·通信与信息工程·

基于FPGA的复数长方阵SVD算法

阎 波，张威威，林水生

(电子科技大学通信与信息工程学院 成都 611731)

【摘要】在OFDM和MIMO系统中普遍使用长方形矩阵复数奇异值分解运算。针对传统算法运算量大，迭代次数多的问题，提出了一种基于householder和双边Jacobi的混合优化算法。该算法首先通过householder变换将矩阵化解为二对角矩阵；然后提取2×2复矩阵；再进行改进型复数双边Jacobi变换。兼具有QR算法的高精度和Jacobi算法的低硬件实现成本的优点。给出了2×8的CSVD的FPGA硬件实现方案并进行了板级测试。测试结果表明，该混合优化算法较传统算法在硬件资源上节省26%，延时缩短10倍，在同等位宽下计算精度至少提高了一个数量级。

关 键 词 复数奇异值分解； 可编程逻辑阵列； householder； Jacobi； 长方矩阵

中图分类号 TP33； TN4 文献标志码 A doi:10.3969/j.issn.1001-0548.2015.04.001

Singular Value Decomposition Algorithm of Rectangular Complex Matrix Based on FPGA

YAN Bo, ZHANG Wei-wei, and LIN Shui-sheng

(School of Communication and Information Engineering, University of Electronic Science and Technology of China Chengdu 611731)

Abstract Rectangular matrix complex singular value decomposition (CSVD) is widely used in orthogonal frequency division multiplexing (OFDM) and multiple input and multiple output (MIMO) systems. In view of large iteration computation of traditional algorithms, a householder and Jacobi based mixed optimized algorithm is proposed which diagonalizes a general complex matrix and carry out an improved complex two-sided Jacobi transform. This method combines the advantages of high precision of QR and the simple hardware structure of Jacobi. A 2×8 CSVD design is implemented on field programmable gate array (FPGA) by using MATLAB simulation and Xilinx platform. Compared with traditional algorithms, the mixed optimized algorithm saves 26% hardware resources, shortens delay time by 10 and improve the accuracy of calculation at least one order of magnitude under the same bit width.

Key words CSVD; FPGA; householder; Jacobi; rectangular complex matrix

CSVD在OFDM通信系统中的快衰落信道估计[1]、MIMO系统中的预编码[2]、信号处理[3]、图像处理[4]等领域中得到了广泛的应用。CSVD在工程中应用的关键在于数值计算，为了保证计算精度，通常会采用迭代运算的方法。文献[5]给出了一种直接复数双边Jacobi方法，该方法以一个复数 2×2 为基本单元，并将其对角化，避免扩展矩阵维度。然而一个基本单元算法复杂，耗费的硬件资源较多，也需要多次迭代。文献[6]提出一种复转实的双边Jacobi方法，该方法将矩阵行列维度均扩大一倍，以一个实 2×2 为基本单元，优点是基本单元简单，但是迭代次数增大，资源、延时和精度受限，实时性差，且这些算法对于长方矩阵具有明显的缺点。

本文提出了基于householder和双边Jacobi的混合优化算法，能有效地解决长方复矩阵奇异值分解的硬件实现和精度等问题。首先介绍了CSVD传统的算法即QR算法和双边Jacobi算法，并指出其对于处理长方阵的局限性。在此基础上提出一种混合优化算法，给出其具体运算步骤，并将该方法与现有常见算法性能比较，具体说明混合优化算法在硬件实现上的优势。最后以 2×8 复矩阵为例，给出了混合优化算法在FPGA上的实现及其测试结果。

收稿日期：2014－03－11 ；修回日期：2015 －03－11

基金项目：国家自然科学基金(61301155,61176025)；中央高校基本科研业务费专项资金(ZYGX2012J003)

作者简介：阎波(1973－)，女，教授，主要从事通信信号处理，无线通信系统、通信集成电路设计等方面的研究.

1 复数矩阵奇异值分解

设 $\boldsymbol{M}$ 是一个 $m\times n$ 的复矩阵，如果存在一个分解有：

$$\boldsymbol{M}=\boldsymbol{U}\times\boldsymbol{S}\times\boldsymbol{V}^{\mathrm{H}} \tag{1}$$

式中，$\boldsymbol{U}$ 是 $m\times m$ 酉矩阵；$\boldsymbol{S}$ 是半正定 $m\times n$ 对角矩阵；$\boldsymbol{V}$ 是 $n\times n$ 酉矩阵；$\boldsymbol{V}^{\mathrm{H}}$ 为 $\boldsymbol{V}$ 的共轭转置。这样的分解就称为复矩阵 $\boldsymbol{M}$ 的奇异值分解。$\boldsymbol{S}$ 对角上的值为 $\boldsymbol{M}$ 的奇异值。在工程实际中，只需要求得 $\boldsymbol{V}$ 矩阵即可。

1.1 传统长方阵CSVD算法

目前传统复数长方阵的奇异值分解主要有两种较为常用：QR迭代算法和双边Jacobi算法。QR迭代算法首先采用householder变换将一般复矩阵化为二对角矩阵；然后采用Givens变换的方法交叉将非对角非零元素消零，如图1所示，$\boldsymbol{S}$,$\boldsymbol{T}$ 分别代表行和列Givens矩阵，这一过程又称为“驱逐出境”；经过多次迭代将非对角化为零，最终达到收敛条件。

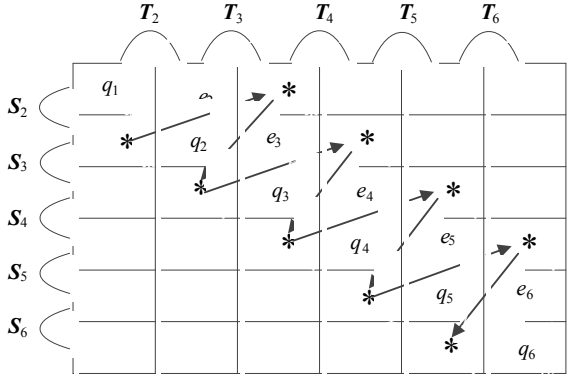

图1 QR迭代“驱逐出境”示意图

双边Jacobi算法常见有：复转实双边Jacobi算法和直接复数双边Jacobi算法。两种算法的本质都是采用Brent-Luk-Van Loan(BLV)[7]脉动阵列迭代算法。所不同的是前者是将 $n\times n$ 复矩阵转换为 $2n\times 2n$ 的实矩阵，以 2×2 矩阵为基本单元；而后者是直接对复 2×2 矩阵进行对角化。

对于复转实算法，首先将一般复矩阵 $\boldsymbol{M}$ 转化为共轭对称矩阵 $\boldsymbol{C}$，即 $\boldsymbol{C}=\boldsymbol{M}^{\mathrm{H}}\boldsymbol{M}$，令 $\boldsymbol{C}=A+B\mathrm{i}$，$(u+\mathrm{i}v)$ 为 $\boldsymbol{C}$ 的奇异值 σ 所对应的奇异向量，则有：

$$(A+\mathrm{i}B)(u+\mathrm{i}v)=\sigma(u+\mathrm{i}v) \tag{2}$$

$$\begin{bmatrix}A & -B\\ B & A\end{bmatrix}\begin{bmatrix}u\\ v\end{bmatrix}=\sigma\begin{bmatrix}u\\ v\end{bmatrix} \tag{3}$$

将复矩阵转成实矩阵后，以一个实 2×2 为基本单元。

对于直接复数双边Jacobi算法，以一个复数为基本运算单元进行对角化。其运算过程主要为以下两步：

$$\begin{bmatrix}\cos\phi\mathrm{e}^{\mathrm{i}\theta_\alpha} & -\sin\phi\mathrm{e}^{\mathrm{i}\theta_\beta}\\ \sin\phi\mathrm{e}^{\mathrm{i}\theta_\alpha} & \cos\phi\mathrm{e}^{\mathrm{i}\theta_\beta}\end{bmatrix}\begin{bmatrix}A_{11}\mathrm{e}^{\mathrm{i}\theta_{11}} & A_{12}\mathrm{e}^{\mathrm{i}\theta_{12}}\\ A_{21}\mathrm{e}^{\mathrm{i}\theta_{21}} & A_{22}\mathrm{e}^{\mathrm{i}\theta_{22}}\end{bmatrix}\times \begin{bmatrix}\cos\varphi\mathrm{e}^{\mathrm{i}\theta_\gamma} & \sin\varphi\mathrm{e}^{\mathrm{i}\theta_\gamma}\\ -\sin\varphi\mathrm{e}^{\mathrm{i}\theta_\delta} & \cos\varphi\mathrm{e}^{\mathrm{i}\theta_\delta}\end{bmatrix}=\begin{bmatrix}W\mathrm{e}^{\mathrm{i}\theta_w} & X\mathrm{e}^{\mathrm{i}\theta_x}\\ 0 & Z\end{bmatrix} \tag{4}$$

$$\begin{bmatrix}\cos\lambda\mathrm{e}^{\mathrm{i}\theta_\xi} & -\sin\lambda\mathrm{e}^{\mathrm{i}\theta_\eta}\\ \sin\lambda\mathrm{e}^{\mathrm{i}\theta_\xi} & \cos\lambda\mathrm{e}^{\mathrm{i}\theta_\eta}\end{bmatrix}\begin{bmatrix}W\mathrm{e}^{\mathrm{i}\theta_w} & X\mathrm{e}^{\mathrm{i}\theta_x}\\ 0 & Z\end{bmatrix}\times \begin{bmatrix}\cos\rho\mathrm{e}^{\mathrm{i}\theta_\zeta} & \sin\rho\mathrm{e}^{\mathrm{i}\theta_\zeta}\\ -\sin\rho\mathrm{e}^{\mathrm{i}\theta_w} & \cos\rho\mathrm{e}^{\mathrm{i}\theta_w}\end{bmatrix}=\begin{bmatrix}P & 0\\ 0 & Q\end{bmatrix} \tag{5}$$

这两种算法各有优缺点：QR迭代算法计算复杂度低，消耗更少的乘除法等硬件资源，但不稳定，当矩阵维度较大时，使得某次Givens矩阵为单位阵，“驱逐出境”会出现不收敛情况[8]，导致QR算法失败；双边Jacobi[9]算法是基于脉动阵列结构的迭代算法，优点是结构简单对称，易于硬件实现，缺点是计算量比QR大，且精度与迭代次数相关。

针对上述两种算法的优缺点，本文提出一种混合优化算法，即混合householder和双边Jacobi的计算方法，优化了传统复数 2×2 复数双边Jacobi的计算方法。

1.2 CSVD混合优化算法

设 $\mathbf{vec\_in}=(x_1,x_2,\cdots,x_n)$，则该向量的householder变换[10-12]步骤如下：

1) 计算输入向量平方和得：

$$\boldsymbol{\Sigma}^2=x_1^2+x_2^2+\cdots+x_n^2 \tag{6}$$

$$\boldsymbol{\sigma}=\mathrm{sign}(x_1)\times\boldsymbol{\Sigma},\ \mathrm{sign}(x_1)=\frac{\mathrm{real}(x_1)+\mathrm{imag}(x_1)\mathrm{i}}{\mathrm{abs}(x_1)} \tag{7}$$

$$\mathrm{division\_factor}=2(\boldsymbol{\Sigma}^2+\mathrm{abs}(x_1)\times\boldsymbol{\Sigma}) \tag{8}$$

2) 构造行向量 $\boldsymbol{u}$，并将其单位化得：

$$\boldsymbol{u}=\mathbf{vec\_in}+\sigma\boldsymbol{e}_{\mathrm{row}},\quad \boldsymbol{e}_{\mathrm{row}}=(0\cdots\underset{\uparrow\ \mathrm{row}}{1}\cdots0) \tag{9}$$

$$\boldsymbol{u}_e=\frac{\boldsymbol{u}}{\|\boldsymbol{u}\|} \tag{10}$$

式中，$\|\boldsymbol{u}\|$ 为向量 $\boldsymbol{u}$ 的模值。

3) 构造householder变换矩阵为：

$$\boldsymbol{H}=I-2\boldsymbol{u}_e^{\mathrm{H}}\boldsymbol{u}_e=I-(2/\mathrm{division\_factor})\boldsymbol{u}^{\mathrm{H}}\boldsymbol{u} \tag{11}$$

4) 向量变换为：

$$\mathbf{vec\_out}=\mathbf{vec\_in}-\boldsymbol{u} \tag{12}$$

对于矩阵的householder变换，则需要逐行逐列调用向量householder变换，并在式(11)用变换矩阵 $\boldsymbol{H}$ 右乘，更新其他行向量。对于 $2\times 4n$ 的复矩阵只需进行行变换，将一般矩阵化简为形如式(13)的二对角矩阵 $\boldsymbol{M}_k$，并提取左上角非零复数 2×2 矩阵 $\boldsymbol{M}$，

则有：

$$\boldsymbol{M}_k=\begin{bmatrix} p_1 & 0 & \cdots & 0 \\ q_1 & q_2 & \cdots & 0 \end{bmatrix} \tag{13}$$

对 $\boldsymbol{M}$ 进行改进型复数双边Jacobi运算，其计算步骤如下：

1) 将复数 2×2 矩阵转成共轭对称复矩阵：

$$\boldsymbol{M}=\begin{bmatrix} p_1 & 0 \\ q_1 & q_2 \end{bmatrix}$$

$$\boldsymbol{M}_{\text{sym}}=\boldsymbol{M}^{\text{H}}\boldsymbol{M}=\begin{bmatrix} A & b_r+b_i\text{i} \\ b_r-b_i\text{i} & D \end{bmatrix} \tag{14}$$

2) 将 $\boldsymbol{M}_{\text{sym}}(1,2)$ 转成幅角形式：

$$\begin{bmatrix} A & b_r+b_i\text{i} \\ b_r-b_i\text{i} & D \end{bmatrix}=\begin{bmatrix} A & B\text{e}^{-\text{i}\theta_b} \\ B\text{e}^{-\text{i}\theta_b} & D \end{bmatrix} \tag{15}$$

3) 将复矩阵化为实矩阵：

$$\begin{bmatrix} 1 & 0 \\ 0 & \text{e}^{-\text{i}\theta_b} \end{bmatrix}\begin{bmatrix} A & B\text{e}^{-\text{i}\theta_b} \\ B\text{e}^{-\text{i}\theta_b} & D \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & \text{e}^{-\text{i}\theta_b} \end{bmatrix}=\begin{bmatrix} A & B \\ B & D \end{bmatrix} \tag{16}$$

4) 计算双边Jacobi变换旋转角为：

$$2\theta=\text{atan}\left(\frac{2B}{D-A}\right) \tag{17}$$

5) 双边Jacobi变换为：

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}^{\text{T}}\begin{bmatrix} A & B \\ B & D \end{bmatrix}\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}=\begin{bmatrix} P & 0 \\ 0 & Q \end{bmatrix} \tag{18}$$

6) $\boldsymbol{V}$ 矩阵计算为：

$$\boldsymbol{V}=\begin{bmatrix} 1 & 0 \\ 0 & \text{e}^{-\text{i}\theta_b} \end{bmatrix}\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{19}$$

2×8 CSVD的 $\boldsymbol{V}_{\text{svd}}$ 矩阵计算：将 2×2 复数 $\boldsymbol{V}$ 矩阵补充到 8×8 单位阵的1,2行列得到复酉矩阵 $\boldsymbol{V}_1$，然后将householder产生的 $\boldsymbol{H}_1$， $\boldsymbol{H}_2$ 与 $\boldsymbol{V}_1$ 矩阵相乘即得到最终输出酉矩阵 $\boldsymbol{V}_{\text{svd}}$，即 $\boldsymbol{V}_{\text{svd}}=\boldsymbol{H}_1\times\boldsymbol{H}_2\times\boldsymbol{V}$。

2　3种算法的比较与分析

以2×4复矩阵为例，比较直接复数双边Jacobi(算法1)，复转实双边Jacobi(算法2)和混合优化算法(算法3)在硬件资源消耗、迭代次数、误差精度之间的差异。

表1给出了3种算法的性能比较，资源消耗为单次迭代次数下的运算量。图2的精度比较结果是基于MATLAB中CORDIC函数浮点仿真结果。

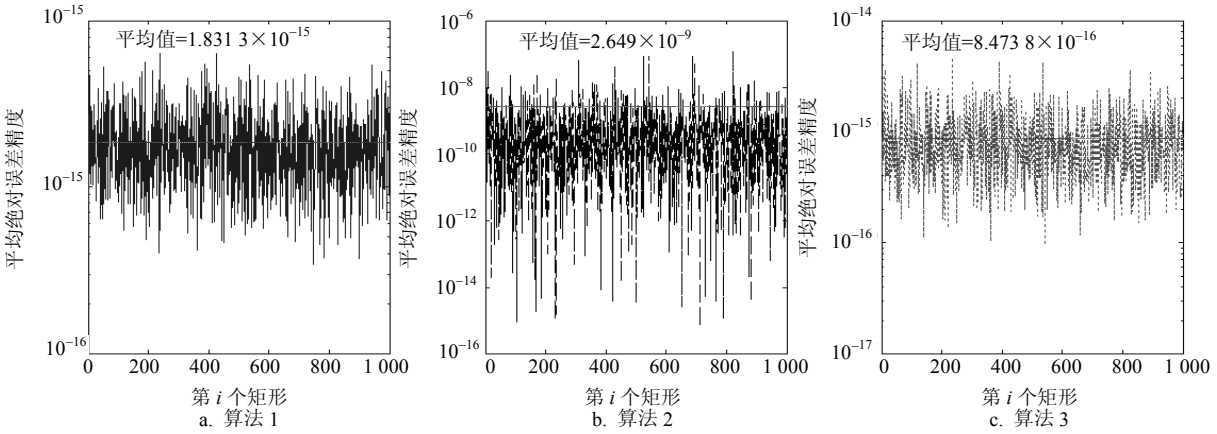

图2　复矩阵2×4在3种算法下平均绝对误差精度比较

表1　复矩阵2×4奇异值分解3种算法性能比较

	硬件资源							迭代次数	平均绝对误差精度
	CORDIC单元				乘法器		除法		
	rotate	translate	arctan	sqrt	复数乘	实数乘			
算法1	32	14	3	0	4	16	0	9	图3
算法2	2	0	2	0	16	100	0	28	图3
算法3	4	2	2	4	9	26	4	0	图3

更为一般地，对于 $2\times N$ 或 $N\times 2$ 的复长方矩阵，算法2和算法3的复杂度比较如图3所示。

在图3中将CORDIC单元、4个实数乘法统一等效为复数乘法。算法3中双边Jacobi运算在算法一基础上进行了优化，极大地减少硬件资源消耗。在单次迭代的情况下，算法2的资源消耗略低于算法3，通过硬件结构上的优化，可以使两种算法在资源消耗上相当。但是增加迭代次数即增加了处理延时，

导致矩阵吞吐率(单位时间处理矩阵个数)急剧下降。且随着矩阵列数的增长，算法2和算法3之间的资源消耗差异将立方增长。

表1中迭代次数并不固定，理论上可由BLV方法总结公式得到，即 $(\log_2 N+1)(N-1)$，而复转实的方法为 $(\log_2(2N)+1)(2N-1)$， N 为复方阵阶数。

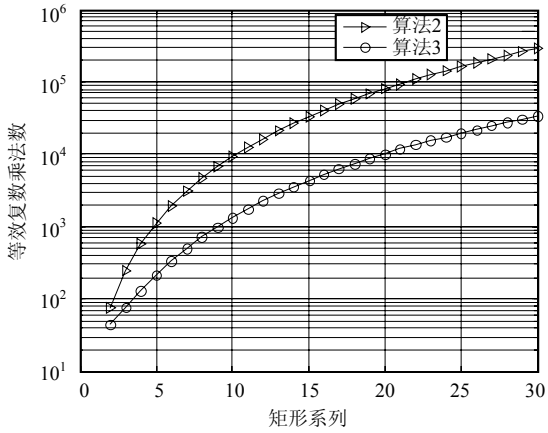

图3 算法2、算法3复杂度比较

计算误差的方法如下：设 $\boldsymbol{M}=\boldsymbol{USV}^{\mathrm{H}}$，消去 $\boldsymbol{U}$，有 $\mathbf{error}=\boldsymbol{M}^{\mathrm{H}}\boldsymbol{M}-\boldsymbol{VS}^{\mathrm{H}}\boldsymbol{SV}^{\mathrm{H}}$，分别取误差矩阵 **error** 各元素的实部和虚部绝对值并取其平均。图2中算法1和算法3精度高于算法2。误差的产生主要取决于CORDIC核，当增加迭代次数后，使得在对复数 2×2 对角化时非对角元素极小，此时 θ_b 应作 0 或 π 处理。而CORDIC核仍求两极小数的比值来计算角度，进而导致计算的角度 θ_b 误差增大，最终导致输出 $\boldsymbol{V}$ 矩阵误差增大。特别是针对 $2\times4n$ 的复矩阵，算法2需要先取其共轭转置并与其本身相乘，再扩展为维度为 $8n\times8n$ 的实矩阵，该矩阵至少包含 $8n$ 个零元素。在迭代过程中容易产生极小的非对角元素。当然可以采取设置门限的方法，若非对角元素绝对值小于某一常数，直接置零。由此，对于奇异值分解的各种算法中迭代算法具有一定的不稳定性。

对于相同维度的长方复矩阵，算法3在资源和精度上要好于算法1和算法2，由于householder变换涉及乘、除、开方，且步骤并不像其他两种算法规则。算法3在实现的复杂度上要略高于其他两种算法。

3 2×8 CSVD的硬件实现结构

CSVD的硬件实现主要包括两个模块：向量householder变换和改进型 2×2 复数双边Jacobi模块。这两个模块的计算是顺序的，因此可以设计为流水线结构。根据具体FPGA芯片资源的多少，可以灵活选择全流水或部分复用实现结构。

3.1 向量householder变换的实现结构

一方面充分考虑资源的可复用程度，合理调配运算顺序；另一方面设计的结构应尽可能简单、模块化。图4设计了一种全流水的行householder变换结构，对于大型矩阵可以复用该模块。

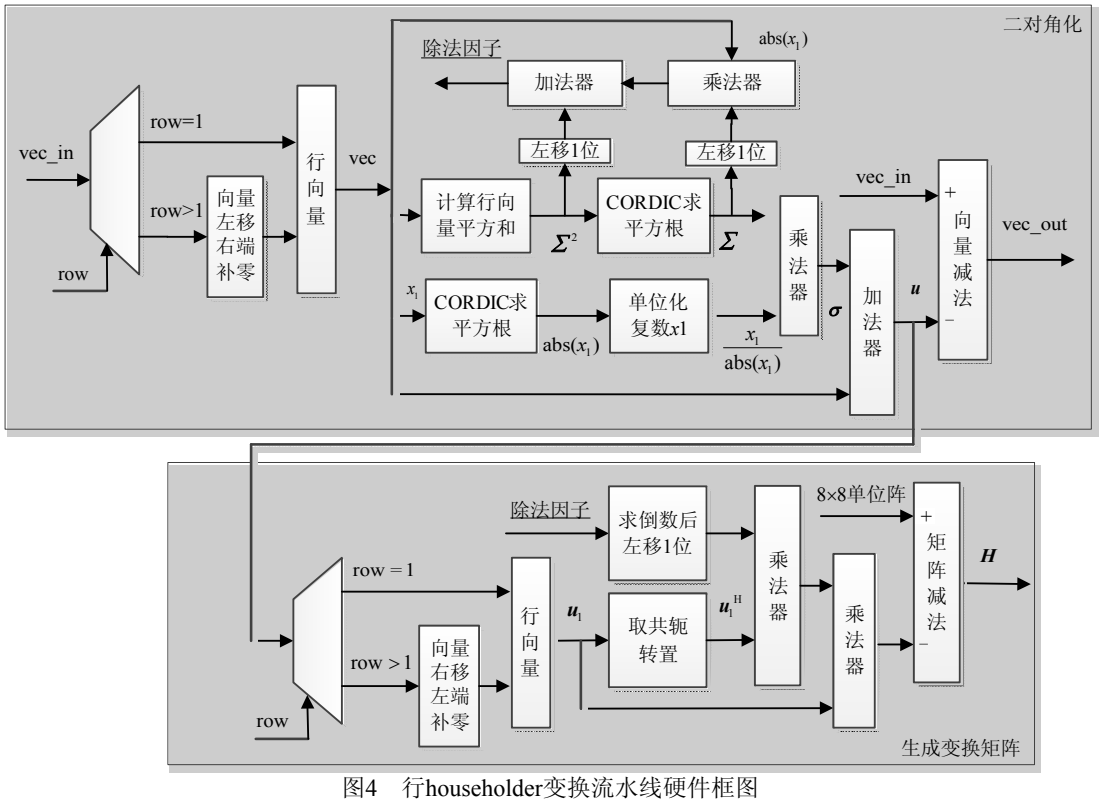

图4 行householder变换流水线硬件框图

3.2　改进型2×2复数双边Jacobi的实现结构

本文提出的复数2×2双边Jacobi方法是在文献[5,7]的基础上针对硬件结构提出的改进。文献[5,7]方法对于一般复矩阵计算量偏大，特别是对于FPGA、DSP等硬件平台。但如果在对角化之前，先将复矩阵转换为共轭对阵矩阵，则计算量会大大减小。额外的资源消耗仅仅是2个复数2×2矩阵相乘。图5的硬件框图相比于文献[5]的硬件框图更省资源。由于不用迭代运算，在精度上可以考虑减少乘法、CORDIC运算的位宽。如Xilinx平台下，一个实数乘法用18×25的位宽，可以保证最大化利用乘法器。其他运算以此为基准进行定点。图5中CORDIC的3种模式均通过Xilinx的CORDIC IP核实现，本文将其设置为全并行模式，以保证整体流水线设计。

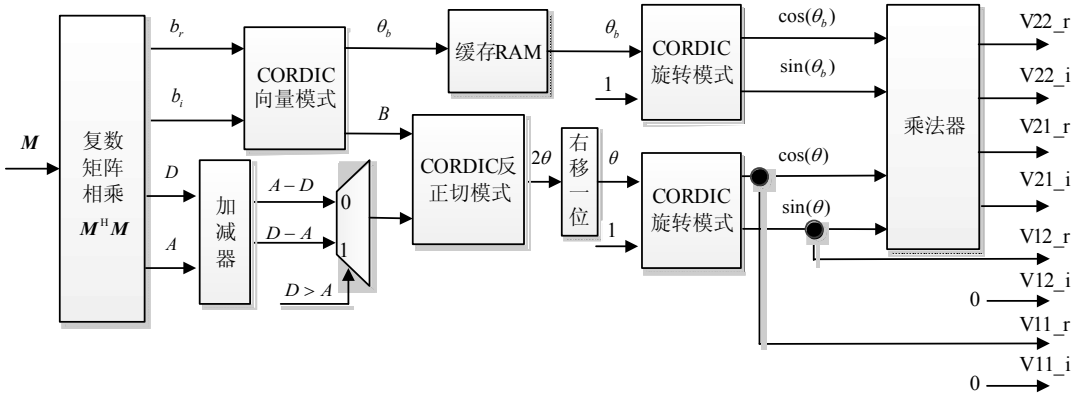

图5　复数2×2对角化流水线硬件框图

3.3　基于FPGA的实现验证

对于输出$\boldsymbol{V}$矩阵维度较大的复数奇异值分解，会耗费更多的FPGA资源如乘法器、除法器和CORDIC核。当DSP资源超过50%以后，布局布线后的最高时钟频率较综合后的最高时钟频率会大幅下降，主要原因在于乘法器核在布局布线过程中会产生较大的线延时，可通过减小乘法器输入输出的扇出解决，也可以通过更改综合工具设置。由布局布线后的结果才能比较准确反映设计的准确性和可靠性。本文设计采用Xilinx的FPGA硬件实现方案，其型号为xc6vlx240t-3ff1156，基本满足设计要求。表2给出了算法2和算法3的资源占用比较，算法2实现平台为Altera Stratix IV。算法2为迭代结构，时钟频率为105 MHz，延时为3 800个时钟；而算法3为全并行，流水结构，布局布线后的最大时钟频率200.240 MHz，延时为330个时钟。算法3在资源和吞吐率上均优于算法2。

表2　复数2×8奇异值分解FPGA资源占用

逻辑资源使用情况	使用量	使用率/%	使用量	使用率/%
	算法2		算法3	
逻辑片寄存器的个数	153 418	36	85 815	28
逻辑片查找表的个数	184 163	43	70 013	46
存储器的位数	69 120	1	17 899	30
RAMB36E1/FIFO36E1的个数	-	-	7	1
RAMB18E1/FIFO18E1的个数	-	-	264	31
DSP48E1的个数	512	50	579	75

4　结 束 语

本文提出了一种主要针对$2\times 4n$或$4n\times 2$的CSVD混合优化算法，若矩阵维度为奇数，需要将矩阵维度向上扩充至偶数。该算法通过对多种传统算法的部分运算整合、改进，极大地减小了CORDIC核的使用且不需要迭代。通过与传统算法对比，该算法比传统算法至少在资源上节省26%，延时缩短10倍，精度提高一个数量级。最后对2×8 CSVD进行FPGA实现，算法上的优势在硬件上得以体现。下一步将对更高维度的长方阵CSVD作探讨。

参 考 文 献

[1] AU E K S, JIN S, MCKAY M R, et al. Analytical performance of MIMO-SVD systems in ricean fading channels with channel estimation error and feedback delay[J]. IEEE Transactions on Wireless Communications,

2008, 7(4): 1315-1325.

[2] SRINIVASAN J, RAJARAM S. FPGA implementation of precoding using low complexity SVD for MIMO-OFDM systems[C]//Information Communication and Embedded Systems (ICICES). [S.l.]: IEEE, 2013.

[3] CHAKROBORTY S, SAHA G. Feature selection using singular value decomposition and QR factorization with column pivoting for text-independent speaker identification [J]. Speech Communication, 2010, 52(9): 693-709.

[4] 胡谋法, 董文娟, 王书宏, 等. 奇异值分解带通滤波背景抑制和去噪[J]. 电子学报, 2008, 36(1): 111-116.
HU Mou-fa, DONG Wen-juan, WANG Shu-hong, et al. Singular value decomposition band-pass-filter for image background suppression and denoising[J]. Acta Electronica Sinica, 2008, 36(1): 111-116.

[5] WANG Y, CUNNINGHAM K, NAGVAJARA P, et al. Singular value decomposition hardware for MIMO: State of the art and custom design[C]//Reconfigurable Computing and FPGAs (ReConFig). [S.l.]: IEEE, 2010.

[6] HAN Q, ZENG L. FPGA Implementation for low-rank channel estimation of OFDM[J]. Journal of Networks, 2012, 7(10): 1631-1638.

[7] HEMKUMAR N D, CAVALLARO J R. A systolic VLSI architecture for complex SVD[C]//Circuits and Systems, ISCAS'92. [S.l.]: IEEE, 1992.

[8] 赵学智, 叶邦彦. 单向收缩QR算法在奇异值分解中的收敛特性[J]. 电子科技大学学报, 2010, 39(5): 762-767.
ZHAO Xue-zhi, YE Bang-yan. Convergence characteristic of single direction shrink QR algorithm in the singular value decomposition[J]. Journal of University of Electronic Science and Technology of China, 2010, 39(5): 762-767.

[9] MA W, KAYE M E, LUKE D M, et al. An FPGA-based singular value decomposition processor[C]//Electrical and Computer Engineering. [S.l.]: IEEE, 2006.

[10] LIU J, ZHANG J. A new maximum simplex volume method based on householder transformation for endmember extraction[J]. IEEE Transactions on Geoscience and Remote Sensing, 2012, 50(1): 104-118.

[11] PEDRAM A, GERSTLAUER A, GEIJN R A V D. Floating point architecture extensions for optimized matrix factorization[C]//Proceedings of the 2013 IEEE 21st Symposium on Computer Arithmetic. [S.l.]: IEEE, 2013: 49-58.

[12] 张贤达. 矩阵分析与应用[M]. 北京: 清华大学出版社有限公司, 2004.
ZHANG Xian-da. Matrix analysis and applications[M]. Beijing: Tsinghua and Springer Publishing House, 2004.

编 辑 税 红