ArmSim：基于ARM处理器的全系统模拟器[1]

邓立波[1]，龙　翔[1]，高小鹏[1]
[1]北京航空航天大学计算机学院，北京(100083)
E-mail: denglibo@cse.buaa.edu.cn

摘　要：模拟器作为嵌入式系统研究的基础研发工具，可辅助系统体系结构调优、软硬件协同设计。本文实现了具有良好配置性及可扩展性的 ArmSim 模拟器，该模拟器是针对 ARM 处理器的全系统模拟器，可在其上运行和调试 ARM 应用级和系统级的目标程序。本文详细描述 ArmSim 的设计与实现细节。

关键词：ARM 处理器　模拟　全系统模拟器
中图分类号:

1. 引言

随着嵌入式系统的飞速发展，嵌入式系统的研究与开发已经成为当今计算机科学的一个重要分支。由于应用领域的特点，嵌入式系统研发通常需要依赖特定的硬件环境。由于对硬件环境的过度依赖，因此传统的研究开发模式具有明显的缺陷，即软件开发与硬件开发无法并行展开。这一方面致使研发周期过长，另一方面也使得设计工作缺乏足够的灵活性。

为了解决上述问题，基于软件的模拟器已经成为嵌入式系统研发中的主要技术手段之一。软件模拟就是用计算机软件来模拟某一特定硬件系统的全部或部分的外部特性和内部功能，实现对目标硬件系统的高度仿真，使得运行在模拟器上的程序无法感知到底层硬件的存在，就如同运行在真实硬件平台上。在嵌入式系统领域，软件模拟技术已经被广泛应用于嵌入式系统软硬件协同设计、嵌入式操作系统开发与评估以及大型嵌入式应用软件性能评估等方面。

本文描述的 ArmSim 是一个基于 C 语言的 ARM 处理器的全系统模拟器。ArmSim 实现了全系统硬件的功能模拟，不仅可以运行 ELF 格式的 ARM 应用级程序，而且可以运行 ELF 映象或二进制映象格式的系统级程序，如 U-Boot。ArmSim 模拟器还支持 GDB 等调试器对模拟器上运行的 ARM 程序进行源代码级远程调试。

本文其它部分的结构如下：第 2 部分对主要的模拟技术进行阐述；第 3 部分详细描述 ArmSim 模拟器的实现；第 4 部分为结论。

2. 模拟技术

全系统模拟器的实现主要包括指令集的模拟和外部设备的模拟。对指令集的模拟是构建所有目标机模拟器的基本要求，而对于构建可以运行目标机系统级程序的全系统模拟器而言，对外部设备的模拟则是其必要的组成部分，是模拟 CPU 和宿主机的通信接口。

2.1 指令集的模拟

对指令集的模拟，最常用的方法包括：解释执行和基于翻译的方法。

[1] [1]本课题得到高等学校博士学科点专项科研基金（项目编号：20030006026）资助

解释执行方式是最简单也是使用得最为广泛的一种指令集模拟方法，如SimOS[1]、Skyeye[2]等都采用了该方式。解释执行的原理是在模拟器内部构建和目标机器对应的CPU、内存等模拟模块，在模拟器执行过程中，利用这些模拟的硬件执行部件模拟真实CPU的工作流程，也就是通常的“取指—译码—执行”循环。穿线代码方式[3][7]是对解释执行方式的一种改进，通过缓存已执行过的指令所对应的模拟函数的地址，实现了一次译码多次执行，从而提高了模拟效率。

在采用基于翻译的方法实现指令集的模拟时，常用的方法有动态翻译[4][5]和静态翻译。动态翻译是在模拟器运行过程中，将目标机器的二进制代码翻译为宿主机的二进制代码的模拟技术。动态翻译一般以一个基本代码段(通常为两个条件语句之间的代码)为基本单元，对每个基本单元通过一次翻译多次执行来提高运行效率。静态翻译是将指令的翻译过程移到了编译阶段，即在编译阶段将目标机器的二进制代码翻译为宿主平台的二进制代码。模拟器运行时直接执行翻译后的宿主平台代码，完全省去了运行时的译码和翻译开销，因而效率也最高。但静态翻译方式不能提供对自修改代码的支持，缺乏通用性。

2.2 外部设备的模拟

在实现外部设备的模拟时，需要考虑两个方面的内容：一是模拟外设的访问；二是模拟外设和其它模拟部件的同步。

2.2.1 外设访问机制

在全系统模拟器中，实现模拟外设的访问主要有两种方法：基于内存空间映射和基于虚拟管脚。基于内存空间映射方法就是模仿真实硬件上采用的线性空间地址映射，将外部设备所占的地址空间映射为整个地址空间的一部分，将模拟的外部设备函数注册到该段地址空间上。这样当模拟CPU访问该段地址空间时，模拟器就会调用注册到该段地址空间上的外部设备函数，实现外部设备的相应动作。在Skyeye[2]中大量采用了这种机制。基于虚拟管脚的外部设备模拟，即模拟外设的管脚，通过定义模拟外设的管脚以及管脚间的连线，从而可以通过设置其高低电平来实现通讯目的[6]。

2.2.2 外设同步机制

在实现模拟外设和其它模拟部件的同步时，主要包括三种模型：串行模型、并行模型和混合模型。目前，在全系统模拟器中，实现模拟外设和其它部件的同步时，广泛采用的模型为串行模型(Skyeye[2]采用该种模型)，该模型在每个指令周期上处理CPU和外设的同步，即模拟CPU每执行完一条指令，都要依次检查所有外设，执行外设的相应动作。当外设通信量较少时，这种方法会浪费大量的CPU时间。并行模型就是每个模拟外设建立并拥有一个独立的线程，外设模拟工作都在各自的线程中完成，而不是交由系统唯一的全局线程完成。模拟开始后，外设线程周期性被系统唤醒，执行工作循环，完成相应的模拟工作后，再次进入休眠，直至下一次被唤醒[6]。通过采用并行模型可以减少CPU时间的浪费，但该模型对线程休眠时间的设定有较高的要求，否则增加的线程切换开销会对整个模拟器的性能产生负面影响。混合模型[6]就是串行模型和并行模型同时使用，根据模拟外设的使用频率决定使用何种模型，从而实现模拟器运行效率的最大化。

3. ArmSim 的实现

ArmSim 实现了对 CPU、存储设备(RAM 和 ROM)、外部设备(开发板和其它外部设备)等硬件设备的模拟。其中，ArmSim 模拟的 CPU 为 ARM720T，包括指令集、MMU、Cache 等，指令集的模拟只实现了 32 位的 ARM 指令，而未对 16 位的 Thumb 指令进行模拟；模拟存储设备的容量取决于宿主机，可以划分为不连续的多块区域，最多不超过 12 块；模拟的开发板为 EP7312，包括开发板的时钟、IO 寄存器、UART 寄存器等，通过开发板的 UART 接口来实现标准控制台输入输出。ArmSim 的总体架构如图 1 所示。

ArmSim 具有良好的可扩展性，主要体现为模拟器中所有的构件都有一个一致的外部接口——初始化函数。模拟器使用者可以在不需要了解整个模拟器内部结构的情况下，只要在单个模拟构件中实现指定的外部接口——初始化函数，就可以实现在模拟器中增加模拟器构件，实现对模拟器进行扩展。ArmSim 通过配置文件的方式来实现模拟器的设备配置和初始化，这使得其具有良好的可配置性。用户只需要更改配置文件中的设备配置信息，即可实现在不同配置的模拟器上运行目标程序。ArmSim 的扩展及配置过程如图 2 所示。

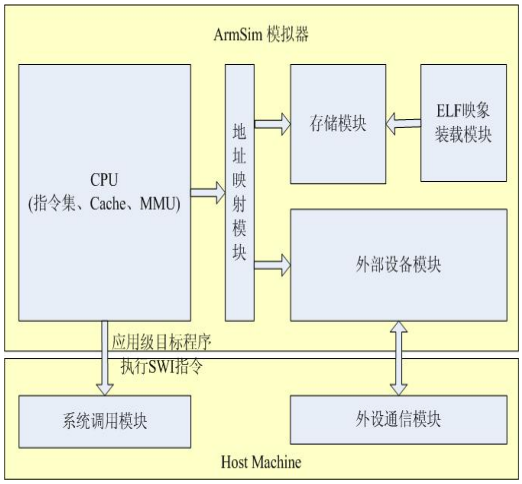

图 1　ArmSim 模拟器的总体架构

图 2 ArmSim 的扩展及配置过程

3.1 指令集的模拟

ArmSim 模拟器的指令集模拟采用简单的解释执行的方法，即“取指—译码—执行”循环，没有流水线的实现，而以功能模拟作为其目标。但由于 ArmSim 良好的可扩展性和可配置性，用户可以根据具体情况重新设计一个模拟效率更高、模拟粒度更细的指令集纳入系统中。

在 ArmSim 指令集的模拟中，指令译码采用函数指针数组的形式，通过函数指针数组项的快速定位来加快指令译码的速度，从而提高了指令集模拟的速度。其示意图如图 3 所示。

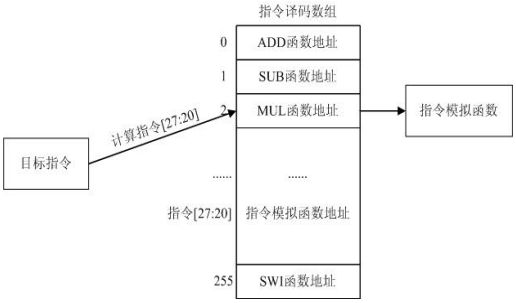

图 3　ArmSim 指令集的译码

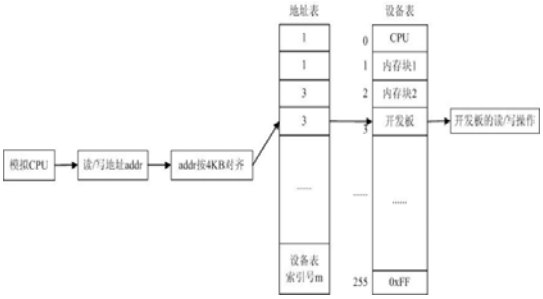

图 4　ArmSim 模拟外设/存储设备的访问

3.2 外部设备的模拟

ArmSim模拟外设的访问方式采用基于内存空间映射的方法来实现，即将模拟外设，包括模拟存储设备，所占的地址空间映射为整个存储空间的一部分，通过统一的访存操作来实现对模拟外设的访问。模拟器将整个 4GB的地址空间用一个具有 2^{20}个项的数组——地址表来表示，每个地址表项表示 4KB大小的地址空间，其内容为该 4KB地址所类属的模拟设备。当CPU访问某存储地址addr时，首先将该存储地址addr按 4KB对齐，其结果为地址表的索引，按该索引获取对应地址表项的内容即为该地址所类属的模拟设备，调用对应模拟设备的相应操作即可完成CPU的访问操作。为实现上述过程，需要在模拟器初始化时将模拟外设，包括模拟存储设备，所占的地址空间映射到地址表的对应表项上，即将模拟外设或模拟存储设备在设备表中的索引号写入地址表的对应表项中，完成模拟设备的地址空间映射。ArmSim模拟外设和存储设备的访问示意图如图 4 所示。

在实现模拟外设和其它模拟部件的同步时，ArmSim 采用了混合模型，即对于使用频率较高、响应延迟要求较小的外设，如控制台输入和控制台输出，采用串行模型；对于使用频率较低、响应延迟要求相对宽松的外设，如串口、网卡等，采用并行模型。对于采用并行模型的外设线程，ArmSim 没有采用周期性的线程唤醒策略，而是采用按需唤醒，即只有当需要对外设进行访问时，才唤醒外设线程，从而进一步减少了模拟器主线程和外设线程之间的切换开销。模拟器主线程通过线程同步机制来实现和外设线程的同步，模拟 CPU 通过中断来实现和模拟外设的同步，整个过程与真实的硬件完全一致。

在设计 ArmSim 模拟器时，将模拟外设与宿主外设之间的通信从外设的模拟中分离出来，所有需要与宿主外设进行通信的工作都包含在独立于模拟外设的外设通信模块中。外设通信模块采用多线程机制实现，每个外设线程负责相应外设的通信，如网卡线程负责实现与物理网卡的通信，串口线程负责实现与物理串口的通信等。在外设通信线程和模拟外设线程之间，用缓冲区来存放二者交互的数据。通过线程同步机制实现外设通信线程、模拟外设线程以及宿主机物理外设之间的同步，从而实现了模拟外设线程和外设通信线程的按需唤醒策略。

采用上述分离以后，模拟的外设构件只包括相应外设的结构及其上的操作，而无需考虑如何通过宿主外设与外部世界通信，从而有效地减少了模拟外设扩展时的工作量。同时，同一种模拟外设可以采用不同的方式实现与宿主平台的通信，如模拟串口可以实现与宿主平台串口的通信，也可以实现与宿主平台标准控制台输入输出的通信。这一方面增加了模拟外设

和宿主外设通信的灵活性，另一方面也为模拟宿主平台不存在的外设提供了可能，进一步增加了模拟器的可扩展性。ArmSim 外设模拟的示意图如图 5 所示。

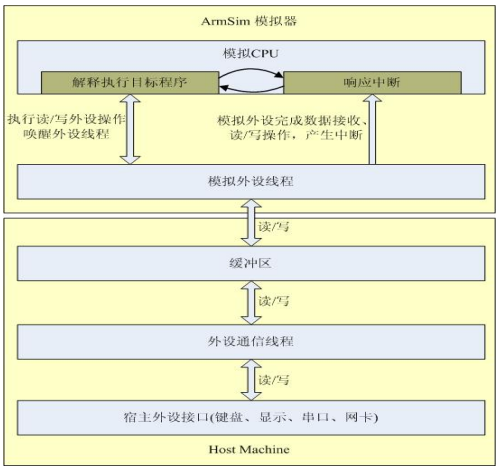

图 5　ArmSim 外设模拟的示意图

4. 结论

ArmSim 是一个基于 C 语言的 ARM 处理器的全系统模拟器，不仅可以运行 ELF 格式的 ARM 应用级程序，而且可以运行 ELF 映象或二进制映象格式的系统级程序，如 U-Boot。ArmSim 通过采用配置文件的方式使得整个模拟器具有非常良好的配置性。ArmSim 还具有良好的可扩展性，用户通过实现所有模拟构件的一致的外部接口——初始化函数，即可方便快速地实现模块扩展。ArmSim 通过模拟外设和宿主外设的通信封装，并独立于外设的模拟，简化了模拟外设扩展时的工作量，同时也进一步提高了模拟器的可扩展性。ArmSim 采用串并行混合模型实现模拟外设和其它模拟部件的同步，减少了模拟 CPU 时间的浪费，并通过并行外设线程的按需唤醒，进一步减少了线程之间的切换开销。

参考文献

[1] Mendel Rosenblum, Stephen Alan Herrod, Emmett Witchel, and Anoop Gupta. Complete computer simulation: The SimOS approach. IEEE Parallel and Distributed Technology, 3(4):34--43, 1995.
[2] 陈渝、李明、杨晔等编著, 源码开放的嵌入式系统软件分析与实践——基于 SkyEye 和 ARM 开发平台，北京航空航天大学出版社, 2004.9, 72--135.
[3] Bell, JamesR. Threaded code. C.ACM 16, 6 (June 1973), 370--372.
[4] E. Witchel et al. Embra: Fast and Flexible Machine Simulation. MMCS, 1996.
[5] Bellard Fabrice, QEMU, a Fast and Portable Dynamic Translator, USENIX, 2005, 41--46.
[6] 柯化成, 嵌入式系统全系统模拟器框架设计和实现, 杭州：浙江大学硕士学位论文, 2006, 23-29.
[7] R. C. Bedichek, "Talisman: Fast and accurate multicomputer simulation," in Proceedings of 1995 ACM SIGMETRICS Conference, pp. 14--24, 1995.

ArmSim：A Full-System Simulator for the ARM Processor

Deng LiBo[1]，Long Xiang[1]，Gao XiaoPeng
School of Compute Science, Beihang University, Beijing, PRC, 100083

Abstract

Simulators have become essential tools for embedded system research and development, which can help architecture optimization and software-hardware codesign。The presented ArmSim simulator is a full-system simulator for the ARM processor, on which ARM application programs and system programs can be executed exactly. We discuss the whole design and the implementation, the configurability and extensibility are considered.

Keywords: *ARM Processor* *Simulation* *Full-System Simulator*

作者简介：邓立波，1976 年生，硕士研究生，研究方向：计算机系统结构；
龙　翔，1963 年生，教授，博士，博士生导师，研究方向：计算机系统结构。
高小鹏，1970 年生，副教授，博士，研究方向：计算机系统结构。