基于 FPGA 的新型浮点 FFT 处理器设计

范 展，梁国龙，刘 洋
（哈尔滨工程大学 水声工程学院，黑龙江 哈尔滨 150001）

摘 要：针对现有 FFT 算法结构复杂、难以并行扩展的问题，提出了一种改进的 FFT 算法，在此基础上设计了一种基于浮点运算的 FFT 处理器，并进行了仿真验证。结果表明，新算法大大简化了系统结构，减少了系统的硬件开销，非常容易并行实现，且显著提高了运算效率，完成一次 N 点的 FFT 运算只需要 $N/2$ 个时钟，完全满足实时信号处理的要求。

关键词：改进的 FFT 算法；浮点运算；实时信号处理

Design of a new floating-point FFT processor based on FPGA

FAN Zhan, LIANG Guo Long, LIU Yang
(College of Underwater Acoustical Engineering, Harbin Engineering University, Harbin 150001, China)

Abstract: Aiming at the problem in the existing FFT algorithm structure, which is not only complex but difficult to realize parallel expansion, a way of improved FFT algorithm is proposed, and a new floating-point FFT processor is designed in this paper. Simulation based on the Modelsim has carried on. The results indicated that the new algorithm is able to simplify the system structure greatly, reduce hardware consumption, and is easy to carry out parallel realization. Completing a N-points FFT only needs N/2 clock periods, it can satisfy the requests of real-time signal processing.

Key words: Improved FFT algorithm; Floating-point operate; Real-time signal processing

采用 FPGA 实现 FFT 是目前研究的热点。新一代 FPGA 内部嵌入了大量高速数字信号处理模块和 RAM 模块，容易借助全并行流水线技术实现 FFT。目前已经有不少学者进行了这方面的研究，参考文献[1]-[4]均以 FPGA 为硬件平台实现了 FFT 算法，系统的运算速度相比于 DSP 有了很大提高，基本能够满足某些高速实时信号处理的要求。但是这些算法的实现过程较为复杂，这主要是因为各流水线级的数据流向不一致，所以需要为每级单独设计地址产生程序，从而使系统的控制时序变得复杂，不利于算法的扩展。

本文在详细分析现有基 2 FFT 算法的基础上，针对其所存在的一些问题，对算法流程进行了适当改进，使之更加适合流水线方式下的并行处理。在此基础上设计了一种切实可行的 32 位浮点 FFT 处理器。该处理器的特点是结构简单，非常容易设计实现，易于扩展，且运算速度进一步提高，系统稳定工作后，完成一次 N 点的 FFT 变换只需要 $N/2$ 个时钟，可实时进行 50%重复的 FFT 运算。

1 改进的并行 FFT 算法

蝶形运算是基 2 FFT 算法的基本运算单位，为了进一步提高 FFT 处理器的性能，本文从两个方面对基本蝶形运算进行了改进：

(1)对流水线级做了更细的划分，将基本蝶形运算单元分成两级进行，分别完成复数乘法运算和复数加法运算。

(2)各加法运算级的运算结果不再顺序存储，而是将偶地址和奇地址单元对应的运算结果分别顺序存入存储空间的上下两个半区。

以 N（$N=2^M$，M 为整数）点的 FFT 运算为例，采用 $2M-1$ 级流水线实现。用 y_j 表示第 j 级流水线的运算结果。当 $j=2r-1$，$r\in[1,M]$时，该流水线级只进行复数加法运算，其运算过程可表示为：

$$y_j(i)=\begin{cases}y_{j-1}(2i)+y_{j-1}(2i+1), i\in[0,N/2)\\ y_{j-1}(2i-N)-y_{j-1}(2i+1-N), i\in[N/2,N)\end{cases} \tag{1}$$

从式(1)可以看出，每个加法运算级需要完成 N 次复数加法运算。而当 $j=2r$，$r\in[1,M)$时，该流水线级只进行复数乘法运算，其运算过程可表示为：

$$y_j(2i)=y_{j-1}(2i), i\in[0,N/2) \tag{2}$$

$$y_j(2i+1)=\begin{cases}y_{j-1}(2i+1), i\in[0,S_j)\\ y_{j-1}(2i+1)W_N^{d_i}, i\in[S_j,N/2)\end{cases} \tag{3}$$

式(3)中，$S_j=N\times2^{-j/2}/2$，$d_i=S_j\times ceil[(i+1)/S_j-1]$，$ceil[\bullet]$是对括号中的内容取整。可以看出，第 j 级流水线需要完成 $N(1-2^{-j/2})/2$ 次复数乘法运算。

针对 8 点的 FFT 运算，图 1 给出了改进后的算法数据流。将整个数据处理流程分成五级流水线，第一、三、五级只进行复数加法运算，第二、四级进行复数乘法运算。图中虚线箭头只表示数据的存储方向，不进行任何算术运算。

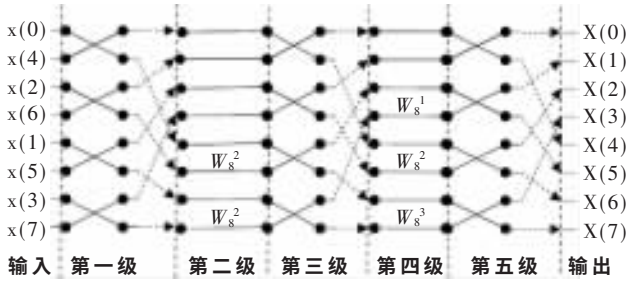


图 1　改进后的 FFT 算法数据流

改进后，由于基本蝶形运算的复数乘法运算和复数加法运算分成两级进行，所以各流水线级的运算量与改进前相比变小了，这使各级由运算产生的硬件延时变小，所以系统可以在更高的时钟频率下正常工作。另一方面，所有加法运算级的数据流向已经变得完全一致，因此，不再需要为每级单独设计控制程序，而且系统的稳定性也得到了提高。同时，所有乘法运算只在该级的奇地址单元进行，而且是到最后一个单元结束，这些特点增强了乘法运算级的继承性。

2 FFT 处理器设计

FFT 处理器的结构如图 2 所示，主要包括：输入和输出处理模块、加法运算单元、乘法运算单元、旋转因子存储器 ROM、数据存储器 RAM 和控制器。图中，输入信号 Din 和输出信号 Dout 都是 32 位浮点数，采用 IEEE 754 标准。EN 是系统使能信号，CLK 是系统时钟信号，ISync 为输入帧同步信号，OSync 为输出帧同步信号。下面将详细介绍各部分的功能。

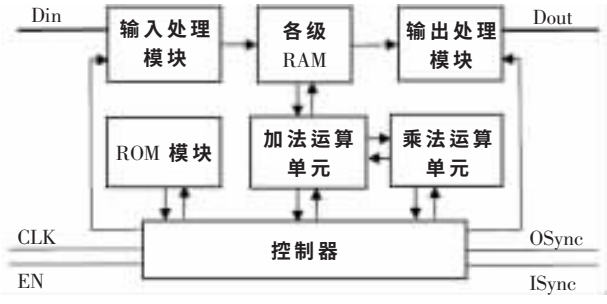


图 2　FFT 处理器结构图

2.1 输入和输出处理模块设计

本文设计的 FFT 处理器采用时域抽取法实现，处理器的输入数据在时域上按一定的倒序规则打乱，经变换后，输出的 FFT 频域信号是顺序排列的。输入处理模块从总线 Din 读取数据(输入的 32 位复数据的实部和虚部)，ISync 是输入帧同步信号。对于 50%重复的 FFT 运算，每一帧输入数据都包含 50%的历史数据。在流水线方式下，为了保证数据的连续操作，在输入模块中建立两组大小相同的 RAM 同时存储输入数据。当一帧数据变换完成后，控制器产生一个输出帧同步信号 OSync，变换结果会通过输出数据总线 Dout 顺序输出。

2.2 加法运算级设计

加法运算级包括加法运算单元和 RAM 模块两个部分，加法运算单元执行浮点加法运算，RAM 模块则保存运算产生的中间结果。

2.2.1 加法运算单元设计

进行浮点加法运算时，由于两个操作数的指数往往不相等，所以要将尾数对齐后再运算。浮点加法运算的具体过程归纳如下[5]：

(1)阶码相减：比较阶码大小，将两个数的阶码做差得到 $\Delta E=|E_A-E_B|$；

(2)交换操作数的位置：根据第(1)步中的比较结果，交换两个浮点数的位置，使得大数总是以被加数的形式出现；

(3)有效数移位对阶：根据 ΔE 的结果，把较小的浮点数有效数右移，前端补零，使得两个操作数具有相同的指数；

(4)有效数相加：根据移位后的有效数和操作数的符号完成有效数的加减运算；

(5)规格化：根据运算结果对浮点有效数字做规格化处理。

2.2.2 RAM 模块设计

为了与加法运算单元的速度相匹配，RAM 模块必须在每个时钟周期完成两次读数和两次写数操作。而 FPGA 内部的 RAM 一个时钟周期最多只能进行一次读数和一次写数操作，这便成了系统的速度瓶颈。通过观察发现，每个时钟周期，加法运算单元产生的两个运算结果分别存入了 RAM 的上下两个半区，而下一个流水线级读取的两个数据分别从该 RAM 模块的奇地址和偶地址单元顺序取出。基于这些特点，设计了一种能在一个时钟完成两次读数和两次写数操作的 RAM，它的具体实现过程是：将整个 RAM 分成容量相等的四个部分，如图 3 所示(图中，除了作为控制信号的地址线外，其他地址总线均被省略)。存储数据时，第一个数存入 RAM1 或 RAM2 中，第二个数存入 RAM3 或 RAM4 中，存数地址的最低位(waddr(LSB))作为所有 RAM 的控制信号；读取数据时，第一个数从 RAM1 或 RAM3 读出，第二个数从 RAM2 或 RAM4 读出，读数地址的最高位(raddr(MSB))作为所有 RAM 的控制信号。

加法运算单元产生的运算结果并不是按原址存储的，如果采用单组 RAM，会造成数据还没有被完全读取

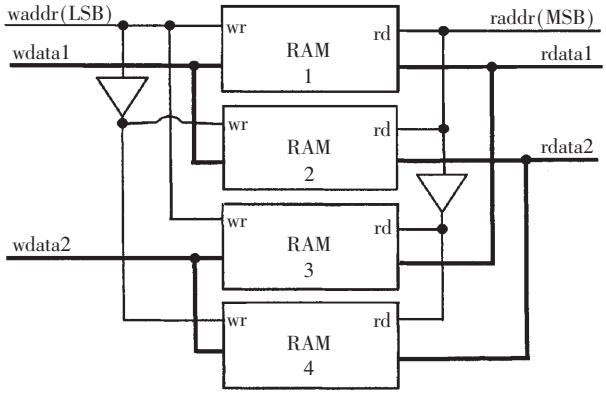


图 3 RAM 模块设计

就被新数据覆盖的冲突。为了解决这个冲突，采用乒乓结构的 RAM 模块进行数据存取，即建立 2 组大小相同的 RAM，当数据往 A 组 RAM 中存储时，下一级运算单元从 B 组 RAM 中读取上次存储的结果。通过模块选择信号实现 A、B 两个 RAM 模块之间的切换。

2.3 乘法运算级设计

乘法运算单元是 FFT 处理器的关键路径。Stratix 系列 FPGA 内部嵌入了大量乘法器资源，这给设计带来了便利。浮点乘法运算主要有如下几个步骤：

(1)阶码相加：将两个乘数的阶码部分相加，这里可能会出现溢出的情况，如出现上溢，就将结果置为浮点格式所能够表示的最大的数；如出现下溢，则将该浮点数置 0。

(2)尾数相乘：将两个乘数的尾数部分添 1 后相乘，并根据乘法结果的最高位调整阶码。

(3)规格化尾数：将尾数规格化处理，并确定浮点数的符号和阶码。

2.4 ROM 模块设计

ROM 模块用来存储乘法运算级所需的旋转因子。Quartus II 软件中的 MegaWizard 工具可以将 *.mif 格式文件中的数据固化到 ROM 模块中。具体的实现方法是：先通过 MATLAB 计算出所有旋转因子的实部和虚部，并按 32 位浮点数的格式存储到 *.mif 文件中，在 Quartus II 软件环境下，利用 MegaWizard 工具建立一个 ROM 元件，然后选择指定的 *.mif 文件作为该 ROM 的配置文件，这样便完成了对 ROM 的初始化。当初始化完成后，只需要将生成的 ROM 模块包含到工程中即可。

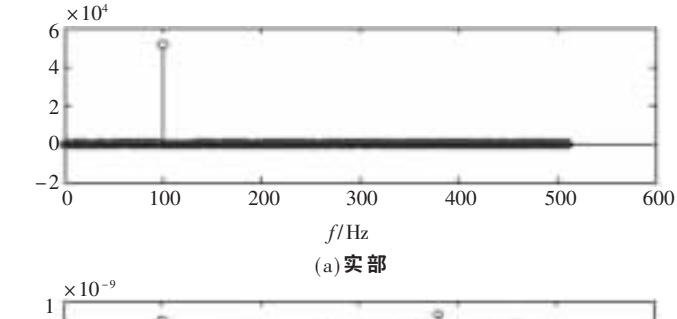
2.5 控制器设计

控制器是整个 FFT 处理器的核心，它主要完成两部分的功能：(1)提供输入和输出处理模块的使能信号。在 EN 为高电平的情况下，当检测到外部输入信号 ISync 的上升沿时，立即启动输入处理模块，从输入数据总线 Din 上读取数据。一帧数据变换完成后，控制器会产生输出帧同步信号 OSync，并控制输出处理模块同步输出运算结果。(2)提供各乘法运算级所需的旋转因子。所有旋转因子都按相应的数据格式和顺序存储在 ROM 模块中。运算过程中，需要为各乘法运算级分别计算旋转因子的存储地址，这些工作都是由控制器完成的。

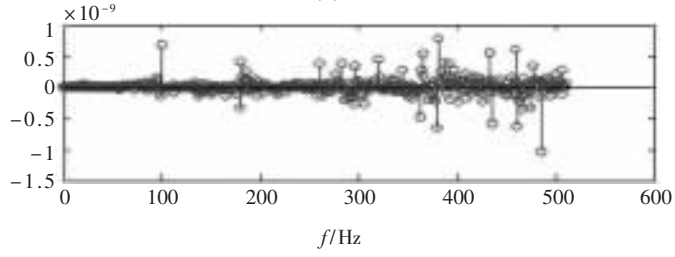
3 仿真验证

为了证明 FFT 处理器的正确性，对 1 024 点的输入数据进行测试试验。在 MATLAB 中有 Link for Modelsim 工具，它提供了 MATLAB 与 Modelsim 之间的数据接口。通过 Link for Modelsim 工具，可以实现 MATLAB 与 Modelsim 之间的数据传输，从而实现 MATLAB 与 Modelsim 之间对测试系统的联合仿真。

输入信号为 $x=100\cos(200\pi k/1024), k=0,1,2,\cdots,1023$。输入信号通过 Modelsim 仿真后将得到的结果返回 MATLAB 中。图 4 是仿真运算结果的实部值和虚部值。图 5 是仿真结果的实部和虚部与由 MATLAB 中的 fft 函数计算结果之间的误差。从中可以看出，本 FFT 处理器的运算误差已经控制在 10^{-9} 量级。

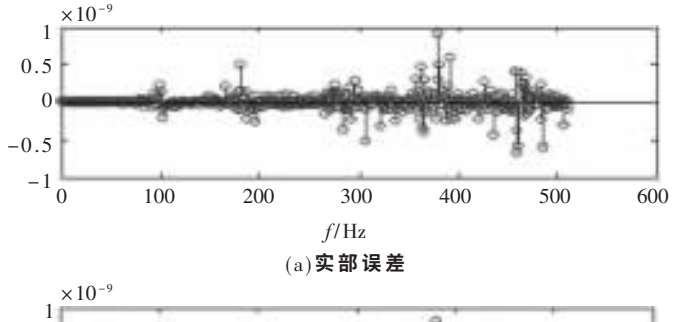


(a)实部

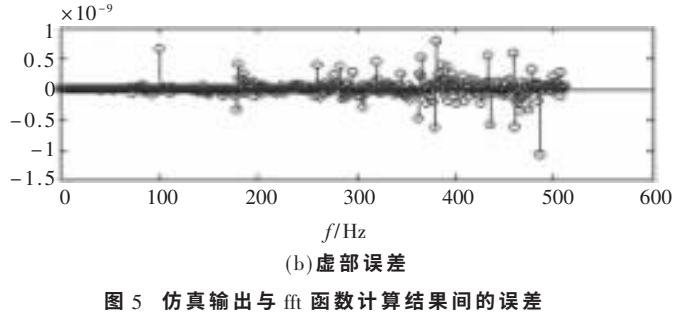


(b)虚部

图 4 仿真输出的实部和虚部



(a)实部误差



(b)虚部误差

图 5 仿真输出与 fft 函数计算结果间的误差

本文对基 2 FFT 算法进行了适当改进，在此基础上设计了一种基于 32 位浮点运算的 FFT 处理器。仿真结果表明，改进后的 FFT 处理器的运算效率进一步提高，完成一次 *N* 点的 FFT 运算只需要 *N*/2 个时钟，可实时进行 50%重复的 FFT 运算，并且开发工作变得更加简单，非常容易设计实现。该算法的另一个优点是可扩展性强，虽然在设计过程中对流水线级做了更细的划分，但是各运算级有较好的继承性，所以只需要分别开发出一级加法运算模块和乘法运算模块即可，开发工作量并不会随运算点数的增加而增大。同时，由于浮点运算固有的高精度性，使得浮点 FFT 处理器具有较高的实际应用价值，可以广泛应用于对精度要求较高的数字信号处理领域。

参考文献

[1] 李伟，孙进平，王俊，等.一种基于 FPGA 的超高速 32K 点 FFT 处理器[J].北京航空航天大学学报，2007，33（12）：1440-1443.

[2] LIN Y W，LIU H Y.A 1-Gs/s FFT/IFFT processor for UWB applications[J].IEEE Journal of Solid-state Circuits，2005，40(8)：1726-1735.

[3] 王旭东，刘渝.全并行结构 FFT 的 FPGA 实现[J].南京航空航天大学学报，2006，38(2)：96-100.

[4] 吴松炎，管云峰.非基 2 点 FFT 处理器的设计与实现[J].电路与应用，2007，31(1)：24-26.

[5] 金席，高小鹏，龙翔.浮点乘累加处理单元的 FPGA 实现[J].计算机与数字工程，2006，34(10)：165-168.