DOI:10.3969/j.issn.1003-5060.2025.05.006

大规模 BBDF 稀疏矩阵求解硬件加速器

张多利， 孙贺云， 胡 锐
(合肥工业大学 微电子学院,安徽 合肥 230601)

摘 要:针对现有电力系统仿真方程求解加速效果有限、内存消耗大等问题,文章设计并完成了一种大规模分块对角加边形式(bordered block diagonal form,BBDF)稀疏矩阵求解器。采用稳定双共轭梯度迭代法,根据电力系统方程的分块对角加边特征进行分块计算,提出系数矩阵嵌套行压缩存储策略,显著降低计算过程中的存储访问负担和内存占用;优化算法任务,缩减算法的执行时间;利用可重构技术将多个任务中相似的组合计算、单个任务中相同的计算动态分配到同一计算电路,形成多层级复用的折叠式计算结构,实现求解器计算资源的高效利用;采用并行、流水等多种方法挖掘并行度,加速方程求解。实验结果表明,该求解器支持多种系数矩阵具有分块对角加边特征的大规模稀疏线性方程组的求解,相较于已有工作,能以更少的硬件资源收获 30～32 倍的加速比。

关键词:分块对角加边形式(BBDF);双稳态共轭梯度(BiCGStab)算法;嵌套行压缩存储;折叠技术;可重构技术;可编程门阵列(FPGA)
中图分类号:TN47 **文献标志码**:A **文章编号**:1003-5060(2025)05-0614-08

Hardware accelerator for solving large-scale BBDF sparse matrix

ZHANG Duoli, SUN Heyun, HU Rui
(School of Microelectronics, Hefei University of Technology, Hefei 230601, China)

Abstract: Aiming at the problems of limited acceleration effect and large memory consumption of existing power system simulation equation solving, this paper completed the design of large-scale bordered block diagonal form (BBDF) sparse matrix solver. Using the stable biconjugate gradient iteration method, block calculation is carried out based on the block diagonal edge feature of the power system equation. A coefficient matrix nested row compression storage strategy is proposed, which significantly reduces the storage access burden and memory occupation during the calculation process. The algorithm tasks are optimized and the algorithm execution time is reduced. By utilizing reconfigurable technology, similar combination calculations from multiple tasks and the same calculations from a single task are dynamically allocated to the same computing circuit, forming a multi-level reusable foldable computing structure, achieving efficient utilization of solver computing resources. Multiple methods such as paralleling and pipelining are used to explore parallelism and accelerate equation solving. The experimental results show that the solver supports the solution of large-scale sparse linear equations with block diagonal edge features for multiple coefficient matrices, and achieves an acceleration ratio of 30-32 times with less hardware resources compared to existing work.

Key words: bordered block diagonal form(BBDF); biconjugate gradient stabilized(BiCGStab) algorithm; nested row compression storage; folding technology; reconfigurable technology; field programmable gate array (FPGA)

收稿日期:2023-04-28;**修回日期**:2023-06-02
基金项目:国家自然科学基金资助项目(61874156);安徽省高校协同创新资助项目(GXXT-2019-030)
作者简介:张多利(1976—),男,黑龙江七台河人,博士,合肥工业大学研究员,硕士生导师,通信作者,E-mail:zhangduoli@hfut.edu.cn.

0 引　言

电力系统全过程动态仿真将电力系统机电暂态、中期和长期动态过程有机结合，描述系统受到扰动之后整个连续的动态过程。应用吉尔算法将描述电力系统相关元件动态特性的微分方程转化成差分方程，采用牛顿法对差分方程和网络代数方程进行联立求解是电力系统动态仿真目前公认的处理方式，在此过程中大规模稀疏线性方程的求解成为了关键问题[1-3]。

业界进行了大量研究工作以加速电力系统方程（一种特殊的稀疏矩阵）的求解。文献[4]基于因子表路径树引入网络分割方法，将电力网络方程写成分块对角加边形式（bordered block diagonal form，BBDF），发展了电力网络方程的并行求解方法；文献[5]依据暂态稳定联立方程的双层 BBDF 结构，将动态元件、分区系统的计算分配到多核 CPU 进行处理，采用双稳态共轭梯度（biconjugate gradient stabilized，BiCGStab）算法将边界系统的计算分配到图形处理器（graphics processing unit，GPU）进行并行加速计算，达到了 7 倍的加速效果；文献[6]采用 BiCGStab 迭代法求解线性方程组的潮流算法，利用 CPU-GPU 异构运算架构，将 GPU 用于并行处理计算量密集的线性方程组求解，将 CPU 用于处理潮流算法的其他步骤，收获了 11.8 倍的平均加速。

上述研究从多个角度对电力系统方程的求解进行了优化和加速，但加速比有限，且代价昂贵，所需的算力、存储量不容忽视。文献[7]针对共轭梯度算法软件执行效率低、实时性差的缺点提出一种基于可编程门阵列（field programmable gate array，FPGA）的共轭梯度求解器；文献[8]和文献[9]针对潮流计算，先、后提出并发展了静态调度算法，将稀疏矩阵求解过程中的三角分解、前代、回代过程静态映射到多个 FPGA 的处理单元，最终在 256 个基本单元（primary element，PE）配置下收获了 5.9 倍的加速比。

FPGA 具备加速稀疏矩阵求解的可行性，但目前相关的研究较少，且已知研究未能结合相关领域方程系数矩阵的特点充分发挥 FPGA 的并行计算优势。采用 FPGA 进行定制化设计，对于提升电力系统仿真的速度和效率具有重大价值[10]。

线性方程求解分为直接法和迭代法，根据电力系统仿真阶段的不同，2 种方法均有各自的适用场景[11-12]。本文的研究对象为电力系统全过程动态仿真中的雅可比矩阵的求解，该矩阵具有稀疏 BBDF 特征，且方程阶数高、模拟的设备和系统种类众多，需考虑中长期动态仿真的复杂模型，系数矩阵在每个计算周期都发生变化[13]。直接求解法中常用的符号分解、因子路径法等加速计算的方法并不适用；迭代法的计算具有重复固定、易于开发计算吞吐率的特点，适宜 FPGA 实现；在求解非对称正定矩阵的同类型迭代法中，BiCGStab 法较双共轭梯度法有更快、更稳定的收敛效果，较广义最小余差法使用更少的存储空间[14]，可作为求解器的基本求解算法。

本文设计的大规模 BBDF 稀疏矩阵求解硬件加速器，利用电力系统矩阵的分块特征将计算缩小到分块程度，通过数据预读取，降低存储访问的负担和计算过程中的内存占用；通过对 BiCGStab 算法的优化设计，节约计算时间、缩减计算资源，加速电力系统方程的求解。

1 电力系统方程的求解方法

1.1 电力系统方程介绍

电力系统的全过程动态方程可用 2 组方程式来描述：

$$\begin{cases} \boldsymbol{Y}' = F(\boldsymbol{Y},\boldsymbol{X},t) \\ G(\boldsymbol{Y},\boldsymbol{X},t) = \boldsymbol{0} \end{cases} \tag{1}$$

其中：$\boldsymbol{Y}$ 为描述系统动态元件的状态变量向量；$\boldsymbol{X}$ 为描述系统状态变量之间相互联系的代数变量向量；t 为动态响应所经历的时间。

对于刚性微分方程的求解，需采用牛顿迭代方法进行校正计算，将微分方程和代数方程联立求解的牛顿迭代公式[1]为：

$$\begin{bmatrix} \boldsymbol{Y}_{k+1}^{i+1} \\ \boldsymbol{X}_{k+1}^{i+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{Y}_{k+1}^{i} \\ \boldsymbol{X}_{k+1}^{i} \end{bmatrix} + \boldsymbol{M}^{-1} \begin{bmatrix} \Delta F_{k+1}^{i}(\boldsymbol{Y},\boldsymbol{X},t) \\ G_{k+1}^{i}(\boldsymbol{Y},\boldsymbol{X},t) \end{bmatrix} \tag{2}$$

雅可比矩阵 $\boldsymbol{M}$ 的结构整体上归属于 BBDF 矩阵，可分为 $\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$、$\boldsymbol{D}$ 四大分块，如图 1 所示。

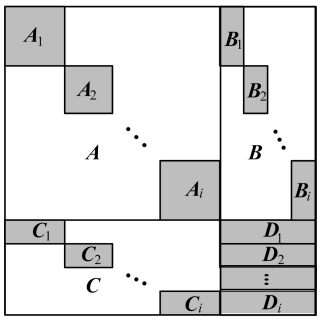

图 1　电力系统仿真方程的系数矩阵结构

图 1 中:$\boldsymbol{A}$ 块阶数是另一大分块 $\boldsymbol{D}$ 的 3～5 倍,包含一系列大小不一致的对角分块 $\boldsymbol{A}_i$(i=1,2,3,…);$\boldsymbol{B}$、$\boldsymbol{C}$ 的子块 $\boldsymbol{B}_i$(i=1,2,3,…)、$\boldsymbol{C}_i$(i=1,2,3,…)与 $\boldsymbol{A}_i$ 相对应,每个 $\boldsymbol{B}$ 块一般只有 2 到 4 列,每个 $\boldsymbol{C}$ 块一般只有 2 到 4 行,$\boldsymbol{C}$ 块内部的行和 $\boldsymbol{B}$ 块内部的列相互独立;$\boldsymbol{D}$ 块的子块 $\boldsymbol{D}_i$(i=1,2,3,…)与 $\boldsymbol{C}_i$ 的行相对应[13]。

1.2 BiCGStab 算法

由电力系统全过程仿真方程的系数矩阵 $\boldsymbol{M}$、解向量 $\boldsymbol{x}$、右端向量 $\boldsymbol{b}$ 组成的线性方程为:

$$\boldsymbol{Mx} = \boldsymbol{b} \tag{3}$$

如上文所述,采用 BiCGStab 算法求解电力系统方程(3),算法流程[15]如下。

输入:稀疏矩阵 $\boldsymbol{M}$,右端向量数组 $\boldsymbol{b}$,预期误差精度 P_{eps}

输出:方程的解向量数组 $\boldsymbol{x}$,结果误差 ε

$\boldsymbol{x}$←ones(N,1)

$\boldsymbol{v}_i$←zeros(N,1) ,$\boldsymbol{p}_i$←zeros(N,1)

ρ_1←1,α←1,ω←1

$\boldsymbol{r}_0$←$\boldsymbol{b}-\boldsymbol{Mx}$,$\boldsymbol{r}$←$\boldsymbol{r}_0$

for P_{iter}

1 to $P_{\mathrm{max,iter}}$ by +1 do

ρ_2←$\boldsymbol{r}_0^{\mathrm{T}}\boldsymbol{r}$

β=$\rho_2\alpha/(\rho_1\omega)$

$\boldsymbol{p}$←$\boldsymbol{p}-\omega\boldsymbol{v}$,$\boldsymbol{p}$←$\boldsymbol{r}+\beta\boldsymbol{p}$

$\boldsymbol{v}$=$\boldsymbol{Mp}$

α=$\rho_2/(\boldsymbol{r}_0^{\mathrm{T}}\boldsymbol{v})$,$\rho_1$=$\rho_2$

$\boldsymbol{s}$←$\boldsymbol{r}-\alpha\boldsymbol{v}$,$\boldsymbol{x}$←$\boldsymbol{x}+\alpha\boldsymbol{p}$

$\boldsymbol{t}$=$\boldsymbol{Ms}$

ω=$(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{s})/(\boldsymbol{t}^{\mathrm{T}}\boldsymbol{t})$

$\boldsymbol{r}$←$\boldsymbol{s}-\omega\boldsymbol{v}$,$\boldsymbol{x}$←$\boldsymbol{x}+\omega\boldsymbol{s}$

ε=$\|\boldsymbol{r}\|$

if $\varepsilon<P_{\mathrm{eps}}$ then

return ($\boldsymbol{x}$,ε)

break

end for

2 求解器的设计方案

2.1 系数矩阵的嵌套行压缩存储格式

存储格式与计算相辅相成,很大程度上影响着计算的吞吐率,求解器以分块尺度执行 BiCGStab 算法,要求存储格式能够适应分块数据包的访问需求。

基于传统的行压缩(compressed sparse row,CSR)格式提出本设计中系数矩阵所使用的嵌套行压缩存储格式,如图 2 所示。

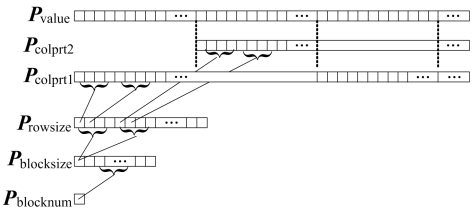

图 2 嵌套行压缩存储格式

该存储格式由 4 个数组和 1 个变量组成,具体描述如下。

1) 数组 $\boldsymbol{P}_{\mathrm{value}}$:按 $\boldsymbol{A}_1\boldsymbol{B}_1$,$\boldsymbol{C}_1\boldsymbol{D}_1$,…,$\boldsymbol{A}_i\boldsymbol{B}_i$,$\boldsymbol{C}_i\boldsymbol{D}_i$ 的块顺序依次按行存储每个子块的非 0 元素值。

2) 数组 $\boldsymbol{P}_{\mathrm{colprt1}}$ 和 $\boldsymbol{P}_{\mathrm{colprt2}}$:记录 $\boldsymbol{P}_{\mathrm{value}}$ 中每个非 0 元素的列号,其中 $\boldsymbol{A}_i\boldsymbol{B}_i$($i$=1,2,3,…)块的列号记录在 $\boldsymbol{P}_{\mathrm{colprt1}}$,$\boldsymbol{C}_i\boldsymbol{D}_i$ 块的列号记录在 $\boldsymbol{P}_{\mathrm{colprt2}}$。

3) 数组 $\boldsymbol{P}_{\mathrm{rowsize}}$:按 $\boldsymbol{A}_1\boldsymbol{B}_1$,$\boldsymbol{C}_1\boldsymbol{D}_1$,…,$\boldsymbol{A}_i\boldsymbol{B}_i$,$\boldsymbol{C}_i\boldsymbol{D}_i$ 的块顺序依次记录子块每行的非 0 元素个数,用于预读取时产生 $\boldsymbol{P}_{\mathrm{value}}$ 的索引,并在读取矩阵时标识换行位置。

4) 数组 $\boldsymbol{P}_{\mathrm{blocksize}}$:由 $\boldsymbol{A}_i$ 子块的行数 $\boldsymbol{P}_{\mathrm{blocksizeA}}$ 和 $\boldsymbol{C}_i$ 子块的行数 $\boldsymbol{P}_{\mathrm{blocksizeC}}$ 拼接而成,用于产生 $\boldsymbol{P}_{\mathrm{rowsize}}$ 的索引,并产生 $\boldsymbol{A}_i\boldsymbol{B}_i$、$\boldsymbol{C}_i\boldsymbol{D}_i$ 块切换的标识,以区分针对 $\boldsymbol{A}_i\boldsymbol{B}_i$ 和 $\boldsymbol{C}_i\boldsymbol{D}_i$ 块的不同操作。

5) 变量 $\boldsymbol{P}_{\mathrm{blocknum}}$:记录矩阵的 $\boldsymbol{A}_i$、$\boldsymbol{B}_i$、$\boldsymbol{C}_i$、$\boldsymbol{D}_i$ 子块的总个数,即 i 的最大值,用于产生 $\boldsymbol{P}_{\mathrm{blocksize}}$ 的索引。

嵌套行压缩存储格式中,按照分块尺度存储行非 0 元素个数,分别存储位宽差异较大的 $\boldsymbol{P}_{\mathrm{colprt1}}$、$\boldsymbol{P}_{\mathrm{colprt2}}$,相较于传统行压缩格式占用更少的存储,有助于提高片外存储芯片数据总线的有效带宽;系数矩阵 $\boldsymbol{M}$ 主要用于 BiCGStab 算法的矩阵向量乘操作,在分块计算中需按矩阵子块划分数据包,嵌套压缩存储的各个数组间的索引关系具有一致性,解压缩、数据分包过程规整有助于流水实现。

基于上述存储格式,计算过程中分块矩阵数据包的解压缩读取流程如下。

1) 顺序取出一个 $\boldsymbol{P}_{\mathrm{blocksize}}$ 数组的数据,取高 5 位记录为 m_i,取低 3 位记录为 n_i,将 m_i 和 n_i 相加作为将要读取的 $\boldsymbol{P}_{\mathrm{rowsize}}$ 数组的个数并记为 P_{cnt1}。

2) 顺序取出一个 $\boldsymbol{P}_{\mathrm{rowsize}}$ 数组的数据,作为将要读取的 $\boldsymbol{P}_{\mathrm{value}}$ 数组的个数并记为 P_{cnt2}。

3) 顺序取出 P_{cnt2} 个 $\boldsymbol{P}_{\mathrm{value}}$ 数组的数据作为分块矩阵的非 0 元素值;顺序取出 m_i 个 $\boldsymbol{P}_{\mathrm{colprt1}}$ 数组的数据和 n_i 个 $\boldsymbol{P}_{\mathrm{colprt2}}$ 数组的数据,作为分块矩阵

的非 0 元素列号。

4）重复步骤 2）、步骤 3）过程至当前子块的元素全部取出，并将这些数据划分为一个分块矩阵数据包。

相应地，分块向量数据包的读取流程如下。

1）同分块矩阵数据包读取流程中步骤 1）。

2）通过顺序递增或分区间递增的方式，顺序取出 P_{cnt1} 个指定向量的元素做为一个分块向量数据包。

2.2　任务优化划分

BiCGStab 算法的执行过程中，向量加减、向量数乘、矩阵向量乘可以实现分块尺度的流水；但向量积需要在完成分块向量积计算后对所有分块向量积结果进行累加，向量积前、后的计算因数据的相关性而阻塞。根据发生阻塞的位置对 BiCGStab 算法进行任务划分，得到的 5 个计算任务具体描述见上文算法流程。

统计每个计算任务中的计算类型，可以观察到部分计算任务的计算类型在时间与空间上没有交叉，可以将相应的计算模块进行合并。由于计算模块之间是流水的，这样的合并方式隐藏了被合并的计算任务的计算时间，且计算资源消耗没有增加。优化的 BiCGStab 算法的任务流图如图 3所示。

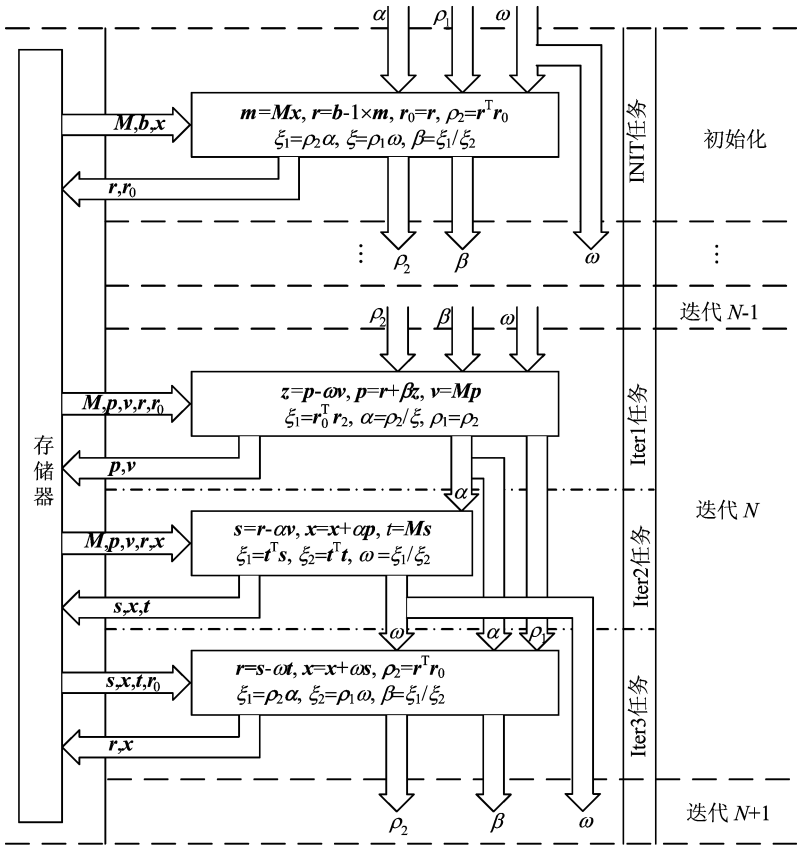

图 3　优化的 BiCGStab 算法的任务流图

基于上文所述，分块矩阵数据包、分块向量数据包的读取过程和算法任务优化划分方式的分块 BiCGStab 算法流程如下。

1）进入 INIT 任务。顺序读取一组分块矩阵 $\boldsymbol{M}_i$（$i=1,2,\cdots$）和分块向量 $\boldsymbol{x}_i$、$\boldsymbol{b}_i$，完成如图 3 所示 INIT 任务的组合计算；重复这一过程至分块计算全部完成后进入步骤 2）。

2）进入 Iter1 任务。顺序读取一组分块矩阵 $\boldsymbol{M}_i$ 和分块向量 $\boldsymbol{p}_i$、$\boldsymbol{v}_i$、$\boldsymbol{r}_i$、$\boldsymbol{r}_{0i}$，完成如图 3 所示 Iter1 任务的组合计算；重复这一过程至分块计算全部完成后进入步骤 3）。

3）进入 Iter2 任务。顺序读取一组分块矩阵 $\boldsymbol{M}_i$ 和分块向量 $\boldsymbol{p}_i$、$\boldsymbol{v}_i$、$\boldsymbol{r}_i$、$\boldsymbol{x}_i$，完成如图 3 所示 Iter2 任务的组合计算；重复这一过程至分块计算全部完成后进入步骤 4）。

4）进入 Iter3 任务。顺序读取一组分块向量 $\boldsymbol{s}_i$、$\boldsymbol{t}_i$、$\boldsymbol{x}_i$、$\boldsymbol{r}_{0i}$，完成如图 3 所示 Iter3 任务的组合计算，并将第 1 组向量加减、向量数乘的计算结果导出计算模平方和用于终止判断；重复这一过程至分块计算全部完成后，进行终止判断，若满足精度

则退出任务循环,否则重复步骤 2)~步骤 4)进行下一次迭代计算。

2.3 计算结构的优化设计

分块 BiCGStab 算法的 4 个计算任务复用同一组计算模块,通过动态配置的可重构技术实现不同任务的组合计算。

参考图 3,分析单个计算任务中计算模块的工作流程。以 Iter2 任务为例,由于计算第 i 组的 $\boldsymbol{t}_i=\boldsymbol{M}_i\boldsymbol{s}_i$ 需要第 i 组的 $\boldsymbol{s}_i=\boldsymbol{r}_i-\alpha\boldsymbol{v}_i$ 的计算结果,为避免计算结果 $\boldsymbol{s}_{i+1}$ 被覆盖,第 $i+2$ 组 $\boldsymbol{s}_{i+2}=\boldsymbol{r}_{i+2}-\alpha\boldsymbol{v}_{i+2}$ 的计算需要在第 i 组矩阵向量乘计算结束,且 $\boldsymbol{s}_{i+1}$ 开始传递给矩阵向量乘计算模块时开始;同时 $(\boldsymbol{t}_i^{\mathrm{T}}\boldsymbol{s}_i)$ 与 $(\boldsymbol{t}_i^{\mathrm{T}}\boldsymbol{t}_i)$ 依赖矩阵向量乘计算的输出,也会被矩阵向量乘计算阻塞。

流水结构下各类计算的耗时与参与计算的数据量有关,分块的矩阵向量乘与其他分块的向量计算间有较大的数据量差异和计算时间差异,因此向量计算在工作中存在长时间的空闲状态。将向量计算的相关模块复用,先后完成同类型计算,形成多层复用的折叠式结构,以节省更多的计算资源,如图 4 所示。其中,op 表示加法器的模式选择信号。

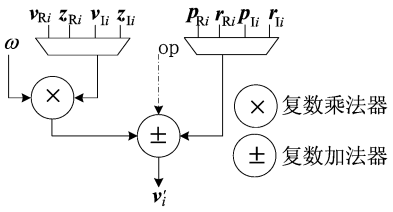

图 4 向量加减、向量数乘计算的折叠结构

折叠优化前、后上述复数计算所需要的实数计算器件的个数见表 1 所列。

表 1 优化前、后计算器件的使用情况

计算类型	使用数量		优化比例/%
	优化前	优化后	
乘法器	74	26	64.86
加法器	61	19	68.85
除法器	2	2	0

2.4 计算模块的块流水机制

计算模块的工作流程示意图如图 5 所示。

结合分块计算设计计算模块间的块流水阻塞机制:图 5a 所示为没有输出阻塞时计算模块的工作流程,t_1 时刻数据包 1 开始输入,相关计算开始执行,t_3 时刻计算完成,数据包 1 开始输出,同时数据包 2 开始输入并执行相关计算;图 5b 所示为有输出阻塞时计算模块的工作流程,t_3 时刻输出阻塞导致计算结果数据包无法输出,此时输入被同步阻塞,至 t_4 时刻输出阻塞释放后输出计算结果,并释放输入阻塞接收数据包 2 执行相关计算。

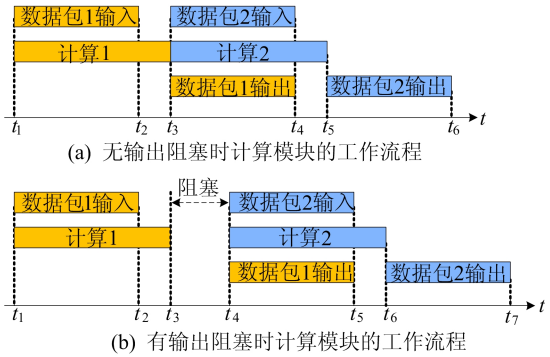

图 5 计算模块的工作流程示意图

这一机制中,后一级计算模块准备后进行一个完整的分块矩阵或向量数据包的传递。由于输入和计算没有中断或阻塞,避免了全流水机制下可能发生的前一级计算模块的输入和计算赶超输出,导致的输出数据被覆盖的错误,又保证了计算模块的高效工作。

3 求解器的设计实现

大规模 BBDF 稀疏矩阵求解器的整体架构如图 6 所示,由任务管理器、存储模块、数据读取接口、数据写回接口、主运算器以及用于 PCI-E 传输的 AXI 总线构成。

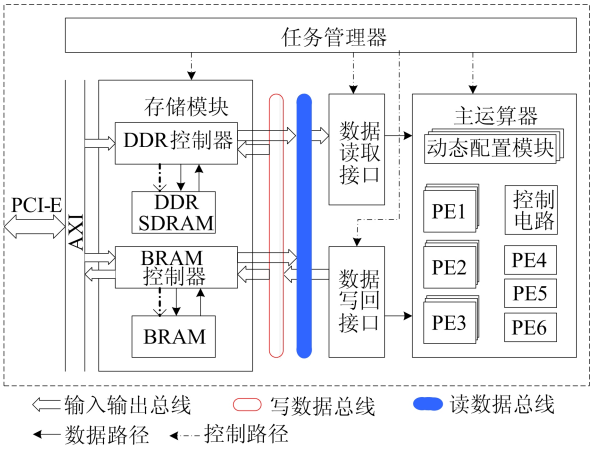

图 6 求解器的整体架构

求解器的工作流程分为 3 个阶段:

1) 数据准备阶段。由上位机下发原始数据至求解器的存储模块,复位解向量和其他过程向量。

2）计算阶段。采用上文所述的任务优化划分和多层级复用的折叠式计算结构，由任务管理器控制存储模块、数据接口模块和主运算器协调工作，实现分块 BiCGStab 算法。

3）数据回传阶段。在状态机结束循环跳转后将解向量、残差向量等数据回传至上位机。

任务管理器作用于求解器工作流程中的计算阶段。内置状态机根据优化的任务划分方式将计算划分为相应的 4 个状态，输出状态控制信号用于其他模块的控制；Iter3 状态下进行算法的终止判断，当满足终止条件后跳出状态循环。

存储模块中设置了 DDR SDRAM 和 Block RAM(BRAM)2 类存储资源，用于存储系数矩阵 $\boldsymbol{M}$、右端向量 $\boldsymbol{b}$、解向量 $\boldsymbol{x}$ 及其他过程向量，并将这些数据的存储端口进行封装，配合数据接口模块进行读写操作。

数据读取接口和数据写回接口作为存储模块与主运算器之间的桥梁，实现主运算器与存储模块之间的通信。求解器内的数据众多，且存在一定的访问规律，根据多种数据的读写时序设置读控制单元、写控制单元，对数据进行批量有序的预读取和写回，并通过内置交叉开关将数据配置到计算模块的指定通道。此外还设置了单一地址访问路径配合计算过程中的单一数据访问控制。

主运算器是稀疏矩阵求解器的计算核心，计算模块和动态数据路径配置模块(data path dynamic configuration，DPDC)相互穿插，形成多层复用的并行折叠式结构，如图 7 所示。主运算器中包含向量加减向量数乘、矩阵向量乘、向量积、四循环累加器 4 个计算模块和复数乘、复数除 2 个计算器件。由任务管理器的状态输出信号配置计算模块之间的数据路径，利用可重构原理，在不同任务状态下将计算模块重新组合，实现不同任务的组合计算。

Iter1 任务状态下单路计算模块之间的数据路径如图 8 所示。

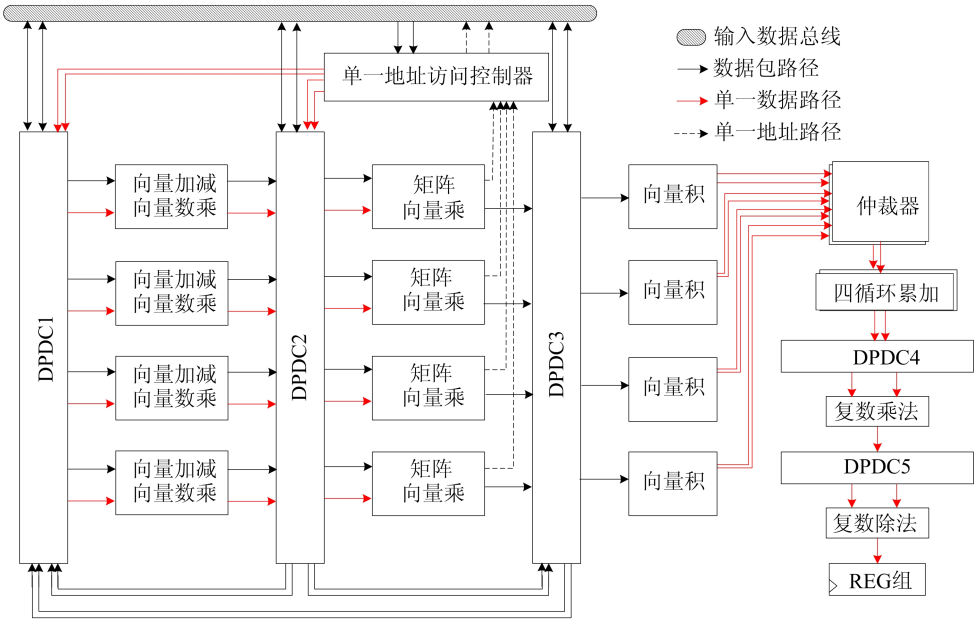

图 7　主运算器的实现结构

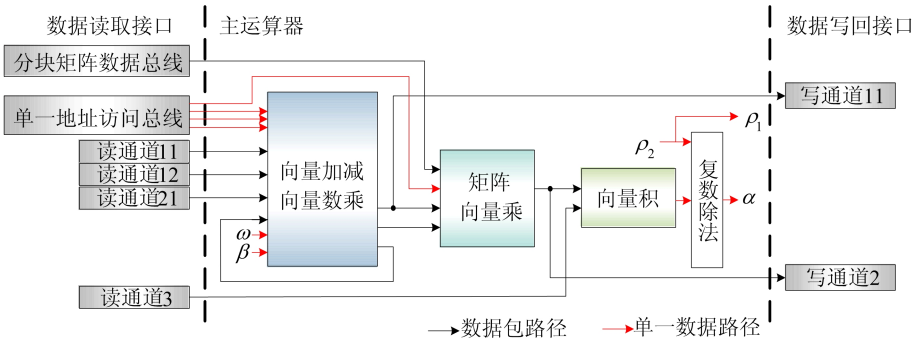

图 8　Iter1 任务状态计算模块的数据路径

矩阵向量乘计算中小部分参与计算的向量元素在分块向量中未命中,通过矩阵元素列号获取该元素的索引,经单一地址访问直接或间接获得该元素。图 7 中单一地址访问控制器用于解决 4 路矩阵向量乘模块的多路单一地址访问冲突,根据 4 路单一地址输入端口的优先级控制各路地址线的信号传递。

4 功能验证与性能分析

本节针对 BBDF 稀疏矩阵求解器进行功能验证和性能测试。设计实现的求解器工作主频为 100 MHz,FPGA 芯片型号为 Xilinx XC7VX690T-2FFG,资源消耗情况见表 2 所列。

表 2 硬件资源使用情况

资源类型	使用量	可用量	使用占比/%
LUT	57 843	433 200	13
LUTRAM	3 651	174 200	2
FF	68 314	866 400	8
BRAM	1 145	1 470	78
DSP	132	3 600	4
PLL	1	20	5

结合批量求解任务的求解速度、结果误差的软硬件对比分析,测试求解器功能的完备性、可靠性和高效性。

实验所使用的算例均借助 MATLAB 的随机函数,按照电力系统矩阵的 BBDF 特征构造不同阶数、不同稀疏度的多种算例进行实验,迭代求解的目标精度为 10^{-6}。实验所使用的软件载体主要参数为 Intel(R) Core(TM) i9-10900K CPU @ 3.70 GHz,96.0 GiB 内存,采用 MATLAB 作为编程工具产生标准计算结果和软件执行时间,并作为硬件性能分析的参考。

求解器采用双精度浮点数据格式计算,当残差向量的模平方和达到变换后的目标精度时终止循环迭代,这使得迭代次数较软件略有浮动,但避免了开方带来的线性误差、资源消耗以及计算时间消耗,且迭代次数的正向浮动使得解向量具有更高的准确性。

取 2 000 组算例的解向量,在 MATLAB 中计算残差向量的标准差之和作为结果误差,计算求解器的结果误差相对于理论值的偏移量并绘制散点图,如图 9 所示,其中负偏移量代表解向量具有更高的精度。

实验 12 组,每组 10 个算例的软、硬件实验的平均耗时及硬件求解加速比见表 3 所列。

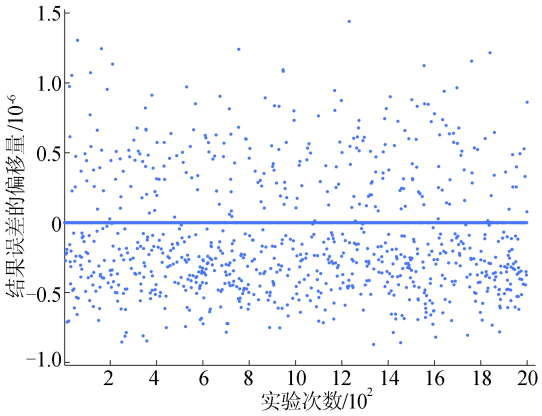

图 9 硬件求解的结果误差分布

表 3 算例的软、硬件求解耗时及硬件求解加速比

实验组别	软件实验耗时/s	硬件实验耗时/s	加速比/倍
1	5.261 8	0.170 0	30.937 4
2	7.259 1	0.239 3	30.505 3
3	27.087 7	0.882 1	31.473 4
4	41.116 0	1.338 3	30.115 1
5	1.055 0	0.033 5	31.424 3
6	1.684 1	0.053 3	31.482 2
7	2.261 6	0.072 1	30.973 7
8	5.359 5	0.166 1	32.235 0
9	4.074 9	0.133 6	30.793 4
10	6.381 6	0.208 7	30.497 7
11	12.045 8	0.384 8	31.304 0
12	38.660 5	1.253 5	30.948 4

将本文方法(求解器)的加速效果与文献[6]、文献[7]、文献[9]的进行对比,见表 4 所列。

表 4 4 种方法加速效果对比

方法	平均矩阵阶数	加速比/倍
文献[6]	760 350	11.884 1
文献[7]	3 755	2.698 2
文献[9]	21 464	5.900 0
本文	47 498.8	30.821 8

表 4 中:文献[6]采用不完全 LU 分解为预处理的 BiCGStab 算法,利用 GPU 并行处理线性方程组求解,迭代法精度取 10^{-6},与求解库 SuperLU 传统方法对比计算加速比;文献[7]采用 CG 迭代法并利用 FPGA,未结合电力系统矩阵的 BBDF 特征进行的通用稀疏矩阵求解器设计;本文用 FPGA 进行定制化电路设计,收获了较其他方法更高的加速比;文献[9]采用直接求解法,利用 GPU 通过静态调度算法构建静态指令流,利用 FPGA(XCVU37P 芯片)加速三角分解、前代、

回代过程。

本文方法与文献[9]方法的硬件资源使用情况对比见表 5 所列。

表 5　硬件资源使用情况对比

资源类型	文献[9]方法	本文方法
LUT	336 646	57 843
LUTRAM	7 109	3 651
FF	186 534	68 314
BRAM	1 074	1 145
URAM	640	0
DSP	2 560	132

针对系数矩阵具有相似特征的稀疏矩阵求解问题，本文设计使用更少的资源实现了更高加速比。

综上所述，求解器的加速效果主要来源于：

1）大规模稀疏线性方程求解的计算重复固定，适合 FPGA 开发。

2）BiCGStab 算法经过分块处理具有更规整、高密度的计算特点，易于应用多种加速设计方法，开发计算吞吐率。

3）数据接口的预读取、写回操作使得存储访问变得规整，缩减了存储访问的时间。

4）任务优化划分和折叠式计算结构隐藏了部分计算时间；计算模块内的流水设计、计算模块间的块流水设计保证主体计算具有较高的计算效率；并行化设计使得加速效果更加显著。

5　结　　论

本文针对电力系统仿真方程的特点，采用嵌套行压缩存储策略，多层级复用的折叠式计算结构，实现了具有一定通用性的大规模 BBDF 稀疏矩阵求解器。该求解器通过任务管理、存储管理、数据管理、计算核心 4 个部分的协同工作实现系数矩阵具有 BBDF 特征的大规模稀疏线性方程组的快速求解。实验结果表明，在批量求解各种算例时，本文求解器相对于通用处理器的平均加速比为 30～32 倍，满足精度要求，具备功能完备性、可靠性和高效性。

[参　考　文　献]

[1] 汤涌. 电力系统全过程动态（机电暂态与中长期动态过程）仿真技术与软件研究[D]. 北京：中国电力科学研究院，2002.

[2] OMAR FARUQUE M D, STRASSER T, LAUSS G, et al. Real-time simulation technologies for power systems design, testing, and analysis[J]. IEEE Power and Energy Technology Systems Journal, 2015, 2(2): 63-73.

[3] SHU J, XUE W, ZHENG W. A parallel transient stability simulation for power systems[J]. IEEE Transactions on Power Systems, 2005, 20(4): 1709-1717.

[4] 洪潮. 电力系统暂态稳定计算的一种时间并行算法[J]. 电网技术，2003(4)：31-35.

[5] 江涵，江全元. 基于 GPU 计算平台的大规模电力系统暂态稳定计算[J]. 电力系统保护与控制，2013，41(4)：13-20.

[6] 唐坤杰，董树锋，宋永华. 基于不完全 LU 分解预处理迭代法的电力系统潮流算法[J]. 中国电机工程学报，2017，37(增刊 1)：55-62.

[7] 宋庆增，顾军华. 共轭梯度求解器的 FPGA 设计与实现[J]. 计算机应用，2011，31(9)：2571-2573，2588.

[8] 吴志勇，王晞阳，陈继林. 一种基于 FPGA 并行加速的稀疏矩阵求解方法[J]. 电力系统保护与控制，2021，49(11)：155-162.

[9] 王晞阳，陈继林，李猛，等. FPGA 架构上面向稀疏矩阵求解的静态调度算法[J]. 计算机工程，2022，48(7)：199-205，213.

[10] 徐晋，汪可友，李国杰. 电力电子设备及含电力电子设备电力系统实时仿真研究综述[J]. 电力系统自动化，2022，46(10)：3-17.

[11] CHANIOTIS D, PAI M A. Iterative solver techniques in the dynamic simulation of power systems[C]//2000 Power Engineering Society Summer Meeting. [S. l.]: IEEE, 2000: 609-613.

[12] HUANG S, DINAVA HI V. Performance analysis of GPU-accelerated fast decoupled power flow using direct linear solver[C]//Electrical Power and Energy Conference. [S. l.]: IEEE, 2017: 1-6.

[13] 宋新立. 电力系统全过程动态仿真算法与模型研究[D]. 天津：天津大学，2014.

[14] 邓亮章. 大型稀疏线性方程组的 Krylov 子空间方法[D]. 厦门：厦门大学，2009.

[15] VORST V D H A. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems[J]. SIAM Journal on Scientific and Statistical Computing, 1992, 13(2): 631-644.

（责任编辑　胡亚敏）