文章编号: 1671-7449(2009)05-0416-07

基于流水线的 FFT 快速计算方法与实现技术

董　志, 张羿猛, 黄芝平, 唐贵林, 刘纯武

(国防科技大学 机电工程与自动化学院, 湖南 长沙 410073)

摘　要: 针对目前 CDMA 快速码捕获系统对捕获速度要求越来越高,在分析快速傅里叶算法理论的基础上, 结合 FPGA (Field Programmable Gate Array) 的独特硬件结构,提出一种基于流水线的 FFT (Fast Fourier Transform) 快速实现方法,并对该方法进行了 matlab 仿真、ISE 仿真和 FPGA 实验. 研究结果证明:相比于传统的FFT 实现方法,在保证计算精度的基础上,该方法实现了 FFT 计算数据的连续输入与输出,减小了捕获时延,缩短了至少 1/3 的计算时间,在100 MHz 时钟时,完成 4096 点的 FFT 运算只需要 42.05 μs,为高速信号处理系统提供了一种更好的时频转换方法.

关键词: 流水线; 快速傅立叶变换; FPGA; matlab
中图分类号: TN911.72　　**文献标识码:** A

Theory and Application of FFT Based on Pipelined Stream

DONG Zhi, ZHANG Yimeng, HUANG Zhiping, TANG Guilin, LIU Chunwu
(College of Mechatronics Engineering and Automation, National University of Defense Technology, Changsha 410073, China)

Abstract: In order to reduce the calculation time of code acquisition in CDMA system, this paper analyzes the theory of FFT and the unique structure of FPGA, presents a new method for computing FFT based on pipelined stream. After Matlab simulation, ISE simulation and FPGA experiment, the results show that: the method has high precision and uses only 42 μs, which is reduced by at least 1/3 of the computing time comparing with previous methods, to compute 4096 points FFT with the clock of 100 MHz. It offers a better DFT method for high speed digital signal processing.

Key words: stream; fast fourier transform; field programmable gate array; matlab

FFT 算法[1]是离散傅里叶变换 (DFT) 的一种高效算法,它被广泛地运用于图像处理[2]、音频编码[3]、频谱分析等数字信号处理 (DSP) 领域,其变换长度 N 一般要求很大,而且都要求在很短的时间内完成数千点的 FFT 运算. 相比于 FPGA,DSP 处理器时延较大、接口不够灵活, 而且没有 FFT 运算所需的巨量存储器, 需外置特定的接口、控制芯片和 RAM, 而且 DSP 内部一般只有 1 个或 2 个 DSP 核,运算能力受限. 如果采用专用的 FFT 处理芯片,虽然速度能达到要求, 但其外围电路复杂、可扩展性差,并且价格昂贵. FPGA 具有可配置性强、速度快、密度高、功耗低的特点,而且目前的 FPGA 内部最多集成

* **收稿日期:** 2009-01-03
基金项目: 国家发改委产业化应用示范工程基金资助项目(2006942)
作者简介: 董志(1979-), 男, 博士生, 主要从事网络化测试技术研究.

有上百个DSP核,易于实现并行运算,适合于运算量大的前端数字信号处理. 根据某CDMA快速码捕获系统对捕获速度要求比较高的特点,提出了一种基于流水线的FFT快速实现方法.

相比于基-4 FFT,基-2 FFT具有延时更小的特点. 因此,本文在基-2 FFT的基础上提出了一种改进结构的快速算法,使整个FFT运算实现流水线操作,并通过实验证明了这种算法的有效性. 该方法同样适用于基-4 FFT结构.

1　基-2 FFT的基本原理

长度为$N=2^L$的有限长序列$x(n)$,$(n=0,1,N-1)$的DFT为

$$x(k)=\sum_{n=0}^{N-1}x(n)W_N^{kn},\ k=0,1,\cdots,N-1, \tag{1}$$

式中:$W_N^{kn}=\exp\{-j2\pi kn/N\}$为旋转因子.

上述DFT的运算中存在大量重复的算法,Cooley和Tukey提出的快速傅里叶变换算法,即基-2FFT有效地减少了乘法的次数. 首先将序列$x(n)$按n的奇偶分成两组.

$$\left.\begin{aligned}x(2r)&=x_1(r)\\x(2r+1)&=x_2(r)\end{aligned}\right\},r=0,1,\cdots,\frac{N}{2}-1, \tag{2}$$

则可将DFT化为

$$X(k)=DFT[x(n)]=\sum_{\substack{n=0\\ n\text{为偶数}}}^{N-1}x(n)W_N^{nk}+\sum_{\substack{n=0\\ n\text{为奇数}}}^{N-1}x(n)W_N^{nk}=$$

$$\sum_{r=0}^{N/2-1}x(2r)W_N^{2rk}+\sum_{r=0}^{N/2-1}x(2r+1)W_N^{(2r+1)k}=\sum_{r=0}^{N/2-1}x_1(r)(W_N^2)^{rk}+\sum_{r=0}^{N/2-1}x_2(r)(W_N^2)^{rk}.$$

由于$W_N^2=e^{-j\frac{2\pi}{N}2}=e^{-j\frac{2\pi}{N/2}}=W_{N/2}$,则

$$X(k)=\sum_{r=0}^{N/2-1}x_1(r)W_{N/2}^{rk}+W_N^k\sum_{r=0}^{N/2-1}x_2(r)W_{N/2}^{rk}=X_1(k)+W_N^kX_2(k), \tag{3}$$

式中:$X_1(k)$及$X_2(k)$分别是$x_1(r)$及$x_2(r)$的$N/2$点DFT

$$X_1(k)=\sum_{r=0}^{N/2-1}x_1(r)W_{N/2}^{rk}=\sum_{r=0}^{N/2-1}x(2r)W_{N/2}^{rk}, \tag{4}$$

$$X_2(k)=\sum_{r=0}^{N/2-1}x_2(r)W_{N/2}^{rk}=\sum_{r=0}^{N/2-1}x(2r+1)W_{N/2}^{rk}. \tag{5}$$

由公式(3)可以看出,一个N点的DFT已分解成两个$N/2$点的DFT,它们按公式(3)有组合成一个N点的DFT. 同时根据应用系数的周期性,即$W_{N/2}^{rk}=W_{N/2}^{r(k+N/2)}$,可以得到

$$X_1(k)=X_1\left(\frac{N}{2}+k\right),\ X_2(k)=X_2\left(\frac{N}{2}+k\right).$$

这样,只要求出0到$\left(\frac{N}{2}-1\right)$区间内的所有$X_1(k)$及$X_2(k)$值,即可求出0到$(N-1)$区间内的所有$X(k)$值,大大节省了运算. 由于$N=2^L$,因而可以依次类推,直到分解为2点的DFT. 这种方法称为按时间抽取的FFT算法(DIT),还有一种算法称为按频率抽取的FFT算法(DIF). 前者使用相对比较普遍,本文重点介绍DIT方法.

2　基-2 DIT FFT的同址计算方法

如图1所示为传统基2-FFT的实现流图. 每一级运算都需要$N/2$(N表示该级输入点数)个基2蝶形运算来完成,每个蝶形因子的运算包括1次复乘和2次复数加法. 4096点的FFT运算一共需要$\log_2(4096)=12$级蝶形运算因子. 蝶形因子的计算是FFT计算的核心部分,以其中的第m级蝶形因子为例,如图2所示. 可以建立式(6)的方程

$$\begin{aligned}X_m[p]&=X_{m-1}[p]+W_N^rX_{m-1}[q],\\X_m[q]&=X_{m-1}[p]-W_N^rX_{m-1}[q].\end{aligned} \tag{6}$$

在传统的 FFT 算法中,每一级的基 2 蝶形因子采用同址计算的方法. 所谓同址计算,即首先将上一级输出的数据送入数据存储 RAM 中,在进行蝶形运算的时候,蝶形因子单元从数据存储 RAM 中读出数据,经过计算得到的中间结果仍然存入相同的数据存储 RAM 去,覆盖输入的数据,直到整个这一级的蝶形因子全部运算结束,将最后结果输出到下一级. 在图 2 中, $X_m[p]$ 与 $X_{m-1}[p]$ 且 $X_m[q]$ 与 $X_{m-1}[q]$ 都占用相同的存储空间,依次重复运算,直到第 12 级运算结束,RAM 中得到的就是最后计算结果. 同址运算最大的缺点就是只有等 12 级蝶形因子全部算完以后,才能计算下一帧的数据,而且整个 FFT 模块始终只有一级蝶形因子在做计算,势必造成计算效率不高,计算速度比较低,该方法不能满足对实时处理速度要求比较高的系统. 为了提高计算速度,本文提出了基于流水线形式的 FFT 计算方法.

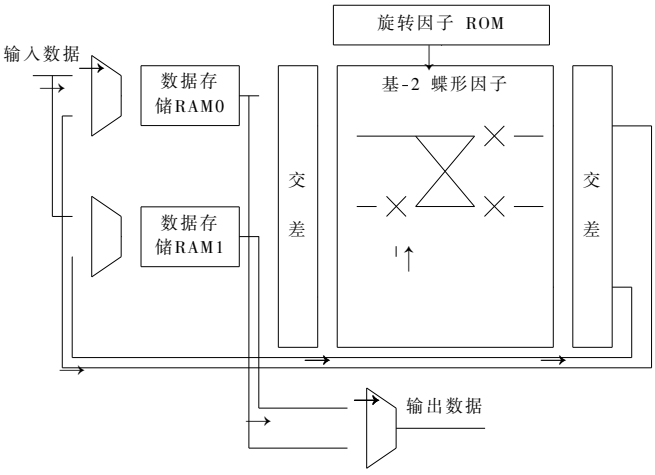

图 1 基-2 FFT 流程图

Fig. 1 The flow chart of Radix-2 FFT

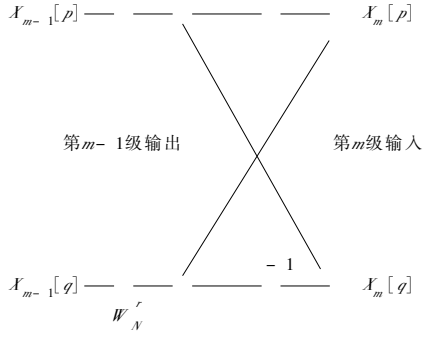

图 2 公式(6)的流程图

Fig. 2 The flow chart of equation (6)

3 基于流水线的 FFT 方法

基于图 1 的基-2FFT 流图,本文提出了一种改进的 FFT 方法. 如图 3 所示,相比于图 1,改进了同址计算方法,在第 $m-1$ 级计算结束之后,将计算结果直接存入下一级的输入端 RAM 内,同时第 $m-2$ 级的输出结果存入到第 $m-1$ 级输入 RAM 中,依此类推,数据就可以连续不断地输入,12 级蝶形因子不间断地同时进行计算,从而实现流水线操作. 该方法最大限度地缩短了每级之间的延时,保证系统的无缝隙流水线操作.

同时,增加的伺服控制部分保证各级蝶形因子之间不会出现数据的读写地址重叠.

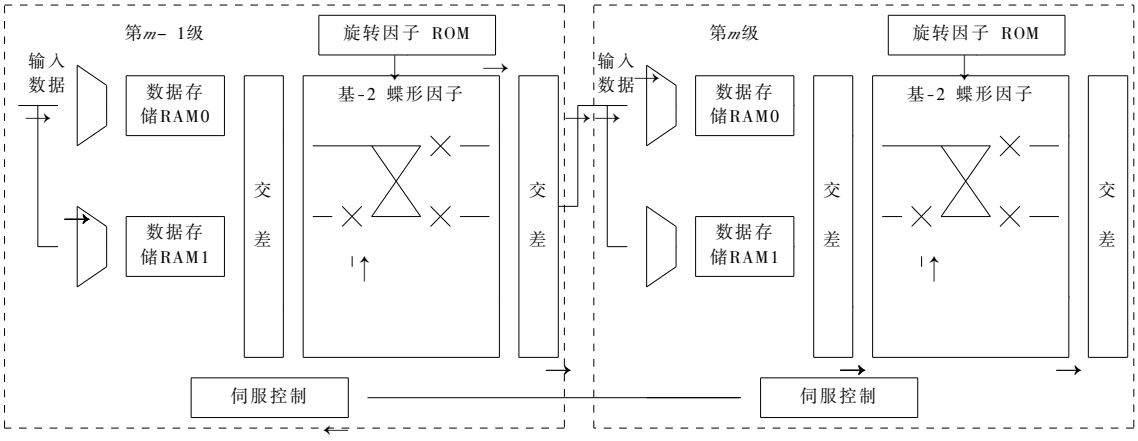

图 3 基于流水线的 FFT 流程图

Fig. 3 The flow chart of pipelined stream FFT

3.1　伺服控制

伺服控制部分用来进行RAM的读写控制以及每一级蝶形因子计算开始与结束的控制. 伺服控制流程如图4所示. 增加的伺服控制单元是为了保证12级蝶形因子能够连续不断地进行计算,避免同一级的RAM读写地址重合,造成数据丢失.

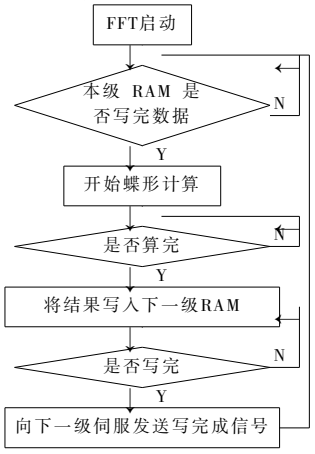

图 4　伺服控制流程图

Fig. 4　The flow chart of servo control

3.2　误差分析

FFT算法的误差分析对于FFT硬件处理器的设计非常重要,利用误差可大致估计出所需的运算字长和变换点数,并可得到较准确的FFT输出结果精度和信噪比. 在CDMA捕获系统中,捕获的标准是根据峰值与均值的比值来确定是否捕获成功,阈值通常设在30 dB以上. 根据参考文献[6]中提出的截断型定点FFT噪信比公式

$$R_{NS} = Q^2\left[\frac{23}{24}N^2 - \frac{31}{8}N + \frac{14}{3}\right], \tag{7}$$

式中:N为FFT的点数;Q为量化步长(总位宽减1位符号位). 由公式(7)可以得到,要保证信噪比在30 dB以上,数据位宽必须大于等于14位,为了计算方便取位宽为16位. 得到$R_{NS} = 2^{-6}$,满足系统要求.

4　仿真与实验验证

4.1　Matlab仿真

Matlab仿真的目的主要是为了验证硬件FFT计算的结果是否正确,因此利用matlab构造了一组线性累加的数据,使用matlab自带的fft函数进行了计算. 为了方便同硬件实验数据对齐,数据的起始点不是从1开始,matlab程序如下:

```
for i = 1:4096
int 16 a(i) = i;
int 16 b(i) = 1 + a(i);
end
int 16 c = complex(a,b)
int 16 d = fft(c,4096)
```

d是最后结果. 计算结果如表1所示.

表 1　FFT的matlab仿真结果

Tab. 1　The simulation results of matlab

1 024+ 1 024i	- 326+ 325i	- 163+ 163i	- 109+ 108i	- 82+ 81i	- 65+ 6 e5i	- 5 e5+ 54i	- 47+ 46i
- 41+ 41i	- 37+ 36i	- 33+ 32i	- 30+ 29i	- 27+ 27i	- 25+ 25i	- 24+ 23i	- 22+ 22i
- 21+ 20i	- 19+ 19i	- 18+ 18i	- 17+ 17i	- 17+ 16i	- 16+ 15i	- 15+ 15i	- 14+ 14i
- 14+ 13i	- 13+ 13i	- 13+ 12i	- 12+ 12i	- 12+ 11i	- 12+ 11i	- 11+ 11i	- 11+ 10i

表1中列出了计算结果的前面32个数. 由于matlab中采用浮点运算,运算结果精度相对高一些,因此结果也没有进行截断,之所以将最终结果除以8 192,也是为了与硬件仿真一致,因为硬件仿真进行了13位的位宽截断,以使最终输出为16 bit位宽,达到数据位宽一致.

4.2　FFT在ISE中的行为仿真

仿真是数字系统设计中不可或缺的一个环节,通过仿真可以对整个设计的功能和时序进行验证. 程序设计完成后,可以用ISE中自带的Simulator工具进行行为仿真,查看仿真时序波形与设计是否一致,以便进行优化. 图5所示为FFT的仿真结果,仿真的时钟采用50 MHz. 与matlab的计算结果相比较,可以看出仿真结果与matlab计算的结果基本一致. 存在的误差是由于位宽截断时导致的,但是误差在允许范围之内,可以根据对结果的精度要求,对位宽进行调整,增加输出位宽,从而增加计算结果的精度. 图5和图6是对传统的基2-FFT、本文提出的基于流水线的FFT方法以及matlab计算结果的对比,如表2所示.

表 2 三种计算方法计算结果的对比

Tab. 2 Comparing the results of those three methods

序号	1	2	3	4	5	6	7	8
matlab	1 024+ 1 024i	- 326+ 325i	- 163+ 163i	- 109+ 108i	- 82+ 81i	- 65+ 6e5i	- 55+ 54i	- 47+ 46i
基-2	1 026+ 1 026i	- 330+ 323i	- 167+ 161i	- 112+ 106i	- 85+ 80i	- 69+ 63i	- 58+ 52i	- 50+ 45i
流水线	1 025+ 1 026i	- 328+ 324i	- 165+ 161i	- 111+ 106i	- 83+ 79i	- 67+ 63i	- 56+ 52i	- 49+ 44i
序号	9	10	11	12	13	14	15	16
matlab	- 41+ 41i	- 37+ 36i	- 33+ 32i	- 30+ 29i	- 27+ 27i	- 25+ 25i	- 24+ 23i	- 22+ 22i
基-2	- 44+ 39i	- 40+ 35i	- 36+ 31i	- 33+ 27i	- 31+ 25i	- 26+ 24i	- 25+ 24i	- 23+ 22i
流水线	- 43+ 39i	- 38+ 34i	- 34+ 31i	- 32+ 28i	- 29+ 25i	- 27+ 23i	- 25+ 21i	- 24+ 22i

在图 5 和图 6 中,“xk _re”和“xk _im”分别为计算结果的实部与虚部,“dv”为数据有效标志,可以看出这两种方法的仿真结果与表 1 给出的结果具有高度的一致性,进而验证了本文提出方法的正确性. 图 7 中的三条标记线从左至右依次为: 开始计算的时间、基于流水线 FFT 方法的结束时间、基 2-FFT 计算结束的时间. 可以明显地看出,基于流水线 FFT 的时间只有基 2-FFT 的 2/3.

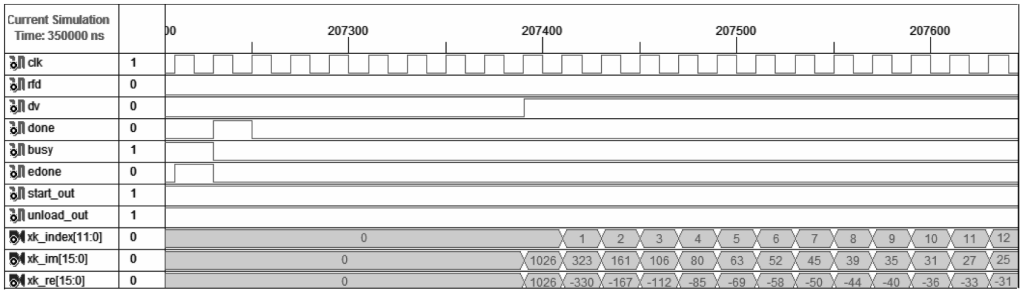

图 5 传统基-2 FFT 的仿真结果

Fig. 5 Te simulation results of radix-2 FFT

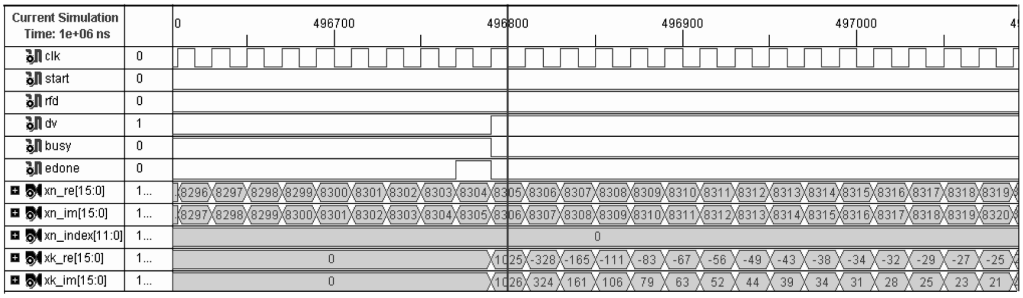

图 6 基于流水线 FFT 的仿真结果

Fig. 6 The simulation results of pipelined stream FFT

4.3 FFT 的硬件实验

ChipScope Pro 是 XILINX 推出的集逻辑分析、总线分析和虚拟输入输出为一体的虚拟分析和调试软件. 通过它可以对 FPGA 进行实时地分析和调试而无需额外的逻辑分析设备,只需将一根 JTAG 借口的下载电缆链接到要调试的 FPGA 器件. ChipScope 的工作原理为:通过在 FPGA 设计中插入 ICON(控制核)和 ILA(逻辑分析核),把需要分析的 FPGA 内部信号的采样点写入 FPGA 内部的 BLOCKRAM 中,软件将 JTAG 接口和 FPGA 相连,读出 BLOCKRAM 中的值并把信号的波形实时显示出来,从而实现逻辑分析的功能.

文中采用的是 4096 点的 FFT 运算,将编译综合之后生成的 bit 文件下载到 FPGA 中. 传统的基-2

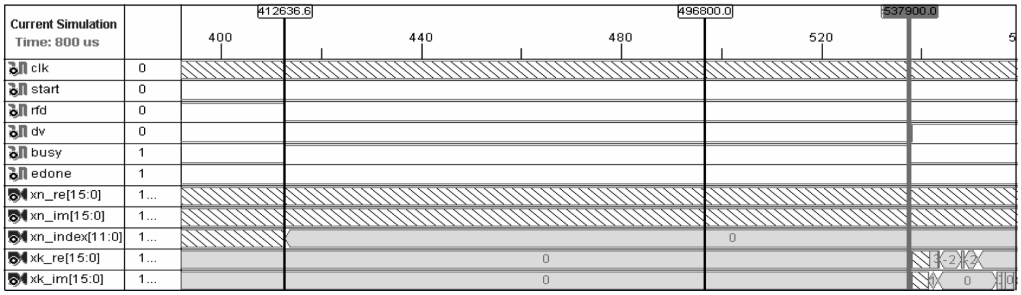

图 7　两种方法仿真时间的对比

Fig. 7　The comparing of Radix-2 FFT and pipelined stream FFT

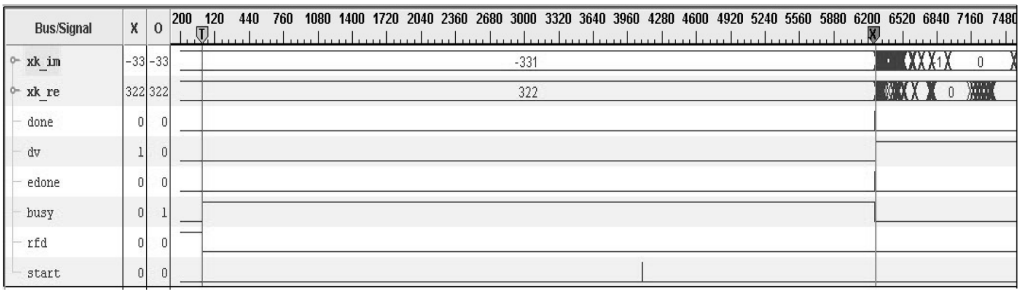

图 8　传统基-2 FFT 的计算时间

Fig. 8　The computation time of Radix-2 FFT

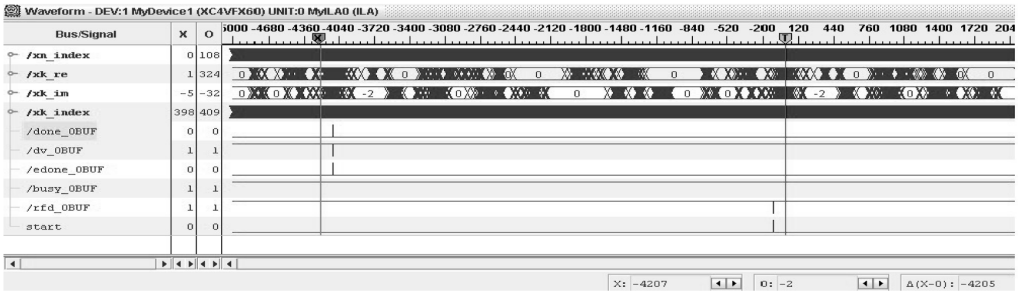

图 9　基于流水线 FFT 的计算时间

Fig. 9　The computation time of pipelined stream FFT

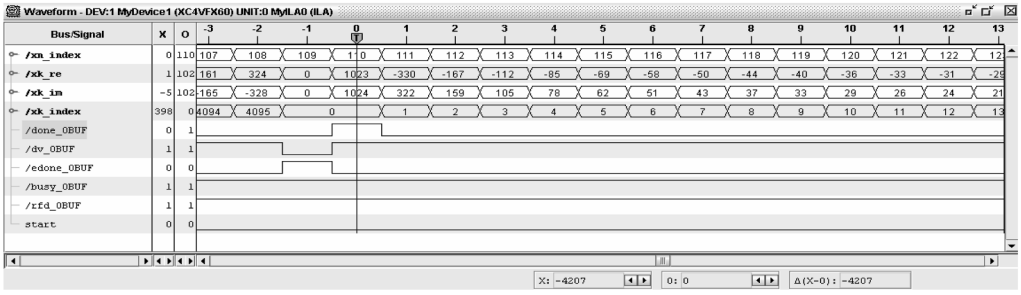

图 10　基于流水线 FFT 的计算结果

Fig. 10　The computation results of pipelined stream FFT

FFT 方法与基于流水线的 FFT 方法的计算时间分别如图 8 和图 9 所示. 可以看到基-2 FFT 方法的整个运算结果持续 6271 个时钟周期,而 FPGA 采用的时钟为 100 MHz,整个 FFT 过程耗时为 62.71 μs.

基于流水线的 FFT 方法的计算时间为 42.05 μs,可见本文提出的基于流水线的 FFT 方法要比传统的基-2 FFT 算法快 1/3 的时间,而相比于参考文献 [7] 中的方法,本文采用的 4096 点的 FFT 所用的时间只有文献 [7] 中 1024 点 FFT 时间的 1/3.

图 10 给出了基于流水线 FFT 的计算结果,"xn _index" 表示输入数据的编号,"xk _index"表示输出数据的编号,对比表 1 和图 10 的计算结果可知:基于流水线 FFT 的计算结果非常准确,与两种仿真方法得到的结果一致. 在图 9 中"O"指针与"T"触发指针重合,因为触发采用的"dv"信号的上升沿. 从图 7 右下角的 X-O 的距离也可以看出整个过程占用 4205 个时钟周期,相当于 42.05 μs 的时间.

5 结 论

本文在分析快速傅里叶变换 (FFT) 算法的基础上,提出并实现了一种基于流水线的 FFT 实现方法,并对该算法进行了 matlab 仿真以及 ISE 的功能仿真,最后在 FPGA 中进行了实验验证. 实验结果证明,基于流水线的 FFT 方法相比于传统的方法节约了至少 1/3 的计算时间,为高速数字信号处理提供了一种很好的时频转换方法.

参考文献:

[1] Cooley J W, Turkey J W. An algorithm for the machine calculation of complex fourier series[J]. Math Computer, 1965, 19: 297-301.

[2] Udo A, Udo Z, Stefan R. FFT based disparity estimation for stereo image coding[C]. Proceedings of the IEEE International Conference on Image, 2003,1: 14-17.

[3] Noll P. MPEG digital audio coding[J]. IEEE Transactions on Signal Processing Magazine. 1997, 14(5): 59-81.

[4] 甘秋歌. 快速傅里叶变换 (FFT) 的另一种推导方法及实现[J]. 西南民族大学学报(自然科学版), 2007, 33(2): 275-279.
Gan Qiuge. A new deduction and realization of Fast Fourier Transformation (FFT)[J]. Journal of Southwest University for Nationalities(Natural Science Edition), 2007, 33(2): 275-279. (in Chinese)

[5] Xilinx. Fast Fourier Transform v4.1[Z]. USA, 2007.

[6] 贾玉臣, 吴嗣亮. 快速傅里叶变换的误差分析[J]. 北京理工大学学报, 2005, 25(8): 739-742.
Jia Yuchen, Wu Siliang. Error analysis of fast fourier transform[J]. Transactions of Beijing Institute of Technology, 2005, 25(8): 739-742. (in Chinese)

[7] 袁浩浩, 张记龙, 徐振峰. 激光告警机中波长检测系统的 FPGA 实现[J]. 微计算机信息, 2007, 23(5): 193-195.
Yuan Haohao, Zhang Jilong, Xu Zhenfeng. The accomplishment of wavelength detection in the laser warning based on FPGA[J]. Information of Microcomputer, 2007, 23(5): 193-195. (in Chinese)