Website：ycyk.brit.com.cn

一种适用于快跳频系统的空域抗干扰算法实现

赵吉喆，王 昊，张 炜
（中国人民解放军国防科技大学 长沙 410003）

摘要：针对调频系统中频率变化快、信号环境高度动态化等挑战，本文提出了一种适用于FPGA（现场可编程门阵列）实现的改进型对角加载SMI（采样矩阵求逆）算法，以增强系统在复杂环境下的抗干扰能力。改进型对角加载SMI算法相较于传统的SMI算法，能够更有效地应对低快拍数和复杂干扰环境下的信号处理需求，且对角加载因子的计算复杂度较低。本文简要介绍了改进型对角加载SMI算法的基本原理以及对角加载因子的计算方法，详细阐述了其在FPGA实现上的改进和性能分析。首先，利用高层综合（Vivado HLS）技术完成了低复杂度的对角加载因子的估计，实现了最优权矢量计算并简化了设计流程。随后，将HLS生成的IP核打包并集成至工程中实现波束合成。仿真结果表明：对角加载SMI算法在FPGA平台上可以达到70 dB以上的信干比改善，表现出显著的干扰抑制效果算法还可在4 734个时钟周期内完成抗干扰权值计算与波束合成更新，速度快，能够满足10 000跳每秒以上跳频信号抗干扰的实时处理需求。

关键词：空域抗干扰算法；对角加载；跳频系统
中图分类号：TN975 **文献标志码：**A **文章编号：**2095-1000(2025)01-0081-09
DOI：10.12347/j.ycyk.20241015001
引用格式：赵吉喆，王昊，张炜. 一种适用于快跳频系统的空域抗干扰算法实现[J]. 遥测遥控，2025，46(1)：81–89.

Implementation of A Spatial Anti-Interference Algorithm Suitable for Fast Frequency-Hopping Systems

ZHAO Jizhe，WANG Hao，ZHANG Wei
（National University of Defense Technology，Changsha 410003，China）

Abstract：To address challenges such as fast frequency changes and highly dynamic signal environments in frequency-hopping systems，this paper proposes an improved diagonally loaded SMI（Sample Matrix Inversion）algorithm suitable for FPGA（Field-Programmable Gate Array）implementation to enhance the system's anti-interference capability in complex environments. Compared with the traditional SMI algorithm，the improved diagonally loaded SMI algorithm is more effective in handling signal processing demands under low snapshot numbers and complex interference environments，with a lower computational complexity for the diagonal loading factor. This paper briefly introduces the basic principles of the improved diagonally loaded SMI algorithm and the calculation method of the diagonal loading factor，while providing a detailed explanation of its FPGA implementation and performance analysis. Firstly，low-complexity estimation of the diagonal loading factor is achieved using high-level synthesis（Vivado HLS）technology，enabling the optimal weight vector calculation and simplifying the design process. Subsequently，the IP core generated by HLS is packaged and integrated into the project to implement beamforming. Simulation results show that the diagonally loaded SMI algorithm can achieve an interference-to-signal ratio（SIR）improvement of over 70 dB on the FPGA platform，demonstrating significant interference suppression effects. Additionally，the algorithm completes anti-interference weight calculation，and beamforming update within 4734 clock cycles，ensuring fast performance and meeting the real-time processing requirements for frequency-hopping signals with over 10 000 hops per second.

Keywords: Spatial anti-jamming algorithm; Diagonal loading; Frequency-hopping system
Citation: ZHAO Jizhe, WANG Hao, ZHANG Wei. Implementation of A Spatial Anti-Interference Algorithm Suitable for Fast Frequency-Hopping Systems[J]. Journal of Telemetry, Tracking and Command, 2025, 46(1): 81–89.

基金项目：国家自然科学基金面上项目：基于可重构智能表面的安全高谱效通信技术研究（62171445）
收稿日期：2024-10-15；**修回日期：**2024-12-01

0 引言

通信对抗作为现代战争中信息战的一个重要组成部分，旨在通过对对方通信系统的干扰和破坏来削弱其作战能力，同时保护自己的作战能力。在信息化战争中，争夺制电磁权是双方作战的焦点，空域抗干扰技术逐渐成为通信对抗中的关键一环[1]。空域抗干扰技术通过在空间上对接收信号进行选择性处理，抑制干扰源，保护通信系统的正常工作，从而在复杂的电磁环境中保持通信的可靠性和稳定性。

自适应波束形成技术是实现空域抗干扰的核心技术之一。自适应波束形成算法可以通过自动调整权重来实现所需的空间和频率滤波，增加通信容量和频谱复用率，提高通信质量。其在雷达、无线通信等领域均应用广泛，尤其在对抗复杂电磁干扰的场景下显示出巨大的潜力[2]。它可以在期望信号方向形成主波束，在干扰方向形成零陷，从而达到增强期望信号、抑制干扰的效果。自适应波束形成算法主要包括反馈控制方法与直接求解方法。其中，采样矩阵求逆(SMI)算法是直接求解方法的一种，其通过求解协方差矩阵的逆矩阵来计算最优权矢量。反馈控制方法的常见算法包括最小均方误差(LMS)算法以及递归最小二乘(RLS)算法，LMS算法实现简单且计算量较小，但收敛速度较慢，难以快速适应信号环境的变化[3]；RLS算法收敛速度快，计算复杂度高，其递归更新过程对输入信号的统计特性变化极为敏感，难以满足跳频系统的即时处理需求。相比之下，SMI算法收敛速度快、结构较为简单，能够在跳频系统这种特殊环境下提供较好的性能。跳频系统中的采样点少、频率变化快，导致信道环境瞬息万变，传统算法往往难以在此种环境下发挥作用[4]。SMI算法可以通过较少的样本数迅速收敛以适应对频率和信道快速变化的场景。

虽然SMI算法有以上优势，但是当样本数有限时，小特征值的扰动仍可能导致主瓣偏移、波束畸变等问题[5]，由于跳频系统的信道环境和干扰源快速变化并且快拍数有限，SMI算法在跳频系统的应用仍然面临一定的挑战。

本文采用改进型对角加载这一方式对SMI算法进行改进，该方法的对角加载因子确定过程简便，对角加载值自适应变化，可更好应对小特征值相关的噪声，一定程度上克服本身目标定位误差引起的导向矢量指向偏差、系统中器件引起的误差、信号模型误差等等。FPGA作为一种具有高度并行计算能力和灵活硬件配置特点的硬件平台，为该算法的高效实现提供了理想的基础。本文详细描述了改进型对角加载SMI算法在FPGA平台上的高效实现。与文献[6]全部采用RTL级语言进行书写相比，该方法不仅简化了计算流程，还提升了算法计算速度，使其在实际应用中更具优势。

1 对角加载SMI算法原理

采样矩阵求逆(SMI)算法是一种常用的波束成形算法，它采用线性约束最小方差(LCMV)准则来调整最优权值，使输出噪声方差最小化。LCMV准则要求事先知道有用信号的方向，并根据该方向计算出最佳权值。SMI算法通过从采样快拍中估计阵列相关矩阵，再直接反演Wiener-Hopf(一种用于解决特定类型积分方程的技术)方程来计算自适应权向量[7]。由于SMI算法具有较快的收敛速度，并且无须迭代，可以将数据进行批处理，因而在大规模计算中节约了时间。根据线性约束最小方差准则可以得到该算法的最优权矢量表达式为：

$$\boldsymbol{w}_{\mathrm{SMI}}=\frac{\boldsymbol{R}_{XX}^{-1}\boldsymbol{a}(\theta_0)}{\boldsymbol{a}^{\mathrm{H}}(\theta_0)\boldsymbol{R}_{XX}^{-1}\boldsymbol{a}(\theta_0)} \tag{1}$$

其中，$\boldsymbol{R}_{XX}$表示信号协方差矩阵，$\boldsymbol{a}(\theta_0)$为信号导向矢量，θ_0为期望信号入射角度，$\boldsymbol{R}_{XX}^{-1}$表示信号协方差矩阵的逆。然而，在实际应用中，无法准确得到信号协方差矩阵，一般通过采样数据得到估计协方差矩阵：

$$\tilde{\boldsymbol{R}}_{\mathrm{XX}}=\frac{1}{K}\sum_{k=1}^{K}\boldsymbol{x}(\boldsymbol{k})\boldsymbol{x}^{\mathrm{H}}(\boldsymbol{k}) \tag{2}$$

则最优权矢量可以表示为

$$\boldsymbol{w}_{\mathrm{SMI}}=\frac{\tilde{\boldsymbol{R}}_{XX}^{-1}\boldsymbol{a}(\theta_0)}{\boldsymbol{a}^{\mathrm{H}}(\theta_0)\tilde{\boldsymbol{R}}_{XX}^{-1}\boldsymbol{a}(\theta_0)} \tag{3}$$

SMI算法在信噪比变化和低快拍数等因素的影响下，可能会出现亚噪声升高、主瓣偏移和波束失真等问题。这些问题的根源在于采样协方差矩阵的估计值与理论值之间的误差。因此，SMI性能优化的关键在于信号协方差矩阵的优化，常见的优化方法是通过对协方差矩阵进行加权校正。当样本数量太小且协方差矩阵未得到充分估计时，SMI算法中主波束失真成为主要误差形式。此外，当阵列采样协方差矩阵不包含期望信号或期望信

号较小时，采样快拍数是影响SMI算法性能的关键因素。为了应对这些问题，本文采用了改进型对角加载的方式提升算法的性能。该方法可以通过估计所获得的协方差矩阵来校正协方差矩阵，同时对角加载量的确定更为简便，经过对角加载之后该算法的权矢量变为：

$$\boldsymbol{w}_{\mathrm{LSMI}}=\frac{(\tilde{\boldsymbol{R}}_{XX}+L\boldsymbol{I})^{-1}\boldsymbol{a}(\theta_0)}{\boldsymbol{a}^{\mathrm{H}}(\theta_0)(\tilde{\boldsymbol{R}}_{XX}+L\boldsymbol{I})^{-1}\boldsymbol{a}(\theta_0)} \tag{4}$$

其中，$\boldsymbol{I}$为单位阵，L为加载量；定义方向图函数为$G(\boldsymbol{w},\theta)=\boldsymbol{w}^{\mathrm{H}}\boldsymbol{a}(\theta)$。此时，其方向图表达式为：

$$G(\boldsymbol{w}_{\mathrm{LSMI}},\theta)=G(\boldsymbol{a}_q,\theta)-\sum_{i=1}^{M-1}\left(\frac{\hat{\lambda}_i-\hat{\lambda}_M}{\hat{\lambda}_i+L}\boldsymbol{a}_q^{\mathrm{H}}\boldsymbol{u}_i\right)G(\boldsymbol{u}_i,\theta) \tag{5}$$

其中，$\boldsymbol{a}_q$即为$\boldsymbol{a}(\theta_0)$，λ_i为协方差矩阵对应特征值，λ_M为信号子空间对应特征值，u_i为信号子空间对应特征向量。当L不太大时，对于干扰对应的大特征值：

$$\frac{\hat{\lambda}_i-\hat{\lambda}_M}{\hat{\lambda}_i+L}\approx 1 \tag{6}$$

对于小特征值：

$$\frac{\hat{\lambda}_i-\hat{\lambda}_M}{\hat{\lambda}_i+L}\ll\frac{\hat{\lambda}_i-\hat{\lambda}_M}{\hat{\lambda}_i} \tag{7}$$

通过上述分析可以发现，对角加载后的波束在小特征值的情况下抑制波束形成过程中的扰动，从而改善了波束形成方向图的畸变问题。当加载量较大时，波束形成抗干扰性能下降；而当加载量过小时，干扰不能得到有效抑制[8]。为了更好地确定加载量的选取，可以将对角加载后的矩阵表示为如下形式：

$$\tilde{\boldsymbol{R}}_{\mathrm{XX}}=\boldsymbol{R}_{\mathrm{XX}}+\varepsilon\boldsymbol{B}+L\boldsymbol{I} \tag{8}$$

其中，$\boldsymbol{B}$为均值为0、标准差为1的误差矩阵，ε为估计误差常数，L为待确定的对角加载量。求逆可得：

$$\tilde{\boldsymbol{R}}_{XX}^{-1}=(\boldsymbol{R}_{XX}+L\boldsymbol{I})^{-1}\left[\boldsymbol{I}+\varepsilon\boldsymbol{B}(\boldsymbol{R}_{XX}+L\boldsymbol{I})^{-1}\right]^{-1} \tag{9}$$

由矩阵求逆定理可化简，上式可约等于为

$$(\boldsymbol{R}_{XX}+L\boldsymbol{I})^{-1}\left\{\boldsymbol{I}-\frac{\varepsilon}{L+\delta_n^2}\boldsymbol{B}[\boldsymbol{I}-\boldsymbol{A}(\boldsymbol{A}^{\mathrm{H}}\boldsymbol{A}+(\delta_n^2+L)\boldsymbol{A}^{-1})^{-1}\boldsymbol{A}^{\mathrm{H}}]\right\} \tag{10}$$

其中，δ_n^2表示接收信号中噪声分量的方差，$\boldsymbol{A}$为投影分解的矩阵。由公式(10)可以看出，自适应波束形成性能的降低是由第二项引起的，为了维持良好的波束形成效果，令

$$\frac{\varepsilon}{L+\delta_n^2}<1 \tag{11}$$

为了进一步研究公式(10)第二项对波束畸变产生的影响，令$L=0$,设置ε/δ_n^2的值分别为0，0.5，2。观察公式(10)第二项对波束形成的影响，仿真结果如图1所示。

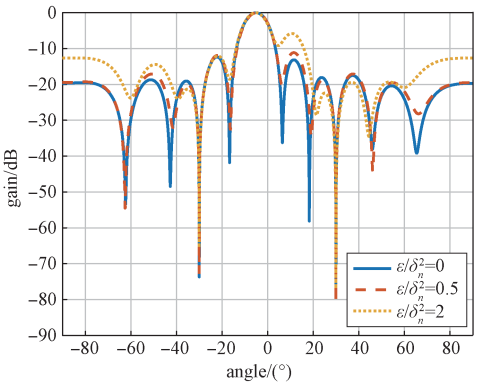

图1　不同误差常数对波束畸变的影响

Fig. 1　The impact of different error constants on beam distortion

上述仿真表明，ε/δ_n^2为0时，由于公式(10)第二项分量不存在，此时可以得到最优波束；当设置ε/δ_n^2为0.5时，波束出现少许畸变，但仍近于最优波束；当ε/δ_n^2增大至2时，波束旁瓣显著抬高，波束畸变进一步加重，验证了公式(11)在实际应用中的可行性。

由公式(11)可以看出对角加载量L选取的最小值应该为ε，也就是估计矩阵与实际矩阵之间的误差，由于对角线元素通常反映了系统的重要特征和信号的主要成分，因此其标准差能够有效表征估计矩阵在信号处理过程中与实际矩阵之间的偏差，这里用对角线元素的标准差代替估计矩阵与实际矩阵之间的误差。即对角加载量应为

$$L=\mathrm{std}(\mathrm{diag}(\tilde{\boldsymbol{R}}_{\mathrm{XX}})) \tag{12}$$

对角加载SMI算法的计算流程图如图2所示。

对上述方法进行仿真测试，仿真参数设置如表1所示。

对两种算法得到的波束形成图进行观察，如图3所示，可以看出两种算法均能在干扰来向形成零陷，但没有进行对角加载的SMI算法出现了旁瓣被抬高等问题。比较两种算法的输出信干比(SIR)可得SMI算法输出信干比为18.66 dB，而对角加载后的输出信干比为19.44 dB，对干扰抑制效果更好，验证了对角加载算法的有效性。

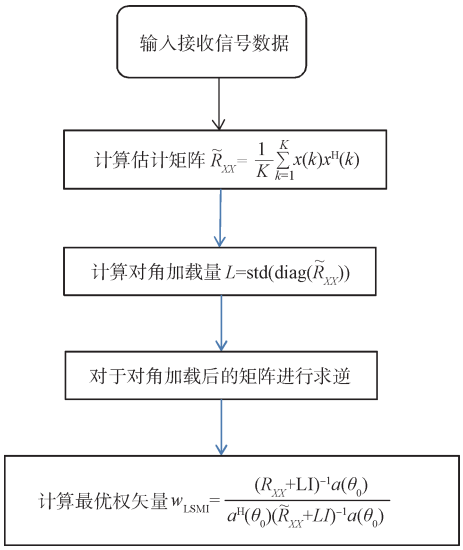

图 2 对角加载 SMI 算法流程图

Fig. 2 Flowchart of the diagonally-loaded SMI algorithm

表 1 仿真参数设置

Table 1 Simulation parameter settings

参数名称	数值	参数名称	数值
输入信息比	−39.09 dB	干扰角度	±30°
快拍数	300	阵元数量	6
阵元间距	半波长	输入信噪比	−10 dB

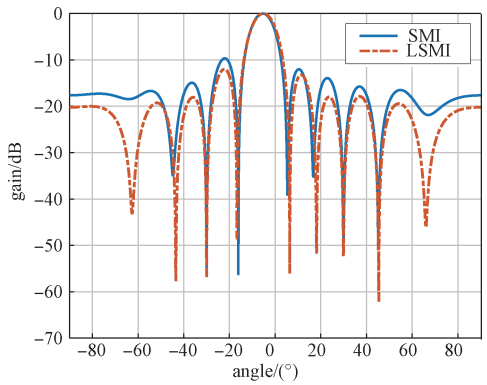

图 3 SMI、LSMI 算法波束形成图

Fig. 3 Beamforming diagrams for SMI and LSMI algorithms

2 对角加载 SMI 算法的 FPGA 实现

2.1 HLS 生成计算最优权值的 IP 核

由上述分析可知，该算法在 FPGA 的实现可以分为求解权最优权矢量以及波束合成两部分。传统实现方法往往全部使用 RTL(寄存器传输级)级语言(如 Verilog)进行设计，这增加了开发复杂度和难度。因此，本文选择使用高层次综合(HLS)工具生成硬件 IP 核来进行最优权矢量的计算，减少设计复杂性，并实现硬件加速。后续将生成的 IP 核例化到工程文件后进行波束合成。相比于传统的 FPGA 实现权值求解，使用 HLS 可以大大简化矩阵求逆这一过程的复杂度[9]。权值求解模块采用的是 AXI14-Stream 接口，AXI4-Stream 是一种高效的数据传输协议，专为在 FPGA 中实现高吞吐量数据流传输而设计。它不需要地址握手，仅用于源端和目标端之间的纯数据流传输，非常适合连续的数据处理。在 SMI 算法的实现中，该接口适用于传输接收信号、信号导向矢量以及最优权值向量。AXI4-Stream 的流式传输机制确保了多个数据通道可以并行传输，减少延迟，确保实时性。该模块的输入包括来自 6 个通道接收到的信号、期望信号的导向矢量以及各个有效信号。每个阵元的输入为 1 024 个定点数数据，每个数据由 32 位组成，其中高 16 位用于存储数据的实部，低 16 位用于存储虚部。导向矢量输入为 64 位双精度浮点数数据，高 32 位存储虚部，低 32 位存储实部。由于实际环境中难以直接获取精确的信号协方差矩阵，通常使用时间平均的协方差矩阵作为替代。

在 HLS 设计中，整体流程图如图 4 所示，首先对输入信号与其共轭进行逐点相乘，然后将结果除以样本数。为简化计算，采用位移操作来实现除法，转化得到定点数形式的协方差矩阵。该矩阵的对角线元素为实数，上下三角矩阵为共轭对称结构。接着，按照上一章讨论的对角加载方法，在主对角线元素上添加对角加载值。对角加载的流程图如图 5 所示，首先定义了一个函数用于计算标准差，将矩阵对角线元素的实部输入到该函数中可以计算得到对角加载因子，之后将计算得到对角加载因子添加到矩阵当中，完成对角加载。

由于经过对角加载处理后的矩阵依然是对称正定矩阵，因此可以采用 Cholesky 分解法对该矩阵进行求逆运算。Cholesky 分解是一种将正定矩阵分解为下三角矩阵及其共轭转置的分解方法，广泛用于求解对称正定矩阵的线性方程组和矩阵求逆问题。对于一个给定 $n\times n$ 对称正定矩阵 $\boldsymbol{A}$，Cholesky 分解可以表示为 $\boldsymbol{A}=\boldsymbol{L}\boldsymbol{L}^{H}$，$\boldsymbol{L}$ 是一个 $n\times n$ 的下三角矩阵，$\boldsymbol{L}^{H}$ 表示该下三角矩阵的共轭转置，分解的过程包括按元素递归地计算 $\boldsymbol{L}$ 的各个元素[10]。对于对角线元素 L_{ii}，计算公式为：

$$L_{ii}=\sqrt{A_{ii}-\sum_{k=1}^{i-1}L_{ik}^{2}} \tag{13}$$

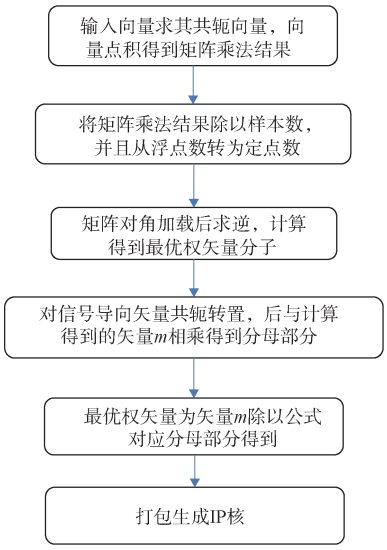

图 4　HLS 实现整体流程图

Fig. 4　Overall flowchart for HLS implementation

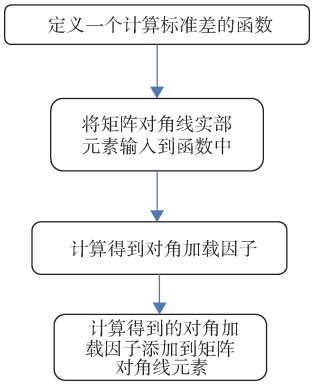

图 5　对角加载的流程

Fig. 5　Flowchart of diagonal loading

对于非对角线元素，$L_{ij}(j<i)$，计算公式为：

$$L_{ij}=\frac{1}{L_{jj}}\left(A_{ij}-\sum_{k=1}^{j-1}L_{ik}L_{jk}\right) \tag{14}$$

这一过程从矩阵的左上角开始，逐步递推计算下三角矩阵的每个元素，直到整个下三角矩阵被求出。

矩阵求逆以及计算最优权矢量分子的实现流程图如图 6 所示，首先调用 cholesky_top 函数计算得到下三角矩阵 $\boldsymbol{L}$，后对得到的下三角矩阵进行共轭转置操作得到矩阵 $\boldsymbol{L}_t$。之后接收矩阵 $\boldsymbol{L}$，并通过遍历矩阵的行和列，对其每个元素执行共轭操作，并将结果存储到矩阵 $\boldsymbol{L}_t$，使用了 HLS 指令#pragma HLS UNROLL，以完全展开内部循环，加速硬件实现。此外，这一指令确保矩阵在硬件中按行进行了完全分块，这进一步提高了数据访问的并行

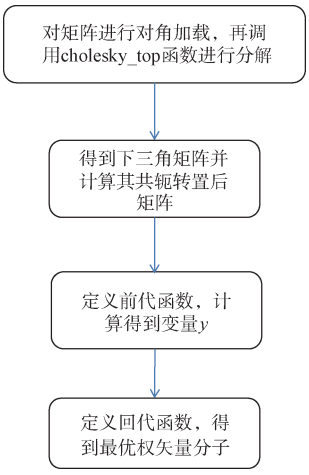

图 6　矩阵求逆流程图

Fig. 6　Flowchart for matrix inversion

性。之后定义了一个实现前代计算的前代函数，该函数用于求解线性方程组 $\boldsymbol{Ly}=\boldsymbol{a}_q$ 中的变量 $\boldsymbol{y}$，$\boldsymbol{a}_q$ 指的是输入的信号导向矢量，即 $\boldsymbol{a}(\theta_0)$。这一函数在 HLS 中是通过内外两层循环来实现的，可以实现方程

$$\boldsymbol{L}[i][i]\cdot\boldsymbol{y}[i]+\sum_{j=0}^{i-1}\boldsymbol{L}[i][j]\cdot\boldsymbol{y}[j]=\boldsymbol{a}_q[i] \tag{15}$$

从而可以得到变量 $\boldsymbol{y}$ 的值。定义了一个执行回代计算的函数用于求解线性方程组 $\boldsymbol{L}_t*\boldsymbol{m}=\boldsymbol{y}$，其中 $\boldsymbol{m}$ 是定义的中间变量，也是式子(4)权矢量的分子部分。回代过程也由层循环组成，外层循环从最后一行开始，向上遍历每一行最后实现了方程。

$$\boldsymbol{L}_t[i][i]\cdot\boldsymbol{m}[i]+\sum_{j=i+1}^{N-1}\boldsymbol{L}_t[i][j]\cdot\boldsymbol{m}[j]=\boldsymbol{y}[i] \tag{16}$$

通过式(16)得到变量 $\boldsymbol{m}$ 的值。上述定义的两个函数都使用了 HLS 指令#pragma HLS PIPELINEII=8，使得外层和内层循环的计算可以在 8 个时钟周期内流水线执行，优化硬件实现的性能。

之后定义了函数用于实现信号导向矢量的转置、复数向量的乘积运算等等，得到了共轭转置之后的信号导向矢量，通过复向量乘法得到了式子(4)的分母部分。最后权矢量的值为向量 $\boldsymbol{m}$ 除以计算得到的式子(4)分母。这一过程使用了指令#pragma HLS ARRAY_PARTITION，#pragma HLS INTERFACE axis register both port 等等，将数组分割成多个小数组，使得计算出来的权值中每个数组元素都会被映射到独立的硬件资源上。这种分割允许并行访问数组的每个元素，从而提高了并行计算的性能，同时该指令指定了输入输出接口

的类型为AXI4-Stream，减少了等待时间并确保数据传输的稳定性[11]。这些指令结合使用，确保了数据流的高效传输和处理，大大降低了数据延迟。最终，HLS工具将代码、IP核的接口定义、寄存器映射等打包为一个独立的IP核模块，生成相应的IP核描述文件例化到后续工程当中。

2.2 计算数据输出

如图7所示为计算数据输出的流程，在将生成的IP核例化到整个文件后，定义了一个状态机，用于控制权向量数据的读取、处理和输出。首先使用复位信号使得系统进入初始状态，在复位信号解除后，状态机根据设置的触发信号进行触发，进入数据读取和处理阶段。当对应ready信号为高时，状态机通过多个状态(从状态0到状态6)逐步处理输入的信号导向矢量。其中状态0到5：在每个时钟周期内，128位信号导向矢量共6个，被依次写入信号导向矢量，同时通过设置相应的有效信号表明有效。状态机不断递增状态值以读取和传输下一部分数据。当所有数据传输完毕时，对应有效信号被置低，表示传输结束，系统进入等待状态。

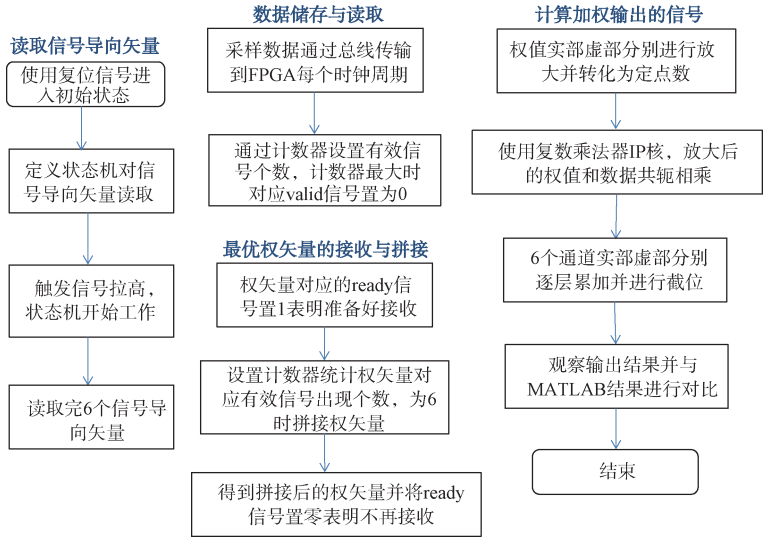

图7 波束合成整体流程图

Fig. 7 Overall flowchart of beamforming

对于接收信号的数据控制，接收到的采样数据通过总线传输到FPGA每个时钟周期中，每个时钟读取一组数据。同时设置了一个计数器，当新的采样数据到达时，计数器同步递增，数据对应的有效信号设置为1直到读取了每个通道的1 024个定点数数据。设置在计数器达到预设最大值时，传输结束，对应有效信号被置低。该机制在计数器达到最大值后自动复位，为下一轮的ADC数据传输作好准备。

之后进行权矢量的接收与拼接，首先将对应的ready信号拉高表明准备接收输出的权矢量，同时设置另一个计数器，当该信号有效且接收计数器小于6时，系统将接收到的128位权向量数据存储到新的数组中，并递增计数器以存储下一个数据。当计数器值为6时，系统将接收到的权向量数据拼接为一个768位的信号，并将ready信号拉低，表示权向量接收完成。为了提高硬件资源的利用效率，在权值与输出信号进行共轭相乘之前，首先对接收到的浮点数权值矢量进行转换。浮点数计算虽然精度高，但在FPGA等硬件实现中资源消耗较大[12]。因此，选择将浮点数权值转换为定点数表示。考虑到权值矢量通常为小于1的小数，为了在定点数表示中尽可能多地保留其有效位，采取了先将权值放大2^{28}倍，再进行定点数转换的策略。这种方法不仅有效减少了硬件资源开销，还能在转换过程中尽量保持数据的精度，确保后续运算的准确性。实部和虚部分别通过各自的模块进行转换，输出定点数格式的复数权值。在完成复数乘法后，采用复数乘法器的IP核对权值和输入的数据流相乘，并在IP核中设置截取后16位的位宽，这一操作意味着将扩大后的权值缩小了2^{16}倍。对于计算出来的6个通道的结果进行逐层累加，每次仅对两个输出结果的实部和虚部分别进行累加，这一方式减少硬件资源的使用，并有效降低延迟，

同时再对最后的输出结果截取 12 位的位宽，确保了数据没有被放大，同时有效保证了输出结果的精度。

3　仿真与验证

3.1　验证方案

使用天线阵列和多通道模数转换器(ADC)采集真实的信号数据，以验证 SMI 算法在 FPGA 上的实现性能。首先，通过天线阵列接收来自多个方向的信号，包括期望信号和干扰信号。天线阵列的输出经过多通道 ADC 进行数字化采样，得到多通道的时域数据。仿真参数设置如表 2 所示。

测试过程中，输入信号由期望正弦信号、接收机通道热噪声和人为施加的单频干扰组成，期望信号入射角 10°，干扰信号入射角 30°，以此来验证设计的鲁棒性和功能正确性。

表 2　硬件测试参数设置

Table 2　Hardware test parameter settings

参数名称	数值	参数名称	数值
输入信息比	−44.91 dB	干扰数量	1
快拍数	1 024	通道数量	6

3.2　结果对比分析

之后对结果进行验证，并将其与 MATLAB 结果进行对比。首先，对最优权矢量的数值进行比较。由于 FPGA 输出结果为以 16 进制表示的双精度浮点数，使用 MATLAB 将各个通道 16 进制的双精度浮点数数据转换为可读数值格式，便于进一步分析。结果对比如图所示，其中图 8 以及图 9 分别为 MATLAB 计算所得最优权矢量以及 FPGA 计算所得最优权矢量，图 10 为 MATLAB 将这些 16 进制的双精度浮点数数据转换为可读数值格式的结果：

```
-0.079432880453135 + 0.1328464533531781
 0.143706093956615 - 0.0738474641334161
 0.005710360231784 + 0.2224135238718211
 0.152234179872610 - 0.1911166171811211
 0.118746200247958 + 0.0464799569616431
 0.126126500423900 - 0.0530082131821611
```

图 8　MATLAB 计算所得最优权矢量

Fig. 8　Optimal weight vector calculated by MATLAB.

图 9　FPGA 输出结果图

Fig. 9　Output result graph from FPGA

```
-0.079612252917440 + 0.132855099650505i
 0.143778567534486 - 0.073663318331757i
 0.005765151898388 + 0.222515914276350i
 0.152123035609364 - 0.191159886212748i
 0.118739667212932 + 0.046448059685424i
 0.125927288839044 - 0.053027191502990i
```

图 10　FPGA 输出结果转为 10 进制的绘制结果图

Fig.10　Output results from FPGA converted to decimal

由于 FPGA 输出的结果为 64 位 16 进制数，其中高 32 位代表实部，低 32 位代表虚部，因此使用 MATLAB 对 FPGA 计算得到的权值进行转换并进行观察。从上图结果可以看出，FPGA 仿真结果与 MATLAB 相比，误差主要集中在有效数字的第四位，但整体结果依然较为准确。导致该误差的原因可能在于，输入到 FPGA 的数据需要等待对应信号有效后才能开始计算，进而导致生成的估计矩阵数据顺序与 MATLAB 不一致。此外，FPGA 在复数计算中采用的是单精度浮点数，而 MATLAB 通常使用双精度浮点数。对于矩阵求逆等高精度运算，精度差异进一步放大，导致误差增加。再观察抗干扰之后的波形输出。由于输入输出的信号均为复信号，这里分别对输出信号实部虚部进行观察，图 11 为 MATLAB 仿真得到的处理后信号实部与未处理之前的输入信号实部的波形图，图 12 为 FPGA 显示的抗干扰之后的数据实部虚部分别的波形图结果。

在 MATLAB 中，选取了信号的实部进行观察。可以看出，输出信号同样呈现正弦波形，与输入信号保持一致。尽管由于输入信噪比较低，输出信号中出现了一定的毛刺现象，然而，这些毛刺并不影响对角加载后 SMI 算法的有效性验证。通过观察 FPGA 的输入输出波形，可以看到其输出结果的实部和虚部也均为正弦波，同时通过 ILA (集成逻辑分析器)将计算得到的数据导出进行频谱分析，可以计算得到抗干扰后的输出信干比为 30.41 dB，相比于输入信干比可以达到 70 dB 以上

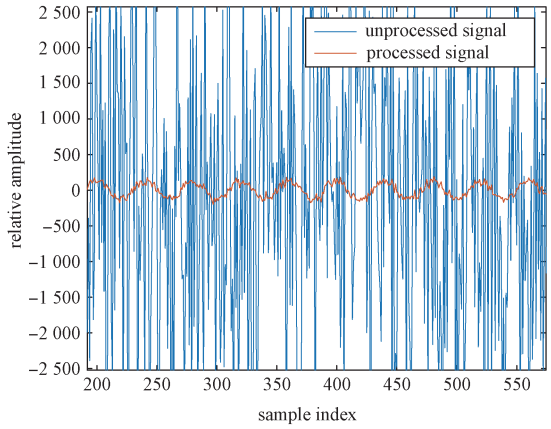

图 11　MATLAB 仿真抗干扰前后信号波形图

Fig. 11　Signal waveforms before and after interference suppression

的信干比增益，抗干扰性能优良。对该算法消耗的时钟个数进行统计，可以在 3 676 个时钟周期内完成抗干扰权值计算，在 4 734 个时钟周期内完成抗干扰权值计算与波束合成更新。本文模块的工作时钟是 192 MHz，权值计算时间为 19.15 μs，可以支持 10 000 跳每秒的跳频信号的快速收敛。

3.3　资源消耗

将统计得到的资源消耗通过表 3 的形式展示出来，可以看出最优权矢量计算模块是资源消耗的主要部分，这是由于该模块涉及大量的复杂计算，如复数运算和矩阵求逆，导致其在逻辑单元和 DSP 资源上占用较高。同时，该模块还占用了少量的 Block RAM，用于存储中间数据或暂存运算结果。其次，输出计算结果模块的资源消耗较为轻量化，该模块使用了 12 个浮点数乘法器，6 个复数乘法器以及 10 次加法，主要承担一些简单的算术操作，对逻辑和计算资源的依赖相对较低。ILA 输出观测模块的资源使用主要集中在 Block RAM 上，该模块主要用于数据存储和输出观测。相比于传统的 CPU 或 GPU 实现，FPGA 设计在计算性能和功耗上具有明显优势，尤其是在对实时性和低延迟要求较高的应用中。此外，与文献[13]采用纯粹的 Verilog 设计相比，HLS 生成代码在资源消耗上差别不大，但简化了设计流程，提升了速度，算法在 4 734 个时钟时计算得到了最优权矢量的值以及波束合成的更新，计算速度较快。

表 3　不同模块资源消耗表

Table 3　Resource consumption of different modules

模块名称	CLBLUTs	CLB Registers	Block RAM	DSPs
最优权矢量计算	12.76%	8.03%	2.99%	14.83%
输出计算结果	1.16%	1.35%	0.00%	2.93%
ILA 输出观测	0.46%	0.41%	3.33%	0.00%

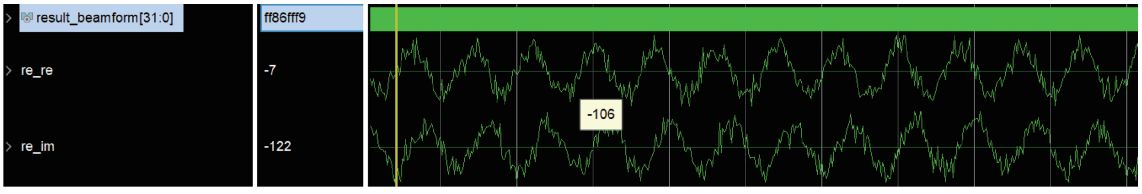

图 12　FPGA 仿真抗干扰后信号波形图

Fig. 12　Signal waveform after interference suppression in FPGA simulation

4　结束语

本文实现并验证了一种适用于快跳频系统的空域抗干扰算法，成功展示了其在跳频系统中的有效性。通过对 FPGA 仿真平台和 MATLAB 仿真结果的比较，分析了不同实现方法之间的数值精度差异与性能表现。在 FPGA 设计中，利用 HLS 生成了权矢量计算模块，并通过 Verilog 实现了复数运算及波束形成的硬件架构。仿真结果表明，该算法 1 024 快拍数的条件下达到 70 dB 以上的信干比增益，抗干扰性能良好。同时，资源消耗分析表明，系统在不同资源的使用上实现了良好的平衡，与纯粹 RTL 级设计相比资源消耗差别不大。FPGA 平台可以在保证一定精度的前提下，在 19.15 μs 完成抗干扰权值计算，显著提升了运算速度和并行处理能力，充分验证了该抗干扰算法在快速跳频系统中的有效性。未来可考虑空时联合抗干扰处理来进一步提升抗干扰能力。

参考文献

[1] 张羽洁, 王炜, 熊政, 等. 俄乌冲突中的联合通信与通信对抗探析[J]. 中国军转民, 2024, 1(1): 197-199.

[2] 陈明建, 胡振彪, 陈怀进, 等. 稳健自适应波束形成算法综述[J]. 探测与控制学报, 2023, 45(5): 7-15, 21.
CHEN Mingjian, HU Zhenbiao,CHEN Huaijin, et al. Review of robust adaptive beamforming algorithms[J]. Journal of Detection & Control, 2023, 45(5): 7-15, 21.

[3] 刘千里. 基于FPGA的逆QR分解SMI算法的并行实现方法[J]. 计算机工程与应用, 2012, 48(26): 71-75, 161.
LIU Qianli. Implementation method of inverse QR decomposition SMI algorithm parallel processing based on FPGA[J]. Computer Engineering and Applications, 2012, 48(26): 71-75, 161.

[4] 王世练. 认知通信抗干扰[M]. 北京: 国防工业出版社, 2023.

[5] 张小飞, 汪飞, 徐大专, 等. 阵列信号处理的理论和应用[M]. 北京: 国防工业出版社, 2010.

[6] 张国义. 自适应数字波束形成及抗干扰研究[D]. 江苏: 南京理工大学, 2021.

[7] DUGGINENI C, KURAPARTHI S, MEENAKSHI K, et al. Comparison of RLS, LMS and SMI algorithms for smart antennas[C]//2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India. New York: IEEE, 2021: 599-602.

[8] FAN Z, FU X B. An improved SMI algorithm based on sampled data[C]//2022 2nd International Conference on Frontiers of Electronics, Information and Computation Technologies (ICFEICT), Wuhan, China, 2022. New York: IEEE, 2022: 217-221.

[9] 李丽, 张巍. 改进Cholesky分解算法的设计与FPGA实现[J]. 电讯技术, 2020, 60(7): 845-849.
LI Li, ZHANG Wei. Design and FPGA implementation of an improved cholesky factorization algorithm[J]. Telecommunication Engineering, 2020, 60(7): 845-849.

[10] 凌元, 韩文俊, 孙健. 基于HLS的矩阵求逆算法设计优化[J]. 电子技术与软件工程, 2021(22): 93-96.

[11] 杨永舟, 黄秀琼. 基于HLS的复数矩阵QR分解求逆算法的实现与优化[J]. 电子技术, 2021, 50(7): 74-78.
YANG Yongzhou, HUANG Xiuqiong. Realization and optimization of inverse algorithm of complex matrix QR decomposition based on HLS[J]. Electronic Technology, 2021, 50(7): 74-78.

[12] 郑张笑. 可配置ADBF权值计算IP核的FPGA设计与实现[D]. 南京: 南京大学, 2018.

[13] 田佳月. 基于FPGA的采样矩阵求逆算法的硬件实现[D]. 哈尔滨: 哈尔滨工程大学, 2013.

[作者简介]

赵吉喆　2001年生，硕士。

王　昊　1989年生，博士，副研究员。

张　炜　1972年生，博士，教授。

(本文编辑：潘三英)

(英文编辑：赵尹默)