第33卷 第10期 计 算 机 工 程 2007年5月
Vol.33 No.10 Computer Engineering May 2007
·开发研究与设计技术· 文章编号：1000—3428(2007)10—0280—03 文献标识码：A 中图分类号：TH166

Javacard CPU的设计与实现

张德学，郭 立，傅忠谦，何 力
（中国科技大学电子科学与技术系，合肥 230026）

摘 要：支持Javacard技术标准是智能卡的发展方向，目前的Javacard系统大多是采用软件虚拟机的方式来解释执行或者通过just-in-time方式执行Java指令，系统软件平台本身占用了大量的资源，且执行效率不高。解决这些问题的方法就是实现硬件Javacard指令处理器。该文给出了一种基于微码的Javacard指令处理器的FPGA设计和实现，以Javacard CPU为核心搭建Javacard CPU测试平台，并将其集成在一块FPGA上实现。

关键词：Javacard；处理器；智能卡

Design and Implementation of Javacard CPU

ZHANG Dexue, GUO Li, FU Zhongqian, HE Li
(Department of Electronic Science and Technology, University of Sciences and Technology of China, Hefei 230026)

【Abstract】Javacard is the development direction of smart card. Most of the Javacard systems use the software virtual machine to execute Javacard instruction. The system takes up a great deal of hardware resources, and it is unefficient. The resolution is to implement the hardware Javacard instruction processor. This paper describes a design of Javacard hardware processor based on microcode.

【Key words】Javacard; CPU; Smartcard

1 概述

1.1 Javacard简介

智能卡是指集成了CPU、ROM、RAM、COS(芯片操作系统)和EEPROM，能储存信息和图像，具备读/写能力，信息能受到加密保护的便携卡。智能卡最基本的标准是ISO/IEC7816。智能卡在银行、电信等行业中得到了广泛的应用，但在发展过程中也有很多的障碍，主要有：各个厂商指令集不统一；编程接口APIs太复杂；开发环境不通用，为新卡开发需要熟悉新卡开发环境；各个厂商的系统不兼容，专卡专用。开发门槛过高影响了智能卡的发展。市场对智能卡的发展提出了新的要求，Sun公司提出了Java Card开放标准[1]。Javacard技术是将智能卡与Java技术相结合的产物，它将Java平台应用到像智能卡这样高度特殊化、内存和处理能力比J2ME设备还受限的设备上。它克服了智能卡硬件和软件开发技术太专业、开发周期长等阻碍智能卡普及的缺点，允许智能卡运行Java编写的应用程序。Javacard技术继承了许多Java语言的优点，制定了一个安全、便捷且多功能的智能卡平台。

Javacard规范主要包括：Javacard虚拟机规范，Javacard编程接口(API)和 Javacard运行环境规范。

Javacard 最小的硬件配置要求为：512B RAM，24KB ROM，8KB EEPROM，8位处理器。典型的Javacard 设备有8或16位的CPU，3.7MHz时钟频率，1KB大小的RAM和大于16KB的非易失存储器(EEPROM或FLASH)。高性能的智能卡还带有独立的处理器和加密芯片以及密码信息存储。

Javacard 系统的实现有基于软件虚拟机和基于硬件两种方法。基于软件虚拟机方法是在非Java处理器硬件上用软件方法模拟实现Javacard平台，在此平台上实现Javacard应用。基于硬件方法是硬件逻辑实现Javacard处理器，在此硬件基础上实现Javacard平台，再在此平台上实现Javacard应用。

1.2 Java处理器的实现方式比较

Java处理器的实现方式如下：

(1)通用CPU+OS+Java软件解释器，软件解释执行Java指令；

(2)通用CPU+OS+Java JIT(just-in-time)编译器，按块编译执行Java指令；

(3)Java加强CPU+OS+特殊的Java编译器，充分使用Java加强硬件的优势；

(4)Java硬件CPU，本地支持执行Java指令，效率最高。

目前的Java系统是基于软件虚拟机实现的，软件解析执行 Java 指令，如方式(1)和方式(2)；但智能卡的硬件能力远远不如PC，ROM、RAM资源非常有限，用软件实现Javacard虚拟机，需要软件Javacard指令解释器，将Java指令转换到本地CPU的指令集，由于其速度慢、实现虚拟机本身占用内存资源，因此不适合在资源有限的硬件中应用。

方式(3)要求CPU硬件实现部分Java指令，它需要特殊的编译器来充分发挥Java加强CPU的功能。

方式(4)是最终的解决方法，Java指令的执行不再需要先转换到宿主CPU的本地指令集上去，是最有效率的方法，同时，它不占用RAM等软件资源，可以给应用程序提供更多的资源。

鉴于支持Javacard标准的智能卡将被大量使用，为提高应用程序执行速度，节省资源，充分利用资源，硬件实现Javacard指令处理器是有重要意义的。

作者简介：张德学(1977－)，男，博士生，主研方向：集成电路设计研究；郭 立，教授、博导；傅忠谦，副教授；何 力，助教

收稿日期：2006-05-22 **E-mail：**dxzhang@ustc.edu

1.3 Java CPU 的研究现状

根据应用的不同，Java 虚拟机所要支持的指令集和特性也有差异，Java 指令处理器大体分为通用型、嵌入式应用和 Javacard 应用。通用型 Java 处理器实现完整的 Java 虚拟机标准。嵌入式应用中的 Java 处理器可以根据应用系统的实际需要取舍一些 Java 特性。Javacard 标准中的 Java 处理器是 Java 虚拟机标准的一个子集，只需要能完成智能卡应用中的必要操作，同时它要求尽量少地占用资源和低功耗。

通用型的 Java 处理器实现有 Sun 公司的 PicoJava I 和 PicoJava II。PicoJava 系列实现了完整的 Java 指令集，旨在提供高性能，加强的硬件浮点支持，6 级指令流水线和多重指令调度。要达到这种性能，付出的代价是逻辑门数很高。用现有水平的 FPGA 来完全实现 PicoJava 是不实际的。

ARM 的 Jazelle 技术，延伸 RISC 的基础框架，让 ARM 晶片直接执行 Java 字节码，目前支持 95%的 bytecode 指令，其余的指令通过转换为 ARM 指令串实现。这种芯片实际是同时支持了两套指令集，属于 Java 加强 CPU。

最近非常活跃的一个Java 嵌入式CPU设计的项目JOP (Java Optimized Processor)[2,3]，致力于实现一个用于嵌入式设备的Java处理器。该项目是一个开源项目，目前已经取得了很好的成果。该项目的设计中为IO操作和一些系统任务引入了一些非Java指令，在此系统上完成的Java应用可能会有兼容性问题。

对于专用于 Javacard 上的 Java 处理器芯片尚无实现报道，本文描述了 Javacard CPU 设计。系统采用 Verilog 描述，设计成一个可以灵活配置、方便修改、资源占用少、兼容性好、可以在普通 FPGA 中实现的软核。

2 Javacard CPU 的设计

2.1 Java CPU 的硬件实现技术

在CPU的设计中，当从内存中取出下一条指令时，执行这条指令有 2 种处理方法：硬件逻辑和微码序列[4]。硬件逻辑方法就是使用译码器、锁存器、计数器和其他一些逻辑部件转移和操作数据，完成指令功能。微码方式是内部实现一个非常简洁、快速的微码处理器，此微码处理器的每条指令对应很简单的硬件动作(一般是单周期指令)，将要执行的CPU指令作为索引，索引到微码ROM中的某个地址，通过执行此地址处的一组微码完成指令的功能。硬件逻辑方法的优点是能设计出更快的CPU，缺点是难以实现复杂的指令集，同时会导致芯片面积增大。微码方法的优点是能减少芯片的面积，能实现复杂指令集，缺点是速度可能会慢一些。实际的CPU设计中采用哪种方法是个权衡利弊的问题，在速度不是关键的时候，微码方法是个很好的选择。

Java 语言是完全的面向对象语言，它的指令集也是为面向对象思想而设计的，在指令集层次上支持面向对象操作，这样的指令要完成的操作是很复杂的，很难用硬件直接实现，本文采用了微码设计。

2.2 wishbone 接口

wishbone 接口是 opencores 发布的片内总线接口标准，简洁高效。在此次的 Javacard CPU 设计中，接口采用了 wishbone 接口。

2.3 Javacard CPU 结构

本文设计的 Javacard CPU 核心部分是微码处理器，用微码指令序列实现 Javacard 指令。微码处理器主要由以下几个部分组成：主控逻辑 core，运算单元 ALU，内部堆栈单元 Stack，微码 ROM，微码指令指针调整模块 Mcpc，外存读写接口 memrw，通过 wishbone 总线连接外部 RAM、ROM、I/O。各模块之间连接关系、数据通路、控制通路以及应答信号连接见图 1。

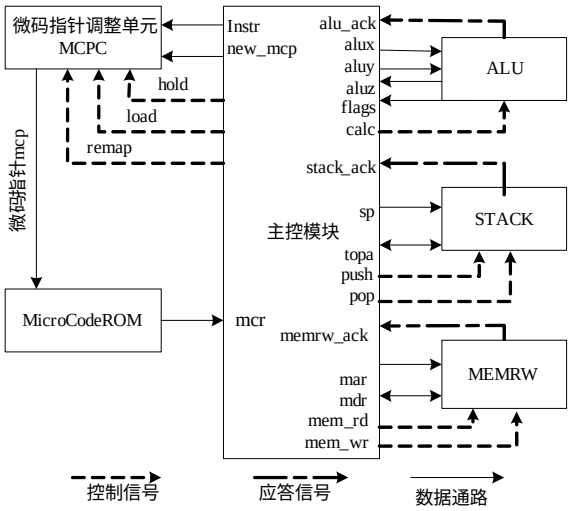

图 1 数据通路和控制通路

2.4 微码处理器各模块接口及功能简介

按照功能将设计划分为几个功能模块，每个模块易于描述和实现。

(1)运算单元 alu: module alu(x,y,op,z,flag,calc,rst,ack,clk)。其中，x,y 是输入操作数，op 是操作码，z 是输出结果，flag 是输出运算结果标志，calc 是运算使能控制信号，ack 是运算结束应答。本模块完成 op 定义的运算，并给出标志位和应答。

(2)内部堆栈 stack: module stack(clk,rst,pop,push,data_i, data_o,sp, ack)。其中，pop,push 是堆栈的弹出、压入操作信号，data_i,data_o 是数据输入输出，sp 是堆栈指针，ack 是堆栈操作结束应答。本模块根据 pop,push 信号对堆栈进行操作。

(3)微码 ROM：module microcodeROM(mcp,mcr)。其中，mcp 是微码 ROM 的指针，mcr 是微码寄存器。根据微码指针 mcp，在 mcr 上输出 mcp 处的微码数据。

(4)微码指令指针调整模块 mcpc: module mcpc(clk,rst, load,new_mcp,hold,remap,instr,mcp)。微码指针的操作有 3 种：保持，重加载，重映射。重加载是用 new_mcp 的值作为新的 mcp 值。重映射是将 CPU 指令 instr 对应的微码序列首地址作为新的 mcp 值。

1)load 信号有效，用 new_mcp 的值给 mcp 赋值；

2)hold 信号有效，保持 mcp 值不变；

3)remap 信号有效，则将 CPU 指令 instr 作为索引，得到 instr 指令对应的微码序列首地址，将首地址赋给 mcp。

以上 3 个信号均无效时，每时钟 mcp 则自动加 1。

(5)外存读写接口 memrw。

module memrw(clk,addr,data_read_in,data_write_out,ack,rst,rd, wr,wb_stb_out,wb_cyc_out,wb_ack_in,wb_addr_out,wb_data_in,wb_data_out,wb_we_out);

对外接口采用了开源的 wishbone 总线标准，wb*信号是 wishbone 相关信号。根据 rd，wr 读写信号，操作 wishbone 信号，等待 wishbone 的应答，然后将数据和应答信号反馈给主控模块。

2.5 Javacard CPU 设计的特点

Javacard CPU 在设计上具有如下特点：

(1)主控模块与其他从模块之间用使能信号和应答信号保持同步,从而可以不假设从模块在规定的时间内完成操作,从模块在完成操作后给出应答信号即可，从而可以匹配不同速度的从模块。

(2)微码指令的设计：所有的微码指令为单指令，即不带任何操作数。微码指令本身包含所需操作的信息，如在哪 2 个寄存器之间转移数据。对于跳转操作等必须带后续操作数的指令，本文采取了变通的方法，先将所需操作数存入内部寄存器，再执行跳转等指令。

详细例解：微码定义为 16bits。bit15 指示本微码是指令还是数据，bit15==1 表示是数据，此时微码的低 8 位是一个数据，处理此微码时，要将此 8 位数据提取出来，存入内部寄存器。bit15==0 表示是指令。当需要执行一个跳转 Jmp 0x0809 时，用微码方法表示为：

0x8008 //bit15==1,是数据型微码

0x8009

JMP //指令型微码助记符

执行时，遇到前面的两个数据型微码，会将 08 和 09 存入内部 16 位数据寄存器的高低 8 位，执行 JMP 指令时，隐含使用此内部数据寄存器。

(3)所有的微码指令是单周期指令。由于采用了特点(2)中所述的单指令微码，微码指令执行时不需要读取后续操作数的周期,在执行当前微码指令的同时读取下一条微码指令,可以做到每时钟执行一条微码。

(4)简洁的主控模块状态机。所有 Javacard 指令均由微码执行，而不采用硬件陷入、软件模拟方式，简化了主控逻辑的设计，仅有 2 个状态：EXEC_MC 和 HLT，CPU 复位后，一直处于执行微码 EXEC_MC 状态,直至遇到 HLT 微码指令。

(5)IO 采用内存映射方式统一编址，避免了引入非 Java 指令，保证了兼容性。

2.6 系统微码实现

系统采用微码实现，用微码序列控制读取 Java 指令、存储数据，实现 Java 指令。Javacard 指令被解释执行的过程如下:读取 Javacard pc 处的 Javacard 指令至指令寄存器 Instr，发出 remap 信号给微码指针调整模块 MCPC，微码指针寄存器 mcp 得到新的 Javacard 指令对应的微码序列首地址，mcp 的变化使微码指令寄存器 mcr 变化为该微码序列的首个微码指令，再由微码处理器执行此 mcr 中的微码。

3 Javacard CPU 测试平台的 FPGA 实现

3.1 外围接口和模块、测试平台框架

测试平台是以一块 xc2s200 芯片为核心的简单开发板，全部设计都在此芯片内实现，包括 CPU 逻辑、存储单元等，板上的 8 位 LED 指示灯用作 I/O 输出端口。测试平台的框架结构见图 2。

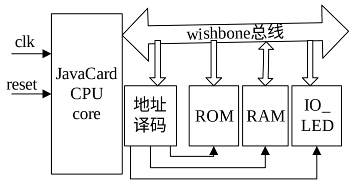

图 2 测试平台结构

3.2 结果说明

本设计是用 verilog 语言实现的，内部使用 16 位数据总线，对外是 8 位的 wishbone 总线，内部堆栈大小 200H，微码 ROM 为 4KB,外接 512B 的 ROM 和 512B 的 RAM。Javacard 定义了 187 条指令，其中 47 条指令是涉及 32 位整型数的，对 32 位整型数的支持是可选的，本次没有实现对 32 位整型数操作的指令，遇到未定义的指令的操作为宕机。共定义了 109 条微码指令。用微码指令序列来完成系统初始化操作和解释 Javacard 指令，用了 3 273 条微码指令序列，约合每条 Javacard 指令用 17 条微码指令来执行(主要是有些面向对象的复杂指令需要更多的微码来解释)。整个系统占用资源很少：4 个 block RAM，2 052 个 SLICE，可以在普通 FPGA 上实现。将测试程序下载到板上的 ROM 中，以 24MHz 时钟运行通过，验证了 Javacard 指令处理的正确性，性能上也完全满足 Javacard 虚拟机标准要求。

4 总结

实现 Javacard 硬件 CPU 是 Javacard 的发展方向,它不需要很高的性能，而是需要低成本、资源占用少、低功耗等特性。Javacard 指令集是面向对象的复杂指令集，很难直接用硬件实现。采用微码方式实现是很好的选择，每一条微码都对应一个很简单的硬件动作，硬件很容易实现，并且使用的资源也较少。用微码序列来完成 Javacard 指令，使硬件设计保持简洁、灵活、易于修改，有些改动只需要重写微码序列而不需要更改硬件设计；添加新的功能支持也只需要修改微码，如硬件实现加密方法调用接口。Javacard 硬件 CPU 的实现必将促进 Javacard 的应用。

参考文献

1 Microsystems Sun Inc.. Java Card Platform Specification[EB/OL]. 2006. http://java.sun.com/products/javacard/specs.html.

2 Schoeberl M. JOP: A Java Optimized Processor[C]//Proc. of Java Technologies for Real-time and Embedded Systems Conference. 2003.

3 Schoeberl M. Java Technology in an FPGA[C]//Proceedings of the International Conference on Field-progRAMmable Logic and Its Applications. 2004.

4 Hyde R. The Art of Assembly Language[EB/OL]. 2004-02. http://webster.cs.ucr.edu/AoA/Linux/HTML/AoATOC.html.

~~~~~~

(上接第 264 页)

参考文献

1 Lee T. Modern Computer-aided Maintenance of Manufacturing Equipment and System, Review and Prospective[J]. Computer & Industrial Engineering, 1995, 28(4): 793-811.

2 李林功, 李继凯, 谷金宏. 嵌入式系统集成开发环境的构成[J]. 计算机工程, 2001, 27(5): 146-148.

3 李　东, 曹忠升, 冯玉才, 等. 移动数据库技术研究综述[J]. 计算机应用研究, 2000, 17(10): 4-8.

4 韩立燕, 尹仕任, 青学江. 故障诊断专家系统应用研究[J]. 西南交通大学学报, 1997, 32(5): 534-540.

5 李　旭, 刘辉林, 徐心如. 故障诊断专家系统的归纳学习方法[J]. 小型微型计算机系统, 1997, 18(6): 70-74.

6 冯建农, 赵　铭. 故障诊断专家系统推理机研究[J]. 华南理工大学学报, 1997, 25(3): 19-25.
~~~~~~