VHDL 语言设计可综合的微处理器内核*

张　楷, 汤志忠
(清华大学 计算机科学与技术系, 北京 100084)

摘　要: 详细介绍了用 VHDL 语言设计可逻辑综合的教学实验用 CPU 的过程。CPU 指令系统构架采用 RISC 结构, 设计上使用结构化编程方法, 将 CPU 内核按照功能划分为不同的模块, 采用 VHDL 语言设计每一个模块的内部功能和外围接口。所有的功能模块组合起来后, 通过 EDA 工具进行 CPU 内核的逻辑综合和功能仿真, 最后在可编程逻辑器件上实现这个完整的 CPU 内核。

关键词: VHDL; 微处理器; 逻辑综合
中图法分类号: TP202　　**文献标识码**: A　　**文章编号**: 1001-3695(2004)06-0123-02

Synthetic VHDL Microprocessor Core Design

ZHANG Kai, TANG Zhi-zhong
(Dept. of Computer Science & Technology, Tsinghua University, Beijing 100084, China)

Abstract: This paper describes a synthetic RISC architecture CPU design and implementation in detail for teaching experiment. To use structure programming method, making the instruction system architecture design, then divide the microprocessor into different function units. Write VHDL code to describle internal function and external interface of each unit. CPU core was simulated and synthesized by EDA tools after combine all units. In the end, CPU core was implemented in programmable logic device.

Key words: VHDL; CPU; Logic Synthesis

电子设计自动化(EDA)在计算机设计过程中扮演着越来越重要的角色, 设计人员几乎可以在 EDA 工具上完成整个系统的开发。电子设计自动化的重要特征就是要求采用形式化的语言来描述硬件电路的功能, 即使用硬件描述语言来描述硬件电路。VHDL 作为一种标准化的硬件描述语言, 是进行硬件电路设计的必备工具。目前国际上知名大学中, 计算机组成与结构的课程实验几乎都采用可编程逻辑器件作为硬件设计平台, 通过硬件描述语言进行 CPU 以及外围接口的设计、仿真和验证。为满足计算机组成与结构的实验要求, 本研究使用 VHDL 语言设计了可逻辑综合的 16 位微处理器内核。

1　VHDL 语言概述

VHDL(VHSIC Hardware Description Language)是指非常高速集成电路的硬件描述语言[2]。目前有国际标准 IEEE STD 1076-1987, 1076-1993 两个版本。VHDL 语言具有以下的优点:

(1) 功能强大, 设计灵活。VHDL 具有强大的语言结构, 可以用简洁明确的源代码来描述复杂的逻辑控制, 并且具有多层次的设计描述功能, 可以直接生成电路级描述。

(2) 易于共享和复用。VHDL 有丰富的仿真语句和库函数, 各种设计模块可以重复使用, 随时可以对设计进行仿真。

(3) 独立于硬件的设计, 与工艺无关。设计者可以不考虑硬件的结构和最终设计实现的目标器件是什么, 而进行独立的设计。

(4) 具备很强的移植能力。因为 VHDL 是一种标准化的硬件描述语言, 同一个设计中可以被不同的工具支持, 使得设计描述的移植十分方便。

可逻辑综合的微处理器采用流行的 VHDL 语言进行设计, 根据它所能够完成的总体功能, 采用自顶向下的设计方法, 将它分成不同的功能单元, 每一个单元实现一定的逻辑功能。

2　指令系统构架

2.1　指令类型

机器语言的集合称为指令集, 微处理器的设计首先是指令系统构架的设计。MIPS 系列 CPU 是精简指令集计算机(RISC)中的代表, 本研究中 CPU 指令集的设计参考 MIPS 处理器的特点, 指令的格式依照"简单的一定是规则的"设计准则[3], 采用 16 位的固定指令长度。操作码和操作数格式十分规整, 其中操作码占用 16 位指令字的高 4 位, 功能区 func 占用低 4 位。指令的执行过程分成取指周期和指令译码执行两大部分。由于内存访问指令在所有的指令中花费最长的时间, 所以将 CPU 指令执行的周期数分为两类, 内存访问指令执行需要 2 个时钟周期完成; 其他所有指令执行都在 1 个时钟周期内完成。

CPU 的寻址方式支持寄存器寻址、立即数寻址、PC 相对寻址方式以及直接寻址方式。为了保证微处理器的指令能够完成所有基本操作, 指令类型按照功能可以分成数据转移指令、算术运算指令、移位指令、逻辑运算指令、控制指令。其中数据

收稿日期: 2003-06-14; 修返日期: 2003-08-28
基金项目: 国家高技术研究发展计划"863"计划资助项目(2001AA111060)

转移指令完成寄存器到寄存器间,寄存器到数据存储器间的数据转移;算术运算指令负责操作数的算术运算;移位指令将操作数进行逻辑移位或算术移位;逻辑运算指令负责操作数的逻辑运算;控制指令包括条件或非条件的指令跳转。每一类指令包含的具体指令如表 1 所示。

表 1 CPU 的指令集

指令类别	指令数目	指令内容
数据转移指令	5	MOV, LR, SR, LHI, LLI
算术运算指令	7	ADD, ADC, SUB, SBC, ADDI, INC, DEC
移位指令	3	SLL, SRL, SRA
逻辑运算指令	7	AND, OR, XOR, NOR, ANDI, ORI, XORI
条件控制指令	6	BEQ, BNE, BLTZ, BLEZ, BGTZ, BGEZ
非条件控制指令	2	JR, JMP

2.2 指令扩展

CPU 的指令格式设计十分规整,具有很好的可扩展性。指令的编码在设计时采用层次化的设计,将不同类型和功能的指令进行编码,分别映射到操作码区和功能区中。需要指令扩展时可以依据指令类型方便地添加新指令到相应的编码区域中,进行指令集的完善。

3 功能单元的设计

3.1 CPU 运行过程

CPU 首先在时钟的上升沿从指令存储器的起始地址处读取一条指令,存放在指令寄存器 IR 中。IR 中的指令送入到控制单元 CONTROL 中进行指令译码,根据 16 位指令字的内容生成相应的控制信号,完成对所有功能单元的控制。寄存器索引控制信号完成对 16 个寄存器文件的选择,被选择的 16 位源寄存器的内容和目的寄存器的内容送入到多路选择器 MUX 中根据指令的类型进行端口选择;选择后的数据输出分别作为累加器 ALU 的输入端口 A 和 B,在 ALU 中进行相应的运算。运算结果的输出送入 MUX 中来决定是否通过回写端口写入到目的寄存器中,同时运算结果送入到 PC 中去,进行下一地址的计算,完成当前指令的执行。

分析每一类指令执行时需要的功能单元,可以将 CPU 的功能单元分为存储器 MEM、程序计数器 PC、累加器 ALU、寄存器文件 REG_BANK、控制单元 CONTROL、多路选择器 MUX 六个部分。为了临时数据能够在下一个周期使用,完成数据的稳定写入和读出,需要添加指令寄存器 IR、内存数据寄存器 MDR、写内存寄存器 WDR、标志寄存器 FLAG 作为临时寄存器。每一个模块的内部功能和外部端口都使用 VHDL 硬件描述语言进行设计[4]。

3.2 微处理器时钟

在设计 CPU 的功能模块之前,需要决定如何逻辑实现和处理器锁存数据。CPU 中有两种不同的逻辑元件:①ALU, MUX, CONTROL 单元的元件都是组合逻辑电路,它们的输出仅仅依赖于当前的输入状态,没有内部存储功能;②存储器、寄存器文件、临时寄存器都是状态单元,它的输出不仅依赖于输入,还有其自身内部的状态。时钟用来决定状态何时被写入,一个状态可以在任意时刻读取。CPU 的时钟采用边沿触发的方式,整个系统采用单时钟电路,即提供一个系统时钟,每当时钟边沿到来的时候,向状态元件写入数据。

3.3 ALU 设计

算术逻辑单元 ALU 是绝大多数指令必须经过的单元,所有的运算都在 ALU 中完成,根据指令的类型对两个 16 位的输入数据进行运算。4 位控制输入信号决定 ALU 的操作类型,可以有 16 种不同的功能组合完成加法、减法、移位、比较等相应的运算功能。VHDL 编写的 ALU 结构体伪代码如下:

```
architecture logic of ALU is-- 结构体声明
信号声明;
begin--结构体 ALU 开始
alu_proc: process(a_in, b_in, alu_function) --ALU 过程开始
变量声明;
16 位数据加法;
16 位数据减法;
case alu_function is; --ALU 功能选择
when ALU 操作类型 = >
根据 alu_function 内容完成 ALU 的运算
end case; --ALU 功能选择结束
ALU 运算结果输出;
end process; --过程结束
end; --ALU 结构体结束
```

3.4 控制单元设计

控制单元负责整个 CPU 的运行控制。在每一个时钟周期的上升沿,指令寄存器 IR 从内存中读取指令字后,控制单元必须能够根据操作码,为每个功能单元产生相应的主控制信号,并给 ALU 提供 4 位控制信号。对于不同的指令,同一个功能单元的输入不同,需要多路选择器 MUX 来对数据通路中功能单元的输入进行选择。控制单元根据 16 位指令字的 4 位操作码和 4 位功能区进行译码,功能区 func1 占用操作码区的 0000H,功能区 func2 占用操作码区的 0001H。VHDL 设计的控制单元结构体伪代码如下:

```
architecture logic of control is-- 结构体声明
控制信号类型声明,包括 ALU 运算类型、原操作数类型、目的操作数类型、PC 源类型、分支控制类型;
begin--结构体 control 开始
control_proc: process(opcode) -- 控制单元过程
变量声明;
begin --过程开始;
case op(3 downto 0) is-- 操作码区译码;
when "000" = >case func(3 downto 0) is;
when "000" = > --功能区 func1 译码
产生控制信号;
end case; -- 功能区 func1 译码完成
when "001" = >case func(3 downto 0) is
when "000" = >--功能区 func2 译码
产生控制信号;
end case; -- 功能区 func1 译码完成
end case; -- 操作码区域译码完成
控制信号译码输出;
end process; --过程结束
end; -- 结构体结束
```

3.5 逻辑综合和仿真

CPU 的每一个功能单元通过 VHDL 语言设计完成后,定义所使用的功能单元接口和它们之间的信号连接关系,将所有功能模块整合起来,进行 CPU 顶层的 VHDL 结构设计。

MAX + PLUS II 是 Altera 公司的可编程逻辑器件开发系统[5],支持 VHDL 语言、VerilogHDL 语言输入以及图形化的输入方式。整个 CPU 设计都在 MAX + PLUS II 的设计环境中进行,各个功能部件的设计以及整体内核都采用 VHDL 语言作为输入方式。

(下转第 173 页)

作(OUT07IRQ = bmEP1) 。

3.2　驱动程序

USB 驱动程序是分层的, 从下至上分为主机控制器驱动、总线驱动、设备驱动。每层分担一部分数据传输任务。

主机控制器位于计算机主板上, 其驱动程序已经由制造厂商开发好并且嵌入了 Windows 操作系统的驱动程序库, 总线驱动也已由 Windows 所包含。开发者要编写的只是设备驱动程序, Cypress 公司提供了通用的设备驱动程序 ezusb. sys 及相应的例子, 可以使用 NuMega 公司的驱动程序开发包 Driverstudio 对驱动程序进行再开发, 使之更适用于所设计的 USB 设备。驱动程序中还包括. INF 文件, 这个文件与主机识别使用哪个驱动程序紧密相连。在识别使用哪个驱动时, Windows 把获得的描述符与它的 INF 文件中的信息进行比较(主要是比较 VID, PID) , 然后选择加载, 此任务由 Windows 的设备管理器实现。USB 设备驱动基于 WDM, 适用于 Win98 与 WinNT 平台。

3.3　主机控制程序

由于 EZ-USB 是基于 RAM 的, 也就是说它用 RAM 来存储程序代码和数据, 所以在进行数据传输之前, 需要将 8051 程序由主机下载到 EZ-USB 的 RAM 中, 此下载可通过两个途径: 串口或 USB 总线。因此首先需要一个固件下载模块, 它主要实现下载固件, 获取设备信息两个功能。

如果只是将 8051 程序下载到 EZ-USB 中, 此时还不能达到数据传输的目的。主机控制程序首先由 CreatFile( ) 打开设备, 然后就可以调用 Windows 的 API 函数 DeviceIoControl( ) 发送 IRP 请求以实现对 USB 设备的各种控制及数据读写操作, DeviceIoControl( ) 充当了应用程序和设备驱动程序之间的接口, 以下将作详细说明。当需要执行大块数据读写任务的时候, 可以调用另两个 API 函数: ReadFile( ) 与 WriteFile( ) 。

```
BOOL DeviceIoControl( HANDLE hDevice,
DWORD dwIoControlCode,
LPVOID lpInBuffer,
DWORD nInBufferSize,
LPVOID lpOutBuffer,
DWORD nOutBufferSize,
LPDWORD lpBytesReturned,
LPOVERLAPPED lpOverlapped) ;
```

其中, hDevice 表示设备句柄, 也就是由 CreateFile( ) 得到的返回值; dwIoControlCode 是操作代码, 一般为类似 IOCTL _ × × × ×的宏(可以自定义) , 它一定要与设备驱动程序相匹配, 包含同样的头文件; lpInBuffer 定义输入缓冲区指针, 具体取决于参数 dwIoControlCode; nInBufferSize 定义输入缓冲区大小; lpOutBuffer 定义输出缓冲区指针, 具体取决于参数 dwIoControlCode; nOutBufferSize 定义输出缓冲区大小; lpBytesReturned 表示实际装载到输出缓冲区中的字节数; lpOverlapped 这个结构用于重叠操作, 主要针对同步传输, 一般赋为 NULL。

图 3 为该 USB 接口的电调谐滤光器件联机控制系统的结构图。

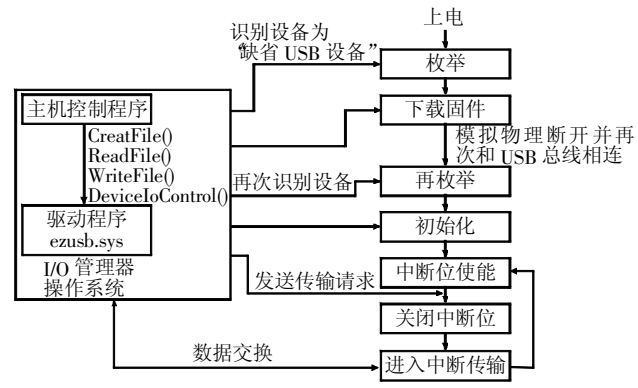

图 3　基于 USB 接口的电调谐滤光控制系统结构

参考文献:

[1]　美] Jan Axelson. USB 大全[M] . 北京: 中国电力出版社, 2001.

[2]　niversal Serial Bus Specification(Revision 1. 1) [M] . American: Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, NEC Corporation, 1998.

[3]　Z-USB Technical Reference Manual(version 1. 9) [M] . American: Cypress Semiconductor Corporation, 2000.

作者简介:

孙勇(1980-) , 男, 硕士研究生, 主要从事 CPLD、计算机接口技术、图像采集系统和光谱仪器的研发工作; 曾立波(1963-) , 男, 教授, 主要研究方向为图像处理和光谱仪器; 张宏波(1980-) , 男, 博士研究生, 主要从事计算机接口技术、光谱仪器的研发工作。

(上接第 124 页)

为了满足 CPU 的测试要求, 设计测试端口输出, 保证能够对 CPU 实现的所有指令进行仿真测试和功能验证, 在 MAX + PLUS II 内部集成一个波形编辑器和仿真器, 能够对 VHDL 逻辑综合后的 CPU 进行仿真。根据指令集的分类编写测试数据, 将测试数据作为仿真矢量在 MAX + PLUS II 中对于 CPU 进行仿真测试, 分析每一条指令的运行结果是否正确。最后对经过验证的 CPU 文件进行编译、逻辑综合、目标器件适配, 然后可以下载到目标器件中去, 完成 CPU 的设计。

4　结论

VHDL 硬件描述语言为计算机硬件设计提供了很大的方便, 结合可编程逻辑器件的灵活性、通用性的特点, 实现的 CPU 内核配合外围接口电路可以替代传统的计算机组成与结构实验计算机。实验人员不仅理解 CPU 的内部组成与原理, 并且能够学习和掌握流行的硬件设计方法, 进行自主性设计实验。

参考文献:

[1]　王爱英. 计算机组成与结构(第二版) [M] . 北京: 清华大学出版社, 1995.

[2]　姜立东. VHDL 程序设计及应用[M] . 北京: 北京邮电大学出版社, 2001.

[3]　ohn L Hennessy, David A Patterson. 计算机组织与设计硬件/软件接口(英文版, 第二版) [M] . 北京: 机械工业出版社, 1999.

[4]　卢毅, 赖杰. VHDL 与数字电路设计[M] . 北京: 科学出版社, 2001.

[5]　廖裕评, 陆瑞强. CPLD 数字电路设计——使用 MAX + PLUS II [M] . 北京: 清华大学出版社, 2001.

作者简介:

张楷(1974-) , 男, 硕士研究生, 主要研究方向为计算机组成与原理、计算机体系结构、高速计算机和总线物理层、数据层测试等; 汤志忠(1946-) , 男, 教授, 博士生导师, 主要研究方向为高性能计算机体系结构、并行算法、并行编译技术等。