基于 FPGA 的高速浮点 FFT 的实现研究

刘 健,史彩娟,赵丽莉
(河北联合大学 信息学院,河北 唐山 063009)

摘 要:研究了利用 FPGA 实现浮点 FFT 的技术,提出了一种循环控制、RAM 访问和蝶形运算三大模块以流水线方式协同工作的方案,结合数据缓冲和并行处理技术,讨论了蝶形运算单元的工作机制。浮点乘法器采用并行 Booth 编码和 3 级 Wallace 压缩树的结构,浮点加法器中采用独立的定点加法器和减法器,使运算得以高速进行。RAM 读/写时序和运算参数都可利用寄存器设置。本设计已在 Cyclone-II系列芯片 EP2C8Q208 中实现,200 MHz 主频下,采用外部 RAM,完成 1 024 点复数 FFT 只需 750 μs。

关键词:FPGA;浮点 FFT;蝶形运算;Booth 编码;Wallace 压缩树
中图分类号:TN911 **文献标识码**:B **文章编号**:1674-7720(2012)14-0079-03

Research of high speed floating point FFT based on FPGA

Liu Jian, Shi Caijuan, Zhao Lili
(Information Institute, Hebei United University, Tangshan 063009, China)

Abstract: The paper discusses technology of implementing floating point FFT on FPGA. Scheme is given that loop control, RAM access and butterfly operation work together in the way of assembly line. Combined with data buffer and parallel processing technology, working mechanism of butterfly operation is discussed. Floating point multiplier uses structure of parallel Booth encoding and three steps Wallace compression tree, independent fixed point adder and subtractor are used in designing floating point adder, so that operation can be completed in high speed. RAM R/W timing and operation parameters can be setted by register. Design has been implemented by Cyclone-II series chip EP2C8Q208. At 200 MHz, using of external RAM, time of completing 1 024-point complex FFT is only 750 μs.

Key words: FPGA; floating point FFT; butterfly operation; Booth encode; Wallace compression tree

数字信号处理是一门涉及许多学科而又极为广泛应用于语音/图像处理、仪器仪表、自动控制、通信等许多领域的学科。DFT(离散傅里叶变换)为其核心运算,DFT 的快速算法即 FFT(快速傅里叶变换)的高效实现意义重大。正是此算法的出现才使得 DSP 技术得以应用并迅速发展,其运算速度是 DSP 芯片的核心性能之一。高速度和高精度 FFT 的硬件实现方案已成为 DSP 领域的一大研究热点。硬件实现 FFT 算法的方案主要存在两个方向:一是专用 DSP 芯片,存在大量的冗余运算,需要许多跳转操作,运算速度较慢,且硬件接口单一;二是通过 FPGA 实现,FPGA 硬件可编程,能够并行处理,容易设计流水线结构,故非常适合实现 FFT 算法。

利用 FPGA 实现 FFT 的最大难点在于精度,目前可行的方案多为定点运算或块浮点运算,精度不高[1]。本文给出一种通过 FPGA 实现单精度浮点 FFT 的方法,在实现高精度的同时,通过流水线设计和并行处理技术使运算得以高速执行,并且接口方式可自由配置,实用性强且应用前景广阔。

1 按时间抽选的基 2-FFT 算法

设输入序列长度为 $N=2^M$(M 为正整数),将该序列按时间顺序的奇偶分解为越来越短的子序列,称为基 2 按时间抽取的 FFT 算法,也称为 Coolkey-Tukey 算法。其中基 2 表示 $N=2^M$,M 为整数。若不满足这个条件,可以人为地加上若干零值(加零补长),使其达到 $N=2^M$。算法可描述为[2]:

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

$$X(k)=X_1(k)+W_N^k X_2(k) \quad k=0,1,\cdots,N/2-1 \qquad (2)$$

$$X(k+N/2)=X_1(k)-W_N^k X_2(k) \quad k=0,1,\cdots,N/2-1 \qquad (3)$$

依此可逐层分解直到子序列长度为 2，即先对输入数据 $x(n)$ 作倒序处理。求得每组子序列的 DFT（2 点），并以蝶形运算为基本运算单元依式(2)和式(3)的规律逐层求得更多点数的 DFT，直到求出 $X(k)$，$k=1,\cdots,N-1$。输入数据和输出数据存储在相同的 RAM 空间，蝶形运算所涉及的数据地址可通过三重循环结构中的循环变量得出，实现流程如图 1 所示。外循环控制 FFT 执行层数，中循环控制每层中各子序列的运算，内循环控制具体子序列的蝶形运算执行。

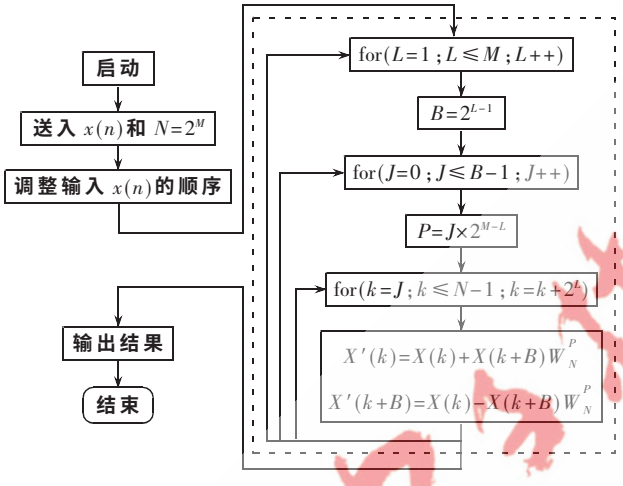

图 1 基 2-FFT 的执行流程

2 利用 FPGA 实现浮点 FFT 的方案

利用 FPGA 实现的浮点 FFT 可通过多种形式与系统其他电路接口，可以利用 MCU（微控制器）芯片控制 FPGA 做运算，也可把单片机或 ARM 核做到 FPGA 内部，即以 FPGA 为核心设计系统。此外，用以存储运算数据的 RAM 也可根据应用情况选择独立 RAM 芯片或用 FPGA 集成的外部 RAM（容量大，但速度慢）。这里考虑更一般的情况，通过一片独立的 MCU 芯片做控制，数据存储用 32 bit 外部 RAM（由 2 片 16 bit RAM 扩展），做 FFT 运算前 MCU 先将待转换的数据以倒序的形式存储到 RAM（也可正序存储，倒序由 FPGA 完成），然后发送启动命令，FFT 模块接管 RAM 进行 FFT 运算，完成时写入状态寄存器，MCU 通过查询或中断得知这一状态，重新控制 RAM 并将转换后的数据读出，如图 2 所示。

RAM 和命令/状态及其他功能寄存器统一编址，运算系统的时钟主频、FFT 点数、正弦向量表地址、输入/输出数据地址、RAM 读/写时序等都可通过 MCU 芯片向 FPGA 内相应的寄存器写入数据设置。FFT 通过 3 大功能模块协同工作实现，即循环控制模块 M1、RAM 访问模块 M2 和蝶形运算模块 M3。3 个模块同时运行，当蝶形运算执行时，RAM 访问模块执行上次 M3 输出数据的存储和下次输入数据的读取，同时循环控制模块完成循环变量的更新及下个周期中 RAM 读/写所需地址的计算。在每个新周期启动的时钟边沿时刻，模块间完成数据交换，即蝶形运算模块将运算结果存入 RAM 访问模块 M2 的输出缓冲器内，同时把 M2 的输入缓冲区里的数据读出，而循环控制模块把新地址送入 M2 的地址寄存器组中。其中 RAM 访问模块 M2 的数据读/写的切换时序以及具体每次读或写时的控制信号 CE、WE 和 OE 等的时序都可通过寄存器设置。循环控制模块 M1 中设置 3 个循环变量寄存器，代表图 1 中的 k、J 和 L。由当前的 k、J、L 值通过组合逻辑算出下次的 k、J、L 值，并在同一时钟边沿置入到相应的寄存器中。此外，由 k、J 和 L 值经组合逻辑运算可得出蝶形运算所涉及数据的地址偏移量。由于本方案采用了流水线思想和数据缓冲机制，故可使得系统效率达到最高。整个方案最关键的设计为蝶形运算模块 M3 的结构。

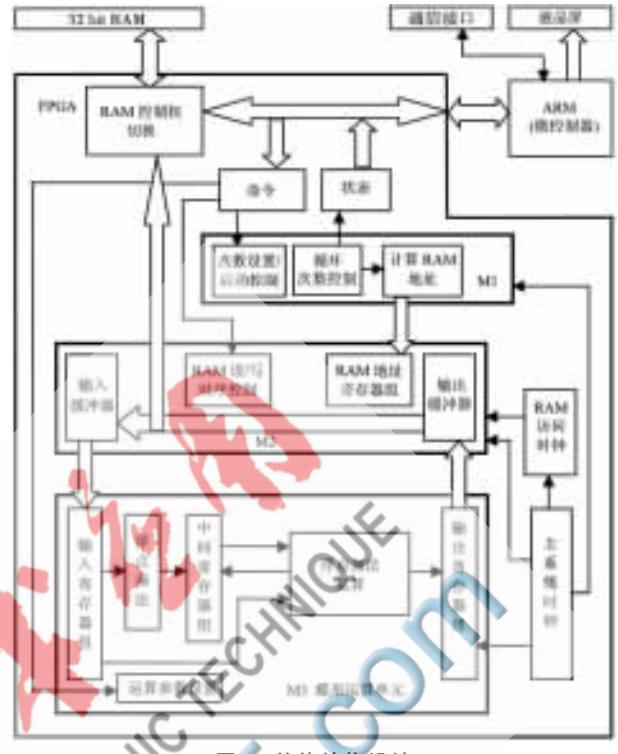

图 2 总体结构设计

3 蝶形运算单元

蝶形运算单元的设计是 FFT 的核心，其复杂性很高，且硬件资源消耗量巨大，其中浮点乘法器及浮点加法器的设计和配置情况尤为重要，因此设计中需兼顾速度、资源占用和协调等因素。由图 1 可知蝶形运算共需作 4 次浮点乘法和 6 次浮点加法，考虑资源占用情况，这里采用 1 个浮点乘法器和 2 个浮点加法器（可根据具体 FPGA 芯片调整）。4 次浮点乘法依次用乘法器用 4 个子周期实现，2 个浮点加法器并行运行，用 3 个子周期完成 6 次浮点加法运算，因此整个蝶形运算周期共需 7 个时钟子周期。浮点加法器 1 用来计算实部相关的运算，浮点加法器 2 用来完成虚部相关的运算。$R(\ )$和 $I(\ )$分

别表示求括号中复数的实部和虚部，执行过程为：前 4 个子周期依次求出两个复数实部/虚部分别相乘的结果，即 $R(X(k+B))R(W_N^P)$、$R(X(k+B))I(W_N^P)$、$I(X(k+B))R(W_N^P)$、$I(X(k+B))I(W_N^P)$；第 5 子周期的两次浮点加法运算得到两个复数相乘后的实部和虚部，即 $R(X(k+B)W_N^P)$和 $I(X(k+B)W_N^P)$；第 6 和第 7 子周期分别利用两个浮点加法器并行运算完成两个复数相加和相减，即第 6 子周期得到 $R(X'(k))$和 $I(X'(k))$，第 7 子周期得到 $R(X'(k+B))$和 $I(X'(k+B))$。

4 浮点乘法器

在通常的数字信号处理应用中，单精度浮点数即可满足处理精度要求。常用的浮点格式为 IEEE 754 标准[3]，其格式为符号位 $S[31]$，指数部分 $E[30:23]$，尾数部分 $M[22:0]$。S 为 0 时表示正数，为 1 时表示负数；指数部分 E 为实际指数加 127 偏移量的结果，取值范围为 $[1,254]$；M 有 23 位，再加上小数点左边一位隐含的 1 总共 24 位构成尾数部分。由它表示的浮点数的值 V 可以表示为：

$$V=(-1)^S\times 2^{E-127}(1.M) \tag{4}$$

浮点乘法积的符号位只需作异或即可求得，指数利用两个乘数的指数字段相加后再减去 127 求得（因为如式(4)指数字段是实际指数加 127 的偏移量得到的），但此时的指数尚需处理。因为尾数相乘后可能会进位 1 位，所以为了提高速度，将偏置后的指数分别做直接输出和加 1 输出，通过尾数相乘后的结果控制数据选择器选择最后的指数值。浮点乘法最主要的运算是进行位扩展后 24 位尾数的定点乘法。定点乘法主要步骤包括并行 Booth 编码、部分积产生、部分积压缩、进位传播加法。对进位传播加法的结果进行舍入和规格化处理后可得到乘积的尾数部分。浮点乘法器的结构如图 3 所示。

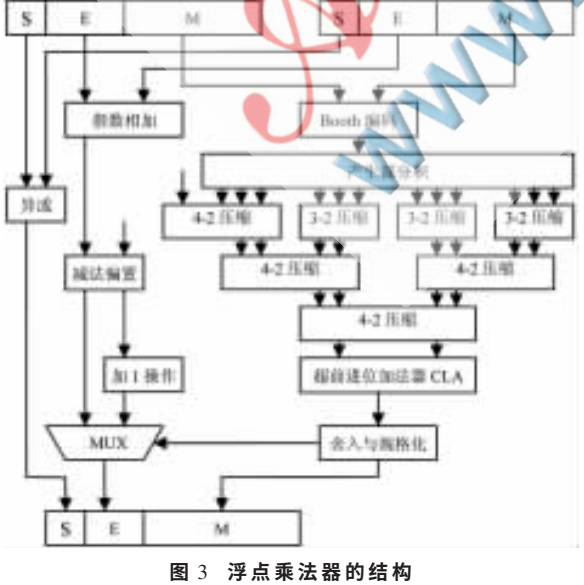

图 3 浮点乘法器的结构

并行 Booth 编码：n 位二进制数 Y 为：

$$Y=\sum_{i=0}^{n-1}Y_i2^i=\sum_{i=0}^{n/2-1}(Y_{2i-1}+Y_{2i}-2Y_{2i+1})2^{2i}+Y_{n-1}2^n$$

其中 $Y_{-1}=0$，括号中表达式的值可以是{−2，−1，0，+1，+2}。如乘数 B 按此方式以 3 位一组分成重叠的组，每组可能对被乘数 A 进行的并行乘法操作构成集合{−2X，−X，0，X，2X}，这些乘法操作的实现方法为：

(1)−2X：将 X 左移一位，再取反，并在其最低位加 1；

(2)−X：将 X 取反，在其最低位加 1；

(3)0：将 X 的各位全都用 0 代替；

(4)X：X 的各位保持不变；

(5)2X：将 X 左移一位。

被乘数通过组合逻辑输出上述 5 种结果，乘数构成重叠的 12 组 3 位数，分别控制 5 选 1 的数据选择器选择相应的输出，经偏移后补充前后数据即可得到各个部分积，加上最高位补充项共 13 个部分积。如何进行部分积压缩是区别各种算法的关键所在，最简单的部分积压缩方法是采用迭代的方法，通过移位累加操作完成，但这种算法速度过慢，N 个部分积的加法需要 N 个时钟周期，对于高位宽的浮点数乘法时间消耗太大。这里采用 Wallace 树的方法对部分积进行压缩，即利用保留进位加法器将部分积逐级压缩，考虑到部分积的数量采用了 3 级压缩，共用到 3 个 3−2 压缩器和 4 个 4−2 压缩器，最终得到 Carry 和 Sum，再将两者用超前进位加法器 CLA 相加。经过舍入和规格化后，结合算出的符号位和指数部分得到最终乘积。

5 浮点加法器

由于浮点数带符号，浮点加法实际可能做加法或减法操作，因此需要先进行加/减法判断，即符号位相同做加法，相反则做减法。但确定最终的符号还需知道两数绝对值的大小关系，绝对值的大小判断可根据两数的指数部分和尾数部分确定，指数大则绝对值大，指数相同时尾数大的绝对值大。大小判断结果结合任一加数的符号位以及加/减法关系即可确定和的符号位。因为可能做加法或减法，所以多数浮点加法器实现方案都将数据先做补码处理[4]，相加后再求回原码，这样就多进行了两次进位加 1 操作，将使速度降低。本设计为克服以上缺点，采用独立的定点加法器和定点减法器以避免求补码。根据加/减法判断结果使定点加法器或定点减法器工作，定点加/减法器的输入数据 A 为绝对值大的数，B 为绝对值小的数，直接利用大小判断结果控制数据选择器选择大绝对值数的尾数送入 A，B 的数据则需先求出两数的指数差，并依此将选择出的小绝对值数的尾数移位方可得到。加法或减法的结果根据实际操作经数据选择器选出，然后检测出结果前导 0 的个数，并依此做舍入处理，结合前导 0 的数目和舍入时进位的情况将尾数

移位规格化。同时,综合两数的指数、前导0数和舍入进位值可算出调整后的指数值。

6 硬件实现

利用Quartus-II编写了各模块程序并在仿真通过后,在Cyclone-II系列FPGA芯片EP2C8Q208中实现了该设计,运算控制由ARM7系列芯片LPC2214完成。EP2C8Q208采用TSMC的90 nm低K工艺,LEs的数量为8 256,整个设计的资源占用率为65%,浮点乘法运行速度可高达80 MFLOPS,在利用锁相环提供的200 MHz的系统时钟下完成1 024点复数FFT只需750 μs。若选用高速大容量的FPGA芯片,采用内置RAM,调整蝶形运算单元的配置,如增加浮点乘法器的数量后,速度还可大幅提高。

本文给出了一种基于FPGA实现浮点FFT的方案,蝶形运算、RAM访问和循环控制3个模块的同步及模块间的数据交换机制利用流水线思想设计,在蝶形运算中采用了并行处理技术,浮点乘法器和浮点加法器的设计在速度、面积以及结构均衡等方面的性能都较优越。基于MCU与FPGA共享RAM的方案设计非常实用,可以很容易地与各种MCU或MPU接口,使高速浮点FFT可以广泛应用于多种嵌入式设备中。

参考文献

[1] 张小妍,邵杰.高速浮点运算单元的FPGA实现[J].信息化研究,2009,35(11):24-27.

[2] 钱文明,刘新宁,张艳丽.基于Cyclone系列FPGA的1024点FFT算法的实现[J].电子工程师,2007,33(2):12-14.

[3] 张亚宜,米琦,高倩.基FPGA的FFT处理器设计[J].中国民航大学学报,2007,25(2):12-15.

[4] 连冰,宫丰奎,张力,等.基于FPGA的快速傅里叶变换[J].国外电子元器件,2003(12):26-28.

(收稿日期:2012-02-24)

作者简介:

刘健,男,1979年生,讲师,硕士研究生,主要研究方向:嵌入式系统设计。

史彩娟,女,1977年生,讲师,硕士研究生,主要研究方向:数字图像处理。

赵丽莉,女,1978年生,讲师,硕士研究生,主要研究方向:数字信号处理。