第38卷第3期 中国科学技术大学学报 Vol. 38, No. 3
2008年3月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Mar. 2008

文章编号:0253-2778(2008)03-0288-05

面积优化与低功耗设计的 JavaCard 处理器

张德学,郭 立,傅忠谦

(中国科学技术大学电子科学与技术系,安徽合肥 230026)

摘要:AOJCP(area-optimized JavaCard processor)是一种自主设计的基于微码、面积优化、低功耗的 JavaCard 硬件处理器. 首先,描述了 AOJCP 中的微码处理器总体框架、单周期与控制-应答机制结合的微码设计、JavaCard 指令执行过程;然后在 FPGA(field programmable gate array)中实现. 结果表明,AOJCP 占用1 410 LogicCells(约合 16 k 门),最高频率 36.2 MHz,典型时钟频率 3.7 MHz 时,功耗为 48 mW.

关键词:JavaCard;智能卡;处理器

中图分类号:TP332.1 **文献标识码**:A

Area-optimized and low power consumption JavaCard processor

ZHANG De-xue, GUO Li, FU Zhong-qian

(*Department of Electronic Science and Technology, USTC, Hefei* 230026, *China*)

Abstract: AOJCP is a microcode based, area-optimized, low power consumption JavaCard processor. The overall architecture of microcode processor, the microcode design unifying single-cycle and control-acknowledge mechanism and execution process of JavaCard instruction are discribed. The implementation result in FPGA(field programmable gate array) shows that AOJCP takes up 1 410 LogicCells(about 16 k gates), that the highest estimated frequency is 36.2 MHz, and that the power consumption is 48 mW when it runs at 3.7 MHz.

Key words: JavaCard; smartcard; processor

0 引言

JavaCard[1]是智能卡与 Java 技术相结合的产物,它是新一代智能卡的标准,应用前景广阔. 据 Java Card Forum(JCF)2005 年的统计[2],JCF 成员发行了近 10 亿张 JavaCard.

JavaCard 最小的硬件配置要求为:512 bytes RAM, 24kb ROM, 8 kb EEPROM (electrically erasable programmable ROM),8 位处理器. 典型的 JavaCard 设备有 8 或 16 位的处理器,3.7 MHz 时钟频率,1 k 的 RAM 和大于 16 k 的非易失存储(EEPROM 或 flash). 高端的智能卡还带有加密芯片.

JavaCard 系统的实现有基于软件和基于硬件两种方法. JavaCard 设备的资源一般都非常有限,硬件方式实现 JavaCard 处理器与软件方式相比,具有资源占用更少;运行 Java 程序效率更高;可根据应用需要扩展或裁减(如扩展AES加解密硬件模

收稿日期:2006-06-29;**修回日期**:2006-10-23

作者简介:张德学,男,1977 年生,博士生/助教. 研究方向:集成电路设计与应用研究. E-mail: dxzhang@ustc.edu

通讯作者:郭立,教授. E-mail: lguo@ustc.edu.cn

表 1　Java 处理器项目

Tab. 1　Java processor projects

项目名称	支持标准	实现方式	占用资源	速度/MHz	源码
JSM[3]	JavaCard	软核,FPGA	2 400 LCs	7.5	否
LavaCore[4]	Java	软核,FPGA	3 800 LCs	20	否
Komodo	Java,50 指令	软核	2 600 LCs	20	否
PicoJava[5]	Java	RTL 设计	440 k 门	N/A	是
JOP[6]	Java	FPGA	1 831 LCs,3.25 kb	101	是
Moon	Java	FPGA	3 660 LCs,4 kb	N/A	否
aJile[7]	Java	ASIC	25 k 门+ROMs	100	否

【注】 “占用资源”中,单位 LC 指 LogicCell,是 FPGA 中的基本单元,代表一个 4 输入查找表 LUT 加一个 D 触发器,约等效于 ASIC 设计的 12 门.

块)等优点.

国际上有不少的 Java 硬件处理器项目,专门的 JavaCard 硬件处理器设计项目尚不多见,表 1 给出了典型的 Java 硬件处理器.

表 1 中的项目大部分是商业项目(产品),不公开设计文档和源码;设计追求高性能,占用面积过大,不适合在资源非常有限的 JavaCard 设备上使用,因而,本文提出了一种以面积优化和低功耗为目标的 JavaCard 硬件处理器设计-AOJCP (area-optimized JavaCard processor).

1　AOJCP 的设计

1.1　设计决策

1.1.1　基于微码处理器设计

Java 语言是完全面向对象的语言,它的指令集是面向对象思想而设计的,在指令层次上支持面向对象操作,同时 Java 指令集也是面向堆栈的指令集,这样的指令要完成的操作很复杂,很难用硬件逻辑直接实现,至今没有完全用硬件逻辑实现 Java 指令集的设计,而用微码方式实现是很好的选择.

JOP(Java optimized processor)是开源项目,它采用微码处理器设计,但它的微码指令是面向堆栈操作的,Java 指令集本身也是面向堆栈设计的,因而 JOP 的一些微码与 Java 指令有一一对应关系,可以用较少的微码解释 Java 指令集. 由于面向堆栈的微码指令集仍是复杂指令集,需要较多的硬件逻辑来实现,不利于降低功耗,考虑到采用 RISC (reduced instruction set computing)架构能设计出高性能、低功耗的处理器,因而 AOJCP 的微码处理器采用了精简指令集.

1.1.2　优先使用微码完成硬件操作

AOJCP 的设计目标是在用硬件和微码都能完成某项功能的情况下,首选微码用于完成操作,以减少芯片面积. 如 JavaCard 指令取指操作、除法运算、取模运算,都使用微码程序完成.

1.1.3　基于 FPGA 的低功耗设计

JavaCard 处理器需要低功耗,AOJCP 要用 FPGA 实现和验证,因而重点考察 FPGA 上的低功耗设计技术. FPGA 设计的总功耗包括静态功耗和动态功耗两个部分. 其中,静态功耗是指逻辑门没有开关活动时的功率消耗,主要由泄漏电流造成,随温度和工艺的不同而不同. 静态功耗主要取决于所选的 FPGA 芯片产品. 动态功耗是指逻辑门开关活动时的功率消耗. 动态功耗是电压 V_{dd}、电容 C、资源使用率 U、翻转频率 f 的函数. 其计算公式[8]为

$$P = V_{dd}^2 \sum C_i \cdot U_i \cdot f_i \tag{1}$$

式中,U_i 代表了 FPGA 中不同类型的资源使用率,总功耗是对不同类型资源功耗的求和. 基于 FPGA 的设计中,选定 FPGA 芯片后,其静态功耗基本确定,可以通过降低动态功耗来获得整体的低功耗. 考察动态功耗计算公式,可以从如下方面努力:

(Ⅰ) 减少设计的资源占用,降低资源使用率;

(Ⅱ) 尽量使用 FPGA 芯片上提供的专门硬件资源实现功能;

(Ⅲ) 使能设计,从降低翻转频率入手,仅在需要的时候执行寄存器翻转.

1.2　微码处理器总体框图

AOJCP 硬件的核心部分是微码处理器,微码处理器主要由六个部分组成:运算单元 ALU、内部堆栈单元 stack、MicrocodeROM、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)、主控逻辑 core.

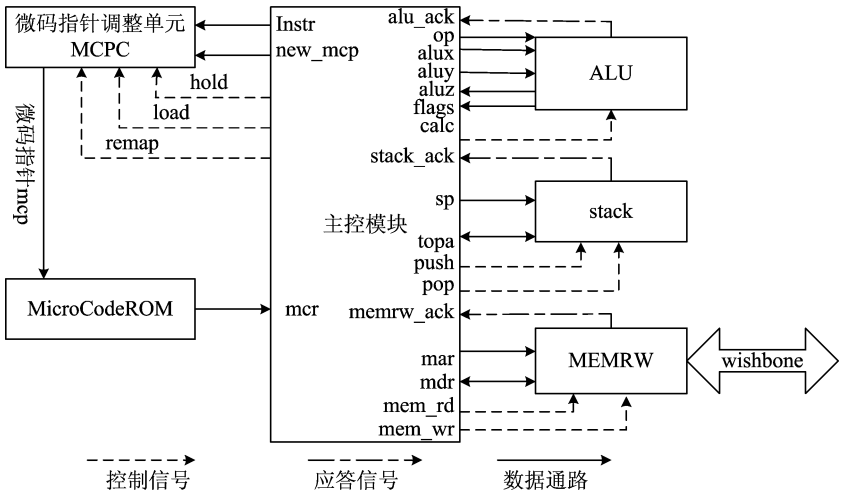

图 1 数据通路和控制通路

Fig. 1 Data-path and control-path

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(mcp 是微码指针,mcr 是微码指令寄存器,mar 是内存地址寄存器,mdr 是内存数据寄存器,Instr 是 JavaCard 指令寄存器,new_mcp 是新微码指针寄存器).

右侧三个从模块采用低功耗设计,仅在控制信号有效时执行相应的操作,操作完成后给出应答信号. 左侧是微码指针调整模块.

1.3 微码指令设计

微码对应基本的硬件操作,设计为 16bit、不带后续操作数,全部操作内含在微码中. 微码程序运行时需要的立即数也编码在指令中,由微码中的最高位指示该微码是指令型微码还是数据型微码. 微码用硬件逻辑实现,完成如下操作:寄存器间值的传送;发控制信号;寄存器+1,-1 等简单运算;算术和逻辑运算;跳转.

AOJCP 中主、从模块之间通过控制-应答机制通信,主控模块发出控制信号,从模块响应控制信号,完成操作后给出应答信号,主控模块检测到应答信号后继续程序的运行. 采用控制-应答机制,主控模块可以不假设从模块操作的执行时序,从而可适应以不同方式实现的从模块.

AOJCP 的微码均设计为单周期执行,这样有利于简化主控模块的设计,但某些指令涉及从模块的操作,需要等待从模块的应答,对这样的操作,有不同的实现方法. JOP 项目中为此设计了两类指令,分别对应发出控制信号指令和检测应答信号指令,在微码程序中两条指令连用,保证从模块的操作完成. AOJCP 的微码设计是将单周期与控制-应答机制相结合,指令实现逻辑中内建检测应答信号的硬件逻辑,在微码程序中只需要一条指令. 指令的逻辑设计如图 2 所示.

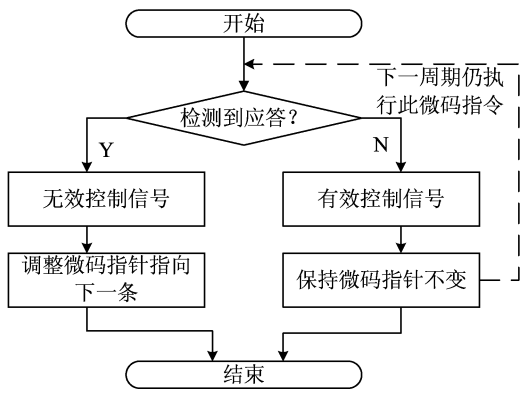

图 2 单周期与控制-应答结合的微码设计

Fig. 2 Microcode design that unified single-cycle and control-acknowledge mechanism

设计中有多种需要从模块完成的操作. 以外存写操作为例,分别用 JOP 方法和 AOJCP 方法做了简化的设计实现(为缩短 verilog 代码长度,将 begin-end 用{}代替,无关代码省略).

```
JOP 方法:case(microcode)
wait_MEMWR:{ if(! memrw_ack) { hold_mcp<=~hold_mcp; }//未检测到应答,保持微码指针不变
              else { memwr<=0; } }// 检测到应答,无效写信号
MEMWR:{ memwr<=1; } endcase//有效写信号
```

```
AOJCP 方法:case(microcode)
MEMWR:{ if(! memrw_ack) { memwr<=1; hold_
        mcp<=~hold_mcp; }//见图 2
      else{ memwr<=0; } } endcase
```

上述两例的 FPGA 实现结果见表 2.

表 2　JOP 与 AOJCP 微码实现方法比较

Tab. 2　Comparison of microcode implemetation between JOP and AOJCP

方法	微码个数	LogicCells	等效门数
JOP	2	9	64
AOJCP	1	6	46
差值	1	3	18

对比 JOP 方法,AOJCP 方法无需专门的检测应答信号的指令,减少了微码指令个数,简化了微码程序的编写,缩短了微码程序长度,当微码程序中大量使用此类操作时,效果将很明显. 同时,实验表明 AOJCP 占用更少的资源,因而,在微码设计中,使用了 AOJCP 方法.

采用单周期与控制-应答机制相结合的微码设计,在指令需要从模块应答时,指令可能会被执行几次直到检测到应答信号,但从主控逻辑考虑,每个微码指令均是单周期执行,设计主控逻辑时仅需要考虑单周期情况,简化了主控逻辑的设计.

1.4　主控逻辑

主控逻辑是微码处理器的核心,负责微码执行、发出控制信号指示从模块动作、控制微码程序流程等. 微码处理器的任务就是执行微码,主控状态机只有两个状态:执行微码 EXEC_MC 和宕机 HLT. 系统复位后,进入 EXEC_MC 状态,当遇到非法指令时,进入 HLT 状态. 主控逻辑流程如图 3 所示.

1.5　测试

将 JavaCard 处理器核通过 wishbone 总线连接外部 ROM,RAM 等构成测试系统. 测试程序使用 Sun 的 JavaCard kit 2.2.1 开发并编译. 测试程序 1:创建长度为 6 的 short 类型数组,写入数值,做冒泡排序(含比较、跳转、循环等操作);测试程序 2:创建一个新对象,调用对象的两 short 类型数求和方法计算,然后返回并将结果保存到静态域(包含操作对象的各个方面,创建对象、调用方法、访问静态域、访问实例域、方法返回等),测试结果见表 3.

AOJCP 与 JOP、aJile-100、TINI 中典型 Java 指令执行周期的比较见表 4(JOP、Jile-100、TINI 的数据来自文献[6]).

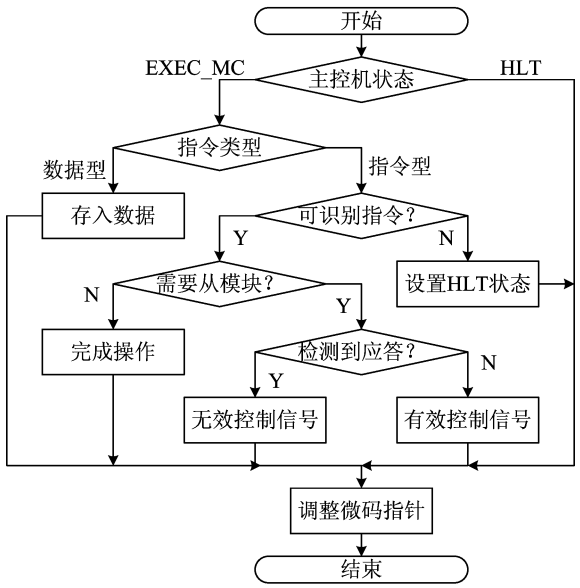

图 3　主控逻辑流程

Fig. 3　Flow chart of main control logic

表 3　测试程序总结表

Tab. 3　Summary of test routines

比较对象	测试 1	测试 2
初始化微码个数	设置 JavaCard 系统寄存器,用了 99 条微码指令	
执行微码个数	11 575	1 173
执行 JavaCard 指令个数	458	22
微码/JavaCard 指令	25	48.8
VCD 文件(用于功耗分析)	sort.vcd	class_method.vcd

表 4　典型指令周期比较

Tab. 4　Comparison of executive cycles for typical instructions

比较对象	AOJCP	JOP	aJile-100	TINI
sadd/iadd(加法)	21	2	8	789
getfield(取域)	48	25	23	2 398
invokestatic(执行静态方法)	186	101	92	5 869
if_acmpne_w(比较跳转)	42	6	18	1 265

【注】 TINI 是用软件实现的,运行在增强型 8051 处理器上,20 MHz.

JavaCard 的应用对处理器性能的要求要远小于 Java 嵌入式应用,用软件虚拟机方式实现时,典型的内部时钟为 3.7MHz 就可满足要求,因此 AOJCP 可满足实际应用要求.

2　AOJCP 的实现结果

以 Xilinx XC2S400E 为目标做实现,整个系统(含外部 ROM,RAM)实现逻辑共占用1 410 LCs(约合 ASIC 的 16 k 门),存储占用 23 个 4 kbit block RAM,是目前最小的 JavaCard 硬件处理器实

现;最高频率 36.2 MHz. 使用 Xilinx XPower[9] 工具来估算 AOJCP 的功耗. XPower 分析时,使用仿真测试程序产生的 VCD(value change dump)文件,以获得准确的翻转频率参数. 不同频率下功耗分析的结果见表 5.

表 5 AOJCP 功耗表

Tab. 5 Power consumption of AOJCP

频率/MHz	功耗(sort.vcd)/mW	功耗(class_method.vcd)/mW
0	38	38
3.7	48	48
10	65	61
20	90	86
36.2	122	126
100	279	280

【注】 3.7 MHz 是典型的 JavaCard 时钟频率;36.2 MHz 是 AOJCP 支持的最高频率;100 MHz 是用于与 aJile-100 比较.

对比 aJile-100 的功耗:aJile-100 是用 ASIC 实现的商业 Java 处理器,它基于 aJile 的微码处理器核 JEM2,配合硬件陷入机制和相应的软件,可支持完整的 Java 虚拟机指令集,硬件实现上下文切换,支持 Java 的线程模型,可用于有实时要求的场合,但 aJile-100 比 AOJCP 要复杂,AOJCP 设计用于 JavaCard 系统,仅支持 JavaCard 指令集. 100 MHz 频率时,aJile-100 工作模式下功耗 257 mW;用同一工艺下的 FPGA 和 ASIC 实现同一设计,FPGA 功耗大约是 ASIC 实现的 4～8 倍. 考虑到 ASIC 与 FPGA 功耗的差别,用 ASIC 实现的 AOJCP 的功耗必将更低. 这里的比较仅有参考意义,因为不同工艺实现的 FPGA 芯片功耗参数也有很大差异.

3 结论

AOJCP 是一种基于微码、面积最小、低功耗设计的 JavaCard 处理器. 单周期与控制-应答机制结合的微码设计节省了资源,简化了主控逻辑的设计. 采用控制-应答机制,AOJCP 能方便地扩展硬件模块,目前已可选扩展了的 AES 硬件加、解密模块. AOJCP 面积小、功耗低,适于在 JavaCard 设备上使用. 将 AOJCP 与 JavaCard kit 连调成功后,作为系统平台,可广泛运行各种 JavaCard 应用.

参考文献(References)

[1] Sun developer network. Java card technology [S]. http://java.sun.com/products/javacard.

[2] Java card forum. General marketing presentation[EB/OL]. http://www.javacardforum.org/03_documents/00_documents/marketingpresentationgeneral.pdf.

[3] Golatowski F, Preuss S, Ploog H, et al. Integration of Java processor core JSM into smart devices [C]// Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation. Antibes Juan les Pins, France: IEEE Press, 2001: 699-702.

[4] Bose B, Tun M E. LavaCORE - a configurable Java processor[EB/OL]. http://www.xilinx.com/xcell/xl37/xcell37_20.pdf.

[5] Sun Microsystems Inc. PicoJava-II microprocessor core architecture data sheet[EB/OL]. http://www.sun.com/microelectronics/PicoJava/.

[6] Schoeberl M. JOP: a Java optimized processor for embedded real-time systems [D]. Vienna University of Technology, 2005.

[7] aJile Systems Inc. aJile systems: low-power direct-execution Java microprocessors for real-time and networked embedded applications[EB/OL]. http://www.ajile.com/downloads/aJile-white-paper.pdf.

[8] Degalahal V, Tuan T. Methodology for high level estimation of FPGA power consumption [C]// Proceedings of the 2005 Conference on Asia South Pacific Design Automation. Shanghai: ACM Press, 2005:657-660.

[9] Naoya S, Masakazu U, Yasuhiro T. Design techniques to reduce power consumption for ARM946E-S core[J]. Oki Tekunikaru Rebyu, 2005,72(3): 52-55.