

**研究生学位论文开题报告**

**报告题目 基于SPARC模拟器的多核调试技术研究与实现**

**学生姓名 李戈 学号 2014E8013261153**

**指导教师 唐志敏 职称 研究员**

**学位类别 工学硕士**

**学科专业 计算机技术**

**研究方向 计算机体系结构**

**培养单位 中国科学院计算技术研究所**

**填表日期 2016.08.15**

**中国科学院大学制**

**填 表 说 明**

1. 本表内容须真实、完整、准确。
2. “学位类别”名称填写：哲学博士、教育学博士、理学博士、工学博士、农学博士、医学博士、管理学博士，哲学硕士、经济学硕士、法学硕士、教育学硕士、文学硕士、理学硕士、工学硕士、农学硕士、医学硕士、管理学硕士等。
3. “学科专业”名称填写： “二级学科”全称。

**报告提纲**

1. 选题的背景及意义
2. 国内外本学科领域的发展现状与趋势
3. 课题主要研究内容、预期目标
4. 拟采用的研究方法、技术路线、实验方案及其可行性分析
5. 已有科研基础与所需的科研条件
6. 研究工作计划与进度安排
7. 参考文献

目录

[1 本文的研究背景和意义 6](#_Toc460794608)

[1.1 嵌入式多核调试技术 6](#_Toc460794609)

[1.2 基于SPARC架构模拟器的多核调试 6](#_Toc460794610)

[2 国内外本学科领域的发展现状与趋势 7](#_Toc460794611)

[2.1 常用多核调试技术 7](#_Toc460794612)

[2.1.1 硬件调试 7](#_Toc460794613)

[2.1.2 软件调试 8](#_Toc460794614)

[2.1.3 常用技术对比 9](#_Toc460794615)

[2.2 SPARC架构模拟器 10](#_Toc460794616)

[2.2.1模拟器框架结构 10](#_Toc460794617)

[2.2.2 BM3803处理器核 10](#_Toc460794618)

[3 课题主要研究内容、预期目标 11](#_Toc460794619)

[3.1 主要研究内容 11](#_Toc460794620)

[3.2 预期目标 11](#_Toc460794621)

[4 拟采用的研究方法和技术路线 11](#_Toc460794622)

[4.1 可扩展的多核调试协议的交互机制研究 12](#_Toc460794623)

[4.1.1 RSP协议 12](#_Toc460794624)

[4.1.2 Flex集成以及接口设计 13](#_Toc460794625)

[4.2同步多核调试机制。 14](#_Toc460794626)

[4.2.1 断点检测机制 14](#_Toc460794627)

[4.2.2 “同启同停”调试机制 15](#_Toc460794628)

[4.3 异步多核调试机制 16](#_Toc460794629)

[4.3.1 调试事件处理机制 17](#_Toc460794630)

[4.3.2 异步调试功能 17](#_Toc460794631)

[4.3.3 模拟器处理流程 18](#_Toc460794632)

[5 已有科研基础与所需的科研条件 18](#_Toc460794633)

[5.1 已有科研基础 18](#_Toc460794634)

[5.2 所需的科研条件 19](#_Toc460794635)

[6 研究工作计划与进度安排 19](#_Toc460794636)

[7 参考文献 19](#_Toc460794637)

# 1 本文的研究背景和意义

## 1.1 嵌入式多核调试技术

嵌入式技术已被广泛应用于科学研究、工程设计、军事技术等各个领域，成为当前 IT 领域中炙手可热的技术。而随着近年来计算机硬件水平的高速发展和用户应用需求的提高，基于多核处理器的嵌入式系统以其高性能、低功耗等优势，在当前嵌入式系统中得到了广泛的应用[1-3]。基于多核处理器芯片的开发在提高了计算机运算速度的同时也为嵌入式系统的开发带来了很大的困难和挑战[4]。在嵌入式系统的开发过程中，系统开发平台是最基本的，而其本身的实现也是难度最大的，因为开发平台中涉及到的编译器、调试器等工具，都与目标板的体系结构相关。因此，设计实现一套良好的基于多核处理器的平台工具，便成为了嵌入式系统开发中的重点以及难点。其中好的调试器，可以使得那些需要运行在多核上的应用程序的实现更简单快捷，加快整个嵌入式系统的开发速度。

在硬件处理能力不断增强的过程中，嵌入式系统软件的调试手段也在经历着一系列的变迁，原来的调试手段已经逐渐退出历史舞台，新的调试手段也在不断涌现。在嵌入式系统软件调试发展过程中，主要使用的调试手段[5]有以下几种：ROM Monitor方式，Debug Stub/Server，在线仿真（In-Circuit Emulation，ICE）方式，片上调试（On-Chip Debug，OCD）方式，以及模拟器调试。由于嵌入式系统往往是资源有限的计算机系统，CPU 处理速度慢、自带存储空间小、没有人机交互设备等都给嵌入式系统软件调试带来了很大阻碍。调试手段的好坏直接影响到软件的开发效率以及软件的质量。为此，使用一套操作方便、安全性好、可靠性高及性能完善的调试工具对嵌入式系统软件开发人员来说非常重要[6]。

## 1.2 基于SPARC架构模拟器的多核调试

在航天领域中，要求航天器中的计算机系统(星载计算机系统)[7]可以在特殊的环境下高效完成复杂应用的处理，同时还要控制成本。目前，在世界范围内星载计算机系统中所使用的处理器架构只有两种，一种是由美国使用的POWERPC架构，另一种就是欧洲主导的SPARC架构。处于战略安全问题的考虑，中国星载计算机系统大多采用开源的SPARC架构。因此，完成基于SPARC架构的应用开发和调试是十分必要的。

模拟器通过软件程序模拟实际硬件的执行过程，可以完整模拟硬件的体系结构，在目标机编译环境下编译后的应用程序可以不加修改的在模拟器上运行。

与硬件调试相比，模拟器不需要特殊硬件的支持，节省开发成本；同时，在设计新系统时，需要更改现有架构，没有完全符合要求的硬件作为目标机，此时采用基于模拟器的调试方法，可以在研发初期完成对系统的调试，保证设计的正确性。

GDB提供了GDBServer和Stub两种常用软件调试方法。GDBServer是轻量级调试程序，运行在目标机的操作系统上，通过系统调用访问被调试程序的地址空间和寄存器等信息；Stub在生成可执行文件时将调试代理编译到被调试程序中，在没有操作系统的机器上，需要修改串口驱动程序和异常处理，是一种侵入式的调试方式。与这两种调试方式相比，基于模拟器的调试直接在模拟器内部实现了调试的功能，不需要操作系统的支持，在开发初期不具备操作系统的情况下也可以尽早实现调试功能；此外，模拟器调试不具有侵入性，不需要修改被调试程序，实现调试功能与应用程序的并行开发。

目前的模拟平台是基于SPARC架构的全新多核模拟平台，采用了模块化的设计思想，将各个模拟模块注册到模拟器框架中执行。在现有平台中，没有支持对应用程序的多核调试，在其他相关研究中，也只有对SPARC架构模拟器的单核调试功能的研究[6]。

因此，在现有多核模拟平台上，研究与实现多核程序调试，对于加快模拟系统的开发和调试具有十分重要的意义。

本文中模拟器的处理器核采用了BM3803MG[8]，BM3803MG 是基于 SPARC V8 体系结构的 32 位 RISC 嵌入式处理器, 可用于板上嵌入式实时计算机系统,能够满足各种航天应用的功能以及性能指标要求,只要加上存储器和与应用相关的外围电路,就可以构成完整的单板计算机系统。

GDB 是一款开放性的源码调试器，同时高版本的GDB(7.0以后)还增加了对多核调试的支持，因此选用了GDB作为宿主机端调试器，并针对调试协议进行模拟器多核调试的研究。

# 2 国内外本学科领域的发展现状与趋势

## 2.1 常用多核调试技术

在嵌入式系统中，受限于目标机的性能、功耗等问题，绝大部分调试都采用远程调试。而对于定位为航天应用的计算机系统，必然需要采用远程调试的策略。

常用的远程多核调试策略分为硬件调试和软件调试两大类。

### 2.1.1 硬件调试

硬件调试方法，一般需要在宿主机和目标机之间通过特殊硬件来连接，如JTAG、EJTAG、Trace 等，通过使处理器进入调试态来查看处理器核的内部运行情况。硬件调试器通常用于较底层的系统开发过程中的调试，如开发Bootloader程序，操作系统内核的开发或移植过程，或者设备驱动程序的开发等。硬件调试方法优点是：速度较快，并且因为采用了外部调试硬件，所以调试程序本身不占用目标机的系统资源，调试环境和最终的程序运行环境基本一致，不改变程序执行的行为，因此具有低侵入性。缺点是：由于调试硬件价格较高，不利于团队的整体开发，芯片面积和功耗都有所增加。常用的硬件调试方法有：

1. ICE (In Circuit Emulator)

在线仿真器ICE，是一种完全仿造调试目标CPU设计的设备，可以执行目标机 CPU 的指令，但ICE的CPU引脚线比目标机实际CPU的引脚线多，用于将内部信号输出到目标机上[9]。目标机对用户来说是完全透明的、可控的。在调试时，目标机与ICE之间用仿真头连接，ICE与宿主机之间用串口、并口或以太网口等连接。ICE可以真正运行所有目标 CPU 的动作，ICE上的内存也可以被映射到用户的程序空间，从而给调试过程带来很多便利。这样在采用 ICE调试时可以连接目标机，也可以不连接目标机。宿主机端运行的调试器通过 ICE来控制运行在目标机上的被调试程序。

EmbeddedICE-RT就采用了ICE的调试方式。EmbeddedICE-RT是ARM公司提出的基于ARM7以上CPU的调试标准，关键技术是在处理器芯片内部加入了EmbeddedICE宏单元，完成调试功能。

1. OCD (On-Chip Debug)

片上调试的设计思想是，在处理器设计初期就考虑了调试的需求[10]，在芯片内部加入专门的调试处理逻辑。根据研究，绝大部分用户仅仅使用了一些基本的调试功能,如断点、单步、处理器访问、内存访问等， OCD 将实时跟踪与运行控制分开来，将运行控制放到目标机系统的 CPU 核内由专门的调试控制逻辑模块来实现，并用一个专用的串行信号接口开放给用户。采用这种方式宿主机端的调试器可直接向目标机发送命令，并读写目标机的内存和寄存器，控制目标程序的运行。OCD技术的体系结构非常多,包括PowerPC、MIPS、ARM、x86等,这中间绝大部分体系结构的调试访问标准都是JTAG[10]。

### 软件调试

软件调试方法，通过在目标机上植入调试代理，辅助宿主机对被调试程序的执行过程进行控制，以便随时查看和修改被调试程序执行的状态。这种方法通常用于应用软件的开发，少数情况下亦可用于底层系统的开发，宿主机和目标机之间一般通过串口或网口来连接。常用的软件调试方法有：

1. ROM monitor

对于嵌入式系统来说,其地址空间里通常都会有一段只读空间,ROM monitor的调试手段正是利用了这一点,通过在嵌入式系统只读空间中写入辅助调试的软件,达到调试程序的目的[11]。ROM monitor有两个功能:用来控制嵌入式目标板上运行的程序;负责与PC端的调试器进行调试信息通信,一般使用远程调试协议(RSP, Remote Serial Protocol)[12]来进行通信。由于ROM monitor是驻留于只读空间,当嵌入式系统上电后,首先会执行这段ROM monitor代码, 达到对目标系统的控制,然后通过PC端调试器与ROM monitor相连,这样便可以进行调试,基本的ROM monitor包含了基本调试使用的命令,如断点、单步、寄存器访问、内存访问等。比较复杂的ROM monitor还有一些高级功能,如系统性能分析、代码密度分析、ROM空间访问等。MicroController Pros公司开发的MON-51，可以对8051系列单片机进行调试，它实现了单步执行、连续执行、设置断点等基本调试功能[22]。

1. Debug Stub/Server

调试桩(Debugging Stub)和调试代理(Debugging Server),虽然他们也是需要运行在嵌入式系中,但是在使用中,并不需要烧写到ROM中去,而且系统运行开始的时候,并不需要首先运行调试桩和调试代理。要使用调试桩或调试代理,首先需要做的是使用其他手段将软件下载到嵌入式目标板上,然后运行程序,它们都是通过RSP协议与PC机软件进行通信的,且也都能完成基本的调试命令,如断点、单步、寄存器访问、内存访问等。

调试桩使用时不需要嵌入式操作系统的软件支持,它需要与嵌入式软件一起绑定运行,即需要在系统的编译阶段将调试桩功能加入进去,这种调试方式是一种侵入式的调试,一般用来调试嵌入式底层的一些软件,如板级支持包、硬件抽象层等;调试代理则需要嵌入式类Unix系统的软件调试接口的支持,它的运行机制与本地机的调试器运行类似,通过系统接口达到对目标程序的控制,是一种非侵入式的调试方式。使用调试代理时,嵌入式目标板首先运行类Unix操作系统,然后运行调试代理和被调试的应用程序,用户可以通过调试代理来调试嵌入式应用程序。

为了实现远程调试，GDB针对不同的平台和操作系统提供了GDBServer和GDB Stub。将编译好的GDBServer下载到目标机上后进行调试，或者将GDBStub与应用程序一起编译成可执行文件完成调试功能。

1. 模拟器调试

通常使用的Simulator是指令级的模拟器,它相当于在宿主机上虚拟了一台目标机。该目标机可以是和宿主机的CPU不同的类型。利用指令集模拟器进行的交叉调试是一种完全软件模拟的调试方法,根本不需要目标板的支持,就连I/O等设备也都是软件模拟的。通过基于RSP实现的软件调试接口与宿主机GDB进行调试会话,实现调试。使用模拟器调试的优点在于,这种调试是一种纯软件的模拟调试,其可扩展性非常好,常常可以根据需求很快的变动,这在嵌入式软件开发的起步阶段是非常有实用价值的,可以大大降低研发成本。

在学术研究中，有一些针对不同平台模拟器调试的研究成果。北京航空航天大学研究了了ARM体系结构下的模拟器设计以及调试模块的实现[23]；在SPARC平台上，也有对模拟器调试的研究，但还仅限于单核调试方面[6]。

### 常用技术对比

ICE有非常多的调试手段,其中实时跟踪是ICE提供的最有特色的调试手段,它可以获取处理器的运行情况但又不会占用系统运行的时钟周期,是典型的非侵入(nonintrusive)调试。但ICE也有一些缺点:过高的调试成本;较差的可扩展性;适用不够广泛。

OCD的功能单元在CPU内部,流片时会预留出芯片的访问管脚，由于JTAG占用了非常少的硬件资源，因此大部分实现中都使用JTAG连接主机与目标机。但因为JTAG属于串口通信,较慢的数据传输速度势必限制了使用OCD实现的CPU的调试,且OCD技术还不能够实现诸如实时跟踪这种复杂的调试功能。

ROM monitor的调试方式在实际使用中并不普遍，主要由于：本身开发难度较大;平台相关度大。

软件调试方法不需要专用的硬件支持，尤其是在系统开发初期，没有实际的硬件环境，只能使用软件调试方法。GDB Stub需要在编译期间将调试功能写入应用程序中，这对编译器和程序编写能力要求较高；GDB Server相当于运行在目标机上的一个应用程序，通过操作系统提供的相关接口，访问其它应用程序的运行情况，这需要目标机上有操作系统的支持。这种方式由于其非侵入性，在软件调试中经常被使用。

在基于SPARC架构的多目标调试项目中，实验环境为基于SPARC V8架构的模拟器。因此，硬件调试方法不可行，只可以利用软件方式进行调试。使用GDB Stub方式需要在应用程序编译期间将调试功能编译到程序中，这对于大量涉及保密的航天应用来说，是不现实的。被普遍使用的GDB Server技术，虽然可以实现非侵入式调试，但需要操作系统的支持。而在模拟器设计初期阶段，还没有实现操作系统的模拟，同时模拟器的结构以及应用程序调试功能的需求，在这一阶段可能经常发生变动。为了加快调试功能的开发，采用模拟器调试是最合适的选择。

在现有的基于模拟器调试研究中[6][23]，主要研究重点还在单核调试上，几乎没有多核调试方面的相关研究。与单核调试相比，多核调试的难度和复杂度更大。在多核调试中，同步模式是相对简单的调试模式，但与单核调试相比，还需要考虑多核程序执行的不确定性，调试过程中处理器核的调度，如何对不同处理器核设置断点等问题。在同步调试模式下，GDB与目标机在同一时间只有一个在运行，当一个处理器核遇到断点暂停执行时，其余核也会暂停执行；为了实现GDB与目标机的异步执行，并且在调试某个核时不影响其他核的执行，引入了异步模式。在异步模式下，需要考虑如何缓存调试事件，GDB与目标机之间消息同步，如何保证调试某个处理器而保持其他处理器继续运行等一系列的问题。

相比于传统的基于模拟器的单核调试，多核调试中一个重点和难点就是多个处理器核的调度问题。在同步模式下，多个核需要同时运行，同时暂停。在此模式下，模拟器顺序调度每个处理器核，保证它们都执行相同的周期数，模拟同步运行的效果；在调试时，对于读取寄存器或内存地址等命令，保证所有处理器核都不会被被调度，对于单步或continue等命令，仍采用同步调度方案，保证核的同步执行。在异步模式下，使用多个线程模拟多个处理器核，每个线程内部处理自身的调度命令，线程之间的同步只发生在程序中需要同步的地方，这样保证每个核在调试时互不影响，达到异步调试的目的。

## SPARC架构模拟器

### 2.2.1 模拟器框架结构

模拟器基于SimICT框架，多个处理器核共享内存。处理器核、RAM等都作为模拟器中的模块，需要在框架中注册，运行时由框架调用模块的接口。模拟器调试主体部分结构如图1所示：

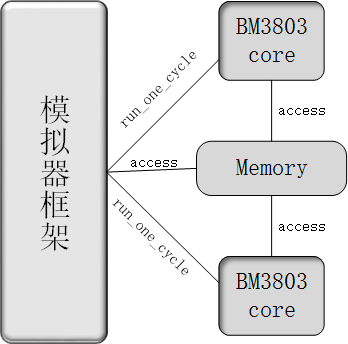


图1 模拟器调试主体部分

BM3803 是基于SPARC V8体系结构的32位RISC嵌入式处理器, 可用于板上嵌入式实时计算机系统,能够满足各种航天应用的功能以及性能指标要求,只要加上存储器和与应用相关的外围电路,就可以构成完整的单板计算机系统。

### 模拟器远程调试

在基于SPARC模拟器的远程调试中，宿主机运行GDB程序，模拟器作为目标机实现GDBServer的功能，两者通过TCP/IP网络进行连接，采用RSP(Remote Serial Protocol)协议进行通信，在宿主机完成对模拟器上应用程序的交叉调试。图2给出了远程调试系统的模型。

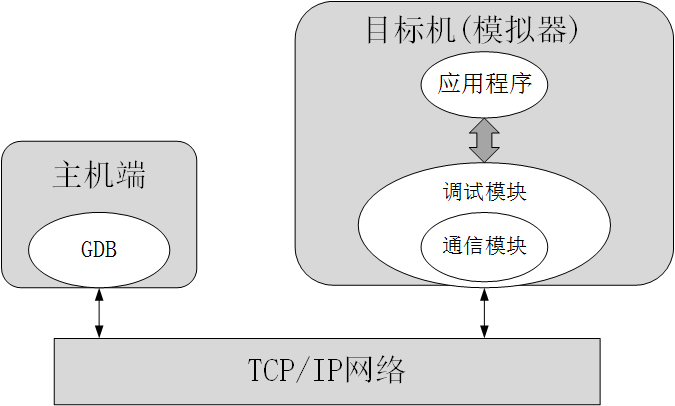


图2 模拟器远程调试

# 3 课题主要研究内容、预期目标

## 主要研究内容

本文的主要工作是研究SPARC架构模拟器的多核调试技术，并实现多核调试功能。嵌入式系统的多核调试已经越来越成为嵌入式领域的重点研究方向，比较成功的调试工具有ARM RealView Debugger，TotalView Debugger，WindRiver Workbench，DS-5 Debugger等，这些调试工具有的是针对某种架构的工具，有的需要硬件支持，并且作为商用工具，都没有开源并且使用成本较高。在国内外的研究中，基于SPARC架构的多核调试几乎没有公开发表的研究。

针对SPARC架构模拟器以及多核调试的需求，本文主要进行以下几个方面的研究：

* **可扩展的多核调试协议的交互机制研究。**本文采用开源且功能强大的GDB作为调试工具，GDB与目标机通过RSP(Remote Serial Protocol，远程串行协议)进行交互。需要考虑如何高效的完成交互过程，同时考虑以后如何支持更多的协议类型。
* **设计基于SPARC架构的断点处理机制。**为了避免断点检测中冗余的断点判断过程，设计断点指令，利用trap机制，完成程序自动跳转至断点处理过程进行调试。
* **异步多核调试机制。**在异步调试中，设计宿主机与模拟器的异步交互方法；在现有调度机制上，设计异步调度机制支持异步调试；在异步调度方案中，利用程序局部性、trace cache、穿线码等技术对模拟器异步调度的执行过程进行加速。

## 3.2 预期目标

* 完成模拟器与宿主机的可扩展的交互模块；
* 与传统的模拟器断点检测方法相比，新的断点检测方法可以大幅降低断点检测的额外开销；
* 完成模拟器与主机端的异步交互；
* 设计异步调度机制，并且加速异步调试过程，获得理想的加速效果

# 4 拟采用的研究方法和技术路线

在SimICT模拟器框架的基础上，结合对SPARC架构处理器核的模拟，实现多核调试功能。模拟器使用C语言实现，用户调试界面基于Eclipse插件开发。

## 4.1 可扩展的多核调试协议的交互机制研究

在基于模拟器的调试中，用模拟器模拟目标机，模拟器可以运行在主机上也可以运行在远程服务器上，模拟器与主机通过网络或串口连接。利用RSP(Remote Serial Protocol)完成主机与模拟器之间的交互。图2给出了两者之间的交互过程。

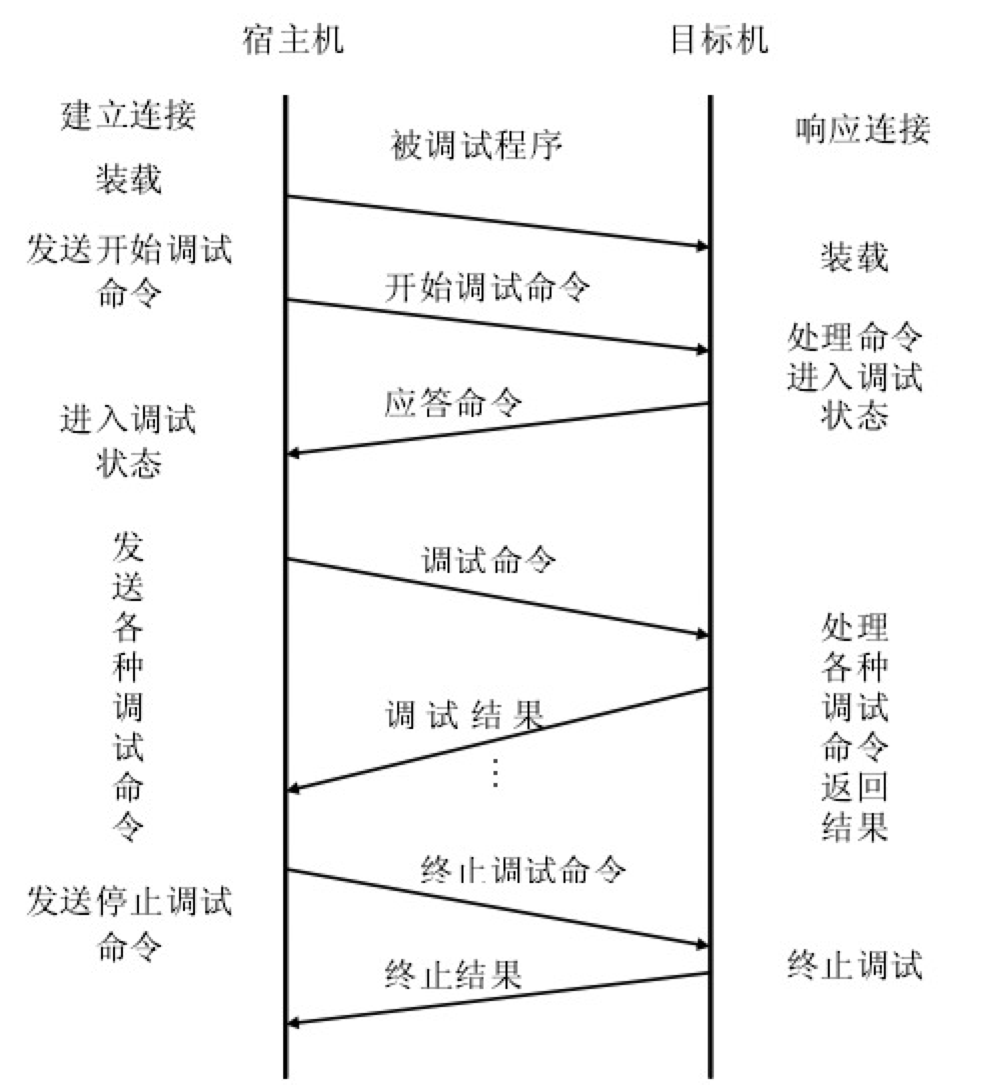


图3 基于GDB的远程调试交互过程

### 4.1.1 RSP协议

宿主机与模拟器之间通过RSP协议[14]交互。RSP制定了一系列具备一定格式的调试命令。RSP 协议包含了起始位，数据，结束位和校验和。一个典型的 RSP 协议包的结构如图3：

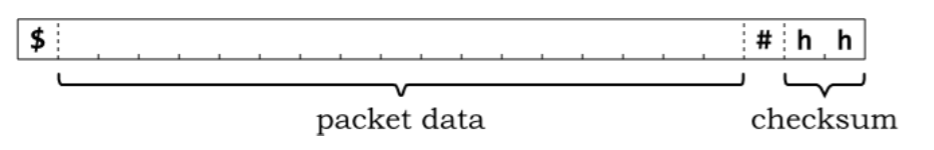


图4 RSP协议数据包格式

每个数据包都以$开口，packet data指出来命令类型，命令的参数以及具体数据，数据部分以#结尾，checksum是packet data部分的校验和模256后的16进制表示。

### 4.1.2 Flex集成以及接口设计

RSP协议中除了常用的调试命令外，还有很多高级命令。在实现调试功能初期，无法全部支持所有命令，同时，为了应对之后RSP协议的扩展，需要设计一套可扩展性强的交互接口。

Flex是一个自动化工具，在.l文件中定义一系列规则以及对应的动作(action)，经过编译后，可以按照定义好的规则自动生成一个C函数yylex()，也称为扫描器（Scanner）[15]。这个C函数把文本串作为输入，按照定义好的规则分析文本串中的字符，找到符合规则的字符序列后，就执行在规则中定义好的动作。

在模拟器交互设计中，将目前支持的调试命令对应的正则表达式作为Flex中的规则，同时指定规则中的action为解析协议的功能函数。rsp\_parse.l中除了包含了这些规则以及action，还包含rsp\_parse()。rsp\_parse()是交互过程中向模拟器展示的接口，在内部调用了Flex生成的yylex()，将输入的协议解析成定义好的rsp\_packet结构体返回给模拟器，图4给出了协议解析的处理过程。

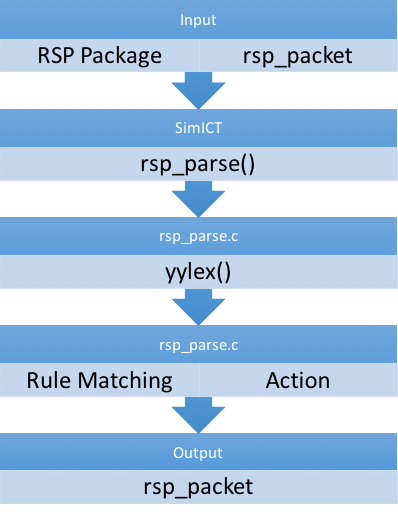


图5 协议解析过程

将Flex集成到模拟器中后，模拟器只需要调用rsp\_parse()接口就可以完成协议数据的解析。同时，如果要增加新的协议命令，只需要在修改rsp\_parse.l，增加新的规则和action，并完成对新命令的解析。这些操作对模拟器都是屏蔽的，在模拟器中只需要在rsp\_packet中增加新的命令类型即可，这极大提高了模拟器交互的可扩展性。

## 4.2基于SPARC架构的断点处理机制。

在传统的模拟器断点处理中，处理方法是，每执行一条指令前，都需要判断当前PC值是否是断点地址，而这些判断过程大部分都是冗余的。本部分的研究目的在于，设计breakpoint指令，减少冗余的断点判断过程，加快调试过程。

### 4.2.1 传统断点检测

在传统断点检测过程中，模拟器维护一个全局断点列表。在多核同步调试中，采用轮询调度，每次调度一个处理器核执行一条指令前，都需要将PC与断点列表中的地址进行比较，判断是否运行到断点处。图6给出了这种检测过程的流程图，其中的check\_breakpoint方法完成的就是将PC与断点列表进行比较的过程。

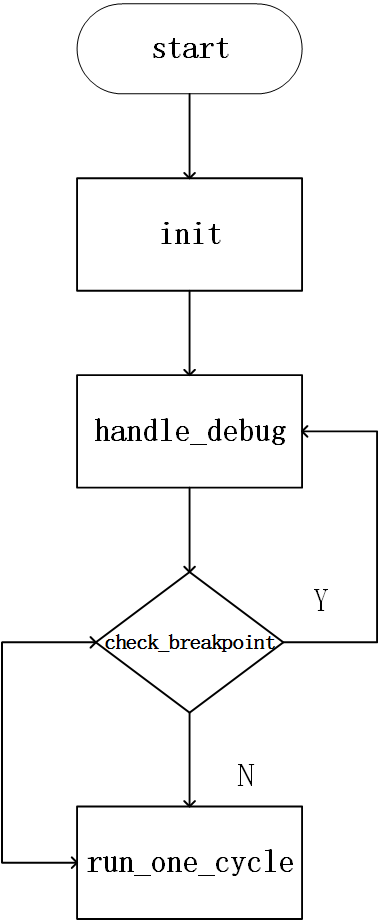


图6 传统断点检测过程

由于一般调试过程中，断点的数量不会很多，因此，这些判断过程绝大部分都是冗余的，大大降低了调试的速度。因此，需要设计一种新的断点处理机制，减少这些冗余的比较，只在真正可能是需要暂停的地址处，进行断点检测。

### 4.2.2 基于SPARC架构的breakpoint指令

为了减少断点判断的次数，提出设计一种基于SPARC架构的断点指令，只有当处理器遇到断点指令时，才会进行判断该地址是否需要进入调试过程。这样，断点判断的次数就从每条指令执行一次降低为只在断点指令处执行。显然降低了调试开销。

SPARC架构中没有定义断点指令，因此需要根据架构特点及其指令集，设计一条breakpoint指令，满足断点调试的需求。

SPARC架构提供了trap机制，并且允许设计者设计并实现软件trap类指令，本项目中的breakpoint指令，就基于其trap指令进行设计。

每个SPARC处理器核会维护一个trap向量表(Trap Table)，用来保存每个trap对应的handler程序的前4条指令。TBR(Trap Base Register)用来指示trap table的表项，图7给出了TBR的具体fields，其中TBA保存了trap table基地址的高20位，由内核写入，tt(trap type)给出了trap的类型，根据不同的tt跳转到trap table的不同表项。图8给出了SPARC架构下支持的trap类型，其中0x00到0x7F是硬件trap，而0x80到0xFF表示软件trap指令的类型。

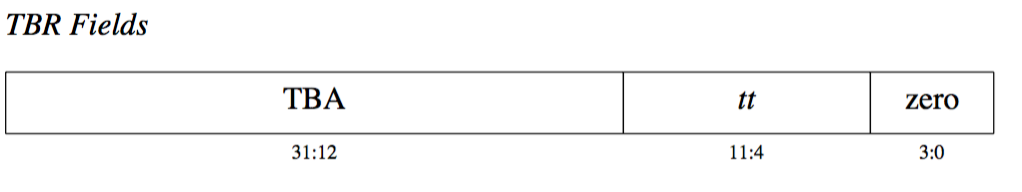


图7 TBR寄存器

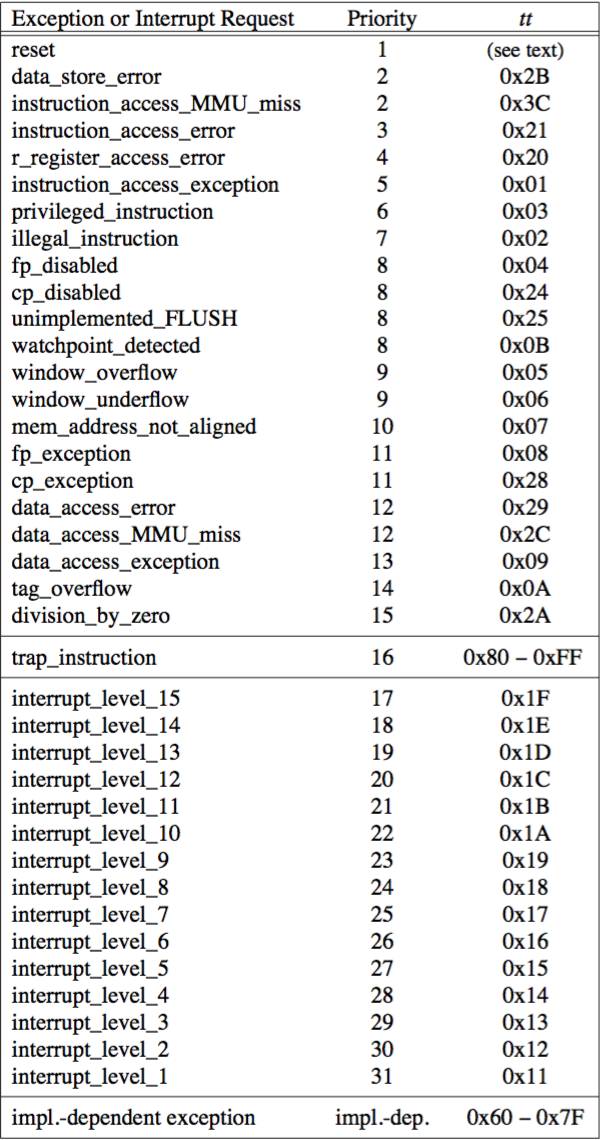


图8 trap类型表

利用trap机制，定义breakpoint指令，指令格式如图9所示。其中10和111010表示该指令为trap类指令，cond为1000表示总是进入trap处理过程，rs1为0即使用r[0]寄存器，imm7设置为0。tt为r[rs1]的低7位与imm7的和加128，因此breakpoint指令的tt值为0x80。

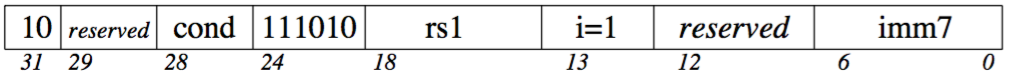


图9 breakpoint指令格式

在breakpoint指令对应的trap向量表项中，存放跳转命令，跳转到模拟器中handle\_debug()的处理过程，进行调试处理。

### 4.2.3 断点处理机制

在breakpoint指令的基础上，设计新的断点处理机制。当模拟器收到设置断点的命令时，将目标地址的指令保存，并替换为breakpoint指令，同时将断点信息保存到全局的断点信息表中。

在同步调试过程中，模拟器顺序调度每个处理器核，当遇到breakpoint指令时，首先将breakpoint指令替换为原始指令。查询断点信息表判断断点所在的核是否是当前核，如果是则进入调试过程，否则单步执行这条指令，执行完后再替换为breakpoint指令，避免其它核错过断点。

在跳转到调试处理过程时，模拟器保存线程，将PC和nPC的值保存到r[17]和r[18]中，同时将trap向量表中第一条指令的地址写入nPC。

## 4.3 异步多核调试机制

在模拟器多个处理器核上执行的多个程序一般是完成整个程序的不同部分，相当于一个进程的多个线程。在同步机制下，当一个处理器核上的程序遇到断点暂停后，其余处理器也会暂停执行，这与实际应用情况不是十分吻合。在实际中，一个处理器核出现中断时，允许其他处理器继续执行，同时在宿主机也应该允许对其它处理器核的调试[20]。

在模拟器中，为了更精确模拟多个处理器核的执行情况，设计异步调度方法，支持异步调度；并且在异步调度的基础之上对模拟器进行加速，提高运行和调试效率。

### 4.3.1 调试事件处理机制

在异步模式下，宿主机与目标机端交互不再像同步模式中那样互相等待。用户在宿主机输入调试命令后，可以在调试信息返回前，继续输入调试命令，查看当前核的状态或着切换调试核等。

为了保证异步交互，需要在宿主机与目标机端(模拟器)设立缓存队列，采用事件缓存策略，对未处理的消息进行缓存。

在宿主机，讲用户输入以及返回的调试信息，作为事件缓存到队列中，通过select()系统调用[21]选取事件进行处理。如果是用户输入事件，则调用响应的命令解析模块，解析用户输入命令并发送RSP数据包给目标机；如果是目标机返回的调试信息，则调用对应的调试处理模块对信息进行处理。

在目标机端，由于多个处理器核异步执行，每一个处理器核维护一个调试事件队列，模拟器框架根据调试命令中的线程号，将调试命令缓存到对应的队列中。每个处理器执行指令前，查看私有的事件队列是否有未处理的命令，如果有则进入调试过程，将调试结果返回给宿主机，否则正常执行程序。

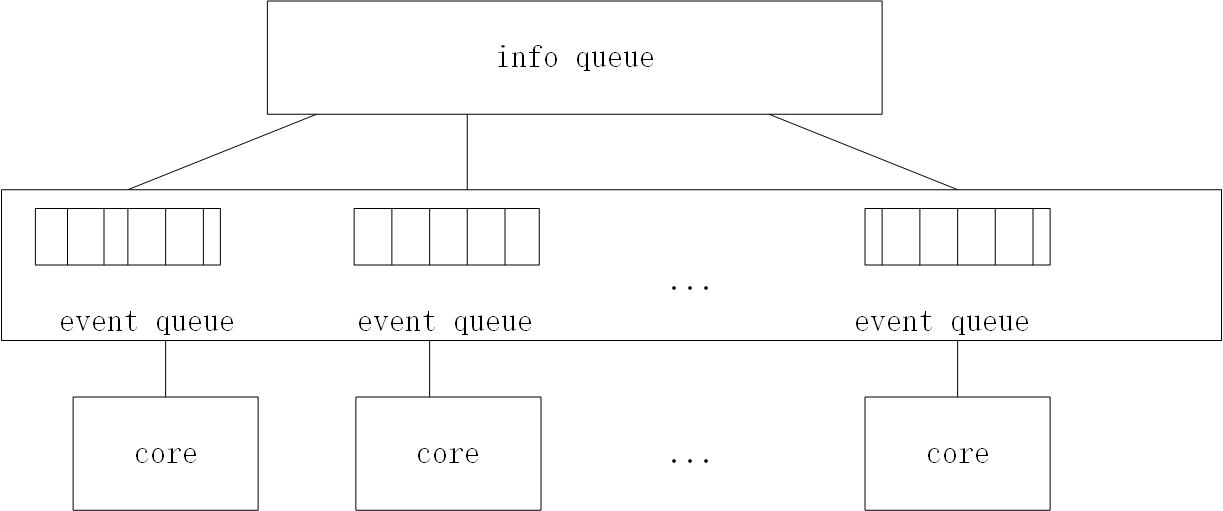


图10 多核调试事件队列

图10给出了模拟器上多个调试事件队列的设计方案，当注册一个处理器核到框架中时，为其开辟私有的事件队列，保存核相关的调试信息。

### 4.3.2 异步调度

现有的模拟器中采用轮询同步调度每个处理器核，当所有处理器核都执行完一条指令后，才进行下一条指令的执行。当遇到断点时，转入调试程序，在调试过程中，其它处理器核都不会被调度。而在异步调试模式下，一个处理器核的调试不应该影响其它处理器核的运行，即某个处理器核在调试过程中时，其它处理器核可以正常执行应用程序指令。

模拟器启动调试时，先将设置好的断点信息保存并且完成指令替换。在调度过程中，如果没有遇到断点指令则每个处理器核都正常运行程序；如果遇到断点指令，并且需要进入调试过程，则当前核等待后续调试命令，或返回调试信息，其它核在这次调度中，正常执行指令。

在调试过程中，根据实际的需求，一个核在调试时，其它核正常运行，但不应该没有限制的运行下去。因此，支持用户设置自定义阈值，指定运行最快的处理器核可以超前被调试核多少条指令。因此，为每个处理器核保存当前执行的指令条数，并且动态维护全局执行最慢的处理器核的指令条数，当调度到某个处理器核，发现二者之差达到阈值，则暂停该处理器核的执行。

### 4.3.3 异步调度下的加速机制

在上一节提出的异步调度方案中，每次处理器核调度，只执行一条指令。这对于模拟器的运行，意味着每调度一次都要访问不同的内存地址去取指令和数据，显然对于模拟器的运行速度有很大影响。

设计合适的指令窗口，每次调度会执行完窗口内的所有指令，再切换到下一个处理器核，这种改进的调度方式可以更好的使模拟器利用程序运行的局部性，提高运行速度。

在模拟器执行过程中，对于指令的取指、译码是很耗费时间的，针对这个过程进行优化。利用Trace Cache缓存执行过的指令，当发现指令在Trace Cache中时，直接执行，省去取指、译码的过程；分离程序的基本块，利用穿线码技术，当执行到之前执行过的基本块时，不需要译码、取指，可以直接执行基本块中后续的指令。

# 5 已有科研基础与所需的科研条件

## 5.1 已有科研基础

1.已经完成了双核SPARC模拟器的基本功能。

2.研究了GDB调试协议。

3.完成了部分eclipse插件定制开发。

## 5.2 所需的科研条件

实验环境需求：SPARC模拟器，eclipse开发环境。

# 6 研究工作计划与进度安排

2016.8~2016.11 完成基于SPARC模拟器框架的同步调试功能实现。

2016.11~2017.3 完成异步调试功能的研究，测试模拟器的调试功能，准备中期答辩。

2017.3~2017.5 完成毕业论文和答辩工作。

# 7 参考文献

[1] 高效伟.多处理器并行处理的新发展 Linux 操作系统内核对 SMP（对称多处理器）的支持，计算机应用研究,2002(9):62-67

[2] Jeff Parkhurst, John Darringer, Bill Grundmann. From Single Core to Multi-Core: Preparing for a new exponential.

[3] Asawaree Kalavade, Joe Othmer, Bryan Ackland. Software environment for a multiprocessor DSP. Annual ACM IEEE Design Automation Conference, 1999,827-830

[4] 彭冬脉，基于GDB的双核调试器的研究与实现，电子科技大学学位论文，2010

[5] 罗克露，陈云川.嵌入式软件调试技术[M].北京：电子工业出版社，2009.

[6] 范涛, SPARC平台模拟器源码级调试系统的研究与实现. 计算机工程与应用[J]. 2013,49(4)

[7] http://www.eepw.com.cn/article/114121.htm

[8] BM3803产品用户手册(C)

[9] 罗蕾.嵌入式实时操作系统及应用开发.北京:北京航空航天大学出版社,2005，77-81

[10] 蒋龙,基于GDB的嵌入式多任务调试器的设计实现与集成.浙江大学学位论文,2014

[11] 张克丰.嵌入式系统的调试方法研究及调试器后台处理模块设计.西安电子科技大学学位论文,2012.

[12] GNU organization. The GDB remote serial protocol[EB/OL].2011-08.

[13] 谭金伟.嵌入式JTAG仿真调试器的研究与实现.电子科技大学学位论文,2004.

[14] Jeremy Bennett. Howto: GDB Remote Serial Protocol

[15] <http://www.cnblogs.com/wanghetao/archive/2011/11/07/2240193.html>

[16] 何克右,周彩贞.Linux2.6 进程调度机制的剖析.华中师范大学学报，2007,41(4):520523

[17] http://blog.csdn.net/guojin08/article/details/25457311

[18] The gnu Source-Level Debugger. Debugging with gdb Tenth Edition.

[19] 李国徽, 诸方舟, 王安军, 马艳, 曹粟, 黄瑛, 李艳红. 事件循环机制在嵌入式多线程应用级调试器设计中的应用. 中国科技论文, 2013, 15(1): 67~71

[20] K. Chatterjee, L. De Alfaro, V. Raman, et al. Analyzing the impact of change in multi- threaded programs. In: Proceedings of the 13th international conference on Fundamental Approaches to Software Engineering. Berlin: Springer Verlag, 2010.

[21] 黄瑛，基于non\_stop调试模式的嵌入式应用级交叉调试器的研究, 华中科技大学学位论文，2013

[22] <http://microcontrollershop.com/raisonance_rommonitor.php>

[23] 于婷，基于模拟器远程调试系统的研究与实现.微计算机信息，2008，24(1)