

# 基于CORDIC算法的高精度浮点对称矩阵 特征值分解的FPGA实现

陈刚 陈旭 徐元 边旻 鲁华祥

(中国科学院半导体研究所神经网络实验室 北京 100083)

**摘要** 为了提高实对称矩阵特征值分解算法的速度,在FPGA上设计并实现了符合IEEE-754标准的单精度浮点(32-bit)CORDIC算法,在保证运算精度的前提下,最大限度地优化资源和速度。整个设计是在Xilinx公司的Spartan-3 xc3sd1800a-5 FPGA上实现的。测试结果显示:1)对于 $3 \times 3$ 的实对称矩阵(每个矩阵元素是32-bit浮点数),精度达到 $2^{-20}$ ;2)设计实现的结构占用2467个slices,占芯片总资源的14%,最高运行时钟频率达到154 MHz。

**关键词** 特征值分解,CORDIC,FPGA,浮点数据

中图分类号 TP332 文献标识码 A

## Floating-point CORDIC-based Real-valued Symmetric Matrix Eigenvalue Decomposition on FPGA

CHEN Gang CHEN Xu XU Yuan BIAN Yi LU Hua-xiang

(Lab of Artificial Neural Networks, Institute of Semiconductors, CAS, Beijing 100083, China)

**Abstract** This paper presented the architecture of real-valued symmetric matrix eigenvalue decomposition based on rotation mode CORDIC algorithm on FPGA. The proposed architecture adopts a direct mapping by 32 floating-point CORDIC-based process units that can compute the eigenvalue decomposition for a  $3 \times 3$  real symmetric matrix. In order to achieve a comprehensive resource and performance evaluation, the whole design was realized on Spartan-3 xc3sd1800a-5 FPGA. The evaluation results show that 1) the proposed architecture satisfies  $3 \times 3$  eigenvalue decomposition for the  $2^{-20}$  accuracy requirement when the word length of the data path is 32-bit; 2) occupies about 2467(14%) slices, and the maximum frequency is 154MHz.

**Keywords** Eigenvalue decomposition, CORDIC, FPGA, Floating data

## 1 引言

在许多科学和工程领域中,矩阵特征值和特征向量的计算是必不可少的,其在数字信号处理、盲源分离、频谱分析、数字图像处理、最优化问题、电力电子等领域都有涉及<sup>[1]</sup>。其中大都针对的是对称矩阵, Jacobi 迭代是应用最广的算法之一。同时,为了满足一些高性能、实时性的应用需求,需要研究在硬件中实现矩阵特征值和特征向量的计算。近年来,随着数字硬件和可编程逻辑器件的大力发展,特别是FPGA性能的不提高,基于FPGA的计算加速已经逐渐成为提高计算速度和计算效率的重要手段之一。Jacobi 迭代本身固有的并行特点,使其在FPGA平台上的执行具有重要的速度优势。近年来,已有一些关于在FPGA上实现Jacobi 迭代的研究<sup>[2-4]</sup>。这些研究在不同方面取得了相应的成果,为后续研究提供了重要的参考。但这些研究基本上都是基于定点数实现的。由于高精度计算、图形加速、数字信号处理等应用对浮点处理的要求越来越高,以往芯片的面积大小限制了浮点单元的实现,但是集成电路工艺的不断发展和特征尺寸的不断缩小以及芯片面积的不断增大,为浮点处理单元的设计与实现提供了基

础。本文利用FPGA设计实现对称矩阵特征值求解硬件系统。设计中采用浮点运算单元,其相比定点设计能提供更高的精度和更大的动态范围。

本文第2节简要介绍Jacobi迭代方法及使用CORDIC结构实现Jacobi算法;第3节给出高精度浮点对称矩阵特征值分解的FPGA实现;第4节给出实施的结果;最后给出结论。

## 2 算法原理

### 2.1 雅克比(Jacobi)迭代法

设线性方程组 $AX=B$ ,其中 $A$ 是 $n \times n$ 维的对称矩阵,为非奇异阵,且 $a_{ii} \neq 0 (i=1, \dots, n)$ ,  $X$ 是 $n$ 维未知量, $B$ 是 $n$ 维右端项。 $A$ 可分解为 $A=D+L+U$ 。

其中:

$$D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, L = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & 0 \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 0 \end{bmatrix}$$

到稿日期:2012-07-10 返修日期:2012-11-12 本文受国家自然科学基金(61076014),中科院战略性先导科技专项(XDA06020700)资助。

陈刚(1985—),男,博士生,主要研究领域为智能信息处理, E-mail: chengang08@semi.ac.cn; 陈旭(1978—),女,博士,副研究员,主要研究领域为智能信息处理; 徐元(1984—),男,硕士生,主要研究领域为电路与系统。

$$U = \begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ 0 & 0 & 0 & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}$$

改写线性方程组,将第  $i$  组方程( $i=1, \dots, n$ )表示为  $y_i$  的表达式:

$$y_i = \frac{(b_i - \sum_{j=1, j \neq i}^n a_{ij} y_j)}{a_{ii}} \quad (i=1, \dots, n) \quad (1)$$

一直迭代到满足  $\max |x_i - y_i| < \epsilon$  或达到最大迭代次数为止,初始向量为  $x(0)$ ,则雅克比迭代法的分量形式<sup>[5]</sup>为:

$$x_i^{(k+1)} = \frac{(b_i - \sum_{j=1, j \neq i}^n a_{ij} x_j^{(k)})}{a_{ii}} \quad (i=1, \dots, n) \quad (2)$$

## 2.2 CORDIC 算法原理

由第 2 节分析可知, Jacobi 迭代算法中涉及到大量的乘、除运算,若要实现浮点精度,则直接用硬件实现比较困难。因此,本文采用计算超越函数的 CORDIC 算法来实现其中的乘、除运算,这会极大简化硬件实现的难度。下面简要介绍 CORDIC 算法的原理。

CORDIC (Coordinate Rotational Digital Computer) 算法是 Volder<sup>[6]</sup> 于 1959 年在美国航空控制系统的设计中提出来的,它是一种用于计算超越函数的循环迭代算法。其基本思想是使一系列与运算基数相关的角度不断偏摆,从而逼近所需旋转的角度。从广义上讲,它是一个数值计算逼近的方法。这些固定的角度与计算基数有关,运算只有移位和加减。可通过该算法不同的实现形式(如圆周模式、线性模式、双曲线模式等,如表 1 所列)来进行乘、除、平方根、三角函数、向量旋转(即复数乘法)以及指数等运算。整个算法由于只有加减和移位操作,因此非常适合用硬件来实现。

表 1 CORDIC 算法模式

| 模式         | 角度 $\theta_k$        | 移位序列            | 半径因子          |
|------------|----------------------|-----------------|---------------|
| 圆周 $m=1$   | $\tan^{-1}(2^{-k})$  | 0, 1, 2, ...    | $K_1=1.65$    |
| 线性 $m=0$   | $2^{-k}$             | 1, 2, ...       | $K_0=1.0$     |
| 双曲线 $m=-1$ | $\tanh^{-1}(2^{-k})$ | 1, 2, 3, 4, ... | $K_{-1}=0.80$ |

在每次迭代中, CORDIC 算法实现以下映射:

$$\begin{bmatrix} X_{k+1} \\ Y_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & m\delta_k 2^{-k} \\ \delta_k 2^{-k} & 1 \end{bmatrix} \begin{bmatrix} X_k \\ Y_k \end{bmatrix} \quad (3)$$

$$Z_{k+1} = Z_k + \delta_k \theta_k \quad (4)$$

其中  $\theta_k$  在表 1 中已经给出,  $\delta_k = \pm 1$ , 两个旋转方向是  $Z_k \rightarrow 0$  和  $Y_k \rightarrow 0$ 。

## 2.3 单精度浮点数

如图 1 所示,单精度浮点数据由符号位  $s$ 、指数部分  $e$ 、小数部分  $f$  构成。其值的计算公式如下:

$$v = (-1)^s \cdot 2^{(e-127)} \cdot 1.f \quad (5)$$

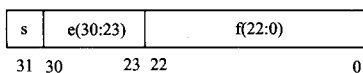


图 1 单精度浮点数存储结构

根据浮点数的结构特点,实现一个 32 位浮点数和  $2^{-k}$  的乘积,仅需要把浮点数相应的指数部分减去  $k$  即可,按照 IEEE-754 标准,仅需要实现一个 8bits 的减法器。而与  $\pm 1$  的乘积,只需要相应地改变浮点数的符号位。式(3)中的运算将很容易实现。

## 3 硬件结构

本设计针对高精度浮点型对称矩阵特征值分解问题,处理 32 位浮点型的数据,尽管 Xilinx 公司为其 FPGA 芯片提供了丰富的浮点数据 IP 核来实现浮点数据的加、减、乘、除、开方,以及比较大小,但是这些 IP 核都相当耗费资源。因此,在设计中不仅要考虑系统的速度,同时也要兼顾资源的消耗,所以本文提出算法硬件结构,如图 2 所示。由图可见,本设计的硬件结构主要包括以下几个模块:数据控制器;FFU 模块;SFU 模块;VFU 模块。

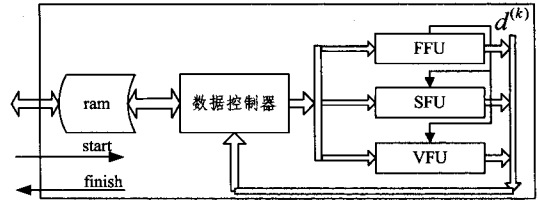


图 2 系统结构图

其中, FFU 和 SFU 是用于计算特征值的模块, FFU 实现的是对角线元素的 CORDIC 迭代,其内部结构如图 3 所示,而 SFU 实现的是非对角线元素的迭代。VFU 是用于计算特征向量的模块。整个系统的工作流程图如图 4 所示。

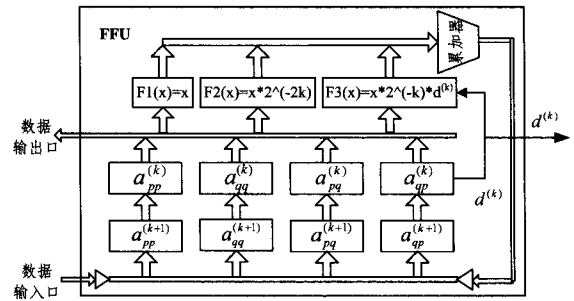


图 3 系统内部 FFU 结构示意图

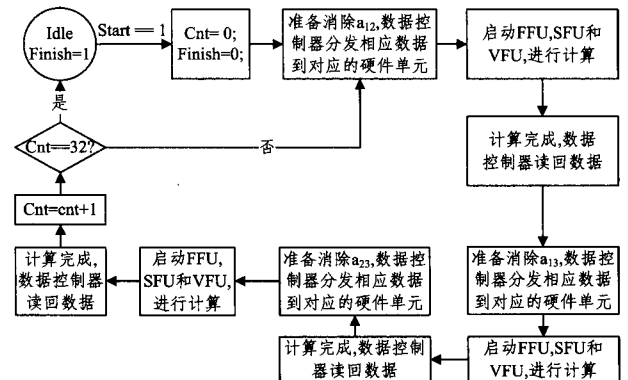


图 4 系统工作流程图

(1) 在 FFU 模块内部实现如下运算:

$$a_{pp}^{(k+1)} = a_{pp}^{(k)} + d^{(k)} \times 2^{-k} (a_{pq}^{(k)} + a_{qp}^{(k)}) + 2^{-2k} a_{qq}^{(k)} \quad (6)$$

$$a_{qp}^{(k+1)} = a_{qp}^{(k+1)} = -2^{-k} d^{(k)} (a_{pp}^{(k)} + a_{pp}^{(k)} - 2^{-2k} a_{qp}^{(k)} + 2^{-k} d^{(k)} a_{qq}^{(k)}) \quad (7)$$

$$a_{qq}^{(k+1)} = a_{qq}^{(k)} + 2^{-k} d^{(k)} (a_{qp}^{(k)} + a_{pq}^{(k)}) + 2^{-2k} a_{pp}^{(k)} \quad (8)$$

(2) 在 SFU 模块内部实现如下运算:

$$a_{pj}^{(k+1)} = a_{pj}^{(k)} + 2^{-k} d^{(k)} a_{iq}^{(k)}, j \neq p, q \quad (9)$$

$$a_{iq}^{(k+1)} = a_{iq}^{(k)} + 2^{-k} d^{(k)} a_{pj}^{(k)}, i \neq p, q \quad (10)$$

(3) 在 VFU 模块内部实现如下运算:

$$e_{jp}^{(k+1)} = e_{jp}^{(k)} + 2^{-k} d^{(k)} e_{jq}^{(k)}, j=1,2,3 \quad (11)$$

$$e_{jq}^{(k+1)} = e_{jq}^{(k)} + 2^{-k} d^{(k)} e_{jp}^{(k)}, j=1,2,3 \quad (12)$$

结合图 3 所示的 FFU 内部结构图, FFU 工作过程如下:

1) 数据控制器把数据存入 FFU 对应的位置  $a_{pp}^{(k)}, a_{pq}^{(k)}, a_{qp}^{(k)}, a_{qq}^{(k)}$  4 个寄存器,  $d^{(k)}$  即是由  $a_{pq}^{(k)}$  的符号位引出。设置  $cnt=0$ ; 并启动 FFU。

2) 在 FFU 的控制下数据按照算法的要求流经不同的函数模块, 并把得到的计算结果进行累加, 分别计算  $a_{pp}^{(k+1)}, a_{pq}^{(k+1)}, a_{qp}^{(k+1)}, a_{qq}^{(k+1)}$ 。

3) 更新。令  $a_{pp}^{(k)} = a_{pp}^{(k+1)}, a_{pq}^{(k)} = a_{pq}^{(k+1)}, a_{qp}^{(k)} = a_{qp}^{(k+1)}, a_{qq}^{(k)} = a_{qq}^{(k+1)}, cnt = cnt + 1$ 。

4) 如果  $cnt == 64$ , 计算结束, 否则转到 2)。

VFU 和 SFU 的结构和计算过程类似, 这里不再赘述。

## 4 硬件实现与测试

整个设计采用 Verilog 语言描述, 在 Xilinx 的 xc3sd1800a-5 cs484 FPGA 上进行了实现。为验证系统的正确性, 同时测试系统的性能, 本文用表 2 所列的矩阵 X 对系统进行了测试。表 3 给出了硬件实现的综合报告。

表 2 测试数据及结果

| 十六进制数据  |                                                                                                                                        |  | 浮点数据                                                                                                                                                |  |  |
|---------|----------------------------------------------------------------------------------------------------------------------------------------|--|-----------------------------------------------------------------------------------------------------------------------------------------------------|--|--|
| 原始实对称矩阵 | $X = \begin{bmatrix} 405b851f & 402ae148 & 4099999a \\ 402ae148 & 3f99999a & 40666666 \\ 4099999a & 40666666 & 40a00000 \end{bmatrix}$ |  | $X = \begin{bmatrix} 3.43 & 2.67 & 4.8 \\ 2.67 & 1.2 & 3.6 \\ 4.8 & 3.6 & 5 \end{bmatrix}$                                                          |  |  |
| 特征值     | $D = \begin{bmatrix} BF825FBB & 2F5CA1FF & 322F0A7A \\ 2F5CA1FF & BEE765CA & 31FE3A14 \\ 322F0A7A & 31FE3A14 & 41319BA0 \end{bmatrix}$ |  | $D = \begin{bmatrix} -1.0185464 & 0 & 0 \\ 0 & -0.4519481 & 0 \\ 0 & 0 & 11.1004943 \end{bmatrix}$                                                  |  |  |
| 特征向量    | $E = \begin{bmatrix} BE74CC6F & 3F46F593 & 3F1503FE \\ BF3CC62B & BF090B3A & 3ED2E50B \\ 3F21B97A & BEA959F6 & 3F33798B \end{bmatrix}$ |  | $E = \begin{bmatrix} -0.2390611 & 0.7771846 & 0.5820921 \\ -0.7373988 & -0.5353275 & 0.4119037 \\ 0.6317363 & -0.3307644 & 0.7010733 \end{bmatrix}$ |  |  |

表 3 综合报告

| Vendor                     | Xilinx                  |
|----------------------------|-------------------------|
| Family                     | Spantan-3dsp            |
| Device                     | xc3sd1800a-5            |
| Number of Slice Flip Flops | 3020 out of 33280 (9%)  |
| Number of 4 input LUTs     | 3121 out of 33280 (9%)  |
| Slice                      | 2467 out of 16640 (14%) |
| Maximum frequency          | 154MHz                  |

**结束语** 本文利用 FPGA 设计实现了符合 IEEE-754 标准的单精度浮点 (32-bit) 雅可比算法 (基于 CORDIC 算法实现) 用于求解高精度浮点实对称矩阵的特征值和特征向量。该设计结合 FPGA 自身的特点, 采用流水线结构的设计思想, 在保证运算精度的前提下, 最大限度地优化了资源和速度。该设计采用浮点运算单元, 相比定点设计提供了更高的精度和更大的动态范围。测试结果表明: 1) 对于  $3 \times 3$  的实对称矩阵 (每个矩阵元素是 32-bit 浮点数), 精度达到  $2^{-20}$ ; 2) 设计实现的结构占用 2467 个 slices, 占芯片总资源的 14%, 最高运行时钟频率达到 150MHz。

## 参考文献

- [1] Bravo I, Jimenez P, Mazo M, et al. Implementation in FPGAs of Jacobi Method to Solve the Eigenvalue and Eigenvector Problem [C] // International Conference on Field Programmable Logic and Applications. 2006; 1-4
- [2] 袁生光, 沈海斌. 基于 Jacobi 算法对称矩阵特征值计算的 FPGA 实现[J]. 机电工程, 2008, 25(10): 80-82
- [3] 宋庆增, 顾军华, 张金珠. 基于 FPGA 的 Jacobi 迭代求解器研究[J]. 计算机工程与应用, 2011, 47(29): 74-77
- [4] Wang Tao, Wei Ping. Hardware Efficient Architectures of Improved Jacobi Method to Solve the Eigen Problem [C] // 2nd International Conference on Computer Engineering and Technology (ICCET). V6, 2010; 22-25
- [5] 张贤达. 矩阵分析与应用[M]. 北京: 清华大学出版社, 2004
- [6] Volder J E. The CORDIC Trigonometric computing technique [J]. IRE Transactions on Electronics Computers, 1959, 8(3): 330-334

(上接第 18 页)

- [26] Gu X H, Gong W G, Yang L P. Regularized locality preserving discriminant analysis for face recognition[J]. Neurocomputing, 2011, 75(5): 201-215
- [27] Yu H, Yang J. A direct LDA algorithm for high-dimensional data-with application to face recognition[J]. Pattern Recognition, 2001, 34(10): 2067-2070
- [28] Wang X, Tang X. Dual-space linear discriminant analysis for face recognition[C] // Proceeding of IEEE Conference on Computer Vision and Patter Recognition. 2004; 564-569
- [29] Chu D, Thye G S. A new and fast implementation for null space based linear discriminant analysis [J]. Pattern Recognition, 2010, 43(4): 1373-1379
- [30] Yang L P, Gong W G, Gu X H, et al. Bagging null space locality

preserving discriminant classifiers for face recognition[J]. Pattern Recognition, 2009, 42: 1853-1858

- [31] Xu D, Yan S C, Lin S, et al. Convergent 2-D subspace learning with null space analysis[J]. IEEE Transactions on Circuits and Systems for Video Technology, 2008, 18(12): 1753-1759
- [32] Eschenauer H, Koski J, Osyczka A. Multicriteria design optimization[M]. Springer-Verlag, 1990
- [33] Georgiades A S, Belhumeur P N, Kriegman D J. From few to many: Illumination cone models for face recognition under variable lighting and pose[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001, 23(6): 643-660
- [34] Guan Z, Wang C, Chen Z, et al. Efficient face recognition using tensor subspace regression[J]. Neurocomputing, 2010, 73(13): 2744-2753