

基于 MCAPI 的嵌入式多核通信机制的研究

王 欣^{1,2},邱 昕¹,慕福奇¹,冷永清¹

(1 中国科学院 微电子研究所,北京 100029;2 中国科学院大学,北京 100049)

摘 要: 随着嵌入式系统的复杂性增加,多核处理器越来越多地被使用,多核间的通信机制的研究也成为热点之一. 针对嵌入式非对称多核处理器(asymmetric multiprocessing,AMP)的特点,并且为了方便上层应用移植,提出了基于多核通信接口框架(Multicore Communications API, MCAPI)的多核通信机制. 最后将该机制移植到 ZYNQ7000 进行实验测试,测试结果表明该通信机制可以满足实时嵌入式系统的需求.

关键词: 多核通信;MCAPI;AMP;ZYNQ7000

中图分类号: TP368.1

文献标识码: A

文章编号: 1000—7180(2017)11—0085—04

Study of MCAPI-Compliant Communication Mechanism in Embedded Multi-core System

WANG Xin^{1,2}, QIU Xin¹, MU Fu-qi¹, LENG Yong-qing¹

(1 Institute of Microelectronics, Chinese Academy of Sciences, Beijing 100029, China;

2 University of Chinese Academy of Sciences, Beijing 10049, China)

Abstract: With the increasing complexity of embedded systems, multi-core processors are increasingly being used, and multi-core communication mechanism also has become one of the hot spots. For the characteristic of embedded multi-core processors architecture, this proposes a communication mechanism that based on MCAPI to improve the portability. Then conducting the experimental test on the ZYNQ7000 platform. At last, the results show that this mechanism can meet the need of real-time embedded system.

Key words: multi-core communication; MCAPI; AMP; ZYNQ7000

1 引言

近年来,随着智能硬件、物联网行业的迅猛发展,嵌入式系统在各个领域都得到了广泛的应用. 为了适应日渐复杂的应用需求,并且降低系统的功耗,嵌入式系统开始采用多核处理器作为硬件平台,但支持多核处理器的实时嵌入式操作系统不多^[1],且支持多核的嵌入式 Linux 又不能满足嵌入式系统的实时性要求,所以采用非对称多处理(asymmetric multiprocessing, AMP)的操作系统环境,在多个核上运行不同的操作系统成为了多核嵌入式实时系统的主流,如 TI 的 OMAP 系列等.

由于当前主流的嵌入式操作系统 UCOS、FreeRTOS 等都只针对单核系统,即各个处理器内核都运行各自独立的操作系统,所以 AMP 模式下,多核间的通信就成为了多核实时嵌入式系统需要解决的主要问题.

TI 的 sys/bios 实时操作系统的核间通信组件(inter-processor communication, IPC)^[2] 提供了 AMP 多核通信机制,但是并不开源,使用该组件的应用移植到其他厂商的平台并不方便;VxWorks 操作系统^[3]采用类似于 Windows Socket 的方式,允许进程不必通过共同的地址空间共享来进行通信,并且屏蔽了底层硬件,但也不开源;RTEMS 操作系

收稿日期: 2017—02—10; 修回日期: 2017—03—20

基金项目: 中科院国防科技创新基金(CXJJ-16M124);国家自然科学基金青年科学基金(61501455);北京市自然科学基金面上项目(4162068)

统^[4]采用包传递的通信方式,每个处理器建立一个 MPCl server(服务器线程),所有的消息传递都要经过该线程进行处理,效率不够高.

因此,本文从高效并且易于移植的角度出发,提出了一种基于 MCAPI^[5] 框架的 AMP 多核通信机制,基于此框架,在适应嵌入式应用的基础上,仍然可以保持灵活的性能. 本文的机制将实现分为移植层和逻辑层,移植层结合相应的操作系统,不仅多核间消息传递高效,且不影响内核的任务调度;逻辑层接口即为 MCAPI,减轻上层软件的开发和移植负担.

因为 AMP 架构下,软件环境一般为嵌入式 Linux 和实时嵌入式操作系统结合,所以本文分别以 Linux 和 FreeRTOS 为例介绍了该通信机制的实现,并在 Xilinx 的 ZYNQ7000 系列平台上进行了移植测试.

2 MCAPI 介绍

MCAPI^[5-6] 是一个由 The Multicore Association 定义的轻量级开源多核通信框架,提供一套标准 API 接口,旨在支持基于消息传递形式的嵌入式多核通信核同步编程.

在 MCAP 中定义了域(Domain)、节点(Node)、端点(Endpoints)和通道(Channel)等概念. 其中,域由多个节点组成,其依赖于具体实现,既可以是具有多个处理器核的单个芯片,也可以是板上的多个处理器. 节点可以是线程、处理器或硬件加速器等,通常将处理器核心作为节点. 端点类似于套接字(socket)的通信点,使用<domain node port>三元组进行标识,并且拥有一系列属性(attribute),包括服务质量、缓存和超时等. 通道则是端点之间点到点的单向先入先出队列(FIFO).

另外,MCAPI 提供了 3 种基本的通信方式:消息(Messages)、包通道(Packet Channels)和标量通道(Scalar channel),3 种方式都提供阻塞和非阻塞调用. 其中,消息是最常用的通信类型,不建立连接,提供了端点之间收发数据的机制,支持优先级动态调整,可用于同步、初始化和负载平衡,如图 1 所示. 包通道和标量通道提供了轻量级的流通信机制,因为建立连接,传输的数据不需要携带收发端口的相关信息,所以相应的程序开销很小. 如图 2 所示.

3 通信机制的实现

3.1 系统架构

本文提出的基于 MCAPI 的多核通信机制^[7],

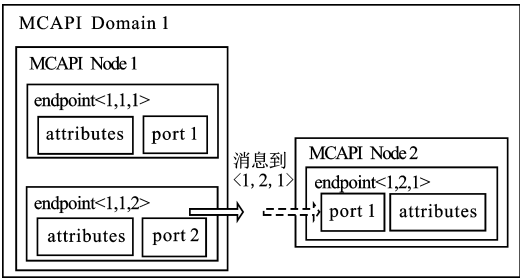


图 1 MCAPI 消息

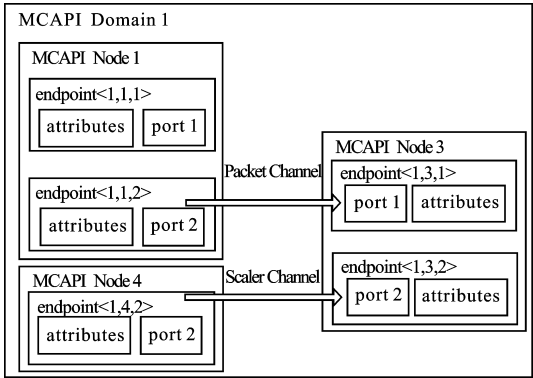


图 2 MCAPI 通道

主要分为逻辑层、操作系统移植层、传输层和移植层,如图 3 所示.

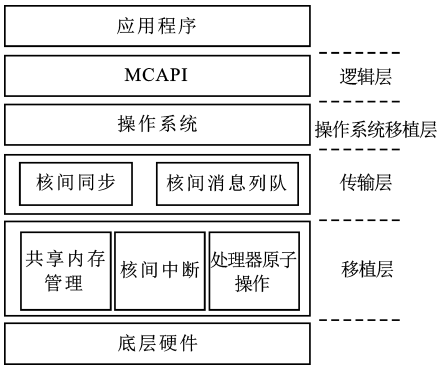


图 3 多核通信机制架构

其中,移植层与硬件架构有关,主要功能是共享内存管理、处理器支持的原子操作以及核间中断. 通过共享内存管理与原子操作实现基本的核间消息队列;使用核间中断进行消息发送通知和接收后的反馈.

传输层主要负责核间同步和核间消息队列实现,是 MCAPI 各种通信方式实现的基础.

操作系统移植层则包括两部分:一部分向下通过系统调用来进入传输层;另一部分向上提供进程控制和进程间通信,为 MCAPI 逻辑层服务.

而 MCAPI 的逻辑层,主要负责消息、包通道和

标量通道的具体逻辑实现,并为上层应用提供标准的 API 接口.

3.2 Linux 机制实现

Linux 作为宏内核,被分为内核态与用户态,其中进程调度和内存等硬件管理都需要在内核态,所以将需要操作共享内存和核间中断的移植层处于内核态,将操作系统移植层则实现在用户态,实现机制如图 4 所示.

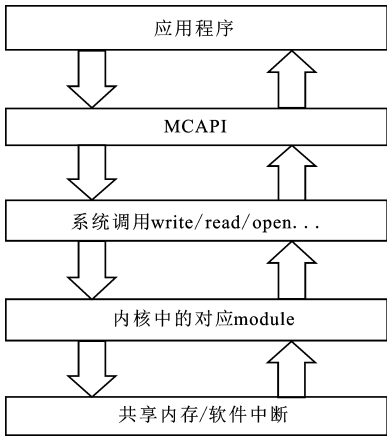


图 4 Linux 机制实现

移植层以模块(module)的形式被接入内核.当发送消息时,上层应用首先调用相应的 MCAPI 接口,该接口通过系统调用 write() 进入内核态,在内核态将需要发送的数据放入对应的核间消息队列中,然后触发对应的核间中断通知相应的核接收数据;接收消息时,则是在绑定的核间中断服务函数里从对应队列中取得数据,放入本地实现的队列中,上层应用通过系统调用 read() 获取数据.

当发送或者接收需要阻塞时,使用 Linux 内核的等待队列(wait queue)实现,将相应进程变为可中断睡眠状态(interruptible state);在核间中断服务函数中再唤醒相应进程.

3.3 FreeRTOS 机制实现

FreeRTOS 作为微内核,没有内核态与用户态之分,所谓的内核仅仅实现了任务调度、任务间通信和内存管理等操作系统基本功能,所以通信机制实现相对于 Linux 更容易,实现机制如图 5 所示.

其中,底层驱动(driver)即为移植层和传输层实现,即进行消息传输,以及软件中断的触发响应.上层应用程序调用 MCAPI 接口,该接口直接调用底层驱动来进行核间通信.

当任务需要阻塞时,则是利用 FreeRTOS 队列初始化时生成的相应事件列表(event list),将需要

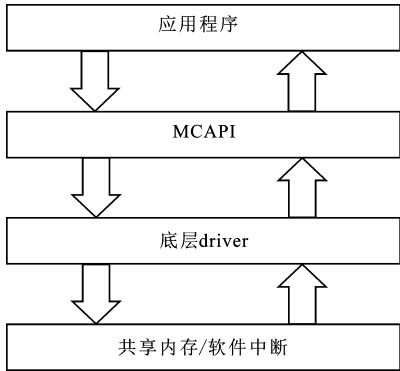


图 5 FreeRTOS 机制实现

被阻塞任务的任务控制块(task controller block, TCB)从运行列表(running list)移除并放入对应的事件列表,从而实现任务从运行态(running state)进入阻塞态(blocked);而需要唤醒时,则重新被放入运行列表.

4 移植与测试

4.1 通信机制移植

本文将该通信机制移植到了 ZYNQ7000 开发平台上,其中 ZYNQ7000 平台^[8]是 Xilinx 公司推出的一款集成 ARM Cortex-A9 双核处理器和 FPGA 可编程逻辑资源的 SoC.其中,核 0 运行 Linux,核 1 运行 FreeRTOS,主频均为 666 MHz,L1 L/D cache 均为 32 kB,DDR3 为 1 GB 且频率为 553 MHz.根据文献[7,9],多核通信性能主要集中在 3 个指标:占用内存大小,通信延迟,流量(数据速率).

4.2 性能测试与结果分析

首先分析整个机制所需要占用的内存大小,结果如表 1 所示.

表 1 占用内存大小			kB
操作系统	模块	链接库	总大小
Linux	11.6	122.3	133.9
FreeRTOS	无	83.4	83.4

由数据可知,对于使用多核处理器的嵌入式系统,该机制只需要占用大约 230 kB 的内存,并不会对嵌入式系统产生负担.

而针对通信延迟的测试,首先通过调用阻塞接收来使某个接收进程阻塞,在另一核发送消息后,被阻塞的接收进程会从阻塞态变为运行态,通过统计发送 1-8 kB 消息的状态下,从发送消息到另一核被阻塞进程从阻塞态恢复运行的平均时间,确定出相应的通信延迟,最终结果如表 2 所示.

表 2 通信延迟 μs

发送方向	平均时延
Linux $\rightarrow$ FreeRTOS	4.6
FreeRTOS $\rightarrow$ Linux	11.3

根据实验结果,FreeRTOS 的时延大约为 4.6 μs ,因为测试得到 FreeRTOS 单核运行时任务切换的时间约为 1~2 μs ,所以多核通信机制下时延 4.6 μs 满足嵌入式系统需求.而 Linux 由于接收到数据,需要从内核态切换到用户态,且 Linux 本身的非实时性,所以延时相对 FreeRTOS 要大.

流量测试与时延测试类似,统计不同类型下调用发送数据 API 发送 1~4 MB 数据到接收核接收到数据的时间,计算出对应的流量,最终结果如表 3 所示.因为标量通道只能发送 8-64 bit 数据,不具有代表性,所以在测试中忽略.

表 3 流量

数据类型	数据大小	数据速率/(MB/s)
消息	512B	266.05
	4 kB	364.60
	32 kB	385.57
	4 MB	396.01
包通道	512B	326.83
	4 kB	394.04
	32 kB	405.00
	4 MB	409.92

因为包通道是建立连接的通信方式,所以发送数据中不需要携带标识发送接收端点的信息,所以通信效率比消息要高,且二者的测得的流量大约为 300~400 MB/s,满足嵌入式系统的一般需要.

5 结束语

本文提出了一种嵌入式处理器多核 AMP 模式通信机制,并为了提高移植性,采用了 MCAPI 开源架构,将通信机制分为逻辑层、操作系统移植层、传输层和移植层.并将该机制移植到了 ZYNQ7000 平台上进行测试.测试结果表明:该机制拥有很低的延迟和很好的数据速率,可以满足实时嵌入式系统的性能要求.

参考文献:

[1] 肖学甲. 基于 AMP 架构的多核间任务同步与通信的设计与实现[D]. 西安:西安电子科技大学, 2011.

[2] Texas Instruments. SYS/BIOS inter-processor communication (IPC) and I/O user's guide[Z]. Texas Instruments, 2011.

[3] Wind River. VxWorks AMP programmer's guide 6.7 [Z]. Wind River, 2008.

[4] 牛莹皎. 支持多核处理器的 RTEMS 嵌入式操作系统的研究[D]. 哈尔滨:哈尔滨工程大学, 2013.

[5] Prophet G. Multicore Association's MCAPI 2.0 specification adds multicore messaging flexibility[J]. Edn Europe, 2011.

[6] Mentor Graphics. Multicore communications API (MCAPI) specification V2.015[Z]. Mentor Graphics, 2015.

[7] Virtanen J, Matilainen L, Salminen E, et al. Implementation of multicore communications API[C]// International Symposium on System-on-Chip. IEEE, 2014:1-6.

[8] Crockett L H. The Zynq book : embedded processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 all programmable SoC [M]. Strathclyde Academic Media, 2014.

[9] Clauss C, Pickartz S, Lankes S, et al. Towards a multicore communications API implementation (MCAPI) for the intel single-chip cloud computer (SCC)[C]// International Symposium on Parallel and Distributed Computing. IEEE Computer Society, 2012:148-155.

作者简介:

王欣 男,(1992-),硕士研究生.研究方向为嵌入式系统. E-mail:wangxin3@ime.ac.cn.

邱昕 男,(1980-),博士,副研究员.研究方向为无线通信.

慕福奇 男,(1957-),研究员.研究方向为无线通信.

冷永清 男,(1981-),博士,助理研究员.研究方向为微波固态功率技术、高效率功率放大器、微波发射机和接收机.