第 34 卷 第 24 期 计 算 机 工 程 2008 年 12 月
Vol.34 No.24 Computer Engineering December 2008
·工程应用技术与实现· 文章编号：1000—3428(2008)24—0238—03 文献标识码：A 中图分类号：TP301.6

高性能基 4 快速傅里叶变换处理器的设计

段小东，顾立志
(华侨大学机电及自动化学院，泉州 362021)

摘 要：研究并设计高性能基 4 快速傅里叶变换(FFT)处理器。采用基 4 算法、流水线结构的蝶形运算单元，提高了处理速度，使芯片能在更高的时钟频率上工作。运用溢出检测状态机对每个蝶形运算单元输出的数据进行块浮点检查，确保对溢出情况进行正确判断。验证与性能评估结果表明，该 FFT 处理器具有较高性能。

关键词：快速傅里叶变换算法；基 4 算法；块浮点算法；蝶形运算单元

Design of High Performance Radix-4 FFT Processor

DUAN Xiao-dong, GU Li-zhi
(College of Mechanical Engineering and Automation, Huaqiao University, Quanzhou 362021)

【Abstract】 This paper studies and designs a high performance radix-4 Fast Fourier Transform(FFT) processor. Based on the radix-4 algorithm, butterfly-like computing cell of pipeline structure, the processor is improved to work at a more rapidly processing speed and a higher clock frequency. By using the spilling state machine, the spilling detection for the output data of each butterfly-like computing cell with the block floating-point algorithm is reliable to determine the state of spilling. Validation and performance evaluation results show that the FFT processor has high performance.

【Key words】 Fast Fourier Transform(FFT) algorithm; Radix-4 algorithm; block floating-point algorithm; butterfly-like computing cell

1 概述

离散傅里叶变换(Discrete Fourier Transform, DFT)使数字信号处理可以在频域上以数字运算方法进行。DFT 是一种用于描述离散信号时域表示与频域表示关系的数学工具，它显著提高了数字信号处理的灵活性，其多种快速算法实现了信号实时处理和设备简化。DFT 具有重要理论意义，且在各种数字信号处理中发挥着核心作用。

DFT 是连续傅里叶变换在离散系统中的表示形式，由于 DFT 的计算量很大，因此在很长一段时间内其应用受到限制。Cooley 和 Tukey 于 20 世纪 60 年代提出快速傅里叶变换(Fast Fourier Transform, FFT)算法。该算法可以快速计算 DFT，明显降低了运算量，提高了 DFT 的运算速度，使运算时间缩短 1 个~2 个数量级。FFI 算法使 DFT 在实际应用中被广泛使用。FFT 在数字通信、语音处理、图像处理、仿真、系统分析、雷达、声纳、光学、医学影像、天文、地震信号分析等多个领域得到了应用。

2 基 4FFT 算法

FFT 算法的基本思想如下：利用旋转因子 W_N^{kn} 的 3 个特性，即周期性、对称性和可约性，将原有 N 点长序列的 DFT 分解成 2 个或多个较短序列的 DFT，并合并 DFT 运算中可以合并的项；计算短序列的 DFT 后，重新组合成原序列的 DFT，使运算量显著减少，从而达到提高速度的目的。上述分解方法可以分为 2 类：(1)对时间序列 $x(n)$进行逐次分解，称为时域抽取 FFT 法(Decimation-In-Time-FFT, DIT-FFT)；(2)对傅里叶变换序列 $X(k)$进行分解，称为频域抽取 FFT 法(Decimation-In- Frequency-FFT,DIF-FFT)。本文采用 DIT-FFT 法。

2.1 基 4 DIT-FFT 算法

DFT将时域信号转换成频率信号，长度为N的有限长序列$x(n)$的离散傅里叶变换[1]可以用式(1)表示。

$$X(k)=DFT[x(n)]=\sum_{n=0}^{N-1}x(n)W_N^{kn} \tag{1}$$

其中，$k,n=0,1,\cdots,N-1;W_N^{kn}=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}kn}$。

基 4 DIT-FFT算法的基本原理[2]如下：将一个N点的FFT分解为 4 个$N/4$ 点的序列，分别进行DFT计算；再将每个$N/4$点进一步分解为 4 个$N/16$ 点的计算，依此类推。同理，对于多点数(4^m)可以进行多级分解。对长度为 256 点的$x(n)$序列采用基 4 DIT-FFT算法可得

$$X(k)=X(k_3k_2k_1k_0)=\sum_{n_0=0}^{3}\sum_{n_1=0}^{3}\sum_{n_2=0}^{3}\sum_{n_3=0}^{3}x(n_3n_2n_1n_0)\cdot W_N^{k_0 4^3 n_3}W_N^{(k_1 4+k_0)4^2 n_2}W_N^{(k_2 4^2+k_1 4+k_0)4n_1}W_N^{(k_3 4^3+k_2 4^2+k_1 4+k_0)n_0} \tag{2}$$

基 4 蝶形运算如下：

$$A^*=A+BW_N^p+CW_N^{2p}+DW_N^{3p} \tag{3}$$

$$B^*=A-\mathrm{j}BW_N^p-CW_N^{2p}+\mathrm{j}DW_N^{3p} \tag{4}$$

$$C^*=A-BW_N^p+CW_N^{2p}-DW_N^{3p} \tag{5}$$

$$D^*=A+\mathrm{j}BW_N^p-CW_N^{2p}-\mathrm{j}DW_N^{3p} \tag{6}$$

其中，A^*,B^*,C^*,D^*是蝶算后的数据符号表示，而非共轭。

基 4 蝶形运算符号可以用图 1 表示。

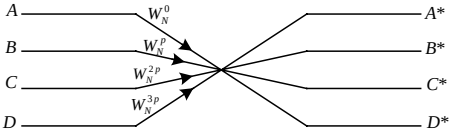

图 1 基 4FFT 蝶形运算符号

作者简介：段小东(1984－)，男，硕士研究生，主研方向：计算机应用技术，虚拟检测；顾立志，教授、博士生导师

收稿日期：2008-07-12 **E-mail：**2006dxd@163.com

2.2 算法比较与选择

人们通常以乘法次数和加法次数来衡量一个算法的性能。例如，对基 4FFT描述如下：每个蝶算为 3 个复数乘(每个复数乘为 4 个实数乘和 2 个实数加)，N个复数点，分为$\log_4 N$级，每级为$N/4$ 个基 4 蝶算。表 1 是 4 种常用基数算法运算量的比较[3]，其中，$r=\mathrm{lb}N$。

表 1　4 种常用基数算法运算量的比较

算法	实数乘法	实数加法
基 2	$(2r-4)N+4$	$(3r-2)N+2$
基 4	$(1.5r-4)N+4$	$(2.75r-2)N+2$
基 8	$(1.333r-4)N+4$	$(2.75r-4)N+4$
基 16	$(1.312\ 5r-4)N+4$	$(2.718\ 75r-2)N+4$

一般来说，基数越高总计算量越少，但判断一个算法的优劣不仅要考虑计算量，还应考虑算法复杂性。如表 1 所示，从基 2 到基 4，乘法和加法的运算次数发生了较大跳变，基 4 FFT 总的复数乘法次数比基 2 FFT 减少了 1/4。而从基 4 到基 8 以至基 16，运算次数变化的幅度不再明显。就复杂性来看，基 2 算法是最容易控制的，操作起来最简单；基 4 算法控制稍复杂，但仍然具有与基 2 算法的可类比性；基 8 和基 16 算法的控制复杂度与基 4 相比，跳变很明显。综合考虑运算速度和控制复杂度，基 4 算法在 FFT 处理器实现中具有最高的性价比。因此，本文选用基 4 FFT 算法来实现 FFT 处理器的设计。

3　FFT 处理器设计

本文设计的 FFT 处理器能完成定点复数傅里叶变换和傅里叶逆变换，数据总线宽度为 16 位，片内大容量 RAM 支持最高达 4 096 点的复数变换及 1 024 点的连续变换，复数输入数据和加权系数均为 32 位，其中，实部和虚部各 16 位。

3.1 算法设计

采用定点运算时，为了避免数据在运算过程中溢出，需要在每级运算中将数据右移，但这会导致数据的动态范围变差，因此，采用块浮点算法。块浮点算法基于数据块的自增益思想，在一个数据块上实现浮点，即一组数据共用一个移位因子，它在硬件上以独立的数据字段存储，硬件实现所需代价小于传统浮点运算，是浮点运算、定点运算的较好折中。块浮点数据块中的移位因子取决于整个数据块中所有数据的最大值，如果数据块中有一个数据较大，则该数据块共用一个较大的因子。如果数据块中的数据都较小，该数据块就共用一个较小的因子。

3.2 处理器设计

实现 FFT 算法的 4 种常用硬件结构如下：

(1)递归结构。其运算单元为一个基 4 蝶形运算单元，每级蝶形运算单元按递归方式运行，该结构使蝶形运算单元一直处于“忙”的状态。递归结构具有占用资源少、结构简单、控制逻辑简单、有利于提高系统稳定性等优点。

(2)级联结构。采用多个蝶形运算单元，每个蝶形运算单元负责一级处理。级联结构可以分为 2 种情况：1)利用“乒乓”存储器进行中间数据缓存以便下一级蝶形运算的数据抽取；2)采用交换器和延时器对每级蝶形运算的输入进行数据重排，延时器可采用 FIFO 构成，以实现流水线结构，减少存储器要求。

(3)并行结构。采用$N/4$ 个蝶形运算单元，若蝶形运算单元的计算时间为T，则每级蝶形运算所需时间也是T，整个变换需要迭代$\log_4 N$次，变换时间为$T\log_4 N$，该结构的运算速度很快，但硬件开销很大，对储存器带宽的要求很高。

(4)阵列结构。将 FFT 算法流程图中的每个运算节点都与一个蝶形运算单元相对应，该结构综合了级联结构与并行结构的特点，即每级蝶形运算由 $N/4$ 个蝶形运算单元并行运算，因此，每隔一个蝶形运算时间 T，完成一级蝶形运算。对于连续输入的多个序列，由于采用流水线结构运算，因此每个序列的 FFT 时间仅为 T，需要采用 $N/4\log_4 N$ 个蝶形运算单元来提高速度。

上述 4 种结构中用到的蝶形运算单元数目与 FFT 所需时间的乘积是常数。可见，FFT 运算速度的提高是以相应硬件开销为代价的，选择不同 FFT 运算结构，必须在运算速度和硬件开销间进行折中。随着硬件量的增加，各个运算单元的任务会越来越单一，因此，其内部模块可以做得更紧凑，地址和控制逻辑变得简单，有些逻辑甚至不再需要。

考虑速度和硬件资源的因素，本设计综合利用并行结构及递归结构的部分特点，在提高转换速度的同时确保控制电路不会太复杂。

FFT 处理器设计框图如图 2 所示。整个 FFT 处理器包括 6 个部分：蝶算单元，块浮点模块，“乒乓”RAM，旋转因子存储单元 ROM，控制单元和输入、输出单元。

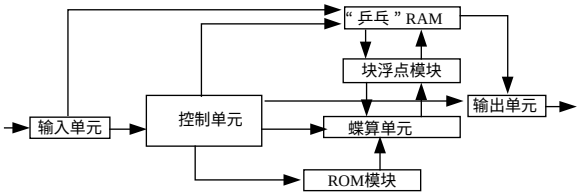

图 2　FFT 处理器设计框图

3.3 模块设计

3.3.1　蝶算单元

为了提高运算速度，考虑到双端口 RAM 的特点，本文把蝶算单元设计为流水线结构，每个算术运算结果都用寄存器保存，并选用流水线结构的乘法器。按上述方法设计的蝶算单元在 4 个时钟周期完成一次基 4 蝶算，即处理 4 个数据，相当于一个时钟周期处理一个数据。此方法虽然使用了较多乘法器和寄存器，但极大提高了运算速度，可以使芯片工作在更高的时钟频率上。图 3 描述了蝶算单元的处理流程。

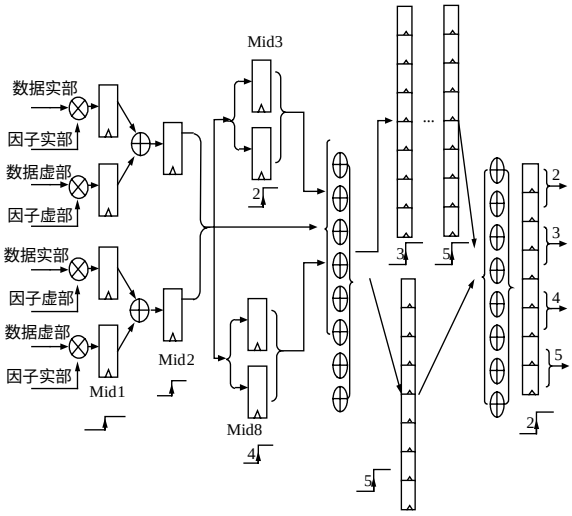

图 3　蝶算单元的处理流程

3.3.2　块浮点模块

块浮点的应用有效提高了数据的动态范围，其功能实现

主要由状态机完成。图 4 描述了溢出检测状态机，其中，①~④表示状态转移条件(被检测数据的高 4 (3)位)；①为 0000 或 1111，表示没有溢出；②为 0001 或 1110，表示溢出 1 bit；③为 001X 或 110X，表示溢出 2 bit；④为 01XX 或 10XX，表示溢出 3 bit。

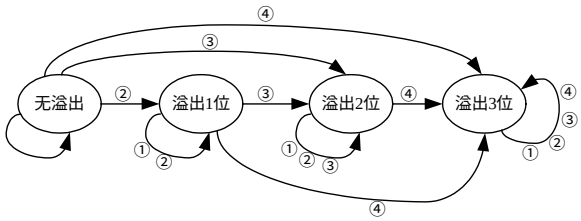

图 4　溢出检测状态机

为了确定溢出情况，对每个蝶算单元输出的数据都进行块浮点检查。在每级结束时，对 16 个蝶算单元的溢出情况进行汇总，产生一个总的溢出指示，根据这个溢出指示在下一级蝶算时，对输入数据进行左移。没有溢出时，左移 2(3)位；溢出 1 位时，左移 1(2)位；溢出 2 位时，左移 0(1)位；溢出 3 位时，左移 0 位。

溢出检测针对 3 bit 溢出或 2 bit 溢出，但在实际检测时，2 bit 溢出可以取高 3 位，并在更高位上扩展一个符号位，从而只用一个状态机就可以实现 2 类检测。

因为只有当 2 个符号不相同的数据进行加法操作时上述判断才正确，所以通常情况下，上述溢出检测方法是不可靠的。当 2 个符号相同的数据进行加法操作时，即使没有溢出，也可能出现 01XX 或 10XX 的情况(包括 001X 或 110X)，从而导致误判。因为在 FFT 蝶算中，2 个数据一定会被同时进行加法和减法操作，所以当蝶算后数据的被检测位出现上述状态时，一定存在实际溢出，因此，对蝶算数据使用该检测方法进行溢出检测是可靠的，并有效减少资源消耗。

3.3.3　“乒乓”RAM

芯片内部集成的RAM最多可以存储 4 096 点复数。由于处理 1 024 点FFT变换时，需要同时进行加载、运算、输出操作，因此RAM采用“乒乓”操作方式，其中一块用于存储当前级的输入数据，另一块用于存储前一级的结果。当本次运算完毕后，2 块RAM交换功能，使数据及时提供给蝶算单元，避免了额外的等待操作，提高了系统处理速度[4]。

1 024 点内部 RAM 的转换结构如图 5 所示。RAM 被分成 4 个逻辑块 A, B, C, D，其中，A, B 以“乒乓”方式载入并转换数据；C, D 以“乒乓”方式输出转换后的数据。下次要被转换的数据输入后存放在 RAM-A 中，此时，RAM-C 在输出上次的转换结果；RAM-D 在进行本次 FFT 转换。进行本次转换时，数据从 RAM-B 中取出，运算后存放在 RAM-D 的相应地址中。依此类推，在下一个 LOAD-TRANSFORM-DUMP 过程中，数据输入后存放在 RAM-B 中，RAM-A 中的数据进行当前 FFT 转换，中间数据及最后结果存放在 RAM-C 中，RAM-D 正在输出上次的转换结果。

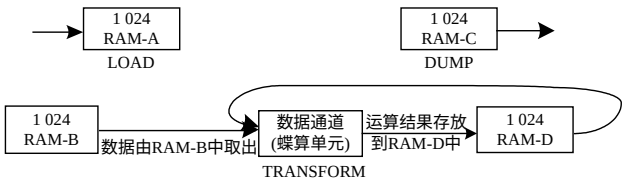

图 5　1 024 点内部 RAM 的转换结构

3.3.4 控制字的载入

本文采用输入控制字对芯片进行配置，使电路可以工作在多种模式下。为了简化设计，控制字的载入没有使用常用的单片机读写方法，而是利用复位信号的上升沿载入虚部端口数据来实现。虚部端口被定义为复用端口，当复位有效时，该端口上的数据为控制字；复位无效时，该端口数据是被转换数据序列的虚部或实数转换时的一组实数。

4　FFT 处理器验证与性能评估

针对本文提出的FFT处理器结构采用Xilinx公司的Virtex-II系列XC2V250-4FG456器件进行了 4 096 点复数及 1 024 点连续FFT处理器的VLSI结构验证。由于此器件包含大量的 16×16 位硬件乘法器、片内可配置RAM块和触发器资源，因此便于硬件设计验证。输入输出数据为 16 位，当系统工作频率为 80 MHz时，完成 1 024 点复数FFT运算所需时间仅为 11.8 μs。如图 6 给出了部分仿真波形，表 2 比较了 7 种FFT处理器的性能指标[5]。

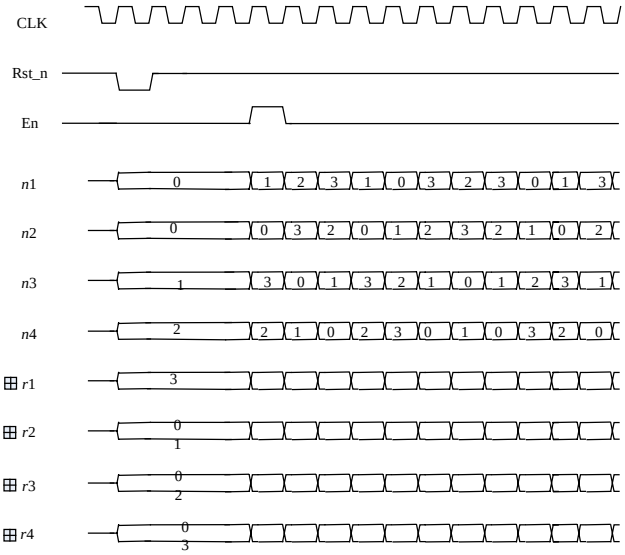

图 6　部分仿真波形

表 2　FFT 处理器的性能指标

处理器	FFT 点数	数据宽度/bit	时钟频率/MHz	处理时间/μs
M.Wosnitza,ETH	1 024	32	66	80.0
Tiger SHARC	1 024	32	250	41.0
Spiffee	1 024	20	173	30.0
Sicom,SNC960A	1 024	16	65	20.0
Altera FFT IP	4 096	16	94	36.0
本文处理器	1 024	16	80	11.8
本文处理器	4 096	16	80	25.0

由表 2 可见，本文提出的基 4“乒乓”RAM存储结构的控制部件简单、地址生成速度快，它使数据可以及时提供给蝶形运算单元，无须额外等待操作，提高了系统的处理速度。该结构对各种形如$N=4^n$的FFT运算能达到较高处理性能。

5　结束语

本文高性能 FFT 处理器可以完成 4 096 点的 FFT，通过“乒乓”RAM 可以实现 1 024 点 FFT 的连续变换，与原有设计相比，有效提高了系统的处理速度，在系统时钟为 80 MHz 的情况下，完成 4 096 点复数 FFT 运算只需 25 μs。该处理器利用块浮点算法有效提高了数据精度，且在复位信号的上升沿读入控制字，使用户可以方便地配置其工作模式。

(下转第 243 页)