计算机科学
COMPUTER SCIENCE

基于FT-M7002的复数域行向量矩阵乘法移植与优化

莫尚丰, 周振芬, 胡勇华, 徐敏敏, 毛春献, 袁钰迪

引用本文

莫尚丰, 周振芬, 胡勇华, 徐敏敏, 毛春献, 袁钰迪. 基于FT-M7002的复数域行向量矩阵乘法移植与优化[J]. 计算机科学, 2023, 50(11A): 220900277-6.

MO Shangfeng, ZHOU Zhenfen, HU Yonghua, XU Minmin, MAO Chunxian, YUAN Yudi. Transplantation and Optimization of Row-vector-matrix Multiplication in Complex Domain Based on FT-M7002 [J]. Computer Science, 2023, 50(11A): 220900277-6.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.220900277

基于 FT-M7002 的复数域行向量矩阵乘法移植与优化

莫尚丰　周振芬　胡勇华　徐敏敏　毛春献　袁钰迪
1 湖南科技大学计算机科学与工程学院　湖南 湘潭 411201
2 服务计算与软件服务新技术湖南省重点实验室　湖南 湘潭 411201

摘　要　FT-M7002 是我国自主研发的高性能 DSP,具有强大的向量处理能力。为有效地发挥它的性能优势,亟待优化移植面向 FT-M7002 的高效 VSIP 函数库。复数域行向量矩阵乘法是 VSIP 库中频繁使用的算法,在数字通信、图像处理等应用领域中大量使用。文中在 FT-M7002 DSP 上研究优化复数域行向量矩阵乘法算法,通过改变计算矩阵列向量为计算矩阵行向量、向量化、循环展开和软件流水等手段提升算法性能。测试结果表明:优化后的向量 ***C*** 算法相比 VSIP 库函数获得了 6.2～20.6 的加速比,汇编优化算法相比向量 ***C*** 算法获得了 3.4～14.3 的加速比,加速效果明显。

关键词:矩阵乘法;数字信号处理器;单指令多数据流;VSIPL
中图法分类号　TP313

Transplantation and Optimization of Row-vector-matrix Multiplication in Complex Domain Based on FT-M7002

MO Shangfeng,ZHOU Zhenfen,HU Yonghua,XU Minmin,MAO Chunxian and YUAN Yudi
School of Computer Science and Engineering,Hunan University of Science and Technology,Xiangtan,Hunan 411201,China
China Hunan Key Laboratory for Service computing and Novel Software Technology,Xiangtan,Hunan 411201,China

Abstract　FT-M7002 is a high-performance DSP independently developed in China,with powerful vector processing capability. In order to give full play to its performance advantages,it is urgent to optimize and transplant the efficient VSIP function library for FT-M7002. Row vector matrix multiplication in complex domain is a frequent algorithm used in VSIP library,which is widely used in digital communication,image processing and other application fields. In this paper,we study the optimization algorithm of row vector matrix multiplication in complex domain on FT-M7002 DSP,and improve the performance of the algorithm by changing the column vector of the computation matrix to the row vector of the computation matrix,vectorization,loop expansion and software pipelining. The test results show that the optimized vector C algorithm achieves a speedup ratio of 6.2～20.6 compared with the VSIP library function,and the assembly optimization algorithm achieves a speedup ratio of 3.4～14.3 compared with the vector C algorithm. The speedup effect is obvious.

Keywords　Matrix multiplication,Digital signal processor,SIMD,VSIPL

矩阵乘法是数字信号处理领域中的基本操作,被广泛应用于各种计算中,例如过程控制、实时图像处理、深度学习及实时数字信号处理等都用到了大规模的矩阵乘法运算,其计算性能直接影响到系统的整体性能。为了提高矩阵乘法运算的执行效率,大量研究人员对其进行了广泛的研究。文献[1]利用 MPI(Message Passing Interface)技术在多处理器平台上研究了矩阵乘法计算,解决了计算节点矩阵乘法操作负载均衡问题。文献[2]研究了通用 CPU 平台上利用 SIMD(Single Instruction Multiple Data)指令加速矩阵乘法计算,有效地提升了通用 CPU(Central Processing Unit)平台的矩阵乘法性能。文献[3-4]在英伟达 GPU(Graphic Processing Unit)平台基于 CUDA(Compute Unified Device Architecture)并行技术加速矩阵乘法计算,实现了在通用计算平台利用 GPU 加速矩阵计算。文献[5-6]基于 FPGA(Field Programmable Gate Array)实现了利用硬件提升矩阵乘法的计算性能,设计了基于 FPGA 的矩阵乘法运算器,提升了矩阵运算性能。

面对国产向量处理器的硬件架构,已有的矩阵乘法优化方法并不能充分利用新平台 DSP(Digital Signal Processor)微处理器的 SIMD(Single Instruction Multiple Data)的硬件资源,需要研究进一步的优化方法。在 SIMD 体系结构的 DSP 中,由于多个并行单元共享一套取指、译码、派发逻辑指令,控制结构简单,可以有效降低硬件代价,能够在较低的功耗下获得较好的性能。近年来,国内的高性能 DSP 也取得了较大的成就,例如飞腾、龙芯、申威等多个系列处理器。飞腾 FT-M7002 是国防科技大学微电子所自主研发的高性能 SIMD DSP,其片上集成 1 个 RISC CPU 核和 2 个 FT-MT2 DSP 核。FT-MT2 DSP 核芯在 1GHz 运行时,双精度浮点数的峰值性能高达 100 GFlops,单精度浮点数的峰值性能高达

基金项目:湖南省教育厅科研项目(20B242);湖南省自然科学基金(2017JJ3087)
This work was supported by the Research Projects of Hunan Provincial Department of Education(20B242) and Natural Science Foundation of Hunan Province,China(2017JJ3087).
通信作者:莫尚丰(mosfxy@foxmail.com)

200GFlops[7]。硬件不断进步在带来性能提升的同时，也对上层软件库提出了更高的要求。因此，根据硬件特点优化相应的算法软件，充分发挥出国产高性能 DSP 的硬件架构优势具有重要意义。

本文将结合 FT-M7002 处理器的硬件资源，研究 DSP 单核复数域行向量矩阵乘法算法的优化方法。

1 行向量矩阵乘法的基本原理

输入数据为行向量 $\boldsymbol{A}$ 和矩阵 $\boldsymbol{B}$，进行矩阵乘法得到结果行向量 $\boldsymbol{C}$。若 $\boldsymbol{A}$ 是一个数量为 n 的行向量，$\boldsymbol{B}$ 是一个 $n\times m$ 的矩阵，则 $\boldsymbol{C}$ 是一个数量为 m 的行向量。计算公式为：

$$\boldsymbol{A}=\begin{bmatrix}a_0 & a_1 & \cdots & a_{n-1}\end{bmatrix}$$

$$\boldsymbol{B}=\begin{bmatrix}b_{00} & b_{01} & \cdots & b_{0(m-1)}\\ b_{10} & b_{11} & \cdots & b_{1(m-1)}\\ \vdots & \vdots & \ddots & \vdots\\ b_{(n-1)0} & b_{(n-1)1} & \cdots & b_{(n-1)(m-1)}\end{bmatrix}$$

$$\boldsymbol{C}=\boldsymbol{AB}=\begin{bmatrix}\sum_{k=0}^{n-1}a_kb_{k0} & \sum_{k=0}^{n-1}a_kb_{k1} & \cdots & \sum_{k=0}^{n-1}a_kb_{k(m-1)}\end{bmatrix} \tag{1}$$

2 FT-M7002 处理器介绍

FT-M7002 是一款 40 nm 工艺的高性能 SIMD DSP，芯片主频为 1 GHz。单个 DSP 核内拥有 32 kB 的标量存储空间(Scalar Memory，SM)和 512 kB 的向量存储空间(Vector Memory，VM；又叫 Array Memory，AM)，所有的计算核共享 2 MB 的全局共享 Cache，核外拥有 32 GB 的大容量 DDR 存储空间。它具有指令集丰富、VM 容量大、向量运算的宽度大以及数据传输高效等特点，在计算密集型的矩阵算法中能发挥出较大的优势。本文的相关研究基于单个 DSP 核。

FT-M7002 的 DSP 核基于 VLIW(Very Long Instruction Word)、SIMD 技术和标量、向量单元并行结构。DSP 核包含一个五流出的标量处理器单元(Scalar Process Unit，SPU)和一个六流出的向量处理单元(Vector Process Unit，VPU)，两个处理单元以紧耦合的方式工作，具体结构如图 1 所示。

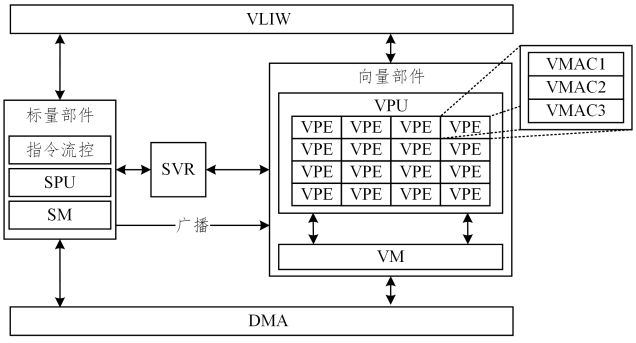

图 1 FT-M7002 DSP 核结构

Fig. 1 FT-M7002 DSP core structure

SPU 只包含一个处理单元，主要负责串行任务处理和程序控制。VPU 由 16 个向量计算引擎(Vector Process Engine，VPE)构成，最多支持 16 路 32 位数据进行向量运算，主要面向密集型的计算提供并行处理。DMA(Direct Memory Access)为内核提供了高速数据传输通路，实现核外 DDR 与 SM 和 VM 的快速数据交换。VM 是 VPU 独享的数据存储器，在访存不冲突时，可以同时支持 2 个向量读/写、2 个 DMA 读/写共 4 个并行请求。通过合理的数据排布，可以实现 DMA 传输和向量访存并行[8]。

3 基于 FT-M7002 的复数域大规模行向量矩阵乘法的分析与实现

3.1 复数域行向量矩阵乘法算法分析

3.1.1 复数域行向量矩阵乘法的串行算法分析

根据式(1)，可以得到计算复数域的行向量矩阵乘法的串行算法，如算法 1 所示。对其中参数做如下定义：$\boldsymbol{A}[n]$表示一个 n 维向量，$\boldsymbol{B}[n,m]$ 表示一个 $n\times m$ 矩阵。

算法 1 复数域行向量矩阵乘法串行算法

```
Input：行向量 A[n]，矩阵 B[n,m]
Output：行向量 C[m]
Begin
1. 初始化
2. for i=0:1:m do
3.    初始化 cTemp=0
4.    for j=0:1:n do
5.       cVecElem=A[j]
6.       cMtxElem=B[j,i]
7.       cTemp=cTemp+cVecElem * cMtxElem //复数相乘并累加
8.    end for
9. C[i]=cTemp
10. end for
end
```

复数乘法计算公式如下：

$$\begin{aligned}(a+bi)(c+di)&=ac+adi+bci+bdi^2\\&=ac+adi+bci-bd\\&=(ac-bd)+(ad+bc)i\end{aligned} \tag{2}$$

从算法 1 可见，计算一个复数域上的两个元素乘法需要对实部和虚部分别进行处理，如式(2)所示，运算结果的实部为两个复数的实部相乘再减去两个复数的虚部相乘，运算结果的虚部为两个复数的实部和虚部交叉相乘的和。依次计算完矩阵的前一列，再计算矩阵的后一列，如此循环，直到所有列全部完成。

算法 1 中存在两个循环，其中循环 1 为外层循环 for $i=0:1:m$ do，m 表示矩阵 $\boldsymbol{B}$ 的列数；循环 2 为内层循环 for $j=0:=1:n$ do，n 表示矩阵 $\boldsymbol{B}$ 的行数；其中循环 2 的主要功能就是逐个计算行向量 $\boldsymbol{A}$ 第 j 个元素 $\boldsymbol{A}[j]$和矩阵的第 j 行第 i 列的元素 $\boldsymbol{B}[j,i]$的复数乘法并累加，循环 1 则控制更新当前矩阵计算的列。由此可见，如果直接对内层循环进行循环展开优化，让多次复数相乘并累加操作并行执行，那么在内层循环结束后需要对多个并行单元中的结果规约求和，一次完整的内层循环只能得到矩阵 $\boldsymbol{B}$ 第 i 列的一个标量结果，这种计算方式在 SIMD DSP 上的执行效率较低。与此同时，这种计算方式需要对矩阵 $\boldsymbol{B}$ 进行列访问，在并行存储器中，矩阵 $\boldsymbol{B}$ 的输入数据大多以行方式存储，这种非连续访存的效率相较于连续访存的效率要低的多。

综上所述，将算法 1 直接进行 SIMD 指令优化并不能充分利用算法以及处理器的特性，因此不能在当前 DSP 上获得程序运行效率的最优解。考虑到矩阵计算的规律，可以改变计算方式，使用行向量 $\boldsymbol{A}$ 第 i 个元素 $\boldsymbol{A}[i]$乘矩阵第 i 行的方式，转列访问为行访问，同时可将内层循环计算复数乘法累加

操作后的不同并行单元之间的规约求和操作转换成不同并行单元内部的累加操作。

算法 2 改进后的复数域行向量矩阵乘法串行算法

```
输入：行向量 A[n]，矩阵 B[[n,m]
输出：行向量 C[m]
begin
1. for i=0:1:n do
2.   cVecElem=A[i]
3.   for j=0:1:m do
4.     if i==0 then
5.       C[j]=0 //初始化
6.     end if
7.     cMtxElem=B[i,j]
8.     C[j]=C[j]+cVecElem * cMtxElem
9.   end for
10.  end for
end
```

根据算法 1 可以得到行向量矩阵乘法的列计算转换成行计算的串行算法，见算法 2。在算法 2 中，循环 2 的主要功能变成逐个计算行向量 $\boldsymbol{A}$ 第 i 个元素 $\boldsymbol{A}[i]$ 和矩阵 $\boldsymbol{B}$ 的第 i 行的第 0，1，2，…，$m-1$ 个元素的复数乘法，并将结果累加。

3.1.2 复数域行向量矩阵乘法的向量化并行算法分析

复数域行向量矩阵乘法的向量化并行算法，简称为向量算法或向量 $\boldsymbol{C}$ 程序。根据 3.1.1 节的算法 2 分析，将循环 2 展开后，数据派发到指定的 VPE 上并行执行。每一次循环 2 执行后会产生一个 m 长度的向量，并需要将其累加到上一次的结果上。为了避免对其进行频繁操作，这里将结果向量的中间值先缓存在 AM 中，并且设置一个临时向量来保存中间值，减少对结果向量的重复访问。对于每一次计算循环 2 时所用到的行向量 $\boldsymbol{A}$ 第 i 个元素 $\boldsymbol{A}[i]$，需要把对应的值从内存中读取并广播到所有的 VPE 内，这样可以节约 DMA 传输时间。

在算法 3 中，VPE_N 表示开启的 N 个 VPE，在 FT-M7002 DSP 上 N 最大为 16。vec_svbcast() 是将元素广播到 VPE 中，vec_muli() 是乘法函数，vec_add() 是加法函数，vec_mula() 是乘加函数，vec_mulb() 是乘减函数，“vecTempI=vec_mula(vecVecI,vecMtixI,vecTempI);”的含义是 vecTempI=vecVecI × vecMtixI+vecTempI。

算法 3 复数域行向量矩阵乘法的向量化并行算法

```
输入：行向量 A[n]，矩阵 B[n,m]
输出：行向量 C[m]
begin
1. 初始化
2. 保存在内存 DDR 中的矩阵 B 的数据通过 DMA 传输到 AM 中
3. for i=0:1:n do
4.   vecVecI=vec_svbcast(scaVecI) //广播 A[i]的虚部值
5.   vecVecR=vec_svbcast(scaVecR) //广播 A[i]的实部值
6.   for j=0:VPE_N:m do
7.     获取矩阵第 i 行 j 到 j+VPE_N 复数数据的实部 vecMtixR 和虚部 vecMtixI
8.     获取结果行向量的 j 到 j+VPE_N 复数数据的实部数据 vecRestR 和虚部 vecRestI
9.     vecTempR=vec_muli(vecVecI,vecMtixI) //式(2)的 bd
10.    vecTempR=vec_mulb(vecVecR,vecMtixR,vecTempR) //式(2)的 ac-bd
11.    vecRestR=vec_add(vecRestR,vecTempR)
12.    vecTempI=vec_muli(vecVecR,vecMtixI) //式(2)的 ad
13.    vecTempI=vec_mula(vecVecI,vecMtixR,vecTempI) 式(2)的 ad+bc
14.    vecRestI=vec_add(vecRestI,vecTempI)
15.    保存结果 vecRestR,vecRestI
16.  end for
17. end for
18. 将 AM 中的结果数据通过 DMA 传输回 DDR 中
end
```

3.2 基于 MT-7002 的复数域行向量矩阵乘法实现与优化

在 3.1 节的分析中，SIMD 向量化需要利用 DMA 完成传输，而在整个程序运行的过程中，数据传输和计算占据了大部分时间。若可以合理优化安排数据传输和数据计算，则可以有效提升程序的运行效率。在实现中需要根据硬件的特点、行向量矩阵乘法的计算原理以及数据量大小分情况讨论，区别处理。

由于 DMA 传输存在时间消耗最小值，因此需要界定向量优化的起始数据量。算法 1 所示的复数域行向量矩阵乘法串行算法又称为标量算法，不同数据量下的标量算法和向量算法的计算节拍数如图 2 所示，可见在矩阵数据量少于 42 时，标量算法明显优于向量算法，这种情况下数据量极小，应当直接调用标量处理函数；其他情况则应当使用优化的向量算法。

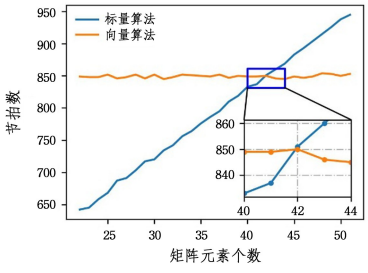

图 2 小数据量计算时间比较

Fig. 2 Computation time comparison of small data

在向量化并行算法中，行向量 $\boldsymbol{A}$ 逐个广播数据到 VPE 寄存器中，故行向量 $\boldsymbol{A}$ 数据不需要从 DDR 通过 DMA 传输到 AM 中，但是矩阵 $\boldsymbol{B}$ 和结果向量 $\boldsymbol{C}$ 的数据需要在 AM 中保存，因此，向量算法分 3 种情况进行处理，如图 3 所示。

1）AM 能放入矩阵 $\boldsymbol{B}$ 和结果向量 $\boldsymbol{C}$ 的全部数据，此时，将矩阵 $\boldsymbol{B}$ 的数据通过 DMA 全部传输到 AM 中，将行向量 $\boldsymbol{A}$ 的数据逐个广播到 VPE 中，与矩阵 $\boldsymbol{B}$ 中每一行数据进行计算，结果也存入 AM 中，计算结束后，通过 DMA 将数据传输回 DDR 内存。

2）AM 可以至少存放矩阵 $\boldsymbol{B}$ 的 2 行数据量和结果行向量 $\boldsymbol{C}$ 的数据，共至少 3 行数据。采用双缓冲策略，将矩阵 $\boldsymbol{B}$ 至少 2 行数据传输到 AM 中，同时结果行向量 $\boldsymbol{C}$ 也缓存在 AM 中，将行向量 $\boldsymbol{A}$ 的数据逐个广播到 VPE 中，与矩阵 $\boldsymbol{B}$ 的数据进行计算，结果缓存在 AM 中。计算结束后，通过 DMA 将最终结果行向量 $\boldsymbol{C}$ 传输回 DDR 内存。

3）AM 不能放入 3 行数据。将矩阵 $\boldsymbol{B}$ 的一行数据分成多块，结果向量 $\boldsymbol{C}$ 的一行数据也按照相同方式分块。保证 AM 能放入矩阵 $\boldsymbol{B}$ 的 2 块数据和结果向量 $\boldsymbol{C}$ 的 1 块数据。采用

双缓冲策略，轮流将矩阵 ***B*** 的数据分块传入 AM 中，与行向量 ***A*** 的数据进行计算，计算结束后，通过 DMA 将数据传输回 DDR 内存。

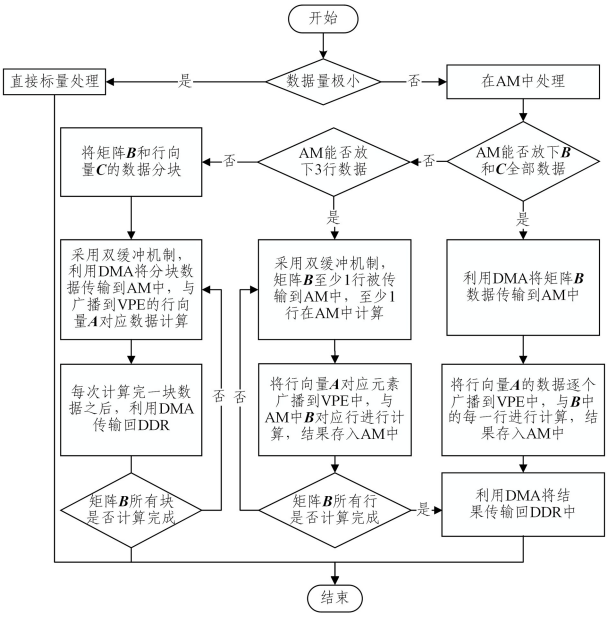

图 3 向量算法流程图

Fig. 3 Vector algorithm flow chart

双缓冲策略的基本思想是根据待计算的内容预先对 AM 空间划分缓冲区，设定好 DMA 传输所需要的传输参数，然后再启动 DMA 传输，开始 DMA 传输后，需要检测 DMA 是否已经传输完毕。但是在开启 DMA 传输之后，检测 DMA 传输结束之前的这段时间允许其他程序运行，因此可以实现在传输数据的同时进行数据计算。

向量算法采用双缓冲策略的具体计算过程中，需要将 AM 均分成 X，Y，Z 3 个区。X 区存放计算结果行向量 ***C*** 的数据，Y 区、Z 区存放矩阵 ***B*** 的数据，划分方式如图 4 所示。计算时先将数据缓存在 Y 区，然后在计算 Y 区数据的同时将下一部分的数据通过 DMA 缓存到 Z 区，在 Y 区计算完成后再计算 Z 区，此时再将下一部分数据缓存在 Y 区，循环往复直到数据计算完成。

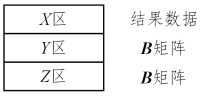

图 4 AM 分区

Fig. 4 AM partition

在 AM 中可以存放矩阵 ***B*** 的至少两行数据和结果行向量 ***C*** 的数据情况下，AM 的 Y 区、Z 区都尽可能多的缓存矩阵 ***B*** 的行数据。在计算时将行向量 ***A*** 的一个元素广播到 VPE 寄存器中，与缓存在 Y 区或者 Z 区的 ***B*** 矩阵的一行相乘，并将此次结果与缓存在 X 区前一行的结果进行累加，再存回 X 区。当所有行都计算完毕之后，通过 DMA 将结果数据传回 AM 中。

若 AM 不能放入 3 行数据，则需要将每一行数据拆分成多块。如图 5 所示，将矩阵 ***B*** 的一行数据分成 3 块，结果向量 ***C*** 的一行数据也分成 3 块。保证 ***AM*** 能放入矩阵 ***B*** 的 2 块数据和结果向量 ***C*** 的 1 块数据量。分批进行计算，计算的方式仍然是将行向量 ***A*** 进行广播与 ***B*** 矩阵进行计算并累加。如图 5 所示，先将 B00 块数据通过 DMA 传输到 AM 中，然后采用双缓冲机制，在计算行向量 ***A*** 数据和 B00 块数据并将结果存入 C0 块的同时，传输 B10 块数据到 AM 中。在 B00 块数据计算和 B10 块数据传输结束后，B10 块的计算和 B20 的传输又可以同时进行，依次处理，等 B30 块的数据计算结束后，C0 块的数据可以从 AM 通过 DMA 传输到 DDR 中。依次这样处理，直至所有数据计算完成。

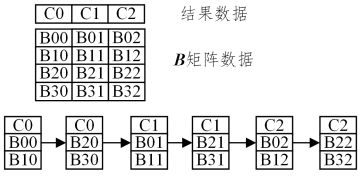

图 5 数据分块计算和传输

Fig. 5 Data chunking calculation and transmission

4 结果分析

向量、信号和图像处理库(VSIPL)是一组标准化的函数，为信号和图像处理应用程序提供可移植的计算中间件。本章介绍在 FT-M7002 硬件板上测试本文算法的性能以及实验结果分析。

为了测试基本算法及其优化算法的性能，分别编写了算法的向量 ***C*** 程序以及汇编程序，并对汇编程序使用循环展开和软件流水优化措施。在测试过程中使用不同的数据类型以及不同的数据量对 VSIP 库源程序、向量 ***C*** 程序及优化汇编程序进行测试。其中 VSIP 库源程序、向量 ***C*** 程序的测试通过－O2 优化选项编译生成 release 工程测试。

4.1 不同数据规模的性能测试

4.1.1 较小数据规模性能测试

为了测试算法性能以及不同数据类型对算法性能的影响，实验使用 32，48，64，80，96，112，128，144，160 阶单精度浮点数 float 复数矩阵和双精度浮点数 double 复数矩阵，分别测试了 VSIP 库程序、向量 ***C*** 程序和优化汇编程序，测得的节拍数如表 1、表 2 所列。

表 1 float 型各程序节拍数

Table 1 Cycles of each float type program

矩阵规模	VSIP 库源程序	向量 ***C*** 程序	优化汇编程序
32	26496	6949	1047
48	59988	10434	1174
64	104944	15727	1522
80	164964	20002	1969
96	236384	26926	2116
112	323092	31958	2663
128	419216	40470	3305
144	532452	46370	3440
160	655680	56374	4189

表 2 double 型各程序节拍数

Table 2 Cycles of each double type program

矩阵规模	VSIP 库源程序	向量 ***C*** 程序	优化汇编程序
32	27044	6628	1069
48	60636	10041	1262
64	107764	15393	1750
80	168204	19664	2363
96	242244	26725	2609
112	329532	31840	3388
128	430484	40611	4302
144	544620	46567	4593
160	672484	57051	5673

根据表 1、表 2 中的数据得到向量 ***C*** 程序相对于 VSIP 库程序、优化汇编程序相对于向量 ***C*** 程序的加速比，具体的加速比如图 6、图 7 所示，其中横坐标表示矩阵的阶数，纵坐标表示加速比。

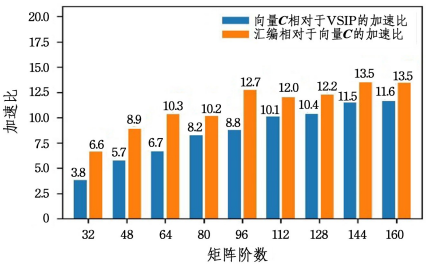

图 6　小规模 float 型加速比

Fig. 6　Float type acceleration ratio of small scale

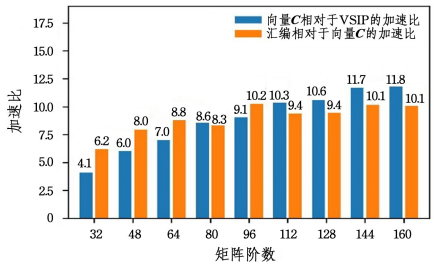

图 7　double 型加速比

Fig. 7　Double type acceleration ratio of small scale

4.1.2　较大数据规模性能测试

实验分别测试了单精度浮点型和双精度浮点型 512，640，768，896，1 024，1 152，1 280，1 408，1 536 阶矩阵，测得的节拍数如表 3、表 4 所列。

表 3　float 型各程序节拍数

Table 3　Cycles of each float type program

矩阵规模	VSIP 库源程序	向量 ***C*** 程序	优化汇编程序
512	6 774 352	404 177	28 157
640	10 746 874	688 234	72 336
768	15 588 359	954 877	90 734
896	21 250 367	1 282 504	130 183
1024	27 779 298	1 658 613	176 568
1152	35 158 706	2 071 408	219 555
1280	43 467 063	2 534 807	271 612
1408	52 545 069	3 042 541	327 879
1536	62 611 030	3 597 424	392 480

表 4　double 型各程序节拍数

Table 4　Cycles of each double type program

矩阵规模	VSIP 库源程序	向量 ***C*** 程序	优化汇编程序
512	7 333 553	489 391	77 724
640	11 548 444	741 815	133 307
768	16 607 166	1 029 074	237 864
896	22 730 173	1 406 176	264 202
1 024	29 602 961	1 860 505	345 362
1 152	37 417 612	2 339 331	438 708
1 280	46 302 386	2 777 472	668 793
1 408	55 888 203	3 456 684	655 033
1 536	66 697 848	3 953 468	963 883

根据表 3、表 4 中的数据得到向量 ***C*** 程序相对于 VSIP 库程序、优化汇编程序相对于向量 ***C*** 程序的加速比，具体的加速比如图 8、图 9 所示。

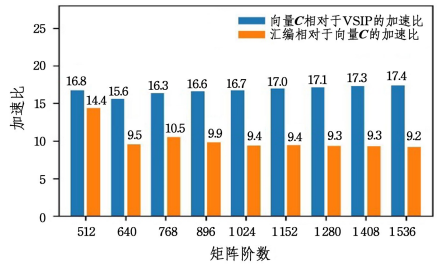

图 8　较大规模 float 型加速比

Fig. 8　Float type acceleration ratio of medium scale

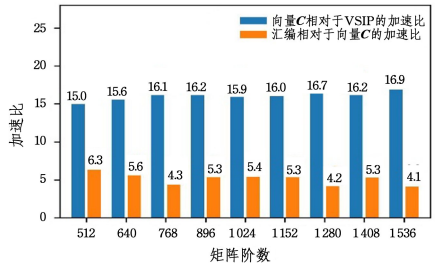

图 9　较大规模 double 型加速比

Fig. 9　Double type acceleration ratio of medium scale

4.1.3　超大数据规模算法性能测试

实验测试了行数为 256、列数分别为 17 920，20 480，23 040，25 600，28 160 规模的矩阵，同时也测试了不同数据类型对算法性能的影响，测得的节拍数如表 5、表 6 所列。

表 5　float 型各程序节拍数

Table 5　Cycles of each float type program

矩阵规模	VSIP 库源程序	向量 ***C*** 程序	优化汇编程序
256 * 17 920	121 954 869	5 953 577	991 773
256 * 20 480	139 736 190	6 782 252	1 124 225
256 * 23 040	157 218 490	9 035 718	1 296 210
256 * 25 600	174 722 728	10 011 341	1 419 341
256 * 28 160	192 199 265	11 015 219	1 552 663

表 6　double 型各程序节拍数

Table 6　Cycles of each double type program

矩阵规模	VSIP 库源程序	向量 ***C*** 程序	优化汇编程序
256 * 17 920	129 102 117	8 583 428	1 915 680
256 * 20 480	147 553 139	9 789 566	2 177 519
256 * 23 040	165 994 745	10 987 159	2 480 482
256 * 25 600	184 423 736	12 240 319	2 744 274
256 * 28 160	202 880 302	13 443 354	3 007 032

根据表 5、表 6 数据得到向量 ***C*** 程序相对于 VSIP 库程序、优化汇编程序相对于向量 ***C*** 程序的加速比，具体的加速比如图 10、图 11 所示。

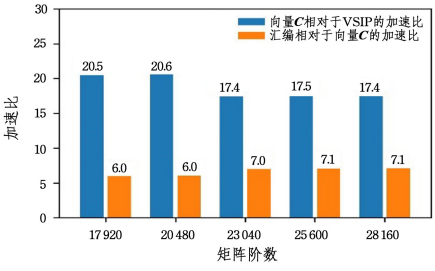

图 10　大规模 float 型加速比

Fig. 10　Float type acceleration ratio of large scale

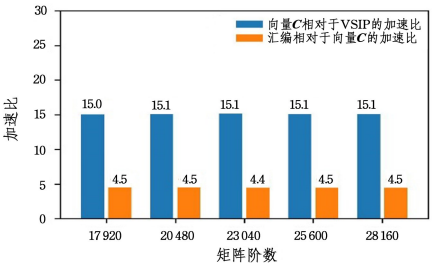

图 11 大规模 double 型加速比

Fig. 11 Double type acceleration ratio of large scale

4.2 性能分析

从图 6－图 11 中向量 ***C*** 程序相对于 VSIP 库程序的加速比可以看出，在对 VSIP 库函数进行向量优化之后，优化后的向量 ***C*** 算法相对于串行 C 算法，处理单精度浮点数获得了 6.64～20.6 的加速比，处理双精度浮点数获得了 6.2～16.8 的加速比；从不同数据量的总体趋势来看，浮点数随着数据量的增加而上升，最后分别稳定在 15 和 17 左右；从处理的数据类型来看，单精度浮点型数据优化效果要优于双精度浮点型。从图 6 和图 7 来看，加速比随着数据量的变化而快速变化，在数据量较少时，由于采用 DMA 传输，存在最小的 DMA 传输时间，向量化计算的优势不大，但随着数据量增加，优势非常明显；从图 8－图 11 来看，数据量大到一定程度时，DMA 传输时间也会增加，因此加速比会趋于稳定。

从图 6－图 11 优化汇编程序相对于向量 ***C*** 程序的加速比可以看出，获得了 3.4～14.3 的加速比，优化加速效果较为明显。由图 6－图 9 可以发现，优化汇编程序相对于向量 ***C*** 程序的加速比随着数据量的增加先上升后趋于平稳，图 6 和图 8 对比更加明显，这是由于数据量越大，DMA 传输时间也越长。图 10 和图 11 显示，极大数据量时，DMA 传输时间更长，所以优化效果相对较差。综合来看，优化达到了增加循环内指令的并行度、优化冗余代码、提高循环执行效率的目的。

结束语 针对 FT-M7002 处理器向量 SIMD 处理的特点，研究了复数域的行向量矩阵乘法算法的优化。根据行向量矩阵乘法的运算特点，改列运算为行运算，避免了内存的非连续访问。同时结合硬件特点对算法进行 SIMD 指令优化，并对算法进行汇编优化。结果表明：优化后的向量 ***C*** 程序相对于 VSIP 库函数获得了 6.2～20.6 的加速比，加速效果明显，经过汇编优化后的程序相对于向量 ***C*** 程序获得了 3.4～14.3 的加速比，加速效果显著。研究针对向量 DSP 单核的行向量矩阵乘法算法的优化，有助于进一步研究针对多核情况的算法优化。

参 考 文 献

[1] ZHANG Y H, LIU X G. Parallel Algorithm of Matrix Multiplication Based on MPI & OpenMP[J]. Computer and Modernization, 2011(7): 84-87.

[2] LIM R, LEE Y, et al. An implementation of matrix-matrix multiplication on the Intel KNL processor with AVX-512[J]. Cluster Computing, 2018, 21: 1785-1795.

[3] LI X W, CUI X. Performance optimization of matrix multiplication and FFT in GPU[J]. Modern Electronics Technique, 2013, 36(4): 80-84.

[4] ZHANG M Y. Parallel implementation of matrix multiplication based on CUDA[J]. Changjiang Information & Communications, 2012(2): 20-21.

[5] SHAO Y M, ZHOU J. Implementation of Customized Instruction for RISC-V CPU Based on FPGA[J]. Software, 2022, 43(1): 161-164.

[6] TIAN X, ZHOU F. Design of field programmable gate array based real time double-precision floating-point matrix multiplier [J]. Journal of Zhejiang University(Engineering Science), 2008(9): 1611-1615.

[7] WANG Y H, LI C, LIU C, et al. Advancing DSP into HPC, AI, and beyond: challenges, mechanisms, and future directions[J]. CCF Transactions on High Performance Computing, 2021, 3(1): 114-125.

[8] LI H X, ZHANG H F. A Cholesky decomposition vector processing algorithm for FT-M7002[J]. Journal of Shaoyang University(Natural ScienceEdition), 2022, 19(3): 9-17.

MO Shangfeng, born in 1977, Ph.D, is a member of China Computer Federation. His main research interests include DSP compilation and embedded system.