MUSIC 算法的 FPGA 实现①

白银山, 李 宏

(西北工业大学 电子信息学院, 西安 710129)

摘 要: 针对多重信号分类(MUSIC)算法计算复杂度高, 难以实时实现的特点, 给出了适用于均匀线阵的实数化预处理算法和实用的空间谱定义, 并选择了适合 FPGA 硬件实现的特征值分解算法, 给出了 MUSIC 算法 FPGA 实现的整体架构. 仿真实验结果表明, 该 FPGA 实现能够完成 MUSIC 算法的准确、快速计算.

关键词: MUSIC; FPGA; 算法选择; 硬件实现

FPGA Implementation of MUSIC

BAI Yin-Shan, LI Hong

(School of Electronic and Information, Northwestern Politechnical University, Xi'an 710129, China)

Abstract: Multiple signal classification(MUSIC) is a computationally complex algorithm that is hard to implement for real-time application. First, for the case of uniform linear array(ULA), the preprocessing algorithm making the following computations in real-value is presented and a practical spatial spectrum is defined. The algorithms for eigenvalue decomposition(EVD) suitable for hardware implementation is selected as well. Then the overall architecture of a FPGA implementation of MUSIC is proposed. Verification results show that this implementation with FPGA can compute MUSIC accurately with high speed.

Key words: multiple signal classification(MUSIC); field programme gate array(FPGA); algorithm selection; hardware implementation

近年来, 在阵列信号处理领域中, 信号子空间类高分辨率波达方向估计是一个十分热门的研究内容. 这类方法可以在不增大阵列实际口径的前提下用信号处理的 "软"手段来提高分辨率.

信号子空间类方法主要由两步组成: 一是对接收数据协方差矩阵进行正交分解来估计信号(或噪声)子空间;二是进行谱峰搜索来提取波达方向估值. 无论是正交分解还是谱值计算都面临着巨大的运算量.

在已报道的 MUSIC 算法在波达方向估计的相关应用中, 大多数是一些对实时性要求不太严格的应用, 如对雷雨天气的研究, 对慢速船舶的定位等. 但在移动通信, 电子侦察以及电子对抗等对实时性要求严格领域中的应用较少, 究其原因, 主要是因为 MUSIC 算法包含非常大的计算量, 现有系统的处理速度难以满足实际应用的需要. 因此, 研究 MUSIC 算法的实时实现有着十分重要的意义.

由于传统 MUSIC 算法[1]包含大量的复数运算, 硬件实现复杂, 有必要将复数运算转化为实数运算. 协方差矩阵特征值分解是 MUSIC 的硬件实现中最为复杂的部分, 特征值分解算法的选择一定程度上决定了整个系统的实时性能. 传统的空间谱定义不适合定点的硬件实现, 有必要给出一种实用的空间谱定义.

对于高速实现 MUSIC 算法, 国内目前大多采用并行 PDSP, 速度在 ms 级. 为此, 国内协方差模块和谱峰搜索模块采用 FPGA 实现, 速度达到了 μs 级, 但特征值分解模块还采用的是 DSP, 所以整个 MUSIC 算法计算时间停留在 ms 量级. 国外采用 FPGA 来实现整个 MUSIC 算法, 速度达到了 μs 量级.

① 基金项目:西北工业大学研究生创业种子基金(Z2013069)

收稿时间:2013-11-30;收到修改稿时间:2014-01-07

1　MUSIC算法及其预处理

设空间中 K 个波长为 λ 的远场窄带信号 $s_k(t)$ ($k=1,2,...,K$)从不同方向 θ 入射到背景噪声为 $n(t)$，相邻阵元间距为 d 的 M 阵元均匀线阵上. 则其 M 个接收数据模型可写成:

$X(t)=AS(t)+N(t)$，其中,

$X(t)=[x_1(t),x_2(t),\cdots,x_M(t)]^T$ 为 t 时刻阵列接收数据矢量;

$S(t)=[s_1(t),s_2(t),\cdots,s_K(t)]^T$ 为辐射信号源矢量;

$N(t)=[n_1(t),n_2(t),\cdots,n_M(t)]^T$ 为独立零均值，方差为 σ 的复高斯白噪声，且与信号不相关;

$A=[\alpha(\theta_1),\cdots,\alpha(\theta_k),\cdots,\alpha(\theta_K)]_{M\times K}$ 为阵列流型，其中，$\phi_k=2\pi d\sin\theta_k/\lambda$ ($k=1,2,...,K$).

$$A=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ e^{-j\phi_1} & e^{-j\phi_2} & \cdots & e^{-j\phi_K} \\ \vdots & \vdots & \ddots & \vdots \\ e^{-j(M-1)\phi_1} & e^{-j(M-1)\phi_2} & \cdots & e^{-j(M-1)\phi_K} \end{bmatrix} \quad (1)$$

从式(1)可以知道阵列的方向矩阵 A 为一个复矩阵，那么由(1)式推导出的阵列输出矢量 $X(t)$ 也是一个复矢量. 因此在应用 MUSIC 算法时，各种计算都是复数运算，并且包含大量的乘法和非线性运算.

然而，可以证明对于一个偶数阵元的对称阵列，可以通过一种简单有效的预处理方法[2]，将复数矩阵 A 转换为实数矩阵，把复矢量 $X(t)$ 用一个实矢量来代替，从而将各种复数计算转换为实数运算. 由于一次复数乘法相当于 4 次实数乘法和 2 次实数加法，一次复数加法相当于 2 次实数加法，因此通过预处理可以大大的减少算法的计算量，从而达到加快MUSIC算法处理速度的目的.

对于偶数阵元的对称阵列，例如 4 元均匀线阵，如图 1 所示，可以分成两个子阵 Sub1, Sub2:

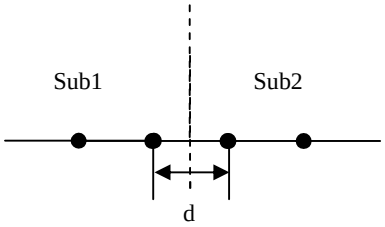

图1　四阵元均匀线阵

该预处理方法的应用可以通过如下方式来进行: 用 Sub1 阵元接收机输出的实部作为接收数据，用 Sub2 阵元接收机输出的虚部作为接收数据. 这样，就完成了预处理操作，而不增加额外的计算量.

2　MUSIC算法的FPGA模块

要高速实现波达方向估计算法，可以考虑三种数字处理器件. ASIC 专用集成电路，其硬件结构固定，不具备通用性，灵活性差，成本较高，实现复杂. PDSP 是基于精简指令集的计算机，其在运算上受制于时钟速率和串行指令流的限制，而且每个时钟内的有用操作受到限制. FPGA 结构是基于半定制门阵列的设计思想而得到的，具有专用集成电路的特点，可以在 FPGA 内部资源许可的条件下进行自由设计. 与 PDSP 相比，不受到运算单元数量的影响，可以用最恰当、最高效的硬件结构实现算法，提供更多的带宽，运算速度相对 PDSP 有较大提高.

对 MUSIC 算法的运算可以划分为三个步骤: 1)计算协方差矩阵; 2)特征值分解; 3)谱峰搜索. 这三个步骤为顺序执行，具有明显的串行性，即执行完前面的模块，才能执行后面的模块. 在进行 FPGA 模块划分时，可以对应 MUSIC 算法三个步骤，设计三个模块来分别实现它们，然后通过模块间互连，实现整个 MUSIC 算法. 用伪代码表示如下:

```
Begin(算法开始)
    输入预处理后的采样数据(4×256)
    计算实对称协方差矩阵 R(4×4)
    对矩阵 R 进行特征值(EVD)分解得特征向量 V
    计算伪谱值并比较得出波达方向(DOA)
End(算法结束)
```

图2给出了经过预处理之后的MUSIC算法的模块划分.

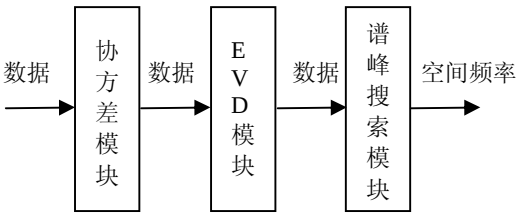

图2　MUSIC算法的FPGA模块

下面分析每个模块功能.

2.1 协方差模块设计

对于协方差模块，其功能是根据 $M\times N$ 的数据矩

阵(M 对应阵元阶数, N 对应采样点数)得到协方差矩阵. 从本质讲, 该模块是计算一个 $M\times N$ 矩阵与其自身转置($N\times M$ 矩阵)相乘, 从而得到一个 $M\times M$ 的对称协方差矩阵. 只需计算协方差矩阵的 4 个主对角元素和 6 个上三角元素. 用乘法累加器即可实现. 乘法累加器如图 3 所示:

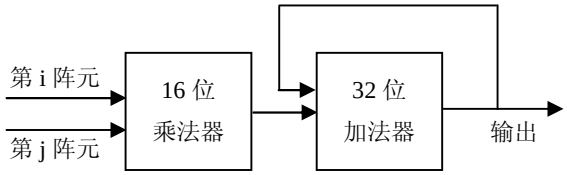

图 3 协方差模块乘法累加器单元

2.2 EVD 模块设计

其功能是根据协方差矩阵计算出特征值和特征向量. 特征值分解算法, 采用 Jacobi 算法. Jacobi 算法是计算实对称矩阵特征值的算法之一, 它使用一系列实平面旋转把实对称矩阵化为对角矩阵. 在 Jacobi 算法中, 对 M 阶对称矩阵 R 的上三角元素按算法依次置 0, 反复迭代后矩阵 R 收敛成对角矩阵, 此时可得到矩阵 R 的特征值和特征向量.

如果令 $p=1$, $q=2$, 那么 Jacobi 算法单次迭代步骤如下:

1)计算反正切:

$$\theta_{opt}=\frac{1}{2}arctg\left(\frac{2r_{pq}}{r_{pp}-r_{qq}}\right)$$

根据 p, q 值从对称矩阵 R 中提取矩阵元素 r_{pq}、r_{pp} 和 r_{qq}, 计算反正切得到旋转角 θ_{opt};

2)计算 2 次坐标旋转:

在第(1)步计算得到旋转角 θ_{opt} 后, 按照式(2)构造旋转矩阵 W_{pq}, 然后按照式(3)算出矩阵 R'. 运算等效于做 2 次坐标旋转.

$$W_{pq}=\begin{bmatrix}1 & & & & & 0\\ & \ddots & & & & \\ & & \cos\theta_{opt} & & \sin\theta_{opt} & & p\text{行}\\ & & & \ddots & & & \\ & & & & \ddots & & \\ & & -\sin\theta_{opt} & & \cos\theta_{opt} & & q\text{行}\\ & & & & & \ddots & \\ 0 & & & & & 1\end{bmatrix} \quad (2)$$

$$R'=W_{pq}RW_{pq}^{T} \quad (3)$$

3)判断:

如果 R' 已经转化为对角矩阵, 则跳出迭代, 此时 R' 即为所求;否则令 $R=R'$, 按照

$$\begin{Bmatrix} r_{12}\rightarrow r_{13}\rightarrow\cdots\rightarrow r_{1N}\\ \quad\ r_{23}\rightarrow\cdots\rightarrow r_{2N}\\ \quad\quad\ \ \rightarrow\cdots\\ \quad\quad\quad\quad\ \ \rightarrow r_{(N-1)N}\end{Bmatrix}$$

的顺序选择下一次 “扫描”对象 r_{pq}(p、q 值改变), 返回第(1)步重新计算.

上述就是 Jacobi 算法单次迭代过程. 在单次迭代后, 矩阵 R 的上三角元素 r_{pq} 约等于 0, 从而在一定的迭代次数后, 对称矩阵 R 转换为对角矩阵.

当用 Jacobi 算法确定特征向量时, 只需要初始特征向量矩阵为 M 阶单位阵 V, 当 “扫描”元素 r_{pq} 时, 特征向量矩阵 V 将只有第 p、q 列元素变化, 设变化后的特征向量矩阵是 V', 则变化公式如下:

$$\begin{bmatrix}v'_{jp}\\ v'_{jq}\end{bmatrix}=\begin{bmatrix}\cos\theta_{opt} & -\sin\theta_{opt}\\ \sin\theta_{opt} & \cos\theta_{opt}\end{bmatrix}\begin{bmatrix}v_{jp}\\ v_{jq}\end{bmatrix} \quad j=1,2\cdots M$$

特征向量矩阵计算本质就是做 1 次坐标旋转. 当 Jacobi 算法进行下次迭代时, 只需要令 $V=V'$ 即可.

Jacobi 算法中包含大量的乘法和非线性运算, 可以采用 CORDIC 算法. CORDIC(Coordinate Rotational Digital Computer, 坐标旋转数字计算机)算法[3]是 Volder 等人于 1959 年在美国航空控制系统的设计中提出来的, 它是一种用于计算一些常用的基本运算函数和算术操作的循环迭代算法. 其基本思想是用一系列与运算基数相关的角度逼近所需旋转的角度.

EVD 模块设计框图如图 4 所示.

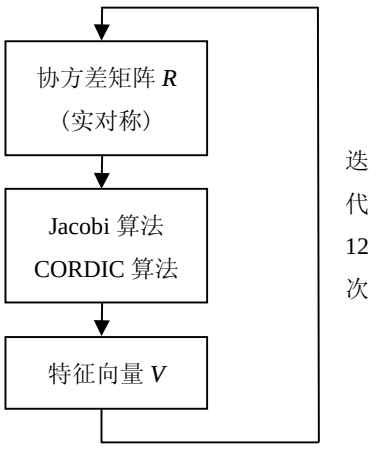

图 4 EVD 模块内部结构

2.3 谱峰搜索模块设计

其功能是根据 EVD 模块输出的特征值和特征向量构造噪声矩阵，然后计算伪谱值．可以把计算 MUSIC 伪谱函数最大值的问题，等价转化为计算 $\|\alpha^H(\theta)U_N\|_2^2$ 最小值问题，由于实数化预处理之后 $\alpha(\theta)$ 和 U_N 的元素均为实数，故可用求实数绝对值来代替求平方，这样既可以提高定点计算时的数值稳定性，又能简化计算[4]，于是将空间谱定义为

$$P_{MUSIC}(\theta)=\sum_{i=K+1}^{4}\left|\alpha^T(\theta)\square v_i\right| \tag{4}$$

式中，$v_i(K<i\le 4)$ 为噪声特征向量．由式(4)可见，可以通过累加 $4-K$ 个实向量内积的绝对值得到空间谱．对于方向矢量，可以把 $\alpha(\theta)(0\le\theta\le 90^\circ)$ 的值进行预先存储．运算过程中只需要从表中读出方向矢量值即可．最后，设计比较单元，根据计算出的谱值找出其 K 个波谷．设计框图如图 5 所示．

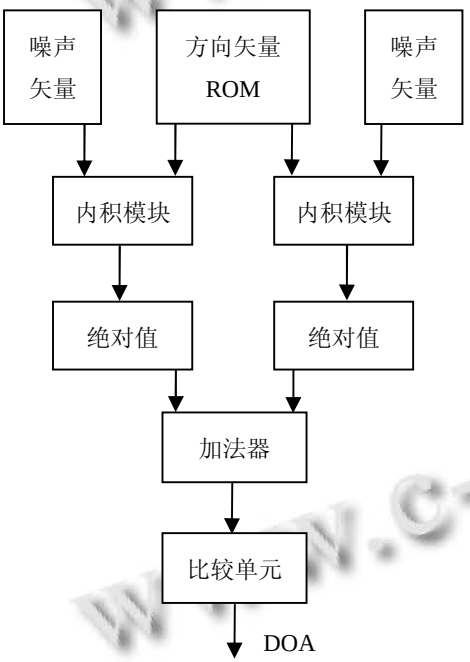

图 5　谱峰搜索模块内部结构

3　MATLAB仿真及FPGA功能仿真

本设计采用 Verilog HDL 作为描述语言，Quartus II 作为开发平台，开发平台运行于 windows 操作系统．开发板的 FPGA 采用 Altera 公司的 Cyclone 系列的 EP3C25Q240C8，该器件有 24624 个 LE，608256 个 RAM 位．

四元均匀线阵，阵元间距 d 为 1 米，2 个输入源信号为单一正弦波，波长 λ 为 2 米，它们的归一化频率 f_1 为 0.015，f_2 为 0.02，方向 θ_1 为 π/3，θ_2 为 π/6，信噪比 snr 为 10dB，快拍数 n 为 256，首先用 MATLAB 进行仿真，仿真图如图 6 所示:

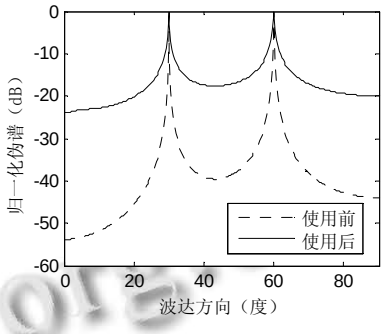

图 6　预处理方法使用前后 MUSIC 算法的 MATLAB 仿真图

当用 FPGA 实现 MUSIC 算法时，三个模块的 Verilog HDL描述分别如下．

首先是协方差模块，即要实现 $R=YY^T$．由于 Y 是 4×256 的矩阵，可以用1024个16位(7位整数, 9位小数)的二进制补码表示，而实现 YY^T 可以按矩阵乘法展开，在VHDL中用乘法和加法实现．这样就求出了 4×4 的实对称协方差矩阵．部分Verilog HDL程序语句如下:

```
//定义存储矩阵Y元素的16位变量，共 4×256 个
//第1行1列
reg signed[15:0] Y1_1 = 16'b1111_1111_1101_0000;
//定义存储 Y 与 Y^T 元素相乘结果的变量，共10×256 个
reg signed[31:0] R11_1 = 0;//第1行1列，第1次相乘.
//定义存储矩阵R元素的32位变量，共10个
reg signed[31:0] R11 = 0;//第1行1列
//以下程序实现乘法累加器功能
//矩阵R的10个上三角元素位置各自有256次相乘
R11_1 = Y1_1*Y1_1;
//把256次相乘结果相加，共10组，得到实对称矩阵R
R11 = R11_1 + R11_2 + … + R11_256;
```

MATLAB计算得到$R(1,3)$ = 925.1, Verilog HDL计算得到$R(1,3)$ = 924.7.

本模块采样率256，时钟50MHz，所以估计模块运算时间为 $1/(50\times 10^6 MHz)\times 256=5.12\mu s$．

其次是EVD模块，即需要求R的特征向量．按照上述CORDIC算法，先计算反正切．这里用到了CORDIC

算法的圆周模式 $y \to 0$，即具有原点的向量 (x_0, y_0) 按如下方式旋转：通过将 y_k 迭代收敛到0，使得向量最后落在 x 轴上，用 $\theta_{k+1} = \theta_k + \xi_i arctg(2^{-k})$ 记录每次旋转的基本角度之和.

CORDIC算法单次迭代首先计算 θ_{opt} 的值，可以令 $y_0 = 2r_{pq}$，$x_0 = r_{pp} - r_{qq}$，$\theta_1 = 0$．当 $y_k = 0$ 时，令 $\theta_{opt} = 1/2 * \theta_{k+1}$ 即可．然后是计算2次坐标旋转，先进行第1次坐标旋转 $W_{pq}R$，它只改变矩阵 R 的第1，2行，可对此两行的每列分别进行坐标旋转．再进行第2次坐标旋转 $W_{pq}RW_{pq}^T$，它只改变矩阵 $W_{pq}R$ 的第1，2列，可对此两列的每行分别进行坐标旋转．计算特征向量时，本质是做1次坐标旋转 VW_{pq}^T，它只改变矩阵 V 的第1，2列，可对此两列的每行分别进行坐标旋转．迭代12次完毕．部分Verilog HDL程序如下：

```
//以下程序是Jacobi算法第1步，计算θopt
// B = arctg(2^-k)*180/π(k=1)
x1_1 <= R11 - R22;//x1_1表示x1的第1次
y1_1 <= R12<<<1;// R12乘以2
if ( y2_1>0 )// ξi =1
    begin
        x3_1 <= x2_1 + (y2_1>>>1);
        y3_1 <= y2_1 - (x2_1>>>1);
        z3_1 <= z2_1 + B;
    end
else// ξi = −1
    begin
        x3_1 <= x2_1 - (y2_1>>>1);
        y3_1 <= y2_1 + (x2_1>>>1);
        z3_1 <= z2_1 - B;
    end
theta_1 <= (z11_1>>>1);//角度值取二分之一
//以下程序是Jacobi算法第2步，第1次坐标旋转
x1_2 <= R11;
y1_2 <= R21;
if ( z2_2>0 )// ξi =1
    begin
        x3_2 <= x2_2 + (y2_2>>>1);
        y3_2 <= y2_2 - (x2_2>>>1);
        z3_2 <= z2_2 - B;
    end
else// ξi = −1
    begin
        x3_2 <= x2_2 - (y2_2>>>1);
        y3_2 <= y2_2 + (x2_2>>>1);
        z3_2 <= z2_2 + B;
    end
```

MATLAB计算得到特征值 $\lambda_3 = 274.2$，Verilog HDL计算得到 $\lambda_3 = 274.1$.

MATLAB计算得到特征向量 $v_3 = [0.8227 \quad 0.1095 \quad -0.3996 \quad -0.3892]^T$，

Verilog HDL计算得到 $v_3 = [0.8231 \quad 0.1099 \quad -0.3991 \quad -0.3898]^T$.

本模块迭代总数为12，单次迭代时钟为13，时钟50MHz，所以估计模块运算时间为：

$1/(50*10^6 MHz) \times 13 \times 12 = 3.12\mu s$.

最后是谱峰搜索模块,部分Verilog HDL程序如下：

```
//计算谱谷值, 2个内积
P_1_1 = ( En_1_1*Alpha_1_1 + En_2_1*Alpha_2_1
        + En_3_1*Alpha_3_1 + En_4_1*Alpha_4_1 );
P_1_2 = ( En_1_2*Alpha_1_1 + En_2_2*Alpha_2_1
        + En_3_2*Alpha_3_1 + En_4_2*Alpha_4_1 );
If ( P_1_1<0 )//负数
    P_1_1 = ␣ P_1_1 + 1; //取绝对值
P_1 = P_1_1 + P_1_2;//绝对值相加
//以下程序段搜索波谷索引值
tempValMin = Pmu_1;//先假设第1个谱值是波谷
if ( P_2 < tempValMin )//如果第2个谱值小于第1个
    begin
        tempValMin = P_2;// 波谷值置第2个谱值
        tempIndexMin = 2;//波谷索引值置2
    end
```

MATLAB计算得到：

$index\_1 = 87$，$index\_2 = 171$

Verilog HDL计算得到：

$index\_1 = 85$，$index\_2 = 172$.

本模块搜索点数为256，时钟50MHz，所以估计模块运算时间为 $1/(50 \times 10^6 MHz) \times 256 = 5.12\mu s$.

4 结语

本文利用有效的实数化预处理算法，特征值分解算法及实用的空间谱定义降低了 MUSIC 算法的实现难度，提高了系统的实时性能．在FPGA实现时，运用合适的定点数长度，既提高了计算精度，又不增加计算量．实验结果表明，计算速度高，精度上能够符合实际要求.

参考文献

1 Schmidt RO. Mutiple Emitter Location and Signal Parameter Estimation. IEEE Trans. on Antennas and Propagation, 1986, 34(3): 276–280.

2 吴仁彪.一种通用的高分辨率波达方向估计预处理新方法.电子科学学刊,1993,15(5):305–309.

3 Volder JE. The CORDIC Trigonometric Computing Technique. IRE Trans. on Electronic Computers, 1959, 8(3): 330–334.

4 徐德琛,刘志文,徐友根.某测向系统中 MUSIC 算法的FPGA 实现.北京理工大学学报,2010,30(9):1107–1111.