Java 硬件实现技术现状及发展

Technology of Java-on-Silicon

桑红石　沈绪榜

（华中理工大学图像识别与人工智能研究所　武汉 430074）

Abstract　Java is architecture neutral, implementation independent and secure. Java is a net oriented language and has been used on network extensively. The technology of Java-on-silicon is introduced in this paper. Java-on-silicon supports Java by hardware. This technology can enhance the speed of Java program. Up to now, there are three main approaches. The first approach is designing microprocessor with architecture just like JVM. The second approach is using a coprocessor to interpret Java codebytes. The third approach is designing parallel processor to support Java's multithread feature. This paper introduces these approaches of Java-on silicon, and discusses the trend of this technology.

Keywords　Java, Java-on-silicon, Microprocessor, Architecture, Embedded device

1. 引言

Java 是面向网络应用的语言、具有平台独立性、处理器兼容性、分布性和安全性等特点；Java 采用字节码作为中间代码，程序代码短小精悍。由于这些特点 Java 十分适于网络应用，已经成为一种网络语言。Java 语言为了实现上述优点，付出的代价是程序运行的时间开销大大增加，执行速度很慢，这是影响 Java 进一步广泛应用的主要原因（有关 Java 执行机制的详细描述见参考文献[1～4,7]。因此，在 Java 语言推广的同时，就开始研究各种提高 Java 运行速度的方法，这些方法概括起来有软件和硬件两条途径。

以软件提高 Java 执行速度的主要努力是以编译执行代替 Java 的解释执行方式。Java 在 Java 虚拟机 JVM（Java Virtual Machine）上运行的标准方式是解释执行，耗费的时间是编译执行的 C 和 C++程序的 15 到 20 倍。由于 Java 的动态特性[5]，Java 程序不可能采用类似 C 程序的做法直接编译生成执行代码。到目前为止，Java 编译技术中效果最好的是及时编译技术 JIT（Just-In-Time）[2,6]。及时编译技术在 Java 程序执行过程中把 Java 字节码替换成处理器的机器码，同一个方法只需要在第一次装入时翻译一次。在程序中多次调用相同方法的情况下，使用 JIT 技术能够显著提高程序的执行速度，甚至可以接近 C 程序的执行速度。JIT 也有其不利的一面：其一，和解释器相比，JIT 将机器码保存在内存中，需要占用更多的存储器容量；其二，解释器性能较为稳定，编译程序的优化部分则容易出错，导致安全隐患[14]。

以硬件支持 Java 语言的方式就是本文讨论的 Java 实现技术。Java 硬件实现技术是设计支持 Java 的微处理器，也称为 Java 芯片或 Java-on-silicon 技术。目前在该技术上的研究主要有以下三方面：

（1）仿照 JVM 的结构设计 Java 专用处理器，以 Java 字节码作为处理器的指令集合；或者设计类似 JVM 的堆栈型处理器，尽量减少翻译、优化 Java 字节码的开销。

（2）设计以协处理器方式工作的 Java 字节码硬件解释器，将字节码对应的机器码固化在存储器中。协处理器读入字节码，将相应的机器码送到处理器执行。

（3）以超长指令字 VLIW（Very Long Instruction Word）结构的并行微处理器支持 Java 程序的线程级并行。处理器内部的功能模块采用 RISC 结构，以求实现处理器的高性能。

下面将分三部分分别介绍 Java 实现技术，并分析各种技术的特点。

2. 支持 Java 的堆栈结构处理器

采用堆栈结构以支持 Java 的微处理器有 Sun 公司的基于 picoJavaII 核的 microJava701 和 Patriot Scientific 公司的 PSC1000 等。这种结构处理器需要解决

桑红石　博士，主要从事嵌入式计算机（MPP）、RISC 处理器的设计以及 Java 技术研究。**沈绪榜**　中科院院士，长期从事嵌入式计算机设计及其芯片实现工作。

的共同问题是尽量消除堆栈对处理器性能的限制。

2.1 microJava701 的系统结构

microJava701 中包括一个 picoJavaII 微处理器核。picoJavaII 微处理器核采用六级 RISC 流水线，具有向前指令合并(forward instruction folding)功能。picoJavaII 核包括六个基本单元模块：整数单元，浮点单元、指令 cache、数据 cache、堆栈管理单元和总线接口单元。picoJavaII 核支持 Java 虚拟机的运行环境。它的特点有多线程同步、多种垃圾回收方法、方法调用、本地变量隐含调用等。

picoJavaII 核的框图如图 1 所示。picoJavaII 核既可用于执行 Java 虚拟机中定义的 Java 字节码，也可用于执行 C/C++代码。picoJavaII 核直接执行大部分常用的 Java 字节码，较为复杂的字节码以微代码的方式执行，最复杂的字节码产生陷阱由软件仿真执行。除了标准字节码，picoJavaII 核还扩展了一些字节码，用于进行直接地址读写、cache 管理和访问内部寄存器，这些指令由操作系统使用，用户不能使用。picoJavaII 核中的指令合并功能可以加速栈顶操作，指令的装载和执行能够在单周期内完成。

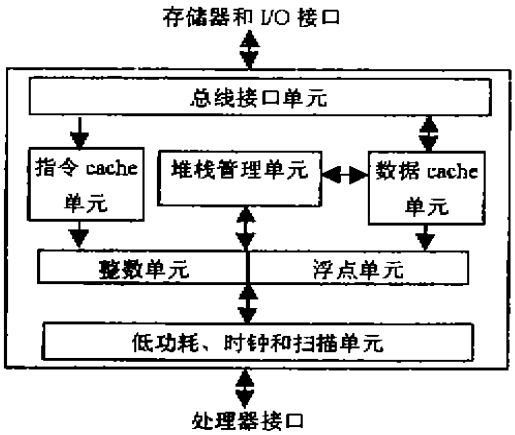

图 1 PicoJavaII 处理器核

picoJavaII 核中的堆栈管理单元包括堆栈 cache、堆栈控制器和传送(dribble)控制器。堆栈 cache 有 64 项，提供 3 个读出口和两个写入口。堆栈控制器为整数单元提供检索操作数的控制信号，也负责控制数据在数据 cache 和堆栈 cache 之间的移动。传送控制器负责在堆栈发生上溢或下溢时在堆栈和存储器之间传送数据。堆栈 cache 中有一个传送控制器专用的读-写口[3]。

Sun 公司开发 Java 芯片是为了提高 Java 程序的执行速度，面向的是嵌入式产品市场。嵌入式产品要求低成本、低功耗，在此基础上再提供令人满意的性能。microJava701 采用堆栈结构支持 Java 字节码，而堆栈又成为程序运行的瓶颈。为了解决这个矛盾，microJava701 采用了复杂的流水线技术和堆栈自动管理技术，其片上 cache 及其控制逻辑也占用了较大的硅片面积，由此导致成本和功耗增加，使 microJava701 芯片运行时功耗高达 3 瓦。因此，尽管 microJava701 能够直接执行 Java 字节码，省去了软件解释或翻译字节码的开销，却牺牲了处理器的性能，且成本高，无法满足用户的要求[3~11,14,15]。

2.2 PSC1000 芯片的系统结构

PSC1000 处理器也是面向嵌入式应用，在设计时努力实现低功耗和低成本，为了节省面积，PSC1000 没有采用流水线和超标量设计，也没有采用指令和数据 cache。因为结构简单紧凑，处理器可以运行在很高的时钟频率上(PSC1000 为 66MHz，PSC1000A 为 100MHz)，同样能够实现较高的性能。PSC1000 采用堆栈结构，大部分 Java 字节码和 PSC1000 的机器指令有对应关系。配合 PSC1000 使用的 JIT 程序只需 20k 字节，而一般的处理器 JIT 程序需要 200k 字节，因此，PSC1000 对 Java 能够提供较好的支持。

PSC1000CPU 包含 10 个主要部件，分别是 32 位微处理器 MPU(MicroProcessing Unit)、虚拟外设单元 VPU(Virtual Peripheral Unit)，全局寄存器、直接存储器访问控制器 DMAC(Direct Memory Access Controller)、中断控制器 INTC(INTerrupt Controller)、片上资源、位输入、位输出、可编程存储器接口 MIF(Memory InterFace)和时钟。PSC1000 处理器的结构框图如图 2 所示。

PSC1000 设计时着重考虑降低系统功耗和成本，其结构有以下特点：

- PSC1000CPU 的指令执行采用硬联控制，速度较快，其特有的“直流”(flow-through)方式允许在上一条指令执行完之前就启动执行下一条指令，因此大多数指令可以在单周期内完成。对于较复杂运算，如乘加、除法和浮点运算，采用多周期完成，由于 CPU 的时钟频率高，复杂运算同样可以较快完成。PSC1000 中没有设置桶式移位器，移位操作采用“智能移位”方式，先按字节移动，再按位移动，以减少移位操作的周期数。

- PSC1000 采用零操作数结构，与 Java 字节码相同，单个指令为一字节，PSC1000 可以同时读出 32 位指令数据，其中最多可以包含 4 条指令，与采用 32 位指令格式的 RISC 处理器相比，相当于提高了存储器的带宽。PSC1000 设有一个 4 字节的指令缓冲器，暂存预取指令，隐含了取指时间，使取指不会影响指令的执行。

- 为了节省芯片面积，PSC1000 充分利用片上资源，实现资源共享。例如：MIF 由 MPU、VPU、DMAC

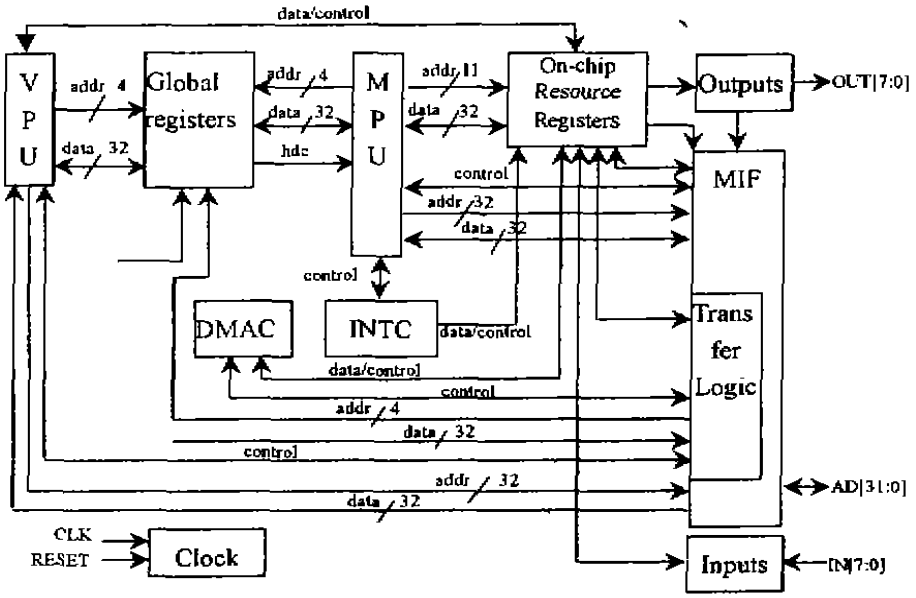

图 2　PSC1000 框图

和位输入、输出共享；全局寄存器由 MPU、VPU、和 MIF 中的传送逻辑共享，等等。

虽然 PSC1000 处理器采用堆栈结构，同样存在堆栈瓶颈的问题，与 microJava701 不同的是，其提高系统性能的方法，是以简单的结构实现较短的时钟周期，因此具有较高的性价比，适合于嵌入式应用，在目前的嵌入式产品低端市场上占据了较大的份额。PSC1000 是以堆栈结构提高 Java 执行速度的一个成功的例子[12~16]。

3. Java 硬件解释器

以协处理器方式工作的 Java 字节码硬件解释器有 Jedi Technologies 公司提出的 JStar 技术。该协处理器可以加入到 RISC 或 CISC 处理器环境中，将 Java 字节码翻译为处理器的机器码。协处理器把字节码对应的处理器机器码固化在存储器中，同时与需要加速的处理器核以及处理器的 cache(或存储器)接口。JStar 读取字节码，检索对应的机器指令，再送入处理器中执行。对处理器而言，相当于直接执行 Java 字节码。JStar 具有较强的扩展性，可以和任何 RISC 或 CISC 处理器配合使用，包括采用 VLIW 或多发射结构的处理器。加入 JStar 不必改动原处理器的指令集合和流水线。JStar 技术可以将 Java 执行速度提高 6 倍[17]。

4. 支持 Java 的并行处理器

采用 VLIW 结构以支持 Java 程序的并行性是 1999 年至今出现的一种新方案，属于这一范畴的有 Sun 公司的 MAJC 结构、IBM 提出的 VLIW 结构等。

MAJC 是 Sun 公司于 1999 年提出的针对 Java Computing 的多处理器系统结构。Java Computing 是指一种基于网络的客户-服务器模型的计算方式，其目的是将计算技术的复杂性由以桌面系统为中心转向以服务器为中心，由服务器进行集中的、专业化的管理。Java 语言因为其平台独立性，适合于实现网络中心的计算方式，因此 Sun 公司称这种计算模式为 Java Computing[18,19]。

MAJC 采用片上多处理器集成技术，旨在开发线程级并行性，以求应用于实时处理声音、图像等大数据流的情况。

MAJC 处理器群(芯片)的结构如图 3 所示。MAJC 为三层结构，最高层为处理器群，其次是处理器，第三层是功能模块。一个芯片为一个处理器群，其中集成了多个处理器；每个处理器包括 1 到 4 个功能模块，每个功能模块相当于一个带有 DSP 功能的 RISC 处理器。每一个功能模块拥有自己的私有寄存器堆、局部指令译码逻辑和状态信息，还有局部布线，这些不与同一处理器中的其它功能模块共享。一个处理器中的所有功能模块可以共享一个全局寄存器堆，该寄存器堆的大小根据需要确定，变化范围从 32 到 512。该全局寄存器堆不能由同一处理器群中的其它处理器访问。每条指令都是发送到某一个功能模块上执行的，一个处理器单元单周期内可以处理的指令数取决于当时能够使用的功能模块的数目。

Sun 公司研究发现，程序中指令级的并行性不超

出4条指令的范围，因此MAJC结构中每个处理器的功能模块下超过4个，MAJC多处理器结构的目的是抽取更高级的线程级并行性，进而支持Java程序的多线程特性[20-22]。

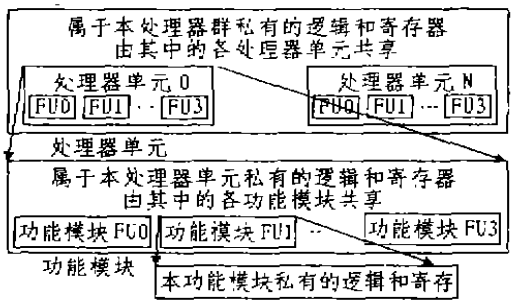

图3 MAJC处理器群的结构

IBM于2000年3月宣布研制支持Java的VLIW结构的微处理器，与MAJC类似，内部采用指令级并行的RISC结构，通过快速编译的方法将Java字节码转换为该处理器的机器码[24]。

目前实用化的VLIW结构的处理器还只限于专用媒体数据处理器，限制其广泛应用的关键原因是该结构微处理器对已有软件目标代码的兼容性较差。如何设计有效的软件翻译器是推广VLIW结构处理器必须解决的问题[24]。

结束语 从Java实现技术的发展过程来看，为了提高Java的执行速度，采用堆栈结构有一定的局限性。堆栈结构微处理器很难达到较高的性能，因此必须尽力降低芯片面积和功耗，以求应用于机顶盒、蜂窝电话、智能卡等要求低成本、低功耗的产品中。高性能的通用微处理器，如Intel公司的Strongarm系列产品，采用RISC系统结构，借助于及时编译技术，同样能够为运行Java程序提供较好的效果，可作为高端嵌入式产品，应用于网络计算机等产品中[25,26]。

设计支持Java的并行处理器是将Java实现技术向高性能处理器的方向发展，其内部基本处理单元一般采用RISC结构，以求提高性能并减少控制逻辑的复杂性。在Java实现技术未来的发展方向中，支持Java的运行环境，减少JVM的开销，是值得探索的途径。支持Java的运行环境包括：支持多线程并行、实现快速现场切换、支持垃圾回收以及支持字节码的安全限制等。目前Sun公司的MAJC系统结构和IBM的VLIW系统结构的微处理器就是向这个方向发展的。

参考文献

1 王克宏，等. Java虚拟机规范 清华大学出版社，1996
2 刘有军译. Visual J++1.1揭密. 电子工业出版社，1998
3 Java Virtual Machine Specification. Available at, http // www. sun. com, SUN Microsystems, Inc, 1997
4 Is Java Too Slow? 1999
5 Solving Java Performance Issues. Intel® WEB99 Mega Tech sessions, 2000
6 Java JIT compile overview. Sun Microsystems, Inc, 1994
7 Java Performance Tuning Available at: http: // patrick. net/jpt, 1999
8 picoJava-I Data Sheet. Sun Microsystems, Inc, 1998
9 Kanellos M. Java chip not picking up steam. Available at, http. // www. cnetinvestor. com/splits, 1998
10 Niccolai J. Analyst: Sun scraps Java chip plans. Available at, http. // www. javaworld. com/javaworld/jw-11-1998, 1998
11 McCarthy J. Sun to take wraps off Java-appliances chip. Available at: http: // www. cnn. con, 1999
12 PSC1000/A Data Sheet. Available at. // www ptsc. com, 1995
13 Wolfe A Patriot takes a lead with its Java-specific processor. Available at: http: // edtn. con/news/mar27
14 Shaw G. Architecture is key to execution in Java. Available at, http: // www. ptsc. com, 1997
15 Java on Patriot's PSC1000 Microprocessor. Available at: http: // www. ptsc. com, 1997
16 The challenge of Java chips. Available at: http: // eetimes. com/columns/wolfes-den, 1998
17 Dennis S. Java Acceleration Device to Increase Speed 6-Fold. Available at: http: // www. newsbytes. com
18 Java Computing: What it Means for the General Manager and CIO. Sun Microsystems, Inc, 1994
19 Tribble B. Java Computing in the Enterprise. Sun Microsystems, Inc, 1996
20 MJAC Architecture Tutorial. Sun Microsystems, Inc, 1999
21 Wade W. Sun Conjures Java CPU For Media Apps. Available at: http: // www. techweb. com/wire/story, 1999-8-2
22 Sun Discloses Details of MAJC Architecture. Available at: http: // www. sun. com/microelectronics/MAJC, 1999-8-17
23 Sun Microsystems Announces the First Implementation of the MAJC Architecture-the Dual Processor MAJC5200. Available at: http: // www. sun. com/microelectronics/ MAJC, 1999-10-6
24 Wolfe A IBM plans VLIW chip that runs Java code. Available at: http: /www. eetimes. com, 2000-3-25
25 SA-1110 Processor. Available at: http: // www. developer. intel. com
26 Intel Strong ARM SA-1100 Microprocessor for Embedded Application. Available at: http: // www. intel. com, 1999