文章编号:1001 - 9081(2011)09 - 2571 - 03

doi:10.3724/SP.J.1087.2011.02571

共轭梯度求解器的 FPGA 设计与实现

宋庆增[1], 顾军华[2]

(1. 河北工业大学 电气工程学院,天津 300401; 2. 河北工业大学 计算机科学与软件学院, 天津 300401)

(qingzengsong@163.com)

摘 要:针对共轭梯度(CG)迭代算法软件执行效率低、实时性差的缺点,提出一种基于现场可编程逻辑门阵列(FPGA)平台的 CG 迭代求解器。设计采用软硬件结合的方式构建整个系统,CG 协处理器执行 CG 迭代算法中计算量大、控制简单的代码,以达到硬件加速的目的。控制复杂、计算量较少的代码则依旧在微处理上执行。设计采用行交错数据流,使得整个系统完全无停顿的运行,提高了计算性能。实验结果表明,与软件执行相比,硬件 CG 协处理器可以获得最高 5.7 倍的性能加速。

关键词:可重构计算;稀疏线性方程组;现场可编程逻辑门阵列;共轭梯度法;行交错数据流

中图分类号: TP31 **文献标志码**:A

Design and implementation of conjugate gradient iterative solver on FPGA

SONG Qing-zeng[1], GU Jun-hua[2]

(1. *School of Electrical Engineering and Automation, Hebei University of Technology, Tianjin* 300401, *China*;
2. *School of Computer Science and Engineering, Hebei University of Technology, Tianjin* 300401, *China*)

Abstract: To overcome the disadvantage of inefficient and bad real-time capability in software version Conjugate Gradient (CG) iterative solver, a CG iterative solver was designed and implemented on Field Programmable Gate Array (FPGA) platform. The design of CG iterative solver was based on hardware/software co-design. Hardware CG co-processor implemented the code of enormous computation and simple control, which could accelerate the system. The code of control complexity and less calculation was still performed in the microprocessor. The use of row-interleaved data flow could make the system not to stall to improve performance. The experimental results illustrate that hardware CG iterative solver can speed up about 5.7 times over the software version of the same algorithm.

Key words: reconfigurable computing; sparse linear equations; Field Programmable Gate Array (FPGA); conjugate gradient algorithm; row-interleaved data flow

0 引言

共轭梯度(Conjugate Gradient, CG)法是求解稀疏线性方程组的有效数值迭代算法,具有较快的收敛速度和二次终止性等优点。石油勘探、大气模拟、电磁场数值计算等领域导出的大规模线性方程组常利用共轭梯度法求解。目前,共轭梯度求解器的实现主要包括两类,即通用处理器(General-Purpose Processor, GPP)和专用集成电路(Application-Specific Integrated Circuit, ASIC)。GPP 方法灵活性强,然而其广泛的灵活性是以牺牲系统的性能和速度为代价换来的,对共轭梯度算法的性能低下,尤其是求解大规模稀疏线性系统。ASIC 法用特定的集成电路,以完全硬件的方式来实现计算任务。尽管 ASIC 在尺寸(晶体管的数量)、复杂性和性能实现上都是最佳的,但是设计和制造它是一个非常耗时和代价昂贵的过程。

近年来,可重构器件的迅速发展极大地弥补了专用 ASIC 的不足。基于现场可编程逻辑门阵列(Field Programmable Gate Array, FPGA)的可重构计算系统可获得接近于专用 ASIC 的计算能力,又可以在应用环境发生变化时改变配置实现新的功能,兼顾了通用 GPP 的可编程性和专用 ASIC 的高效性。对于 CG 迭代算法,Jacobi 迭代算法等计算密集型的数值算法,可重构计算系统具有巨大的速度和功耗优势[1-2]。

本文在基于 FPGA 的可重构计算平台上设计和实现了一种硬件 CG 求解器。设计采用软硬件结合的方法构建整个系统,将自定义的硬件电路作为 CG 协处理器同 Nios Ⅱ软核处理器耦合在一起。CG 协处理器完成计算稠密、控制简单的代码,以达到硬件加速的目的。实验结果表明,相对于运行在 CPU 上的软件实现,CG 协处理器可以得到一定的计算性能的加速。

1 相关工作

自然科学和工程中,很多问题都将归结为求解下面的线性方程组:

$$\boldsymbol{Ax} = \boldsymbol{b} \tag{1}$$

其中:$\boldsymbol{A}$ 是 $n \times n$ 维的稀疏矩阵,$\boldsymbol{x}$ 是 n 维未知量,$\boldsymbol{b}$ 是 n 维右端项。稀疏线性方程组求解的迭代算法中,CG 迭代法是最常用的一种方法。该方法具有算法简单、存储需求小等优点,十分适合于大规模的问题,在诸多领域得到了广泛的应用。CG 迭代算法的搜索方向是互相共轭的,因此具有非常快的收敛速度[4-5]。图 1 给出了 CG 迭代的算法流程。CG 主函数完成初始化,控制迭代次数等操作,主迭代子程序则完成主要的计算任务。

收稿日期:2011 - 03 - 21;**修回日期**:2011 - 06 - 03。 **基金项目**:科技人员服务企业行动项目(2009GJA20014)。

作者简介:宋庆增(1980 -),男,河北保定人,博士研究生,主要研究方向:智能信息处理、电磁场数值解、可重构计算; 顾军华(1966 -),男,河北赵县人,教授,博士,博士生导师,主要研究方向:智能信息处理、电磁场数值解。

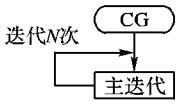

图 1 CG 迭代算法

主迭代程序主要由下面的计算任务组成：

$$\boldsymbol{q}^k = \boldsymbol{A}\boldsymbol{p}^k$$
$$t_k = (\boldsymbol{r}^k,\boldsymbol{r}^k)/(\boldsymbol{q}^k,\boldsymbol{p}^k)$$
$$\boldsymbol{r}^{k+1} = \boldsymbol{r}^k - t^k\boldsymbol{q}^k$$
$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + t^k\boldsymbol{p}^k$$
$$\boldsymbol{\beta}^k = (\boldsymbol{r}^{k+1},\boldsymbol{r}^{k+1})/(\boldsymbol{r}^k,\boldsymbol{r}^k)$$
$$\boldsymbol{p}^{k+1} = \boldsymbol{r}^{k+1} + \boldsymbol{\beta}^k\boldsymbol{p}^k$$

其中：$(\boldsymbol{M},\boldsymbol{N})$ 表示向量 $\boldsymbol{M}$ 和 $\boldsymbol{N}$ 的内积。CG 迭代算法同时具有计算稠密和控制复杂的特点，因此，将整个算法完全映射到 FPGA 芯片中并不是合理的处理方式。通常的做法是对 CG 迭代算法进行软硬件划分，将计算稠密、控制简单的部分以硬件电路的方式执行，以达到硬件加速的目的，算法的其余部分则依旧以软件的形式存在。

目前，基于 FPGA 的 CG 求解器的研究已经取得了一些研究成果。Morris 等人[1]在可重构计算机上设计和实现了稀疏线性方程组的 CG 迭代求解，采用基于 FPGA 和 GPP 混合设计的方法，将算法的计算稠密部分在 FPGA 硬件进行加速。Lopes 等人[3]为了求解稠密线性方程组，采用 CG 算法，与通常的处理方式不同，其在单个芯片上实现了整个 CG 算法，因此其扩展性很差，只能处理不超过 60 维的稠密线性方程组。

已有的算法的要么使用 CRS 格式[1]，要么是稠密矩阵[3]，很少针对偏微分方程导出的稀疏线性系统进行优化设计。偏微分方程通过有限元法和有限差分法导出的大规模稀疏线性方程组往往具有规则的结构，本文针对这类规则的稀疏线性方程组，对硬件 CG 求解器采用更加适合的稀疏矩阵存储格式和行交错的数据流，进行了优化设计。

2 系统架构设计

随着半导体技术的进步，FPGA 的运算速度和存储空间都得到了很大的提高，功能变得更加强大灵活，这使得在其上实现并行数值计算成为可能。然而，并非所有的算法都适合在 FPGA 实现。可重构硬件使用类似于 ASIC 的基于空间并行的执行方式，使其只适合执行程序中计算量大、拥有规则的数据访问模式、控制简单的那部分“计算密集型”代码；而算法中存在的一些计算量相对较少、访问模式比较随机、控制比较复杂的代码则不适合采用硬件执行。为执行这部分程序，则需要在可重构系统集成一个计算能力相对较弱的通用微处理器。

本文采用在 FPGA 内部生成 Nios Ⅱ软核的方法，构建可重构计算系统。Nios Ⅱ作为可重构系统的通用微处理器，执行不适合采用硬件执行的代码，系统的整体架构如图 2。

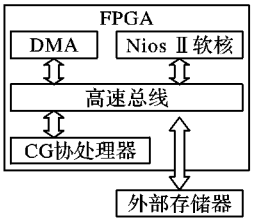

图 2 CG 系统架构

整个可重构计算系统由四个模块组成：DMA 模块、Nios Ⅱ模块、CG 协处理器和外部存储器模块，四个模块通过高速总线连接在一起。其中，Nios Ⅱ执行控制比较复杂、计算量较少的代码，主要的计算任务由 CG 协处理器的硬件电路完成。外部存储器存储计算需要的数据，DMA 模块则负责数据的交换。

根据程序代码自身的性质，对 CG 算法进行软硬件划分，确定使用 CG 协处理器执行的代码是设计的第一步，也是最为关键的一步。不合理的划分，将导致非常低的计算性能，不能达到性能加速的目的。将整个 CG 算法的主迭代映射到 FPGA 执行是可行的实现方式，但是这种方式只适合小规模的稀疏线性系统，可扩展性很差，一般不采用这种设计方式。

CG 算法的主迭代程序执行一次，需要执行一次稀疏矩阵乘向量（$\boldsymbol{q} = \boldsymbol{A}\boldsymbol{p}$）、二次归约操作（$(\boldsymbol{r},\boldsymbol{r})$ 和 $(\boldsymbol{q},\boldsymbol{p})$）、三次向量操作（$\boldsymbol{r} = \boldsymbol{r} - t\boldsymbol{q}, \boldsymbol{x} = \boldsymbol{x} + t\boldsymbol{p}, \boldsymbol{p} = \boldsymbol{r} + \boldsymbol{\beta}\boldsymbol{p}$），以及少量的标量操作。其中 95% 的计算时间用于稀疏矩阵乘向量操作，是整个迭代算法的主体。稀疏矩阵乘向量同时满足计算密集和操作简单的条件，适合以硬件电路的方式执行。因此，本文将稀疏向量乘矩阵映射到 CG 协处理器中以硬件电路的方式执行，其余部分依旧作为软件，在处理器上执行。

3 CG 协处理器设计

CG 协处理器的主要功能是高效地完成稀疏矩阵乘向量。稀疏矩阵乘向量不止是 CG 迭代设计的核心，更被广泛地用于各种数值方法。CG 协处理器由接口模块、计算模块和控制模块组成。接口模块由符合高速总线标准的接口电路组成，与高速总线相连负责数据的输入与输出；计算模块由一组计算单元（Process Element，PE）阵列和局部存储器组成；控制模块的主要功能是控制整个系统的正常运行。

计算模块是整个系统的重点，其他模块的设计都依附于计算模块的设计。计算模块所使用的浮点计算单元都是深度流水的，因此具有很高的运行频率。整个设计的数据通路以流水和并行化的方式进行组织，充分发挥 FPGA 的计算潜力。

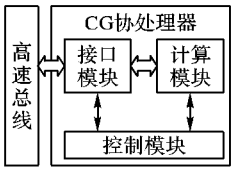

图 3 模块划分

3.1 稀疏矩阵存储格式

与稠密矩阵乘向量不同，稀疏矩阵的每一行的元素个数很少，这将导致在 FPGA 平台上实现稀疏矩阵乘向量的计算效率很大程度上取决于稀疏矩阵的存储策略[6]。而不合理的存储格式使得系统无法高度并行和流水的运行，最终导致性能的下降。有限元或有限差分法得到的稀疏矩阵一般具有很规则的结构，Ellpack-Itpack 格式充分体现了这种矩阵的稀疏结构的特点，具有在 FPGA 上执行计算效率高的优点。

该存储格式假设稀疏矩阵的每行至多有 N_d 个非零元素，需要两个大小为 $n \cdot N_d$ 的实浮点数矩阵和整型矩阵存储。COEF 矩阵用来记录稀疏矩阵中的非零元素，每一行的非零元对应存储在 $\mathrm{COEF}(1:n,1:N_d)$ 中，JCOEF 矩阵用来存储 COEF 中每个元素对应的列号。

3.2 计算模块

实现并行稀疏矩阵乘向量的方法有很多种，其中，内积方

法是最为重要的并行矩阵乘向量实现方法,这种方法将整个计算分解为 n 次的向量内积:

$$y_i = \sum_j A_{ij} * x_j \tag{2}$$

如图 4,设计采用完全的乘加二叉树组成计算阵列。假设稀疏矩阵的每一行的元素数目为 m ,矩阵的维数 n。内积法将需要 m 个浮点乘法器,$m-1$ 个浮点加法器和至少 $n \times m$ 局部存储单元。向量 $\boldsymbol{x}$ 存储在局部存储器中,由于一个局部存储器在一个时钟周期只能提供一个 $\boldsymbol{x}$ 的值,因此向量 $\boldsymbol{x}$ 需要 m 个备份。

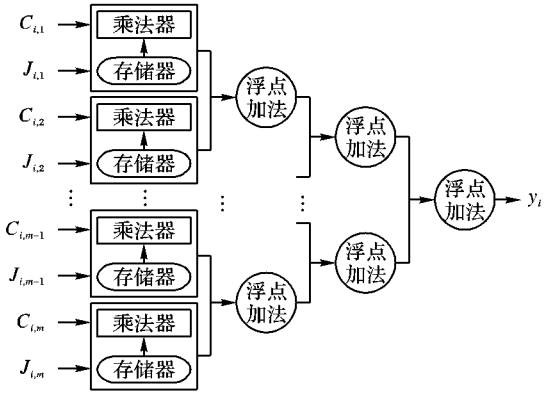

图 4　并行稀疏矩阵乘向量

每个时钟周期,都有 m 个 COEF 和 JCOEF 数值进入计算阵列。JCOEF 整数值作为向量 $\boldsymbol{x}$ 的地址值,在计算阵列的存储器中得到一个 $\boldsymbol{x}$ 值提供给浮点乘法器,COEF 的数值作为另一个乘数提供给浮点乘法器。由于采用完全并行和流水的数据流设计,完成整个计算所需的时钟周期数为 $n + \alpha_a \lceil \mathrm{lb}\, m \rceil + \alpha_m$,这里 α_a 为浮点加法的延迟,α_m 为浮点乘法的延迟,$\lceil \mathrm{lb}\, m \rceil$ 表示向上取整。

受硬件规模的限制,FPGA 上局部存储器的规模 w,和能实现的浮点加法和乘法的数目 s 也是有限的,一般有这样的关系:$\min(s, w/n) \leqslant m \leqslant n$。因此,完全并行的内积设计只适合小规模的稀疏矩阵,对中等规模以上的稀疏矩阵乘向量,无法在 FPGA 上实现完全并行的设计。

如图 5,本文将完全并行的操作分解为并行模块和串行模块两部分,其中并行模块采用与完全并行操作相同的乘加二叉树,只是规模缩小为 k,这里 $k \leqslant \min(s, w/n)$(本例中 $k = 4$)。每个时钟周期,并行的乘加二叉树将生成一个数据(部分的内积结果),串行模块将其累加,生成最终的计算结果。串行模块的设计是整个设计的难点之一,不合理的设计将导致整个系统的流水停顿。

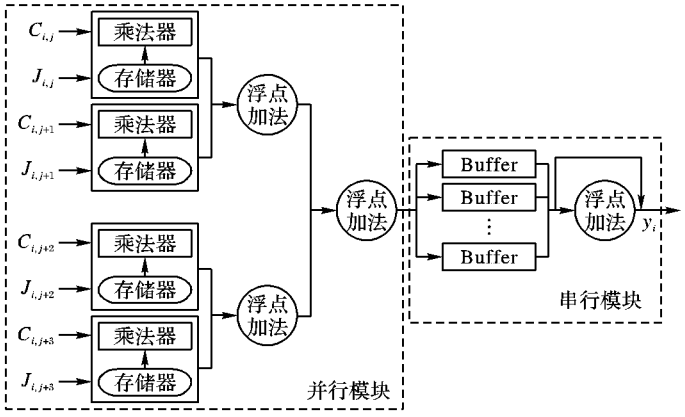

图 5　分解电路

2.3　行交错数据流

众所周知,浮点加法器都是深度流水的,显然简单的反馈加法设计将导致流水停顿,单集的归约电路法虽然能减少流水的停顿时间,但不能完全消除流水停顿。多集归约电路的设计能避免流水停顿,但是其设计复杂,资源消耗更多,同时复杂的设计将降低整个系统的运行频率。

为了避免任何的流水停顿和提高整个系统的运行频率,采用行交错数据流作为内部数据流。如图 5,采用一组 BUFFER 将原本严格的行优先输入的数据改成行交错数据。下面给出了一个实际的例子,其中交错参数 $\alpha_a = 3$,与浮点加法器的延迟相同。由于交错参数和加法延迟完全一致,交错数据流正好可以使浮点加法器完全无停顿的运行,提高整个系统的性能。

交错数据流:

原始顺序:1　2　3　4　5　6　7　8　9　10　11　12　13　14　15　16　17　18

交错顺序:1　7　13　2　8　14　3　9　15　4　10　16　5　11　17　6　12　18

完整的串行模块由一个浮点加法器,一组生成交错数据流的 BUFFER 和控制电路组成。BUFFERER 模块为浮点加法器生成内部行交错数据流,控制电路控制整个串行模块的运行。这里只是为了说明交错数据流的设计思想,没有给出完整的串行模块的设计。

4　分析与实验

Quartus II 9.0 和 Nios Ⅱ 9.0 作为编程工具,在 Altera 公司的 EP2C70 芯片上实现了完整的设计。对照组采用运行在 PC 机上的程序,其 CPU 采用 Intel Pentium D 2.8 GHZ(双核),内存为 512 MB。测试矩阵采用佛罗里达大学的稀疏矩阵集,表 1 给出了测试矩阵的性质,每个稀疏矩阵运行 1 000 次取其平均计算时间,时间单位为秒。与 CPU 上以软件方式执行相比,CG 协处理器计算速度要比 CPU 快 2 倍左右,最高可以得到 5.7 倍的性能加速。

表 1　性能加速比

矩阵名	矩阵维数	非零元数	FPGA 执行时间/s	CPU 执行时间/s	加速比
ash292	292	2 208	0.010	0.585	5.731
rdb8001	800	4 640	0.014	0.038	2.743
nasa2146	2 146	72 250	0.188	0.372	1.975
lnsp3937	3 937	25 407	0.104	0.202	1.952
s1rmq4m1	5 849	262 411	0.674	1.358	2.015
nemeth24	9 506	1 506 550	4.086	7.245	1.773

协处理器计算性能峰值 P_{peak} 的计算公式为 $P_{peak} \approx$ 浮点计算器的数目 × 运行频率。由于 CG 协处理器包含 4 个浮点乘法器和 4 个浮点加法器,运行频率为 114 MHz,因此 $P_{peak} \approx 8 \times 114 = 912$ MFLOPS $= 0.912$ GFLOPS。这里 GFLOPS 和 MFLOPS 都是浮点计算性能的单位,其中一个 GFLOPS 约等于每秒 10^9 次浮点运算。表 2 给出了 CG 协处理器运行测试矩阵时的实际计算性能,计算性能都在 0.4 GFLOPS 以上,最高能达到 0.77 GFLOPS。与软件执行不同,FPGA 上执行往往能达到计算峰值的 50% 以上,甚至 80%。因此当采用更先进芯片和集成更多计算单元时,可以得到更好的计算性能。

(下转第 2588 页)

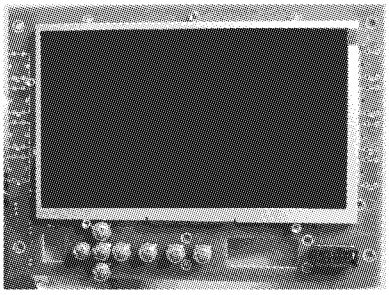

图 10 车载播放器硬件实物图

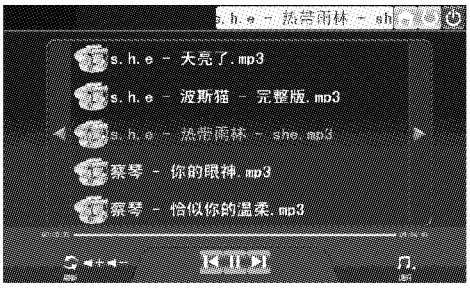

图 11 车载播放器音频播放截图

4 结语

Atom Z510 是 Intel 推出的系列嵌入式处理器之一，搭载有最新款 SCH 控制芯片，主要面向车载娱乐、工业控制与自动化、多媒体网络电话等嵌入式计算领域，提供 Windows/Linux 多操作系统支持。Atom Z510 具备低功耗的特性，支持高清，板载 1 GB 内存，搭载有 SCH US15WP 桥片，提供丰富的接口，使用户可根据需要方便的扩充自己需要的接口。Atom Z510 的这些特性，为利用它制作一款高性能的车载播放器提供了优越条件。本文的播放器建立在 Atom Z510 基础之上，具备图形播放界面，操作简单便捷；同时支持音频、视频播放，支持多种格式的音视频文件，支持所有主流视频编解码标准，支持高清，功能强大，具有很大的实用价值。

参考文献：

[1] 殷建红. 国内车载娱乐信息系统发展现状及趋势[J]. 汽车与配件, 2009(11): 40 - 42.

[2] MACARIO G, TORCHIANO M, VIOLANTE M. An in-vehicle infotainment software architecture based on google android [C]// SIES '09: IEEE International Symposium on Industrial Embedded Systems. Piscataway, NJ: IEEE, 2009: 257 - 260.

[3] Intel Corporation. Intel System Controller Hub [EB/OL]. [2010 - 01 - 26]. http://download. intel. com/design/chipsets/embedded/datashts/319537. pdf.

[4] 李建, 许勇, 苏平. 基于 Atom 的移动装置远程监控系统设计[J]. 计算机系统应用, 2010, 19(4): 5 - 8.

[5] 张江洪. 基于 AU6840 的车载音乐播放器系统设计[J]. 吉首大学学报: 自然科学版, 2009, 30(2): 63 - 65.

[6] 王建国, 马然. 基于 Au1200 处理器的车载多媒体电脑设计[J]. 计算机工程, 2009, 35(23): 243 - 245.

[7] 张海滨, 李挥, 吴晔, 等. 嵌入式高清播放器的设计与实现[J]. 计算机工程与设计, 2010, 31(13): 3084 - 3087.

[8] 李巍, 贾克斌. MPlayer 播放器在嵌入式平台上的应用[J]. 测控技术, 2007, 26(增刊): 286 - 289.

[9] MPlayer 工作组. MPlayer——电影播放器[CP/OL]. [2010 - 10 - 24]. http://www. mplayerhq. hu.

[10] 毕厚杰, 王健. 新一代视频压缩编码标准——H. 264/AVC[M]. 2 版. 北京: 人民邮电出版社, 2009.

[11] BLICKFORD B L, BROWN A K. Vehicle audio system having random access player with play list control: US, 6185163 B1 [P]. 2001 - 02 - 06.

[12] 梁天彧, 李娟, 门爱东. 现代视频编码关键技术及其发展[J]. 电力系统通信, 2006, 27(3): 1 - 4.

[13] BUTLER N, BATOUR J-N, TUCKER D, *et al*. In-vehicle multimedia system: US, 7571038 B2 [P]. 2009 - 08 - 04.

(上接第 2573 页)

表 2 CG 协处理器的计算性能

矩阵名	计算性能/GFLOPS	实际性能占峰值的百分比/%
ash292	0.4310	47.26
rdb8001	0.6612	72.50
nasa2146	0.7676	84.17
lnsp3937	0.4905	53.78
s1rmq4m1	0.7786	85.37
nemeth24	0.7374	80.86

4 结语

本文针对 CG 迭代算法的特点，在基于 FPGA 的可重构硬件平台上设计和实现了优化的硬件 CG 迭代求解器。有效的软硬件划分技术将 CG 代码被分割成两部分，分别以硬件和软件的方式执行。采用 Ellpack-Itpack 格式作为矩阵存储格式，内部电路生成行交错数据流，提高了整个系统的计算性能。实验结果表明，与软件实现相比，本设计平均可以得到 2 倍左右的计算性能加速。

参考文献：

[1] MORRIS G R, PRASANNA V K, ANDERSON R D. A hybrid approach for mapping conjugate gradient onto an FPGA augmented reconfigurable supercomputer [C]// FCCM'06: Proceedings of the 14th IEEE Symposium Field-programmable Custom Computing Machines. Washington, DC: IEEE Computer Society, 2006: 3 - 12.

[2] KASBAH S J, DAMAJ I W. The jacobi method in reconfigurable hardware [C]// WCE 2007: Proceedings of the World Congress on Engineering. [S. l.]: Newswood Limited, 2007, 2: 823 - 827.

[3] LOPES A R, CONSTANTINIDES G A. A high throughput FPGA-based floating point conjugate gradient implementation [C]// ARC '08: Proceedings of the 4th International Workshop on Reconfigurable Computing: Architectures, Tools and Applications, LNCS 4943. Berlin: Springer-Verlag, 2008: 75 - 86.

[4] BAKOS J D, NAGAR K K. Exploiting matrix symmetry to improve FPGA-accelerated conjugate gradient [C]// FCCM 2009: 17th Annual IEEE International Symposium on Field Programmable Custom Computing Machines. Washington, DC: IEEE Computer Society, 2009: 223 - 226.

[5] SUN J, PETERSON G. Sparse matrix-vector multiplication design on FPGAs [C]// FCCM 2007: 15th Annual IEEE International Symposium on Field Programmable Custom Computing Machines. Washington, DC: IEEE Computer Society, 2007: 349 - 352.

[6] ZHUO L, PRASANNA V K. Scalable hybrid designs for linear algebra on reconfigurable computing systems [J]. IEEE Transactions on Computers, 2008, 57(12): 1661 - 1675.

[7] ROLDAO A, CONSTANTINIDES G. 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.

[8] WANG Y, BOYD S. Fast model predictive control using online optimization [J]. IEEE Transactions on Control Systems Technology, 2010, 18(2): 267 - 278.

[9] CONSTANTINIDES G A. Tutorial paper: parallel architectures for model predictive control [C]// ECC'09: Proceedings of European Control Conference. Washington, DC: IEEE Control System Society, 2009: 138 - 143.