文章编号:1674-2974(2025)08-0130-10 DOI:10.16339/j.cnki.hdxbzkb.2025290

超长指令字DSP上的多方向Sobel算法实现与优化

唐俊龙[1†],段美竹[1],时洋[2]
(1. 长沙理工大学 物理与电子科学学院,湖南 长沙 410011;
2. 国防科技大学 计算机学院,湖南 长沙 410003)

摘 要:VLIW(very long instruction word)架构的DSP在图像处理和计算机视觉等实时性应用场景得到广泛应用,高并行性的多方向Sobel算法是这些应用领域的重要算法之一,面向VLIW DSP实现和优化多方向Sobel算法具有重要意义.本文提出了基于VLIW的数据重排Im2col(image to column)加矩阵乘GEMM(general matrix multiplication)优化卷积计算的方法,并采用DMA(direct memory access)双缓冲机制实现数据传输与内核计算的并行,减少了等待数据传输的时间开销,使用该方法在FT-Matrix DSP上实现并优化了多方向Sobel算法.实验结果显示,优化后的算法相比于OpenCV图像库中算法,实现了4.96~8.76倍的加速;比TMS320C6678处理器提升了3.26~6.60倍.这些结果表明,采用VLIW架构的DSP在密集型数据处理方面具有显著优势,在VLIW DSP上实现与优化的图像检测算法具有广阔应用前景.

关键词:超长指令字(VLIW);多方向Sobel算法;数据重排;矩阵乘;DMA双缓冲
中图分类号:TP751 **文献标志码**:A

Implementation and Optimization of Multi-directional Sobel Algorithm on VLIW DSPs

TANG Junlong[1†], DUAN Meizhu[1], SHI Yang[2]
(1. School of Physics and Electronic Science, Changsha University of Science and Technology, Changsha 410011, China;
2. School of Computer Science, National University of Defense Technology, Changsha 410003, China)

Abstract: DSPs (digital signal processors) using VLIW (very long instruction word) architecture are widely used in high real-time application scenarios, such as image processing and computer vision. One of the important algorithms in these application areas is the highly parallel multi-directional Sobel algorithm. Implementing and optimizing this algorithm for VLIW DSPs is of great significance. In this paper, we propose a method of optimizing convolutional computation based on VLIW data rearrangement Im2col (image to column) plus matrix multiplication GEMM (general matrix multiplication), and use DMA (direct memory access) double buffer mechanism to realize the parallelism of data transmission and kernel computation, which reduces the time overhead of waiting for data transmission and the time overhead of kernel computation. The time overhead of waiting for data transmission is reduced, and the multi-directional Sobel algorithm is implemented and optimized on FT-Matrix DSP using this method. The experimental results show that the optimized algorithm achieves 4.96~8.76 times speedup compared

* **收稿日期**:2024-09-06
基金项目:湖南省自然科学基金资助项目(2023JJ40679), Natural Science Foundation of Hunan Province (2023JJ40679)
作者简介:唐俊龙(1973—),男,湖南武冈人,长沙理工大学教授,博士
† 通信联系人,E-mail: tangjl@csust.edu.cn

with the algorithm in OpenCV image library, and 3.26~6.60 times improvement compared with the TMS320C6678 processor. These results show that the DSP with VLIW architecture has significant advantages in intensive data processing, and the image detection algorithm implemented and optimized on VLIW DSP has a broad application prospect.

Key words: very long instruction word (VLIW); multi-directional Sobel algorithm; data rearrangement; general matrix multiplication (GEMM); DMA double buffering

Sobel算子是图像处理中的基础算法,主要用于边缘检测,传统边缘检测算子包括Roberts、Sobel、Canny、Prewitt和Laplacian算子[1-2].其中Sobel算法简单高效,对噪声具有较强的鲁棒性,故而应用广泛[3].它通过计算图像在水平和垂直方向上的梯度,来检测图像的边缘,揭示物体的轮廓和结构.多方向Sobel算法在原理上和传统Sobel算法是一样的,只是增加了对45°和135°两个方向的检测[4-5].然而,Sobel算子的计算过程涉及大量的卷积操作,对处理器的计算能力提出了较高的要求,采用高性能的处理器缩短计算时间,才能满足实时性的要求.

近年来,随着大规模集成电路和信息技术的发展,数字信号处理器(DSP)广泛应用于通信、音频处理、雷达系统和图像处理等领域[6-7].VLIW架构的DSP是一种高性能的并行处理器,VLIW DSP结合了VLIW架构和传统DSP的优点,通过在一个时钟周期内同时执行多个指令,显著提升计算效率和处理速度[8-9].这种架构不仅硬件结构简单,降低了设计和制造成本,减少了功耗,而且具备强大的运算能力,适用于对实时性要求高的应用场景[10].Sobel算法在图像领域应用广泛并且具有较高并行性,本文聚焦于Sobel在VLIW DSP上实现和优化,发挥架构优势,对于推动DSP和Sobel算法的发展具有重要意义.

目前,许多学者对于Sobel算法的优化取得了一定的成果.文献[11]在GPU(图形处理器)上通过将图像分割并行化Sobel算法,提高了处理速度.文献[12]提出了利用缓存优化和数据预取技术减少内存访问延迟,从而提高Sobel算子的执行效率.文献[13]基于FPGA(field programme gate array)提出一种多核矩阵处理器,其采用同构多核结构使数据并行传输和运算.文献[14]基于Zynq-7000系列平台,将ARM处理器与FPGA组合起来并利用异构多核平台实现加速.因为平台和处理器的体系结构差异,上述优化方法并不完全适用.平台架构相似的有文献[15]基于FT-M7002平台,通过将二维卷积核拆分为水平和垂直两个一维模版实现计算加速,但矩阵转置过程中,并行性较差,不能完全发挥架构特性.

本文依据VLIW架构的特性对多方向Sobel算法进行优化,在FT-Matrix DSP平台进行了实现.本文工作主要有以下几个方面:1)根据VLIW特性采用Im2col+GEMM方式优化卷积计算过程;2)使用DMA双缓冲机制,实现数据传输与数据计算的并行,通过向量化、循环展开等优化手段提高数据的并行性;3)针对FT-Matrix DSP的Sobel实例化,展示可行性;4)设计对比实验,测试优化算法的高效性.

1　多方向Sobel算法原理

1.1　传统Sobel算法

Sobel边缘检测算法是利用图像中像素点灰度值的梯度来检测边缘[16].Sobel算法的求梯度值原理是对图像像素矩阵做卷积.Sobel算法最常使用的是两个3×3的卷积核,分为横向和纵向模版,如图1(a)、(b)所示,分别对水平和垂直方向上的像素进行滑动检测,假设图像的部分像素矩阵如图2所示.Sobel算法检测的过程如下:

-1	0	1
-2	0	2
-1	0	1

(a)横向

1	2	1
0	0	0
-1	-2	-1

(b)纵向

图1　卷积核

Fig.1　Convolutional kernel

1)对像素矩阵做卷积,G_1和G_2分别代表横向和纵向边缘检测的图像灰度值,如式(1)和式(2)所示[17].

$$G_1 = (a_3 + 2 \times a_6 + a_9) - (a_1 + 2 \times a_4 + a_7) \quad (1)$$

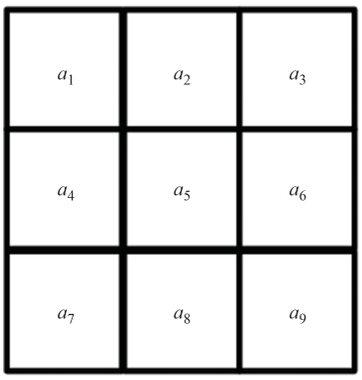

图2 像素矩阵

Fig.2 Pixel matrix

$$G_2 = (a_1 + 2 \times a_2 + a_3) - (a_7 + 2 \times a_8 + a_9) \quad (2)$$

2)通过式(1)与式(2)得到图像每一个像素的横向和纵向灰度值,经过式(3)计算出该点的梯度值G_{xy}.

$$G_{xy} = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y| \quad (3)$$

3)用式(3)得到的像素梯度值和设定的阈值进行比较,判断是否为边缘点,若超过阈值,即视为边缘点,反之,则不认为是边缘点[18-19].

1.2 多方向Sobel算法

传统Sobel算法只能对水平和垂直方向进行检测,对于其他方向的检测效率较低,改进的Sobel算法增加了45°和135°方向检测,模板如图3(a)、(b)所示.通过这两个卷积模板进行卷积得到的梯度结果记为G_3和G_4,见式(4)和式(5).

$$G_3 = (a_2 + 2 \times a_3 + a_6) - (a_4 + 2 \times a_7 + a_8) \quad (4)$$

$$G_4 = (a_2 + 2 \times a_1 + a_4) - (a_6 + 2 \times a_9 + a_8) \quad (5)$$

梯度值G_{xy}是将这四个方向梯度值的绝对值相加求得,如式(6)所示.最后将梯度值与阈值进行比较,判断是否为边缘点.

$$G_{xy} = |G_1| + |G_2| + |G_3| + |G_4| \quad (6)$$

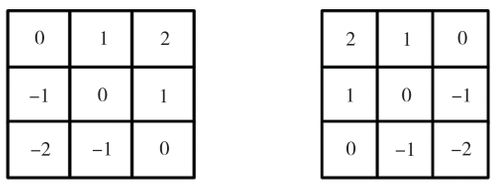

图3 增加的卷积核

Fig.3 Increased convolution kernel

2 算法优化方法

2.1 卷积计算优化

Sobel边缘检测的核心是卷积计算求梯度,也是占运行时间的主要部分,因此根据VLIW架构主要对梯度计算进行优化,并且使用SIMD(single instruction multiple data)指令级并行优化,能够明显减少访存次数,将卷积计算通过Im2col转换成GEMM,滑动窗口展开成矩阵,能连续读取内存块的数据,减少循环开销,提高吞吐量,提升计算效率[20-21],优化流程示意如图4所示.

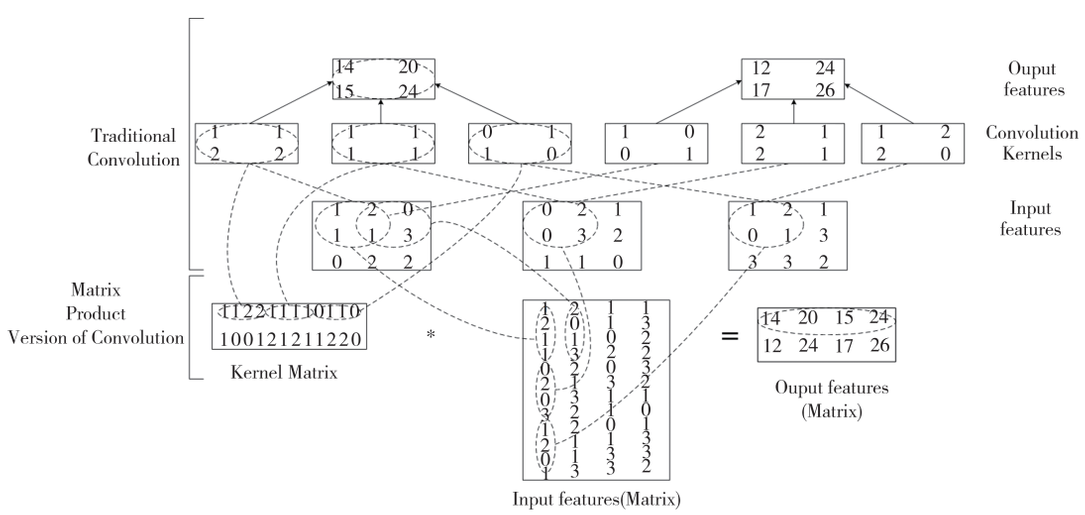

图4 卷积优化流程示意图

Fig.4 Schematic diagram of convolution optimization process

2.1.1 Im2col优化

Im2col过程如图4所示,将滑动窗口内的元素转换成一列,Im2col的结果矩阵高度为$H=K_h \times K_w$,矩阵的宽W由式(7)得出.

$$W = \left(\frac{H_i - K_h}{s} + 1\right) \times \left(\frac{W_i - K_w}{s} + 1\right) \quad (7)$$

式中:K_h是卷积核的高;K_w是卷积核的宽;H_i是输入矩阵的高;W_i是输入矩阵的宽;s是步长.通常Im2col过程是对滑动窗口内所有元素进行排列之后,再对下一个滑动窗口进行重排.由于VLIW结构,一个指令包包含若干标量和向量指令,标量指令通常作用于一个数据,容易造成大量的寄存器空闲.改变策略是先将每个滑动窗口的第一个元素排列,再对第二个元素排列,充分利用寄存器资源.

2.1.2　GEMM优化

传统的GEMM,其基本算法形式是$\boldsymbol{C}=\boldsymbol{A}\times\boldsymbol{B}$,矩阵$\boldsymbol{A}$的规模是$M\times N$,矩阵$\boldsymbol{B}$的规模是$N\times K$,则结果矩阵$\boldsymbol{C}$的规模$M\times K$,矩阵$\boldsymbol{C}$的元素$C_{ij}$的值如式(8)所示.

$$C_{ij}=\sum_{k=0}^{k-1}A_{ik}\times B_{kj} \tag{8}$$

计算某个C_{ij}元素的时候,需要对$\boldsymbol{A}$矩阵和$\boldsymbol{B}$矩阵进行内积运算,累加会对数据产生依赖,无法充分使用所有MAC(multiply accumulate)单元,因此,在VLIW DSP平台实现矩阵乘需要改变计算方式,充分发掘向量单元的并行性.本文采取的方法是将$\boldsymbol{B}$矩阵按列进行划分,每个VPE(vector processing element)中的每个MAC单元都处理不同的列数据,从而计算出$\boldsymbol{C}$矩阵中不同列的结果,这样的并行化的优势在于每个MAC单元处理数据之间不存在任何依赖,可以将所有MAC单元的效率最大化,如图5所示.由于$\boldsymbol{A}$矩阵是卷积核,数据规模不大,可以放置在标量存储单元SM(scalar memory)内,通过全局共享寄存器广播给所有VPE,可以减少占用AM(array memory)空间,同时也能避免VPE读取相同的$\boldsymbol{A}$矩阵数据.

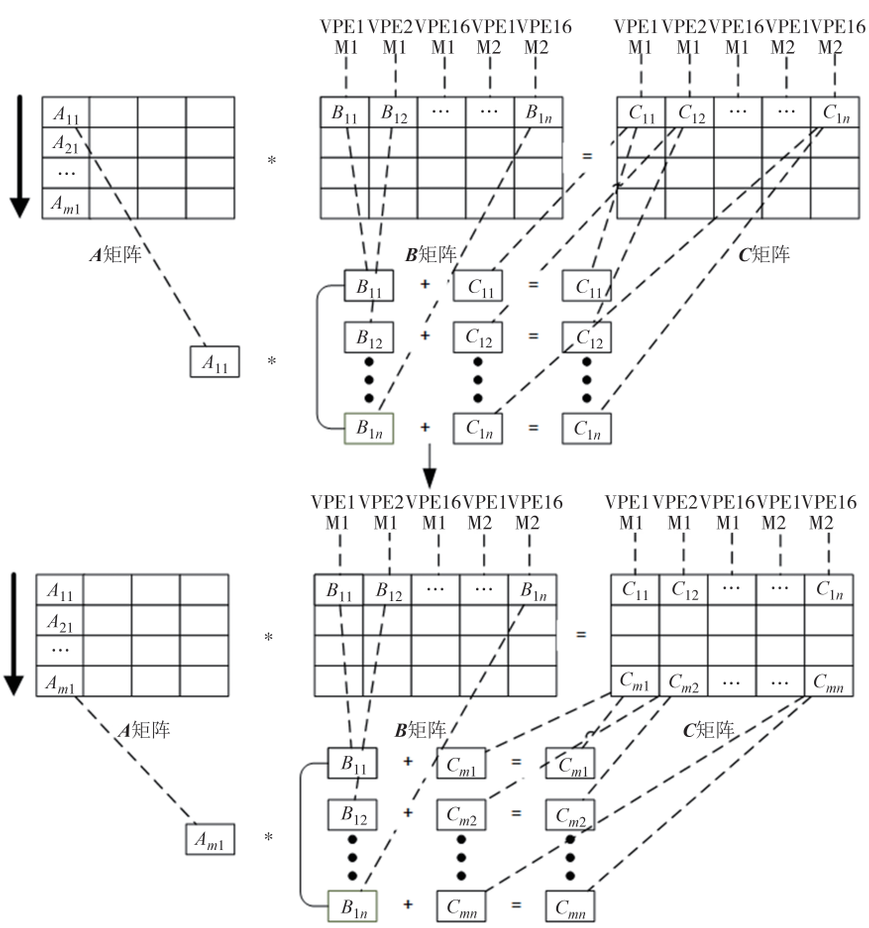

图5　矩阵乘优化化

Fig.5　Matrix multiplication optimization

2.2　DMA传输优化

通常AM空间一般只有几百千字节,图片的像素矩阵空间远大于AM空间,需要将像素矩阵存储在DDR(double data rate)中,通过DMA分块将数据传输到AM空间进行向量运算.DMA传输的时间会影响整个算法的执行效率,因此需要对DMA传输进行优化.如图6所示,采用DMA双缓冲机制,将AM空间划分为AM0,AM1,AM2,AM3四部分.

当数据从DDR中分块传入AM0中,在通过设置关键字判断传输完成之后开始计算,同时,下一块数

据从DDR传入AM2中,AM0中计算的结果存入AM1中.当AM0中计算结束,开始AM2中的数据计算,AM2的计算结果存储到AM3之中,同时DMA将AM1中的结果传出到DDR,并且DMA也会将下一块的数据传入AM0之中,当AM2计算结束之后,AM0开始计算,同时将AM3中的结果传出到DDR中,将下一块计算数据传入AM2中.通过这样的方式,牺牲一部分计算空间用来进行传输与计算并行,可以大幅度减少等待数据传输的时间,从而提高整个运算效率.

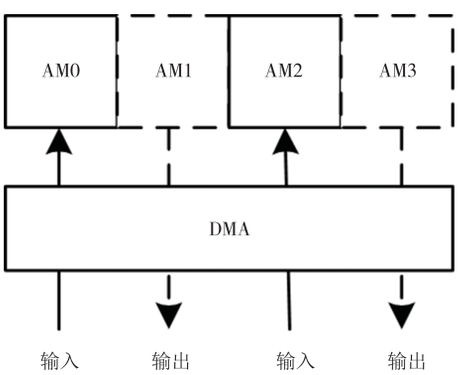

图6 DMA双缓冲机制

Fig.6 DMA double buffering mechanism

2.3 其他优化

在以上优化的基础上,为进一步提升优化效果,做了以下优化:1)对软流水进行优化,每个指令都有其对应的节拍数,需要尽可能地排布流水,使代码减少空转时间,提升计算性能;2)将核心代码中最内层的循环展开优化,通过增加迭代元素来减少循环次数,增加流水并行性;3)使用多核并行优化,通过设置合适的偏移量,多个内核分别计算多通道图片中的一个通道的数据,充分挖掘指令级的并行性以获得最好的加速效果.

3 基于FT-Matrix DSP算法优化实现

VLIW上的多方向Sobel算法实现优化在国防科技大学自主研发的一款FT-Matrix DSP上进行了实例化,并且与原始算法、同系列不同型号的FT-M7002以及TI的TMS320C6678算法进行对比.

3.1 FT-Matrix DSP的架构

FT-Matrix DSP是一款片上集成了一个CPU核、四个DSP核以及一个专用加速器核,单个核内具有32 kB一级数据缓存、512 kB向量存储空间AM,核外拥有62 GB超大容量DDR存储空间的高性能三级存储DSP芯片[22].FT-Matrix DSP继承了传统DSP体系结构的VLIW,包含标量处理单元SPU(scalar processing unit)以及向量处理单元VPU(vector processing unit),2个处理单元以紧耦合方式工作,如图7所示[23].SPU由SPE(scalar processing element)、SM、指令流控部件组成.SPE的功能是接受标量运算的指令,并在译码后执行操作;SM是标量访存寄存器,主要作用是标量数据的访存;VPU包括16个同构运算簇VPE以及混洗/归约部件[24].VPE内部集成4个运算部件,支持定点和浮点运算,可以同时对16路32位数据进行向量运算.因此,VPU是针对数据密集、计算量大的任务进行并行处理,以提高运算效率.直接访存部件DMA是数据传输的中枢,通过配置DMA传输参数,启动对包括AM、SM、GSM(global shared memory)和DDR的访问,实现核内与核外的数据交互[25].综上,FT-Matrix DSP基于VLIW的体系架构对于密集型数据处理具有优势.

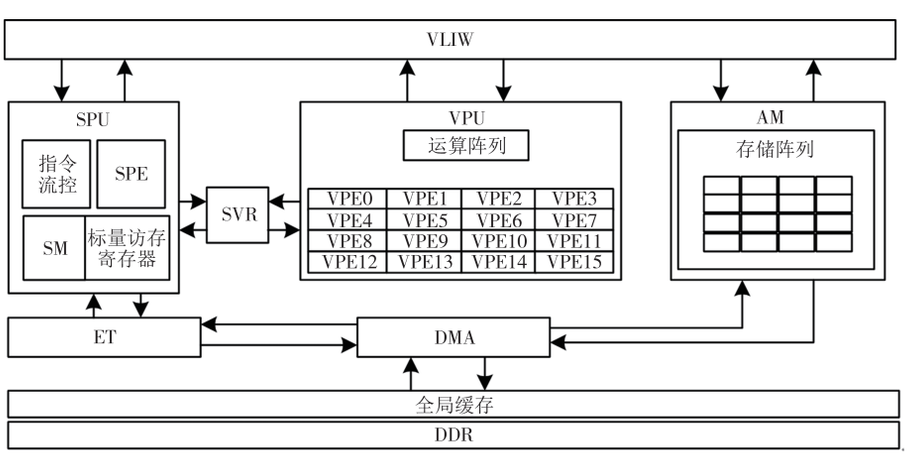

图7 FT-Matrix DSP体系结构内核架构图

Fig.7 Kernel architecture diagram of FT-Matrix DSP architecture

3.2　算法实现流程

Sobel算法在FT-Matrix DSP平台的实现流程如图8所示.首先将像素矩阵存入DDR,卷积核存入SM空间;其次,由于像素矩阵默认数据类型是unsigned char,FT-Matrix DSP向量操作是32位float型,因此需要对数据进行混洗、移位以及打包操作,将数据类型转换成32f;然后将像素矩阵分块从DDR传入AM空间进行Im2col操作,将卷积运算转换为矩阵相乘,完成转换操作,将数据传回DDR;接着进行通用矩阵乘GEMM计算;最后求取梯度幅值,进行阈值判断,得到边缘点.

3.3　FT-Matrix DSP上的Im2col

初始Im2col部分算法是将一个窗口完全重排之后,再对下一个窗口进行重排.如算法1所示,K_h、K_w分别表示卷积核的高和宽,O_h、O_w分别表示滑动窗口经过的高和宽,其数值通过式(9)、(10)计算得出,以决定每层循环的次数.

$$O_h = \frac{H_i - K_h}{s} + 1 \tag{9}$$

$$O_w = \frac{W_i - K_w}{s} + 1 \tag{10}$$

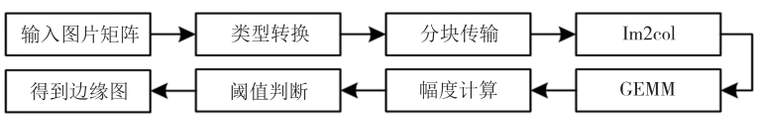

图8　算法实现流程图

Fig.8　Algorithm implementation flowchart

算法核心是先给偏移量or0赋值(算法1第5~9行),然后通过偏移量读取源地址指定的数据,再存储到目的地址(算法1第10~11行).FT-Matrix DSP一个指令包最多并行来自11个功能部件的11条指令,但读/存标量功能部件只有一个标量读/存SLDST(scalar load and store)功能部件,同一个功能单元的指令不能并行,并且对偏移量or0具有依赖性,一次循环只能完全一个数据的重排,没有使用向量指令的功能单元,极大地浪费了寄存器资源,导致不能够充分发挥架构的并行性,严重增加访存次数.

针对这个问题,需要将算法向量化,充分发挥VLIW架构的优势.一个指令包包含两个向量读/存功能单元VLDST0(vector load and store)和VLDST1,支持核内两条Load/Store指令的并行访存,一个VLDDW(vector load double word)指令表示读取的向量数据单位是双字(64位),一次可以最多支持64个32位数据的并行存取.以3×3的卷积核与10×640大小的像素矩阵、步长为1的卷积为例,具体实现的部分算法如算法2所示.核心循环是从源地址读取数据存入目的地址,每次循环对64个窗口的同一位置的元素进行重排(算法2第5~8行);某一位置元素完成重排之后,跳出循环,并且调整读取和存储的地址(算法2第11~14、17~20行).

3.4　FT-Matrix DSP上的GEMM

在FT-Matrix DSP上实现适用VLIW的优化版GEMM,以3×3的卷积核与10×640大小的像素矩阵、步长为1的卷积为例,具体实现的部分代码如算法3所示.核心循环是先从SM中读取卷积矩阵中的一个元素,广播到向量寄存器中(算法3第9~10行).然后两个VLDDW并行读取64个32位数据(算法3第11~12行).接着是4个MAC单元并行4条乘加操作(vector floating-point multiply signed add 32-bit),一次可

算法1　初始Im2col部分算法代码

```
1.    Loop_Kh:
2.    Loop_Kw:
3.    Loop_Oh:
4.    Loop_Ow:
5.    SADDU     oh,kh,r8
6.    SADDU     ow,kw,r9
7.    SMULIU    Wi,r8,r8
8.    SADDU     r8,r9,r9
9.    SMVAGA36  r10:r9,or0
10.   SLDW      *+ar1[or0],r11
11.   SSTW      r11,*ar2++[1]
12.   SADDU     1,ow,ow
13.   SEQ       ow,Ow,r1
14.   [! r1]  SBR       Loop_Ow
15.   SADDU     1,oh,oh
16.   SEQ       oh,Oh,r1
17.   SMOVIL    0,ow
18.   [! r1]  SBR       Loop_Oh
19.   SADDU     1,kw,kw
20.   SEQ       kw,Kw,r1
21.   SMOVIL    0,oh
22.   [! r1]  SBR       Loop_Kw
23.   SADDU     1,kh,kh
24.   SEQ       kh,Kh,r1
25.   SMOVIL    0,kw
26.   [! r1]  SBR       Loop_Kh
```

算法2 Im2col并行算法的部分代码

```
1.      SMOVIL      8,r1
2.  Loop1_1:
3.      SMOVIL      10,r2
4.  Loop1_2:
5.      VLDDW       *ar0++[or0],vr1:vr0
6.      VLDDW       *ar1++[or0],vr3:vr2
7.    VSTDW       vr1:vr0,*ar4++[or0]
8.    VSTDW       vr3:vr2,*ar5++[or0]
9.  [r2]  SSUBU     1,r2,r2
10. [r2]  SBR       Loop1_2
11.     SSUBA       H1:L1,ar0,ar0
12.     SSUBA       H1:L1,ar1,ar1
13.     SSUBA       H2:L2,ar4,ar4
14.     SSUBA       H2:L2,ar5,ar5
15. [r1]  SSUBU     1,r1,r1
16. [r1]  SBR       Loop1_1
17.     SSUBA       H3:L3,ar0,ar0
18.     SADDA       H4:L4,ar0,ar1
19.     SADDA       H4:L4,ar1,ar2
20.     SADDA       H4:L4,ar2,ar3
```

算法3 矩阵乘并行算法的部分代码

```
1.      SMOVIL      80,r2
2.  Loop1_1:
3.      SMOVIL      9,r1
4.      VMOVIL      0,vr6
5.      VMOVIL      0,vr7
6.      VMOVIL      0,vr8
7.      VMOVIL      0,vr9
8.  Loop1_2:
9.      SLDW        *src1++[1],r10
10.     SVBCAST     r10,vr10
11.     VLDDW       *src2++[or0],vr1:vr0
12.     VLDDW       *src3++[or0],vr3:vr2
13.   VFMULAS32   vr0,vr10,vr6,vr6
14.   VFMULAS32   vr1,vr10,vr7,vr7
15.   VFMULAS32   vr2,vr10,vr8,vr8
16.   VFMULAS32   vr3,vr10,vr9,vr9
17. [r1]  SSUBU     1,r1,r1
18. [r1]  SBR       Loop1_2
19.   VSTDW       vr7:vr6,*dst4++[or1]
20.   VSTDW       vr9:vr8,*dst5++[or1]
21.     SSUBA       H1:L1,src1,src1
22.     SSUBA       H2:L2,src2,src2
23.     SSUBA       H2:L2,src3,src3
24. [r2]  SSUBU     1,r2,r2
25. [r2]  SBR       Loop1_1
```

以完成64个数计算(算法3第13~16行).每跳出一次内层循环相当于做完64次卷积(算法3第17~18行).然后将矩阵乘结果双字VSTDW(vector store double word)存储到目的地址(算法3第19~20行).最后调整读取和存储的寄存器地址(算法3第21~23行).完成所有的计算之后,使用DMA双缓冲将AM空间的结果传回DDR的结果地址.

4 算法测试分析

4.1 实验环境

实验平台是基于VLIW架构的FT-Matrix DSP,使用的卷积核大小为3×3、5×5、7×7,图片大小为360像素×640像素、480像素×640像素、500像素×800像素、768像素×1 024像素,进行优化算法的性能测试,然后与优化前对比,与FT-M7002、TMS320C6678平台的优化算法对比,评估优化算法的表现.

4.2 正确性分析

程序性能优化的根本性原则是正确性,不能一味地追求代码效率,导致正确性不能得到保证.通过在FT-Matrix DSP平台上对不同尺寸的图片和卷积核运行优化算法,将结果矩阵与OpenCV库中Sobel算法的结果矩阵进行对比,误差在小数点后四位以上,认为其正确,经过验证,该优化算法在所有测试条件下均能得到正确的结果.

4.3 性能分析

用三种尺寸大小的卷积核、四种不同尺寸的三通道输入图片,对比分析不同平台、不同优化算法的优化效果.

4.3.1 算法优化前后性能分析

表1所示为优化算法与OpenCV库中算法在不同大小卷积核与不同尺寸的图片像素矩阵上的运行周期以及加速比,图9是优化后对比优化前加速效果的柱状图.横坐标是卷积核大小,纵坐标是加速比倍数.从表1可以看出:对于不同大小的卷积核与不同尺寸的输入矩阵,该优化算法可以取得4.96~8.76倍的加速效果,对于同一大小的卷积核,加速比趋近相同,对于不同大小的卷积核,随着卷积核的增加,加速比下降.这是由于Im2col的过程通常以像素矩阵的宽为最小尺度,然后尽量增加行数,然而AM空间有限,不同大小的卷积核与图片尺寸对AM空间的利用率不同,趋势是卷积核和图片尺寸越大,利用率越低,加速效果越差.其中5×5卷积核与500×800像素矩阵时,出现了较大差异,这是由于当卷积核为5×5、像素矩阵为500×800时,AM空间利用率相对于其

他几个尺寸最低,导致访存次数增加,从而增加了整个Im2col的时间,降低了算法性能.

表1　算法优化前后对比

Tab.1　Comparison before and after algorithm optimization

卷积核大小	图片大小/(像素×像素)	运行时间/ns		加速比/倍
		优化前	优化后	
3×3	360×640	18 433 828	2 137 120	8.62
	480×640	24 610 004	2 808 124	8.76
	500×800	31 856 316	3 649 360	8.72
	768×1 024	62 668 168	7 213 200	8.68
5×5	360×640	27 385 114	3 742 639	7.31
	480×640	36 494 627	4 970 379	7.34
	500×800	48 226 313	9 703 850	4.96
	768×1 024	94 232 948	12 548 030	7.50
7×7	360×640	39 998 071	7 720 478	5.18
	480×640	52 165 627	10 349 000	5.04
	500×800	68 478 785	13 163 621	5.20
	768×1 024	138 360 266	25 550 284	5.41

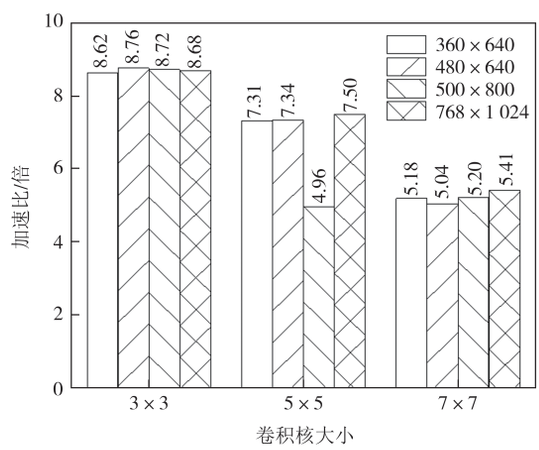

图9　算法优化后较优化前性能加速比柱状图

Fig.9　Speedup ratio after optimization over pre-optimization

4.3.2　与FT-M7002优化算法对比分析

在FT-Matrix DSP平台运行本文优化算法与范明亮等人[15]提出的基于FT-M7002的拆分二维卷积模板为一维卷积的优化算法,将两种优化算法进行比较,使用平台的性能统计记录运行节拍,测试结果如表2所示,加速效果如图10所示.本文的优化算法取得了1.67~4.77倍的加速比.本文优化算法原理上的优势在于:一是拆分法在计算垂直分量的梯度时需要对像素矩阵进行转置,转置过程并行性较差,耗时较长;二是矩阵乘计算的全局并行性更高,拆分法逐行计算,局部并行,局部之外具有数据依赖性,并且数据也有重复访存.

4.3.3　TI平台优化前后对比分析

在TMS320C6678平台中分别运行OpenCV库中

表2　与FT-M7002优化算法性能比较

Tab.2　Performance comparison with FT-M7002 optimization algorithm

卷积核大小	图片尺寸/(像素×像素)	运行时间/ns		加速比/倍
		优化前	优化后	
3×3	360×640	10 196 419	21 371 20	4.77
	480×640	13 059 288	28 081 24	4.65
	500×800	17 069 228	36 493 60	4.67
	768×1 024	33 664 377	72 132 00	4.66
5×5	360×640	11 010 777	37 426 39	2.94
	480×640	14 063 457	49 703 79	2.83
	500×800	18 333 202	97 038 50	1.89
	768×1 024	34 963 881	125 480 30	2.78
7×7	360×640	12 861 589	772 047 8	1.67
	480×640	17 319 618	103 490 00	1.67
	500×800	22 406 557	131 636 21	1.70
	768×1 024	43 087 868	255 502 84	1.68

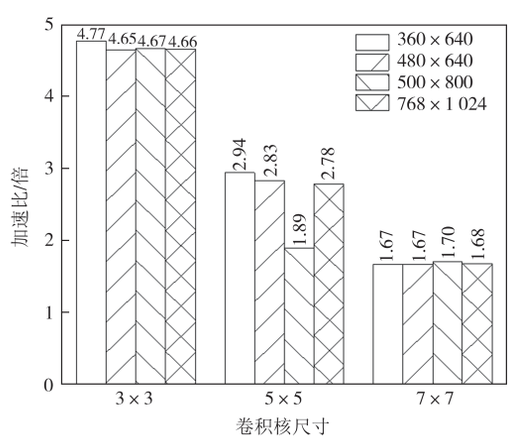

图10　相较FT-M7002优化算法加速比

Fig.10　Speedup ratio compared with FT-M7002 optimization algorithm

Sobel算法和优化算法,使用TI平台的计时函数记录运行时间.实验结果如表3所示,加速效果如图11所示.相较于TI平台原算法,优化后的算法取得1.18~1.60倍的加速比.5×5卷积核优化效果比3×3的好,这是因为优化算法针对梯度计算,其他固定开销优化较少,而7×7大小的优化效果较差是因为在Im2col过程中核越大,耗时越长.

4.3.4　与TI平台对比分析

分别在TMS320C6678和FT-Matrix DSP平台中运行本文优化算法,使用TI平台的计时函数、FT-Matrix DSP平台的性能统计记录运行时间.实验结果如表4所示,加速效果如图12所示.相较于TI平台,本文优化算法在FT-Matrix DSP平台取得3.26~6.60倍的加速比,加速效果明显,多核对比单核,双缓冲传输以及指令级优化具有优势.

表3 TI平台算法优化前后性能比较

Tab.3 Performance comparison before and after algorithm optimization on TI platform

卷积核大小	图片尺寸/(像素×像素)	运行时间/ns		加速比/倍
		优化前	优化后	
3×3	360×640	19 001 771	13 854 121	1.37
	480×640	25 220 489	18 508 003	1.36
	500×800	32 563 627	24 116 032	1.35
	768×1 024	64 263 358	47 541 893	1.35
5×5	360×640	28 985 746	18 247 642	1.59
	480×640	38 481 904	24 377 747	1.58
	500×800	50 578 562	31 678 414	1.59
	768×1 024	100 217 764	62 741 642	1.60
7×7	360×640	39 289 810	33 080 712	1.19
	480×640	52 049 175	44 147 602	1.18
	500×800	68 095 177	57 725 434	1.18
	768×1 024	138 070 335	114 144 275	1.20

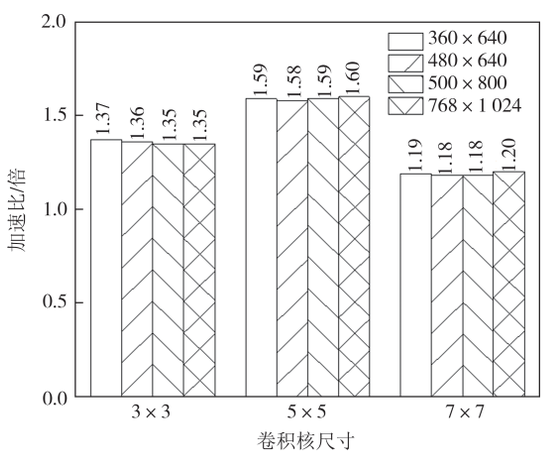

图11 TI平台算法优化后加速比

Fig.11 Speedup ratio after algorithm optimization on TI platform

表4 与TI平台性能比较

Tab.4 Comparison of algorithm performance with TI platform

卷积核大小	图片尺寸/(像素×像素)	运行时间/ns		加速比/倍
		优化前	优化后	
3×3	360×640	13 854 121	2 137 120	6.48
	480×640	18 508 003	2 808 124	6.59
	500×800	24 116 032	3 649 360	6.60
	768×1 024	47 541 893	7 213 200	6.59
5×5	360×640	18 247 642	3 742 639	4.88
	480×640	24 377 747	4 970 379	4.90
	500×800	31 678 414	9 703 850	3.26
	768×1 024	62 741 642	12 548 030	5.00
7×7	360×640	33 080 712	7 720 478	4.28
	480×640	44 147 602	10 349 000	4.27
	500×800	57 725 434	13 163 621	4.38
	768×1 024	114 144 275	25 550 284	4.46

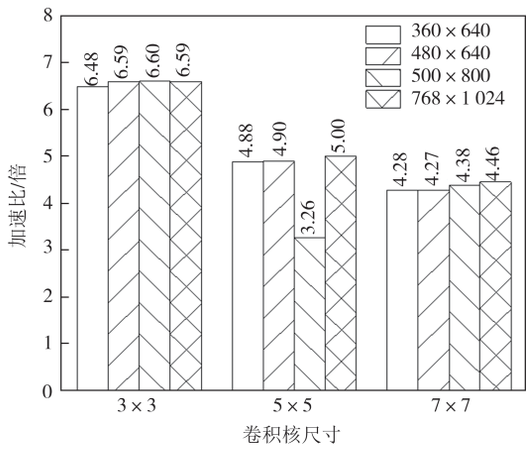

图12 相较TI平台加速比

Fig.12 Speedup ratio compared with TI platform

5 结论

本文提出一种针对VLIW架构的多方向Sobel算法优化方法,在梯度并行计算过程中,充分利用VLIW DSP架构的特点,使用SIMD内嵌指令向量化、多核并行化以及流水排布针对指令级并行优化,采用Im2col+GEMM方式优化核心计算,以提高计算的并行性;使用双缓冲机制优化数据访存,并基于FT-Matrix DSP平台成功设计与实现,进行了实例化.实验结果表明,与库中算法相比,本优化算法整体运行速度提升了4.96~8.76倍;相较面向FT-M7002平台的优化算法取得了1.67~4.77倍的加速效果;在TMS320C6678处理器上运行优化算法取得了1.18~1.60倍加速效果;相较于TMS320C6678,本文优化算法在FT-Matrix DSP平台取得3.26~6.60倍的加速效果.但是,随着卷积核的增大,AM空间利用率下降,加速效果降低,因此后续将优化AM空间的分配,提升运行速度,并实现更多的图像算法在VLIW DSP平台上的优化.

参考文献

[1] YANG S H, HSIAO S J. 266/VVC fast intra prediction using Sobel edge features[J]. Electronics Letters, 2021, 57(1): 11-13.

[2] AHMED A S. Comparative study among sobel, Prewitt and Canny edge detection operators used in image processing[J]. Journal of Theoretical and Applied Information Technology, 2018, 96(19): 6517-6525.

[3] LI L, WANG S Q, ZENG W J, et al. Research on zero watermarking technique based on improved Sobel operator[C]// 2021 IEEE International Conference on Artificial Intelligence and

Industrial Design (AIID). Guangzhou, China. IEEE, 2021: 594-597.

[4] TANG X L, WANG X G, HOU J, et al. An improved sobel face gray image edge detection algorithm [C]//2020 39th Chinese Control Conference (CCC). Shenyang, China. IEEE, 2020: 6639-6643.

[5] SHI L, ZHAO Y F. Edge detection of high-resolution remote sensing image based on multi-directional improved sobel operator [J]. IEEE Access, 2023, 11: 135979-135993.

[6] 赵一霈.DSP的发展与应用[J].电子技术与软件工程,2018(6):92.
ZHAO Y P. Development and application of DSP[J]. Electronic Technology & Software Engineering,2018(6):92.(in Chinese)

[7] 周逢道,韩思雨,綦振伟,等. 基于FPGA+DSP的浅地表频域电磁探测数字处理系统[J]. 湖南大学学报(自然科学版),2016,43(10):94-101.
ZHOU F D, HAN S Y, QI Z W, et al. Digital processing system for shallow surface frequency-domain electromagnetic detection based on FPGA+DSP[J]. Journal of Hunan University (Natural Sciences),2016,43(10):94-101.(in Chinese)

[8] SESHAN N. High VelociTI processing Texas instruments VLIW DSP architecture[J]. IEEE Signal Processing Magazine, 1998, 15(2): 86-101.

[9] MUTO V, ANDREOZZI E, CAPPELLI C, et al. Real-time implementation of a frequency shifter for enhancement of heart sounds perception on VLIW DSP platform[J]. Electronics,2023, 12(20): 4359.

[10] 张帆,葛颖增,窦勇. 超长指令字DSP上的数字图像处理算法优化方法[J]. 微计算机应用,2008(10):1-6.
ZHANG F, GE Y Z, DOU Y. Optimization methods of digital image process algorithm on VLIW DSP [J]. Microcomputer Applications,2008(10):1-6.(in Chinese)

[11] CHANG Q, LI X, LI Y, et al. Multi-directional sobel operator kernel on GPUs[J].Journal of Parallel and Distributed Computing, 2023,177:160-170

[12] ZHANG H, HAN L, XIE J M, et al. Realization and optimization of Sobel edge detection algorithm for Domestic DCU accelerators [C]//MEMAT 2022; 2nd International Conference on Mechanical Engineering, Intelligent Manufacturing and Automation Technology. Guilin, China. VDE, 2022: 1-7

[13] 陈凯,张涛. 多核DSP并行软件设计技术研究[J]. 信息技术,2015,39(7):5-8.
CHEN K, ZHANG T. Research on concurrent software design based on multicore DSP [J]. Information Technology, 2015, 39(7): 5-8.(in Chinese)

[14] 王威.基于ARM-FPGA异构多核平台上图像处理算法的加速研究[D].西安:西安电子科技大学,2019.
WANG W. Research on acceleration of image processing algorithms based on ARM-FPGA heterogeneous multi-core platform [D]. Xi'an: Xidian University, 2019.(in Chinese)

[15] 范明亮,郭子涵,柴晓楠,等. 面向FT-M7002的Sobel边缘检测算法优化实现[J]. 计算机工程,2022,48(6):193-199.
FAN M L, GUO Z H, CHAI X N, et al. Optimized realization of sobel edge detection algorithm for FT-M7002 [J]. Computer Engineering, 2022,48(6):193-199.(in Chinese)

[16] HADI SAPUTRA V, HERWINDIATI D E, SUTRISNO T. Car shape clustering using sobel edge detection with divisive average linkage and single linkage algorithm (case: bus, Sedan, citycar, mpv, and truck)[J]. IOP Conference Series: Materials Science and Engineering,2020,1007(1):012136.

[17] 李丹阳. 基于Soble算子的图像边缘检测优化设计[J]. 数字技术与应用,2017,35(11):137-138.
LI D Y. A optimization design of edge detection based on soble algorithm[J]. Digital Technology and Application,2017,35(11): 137-138.(in Chinese)

[18] TIAN R, SUN G L, LIU X C, et al. Sobel edge detection based on weighted nuclear norm minimization image denoising [J]. Electronics,2021,10(6):655.

[19] RAVIVARMA G, GAVASKAR K, MALATHI D, et al. Implementation of Sobel operator based image edge detection on FPGA[J]. Materials Today:Proceedings,2021,45:2401-2407.

[20] 王庆林,裴向东,廖林玉,等. 多核数字信号处理器矩阵乘卷积算法性能评测[J]. 国防科技大学学报,2023,45(1):86-94.
WANG Q L, PEI X D, LIAO L Y, et al. Evaluating matrix multiplication-based convolution algorithm on multi-core digital signal processors[J]. Journal of National University of Defense Technology,2023,45(1):86-94.(in Chinese)

[21] 刘杰, 迟利华, 谢林川, 等. 矩阵乘在通用DSP上的峰值性能模型[J]. 湖南大学学报(自然科学版), 2013, 40(Z1): 148-152.
LIU J, CHI L H, XIE L C, et al. Peak performance modeling of matrix multiplication on general-purpose DSPs [J]. Journal of Hunan University (Natural Science Edition), 2013, 40(Z1): 148-152.(in Chinese)

[22] 孙广辉.基于FT-M7002的OpenCV移植与优化[D].西安:西安电子科技大学,2019.
SUN G H. OpenCV transplantation and optimization based on FT-M7002[D]. Xi'an: Xidian University, 2019.(in Chinese)

[23] 郭恒亮, 柴晓楠, 韩林, 等. Canny边缘检测算法在飞腾平台上的实现与优化[J]. 计算机工程,2021,47(7):37-43.
GUO H L, CHAI X N, HAN L, et al. Implementation and optimization of canny edge detection algorithm on FT platform[J]. Computer Engineering,2021,47(7):37-43.(in Chinese)

[24] 王梦园.面向飞腾平台图像滤波算法的实现与优化[D].郑州:郑州大学,2021.
WANG M Y. Implementation and optimization of image filtering algorithm for Phytium Platform [D]. Zhengzhou: Zhengzhou University, 2021.(in Chinese)

[25] 郭盼盼,陈梦雪,梁祖达,等. 面向FT-M7002平台点积算法的优化实现[J]. 计算机工程与科学,2022,44(11):1909-1917.
GUO P P, CHEN M X, LIANG Z D, et al. Optimization of dot product algorithms on FT-M7002 [J]. Computer Engineering & Science,2022,44(11):1909-1917.(in Chinese)