第 34 卷 第 1 期 计 算 机 工 程 2008 年 1 月
Vol.34 No.1 Computer Engineering January 2008
·工程应用技术与实现· 文章编号：1000—3428(2008)01—0253—03 文献标识码：A 中图分类号：TP334

基于 FPGA 的 Java 处理器设计

南兆阔，须文波，柴志雷
（江南大学信息工程学院，无锡 214122）

摘 要：针对 Java 技术在嵌入式领域的广泛应用，设计了一个适用于低端嵌入式设备的 32 位环境的 Java 处理器 JPOR。该处理器由 FPGA 芯片实现，采用一种新的 Java 栈结构，指令系统简洁，可以直接执行 Java 字节码，能够对实时 Java 规范(RTSJ)提供有效支持。在 Xilinx SPARTAN-3 平台上通过了功能仿真，表明该 Java 处理器能够在低成本的 FPGA 芯片中实现。

关键词：Java 处理器；RTSJ；实时 Java 平台；栈帧

Design of Java Processor Based on FPGA

NAN Zhao-kuo, XU Wen-bo, CHAI Zhi-lei
(College of Information Engineering, Southern Yangtz University, Wuxi 214122)

【Abstract】 With the Java technology widespread applied in the embedded system domain, a Java processor with a 32-bit environment optimized For RTSJ is designed, and is suitable for low power embedded equipments. The processor is implemented by a FPGA chip, which has a new kind of Java stack structure and has a very simple instruction system. It can exeute the Java byte code directly, and provide effective support to the real-time specification for Java. It passes the function simulation in the Xilinx SPARTAN-3 platform, which shows that the Java processor can be implemented in the low cost FPGA chip.

【Key words】 Java processor; RTSJ; real-time Java platform; stack frame

Java处理器不仅能够利用硬件直接执行Java字节码，而且能够针对Java虚拟机的运行时结构提供专门的硬件支持(如栈操作、多线程管理等)，因此，能够支持实时Java规范(Real-Time Specification for Java, RTSJ)[1]，且具有高性能、低功耗、内存占用少等优点，更适合应用于嵌入式系统。但是较早的Java处理器如PicoJava/PicoJava-II[2]，主要是提高系统的整体性能，用户环境不支持RTSJ中的特性。而由维也纳理工大学Martin Schoeberl博士设计的JOP[3]，虽是一款优秀的嵌入式实时Java处理器，但它的用户环境也不支持RTSJ中的特性，且难于在低端的FPGA芯片中实现。目前，已经在复旦大学嵌入式系统实验室自行开发的计算机体系结构实验仪FD-MCES上实现了 16 位的Java实时处理器的试验平 台[4-5]。网络、通信、多媒体和信息家电的普及无疑为 32 位嵌入式系统高端应用提供了空前巨大的发展空间。在文献[5]的工作基础上由原来的 16 位Java处理器扩展为 32 位处理器JPOR，以增加数据计算和处理的速度和内存寻址空间，增强整体性能。

1 基于 JPOR 的 Java 平台

Java的实时性需要有合适的实时Java平台来支持，单纯Java处理器还不能构成完整平台。而Java平台是由Java虚拟机(JVM)和应用程序接口(API)组成的，为此把应用程序的执行分为初始化和任务执行 2 个阶段，所有非实时的操作，如类的装载、解析和连接、字节码的优化、静态变量空间分配等都在初始化阶段进行，应用程序用到的所有的类都在该阶段装载和初始化。基于JPOR的Java平台的模式如图 1 所示。标准Class文件(包括用户程序编 译后的Class以及Java库中的Class)在装载到嵌入式Java处理器执行之前，先由运行在PC上的Class文件预处理器Cconverter[5]进行预处理。然后把生成二进制格式的映像文件下载到JPOR目标板运行。

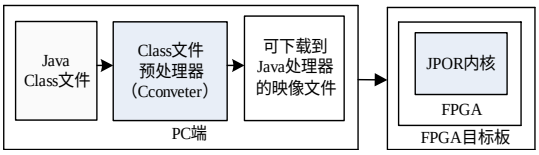

图 1 Java 平台

2 JPOR 的系统结构

2.1 JPOR 的特点

本文支持的JPOR用户环境是在RTSJ基础上的限制版本，提供对异常处理、异步控制转移机制的支持，并遵循类似CLDC1.0(Connected Limited Device Configuration)[6]所规定的限制，如：不支持浮点操作，不支持终结(finalization)，不支持Java本地方法接口(JNI)，不支持用户定义的类装载，不支持Long数据类型和浮点操作。同时，为了确保系统的实时性和最坏执行时间可预测(WCET)，JPOR处理器不提供对接口方法的直接支持，而是限制同一个接口只在一个类中实现，这样保证了接口方法在方法表中的偏移量是固定值。Cconverter 通过预处理提前计算出该偏移量并把invokeinterface替换为invokevirtual，从而保证方法调用的实时性。

2.2 JPOR 的栈帧

栈帧由 3 部分组成：局部变量区，操作数栈和帧数据区。局部变量区和操作数栈的大小在编译时确定并放在 Class 文

基金项目：江南大学青年科学基金资助项目“实时Java平台研究”(005067)
作者简介：南兆阔(1981－)，男，硕士研究生，主研方向：嵌入式系统；须文波，教授、博士生导师；柴志雷，博士研究生
收稿日期：2007-08-31 **E-mail：**zk_nan@163.com

件中。帧数据区的大小是和实现有关的。

(1)局部变量区：用来存放当前方法的局部变量可以通过局部变量指针访问。局部变量区被组织成一个以字为单位的、从 0 开始计数的数组。字节码指令通过从 0 开始的索引来访问局部变量。

(2)操作数栈：操作数栈保存了指令所需要的操作数及中间结果，功能类似普通处理器中的寄存器。操作数栈也被组织成一个以字为单位的数组。但数据的访问只能通过标准的栈操作(压栈和出栈)来进行。

(3)帧数据区：帧数据区中包含了用来支持常量池解析，方法正常返回以及异常处理的数据。

虚拟机每当启动一个方法时，JVM 都会为它分配一个 Java 栈，每个方法对应栈中唯一的栈帧，也即 Java 栈以帧为单位保存线程的运行状态当前的方法位于栈顶。因为本文中 Java 处理器可以直接执行字节码，所以 Java 栈同时也是本地方法栈。线程正在执行的方法称为该线程的当前方法，当前方法使用的栈帧称为当前帧。

如图 2 所示，当要调用新方法时，前一帧将需要传递的参数置于当前帧的操作数栈顶，被调用的方法生成新的栈帧置于栈顶，从前一帧的参数传递区获得方法的参数，并将当前的帧状态等信息保存在当前帧的帧数据区。Java 方法可通过 2 种方式完成：一种是通过 return 正常返回，另一种是通过抛出异常而中止。不管以哪种方式返回，虚拟机都会将当前帧弹出栈并释放掉，这样上一个方法的栈帧就成为当前帧了。

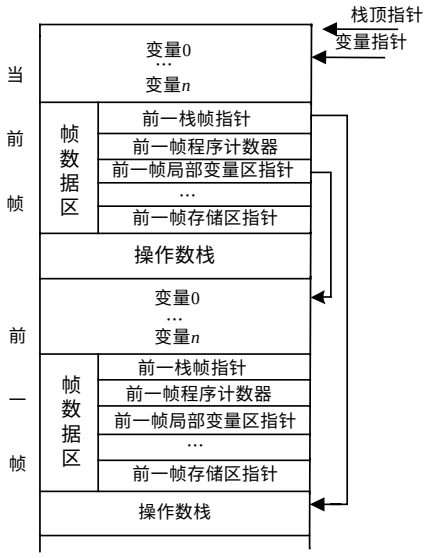

图 2　Java 的栈结构

2.3　JPOR 的指令系统

在 JVM 指令集中，各条指令有很大的差异，既有一些简单的，如算术和逻辑运算指令，也有一些复杂的指令，如 new 和 involestatic 等这样的类装载和校验指令。由于这种差异，并非所有 JVM 指令都可以通过硬件来实现，在 Sun 公司的 Java 处理器 picojava-Ⅱ中使用的一种方法是用硬件来实现一个指令子集，剩下的复杂指令通过软件陷阱调用来实现。

JPOR在这个问题的处理上采用了不同的方案[7]，JPOR的简单的指令如算术逻辑指令适合标准JVM一一对应的，可以在一个时钟周期来完成。对于较为复杂的指令通过一序列的JPOR指令来进行模拟运算执行，它的每一个复杂指令都对应这一段微代码。

2.4　JPOR 的内核

JPOR 的内核组成部分可以分为：取字节码，取指令，译码，执行 4 个部件。如图 3 所示。

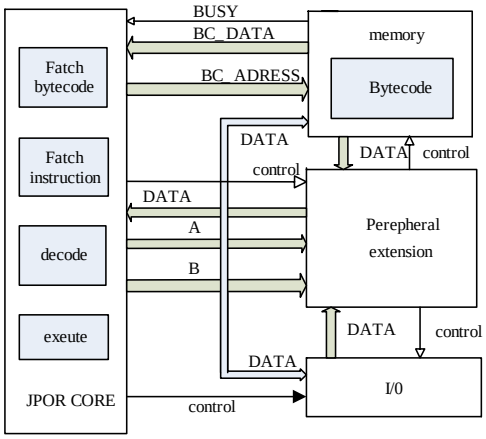

图 3　JPOR 内部框图

(1)取字节码：为了减少取指令时内存访问次数，JPOR 处理器中实现了一个取指令 FIFO(IRSH 寄存器组)，它可以同时存放多个字节码。相邻的字节码被一次性取到 IRSH 中。

(2)译码操作及控制部件：处理器的控制部件经过取指令操作以后，待译码的指令在 IRSH 的高 8 位中，作为控制部件的输入，经过控制部件译码后产生控制信号。为了在低成本，现场可编程门阵列(FPGA)芯片中实现该处理器时提供一定的指令集可重构特性。译码方式采用了类似微程序的方式，在面向不同应用时可以方便地把需要的指令逐条添加进来。同时，该译码部件直接由硬件逻辑在芯片内部构造，译码速度可以和普通的硬接线方式相比。

Java 虚拟机的操作是栈方式的，取栈顶的元素进行操作并把执行结果又压入栈。在 JPOR 处理器中，使用寄存器 A、B 来表示栈顶的两个元素，同时 A、B 为 ALU 提供操作数，计算结果可以保存到指定的寄存器中。对栈操作如下：

(3)压栈操作：(SP)+1 => SP, (B) => stack[SP], (A) => B, MEM[ADDR]=> A。从内存中读一个字压入栈中。最后栈顶的元素在 A 中。

(4)出栈操作：(A)=>MEM[ADDR]; (B)=>A; stack[SP]=>B; (SP)-1=>SP。从栈中弹出一个字保存到内存中。实际是 A(栈顶)中的数据保存到内存中了。

3　实验平台及实现情况

FPGA 是一种可编程逻辑器件，它是一种半定制的通用性器件，用户可以通过对 FPGA 进行编程来实现所需的逻辑功能。与专用集成电路 ASIC 相比，FPGA 具有灵活性高、设计周期短、成本低、风险小等优势，因此得到了广泛的应用。

本文试验开发环境为SPARTAN-3 Starter Board[8]。实现平台如图 4 所示，在文献[4]中已经做了 16 位指令系统，本文在此平台上实现了 32 位指令系统的升级与扩展，并通过高速串口与PC交互。采用了硬件描述语言VHDL设计的JPOR内核及其外围控制，经过编译、综合、布局布线等步骤最后生成了一个.bit目标文件，把该.bit目标文件通过串行下载线下载到目标板的实验芯片中，该实验芯片就成为了一个Java处理器芯片，并完成了其内核和外围控制各个模块的功能仿真测试，以及内核整体的功能仿真测试，指令执行的部分仿真结

果如表 1，部分仿真波形如图 5。

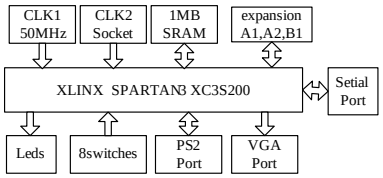

图 4　Spartan 3 实现平台

表 1　部分仿真结果

字节码	JPOR 执行周期
iaload	4
aload	4
invoke	45
Invoke static	42
new	12
return	21
getfield	9
astore_x	7
iconst_x	3
goto	7

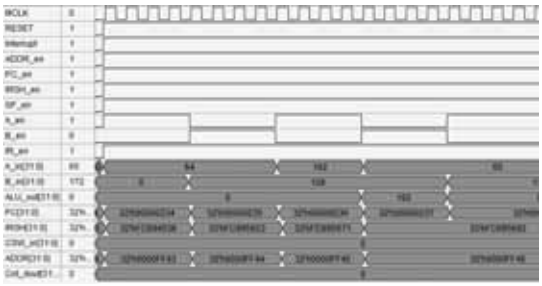

图 5　部分仿真波形界面

4　结束语

本文设计出了一个适用于低端嵌入式设备的实时 32 位环境的 Java 处理器实现模型。该实现模型是一个完整的 32 位环境，包括对 Java Class 文件的装载和预处理以及 Java 字节码的硬件执行，通过分阶段处理来保证运行时的实时性，能够对实时 Java 规范(RTSJ)提供有效支持。同时，通过对 Java Class 文件的预处理 Cconverter 来简化处理器硬件实现。该处理器结构简单，能够直接执行 Java 字节码，能够在低成本的 FPGA 芯片中实现，具备一定的可扩展性，且可以逐步添加字节码。

参考文献

[1] Gosling J, Joy B, Steele G, et al. The Java Language Specification Second Edition[M]. Santa Clara, California: Sun Microsystems, Inc., 2005.

[2] McGhan H, O'Connor M. PicoJava: A Direct Execution Engine for Java Bytecode[J]. Computer, 1998, 31(10): 22 -30.

[3] Schoeberl M. JOP: A Java Optimized Processor for Embedded Real-time Systems[EB/OL]. (2005-03-02). http://www.jopdesign.com.

[4] 柴志雷. Java 实时性及嵌入式实时 Java 处理器研究[D]. 上海: 复旦大学, 2006.

[5] 柴志雷，高丽强，陈章龙，等. 一种用于硬实时 Java 处理器的类转换器设计及实现[J]. 小型微型计算机系统，2006，12(27): 2336-2340.

[6] Spartan-3 Starter Kit Board User Guide[Z]. (2005-05-13). http://www.digilentinc.com/Data/Products/S3BOARD/S3BOARD-rm.pdf.

[7] 杨　骥，毛　峡. 一种堆栈型 Java 处理器的流水线设计[J]. 计算机工程与设计，2004, 25(12): 2357-2359.

[8] Connected Limited Device Configuration(CLDC)[Z]. (2007-01-04). http://java.sun.com/products/cldc/.

（上接第 246 页）

时间间隔。这是因为 MINSATS 在检测到错误时会中止此流程，停止后续信令的发送，若 SCP 在等待信令 20 s 之后，仍未收到，则视为超时，从而中止此次对话。而且，MINSATS 会在此 20 s 的时间间隔之内删除 SCP 发来的任何信令，保证不对下一流程造成影响。若需要验证数据库数据，还要确保此流程已经将数据永久地写入数据库之后再进行验证，这也需要几秒(暂定为 5 s)的时间间隔。将这 2 部分所需时间之和记为 30 s 已经足够。若只测试一个流程，则此 30 s 可以减少至 5 s。

由表 1 可以看出，在使用仿真 SSF 或 CMIN02-SCE 时，测试单个信令流程分别需要 15 min 和 12 min，而使用 MINSATS 时，测试单个信令流程仅需 5.5 min，测试时间分别节省了 63%和 54%。

对于单个业务的功能完备性测试，假设此业务对应的信令流程总数为 120，使用仿真 SSF 对所有流程进行测试所需的时间记为 $T1$，使用 CMIN02-SCE 测试所需的时间记为 $T2$，使用 MINSATS 测试所需的时间记为 $T3$，由表 1 可得：

$T1$=5+5+(2+3)×120=610 min

$T2$=5+3+(2+2)×120=488 min

$T3$=5+0.5×120=65 min

与仿真 SSF 和 CMIN02-SCE 相比，MINSATS 的整体测试时间分别减少了 89%和 87%。

这里，$T3$ 仅是指在工作时间进行的测试，若后台运行，则测试时间只有搭建测试环境所需的 5 min，减少了 99%。

5　结束语

MINSATS的实施使测试时由人工校验信令参数、数据库数据，改为由计算机自动检验，这是以往智能网测试工具所不具备的。而且，MINSATS提供的批量测试的功能，可以极大地提高测试效率、降低开发成本、提高软件质量。同时，MINSATS还需要做大量的增强，如增加支持大话务量测试的功能。虽然MINSATS在多进程运行时，发向SCP的呼叫数可以达到 10 CAPS[5]左右，但是，对于多数业务，这还远不能达到现网的实际要求。在增加了支持大话务量测试的功能后，MINSATS就完全可完成SCP上的功能测试和压力测试。

参考文献

[1] ETSI. GSM 09.78 v7.0.0 CAMEL Application Part(CAP) Specification[S]. 1999-08.

[2] 廖建新，王　晶，郭　力. 移动智能网[M]. 北京: 北京邮电大学出版社, 2000-11.

[3] ETSI. GSM 09.02 v5.6.0 Mobile Application Part(MAP) Specification[S]. 1997-08.

[4] 朱晓民，温　瑜，廖建新，等. SCE 研究进展及其在移动智能网中的实现[J]. 计算机工程, 2005, 31(6): 1-3.

[5] 武家春，刘　川. 移动智能网中 SCP 的性能测试[J]. 电信科学, 2004, 20(2): 45-47.

[6] ITU-T Rec.. Q.1201 Principles of Intelligent Network Architecture[S]. 1992-10.

[7] 桂海源，骆亚国. No.7 信令系统[M]. 北京: 北京邮电大学出版社, 1999-03.