

浮点矩阵向量乘法的 FPGA 设计与优化

薛永江, 宋庆增, 王瑞昆

(天津工业大学 计算机学院, 天津 300387)

摘 要: 提出了一种基于 IEEE-754 的 32 bit、64 bit 浮点数格式, 二叉树数据流(binary tree data flow)的矩阵向量乘法器. 其在 FPGA 上流水线和高高度并行化的高效执行. 以 Altera 公司的 EP2C70 为实现设备, 研究了设计的硬件规模, 时钟速度, 和峰值 GFLOPS 能力.

关键词: 可重构计算; 可编程逻辑门阵列; 矩阵向量乘法

中图分类号: TP30

文献标识码: A

文章编号: 1000-7180(2013)11-0064-04

Design and Optimization of Floating-Point Matrix-Vector Multiply for FPGAs

XUE Yong-jiang, SONG Qing-zeng, WANG Rui-kun

(College of Computer, Tianjin Polytechnic University, Tianjin 300387, China)

Abstract: In this paper, we design a matrix-vector multiplier which is based on IEEE-754 (32 bit, 64 bit) floating point data formats, and the binary tree data flow. The implementation in FPGA platform is pipeline and highly efficient parallel. With Altera EP2C70 as the target device, detailed study of the design of the hardware size, clock speed, capacity and peak GFLOPS.

Key words: reconfiguration computing; FPGA; matrix-vector multiply

1 引言

可重构计算作为一种新的计算体系结构, 采用可编程的硬件模块来实现计算, 填补了软硬件之间的鸿沟. 可重构计算不仅保持了硬件的高性能, 同时还具有接近于软件的灵活性^[1].

基于 FPGA 的可重构计算系统^[2]在功耗和计算性能方面具有优势, 但是在设计方面非常具有挑战性, 本文提出了一种浮点稠密矩阵向量乘法器的优化设计. 浮点格式选择 IEEE-754 的 32 bit 和 64 bit 浮点格式. 二叉树结构的数据流设计, 使得此乘法器在 FPGA 上可以流水线和高高度并行化的执行.

2 背景

2.1 数学背景

稠密矩阵与向量相乘是指给 n 阶稠密矩阵^[3].

$$A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

与 n 的向量 $x = [x_1, x_2, \dots, x_n]^T$, 计算 A 与 x 的乘积 $y = [y_1, y_2, \dots, y_n]^T$, 其串行算法表示如下:

计算 $y = Ax$:

for $k = 1$ to n do

$y_k = 0$;

for $j = 1$ to n do

$y_k = y_k + a_{kj} * x_j$;

end for;

end for;

算法计算量为 n^2 次浮点乘法和 n^2 次浮点加法.

2.2 浮点数据格式

受硬件容量和速度的限制, 早期的 FPGA 中很

少使用浮点数,通常采用定点数作为运算的数据格式。为了确定数据的位宽和小数点的位置,需要仔细的权衡计算精度,速度和硬件资源^[2]。然而,随着芯片制造技术的不断进展,当前的 FPGA 设备通常包含数百万的逻辑门,这使得实现大量浮点计算单元成为可能^[1]。

数值计算中,数据的精度要求高,数据的变化范围大。定点数在表示数值的精度和范围方面远不如浮点数,本文采用 IEEE-754 浮点数据标准的单精度(32 bit)和双精度(64 bit)。IEEE-754 浮点数据标准是最广泛使用的浮点数运算标准,IEEE 754 浮点标准由符号位、指数位和尾数位三部分组成。最高位为符号位 s ,指数位在中间,尾数位在最后。图 1 和图 2 分别标出了单精度浮点数和双精度浮点数格式。双精度和单精度浮点数数值分别按式(1)和式(2)计算:

$$Fp = (-1)^s 2^{e-1023} 1.f \tag{1}$$

$$Fp = (-1)^s 2^{e-127} 1.f \tag{2}$$

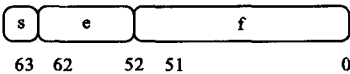


图 1 64 bit 浮点数

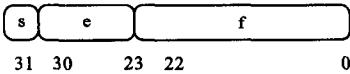


图 2 32 bit 浮点数

2.3 实现平台

以 Altera 公司的 EP2C70 芯片为实现设备,编程环境为 Quartus II 9.0。EP2C70 是一款低密度,低成本的 FPGA 芯片,与高端芯片相比,其速度和密度有很大差距。表 1 中列举了 EP2C70 芯片的硬件资源^[4]。

表 1 浮点加法器的面积和最大频率

EP2C70	资源总数
查找表(Lut)	68 416
寄存器(Reg)	68 416
硬件 DSP 块	300

受硬件规模限制,在 EP2C70 芯片上可生成的乘法器和加法器是有限的。EP2C70 最多只能实现 39 个 64 位浮点加法器,或者 16 个 64 位浮点乘法器,而高端芯片可实现数以百计的浮点加法和乘法。

同时,时钟的最大频率与高端芯片相比也有较大差距。比如 64 位浮点加法最大时钟频率只有 177.06 MHz,浮点乘法只有 118.02 MHz。而高端

芯片浮点加法和乘法能达到 400 MHz 到 500 MHz 之间,是 EP2C70 速度的 3~4 倍。

通过充分利用算法内在的并行性和 FPGA 的细粒度并行结构,设计高度并行的二叉树数据流,并进行了优化。虽然 EP2C70 是低密度,低成本的芯片,我们设计的矩阵向量乘法依旧可以达到非常高计算速度,其中单精度浮点可以达到 9.306 2 GFLOPS,双精度峰值为 3.482 9 GFLOPS。浮点加法器与乘法器分别见表 2、表 3。

表 2 浮点加法器

浮点加法	32 bit	64 bit
查找表(Lut)	264	597
寄存器(Reg)	927	1 742
最大频率/MHz	225.17	177.06
可生成的数量	73	39

表 3 浮点乘法器

浮点加法	32 bit	64 bit
查找表(Lut)	111	354
寄存器(Reg)	464	1 174
硬件 DSP 块	7	18
最大频率/MHz	222.92	118.02
可生成的数量	42	16

3 硬件设计

3.1 二叉树数据流

在 FPGA 硬件上实现矩阵乘向量的方法有很多种,其中内积方法(inner product method)是最为重要的方法^[5]。内积方法将 n 维矩阵乘向量分解为 n 次的向量乘向量,如式(3):

$$y_i = \sum_j A_{ij} * x_j \tag{3}$$

n 维向量乘向量是一个向量约化操作(vector reduction operation),也就是输入两个 n 维向量,输出一个数值。约化操作的最主要的特点是每次计算的都与前面的计算结果密切相关,因此非常适合串行执行^[6]。

为了并行处理 n 维向量乘向量,需要设计合理的数据流。为此,我们构造了二进制的乘加树,见图 3。乘加树共有 $\log_2(n) + 1$ 层, n 个乘法器作为树的第 0 层叶子节点。 $2n$ 个数据在一个时钟周期内进入作为叶子节点的乘法器。浮点乘法的时钟延迟是 11,所以 11 个时钟周期后,结果离开乘法器,进入第一层加法器。 $n - 1$ 个加法器排列成 $\log_2(n)$ 层二叉树结构,每层的加法叶子节点减半。加法器的时钟延

迟是 14,所以经过 $14 \log_2(n)$ 时钟周期后,输出一个数值,此数值即为 n 维向量乘向量的计结果.

经过具有 $\log_2(n) + 1$ 层结构的乘加树计算后, $2n$ 个数据生成一个数据,所需的时钟周期为

$$\alpha_m + \alpha_a \log_2(n)$$

式中, α_m 为浮点乘法延迟周期, α_a 为浮点加法延迟周期,数值见表 4. 浮点乘加操作都是深度流水的,所以 n 维矩阵乘向量可以流水执行所需时钟为

$$n + \alpha_m + \alpha_a \log_2(n) - 1 \tag{4}$$

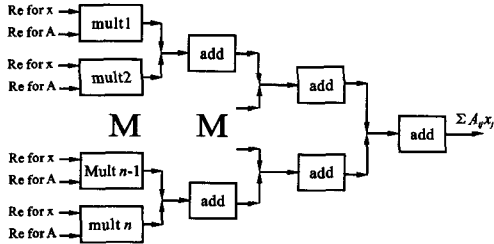


图 3 二叉树的数据流

表 4 时钟延迟

时钟延迟周期	
浮点加法延迟 (α_a)	14
浮点乘法延迟 (α_m)	11

3.2 矩阵划分

n 维向量乘向量需要 n 个浮点乘法和 $n - 1$ 个浮点加法. 受硬件规模的限制, FPGA 能实现的浮点加法和乘法是有限的 m , 对于芯片 EP2C70, 双精度时 $m = 16$, 单精度时 $m = 42$. 即使最大的 FPGA 也只能实现数百个浮点加法和乘法. 相对于 FPGA 可实现浮点加法和乘法数目 m , 向量维数 n 一般是一个非常大的数, 也就是 $n \gg m$.

为此必须将矩阵进行划分, 以适应硬件执行. 本文采用逐列一维块划分, 与通常的逐行一维块划分相比, 这种方法可以避免复杂的约化电路(reduction circuit), 硬件执行效率高, 内存带宽要求低.

矩阵 A 每 p 列划分为一个块(这里 p 的具体数值受硬件资源的约束, p 小于硬件可实现的最大浮点乘法和浮点加法数 m), 为叙述简单, 假设 n 可以被 p 整除. 在 n 不能被 p 整除时, 可以在最后一个分块填入几列 0 元. 逐列一维块划分, 矩阵 $A = [A_0, A_1, A_2, \dots, A_{n/p}]$, 其中

$$A_k = \begin{bmatrix} a_{1,k \times p} & a_{1,k \times p+1} & \cdots & a_{1,k \times p+p-1} \\ a_{2,k \times p} & a_{2,k \times p} & \cdots & a_{1,k \times p+p-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,k \times p} & a_{n,k \times p} & a_{1,k \times p} & a_{1,k \times p+p-1} \end{bmatrix}$$

与之相对应, 向量 x 也每 p 个元素划分为一个块:

$$x = [x^0, x^1, x^2, \dots, x^{n/p}]$$

其中

$$x^k = [x_{k \times p}, x_{k \times p+1}, x_{k \times p+2}, \dots, x_{k \times p+p-1}]$$

所以

$$\begin{aligned} y &= Ax \\ &= [A_0, A_1, A_2, \dots, A_{n/p}] * [x^0, x^1, x^2, \dots, x^{n/p}]' \\ &= \sum_{i=1}^{n/p} A_i x^i. \end{aligned}$$

图 4 中以一个 16×16 矩阵为例, 将矩阵划分成了四个 16×4 的矩阵, 对应的向量也划分成 4 个 4 维向量.

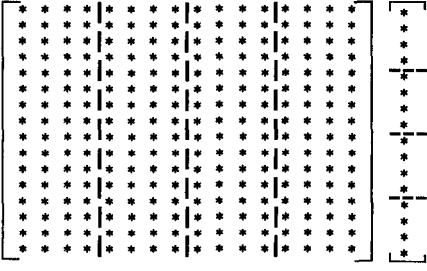


图 4 矩阵的逐列一维划分和对应的向量划分

4 实验结果分析

4.1 速度分析

Quartus II 9.0 作为编程工具, Modelsim SE 6.5 作为仿真工具, 在 Altera 公司的 EP2C70 芯片上实现整个设计.

运行在 PC 机上, 软件执行的矩阵向量乘法作为对照组, 与 FPGA 性能进行对比. PC 机的 CPU 采用 Intel Pentium D 2.8 MHz(双核), 内存为 512 MB, 详细配置如表 5 所示.

表 5 对照组硬件配置信息

主频	一级缓存	二级缓存	内存
2.8/GHz	2x28 kB	2x1 MB	DDR2, 512 MB

如图 5 所示, 与软件执行相比, FPGA 有明显的速度优势. 当使用 8 个浮点加法和浮点乘法器时, FPGA 的运行速度已经超过了软件执行. 当使用 16 个硬件时, FPGA 的执行速度大约是软件执行的三倍以上.

4.2 加速比与并行效率

加速比和并行效率是衡量设计质量的重要指标, 它描述了增加硬件资源对实际可获得的性能改进.

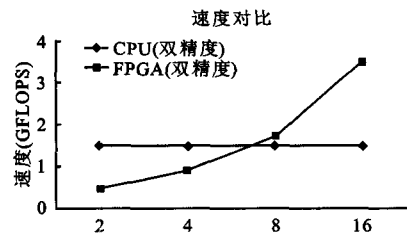


图5 FPGA与CPU速度对比

定义相对加速比为

$$s_p = \frac{t_1}{t_p} \tag{5}$$

式中, t_1 为只一个浮点加法和浮点乘法器时的计算时间, 有下面的计算公式:

$$t_1 = \max(\frac{n^2}{f_{mult}}, \frac{n^2}{f_{add}}) \tag{6}$$

由于无论是单精度还是双精度数, 浮点加法器的运行频率都大于浮点乘法器的运行频率, 所以上式变为

$$t_1 = \frac{n^2}{f_{mult}} \tag{7}$$

式中, t_p 为 p 个浮点加法和浮点乘法器时的计算时间, 所以

$$t_p = \frac{n^2}{p} * \frac{1}{f_p} \tag{8}$$

所以性对加速变为

$$s_p = \frac{t_1}{t_p} = \frac{pf_p}{f_{mult}} \tag{9}$$

并行效率定义相对加速比的倒数:

$$E_p = \frac{s_p}{p} \tag{10}$$

表6中列出了64 bit数据格式下的时钟频率, 加速比和并行效率. 随着硬件规模增加到16个浮点加法和乘法器, 时钟频率下降不足8%((118.02-108.84)/118.02), 具有近似线性的加速比.

表6 加速比与并行效率(64 bit)

	P=2	P=4	P=8	P=16
Clock/MHz	118.02	113.71	115.59	108.84
加速比	2.000 0	3.853 9	7.835 3	14.755 5
并行效率	1.000 0	0.963 5	0.979 4	0.922 2

表7中给出了32 bit数据格式下的时钟频率, 加速比和并行效率. 硬件规模增加到32时, 执行速度变原来的20.87倍, 计算性能得到很好的提升.

5 结束语

本文针对矩阵乘向量运算固有的并行性, 设计

表7 加速比与并行效率(32 bit)

	P=2	P=4	P=8	P=16	P=32
Clock/MHz	213.45	186.25	185.56	172.24	145.41
加速比	1.915 0	3.342 0	6.659 2	12.362 5	20.873 5
并行效率	0.957 5	0.835 5	0.832 4	0.772 7	0.652 3

了基于二叉树数据流的矩阵向量乘法器. 以Altera公司的EP2C70为实现设备, 详细的研究了设计的硬件规模、时钟速度和峰值GFLOPS能力.

与软件执行相比, 双精度运算速度能达到其3倍左右, 具有非常高的速度. 同时这种设计具有良好的并行效率和加速比.

参考文献:

[1] Clive “Max” Maxfield. FPGA设计指南:器件、工具和流程[M]. 杜生海, 邢闻, 译. 北京: 人民邮电出版社, 2007.

[2] Morris G. Mapping sparse matrix scientific applications onto FPGA augmented recon—figurables supercomputers[D]. California: University of Southern California, 2006.

[3] Zhuo L, Prasanna V K. Scalable and modular algorithms for floating point matrix multiplication on reconfigurable computing systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2007, 18(4): 433-448.

[4] Zhuo L, Prasanna V K. Scalable hybrid designs for linear algebra on reconfigurable computing systems [J]. IEEE Transactions on Computers, 2008, 57(12): 1661-1975.

[5] Scrofano R, Zhuo L. Area—efficient arithmetic expression evaluation using deeply pipelined floating—point cores[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2008, 16(2): 167-176.

[6] Baboulin M, Buttari A. Accelerating scientific computations with mixed precision algorithms[J]. Computer Physics Communications, 2009, 180(12): 2526-2533.

作者简介:

薛永江 男, (1983-), 硕士, 助教. 研究方向为嵌入式系统并行计算.

宋庆增 男, (1980-), 博士, 讲师. 研究方向为并行计算.

王瑞坤 男, (1983-), 硕士, 讲师. 研究方向为嵌入式系统.