中国科学 信息科学
SCIENTIA SINICA Informationis

基于三值光学计算机的并行快速Fourier算法实现

彭俊杰, 魏鑫??, 张晓峰, 沈云付 and 付友谊

Citation: 中国科学: 信息科学 **47**, 846 (2017); doi: 10.1360/N112016-00164

View online: http://engine.scichina.com/doi/10.1360/N112016-00164

View Table of Contents:http://engine.scichina.com/publisher/scp/journal/SSI/47/7

Published by the 《中国科学》杂志社

中国科学: 信息科学 2017 年 第 47 卷 第 7 期: 846–862
SCIENTIA SINICA Informationis
论文

基于三值光学计算机的并行快速 Fourier 算法实现

彭俊杰[1*], 魏鑫燏[1,2], 张晓峰[3*], 沈云付[1], 付友谊[1]

1. 上海大学计算机工程与科学学院, 上海 200444
2. 中国科学院上海高等研究院, 上海 201210
3. 毫米波遥感技术国家级重点实验室, 北京 100854
* 通信作者. E-mail: jjie.peng@shu.edu.cn, xiangfan_zxf@sohu.com

收稿日期: 2016–06–21; 接受日期: 2016–08–30; 网络出版日期: 2017–02–27
国家自然科学基金 (批准号: 61572305, 61103054) 和中国航天科工集团二院 “自主” 创新项目资助

摘要 快速 Fourier 变换 (FFT) 是信号处理领域应用十分广泛的算法, 在高速、实时的应用环境中常用硬件并行实现的方法来加快 FFT 的运算速度, 但在一些特定领域如在空间受限、对能耗及散热具有较高要求的航空航天设备中, 传统的电子方法将受到很大的局限, 而三值光学计算机以其能耗低、数据位数众多的优点使得它可能具有广泛的应用前景. 针对这种现状, 本文研究了采用三值光学计算机来实现快速 Fourier 变换的设计方案和方法. 通过对传统基 2、基 4 和基 8 时域抽取快速 Fourier 变换运算过程的分析, 利用三值光学计算机数据位数众多和数据位数易扩展的特点设计了多个并行度更高的快速 Fourier 变换算法, 给出了详细的算法实现流程并进行了各算法间的对比, 分析了实现方案所需的时钟周期和硬件资源, 模拟验证了该实现方案的正确性. 相比于传统基于现场可编程门阵列的并行实现方法, 这种在三值光学计算机上实现的快速 Fourier 变换运算功耗更低、所需时钟周期数更少, 这为在嵌入式设备中高速低功耗地实现快速 Fourier 变换提供了新的解决思路.

关键词 三值光学计算机, 快速 Fourier 变换, 并行计算, MSD 加法器, 嵌入式设备

1 引言

快速 Fourier 变换 (FFT: fast Fourier transformation) [1] 作为在数字信号处理领域的重要工具, 在雷达信号分析、图像处理、语音分析等应用中被广泛使用.

针对高速、实时的应用场合, 目前常见的 FFT 硬件加速器采用基于流水线的单路径延时反馈结构 [2] 或者各级内蝶形变换并行处理方案 [3]. 通常使用现场可编程门阵列 (FPGA) [3~6] 或者数字信号

引用格式: 彭俊杰, 魏鑫燏, 张晓峰, 等. 基于三值光学计算机的并行快速 Fourier 算法实现. 中国科学: 信息科学, 2017, 47: 846–862, doi: 10.1360/N112016-00164
Peng J J, Wei X Y, Zhang X F, et al. Implementation of parallel FFT algorithm on a ternary optical computer (in Chinese). Sci Sin Inform, 2017, 47: 846–862, doi: 10.1360/N112016-00164

处理器 (DSP) [7] 实现 FFT 硬件加速. 但在空间和资源有限的嵌入式设备上, 如在星弹载设备上, 一方面传感器的数量不断增加使得需要实时处理的数据量不断增多, 另一方面又存在设备空间有限、散热能力不足和可用电量有限的问题, 这对传统的通过电子硬件加速的方法来实现 FFT 实时应用提出了挑战, 必须寻求新的解决方案以应对这一挑战.

与电子计算机相比, 三值光学计算机 (TOC: ternary optical computer) 功耗低、数据位数众多、数据位数易增加以及并行性的特点使其更适于解决目前嵌入式设备中高速 FFT 算法实现所面临的问题. 由于 TOC 中的液晶阵列和嵌入式控制部件都是低功耗的设备, 相比于电子计算机能耗更低、发热更少, 同时其众多的数据位以及数据位极易扩展的特性使其更适于以高度并行方式实现 FFT. TOC 目前已具备了初步的理论体系和实验平台, 从 2003 年公开 TOC 的思想开始 [8], 经过降值设计理论 [9] 的提出、TOC 编码器和解码器 [10,11]、改进符号数 (MSD) 加法器 [12~17]、可重构三值光学处理器的实现 [18,19] 以及诸如向量矩阵乘 [20,21]、元胞自动机 [22] 等基于 TOC 的应用实现, 当前 TOC 已具备了实现并行 FFT 算法的条件.

本文一方面利用 TOC 天然的低功耗特点来解决资源和体积有限的设备中高速 FFT 实现的功耗和散热问题; 另一方面充分利用 TOC 数据位众多的优势, 通过减少传统 FFT 算法对离散 Fourier 变换 (DFT) 的分解层数, 在大幅度提高并行程度的同时适当增加计算量, 从而从整体上提高计算速度. 由于 TOC 的数据位数众多且易增加, 因此适当增加计算量的方式对 TOC 而言问题不大.

本文介绍了 TOC 的特点和相应的加法、乘法运算过程, 分析了高度并行的 FFT 运算算法和全并行的 DFT 运算算法在 TOC 上的实现方案及并行层次, 同时分析了它们所需的时钟周期及硬件资源, 进行了模拟实验验证, 最后与同类电子实现方法进行了时钟周期数的对比.

2 相关研究

2.1 FFT 电子实现方案相关研究

电子计算机架构下提高 FFT 运算速度方面, 李小进 [23] 等使用 FPGA 实现了基 2 按时域抽取 (DIT) 的 FFT 算法, 在 1072 个时钟周期内完成了 256 点的 FFT 运算; 赵丽丽 [24] 等使用 GPU 对 FFT 的运算进行了加速, 石长振 [3] 等采用基 4 FFT 算法, 使用 Xilinx 的 FPGA 构造了 4 个基 4 蝶形处理模块, 实现了 FFT 各层内蝶形变换的部分并行. 他们采用的数学方法均为传统 FFT 多层蝶形运算, 并在此基础上进行了层内蝶形变换的部分并行实现.

在传统 FFT 电子实现中存在单级、多级和并行几种不同的实现方法, 在硬件消耗最多、运算速度最快的并行方法中, 通过在硬件上构造全部或者部分蝶形变换运算模块, 使得 FFT 多级运算中每一级里的蝶形运算可以并行完成. 这种基于传统的 FFT 算法的硬件并行实现方法运算速度快, 但存在以下几个问题: 第一是采用传统 FFT 算法时, 多级运算之间存在先后顺序, 如在基 2 时域抽取 FFT 算法中, 一个 1024 点的 FFT 需要进行 10 层运算, 这 10 层运算之间无法并行; 第二是在运算大点数 FFT 时, 目前的硬件资源很难构造出全部的蝶形运算模块, 数据需要存储回馈; 第三是传统方法没有考虑到功耗和散热问题.

2.2 TOC 实现 FFT 的基础

本文中基于 TOC 的并行快速 Fourier 变换实现采用以复数向量矩阵乘为基本运算单元的算法, 以 TOC 中的加法和乘法运算为基础. 随着基于 TOC 的 MSD 加法器 [12~17] 的完成、矩阵向量乘 [20]

表 1　T, W, T′, W′ 和 M 运算的真值表

Table 1　Truth table for T, W, T′, W′, M transformations

a	b	T	W	T′	W′	M
−1	−1	−1	0	−1	0	1
−1	0	−1	1	0	−1	0
−1	1	0	0	0	0	−1
0	−1	−1	1	0	−1	0
0	0	0	0	0	0	0
0	1	1	−1	0	1	0
1	−1	0	0	0	0	−1
1	0	1	−1	0	1	0
1	1	1	0	1	0	1

的实现, 在 TOC 上并行实现快速 Fourier 变换的前提条件已经具备.

2.2.1　MSD 加法运算

MSD 数字系统 1961 年由 Avizienis 提出 [25], 任何一个实数 A 均可由 MSD 数表示如下:

$$A = \sum_i a_i 2^i, \tag{1}$$

其中 a_i 为 1, 0 或者 u. 在 TOC 中使用水平偏振光、垂直偏振光和无光 3 种光的状态来表示 1, u 和 0, 通过液晶和偏振片实现这 3 种状态之间的转变, 从而完成相应运算. 在 TOC 中 u 代表 −1, 本文接下来的内容中 u 均指代 −1.

MSD 加法使用 4 种逻辑运算, 分别称为 T, W, T′ 和 W′, 真值表如表 1 所示 (其中 a 和 b 两列为不同情况下的输入数据, T, W, T′ 和 W′ 列为经过相应真值表转换后的输出数据). 利用 MSD 数的冗余性, 只要运算数据的位数不超过 TOC 的运算器位数, 一次 MSD 加法运算只需 3 步即可完成, 3 个步骤如下:

(1) 对输入加数和被加数同时进行 T, W 运算, T 运算的结果左移一位, 第 0 位补 0;

(2) 对第一步的结果同时进行 T′, W′ 运算, T′ 运算的结果左移一位, 第 0 位补 0;

(3) 对第二步的结果进行 T 运算, 得到最终结果.

2.2.2　MSD 乘法运算

两个一位的 MSD 数的乘法对应的逻辑运算称为 M, 真值表如表 1 所示. 当 n 位数据 $A = a_{n-1}\cdots a_1 a_0$ 和 $B = b_{n-1}\cdots b_1 b_0$ 相乘时, 乘数 B 中第 i 位与被乘数 A 中每一位分别进行 M 运算, 得到的结果左移 i 位, 这个结果称为乘法运算的部分积, 所有 n 个部分积累加得到的结果即为乘法运算的最终结果.

n 个部分积累加的过程使用二叉迭代法完成, 其运算步骤如下:

(1) 把相邻的两个部分积输入一个 MSD 加法器上完成运算, n 为偶数时需要 $n/2$ 个加法器, n 为奇数时需要 $n/2-1$ 个加法器, 加法器输出的结果称为部分和;

(2) 将相邻两个部分和输入一个 MSD 加法器上完成运算, 输出作为下一步运算中的部分和;

(3) 重复第二步直到只剩下一个部分和, 即为 MSD 乘法结果.

3 基于 TOC 的并行 DFT 算法设计

3.1 全并行 DFT 运算

一个 N 点 m 位的离散 Fourier 变换可以表示为

$$X(k)=\sum_{n=0}^{N-1}x(n)\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}kn}=\sum_{n=0}^{N-1}x(n)W_N^{kn},\quad k=0,1,\ldots,N-1. \tag{2}$$

为便于分析, 接下来的内容假设 N 与 m 均为 2 的整数次幂.

式 (2) 中旋转因子 $W_N^r=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}r}=\cos(\frac{2\pi}{N}r)-\mathrm{j}\sin(\frac{2\pi}{N}r)$, $x(n)$ 为输入复数向量的第 n 维, $X(k)$ 为 DFT 的输出复数向量的第 k 维, 为了在 TOC 的实现过程中进行并行层次的分析, 我们将此式表示为复数矩阵向量乘的形式:

$$\begin{bmatrix}1 & 1 & 1 & \cdots & 1\\ 1 & W_N^{1\times1} & W_N^{1\times2} & \cdots & W_N^{1\times(N-1)}\\ 1 & W_N^{2\times1} & W_N^{2\times2} & \cdots & W_N^{2\times(N-1)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & W_N^{(N-1)\times1} & W_N^{(N-1)\times2} & \cdots & W_N^{(N-1)\times(N-1)}\end{bmatrix}\begin{bmatrix}x(0)\\ x(1)\\ x(2)\\ \vdots\\ x(N-1)\end{bmatrix}=\begin{bmatrix}X(0)\\ X(1)\\ X(2)\\ \vdots\\ X(N-1)\end{bmatrix}. \tag{3}$$

因为所有 $X(n)$ $(n=0,1,2,\ldots,N-1)$ 计算时步骤相同, 不失一般性, 取

$$X(2)=W_N^{2\times0}x(0)+W_N^{2\times1}x(1)+\cdots+W_N^{2\times(N-1)}x(N-1)=\sum_{k=0}^{N-1}W_N^{2\times k}x(k). \tag{4}$$

式 (4) 中含有 N 次复数乘法和 $N-1$ 次复数加法. 为了将式 (4) 进一步简化, 写出 $W_N^{2\times k}$ 和 $x(k)$ $(k=0,1,2,\ldots,N-1)$ 的复数形式如下:

$$W_N^{2\times k}=a_k+\mathrm{j}b_k,\quad x(k)=c_k+\mathrm{j}d_k. \tag{5}$$

式 (5) 中 a_k,b_k,c_k,d_k 均为实数, 复数乘法计算如下:

$$W_N^{2\times k}\times x(k)=(a_k+\mathrm{j}b_k)\times(c_k+\mathrm{j}d_k)=(a_kc_k-b_kd_k)+\mathrm{j}(b_kc_k+a_kd_k). \tag{6}$$

由式 (4) 和 (6) 可得

$$\begin{aligned}X(2)&=\sum_{k=0}^{N-1}[(a_kc_k-b_kd_k)+\mathrm{j}(b_kc_k+a_kd_k)]\\&=\left[\sum_{k=0}^{N-1}(a_kc_k)-\sum_{k=0}^{N-1}(b_kd_k)\right]+\mathrm{j}\left[\sum_{k=0}^{N-1}(b_kc_k)+\sum_{k=0}^{N-1}(a_kd_k)\right].\end{aligned} \tag{7}$$

在使用光学方法直接实现 DFT 时, 其并行层次有 3 层:

(1) 在式 (3) 中所有 $X(0),X(1),\ldots,X(i),\ldots,X(N-1)$ 均可并行, DFT 所需总时间为一次 $X(i)$ 的计算时间;

(2) 在式 (7) 中 $\sum_{k=0}^{N-1}(a_kc_k)-\sum_{k=0}^{N-1}(b_kd_k)$ 与 $\sum_{k=0}^{N-1}(b_kc_k)+\sum_{k=0}^{N-1}(a_kd_k)$ 可以并行计算, 而 $\sum_{k=0}^{N-1}(a_kc_k)$ 与 $\sum_{k=0}^{N-1}(b_kd_k)$、$\sum_{k=0}^{N-1}(b_kc_k)$ 与 $\sum_{k=0}^{N-1}(a_kd_k)$ 也可并行, 因此一次 $X(i)$ 的计算时间为一次 $\sum_{k=0}^{N-1}(a_kc_k)$ 与一次实数加/减法的计算时间;

(3) 在 $\sum_{k=0}^{N-1}(a_kc_k)$ 中 $a_0c_0, a_1c_1, \ldots, a_{N-1}c_{N-1}$, 均可并行, 因此一次 $\sum_{k=0}^{N-1}(a_kc_k)$ 的计算时间为一次实数乘法和 $N-1$ 次实数加法的计算时间.

通过上述并行层次的分析可得: 一次 DFT 运算在全并行的情况下的运算时间为一次实数乘法和 N 次实数加法的总运算时间. 由于在 TOC 中采用 MSD 加法 [12], 加法运算时间不随加数或者被加数的长度而改变, 相对于电子计算机而言, TOC 在处理位数极大的数据时拥有很大的时间优势. 与此同时, TOC 位数众多且易于扩展, 具备并行实现 DFT 的硬件条件.

在 DFT 的三值光学实现中, 其相关快速算法 (FFT) 中含有迭代过程, 由于迭代过程中计算存在先后顺序, 因此不能实现完全并行, 但是基于 FFT 的实现对液晶数量的需求会大大减小, 运用基 q 时域抽取 FFT 算法可以将 N 点 DFT 分解成多个 N/q 点 DFT, 结合复数矩阵向量乘的流水线技术能够在不增加过多时钟周期的前提下用更少的液晶完成 DFT, 这更利于在目前的三值光学实验平台上完成较多点数的 DFT 运算, 因此对 FFT 的并行算法进行了研究.

3.2 改进的并行 FFT 运算

采用基 2 时域抽取 FFT 运算:

$$\begin{aligned} &X(k)=X_1(k)+W_N^kX_2(k),\\ &X\left(k+\frac{N}{2}\right)=X_1(k)-W_N^kX_2(k), \end{aligned} \qquad k=0,1,\ldots,N/2-1, \tag{8}$$

其中 $X_1(k)$ 表示次序号为偶数的输入数据的 DFT 结果, $X_2(k)$ 表示次序号为奇数的输入数据的 DFT 结果. 其中 $X_1(k)$ 和 $X_2(k)$ 均为 $N/2$ 维复数向量与 $N/2$ 阶复数方阵的乘法, 将式 (8) 右侧的 $W_N^kX_2(k)$ 展开可得

$$W_N^kX_2(k)=\sum_{r=0}^{\frac{N}{2}-1}x(2r+1)\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}k(2r+1)}. \tag{9}$$

由式 (9) 可知: 当需要计算的 DFT 的点数已知时, 所有由旋转因子组成的矩阵是已知的, 它们相互之间的运算是确定的, 其运算结果在 FFT 开始前就能够求得, 将其看做为参数矩阵. 因此在分析并行 FFT 计算量时, 本文不考虑所有由旋转因子组成的矩阵之间的运算.

由以上分析可知 N 点的 DFT 可看做是两次 $N/2$ 维复数向量与 $N/2$ 阶复数方阵的乘法、一次 $N/2$ 维复数向量加法及一次 $N/2$ 维复数向量减法. 基 2 FFT 可以继续应用到 $X_1(k)$ 和 $X_2(k)$ 上, 此时为了去除迭代, 我们将 $X(k)$ 展开为 4 个 $N/4$ 阶的复数矩阵向量乘运算, 它们可以并行执行, 代价是计算量增大, 如继续对 $X_1(k)$ 和 $X_2(k)$ 进行一次分解, 非迭代的运算方式下需要 4 次 $N/4$ 维复数向量与 $N/4$ 阶复数方阵的乘法、8 次 $N/4$ 维复数向量加/减法, 此时的运算量大于进行一次基 4 FFT 所需的运算量.

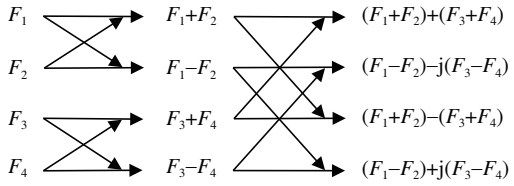

图 1 基 4 FFT 中复数向量加/减法步骤

Figure 1 Operating procedures of plural vector addition/subtraction in Radix-4 FFT algorithm

表 2 不同并行 FFT 算法的计算量 (对 DFT 进行一次分解时)

Table 2 Computation cost of different parallel FFT algorithms (divide DFT once)

Algorithm	Computation	
	Multiplication of a $N/2$-dimensional complex vector and a complex matrix of order $N/2$	Addition/Subtraction of $N/2$-dimensional complex vectors
Parallel Radix-2 FFT	2 times	2 times : 1 layer
Parallel Radix-4 FFT	4 times	8 times : 2 layers; number of addition or subtraction in each layer are 4, 4
Parallel Radix-8 FFT	16 times	28 times : 3 layers; number of addition or subtraction in each layer are 12, 8, 8

采用基 4 时域抽取 FFT 运算:

$$\begin{aligned}
&X(k) = X_1(k) + W_N^{2k}X_3(k) + W_N^kX_2(k) + W_N^{3k}X_4(k),\\
&X\left(k+\frac{N}{4}\right) = X_1(k) - W_N^{2k}X_3(k) - \mathrm{j}\left[W_N^kX_2(k) - W_N^{3k}X_4(k)\right],\\
&X\left(k+\frac{N}{2}\right) = X_1(k) + W_N^{2k}X_3(k) - \mathrm{j}\left[W_N^kX_2(k) + W_N^{3k}X_4(k)\right],\\
&X\left(k+\frac{3N}{4}\right) = X_1(k) - W_N^{2k}X_3(k) + \mathrm{j}\left[W_N^kX_2(k) - W_N^{3k}X_4(k)\right],
\end{aligned} \qquad k=0,1,\ldots,N/4-1, \tag{10}$$

其中 $X_1(k), X_2(k), X_3(k), X_4(k)$ 表示 $N/4$ 点的 DFT 运算. 通过与式 (9) 类似的分析可知: N 点的 DFT 可看作 4 次 $N/4$ 维复数向量与 $N/4$ 阶复数方阵的乘法、8 次 $N/4$ 维复数向量加/减法. 其 4 次复数矩阵向量乘分别为 $X_1(k), W_N^{2k}X_3(k), W_N^kX_2(k), W_N^{3k}X_4(k)$, 分别设其结果为复数向量 F_1, F_2, F_3, F_4, 8 次复数向量加/减法的过程见图 1, 其中包含 8 次加/减法, 共 2 层运算.

采用基 q 时域抽取 FFT 运算:

$$X\left(k+l\frac{N}{q}\right) = \sum_{u=0}^{q-1}\left[W_N^{uk}W_N^{ul}X_{u+1}(k)\right], \quad k=0,1,\ldots,\frac{N}{q-1}, \quad l=0,1,\ldots,q-1, \tag{11}$$

其中 $X_{u+1}(k)$ 表示 N/q 点的 DFT 运算. 在基 4 或者基 2 运算中 W_N^{ul} 仅为 1, -1, j 或者 $-$j, 在 $q>4$ 时 W_N^{ul} 开始出现实部虚部均不为 0 的情况, 根据本节前面的分析, 将 $W_N^{uk}W_N^{ul}X_{u+1}(k)$ 看作一个参数复数矩阵与一个输入复数向量间的乘法运算.

表 2 分析了仅对 DFT 分解一次时不同基 FFT 运算所需的计算量, 其中多个复数矩阵向量乘可以使用同一套硬件系统流水线完成, 而其中的每个复数向量加/减法中各维可以并行, 多个复数向量加/减法部分可并行.

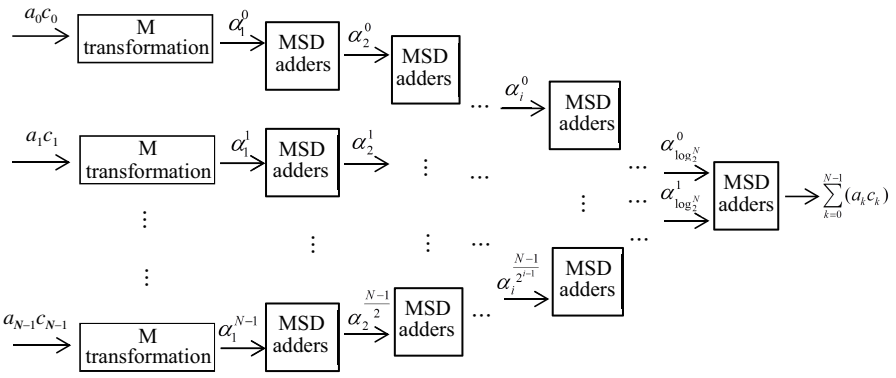

图 2　连续乘加运算的步骤 (α_k^j 为第 k 层第 j 个中间结果向量)

Figure 2　The process of continuous multiplication and addition (α_k^j stands for the j-th intermediate result vector in the k-th step)

4　基于 TOC 的并行 FFT 算法实现与分析

由于此处采用的并行 FFT 算法的实质是将一个 N 点 DFT 运算 (N 阶复数矩阵与 N 维复数向量的乘法) 分解为多个 N/q (其中 q 为 2 的整数次幂) 阶复数矩阵与 N/q 维复数向量的乘法运算, 而复数矩阵向量乘的实现与并行 DFT 运算的实现完全相同, 因此在分析并行 FFT 算法的实现时, 本文首先分析并行 DFT 基于 TOC 的实现, 然后分析并行 FFT 的三值光学实现方法.

4.1　并行 DFT 运算的实现

结合 3.1 小节的分析, 一次 DFT 变换的实现流程如下所述.

(1) 将 $x(0), x(1), \ldots, x(i), \ldots, x(N-1)$ 编码输入.

(2) 并行计算出 $X(0), X(1), \ldots, X(i), \ldots, X(N-1)$, 以其中一项 $X(i)$ 为例:

(2-1) 以流水线的方式计算出 $\sum_{k=0}^{N-1}(a_kc_k), \sum_{k=0}^{N-1}(b_kd_k), \sum_{k=0}^{N-1}(b_kc_k), \sum_{k=0}^{N-1}(a_kd_k)$, 以 $\sum_{k=0}^{N-1}(a_kc_k)$ 为例, 此部分与文献 [21] 中运算形式相似, 见图 2, 步骤如下:

A. 部分积生成. 通过 $N \times m$ 个 M 变换器生成所有 a_kc_k $(k = 0, 1, \ldots, N-1)$ 的部分积, 对于 N 点 m 位的 DFT, $\sum_{k=0}^{N-1}(a_kc_k)$ 运算中将生成 $N \times m$ 个部分积.

B. 部分积相加. 使用二叉迭代法将所有 $N \times m$ 个部分积相加得到 $\sum_{k=0}^{N-1}(a_kc_k)$ 的最终结果.

(2-2) 计算出 $X(i)$ 的实部 $\sum_{k=0}^{N-1}(a_kc_k) - \sum_{k=0}^{N-1}(b_kd_k)$ 和虚部 $\sum_{k=0}^{N-1}(b_kc_k) + \sum_{k=0}^{N-1}(a_kd_k)$.

(3) 解码输出结果.

为了在硬件数量和运行速度之间进行平衡, 对于前面第 (2-1) 步, 我们采取流水线方案而非全并行方案, 基于此, 计算 DFT 中一项 $X(i)$ 的操作时序如图 3, 其中 M 代表 M 变换, 需 1 个时钟周期. 第 i $(i = 1, \ldots, \log_2 N)$ 步加法代表 MSD 3 步式加法, 需 3 个时钟周期. 相对于全并行方案, 流水线方案的硬件需求为其 1/4, 时钟周期数仅增加了 3.

4.2　并行 DFT 三值光学实现的时钟周期及液晶数目分析

结合 4.1 小节的分析, 针对 N 点 m 位的 DFT 运算, 对于每一维 $X(i)$ 的运算过程:

(1) 分析其所需 M 变换器和 MSD 加法器的个数. 完成连续乘加运算 $\sum_{k=0}^{N-1}(a_kc_k)$ 的硬件: M 变换器 $N \times m$ 个, MSD 加法器 $\log_2^{N\times m}$ 层, 第 x 层 MSD 加法器数目为 $N \times m/2^x$. 需要计算

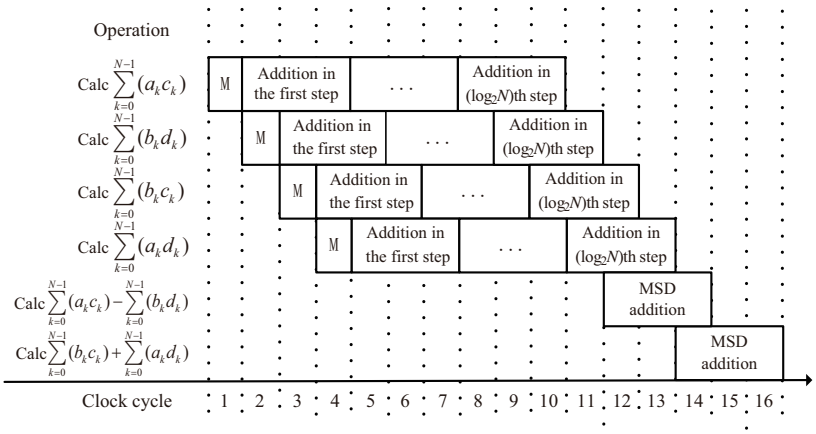

图 3 计算 DFT 中一项 $X(i)$ 时的操作时序

Figure 3 Sequence diagram of pipeline scheme to compute $X(i)$

$\sum_{k=0}^{N-1}(a_kc_k)-\sum_{k=0}^{N-1}(b_kd_k)$ 和 $\sum_{k=0}^{N-1}(b_kc_k)+\sum_{k=0}^{N-1}(a_kd_k)$ 的硬件: MSD 加法器 1 个. 因此共需 M 变换器 $N\times m$ 个, MSD 加法器 $N\times m$ 个.

(2) 分析单个 M 变换器及 MSD 加法器所需液晶数. 单个 M 变换器的液晶数目为 m. 经过 M 变换输入到第一层 MSD 加法的数据位数为 $2m-1$, 每层 MSD 加法会让结果位数增长 2 位, 第 i 层中 MSD 加法器输入的数据位数 P_i 为 $2m-1+2(i-1)$, 因此不同层中 MSD 加法器位数互不相同, 一个 MSD 加法器件中存在 T, W, T′, W′, T 5 种变换, 其中第 i 层 MSD 加法器中 T, W, T′, W′, T 5 个变换器的位数分别为 $P_i, P_i, P_i+1, P_i+1, P_i+2$, 所需液晶数目为 $5P_i+4$, 即 $10m+10i-11$, 最后计算 $X(i)$ 实部和虚部的 MSD 加法器所需液晶为 $10m+10\log_2^{N\times m}-1$.

(3) 分析总的液晶数目. M 变换器的总液晶位数为 $N\times m\times m$.

MSD 加法器的总位数为

$$\left(10m+10\log_2^{N\times m}-1\right)+\sum_{i=0}^{\log_2^{N\times m}}\left[(N\times m/2^i)(10m+10i-11)\right].$$

因此, 计算 $X(i)$ 所需总的液晶数 C_1 应为

$$C_1=N\times m\times m+\left(10m+10\log_2^{N\times m}-1\right)+\sum_{i=0}^{\log_2^{N\times m}}\left[(N\times m/2^i)(10m+10i-11)\right]. \tag{12}$$

(4) 分析时钟周期. M 变换需要 1 个时钟周期, $\log_2^{N\times m}$ 层 MSD 加法需要 $3\log_2^{N\times m}$ 个时钟周期, 流水线运行需要增加 3 个时钟周期, 最后计算 $X(i)$ 实部和虚部的 MSD 加/减法需要 3 个时钟周期, 总的时钟周期 T_1 应为

$$T_1=7+3\log_2^{N\times m}. \tag{13}$$

对于整个 DFT 运算, 共有 N 维需要并行计算的 $X(i)$, 因此 N 点 DFT 计算中液晶数目 $C_{\rm D}^N$ 和时钟周期 $T_{\rm D}^N$ 分别为

$$C_{\rm D}^N=N\times C_1=N^2\times m^2+N\left\{\left(10m+10\log_2^{N\times m}-1\right)+\sum_{i=0}^{\log_2^{N\times m}}\left[(N\times m/2^i)(10m+10i-11)\right]\right\}, \tag{14}$$

$$T_{\mathrm{D}}^{N}=T_{1}=7+3\log_{2}^{N\times m}. \tag{15}$$

最终结果里每一维的实部/虚部的位数 L_N 为

$$L_{N}=2m+2\log_{2}^{N\times m}+1. \tag{16}$$

对于 1024 点 16 位的并行 DFT 运算, 所需时钟周期数为 49, 液晶数目为 3103774720, 并行 DFT 运算时钟周期数极小, 但是液晶数较多, 以目前的硬件条件而言实现较为困难, 因此基于并行 DFT 运算, 本文对并行 FFT 的光学实现进行了分析.

4.3 并行 FFT 运算的实现

基于 3.2 小节的分析, 并行 FFT 运算将 N 点的 DFT 运算分解为多个阶数较小的复数矩阵向量乘的运算, 因此我们仅需构建一套实现较小阶数复数矩阵向量乘运算的三值光学模块, 流水线完成多个相同规模的复数矩阵向量乘运算. 单个复数矩阵向量乘的运算模块与并行 DFT 运算模块完全相同, 这部分在 4.1 小节中已经讨论. 本节主要研究将已得到的复数矩阵向量乘的结果组合成为最终结果的方法.

以基 4 时域抽取 FFT 为例:

采用如图 1 所示运算次序, 首先使用 4.1 小节中的方法通过流水线技术计算出 F_1, F_2, F_3 和 F_4, 对于 N 点 DFT 运算而言, 此时 F_1, F_2, F_3 和 F_4 均为 $N/4$ 维复数向量, 然后使用 $2N$ 个 MSD 加法器同时完成两对复数向量的加/减法, 最后再次使用 $2N$ 个 MSD 加法器同时完成第二层的两对复数向量加/减法, 从而得到最终结果.

光学实现模块见图 4, 其中每个 Adders 模块代表 $N/4$ 个 MSD 加法器; F_i $(i=1,2,3,4)$ 实部/虚部代表由 $N/4$ 维复数向量中的实部/虚部组成的 $N/4$ 维实数向量; $X(k+jN/4)$ $(j=0,1,2,3)$ 实部/虚部代表 $k=0,1,2,\ldots,N/4-1$ 时由最终结果第 $k+jN/4$ 项的实部/虚部组成的 $N/4$ 维实数向量.

对于基 q 时域抽取 FFT 算法, 其计算流程和光学实现方式与基 4 时域抽取 FFT 相同, 区别仅在于计算量和计算层次的不同. 为选取合适的实现算法, 下面分析采用基 4 时域抽取 FFT 算法时所需的时钟周期和液晶数目, 并对比不同基 FFT 算法所需的时钟周期和液晶数目.

4.4 并行 FFT 三值光学实现的时钟周期及液晶数目分析

结合 3.2 小节对于基 4 时域抽取 FFT 算法的分析, 按照其实现流程分析所需时钟周期和液晶数目如下:

(1) 使用流水线技术得到 4 个 $N/4$ 阶复数矩阵向量乘的运算结果 F_1, F_2, F_3 和 F_4 (参看 3.2 小节). 由于在计算单个复数矩阵向量乘的过程中已经使用了流水线技术 (图 3), 因此相对于单个 $N/4$ 阶复数矩阵向量乘运算, 计算 4 个 $N/4$ 阶复数矩阵向量乘时将会增加 3×4 个时钟周期, 液晶数目不增加.

(2) 将 F_1, F_2, F_3 和 F_4 组合得到最终结果 (图 4). 需要两层共 $2\times2N$ 个 MSD 加法器, 各层中单个 MSD 加法器的液晶位数分别为 $5L_{N/4}+4$ 和 $5(L_{N/4}+2)+4$, 因此组合过程将会增加 $4N$ $(5L_{N/4}+9)$ 位液晶, 时钟周期数增加 3×2.

由此得到基于 TOC 的 N 点基 4 时域抽取 FFT 算法所需液晶数目 C_{F}^{N} 和时钟周期 T_{F}^{N} 分别为

$$C_{\mathrm{F}}^{N}=C_{\mathrm{D}}^{N/4}+4N(5L_{N/4}+9), \tag{17}$$

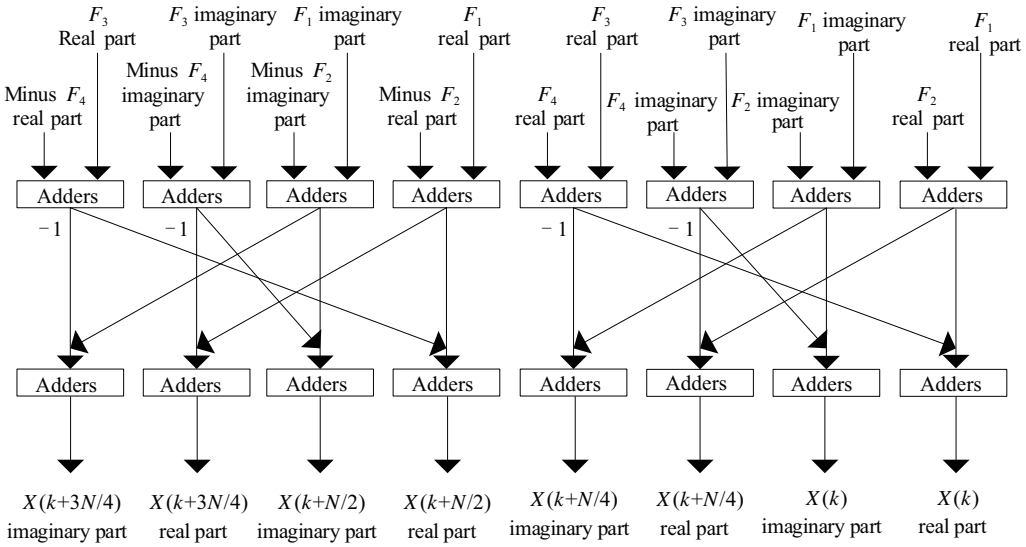

图 4 基 4 时域抽取 FFT 并行光学实现 (一个加法模块代表 $N/4$ 个 MSD 加法器)

Figure 4 Radix-4 DIT-FFT algorithm implementation in TOC (one adders module stands for $N/4$ MSD adders)

表 3 TOC 中不同基 FFT 算法所需时钟周期和液晶数目分析

Table 3 Data bits and clock cycles required by different FFT algorithms implemented on TOC

Algorithm	Clock cycles	Data bits
Parallel DFT	49	3103774720
Parallel Radix-2 FFT	53	776553472
Parallel Radix-4 FFT	61	195188224
Parallel Radix-8 FFT	109	50556672

$$T_{\mathrm{F}}^{N} = T_{\mathrm{D}}^{N/4} + 12 + 6 = 19 + 3\log_2 N \times m. \tag{18}$$

对于 1024 点 16 位的并行基 4 时域抽取 FFT 运算, 所需时钟周期数为 61, 液晶数目为 195188224, 相对于同等规模的并行 DFT 运算, 时钟周期数增加了 8, 但液晶数目减小为 DFT 的 1/4.

基于以上分析, 计算 1024 点 16 位 DFT 时不同基时域抽取 FFT (非迭代形式, 即仅将 DFT 分解一次) 所需的时钟周期和液晶数目, 结果见表 3.

5 模拟实验

5.1 实验描述

由于 TOC 应用研究试验系统 SD11 仍在建设过程中. 本文采用 TOC 的模拟实验平台进行实验, 以验证并行 FFT 算法及其实现策略的正确性. 实验模拟了三值光学计算机中 MSD 加法器、MSD 乘法器和 MSD 数表示下的基 4 和基 8 FFT 运算器. 模拟实验平台硬件环境为: CPU: Intel(R) Core(TM) i5 430 M 2.26 GHz, 内存: 2 GB, 操作系统: 64 位 Ubuntu 14.04, 编译器: g++ 4.8.4. 使用 C++ 编写模拟程序, 本文的实验平台基于文献 [26] 中的实验软件开发而成.

使用字符来表示 MSD 数, 字符 '1' 表示 1, '0' 表示 0, 'u' 表示 −1, 使用 map⟨char, map⟨char,

char⟩⟩ 来存放 MSD 数快速 Fourier 变换所需的真值表, 使用查表的方法模拟 TOC 中各个变换器的功能. 实验详细设计如下:

(1) 变量定义.

(1-1) 定义整型变量 FFT_N, MSD_m 和 MSD_p, 分别表示 FFT 运算的点数, 单个输入 MSD 数的总位数和 MSD 数的小数部分位数;

(1-2) 定义 map⟨char, map⟨char, char⟩⟩ 型变量 T, W, $T1$, $W2$, M 分别表示 MSD 加法过程的 4 个真值表以及 MSD 乘法过程的 M 真值表;

(1-3) 定义二维字符串数组 InputReal[8][FFT_N/8] 和 InputImag[8][FFT_N/8] 分别用于基 8 FFT 运算中存储输入 MSD 数据的实部和虚部;

(1-4) 定义三维字符串数组 TwiddleFactorMatrixReal[16][FFT_N/8][FFT_N/8] 和 TwiddleFactor MatrixImag[16][FFT_N/8][FFT_N/8] 用于存储表 1 中分析得出的基 8 FFT 需要进行的 16 次复数向量矩阵乘中的复数矩阵部分, 这部分矩阵由旋转因子相互运算得到, 在确定 FFT_N 之后即可计算得到这部分数据;

(1-5) 定义二维字符串数组 First_Real[16][FFT_N/8] 和 First_Imag[16][FFT_N/8] 用于存储基 8 FFT 运算中 16 次复数向量矩阵乘得到的 16 个 MSD 复数向量.

(2) 数据输入. 实验过程中随机生成了 128 个 32 位长的 MSD 数, 每两个 MSD 数为一组, 分别表示输入复数的实部和虚部. 使用基 8 FFT 算法时, 将输入数据分为 8 组存到数组 InputReal 和 InputImag 中.

(3) 构造函数 MSDAdder() 模拟 MSD 加法器, 此函数使用 T, W, $T1$ 和 $W2$ 4 个真值表, 基于查表的方式、按照文献 [14] 的运算过程完成两个 MSD 数的加法操作, 返回两个 MSD 加法结果.

(4) 构造函数 MSDMulti() 模拟 MSD 乘法过程, 此函数使用真值表 M 得到所有部分积, 然后使用 MSDAdder() 函数完成部分积相加, 返回两个 MSD 数乘法运算结果.

(5) 构造函数 CalTwiddleFactorMatrix() 完成 TwiddleFactorMatrixReal 和 TwiddleFactorMatrix Imag 的赋值, 主要工作为计算正弦函数和余弦函数的结果并将其转化为字符串表示的 MSD 数.

(6) 构造函数 Com_Radix_8_FFT_MSD() 模拟分解一次的基 8 FFT 运算.

(6-1) 将输入 MSD 数分为 8 组存到 InputReal 和 InputImag 数组中;

(6-2) 调用 MSDMulti() 函数计算 16 次复数向量矩阵乘的结果, 即 TwiddleFactorMatrixReal, TwiddleFactorMatrixImag 表示的 MSD 复数矩阵与 InputReal, InputImag 表示的 MSD 复数向量相乘所得中间结果, 模拟流水线进行 MSD 复数向量矩阵乘的过程.

(6-3) 对上一步得到的结果 First_Real 和 First_Imag 向量进行基 8 FFT 对应的 3 层 MSD 复数向量加、减法, 调用 MSDAdder() 函数完成相应操作, 返回基 8 FFT 运算结果.

通过与基于 TOC 的基 8 FFT 运算模拟相似的程序设计方法, 本文同时实现了基 4 FFT 的运算模拟.

5.2 实验结果

本实验对基于 TOC 的 64 点 32 位的 FFT 运算进行模拟.

随机生成了 128 个 32 位长的 MSD 数, 每两个 MSD 数为一组, 分别表示输入复数的实部和虚部, 设置 MSD 数中 24 位表示小数部分, 8 位表示整数部分. 此部分作为实验用例.

为了验证 TOC 中实现 FFT 运算的正确性, 采用电子计算机中通用的基 2 时域抽取 FFT 计算方法运算, 对比结果. 此部分将生成的 MSD 数转化为十进制数作为输入. 实验用例见表 4 (其中 MSD

表 4 模拟实验输入数据前 6 维

Table 4 First six dimensions of simulation experiment inputs

Input data		MSD binary system	Decimal system
1	Real part	$10u10uuu01u0uu1u0u1100uuuu1uu10u$	105.0817
	Imaginary part	$1110111101u0u11000110u1u01u0u100$	239.1178
2	Real part	$u1100uu010u0u0u1100uuu011u011000$	−37.6586
	Imaginary part	$u0u000uu0u0u10010u110u0uu1uu01uu$	−163.2776
3	Real part	$00u01u1u00110uu1uu11u0u0u01uu0uu$	−26.8343
	Imaginary part	$110uuu0uu0u0u10u1uu111u011u1u111$	162.3563
4	Real part	$0u00u1uu100uu0u11uu0010uuu11u011$	−70.5971
	Imaginary part	$1u0011u110000uuu0u1uu01u00u1110u$	75.4718
5	Real part	$010u0u00u0u00111u01uuu10001u10u0$	43.4004
	Imaginary part	$u0u1u0u0u1uu1u0u1001u0u0100u10uu$	−154.4237
6	Real part	$011u01uuu001u01u1u1uu0u00uu001u1$	80.5362
	Imaginary part	$0011u1u0u11u1u0u01uuuu0111uu1111$	41.8243

表 5 实现并行基 4 FFT 的模拟实验结果

Table 5 Outputs of simulation experiment to implement parallel Radix-4 FFT

Output data		Ternary optical computer		Electronic computer	Result
		MSD binary system	Decimal system		
1	Real part	$u100u01u001u0u10u101uu10000u10u01u$	−283.7847	−283.7847	Correct
	Imaginary part	$1u01100uu01u000001u00100u110u0$	21.6407	21.6407	
2	Real part	$uu10000100u0010u011u00000001u00u00$	−636.4482	−636.4482	Correct
	Imaginary part	$1u0u01u10u00uu1u0011u00u1u00100100u1$	811.3220	811.3220	
3	Real part	$1uu10u1u00010u0u10uu01000u0101u11u1$	720.7079	720.7079	Correct
	Imaginary part	$1u0u11u10u01u0u10u0000u0u0100000010$	470.2106	470.2106	
4	Real part	$1uu1u01u000100u100u0010uu101u00u1$	82.1162	82.1162	Correct
	Imaginary part	$u10u00100uu10u1u0u010u1000u1001u1u0$	−626.5998	−626.5998	
5	Real part	$u011u0u100000uu1000u00u001u00u1u0$	−178.0396	−178.0396	Correct
	Imaginary part	$1uu1u10u11u00001u10uu110u0000001u00u$	698.0423	698.0423	
6	Real part	$1u0000u0010u01000100u1000u10u010u$	124.4080	124.4080	Correct
	Imaginary part	$u10000u100010u01u001u010u1u011uu100$	−519.6082	−519.6082	

数中的 u 表示 -1), 基 4 FFT 模拟实验结果见表 5, 基 8 FFT 模拟实验结果见表 6, 限于篇幅, 仅显示模拟实验中输入输出数据的前 6 维. 得到的模拟实验结果与电子计算机的 FFT 计算结果一致, 这表明本文提出的基于 TOC 的 MSD 数的基 8 及基 4 FFT 并行运算功能正确.

6 与现有方法对比

6.1 算法对比

本文所给出的并行 FFT 算法基于 TOC 平台, 该平台具有数据位众多、并行计算潜力大的特点, 本算法正是充分考虑并利用了光学计算高度并行的特点而改进提出的. 就此基础来说, 本算法与传统 FFT 算法的区别有如下几点:

表 6　实现并行基 8 FFT 的模拟实验结果

Table 6　Outputs of simulation experiment to implement parallel Radix-8 FFT

Output data		Ternary optical computer		Electronic computer	Result
		MSD binary system	Decimal system		
1	Real part	$u10u100100010u100u001u10000u10u01u$	−283.7847	−283.7847	Correct
	Imaginary part	$1u10u00uu00100000001001000u10u0$	21.6407	21.6407	
2	Real part	$u1u1000010u100100uu010000000010u100$	−636.4482	−636.4482	Correct
	Imaginary part	$10u010uu10u011u01u1u00u1u0011uu00u1$	811.3220	811.3220	
3	Real part	$110u1u00010u0u1u0101000u01010u1u01$	720.7079	720.7079	Correct
	Imaginary part	$1u0u11u10u01u00u0u0000u0u01000001u0$	470.2106	470.2106	
4	Real part	$1u0101u000100u10u100011u101u000u$	82.1162	82.1162	Correct
	Imaginary part	$u10u1uu00uu10u0100uu00u000u1001u010$	−626.5998	−626.5998	
5	Real part	$u1u010u10000u10u0u1100u001u00u1u0$	−178.0396	−178.0396	Correct
	Imaginary part	$1u10u00u0100001u1u10u10u000000010u1$	698.0423	698.0423	
6	Real part	$10000u01u0u01000100u1000u10u01u1$	124.4080	124.4080	Correct
	Imaginary part	$u100000u001uu1001001u1uu11u1u1uu100$	−519.6082	−519.6082	

(1) 为充分利用 TOC 的并行计算潜力, 通过对复数矩阵向量乘法多层并行性的分析 (见 3.1 小节), 本文提出的并行 FFT 算法选择复数矩阵向量乘作为基本运算单元, 这与传统 FFT 算法以蝶形运算作为基本运算单元有所不同, 而这一点是由 TOC 数据位众多的特点所支撑的. 举例而言, 在使用本文提出的并行基 4 FFT 算法时, 通过对 DFT 运算进行单次分解 (见 3.2 小节), 以 4 个复数矩阵向量乘作为基本运算单元, 仅通过两层运算就可以完成 1024 点的 FFT, 而在传统基 2 FFT 算法中, 需要 10 层串行的蝶形运算才能得到最终结果, 即使使用传统的基 4 FFT 算法, 也需要 9 层串行的蝶形运算.

(2) 不可否认的是本算法减少时钟周期的同时, 在一定程度上增大了计算量, 但是利用传统算法的分解思想 (见 3.2 小节) 和流水线的实现技巧 (见 4.1 小节), 通过对所需硬件资源的详细分析 (见 4.4 小节以及表 3) 可以看到: 本文提出的算法在极大缩短时钟周期的同时把所需硬件资源的数量控制在了 TOC 平台可接受的范围内, 而这一并行方案对于基于电子计算机的传统 FFT 算法来说在目前是很难实现的.

(3) 基础平台不同导致算法性能存在差异, 本文中的并行 FFT 算法所使用的 MSD 乘法由三值逻辑运算中的 M 运算产生部分积, 使用无进位的 MSD 加法实现部分积相加, 而传统方法中乘法采用位与逻辑电路或者 Booth 编码电路产生部分积, 使用二进制加法实现部分积相加, 不可避免地存在进位延时问题. 这一区别使得本文的算法在处理大点数的 FFT 运算时时间优势更为明显.

6.2　实现效果对比分析

为详细分析基于 TOC 的 FFT 运算与传统电子方法实现 FFT 运算的性能, 与文献 [4~6,23,27,28] 中的电子实现方法进行时钟周期的对比 (部分比较使用了文献 [6] 的数据对比结果), 结果见表 7 和图 5, 图 5 中采用的计算方法和所计算的 FFT 的点数和位数信息对应表 7 中的相应数据, 其中电子方法的数据来自表 7 中 6 种电子实现方法中的最小时钟周期数 (6 种电子实现方法的时钟周期数据均收集自相应参考文献, 由于文献中没有提供所有全面的实验数据, 因此表中存在部分空缺).

从图 5 中可以看出, 当 FFT 运算的点数指数增长时, 基于 TOC 的运算方法所需的时钟周期数仅

表 7 与 FFT 电子实现方法所需时钟周期数对比

Table 7 Clock cycles comparison versus electronic methods

FFT design	Transform time (cycles) FFT size				Device
	16-bit 16-point	16-bit 64-point	16-bit 256-point	16-bit 1024-point	
Electronic method 1 [4]	37	137	525	2065	Spartan-3
Electronic method 2 [5]	–	–	–	1283	XCV2P30
Electronic method 3 [6]	361	–	929	448805	EP2C35
Electronic method 4 [23]	–	–	1072	–	Vertex-II
Electronic method 5 [27]	–	–	255	–	EP3SL70
Electronic method 6 [28]	–	–	255	–	5VSX35T
Minimum clock cycle in electronic methods	36	137	255	1283	–
Our parallel Radix-4 FFT method	43	49	55	61	TOC
Our parallel Radix-8 FFT method	91	97	103	109	TOC

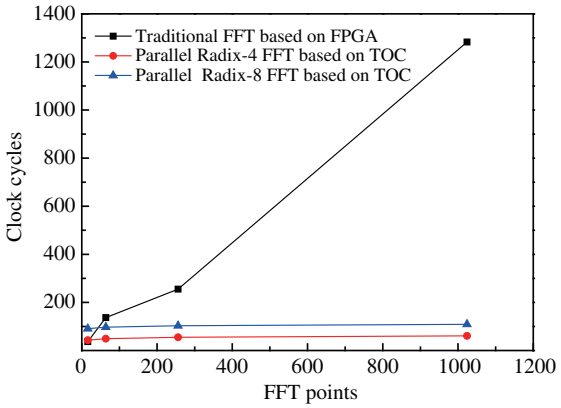

图 5 (网络版彩图) 不同 FFT 计算方法所需时钟周期数对比

Figure 5 (Color online) Comparison of the clock cycles required for different FFT algorithms

线性增长, 这是因为基于 TOC 的 FFT 运算的时钟周期数由乘法过程中部分积累加的二叉迭代次数决定, 当乘数位数指数增长时, 其部分积累加时的二叉迭代层数仅会线性增长, 这一特性使得相比于 FFT 的电子实现方法, 基于 TOC 的 FFT 运算方法对点数较大的 FFT 具有极大的时间优势.

在能耗方面, 对于 FFT 运算而言, 不论运行平台是 FPGA 还是 TOC, 其能耗都分为控制传输部分能耗和运算阶段产生的能耗, 对于数据的控制传输部分而言, 两者都是采用低功耗嵌入式系统, 能耗大致相当. 在运算阶段, 对于 FPGA 等电子运算器件而言, 能耗主要是集中在计算部分, 这部分时间较长, 能量消耗大, 产生的热量集中, 并且随着 FFT 运算点数的增加, 计算量大幅度增加, 相应所需的能耗和产生的热量也会大大增加. 对于以光学器件作为运算部件的 TOC 而言, 其空间调制器在改变光状态时耗能极小, 使得光学计算近乎于零散热解决方案, 而且, 由于光子信息容量大, 并行度高, FFT 运算时点数增加所带来的能量消耗增加量几乎可以忽略不计. 研究表明, 百万像素液晶器件的能耗仅为毫瓦级, 它作为空间调制器构建光学处理器, 其能耗是具有相同计算能力电子 CPU 组的千分之一. 从这个意义上讲, 尽管 TOC 目前尚处于原型系统阶段, 部分过程尚无法进行定量功耗分析, 但是相对

于 FPGA 在进行 FFT 运算, 特别是大规模 FFT 运算时, 能耗消耗具有较大的潜在优势.

7 总结

本文针对空间和资源受限的嵌入式设备中的高速 FFT 实现需求, 设计了基于 TOC 的并行 FFT 算法和相应实现模块, 给出了该方法的运算步骤, 对其正确性通过模拟实验进行了证明, 通过对运算所需的时钟周期进行分析并与基于 FPGA 的 FFT 实现进行对比说明了本方法的有效性. 本文通过对基于 TOC 的 FFT 实现研究, 在新的视角下分析了提高 FFT 运算速度的实现方法, 为在空间和资源有限的嵌入式设备上, 如星弹载设备上实现高速 FFT 运算提出了新的解决方案.

致谢 感谢所有三值光学计算机团队成员, 尤其是徐群博士研究生对本文所做出的贡献.

参考文献

1 James W C, John W T. An algorithm for the machine calculation of complex Fourier series. Math Comput, 1965, 19: 297–301
2 Chang Y N, Parhi K K. An efficient pipelined FFT architecture. IEEE Trans Circu Syst-II: Analog Digit Signal Process, 2003, 50: 322–325
3 Shi C Z, Yang X, Wang Z S. Design and realization of high performance parallel FFT processor. Comput Eng, 2012, 2: 242–244, 247 [石长振, 杨雪, 王贞松. 高性能并行 FFT 处理器的设计与实现. 计算机工程, 2012, 2: 242–244, 247]
4 Zhou B, Peng Y, Daivid H. Pipeline FFT architectures optimized for FPGAs. Int J Reconfig Comput, 2009, 2009: 1–9
5 Chu C, Zhang Q, Xie Y K, et al. Design of a high performance FFT processor based on FPGA. In: Proceedigns of IEEE Asia South-Pacific Design Automation Conference, Shanghai, 2005. 920–923
6 Watanabe C, Silva C, Munoz J. Implementation of split-radix fast fourier transform on FPGA. In: Proceedings of Southern Programmable Logic Conference, Ipojuca, 2010. 167–170
7 Gao L N, Ma X, Liu T F, et al. Research and implementation of ultra large sequences FFT optimized algorithm. J Electron Inf Tech, 2014, 4: 998–1002 [高立宁, 马潇, 刘腾飞, 等. 基于超大点数 FFT 优化算法的研究与实现. 电子与信息学报, 2014, 4: 998–1002]
8 Jin Y, He H C, Lu Y T. Ternary optical computer principle. Sci China Ser F-Inf Sci, 2003, 46: 145–150
9 Yan J Y, Jin Y, Zuo K Z. Decrese-radix design principle for carrying/borrowing free multi-valued and application in ternary optical computer. Sci China Ser F-Inf Sci, 2008, 51: 1415–1426
10 Jin Y, Gu Y Y, Zuo K Z. Theory, technology and progress of a ternary optical computer's decoder. Sci Sin Inform, 2013, 43: 275–286 [金翊, 顾莹莹, 左开中. 三值光学计算机解码器的理论、技术和实现. 中国科学: 信息科学, 2013, 43: 275–286]
11 Zuo K Z, Jin Y, Peng J J, et al. Design of hundred-bit decoder for ternary optical computer. Chin J Lasers, 2009, 4: 823–827 [左开中, 金翊, 彭俊杰, 等. 三值光计算机百位量级解码器的设计. 中国激光, 2009, 4: 823–827]
12 Peng J J, Shen R, Jin Y, et al. Design and implementation of modified signed-digit adder. IEEE Trans Comput, 2014, 5: 1134–1143
13 Peng J J, Shen R, Ping X S. Design of a high-efficient MSD adder. J Supercomput, 2016, 72: 1770–1784
14 Jin Y, Shen Y F, Peng J J, et al. Principles and construction of MSD adder in ternary optical computer. Sci China Inf Sci, 2010, 53: 2159–2168
15 Jin Y, He H C, Ai L R. Lane of parallel through carry in ternary optical adder. Sci China Ser F-Inf Sci, 2005, 48: 107–116
16 Shen Y F, Pan L, Jin Y, et al. One-step binary MSD adder for ternary optical computer. Sci Sin Inform, 2012, 42: 869–881 [沈云付, 潘磊, 金翊, 等. 三值光学计算机一种限制输入一步式 MSD 加法器. 中国科学: 信息科学, 2012, 42: 869–881]
17 Shen Y F, Pan L. Principle of a one-step MSD adder for a ternary optical computer. Sci China Ser F-Inf Sci, 2014, 57: 012107

18 Jin Y, Wang H J, Ouyang S, et al. Principles, structures and implementation of reconfigurable ternary optical processors. Sci China Inf Sci, 2011, 54: 2236–2246
19 Wang H J, Jin Y, Ouyang S. Design and implementation of 1-bit reconfigurable ternary optical processor. Chin J Comput, 2014, 37: 1500–1507 [王宏健, 金翊, 欧阳山. 一位可重构三值光学处理器的设计和实现. 计算机学报, 2014, 37: 1500–1507]
20 Jin Y, Wang X C, Peng J J, et al. Vector matrix multiplication in ternary optical computer. Int J Numer Anal Model, 2012, 9: 401–409
21 Wang X C, Peng J J, Li M, et al. Carry-free vector-matrix multiplication on a dynamically reconfigurable optical platform. Appl Opt, 2010, 49: 2352–2362
22 Peng J J, Teng L, Jin Y. Realization of a tri-valued programmable cellular automata with ternary optical computer. Int J Numer Anal Model, 2012, 9: 304–311
23 Li X J, Chu J P, Lai Z S, et al. FPGA implementation of a high-speed base-2 256-point FFT. J Circ Syst, 2005, 5: 51–55 [李小进, 初建朋, 赖宗声, 等. 高速基 2 FFT 处理器的结构设计与 FPGA 实现. 电路与系统学报, 2005, 5: 51–55]
24 Zhao L L, Zhang S B, Zhang M, et al. High performance FFT computation based on CUDA. Appl Res Comput, 2011, 28: 1556–1559 [赵丽丽, 张盛兵, 张萌, 等. 基于 CUDA 的高速 FFT 计算. 计算机应用研究, 2011, 28: 1556–1559]
25 Avizienis A. Signed digit number representation for fast parallel arithmetic. IRE Trans Electron Comput, 1961, 10: 389–400
26 Hu X J, Jin Y, Ouyang S. A 40-bit multiplication routine of ternary optical computer. J Shanghai Univ (Nat Sci), 2014, 5: 645–657 [胡晓俊, 金翊, 欧阳山. 三值光学计算机的 40 位乘法例程. 上海大学学报 (自然科学版), 2014, 5: 645–657]
27 Altera Inc., FFT MegaCore function user guide, Version 7.0, 2006
28 Xilinx Inc., Xilinx LogiCore Fast Fourier Transform, Version 4.1, 2007

Implementation of parallel FFT algorithm on a ternary optical computer

Junjie PENG[1*], Xinyu WEI[1,2], Xiaofeng ZHANG[3*], Yunfu SHEN[1] & Youyi FU[1]

1. *School of Computer Engineering and Science, Shanghai University, Shanghai* 200444*, China*;
2. *Shanghai Advanced Research Institute, Chinese Academy of Sciences, Shanghai* 201210*, China*;
3. *National Key Laboratory of Science and Technology on Millimeter-wave Remote Sensing, Beijing* 100854*, China*
* Corresponding author. E-mail: jjie.peng@shu.edu.cn, xiangfan_zxf@sohu.com

Abstract The Fast Fourier Transform (FFT) is widely used in modern digital signal processing systems. In order to improve the efficiency of FFT, parallel methods are often used to speed up FFT in high-speed real-time application environments. However, due to the restrictions of size, energy consumption, heat dissipation etc., traditional electronic methods are becoming unsuitable for FFT applications in certain areas, such as aviation, aerospace, etc. Due to its characteristics of a massive number of data bits and low energy consumption, the Ternary Optical Computer (TOC) is a potential solution for these special cases. To verify this possibility, we studied the design and implementation of a high-speed parallel FFT on a TOC. Through analysis of the traditional radix-2, radix-4, and radix-8 DIT FFT operation processes, several FFT algorithms with higher parallelism, implemented on TOC, are designed by taking advantage of the characteristics of TOC. The implementation processes of these algorithms and the differences between them are presented. Meanwhile, the clock cycles and hardware resources required for each algorithm are also discussed. Simulation results reveal that the FFT implementation method is accurate. The algorithms require less power and fewer clock cycles when implemented on a TOC compared to the traditional methods implemented on an FPGA. This provides a new possible solution for high-speed low-power FFT implementation.

Keywords ternary optical computer, fast Fourier transform, parallel computing, MSD adder, embedded device

Junjie PENG was born in 1977. He received a Ph.D. degree in computer application technology from the Harbin Institute of Technology, Harbin in 2005. He is currently an associate professor at the School of Computer Science at Shanghai University. His research interests include architectures of new computer systems, and storage and energy saving in cloud computing.

Xinyu WEI was born in 1992. He received a bachelor's degree from Wuhan University, China in 2014. He is currently pursuing a master's degree at the School of Computer Engineering and Science, Shanghai University. His research interests include optical computing, ternary optical computers, and computer vision.

Xiaofeng ZHANG was born in 1979. He received a Ph.D. degree in 2005 from the Harbin Institute of Technology. He is currently a professor at the National Lab on Milimeterwave Remotesensing, No. 25 Institute, Chinese Aerospace Science and Industry Group. His research interests include precision guidance and millimeter engineering.

Yunfu SHEN was born in 1960. He received a Ph.D. degree in science from Beijing Normal University, Beijing in 1996. He is currently an associate professor at Shanghai University. His research interests include reliability and operation units for ternary optical computers, formalization methods for software and hardware, and model checking.