文章编号：2095-4980(2021)02-0303-05

基于 FPGA 的混合基 FFT 算法设计与实现

侯晓晨，孟 骁，陈 昊

(北京理工大学 信息与电子学院，北京 100081)

摘 要：目前，研究资源节约型的低复杂度混合基快速傅里叶变换(FFT)设计技术具有重要的应用价值。本文基于现场可编程逻辑门阵列(FPGA)平台提出并实现了一种新型混合基 FFT 分解算法。该算法基于原位存储结构设计，采用素数因子分解与库利–图基分解相结合的混合分解模式，在省去了一步旋转因子乘法运算的同时也有效减小了存储空间和运算量，并采用通用蝶形单元模块设计使得算法能够同时适应基 2、基 3、基 4 的 FFT 运算。仿真结果表明，该算法可以极大提高 FFT 处理点数的灵活性，有效节省运算资源。

关键词：快速傅里叶变换；混合基算法；通用蝶形单元；现场可编程逻辑门阵列

中图分类号：TN79^{+}1 **文献标志码：**A **doi：**10.11805/TKYDA2019435

Design and implementation of mixed-radix FFT algorithm based on FPGA

HOU Xiaochen，MENG Xiao，CHEN Hao

(School of Information and Electronics，Beijing Institute of Technology，Beijing 100081，China)

Abstract：Recently, researches on resource-saving mixed-radix Fast Fourier Transform(FFT) technology with low complexity and high efficiency are of vital importance in digital signal processing. In this paper, a new mixed-radix FFT decomposition algorithm based on Field Programmable Gate Array(FPGA) is proposed and implemented. The proposed in-place algorithm adopts a hybrid decomposition mode combining prime factorization algorithm and Cooley–Tukey algorithm, which can save one-step multiplication operation of the rotation factor and also reduce the storage space and operation amount effectively, while it also uses the universal butterfly unit module to accommodate to the radix-2, radix-3, and radix-4 FFT operations. The simulation results indicate that the proposed algorithm can greatly improve the flexibility of FFT processing points and effectively save computing resources.

Keywords：Fast Fourier Transform；mixed-radix algorithm；universal butterfly unit；Field Programmable Gate Array

快速傅里叶变换(FFT)算法具有计算量小的显著优点，在信号处理技术领域得到了广泛应用，现已成为数字信号处理强有力的工具。FFT 硬件实现速度取决于算法效率，目前基于现场可编程逻辑门阵列(FPGA)平台实现的 FFT 算法 IP 核大多采用库利–图基基-2、基-4 或者二者混合算法，要求其变换长度 $N=2^n$。若序列长度不满足 2 的整数次幂时，需要将序列长度补零至最近的 2^n 个点，补零方式会影响运算系统的准确度和实时性。此外，该算法可选点数范围小，灵活性不高，且在长度为非整数倍周期的情况下容易造成频谱泄露，造成频谱失真[1]。因此，研究资源节约型的低复杂度混合基 FFT 设计技术具有重要应用价值。针对基于 FPGA 实现混合基 FFT 技术的问题，国内外学者进行了大量研究，文献[2]中 Demuth 给出了混合基 FFT 数据地址产生方法，但是操作数寻址与旋转因子寻址采用两种不同的方案且寻址过程所需参数过多，不利于硬件实现；鉴于基-2 与基-4 的 FFT 运算结构相对简单，文献[3]与文献[4]分别采用了基-2/4 与基-2/2^k 算法实现混合基 FFT 运算；文献[5]中 Hsiao 提出一种通用混合基 FFT 地址映射方法，该方法需要多个取模操作实现，且其方案中并没有对旋转因子

收稿日期：2019-10-31；**修回日期：**2019-12-03

作者简介：侯晓晨(1992–)，男，在读硕士研究生，主要研究方向为信号处理、通信数据链。email:465383459@qq.com

进行考虑；文献[6]中提出的映射方案仅由一个计数器就可获得一种低复杂度的地址产生方法，不需要额外的求模运算，具有较好的硬件实现结构。混合基 FFT 算法能适应不同长度的点数运算，可以有效降低存储资源，能够大大加强 FFT 在工程应用上的适用性。本文基于 FPGA 平台提出了一种基-2、基-3、基-4 的混合基 FFT 算法，并且结合素因子分解与库利-图基分解对混合基进行分解运算，并进行算法仿真验证。

1 混合基 FFT 分解算法

1.1 索引映射

有限列长为 N 的序列 $x(n)$，其离散傅里叶变换(Discrete Fourier Transform，DFT)的计算公式为：

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

式中 $W_N=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}}$ 称为单位根或旋转因子。对 n,k 分别进行如下映射：

$$\begin{cases}n=f(n_1,n_2)=(N_2n_1+A_2n_2)\bmod N, & n_1,k_1=0,1,\cdots,N_1-1,\quad N=N_1N_2\\ k=\overline{f}(k_1,k_2)=(B_1k_1+N_1k_2)\bmod N, & n_2,k_2=0,1,\cdots,N_2-1\end{cases} \tag{2}$$

函数 f^{-1} 和 $\overline{f}^{-1}$ 可将 n 和 k 从一维索引 $[0,N-1]$ 分别映射成 $[0,N_1-1]$*$[0,N_2-1]$ 的二维索引 (n_1,n_2) 和 (k_1,k_2) 的分解表示，f 和 $\overline{f}$ 分别是这两个映射函数的逆函数(因为这两个映射都是一一映射关系，因此其逆函数必然存在)，系数 A_2 和 B_1 的选取依赖于 N_1 和 N_2 之间的关系。

1.2 素因子算法(PFA)

素因子算法(Prime Factor Algorithm，PFA)采用点数互素的短点数 DFT 运算的组合来完成长点数的运算。这种实现方法不会引入额外的与旋转因子相乘的运算，因而该方法的乘法运算次数相对于库利-图基算法要少得多。素因子算法的特点是既能减少乘法运算次数，还有便于程序实现的结构。在通用机上，它的效率是最高的。不过，该算法是以更加复杂的下标变换为代价来消除旋转因子。将 FFT 序列分为互质的小序列，可先分别计算小序列变换，再得到 N 点变换。N_1 和 N_2 互质，且合理选择 A_2 和 B_1 的值，使其满足[7]：

$$<N_2B_1>_N=N_2,\ <A_2N_1>_N=N_1,\ <A_2B_1>_N=0 \tag{3}$$

式中 $<>_N$ 表示模 N 运算。可取 A_2 和 B_1 满足以下关系：

$$A_2=p_1N_1=q_1N_2+1,\quad B_1=p_2N_2=q_2N_1+1 \tag{4}$$

此时 A_2, B_1,p_1,q_1,p_2,q_2 均为正整数，则可将 DFT 进行如下分解：

$$X(k)=X(k_1,k_2)=\sum_{n_2}\sum_{n_1}x(n_1,n_2)W_N^{(N_2n_1+A_2n_2)(B_1k_1+N_1k_2)}=\sum_{n_2}\sum_{n_1}x(n_1,n_2)W_{N_1}^{n_1k_1}W_{N_2}^{n_2k_2}=\sum_{n_2}\left[\sum_{n1}x(n_1,n_2)W_{N_1}^{n_1k_1}\right]W_{N_2}^{n_2k_2}=\sum_{n_2}X_1(k_1,n_2)W_{N_2}^{n_2k_2} \tag{5}$$

1.3 库利-图基算法(CTA)

库利-图基算法(Cooly-Tukey Algorithm，CTA)1965 年由 J W 库利和 T W 图基提出，是最常见的快速傅里叶变换算法。采用这种算法能使计算机计算离散傅里叶变换所需要的乘法次数大为减少，特别是被变换的抽样点数 N 越多，FFT 算法计算量的节省就越显著。该算法成功地将计算离散傅里叶变换的时间复杂度从 $O(N^2)$ 降低到 $O(N(\log N))$，并且间接导致了 20 世纪 70 年代数字信号处理(Digital Signal Processing，DSP)芯片的迅猛发展。这一方法以分治法为策略递归地将长度为 $N=N_1N_2$ 的 DFT 分解为长度分别为 N_1 和 N_2 的两个较短序列的 DFT，以及与旋转因子的复数乘法。此外，库利-图基算法是将 DFT 分解为较小长度的多个 DFT，因此它可同任一种其他 DFT 算法联合使用。若式(2)中 N_1 和 N_2 不互素，可取 A_2=B_1=1，则 DFT 运算可分解为：

$$X(k)=X(k_1,k_2)=\sum_{n2}\sum_{n_1}x(n_1,n_2)W_{N_1}^{n_1k_1}W_{N_2}^{n_2k_2}W_N^{n_2k_1}=\sum_{n_2}\left\{W_N^{n_2k_1}\left[\sum_{n_1}x(n_1,n_2)W_{N_1}^{n_1k_1}\right]\right\}W_{N_2}^{n_2k_2}=\sum_{n_2}\left[W_N^{n_2k_1}\,X_1(k_1,n_2)\right]W_{N_2}^{n_2k_2} \tag{6}$$

通过与式(2)中的映射关系进行观察对比，如果用数位形式表示对 n 和 k 的分解，可以得到如下表达形式：

$$n=f(n_1,n_2)=(n_1,n_2)_{\mathrm{digit}},\quad k=\overline{f}(k_1,k_2)=(k_2,k_1)_{\mathrm{digit,reverse}} \tag{7}$$

式中：n 和 k 的数位表示是颠倒的，同时颠倒的还有各个数位的基。因此，当需要顺序输入同时保证顺序输出时，在 I/O 的过程中应当进行适当的操作，其他中间步骤不会受到影响。这个性质是 PFA 所不具备的。尽管在

由分解得到一维表示时都是简单的加权求和(PFA 需要额外的取模运算)，但在逆运算时则存在很大的差别。由于 CTA 数位表示的顺序性，它的逆运算可以通过多级计数器实现，而该过程对于 PFA 非常复杂。

2　新型混合基算法设计与实现

通过将 PFA 算法的分解公式(5)与 CTA 算法的分解公式(6)进行对比可以发现，CTA 较之 PFA 中间多了一级旋转因子的乘法，而对于索引的计算，PFA 会比 CTA 复杂很多。对于一个已经给定长度为 N 的 FFT，设计的 FFT 算法是 PFA 与 CTA 的结合，从而使各自的优点能够充分发挥。具体而言，先利用 PFA 将 N 分解为两个互素的因子 N_1 和 N_2，$N_1=2^{t_1}$，$N_2=3^{t_2}$；然后再利用 CTA 对 N_1 和 N_2 分别进行分解。这样做有如下两点好处：首先，PFA 可以节省一步旋转因子的乘法运算；并且，可以将其他步骤旋转因子的基由 N 简化为它的因子 N_1 和 N_2，这将大大减小存储空间和运算量。

对 N_1 分解是为了更高效计算：

$$X_1(k_1,n_2)=\sum_{n_1}x(n_1,n_2)W_{N_1}^{n_1k_1} \tag{8}$$

上式可进一步分解为：

$$\begin{cases} n_1=(n_{1,1},\widetilde{n_{1,2}})=(n_{1,1},n_{1,2},\cdots,n_{1,r},\widetilde{n_{1,r+1}})=(n_{1,1},n_{1,2},\cdots,n_{1,R_1}) \\ k_1=(k_{1,1},\widetilde{k_{1,2}})=(k_{1,1},k_{1,2},\cdots,k_{1,r},\widetilde{k_{1,r+1}})=(k_{1,1},k_{1,2},\cdots,k_{1,R_1}) \end{cases} \tag{9}$$

其中，

$$\begin{cases} \widetilde{n_{1,r}},\widetilde{k_{1,r}}=0,1,\cdots,\widetilde{N_{1,r}}-1;\ \ n_{1,r},k_{1,r}=0,1,\cdots,N_{1,r}-1 \\ \widetilde{n_{1,r}}=(n_{1,r},\widetilde{n_{1,r+1}})=(n_{1,r},n_{1,r+1},\cdots,n_{1,R_1});\ \widetilde{N_{1,r}}=N_{1,r}\widetilde{N_{1,r+1}}=N_{1,r}N_{1,r+1}\cdots N_{1,R_1} \end{cases} \tag{10}$$

可得到类似文献[8]中的递推公式：

$$X_{1,r}(k_{1,1},k_{1,2},\cdots,k_{1,r},\widetilde{n_{1,r+1}},n_2)=\left[\sum\nolimits_{n_{1,r}}X_{1,r-1}(k_{1,1},k_{1,2},\ldots,n_{1,r},\widetilde{n_{1,r+1}},n_2)W_{N_{1,r}}^{n_{1,r}k_{1,r}}\right]W_{\widetilde{N_{1,r}}}^{\widetilde{n_{1,r+1}}k_{1,r}} \tag{11}$$

式(11)中这个迭代过程一共有 R_1 步，称 $W_{\widetilde{N_{1,r}}}^{\widetilde{n_{1,r+1}}k_{1,r}}$ 为旋转因子。在上述的迭代过程中，对于第 r 次迭代，需要将其拆分为蝶形运算与旋转因子乘法两个步骤。一个 $radix-N_{1,r}$ 蝶形运算可表示为：

$$X_{1,r}(k_{1,1},k_{1,2},\cdots,k_{1,r},n_{1,r+1},n_{1,R_1},n_2)=X_{1,r-1}(k_{1,1},k_{1,2},\cdots,n_{1,r},n_{1,r+1},n_{1,R_1},n_2)W_{N_{1,r}}^{n_{1,r}k_{1,r}} \tag{12}$$

其中输入和输出的索引值的区别仅在于第 r 维，即将 $n_{1,r}$ 替换为 $k_{1,r}$。在这一轮迭代过程中，需要对除第 r 维外的所有维度进行遍历，也就是对所有蝶形结构进行遍历。而每个蝶形结构的输入和输出对应着 $n_{1,r}$ 和 $k_{1,r}$ 的所有可能的 $N_{1,r}$ 个取值[9]。

而旋转因子可变换为：

$$W_{\widetilde{N_{1,r}}}^{\widetilde{n_{1,r+1}}k_{1,r}}=W_{N1}^{N_{1,1}\cdots N_{1,r-1}(n_{1,r+1}\cdots n_{1,R_1})k_{1,r}} \tag{13}$$

因此，若查找表中保存有 W_{N_1} 所有指数值，则可通过查找表得出旋转因子的值。显然，只需要计算指数部分的索引值即可。通过对指数部分进行分解，也可有效简化运算，节省系统资源。

由于算法采用的是顺序存储结构，关键在于将多维分解表示转化为一维形式，即将 $(n_{1,1},n_{1,2},\cdots,n_{1,R_1},n_{2,1},n_{2,2},\cdots,n_{2,R_2})$ 转化为 n。实现 $n=f_{\mathrm{PFA}}(n_1,n_2)$，难点在于模 N 的求取运算。将表达式展开，并进行如下形式的转换：

$$n=(N_2n_1+(q_1N_2+1)n_2)\bmod N=(N_2(n_1+q_1N_2)+n_2)\bmod N=N_2n_1^{'}+n_2 \tag{14}$$

式中 $n_1^{'}=(n_1+q_1N_2)\bmod N_1$。从上式中可以发现，运算中已经规避了对 N 的取模操作。因此，结合文献[6]中计数器的设计方案，可以高效获取当前的内存地址。

本算法中，蝶形运算采用高效的维诺格兰傅里叶变换算法(Winograd Fourier Transform Algorithm，WFTA)。如果不同的基采用不同的蝶形单元，则蝶形单元模块将会占用一部分存储资源，因此也需要一种蝶形单元模块能够同时适用于基-2、基-3、基-4。在一个蝶形运算模块内部包含所有 3 种基的蝶形运算[10]。

将蝶形运算模块分为 3 个独立的流水线步骤，每个步骤使用一个 DSP slice，分别对应 pre-add、multiply 以及 post-add 三个操作，基-2、基-3 和基-4 都可以映射到这样的硬件结构，从而降低复杂度并减少资源消耗。以基-3 算法为例，蝶形运算的分解如图 1 所示，其中控制信号 C0 及其乘因子值的配置情况由表 1 给出。对 N_2 的

分解与 N_1 类似，区别在于此时固定的是 k_1 而不是 n_2，对 N_2 进行与 N_1 类似的操作后即可完成高效计算。该混合基 FFT 算法基于基-2、基-3、基-4 进行混合基 FFT 运算，也就是说，长度 N 分解到最后为这 3 个基的组合。此外，由于基-4 运算比基-2 运算效率更高，也更节省资源，因此基-2 的运算量只存在 0 或 1 两种可能。

另外，综合考虑算法精确度及防止数据溢出等问题，还需要加入缩放因子用以调整幅度。为了简化运算，还需要引入计数器，用以在运算中完成对所有蝶形单元的遍历以及辅助对所有操作数在内存中进行索引。运用以上分解的计算之后，内存中存储的 FFT 的结果其实是逆序的。需要逆序输出时，可以顺序输出内存中的结果，当需要正序输出时，需要对输出时的读地址进行调整，通过之前的计算依次读取对应的索引号即可完成。

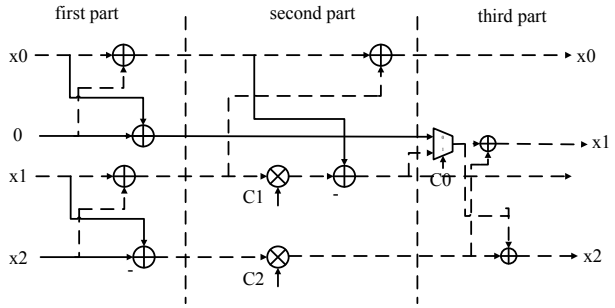

Fig.1 Diagram of radix-3 butterfly circuit
图 1 基-3 蝶形电路图

表 1 控制信号及系数配置

Table1 Configuration of control signals and coefficients

radix	C0	C1	C2
R-2	–	1	0
R-3	0	1/2	$-j\sqrt{3}/2$
R-4	1	1	−j

3 算法仿真验证

上述算法理论上可支持任意基底的混合基运算，考虑到 FPGA 资源限制与算法实现难度，这里只实现基-2、基-3、基-4 的混合基运算。算法的功能测试在 Matlab 和 Vivado 两个平台进行。为了测试算法功能的正确性，首先使用 Matlab 生成测试数据，然后为了与 FPGA 平台对比，需要将浮点型数据转换成定点，同时中间计算过程也需要采用定点运算，最后将 Matlab 与 FPGA 的结果进行比对。

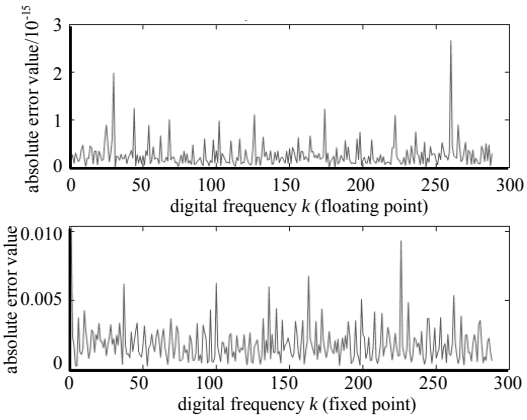

Fig.2 Comparison of absolute errors
图 2 绝对误差对比

目前定点 DSP 对定点数据的处理能力远强于浮点数据，因而在仿真验证中采用 Q15 量化方案。利用 Matlab 生成一个载波频率为 1×10^7 Hz 与 1.2×10^7 Hz 相叠加的双音信号，对该信号做 288 点 FFT 运算，分别利用 Matlab 系统函数、浮点混合基算法以及定点混合基算法进行运算比较。以向量二范数作为表征，可得到标准 FFT 算法与浮点混合基算法的欧氏距离为 $6.426\,585\times10^{-15}$，而标准 FFT 算法与定点混合基算法欧氏距离为 $3.842\,434\times10^{-2}$。图 2 分别给出了浮点混合基算法和定点混合基算法与标准 FFT 算法绝对误差的比较。从仿真结果可以看出，浮点混合基算法与标准 FFT 算法可实现精确度的完美拟合，从而验证了算法的可行性。而定点混合基运算存在一定的精确度损失，这一部分损失是由于浮点运算定点化所引入的，由于采用的是 Q15 的量化方案，存在小数部分的舍位运算，这部分损失可以通过计算得出。在 FPGA 仿真测试中，混合基 FFT 算法采用 Verilog 语言进行编程，运行软件为 Vivado 2017.4，输入数据采用 Matlab 生成的定点数据，利用 testbench 完成仿真测试。可以看到整个混合基 FFT 运算一共进行了 5 级运算，其仿真结果如图 3 所示。FPGA 最终的定点化运算结果导出至 Matlab 中进行比对，其与 Matlab 的定点化运算结果完全一致，从而验证了算法硬件实现的可行性。

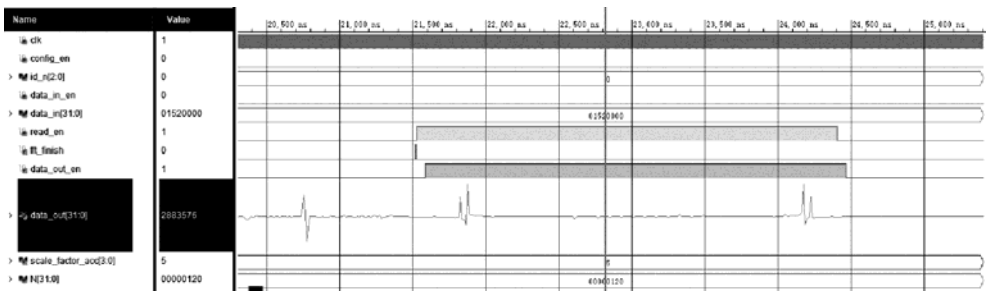

Fig.3 Simulation implementation results of the algorithm based on FPGA
图 3 FPGA 算法仿真实现结果

FPGA 实现中算法整体的资源占用情况与固定基知识产权核(Intellectual Property core，IP 核)的 FFT 资源占用对比情况如表 2 所示。其中，混合基算法采用 288 点 FFT 运算，而固定基算法使用 512 点 FFT 运算。可以看到，无论从 LUTs,registers,block RAM 还是 DSPs，混合基所占用的运算资源均小于固定基，因而可以有效节省系统资源。

表 2 混合基与固定基运算资源占用对比

Table2 Comparison of resource occupancy between mixed-radix and fixed-radix operations

FFT algorithm	slice LUTS	slice registers	block RAM	DSPs
mixed-radix algorithm	383	770	2.5	10
fixed-radix IP core	1311	2038	5.5	12

4　结论

在本论文中，针对需要在 Xilinx 7 Series FPGA 平台上实现高效 FFT 运算的需求，利用素数因子分解与库利-图基分解相结合的混合分解模式，并采用通用蝶形单元模块设计的方法，提出了一种可适应不同点数 FFT 计算的混合基 FFT 算法，并通过 MATLAB 与 Vivado 两个仿真平台验证了算法的可行性。

仿真结果表明，基于新型混合基 FFT 算法可满足基-2、基-3、基-4 的混合基 FFT 运算，相比较固定基 FFT 算法，在相同序列长度范围内，混合基 FFT 可精确计算的序列长度数量大大增加，极大提高了算法的适用性。通过对算法进行优化，在省去了一步旋转因子乘法运算的同时也有效减小了存储空间和运算量。此外，该算法极大提高 FFT 处理点数的灵活性，并能够有效节省运算资源。

参考文献：

[1] 程志鹏,马琪,竺红卫. 混合基 FFT 算法运算量分析[J]. 太赫兹科学与电子信息学报, 2016,14(6):925-928. (CHENG Zhipeng,MA Qi,ZHU Hongwei. Computational complexity analysis on mixed-radix FFT[J]. Journal of Terahertz Science and Electronic Information Technology, 2016,14(6):925-928.)

[2] DEMUTH G L. Algorithms for defining mixed radix FFT flow graphs[J]. IEEE Transactions on Acoustics Speech & Signal Processing, 1989,37(9):1349-1358.

[3] JACOBSON A T,TRUONG D N,BAAS B M. The design of a reconfigurable continuous-flow mixed-radix FFT processor[C]// 2009 IEEE International Symposium on Circuits and Systems. Taipei,Taiwan,China:IEEE, 2009.

[4] XIAO Hao,PAN An,CHEN Yun,et al. Low-cost reconfigurable VLSI architecture for fast Fourier transform[J]. IEEE Transactions on Consumer Electronics, 2008,54(4):1617-1622.

[5] HSIAO C F,CHEN Y,LEE C Y. A generalized mixed-radix algorithm for memory-based FFT processors[J]. IEEE Transactions on Circuits and Systems II: Express Briefs, 2010,57(1):26-30.

[6] MA Cuimei,XIE Yizhuang,CHEN He,et al. Simplified addressing scheme for mixed radix FFT algorithms[C]// 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). Florence,Italy:IEEE, 2014.

[7] TRUONG T K,REED I S,LIPES R G,et al. On the application of a fast polynomial transform and the Chinese Remainder Theorem to compute a two-dimensional convolution[J]. IEEE Transactions on Acoustics Speech and Signal Processing, 1981,29(1):91-97.

[8] REISIS D,VLASSOPOULOS N. Conflict-free parallel memory accessing techniques for FFT architectures[J]. IEEE Transactions on Circuits and Systems I: Regular Papers, 2008,55(11):3438-3447.

[9] 王非非,杜伟韬. 一种旋转因子访存优化的 FFT 算法[J]. 太赫兹科学与电子信息学报, 2011,9(2):206-210. (WANG Feifei,DU Weitao. Optimized design of memory access for twiddle factors in FFT algorithm[J]. Journal of Terahertz Science and Electronic Information Technology, 2011,9(2):206-210.)

[10] 马翠梅. 低复杂度混合基 FFT 研究与设计[D]. 北京:北京理工大学, 2014. (MA Cuimei. Research and design of low complexity mixed base FFT[D]. Beijing:Beijing Polytechnic University, 2014.)