基于 NiosII 的 SSCA 算法实现*

高玉龙，张中兆
(哈尔滨工业大学 通信技术研究所，哈尔滨 150001)

摘 要：基于 ALTERA 公司的 NoisII 软核处理器及其 Avalon 总线架构，提出采用单片 FPGA 实现 SSCA 算法的方法；对算法中适合 DSP 处理的部分采用 NiosII 软核处理器实现，其他部分采用 VHDL 语言实现;给出了算法实现的具体流程,并对并行处理关键技术提出了解决方案。以余弦信号为例进行了验证。

关键词：循环谱 SSCA 算法 NiosII 软核处理器 Avalon 总线

如果随机过程的自相关函数 $R_x(t;\tau)$是周期信号,则称 $x(t)$为二阶循环平稳过程[1],自相关函数称为循环自相关函数。由于 $R_x(t;\tau)$是周期信号,展开成傅立叶级数为：

$$R_x(t;\tau)=\sum_{m=-\infty}^{+\infty} R_x^{\alpha}(\tau)e^{j2\pi\alpha t} \tag{1}$$

其中 $\alpha=\frac{m}{T_0}$,T_0 为 $R_x(t;\tau)$的周期。它的傅立叶系数 $R_x^{\alpha}(\tau)$表示循环频率为 α 的循环自相关强度,同时还是 τ 的函数。对 $R_x^{\alpha}(\tau)$作傅立叶变换[1],得到：

$$S_x^{\alpha}(f)=\int_{-\infty}^{\infty} R_x^{\alpha}(\tau)e^{-j2\pi f\tau}d\tau \tag{2}$$

$S_x^{\alpha}(f)$为循环谱密度函数,也称循环谱。它是循环频率 α 与谱频率 f 组成的二维平面(α, f)的幅度值。

随着对信号循环平稳性研究的深入，循环谱在很多领域得到了广泛的应用。它们对计算信号循环谱的实时性要求很高，因此实现循环谱的关键就是减少其执行时间。现在,循环谱估算法有很多[1-3],成熟的大致分为三种,分别为时域平滑算法中的 FAM 和 SSCA（Strip Spectral Correlation Algorithm）、频域平滑算法中的 FSM（Frequency-Smoothed Method）。文献[3]给出了其数字实现的大致思路。

信号的循环谱一方面反映了信号统计量随时间的变化,弥补了平稳信号处理的不足,更多地反映了信号的本质特征；另一方面认为信号的统计量周期变化,简化了一般的非平稳信号处理。因此它成为近来研究的热点,正进入工程实际应用。但至今为止,还没有文献对循环谱算法进行 FPGA 实现。由于在三种算法中,SSCA 算法计算量最小,因此本文以其为例说明循环谱算法 FPGA 实现的思路和过程。

* 国家 863 资助项目（2004AA001210）

NiosII 是 Altera 公司针对其 FPGA 设计的嵌入式软核处理器[5],它只占芯片内部很少的一部分逻辑单元和存储资源,成本很低。上百兆的性能、灵活的自定义指令集和自定义硬件加速单元以及友好的图形化开发环境 Nios II IDE[6]等特点为它的应用开发提供了无与伦比的便利。而采用 Avalon 结构的总线架构能够进行多路数据的同时处理,克服了传统总线带宽瓶颈,实现最大的数据吞吐量[7]。上述这些特点正好解决了 SSCA 算法强平行性和计算量大的问题,并很好地体现在硬件加速器的运用上。

基于以上原因,本文以 NiosII 软核处理器为主,辅以 VHDL 语言编程,用单个 FPGA 实现 SSCA 算法。这种方案比传统的 DSP+FPGA 方案具有灵活、稳定、成本低、开发周期短等优点。

1 SSCA 算法描述

SSCA 算法是其时域平滑算法的一种改进，其表达式为[2]：

$$S_{xT}^{f_k+q\Delta\alpha}\left(n,\frac{f_k-q\Delta\alpha}{2}\right)\Delta t=\frac{1}{T\Delta t}\sum_r X_T(r,f_k)x^*(r)g(n-r)e^{-j2\pi qr/N} \tag{3}$$

其中：

$$X_T(n,f)=\sum_{-N'/2}^{N'/2-1} a(r)x(n-r)e^{-j2\pi f(n-r)Ts} \tag{4}$$

其中,Δt 为数据采集时间,N 为采样数,q 是循环频率分辨率的倍数。$X_T(r, f_k)$是输入信号 $x(t)$是复解调,它通过 N'点 FFT 实现,$g(n-r)$是 N 阶的平滑窗,一般为矩形窗。频率分辨率为滤波器 $a(n)$的带宽,即 $\Delta f=\Delta a=f_s/N'$,循环频率分辨率为滤波器 $g(n)$的带宽,即 $\Delta\alpha=f_S/N$,时间频率分辨率乘积为 $\Delta f\cdot\Delta t=\frac{N}{N'}$,且要满足测不准原理,即 $\Delta f\cdot\Delta t>>1$。

根据公式 (1)，SSCA 算法的数字实现框图如图 1 所示。

依据图 1,SSCA 算法的步骤如下：

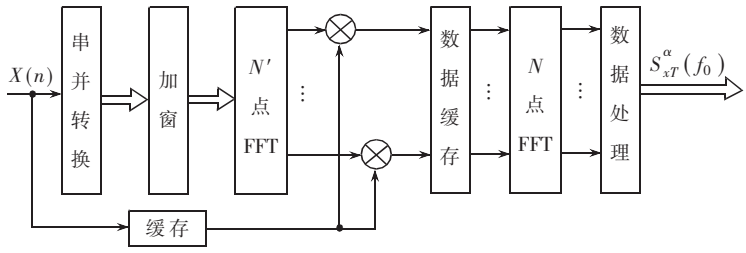

图 1 SSCA 算法数字实现

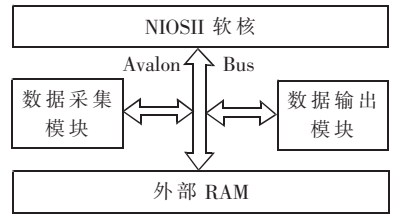

图 2 SSCA 算法 SOPC 实现框图

(1) t_0 时刻的 N'点数据加窗,然后进行 N'点 FFT;

(2) 折叠运算,求出一个周期的频谱,得到信号的复解调表达式,即:

$f_k=k(f_k/N')$, $k=-N'/2,\cdots,N'/2-1$;

(3) 求出 t_0 时刻的复解调的相关,即 $X_T(n,f_k)x^*(n)$,得到 N'个数据;

(4) 重复(1)、(2)、(3)直到时刻 t_N,表示为 $N\times N'$矩阵;

(5) 对 $N\times N'$矩阵中的每一列求 N 点 FFT,得到频率为 $(f_k-q\Delta\alpha)/2=(k/N'-q/N)\frac{f_s}{2}$,$q=-N/2,\cdots,N/2-1$,循环频率为 $f_k+q\Delta\alpha=(k/N'+q/N)f_s$ 的 N 个的数据;

(6) 数据选择。由于进行 N'点的 FFT, 只在 $f_k=k(f_k/N')$, $k=-N'/2,\cdots,N'/2-1$ 处频率有值存在,因此舍去

$$(f_k-q\Delta\alpha)/2=(k/N'-q/N)\frac{f_s}{2}\neq m/N',$$

$q=-N/2,\cdots,N/2-1$,$q=-N'/2,\cdots,N'/2-1$

点处的数据,把频率与循环频率相等的数据加起来。这样得到循环谱的估计值。

根据上面步骤, 可知 CCSA 算法的数据流具有明显的三级并行流水性,即:

(1) 通过 FFT 计算数据的复解调,即 N'点 FFT;

(2) 计算复解调和数据的共轭相关值;

(3) 计算 N 点 FFT,得到循环谱估计值。

2 SSCA 算法的 FPGA 实现

文中使用了 Altera 的 NiosII 嵌入式软核处理器,借助 QuartusII6.0 中 SOPC Builder 以及 NiosIIIDE6.0 软件,在 Altera EP2S30F484C5 芯片上进行 SSCA 算法的系统设计与实现,图 2 是该设计的系统结构图。主要的功能模块全部集中到一片 FPGA 上,实现了 SOPC 的设计思想。外部晶振提供的 50MHz 时钟经过 FPGA 内部锁相环 4 倍频得到 200MHz 的时钟信号,作为 NiosII 软核的系统时钟。

图 2 中各个部分通过 Avalon 总线连接,数据采集模块接受经过 A/D 采样的数据,并把数据按照 SSCA 算法的要求存储到外部 RAM, 数据输出模块负责选择符合算法要求的数据,并输出其值。

图 3 给出采用 SOPC Builder 生成的软核处理器。数据采集模块和数据输出模块采用 VHDL 语言生成,通过图 3 中 data_collection_interface 和 data_output_interface 接口与 NiosII 软核处理器连接。

算法的其他部分在 NiosII IDE 软件中通过 C 语言

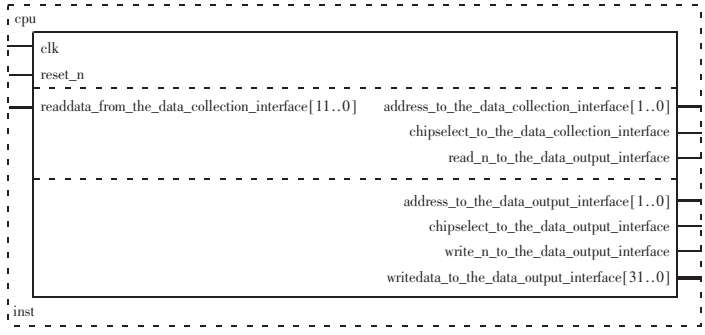

图 3 SOPC Builder 生成的软核

实现。主要包括串并转换、加窗运算、N'点 FFT 数据的顺序调整, N'点 FFT 数据存储的控制。为了提高算法的实时性,FFT 算法、相关运算采用硬件加速器,但为方便说明问题,在图 4 给出了系统主函数流程图对它们采用函数的形式。由于加窗函数、数据的顺序调整经常使用,

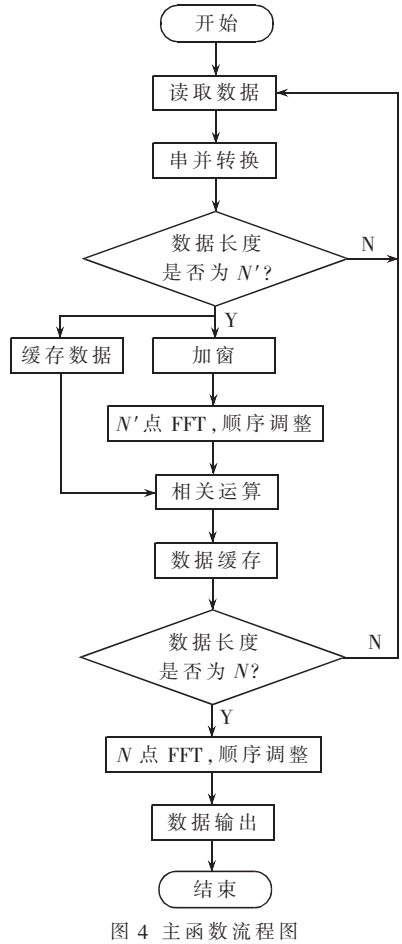

图 4 主函数流程图

因此把它们作为子函数以备调用,其他的处理在主函数中完成。

为了实现并行处理,设置两个缓冲区接收采集的数据。当第一个数据缓冲区接收到 N' 个数据时,启动第二个数据缓冲区,与此同时进入如图 3 所示的处理流程,得到 N 组数据的第一组。依次进行,直到得到 N 组数据。

为了进行实时处理,图 3 的处理流程所用的时间要比接收 N' 个数据短,这主要依靠提高软核处理器的频率来实现。在设计软核处理器中采用 200MHz 的主频就是为了完成这个任务。但同时要求数据速率不能太高,一般都能满足。

进行实时处理的另一个措施就是采用专用硬件加速器,它利用 Avalon 总线架构和主 CPU 通信,因此作为 FPGA 中的定制协处理器,能协助 CPU 同时处理多个数据块。在本文中采用 Altera 公司为 NiosII 系统开发人员提供的效能工具 Nios® II C 语言至硬件加速(C2H)编译器,缩短了开发的周期。采用硬件加速器处理 128K 字节缓冲,重点处理 SSCA 算法的三级并行性,即 N' 点 FFT、共轭相关运算、N 点 FFT。通过计算对比发现比原来采用软件编程快 528 倍,减少了算法的执行时间。

为了验证 FPGA 实现的正确性,以余弦信号 $x(t)=\cos(2\pi f_0 t)$ 为例进行实现。其循环谱的公式[1]为:

$$S_x^{\alpha}(f)=\begin{cases}\dfrac{1}{4}\delta(f-f_0)+\dfrac{1}{4}\delta(f+f_0), & \alpha=0\\ \dfrac{1}{4}\delta(f), & \alpha=\pm 2f_0\\ 0 & \text{others}\end{cases} \tag{5}$$

具体参数是 $N'=64$,$N=4096$,窗函数为汉明窗。采样频率归一化为 1,频率为 0.25Hz。图 5 给出理论上的循环谱的幅度图。图 6 是用 FPGA 实现的循环谱的幅度图。通过比较可以看出图 6 比图 5 中的 4 个主峰的幅度粗,根部有些旁瓣,且幅度值也有所减小。这是因为 FPGA 实现需要对数据进行截断,也就是数据数字处理的有限长效应引起的。在试验中数据采用 32 位处理。如果采用 16 位其效果会差一些。

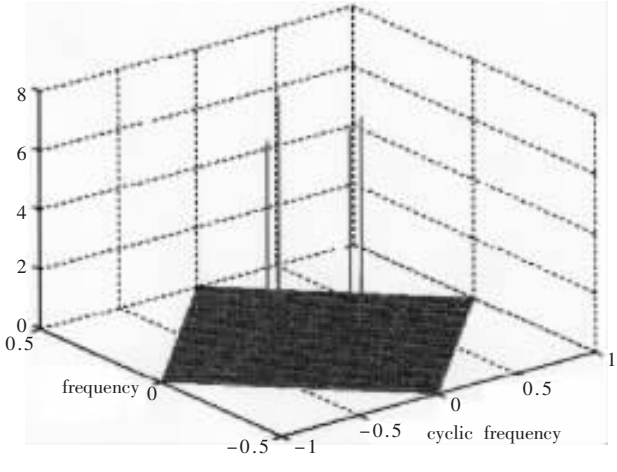

图 5 余弦信号的理论循环谱幅度

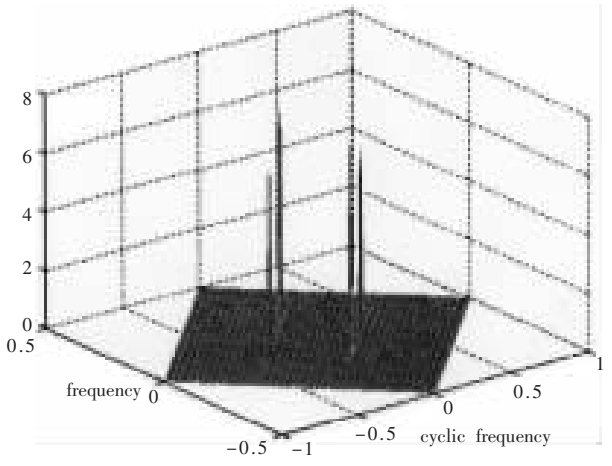

图 6 FPGA 实现的循环谱幅度

通过上述三项措施,大大提高了循环谱算法的实时性。本文提出的实现循环谱方法已经在某调制识别项目中得到应用,达到了项目对实时性的要求。

对 SSCA 算法实现过程稍加改动,就可以实现循环谱的其他算法。

参考文献

[1] 张贤达,保铮.非平稳信号分析与处理,北京:国防工业出版社,1998.

[2] GARDNER W A. Measurement of spectral correlation[J]. IEEE Trans. Acoust., Speech, Signal Processing, 1986,V(34):1111-1123.

[3] BROWN W A, LOOMIS H H. Digital implementations of spectral correlation analyzers[J]. IEEE Transactions on Signal Processing, 1993,V41(2):703-720.

[4] Altera Company.SOPC Builder Data Sheet1[S].2005.

[5] Altera Company.Nios II Processor Reference Handbook[S]. 2005.

[6] Altera Company. Nios II Software Developer's Handbook[S]. 2005.

[7] Altera Company.Avalon Interface Specification[S].2005.

(收稿日期:2006-09-28)