基于超大点数 FFT 优化算法的研究与实现

高立宁 马 潇* 刘腾飞 吴 金
(北京理工大学信息与电子学院 北京 100081)

摘 要：针对应用系统对超大点数快速傅里叶变换(FFT)的性能需求不断提升，以及现有处理平台的资源对实现超大点数 FFT 的制约问题，该文提出一种超大点数 FFT 的实现方法。该方法通过优化铰链因子存储，采用行列号方式访问 2 维矩阵避免了 3 次显性转置，从而节省了内存资源；同时，通过分析处理器的分级存储结构特点，优化了矩阵行列划分规则，进而提高了行列访问效率。实验结果表明，该方法节约了近一半的内存资源，且有效提高了超大点数 FFT 的执行速度。

关键词：信号处理；内存优化；超大点数 FFT；矩阵转置；Winograd 算法
中图分类号：TN911.7 **文献标识码**：A **文章编号**：1009-5896(2014)04-0998-05
DOI: 10.3724/SP.J.1146.2013.00841

Research and Implementation of Ultra Large Sequences FFT Optimized Algorithm

Gao Li-ning Ma Xiao Liu Teng-fei Wu Jin
(*School of Information and Electronic*, *Beijing Institute of Technology*, *Beijing* 100081, *China*)

Abstract: For the increasing requirements of ultra long sequences FFT's efficient implementation in application systems, an implementation method is proposed to realize ultra long sequences FFT. The method can solve resources limitation of existed processing platforms, which can save mass memory resources by optimizing the twiddle factor store and adopting the way of rows and columns instead of three explicitly matrix transposition to access the two-dimensional matrix. The characteristics of processors' hierarchical memory structure is analysed to optimize the rules of matrix partition, which improves the access efficiency. Experiment results show that the performance of execution efficiency of ultra long sequences FFT is improved nearly half of the memory resources.
Key words: Signal processing; Memory optimizing; Ultra long sequences FFT; Matrix transposition; Winograd algorithm

1 引言

快速傅里叶变换(FFT)在雷达、数字通信和图像处理等科学技术领域有着广泛的应用，这使得 FFT 的工程实现具有十分重要的意义。随着现代数字信号处理技术的发展，应用系统中对实现超大点数 FFT 的需求越来越高，这也意味着对数字信号处理器(DSP)的处理性能和内存资源都提出了更高的要求和挑战[1]。

在雷达系统中，高分辨大测绘带宽的合成孔径雷达的飞速发展，对信号处理系统中实现超大点数 FFT 提出了更高的要求。然而，超大点数 FFT 的实现对资源的消耗随着点数的增加而增大，且对超大点数 FFT 的优化效率往往和资源开销的程度直接相关，这都说明处理器资源已经成为制约大点数 FFT 实现的关键因素。因此，如何在节约现有平台资源开销并保证执行效率的条件下实现超大点数 FFT，成为当前研究的重要方向。

很多文献对大点数 FFT 的实现进行了深入的研究，对其性能的优化主要采用了两种方式：一种是通过算法的优化，使 FFT 处理更加适合处理器架构；另一种是增加系统资源，达到并行处理。例如，文献[2,3]中给出了 SingLeton 结构实现定点/浮点 FFT 的方法，采用该结构对蝶形进行重排，使除了第 1 级外的其它级数据都是顺序读取，具有较高运算效率，但是这种方法要采用乒乓缓存，浪费了一倍的存储器；文献[4]采用多片 DSP 分级并行处理一个大点数 FFT，将蝶形运算分配到多个 DSP 分级并行计算，进而提高 FFT 的执行效率，但是该处理方法需要多片 DSP 并行计算，造成了资源浪费，在实际工程应用中成本较高，且增加了开发难度；文献[5]提出了一种基于 FPGA 的高性能并行 FFT 处

2013-06-14 收到，2013-10-31 改回
*通信作者：马潇 mxdreams@163.com

理方案，其采用 4 个蝶形运算单元进行并行处理，优化了处理效率，其缺点是增加了 4 倍的资源消耗；文献[6]采用 Winograd 算法将大点数 FFT 拆成小点数进行处理，虽然在处理中引入了额外的铰链因子相乘运算和 3 次显性转置操作，但是该算法能够有效地降低资源的消耗，能够在现有平台上实现更大点数的 FFT 运算。

本文针对处理器硬件资源对实现超大点数 FFT 的制约，提出一种优化的 Winograd 算法实现超大点数 FFT 的方法。相比传统的 Winograd 处理方法，该实现方法通过优化铰链因子存储，改变矩阵访问方式和限制行列划分等手段使 FFT 处理能够更好的适应处理器的架构，使其在现有平台上可以实现更大点数的 FFT 运算。

2　资源优化分析

文献[6]中 Winograd 算法实现 FFT 的主要流程为：(1)将 1 维序列拆分成 2 维矩阵并转置；(2)行方向 FFT 处理；(3)乘以铰链因子；(4)对(3)的结果转置存储；(5)列方向 FFT 处理；(6)将处理结果转置存储，得到序列的 FFT 结果。该处理算法将大点数 FFT 拆成小点数运算，但是该算法在实现过程中存在两个问题：一是引入的铰链因子需要额外的存储空间，同时增加了乘法运算次数；二是 3 次显性转置需要较多额外的内存空间，且存储转置处理引起的 Cache 丢失对时间影响很大。针对上述问题，本节给出具体的优化方案。

2.1 铰链因子存储优化

为了便于说明，这里重设 2 维序列 $N = M \times L$，M 为列数，L 为行数，并设 n_1 和 n_0 分别表示时域行列序号，k_0 和 k_1 分别表示频域的行列序号。由文献[6]中给出的处理步骤可知，在第(3)步时对行方向 FFT 的处理结果乘以铰链因子，其值如式(1)：

$$Z(n_0,k_0) = e^{\frac{-j2\pi n_0 k_0}{N}} \tag{1}$$

其中 $n_0 = 0,1,\cdots,M-1$，$k_0 = 0,1,\cdots,L-1$。

由式(1)可推导：

$$Z(n_0,k) = e^{\frac{-j2\pi n_0}{N}} \times e^{\frac{-j2\pi n_0}{N}} \times \cdots \times e^{\frac{-j2\pi n_0}{N}} \tag{2}$$

由式(2)可以看出，铰链因子与 n_0 和 k 两个变量有关，且可以将第 k 行的铰链因子分解为 k 个 $e^{\frac{-j2\pi n_0}{N}}$ 相乘，因此，在存储铰链因子时可以只存储第 1 行的铰链因子，而其它行的铰链因子可由第 1 行的铰链因子计算得出。虽然采用计算得出铰链因子会引入新的额外乘法运算，但该处理方式可以大大节省内存资源，这对在资源有限的情况下实现超大点数 FFT 十分重要。为了得出新的额外乘法运算对超大点数 FFT 执行效率的影响，下面对该运算时间在 FFT 总运算时间中所占的比重作进一步分析[7]。

Winograd 算法实现 FFT 运算所需运算量为

$$\left.\begin{aligned} m_{\mathrm{CpxMul}} &= \frac{M}{2}\log_2 M + \frac{L}{2}\log_2 L + N \\ a_{\mathrm{CpxSum}} &= M\log_2 M + L\log_2 L \end{aligned}\right\} \tag{3}$$

采用节省内存空间的方式存储铰链因子会额外引入 N 次乘法运算，又由于处理器进行乘法运算的时间比加法多得多，因此，如果只考虑 FFT 计算时间与乘法次数成正比，则新引入的额外乘法运算时间占 FFT 运算时间的比例为

$$\eta = \frac{N}{\frac{M}{2}\log_2 M + \frac{L}{2}\log_2 L + N} \tag{4}$$

由柯西不等式可以得

$$\eta \le \frac{4}{\log_2(M+L)+4} \tag{5}$$

当 $M = L = \sqrt{N}$ 时，式(5)取最大值：

$$\eta_{\max} = \frac{4}{\log_2\sqrt{N}+5} \tag{6}$$

由式(6)和图 1 可见，新引入的额外乘法运算随着 FFT 点数的增加逐步减小，其对 FFT 的执行效率的影响也在逐渐减低。例如，按照理论分析在 256k 点时，额外乘法运算只占 FFT 乘法总运算时间的 18.49%；然而，该处理方法可以节约 $2(N-2L)$ 的内存空间，这为在现有处理器资源不变的情况下实现更大点数的 FFT 提供了可能。

经以上推导分析，为了节省处理器内存资源，本文采取只存储第 1 行的方法来存储铰链因子，且由此引入的新的额外乘法运算对 FFT 执行时间的影响随点数的增加而逐渐减小；所以，对于超大点数 FFT 的实现来说，该部分的时间开销可以忽略不计。

2.2 矩阵访问优化

相对于 1 维序列的处理，文献[6]的 Winograd

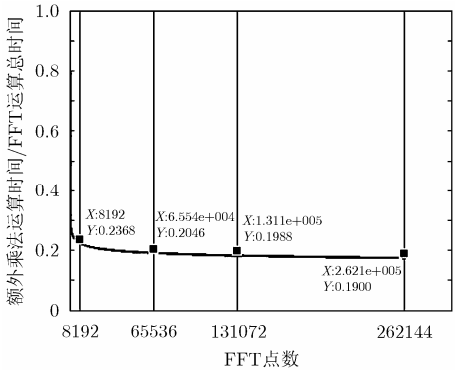

图 1 随点数增加额外乘法运算时间/FFT 乘法总运算时间曲线图

算法将序列映射成 2 维序列处理，对 2 维矩阵的行列访问采用转置成正常顺序后连续读取的方式，对于 FFT 处理，该方法引入了 3 次显性转置，而转置处理需要大量的额外转置内存空间，且矩阵转置会对 FFT 的执行速度产生严重的影响。目前主流 DSP 处理器都采用分级存储结构，一级缓存(Cache)会对数据进行预处理来协助对矩阵数据的访问，从而减少直接对 DRAM 的访问次数；当数据在 Cache 中命中，可以直接从 Cache 中读取所需数据，否则需要重新到 DRAM 中读取新的数据页面，此时会引起数据访问时延[8]。文献[9]采用变换行列号方式实现对 2 维矩阵的访问，即通过行列号来计算数据首地址和访问步长，然后进行数据访问。但其带来的问题是，读取一列数据时，需要每读一个数据均进行行切换，而行切换需要切换时间；读取一行数据时，行数据不能太长，如果超出一级 Cache 的容量，在行 FFT 处理过程中，则会出现因无法从 Cache 命中数据带来的各种问题。因此，需要通过优化行列的拆分规则来最大发挥 Cache 协助数据访问的特点，实现对 2 维矩阵高效访问。

为了尽量减少行切换，可以在列处理时一次读取几列数据，读取列数的多少以及每列含数据量大小均需要根据 Cache 的容纳量计算，以实现对 Cache 的充分利用，但要保证读取列的总数据小于 Cache 大小，否则在列 FFT 处理时会出现 Cache 频繁丢失的情况；图 2 所示为在进行一次列数据访问读取第 i 列的数据时，数据在 Cache 中的布局图。

在行 FFT 处理时为了充分利用 Cache，只有当读一行的数据尽可能填充 Cache 空间时才能充分利用 Cache 的优势来协助数据访问。矩阵行处理访问内存后数据在 Cache 中分布如图 3 所示。

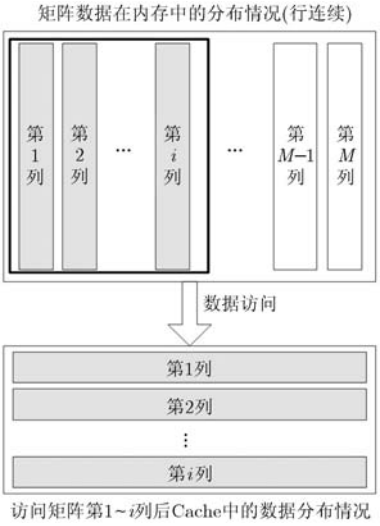

图 2 矩阵列方向访问后的缓存布局

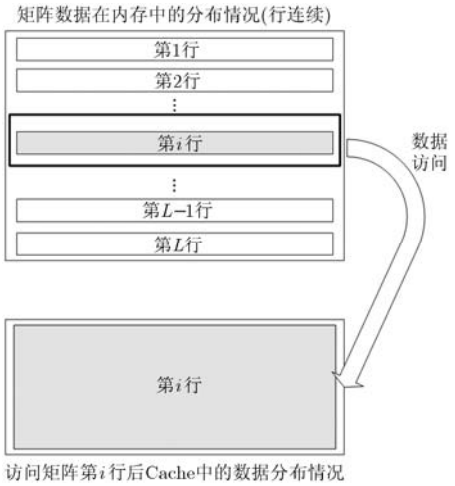

图 3 矩阵行方向访问后的缓存布局

由此，采用行列号方式访问 2 维矩阵，节省了额外的转置存储空间，解决了 3 次显性转置引起时间开销对 FFT 执行效率影响的问题，并结合处理器分级存储的特点对 2 维矩阵行列划分规则进行优化，最大限度发挥 Cache 的优点来提高矩阵访问速度。

上述两小节分别从铰链因子和矩阵转置两个方面对资源及效率进行优化：一是依据铰链因子的规律，采取减少存储铰链因子的数据量来节省内存空间，该方法引入了额外的乘法运算，但经过理论分析该额外运算引起的时间开销对超大点数 FFT 的执行效率影响较小；二是改变矩阵的访问方式，节省了显性转置增加的额外存储空间，同时通过限制行列划分规则，在一定程度上提高了数据访问的速度。表 1 给出了本文方法采用 Winograd 算法的优化方法，实现超大点数 FFT 时对内存资源的需求。其中 $(M,L)_{\max}$ 表示取其中最大的缓冲区，该缓冲区行列可复用。下面通过工程实例来具体说明采用 Winograd 算法的优化方法对超大点 FFT 的实现流程。

3 工程实例

TS201 是 ADI 公司的 TigerSHARC 系列 DSP，

表 1 本文方法实现 FFT 对资源的需求

序号	内存空间
原始数据空间	$2\times N$
旋转因子空间	$M+L$
铰链因子空间	$4\times L$
乒乓缓存空间	$4\times(M,L)_{\max}$
总需求空间	$2\times N+(M+L)+4\times L+4\times(M,L)_{\max}$

其采用超级哈佛结构，最高可支持 600 MHz 的内核时钟，具有双运算单元和双整数 ALU，其内部集成 24M bit 的 DRAM 存储器，整个 DRAM 划分为 6 个分区(BANK)，每个 DRAM 存储器块带有一个 128k bit 的 Cache，用来协助数据访问。它的静态超标量结构使其能够在 1 个周期内执行 4 条指令，完成 24 次 16 bit 定点运算，12 次 32 bit 定点运算 6 次浮点运算[10]。

由于在一个 BANK 里面只能存储 128k 复数点的数据量，因此考虑将原始数据存储在两个不同的 BANK 中，采用 J 和 K 总线同时访问两个 BANK 的数据；另外，TS201 的双运算单元可以一次读取 4 列 8 个复数点的数据，Cache 的大小为 128k bit，按照第 2 节所述的行列划分规则对 1 维矩阵进行划分：M=2048, L=64, i=8；即行处理时，每次读取一行的数据是 2048 个复数点可以刚好填充 Cache 的空间，在每次列处理时，分 8 次读取数据，一次读取 4 列，也刚好可以填充一个 Cache 的空间。因此，在行列 FFT 处理时，只在蝶形运算第 1 级和最后一级存在 Cache 丢失的情况下，其它级运算因为数据都缓存在 Cache 中而提高了数据命中概率。图 4 为 128k 复数浮点 FFT 的程序设计流程图。

本文主要从指令优化和汇编优化两个方面对 128k 复数浮点 FFT 的程序设计进行优化：在指令优化方面，TS201 提供了合适复数运算的指令集，其内部有 4 条 128 bit 宽度的内部总线，通过 J 和 K 总线可以同时实现一个周期内 4 个 32 bit 数据的读和写操作，且其存在两个完全对称的 X/Y 计算单元和指令对齐缓冲器(IAB)为单个时钟周期内运行 4 条指令提供了硬件基础，所以，理论上一个周期内可以完成两个蝶形运算；在汇编优化方面，TS201 采用并行的超标量流水线技术，通过合理安排指令流水，使程序进入核循环时内核运算单元(乘法运算、加减法运算)的操作与内存访问的 IO 操作均并行起来，使程序只在填充和排空时存在一定的串行操作，保证了算法的高效运算[11]。

4　资源开销及速度对比

验证实验采用北京理工大学雷达所研制的 8TS201 通用信号处理板卡进行实测，即通过在 FFT 函数运行前后分别读取 TS201 内部的时钟计数器，计算二者差值并除以 TS201 的处理器主频可以来得到运行时间，运行时间见表 2。

由表 2 可以看出，相对与文献[6]的 Winograd 算法实现 FFT，采用本文的方法对 Winograd 算法实现进行优化，其在小点数时因引入了额外的乘法运算致使执行效率大概降低 16%；但是，在大点数时 FFT 的执行效率却得到了提升，这主要是因为该处理方法采用行列号方式优化矩阵访问使其省去了 3 次显性转置，并通过限制行列划分规则来减少 Cache 的丢失对执行效率的影响。另外，由表 3 可以看出，在超大点数时，本文 Winograd 算法的实现方法比传统方法节省了近一半的内存资源，较其它算法也有巨大优势，这为同一处理器平台上实现更大点数的 FFT 提供了一种可行的方法。

5　结束语

本文针对现有处理器硬件平台对实现超大点数 FFT 的限制因素进行了深入分析，对 Winograd 算法的实现方法进行了资源优化处理，并依据处理器分级存储结构的特点优化了行列划分规则，从而提高了 FFT 处理过程中的行列访问效率；同时，给出了在 TS201 上实现了 128k 复数浮点 FFT 的实现流

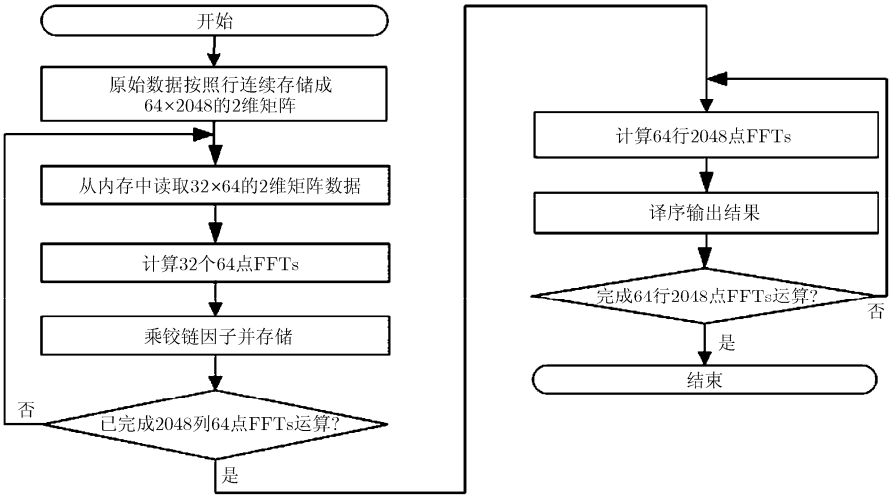

图 4　128k 复数浮点 FFT 程序实现流程图

表 2 FFT 运行时间对比 (μs)

点数	文献[6]	本文
8192	333.55	391.33
16384	722.15	829.57
32768	1528.45	1597.82
65536	3301.36	3190.14
131072	–	5738.99

表 3 FFT 内存开销对比(kB)

点数	原位基2FFT	文献[6]	SingLeton算法	本文
4096	48	64	80	72
8192	96	128	160	104
16384	192	256	320	168
32768	384	512	640	296
65536	768	1024	1280	552
131072	1536	2048	2560	1064

程。实验验证及资源对比表明，在超大点数 FFT 时，本文的处理方法能够节约一半的处理器资源开销，明显提升了大点数 FFT 处理的处理效率，该处理方法实现的超大点数 FFT 具有很高的工程应用价值。

参 考 文 献

[1] 李斌, 田素雷, 孙雪晶. 大点数 FFT 设计中提高资源利用率的方法[J]. 无线电工程, 2011, 41(1): 54–57.
Li Bin, Tian Su-lei, and Sun Xue-jing. Method of improving resource utilization for large point FFT design[J]. *Radio Engineering*, 2011, 41(1): 54–57.

[2] Boris Lerner. Writing efficient floating-point FFTs for ADSP-TS201 TigerSHARC[OL]. http://www.Analog.com/dsp, 2012.

[3] 李欣, 刘峰, 龙腾. 定点FFT在TS201上的高效实现[J]. 北京理工大学学报, 2010, 30(1): 88–91.
Li Xin, Liu Feng, and Long Teng. Efficient implementation of fixed-point FFT on TS201[J]. *Transactions of Beijing Institute of Technology*, 2010, 30(1): 88–91.

[4] 刘莉, 高梅国, 周闰, 等. 大点数FFT的多DSPs并行处理算法及实现[J]. 系统工程与电子技术, 2003, 25(10): 1193–1197.
Liu Li, Gao Mei-guo, Zhou Run, *et al*.. Algorithm and implementation of a large-point FFT under the master-slave parallel multi-processor architecture[J]. *Systems Engineering and Electronics*, 2003, 25(10): 1193–1197.

[5] 石长振, 杨雪, 王贞松. 高性能并行 FFT 处理器的设计与实现[J]. 计算机工程, 2012, 38(2): 242–245.
Shi Chang-zhen, Yang Xue, and Wang Zhen-song. Design and realization of high performance parallel FFT processor[J]. *Computer Engineering*, 2012, 38(2): 242–245.

[6] Boris Lerner. Parallel implementation of fixed-point FFTs on TigerSHARC processors[OL]. http://www.Analog.com/dsp, 2012.

[7] 王世一. 数字信号处理[M]. 北京: 北京理工大学出版社, 1997: 123–131.

[8] 李浩, 谢伦国. 片上多处理器末级 Cache 优化技术研究[J]. 计算机研究与发展, 2012, 49(增刊): 172–179.
Li Hao and Xie Lun-guo. Research development of optimization technology on last level cache in chip multi-processors[J]. *Journal of Computer Research and Development*, 2012, 49(Suppl.): 172–179.

[9] 周永彬, 张军超. 基于软硬件的协同支持在众核上对 1-DFT 算法的优化研究[J]. 计算机学报, 2008, 31(11): 2005–2014.
Zhou Yong-bin and Zhang Jun-chao. Software & hardware co-design for 1-D FFT optimization on many-core architecture[J]. *Chinese Journal of Computers*, 2008, 31(11): 2005–2014.

[10] 刘书明, 罗勇江. ADSP TS20XS 系列 DSP 原理与应用设计[M]. 北京: 电子工业出版社, 2007: 6–7.

[11] Analog Device Inc. ADSP-TS201 TigerSHARC Processor Programming Reference[M]. Norwood: Mass, US, Analog Device Incorporation, 2004: 172–199.

[12] 刘志哲, 仲顺安. 基于分级存储并行运算的 FFT 处理器设计[J]. 北京理工大学学报, 2011, 32(6): 691–684.
Liu Zhi-zhe and Zhong Shun-an. Design of FFT processor based on grade-memory and parallel-computation[J]. *Transactions of Beijing Institute of Technology*, 2011, 32(6): 691–694.

[13] 苏涛, 庄德靖. 大点数 FFT 算法的改进及其实现[J]. 现代雷达, 2005, 27(7): 23–27.
Su Tao and Zhuang De-jing. Improvement and implementation of FFT algorithm for long sequences[J]. *Modern Radar*, 2005, 27(7): 23–27.

[14] Analog Device Inc. TigerSHARC DSP 32 bit REAL/COMPLEX FFT example [EB/OL]. http://www.Analog.com/dsp, 2012.

[15] Analog Device Inc. TigerSHARC DSP complex fixed point FFT example for TS201 and TS101[EB/OL]. http://www.Analog.com/dsp, 2012.

[16] Bailey D H. FFTs in external or hierarchical memory[J]. *Journal of Supercomputing*, 1990, 4(1): 23–35.

高立宁： 男，1981 年生，讲师，高速雷达实时信号处理、目标探测与识别.

马 潇： 男，1987 年生，硕士，高速雷达实时信号处理、目标探测与识别.

刘腾飞： 男，1985 年生，博士，高速雷达实时信号处理、目标探测与识别.

吴 金： 男，1989 年生，硕士，高速雷达实时信号处理.