

一种基于 cache 一致性的多核任务监测机制的实现方法

陈 慧¹,慕福奇^{1,2},张 然²,姜 彬²

(1 中国科学院大学 微电子学院,北京 100049;2 江苏中科羿链通信技术有限公司,江苏 无锡 214135)

摘 要: 为了解决在多核多操作系统这种情况下任务监测的不便,尤其是要处理某核运行异常,仍要有效的捕获和分析该核上的任务状况等问题,提出了一种基于 cache 一致性的多核任务监测机制的实现方法,对任务使用系统资源及自身运行状态与堆栈信息的监测,最后以 Zynq-7000 为硬件平台,在 Linux 和 FreeRTOS 双系统的模式下,进行试验测试,结果表明该机制能有效的监测多核系统的任务状况,为任务管理和异常处理提供方便。

关键词: cache;Linux;FreeRTOS;任务监测;Zynq-7000

中图分类号: TP31

文献标识码: A

文章编号: 1000—7180(2018)10—0019—04

A Implementation Method of Multi-core Task Monitoring Mechanism Based on Cache Consistency

CHEN Hui¹, MU Fu-qi^{1,2}, ZHANG Ran², JIANG Bin²

(1 School of Microelectronics, University of the Chinese Academy of Sciences, Beijing 100049, China;

2 Jiangsu Zhongke Yilian Communication Technology Co., Ltd., Wuxi 214135, China)

Abstract: In order to solve the problem of task monitoring, it is necessary to effectively capture and analyze the problems of the tasks in the core, such as the operation of a exceptional core. In the paper, a multi-core task monitoring mechanism based on cache consistency is proposed to realize the monitoring of the system resources and the operation status and stack information of the task. Then conducting the experimental test on the ZYNQ-7000 platform. In the double system mode of Linux and FreeRTOS, the results show that the mechanism can effectively monitor the multi-core system task status, provide convenience for task management and exception handling.

Key words: cache; Linux; FreeRTOS; task monitoring; Zynq-7000

1 引言

随着智能设备、物联网产业的迅速发展,嵌入式系统开始采用多核处理器作为硬件平台.由于应用场景与实现功能的不同,在多核上运行不同的操作系统成为了一种主流^[1].

在这种模式下,任务不仅在自己所在的系统上运行,有时还需跟其它核的任务进行交互,而且不同操作系统,任务调度机制也不同,这就使得多核任务的监测变得更加复杂,如何在解决核间通信的基础上,有效的监测各核的任务状态,成为了多核嵌入式系统需要解决的一个重要问题.

近年来,研究人员提出了多种核间通信的方

法,总结如下:(1)采用集成的可移植的核间通信模块,比如 MCAPI、MPI、D-Bus、LINUX 等^[2];(2)基于其嵌入式平台与操作系统设计的通信模块,比如使用共享内存、自旋锁、核间中断等机制.这些方法都能有效的解决核间通信的问题,并为任务监测提供核间通信的渠道,但是这种基于软件实现的通信方式,对于任务监测系统来说有如下的不足之处:(1)当芯片中某些核因为异常而被挂死,该核上的核间通信模块也就停止运转,这将导致其核上的任务状态信息无法传递到外界,不方便异常状况的分析;(2)核间通信模块在系统上也是以任务的状态运行,如果设计过于复杂,会加剧任务监测系统的负担.通过实现多核 cache 内容一

致,直接在存储单元中读取数据,可避免上述情况导致的问题.

因此,本文基于 cache 一致性,设计了一种多核任务监测机制,并在搭载了双核 Cortex-A9 处理器的 Zynq-7000 的平台上进行测试,结果证明在实现 cache 一致性的情况下,不管核是否正常工作,都能监测到各核上的任务状态信息.

2 多核任务监测机制的需求

任务监测是对系统当前时间点、空间域上任务状态的捕获与分析.在多核异操作系统中,如何有效的捕获各核的任务数据,并解析出任务使用系统资源及自身运行状态与堆栈的情况,最终以可视化的图表展示出来,是多核任务监测机制要实现的功能,因此多核任务监测机制应该满足如下需求:

(1)实时信息的捕获.信息捕获的前提是任务数据的存储,由于任务监测具有实时性,对于展示监测结果的内核,能从存储单元拿到正确反映其它核状态的信息至关重要.因此本文通过实现 cache 一致性来保证通过存储的数据能正确获取到系统实时的任务信息.

(2)状态信息的分析.从存储单元中拿到的数据只是一堆二进制信息,因此需要按照一定的规则解析出相应的状态消息.

(3)监测信息的展示.信息展示有助于开发人员对系统状态做出合理判断,方便系统异常时的错误定位和原因分析.

针对上述需求,本文以 Xilinx 下搭载了双核 Cortex-A9 处理器的 Zynq-7000 平台为例,提出了基于 cache 一致性,包括信息存储、捕获分析、展示的任务监测机制,为任务管理和异常处理提供方便.图 1 给出了该系统的结构模型, Linux 运行在 core0 上, FreeRTOS 运行在 core1 上,双系统工作在 SMP^[3] 模式下,各自独立的管理其私有资源, L1 级 cache 上存放着任务运行状态相关的实时信息,通过 cache 一致性的实现,完成任务监测信息的同步.其中 Linux 系统因其具有多样的任务查看命令,以及终端显示界面,将任务监测的分析与显示模块放置在 core0 的 linux 系统上.

3 任务监测机制的设计和实现

3.1 任务监测机制的设计框架

针对监测的功能需求,系统有三个主要的部分: cache 一致性的实现、任务信息捕获、数据分析.其中

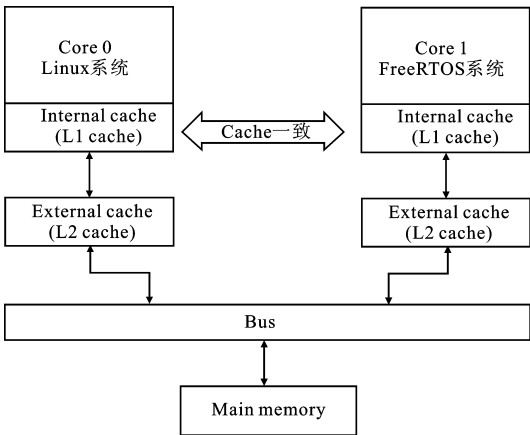


图 1 系统结构模型

cache 一致性是为了解决双核信息交互的问题,使得 core0 能获取到 core1 上实时的任务信息;任务信息捕获则完成 core0 读取到 core1 在指定位置上存储的任务相关的地址信息;数据解析是当 core0 获取到地址信息后,按照一定规则和策略解析地址信息,以获取任务运行时的状态、堆栈、事件、优先级等数据.通过以上的分析,任务监测机制的设计框架如图 2 所示.

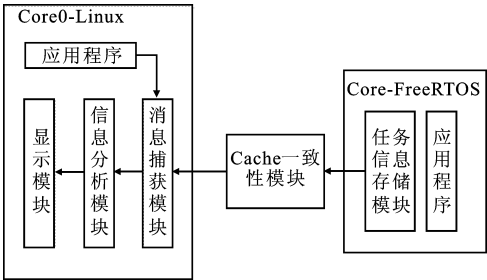


图 2 任务监测机制设计框图

3.2 cache 一致性的实现

cache 一致性可以理解为同一个数据块在多个私有 cache 中的副本是一致的,处理器从每个 cache 中读取的数据都是之前写入此缓存块的最新值^[4].

解决 cache 一致性问题,可以通过软件方法在主存中设置临界代码区,或者将可能出现一致性问题的存储区设为非 cache 来解决.软件解决占用处理器周期,总线带宽;设置为非 cache 模式,又起不到 cache 高速缓存的作用,因此考虑高效的硬件解决方法.硬件方法主要有监听协议(Snoop Protocol)和目录协议(Directory Protol)两种,其中目录协议主要适用于带有互连网络的多处理器系统,而监听协议通过总线监听机制实现 cache 和共享存储器之间的数据一致性^[5].考虑到实现平台在硬件上提供了 SCU(Snoop Control Unit)模块,因此在监听协

议的基础上进行讨论。

Cortex-A9 处理器提供的 SCU 是实现 cache 一致性的控制单元,并且处理 Zynq-7000 双核 L1 级 cache 的一致性。大多数 ARM 处理器采用的是基于 MOESI^[6] 或者 MESI 的一致性协议,本文采用基于 MOESI 协议的 SCU(Snoop Control Unit)方法,该功能的实现需要进行如下的配置:

- (1)SCU 模块是使能的。
- (2)处理器访问的数据位于内部共享区域,比如 Zynq-7000 上的 L1 级 cache。
- (3)Cortex-A9 双核处理器工作在 SMP 模式。该模式的特点是一个操作系统实例同时管理所有的 CPU,但是本论文运行的是 linux 与 FreeRTOS 异操作系统,两个系统管理的任务不一致。因此,需要将操作系统的控制范围局限在单核上,以此来匹配双核的工作模式和系统应用。其设置方式在于对协处理器 CP15:ACTLR 的 SMP 位的配置。一种配置方式如下:

```
MRC    p15, 0, r0, c1, c0, 1; //读取 ACTLR 信息
ORR    r0, r0, # 0x040;      //设置 bit[6] 确定 SMP
MCR    p15, 0, r0, c1, c0, 1; //写回 ACTLR
DSB
```

- (4)MMU 功能是使能的。
- (5)被访问的数据是具有 write-back^[7], write-allocate^[8] 的共享数据。

在相关寄存器配置完成后,SCU 以 MOESI 协议为基础,根据 cache Line 上的状态位,决定处理器的读写操作,以实现 L1 级 cache 的一致性。

3.3 任务消息捕获和分析

任务消息捕获是指 core0 核能读取到 core1 核上的任务数据,任务消息的捕获过程可以概括的描述为三个流程:确定要捕获的信息内容、指定存放空间、core0 读取。

确定要捕获的内容,立足于对 FreeRTOS 系统的分析,其内核的主要工作就是对任务进行管理和调度,从任务的存储结构上来看,任务由图 3 所示的三部分组成:任务程序代码,任务堆栈和任务控制块(TCB)。其中,任务控制块就是关联了任务代码的程序控制块,它记录了任务的基本属性,比如优先级、堆栈信息、任务名字、事件链表和状态链表信息等;任务堆栈则用来保存任务的工作环境;任务程序代码就是任务的执行部分。其中对任务监测的最主要

的就是获取任务控制块的信息。

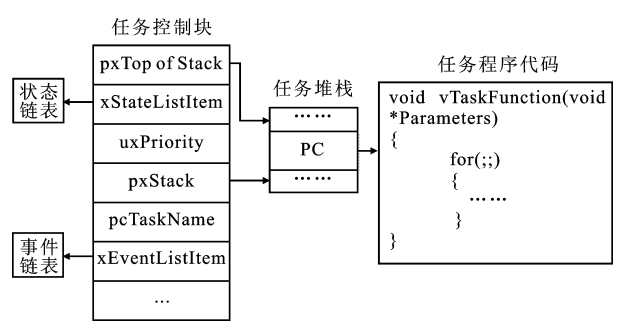


图 3 任务的存储结构

任务控制块提供了一个任务被创建时的基本信息,但调度器是通过状态链表去获取 TCB 信息的,因此还需了解 FreeRTOS 中链表与 TCB 的关系,链表和 TCB 的关系如图 4 所示。

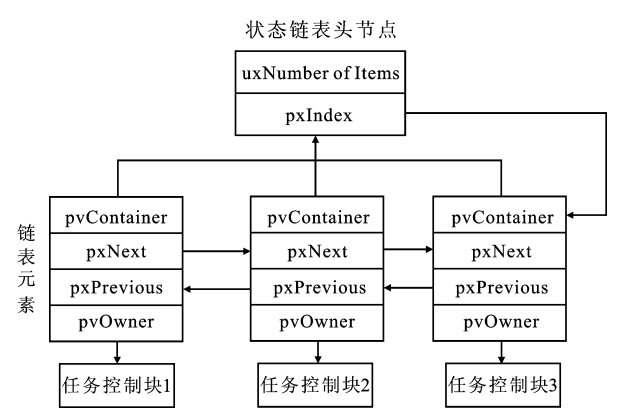


图 4 链表与任务控制块的关系

任务监测要获取实时的数据,所以选择状态链表的首地址作为存储内容,通过链表遍历的方法获取 TCB 中的信息。对 FreeRTOS 内核任务管理相关的源码分析,主要监测五类链表里的信息,已在表 1 中列出。由于核 0 和核 1 是独立,互不干扰的,各自尤其私有的存储空间。因此对于属于核 1 的状态链表的头指针信息,核 0 是无法获取的,所以得将状态链表的头指针地址存储在 core0 能获取到的地方,然后根据链表的结构进行解析,以获取任务相关的信息。利用 Zynq-7000 的开发工具 SDK,可以在链接器脚本 lscript.ld 中定义一块存储空间,用于 core0 访问 core1 的状态链表信息。

任务状态链表的元素是随着任务调度不断变化的,因此在共享空间中,不是存储状态链表的元素信息,而是存放各状态链表的头节点地址,当 core0 核获得地址信息的时候,根据表 1 的链表类型,按照图 4 的链表结构进行解析。任务消息数据的捕获和分析的操作流程如图 5 所示。

表 1 链表分类

链表名字	内容
pxReadyTasksLists	优先级就绪链表中的任务是等待运行的任务,处于就绪态.
xDelayedTaskList	阻塞链表中的任务表示该任务正在等待某个事件,处于阻塞态 该链表记录当调度器被挂起的时候,进入就绪态,但是没有放到就绪链表中的任务,等待调度器开始时重新进行一次调度
xPendingReadyList	
xTasksWaitingTerminationxSuspended-TaskList	表示等待结束的任务,该任务所占有的资源还未释放 表示该链表上的任务处于挂起态

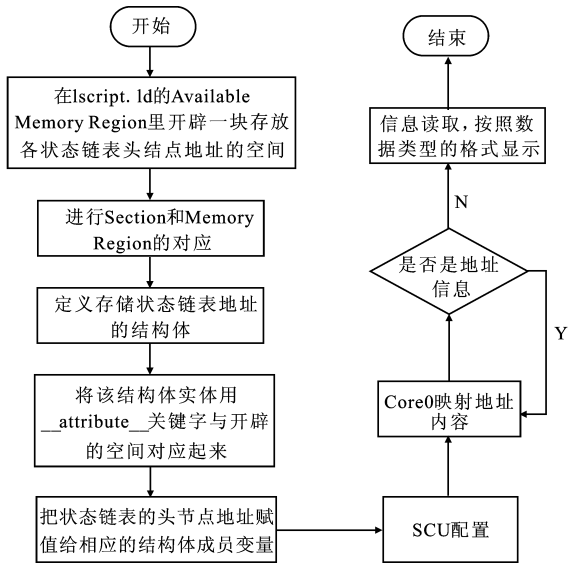


图 5 捕获和分析实现

4 测试结果和分析

4.1 测试平台说明

本文的研究是在 Xilinx 推出的 Zynq-7000 全可编程的可扩展平台上进行的.该平台由两部分组成,分别是集成了 ARM Cortex-A9 双核处理器的处理系统(Processing System,简称 PS)和 FPGA 可编程逻辑(Programmable Logic,简称 PL).测试只在双核处理系统上进行.处理器上搭载的操作系统结构是在 core0 核运行 Linux,core1 核运行 FreeRTOS,论文在对双核 L1 级 cache 一致性进行配置后,在 core0 上查看 core1 上的任务状态,以验证基于 cache 一致性的多核任务监测机制的正确性.

4.2 测试方案设计

由于任务监测机制主要是能正确反映双核上的任务信息,所以测试集中在对比任务信息的正确性上.由于 Linux 系统有丰富的任务查看命令,可借助查看命令对比监测结果;而 FreeRTOS 核的监测信

息是在 Linux 上显示的,需要在 core1 上做本地结果的对比分析.设计方案如下:

(1)让 Linux 和 FreeRTOS 在各自的处理器上运行自己的任务;

(2)采用 FreeRTOS 内核提供的查看系统状态的 API 做本地结果分析;

(3)为了验证在 FreeRTOS 核异常的情况下, Linux 系统仍然能够查看到任务状态,通过 SDK 调试平台,使 FreeRTOS 停止运作;

(4)通过 Linux 查看 FreeRTOS 在停止时刻,任务状态信息;

(5)与 core1 停止的内存信息对比.

4.3 结果验证

在 Zynq-7000 的 SDK 界面中可以得到通过串口打印的显示信息.首先看单核的任务状态信息,图 6 是 FreeRTOS 核任务状态信息.

Task Name	Priority	State	Top Stack	Bottom Stack
task 1	3	0	0x180 ce2d0	0x180 ca7c8
task 2	3	1	0x180 d2720	0x180 ce830
IDLE	0	1	0x180 dac30	0x180 da968
task 3	4	2	0x180 d6818	0x180 d2898
Tmr Svc	15	3	0x180 db3a0	0x180 dadf0

图 6 FreeRTOS 任务信息

再对比 Linux 上显示的 FreeRTOS 任务信息,如图 7 所示.

TCF Debug Process Terminal-P982					
task1	3	Running	0x180ce2d0	0x180ca7cb	
Scheduler State TickCount					
Running	716				
List Name	Task Name	Priority	State	Top Stack	Bottom Stack
Ready Lists	IDLE	0	Ready	180dac30	180da968
Ready Lists	task2	3	Ready	180d2720	180ce830
Delayed List1	task3	4	Block	180d0818	180d2898
Suspended List	TmrSvc	15	Suspended	180b3a0	180dadf0

图 7 Linux 上查看到的异核任务信息

其中图 7 的状态数字标识和图 8 中的状态对应关系如表 2 所示.

表 2 状态对比

数字状态标识	所指示的任务状态
0	Running
1	Ready
2	Block
3	Suspended

(下转第 28 页)

键问题综述[J]. 计算机研究与发展, 2012, 49(6): 1149-1161.

[2] Huang S, Zhou C J, Yang S H, et al. Cyber-physical system security for networked industrial processes[J]. International Journal of Automation & Computing, 2015, 12(6):567-578.

[3] Edward A Lee. CPS foundations[C]//Design Automation Conference. San Diego, California, USA, ACM, 2010:737-742.

[4] 钱志鸿, 王义君. 面向物联网的无线传感器网络综述[J]. 电子与信息学报, 2013, 35(1):215-227.

[5] Liu Y, Dong M, Ota K, et al. Active trust: secure and trustable routing in wireless sensor networks[J]. IEEE Transactions on Information Forensics & Security, 2017, 11(9):2013-2027.

[6] 郑国雄. 基于 SERCOS-III 网络系统高精度同步机制的研究[J]. 物联网技术, 2015, 4(2):40-42.

[7] 张伟功, 周继芹, 李杰, 等. UM-BUS 总线及接入式体

系结构[J]. 电子学报, 2015, 43(9):1776-1785.

[8] 李超, 邱柯妮, 张伟功, 等. 一种新型的动态可重构总线时间同步方法[J]. 微电子学与计算机, 2016, 33(7):60-63.

[9] 成立, 朱漪云, 王振宇, 等. 高速低耗 BiCMOS OC 门及其线与逻辑系统[J]. 江苏大学学报(自然科学版), 2007, 28(2):156-159.

作者简介:

万玛宁 女, (1981-), 硕士, 讲师. 研究方向为通讯总线、高可靠嵌入式系统. E-mail: wanmaning@cnu.edu.cn.

王莹 女, (1984-), 硕士. 研究方向为高可靠嵌入式系统与冗余容错技术.

周继芹 女, (1978-), 博士研究生. 研究方向为计算机系统体系结构与性能评价.

张伟功 男, (1967-), 博士, 研究员. 研究方向为高可靠嵌入式系统.

(上接第 22 页)

对比图 6 和图 7 的数据, 可以知道在实现双核 cache 一致性的情况下, 可以从 core0 核正确获取 core1 核的状态信息, 以反映系统的运行状态.

5 结束语

本文考虑到多核系统的任务不仅在本核上运行, 还会涉及到核间信息的交互, 采用软件实现的核间通信机制, 在某个内核异常的时候, 并不能有效的传递出该核的任务信息, 导致任务监测的失效. 本文提出基于 cache 一致性的多核任务监测机制以解决该问题. 并通过测试用例证明了该方法的可行性, 该方法可以避免多核间复杂的通信机制, 基于硬件提供的便利, 既降低了系统的复杂性, 也为异常情况下, 仍然能有效正确的获取核上的任务信息提供了便利, 为任务管理和异常处理提供了监测手段.

参考文献:

[1] 王欣, 邱昕, 慕福奇, 等. 基于 MCAP1 的嵌入式多核通信机制的研究[J]. 微电子学与计算机, 2017, 34(11): 85-88.

[2] 蔡玉鑫. 嵌入式多核处理器核间通信方法的设计与实现[D]. 西安: 西安电子科技大学, 2015.

[3] 郭海林, 陈香兰. 一种支持多级存储架构的嵌入式多核

操作系统模型[J]. 计算机系统应用, 2015, 24(12): 110-117.

[4] 孙荪. 提高多核处理器片上 Cache 利用率的关键技术研究[D]. 合肥: 中国科学技术大学, 2015.

[5] 刘文佳. 多核处理器 Cache 一致性研究与设计[D]. 哈尔滨: 哈尔滨工程大学, 2010.

[6] 黄安文, 张民选. 多核处理器 Cache 一致性协议关键技术研究[J]. 计算机工程与科学, 2009, 31(S1): 104-108.

[7] Dalui M, Som T, Bansal S, et al. Handling cache write-back and cache eviction for cache coherence [P]. WO2014022397, 2014-02-06. United States Patent Application 20140040561.

[8] Amer Diwan, David Tarditi, Eliot Moss. Memory system performance of programs with intensive heap allocation [J]. ACM Transactions on Computer Systems (TOCS), 1995, 13(3):244-273.

作者简介:

陈慧 女, (1993-), 硕士研究生. 研究方向为嵌入式系统. E-mail: chenhui@ciotc.org.

慕福奇 男, (1957-), 研究员, 博士生导师. 研究方向为无线通信.

张然 男, (1988-), 硕士, 工程师. 研究方向为无线通信.

姜彬 男, (1989-), 硕士, 工程师. 研究方向为嵌入式.