文章编号: 1001- 2486(2009) 05- 0029- 04

基于 FPGA 的带回溯的 Smith-Waterman 算法加速器的设计与实现*

邹 丹, 窦 勇, 夏 飞, 倪时策
(国防科技大学 计算机学院, 湖南 长沙 410073)

摘 要: 针对传统的 Smith-Waterman 硬件算法加速器未保存回溯路径而无法回溯的问题, 通过将计算路径存入外存, 在 FPGA 平台上基于脉动阵列实现了带回溯的 Smith-Waterman 算法加速器, 详细阐述了算法加速器回溯设计中的关键技术以及算法加速器的系统结构。实验表明, 与传统的解决方案相比, 带回溯的算法加速器最高可获得 161 倍加速比, 能够有效提高带回溯的 Smith-Waterman 算法执行效率。

关键词: FPGA; Smith-Waterman 算法; 脉动阵列; 回溯
中图分类号: TP302 文献标识码: A

FPGA-based Smith-Waterman Algorithm Accelerator with Backtracking

ZOU Dan, DOU Yong, XIA Fei, NI Shi-ce
(College of Computer, National Univ. of Defense Technology, Changsha 410073, China)

Abstract: The Smith-Waterman algorithm accelerator with backtracking, which has not been implemented in hardware before, is designed and implemented on FPGA platform with systolic array by storing the path data into DRAM. The key techniques of backtracking design and the architecture of algorithm accelerator are discussed in detail. Compared with the conventional scheme, the FPGA-based accelerator with backtracking is more effective, with the acceleration reaching 161.

Key words: FPGA; Smith-Waterman algorithm; systolic array; backtracking

Smith-Waterman 算法(简称 SW 算法)是生物信息学研究中双序列比对的核心算法之一, 由打分和回溯两部分组成。随着生物序列数据库规模的迅速增长, 序列比对研究对计算能力的需求远远超过计算能力的增长, 使用传统的软件方法进行序列比对已不能满足需要。近年来, 国内外学者在基于 FPGA 平台的 SW 算法加速器方面做了大量研究工作[1-3], 但受限于片上存储资源无法保存完整的计算路径数据, 只能实现无回溯的硬件打分, 回溯部分仍然由软件完成, 软件回溯的执行时间成为提高算法执行效率的瓶颈。

1 Smith-Waterman 算法

SW 算法[4] 用于确定 DNA 或蛋白质序列的相似性区域, 执行过程分为两个步骤: (1) 计算相似性矩阵, 同时存储计算路径; (2) 由最大得分值所在位置开始, 沿计算路径进行回溯, 得到最佳片段对。前者称为打分阶段, 后者称为回溯阶段。

在打分阶段, 计算相似性矩阵的数据依赖关系如图 1 所示。同一反对角线上的元素间不存在数据依赖关系, 因此可从矩阵的左上角出发, 按照对角线方向对位于反对角线上的元素进行并行计算。图 1 中所标数字代表计算的先后次序, 每个处理单元(PE)计算矩阵的一行。

回溯阶段在打分完成后开始。与相似性矩阵计算过程相反, 回溯过程需要从矩阵的最大值元素向前反推, 是一个串行过程。对于任意回溯点 $V(i,j)$, 回溯路径指向 $V(i-1,j)$、$V(i,j-1)$ 和 $V(i-1,j-1)$ 三者中最大值所在位置。垂直路径表示在横轴序列的对应位置插入空位, 水平路径表示在纵轴序列的

* 收稿日期: 2009- 07- 03
基金项目: 教育部"高性能微处理器技术"创新团队资助项目(IRT0614)
作者简介: 邹丹(1984—), 男, 博士生。

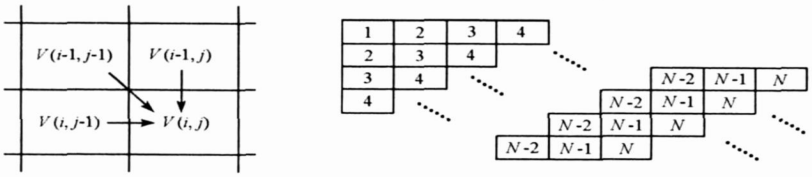

图 1 比对过程中的数据依赖关系

Fig. 1 Data dependency of alignment

对应位置插入空位, 指向对角元素的路径表示两序列在当前回溯点进行了一次匹配。$V(i,j)$ 为 0 时回溯结束。

2 算法特征分析

设输入序列 S、T 的长度分别为 M、N, 则打分阶段产生的计算路径矩阵规模为 $O(MN)$。由于 FPGA 片内存储资源无法满足大规模序列计算路径的存储需求, 必须在打分阶段计算相似性矩阵的同时将计算路径数据存入外存储器, 在回溯阶段再将计算路径分批载入 FPGA。访存带宽限制下的 PE 阵列规模设计和数据调度策略是回溯设计中需要解决的关键问题。

2.1 阵列规模

计算路径的访存带宽需求与 PE 阵列规模 L 成正比, 且与计算路径的表示方法有关。相似性矩阵中的每个位置的回溯选择都包含 4 种可能, 因此采用 2bit 表示每个矩阵元素的回溯选择。对于任何回溯点 $V(i,j)$, 01、10、11 分别代表向 $V(i-1,j)$ 回溯、向 $V(i,j-1)$ 回溯和向 $V(i-1,j-1)$ 回溯, 00 表示回溯过程终止, 则每个时钟周期产生的计算路径数据量为 $2L$ bit。

设算法加速器的时钟频率为 F, DDR 的有效带宽为 B_d, 则计算路径数据的输出带宽 $B_t = LF$ bit/s。为了实现计算路径数据的访存/计算时间重叠, 需满足 $B_t < B_d$, 解不等式, 得到 $L < L_m$, 则无停顿的 PE 阵列规模最大不能超过 L_m。

2.2 数据调度

在打分阶段, 相似性矩阵的中间数据和计算路径数据同时产生且都需要存入外存。为了实现并行访存, 采用双通道设计, 将中间数据与计算路径数据分别存入不同通道的 DRAM。由于计算路径矩阵是二维结构, 这就需要将二维的矩阵结构映射到一维的 DRAM 空间。

在打分阶段, 当阵列规模与 S、T 序列长度满足 $L < M \leqslant N$ 时, 需要将 S 序列分批送入阵列, 每批送入 L 个字符, 然后再将 T 序列送入阵列与 S 序列进行比对。每一批可以完成一个 $N \times L$ 子矩阵的比对, 同时产生一个相同规模的计算路径子矩阵。

设阵列规模 $L = 4r$, 为使存储顺序与数据产生顺序一致, 将计算路径子矩阵每行四个相邻元素合并为一个字节, 字节地址逐行顺序递增, 将连续产生的数据合并存入 DRAM 以提高写效率。如图 2 所示。

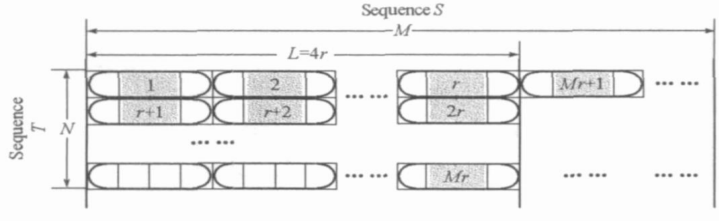

图 2 比对过程中的地址映射关系

Fig. 2 Address mapping of alignment

计算路径矩阵中元素坐标 (x,y) 与 DRAM 空间中字节地址 $A_{x,y}$ 以及字节内偏移 $O_{x,y}$ 满足映射关系:

$$\begin{cases} x = \dfrac{4A_{x,y}}{L} \bmod N + 1 \\ y = \left\lfloor \dfrac{4A_{x,y}}{NL} \right\rfloor \times L + A_{x,y} \bmod \dfrac{L}{4} + O_{x,y} \end{cases} \tag{1}$$

打分完成后, 开始回溯。计算路径数据按块从 DRAM 载入 FPGA, 每块对应一个 $P \times P$ 的计算路径子矩阵, 且 $L \bmod P = 0$。每个时钟周期进行一步回溯, 在每个计算路径子矩阵中的回溯最多需要 P 个时钟周期。当回溯在一个计算子矩阵中进行时, 需要将与其相邻的三个计算路径子矩阵预读入片内 RAM 以减少停顿。如图 3 所示, 若当前回溯位置进入# 6 计算路径子矩阵时, 开始预读# 2、# 3、# 5 子矩阵。实际上, 当回溯位置在# 9 子矩阵内时, # 5 子矩阵已经预读, 故只需预读# 2、# 3 子矩阵。

在每个计算路径子矩阵中的回溯路径长度为 0~ P, 子矩阵内相对应的回溯时间需要 0~ P 个时钟周期。在回溯开始前, 将当前回溯位置所在子矩阵及三个相邻子矩阵载入片内 RAM, 然后启动回溯过程。在回溯位置移出当前子矩阵时, 如果目标子矩阵就绪, 则将回溯位置移入并继续回溯过程, 同时开始预读新的相邻子矩阵。

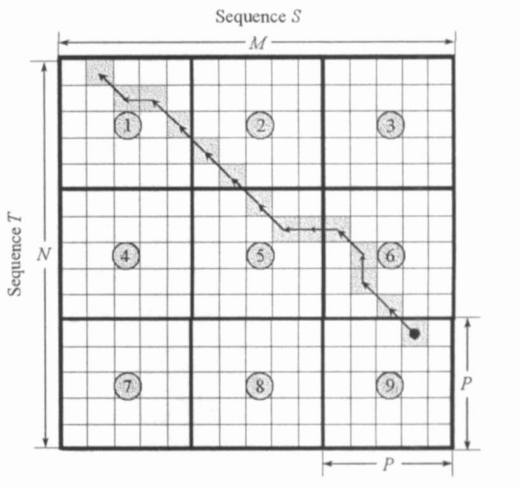

图 3　计算路径数据预读子矩阵

Fig. 3　Prefetch submatrix of calculation path

3　FPGA 实现

如图 4 所示, 基于 FPGA 的带回溯的 SW 算法加速器系统平台由一台通用计算机和 SW 算法加速器构成。主机负责与用户的交互, 将用户输入的序列加载至通道 1 的内存指定区域, 并启动 SW 算法加速器。加速器完成 SW 算法的计算和回溯过程, 将最大得分和最佳片段对写入通道 1 的指定内存区域, 然后向主机发送回溯完成信号, 由主机从内存中取回比对和回溯结果。

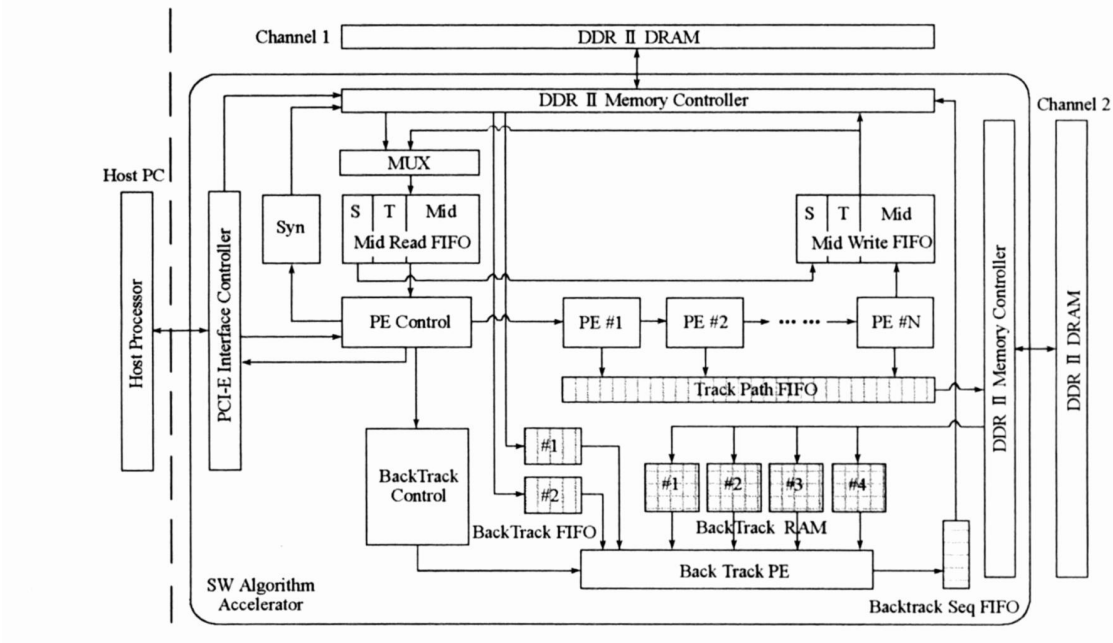

图 4　系统硬件结构

Fig. 4　System hardware architecture

加速器以高性能商用 FPGA 芯片为核心,配置两个 2GB DDR 存储器,通过 PCI-E × 8 数据通道与主机相连,实现通用处理器与算法加速器的协同工作。

FPGA 芯片是算法加速器的核心,其内部由 PCI-E 接口控制器、DDR 存储控制器和 SW 算法逻辑三部分构成。PCI-E 接口控制器基于 Xilinx PCI-E IP 核实现,负责加速器与主机间的通信。DDR 存储控制器负责实现对片外 DDR DRAM 的存储访问。

SW 算法逻辑包括计算逻辑和回溯逻辑两部分。计算逻辑包括 PE 阵列控制模块、PE 阵列、中间数据读/写缓冲以及计算路径缓冲四部分。PE 阵列控制模块负责 PE 阵列的初始化和控制,并为 PE 阵列送入比对序列和中间数据;PE 阵列负责相似性矩阵的并行计算,由 64 个 PE 模块串联构成,每个 PE 模块输出的计算路径数据合并后写入计算路径缓冲,存入通道 2 处的 DRAM。回溯逻辑包括回溯控制模块,回溯处理单元,回溯子矩阵缓存,回溯序列缓冲,序列输出缓冲五部分。回溯计算路径子矩阵规模为 64 × 64。回溯控制模块负责回溯处理单元的启动和初始化。回溯子矩阵缓存负责加载相邻的四个计算路径子矩阵,由 4 个 64 × 64 × 2bit 的 Block RAM 构成。回溯序列缓冲负责加载相应的 S/T 序列片段。回溯处理单元根据计算路径,将相应的序列字符对写入序列输出缓冲,并沿回溯方向移动到下一个回溯位置,重复此过程直到计算路径截止。序列输出缓冲将数据写回到通道 1 的 DRAM 中。

4 实验与性能分析

测试平台的主机配置为 Intel Xeon 2.67GHz CPU, 2GB 主存。SW 算法加速器硬件主要包括 1 片 FPGA 芯片(XC5VLX330),2 条 2GB 的 DDR 存储条。加速器工作频率为 100MHz。

测试了不同长度的 DNA 序列在传统的硬件打分、软件回溯解决方案下的执行时间,并与带回溯的 SW 算法加速器进行了比较。随机生成每种长度的测试序列各 10 组,执行时间为 10 组测试结果的平均值。

传统解决方案执行时间为硬件打分和软件回溯的执行时间之和。其中,硬件打分的执行时间取自 Altera 公司的实验数据[3],软件回溯的执行时间为 ClustalW v1.83 在主机上的实测数据。带回溯的 SW 算法加速器方案的执行时间为从硬件启动到输出打分和回溯结果到内存的时间,见表 1。

表 1 相对传统解决方案的加速比

Tab. 1 Speedup constrasting to conventional scheme

S/T 序列长度(bp)	传统解决方案执行时间(s)	带回溯的加速器执行时间(s)	加速比
1024	0.000355	0.006228	17.54
2048	0.000865	0.063860	73.83
4096	0.002885	0.419961	145.57
8192	0.010798	1.478492	136.92
16384	0.042436	6.871890	161.94

由实验数据可见,基于 FPGA 平台带回溯的 SW 算法加速器优于传统的解决方案,能够有效提高 SW 算法的执行效率。

参 考 文 献:

[1] Yu C W, Kwong K H, Lee K H, et al. A Smith-Waterman Systolic Cell[C]// Proc. 13th Field-programmable Logic and Applications (FPL'03), Springer, Berlin, 2003: 375-384.

[2] Oliver T, Schmidt B, Maskell D. Hyper Customized Processors for Bio-sequence Database Scanning on FPGAs[C]// Proc. 13th International Symposium on Field Programmable Gate Arrays (FPGA'05), Monterey, CA, 2005: 229-237.

[3] Zhang P, Tan G, Gao G R. Implementation of the Smith-Waterman Algorithm on a Reconfigurable Supercomputing Platform[R]. WP-01035-1.0, 2007.

[4] Smith T F, Waterman M S. Identification of Common Molecular Subsequences[J]. Journal of Molecular Biology, 1981, 147(1): 195-197.