DOI:10.13274/j.cnki.hdzj.2007.01.020
信息技术 Information Technology 2007年第1期
中图分类号:TN402 文献标识码:A 文章编号:1009-2552(2007)01-0064-04

16点基4-FFT芯片设计技术研究

晓磊,朱 恩,赵 梅

(东南大学射频与光电集成电路研究所,南京 210096)

摘 要:FFT算法是高速实时信号处理的关键算法之一,在很多领域有广泛应用。文中采用了基-4,按时间抽取FFT算法,完成了16点,32bit位长,定点复数FFT的设计。基-4蝶形单元中采用32位Booth算法乘法器,并使用3级流水线设计,并行的处理四路输入数据,极大地提高了FFT的处理速度。本设计划分为多个功能模块,全部采用Verilog HDL语言描述,并且通过仿真验证。

关键词:FFT;基-4蝶形运算单元;流水线;Verilog HDL

Technology research of a 16-point, radix-4 FFT chip design

DING Xiao-lei, ZHU En, ZHAO Mei

(Institute of RF & OE-Ics, Southeast University, Nanjing 210096, China)

Abstract: The Fast Fourier Transform(FFT) is one of the most important algorithms of high real-time signal processing. In this article, the processor is designed for 16 samples, 32bits, fixed, complex FFT. The radix-4 butterfly unit adopts the 32bits multiplier of Booth algorithm. The 3 levels pipeline structure and 4 parallele data are uesed in the FFT processor. The main purpose of using these techniques is get better performance by balancing the speed and the power consume. This design has some function blocks based on Verilog HDL, and it proves the verity.

Key words: FFT; radix-4 butterflyunit; pipeline; Verilog HDL

0 引言

FFT作为时域和频域转化的基本运算,是数字谱分析的必要前提。FFT方法是信号处理领域核心的算法之一,广泛应用于雷达处理、观测,定时定位处理,数字通信、无线通讯、声纳、图像处理等领域。FFT的实现方法分软件和硬件实现两类,高速实时FFT的实现以硬件实现方法为主,具体又分为ASIC方法、FPGA方法、DSP方法和通用处理机方法(GPP)等。从处理速度、芯片面积和功耗等几方面考虑,ASIC方法性能最好,FPGA方法次之。

为了快速计算DFT,提出了Cooley-Tukey算法,常见的算法有基-2,基-4,分裂基算法等。在硬件实现中,需要考虑的不仅仅是算法运算量,更重要的是算法的复杂性、规整性和模块化。控制简单、实现规整的算法在硬件系统实现中要优于仅仅是在运算量上占优的算法。分裂基算法综合了基-2算法、基-4算法的优点,但L型碟式运算结构在控制上要复杂。基-4算法与基-2算法相比,节省约25%乘法次数,并且可以使存储器和运算部件间的必要通讯减少一半,并行性提高4倍,同样时钟频率下的运算速度是基-2 FFT的2倍。从算法的角度,基-4 FFT比基-2 FFT结构更能实现低功耗。综合考虑,本文采用基-4 FFT算法。

本文以16点FFT处理器的硬件设计为切入点,运用并行结构设计,在低功耗和速度方面做了考虑。目的是今后扩展为大点数,高性能的FFT处理器。

1 FFT算法分析

离散傅里叶变化(DFT)将时域信息转换成频域信息。长度为N的序列$x(n)$的离散傅里叶变换可

收稿日期:2006-09-04

作者简介:丁晓磊(1982-),女,东南大学硕士研究生,2004年毕业于南京航空航天大学,主要研究方向为数字IC设计。

写为：

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad 0 \leqslant n \leqslant N-1 \qquad (1)$$

基－4算法，要求 $N = 4l$（l 为正整数）。输入序列 $x(n)$ 分解为如下4个子序列：

$$x_m(i) = x(4i+m)$$

子序列 $x_m(i)$ 均为 $N/4$ 点序列，设它们的 $N/4$ 点DFT为 $X_m(r)$，则基－4时分蝶式运算公式如下：

$$X(k) = \sum_{m=0}^{N/4-1} x(4m) W_N^{4mk} + \sum_{m=0}^{N/4-1} x(4m+1) W_N^{(4m+1)k} + \sum_{m=0}^{N/4-1} x(4m+2) W_N^{(4m+2)k} + \sum_{m=0}^{N/4-1} x(4m+3) W_N^{(4m+3)k} \qquad (2)$$

其中：$m = 0, 1, 2, \cdots\cdots, N/4-1$；$k = 0, 1, 2, \cdots\cdots, N-1$

又令：

$$A = \sum_{m=0}^{N/4-1} x(4m) W_N^{4mk},\ B = \sum_{m=0}^{N/4-1} x(4m+1) W_N^{4mk},$$

$$C = \sum_{m=0}^{N/4-1} x(4m+2) W_N^{4mk},\ D = \sum_{m=0}^{N/4-1} x(4m+3) W_N^{4mk}$$

令：$W_N^k = W^P$，则：

$$\begin{cases} X(k) = A + BW^P + CW^{2P} + DW^{3P} \\ X(k+N/4) = A - jBW^P - CW^{2P} + jDW^{3P} \\ X(k+2N/4) = A - BW^P + CW^{2P} - DW^{3P} \\ X(k+3N/4) = A + jBW^P - CW^{2P} - jDW^{3P} \end{cases} \qquad (3)$$

再令 $A' = X(k)$，$B' = X(k+N/4)$，$C' = X(k+2N/4)$，$D' = X(k+3N/4)$，则公式可以进一步化简为：

$$\begin{cases} A' = (A + CW^{2P}) + (BW^P + DW^{3P}) \\ B' = (A - CW^{2P}) - j(BW^P - DW^{3P}) \\ C' = (A + CW^{2P}) - (BW^P + DW^{3P}) \\ D' = (A - CW^{2P}) + j(BW^P - DW^{3P}) \end{cases} \qquad (4)$$

16点基－4 FFT需要两级运算，每级有4个蝶形单元。为了做基－4时分FFT运算，输入序列必须做逆序排列。

16点基－4时分FFT，考虑了逆序重排的分步运算流程图如图1所示。

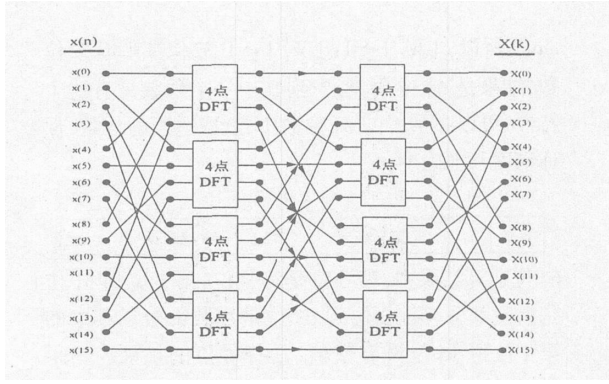


图1　N=16基－4时分FFT运算流程图

2　FFT处理器的硬件实现

2.1　系统总体结构

为了达到高效率和低功耗的要求，综合考虑，本文采用了两块RAM交替使用的工作方式，关键部件采用流水线和并行。整个FFT处理器的系统结构图如图2所示。包括一个控制器，控制整个电路的时序工作，保证系统各部分之间的有机联系与协调通讯；一个地址产生器，在控制器的控制下，产生蝶形运算器所需的各个地址，保证运算器按照时序要求处理数据；一个基－4蝶形运算器；一个输入寄存器，一个输出寄存器，保证每次基－4 FFT运算时，输入输出数据能同时到达运算器；一个存储旋转因子的ROM；一个RAM，其中一组RAM与外部的输入输出接口通讯，另一组RAM用来处理内部数据。

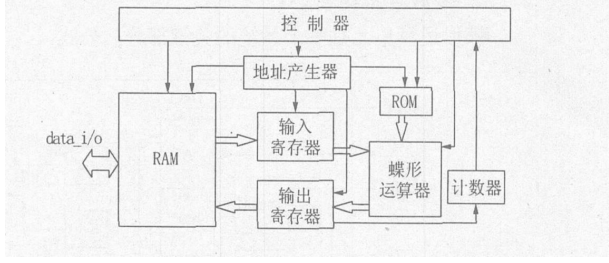


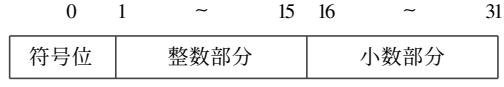
图2　FFT处理器的系统结构图

2.2　基－4 FFT蝶形运算单元

2.2.1　定点数据的处理和Booth乘法器设计

为满足数据的高精度要求，本文采用了32位定点数据，数据处理时，使用补码形式。

32位定点数据表示如下：



待处理的数据和旋转因子均用复数形式表示，存储时，采用RAM和ROM中的存储单元均为64位，前32位是复数的实部，后32位是复数的虚部。

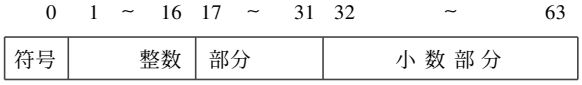
处理的数据范围是：$-(2^{15}-1) \sim (2^{15}-1)$，精确度可以到 2^{-15}。

以公式(3)中 BW^P 为例，令 B 的实部和虚部分别为 B_r、B_i，令 W^P 的实部和虚部分别为 W_r^P、W_i^P。则：

$$BW^P = (B_r + jB_i)(W_r^P + jW_i^P) = (B_r \cdot W_r^P - B_i \cdot W_i^P) + j(B_i \cdot W_r^P + B_r W_i^P)$$

由于式中，W_r^P 的取值是 $\pm\cos\alpha$，W_i^P 的取值是 $\pm$

$\sin\alpha$, 所以, $|W_r^P| \leq 1$, $|W_i^P| \leq 1$。32 位数乘以 32 位数, 结果是 64 位数。考虑到其中一个乘数具有这个特性, 可以在乘得的 64 位数中, 选取适当的位数, 保证精度和准确度。



64 位运算结果中, 第 0 位是符号位, 第 1 位到 31 位是整数部分, 第 32 位到 63 位是小数部分。其中的第 1 位到 16 位没有数值, 是符号位的扩展。结果中, 取符号位, 整数部分的第 17 到 31 位, 和小数部分的前 16 位, 构成乘法结果的 32 位数。这样, 保证了精度和准确度。

由公式(3)可知, 完成一次基－4 蝶形运算, 需要 3 次乘法运算和 8 次加法运算。所以考虑使用 Booth 算法时序乘法器。Booth 算法采用补码表示正数和负数, 便于后面的加法运算。本文采用的 Booth 算法时序乘法器的结构图如图 3 所示。

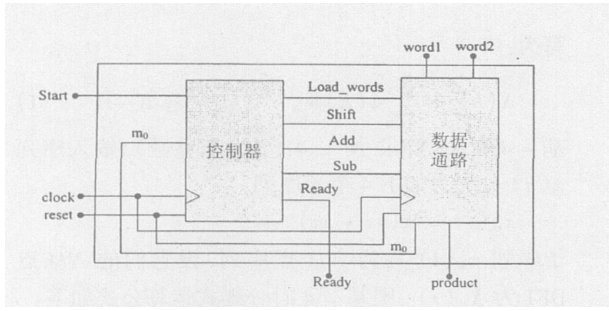


图 3 Booth 乘法器结构

2.2.2 蝶形运算单元设计

蝶形运算单元是整个系统的关键部分。公式(4)是一个基－4 蝶形运算的原理。依据这个原理, 本文采用并行流水线结构, 实现基－4 蝶形运算单元。硬件结构如图 4 所示。由输入寄存器提供 A, B, C, D, W^P, W^{2P}, W^{3P}, 当 7 个数据都在输入存储器中时, 给一个 Start 信号。蝶形运算器开始工作。根据结构, 做了 3 级流水, 第一级时, 3 个复数乘法器并行, 提高预算速度, 运算结构存放在第一组寄存器中。第二级, 第三级如图 4 所示。整个蝶形运算的时间即为第一级复数运算的时间。流水线的使用, 大大节省运算时间。

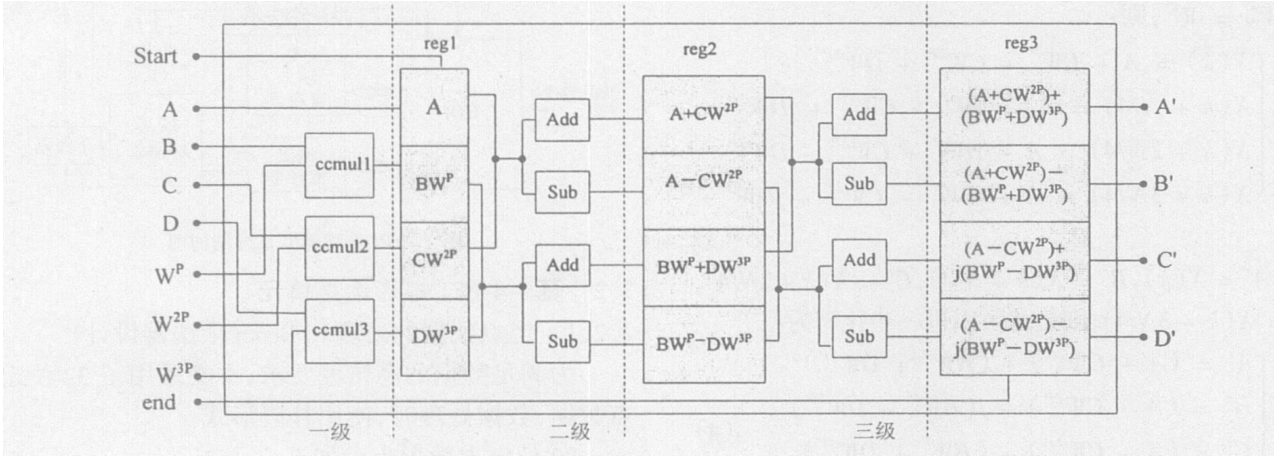


图 4 蝶形单元处理器结构

2.3 地址产生器设计

控制和时序是系统设计的难点, 本文采用的数据存取方式是, 每级反序输入, 正序输出。使得每级迭代时数据存取规律都相同, 降低控制的复杂性, 简化硬件设计。

N 点 FFT 需要 $\log_4 N$ 级运算, 每级有 $N/4$ 次基 4 运算。在进行第 k 次基－4 运算时, 应该在 N 个数据中取出第 k 个、第 $k+N/4$ 个、第 $k+N/2$、第 $k+3N/4$, 这四个数据运算以后结果存放在第 $4k+1$ 个、第 $4k+2$ 个、第 $4k+3$ 个、第 $4k+4$ 个。各级运算都按照这个规律存取数据。本文设计中, 采用两组 RAM, 一组用于读数据, 一组用于写数据, 两组交替使用。硬件实现上, 用 4 个 DPRAM, 产生 RAM 的读地址。16 点的地址可以用 4bit 数据表示。

每一次的四个数据的地址, 就是把除了前两位的尾数一直累加, 按照这个地址到 RAM 中去读数据。第二级也按照这个规律寻址和存储, 如图 1 所示, 第二级的运算结果, 也按照这个寻址方式存储, 即为顺序的输出结果。具体实现如表 1 所示。

表 1 RAM 寻址方式

第一次	第二次	第三次	第四次
00 00 // 0	00 01 // 1	00 10 // 2	00 11 // 3
01 00 // 4	01 01 // 5	01 10 // 6	01 11 // 7
10 00 // 8	10 01 // 9	10 10 // 10	10 11 // 11
11 00 // 12	11 01 // 13	11 10 // 14	11 11 // 15

2.4 控制器设计

时序设计是整个系统设计的关键也是难点,因此控制器是本设计的核心之一。本文中控制器采用有限状态机实现。控制器以时钟节拍向受控电路发出有序的控制信号,控制系统中各个模块按一定的时序执行逻辑操作。

16 点基－4 FFT,本文的设计系统中需要,2 级运算,再加一级同样寻址方式的数据调整,每级运算需要 4 个循环。具体的控制过程如下:16 个时钟周期结束,第一组 RAM1 即读满 16 组数据。此时,控制器控制地址产生器产生地址去读 RAM1 中的数据和 ROM 中的数据,送入蝶形运算器中进行运算,然后再写入 RAM2 中,这样就完成一个 Cycle。完整的做一次 16 点的基－4 FFT 运算,需要经过如表 2 所示的 12 个 Cycle。前 8 个 Cycle,是读数据,做运算,存数据的过程,后四个 Cycle,是将数据按照输出顺序存放到 RAM2 中。整个设计过程,控制器是 12 个状态,由时序和握手信号进行 12 个状态间的转换,完成时序控制。

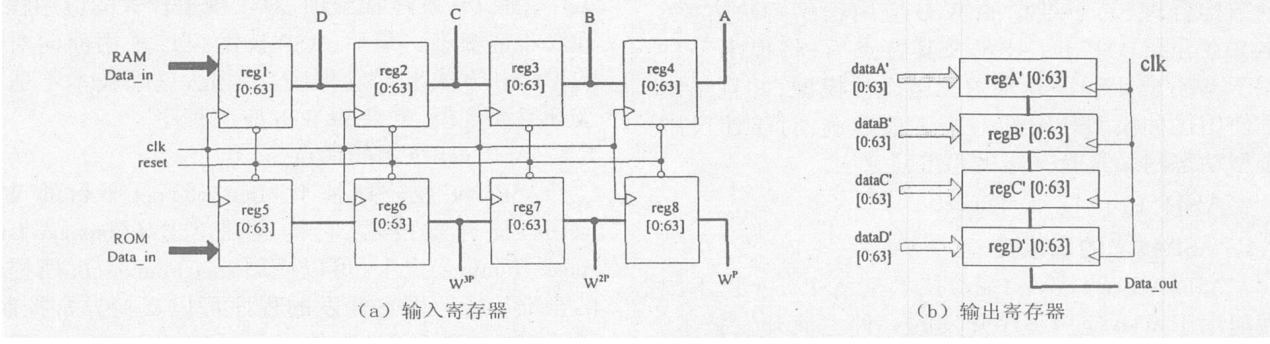
表2 时序过程和状态图

		读数据	运 算	写数据
第一级	Cycle1	从 RAM1[0:16] 中的 0,4,8,12 读	一次蝶形运算	写入 RAM2[0:16] 中的 0,1,2,3
	Cycle2	从 RAM1[0:16] 中的 1,5,9,13 读	一次蝶形运算	写入 RAM2[0:16] 中的 4,5,6,7
	Cycle3	从 RAM1[0:16] 中的 2,6,10,14 读	一次蝶形运算	写入 RAM2[0:16] 中的 8,9,10,11
	Cycle4	从 RAM1[0:16] 中的 3,7,11,15 读	一次蝶形运算	写入 RAM2[0:16] 中的 12,13,14,15
第二级	Cycle5	从 RAM2[0:16] 中的 0,4,8,12 读	一次蝶形运算	写入 RAM1[0:16] 中的 0,1,2,3
	Cycle6	从 RAM2[0:16] 中的 1,5,9,13 读	一次蝶形运算	写入 RAM1[0:16] 中的 4,5,6,7
	Cycle7	从 RAM2[0:16] 中的 2,6,10,14 读	一次蝶形运算	写入 RAM1[0:16] 中的 8,9,10,11
	Cycle8	从 RAM2[0:16] 中的 3,7,11,15 读	一次蝶形运算	写入 RAM1[0:16] 中的 12,13,14,15
第三级	Cycle9	从 RAM1[0:16] 中的 0,4,8,12 读		写入 RAM2[0:16] 中的 0,1,2,3
	Cycle10	从 RAM1[0:16] 中的 1,5,9,13 读		写入 RAM2[0:16] 中的 4,5,6,7
	Cycle11	从 RAM1[0:16] 中的 2,6,10,14 读		写入 RAM2[0:16] 中的 8,9,10,11
	Cycle12	从 RAM1[0:16] 中的 3,7,11,15 读		写入 RAM2[0:16] 中的 12,13,14,15

2.5 输入、输出寄存器设计

控制器和地址产生器共同工作,每一个时钟周期从存储器中按照寻址规律读取一个数值,为了保证蝶形运算单元所需的 7 个数据 $A, B, C, D, W^{P}, W^{2P}, W^{3P}$ 同时到达,故设计了如图 5(a) 所示的输入寄存器结构。

同理,FFT 蝶形单元同时输出 4 个数据 A', B', C', D'。为了保证这四个数据按照顺序写到输出寄存器中,设计如图 5(b) 所示的输出寄存器结构。



(a) 输入寄存器　　(b) 输出寄存器

图5 输入输出寄存器结构

3 性能分析与结果

本设计全部由 Verilog 语言编写实现,关键部件采用流水线设计,并且两组 RAM 交替使用,加快了系统的运算速度。经 modelsim 验证,时序正确。图 6 是在 modelsim 里的部分仿真结果波形。并且数据结果通过 matlab 验证,逻辑正确。本文的系统设计方法行之有效。

(下转第 71 页)

业务对象在建立用户界面层应用程序中，应首先在页面中导入自己定义的名称空间，然后建立自定义类的对象并且使用该对象的属性和方法完成与业务对象的交互，在该程序页面中需要导入 BLL 和 DataAccess 类库空间。

在程序中利用构造函数将数据库连接字符串传到业务逻辑层的控件中，同时调用自定义类的 DataInsert 方法完成任务相应的转换任务。请注意，在这个用户程序中完全没有数据库连接控件，它与数据库的连接任务是通过业务逻辑层来完成的，而实际的业务逻辑层，是调用 DataAccess 中相应类来完成操作数据库操作。这样，程序的结构更加清晰，这也是三层结构应用程序的主要特点之一。

4 结束语

以上对如何建立三层体系结构的 ASP. net 应用程序的理论和实践进行了一番探讨，同时在上面的实际例子中，笔者使用了面向对象的程序设计方法设计了业务逻辑层中的自定义类及在用户程序中使用该类的对象完成了任务，这种设计方法及理念在三层结构的程序设计中经常用到。此外，在例子中笔者使用的 ASP. net 技术，实际的三层结构的应用程序可以用 VB. net，C ♯，Java 等任何一门语言实现。

回顾工程的整个开发过程，三层设计模型在工程设计、开发、测试和维护各个阶段都起到了巨大的作用，而 ASP. net 技术的一些优秀特点对于优化工程的结构也是至关重要的。因此，基于 ASP. net 技术的 Web 应用程序三层设计模型是整个工程得以成功实现的重要基石。

当然，三层体系结构的概念远远不止于此，在优秀的分布式应用开发的过程，用到了面向对象的分析/设计/编程/测试、UML 建模、软件开发过程控制、并行开发、迭代增量开发等诸多先进技术与理念。VS 2005 的开发环境的推出，更是推动了 Web 设计的春天到来，新的 ASP. net 的技术更为完善，更利于程序分析实现，有待于进一步学习实践。

参 考 文 献：

[1] 陈运海. 基于 ASP. net 的电子商城管理系统的构建[J]. 武汉船舶职业技术学院学报，2005(6)：11－14.

[2] 石岩. 论我国电子商务发展战略[J]. 现代情报，2005(8)：38－40.

[3] 秦超，杨华生. ASP. net 项目软件协作开发模式探讨[J]. 大众科技，2006(1)：65－66.

[4] 曾诚. 高性能 ASP. net 应用程序的探讨与研究[J]. 湖北大学学报，2004(3)：19－22.

[5] 毛德祥，罗荣阁. 基于 ASP. net 技术的 Web 应用程序三层设计模型[J]. 微电脑应用，2002(3)：26－30.

[6] Jesse Liberty, Dan Hurwitz. Programming ASP. net[M]. 2004：46－78.

[7] Andrew S. Tanenbaum Computer Networks－Pearson[M]. 2003：23－46.

[8] 胡迎松，彭利文，池楚兵. 基于. net 的 Web 的应用三层结构设计技术[J]. 计算机工程，2003(5)：173－175.

[9] 胡杰. 浅谈如何建立三层体系结构的 ASP. net 应用程序[J]. 青岛职业技术学院学报，2005(12)：67－70.

责任编辑：张荣香

(上接第 67 页)

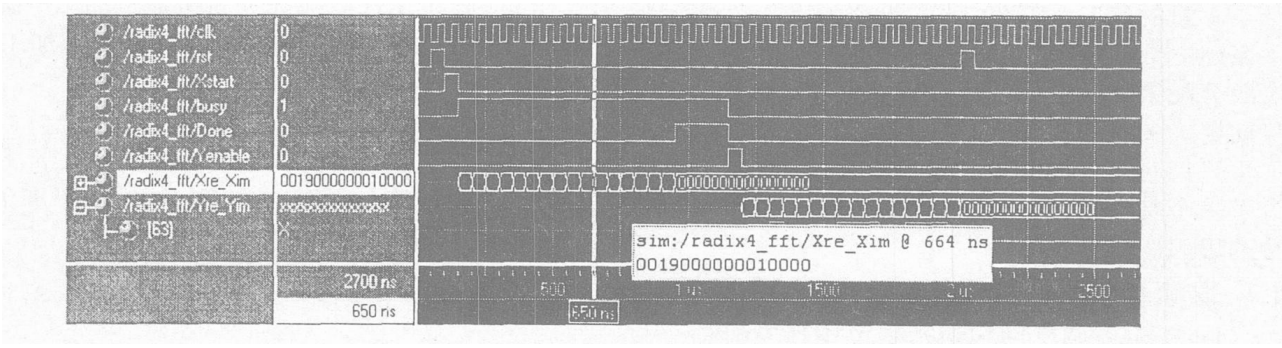


图6 部分波形图

参 考 文 献：

[1] 王志功，朱恩. VLSI 设计[M]. 北京：电子工业出版社，2005.

[2] Michael D Ciletti. Advanced Digital Design with the Verilog HDL [M]. 电子工业出版社，2005.

[3] Keshab K Parhi. VISI Digital Signal Processing Systems Design and Implementation[M]. 机械工业出版社，2004.

[4] 冷建华，李萍，王良红. 数字信号处理[M]. 北京：国防工业出版社，2002.

[5] Chang Yun－Nan, Parhi K K. Efficient FFT Implementation Using Digit－Serial Arithmetic[J]. Circuits and Systems Ⅱ：Analog and Digital Signal Processing, 2003, 50(6).

[6] 汤晓峰，戎蒙恬，等. OFDM 系统中傅里叶变换的硬件实现方法[J]. 计算机工程与应用，2005(25).

[7] Ray Ranjan Varghese, Chanjal G Tharayil. Project Report：Design, Simulation and Synthesis of an FFT Processor using VHDL. Mahatma Gandhi University, Kottayam, Kerala 2001.

[8] 韩泽耀，韩雁，郑为民. 一种高速实时定点 FFT 处理器的设计[J]. 电路与系统学报，2003，7(1).

责任编辑：肖 滨