Operations Research and Fuzziology 运筹与模糊学, 2024, 14(3), 102-108
Published Online June 2024 in Hans. https://www.hanspub.org/journal/orf
https://doi.org/10.12677/orf.2024.143249

Hans汉斯

求解电路方程组的改进分块对角加边方法

陈炳旭

北京邮电大学理学院，北京

收稿日期：2024年3月18日；录用日期：2024年6月5日；发布日期：2024年6月13日

摘　要

针对电路仿真中瞬态分析底层产生的超大规模稀疏线性方程组的求解，分块对角加边(Bordered Block Diagonal, BBD)方法是一类经典的方法。本文提出了一种改进的BBD方法，通过在边界分解时引入以列为基础单位的动态并行分解，缓解了经典BBD方法中线程负载不均的问题，同时增强了并行性。使用8个电路方程的矩阵进行了数值实验，实验结果显示对于测试矩阵的LU分解速度，本文提出的改进方法在2线程和8线程情况下相比经典BBD方法均有一定的提升。

关键词

稀疏LU分解，电路仿真，BBD方法

Improved Bordered Block Diagonal Method for Solving Circuit Equation Systems

Bingxu Chen

School of Science, Beijing University of Posts and Telecommunications, Beijing

Received: Mar. 18th, 2024; accepted: Jun. 5th, 2024; published: Jun. 13th, 2024

Abstract

The Bordered Block Diagonal (BBD) method is a classic approach for solving the large and sparse linear equation systems generated at the bottom level in transient analysis of circuits simulation. In this paper, an improved BBD method is proposed, which introduces a fine-grained parallel decomposition based on columns during the boundary decomposition to alleviate the issue of uneven thread workload in the classical BBD method, while enhancing parallelism. Numerical experiments were conducted using a matrix of 8 circuit equations, and the results show that the pro-

文章引用: 陈炳旭. 求解电路方程组的改进分块对角加边方法[J]. 运筹与模糊学, 2024, 14(3): 102-108.
DOI: 10.12677/orf.2024.143249

posed improvement method achieves a certain speedup in LU decomposition for the test matrix compared to the classical BBD method when using 2-thread and 8-thread scenarios.

Keywords

Sparse LU Decomposition, Circuit Simulation, Bordered Block Diagonal Method

1. 引言

随着现代集成电路技术蓬勃发展，电子设计自动化(Electronic Design Automation, EDA)成为了电子领域里的重要课题。EDA 指的是是指利用计算机辅助设计软件，来完成超大规模集成电路芯片的功能设计、综合、验证、物理设计(包括布局、布线、版图、设计规则检查等)等流程的设计方式，它极大促进了半导体工业的发展。现代超大规模集成电路的发展趋势是将更多的功能集成到单个芯片中。为了促进这一点，现代集成电路的规模非常大，电子系统也一代又一代地变得更加复杂。此外，电子设备正在频繁升级，芯片供应商面临着上市时间的巨大挑战。如果不能以尽量短的周期设计高质量的电路，就无法应对市场的挑战，因此高质量的电路仿真是必不可少的。

电路仿真是利用计算机软件模拟和分析电路的行为和性能的过程，是 EDA 的核心技术之一。通过电路仿真，工程师可以在实际制造之前评估电路的功能、稳定性和性能，从而节省时间和成本。现代的电路仿真基于 SPICE (Simulation Program with Integrated Circuit Emphasis)仿真器[1]。对于瞬态分析的仿真是 SPICE 仿真的核心之一，其底层是一系列稀疏线性方程组。

分块对角加边(Bordered Block Diagonal, BBD)方法是一类用来求解稀疏线性方程组的经典并行方法。经典 BBD 方法的思路是将方程组的系数矩阵通过行列重排序转化为 BBD 结构，然后再使用 LU 分解对方程组进行求解，并且在 LU 分解时利用 BBD 的分块结构对 LU 分解进行并行加速。该方法的分块结构为并行分解提供了良好的数据局部性，但同时也导致了以负载不均衡为主的一系列缺点。电路方程组的系数矩阵极其稀疏，并且电路本身就可以根据功能模块进行分块，这些特点导致电路方程组的系数矩阵可以较为容易地通过行列变换转化为高质量的 BBD 结构，因此从理论上看，该方法对于电路方程组有着良好的适配性。然而，其 BBD 结构也导致了负载不均衡等缺点，对求解的速度产生不良影响。因此，本文的研究希望能改进经典的 BBD 方法，克服它的缺点以更快地求解电路方程组。

2. 背景

2.1. 研究意义

现代电子设计自动化(Electronic Design Automation, EDA)的电路设计中，SPICE (Simulation Program with Integrated Circuit Emphasis)仿真器[1]的诞生彻底改变了电路的设计流程。SPICE 是一个通用的集成电路仿真器，它能够在实际建造电路之前，在各种条件和场景下通过仿真来模拟电路的能力，从而节省大量时间和成本。各类基于 SPICE 的仿真器层出不穷，极大地推动了 EDA 行业的发展，SPICE 类仿真器已经成为了标准的晶体管级集成电路仿真工具。

EDA 技术和半导体工业的发展带动了现代超大规模集成电路产业，其发展趋势是将更多的功能集成

到单个芯片中。因此集成电路的规模越来越大、系统的复杂性也越来越强，对电路仿真器的速度和精度提出了挑战。

基于 SPICE 的现代电路仿真器能够提供各种各样的电路分析，包括直流(Direct Current, DC)分析、交流(Alternating Current, AC)分析、瞬态分析、噪声分析、灵敏度分析、极点－零点分析等。其中，计算静态工作点的直流分析是几乎所有其他模拟的基本起点。瞬态分析(Transient Analysis)，或称时域仿真，它是对电路从进入瞬态开始到下一个稳态时间点之间电路上电信号变化的模拟，是 SPICE 提供的所有仿真功能中，在模拟和混合信号电路设计和验证中使用最广泛一个。本文研究瞬态仿真底层稀疏线性方程组的加速求解，对实际 SPICE 类仿真器的发展乃至整个集成电路产业的发展具有重要意义。

2.2. 相关工作

国内外许多学者对电路方程组的求解进行了研究，开发了多种多样的求解算法[2] [3] [4]，主流的方法可以分为直接法和迭代法两大类。相比于直接法，迭代法具有内存需求少、并行化较容易等优点。但在电路仿真中，对求解线性方程组的迭代方法的研究较少，这是因为迭代法在电路方程组的求解中，其收敛性和鲁棒性可能产生问题。直接法在并行理论上和实际实现上的困难程度都要高于迭代法，但直接法可以应对更多更广类型的问题。同时，瞬态分析的仿真，其底层由通过牛顿－拉夫逊方法得到的一系列稀疏线性方程组构成。这一系列线性方程组具有相同的非零结构，因此直接法能够重复利用得到的 LU 因子非零结构以节省大量的时间，这也是在这类问题上直接法相比迭代法的一大优势。

BBD 方法是一类求解稀疏线性方程组的经典直接法[5] [6] [7] [8] [9]，它来源于有限元方程中区域分解理论。经典的 BBD 方法具有良好的数据局部性，但其并行结构容易导致并行负载不均衡问题。本文基于经典的 BBD 方法进行改进，平衡了各个线程的负载并增加并行性，希望以此来加速电路稀疏方程组的求解。

3. 经典分块对角加边方法

考虑电路方程组的一般形式：

$$Ax = b$$

其中系数矩阵 A 为稀疏的 n 维非奇异方阵，x 为 n 维未知向量，b 为 n 维右端项。经典 BBD 方法会基于图划分的理论，通过行列重排序将系数矩阵转化为如下所示的具有分块结构的矩阵：

$$\begin{bmatrix} D_1 & & & C_1 \\ & \ddots & & \vdots \\ & & D_p & C_p \\ R_1 & \cdots & R_p & G \end{bmatrix}$$

这种“箭头”形结构就被称为 BBD 结构。其中，$n_i \times n_i$ 维矩阵 D_i 称为对角块，$n_G \times n_i$ 维矩阵 R_i 称为行耦合块，$n_i \times n_G$ 维矩阵 C_i 称为行耦合块，$i = 1,2,\cdots,p$，$n_G \times n_G$ 维矩阵 G 称为边界块，且有 $\sum_{i=1}^{p} n_i + n_G = n$。具有 BBD 结构的矩阵称为 BBD 矩阵。

对系数矩阵的行列变换实质上将原方程组转化为了如下方程组：

$$(PAQ)\left(Q^{-1}x\right) = Pb$$

其中，P 是行置换矩阵，Q 是列置换矩阵。经典 BBD 方法通过 LU 分解求解该方程组得到 $\left(Q^{-1}x\right)$，然后再通过简单的转换就可以得到原始方程组的解。经典 BBD 方法中作为重点的 LU 分解如下所示：

$$\begin{bmatrix} D_1 & & & C_1 \\ & \ddots & & \vdots \\ & & D_p & C_p \\ R_1 & \cdots & R_p & G \end{bmatrix} = \begin{bmatrix} L_1 & & & \\ & \ddots & & \\ & & L_p & \\ L_{R1} & \cdots & L_{Rp} & L_G \end{bmatrix} \begin{bmatrix} U_1 & & & U_{C1} \\ & \ddots & & \vdots \\ & & U_p & U_{Cp} \\ & & & U_G \end{bmatrix}$$

通过其结构，我们可以发现，在有 p 个线程的情况下，每个线程都可以分配到一个对角块、一个行耦合块、一个列耦合块，并且每个线程所分配到的块的处理与其他线程分配的块完全无关，因此这些线程对对角块、行耦合块、列耦合块的处理是可以并行的。假设使用列压缩存储方式对矩阵进行存储，则经典 BBD 方法的 LU 分解过程如图 1 所示。

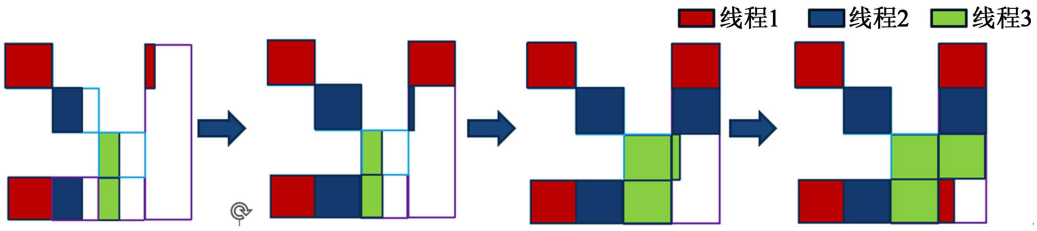

Figure 1. Classic BBD method LU decomposition process
图 1. 经典 BBD 方法 LU 分解流程

图中不同的颜色代表不同的线程。各线程从左到右以列为单位分别处理自己分配到的任务。在所有对角块、行耦合块、列耦合块的处理全部完成后，最后由一个线程对边界块进行顺序分解。这就是经典 BBD 方法的 LU 分解。

它的一大优势是更新与分解的分块完成。这使得在实际实现中，任意块(包括对角块、耦合块、边界块)的内部都可以存储在一大段连续的内存空间中，因此拥有较好的数据局部性，这在计算机实现的难易度以及程序运行速度上都有一定的优势。但它的缺点也很明显：于各个对角块和各个耦合块的维度和稀疏性的不同，每个线程分配到的任务量也不同，因此线程完成任务的时间也不同。则当后完成任务的线程还在工作时，先完成的线程将会无事可做，这就是线程负载不均导致的计算资源浪费。

4. 改进分块对角加边方法

针对线程负载不均的缺点，本文提出的改进方法选择在列耦合块和边界的处理上进行改进。改进方法的 LU 分解过程如图 2 所示：

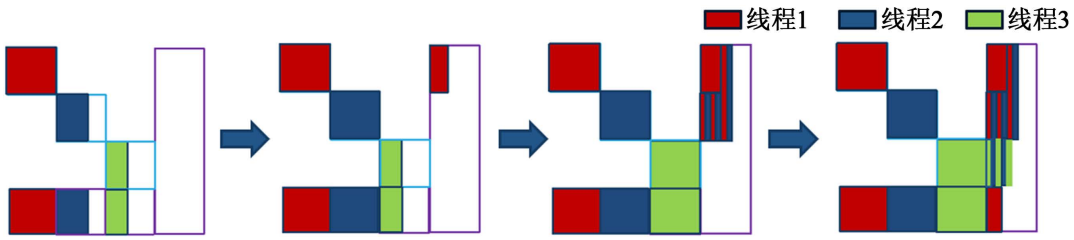

Figure 2. Improved BBD method LU decomposition process
图 2. 改进 BBD 方法 LU 分解流程

在分解开始前，本方法会通过静态符号分解对各线程在对角块和行耦合块的工作量进行估计，然后进行重排序使得对角块和行耦合块的工作量从左到右递增，然后再开始 LU 分解。在分解的开始，与经典 BBD 方法相同，各线程按列分解分配到的任务；然后，当有些线程完成了对角块与行耦合块的处理后，

它们会一起对列耦合块中已经可以处理的部分，即左侧的对角块已完成处理的那些列耦合块部分，并行进行处理，由 LU 分解的数值依赖关系[10]可知这些列的并行处理并不会造成冲突；最后，当所有线程都完成了对角块与行耦合块的处理后，将边界块分配给某一线程进行处理，其他线程共同处理列耦合块部分，即列耦合块和边界块的块间并行、列耦合块内部的块内列并行同时进行。该改进方法在列耦合块处进行的以列为单位的动态并行分解，使得在对角块与行耦合块阶段工作量较少的线程在进行列耦合块分解时能分配到更多的任务，能够有效地缓解线程负载不均问题。

需要注意的是，在最后的列耦合块与边界块的并行分解阶段，可能会发生边界块分解进度将要超过列耦合块的情况，如图 3 所示：

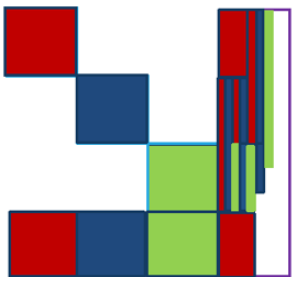

Figure 3. Conflict in the parallel processing of boundary block and column coupling block
图 3. 并行处理边界块与列耦合块时的冲突

此时边界块中将要对下一列进行处理，但该列在列耦合块中还未完成处理，由 LU 分解的数值依赖关系可知，边界块中的列的处理需要在列耦合块中的同一列已处理完成后才能进行，因此此时边界块的处理不得不暂停，因此浪费了线程的计算资源。对于这种情况，本方法的处理方式如图 4：

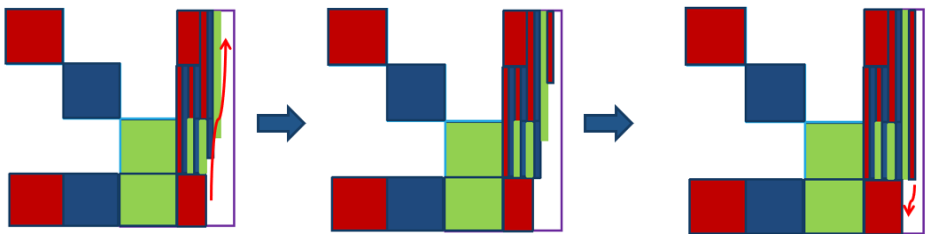

Figure 4. Using dynamic scheduling to solve the problem
图 4. 使用动态调度解决问题

将负责处理边界块的线程调度到列耦合块中，进行列耦合块待处理列中的新一列的处理，然后等列耦合块中新一列和原先的列的处理都完成后，再将该线程调度回边界块的处理。以这样的方式保证线程的忙碌，使计算资源的浪费问题在一定程度上得到了缓解。

5. 数值实验

数值实验中使用的经典 BBD 方法以及本文提出的改进 BBD 方法均用 C++实现，编译时不使用优化。所有测验均在配备 AMD Ryzen 7 4800H 的 Windows 环境下进行。

5.1. 测试用例介绍

数值实验使用的测试用例来自实际的电路方程组，共 8 个矩阵，它们都是非奇异且高度稀疏的，其基本信息如表 1 所示。

Table 1. Basic information of test cases
表 1. 测试用例基本信息

名称	维度	总非零元数	平均每列非零元数
autols	33,822	186,186	5.50
aaadc	194,262	1,289,026	6.64
avdac	228,366	2,182,647	9.56
buck	250,750	1,828,185	7.29
dll	234,405	2,046,883	8.73
serdes	549,375	2,616,120	4.76
hlmc	704,359	3,992,118	5.67
topsim	261,895	2,864,714	10.94

5.2. LU 分解速度比较

在进行 LU 分解速度比较前，我们会通过预处理将所有矩阵通过行列变换转换为 BBD 矩阵，并且根据其线程数 p 调整对角块数量使其一致。预处理完成后，我们测试并比较 2 线程经典 BBD 方法、2 线程改进 BBD 方法、8 线程经典 BBD 方法、8 线程改进 BBD 方法的 LU 分解速度。对于每个测试用例，每种方法执行 30 次 LU 分解，将结果的分解时间取平均值，单位为毫秒，结果如表 2 所示：

Table 2. Comparison of LU decomposition speed
表 2. LU 分解速度比较

测试用例	2 线程经典 BBD	2 线程改进 BBD	8 线程经典 BBD	8 线程改进 BBD
autols	45	42	29	24
aaadc	65	69	48	53
avdac	1907	1833	1053	882
buck	403	391	167	158
dll	3258	3247	1439	1304
serdes	868	856	583	650
hlmc	2169	2143	305	297
topsim	4283	4172	2998	2491

从实验结果可以看出，在 2 线程时和 8 线程时，本文提出的改进 BBD 方法在大部分测试用例上表现好于经典 BBD 方法。

6. 结论

对于电路方程组的快速求解问题，本文提出了一种改进的分块对角加边方法。通过分析，我们发现经典 BBD 方法存在线程负载不均的缺点。因此在经典 BBD 方法的基础上，本文提出的改进方法在列耦合块与边界分解时引入以列为基础单位的动态并行分解，缓解了负载不均问题。数值实验表明，相比经典 BBD 方法，改进方法在 LU 分解速度上有一定优势。

参考文献

[1] Nagel, L.W. and Pederson, D.O. (1973) SPICE (Simulation Program with Integrated Circuit Emphasis). Memo No.

ERL-M382, Electronics Research Laboratory, University of California, Berkeley.

[2] Li, P. (2012) Parallel Circuit Simulation: A Historical Perspective and Recent Developments. *Foundations and Trends® in Electronic Design Automation*, **5**, 211-318. https://doi.org/10.1561/1000000020

[3] Thornquist, H.K. and Rajamanickam, S. (2014) A Hybrid Approach for Parallel Transistor-Level Full-Chip Circuit Simulation. In: *International Conference on High Performance Computing for Computational Science*, Springer International Publishing, 102-111. https://doi.org/10.1007/978-3-319-17353-5_9

[4] Bollhöfer, M., Schenk, O., Janalik, R., Hamm, S. and Gullapalli, K. (2020) State-of-the-Art Sparse Direct Solvers. In: *Parallel Algorithms in Computational Science and Engineering*, Birkhäuser, 3-33. https://doi.org/10.1007/978-3-030-43736-7_1

[5] Chan, K.W. (2001) Parallel Algorithms for Direct Solution of Large Sparse Power System Matrix Equations. *IEE Proceedings—Generation, Transmission and Distribution*, **148**, 615-622. https://doi.org/10.1049/ip-gtd:20010583

[6] Zecevic, A.I. and Siljak, D.D. (1994) Balanced Decompositions of Sparse Systems for Multilevel Parallel Processing. *IEEE Transactions on Circuits and Systems I*: *Fundamental Theory and Applications*, **41**, 220-233. https://doi.org/10.1109/81.273921

[7] Duff, I.S. and Scott, J.A. (2005) Stabilized Bordered Block Diagonal Forms for Parallel Sparse Solvers. *Parallel Computing*, **31**, 275-289. https://doi.org/10.1016/j.parco.2004.12.008

[8] Rajamanickam, S., Boman, E.G. and Heroux, M.A. (2012) ShyLU: A Hybrid-Hybrid Solver for Multicore Platforms. 2012 *IEEE* 26*th International Parallel and Distributed Processing Symposium*, Shanghai, 21-25 May 2012, 631-643. https://doi.org/10.1109/IPDPS.2012.64

[9] Li, L., Liu, Z., Liu, K., Shen, S. and Yu, W. (2023) Parallel Incomplete LU Factorization Based Iterative Solver for Fixed-Structure Linear Equations in Circuit Simulation. *Proceedings of the* 28*th Asia and South Pacific Design Automation Conference*, 339-345. https://doi.org/10.1145/3566097.3567882

[10] Gilbert, J.R. and Peierls, T. (1988) Sparse Partial Pivoting in Time Proportional to Arithmetic Operations. *SIAM Journal on Scientific and Statistical Computing*, **9**, 862-874. https://doi.org/10.1137/0909058