基于 FPGA 的基 8-FFT 处理器设计

宋 玮[1],李如玮[1],代栋敏[2]

1. 北京工业大学电子信息与控制工程学院,北京 100124
2. 北京理工大学信息与电子学院,北京 100081

摘要 提出了在现场可编程门阵列(FPGA)上实现 4096 点基 8 快速傅里叶变换(FFT)算法的设计方案。方案对蝶形器、旋转因子产生器和输入/输出接口进行了分析和优化,整个算法的流程采用了流水线的工作方式,提高了运算速度并减小了 FPGA 内部资源的占用。通过仿真测试,并同 Matlab 定点模型进行了对比。本设计方案在 100MHz 的时钟下,完成 4096 点基 8-FFT 运算需要 2.048μs,完全满足高速数字信号处理的要求。

关键词 快速傅里叶变换;FPGA;蝶形器;旋转因子

中图分类号 TN47 **文献标识码** A **文章编号** 1000-7857(2010)16-0067-04

Radix-8 FFT Processor Design Based on FPGA

SONG Wei[1], LI Ruwei[1], DAI Dongmin[2]

1. *College of Electronic Information and Control Engineering, Beijing University of Technology, Beijing 100124, China*
2. *School of Information and Eletronics, Beijing Institute of Technology, Beijing 100081, China*

Abstract A design of 4096-point radix-8 FFT is implemented on Field-Programming Gate Array (FPGA). Traditional radix-2 and radix-4 FFT processors could not satisfy the requirements of modern high-speed digital signal processing, so the radix-8 shared-memory architecture is used at the top-level. The butterfly module, the twiddle factor generation module, the input-output interface module are analyzed and optimized. A novel method to generate twiddle factors is proposed and compared with the traditional method. The pipeline style design increases the computing speed and decreases the FPGA resource utilization. Simulation verification is done and the result is compared with that of Matlab fixed-point model. The design is finally programmed to an Altera EP2S60F672I4 device and is verified with the help of a digital signal processor. The computing results with various input patterns are retrieved to Matlab and compared with the fixed-point model bit by bit. Under the clock frequency of 100MHz, the design takes 2.048μs to finish 4096-point radix-8 FFT, so it can meet the requirement of high speed digital signal processing.

Keywords fast Fourier transform; FPGA; butterfly module; twiddle factor

0 引言

离散傅里叶变换(Discrete Fourier Transform,DFT)是数字信号处理中的一种重要算法,但计算量大、运算时间长的缺点使其使用范围有限。1965 年,Cooley 和 Tukey 提出快速傅里叶变换(Fast Fourier Transform,FFT)算法,利用旋转因子的周期性和对称性,把长序列的 DFT 分解成更小点数的 DFT,从而减少乘法次数,提高计算效率,此算法一经问世就得到极大关注,发展迅速,出现了很多相关的算法。1968 年,Bergland 对 FFT 的基 2、基 4 算法进行了改进,推导出了基 8、基 16 算法。1977 年,Kolba 和 Park 提出素因子算法(Prime Factor Algorithm,PFA),与基 2 类算法结构完全不同,此算法虽然简便但必须满足 DFT 长度 N 为若干互素因子之积的条件。之后,Winograd 提出了以素因子算法为基础的 Winograd 傅里叶变换算法(Winograd Fourier Transform Algorithm,WFTA),此算法在 $N\leqslant 16$ 点时结构简单,运算量少,但是大于 16 点时仍存在结构复杂的缺点。PFA 和 WFTA 算法相对复

收稿日期:2010-07-13

作者简介:宋玮,研究方向为通信工程,电子信箱:bensoning@gmail.com;李如玮(通信作者),副教授,研究方向为语音信号处理与编码、小波变换在语音信号处理中的应用、语音增强等,电子信箱:liruwei@bjut.edu.cn

杂,不利于硬件实现,在数字信号处理领域常用的是基 2 和基 4 的 FFT 算法。FFT 算法被广泛运用于雷达系统、图像处理、通信系统、航天航空等领域,随着对系统性能指标要求的提升,传统的基 2、基 4 算法已不能满足雷达、通信和电子对抗等高速处理系统的计算要求,采用高基数算法成为提高 FFT 处理速度的主要途径之一[1]。本文采用按时间抽取的基 8-FFT 算法实现 4096 点的超高速 FFT 处理器。

1 系统顶层模块设计

设计采用共享内存架构,按照功能将 FFT 处理器划分为地址产生、旋转因子生成、内存接口和蝶形器模块。各个模块之间的连接关系如图 1 所示。图中,fft_top 为顶层模块名称,其外部信号含义是:addr 为输入数据的地址,data_in 为输入的数据,data_out 为输出的数据,done 为 FFT 模块计算完成标志;其内部信号含义是:tf_addr 为旋转因子的地址,addr0~addr7 分别为蝶形运算数据 x0~x7 对应的地址,x0~x7 为蝶形器的输入值,y0~y7 为蝶形器的输出值。

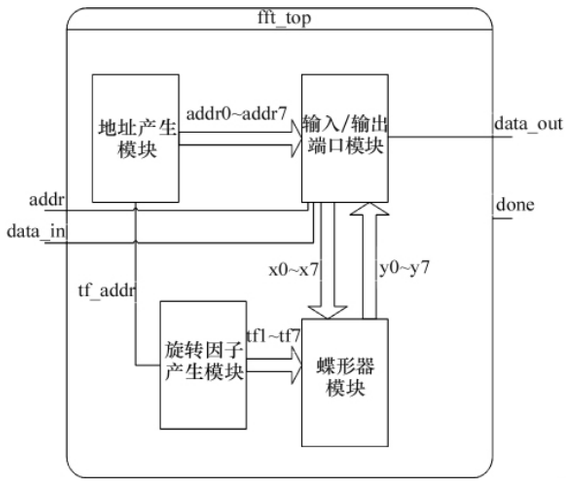


图 1 系统顶层架构

Fig. 1 Top-level architecture

2 各个模块的设计

按照系统顶层架构中罗列的模块,下面分别介绍各模块的设计思路。

2.1 地址产生模块

地址产生模块负责产生 8 个数据地址和 1 个旋转因子地址。由于采用原位计算,所以读数据和写数据的地址其实是一个,并且写数据地址是读数据地址的流水延迟。FFT 的每一级数据地址包括一个组计数器和一个蝶形计数器,组计数器的范围是 0~511,蝶形计数器的范围是 0~7。组计数器对于每个蝶形运算不变,在每级的运算中改变;蝶形计数器在每级的蝶形运算中随着地址的改变而改变。地址产生的变化方式如表 1 所示。

表 1 地址产生方案

Table 1 Address generation scheme

级	数据地址	旋转因子地址
0	{g[8:0], b}	0
1	{g[8:3], b, g[2:0]}	{g[2:0], 6′b$_0$}
2	{g[8:6], b, g[5:0]}	{g[5:0], 3′b$_0$}
3	{b, g[8:0]}	{g[8:0]}

2.2 旋转因子产生模块

在基 8-FFT 中,在已经得到旋转因子(twiddle factor) tf_1 的情况下,旋转因子 $tf_k(k=2,3,4,5,6,7)$ 由下式确定

$$tf_k=tf_1^k \tag{1}$$

已知 $tf_1=e^{-j\theta_1}$,则

$$tf_k=(e^{-j\theta_1})^k=e^{-jk\theta_1}=e^{-j\theta_k} \tag{2}$$

所以

$$\theta_k=k\theta_1 \tag{3}$$

第 $k(k=2,3,4,5,6,7)$ 个旋转因子对应的角度是第一个旋转因子对应角度的 k 倍,并且这些角度位于一个单位圆上。单位圆上旋转角度的增长是线性的,同时角度恰好正比于旋转因子查找表的地址。旋转因子产生模块从地址产生模块获得第一个旋转因子 tf_1 的地址,通过二进制的移位和加法生成其余旋转因子的地址。给定第一个旋转因子的地址 addr_tf$_1$={3′b000,addr_tf},则其余旋转因子地址为

addr_tf$_2$={2′b00,addr_tf,1′b0}

addr_tf$_3$=addr_tf$_1$+addr_tf$_2$

addr_tf$_4$={1′b0,addr_tf,2′b00}

addr_tf$_5$=addr_tf$_3$+addr_tf$_2$

addr_tf$_6$=addr_tf$_4$+addr_tf$_2$

addr_tf$_7$=addr_tf$_4$+addr_tf$_3$

因为只有移位和加法运算[2],以上公式便于硬件实现。

对于现场可编程门阵列(Field-Programmable Gate Array,FPGA)内部的旋转因子,只读内存(ROM)只储存 0~-π/4 的旋转因子的正值。地址产生模块产生的旋转因子首地址首先送入移位/加法模块,产生 7 个旋转因子地址。这 7 个地址的高位经过流水延迟后送入解码模块,低位送入旋转因子 ROM 堆。ROM 堆给出的数据经过解码模块解出正确的旋转因子数据并送出。设计框图如图 2 所示。

对关键路径的分析可知,关键路径不位于旋转因子产生模块。传统方法需要 7 片容量为 1024×36b 的 ROM,而在满足时序要求下,使用文中所述方法计算旋转因子所需 FPGA 内部 ROM 的容量减小至 3 片双口和 1 片单口 ROM,每片 ROM 容量为 512×36b,从而减少了 FPGA 内部 ROM 资源占用率。FPGA 内部相同容量的双口 ROM 和单口 ROM 占用的内存单元相同,此方法可以缩减旋转因子 ROM 面积至传统方法的 29%,同时不带来时序影响。

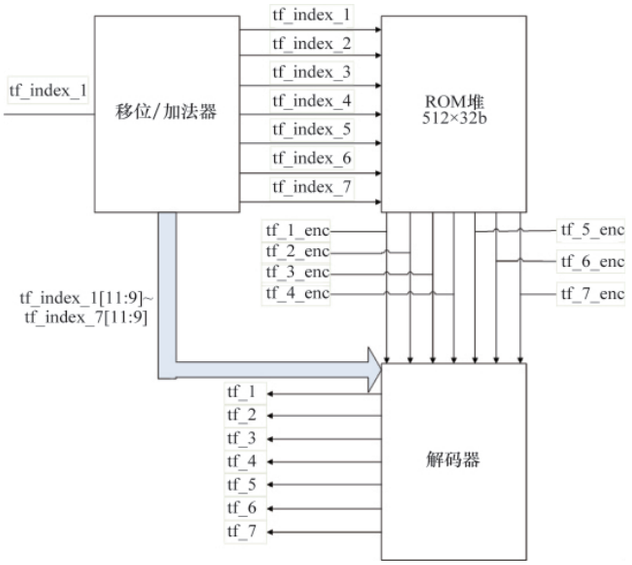


图 2　旋转因子产生器框图

Fig. 2　Twiddle factor generator diagram

2.3　内存接口设计

基 8 蝶形器需要同时取到 8 个参与运算的数据，这就要求至少将内存分为 8 块(bank)，并且每次取数据都保证分别从每块内存中取得并不发生冲突。根据基 8-FFT 算法取地址的特点，使用以下方案来保证每次取地址不发生块冲突。

给定一个地址 addr(对于 4096 点 FFT 运算，此地址为 12 位)对应的块号码

```
bank=addr[11:9]+addr[8:6]+addr[5:3]+addr[2:0];
```

对应的每块内存中的地址为

```
bank_addr=addr[11:3];
```

在 Matlab 中对此取地址方案进行验证，结果表明 4 级运算中都不存在取地址冲突的问题。

内存接口模块除了负责运算状态下的读取、写入数据之外，还负责输入和输出状态下的数据分配。对于本设计的按时间抽取 FFT 算法，要求输入逆序，输出正序。输入的数据需要先经过按三位翻转

```
addr_in={addr[2:0],addr[5:3],addr[8:6],addr[11:9]};
```

这样得到的地址按照上面提到的块内存访问方案进入其中的一块内存中。输出数据与输入数据的操作类似，只是不需要进行按三位翻转。

2.4　蝶形器设计

利用 DFT 的运算公式以及旋转因子的对称性和周期性得到基 8-FFT 的算法。按时间抽取(Decimation In Time，DIT)的基 8-FFT 算法是将输入序列在时域上的次序按 $8r$、$8r+1$、$8r+2$、$8r+3$、$8r+4$、$8r+5$、$8r+6$、$8r+7$ 抽取，对于长度为 4096 点的 DFT 运算，可以采用 4 次分解，最后分解成 8 点的 DFT 的运算[3-5]。由 DFT 的定义[6]，有

$$X(k)=\mathrm{DFT}[x(n)]=\sum_{n=0}^{N-1}x(n)W_N^{nk}\quad(k=0,1,2,\cdots,N-1)\quad(4)$$

其中，$x(n)$为列长为 $N(n=0,1,2,\cdots,N-1)$的输入序列。

再利用旋转因子 W_N^{nk} 的周期性和对称性，将式(4)化为

$$\begin{aligned}X(k)=&\sum_{r=0}^{(N/8)-1}x(8r)W_{N/8}^{rk}+W_N^{k}\sum_{r=0}^{(N/8)-1}x(8r+1)W_{N/8}^{rk}+\\&W_N^{2k}\sum_{r=0}^{(N/8)-1}x(8r+2)W_{N/8}^{rk}+W_N^{3k}\sum_{r=0}^{(N/8)-1}x(8r+3)W_{N/8}^{rk}+\\&W_N^{4k}\sum_{r=0}^{(N/8)-1}x(8r+4)W_{N/8}^{rk}+W_N^{5k}\sum_{r=0}^{(N/8)-1}x(8r+5)W_{N/8}^{rk}+\\&W_N^{6k}\sum_{r=0}^{(N/8)-1}x(8r+6)W_{N/8}^{rk}+W_N^{7k}\sum_{r=0}^{(N/8)-1}x(8r+7)W_{N/8}^{rk}\\&(r=0,1,2,\cdots,(N/8)-1)\end{aligned}\quad(5)$$

令 $A=\sum_{r=0}^{(N/8)-1}x(8r)W_{N/8}^{rk}$，$B=\sum_{r=0}^{(N/8)-1}x(8r+1)W_{N/8}^{rk}$，$C=\sum_{r=0}^{(N/8)-1}x(8r+2)W_{N/8}^{rk}$，$D=\sum_{r=0}^{(N/8)-1}x(8r+3)W_{N/8}^{rk}$，$E=\sum_{r=0}^{(N/8)-1}x(8r+4)W_{N/8}^{rk}$，$F=\sum_{r=0}^{(N/8)-1}x(8r+5)W_{N/8}^{rk}$，$G=\sum_{r=0}^{(N/8)-1}x(8r+6)W_{N/8}^{rk}$，$H=\sum_{r=0}^{(N/8)-1}x(8r+7)W_{N/8}^{rk}$，$W_N^k=W^P$，可得

$$X(k)=A+W^PB+W^{2P}C+W^{3P}D+W^{4P}E+W^{5P}F+W^{6P}G+W^{7P}H\quad(6)$$

$$\begin{aligned}X(k+N/8)=&A+\frac{\sqrt{2}}{2}(1-\mathrm{j})W^PB-\mathrm{j}W^{2P}C-\frac{\sqrt{2}}{2}\cdot\\&(1+\mathrm{j})W^{3P}D-W^{4P}E-\frac{\sqrt{2}}{2}(1-\mathrm{j})W^{5P}F+\\&\mathrm{j}W^{6P}G+\frac{\sqrt{2}}{2}(1+\mathrm{j})W^{7P}H\end{aligned}\quad(7)$$

$$\begin{aligned}X(k+2N/8)=&A-\mathrm{j}W^PB-W^{2P}C+\mathrm{j}W^{3P}D+W^{4P}E-\\&\mathrm{j}W^{5P}F-W^{6P}G+\mathrm{j}W^{7P}H\end{aligned}\quad(8)$$

$$\begin{aligned}X(k+3N/8)=&A-\frac{\sqrt{2}}{2}(1+\mathrm{j})W^PB+\mathrm{j}W^{2P}C+\frac{\sqrt{2}}{2}\cdot\\&(1-\mathrm{j})W^{3P}D-W^{4P}E+\frac{\sqrt{2}}{2}(1+\mathrm{j})W^{5P}F-\mathrm{i}W^{6P}G-\\&\frac{\sqrt{2}}{2}(1-\mathrm{j})W^{7P}H\end{aligned}\quad(9)$$

$$\begin{aligned}X(k+4N/8)=&A-W^PB+W^{2P}C-W^{3P}D+\\&W^{4P}E-W^{5P}F+W^{6P}G-W^{7P}H\end{aligned}\quad(10)$$

$$\begin{aligned}X(k+5N/8)=&A-\frac{\sqrt{2}}{2}(1-\mathrm{j})W^PB-\mathrm{j}W^{2P}C+\frac{\sqrt{2}}{2}(1+\mathrm{j})W^{3P}D-\\&W^{4P}E+\frac{\sqrt{2}}{2}(1-\mathrm{j})W^{5P}F+\mathrm{i}W^{6P}G-\frac{\sqrt{2}}{2}(1+\mathrm{j})W^{7P}H\end{aligned}\quad(11)$$

$$\begin{aligned}X(k+6N/8)=&A+\mathrm{j}W^PB-W^{2P}C-\mathrm{j}W^{3P}D+W^{4P}E+\\&\mathrm{j}W^{5P}F-W^{6P}G-\mathrm{j}W^{7P}H\end{aligned}\quad(12)$$

$$\begin{aligned}X(k+7N/8)=&A+\frac{\sqrt{2}}{2}(1+\mathrm{j})W^PB+\mathrm{j}W^{2P}C-\frac{\sqrt{2}}{2}\cdot\\&(1-\mathrm{j})W^{3P}D-W^{4P}E-\frac{\sqrt{2}}{2}(1+\mathrm{j})W^{5P}F-\\&\mathrm{j}W^{6P}G+\frac{\sqrt{2}}{2}(1-\mathrm{j})W^{7P}H\end{aligned}\quad(13)$$

蝶形器设计如图 3 所示。

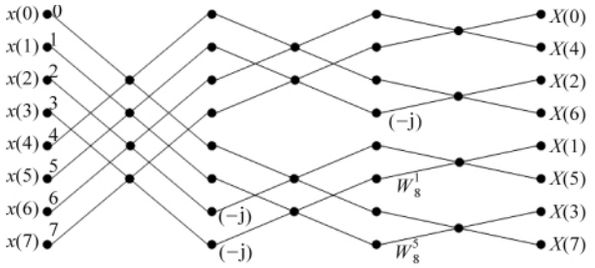


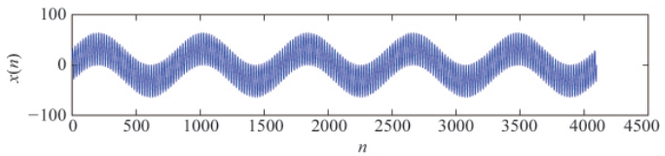
图 3　碟形器图

Fig. 3　Butterfly diagram

3　验证和测试

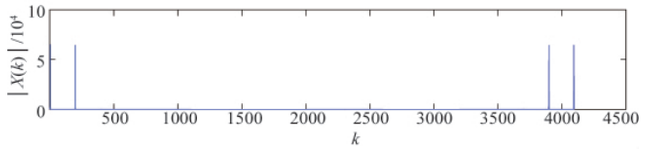
首先在 Matlab 中建立基 8-FFT 处理器的定点处理模型，然后在 Mentor Graphics QuestaSim 中使用 Verilog HDL 完成寄存器传输级(Register Transfer Level，RTL)编码。通过 Matlab 定点模型及 RTL 编码对数据处理结果的对比进行前仿真验证，测试数据包括正弦波、调制正弦波、白噪声、全 0 数据、固定数据。结果表明，RTL 编码与 Matlab 定点模型结果一致。

针对 Altera EP2S60F672I4 器件综合和布局布线后，占用 4746 个逻辑单元，217088b 内存和 64 个 DSP 模块。布局布线后对 FPGA 进行编程，通过 TI TMS320C6416 的 EMIFB 接口对 FPGA 输入/输出数据。输入的测试数据与前仿真时一致。FPGA 运算结果由数字信号处理器（Digital Signal Processor，DSP）取回并从 Code Composer Studio 导出到 Matlab，与 Matlab 定点结果自动进行二进制逐位对比。由于 Matlab 定点模型通过 Matlab 内部的浮点 fft 函数验证正确，同时与 FPGA 实现结果按二进制逐位一致，所以可以证明 FPGA 的实现正确。图 4 为 FPGA 运算结果与 Matlab 中的 fft 函数结果对比。



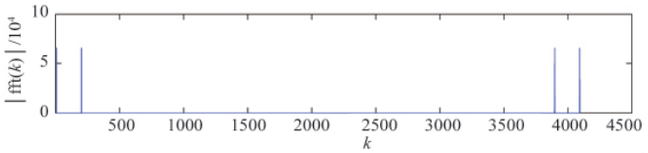
(a) 输入数据

(a) Input data



(b) FPGA 实现的结果

(b) Computing results of FPGA implementation



(c) Matlab 的 fft 函数运算结果

(c) Computing results of Matlab fft function

图 4　运算结果与 Matlab 中的 fft 函数结果对比

Fig. 4　Computing results compared with the fft function of Matlab

4096 点基 8-FFT 共需要 4 级运算，每级需要计算 512 个蝶形，每个蝶形需要 1 个时钟周期，所以完成 4096 点 FFT 运算的总时钟周期数为 4×512=2048；FPGA 编程后，100MHz 时钟频率下完成一次 FFT 运算需要的时间为 2.048μs。

4　结论

针对现代高速信号处理系统的要求，本文提出了一种基 8-FFT 处理器的设计和实现。通过对内部各个模块的分析和优化，提高了运算速度并占用较少的 FPGA 内部资源。实验结果表明，本设计满足雷达、通信和电子对抗等高速处理系统的计算要求。

参考文献(References)

[1] 张竺君，钱建平. 基于 FPGA 的超高速 FFT 处理器的设计[C]. 第四届江苏省电机工程青年科技论坛，南京：江苏省电机工程学会，2009.
Zhang Zhujun, Qian Jianping. High-speed FFT Processor Design Based on FPGA [C]. The 4th Jiangsu Young Electric Scientists & Engineers Forum, Nanjing: Jiangsu Society for Electrical Engineering, 2009.

[2] Jacobson A T, Truong D N, Baas B M. The design of a reconfigurable continuous-flow mixed-radix FFT processor [DB/OL]. 2009, http://www.ece.ucdavis.edu/vcl/pubs/2009.05.ISCAS.FFT/Dean_ISCAS_2009.pdf.

[3] Jia L, Gao Y, Tenuunen H. Efficient VLSI implementation of radix-8 FFT algorithm [C]//Proc 1999 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing. Victoria: IEEE, 1999.

[4] Widhe T, Melander J, Wanhammar L. Design of efficient radix-8 butterfly PEs for VLSI [C]//ISCAS'97, Proceedings of 1997 IEEE International Symposium on Circuits and Systems. Hongkong: IEEE International Symposium on Circuits and Systems, 1997.

[5] 林晗，夏宇闻，陈杰. 一种改进型基-8 FFT 算法及其 ASIC 实现[J]. 中国集成电路，2003，52(9)：68-72.
Lin Han, Xia Yuwen, Chen Jie. *China Integrated Circuit*, 2003, 52(9): 68-72.

[6] Bouguezel S, Ahmad M O, Swamy M N S. Improved radix-4 and radix-8 FFT algorithms [C]//ISCAS'04, Proceedings of the 2004 International Symposium on Circuits and Systems. Vancouver: IEEE International Symposium on Circuits and Systems, 2004.

(责任编辑　刘志远)

本期好玩的数学——巧妙填数答案