计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2017,26(12):124−129 [doi: 10.15888/j.cnki.csa.006115]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

面向高性能 DSP 的图像滤波库优化①

郑晓松, 顾乃杰, 叶　鸿

(中国科学技术大学 计算机科学与技术学院, 合肥 230027)
(中国科学技术大学 安徽省计算与通信软件重点实验室, 合肥 230027)
(中国科学技术大学 先进技术研究院, 合肥 230027)

摘　要: 滤波函数在图像处理领域起着至关重要的作用. 传统的滤波实现方法以窗口为处理单位, 但窗口尺寸较小导致指令流水线频繁中断. 针对该问题, 本文提出了算法切片的优化方法. 首先, 对滤波算法进行切片, 使得每个切片处理滤波窗口中的一个像素点; 接着, 根据切片处理的像素点在滤波窗口中的位置计算出有效处理范围, 去除图像边缘处理中复杂的条件判断; 最后, 按列方向进行多簇流水, 让流水线重复执行大量相同的指令序列. 结合 BWDSP 1042 的特殊指令和硬件逻辑, 对中值滤波等图像滤波函数进行了优化. 实验结果表明: 在四簇流水模式下, 所有图像滤波函数性能均提升了 51 倍以上.

关键词: 数字信号处理器; 超流水线; 多簇; 图像滤波; 软件流水化; 中值滤波

引用格式: 郑晓松,顾乃杰,叶鸿.面向高性能 DSP 的图像滤波库优化.计算机系统应用,2017,26(12):124–129. http://www.c-s-a.org.cn/1003-3254/6115.html

Optimization of Image Filtering Library for High-Performance DSP

ZHENG Xiao-Song, GU Nai-Jie, YE Hong

(School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)
(Anhui Province Key Laboratory of Computing and Communication Software, University of Science and Technology of China, Hefei 230027, China)
(Institute of Advanced Technology, University of Science and Technology of China, Hefei 230027, China)

Abstract: Filter functions play a significant role in image processing. The traditional implementation method takes the window as the processing unit, whose size is so small that the pipeline is interrupted frequently. This paper proposes an optimization method of algorithm slicing to settle this problem. First, the filtering algorithm is sliced so that each slice processes one pixel in the filter window. Then, the effective processing range is calculated from the position of the pixel in the filter window to remove the complex conditional judgment of the edge of the image. Finally, the software pipeline is carried out in the column direction, allowing the pipeline to repeat a large number of identical instruction sequences. Combined with BWDSP 1042 special instructions and hardware logic, the median filter and other image filtering functions are optimized. The experimental results show that the performance of the entire image filtering functions is improved by more than 51 times in the four-cluster pipeline mode.

Key words: digital signal processor(DSP); superpipline; multi-cluster; image filtering; software pipeline; median filter

1　引言

图像滤波库涵盖了图像处理领域核心的滤波函数, 包括维纳滤波[1](Wiener filter), 中值滤波[2](Median filter), 高斯滤波[3](Gauss filter), 拉普拉斯滤波[4](Laplacian filter) 等函数. 这些函数要在尽可能保留细节特征的前提下对目标图像进行去噪处理, 是图像复

① 基金项目: 安徽省自然科学基金 (1408085MKL06); 高等学校学科创新引智计划基金 (B07033)
收稿时间: 2017-03-15; 修改时间: 2017-04-05; 采用时间: 2017-04-13

原、分割、特征提取、图像识别等后续工作的基础[5]. 因此, 对图像滤波库进行优化, 提高函数处理效率, 具有十分重要的意义.

图像滤波处理需要对输入图像中每个像素及其周围的邻域进行操作, 待处理数据量庞大. 随着现代图像采集设备精密度越来越高, 图像尺寸也越来越大, 计算量也随之增加. 为了提高图像的质量和显示效果, 往往需要使用一些复杂的滤波算法, 这些算法不仅时间复杂度高, 而且数据相关性大, 计算过程非常耗时.

为了克服上述难题, 研究人员一方面从硬件加速的角度来提高计算速度. 文献[6]和文献[7]对 3×3 大小的滤波窗口中值求解过程进行硬件实现, 来提高中值滤波函数的效率. 该方法只针对特定窗口大小的中值滤波, 缺乏足够的通用性. 另一方面从并行加速的角度来提高处理速度——将图像滤波函数移植到通用数字信号处理器 (Digital signal processor, DSP) 上. 文献[8]通过去除相邻窗口的重复计算, 在 TMS320C6416 平台上对均值滤波等函数进行优化, 性能提升了 4.3 倍以上. 文献[9]利用相邻窗口的相关性, 减少了计算与存取数据的次数, 在 TMS320C6416 平台上对低通滤波器进行优化, 性能提升 7.5 倍. 文献[10]选用快速排序算法进行排序, 在 TMS320C6000 系列 DSP 上对中值滤波函数进行软件流水化, 使得函数处理速度提升了 2.8 倍. 上述优化实现都以滤波窗口为处理单位, 未考虑到由于循环体长度较小而导致指令流水线频繁中断的问题.

随着计算机技术的发展, 现代高性能 DSP 计算能力更加强大. 现代高性能 DSP 普遍采用多簇 (multi-cluster) 系统架构和超流水线 (superpipline)、超长指令字 (Very long instruction word, VLIW) 等技术[11,12]. 如何同时发挥多种硬件技术的计算能力, 成为当前研究的热点.

本文提出的算法切片方法通过对复杂的图像滤波算法进行分解, 细化最小处理单位, 并按列方向多簇流水化, 来让流水线重复执行大量相同的操作序列, 减少流水线中断次数, 同时提高流水线加速比. 结合 BWDSP 1042 平台的特性, 对图像滤波库进行优化, 实验结果表明算法切片方法能够大幅度提升函数处理效率.

本文剩余部分组织如下: 第 2 节介绍实验平台的体系结构, 第 3 节介绍本文使用的优化方法, 第 4 节介绍如何结合平台特性进行算法实现, 第 5 节是实验结果与分析, 第 6 节总结本文工作并指出下一步工作方向.

2 平台简介

BWDSP 系列处理器由国内某研究所设计制造, 拥有完全的自主知识产权, 其成功应用打破了国外高端数字信号处理器芯片对中国高性能计算领域的垄断[13]. 因此, 针对该平台的特点进行函数移植与优化, 充分挖掘处理器计算能力, 具有十分重要的国产化意义.

BWDSP 1042 处理器采用哈佛结构, 以及超流水线、超长指令字等并行技术, 适用于雷达信号处理、图像处理等计算领域. BWDSP 1042 处理器的体系结构如图 1 所示, 下面是简要的平台介绍.

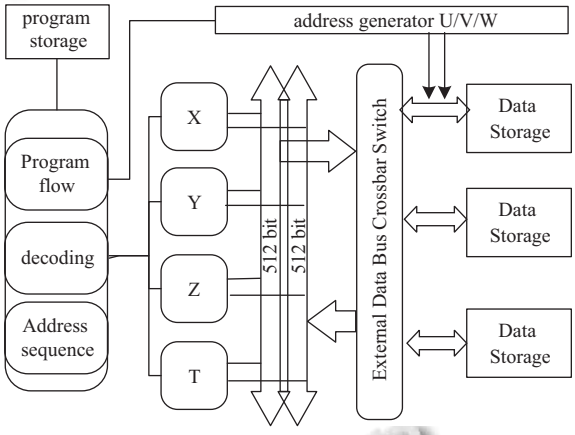

图 1 BWDSP1042 系统架构

(1) 包含 4 个结构、功能完全相同的簇, 分别用 X, Y, Z, T 表示. 每簇含有 128 个通用寄存器, 8 个算术逻辑单元 (Arithmetic logic unit, ALU), 8 个乘法器 (Multiplier, MUL), 4 个移位器 (Shifter, SHF), 1 个超算器 (Super unit, SPU).

(2) 包含 3 个结构、功能完全相同且相互独立的地址产生器, 分别用 U, V, W 表示. 支持“两读一写”或“两写一读”. “两读一写”是指一个指令周期内每个簇读取 2 个 32 位的复数、写回 1 个 32 位的复数. “两写一读”是指一个指令周期内每个簇写回 2 个 32 位的复数、读取 1 个 32 位的复数. 也即同时进行读写时, 数据读总线和数据写总线宽度之和不得超过 768 位.

(3) 采用超长指令字技术, 共 16 个指令槽. 在没有资源冲突的情况下, 最多可以同时执行 16 条单字指令. 对于条件跳转等分支指令只能放在第一个指令槽, 而且每个指令行最多只能有一条该类指令.

(4) 流水线共有 13 级, 其中取指令 3 级 (FE0~FE2), 指令缓冲池 3 级 (IAB1~IAB3), 指令译码 4 级 (DC1~DC4), 指令发射 1 级 (AC), 指令执行 1 级 (EX), 指令结果写回 1 级 (WB). 采用多级取指是为了支持分支预测, 以减少流水线中断.

3 优化方法

图像滤波算法实现时, 以滤波窗口 (窗口大小为 $p\times q$) 为处理单位进行软件流水, 由于第 i 行最后一个元素的存储地址和第 i+1 行第一个元素的存储地址不连续, 对每个窗口遍历一次, 要产生 p 次因存储地址不连续而导致的流水线中断. 对于 $m\times n$ 的输入图像 **F**, 共产生此类型流水线中断约 $m\times n\times p$ 次. 现代超流水线处理器对指令流水线划分更加精细, 用以提高并行能力, 因此一个流水线中断会造成多个指令周期的流水线停顿.

3.1 算法切片

对于超流水处理器, 重叠执行的指令条数越多, 流水线加速比越趋向于流水线级数, 超流水技术得以充分发挥[14]. 分支预取失败和循环体长度较小都是造成流水线停顿的重要原因, 可以从这两方面进行优化. 现代高性能 DSP 普遍采用零开销循环技术, 在处理器中加入硬件逻辑, 来提高循环体结尾处分支预取的准确度, 可以有效减少流水线的性能开销. 对于循环体长度较小的问题, 可以使用软件优化技术, 通过改进计算方法来解决.

任何一个复杂的算法 S 都是由读操作 R、写操作 W 和运算操作 Cal 组成的一个操作序列, 如 S = {R, Cal^+, R, Cal^+, ……, R, R, Cal^+, W}(Cal^+表示至少有一个运算操作). 将该操作序列切分成多个子序列 S_i(简称为切片), 使得每个切片包含更少的操作. 每个切片 S_i 的第一个操作从内存中读数据, 后面的几个操作对它进行处理. 为了保证切片后计算结果的正确性, 在每个切片的尾部添加一些操作, 用于将该切片的计算结果与之前的计算结果进行合并, 或者直接保存计算结果, 以等待后续切片读取. 算法 S 的切片如下:

S_1 = {R, Cal^+, W}

S_2 = {R, Cal^+, R, Cal, W}

……

S_n = {R, R, Cal^+, R, Cal, W}

对图像滤波算法 S 进行切片, 使得每个切片处理滤波窗口中的一个像素点. 若算法 S 的空间复杂度为 O(1)(不包括输出图像所占用的内存空间), 则任一切片 S_i 的计算只可能依赖前面常数个切片的计算结果. 因为算法 S 的空间复杂度为 O(1), 计算过程中只需要使用常数个变量来保存中间计算结果, 所以一个切片的计算只可能依赖于前面常数个切片的计算结果. 保存所有滤波窗口同一切片的计算结果只需要 O($m\times n$) 个临时空间.

算法切片对一个滤波窗口中的操作序列进行分解, 切片成为最小的处理单位. 由于各滤波窗口之间没有数据依赖关系, 不同窗口的切片可以交叉处理. 出于存储地址规律性的考虑, 选择同时处理所有滤波窗口的同一个切片. 如图 2 所示, 以矩阵 P 中像素点为中心的滤波窗口, 其位置 (-1, -1) 对应的像素点组成矩阵 Q. 对输入图像所有滤波窗口执行同一个切片包含的操作, 就是对输入图像子矩阵 **F**'中的每个像素点执行同样的操作 (去除边缘的无效部分). 由于输入图像的尺寸比较大, 可以充分发挥流水线的优势.

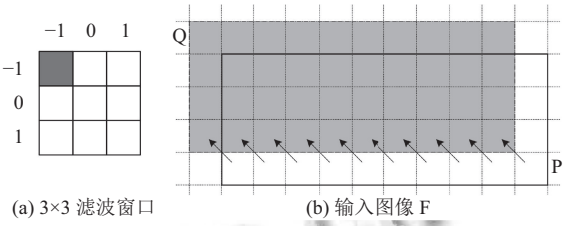

图 2 滤波窗口对应关系示意图

假设要对一个滤波窗口求 Hadamard 乘积 (所有线性滤波器的基本操作), 进行切分后, 每个切片 S_i 包含如下操作: 读入滤波窗口的第 i 个像素点, 将其乘以一个系数, 再与前面的计算结果进行合并. 按照切片顺序依次进行软件流水, 最多产生 $m\times p\times q$ 次流水线中断, 较切片前大幅度减少, 而且流水线加速比也有所提高.

使用算法切片优化方法, 每个滤波窗口的原有操作数目保持不变, 只在切片中增加了少量的内存读写操作. 虽然该方法增加了数据总线的压力, 但是现代高性能 DSP 普遍采用高带宽的数据总线, 以配合处理器并行计算能力的提升, 这使得上述以增加数据总线带宽来减少流水线中断的策略可以有效提高函数性能.

对空间复杂度较高的滤波算法施用算法切片方法, 可能会因为需要的临时存储空间太多, 而难以满足实际应用需要. 可以使用以时间换空间的策略, 设计一个

时间复杂度更高, 但空间复杂度为 O(1) 的算法.

3.2 边缘处理优化

以输入图像边缘的像素点为滤波窗口中心, 与窗口中某些位置对应的像素点可能超出了输入图像的有效范围. 为了保证计算结果的正确性, 边缘处理时要判断像素位置的有效性. 在尺寸大小为 $m\times n$ 的图像中, 有 $m\times n-(m-p+1)\times(n-q+1)$ 个滤波窗口需要进行特殊处理, 大量的条件判断导致超流水线处理器效率低下.

通过算法切片后, 可以根据窗口中当前位置相对于窗口中心的偏移量计算出输入图像的有效处理范围. 假设当前处理像素点与滤波窗口中心的偏移量为 (x, y), 则与输入图像中 $\mathbf{F}_{i,j}(0\le i\le m-1, 0\le j\le n-1)$ 对应的像素点为 $\mathbf{F'}_{i,j}(x\le i\le m+x-1, y\le j\le n+y-1)$. 若 $x\ne 0$ 或 $y\ne 0$, 均会导致一些位置的计算无效. 使用算法切片方法, 只需缩小处理范围, 抑制无效区域的计算, 便可以得到正确的结果. 故有效的处理范围为 $\mathbf{F'}_{i,j}(\max(0, x)\le i\le m-1+\min(0, x);\ \max(0, y)\le j\le n-1+\min(0, y))$. 算法切片方法只需通过 $p\times q$ 次计算来求出有效处理范围, 避免了传统实现方式中图像边缘像素点处理的复杂性.

3.3 按列处理

通过算法切片后, 需要多次遍历输入图像的子矩阵, 由于图像像素数据在内存中按行存储, 一般选择按行方向处理. 在分簇结构的处理器中, 依次读入相邻的几个像素, 让多个簇同时进行处理. 但是每行最后剩余的待处理数据个数可能比处理器簇数要少, 不能让多个簇同时执行, 此时需要进行尾数处理.

对于 BWDSP 1042 这类无数据 cache 结构的 DSP, 不需要考虑存储空间的局部性, 可以全部按列进行软件流水. 以 s 列为基本单位 (s 与处理器簇数以及每个簇能够处理的数据个数相关) 依次进行软件流水, 最后对剩余的几列进行尾数处理.

求输入图像 F 滤波窗口的 Hadamard 乘积, 按行软件流水化, 大约有 $m\times p\times q$ 次流水线中断. 全部按列方向处理最多有 $[(n \bmod s)+1]\times p\times q$ 次流水线中断, 而且所有像素点的计算都在流水线中完成, 不需要通过串行的方式处理尾数. 常见高性能 DSP 分成 4 簇或 8 簇, 并且每个簇能够处理 1~2 个像素点, 对于大部分图像尺寸, 按列方向进行软件流水, 流水线中断次数更少.

4 算法实现

图像滤波库中维纳滤波, 高斯滤波, 拉普拉斯滤波等函数, 由于空间复杂度低, 可以直接使用算法切片方法进行优化. 对于中值滤波等空间复杂度高的函数, 需要分别进行算法改进. 本节以常用的中值滤波函数为例介绍如何使用算法切片方法, 并结合平台特性进行优化.

4.1 迭代中值算法

对于一个长度为 n, 元素互异的数组, 可以在空间复杂度为 O(1) 的限制内求出中值. 首先, 遍历数组求出最大值 m_1; 当第二次遍历数组时, 选出所有小于 m_1 的最大值 m_2; 以此类推, 当第 i+1 次遍历数组时, 在所有小于 m_i 的数中, 选择最大值 m_{i+1}. 按照该规则, 通过多次迭代可以求出中值.

上述算法仅适用于没有重复元素的数组, 不能直接应用到中值滤波函数的求解过程中. 可以通过一种映射规则, 将普通数组映射成元素互异的数组, 来解决该问题.

假设有一整数数组 H, 其元素个数为 $n(n\ge 2)$. 定义数组 G 为 $G_i=H_i\times n+i$(i 为元素下标, $0\le i\le n-1$), 则对于任意下标为 i、$j(0\le i<j\le n-1)$ 的两个元素, 存在以下两种情况:

1) 当 $H_i\le H_j$ 时, 易证 $G_i<G_j$;

2) 当 $H_i>H_j$ 时, $G_i>G_j$ 的证明如下:

$\because H_i>H_j$

$\therefore H_i\times n>H_j\times n$

$\because H_i, H_j$ 为整数且 $n\ge 2$

$\therefore H_i\times n\ge(H_j+1)\times n$

$\because G_i=H_i\times n+i\ge H_i\times n;\ G_j=H_j\times n+j<H_j\times n+n$

$\therefore G_i>G_j$

从上述证明可以看出, 数组 G 中的元素均不相同. 若要计算数组 H 的中值, 只需先求出无重复元素数组 G 的中值 med, 最后将 med 整除 n, 可以得到数组 H 的中值. 灰度值、RGB 图像单通道值均为无符号整数, 满足上述限制条件, 而且它们只占 1~2 个字节, 在变量分配空间足够的情况下, 不会产生溢出. 对滤波窗口应用迭代中值算法, 根据窗口位置即可计算出对应数组中的偏移量, 如算法 1 所示.

对迭代中值滤波算法进行切片, 可以将核心计算过程分成 $[(p\times q+1)/2]\times(p\times q+1)$ 个操作序列. 图 3 给出了迭代中值滤波各切片间的依赖关系, S_{ij} 表示第 i 轮遍历的第 j 个切片. S_{i0} 用于初始化临时矩阵, S_{i1}~S_{in} 分别读入窗口中的一个像素, 然后依次执行算法 1 中的

8~10 行对应的操作, 并对计算结果进行合并与保存.

算法1. 迭代中值滤波算法

输入: a window **W** of Image F

输出: med, median of window **W**

```
1.  S = p×q
2.  last = +∞
3.  for k = 1 to (S+1)/2 do
4.     med = 0
5.     offset = 0
6.     for i = 0 to p-1 do
7.        for j = 0 to q-1 do
8.           temp = W_{i,j}×S+offset
9.           if temp<last do
10.             med = max(med, temp)
11.          offset += 1
12.    last = med
13. med = last/S
```

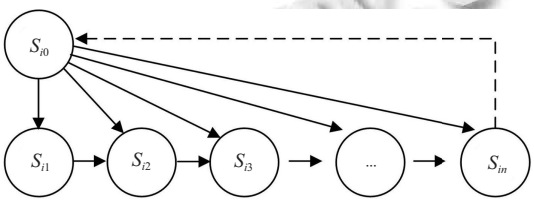

图 3 迭代中值滤波切片依赖关系

4.2 条件判断

直接使用 BWDSP 1042 中的条件跳转指令实现算法中的判断语句, 会因为分支预测失败导致流水线中断. 而且对于多簇处理器而言, 多个簇进行相同的条件判断, 得出的结果可能不一致, 从而导致各个簇的指令流不同.

为了减少因条件判断导致的流水线中断, 可以采用 BWDSP 1042 中特殊的硬件逻辑. 对于简单的数据选大选小问题 (如算法 1 第 10 行), 直接使用 max、min 指令来计算. 对于控制程序流向的条件判断 (如算法 1 第 9 行), 可以利用 BWDSP 1042 中 CPred 条件执行控制器通道来控制各分支指令的执行. 首先 CPred 通道获取条件判断的结果, 然后根据判定结果抑制无效分支中所有指令的执行. 通过使用这些硬件逻辑, 能够有效的减少流水线的中断次数, 使程序更加高效.

4.3 零开销循环

图像滤波处理至少需要四重循环, 内层循环的流水线中断对整体性能的影响较大. 如迭代中值滤波函数需要五重循环, 共计$[(p\times q+1)/2]\times(p\times q\times m\times n)$ 次循环迭代. 使用普通的 if 指令来判断循环是否结束, 每个循环体结尾处都会产生大量的空操作, 导致计算资源浪费.

现代高性能 DSP 普遍采用零开销循环技术来解决循环体结尾处的流水线停顿, BWDSP 1042 也不例外. BWDSP 1042 使用零开销循环寄存器来保存循环体的长度, 指令流水线在 FE2 级根据该专用寄存器的值, 提前判断并预取循环体开始指令行的代码, 使得循环在执行过程中不会产生流水线停顿.

5 实验结果与分析

根据第 3、4 节描述的优化方法, 在 BWDSP 1042 上对图像滤波库中的函数进行了测试. 实验平台为 BWDSP 1042 模拟器, 操作系统为 32 位 Windows 7, 集成开发环境是 ECS(Efficient coding studio) 2.0.0.1, 使用 BWCC 编译器.

本文首先对空间复杂度高的中值滤波算法进行测试, 以尺寸大小为 256×256 的 lena 图像作为测试用例. 分别选用大小为 3×3、5×5 的窗口进行中值滤波, 处理结果如图 4 所示. 经比较, 迭代中值滤波算法的计算结果与传统中值滤波算法的计算结果相同.

(a) 有椒盐噪声的 lena 图像　(b) 3×3 窗口迭代中值滤波结果　(c) 5×5 窗口迭代中值滤波结果

图 4 迭代中值滤波算法计算结果

表 1 列出了传统中值滤波算法和迭代中值滤波算法的计算耗时情况. 可以看出, 虽然在串行计算方式下, 迭代中值滤波算法所需要的指令周期数远远多于传统中值滤波算法, 但使用算法切片优化方法后, 迭代中值滤波算法的指令周期数更少, 这主要是因为迭代中值滤波的时间复杂度高于传统中值滤波算法, 而前者能够通过使用算法切片方法来加快计算速度. 还可以看出, 以滤波窗口为单位进行软件流水, 函数性能提升都不超过 5 倍, 主要是由于流水线频繁中断, 且流水线加速比很小. 使用算法切片方法细分迭代中值滤波函数处理单位, 使得超流水线的优势得到充分发挥, 函数性能较切片前进一步提升了 51 倍以上.

本文接着对图像滤波库中空间复杂度为 O(1) 的

函数进行了优化与测试. 由于不需要额外存储空间, 可以直接对这些函数进行切片. 在四簇流水模式下, 图像滤波库中部分核心函数切片前后的加速情况如图 5 所示 (输入图像的尺寸大小为 256×256). 所有函数的加速比均在 52 倍以上, 而且随着窗口的增大, 加速比也随之增大.

表 1 中值滤波函数测试结果 (单位: 万个指令周期)

算法实现	传统中值滤波算法		迭代中值滤波算法		
	串行	流水化	串行	切片前流水化	切片后流水化
3×3滤波窗口	29550	15692	35417	17492	337
5×5滤波窗口	95262	46847	203570	49626	762

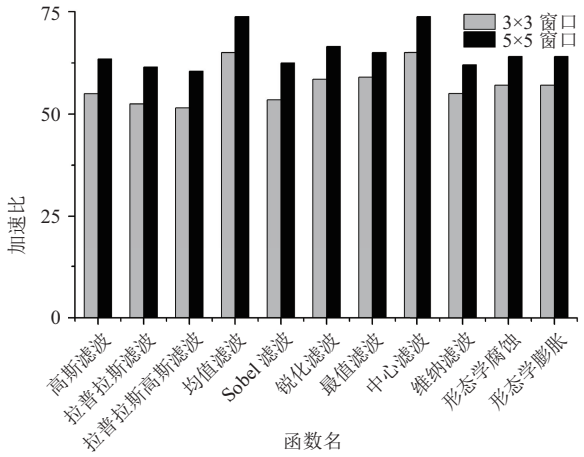

图 5 图像滤波函数加速比

6 结语

本文通过对复杂的图像滤波算法进行切片, 并按列方向多簇流水化, 让流水线重复执行大量相同的操作序列, 以减少流水线中断次数, 同时提高流水线加速比. 结合 BWDSP 1042 平台的特殊硬件逻辑, 对中值滤波等图像滤波处理函数进行了优化. 实验结果表明, 使用算法切片优化方法, 能够充分发挥现代高性能处理器的优势, 大幅度提升函数的处理效率. 下一步将对其他图像处理函数进行平台移植与优化, 以实现全面且高效的图像处理函数库.

参考文献

1 Lim JS. Two-Dimensional Signal and Image Processing. New Jersey: Prentice Hall, 1990.

2 Li YQ, Su GD. Design of high speed median filter based on neighborhood processor. 2015 6th IEEE International Conference on Software Engineering and Service Science (ICSESS). Beijing, China. 2015. 648–651.

3 金龙, 王洪元, 张继, 等. 实时 DSP 图像处理高斯滤波优化. 制造业自动化, 2014, 36(12): 63–66.

4 Badamchizadeh MA, Aghagolzadeh A. Comparative study of unsharp masking methods for image enhancement. Third International Conference on Image and Graphics (ICIG'04). Hong Kong, China. 2004. 27–30.

5 Gonzalez RC, Woods RE. Digital Image Processing. 3rd ed. New Jersey: Prentice-Hall, 2006: 166–213.

6 Maheshwari R, Rao SSSP, Poonacha PG. FPGA implementation of median filter. Proc. of the Tenth International Conference on VLSI Design, 1997. Hyderabad, India. 1997. 523–524.

7 Benkrid K, Crookes D, Benkrid A. Design and implementation of a novel algorithm for general purpose median filtering on FPGAs. IEEE International Symposium on Circuits and Systems, 2002. ISCAS 2002. Phoenix-Scottsdale, AZ, USA. 2002. 4. IV-425–IV-428.

8 雷涛, 周进, 吴钦章. DSP 实时图像处理软件优化方法研究. 计算机工程, 2012, 38(14): 177–180. [doi: 10.3969/j.issn.1000-3428.2012.14.053]

9 雷涛, 曹晓伟, 吴钦章. 实时 DSP 图像处理空间低通滤波模块优化. 光电工程, 2012, 39(5): 116–120.

10 黄德天, 陈建华. DSP 图像处理的程序优化. 中国光学与应用光学, 2009, 2(5): 452–459.

11 Eyre J, Bier J. The evolution of DSP processors. IEEE Signal Processing Magazine, 2000, 17(2): 43–51. [doi: 10.1109/79.826411]

12 洪一, 方体莲, 赵斌, 等. "魂芯一号"数字信号处理器及其应用. 中国科学: 信息科学, 2015, 45(4): 574–586.

13 甄扬, 顾乃杰, 叶鸿. 数字信号变换函数在多簇 VLIW DSP 上的优化. 计算机工程, 2016, 42(3): 47–52.

14 齐广玉, 张功萱. 超标量、超流水处理机的性能分析. 小型微型计算机系统, 1996, 17(9): 25–30.