exploiting the parallel calculation of the covariance matrix, the parallel Jacobi algorithm of multiple cleaning of eigenvalue decomposition (EVD), the multi-level spectral peak search method, and the pipelining of each module. The multi-level and multi-group cleaning in the EVD module reuse the resources occupied by a group of cleaning, and the spectral peaks are serially searched in the peak search module. Consequently, compared with the complex-valued MUSIC, the hardware resource consumption and computation burden is dramatically reduced in the proposed scheme.

Keywords: DOA estimation; FPGA; MUSIC algorithm; Polarization sensitive array

1.Introduction

Direction of arrival (DOA) estimation is one of the main parts in array signal processing [1], and has gained considerable attentions in many fields, including the spectrum sharing [2] and automotive radar [3, 4]. For scalar arrays, many effective methods have been developed in the last decades [5–7]. However, the capability of scalar array to deal with the complex environment is limited. To overcome the limitations in scalar arrays, the polarization sensitive array, one kind of the vector arrays, has been investigated [8, 9]. The polarization sensitive array element can obtain both the spatial information and polarization information from the incident electromagnetic wave in space according to its polarization sensitive characteristics. Compared with ordinary array, polarization sensitive array has the advantages of high anti-interference ability, clear resolution and strong signal detection ability [10]. Therefore, the polarization sensitive array has a wide range of applications [11–13].

On the other hand, multiple signal classification (MUSIC) algorithm, as a classical DOA estimation algorithm, has the advantages of high precision, high resolution and universal array applicability [14]. The use of MUSIC in polarization sensitive array has been well studied in theory [15]. However, the corresponding hardware implementation scheme has not been investigated before. Therefore, it is of great significance to develop the hardware implementation of the MUSIC algorithm based on polarization sensitive arrays. At present, DOA estimation is mostly embedded in the multi-core Digital Signal Processor (DSP), e.g., TMS320C6678, since many resources are integrated in the DSP and can be directly called. However, tasks are serially processed in DSP, thus limiting the running time of MUSIC algorithm to millisecond level. Such running time usually cannot meet the real-time requirement of DOA estimation. With the rapid development of Field Programmable Gate Array (FPGA) technology, the capacity of FPGA chips is getting larger and larger, and the provided IP libraries are becoming more and more comprehensive. Consequently, the advantages of high FPGA parallelism are gradually revealed, providing an outstanding alternative for the hardware implementation to improve the real-time performance of DOA estimates. Reference [16] proposed an antenna array reduction and covariance normalization technology, where the one-dimensional search is realized on ZYNQ (XC7Z020-CLG484-1) platform. As a result, DOA estimation can be completed in only $2\mu s$ at 100MHz working frequency. However, this technology is only applicable to conventional scalar array. In addition, the corresponding algebraic eigenvalue decomposition (EVD) method is only suitable for the case that the dimension of covariance matrix is less than 5, which is rather limited. Reference [17] migrated the two-dimensional search of MUSIC algorithm to the Xilinx Virtex-6LX130T FPGA, where the serial systolic array structure of Jacobi algorithm and the multi-scale peak search method are used. It takes $1ms$ to complete one DOA estimation at the operating frequency of 100MHz. In reference [18], the classical MUSIC algorithm on the uniform symmetric circular array was implemented and a MUSIC algorithm with complete real-valued calculation was proposed. In order to reduce the resource consumption of parallel Jacobi algorithm, a novel systolic array structure is proposed in [18]. However, it brings obstacles to the real-time performance of DOA estimation, and the running time on hardware is in milliseconds. Aiming at the heavy computational load and time-consuming problem of EVD, reference [19] optimized the Jacobi algorithm and proposed an one-time rotation acceleration method of parallel Jacobi algorithm. Nevertheless, the running time is still more than $100ms$. Reference [20] gave the theoretical analysis of several matrix decomposition algorithms on DOA estimation performance when implemented on FPGA. The conclusion is that the time-consuming of these decomposition methods is at the microseconds level. So far, the research on the FPGA implementation of MUSIC algorithm with high resolution is based on traditional scalar array. The research on the FPGA implementation of MUSIC algorithm based on polarization sensitive array is still open, and the research on improving the real-time performance of DOA estimation is also of great significance.

In this paper, an efficient hardware implementation scheme of 2D MUSIC algorithm based on polarization sensitive arrays is investigated under the FPGA framework. In order to improve the hardware efficiency, a real-valued preprocessing

method is proposed via fully exploiting the characteristics of the receive array and the property of the corresponding steering vector. We consider a distributed polarization sensitive array with all the elements placed as a centrosymmetric uniform circular array in this paper. According to the centrosymmetric structure, a linear transformation matrix is constructed to describe the relationship between the real and imaginary part of the steering vector. The linear transformation matrix is multiplied by the received signal to obtain real-valued received data. By performing real-valued processing, the calculation of the polarization MUSIC algorithm is effectively simplified.

On the other hand, the FPGA implementation scheme mainly consists of 5 modules, i.e., real number preprocessing, covariance matrix calculation, EVD, noise subspace determination, and spectral peak search. The time-consuming of the algorithm is shortened by adopting parallel calculation in the covariance matrix calculation module, parallel Jacobi algorithm with multiple sweeps in the EVD module, and multi-level search in the spectral peak search module. According to the characteristics of independent calculation of each module, the pipeline working mode is designed, which can fully utilize the characteristics of FPGA parallel computing. From the experimental results, the proposed scheme can complete one DOA estimation based on polarization sensitive array in μs level, while the estimation accuracy is well guaranteed.

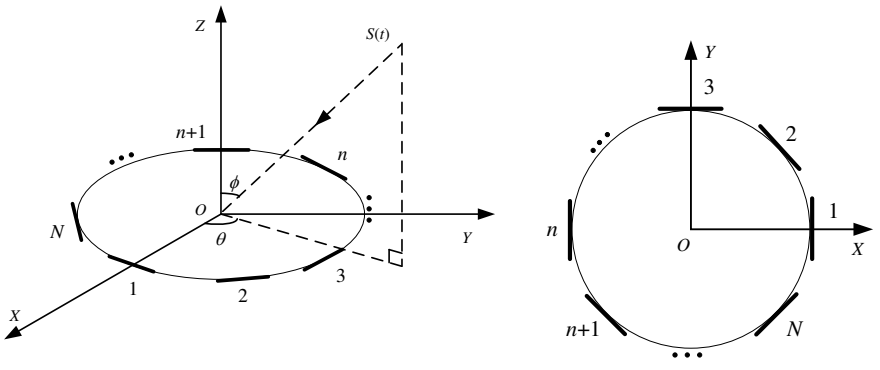
Dominant contributions of this paper are summarized as follows: (1) The FPGA implementation of MUSIC based on polarization sensitive array is discussed in this paper for the first time. Previous relevant research mainly focuses on the scalar array. (2) A real-valued preprocessing is proposed to simplify the subsequent calculations, where the received signal is transformed to real-valued form by considering the centrosymmetric structure of the uniform circular receive array. (3) The whole FPGA implementation scheme is developed, including the design of each module and the connection between modules.

The rest of the paper is organized as follows. In Sect.2, the principle of the 2D polarization MUSIC algorithm is introduced. In Sect.3, the implementation of the 2D polarization MUSIC on FPGA is presented in detail. Sect.4 shows the results of several typical experiments and gives the performance evaluations. Sect.5 concludes this paper.

2.MUSIC algorithm based on Polarization Sensitive Array

The array shown in Figure 1 is a distributed polarization sensitive array consisting of single dipoles, where different dipoles have different polarization modes and directions.

Consider a distributed polarization sensitive array composed of N (N=2K or N=2K+1) dipoles, and assume that M far-field narrowband fully polarization signals impinge on the array, where the azimuth and elevation of the mth signal are θ_m and ϕ_m, and the polarization auxiliary angle and polarization phase are γ_m and η_m, respectively. In addition, assume that the noise is additive Gaussian white noise (AWGN) with zero mean and variance σ^2, and the incident signals are uncorrelated and independent of the noise. Then, the received signal of the polarization sensitive



(a) Array Stereoscopic Schematic (b) Array planar structure

Figure 1 Distributed polarization sensitive array

array is expressed as:

$$\boldsymbol{X}(t) = \sum_{m=1}^{M} \boldsymbol{a}_{\theta_m,\phi_m,\gamma_m,\eta_m} s_m(t) + \boldsymbol{N}(t) = \boldsymbol{A}\boldsymbol{S}(t) + \boldsymbol{N}(t) \tag{1}$$

where $\boldsymbol{A} = [\boldsymbol{a}_{\theta_1,\phi_1,\gamma_1,\eta_1}, \cdots, \boldsymbol{a}_{\theta_M,\phi_M,\gamma_M,\eta_M}]$ is the spatial and polar domain joint steering vector matrix of the signal, $\boldsymbol{S}(t) = [s_1(t), \cdots, s_M(t)]^{\mathrm{T}}$ is the vector of M independent incident sources, and $\boldsymbol{N}(t) = [n_1(t), \cdots, n_N(t)]^{\mathrm{T}}$ is the noise vector.

In equation (1), $\boldsymbol{a}_{\theta_m,\phi_m,\gamma_m,\eta_m}(m = 1, 2, \cdots, M)$ is the steering vector of the mth incident signal. Denote $\boldsymbol{a}_{\theta_m,\phi_m,\gamma_m,\eta_m}$ by $\boldsymbol{a}_{\theta,\phi,\gamma,\eta}$, and the corresponding definition is:

$$\begin{aligned}\boldsymbol{a}_{\theta,\phi,\gamma,\eta} &= \underbrace{\begin{bmatrix} u_1(\theta,\phi) & & & \\ & u_2(\theta,\phi) & & \\ & & ... & \\ & & & u_N(\theta,\phi) \end{bmatrix}}_{\boldsymbol{\Upsilon}_{\theta,\phi}} \boldsymbol{B} \underbrace{\begin{bmatrix} -\sin\theta & \cos\phi\cos\theta \\ \cos\theta & \cos\phi\sin\theta \end{bmatrix}}_{\boldsymbol{\Xi}_{\theta,\phi}} \underbrace{\begin{bmatrix} \cos\gamma \\ \sin\gamma e^{j\eta} \end{bmatrix}}_{\boldsymbol{h}_{\gamma,\eta}} \\ &= \boldsymbol{\Upsilon}_{\theta,\phi}\boldsymbol{\Xi}_{\theta,\phi}\boldsymbol{h}_{\gamma,\eta} = \boldsymbol{D}_{\theta,\phi}\boldsymbol{h}_{\gamma,\eta}\end{aligned} \tag{2}$$

where $\boldsymbol{B} = [\boldsymbol{B}_1, \boldsymbol{B}_2, ..., \boldsymbol{B}_N]^{\mathrm{T}}$ is the polarization sensitive matrix depending on the polarization sensitive array structure and polarization mode of each dipole. For distributed polarization sensitive array, if the pointing angle of the nth array element is α_n, then $\boldsymbol{B}_n = [\cos\alpha_n, \sin\alpha_n]^{\mathrm{T}}$.

For the nth array element, the spatial steering vector $u_n(\theta,\phi)$ has the following form:

$$u_n(\theta,\phi) = \exp\{-j2\pi f\tau_n\} \tag{3}$$

where f is the signal frequency and τ_n is the spatial delay between different array elements. For a planar array, if the origin is taken as the reference point, the specific form is:

$$\tau_n = \frac{1}{c}(x_n \sin\theta\cos\phi + y_n \sin\theta\sin\phi) \tag{4}$$

where $c = 3 \times 10^8$ m/s is the speed of light and (x_n, y_n) is the planar coordinate of the nth array element.

The covariance matrix of the received signals is:

$$\boldsymbol{R} = \mathrm{E}\left[\boldsymbol{X}\boldsymbol{X}^{\mathrm{H}}\right] = \boldsymbol{A}\mathrm{E}\left[\boldsymbol{S}\boldsymbol{S}^{\mathrm{H}}\right]\boldsymbol{A}^{\mathrm{H}} + \sigma^2\boldsymbol{I} = \boldsymbol{A}\boldsymbol{R}_S\boldsymbol{A}^{\mathrm{H}} + \sigma^2\boldsymbol{I} \tag{5}$$

Since signal and noise are independent of each other, the covariance matrix can be decomposed into signal subspace and noise subspace, where $\boldsymbol{R}_S$ is the covariance matrix of the signal and $\boldsymbol{A}\boldsymbol{R}_S\boldsymbol{A}^{\mathrm{H}}$ is the signal part [21]. The EVD of the covariance matrix is:

$$\boldsymbol{R} = \boldsymbol{U}_S\boldsymbol{\Sigma}_S\boldsymbol{U}_S^{\mathrm{H}} + \boldsymbol{U}_N\boldsymbol{\Sigma}_N\boldsymbol{U}_N^{\mathrm{H}} \tag{6}$$

where $\boldsymbol{U}_S$, referred as the signal subspace, is composed of eigenvectors corresponding to M largest eigenvalues, and $\boldsymbol{U}_N$ is the noise subspace composed of eigenvectors corresponding to the remaining $N - M$ small eigenvalues [22].

Since the steering vectors associated with the signal incident directions are another group of bases of the signal subspace, then, utilizing the orthogonality between signal and noise subspace, the spatial and polar domain spectral function is constructed as:

$$P_{\mathrm{MUSIC}} = \frac{1}{\boldsymbol{a}_{\theta,\phi,\gamma,\eta}^{\mathrm{H}}\boldsymbol{U}_N\boldsymbol{U}_N^{\mathrm{H}}\boldsymbol{a}_{\theta,\phi,\gamma,\eta}} \tag{7}$$

It is observed that equation (7) has four unknown variables, i.e., θ, ϕ, γ and η. Therefore, a four-dimensional spectral peak search is required to obtain the estimated DOAs. Such kind of search has quite a lot of search nodes, thus leading to a high time consumption when implemented in hardware. To cut down the search time of spectral peak search, it is necessary to reduce the degree of freedom of the spectral function.

According to equation (2), the MUSIC spectrum in equation (7) can be further expressed as:

$$\begin{aligned} P_{\mathrm{MUSIC}} &= \frac{1}{(\boldsymbol{D}_{\theta,\phi}\boldsymbol{h}_{\gamma,\eta})^{\mathrm{H}}\boldsymbol{U}_N\boldsymbol{U}_N^{\mathrm{H}}(\boldsymbol{D}_{\theta,\phi}\boldsymbol{h}_{\gamma,\eta})} \\ &= \frac{1}{\boldsymbol{h}_{\gamma,\eta}^{\mathrm{H}}\left(\boldsymbol{D}_{\theta,\phi}^{\mathrm{H}}\boldsymbol{U}_N\boldsymbol{U}_N^{\mathrm{H}}\boldsymbol{D}_{\theta,\phi}\right)\boldsymbol{h}_{\gamma,\eta}} \end{aligned} \tag{8}$$

where $\boldsymbol{h}_{\gamma,\eta}$ is called the polarization vector, which is a full rank vector independent of DOA parameters, defined as:

$$\boldsymbol{h}_{\gamma,\eta} = \begin{bmatrix} \cos\gamma \\ \sin\gamma e^{j\eta} \end{bmatrix} \tag{9}$$

Therefore, $\boldsymbol{h}_{\gamma,\eta}$ in equation (8) can be omitted, and the spatial spectral function of polarization MUSIC is simplified to:

$$P_{\mathrm{MUSIC}}(\theta,\phi)=\underset{\theta,\phi}{\arg\max}\left[\det\left(\boldsymbol{D}_{\theta,\phi}^{\mathrm{H}}\boldsymbol{U}_N\boldsymbol{U}_N^{\mathrm{H}}\boldsymbol{D}_{\theta,\phi}\right)\right]^{-1} \tag{10}$$

Equation (10) is called the reduced-dimensional spectral function of polarization MUSIC algorithm.

3.FPGA implementation of polarization MUSIC algorithm

The uniform circular polarization sensitive array with centrosymmetric structure, as shown in Figure 1, is adopted in this paper, since uniform circular array is popular in industry. On the Virtex7-690T FPGA chip of Xilinx Company, the Verilog or System Verilog language is used to carry out comprehensive simulation on the Vivado platform.

Since the received signal is complex-valued data, the complex EVD and spectral peak search process of the MUSIC algorithm will consume a lot of hardware and time resources during FPGA implementation, bringing huge obstacles to improve the real-time performance of DOA estimation. Therefore, we hereby propose a real-valued preprocessing method that are suitable for uniform circular polarization sensitive arrays, which will reduce the computational complexity exponentially and improve the real-time performance of DOA estimation. Figure 2 shows the overall flow chart of the algorithm implementation.

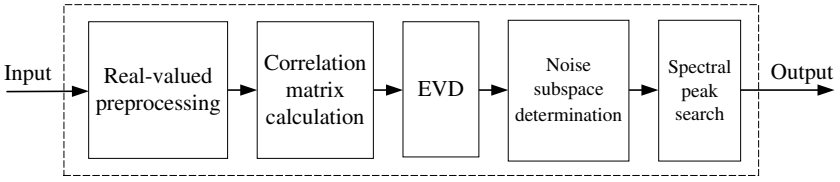


Figure 2 Algorithm Implementation Flowchart

3.1 Real-valued preprocessing

The steering vector omitting the polarization vector $\boldsymbol{h}_{\gamma,\eta}$ is:

$$\begin{aligned}\boldsymbol{D}_{\theta,\phi}&=\begin{bmatrix}u_1(\theta,\phi)\\u_2(\theta,\phi)\\\vdots\\u_N(\theta,\phi)\end{bmatrix}\begin{bmatrix}\cos\alpha_1 & \sin\alpha_1\\\cos\alpha_2 & \sin\alpha_2\\\vdots & \vdots\\\cos\alpha_N & \sin\alpha_N\end{bmatrix}\begin{bmatrix}-\sin\theta & \cos\phi\cos\theta\\\cos\theta & \cos\phi\sin\theta\end{bmatrix}\\&=\begin{bmatrix}\boldsymbol{D}_{\mathrm{H}} & \boldsymbol{D}_{\mathrm{V}}\end{bmatrix}\end{aligned} \tag{11}$$

According to the central symmetry relationship of the distributed polarization sensitive array element, when N is even, the coordinates and pointing angles of the 1st and $(K+1)$th array elements satisfy:

$$(x_{K+1},y_{K+1})=(-x_1,-y_1) \tag{12}$$

$$\alpha_{K+1} = \alpha_1 + \pi \tag{13}$$

According to equation (4), we get:

$$\tau_{K+1} = -\tau_1 \tag{14}$$

Therefore, the 1st and $(K+1)$th entry of $\boldsymbol{D}_{\mathrm{H}}$ are respectively expressed as:

$$\begin{aligned} \boldsymbol{D}_{\mathrm{H}}(1) &= u_1(\theta,\phi)\left(\cos\theta\sin\alpha_1 - \sin\theta\cos\alpha_1\right) \\ &= \left[\cos\left(2\pi f\tau_1\right) - j\sin\left(2\pi f\tau_1\right)\right]\left(\cos\theta\sin\alpha_1 - \sin\theta\cos\alpha_1\right) \end{aligned} \tag{15}$$

$$\begin{aligned} &\boldsymbol{D}_{\mathrm{H}}(K+1) = u_{K+1}(\theta,\phi)\left(\cos\theta\sin\alpha_{K+1} - \sin\theta\cos\alpha_{K+1}\right) \\ &= \left[\cos\left(2\pi f\tau_{K+1}\right) - j\sin\left(2\pi f\tau_{K+1}\right)\right]\left(\cos\theta\sin\alpha_{K+1} - \sin\theta\cos\alpha_{K+1}\right) \\ &= \left[\cos\left(2\pi f\tau_1\right) + j\sin\left(2\pi f\tau_1\right)\right]\left(-\cos\theta\sin\alpha_1 + \sin\theta\cos\alpha_1\right) \\ &= \left[-\cos\left(2\pi f\tau_1\right) - j\sin\left(2\pi f\tau_1\right)\right]\left(\cos\theta\sin\alpha_1 - \sin\theta\cos\alpha_1\right) \end{aligned} \tag{16}$$

where the Eulerian formulation is exploited. According to equation (15) and (16), it is obtained that the real parts of $\boldsymbol{D}_{\mathrm{H}}(1)$ and $\boldsymbol{D}_{\mathrm{H}}(K+1)$ are opposite, while the imaginary parts are the same. Additionally, there is the same symmetrical relationship between $\boldsymbol{D}_{\mathrm{V}}(1)$ and $\boldsymbol{D}_{\mathrm{V}}(K+1)$. The same relationship also exists for other array elements with distance K. When N is odd, the centrosymmetric elements in the array also meet the above symmetry relationship. Therefore, we construct a linear transformation matrix $\boldsymbol{U}$ to describe such relationship as:

$$\boldsymbol{U} = \begin{cases} \begin{bmatrix} \boldsymbol{I}_K & -\boldsymbol{I}_K \\ i\boldsymbol{I}_K & i\boldsymbol{I}_K \end{bmatrix}, N = 2K \\ \begin{bmatrix} \boldsymbol{I}_K & \boldsymbol{O} & -\boldsymbol{I}_K \\ \boldsymbol{O}^T & \sqrt{2} & \mathbf{O}^T \\ i\boldsymbol{I}_K & \boldsymbol{O} & i\boldsymbol{I}_K \end{bmatrix}, N = 2K+1 \end{cases} \tag{17}$$

where $\boldsymbol{I}_K$ is the identity matrix of order K and $\boldsymbol{O}$ is the zero-valued column vector with proper dimension. Performing this linear transformation on the received signal $\boldsymbol{X}(t)$ yields:

$$\boldsymbol{Y}(t) = \boldsymbol{U}\boldsymbol{X}(t) = \boldsymbol{U}\boldsymbol{A}\boldsymbol{S}(t) + \boldsymbol{U}\boldsymbol{N}(t) \tag{18}$$

At this time, $\boldsymbol{Y}(t)$ is still complex data, but useful information only exists in the real part. On the other hand, for hardware implementation, the real and imaginary data are separately input into the system as IQ data stream. Thus, we only need to consider the real part of $\boldsymbol{Y}(t)$. Figure 3 shows a schematic diagram of the FPGA implementation of the real-valued processing when N is even. Subtracting the real parts and adding the imaginary parts for all pairs of elements that are of distance K, the complex-valued data is transformed into real-valued data. When $N = 10$, 10 adder IP cores are required to complete the calculation, and only 2 clock cycles are consumed, which is easy to implement.

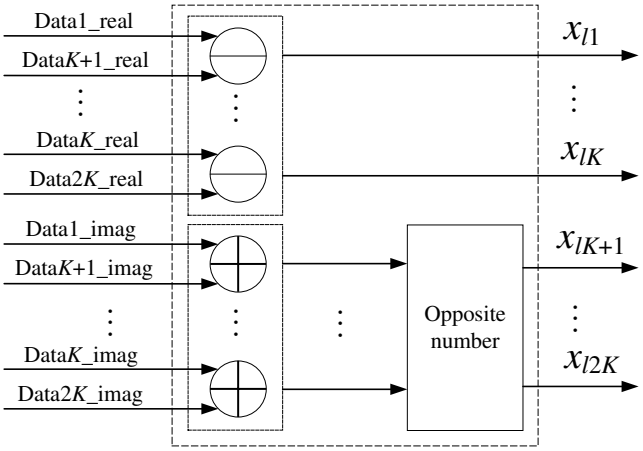


Figure 3 Schematic diagram of the implementation of real-valued preprocessing

After preprocessing, the steering vector becomes:

$$\begin{aligned}\boldsymbol{D}_{\mathrm{R}} &= \boldsymbol{U}\,\boldsymbol{D}_{\theta,\phi} \\ &= \begin{cases} \begin{bmatrix} 2\mathrm{Re}\left(\boldsymbol{D}_{\mathrm{H}}(1:K)\right) & 2\mathrm{Re}\left(\boldsymbol{D}_{\mathrm{V}}(1:K)\right) \\ -2\mathrm{Im}\left(\boldsymbol{D}_{\mathrm{H}}(K+1:2K)\right) & -2\mathrm{Im}\left(\boldsymbol{D}_{\mathrm{V}}(K+1:2K)\right) \end{bmatrix}, N = 2K \\ \begin{bmatrix} 2\mathrm{Re}\left(\boldsymbol{D}_{\mathrm{H}}(1:K)\right) & 2\mathrm{Re}\left(\boldsymbol{D}_{\mathrm{V}}(1:K)\right) \\ \sqrt{2}\mathrm{Re}\left(\boldsymbol{D}_{\mathrm{H}}(K+1)\right) & \sqrt{2}\mathrm{Re}\left(\boldsymbol{D}_{\mathrm{V}}(K+1)\right) \\ -2\mathrm{Im}\left(\boldsymbol{D}_{\mathrm{H}}(K+2:2K)\right) & -2\mathrm{Im}\left(\boldsymbol{D}_{\mathrm{V}}(K+2:2K)\right) \end{bmatrix}, N = 2K+1 \end{cases} \end{aligned} \tag{19}$$

So far, the subsequent calculation of polarization MUSIC algorithm can be completely finished by real values. Compared with complex-valued calculation, it can save a lot of hardware resources, shorten the calculation time, and improve the real-time performance of DOA estimation.

3.2 Calculation of covariance matrix

The formula to obtain the signal covariance matrix $\boldsymbol{R}$ in practice is:

$$\boldsymbol{R} = \mathrm{E}[\boldsymbol{X}\,\boldsymbol{X}^{\mathrm{H}}] \approx \frac{1}{L}\sum_{l=1}^{L} \boldsymbol{x}_l \boldsymbol{x}_l^{\mathrm{H}} \tag{20}$$

where $\boldsymbol{X}$ is the signal data received by the array, and L is the number of snapshots. Hereby, it is only required to calculate the value of the upper triangular element, since $\boldsymbol{R}$ is a real symmetric matrix after real-valued preprocessing. In addition, the number of snapshots can be omitted in the calculation, since the averaging operation has no effect on subsequent calculations. To sum up, the covariance matrix calculation formula can be expressed as the following form:

$$\widetilde{\boldsymbol{R}} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_L][\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_L]^{\mathrm{T}} = \sum_{l=1}^{L} \boldsymbol{x}_l \boldsymbol{x}_l^{\mathrm{T}} \tag{21}$$

In the above equation, $\boldsymbol{x}_l\,(l=1,2,...,L)$ is the data obtained by the real-valued preprocessing of the lth received signal snapshot. The detailed expression of $\boldsymbol{x}_l{\boldsymbol{x}_l}^{\mathrm{T}}$ is:

$$\boldsymbol{x}_l{\boldsymbol{x}_l}^{\mathrm{T}}=\begin{bmatrix} x_{l1} \\ x_{l2} \\ \vdots \\ x_{lN} \end{bmatrix}\begin{bmatrix} x_{l1} & x_{l2} & \cdots & x_{lN} \end{bmatrix}=\begin{bmatrix} x_{l1}^2 & x_{l1}x_{l2} & \cdots & x_{l1}x_{lN} \\ & x_{l2}^2 & \cdots & x_{l2}x_{lN} \\ & & \ddots & \vdots \\ & & & x_{lN}^2 \end{bmatrix} \tag{22}$$

where $x_{ln}(l=1,2,\ldots,L,n=1,2,\ldots,N)$ is the nth element in $\boldsymbol{x}_l$. The matrix has a total of $N(N+1)/2$ upper triangular elements, and each element is obtained by a multiplier (Multiplier IP core) and an addition accumulator (Accumulator IP core).

3.3 Eigenvalue Decomposition

After obtaining the covariance matrix, we need to perform the corresponding EVD to get the eigenvalues and eigenvectors. The Jacobi algorithm is one of the most commonly used EVD algorithms, where a series of rotation transformations are performed to transform the matrix into a diagonal matrix [23]. In this paper, the parallel Jacobi algorithm is selected to make full use of the parallel computing advantages of FPGA and Jacobi algorithm. Using the systolic array structure [24], the upper triangular elements of the covariance matrix of order $N\times N$ are divided into $N-1$ groups, and each group contains $[K\times(K+1)]/2$ matrix processing units of order 2×2. According to the position of each unit in the covariance matrix, it can be divided into the diagonal and off-diagonal unit. The parallel grouping and scheduling rules ensure that the processing units in the same group are independent and parallel to perform Jacobi rotation. When the number of array elements is $N=10$, the parallel Jacobi grouping is shown in Table 1:

Table 1 10×10 Matrix Parallel Jacobi Grouping

Serial number	Jacobi Grouping
S1	{(1,2) (3,4) (5,6) (7,8) (9,10)}
S2	{(1,4) (2,6) (3,8) (5,10) (7,9)}
S3	{(1,6) (4,8) (2,10) (3,9) (5,7)}
S4	{(1,8) (6,10) (4,9) (2,7) (3,5)}
S5	{(1,10) (8,9) (6,7) (5,4) (2,3)}
S6	{(1,9) (10,7) (8,5) (6,3) (4,2)}
S7	{(1,7) (9,5) (10,3) (8,2) (6,4)}
S8	{(1,5) (7,3) (9,2) (10,4) (8,6)}
S9	{(1,3) (5,2) (7,4) (9,6) (10,8)}

Then, the value of the rotation angle ϑ is calculated through the diagonal unit, and the calculation formula is:

$$\vartheta=\begin{cases} \frac{1}{2}\arctan\left(\frac{2b}{a_{ii}-a_{jj}}\right), a_{ii}\neq a_{jj} \\ \frac{\pi}{4}sign\,(b)\,, a_{ii}=a_{jj} \end{cases} \tag{23}$$

where a_{ii} and a_{jj} are the diagonal elements of the diagonal unit $\begin{bmatrix} a_{ii} & b \\ b & a_{jj} \end{bmatrix}$, and b is the off-diagonal element. Similarly, in the off-diagonal unit $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, a and

d are diagonal elements, b and c are off-diagonal elements. The rotation matrix corresponding to this diagonal unit is:

$$W_{ij} = \begin{bmatrix} \cos\vartheta & \sin\vartheta \\ -\sin\vartheta & \cos\vartheta \end{bmatrix} \tag{24}$$

The elements of the rotation matrix corresponding to all the diagonal units are arranged into a matrix according to the rows and columns of the diagonal elements of the diagonal unit, and the other positions are filled with 0, which is the rotation matrix of the entire covariance matrix.

The Jacobi rotation operation is performed on the diagonal unit according to the following equation:

$$\begin{bmatrix} a'_{ii} & b' \\ b' & a'_{jj} \end{bmatrix} = \begin{bmatrix} \cos\vartheta & \sin\vartheta \\ -\sin\vartheta & \cos\vartheta \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} a_{ii} & b \\ b & a_{jj} \end{bmatrix} \begin{bmatrix} \cos\vartheta & \sin\vartheta \\ -\sin\vartheta & \cos\vartheta \end{bmatrix} \tag{25}$$

At the same time, the Jacobi rotation operation is performed on the off-diagonal unit according to the following formula:

$$\begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix} = \begin{bmatrix} \cos\vartheta & \sin\vartheta \\ -\sin\vartheta & \cos\vartheta \end{bmatrix}^{\mathrm{T}} \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} \cos\vartheta & \sin\vartheta \\ -\sin\vartheta & \cos\vartheta \end{bmatrix} \tag{26}$$

Note that the data of the rotation matrix is transferred to all off-diagonal units simultaneously so that all units can perform Jacobian rotation and update the eigenvector matrix parallelly. The initial eigenvector matrix of the covariance matrix is set as $\boldsymbol{V}_0 = \boldsymbol{I}_N$, where $\boldsymbol{I}_N$ is the identity matrix of order N. Then, each time when the rotation is performed, the eigenvectors are updated as:

$$\begin{bmatrix} \boldsymbol{v}'_i \\ \boldsymbol{v}'_j \end{bmatrix} = \boldsymbol{W}_{ij} \begin{bmatrix} \boldsymbol{v}_i \\ \boldsymbol{v}_j \end{bmatrix} \tag{27}$$

where $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ represent the ith row and the jth row of the eigenvector matrix, respectively.

Since each group performs the same iterative calculation, in order to save resources, a group of hardware resources are repeatedly used $N-1$ times to complete the cleaning. Thus, it is necessary to exchange data at the input end and the output end. Data exchange is performed according to the grouping of parallel Jacobian decomposition, and the exchange formula is:

$$\boldsymbol{R}_{i+1} = \boldsymbol{W}_i \boldsymbol{R}_i \boldsymbol{W}_i^{\mathrm{T}} = \boldsymbol{C}_i \left[\boldsymbol{W}_1 \left(\boldsymbol{H}_i \boldsymbol{R}_i \boldsymbol{H}_i^{\mathrm{T}} \right) \boldsymbol{W}_1^{\mathrm{T}} \right] \boldsymbol{C}_i^{\mathrm{T}} \tag{28}$$

where $\boldsymbol{W}_i$ represents the rotation matrix of the ith group with $i = 1, 2, \cdots, N-1$, and $\boldsymbol{H}_i$ and $\boldsymbol{C}_i$ represents the input and output switching matrix, respectively. Since the hardware resources of the first group are used cyclically, the remaining $N-2$ groups are subjected to elementary row-column transformations based on the rotation matrix of the first group. In addition, $\boldsymbol{H}_i$ and $\boldsymbol{C}_i$ are mutually reversed.

The first-level cleaning is completed until $N-1$ groups of Jacobi rotations are finished. In order to save hardware resources, the data exchange operation involves no matrix multiplication. Instead, a state machine is used to directly perform register assignment operations. For a 10-order matrix, each level of cleaning needs to complete 9 groups of Jacobi rotations in Table 1. The input control module exchanges the input data according to the new positions of the elements in the group, and the output control module exchanges the updated data according to the original positions in the matrix.

When performing EVD, the more levels the cleaning is performed, the closer the eigenvalue matrix is to the diagonal matrix, that is, the smaller the error is. However, at the same time, the time consumption is increased. Therefore, The number of cleaning levels should be selected according to actual requirements. The FPGA implementation structure of each group of Jacobi rotations is shown in Figure 4. The arc-tangent value is calculated by calling Arc Tan mode of CORDIC IP core so as to obtain the rotation angle, while the sine and cosine value of the rotation angle is calculated by calling the Sin and Cos mode of the CORDIC IP core. In order to reduce the error of EVD, this paper adopts the three-level cleaning structure as shown in Figure 5.

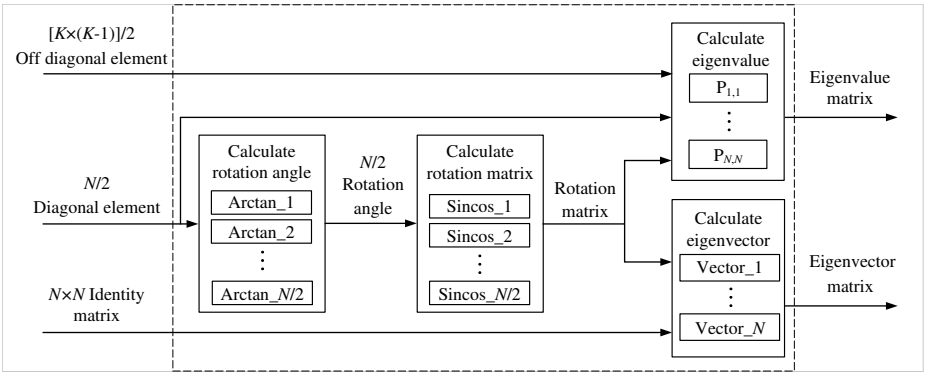


Figure 4 Single group of Jacobi rotations

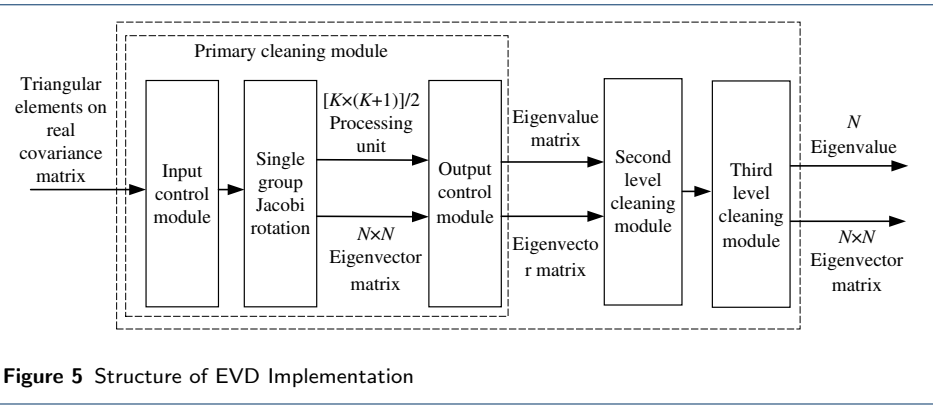


Figure 5 Structure of EVD Implementation

3.4 Noise subspace determination

In this module, the eigenvalues are firstly sorted, and the noise subspace is then determined via the rearranged results. The N eigenvalues are sorted in descent order, where each eigenvalue is compared with the other N eigenvalues in turn, and the number greater than or equal to the current eigenvalue is the updated sorting result.

The comparison of eigenvalues is carried out through the operators "$<$" and "$==$". Since the eigenvalue indicates the power of the corresponding signal or noise, the eigenvalue must be positive. All the N comparison units are launched in parallel. The input of the ith comparison unit is λ_i (fixed value) and λ_j $(j = 1, 2, ..., N)$ (sequential input), where i and j are input as flag bits at the same time. If the comparison result of the operator "$<$" is 1, then output the result $k + 1$, where k represents the current rank of λ_i. Furthermore, if the comparison result of the operator "$<$" is 0, and the comparison result of the operator "$==$" is 1, the output result is also $k+1$. Otherwise, keep the output result k unchanged until the comparison is complete. The final result output is the index of the rearranged λ_i $(i = 1, 2, ..., N)$. Figure 6 shows the FPGA implementation schematic diagram of eigenvalue sorting. According to the eigenvalue sorting results, the corresponding eigenvectors are selected to compose the noise subspace.

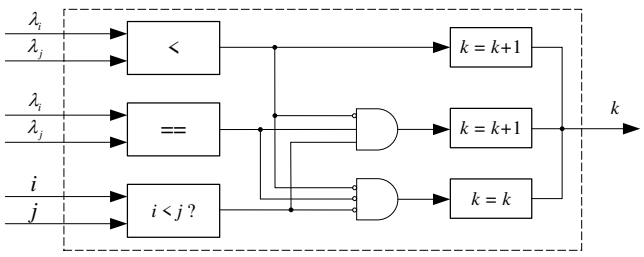


Figure 6 Eigenvalue sorting FPGA implementation structure

3.5 Spectral peak search

In order to search for spectral peaks, the steering vector should be calculated first. Then, calculate the spectral function using the steering vector and the noise subspace. Finally, the spectral peak search is performed.

The real-valued steering vector of the two-dimensional array is:

$$\boldsymbol{D}_{\mathrm{R}}(\theta,\phi) = \begin{bmatrix} \cos(2\pi f\tau_1) \\ \vdots \\ \cos(2\pi f\tau_{N/2}) \\ \sin(2\pi f\tau_{N/2+1}) \\ \vdots \\ \sin(2\pi f\tau_N) \end{bmatrix} \begin{bmatrix} \cos\alpha_2 & \sin\alpha_2 \\ \vdots & \vdots \\ \cos\alpha_{N/2} & \sin\alpha_{N/2} \\ \cos\alpha_{N/2+1} & \sin\alpha_{N/2+1} \\ \vdots & \vdots \\ \cos\alpha_N & \sin\alpha_N \end{bmatrix} \begin{bmatrix} -\sin\theta & \cos\phi\cos\theta \\ \cos\theta & \cos\phi\sin\theta \end{bmatrix}$$

(29)

where $(\theta, \phi)$is the DOA of the searching grid, and the corresponding sine and cosine values can be stored in the ROM according to the search range and step. The schematic diagram of the implementation of the steering vector calculation is shown in Figure 7.

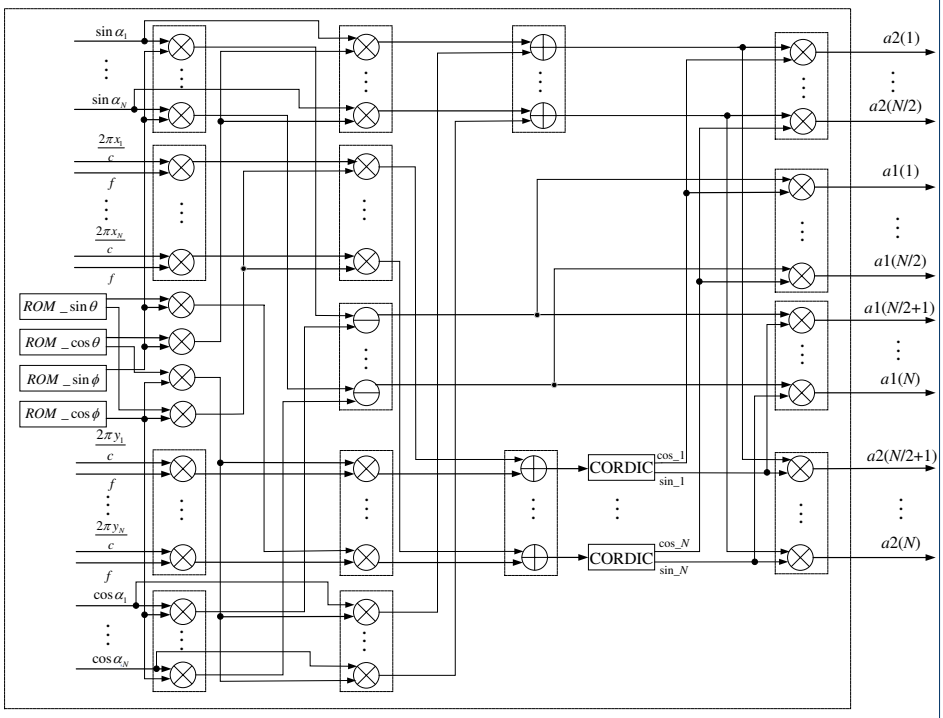


Figure 7 Steering vector calculation diagram

It can be seen from the above analysis that the spectral function of the reduced-dimensional polarization MUSIC after real-valued processing is expressed as:

$$P_{\mathrm{MUSIC}} = \underset{\theta,\phi}{\arg\max} \left[\det \left(\boldsymbol{D}_{\mathrm{R}}^{\mathrm{T}}(\theta,\phi)\, \widetilde{\boldsymbol{U}}_N \widetilde{\boldsymbol{U}}_N^{\mathrm{T}} \boldsymbol{D}_{\mathrm{R}}(\theta,\phi) \right) \right]^{-1} \tag{30}$$

where $\boldsymbol{D}_{\mathrm{R}}(\theta,\phi)$ and $\widetilde{\boldsymbol{U}}_N$ are the real-valued steering vector and noise subspace, respectively. To simplify the calculation, the pseudo spectral function is constructed as:

$$P(\theta,\phi) = \left| \det \left(\sum_{i=1}^{N-M} \boldsymbol{D}_{\mathrm{R}}^{\mathrm{T}}(\theta,\phi)\, \widetilde{\boldsymbol{U}}_N(i)\, \widetilde{\boldsymbol{U}}_N^{\mathrm{T}}(i)\, \boldsymbol{D}_{\mathrm{R}}(\theta,\phi) \right) \right| \tag{31}$$

where $\widetilde{\boldsymbol{U}}_N(i)$ is ith column vector of $\widetilde{\boldsymbol{U}}_N$. The spectral peak search is equivalently converted into the spectral valley search.

Hereby, we take the ith column $\boldsymbol{D}_{\mathrm{R}}^{\mathrm{T}}(\theta,\phi) \times \widetilde{\boldsymbol{U}}_N(i) \times \widetilde{\boldsymbol{U}}_N^{\mathrm{T}}(i) \times \boldsymbol{D}_{\mathrm{R}}(\theta,\phi)$, referred as the internal cell, as an instance to illustrate the spectrum calculation process in

FPGA. The corresponding implementation structure is shown in Figure 8. Considering the single source scenario, the noise subspace is a matrix of order $N \times (N-1)$, indicating that N-1 internal cells are required. Figure 9 is a schematic diagram of the spectral function calculation.

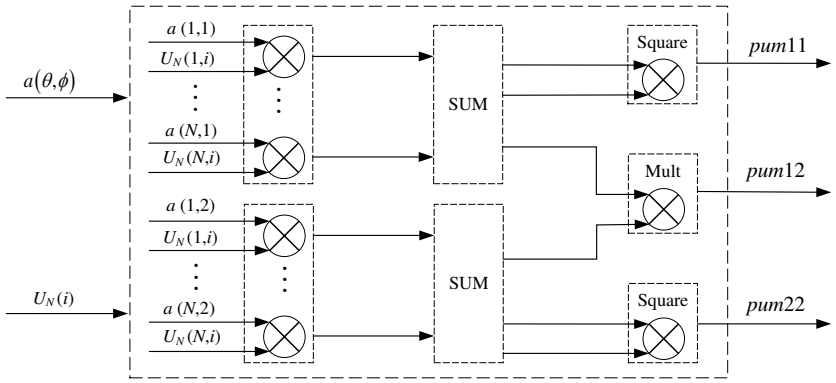


Figure 8 The structure of the internal cell

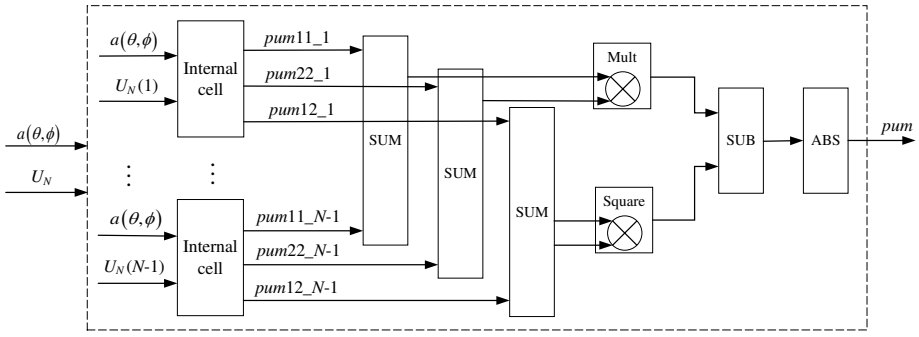


Figure 9 Spectral function calculation

Assume that the search ranges for azimuth and elevation are R_θ and R_ϕ, and the search steps are Δ_θ and Δ_ϕ, respectively. Let $I = R_\theta/\Delta_\theta$ and $J = R_\phi/\Delta_\phi$. Then, (θ, ϕ) can be reconstructed into a new matrix of size $I \times J$, which allows the spectral function $P(\theta, \phi)$ to be represented as a $\boldsymbol{P}$ matrix of order $I \times J$, namely:

$$\boldsymbol{P} = \begin{bmatrix} P(\theta_1,\phi_1) & P(\theta_2,\phi_1) & \cdots & P(\theta_I,\phi_1) \\ P(\theta_1,\phi_2) & P(\theta_2,\phi_2) & \cdots & P(\theta_I,\phi_2) \\ \vdots & \vdots & \ddots & \vdots \\ P(\theta_1,\phi_J) & P(\theta_2,\phi_J) & \cdots & P(\theta_I,\phi_J) \end{bmatrix} \tag{32}$$

In this way, the spectral peak search problem of the two-dimensional polarization MUSIC algorithm can be transformed into the problem of searching for local minimization value in $\boldsymbol{P}$.

For notational simplicity, we use $P_{i,j}$ to represent $P(\theta_i, \phi_j), i = 1, 2, ..., I, j = 1, 2, ..., J$. If $P_{i,j}$ corresponds to a spectral valley, it should be satisfied that the value of $P_{i,j}$ is minimum in the adjacent 3×3 region. The schematic diagram of the spectral valley value associated with the spatial positions is shown in Figure 10. By searching for all spectral valley positions, and sorting the results in ascent order, the azimuth and elevation corresponding to the first M minimum spectral valleys are the estimated DOA parameters.

	j-1	j	j+1
i-1	$P_{i\text{-}1,j\text{-}1}$	$P_{i\text{-}1,j}$	$P_{i\text{-}1,j+1}$
i	$P_{i,j\text{-}1}$	$P_{i,j}$	$P_{i,j+1}$
i+1	$P_{i+1,j\text{-}1}$	$P_{i+1,j}$	$P_{i+1,j+1}$

Figure 10 Schematic diagram of the valley value of the spectrum

It is seen from the searching progress that the entire spatial domain should be scanned to obtain all the valleys. In this paper, the azimuth is slowly changed, while the elevation is fast changed. Specifically, the azimuth is fixed and search valleys for the entire elevation first. Then, change the azimuth and repeat the searching operation. In the FPGA implementation, data transmission is performed through the FIFO IP core to realize the nine-square grid data comparison.

To further reduce the computational burden, we adopt a two-stage search strategy here. The first stage is a rough search, where a large search step is used. The spectral valley results $(\hat{\theta}_m, \hat{\phi}_m)\,(m = 1, 2, \cdots, M)$ within the search range are obtained through the previously demonstrated spectrum searching. Then, input the results serially to the second stage for fine search with a small search step, where the search range is properly selected around the searching results in first stage. Finally, all the DOAs $(\theta_m, \phi_m)\,(m = 1, 2, \cdots, M)$ are estimated. Figure 11 shows the realization of spectral valley search for the case of 2 sources, where all the valleys are obtained first and the 2 smallest valleys are then selected.

3.6 Overall block diagram

The overall block diagram consisting of previous modules is shown in Figure 12. The input of the system is the IQ data stream of N channel, as well as the number of snapshots and signal frequency. On the other hand, the output is the estimated DOAs and the corresponding spectral valley values. First, real-valued preprocessing is performed on the input IQ data stream, and the covariance matrix is calculated according to the obtained results. Then, the upper triangular element of the real symmetric covariance matrix is input into the EVD module, where the parallel Jacobi algorithm is used. The acquired eigenvalues and eigenvectors are next input into the noise subspace determination module, where the eigenvalues are sorted first,

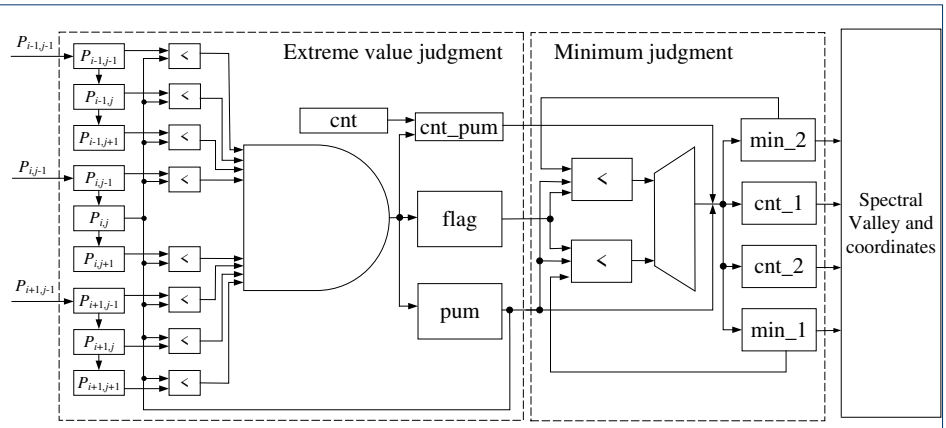


Figure 11 Extreme value judgment and minimum value judgment

and the noise subspace is then obtained by selecting the eigenvectors corresponding to the $N - M$ smallest eigenvalues. Finally, the spectral peak search module are exploited to estimate the DOA parameters.

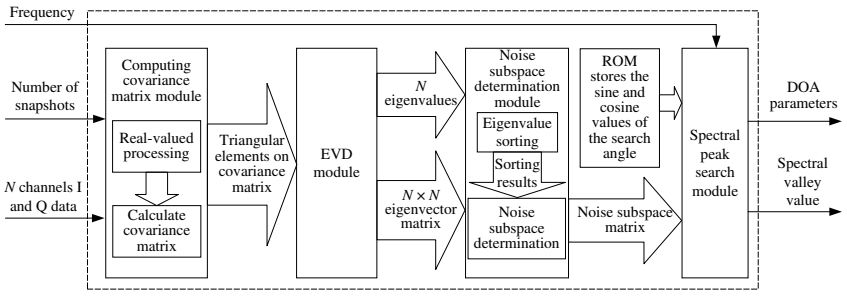


Figure 12 Overall block diagram of algorithm implementation

In addition, the pipeline processing structure, as shown in Figure 13, is adopted because the time-consuming of the spectral peak search module is greater than the sum of the other modules. Therefore, the running time of one estimation is dominated by the spectral peak search module.

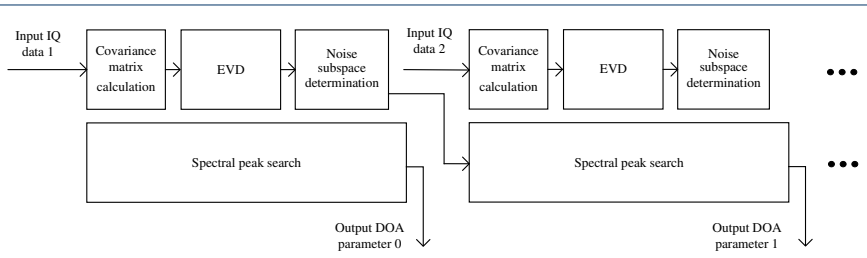


Figure 13 Data pipeline processing block diagram

4.Experimental Results and Performance Evaluation

4.1 FPGA Implementation Results and Resource Occupancy

In the first experiment, we examine the hardware resource occupancy and time consumption of the proposed scheme. Restricted by FPGA capacity, a 10-element uniform circular array is used in the proposed real-valued polarization MUSIC algorithm, while only 8 elements are used in the traditional complex-valued polarization MUSIC algorithm. The signal-to-noise ratio (SNR) and the number of snapshots is 20 dB and 100, respectively. Setting the coarse and fine search step as 2° and 0.2°, respectively, the consumed hardware resources and running time are shown in Table 2 and Table 3, respectively, where the searching range is $\theta \in [0, 360^{\circ}]$ and $\phi \in [0, 30^{\circ}]$. It is seen from Table 2 that, for the LUT, FF and DSP resource, the utilization of the proposed real-valued MUSIC is much less than that of the traditional MUSIC, indicating that the amount of computation is significantly reduced via the proposed scheme. In addition, from Table 3, the two algorithms have the largest difference in the EVD module in terms of time consumption. The main reason is that the complex calculation of the traditional MUSIC doubles the covariance matrix dimension in the EVD, while this issue is avoided by the real-valued preprocessing in the proposed scheme. When the operating frequency of the chip is 100MHz, it takes $34.60\mu s$ for the proposed real-valued MUSIC to finish the spectrum peak searching in one estimation due to the pipeline design. Note that the proposed scheme exploits more array elements than the conventional MUSIC. The improvement on time consumption will be further improved if compared with conventional MUSIC based on 10-element array. However, the hardware resources of FPGA cannot support the running of the conventional MUSIC based on 10-element array. Therefore, an 8-element array is considered in conventional MUSIC for comparison.

Table 2 Device utilization summary

		Real-valued Polarization MUSIC[1]		Polarization MUSIC[2]	
Resource	Available	Utilization	Proportion (%)	Utilization	Proportion (%)
LUT	433200	84658	19.54	268978	62.09
LUTRAM	174200	439	0.25	346	0.2
FF	866400	95185	10.99	214345	24.74
BRAM	1470	7	0.48	6	0.41
DSP	3600	1311	36.42	2979	82.75

[1] 10 array elements.
[2] 8 array elements.

Table 3 Timing reports

	Real-valued Polarization MUSIC[1]		Polarization MUSIC[2]	
Module	Clock cycle	Time(μs) [3]	Clock cycle	Time(μs) [3]
Covariance matrix calculation	105	1.05	105	1.05
EVD	1769	17.69	2939	29.39
Determine Noise Subspace	17	0.17	23	0.23
Spectral Peak Search	3460	34.60	3472	34.72
Total	5396	53.96	6539	65.39

[1] 10 array elements.
[2] 8 array elements.
[3] Time-consuming when the clock frequency is 100MHz.

4.2 Evaluation of estimation accuracy of proposed real-valued polarization MUSIC

In this part, the estimation accuracy of the proposed real-valued polarization MUSIC is compared with the conventional polarization MUSIC under the numerical

simulation environment. Considering the single source scenario, the DOA parameters of the source are randomly given in the searching range, and the polarization parameter is fixed as $(\gamma, \eta) = (45°, 15°)$. Note that 10-element uniform circular array is used for both the two algorithms here. In the simulation, the SNR ranges from 0 dB to 28 dB, and 100 signal snapshots are used. The searching step is set as 0.2°. The estimation accuracy is evaluated via the root mean square error (RMSE), where 1000 Monte-Carlo experiments are conducted in each SNR. The result is shown in Figure 14. It is observed that the RMSE of the real-valued polarization MUSIC algorithm is slightly lower than that of the polarization MUSIC algorithm in low SNR region. However, as SNR increases, the two methods achieve about the same estimation accuracy. Based on these simulation results, it can be concluded that the proposed real-valued polarization MUSIC algorithm has important engineering application value.

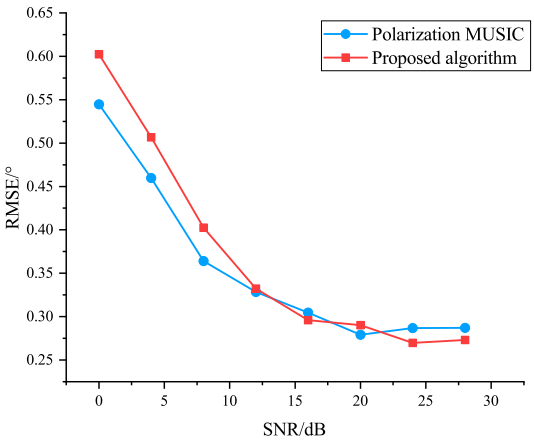


Figure 14 RMSE vs. SNR curve

5.Conclusion

In this paper, the implementation of polarization MUSIC algorithm on FPGA was investigated for the first time. To address the resource occupancy and time consumption issue in conventional complex-valued polarization MUSIC, a real-valued preprocessing was proposed by utilizing the centrosymmetric structure of uniform circular array, where the received data and steering vector are converted into real value via a linear transformation. By operating real-valued covariance matrix, the time consumption in EVD is dramatically reduced. Furthermore, the whole implementation scheme on FPGA of the proposed real-valued MUSIC was constructed in this paper. The design of each module was illustrated in detail. To effectively connect all the modules, a pipeline implementation structure was also designed in this paper. According to the experimental results, it takes 53.96μs to finish one entire DOA estimation with an operating frequency of 100MHz, while the estimation accuracy is guaranteed. Compared with the existing relevant methods, the proposed

implementation scheme has the advantages of high precision, outstanding real-time performance and low resource consumption, which are of great engineering application value.

Acknowledgements
Not applicable

Authors' contributions
Lutao Liu and Ying Cao proposed the original idea. Ying Cao wrote the paper under the guidance of Lutao Liu. Ying Cao and Muran Guo designed the experiment and provided all of the figures and data. Muran Guo proofread the paper. All authors read and approved the final manuscript.

Funding
This work is supported by the National Natural Science Foundation of China under Grant 62071137 and 62001136.

Abbreviations
DOA Direction of Arrival
MUSIC Multiple Signal Classification
FPGA Field Programmable Gate Array
EVD eigenvalue decomposition
DSP Digital Signal Processor
AWGN Additive Gaussian White Noise
SNR Signal-to-Noise Ratio
RMSE Root Mean Square Error

Availability of data and materials
Not applicable

Ethics approval and consent to participate
Not applicable

Competing interests
The authors declare that they have no competing interests.

Consent for publication
All authors agree to publication.

Author details
[1]The College of Information and Communication Engineering, Harbin Engineering University, Harbin, China. [2]Key Laboratory of Advanced Marine Communication and Information Technology, Ministry of Industry and Information Technology,Harbin Engineering University, Harbin, China.

References

1. L, V.T.H.: Optimum Array Processing: Part IV of Detection, Estimation, and Modulation Theory. Wiley Interscience, New York, NY, USA (2002)
2. Ahmed, A., Zhang, Y.D., Gu, Y.: Dual-function radar-communications using QAM-based sidelobe modulation. Digit Signal Process **82**, 166–174 (2018). doi:10.1016/j.dsp.2018.06.018
3. Tian, Y., Liu, S., Liu, W., Chen, H., Dong, Z.: Vehicle positioning with deep learning-based direction-of-arrival estimation of incoherently distributed sources. IEEE Internet Things, 1–1 (2022). doi:10.1109/JIOT.2022.3171820
4. Wan, L., Sun, Y., Sun, L., Ning, Z., Rodrigues, J.J.P.C.: Deep learning based autonomous vehicle super resolution DOA estimation for safety driving. IEEE Trans. Intell. Transp. **22**(7), 4301–4315 (2021). doi:10.1109/TITS.2020.3009223
5. Qin, S., Zhang, Y.D., Amin, M.G.: Generalized coprime array configurations for direction-of-arrival estimation. IEEE T Signal Proces **63**(6), 1377–1390 (2015). doi:10.1109/TSP.2015.2393838
6. Zheng, H., Shi, Z., Zhou, C., Haardt, M., Chen, J.: Coupled coarray tensor CPD for DOA estimation with coprime L-shaped array. IEEE Signal Proc Let **28**, 1545–1549 (2021). doi:10.1109/LSP.2021.3099074
7. Zhou, C., Gu, Y., Shi, Z., Zhang, Y.D.: Off-grid direction-of-arrival estimation using coprime array interpolation. IEEE Signal Proc Let **25**(11), 1710–1714 (2018). doi:10.1109/LSP.2018.2872400
8. Rumsey, V.H., Deschamps, G.A., Kales, M.L., Bohnert, J.I., Booker, H.G.: Techniques for handling elliptically polarized waves with special reference to antennas: Introduction. Proceedings of the IRE **39**(5), 533–534 (1951). doi:10.1109/JRPROC.1951.233134
9. Nehorai, A., Paldi, E.: Vector-sensor array processing for electromagnetic source localization. IEEE T Signal Proces **42**(2), 376–398 (1994). doi:10.1109/78.275610
10. Xu, Y., Liu, Z., Gong, X.: Polarization Sensitive Array Signal Processing. Beijing Institute of Technology Press, Beijing (2013)
11. Dai, M., Ma, X., Liu, W., Weixing, S., Han, Y., Xu, H.: Monopulse based DOA and polarization estimation with polarization sensitive arrays. Signal Processing **195**, 108487 (2022). doi:10.1016/j.sigpro.2022.108487
12. Chen, T., Yang, J., Wang, W., Guo, M.: Generalized sparse polarization array for DOA estimation using compressive measurements. Wireless Communications and Mobile Computing, 5539709 (2021). doi:10.1155/2021/5539709

13. Wang, X., Huang, M., Wan, L.: Joint 2D-DOD and 2D-DOA estimation for coprime emvs–mimo radar. Circuits Syst Signal Process **40**(6), 2950–2966 (2021). doi:10.1007/s00034-020-01605-5
14. Schmidt, R.: Multiple emitter location and signal parameter estimation. IRE Trans. Antennas Propag. **34**(3), 276–280 (1986). doi:10.1109/TAP.1986.1143830
15. Cheng, C., Hua, Y.: Performance analysis of music and pencil-music algorithms for diversely-polarized array. In: Proceedings of ICASSP '94. IEEE International Conference on Acoustics, Speech and Signal Processing: 19-22 April 1994; Adelaide, SA, Australia, vol. 42, pp. 3150–3165 (1994). doi:10.1109/ICASSP.1994.389836
16. Butt, U.M., Khan, S.A., Ullah, A., Khaliq, A., Reviriego, P., Zahir, A.: Towards low latency and resource-efficient FPGA implementations of the MUSIC algorithm for direction of arrival estimation. IEEE Trans. Circuits Syst. **68**(8), 3351–3362 (2021). doi:10.1109/TCSI.2021.3083280
17. Huang, K., Sha, J., Shi, W., Wang, Z.: An efficient FPGA implementation for 2-D MUSIC algorithm. Circuits Syst Signal Process **35**(5), 1795–1805 (2016). doi:10.1007/s00034-015-0144-z
18. Xu, D., Liu, Z., Qi, X., Xu, Y., Zeng, Y.: A FPGA-based implementation of MUSIC for centrosymmetric circular array. In: Yuan, B., Ruan, Q., Tang, X. (eds.) 9th International Conference on Signal Processing: 26-29 Oct. 2008; Beijing,China, pp. 490–493 (2008). doi:10.1109/ICOSP.2008.4697177
19. Shi, Z., He, Q., Liu, Y.: Accelerating parallel Jacobi method for matrix eigenvalue computation in DOA estimation algorithm. IEEE Trans. Veh. Technol. **69**(6), 6275–6285 (2020). doi:10.1109/TVT.2020.2984705
20. Hussain, A.A., Tayem, N., Soliman, A.-H.: Matrix decomposition methods for efficient hardware implementation of DOA estimation algorithms: A performance comparison. In: 2019 4th International Conference and Workshops on Recent Advances and Innovations in Engineering (ICRAIE): 27-29 Nov. 2019; Poornima Coll Engn, Dept Comp Sci, Malaysia, pp. 1–7 (2019). doi:10.1109/ICRAIE47735.2019.9037778
21. Zhang, M., Lyu, T.: Multi-parameter estimation based on improved MUSIC algorithm for polarization sensitive array. In: Wang, L., Chung, K. (eds.) 2019 International Workshop on Electromagnetics: Applications and Student Innovation Competition (iWEM): 18-20 Sep. 2019; Qingdao, China, pp. 1–4 (2019). doi:10.1109/iWEM.2019.8887887
22. Wang, Y.: Theory and Algorithm of Spatial Spectrum Estimation. Tsinghua University Press, Beijing (2004)
23. Shi, H., Jiang, Z., Liu, Q., Cai, X.: An efficient fPGA parallel implementation for MUSIC algorithm. In: 4th International Conference on Environmental Science and Material Application (ESMA): 15-16 Dec. 2018; Xian, China, vol. 252, p. 032168 (2019). doi:10.1088/1755-1315/252/3/032168
24. Brent, R.P., Luk, F.T., Van Loan, C.: Computation of the singular value decomposition using mesh-connected processors. Technical report, Cornell University (1982)