

论文原创性声明

本人郑重声明：此处所提交的论文《基于FPGA的新型浮点FFT处理器设计》是本人在导师指导下进行研究工作所取得的研究成果。据我所知，除了文中特别加以标注的地方外，论文中不包含其他人已经发表或撰写过的研究成果。本声明的法律结果将完全由本人承担。

作者签字：范展

2008年3月9日

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 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

中图分类号: TN409

文献标识码: A

文章编号:

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 difficultly 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 an N points of FFT only needs N/2 clock periods, it may satisfy the requests of real-time signal processing.

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

1 引言

FFT作为数字信号处理的关键技术之一, 在信号处理领域扮演着非常重要的角色^[1]。目前用于实现FFT的硬件平台主要有DSP、ASIC和FPGA。用DSP实现FFT的处理速度相对较慢, 往往难以满足高速实时信号处理的要求; ASIC虽然速度很快, 但外围电路相对复杂, 不易扩展, 且价格昂贵; 采用FPGA实现FFT兼有二者的优点。ALTERA公司推出的新一代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, 不但性能稳定、经济性好, 而且可以大大缩短计算耗时。目前已经有不少学者进行了这方面的研究, 文献[2]~[9]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, 这使系统的控制时序变得复杂, 开发难度增大, 不利于算法的扩展, 也限制了它的推广和应用。

本文在详细分析现有基 2 FFT 算法的基础上, 根据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。RAM 模块的设计是本 FFT 处理器的一大关键, 因为它的存取速度直接关系到系统的性能。Stratix 系列 FPGA 内部嵌入了大量 RAM 模块, 为设计带来了便利, 但这些 RAM 一个时钟最多只能进行一次读数和一次写数操作, 本文结合改进后的算法对 RAM 模块进行适当改进, 在不需要额外增加 RAM 资源的情况下, 改进后的 RAM 可以在一个时钟完成两次读数和两次写数操作。为了验证 FFT 处理器的性能, 采用 Modelsim

软件结合 Matlab 软件做了详细的仿真分析, 仿真结果表明, 该 FFT 处理器不仅运算速度快, 而且精度很高, 可以广泛应用于对精度要求较高的实时数字信号处理领域。

2 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) 对流水线级作了更细的划分, 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:

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

从上式可以看出, 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:

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

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

式中, $S_j = N \cdot 2^{-j/2} / 2$, $d_i = S_j \cdot \text{ceil}[(i+1)/S_j - 1]$, $\text{ceil}[\bullet]$ 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 5 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。

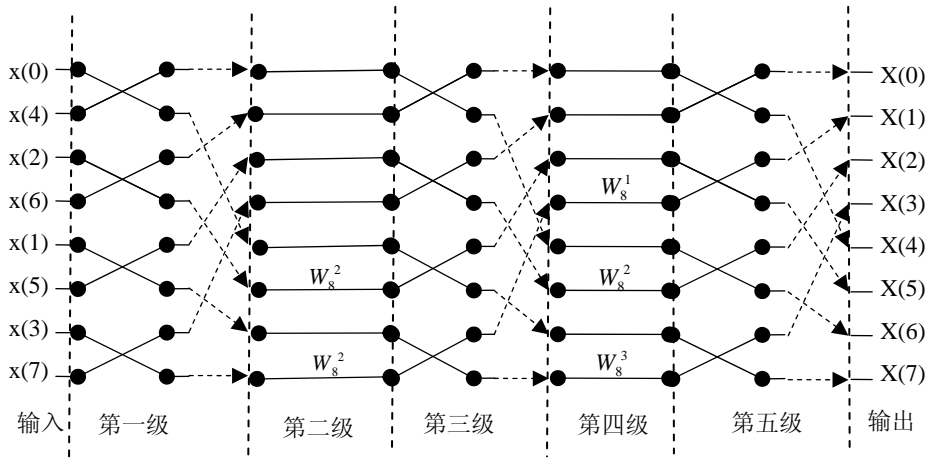


图 1 改进后的 FFT 算法数据流

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已经变得完全一致, 因此, 所有加法运算级可以复用同一个运算模块, 而不再需要为每级单独设计控制程序, 这大大简化了 VHDL 的开发工作, 而且系统的稳定性也得到了提高。同时, 各乘法运算级的运算规律也很明显, 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, 使系统总体的控制时序得到简化, 开发工作变得简单。

3 FFT 处理器设计

在 FFT 处理器工作方式的设计上, 充分利用 Stratix 系列 FPGA 内嵌乘法器和存储器资源丰富的特点, 采用全并行流水线方式实现。FFT 处理器的结构如图 2 所示, 从图中可以看出该处理器包括: 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, 当 EN 为 ‘1’ 时, 系统开始正常工作。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。以下将详细介绍各部分的功能。

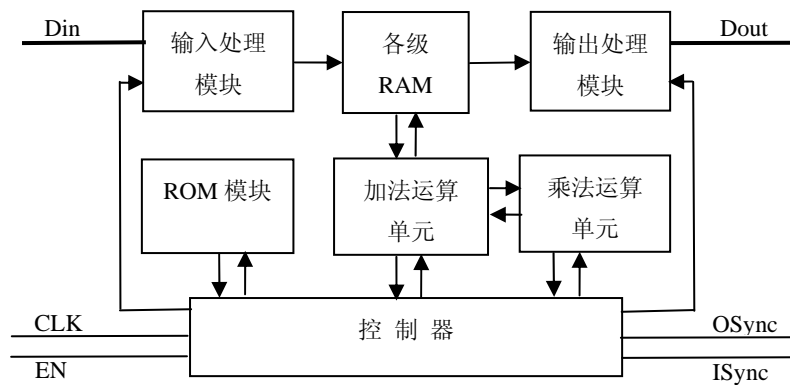


图 2 FFT 处理器结构图

3.1 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), 并在 CLK 的上升沿对数据进行倒序存储。ISync 是输入帧同步信号, ISync 由 ‘0’ 变为 ‘1’ 时表示输入数据帧的第一个数开始输入。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。两组 RAM 的写操作控制时序完全一样, 但是写数地址不同, 对于 N 点的 FFT 运算, 二者正好相差 N/2。这样, 每次输入 N/2 个数据后就会有一组 RAM 存满, 从下一个时钟开始, 这组已经存满的 RAM 将提供给下一级运算单元进行处理。

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。本文设计的 FFT 处理器每个时钟会产生两个运算结果, 这两个运算结果的地址分别指向上下两个半区, 见图 1。对于离散付氏变换, 输出的频域信号总是以中点为轴对称分布, 所以前 N/2 个点已经包含了所有频域信息。基于以上原因, 在输出处理模块, 每个时钟周期只需要输出第一个数即可, 所以一帧数据经过 FFT 变换后只会输出 N/2 个数据。

3.2 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。浮点加法器的设计是难点,也是关键,它除了要保证运算速度,使得每个时钟完成一次浮点加法运算,还需要保证运算精度;另外,如何保证运算速度与数据存取速度相互匹配也是设计的关键。由前面的分析可以看出,本 FFT 处理器的运算速度是数据存取速度的两倍,这就要求设计的 RAM 模块每个时钟需要完成两次读数和两次写数操作。

3.2.1 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。为了实现尾数对齐,可以右移较小的数也可以左移较大的数。这两种操作都会导致数字丢失,一般来说,右移较小的数而丢失的数字所造成的影响要小一些。浮点加法运算的具体过程归纳如下^[10]:

- 1) 阶码相减:比较阶码大小,将两个数的阶码作差得到 $\Delta E = |E_A - E_B|$;
- 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。这样做的目的在于减少硬件的需求量,在下一步中仅需要一个移位逻辑就可以完成运算;
- 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) 转换:当有效位的结果为负数时,转换为符号、有效数的表示方式。因为有效数采用原码表示,所以转换需要一个求补操作,即包括一个加法步骤;
- 6) 前导零检测:对于减法,判断由于减法结果产生的左移数量;对于加法,可能需要右移一位或不移。对前导零判定的结果进行编码以确定规格化移位;
- 7) 规格化:根据前导零的检测结果对浮点有效数字计算结果作规格化处理,有效数字左移、指数相应调整。

对于 N 点的 FFT 运算,加法运算级一共需要进行 N 次复数加法运算。每个时钟周期,加法运算单元要从上一个流水线级读取两个数据,并对这两个数分别进行复数加法运算,然后将运算产生的两个结果同时存入 RAM 模块中。通常,完成一次复数加法运算需要进行两次实数加法,所以需要为每个加法运算级分配 4 个浮点加法器,最终可以保证 N/2 个时钟完成一帧数据的处理。

3.2.2 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。图中，三角形符号表示对该信号取非，通过这种简单的逻辑控制即可实现各个 RAM 模块之间的数据切换。

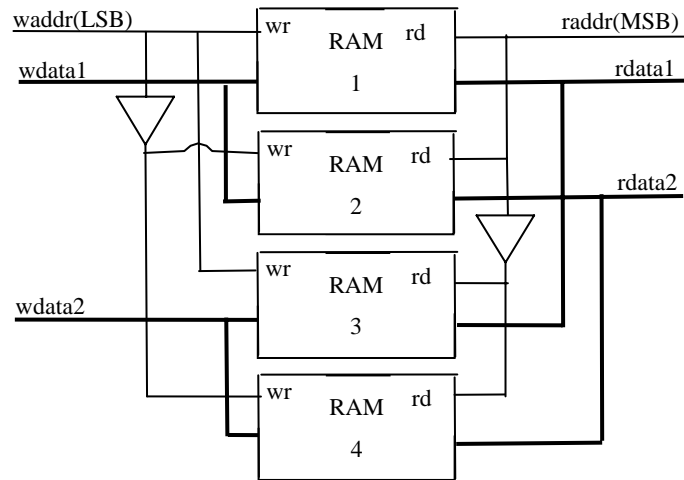


图 3 RAM 模块设计

根据改进后 FFT 算法的流程可以看出，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，如图 4 所示。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。通过组选择信号实现 A、B 两个 RAM 模块之间的切换。组选择信号是一个方波信号，N/2 个时钟进行一次切换。

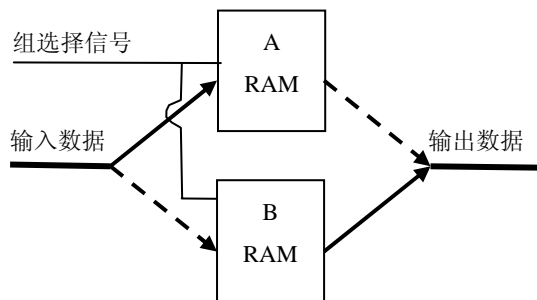


图 4 乒乓结构的 RAM 模块

在上述基本定义的基础上，在 Quartus II 软件环境下对 RAM 模块进行了设计综合与时序仿真，图 5 是仿真时序图。图中 chipselect 信号是组选择信号，raddr 是读数地址，waddr 是写数地址，rdata1_Re、rdata1_Im 和 rdata2_Re、rdata2_Im 分别是 RAM 中读出的第一个数和第二个数的实部与虚部，wdata1_Re、wdata1_Im 和 wdata2_Re、wdata2_Im 是写入 RAM 的第一个数和第二个数的实部与虚部，从图中时序可以看出，该 RAM 模块的设计完全满足实际需求。

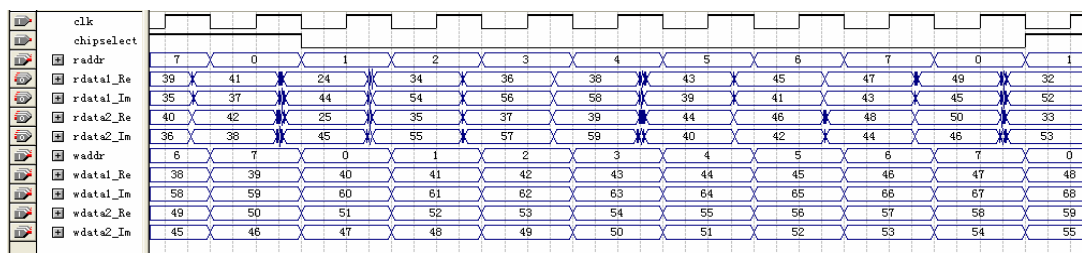


图 5 RAM 模块仿真时序图

3.3 乘法运算级设计

乘法运算单元是 FFT 处理器的关键路径，其运算速度直接关系到系统的整体性能。与浮点加法器相比，浮点乘法器的结构更为简单，而且 Stratix 系列 FPGA 内部嵌入了大量乘法器资源，每个乘法器都可以在一个时钟周期内完成一次实数乘法运算，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，根据乘法结果的最高位调整阶码。由于乘法结果最多只需一次左移就可以规格化，因而如果最高位为1，则不需调整阶码，如果为0，阶码只需减 1 即可；
- 3) 规格化尾数，完成浮点格式输出：将尾数规格化处理，并确定浮点的符号和阶码，将最终乘积的符号位、指数位和尾数位表示成32位浮点数的格式输出。

由改进后的 FFT 算法流程可以看出，每个乘法运算单元最多只需完成 $N/2$ 次复数乘法运算，且每个时钟周期最多进行一次。通常，完成一次复数乘法运算需要进行 4 次实数乘法和 2 次实数加法运算，所以，需要为乘法运算单元分配 4 个浮点乘法器和 2 个浮点加法器，这样便可以保证乘法运算级在 $N/2$ 个时钟完成一帧数据的处理。

如何正确存取运算结果同样也是乘法运算单元非常关键的问题。从图 1 给出的算法数据流中可以看出，每个乘法运算级都是从上一个流水线级顺序读取数据，而运算结果是按原址产生的。利用这一特点，可以将运算结果直接送给下一个流水线级，而不需要保存到 RAM 模块中，这便大大简化了系统结构，而且节省了 RAM 资源。这种方案的具体实现方法是：乘法运算单元提前一个时钟从上一级的 RAM 中读取两个数据，并与控制单元提供的旋转因子进行浮点乘法运算，然后将运算结果直接送给下一级。这种设计可以保证对数据的连续操作，因为相邻两级之间的时序是完全匹配的。

3.4 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 Plug-In Manager 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 Plug-In Manager 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。

3.5 控制器设计

控制器是整个 FFT 处理器的核心，作为各个功能单元的总线接口，它主要完成以下工作：

- 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，并对读取的数据进行倒序存储。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。中间各运算级的使能信号由上一个流水线级产生。
- 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。根据算法的数据流图可以看出，运算过程中，同一时刻各乘法运算级所需要的旋转因子并不完全相同，因此，针对不同乘法运算级需要分别计算其所需要的旋转因子在 ROM 中的存储地址，这些工作都是由控制器完成的。控制器从 ROM 模块中

读取数据后，将所读的数据传给乘法运算单元。

4 仿真实验

为了证明 FFT 处理器的正确性，对 1024 点的输入数据进行测试试验。在 MATLAB7.01 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。由于 1024 点 FFT 的数据量很大，为了方便比较，通过 Link for Modelsim 工具将 Modelsim 仿真后的数据传入 MATLAB 中，然后与 MATLAB 提供的 fft 函数运算的结果进行比较，来对系统进行检验。

输入信号为 $x = 100 \cdot \cos(100 \cdot 2 \cdot \pi \cdot k / 1024)$ $k=0,1,2,3,\dots,1023$ ，输入信号通过 Modelsim 仿真后将得到的结果返回 MATLAB 中。图 6 和图 7 分别是仿真运算结果的实部值和虚部值。图 8 和图 9 分别是仿真结果的实部和虚部与由 MATLAB 中 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了 10^{-9} 量级。

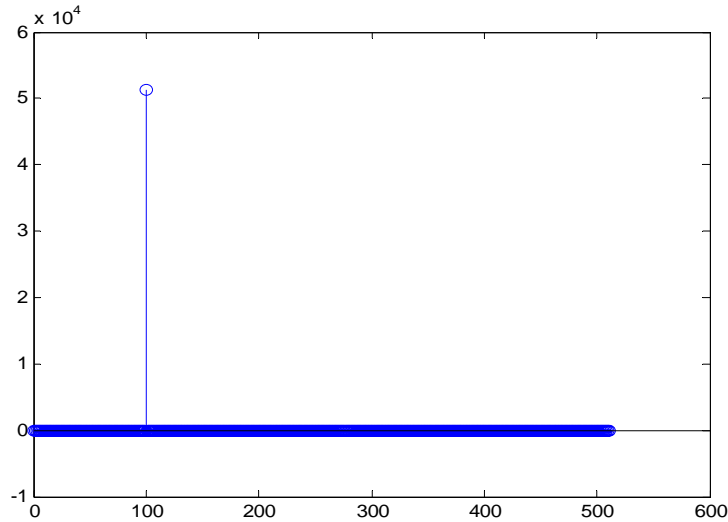


图 6 仿真输出的实部

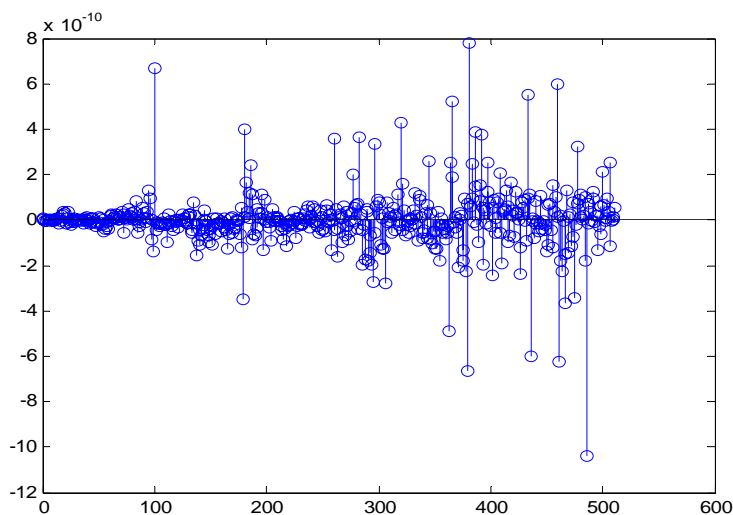


图 7 仿真输出的虚部

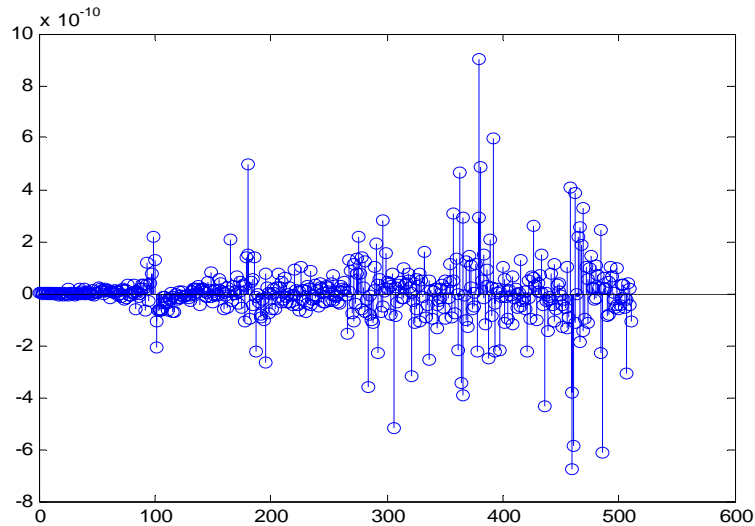


图 8 仿真输出实部与真实值间的误差

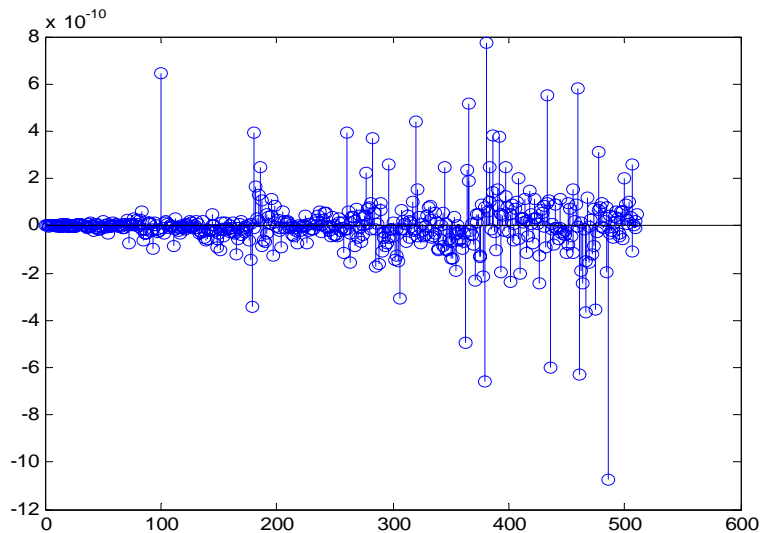


图 9 仿真输出虚部与真实值间的误差

5 结论

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作了更细的划分,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] Oppenheim A V, Schaffer R W. Discrete-time signal processing[M]. Englewood Cliffs,NJ: Prentice Hall, 1998.
- [2] 李伟, 孙进平, 王俊, 李少洪. 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.

- [3] Lin YW, 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.
- [4] 谭征,张晓林. 一种基于 FPGA 的超高速 FFT 处理器设计[J]. 遥测遥控, 2005, 26 (6) : 46 – 49.
- [5] 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.
- [6] 满峰,汶德胜,朱家佳. 基于 FPGA 的高速处理器的设计与实现[J]. 科学技术与工程,2006,9(6):2657–2660.
- [7] 吴松炎,管云峰等. 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.
- [8] 朱晓航,刘亚男. OFDM 中的 FFT 模块设计及其 FPGA 实现[J]. 电子科技,2007,2(15):33 – 36.
- [9] Benzl A O, Grigisl P C. A broadband FFT spectrometer for radio and millimeter astronomy[J]. Astronomy & Astrophysics,2005, 442 (2) : 767 – 773.
- [10] 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.

作者简介:

范展: 男, 1983 年生, 硕士研究生, 主要研究方向为信号与信息处理、通信。

梁国龙: 男, 1964 年生, 教授, 博士生导师, 主要研究方向为信号与信息处理、水下定位与导航、水声通信、鱼雷自导。

刘洋: 男, 1980 年生, 助教, 主要研究方向为信号与信息处理、水声通信。

Email: ff_131486@sina.com.cn

电话: 13836110237

通信地址: 哈尔滨工程大学水声工程学院 31# 461 室 邮编: 150001