文章编号:1000 - 582X(2004)09 - 0033 - 04

FPGA 实现流水线结构的 FFT 处理器*

朱冰莲,刘学刚
(重庆大学 通信工程学院,重庆 400030)

摘 要:针对高速实时信号处理的要求,介绍了用现场可编程逻辑阵列(FPGA)实现的一种流水线结构的 FFT 处理器方案。该 FFT 处理器能够对信号进行实时频谱分析,最高工作频率达到 75 MHz。通过对采样数据进行加窗处理来减少了频谱泄漏产生的误差。为了提高 FFT 工作频率和节省 FPGA 资源,采用了由 1 024 点复数 FFT 计算 2 048 点实数 FFT 的算法。此外还介绍了一种计算复数模值的近似算法。

关键词:流水线;快速傅立叶变换;现场可编程逻辑阵列
中图分类号:TN 911.6 **文献标识码**:A

FFT 算法多种多样,按数据组合方式不同一般分为按时间抽取和按频率抽取,按数据抽取方式的不同又可分为基 2,基 4 等。FFT 的实现方法也多种多样,可以用软件或硬件实现,也可以用软硬件结合的方式实现。用软件实现计算速度很慢,一般用于离线处理,软、硬件结合方式实现如用单片机或 DSP 实现在速度不高的情况可以实现在线实时处理,但是在高速的场合仍然不能满足要求[1]。针对快速信号处理的要求及 FPGA 器件的特点,提出了一种基于 FPGA 实现的基 2 抽取流水线结构的 FFT 算法。为了提高 FFT 工作频率和节省 FPGA 资源,利用 1 024 点复数来计算 2 048 点实数的 FFT。此外为了减少频谱泄漏的影响,在 FFT 运算前先对数据进行了加窗处理。观察信号的频谱分布,通常需要对 FFT 运算后的复数数据进行取摸运算,但是硬件实现取模运算非常困难。笔者提出了复数取模的一种近似算法,平均误差不超过 0.6 %。

1 FFT 处理器的 FPGA 设计

FFT 处理器是在 ALTERA 公司的 Quartus 系统中开发的,选用基于查找表、乘积项、嵌入式存储器的多核结构的 APEX20K 系列器件。乘法器、双口 RAM、ROM 通过调用库中模块实现,加法器、饱和处理运算、复数求模运算、地址产生单元及其缓冲单元使用 VHDL 语言编程实现。

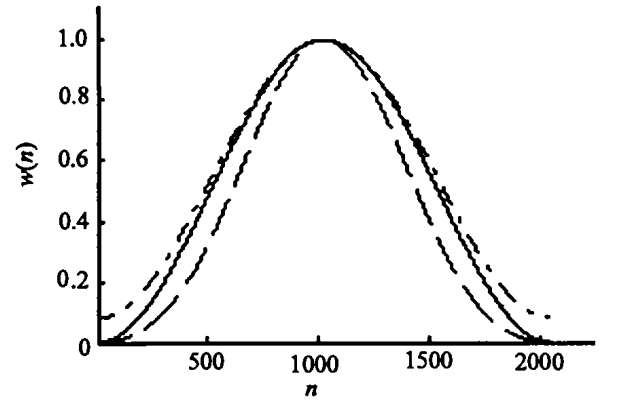
1.1 加窗处理运算

为了减少时域截断造成的频谱泄漏误差,最常用的方法就是变换前对采样数据进行加窗处理运算[2-4]。用于信号处理的窗函数很多,工程上常用的是矩形窗、汉宁窗、高斯窗、海明窗、布拉克曼窗等。由图 1 可以看出,海明窗的主瓣宽度比较窄,旁瓣衰减在 40 dB 以上,而且窗函数的形状比较平坦,可以用比较少的位数对窗函数数值进行量化。高斯窗的形状虽然可以调节,但是它的主瓣衰减太慢。

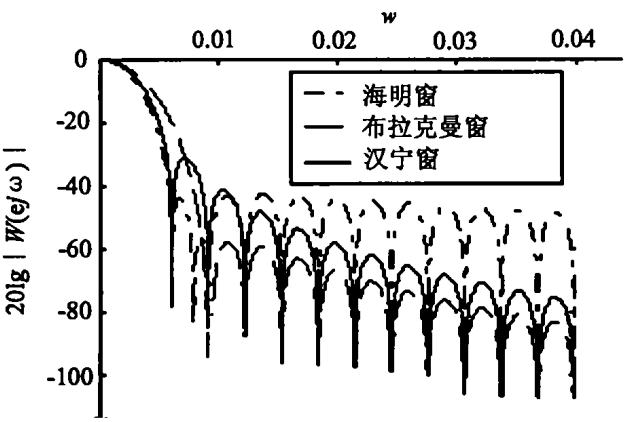
笔者设计的 FFT 处理器是用来分析叠加有多个窄带干扰的直接序列扩频通信系统中的信号,要求精确给出每个窄带干扰的中心频率及其干扰强度的相对大小,因此海明窗更适合本文中的信号处理要求。由于窗函数的值是中心对称的,因此只需存储窗函数的前 $N/2$ 个点,这样可以节省一半的 ROM 存储空间。海明窗函数表达式如下:

$$w(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right)$$
$$n = 0,1,\cdots,N-1 \qquad (1)$$

* 收稿日期:2003 - 09 - 25
基金项目:重庆市科委应用基础研究基金资助项目(7964)
作者简介:朱冰莲(1959 -),女,四川富顺人,重庆大学副教授,博士,主要从事信号处理及其应用的研究。



(a) 时域



(b) 频域

图1 几种窗函数的实频波形比较

1.2 流水线结构的FFT实现

1.2.1 N点复数FFT计算$2N$点实数FFT的算法

首先把输入序列x(n),n = 0,1,…,2N − 1按奇偶分组,生成2个新的序列 $x_1(n)$ 和 $x_2(n)$:

$$x_1(n) = x(2n) \quad n = 0,1,\cdots,N-1 \tag{2}$$

$$x_2(n) = x(2n+1) \quad n = 0,1,\cdots,N-1 \tag{3}$$

在将 $x_1(n)$ 和 $x_2(n)$ 组成一个复数序列 $y(n)$:

$$y(n) = x_1(n) + jx_2(n) \tag{4}$$

根据DFT的性质容易导出[5]:

$$Y(k) = X_1(k) + jX_2(k) \quad k = 0,1,\cdots,N-1 \tag{5}$$

$$X_1(k) = \frac{1}{2}[Y(k) + Y^*(N-k)]$$

$$k = 0,1,\cdots,N-1 \tag{6}$$

$$X_2(k) = \frac{1}{2j}[Y(k) - Y^*(N-k)]$$

$$k = 0,1,\cdots,N-1 \tag{7}$$

$$X(k) = X_1(k) + W_{2N}^k X_2(k)$$

$$k = 0,1,\cdots,N-1 \tag{8}$$

$$X(k+N) = X_1(k) - W_{2N}^k X_2(k)$$

$$k = 0,1,\cdots,N-1 \tag{9}$$

可见计算出复数序列 $Y(k)$ 后,通过一个蝶形运算可以得到 $X_1(k)$ 和 $X_2(k)$,再经过一个蝶形运算即可得到所求的 $X(k)$。该算法最大的优点是能够提高FFT处理器的工作频率。因为此算法不直接的计算 $X(k)$,而是先计算 $Y(k)$ 再由 $Y(k)$ 计算 $X(k)$,计算 $Y(k)$ 时每一级蝶形运算的数量减少了一半,这样输入采样序列的速率可以提高一倍,也即是FFT的工作频率提高了一倍。

1.2.2 FFT的FPGA实现框图

1 024点复数FFT运算,按照基2频率抽取运算分成10级,每级包括1个双口RAM,1个地址产生器,1个旋转因子ROM表,1个蝶形运算单元,2个选择缓冲单元,如图2所示。

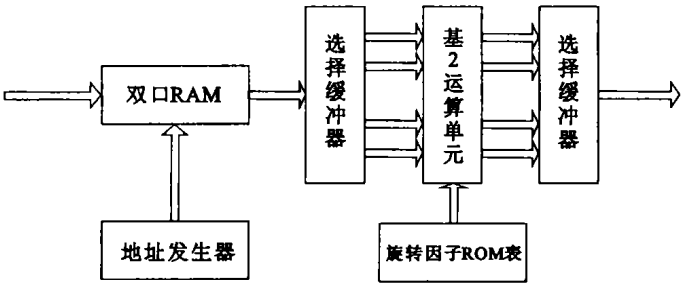


图2 每级FFT运算框图

为了简化地址单元电路,将复数数据的实部虚部组合成一个数据存储在RAM中。由于使用双口RAM,当一个存储单元中的数据读出做运算时,该存储单元就能够存储上一级来的数据,因此这种结构的FFT可以进行流水线操作,能够对信号样本进行实时连续的运算。选择缓冲器的用途是拉齐数据,将RAM输出的2个复数数据拆成4个实数数据输入到蝶形运算单元,完成蝶形运算后的数据进入选择缓冲器组合成2个复数数据输出。

1.2.3 地址产生单元电路

为了简单,用8点FFT实例说明地址产生单元的设计。假设以自然顺序进入FFT处理单元的数据为 $y(n)$,$y(0)$ ~ $y(7)$ 组成第1个数据块,$y(8)$ ~ $y(15)$ 组成第2个数据块用,依次类推。用 $y^1(n)$ 表示FFT的第1级输出,$y^2(n)$ 表示FFT的第2级输出,$y^3(n)$ 表示FFT的第3级输出。

由表1可以看出各块数据在第1级双口RAM中的存储地址顺序应该为:第1块数据按照自然顺序存储;第2块数据按照 $0,\frac{N}{2},1,\frac{N}{2}+1,2,\frac{N}{2}+2,\cdots,i,\frac{N}{2}+i,\cdots,\frac{N}{2}-1,N-1$ 顺序存储;第3块数据按照 $0,\frac{N}{4},\frac{N}{2},$

$\frac{N}{2}+\frac{N}{4},1,\frac{N}{4}+1,\frac{N}{2}+1,\frac{N}{2}+\frac{N}{4}+1,\cdots,\frac{N}{4}-1,\frac{2N}{4}-1,\frac{N}{2}+\frac{N}{4}-1,\frac{N}{2}+\frac{2N}{4}-1$ 顺序存储；第 4 块数据又是按照自然顺序存储。通过分析数据在第 1 级双口 RAM 中的存储地址顺序，可以发现一个规律，下一块数据的存储地址可以由上一块数据的存储地址循环右移得到，即把地址的最低位移到地址的最高位，其余地址向右移。

表 1　FFT 数据在双口 RAM 中的存储顺序

数据在第一级双口 RAM 中的存储顺序					数据在第二级双口 RAM 中的存储顺序			
RAM 地址	第 1 块数据	第 2 块数据	第 3 块数据	第 4 块数据	RAM 地址	第 1 块数据	第 2 块数据	第 3 块数据
0	$y(0)$	$y(8)$	$y(16)$	$y(24)$	0	$y^1(0)$	$y^1(8)$	$y^1(16)$
1	$y(1)$	$y(10)$	$y(20)$	$y(25)$	1	$y^1(4)$	$y^1(10)$	$y^1(20)$
2	$y(2)$	$y(12)$	$y(17)$	$y(26)$	2	$y^1(1)$	$y^1(9)$	$y^1(17)$
3	$y(3)$	$y(14)$	$y(21)$	$y(27)$	3	$y^1(5)$	$y^1(11)$	$y^1(21)$
4	$y(4)$	$y(9)$	$y(18)$	$y(28)$	4	$y^1(2)$	$y^1(12)$	$y^1(18)$
5	$y(5)$	$y(11)$	$y(22)$	$y(29)$	5	$y^1(6)$	$y^1(14)$	$y^1(22)$
6	$y(6)$	$y(13)$	$y(19)$	$y(30)$	6	$y^1(3)$	$y^1(13)$	$y^1(19)$
7	$y(7)$	$y(15)$	$y(23)$	$y(31)$	7	$y^1(7)$	$y^1(15)$	$y^1(23)$

第 1 级的地址产生可以用从 0 到 $N-1$ 的地址计数器来产生，用一个块数计数器即 0 ~ $\log_2 N-1$ 来控制地址计数器的循环移位，当块计数器为 0 时，地址计数器输出不移位，当块计数器为 1 时，地址计数器输出循环右移一位，同理，当块计数器为 2 时，地址计数器输出循环右移2位，依次类推。电路原理图如图3 所示。

同样的分析方法，可以得到第 2 级地址产生规律：当块计数器为偶数时，地址输出等于计数器输出的地址；当块计数器为奇数时，地址输出等于交换计数器输出的最低位和最高位得到的地址。经过分析，第 3 级及其以后各级的地址产生方法与第 2 级地址产生方法相同，只是地址宽度逐级递减。电路原理图如图 4 所示。

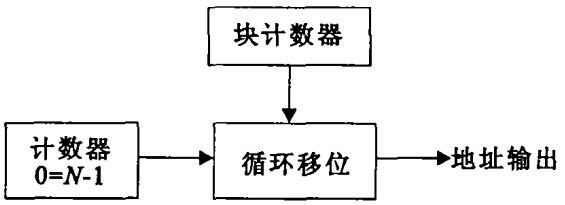


图 3　第一级地址产生电路

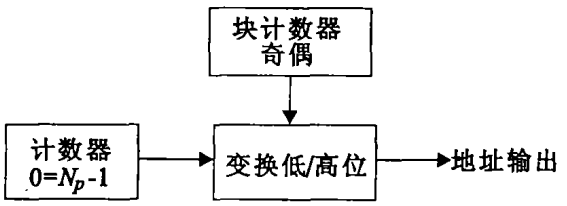


图 4　第二级地址产生电路

1.2.4　蝶形运算单元电路

蝶形单元是 FFT 算法的基本的操作，它由 1 个复数加法器、1 个复数减法器和 1 个旋转因子的复数乘法器组成。图 5 为蝶形处理单元示意图，A，B 表示蝶形结输入的复数数据，C，D 表示蝶形结的输出结果，W_N^k 表示旋转因子。为了防止运算结果溢出，在蝶形结的 2 个支路都乘以 1/2 的比例因子[5]。

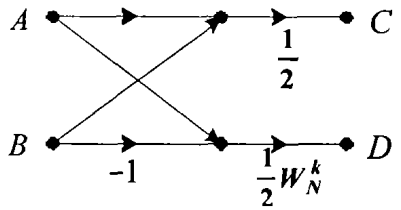


图 5　蝶形处理单元示意图

将复数用实部、虚部表示：$A=a_r+ja_i$，$B=b_r+jb_r$，$C=c_r+jc_i$，$D=d_r+jd_i$ 则有

$$c_r=\frac{1}{2}(a_r+b_r) \tag{10}$$

$$c_i=\frac{1}{2}(a_i+b_i) \tag{11}$$

$$d_r=\frac{1}{2}\left[(a_r-b_r)\cos\left(\frac{2\pi k}{N}\right)+(a_i-b_i)\sin\left(\frac{2\pi k}{N}\right)\right] \tag{12}$$

$$d_i=\frac{1}{2}\left[(a_i-b_i)\cos\left(\frac{2\pi k}{N}\right)-(a_r-b_r)\sin\left(\frac{2\pi k}{N}\right)\right] \tag{13}$$

由式(10) 和(11) 可知 c_r 和 c_i 肯定不会溢出。由式(12) 和(13) 可知 d_r 和 d_i 有可能溢出，但是 d_r 和 d_i 溢出的概率很小，因此可以对 d_r 和 d_i 作饱和处理，即计算结果正向溢出时，d_r 和 d_i 用最大正值表示；计算结果反向溢出时，d_r 和 d_i 用最大负值表示；计算结果不发生溢出时，d_r 和 d_i 等于计算值。虽然饱和处理会引入一定的误差，但是由于溢出的概率很小，所以误差可以忽略。

1.3　求复数模值算法

对 FFT 变换结果取模，既能节省存储 FFT 变换结果需要的 ROM 空间，又方便观察信号的频谱分布。但

是用硬件来实现复数求模运算很困难[6]，因而在此采用了一种近似算法。

由三角公式、求模公式可以推导出 $|X(k)|$ 的范围如下：

$$\max[|\mathrm{Re}X(k)|,|\mathrm{Im}X(k)|] \leqslant |X(k)| \leqslant \max[|\mathrm{Re}X(k)|,|\mathrm{Im}X(k)|] + \frac{\min[|\mathrm{Re}X(k)|,|\mathrm{Im}X(k)|]}{1+\sqrt{2}} \quad (14)$$

取 $|X(k)|$ 的近似值为：

$$|X(k)| \approx \max[|\mathrm{Re}X(k)|,|\mathrm{Im}X(k)|] + \frac{\min[|\mathrm{Re}X(k)|,|\mathrm{Im}X(k)|]}{4} \quad (15)$$

这样便利用简单的移位和加法运算代替复杂的乘法和求平方根运算，大大简化了硬件结构节省了资源。该近似方法平均误差为 0.6 %，在 $\left(\frac{\pi}{4},\frac{3\pi}{4},\frac{5\pi}{4},\frac{7\pi}{4}\right)$ 点误差最大，为 11.6 %。

2 FFT 处理器的软件仿真结果

利用 ALTERA 公司提供的 Quartus 开发系统，选用 APEX20K 系列中的 EP20K300E 逻辑器件进行时序仿真的结果如表 2 所示。

表 2 FFT 处理器的仿真实验结果

输入信号位数/bit	元器件选择	FFT 最大工作频率/MHz	占用 LE 单元数	所需 ROM 容量/bit	变换时间/μm
8	EP20K300 EQC240 - 1	75.32	7989	130776	257.92
	EP20K300 EQC240 - 2	65.29			
	EP20K300 EQC240 - 3	56.17			

3 结 论

实验结果表明，按照笔者的方法设计的 FFT 处理器，可以有效的抑制频谱泄漏的影响，频谱分析精度比较高，而且可以对信号进行连续实时的频谱分析，最高工作频率达到 75 MHz。

参考文献：

[1] 韩颖，王旭，吴嗣亮. FPGA 实现高速加窗复数 FFT 处理器的研究[J]. 北京理工大学学报，2003，23(3)：381 - 385.

[2] LIU JINMING，YING HUAIQIAO. The Dominant Frequency Position's Influence on FFT spectrum leakage[J]. IEEE Proceeding of ICSP，96：241 - 244.

[3] 郑星亮，程洁，魏任之. 数字信号处理中的窗效应及窗函数的应用原则[J]. 北京联合大学学报，1997，11(2)：33 - 37.

[4] 潘文，钱俞寿，周鹗. 基于加窗插值 FFT 的电力谐波测量理论(I)窗函数研究[J]. 电工技术学报，1994，9(1)：50 - 54.

[5] 程佩青. 数字信号处理[M]. 北京：清华大学出版社，2002.

[6] PAUL T，CAPOZZA. A Single - Chip Narrow Band Frequency Domain Excisor for a Global Positioning System (GPS) Receiver[J]. IEEE Journal of Solid - State Circuits. 2000，35(3)：401 - 411.

FPGA Implementation of Pipelined FFT

ZHU Bing-lian, LIU Xue-gang

(Communication Engineering College，Chongqing University，Chongqing 400030，China)

Abstract：A pipelined FFT processor designed for fast and real - time requirements with FPGA is introduced. This FFT processor can be used to real - time frequency analysis and its working frequency can reach to 75 MHz. The leakage error is reduced through multiply the sampled signal by a weighting window. In order to improve FFT's working frequency and economize FPGA resources, an algorithm of 1024 - point complex to compute 2048 - point real data is adopted. In addition, an approximate algorithm to compute module value of complex number is introduced.

Key words：pipeline；FFT；FPGA

(编辑 吕赛英)