doi: 10.13413/j.cnki.jdxblxb.2018.01.26

基于 FPGA 架构的可变点 FFT 处理器设计与实现

才 华, 陈广秋, 刘广文, 耿振野, 杜兆圣
(长春理工大学 电子信息工程学院, 长春 130022)

摘要: 通过对传统的基-4 快速 Fourier 变换(FFT)算法进行优化, 降低基-4 算法的复杂度, 使其具有基-2 算法的蝶形结构. 采用优化后的基-4/2 混合基算法及流水线基-2^2 单路延时反馈(R2^2SDF)结构设计可变点 FFT 处理器, 并对输出结果进行功能和信号仿真验证. 结果表明, 该处理器的有效性和执行效率均表现良好.

关键词: 正交频分多址技术; 快速 Fourier 变换; 蝶形运算; 流水线; 基-2^2 单路延时反馈
中图分类号: TN47 **文献标志码**: A **文章编号**: 1671-5489(2018)01-0151-08

Design and Implementation of Variable Points FFT Processor Based on FPGA Architecture

CAI Hua, CHEN Guangqiu, LIU Guangwen, GENG Zhenye, DU Zhaosheng
(*School of Electronic and Information Engineering, Changchun University of Science and Technology, Changchun* 130022, *China*)

Abstract: The complexity of radix-4 algorithm was reduced by optimizing the traditional radix-4 fast Fourier transform (FFT) algorithm, which retained the butterfly structure of radix-2 algorithm. The optimized mixed radix-4/2 and pipeline radix-2^2 single-path delay feedback (R2^2SDF) structure were adopted to design the variable points FFT processor, and the output results were verified by the function and signal simulation. The results show that the FFT processor is excellent in validity and efficiency.

Key words: orthogonal frequency division multiple access (OFDMA); fast Fourier transform (FFT); butterfly operation; pipeline; radix-2^2 single-path delay feedback (R2^2SDF)

快速 Fourier 变换(FFT)作为一种有效计算离散 Fourier 变换(DFT)的方法, 在通信、滤波及数字谱分析等领域应用广泛. 利用现场可编程门阵列(FPGA)可设计 FFT 处理器的硬件架构[1].

随着 FFT 算法的不断完善, 在基-2FFT 算法[2]的基础上, 文献[3-6]又提出了基-4、基-8 和基-16 固定基以及分裂基等算法. 随着基数 r 的增加, 算法分解级数逐渐减少, 所需运算量(乘法和加法)也逐渐减少, 但其算法控制的复杂度增大. 由于可实现的点数受到限制, 因此需引进混合基算法兼顾 FFT 的运算量和复杂度[7].

常用的 FFT 处理器硬件结构有 4 种[8]: 顺序结构、流水线结构、并行结构和阵列结构. 其中顺序

收稿日期: 2017-02-15.

作者简介: 才 华(1977—), 男, 汉族, 博士, 副教授, 从事图像处理与机器视觉的研究, E-mail: caihua@cust.edu.cn. 通信作者: 陈广秋(1977—), 男, 汉族, 博士, 副教授, 从事高速数据采集与处理的研究, E-mail: guangqiu_chen@126.com.

基金项目: 吉林省科技发展计划项目(批准号: 20130101179JC)和教育部留学归国人员科研启动基金(批准号: 教外司留 1685).

结构运算速度慢，实时性差；流水线结构比顺序结构的运算速度提高了 $\log_r N$ 倍(其中：N 为序列点数；r 为基数)，所需的硬件资源有所增加；阵列结构的运算速度最快，但所需硬件资源和功耗也最大.由于流水线结构包含多个独立的蝶形运算单元，每个单元负责一级蝶形运算，各级蝶形运算单元间采用流水线方式进行工作，通过增减结构中蝶形运算单元可实现不同点数序列的 FFT，此外流水线结构还具有芯片面积小、功耗低以及高数据吞吐量等优点，因此可采用流水线结构处理硬件资源与处理速度间的关系.

正交频分多址技术(OFDMA)是基于正交频分复用技术(OFDM)的新一代无线接入技术，在 IEEE802.16e 物理层标准中，不同带宽的 OFDMA 系统采用的 FFT 点数不同，如 3 M 带宽采用 256 点，10 M 带宽采用 1 024 点，20 M 带宽采用 2 048 点等[9-10]. 本文利用 FPGA 硬件架构，采用优化的基-4/2 混合基分解算法及流水线硬件结构实现可变点 FFT 处理器的设计，并将其应用于 OFMDA 系统中.

1 按频率抽取基-2/4 混合基算法原理

1.1 按频率抽取基-2FFT 算法原理

设序列点数 $N=2^L$(L 为整数)，按频率抽取(DIF)基-2FFT 算法将输入 $x(n)$ 按 n 的顺序分为前后两部分，将结果 $X(k)$ 按 k 的奇偶进行分组[11-12]. 其中

$$\begin{aligned}X(k)=&\sum_{n=0}^{N/2-1}x(n)W_N^{nk}+\sum_{n=N/2}^{N-1}x(n)W_N^{nk}=\\&\sum_{n=0}^{N/2-1}[x(n)+W_N^{kN/2}x(n+N/2)]W_N^{nk}=\\&\sum_{n=0}^{N/2-1}[x(n)+(-1)^k\cdot x(n+N/2)]W_N^{nk},\quad k=0,1,\cdots,N-1,\end{aligned}\tag{1}$$

按 k 的奇偶性，将 $X(k)$ 划分为

$$\begin{cases}X(k)\big|_{k=2r}=\sum_{n=0}^{N/2-1}[x(n)+x(n+N/2)]W_{N/2}^{nr},\\X(k)\big|_{k=2r+1}=\sum_{n=0}^{N/2-1}[x(n)-x(n+N/2)]W_N^nW_{N/2}^{nr},\quad r=0,1,\cdots,N/2-1,\end{cases}\tag{2}$$

由式(2)可得 DIF 基-2 蝶形运算单元，如图 1 所示.

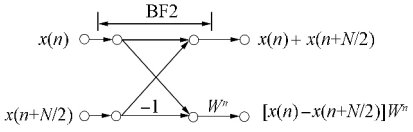


图 1 DIF 基-2 蝶形运算单元

Fig.1 Radix-2 DIF butterfly operation unit

1.2 按频率抽取基-4FFT 算法原理

设序列点数 $N=4^L$(L 为整数)，DIF 基-4FFT 算法将输入 $x(n)$ 按 n 的顺序分为前后 4 组，将运算结果 $X(k)$ 按 $k=4r$，$k=4r+1$，$k=4r+2$ 和 $k=4r+3\left(r=0,1,\cdots,\frac{N}{4}-1\right)$ 进行分组. 其中

$$\begin{aligned}X(k)=&\sum_{n=0}^{N/4-1}x(n)W_N^{nk}+\sum_{n=N/4}^{N/2-1}x(n)W_N^{nk}+\sum_{n=N/2}^{3N/4-1}x(n)W_N^{nk}+\sum_{n=3N/4}^{N-1}x(n)W_N^{nk}=\\&\sum_{n=0}^{N/4-1}x(n)W_N^{nk}+\sum_{n=0}^{N/4-1}x(n+N/4)W_N^{Nk/4}\cdot W_N^{nk}+\\&\sum_{n=0}^{N/4-1}x(n+N/2)W_N^{Nk/2}\cdot W_N^{nk}+\sum_{n=0}^{N/4-1}x(n+3N/4)W_N^{3Nk/4}\cdot W_N^{nk}=\\&\sum_{n=0}^{N/4-1}[x(n)+x(n+N/4)W_N^{Nk/4}+x(n+N/2)W_N^{Nk/2}+x(n+3N/4)W_N^{3Nk/4}]\cdot W_N^{nk},\end{aligned}\tag{3}$$

对 $X(k)$ 进行分组

$$\begin{cases} X(k)\big|_{k=4r}=\sum\limits_{n=0}^{N/4-1}[x(n)+x(n+N/4)+x(n+N/2)+x(n+3N/4)]\cdot W_{N/4}^{nr}=\sum\limits_{n=0}^{N/4-1}A\cdot W_{N/4}^{nr},\\ X(k)\big|_{k=4r+1}=\sum\limits_{n=0}^{N/4-1}[x(n)-jx(n+N/4)-x(n+N/2)+jx(n+3N/4)]W_N^n\cdot W_{N/4}^{nr}=\sum\limits_{n=0}^{N/4-1}B\cdot W_{N/4}^{nr},\\ X(k)\big|_{k=4r+2}=\sum\limits_{n=0}^{N/4-1}[x(n)-x(n+N/4)+x(n+N/2)-x(n+3N/4)]W_N^{2n}\cdot W_{N/4}^{nr}=\sum\limits_{n=0}^{N/4-1}C\cdot W_{N/4}^{nr},\\ X(k)\big|_{k=4r+3}=\sum\limits_{n=0}^{N/4-1}[x(n)+jx(n+N/4)-x(n+N/2)-jx(n+3N/4)]W_N^{3n}\cdot W_{N/4}^{nr}=\sum\limits_{n=0}^{N/4-1}D\cdot W_{N/4}^{nr}. \end{cases} \tag{4}$$

由式(4)可得 DIF 基-4 蝶形运算单元，如图 2 所示. 由图 2 可见，基-4 比基-2 蝶形运算复杂，结构差别较大，规律性较差，不适合硬件实现混合基运算，因此需对上述算法进行优化. 将式(4)中的序列重新分组，可得优化的 DIF 基-4FFT 算法为

$$\begin{cases} X(k)\big|_{k=4r}=\sum\limits_{n=0}^{N/4-1}\{[x(n)+x(n+N/2)]+[x(n+N/4)+x(n+3N/4)]\}\cdot W_{N/4}^{nr}=\sum\limits_{n=0}^{N/4-1}A'\cdot W_{N/4}^{nr},\\ X(k)\big|_{k=4r+1}=\sum\limits_{n=0}^{N/4-1}\{[x(n)-x(n+N/2)]-j[x(n+N/4)-x(n+3N/4)]\}W_N^n\cdot W_{N/4}^{nr}=\sum\limits_{n=0}^{N/4-1}B'\cdot W_{N/4}^{nr},\\ X(k)\big|_{k=4r+2}=\sum\limits_{n=0}^{N/4-1}\{[x(n)+x(n+N/2)]-[x(n+N/4)+x(n+3N/4)]\}W_N^{2n}\cdot W_{N/4}^{nr}=\sum\limits_{n=0}^{N/4-1}C'\cdot W_{N/4}^{nr},\\ X(k)\big|_{k=4r+3}=\sum\limits_{n=0}^{N/4-1}\{[x(n)-x(n+N/2)]+j[x(n+N/4)-x(n+3N/4)]\}W_N^{3n}\cdot W_{N/4}^{nr}=\sum\limits_{n=0}^{N/4-1}D'\cdot W_{N/4}^{nr}. \end{cases} \tag{5}$$

由式(5)可得优化后的 DIF 基-4 蝶形计算单元，如图 3 所示.

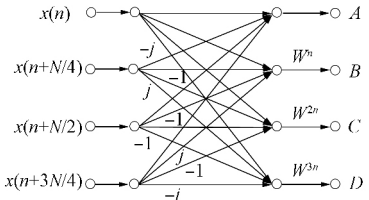


图 2　传统 DIF 基-4 蝶形运算单元

Fig.2　Traditional radix-4 DIF butterfly operation unit

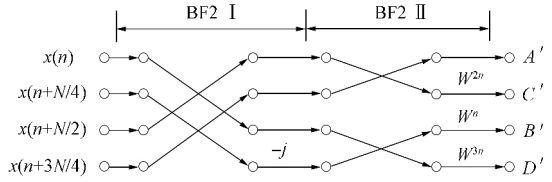


图 3　优化后的 DIF 基-4 蝶形运算单元

Fig.3　Optimized radix-4 DIF butterfly operation unit

通过计算可知，优化后的 DIF 基-4 蝶形运算比传统的 DIF 基-4 蝶形运算可减少 4 个复数加法运算，其结构与 DIF 基-2 蝶形结构相同，信号流图具有较强的规律性，适合硬件实现混合基运算. 图 4 为优化后 $N=16$ 的 DIF 基-4FFT 流图.

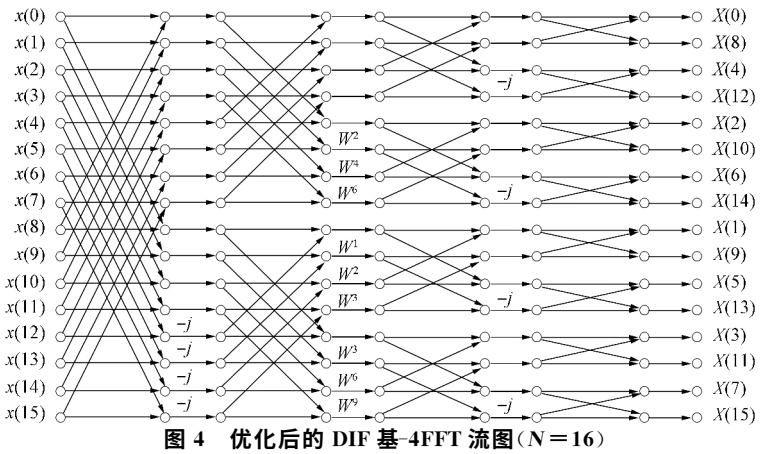


图 4　优化后的 DIF 基-4FFT 流图($N=16$)

Fig.4　Flow graph of optimized radix-4 DIF FFT ($N=16$)

由图4可见，优化后DIF基-4FFT与DIF基-2FFT的各级流图在结构形式上一致，仅旋转因子不同.

2 混合基FFT算法原理

N点DFT的计算表达式为

$$X(k)=\sum_{n=0}^{N-1}x(n)W_N^{nk},\qquad k=0,1,\cdots,N-1,\tag{6}$$

其中$N=r_1r_2\cdots r_L$为复合数，按整数的多基多进制表示形式，式(6)中的n和k可分别表示为

$$\begin{cases} n=(n_{L-1}n_{L-2}\cdots n_1n_0)_{r_1r_2\cdots r_L}=n_{L-1}(r_2r_3\cdots r_L)+n_{L-2}(r_3r_4\cdots r_L)+\cdots+n_1r_L+n_0,\\ k=(k_{L-1}k_{L-2}\cdots k_1k_0)_{r_Lr_{L-1}\cdots r_1}=k_{L-1}(r_{L-1}\cdots r_1r_0)+k_{L-2}(r_{L-2}\cdots r_1r_0)+\cdots+k_1r_1+k_0,\end{cases}\tag{7}$$

其中：$n_i=0,1,\cdots,r_{L-i}-1$；$k_i=0,1,\cdots,r_{i+1}-1$，$i=0,1,\cdots,L-1$. 将式(7)中n和k的值代入式(6)可得

$$\begin{aligned}X(k)=&\ X(k_{L-1}(r_{L-1}r_{L-2}\cdots r_1)+k_{L-2}(r_{L-2}r_{L-3}\cdots r_1)+\cdots+k_1r_1+k_0)=\ X[k_{L-1},k_{L-2},\cdots k_1,k_0]=\\&\sum_{n_0=0}^{r_L-1}\sum_{n_1=0}^{r_{L-1}-1}\cdots\sum_{n_{L-2}=0}^{r_2-1}\sum_{n_{L-1}=0}^{r_1-1}x[n_{L-1}(r_2r_3\cdots r_L)+n_{L-2}(r_3r_4\cdots r_L)+\cdots+n_1r_L+n_0]\times\\&W_{r_1r_2r_3\cdots r_L}^{[nL-1(r2r3\cdots rL)+nL-2(r3r4\cdots rL)+\cdots+n1rL+n0][kL-1(rL-1rL-2\cdots r1)+kL-2(rL-2rL-3\cdots r1)+\cdots+k1r1+k0]}=\\&\sum_{n_0=0}^{r_L-1}\sum_{n_1=0}^{r_{L-1}-1}\cdots\sum_{n_{L-2}=0}^{r_2-1}\left\{\left[\sum_{n_{L-1}=0}^{r_1-1}x[n_{L-1},n_{L-2},\cdots,n_1,n_0]\cdot W_{r_1}^{n_{L-1}k_0}\right]\cdot W_{r_1r_2r_3\cdots r_L}^{[nL-2(r3r4\cdots rL)+nL-3(r4r5\cdots rL)+\cdots+n1rL+n0]\cdot k0}\right\}\times\\&W_{r_1r_2r_3\cdots r_L}^{[nL-2(r3r4\cdots rL)+nL-3(r4r5\cdots rL)+\cdots+n1rL+n0][kL-1(rL-2rL-3\cdots r1)+kL-2(rL-3rL-4\cdots r1)+\cdots+k2r2r1+k1r1]}=\\&\sum_{n_0=0}^{r_L-1}\sum_{n_1=0}^{r_{L-1}-1}\cdots\sum_{n_{L-3}=0}^{r_3-1}\left\{\left[\sum_{n_{L-2}=0}^{r_2-1}X_1'[k_0,n_{L-2},\cdots,n_1,n_0]\cdot W_{r_2}^{n_{L-2}k_1}\right]\cdot W_{r_2r_3\cdots r_L}^{[nL-3(r4r5\cdots rL)+\cdots+n1rL+n0]\cdot k1}\right\}\times\\&W_{r_1r_2r_3\cdots r_L}^{[nL-3(r4r5\cdots rL)+nL-4(r5r6\cdots rL)+\cdots+n1rL+n0][kL-1(rL-1rL-2\cdots r1)+kL-2(rL-2rL-3\cdots r1)+\cdots+k2r2r1]}=\\&\sum_{n_0=0}^{r_L-1}\sum_{n_1=0}^{r_{L-1}-1}\cdots\sum_{n_{L-4}=0}^{r_4-1}\left\{\left[\sum_{n_{L-3}=0}^{r_3-1}X_2'[k_0,k_1,n_{L-3},\cdots,n_1,n_0]\cdot W_{r_3}^{n_{L-3}k_2}\right]\cdot W_{r_3r_4\cdots r_L}^{[nL-4(r5r6\cdots rL)+\cdots+n1rL+n0]\cdot k2}\right\}\times\\&W_{r_1r_2r_3\cdots r_L}^{[nL-4(r5r6\cdots rL)+nL-5(r6r7\cdots rL)+\cdots+n1rL+n0][kL-1(rL-1rL-2\cdots r1)+kL-2(rL-2rL-3\cdots r1)+\cdots+k3r3r2r1]}=\cdots\cdots=\\&\sum_{n_0=0}^{r_L-1}\left\{\left[\sum_{n_1=0}^{r_{L-1}-1}X_{L-2}'[k_0,k_1,\cdots,k_{L-3},n_1,n_0]\cdot W_{r_{L-1}}^{n_1k_{L-2}}\right]\cdot W_{r_{L-1}r_L}^{n_0k_{L-2}}\right\}\cdot W_{r_L}^{n_0k_{L-1}}=\\&\sum_{n_0=0}^{r_L-1}X_{L-1}'[k_0,k_1,\cdots,k_{L-3},k_{L-2},n_0]\cdot W_{r_L}^{n_0k_{L-1}}.\end{aligned}\tag{8}$$

由式(8)可知，当满足$r_1=r_2=\cdots=r_{L-1}=r_c$时，可将$N$点DFT分解为$(L-1)$个基-$r_c$FFT及一个基-$r_L$FFT级联的形式，从而缩短完成DFT运算所需时间，并解决基-r_cFFT算法无法实现r_c非整数次幂DFT算法的问题，因此本文提出的可变点FFT算法可将DIF基-4和基-2进行级联计算，且优化后的DIF基-4与基-2算法具有相同蝶形单元结构，更适合硬件实现混合基运算[13-14].

3 可变点FFT处理器的硬件架构设计及仿真和验证

3.1 FFT处理器流水线结构

基-2或基-4FFT处理器主要有4种流水线结构[15-16]，分别为基-2多路延时转换(R2MDC)结构、基-2单路延时反馈(R2SDF)结构、基-4单路延时反馈(R4SDF)结构与基4多路延时转换(R4MDC)结构. R2SDF和R4SDF比R2MDC和R4MDC能更有效利用存储器，R4SDF比R2SDF能更有效利用乘法器，但R2SDF比R4SDF具有更简单的蝶形结构及更低的控制复杂度. 在混合基算法中，基-2FFT流水线结构采用R2SDF结构，优化后的基-4FFT流水线结构采用改进的R2SDF结构，称为

基-2^2 单路延时反馈(R2^2SDF)结构. 以 16 点 FFT 为例，R2SDF 结构如图 5 所示.

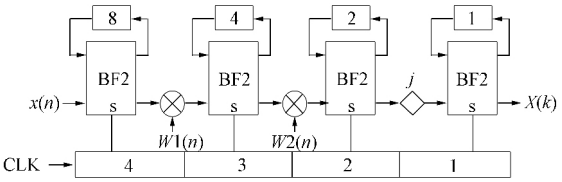


图 5　R2SDF 结构($N=16$)

Fig.5　Structure of R2SDF ($N=16$)

首先将输入数据分成上下两部分，上半部分数据串行输入第一级延时缓存器中，下半部分第一个数据与缓存单元中的第一个数据送入第一级基-2 蝶形单元(数据点间距为 $N/2$)进行运算，将二者之和送到下一级运算单元，二者之差送到本级的延时缓存器中，覆盖第一个数据，对所有数据依次进行上述处理，可得第一级蝶形运算的全部结果，结果的上半部分依次送入下一级继续计算，下半部分依次存入本级的延时缓存单元；对进入第二级基-2 蝶形运算单元的数据也分为上下两部分，上半部分数据串行输入第二级延时缓存器中，下半部分第一个数据与缓存单元中的第一个数据送入第二级基-2 蝶形单元进行运算(数据点间距为 $N/4$)，各级基-2 蝶形运算单元均采用相同的处理机制，从而保证各级数据流的连续性，最后得到计算结果.

由图 1 和图 2 可知，优化后的基-4 蝶形单元与基-2 蝶形单元具有相同结构，仅在 BF2Ⅰ阶段需乘以一个$-j$，对 R2SDF 结构进行改进得到优化后的基-4FFT 流水线单路延时反馈结构，其数据流的计算过程与 R2SDF 结构相同，如图 6 所示.

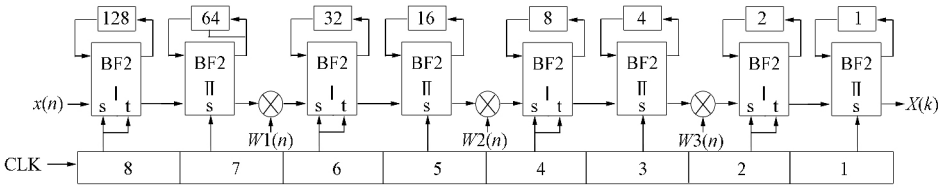


图 6　R2^2SDF 结构($N=256$)

Fig.6　Structure of R2^2SDF ($N=256$)

在图 6 的 BF2Ⅰ单元中，t 为控制输出与$-j$相乘的时钟，可实现实部与虚部位置互换. 不同流水线结构所需硬件资源及控制复杂性的比较列于表 1. 由表 1 可见，R2^2SDF 流水线结构在乘法器和存储器所需数量均最少，因此本文采用 R2^2SDF 结构.

表 1　不同流水线结构所需硬件资源及控制复杂性的比较

Table 1　Comparisons of hardware requirement and control complexity in different pipeline structures

结构	乘法器	加法器	存储器	控制复杂性
R2MDC	$2(\log_4 N-1)$	$4\log_4 N$	$3N/2-2$	简单
R2SDF	$2(\log_4 N-1)$	$4\log_4 N$	$N-1$	简单
R4SDF	$\log_4 N-1$	$8\log_4 N$	$N-1$	中等
R4MDC	$3(\log_4 N-1)$	$8\log_4 N$	$5N/2-4$	简单
R2^2SDC	$\log_4 N-1$	$4\log_4 N$	$N-1$	简单

3.2　可变点 FFT 处理器的硬件架构设计

采用基-4/2 混合基算法和流水线 R2^2SDF 结构设计可变点 FFT 处理器的硬件架构，如图 7 所示. 由图 7 可见，可通过增减蝶形单元实现不同点数的 FFT，从而实现 OFDMA 系统的核心功能. 各级运算模块结构类似，均包括控制单元、蝶形运算数据存储单元、旋转因子存储单元、复数乘法运算单元和蝶形运算单元五部分. 其中蝶形运算单元为核心部分，该单元完成 BF2，BF2Ⅰ和 BF2Ⅱ的复数加法运算，其运算单元结构如图 8 所示.

3.3　功能仿真验证

采用 Matlab 软件产生一个 $64=4^3$ 点的序列，作为仿真软件 Modelsim 和 FFT 处理器的输入，Modelsim 仿真结果如图 9(A)所示，通过 Quartus 中的 Signaltap 逻辑分析仪采样得到 FFT 处理器运行结果，如图 9(B)所示. 由图 9 可见，(A)和(B)的结果一致，表明设计的 FFT 处理器各功能模块及整

个系统满足设计要求，功能与时序正确.

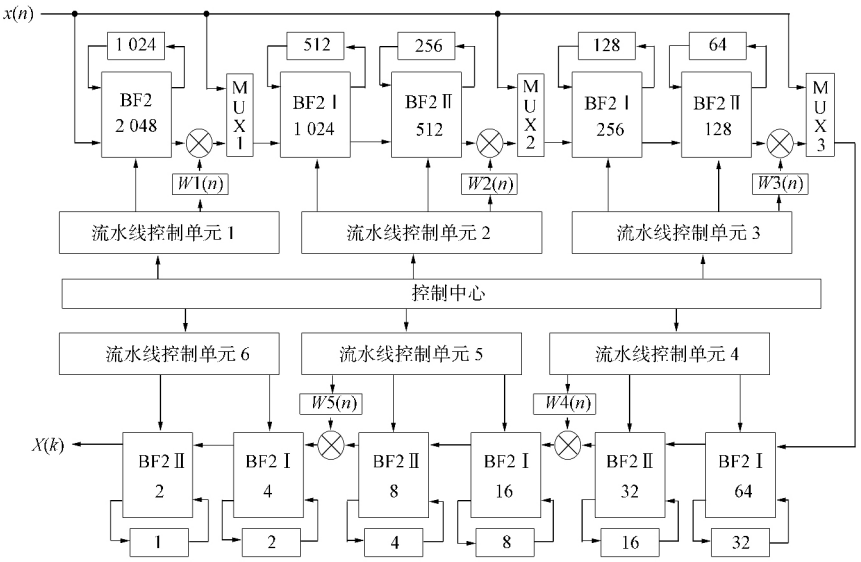


图 7 可变点 FFT 处理器的硬件架构

Fig.7 Hardware architecture of variable points FFT processor

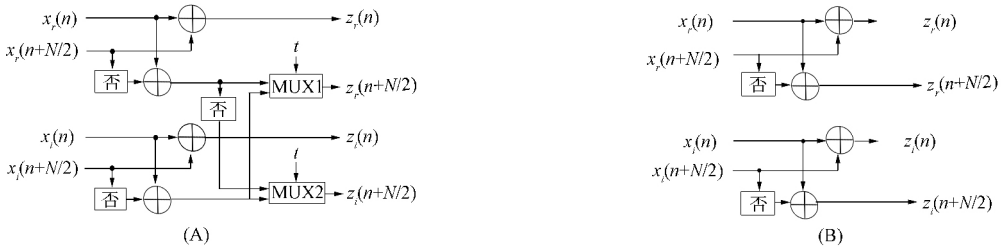


图 8 BF2Ⅰ(A)和 BF2/BF2Ⅱ(B)的运算单元结构

Fig.8 Operation unit structure of BF2Ⅰ (A) and BF2/BF2Ⅱ (B)

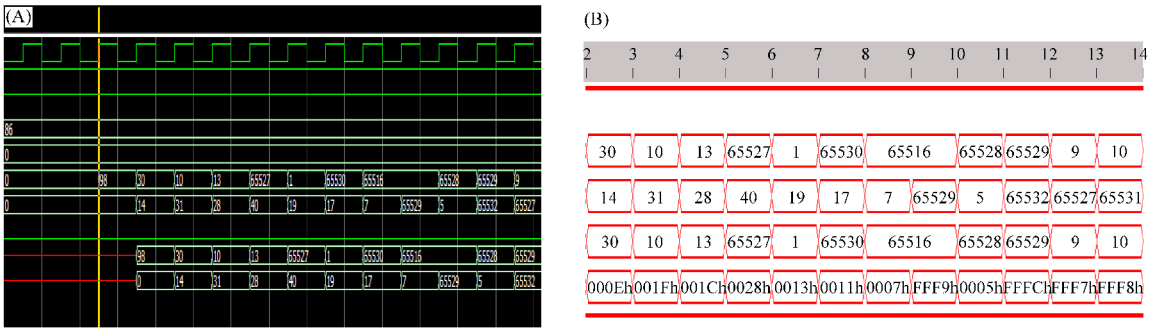


图 9 Modelsim 仿真结果(A)与 FFT 处理器运行结果(B)

Fig.9 Simulation results of modelsim (A) and operation results of FFT processor (B)

3.4 信号仿真验证

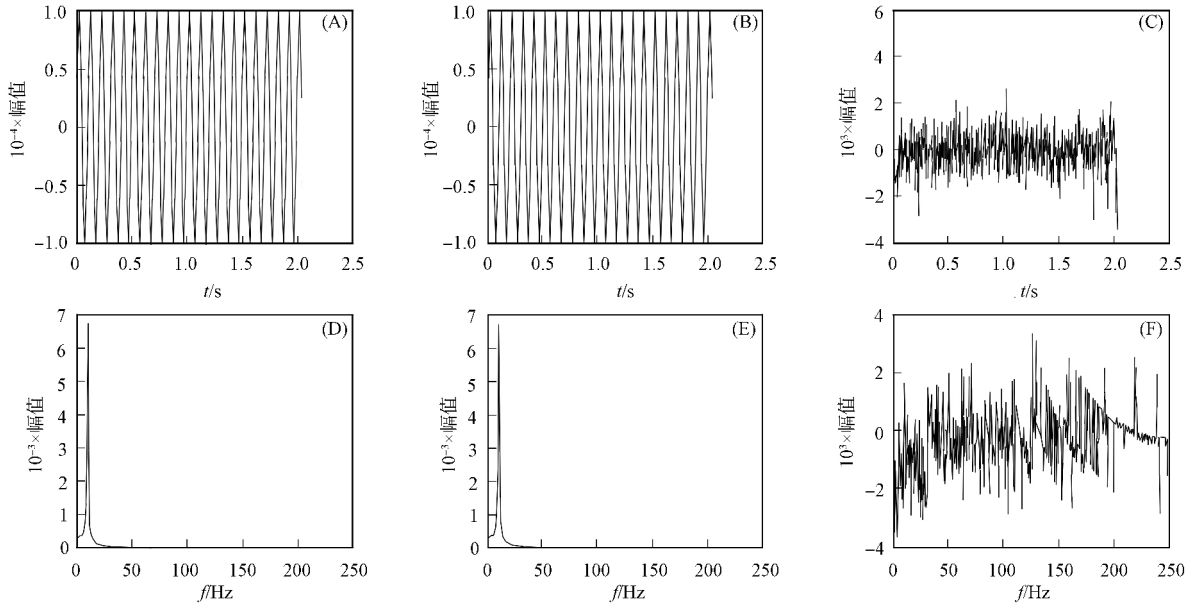
利用 Matlab 软件对正弦波和锯齿波进行采样，得到输入序列，将 FFT 处理器运算结果通过 Matlab 做 ifft 和生成频谱，并与 Matlab 中 fft( )函数产生的频谱进行比较.

3.4.1 **正弦波信号仿真** 利用 Matlab 函数产生一组 $1\ 024=4^5$ 点正弦波序列点，信号的采样频率为 500 Hz，为显示方便，幅值放大 10^4 倍.

$$x(t)=\sin(2\pi\times 10\times t). \tag{10}$$

通过 Matlab 中 fft( )函数产生的频谱和 FFT 处理器运行结果如图 10 所示. 由图 10 可见，FFT 处理器输出的结果通过函数 ifft( )得到的时域信号与输入正弦波信号相同，输出的频谱与 Matlab 所得

频谱一致，时域误差与频域误差极小.

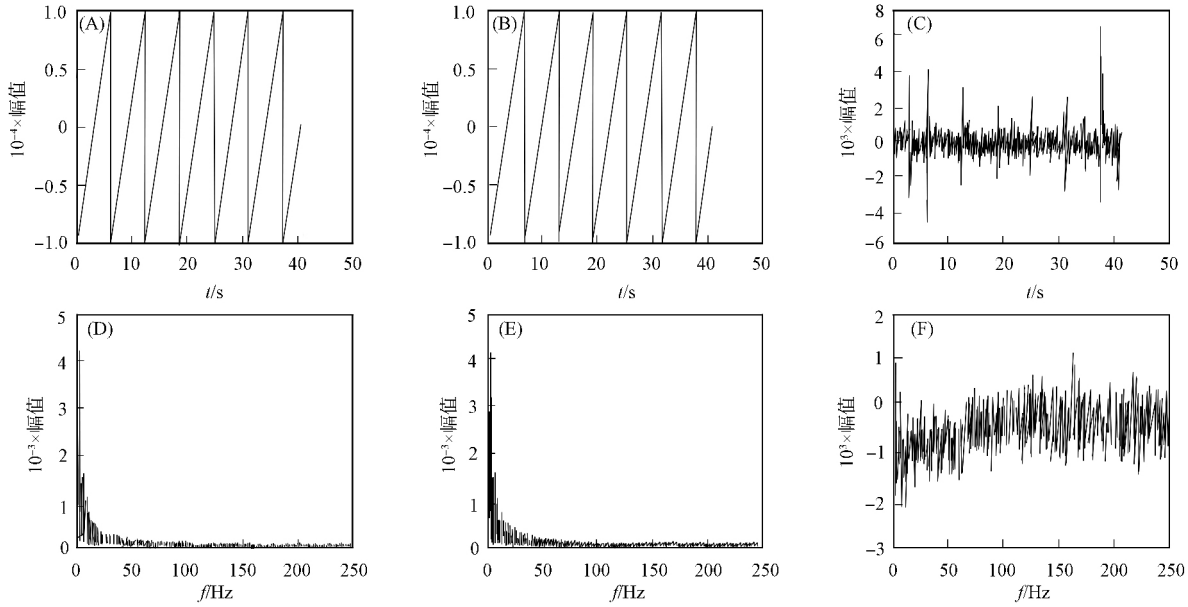


(A) 正弦信号时域信号($N=1\ 024$)；(B) 利用 IFFT 得到的时域信号；(C) 时域信号误差；
(D) Matlab FFT 运算结果；(E) 本文 FFT 运算结果；(F) 频域信号误差.

图 10　1 024 点正弦波运算结果

Fig.10　Operation results for 1 024 points sine wave

3.4.2　**锯齿波信号仿真与验证**　利用 Matlab 软件产生一组 $2\ 048=2\times4^5$ 点锯齿波序列点，作为输入信号，信号的采样频率为 50 Hz，为显示方便，幅值放大 10^4 倍，通过 Matlab 中 fft( )函数产生的频谱和 FFT 处理器运行结果如图 11 所示. 由图 11 可见，FFT 处理器输出的结果通过函数 ifft( )得到的时域信号与输入三角波信号相同，输出的频谱数据与 Matlab 所得频谱一致，时域误差与频域误差较小.



(A) 锯齿波信号时域信号($N=2\ 048$)；(B) 通过 IFFT 得到的时域信号；(C) 时域信号误差；
(D) Matlab FFT 运算结果；(E) 本文 FFT 运算结果；(F) 频域信号误差.

图 11　2 048 点锯齿波信号运算结果

Fig.11　Operation results for 2 048 points sawtooth wave

综上，本文设计了一种基于 FPGA 的可变点 FFT 处理器，采用 DIF 基-4/2 混合基算法，通过优

化使得基-4算法流图中具有基-2蝶形结构，有效减少了蝶形迭代的次数，降低了运算的复杂度，采用流水线 $R2^2SDF$ 结构，可减少所需存储器和乘法器的数量，提高各级间的运算速度，每级蝶形运算可在部分数据完成计算和存储后即开始新一级运算，实现多级运算交叉进行，进一步提高了FFT运算速度，降低控制难度. 最后通过实验对FFT处理器进行功能和信号的仿真验证，实验结果表明，FFT处理器的有效性和执行效率均满足OFDMA系统应用的需求.

参 考 文 献

[1] CHEN Jiyang, LEI Yuanwu, PENG Yuanxi, et al. Configurable Floating-Point FFT Accelerator on FPGA Based Multiple-Rotation CORDIC [J]. Chinese Journal of Electronics, 2016, 25(6): 1063-1070.

[2] 刘宝军，王中训，钟强，等. 基于FPGA的FFT算法设计与实现 [J]. 光电技术应用，2016，31(3)：46-49. (LIU Baojun, WANG Zhongxun, ZHONG Qiang, et al. Design and Implementation of FFT Algorithm Based on FPGA [J]. Electro-Optic Technology Application, 2016, 31(3): 46-49.)

[3] Prabhu E, Mangalam H, Karthick S. Design of Area and Power Efficient Radix-4 DIT FFT Butterfly Unit Using Floating Point Fused Arithmetic [J]. Journal of Central South University, 2016, 23(7): 1669-1681.

[4] 樊恩辰，姚馨，何书专，等. 基于SystemC的可配置FFT周期精确模型 [J]. 微电子学与计算机，2014，31(11)：83-87. (FAN Enchen, YAO Xin, HE Shuzhuan, et al. Cycle-Accurate Configurable FFT Modeling with System C [J]. Microelectronics & Computer, 2014, 31(11): 83-87.)

[5] Huang S J, Chen S G. A High-Throughput Radix-16 FFT Processor with Parallel and Normal Input/Output Ordering for IEEE 802.15.3c Systems [J]. IEEE Transactions on Circuits & Systems Ⅰ: Regular Papers, 2012, 59(8): 1752-1765.

[6] Joshi S P, Paily R. Distributed Arithmetic Based Split-Radix FFT [J]. Journal of Signal Processing Systems, 2014, 75(1): 85-92.

[7] Kim E J, Lee J H, Sunwoo M H. Novel Shared Multiplier Scheduling Scheme for Area-Efficient FFT/IFFT Processors [J]. IEEE Transactions on Very Large Scale Integration Systems, 2015, 23(9): 1689-1699.

[8] 王英喆，杜蓉. 基于FPGA流水线结构并行FFT的设计与实现 [J]. 电子设计工程，2015，23(4)：47-50. (WANG Yingzhe, DU Rong. A Pipelining Architecture Design of Parallel FFT Based on FPGA [J]. Electronic Design Engineering, 2015, 23(4): 47-50.)

[9] Harikrishna K, Rama R T, Labay V A. FPGA Implementation of FFT Algorithm for IEEE 802.16e (Mobile WiMAX) [J]. International Journal of Computer Theory and Engineering, 2011, 3(2): 197-203.

[10] Yang K J, Tsai S H, Chuang G C H. MDC FFT/IFFT Processor with Variable Length for MIMO-OFDM Systems [J]. IEEE Transactions on Very Large Scale Integration Systems, 2013, 21(4): 720-731.

[11] 许朋，周立青，刘宇航，等. 基于FPGA的高性能浮点型FFT处理器设计 [J]. 武汉大学学报(工学版)，2015，48(1)：120-124. (XU Peng, ZHOU Liqing, LIU Yuhang, et al. Design of a High-Performance Floating-Point Fast Fourier Transform Processor Based on FPGA [J]. Engineering Journal of Wuhan University, 2015, 48(1): 120-124.)

[12] 苏斌，刘畅，潘志刚. 基于FPGA的高速浮点FFT/IFFT处理器设计与实现 [J]. 中国科学院大学学报，2015，32(2)：259-263. (SU Bin, LIU Chang, PAN Zhigang. Design and Implementation of High-Speed Floating Points FFT Processor Based on FPGA [J]. Journal of University of Chinese Academy of Sciences, 2015, 32(2): 259-263.)

[13] 周景龙. 基于高速FFT结构的频域抗干扰算法的FPGA实现 [J]. 微电子学与计算机，2014，31(5)：32-35. (ZHOU Jinglong. An FPGA Implementation of Frequency-Domain Anti-jamming Algorithm Based on a Structure of High-Speed FFT [J]. Microelectronics & Computer, 2014, 31(5): 32-35.)

[14] Lakshmi D V V, Satya N C, Anil K R. Butterfly Design for RADIX-4K DIF FFT [J]. International Journal of Research in Computer and Communication Technology, 2014, 3(10): 1348-1353.

[15] WANG Zeke, LIU Xue, HE Bingsheng, et al. A Combined SDC-SDF Architecture for Normal I/O Pipelined Radix-2 FFT [J]. IEEE Transactions on Very Large Scale Integration Systems, 2015, 23(5): 973-977.

[16] 王天. 基于混合基算法的快速傅立叶变换和反变换在VDSL2中的应用与实现 [D]. 西安：西安电子科技大学，2009. (WANG Tian. Application and Implementation of FFT/IFFT Based on Mixed Radix in VDSL2 System [D]. Xi'an: Xidian University, 2009.)

(责任编辑：王 健)