文章编号:1000 - 582X(2005)03 - 0072 - 04

基于 CPLD 实现可扩展(I)FFT 处理器的设计*

刘晓明[1],熊　东[2],孙　学[1],鲁俊成[1]
(重庆大学 1. 通信工程学院;2. 电气工程学院,重庆　400030)

摘　要:提出了基于 CPLD(复杂可编程逻辑器件)实现傅立叶变换点数可灵活扩展的高速 FFT 处理器的结构设计以及各功能模块的算法实现,包括高组合数 FFT 算法的流水线实现结构、读/写 RAM 地址规律、补码实现短点数 FFT 阵列处理结构以及补码实现 CORDIC(坐标旋转数字计算机)算法的流水线结构等,输入数据速率为 20 MHz 时,1024 点 FFT 运算时间约为 50 us。

关键词:快速傅里叶变换;坐标旋转数字计算机;复杂可编程逻辑器件
中图分类号:TN402　　**文献标识码**:A

DFT(离散傅立叶变换)作为将信号从时域转换到频域的基本运算,在各种数字信号处理中起着核心作用,其快速算法 FFT(快速傅立叶变换)在无线通信、语音识别、图像处理、数字滤波和频谱分析等领域有着广泛的应用[1]。特别是随着 OFDM(正交频分复用)技术的出现,不同 OFDM 系统需要不同变换点数 FFT,如何更快速、更灵活地实现 FFT 变得越来越重要。

1　组合数 $N = r_1 r_2$ 点混合基 FFT 原理及其快速算法的 MATLAB 仿真

1.1　$N = r_1 r_2$ 点 DFT 原理

计算 N 点 DFT[1]:

$$X(k) = \mathrm{DFT}[x(n)] = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$
$$(k = 0,1,\cdots,N-1) \qquad (1)$$

若 $N = r_1 r_2$ 的组合数,可将 $n(n < N)$ 表示为:

$$n = n_1 r_2 + n_0$$
$$(n_1 = 0,1,\cdots,r-1),(n_0 = 0,1,\cdots,r_2 - 1)$$

将表示为:　　$k = k_1 r_1 + k_0$

$$(k_1 = 0,1,\cdots,r_2 - 1),(k_0 = 0,1,\cdots,r_1 - 1)$$

令　　$x(n) = x(n_1 r_2 + n_0) = x(n_1, n_0)$

$$X(k) = X(k_1 r_1 + k_0) = X(k_1, k_0)$$

则式(1)可以表示为:

$$X(k_1,k_0) = \sum_{n_0=0}^{r_2-1} \{[\sum_{n_1=0}^{r_1-1} x(n_1,n_0) W_{r_1}^{n_1 k_0}] W_N^{n_0 k_0}\} W_{r_2}^{n_0 k_1} \qquad (2)$$

式(2)的意义在于,计算组合数 $N = r_1 r_2$ 点 DFT,等价于先求出 r_2 组 r_1 点的 DFT,其结果经过对应旋转因子 $W_N^{n_0 k_0}$ 的相位旋转后,再计算 r_1 组 r_2 点的 DFT。实际应用中,DFT 往往用它的快速算法 FFT 实现,因而式(2)中的点 DFT 和 r_2 点 DFT 都用 r_1 点 FFT 和 r_2 点 FFT 实现。

1.2　信号的直接 FFT 和混合基 FFT 的 MATLAB 仿真

式(2)提供了实现变换点数可扩展的 FFT 处理器的理论基础,使得大点数(比如 $N > 1024$)FFT 实现时便于模块化设计,实现结构也更加灵活[2]。通过 MATLAB 仿真组合数 $32 = 8 \times 4$ 点混合基 FFT,直观地说明式(2)理论的实现过程;其结果与 32 点直接 FFT 结果进行比较,证明理论的正确性和具有模块化实现的特性。

1.2.1　信号的直接 FFT 的 MATLAB 仿真

$f(t) = 225\cos(t) + 200\cos(3t + 3pi/4) + 175 \cdot \cos(9t - pi/4)$ 进行 32 点/周期的抽样得到周期序列 $f(n)$,取其 1 个周期的 32 个抽样值进行 32 点 FFT(调用 MATLAB 中的函数"fft"对 $f(n)$ 作 FFT),输出 $F(k)$

* 收稿日期:2004 - 10 - 08
基金项目:国家自然科学基金资助项目(50305037)
作者简介:刘晓明(1963 -),男,重庆人,重庆大学教授,博士后,主要从事测控系统数字化、软化、智能化的研究。

和其幅度值$|F(k)|$，如图 1 所示。设信号周期为 T，抽样周期 Ts，则 $NTs=32Ts=T$；频率分辨率 $=1/(NTs)=1/T=$信号频率，所以 $f(t)$ 的 1 次、3 次和 9 次谐波分量分别对应 $F(1)$、$F(3)$ 和 $F(9)$，在图 1 的 $F(k)$ 的极坐标图中以实线表示，虚线表示它们的共轭对称分量。由于 $f(t)$ 没有直流分量和其他谐波分量，所以在其他 k 值处 $F(k)$ 为 0，与理论相符。

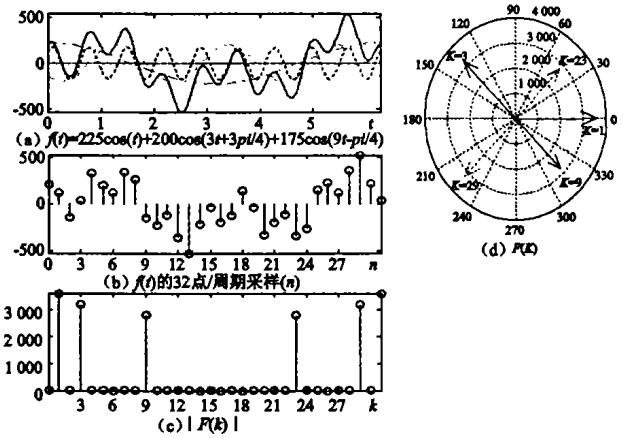

图 1　$f(t)$、$f(n)$、$F(k)$ 以及 $|F(k)|$ 的仿真图

1.2.2　信号的混合基 FFT 的 MATLAB 仿真

按照图 2 的实现结构，对 $f(n)$ 的 8×4 点混合基 FFT 的运算过程进行 MATLAB 仿真，结果如图 3 所示。

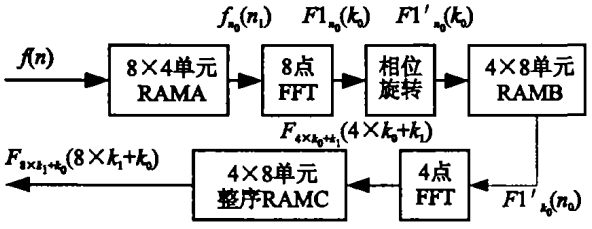

图 2　8×4 点混合基 FFT 运算过程

首先将 32 点采样数据 $f(n)$ 按参变量 n_0，变量 n_1 进行 $N=8\times4$ 分组得到 $f_{n_0}(n_1)$，其中 $(n_1=0,1,\cdots,7;\ n_0=0,1,2,3)$，对 4 组 $f_{n_0}(n_1)$ 分别求 8 点的 FFT 输出 $F1_{n_0}(k_0)$ $(k_0=0,1,\cdots,7)$；接着对 $F1_{n_0}(k_0)$ 进行相位旋转，乘以相位旋转因子 $W_N^{n_0k_0}$ 得到 $F1'_{k_0}(n_0)$；按参变量 k_0，变量 n_0 对 $F1'_{k_0}(n_0)$ 进行 4×8 分组得到 $F1'_{k_0}(n_0)$，如图 3 所示；对 8 组 $F1'_{k_0}(n_0)$ 分别求 4 点的 FFT 输出 $F_{4\times k_0+k_1}(4\times k_0+k_1)$；最后对 $F_{4\times k_0+k_1}(4\times k_0+k_1)$ 整序输出 $F_{8\times k_1+k_0}(8\times k_1+k_0)$，如图 4 所示。

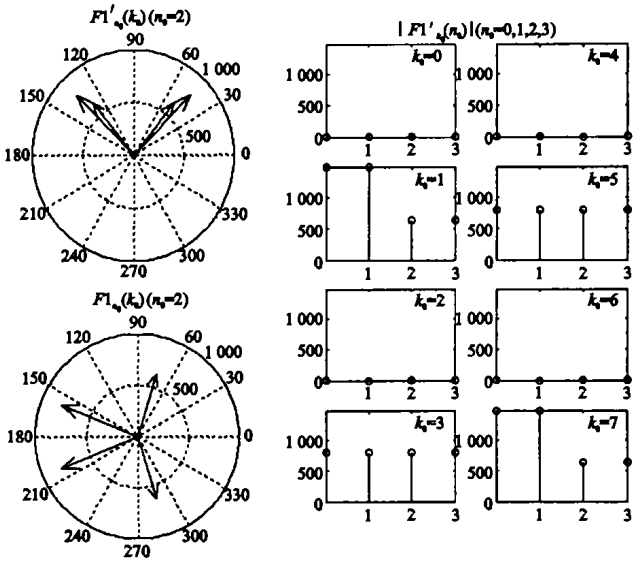

图 3　$F1_2(k_0)$ 的 8 点数据相位旋转后 $F1'_2(k_0)$ 以及 $|F1'_{n_0}(k_0)|$ 的重新分组 $|F1'_{k_0}(n_0)|$

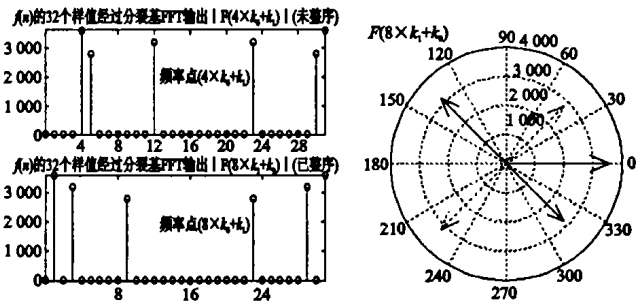

图 4　$|F1'_{k_0}(n_0)|$ 的 FFT 运算模值 $|F_{4\times k_0+k_1}(4\times k_0+k_1)|$ 以及整序后的 $F_{8\times k_1+k_0}(8\times k_1+k_0)$ 和其模值

1.2.3　直接 FFT 和混合基 FFT 的 MATLAB 仿真结果比较

比较图 1 中对 $f(n)$ 进行 32 点直接的 FFT 结果和图 3 中对 $f(n)$ 进行 8×4 点混合基的 FFT 结果可以发现，22 种方法的变换结果的幅度和相位都相同，说明混合基 FFT 的算法原理完全正确；同时也证明了长点数的 FFT 可以通过短点数 FFT 处理器的结构扩展加以计算。

2　IDFT 和 DFT 的相互转换关系

IDFT 和 DFT 的相互转换关系如下：

$$x(n)=\mathrm{IDFT}[X(k)]=\frac{1}{N}\{\sum_{k=0}^{N-1}[X(k)]^*W_N^{nk}\}^*=\frac{1}{N}\{\mathrm{DFT}[X^*(k)]\}^*$$

其中 $(n=0,1,\cdots,N-1)$。计算 IDFT 时，只需要把输入数据和输出数据取其共轭，就可以在 DFT 的原有结构上实现 IDFT。

3 FFT 实现结构

3.1 FFT 处理器整体结构

根据式(2)的算法原理设计 FFT 处理器的可扩展结构,其结构原理框图如图 5 所示。采用流水线模块化级联结构,设计模块可以重复利用,减少了设计量[4-5]。

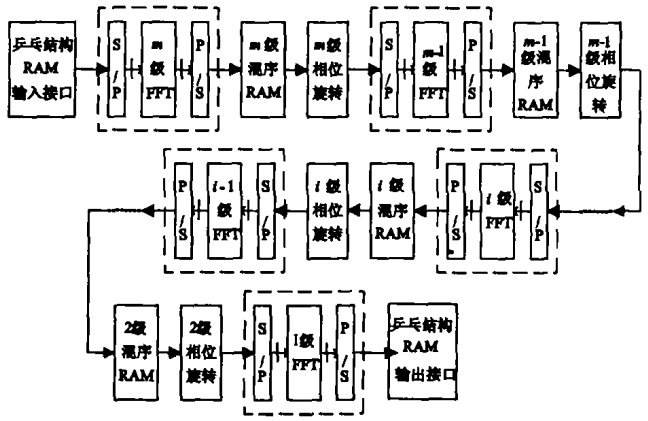

图 5 可扩展 FFT 处理器流水线结构的原理模型

3.2 短点数 FFT 阵列结构

由于 $W_2^{nk}=e^{-(\pi)nk}$, $W_4^{nk}=e^{-(\pi/2)nk}$, $W_8^{nk}=e^{-(\pi/4)nk}$, 所以 2 点、4 点和 8 点 FFT 用 Cooley - Tukey 算法结构实现时,有大量的复数乘法实际上转化为加减运算,所以用阵列结构实现不但具有速度快的优点,而且所用器件资源少。

3.3 相位旋转运算单元

对于前几级短点数 FFT 级间相位旋转处理单元,可以采用 ROM 存储旋转因子与数据复乘进行相位旋转的传统方法。当需要的 ROM 存储单元较多时,采用流水线结构 CORDIC 运算器[3,6]。

复数 $P=x+jy$ 旋转角度 θ 得到 Q 的表达式:

$$Q=(x+jy)e^{j\theta}=(x\cos\theta-y\sin\theta)+j(y\cos\theta+x\sin\theta) \quad (3)$$

如果旋转角度 θ 可以分解成 n 个小角度 ϕ_i 之和,即:

$$\theta=\phi_0\pm\phi_1\pm\phi_2\pm\cdots\pm\phi_{n-1}=\sum_{i=0}^{n-1}\delta_i\phi_i$$

其中$(\phi_i>0;\delta_i=\pm1),(i=0,1,\cdots,n-1)$

令 $x=R_0, y=I_0$ 则旋转 $\delta_i\phi_i$ 角度得到:

$$R_{i+1}=[R_i-I_i\tan(\delta_i\phi_i)]\cos(\delta_i\phi_i)$$

$$I_{i+1}=[I_i+R_i\tan(\delta_i\phi_i)]\cos(\delta_i\phi_i)$$

由于有以下性质:

$$\cos\phi_0\cos\phi_1\cos\phi_2\cdots\cos\phi_{n-1}=1/(\prod_{i=0}^{n-1}\sqrt{1+2^{-2i}})\xrightarrow{n>12}1/1.647$$

所以在电路设计时,更常用迭代公式

$$\begin{cases}\hat{R}_{i+1}=\hat{R}_i-\hat{I}_i\delta_i2^{-i}\\ \hat{I}_{i+1}=\hat{I}_i+\hat{R}_i\delta_i2^{-i},\ (x=\hat{R}_0,y=\hat{I}_0)\end{cases} \quad (4)$$

只需要进行简单的移位和加减运算计算出 $\hat{R}_n$, $\hat{I}_n$,最后经过 1.647 倍的幅度校正即可计算出 R_n, I_n。设计时采用补码实现移位和加减运算,实现结构简单,消耗资源少,如图 6 所示。

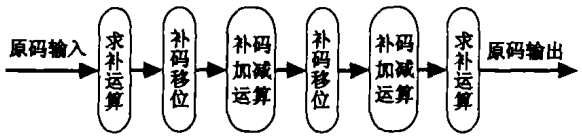

图 6 CORDIC 处理器的补码实现框图

3.4 RAM 结构

设计的 RAM,每个存储单元为 32 bit,高 16 位为复数的实部,低 16 位为复数的虚部。输入输出数据接口用 RAM 设计为乒乓结构,基本思想就是用 2 块相同的 RAM 交替读出或交替写入外部输入数据;级与级之间数据混序用 RAM 设计为读/写 RAM,基本思想就是在同一块 RAM 中用 2 个时钟完成一次读/写操作,即在同一地址单元进行前一时钟读,后一时钟写操作。

3.4.1 输入数据接口用 RAM

同一个 FFT 处理器的不同点数处理模式的输入接口用 RAM 可以用统一的 RAM 地址信号读写不同变换点数的外部数据。例如 512 点 FFT 处理器的输入数据接口用 RAM 可以分解成$((1\times8)\times8)\times8$ 个存储单元,即 8 个 8 单元 RAM 扩展成一个 64 单元 RAM,8 个 64 单元 RAM 扩展成一个 512 单元 RAM。计算 64 点 FFT 时,利用 512 单元 RAM 的第一块 64 单元 RAM,计算 8 点 FFT 时,利用 64 单元 RAM 的第一块 8 单元 RAM 用于输入数据接口。

3.4.2 输出整序用 RAM 地址发生器

这里举例 4×4FFT 输出整序用 RAM 的读写地址规律来说明其工作原理,如图 7 所示。

数据次序为 counter[1..0]&counter[3..2],数据写入 RAM 中的地址信号为 $address_1$ = counter[3..2]& counter[1..0],读 RAM 中的数据的地址信号为 $address_2$ = counter[1..0]&counter [3..2]。采用乒乓结构的 2 块 RAM 实现独立地同一块 RAM 数据的读写操作。

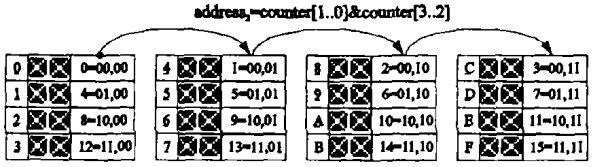

图 7 4×4FFT 输出整序用 RAM 地址规律示意图

3.4.3 级与级之间数据混序用流水线读/写 RAM 地址发生器

级与级之间数据混序用流水线读/写 RAM 在整个 FFT 处理器的结构中是一个关键模块，它的地址控制逻辑也最为复杂，由于地址有一循环移位的规律，虽然设计难度大，但根据规律进行 FFT 的结构扩展还是很方便的。

4 结 语

以混合基 DFT 算法为基础，设计出了应用于多模式正交频分复用(OFDM)系统的变换点数可扩展的流水线方式的复数(I)FFT 处理器的实现结构。读/写 RAM 工作在 40 MHz 时钟，输入/输出数据速率为 20 MHz时，1024 点 FFT 运算时间约为 50 us。采用模块化设计结构，便于系统调试和实现，而且模块化设计可以重复利用设计模块，避免重复相同的设计，从而缩短芯片设计开发时间；而采用混合基 DFT 算法设计系统结构，更易于 FFT 处理器的结构扩展；整个(I)FFT 设计结构新颖，实现容易，具有一定实用价值。

参考文献：

[1] 程佩清. 数字信号处理教程[M]. 北京：清华大学出版社，2001.

[2] 侯伯亨，顾新. VHDL 硬件描述语言与数字逻辑电路设计[M]. 西安：西安电子科技大学出版，1999.

[3] STEPHAN W MONDWURF. Benefits of the Cordic-algorithm in a Versatile Cofdm Modulator/Demodulator Design[Z]. Fourth IEEE International Caracas Conference on Devices, Circuits and Systems, Aruba, April 17 - 19, 2002.

[4] MA Y, WANHAMMAR L. A Hardware Efficient Control of Memory Addressing for High Performance FFT Processors[J]. IEEE Transactions on Signal Processing, 2000, 48(3): 917 - 921.

[5] VOLDER J E. The CORDIC Trigonometric Computing Technique[J]. IRE Trans on Electronic Computers, 1959, 8(3):330 - 334.

[6] DESPAIN A M. Fourier Transform Computers Using CORDIC Iterations[J]. IEEE Trans. on Computers, 1993, c - 23(10): 993 - 1 001.

Design of the Expandable (I) FFT Implemented in the CPLD

LIU Xiao-ming[1], *XIONG Dong*[2], *SUN Xue*[1], *LU Jun-cheng*[1]

(1. College of Communication Engineering;
2. College of Electrical Engineering, Chongqing University, Chongqing 400030, China)

Abstract: The architecture of scalable length and high speed FFT processor based on CPLD (Complex Programmable Logic Device) is proposed, including the pipeline architecture of the radix mixed FFT algorithm, the address regularity of the read-then-write RAM, the array architecture of short-length FFT and the pipeline complement architecture of CORDIC (Coordinate Rotation Digital Computer) algorithm. If the input-data velocity is 20 MHz, the time expended on 1024-point FFT is 50 us.

Key words: FFT; CORDIC; CPLD

(编辑　张　苹)