ISSN 1000-0054 清华大学学报（自然科学版） 2016 年 第 56 卷 第 9 期 9/18
CN 11-2223/N J Tsinghua Univ (Sci & Technol), 2016, Vol. 56, No. 9 963-968

基于矩阵三角化分解的 Cholesky 分解及 FPGA 并行结构设计

刘书勇， 林俊宇， 吴艳霞， 张博为

（哈尔滨工程大学 计算机科学与技术学院，哈尔滨 150001）

摘 要：矩阵运算是高性能计算中核心问题之一，矩阵分解是提高矩阵运算并行性的重要途径，飞速发展的 FPGA 为并行运算结构提供了有力的环境支持。该文基于子矩阵更新同一化算法实现了 Cholesky 分解，基于 FPGA 设计了相应的并行结构。实验结果表明：与通用处理器的软件实现相比，本文实现的 Cholesky 分解的 FPGA 并行结果在核心计算性能上可以取得 10 倍以上的加速比，该算法针对矩阵三角化计算过程具有更高的数据和流水并行性。

关键词：矩阵三角化分解；Cholesky 分解；并行结构；现场可编程门阵列

中图分类号：TP302.1 **文献标志码**：A

文章编号：1000-0054(2016)09-0963-06

DOI：10.16511/j.cnki.qhdxxb.2016.21.060

Cholesky decomposition and parallel structure design based on matrix triangularization decomposition

LIU Shuyong, LIN Junyu, WU Yanxia, ZHANG Bowei

(Collage of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)

Abstract: Matrix computing is one of the core problems in high performance computing with matrix decomposition being an important way to improve the parallelism of matrix computations. FPGA gives a powerful environment for parallel computing. This study uses Cholesky decomposition based on a hardware-adaptive parallel sub-matrix identity updating algorithm. Its parallel structure is based on FPGA. Tests show that this structure achieves more than 10 fold speedup compared to general-purpose processors in terms of the kernel computational speed because the algorithm has better data-parallelism and pipeline-parallelism during matrix triangularization.

Key words: matrix triangularization decomposition; Cholesky decomposition; parallel structure; field programmable gate array

当前，对高性能矩阵三角化分解的研究主要从通用计算[1]、脉动阵列[2]和 FPGA 实现[3] 3 个方向开展。在高性能通用计算领域内，从计算特征及应用领域考虑，对矩阵三角化分解的研究主要在基于单指令流多数据流（SIMD）或多指令流多数据流（MIMD）的向量机、共享存储的多处理机等技术[4-10]的并行计算机体系结构上开展。一些研究项目在单处理器以及通用计算图形处理器（GPGPU）上也有良好的表现。该领域的研究通过向程序设计者提供具有标准接口和较好移植性的数值计算模块，以在目标体系结构上实现算法的高性能计算。许多高层编程语义数值计算库如基于 C++的 cvmlib、英特尔数学核心函数库（Intel MKL）和矩阵计算模板库（matrix template library，MTL）等则通过调用 1 个或若干基础数值计算子程序库来实现。在业界已有的研究成果中，以 BLAS、LAPACK 和 LINPACK 等[11-12]为代表的一系列数值线性代数软件包最具有代表性。使用线性或二维脉动阵列机实现三角化分解的研究集中在 20 世纪 80 年代和 90 年代初，往往针对某些特定科学工程领域的应用。该结构是 VLSI 设计中一种经典体系结构，通过消除阵列中的广播路径实现三角化分解中流水并行的最大化，往往占用较大的片上面积，产生较大的功耗。近年来，更多的研究者在固定面积的 FPGA 上设计具有较高性能的矩阵三角化分解并行结构。归纳起来，这些设计分别选择从单个处理单元（processing element，PE）设计、存储层次设计、阵列结构设计等方面开展优化，实现 FPGA 矩阵三角化分解的研究以及针对软件数值线性代数软件包的 FP-

收稿日期：2016-04-11

基金项目：国家自然科学基金面上项目（61003036）；
计算机体系结构国家重点实验室开放课题（CARCH201301）；
中央高校基本科研业务经费专项基金（HEUCF100606）

作者简介：刘书勇（1978—），男，讲师。

通信作者：林俊宇，讲师，Email：linjunyu@hrbeu.edu.cn

GA 设计[13-15]。从实现结构角度来说，它们大多是针对某种具体矩阵分解的一维线性阵列及二维多边形阵列以不同并行算法加以实现[16]。这些设计在科学及工程应用领域已经得到广泛应用。但是，对 FPGA 矩阵三角化分解的研究仍有可提升的空间。从矩阵计算类属的角度考虑，专用的矩阵分解并行算法及阵列结构很难直接复用到其他矩阵分解的实现中，而以通用化为设计目标的结构则在计算性能上无法与专用阵列结构相比。此外，已有的一些矩阵分解细粒度并行算法往往需要对算法本身进行较大的改动，通过增加硬件操作原语来实现算法指令级的并行[17-18]。在设计一类具有共同计算特征的矩阵分解应用中，对计算过程和算法本身细粒度并行的分析和优化成为提高计算阵列性能的关键。矩阵三角化分解过程中均含有三角化计算，这一数值计算过程同样具有规律性较强的数据排列与流动特征，非常适合进行较细粒度的 FPGA 并行优化设计。

本文针对矩阵三角化分解中共有的三角化计算迭代过程，通过分析其中子矩阵更新过程的线性规律，提出一种具有一般性的子矩阵更新同一化并行算法，进而提出基于该算法的计算复杂度较低的矩阵三角化分解并行结构，针对 Cholesky 矩阵三角化分解在并行结构上的高性能实现及优化方法开展研究。

1 研究背景

本文提到的矩阵和向量均在实数范围内，矩阵皆为方阵。

单个矩阵的三角化分解将矩阵 $\boldsymbol{A}$ 分解为正交矩阵与三角矩阵之积，或分解为一个上三角矩阵与一个下三角矩阵之积。三角化过程可以总结为：给定 $n\times p(p\leqslant n)$ 维的矩阵 $\boldsymbol{X}$，根据式(1)求 $n\times n$ 维的非奇异矩阵 $\boldsymbol{M}$ 以及 $p\times p$ 维的上三角矩阵 $\boldsymbol{R}$。

$$\boldsymbol{MX}=\begin{bmatrix}\boldsymbol{R}\\0\end{bmatrix}. \tag{1}$$

关于子矩阵更新同一化算法请参考文[19]。

2 Cholesky 分解及 FPGA 并行结构设计

Cholesky 分解将 $n\times n$ 维对称正定矩阵 $\boldsymbol{A}$(其中元素为 a_{ij}，$1\leqslant i,j\leqslant n$)分解成 $\boldsymbol{A}=\boldsymbol{LL}^{\mathrm{T}}$ 的形式，$\boldsymbol{L}$(其中元素为 l_{ij}，$1\leqslant j\leqslant i\leqslant n$)为对角线元素为正数的下三角矩阵。$L$ 与 L^T 如式(2)所示。

$$\boldsymbol{L}=\begin{bmatrix}l_{11}&0&\cdots&0&\cdots&0\\l_{21}&l_{22}&\cdots&0&\cdots&0\\\vdots&\vdots&&\vdots&&\vdots\\l_{k1}&l_{k2}&\cdots&l_{kk}&\cdots&0\\\vdots&\vdots&&\vdots&&\vdots\\l_{n1}&l_{n2}&\cdots&l_{nk}&\cdots&l_{nn}\end{bmatrix},$$

$$\boldsymbol{L}^{\mathrm{T}}=\begin{bmatrix}l_{11}&l_{21}&\cdots&l_{k1}&\cdots&l_{n1}\\0&l_{22}&\cdots&l_{k2}&\cdots&l_{n2}\\\vdots&\vdots&&\vdots&&\vdots\\0&0&\cdots&l_{kk}&\cdots&l_{nk}\\\vdots&\vdots&&\vdots&&\vdots\\0&0&\cdots&0&\cdots&l_{nn}\end{bmatrix}. \tag{2}$$

$\boldsymbol{L}$ 可以通过式(3)和(4)求得。

$$l_{ij}=\begin{cases}a_{ij}/l_{jj}, & i>j=1;\\\left(a_{ij}-\sum\limits_{k=1}^{j-1}l_{ik}l_{jk}\right)\Big/l_{jj}, & i>j>1;\end{cases} \tag{3}$$

$$l_{ii}=\sqrt{a_{ii}-\sum_{k=1}^{i-1}l_{ik}^2}. \tag{4}$$

将 Cholesky 分解的执行过程由计算 $\boldsymbol{L}$ 的过程转化为计算 $\boldsymbol{L}^{\mathrm{T}}$ 的等价过程(简称 $\boldsymbol{L}^{\mathrm{T}}$-SC)。从矩阵几何特征的角度，$\boldsymbol{L}^{\mathrm{T}}$-SC 将更新矩阵求解下三角矩阵过程变为求解上三角矩阵过程，如图 1 所示，其中“×”为无关项。

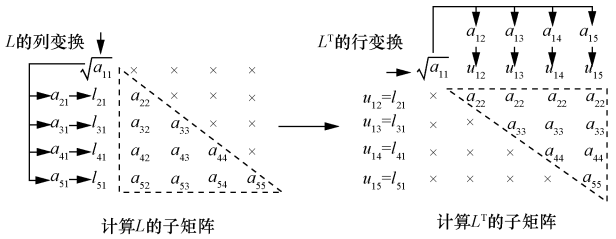


图 1 $\boldsymbol{L}^{\mathrm{T}}$-SC 的计算过程

因此，$\boldsymbol{L}^{\mathrm{T}}$-SC 基于列 k 的 1 次子矩阵更新亦可以表示为式(5)和(6)。

$$L^{\mathrm{T}}_{(1,1)}=\begin{bmatrix}u_{11}&u_{12}&u_{13}&u_{14}\\\times&a'_{22}&a'_{23}&a'_{24}\\\times&\times&a'_{33}&a'_{34}\\\times&\times&\times&a'_{44}\end{bmatrix},$$

$$\begin{bmatrix}u_{11}\\u_{12}\\u_{13}\\u_{14}\end{bmatrix}^{\mathrm{T}}=\begin{bmatrix}\sqrt{a_{11}}\\a_{12}/\sqrt{a_{11}}\\a_{13}/\sqrt{a_{11}}\\a_{14}/\sqrt{a_{11}}\end{bmatrix}^{\mathrm{T}}, \tag{5}$$

$$\boldsymbol{A}'=\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ \times & a'_{22} & a'_{23} & a'_{24} \\ \times & \times & a'_{33} & a'_{34} \\ \times & \times & \times & a'_{44} \end{bmatrix}=$$

$$\boldsymbol{T}_{1,1}\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & \times & a_{33} & a_{34} \\ a_{41} & \times & \times & a_{44} \end{bmatrix},$$

$$\boldsymbol{T}_{1,1}=\begin{bmatrix} 1 & 0 & 0 & 0 \\ -a_{21}/a_{11} & 1 & 0 & 0 \\ -a_{31}/a_{11} & 0 & 1 & 0 \\ -a_{41}/a_{11} & 0 & 0 & 1 \end{bmatrix}. \tag{6}$$

引入置换矩阵 $\boldsymbol{P}$，令：

$$\boldsymbol{P}=\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{7}$$

则 $\boldsymbol{L}^{\mathrm{T}}$-SC 的求解过程亦可以表示为式(8)的计算过程。$\boldsymbol{L}^{\mathrm{T}}$-SC 每一步骤的计算对象均为上三角矩阵，因此每完成 1 次矩阵更新，有效上三角矩阵的维数减 1。

$$\begin{cases} \boldsymbol{A}^{(t)}=\boldsymbol{P}(\boldsymbol{T}_{1,1}\boldsymbol{A}^{(t-1)})\boldsymbol{P}, & 0<t\leqslant n; \\ \boldsymbol{A}^{(0)}=\boldsymbol{A}. \end{cases} \tag{8}$$

将 $\boldsymbol{Z}$ 看成更新后的 $\boldsymbol{A}$，z_{ij} 为更新后的 a_{ij}。设：$z^{(t)}_{i-1,j-1}=g^{(t)}(i,j,k)=z^{(t-1)}_{i,j}-(z^{(t-1)}_{i,1}/z^{(t-1)}_{1,1})z^{(t-1)}_{1,j}$，$1<i,j\leqslant n$，$1\leqslant k\leqslant n$ 且 $0<t\leqslant n$，$\boldsymbol{Z}^{(0)}=\boldsymbol{Z}$。则使用子矩阵同一化并行算法后，子矩阵的更新过程可以表示为式(9)。

$$\boldsymbol{A}^{(1)}_{3\times 3}=$$

$$\begin{bmatrix} g^{(1)}(2,2,1) & g^{(1)}(2,3,1) & g^{(1)}(2,4,1) & \times \\ g^{(1)}(2,3,1) & g^{(1)}(3,3,1) & g^{(1)}(3,4,1) & \times \\ g^{(1)}(2,4,1) & \times & g^{(1)}(4,4,1) & \times \\ \times & \times & \times & \times \end{bmatrix}=$$

$$\boldsymbol{T}_{\mathrm{d}}\boldsymbol{A}_{4\times 4}=\boldsymbol{T}_{d}\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ \times & a_{22} & a_{23} & a_{24} \\ \times & \times & a_{33} & a_{34} \\ \times & \times & \times & a_{44} \end{bmatrix}, \tag{9a}$$

$$\boldsymbol{T}_{\mathrm{d}}=\begin{bmatrix} -a_{12}/a_{11} & 1 & 0 & 0 \\ -a_{13}/a_{11} & 0 & 1 & 1 \\ -a_{14}/a_{11} & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}=$$

$$\boldsymbol{P}\begin{bmatrix} 1 & 0 & 0 & 0 \\ -a_{12}/a_{11} & 1 & 0 & 0 \\ -a_{13}/a_{11} & 0 & 1 & 0 \\ -a_{14}/a_{11} & 0 & 0 & 1 \end{bmatrix}. \tag{9b}$$

$\boldsymbol{L}^{\mathrm{T}}$-SC 的计算依赖图如图 2 所示。在该依赖图中，以 k 轴的正方向为时间维度，有：$a^{(t)}_{i,j}$ 依赖于 $a^{(t)}_{i+1,\,j+1}$，$0<j\leqslant i\leqslant n$。即只更新除去右端边界列的子三角矩阵，且 $a^{(t)}_{1,1}$ 始终为下一次子矩阵更新计算的主元。根据 $\boldsymbol{L}^{\mathrm{T}}$-SC 的计算依赖图构建的阵列结构与时空图如图 3 所示。将子矩阵更新趟间依赖映射为连结斜上方邻接 PE 单元的输出边。为首行 PE 单元($a_{1,2}$，$a_{1,3}$，…，$a_{1,n}$)增加旁路边($a_{i,j}=a_{j,i}$)，以获取子矩阵行更新行变换关键向量计算的数据并行。对应的 $\boldsymbol{L}^{\mathrm{T}}$-SC 并行结构 PE 的配置和 $\boldsymbol{L}^{\mathrm{T}}$-SC 总体并行结构分别如图 4 和 5 所示。

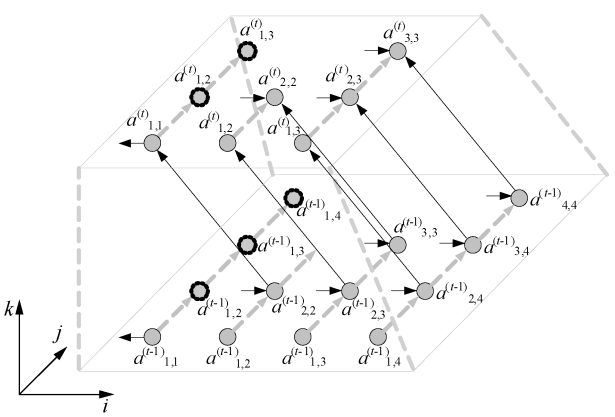


图 2 $\boldsymbol{L}^{\mathrm{T}}$-SC 的计算依赖图

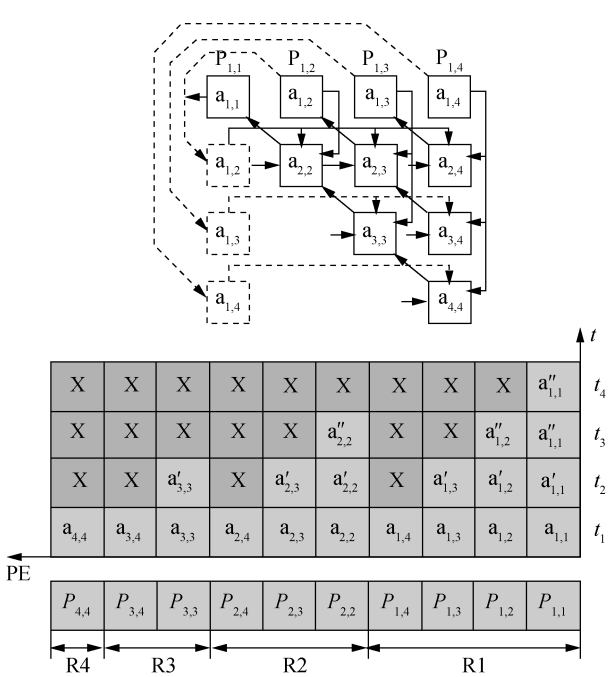
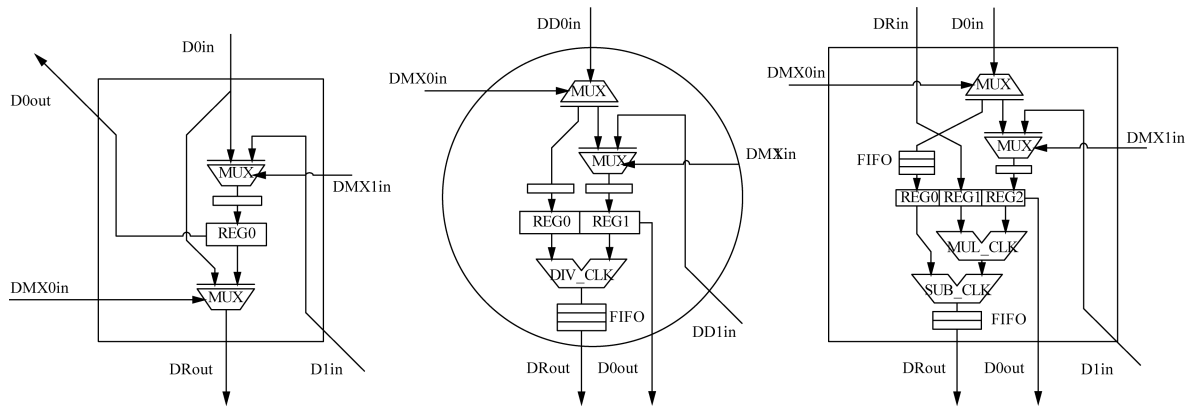


图 3 $\boldsymbol{L}^{\mathrm{T}}$-SC 的阵列结构与计算时空图



(a) 寄存单元rPE (b) 边界cPE (Boundary cPE-1) (c) 内部cPE (Internal cPE)

图 4 L^T-SC 并行结构 PE 的配置

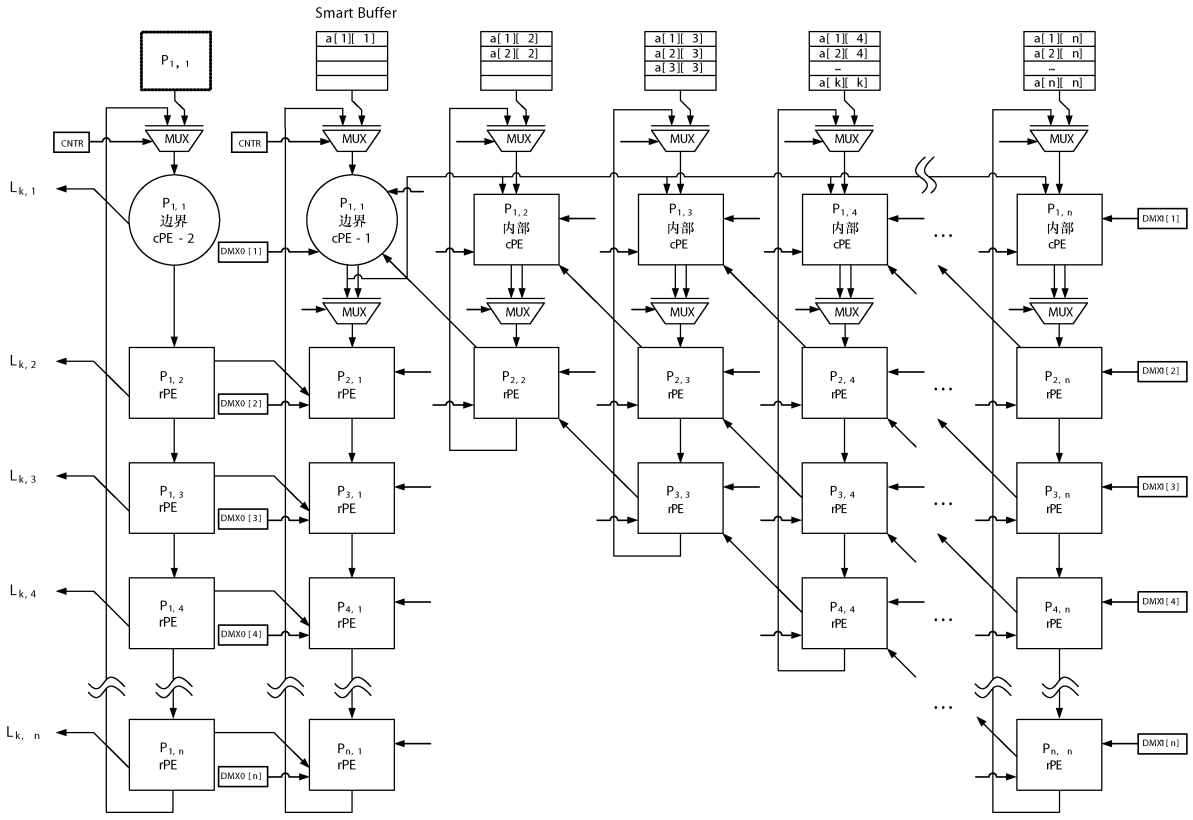


图 5 L^T-SC 总体并行结构

3 实验结果

本文提出的设计采用 VHDL 语言实现。使用 Mentor Graphic Modelsim v6.5 仿真验证工具进行仿真后在 Xilinx ISE 11.1 工具中完成综合，并给出资源占用结果；对设计的性能进行估计与分析；最后将设计加载到 FPGA 上进行实际运行与功能验证。所使用的 FPGA 分别为 xc5vlx330t 和 xc5vlx110t。

3.1 FPGA 资源占用

本文涉及的浮点计算如加法、减法、乘法、除法以及开平方等均使用开源算术浮点库 libhdlfltp 的单精度流水浮点计算部件实现，遵照 IEEE754 标准的浮点数格式。

表 1 给出 L^T-SC 并行结构中 PE 单元的综合结果，包括 BcPE-1、IcPE、BcPE-2 和 rPE 等 4 种 PE 单元的资源使用和最大频率。可以看到，因内部没有配置浮点运算部件，rPE 的资源占用非常少。

表 1 LT-SC 并行结构中 PE 单元的综合结果

单元	寄存器个数	查找表个数	查找表触发器个数	DSP48E 个数	最大频率/MHz
BcPE-1	1 156	1 212	1 228	0	188.56
IcPE	338	847	868	2	169.21
BcPE-2	2 034	1 942	2 054	0	150.76
rPE	34	40	42	0	550.01

3.2 性能比较

用于比较的台式计算机配置为主频 2.33 GHz 英特尔酷睿 2 处理器、4 GB 主存及 CentOS 5.5 操作系统。C 语言编译器为 gcc 4.3，优化选项设为 O2。图 6 给出 $\boldsymbol{L}^{\mathrm{T}}$-SC 阵列并行结构与通用处理器上软件实现相比所获得的加速比和最大频率随矩阵规模(阵列规模)增加的变化。可以看到，在各个矩阵规模上的计算中 $\boldsymbol{L}^{\mathrm{T}}$-SC 阵列并行结构均可以取得 10 倍以上的加速比，即使最大频率在降低，加速比也随着矩阵规模的增加逐步提升。需要指出的是，将矩阵规模从 10×10 维增加到 18×18 维过程中，随着 IcPE 数量的大量增加，FPGA 上的 DSP48E Slice 资源全部消耗，故最大频率迅速下降。而矩阵规模接近 32×32 维时，随着 FPGA 上 LUT Flip-Flop pairs 资源的耗尽，最大频率大幅度下降。

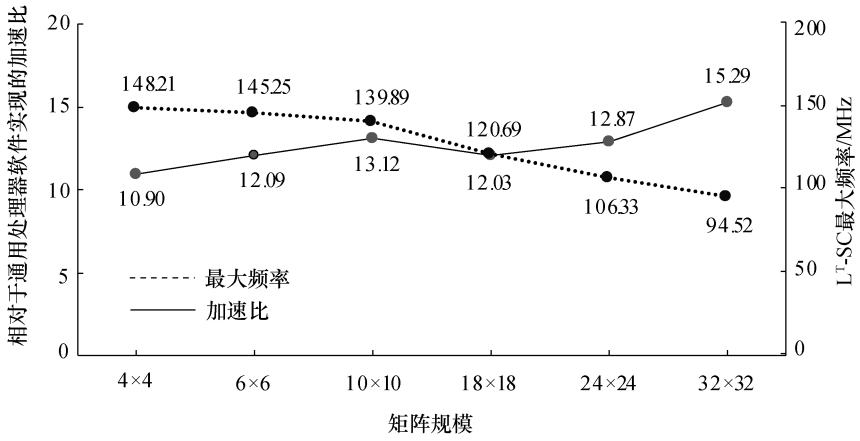


图 6 $\boldsymbol{L}^{\mathrm{T}}$-SC 阵列并行结构与通用处理器软件实现的加速比和最大频率随矩阵规模增加的变化

4 结 论

本文基于子矩阵更新同一化算法，针对 Cholesky 分解的求解特征，提出以求解矩阵 $\boldsymbol{L}$ 的转置阵 $\boldsymbol{L}^{\mathrm{T}}$ 的 $\boldsymbol{L}^{\mathrm{T}}$-SC 算法，并在分析 $\boldsymbol{L}^{\mathrm{T}}$-SC 线性代数特征的基础上，给出了 $\boldsymbol{L}^{\mathrm{T}}$-SC 的并行结构实现方案。在并行结构的具体实现中，采用 FPGA 协处理器浮点数学库 libhdlfltp 支持阵列 PE 的浮点流水运算。理论分析表明，该算法针对矩阵三角化计算过程具有更高的数据并行性与流水并行性；实验结果表明，针对 Cholesky 分解，基于 $\boldsymbol{L}^{\mathrm{T}}$-SC 算法在 FPGA 上的实现在核心计算性能上优于在通用处理器上的软件实现，可达到 10 倍以上的加速比。但作为实际运行硬件的可重构计算验证平台中的通信开销所占据整体运行时间比例过大，已经成为整体计算加速的瓶颈，下一步应加以改进。

参考文献 (References)

[1] Satchidanand G H, Sotirios G Z. FPGA implementation of a Cholesky algorithm for a shared-memory multiprocessor architecture [J]. *Parallel Algorithms and Applications*, 2004, **19**(4): 211-226.

[2] Kung H T, Leiserson C E. Systolic arrays technical report [R]. Pitsburg, PA, USA: Carneige Mellon University, 1978.

[3] YANG Depeng, Gregory D P, LI Husheng. Compressed sensing and Cholesky decomposition on FPGAs and GPUs [J]. *Parallel Computing*, 2012, **38**(8): 421-437.

[4] Alfredo B, Julien L, Jakub K, et al. A class of parallel tiled linear algebra algorithms for multicore architectures [J]. *Parallel Computing*, 2009, **35**(1): 38-53.

[5] DOU Yong, Vassiliadis S, Kuzmanov G, et al. 64-bit floating-point FPGA matrix multiplication [C]//Proceedings of the 13th ACM/SIGDA International Symposium on Field Programmable Gate Arrays(FPGA'05), New York, NY, USA: ACM, 2005: 86-95.

[6] Vasily V, James W D. LU, QR and Cholesky factorizations using vector capabilities of GPUs [R]. Berkeley, CA, USA: EECS Department, University of California, Berkeley, 2008.

[7] Oleg M, Volodymyr L, Anatoli S, et al. Parallel implementation of Cholesky LLT-algorithm in FPGA-based processor [J]. *Parallel Processing and Applied Mathematics*, 2008, **49**(67): 137-147.

[8] Eunice E S, CHU Peiyun. Efficient and optimal parallel algorithms for Cholesky decomposition [J]. *Journal of Mathematical Modeling and Algorithms*, 2003, **2**(3): 217-234.

[9] Aatonio R, George A. A high throughput FPGA-based floating point conjugate gradient implementation for dense matrices [J]. *ACM Transactions on Reconfigurable Technology and Systems*, 2010, **3**(1): 1-19.

[10] Badia J M, Vidal A M. Parallel algorithms to computer the eigenvalues and eigenvectors of symmetric Toeplitz matrices [J]. *Parallel Algorithms and Applications*, 1998, **13**(1): 75-93.

[11] Anderson E, BAI Z, Bischof C, et al. LAPACK Users Guide [M]. 3rd ED. Philadelphia, PA, USA: The society of industrial and applied mathematics, 1999.

[12] Blackford L S, Choi J, Cleary A, et al. ScaLAPACK Users Guide [M]. Philadelphia, PA, USA: The Society of Industrial and Applied Mathematics, 1997.

[13] WU Guiming, DOU Yong, SUN Junqing, et al. A high performance and memory efficient LU decomposer on FPGAs [J]. *Computers, IEEE Transactions*, 2012, **61**(3): 366-378.

[14] WU Guiming, DOU Yong, Gregory D P. Blocking LU decomposition for FPGAs [C]//Proceedings of the 18th IEEE Symposium on Field-Programmable Custom Computing Machines(FCCM'10). Charlotte, NC,USA: IEEE, 2010, 109-112.

[15] Haridas S G, Sotirios G Z. FPGA implementation of a Cholesky algorithm for a shared memory multiprocessor architecture [J]. *International Journal of Parallel, Emergent and Distributed Systems*, 2004, **19**(4): 211-226.

[16] 郭磊, 唐玉华, 周杰, 等. 基于 FPGA 的 Cholesky 分解细粒度并行结构与实现 [J]. 计算机研究与发展, 48 (Suppl), 2011: 258-265.
GUO Lei, TANG Yuhua, ZHOU Jie, et al. Fine-grained architecture and implementation for Cholesky decomposition on FPGA [J]. *Journal of Computer Research and Development*, 2011, **48**(suppl): 258-265. (in Chinese)

[17] WANG Xiaojun, Leeser M. A truly two-dimensional systolic array FPGA implementation of QR decomposition [J]. *ACM Transactions on Embedded Computing Systems* (*TECS*), 2009, **9**(1): 17-18.

[18] 邬贵明, 窦勇, 王淼. Cholesky 分解细粒度并行算法 [J]. 计算机工程与科学, 2010, **32**(9): 102-106.
WU Guiming, DOU Yong, WANG Miao. A fine-grained parallel algorithm for the Cholesky decomposition [J]. *Computer Engineering & Science*, 2010, **32**(9): 102-106. (in Chinese)

[19] 刘书勇,吴艳霞,张博文,等. 基于可重构计算系统的矩阵三角化分解硬件并行结构研究 [J]. 电子学报, **43**(8), 2015: 1642-1650.
LIU Shuyong, WU Yanxia, ZHANG Bowen, et al. Research of parallel hardware architecture for matrix triangularization decomposition based on reconfigurable computing system [J]. *ACTA Electronica Sinica*, 2015, **43** (8): 1642-1650. (in Chinese)

(上接第 962 页)

[5] Vogt P, Nentwich F, Jovanovic N, et al. Cross site scripting prevention with dynamic data tainting and static analysis [C]// Proceedings of the 14th Annual Network and Distributed System Security Symposium. San Diego, CA, USA: Internet Society, 2007.

[6] Minded Security. DOMinatorPro: Securing next generation of Web applications [Z/OL]. (2012-09-30). [2015-04-07]. https://dominator.mindedsecurity.com.

[7] Lekies S, Stock B, Johns M. 25 million flows later: Large-scale detection of DOM-based XSS [C]// Proceedings of the 20th ACM Conference on Computer and Communications Security. New York, NY, USA: ACM, 2013: 1193-1204.

[8] Saxena P, Hanna S, Poosankam P, et al. FLAX: Systematic discovery of client-side validation vulnerabilities in rich Web applications [C]// Proceedings of the 17th Annual Network and Distributed System Security Symposium. San Diego, CA, USA: Internet Society, 2010.

[9] Phung P H, Sands D, Chudnov A. Lightweight self-protecting JavaScript [C]// Proceedings of the 4th International Symposium on Information, Computer, and Communications Security. New York, NY, USA: ACM, 2009: 47-60.

[10] International Secure Systems Lab. NoMoXSS[Z/OL]. (2006-3-29). [2015-04-07]. http://seclab.tuwien.ac.at/projects/jstaint/files/testing.zi