第 22卷第 1期　　四川理工学院学报 (自然科学版)　　Vol. 22　No. 1
2009年 2月　　Journal of Sichuan University of Science & Engineering(Natural Science Edition)　　Feb. 2009
文章编号: 1673-1549(2009)01-0087-05

嵌入式微控制器 PicoBlaze的 PLD应用研究

古世甫[1], 游志宇[1], 杜　杨[2], 董秀成[1]
(1.西华大学电气信息学院, 成都　610039; 2.中国科学院光电技术研究所, 成都　610209)

摘　要: PicoBlaze是 Xilinx公司开发的嵌入式 8位微控制器 IP核。在简要介绍 PicoBlaze结构原理、性能和基本开发流程的基础上,重点论述了 PicoBlaze的 PLD实现方法,并在 MAX II570T144试验板上进行应用验证。同时对 PicoBlaze核存在的 Bug提出适当的解决方法。

关键词:微控制器; PicoBlaze; PLD; MAX II; VHDL
中图分类号: TN402　　**文献标识码**: A

引　言

长期以来,微控制器以其性价比高、体积小、功能灵活等方面的独特优点,被广泛应用在各领域。为了同时达到对处理速度和控制灵活性等方面的需求,设计嵌入式系统常采用微控制器和可编程逻辑器件 (PLD)共同搭建的方案。随着大规模可编程逻辑器件及 EDA技术的发展, IP核技术在 PLD中的应用,特别是 MCU IP核技术的发展,出现了性能不同的嵌入式 MCU IP软核。使得将 MCU、存储器和一些外围电路集成到同一个可编程器件上成为现代嵌入式系统设计的发展方向和趋势。形成一种特殊的基于芯片的可编程片上嵌入式系统解决方案。

PicoBlaze是 Xilinx公司设计开发的 8位微控制器软核,它是为 Virtex系列 FPGA、Spartan系列 FPGA和 CoolRunner-II系列 CPLD设计的嵌入式专用 MCU IP核。PicoBlaze是一个由 VHDL语言实现的完全嵌入式 MCU IP核,不需要预编译,可直接由布局布线工具嵌入到容量大一点的 PLD器件中,设计灵活方便,运行速度快,占用资源少。在实际应用中,虽然 PicoBlaze也能被用于数据处理,但它更适用于那些状态复杂而对时序要求稍低的设计。因此 PicoBlaze又被命名为常量编码可编程状态机 (KCPSM, (K) constant Coded Prograramable State Machine)。目前的 KCPSM3是经过优化和改进设计的最新版 PicoBlaze微控制器,它在前期版本 (PSM、KCPSM和 KCPSM2)的基础上增加了 8条新的指令。指令的增加不可避免要增加处理器的复杂度,在 FPGA实现占用资源方面 KCPSM3比 KCPSM增加了约 26%,比 KCPSM2增加了约 14%。这样虽然在设计和实现上存在一定的风险,但却很好地满足了编程的基本需求。改善了代码编写的效率,从而降低了对外围逻辑设计的需求。

Xilinx公司提供面向 FPGA (KCPSM、KCPSM2、KCPSM3)和面向 CPLD的两种类型 PicoBlaze MCU IP核。面向 FPGA的 PicoBlaze也是采用 VHDL语言实现,但其内部调用了大量 Xilinx的逻辑功能模块,不利于在其他公司的 PLD上进行综合和实现。而面向 CPLD的 PicoBlaze完全是采用 VHDL描述实现的,与工艺无关,可以在多种资源充足的 FPGA/CPLD上进行逻辑综合和实现。由于面向 FPGA的 PicoBlaze与面向 CPLD的 PicoBlaze逻辑结构基本相同,因此本文选择面向 CPLD的 PicoBlaze IP核来分析它的逻辑结构、指令系统,并在此基础上详细论述 PicoBlaze的 PLD实现方法。同时在 Altera的 PLD试验板上进行验证,对遇到的问题提出解决方法。

1　PicoBlaze的结构原理与指令集

嵌入式微控制器 PicoBlaze IP核提供 49条 16位指令, 8个全局寄存器, 256个直接或间接的可设定地址的端口, 1个可屏蔽的中断,使得 PicoBlaze IP核很好的解决了常量编码可编程状态机的问题。

1.1　PicoBlaze的结构原理

收稿日期: 2008-08-28
作者简介:古世甫 (1982-),男,四川广安人,助理实验师,要从事嵌入式系统开发、图象信号处理及应用方面的研究。

PicoBlaze IP核由全局寄存器、计算逻辑单元(ALU)、程序流控制标志和复位逻辑、输入/输出(I/O)、中断控制器等几大部分构成。其内部结构框图如图1所示。

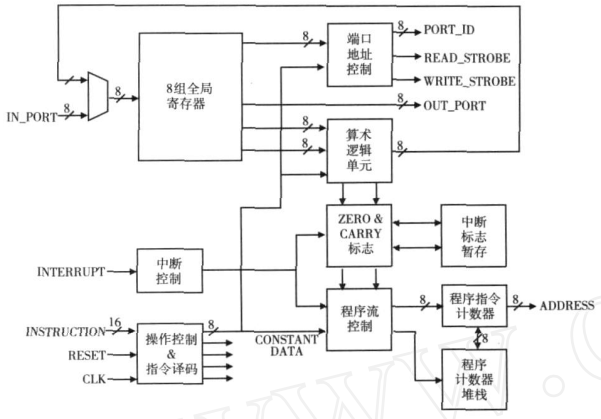

图1 PicoBlaze微控制器内部结构框图

全局寄存器:提供8个8位全局寄存器,s0～s7。寄存器的操作非常灵活的,没有为特殊任务保留寄存器,任何寄存器的优先权都是一样的。

算术逻辑单元(ALU):提供了8位处理器需要的所有简单操作。执行所有操作都是用任意一个寄存器提供的操作数完成。若操作需两个操作数,则由另一寄存器指定或在指令中嵌入一个8位常量值。在不增加程序大小的前提下,嵌入常量值操作数,增强了简单的指令特性。比如INCREMENT与ADD 1指令是等价的。若操作超过8位,则有一选项(增加或减少)可供选择。二进制操作码(LOAD、AND、OR、XOR)可操作和测试二进制数。

程序流控制标志:ALU操作后的结果影响ZERO和CARRY两个标记。用有条件或无条件的程序流控制指令决定程序执行的顺序。JUMP指令指定在程序空间内的绝对地址。CALL指令将程序定位到用一段代码写的子程序的绝对地址,同时将返回地址压栈。可以使用嵌套CALL指令。

复位逻辑:复位信号强迫程序回到初始状态,即程序从地址00开始执行,中断被屏蔽,状态标记和堆栈也同时复位,但全局寄存器中内容不受影响。

输入/输出(I/O):PicoBlaze提供256个输入端口和256个输出端口。由端口总线提供一个8位地址值与一个READ或WRITE选通脉冲信号一起指定访问的端口。这个端口地址值或为一确定值或由任意一寄存器中内容指定。当访问由分布式或块状RAM组成的内存时,最好用直接寻址。当进行输入操作时,输入端口上的值在输出一个READ_STROBE输出脉冲时,即表示进行了一次输入操作。

中断控制器:PicoBlaze提供一个中断输入信号。只要用一些简单的组合逻辑,多个信号就可进行组合并被应用于这一中断。程序中可定义此中断是否被屏蔽,默认值是中断被屏蔽。一个被激活的中断信号使能程序执行"CALL FF"指令(FF即255,程序存储器的最后一个位置),然后设计者为此定义的放在此处的一段服务程序被执行。一般在此地址放一条JUMP指令,跳转到中断服务程序处。中断进程屏蔽其它中断,RETURNI指令保证在中断程序结束后,标记和控制指令回到原先的状态。

PicoBlaze IP核结构简单,占用资源少,性能超过了传统独立元器件组成的微控制器,使得PicoBlaze在数据处理和控制算法领域有着广泛的应用前景。

1.2 PicoBlaze的指令集

PicoBlaze微控制器采用RISC指令集,每一条指令的执行需要两个时钟周期,其指令时序图如图2所示。它的指令集分为六类,分别是控制程序转移类指令、循环移位类指令、中断类指令、逻辑操作类指令、输入/输出类指令、算术运算类指令。

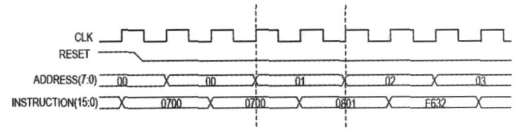

图2 PicoBlaze微控制器指令执行时序图

从指令系统可看出,PicoBlaze处理器是一种非常简单且紧凑的8位处理器IP核。与传统的8位MCU相比,它是一个由VHDL实现的软处理器IP核,可以在各种PLD上实现,设计灵活方便。因此PicoBlaze将会在基于可编程逻辑和MCU的应用领域中发挥积极作用。

2 PicoBlaze微控制器开发流程

PicoBlaze微控制器应用开发分为四个步骤:代码编辑、软件调试、程序编译和系统集成。

2.1 代码编辑

代码的编写可使用普通的文本编辑器完成。如:记事本(Notepad)或写字板(Wordpad)。值得注意的是,编写好的程序要保存为编译器支持的asm格式。即文件扩展名为.asm,并且文件名不可多于8个字符,不能用中文字符。

2.2 软件调试

处理器应用开发者常常使用指令集仿真器(ISS)进行诸如单步运行、连续运行、断点设置这样的手段来调试程序。Mediatronix免费提供的pBlazIDE正是这样的辅助工具。pBlazIDE是PicoBlaze汇编程序的编辑、调试集成环境。利用它可以方便地仿真所写程序的输入输

出、寄存器内值的变化及程序指令的顺序执行过程,便于检查程序的错误。此外,pBlazIDE还能够实现代码覆盖率、运行时间统计、基于端口功能的系统级模拟等功能。使用 pBlazIDE进行调试需要注意的是 pBlazIDE不能直接支持 PicoBlaze的指令集,必须用该软件的文件转换接口进行文件转换后才能进行调试。同时,pBlazIDE也不能实现与外围可编程逻辑的联合调试。但对于一般应用的代码调试 pBlazIDE完全能满足需求。

2.3　程序编译

PicoBlaze的汇编程序需要使用 Xilinx公司提供的汇编器 ASM.exe进行编译。将编辑好的汇编程序文件和编译器放置在同一个工作目录下。在命令行系统环境运行 ASM.exe程序,并输入编写好的汇编源程序 Filename.asm后,汇编编译系统就会生成设计需要的文件,具体的文件组织如图 3所示。其中 Filename.vhd是编译器自动生成的用户应用程序指令代码 VHDL文件,用来构成应用程序存储器 ROM模块。当编译器发现错误时会立刻停止编译程序,给出错误提示信息,设计者可根据提示修改代码、重新编译。

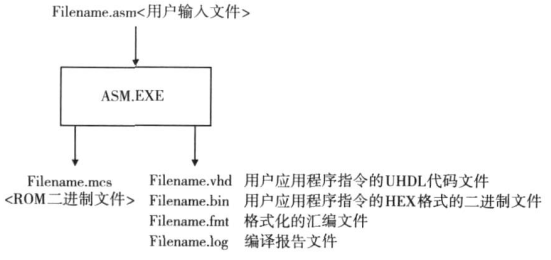

图 3　PicoBlaze汇编编译器执行处理过程

编译应用程序也可以采用批处理的方式:首先将编辑好的汇编程序文件和编译器放置在同一个工作目录下,并在该目录下建一个 Compile.bat的批处理文件,在批处理文件中输入 ASM.exe Filename.asm并保存。然后用鼠标双击 Compile.bat文件即可完成编译。

2.4　系统集成

在上层设计中调用编译好的 PicoBlaze模块(包括汇编编译器生成的程序存储器 ROM模块和 PicoBlaze MCU模块文件),并加入定制的外围逻辑接口就能构成整个系统。对于整个系统的仿真、综合、适配、下载与 FPGA/CPLD逻辑设计流程一致,此处就不再赘述。

3　PicoBlaze IP核的 PLD应用实现

本文以一个简单的流水灯试验为例子,详细论述 PicoBlaze微控制器 IP核的 PLD应用实现。PLD芯片选用 Altera公司的 MAXII570T144,所以我们可以利用 Altera公司的集成开发环境 Quartus II来完成实际应用系统构建。

3.1　PicoBlaze硬件系统构建

在 Quartus II中新建应用工程,选择目标器件,设置相关工程参数。

将 PicoBlaze IP Core的 VHDL源代码拷贝到工程目录下,并将源代码添加到工程中。

在 Quartus II中用 VHDL语言编写一个时钟分频模块,对目标试验板上的系统时钟进行分频,以满足 PicoBlaze的时钟要求。

在工程目录下新建一个应用程序文件夹,将汇编编译器拷贝到该目录下,同时在该文件夹下新建一个 Compile.bat批处理文件及一个空的汇编程序文件。注意汇编程序文件保存的后缀名为.asm。双击 Compile.bat批处理文件编译生成一个用户应用程序指令 VHDL代码文件,即程序存储器 ROM模块;然后将此 VHDL文件拷贝到 QuartusII工程目录下,并添加到工程中。注意此处生成的用户应用程序指令 VHDL代码文件是一个指令代码全为 0的程序存储器 ROM模块。

在 Quartus II中用 VHDL语言编写一个应用工程的顶层文件,并加入系统所需的外围逻辑接口。然后将系统中的各模块连接起来,编译工程。

本文设计一个流水灯试验,硬件系统包含四个模块:一个时钟分频模块、一个 ROM程序存储模块、一个 PicoBlaze处理器模块和一个输出端口模块。其编译后生成的 RTL视图如图 4所示。

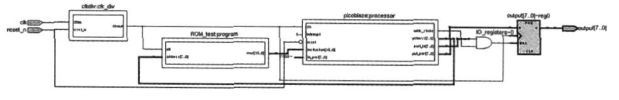

图 4　应用工程顶层 RTL图

3.2　PicoBlaze应用程序创建

在硬件系统模块建立起来后,就可以根据具体应用编写应用程序。在构建系统硬件模块时创建的应用程序文件夹下,利用文本编辑器编写流水灯汇编应用程序代码,并用 pBlazeIDE进行指令代码仿真、调试。在调试正确后双击 Complie.bat批处理文件编译生成新的 PicoBlaze应用程序指令 VHDL代码文件,即构成新的程序存储器 ROM模块。本流水灯试验的汇编代码如下所示:

```
        Constant shifter - port, 01;端口声明
        Namereg s7, shifter - reg;寄存器声明
start:  load shifter - reg, 01;寄存器初始化
Loop1:  Output shifter - reg, shifter - port;输出
        RR shifter - reg;右循环移位
        Jump loop1;无条件跳转
```

3.3　PicoBlaze实际应用系统实现

将汇编编译器生成的 PicoBlaze应用程序指令

VHDL代码文件拷贝到 Quartus II工程文件目录下,替代原创建 PicoBlaze硬件系统时使用的 VHDL代码文件。然后重新编译 Quartus II工程生成 MAXII570T144的配置文件. pof,将应用程序指令代码嵌入(固化)到 ROM中。最后利用 QuartusII的 Programmer将配置文件. pof下载到 MAXII570T144试验板上进行验证,实现 PicoBlaze的实际应用系统设计。

4 IP核 Bug解决及存储空间扩展

由于 PicoBlaze IP Core是开放的源代码, xilinx公司并不保证源代码的完全正确性,并不对程序中存在的小错误负任何责任,因此使用中需要特别小心。在上面的流水灯试验中,当把配置文件下载到 MAXII570T144试验板后,流水灯不能正常运行。说明 PicoBlaze IP核、应用程序及整个硬件系统模块中某个或几个存在问题,因此我们必须采取一定的方法和步骤来定位问题所在。

首先查看应用程序源代码,排除是源代码的问题。然后仔细分析硬件系统模块,也排除是硬件系统模块的问题。于是我们可以定位到问题出在 PicoBlaze IP核中。为了解决 IP核存在的问题,我们必须仔细分析其源代码,并借助仿真工具进行仿真。采用仿真工具对上面的试验进行仿真,其仿真波形图如图 5所示。

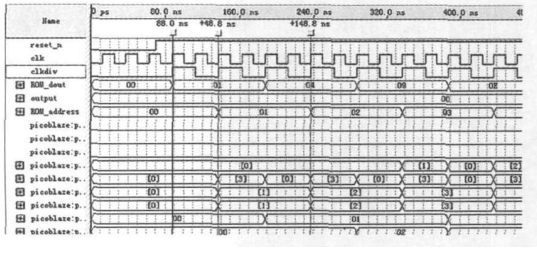

图 5 PicoBlaze IP核时序仿真波形图

从图中可以看出,系统复位后,在第一个时钟周期的上升沿程序存储器才输出其地址为 00处的第一条指令代码,而与此同时 PicoBlaze MCU也开始执行第一条指令代码了。此时的指令代码应为第一个时钟周期上升沿时刻前从 ROM中输出的指令代码,也即系统复位后程序存储器 ROM输出的值,此值应为全 0,而不是应用程序中的第一条指令代码的值。因此相当于 PicoBlaze MCU执行了 load s0,00指令,而不是执行程序中的 load shifter_reg,01指令。在第一条指令执行完后,寄存器 s7中的值仍然为 0;接下来执行第二条指令从端口输出 s7中的值,即输出为 0;然后执行 s7循环移位,移位后 s7的值仍然为 0,接着再从端口输出 s7中的值。这样周而复始,始终从端口输出 0值,因此在上面的验证试验中看不到任何反应。在 PicoBlaze MCU执行第一条指令的过程中,使程序计数器 PC自动加 1,指向下一条指令代码,程序存储器 ROM便在第二个时钟周期后输出第二条指令代码。在第三个时钟周期, PicoBlaze MCU便开始执行第二条指令代码,这样第一条指令代码没有被正确的执行。经过仔细分析,确定问题就出在此处。系统复位后, PicoBlaze MCU执行的第一条指令不是程序中的指令代码。

既然问题出在系统复位后, PicoBlaze MCU执行的第一条指令不是程序中的指令代码。那么能否让 PicoBlaze MCU在复位后执行程序代码中的第一条指令呢?答案是肯定的。只要让程序存储器 ROM在 PicoBlaze MCU执行第一条指令之前就输出第一条指令代码就能解决问题。通过仔细分析汇编编译器生成的 PicoBlaze应用程序指令 VHDL代码文件形成的程序存储器 ROM模块与 PicoBlaze IP核的连接关系,提出以下几种解决方法:

第一种:修改编译生成的 PicoBlaze应用程序指令 VHDL代码文件,添加一个异步复位信号,在复位信号有效时,输出程序指令代码的第一条指令。这样在系统复位后第一个时钟周期的上升沿 PicoBlaze MCU便可以执行其输出的第一条指令。其仿真波形与图 5相似,只是 ROM_dout变化了。

第二种:不修改硬件系统源代码的情况下(包括不修改编译生成的 PicoBlaze应用程序指令 VHDL代码),编写应用程序代码时在程序的第一条指令处放置一条与应用无关的任何一条指令代码,而真正的指令代码从第二条开始编写。因为 PicoBlaz IP核执行的第一条指令始终是 load s0,00指令,而忽略程序代码中的第一条指令。其仿真波形与图 5相似,只是 ROM_dout变化了。

第三种:在程序存储器 ROM与 PicoBlaze IP核之间加一个时钟延迟单元,让程序存储器 ROM的时钟延迟一个时钟周期后再给 PicoBlaze IP核。也就是让 PicoBlaze IP在系统复位后,推迟一个时钟周期才开始运行,而程序存储器 ROM在复位后第一个时钟的上升沿就输出了程序代码中的第一条指令代码。这样在下一个时钟上升沿到来时, PicoBlaze就能执行正确的指令代码。对于这个时钟延迟单元,可以放在应用工程的顶层设计文件中,加到程序存储器 ROM模块和 PicoBlaze IP模块之间,如图 6所示。也可以将这个时钟延迟单元加到 PicoBlaze IP核中,与 IP核集成在一起,如图 7所示。添加延迟单元后的时序仿真波形如图 8所示。

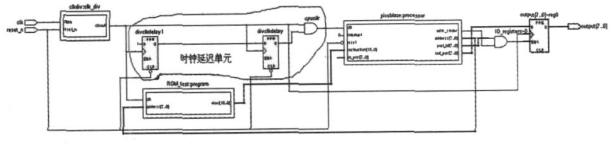

图 6 时钟延迟单元加到顶层设计文件

对上面提出的三种方法,分别对流水灯试验硬件代

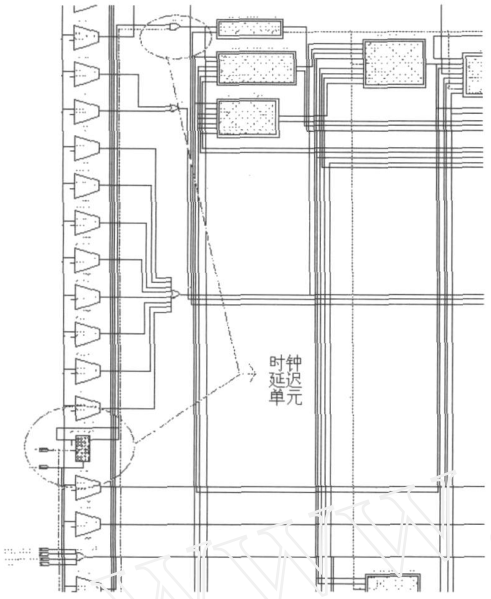

图 7　时钟延迟单元集成到 PicoBlaze IP 核内

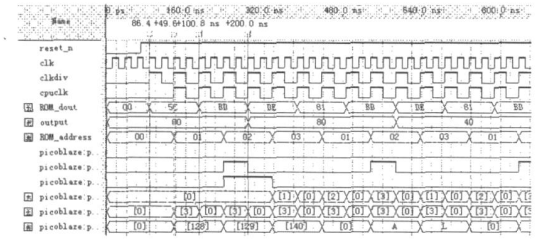

图 8　加时钟延迟单元后 IP 核时序仿真波形图

码及应用程序进行修改,重新编译下载到 MAX-II570T144试验板上进行验证,都获得了预期的试验效果。说明 PicoBlaze IP中存在的 Bug得到了正确解决。

5　结束语

嵌入式微控制器 PicoBlaze是一个典型的 8位微处理器,与传统的 8位 MCU相比,它是一个由 VHDL实现的软处理器 IP Core,可以在各种 FPGA和 CPLD上实现,设计灵活方便。本文通过一个具体的试验,修正了 PicoBlaze IP中的 Bug,并在试验板上进行了验证。其较高的处理性能和较少的资源占用,预示其在基于可编程逻辑和 MCU的应用领域中具有广泛的应用前景。

参考文献:

[1] 董代洁,郭怀理,曹春雨. 基于 FPGA的可编程 SoC设计 [M]. 北京:北京航空航天大学出版社, 2006.
[2] 徐志军,徐光辉. CPLD/FPGA的开发与应用 [M]. 北京:电子工业出版社, 2002.
[3] 赵俊超. 集成电路设计 VHDL教程 [M]. 北京:北京希望电子出版社, 2002.
[4] 徐光辉,程东旭,黄 如. 基于 FPGA的嵌入式开发与应用 [M]. 北京:电子工业出版社, 2006.
[5] 李洪伟,袁斯华. 基于 Quartus II的 FPGA/CPLD设计 [M]. 北京:电子工业出版社, 2006.
[6] Xilinx Corporation, PicoBlaze 8-bit Embedded Microcontroller Datasheet http://www.xilinx.com, 2004.
[7] 江思敏. VHDL数字电路及系统设计 [M]. 北京:机械工业出版社, 2006.
[8] 侯伯亨. VHDL硬件描述语言与数字逻辑电路设计 [M]. 西安:西安电子科技大学出版社, 2001.

PLD Applied Research of Embedded Microcontroller PicoBlaze

GU Shi-fu[1], *YOU Zhi-yu*[1], *DU Yang*[2], *DONG Xiu-cheng*[1]
(1. School of Electrical and Information Engineering, Xihua University, Chengdu 610039, China;
2. Institute of Optics Electronics, Chinese Academy of Sciences, Chengdu 610209, China)

Abstract: PicoBlaze is the embedded 8-bit Microcontroller IP Core developed by Xilinx. In briefly introducing the PicoBlaze structure principle, the performance and the basic development process, this paper elaborates PLD realizes method for PicoBlaze, and carries on the application confirmation on the MAX II570T144 experimental board. Meanwhile, proposes the suitable solution to PicoBlaze IP Core existent Bug.

Key words: microcontroller; PicoBlaze; PLD; MAX II; VHDL