基于异构嵌入式计算平台的 PCIe 通信软件设计

田 浩[1] 马 超[1] 王 晨[1]
TIAN Hao MA Chao WANG Chen

摘 要 随着嵌入式领域处理器性能的高速发展，越来越多的嵌入式平台通过异构多核处理器（heterogeneous multi-processor unit）架构进行复杂度更高的计算工作，但多核处理器的核间通信效率对平台的高性能计算能力有着很大的影响。首先对异构嵌入式计算平台在多处理器之间的通信所存在的这一问题进行了分析，对硬件所存在的特定的场景进行了考虑，最后实现了平台上处理器之间基于PCIe总线的通信驱动。驱动在强调毫秒级高实时性的应用场景下，通过 PCIe 双缓冲区的设计，不仅保证了高可靠性，还具备了低延迟和高传输速率，为后续基于平台的计算密集型软件应用提供了良好的基础。

关键词 PCIe；Linux；RTOS；异构

doi：10.3969/j.issn.1672-9528.2024.04.016

0 引言

异构嵌入式计算平台是一种拥有着多个不同架构处理器内核的计算平台，它的架构可以根据不同的嵌入式领域的具体应用来进行针对性的设计，因此受到各个行业的广泛关注。在一些特定的应用场景下，异构计算平台的处理器之间需要以微秒级进行频繁的信息交互，如何准确高效地在多个处理器核心之间进行通信成为一个亟待解决的关键问题 [1]。

PCI-Express（peripheral component interconnect express，PCIe）是一种高速串行计算机扩展总线标准，与它的前代 PCI 和 PCI-X 相比，有着更高的速度和性能，并且可扩展性和兼容性也更加优秀，由于其高可靠性、低延时和高传输速率，PCI Express 被广泛用于嵌入式异构平台上 [2]。

本文软件所使用的嵌入式异构计算平台架构采用了两个处理器的架构：某款面向桌面应用的通用 4 核处理器作为主处理器，某款集成了可编程逻辑（programmable logic，PL）以及 AI 加速引擎的高性能 4 核处理器作为协处理器。通过上述对相关文献的阅读，本文采用 PCI Express 总线来实现这两个处理器之间的数据交互，即使用 PCI Express 总线作为平台处理器之间的通信总线（后续简称 PCIe 总线）。

1 软件总体架构

对于本文的计算平台而言，它的不同处理器所采用的操作系统（operation system，OS）并不一致。针对这一问题，本文通过不同 OS 的运行方式以及不同 OS 下的软件对物理地址、内存地址的操作方式等方法进行分析，以分析结果为基础，以此设计平台上两个处理器之间的 PCIe 通信驱动软件。

本文平台主处理器的操作系统采用了某一款可靠性高的嵌入式实时操作系统（real-time operating system，RTOS），其提供了分区调度、通信等功能，能够有效地降低分区应用间的相互干扰，以此来保证实时性，能够及时响应高优级的任务，确保数据收发的稳定 [3]。协处理器采用了 Linux OS 作为其操作系统，其作为一款嵌入式操作系统，具有免费开源、模块化程度高、性能安全稳定、可移植性高等优点，与其他嵌入式操作系统相比，Linux 对比大部分 AI 框架的兼容性都做得更加出色，在运行 AI 软件方面有着简洁高效的优势 [4]。因此，协处理器使用 Linux OS 保证更高的性能，用来支持计算密集型应用的稳定运行。

在上文中提到，平台的两个处理器之间通过 PCIe 总线进行通信。具体来说，PCIe 总线可以挂接各种不同的设备来形成一个拓扑结构，这些设备主要分为两种模式，分别为 RC（root complex）和 EP（endpoint），这两种设备类型在 PCIe 拓扑结构中扮演着不同的角色 [5]。RC 设备是 PCIe 拓扑结构的起始点，用于连接 CPU/ 内存子系统和 I/O 设备，主要功能有完成存储器域到 PCIe 总线域的地址转换、物理信号的转换等；EP 设备是 PCIe 拓扑结构的终点，通常表示一个串行或者 I/O 设备，其为系统提供了灵活性和可扩展性，可以根据不同需求来挂接不同类型的设备。

本文的架构将主处理器作为 PCIe 结构中的 RC 设备（可以抽象地理解整个主处理器就是 RC），协处理器作为 PCIe 结构中的 EP 设备，以此来响应 RC 发送过来的请求命令，一般包括数据的读写等。综上，本文通过 PCIe 总线上 RC 和 EP 之间的通信来实现主协处理器之间数据的交互，所以

1. 航空工业西安航空计算技术研究所 陕西西安 710068

PCIe 通信软件的架构可以分为两个部分：主处理器通信软件功能主要包括对 PCIe 总线下的设备进行初始化、发送数据到协处理器、接收协处理发送过来的数据等；协处理器通信软件功能主要包括数据的收发以及虚拟地址到物理地址的转换等。软件架构如图 1 所示。

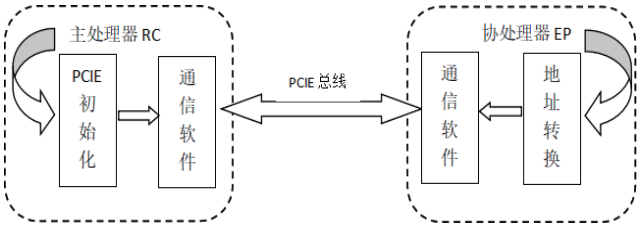

图 1 PCIe 通信软件架构图

2 主处理器通信软件设计

2.1 PCIe 初始化

本文在上一章节中提到，主处理器的操作系统选择了一款 RTOS 而不是功能更加强大的 LinuxOS，这主要是由于 LinuxOS 虚拟内存的访问时间不可控，而本文中的主处理器需要在毫秒级的周期任务里准确地进行任务的收发。为了避免这一问题，选择了不支持虚拟内存的硬实时系统。这也导致了一些缺点：比如有多少物理内存就只能用多少，比起非实时操作系统的灵活度要差很多 [6]。

主处理器作为整个 PCIe 通信架构的 RC，扮演着收发和转发数据的重要角色。如果主处理器 RC 要准确地与作为 EP 的协处理器进行通信，需要访问到 EP 的 BAR（base address register）空间。BAR 空间用于映射设备内存或者 I\O 资源的一种机制，为了能够获取协处理器的 BAR 空间基地址，需要通过对 PCIe 上挂接的设备进行扫描，以此来确定设备的设备号、总线号、函数号，这是设备在总线上的唯一标识 [7]。

由以上可知，主处理器在通过 PCIe 进行数据收发之前，要先对 PCIe 总线上的设备进行初始化扫描，确定每个设备的 busNo、deviceNo、functionNo，具体流程如下。

PCIe 初始化功能流程：

1 获取当前总线号；
2 循环最大 PCIe 总线设备次；
3 循环最大 PCIe 总线功能次；
4 如果需要跳过该设备扫描，
5 跳出本次循环扫描；
6 如果没有找到多功能设备
7 则继续下一次循环查找；
8 读取 PCIe 配置头类型；
9 读取 PCIe 设备的 vendorId；
10 若 vendorId 不为 0xffff 或者 vendorId 不为 0；
11 则继续下一次循环查找；
12 若功能 ID 不为 0，
13 置 PCIe 设备多功能标志位为多功能；
14 读取设备的 deviceId；
15 读取设备的 CLASS_DEVICE；
16 记录 PCIe 设备信息；
17 调用 PCIe 总线自动配置函数，获取总线号；
18 若总线号大于当前下游总线号，则获取到的总线号为下游总线号；
19 否则取当前总线号为下游总线号。

在上述的 PCIe 初始化之后，就可以得到主处理器在 PCIe 总线上的设备号、总线号、函数号，以此得到其 PCIe 空间的基地址。

接下来，主处理器要实现 PCIe 的数据收发功能，数据的搬运通过处理器的 FPGA 实现，软件需要下达数据搬运的命令以及通过读取相应状态寄存器的值判断搬运是否结束。为了保证数据的实时性，软件做了一个双数据缓冲区的设计。收发数据各自拥有两块 PCIe 空间作为其数据的缓冲区，通过读取 FPGA 相应的寄存器来判断是否有搬运新数据，以此来进行两个缓冲区之间的切换。

2.2 主处理器数据收发

本文的 PCIe 通信驱动主要有收和发两个部分，其中发送指令数据通过判断 FPGA 上的状态寄存器来判断上一次发送是否完成，若上一次发送完成，则将数据搬运到主处理器平台 FPGA 相应的 PCIe 缓冲区，通过读写 FPGA 中的寄存器来控制其进行数据的搬运，具体流程如下。

PCIe 通信发送驱动功能流程：

1 定时判断 FPGA 中的状态寄存器有效位是否重新置为 0；
2 如果没有重新置为 0，
3 则返回错误值；
4 如果重新置为 0，
5 将待发送输入放入 PCIe 的指定传输空间中；
6 如果传入 ID 号无效，
7 则返回错误值；
8 如果传入 ID 号有效，
9 将 ID 号和数据长度写入发送数据的 FPGA 寄存器，并将发送位置 1。

对于主处理器 PCIe 通信的接收驱动来说，软件需要去读取逻辑提供的寄存器的值，判断当前收取到的数据是否有效，如果有效，通过相应寄存器去确定需要搬运数据的大小，最后从逻辑的 PCIe 缓冲区将数据搬运出来。本文设计的通信

驱动是周期收发的，为了避免频繁读取逻辑的寄存器，本文设计了一个结构体用来存储有效数据的总数、ID 值以及长度，这样只需要读取以此判断数据有效的寄存器，减轻了 CPU 负载压力。

PCIe 通信接收驱动设计分为两个部分，第一部分通过读取逻辑的数据有效寄存器来获取有效数据的 ID、长度，并将其保存到构造好的结构体中；第二部分通过结构体中有效的数据 ID 去判断当前数据是否有效，获取其数据长度，最后在相应的 PCIe 缓冲区去搬运数据，具体流程如下。

PCIe 通信接收驱动功能流程：

1 初始化有效接收数据结构体；

2 如果当前 ID 值无效

3　则返回错误值；

4 如果当前 ID 值有效，

5　获取 ID 对应的数据长度；

6　将对应大小的数据从 PCIe 缓冲区搬运到内存；

7　返回正确值。

3 协处理器通信软件设计

3.1 地址空间转换

在 Linux 操作系统下，为了避免多个进程之间的内存地址相互影响，例如一个程序修改了另一个程序使用的地址，从而导致该程序崩溃，引入了虚拟内存的概念。通过操作系统单独为每一个进程分配一套独立的虚拟地址，使进程之间相互隔离开来 [8]。

相比较于单片机设备可以直接通过在代码中读写物理地址来操作寄存器，在 Linux 的用户空间如果需要访问真实的物理地址，就要配置一套虚拟地址到物理地址的映射机制，根据这样的需求，本文根据 LinuxOS 的特性实现了这一套机制。

PCIe 通信功能需要访问的物理地址有两个部分，一个是 FPGA 寄存器所在的物理地址，另一个是用来暂存 PCIe 数据的缓冲区。在 Linux 的文件系统中，其外部设备的访问端口全部以文件的形式存放在 /dev 目录下，本文通过外部设备驱动的形式对两段物理地址进行封装，分别为 dev/udmabuf0 和 dev/udmabuf1。通过 Linux 系统函数 open 去打开这两个文件，通过 mmap 系统函数去实现这两段物理地址到 Linux 用户空间地址之间的映射 [9]。

通过调用 mmap 函数可以对设备进行映射，函数结果会返回一个指向对应物理地址的指针，通过这一指针去实现读写相应物理地址的接口 [10]。

对于 udmabuf0 这一设备来说，本文主要是去读写和 FPGA 有关的寄存器的物理地址，所以这里设计了 readReg、writeReg 这两个接口用于对相关寄存器的读写。readReg 作为一个读取寄存器的接口，它的参数为寄存器的地址，返回值为相应地址的内容，对于其传入的参数，需要判断其是否在正确的地址上，具体流程如下。

readReg 功能流程：

1 对传入的地址做以判断；

2 如果地址在正确的地址范围内，

3　返回对应地址 mmap 指针；

4 如果地址不在正确的地址范围内，

5　返回一个错误值。

writeReg 作为一个写寄存器的接口，参数有两个，一个是寄存器的地址，另一个是写入寄存器的内容（写入值的类型需要根据寄存器的位数加以限制），无返回值。对于其传入的参数，需要判断其是否在正确的地址上，具体流程如下。

writeReg 功能流程：

1 对传入的地址做以判断；

2 如果地址在正确的地址范围内，

3　返回对应地址 mmap 指针；

4 如果地址不在正确的地址范围内，

5　返回一个错误值。

根据上文描述，udmabuf1 设备所封装的物理地址是作为缓冲区提供给 FPGA 的，用于缓存 FPGA 从 PCIe 搬运过来的数据。本文设计了两个接口即 readMem 和 writeMem，用来读写这一段作为缓冲区的物理地址。为了保证读数据和写数据之间不产生影响，这里设计了两个缓冲区，将需要读写的数据分别放在不同的缓冲区中，两个缓冲区之间的间隔需要根据读写数据的最大 size 对其缓冲区首地址进行偏移。

readMem 作为一个读取缓冲区数据的接口，负责将 FPGA 搬运到 PSDDR 物理地址的数据映射到内存的虚拟地址中，以此实现内存中的应用对其的读写。函数接口总共有三个参数：第一个是用户空间下用于接收数据的缓冲区虚拟地址，第二个是读取数据的物理地址偏移，第三个是读取数据的大小，如图 2 所示。

```
/**
 * 将数据从物理地址转换到linux下的虚拟地址
 *
 * @param  dstBuf              虚拟地址指针
 * @param  offset              物理地址的偏移
 * @param  size                数据大小字节数
 */
void readMem(unsigned char *dstBuf,unsigned int offset,unsigned int size)
```

图 2 readMem 函数接口示例

writeMem 作为一个向缓冲区写入数据的接口，负责将内存中用户处理后的数据写入 PSDDR 的物理地址中，以此实现虚拟地址到物理地址的转换。函数接口总共有三个参数：第一个参数是用户空间下准备写入数据的虚拟首地址，第二个参数是向 udmabuf1 偏移多少的地址写入数据，第三个参数是写入数据的大小，如图 3 所示。

```
/**
 * 将数据从linux下的虚拟地址转换到ps_ddr物理地址
 *
 * @param  srcBuf              虚拟地址指针
 * @param  offset              物理地址的偏移
 * @param  size                数据大小字节数
 */
void writeMem(char *srcBuf,unsigned int offset,unsigned int size)
```

图 3 writeMem 函数接口示例

3.2 协处理器数据收发

协处理器主要软件层实现了数据通过 PCIe 链路的收发工作。与主处理器 PCIe 通信软件类似，协处理器的通信驱动也分为收取和发送两个部分。对于整个软件架构来说，主处理器通过 PCIe 将数据发送到协处理器，协处理器接收后再将处理结果返回主处理器。对于协处理器的 PCIe 接收功能来说，它分为两个部分，首先是读取 FPGA 的寄存器去判断数据是否有效，若有效则通过写寄存器去控制 FPGA 搬运数据到相应的 PSDDR 物理地址，最后通过 readMem 接口将数据所在的这一段物理地址映射到虚拟地址，具体流程如下。

PCIe 通信接收功能流程：

1 判断接收的数据 ID 是否有效；

2 如果接收的数据无效，

3 返回错误值；

4 如果接收的数据有效，

5 判断上一次搬运是否结束；

6 如果上一次搬运未结束，

7 返回错误值；

8 如果上一次搬运结束，

9 向逻辑寄存器中写入搬向的PSDDR的物理地址，

10 向逻辑寄存器写入搬运命令。

对于协处理器的 PCIe 发送功能来说，它首先是读取 FPGA 的寄存器去判断上一次发送过程是否结束，如果结束，则进行新一次的搬运。通过调用 writeMem 接口，把需要发送的数据搬运到 PSDDR 物理地址所对应的虚拟地址上，操作系统会将数据映射到真实的 PSDDR 的物理地址，具体流程如下。

PCIe 通信发送功能流程：

1 判断上一次发送过程是否结束；

2 如果上一次发送未结束，

3 返回错误值；

4 如果上一次发送结束，

5 把需要发送的数据搬运到 PSDDR 物理地址所对应的虚拟地址上。

4 结语

本文基于异构嵌入式计算平台在多处理器之间的通信所存在的痛点问题进行了分析，对硬件所存在的特定的场景进行了考虑，最后基于 PCIe 总线实现了平台上多处理器之间的通信，在强调毫秒级高实时性的应用场景下，通过双缓冲区的设计，不仅保证了高可靠性，还具备了低延迟和高传输速率，为后续软件应用提供了良好的基础。

参考文献：

[1] 王博 . 一种基于异构计算平台的 FPGA 动态重配置方法 [J]. 电子质量 , 2019(1):23-29.

[2] 徐健 , 张建泉 , 张健 . 基于 PCIe 非透明桥的嵌入式异构平台设计 [J]. 微电子学与计算机 ,2018,35(1):26-30.

[3] 郝玉锴 , 戴小氐 , 崔西宁 . 综合化模块化航空电子架构航电系统飞行管理模块的设计 [J]. 科学技术与工程 , 2021, 21(16): 6923-6929.

[4] 吴松 , 王坤 , 金海 . 操作系统虚拟化的研究现状与展望 [J]. 计算机研究与发展 ,2019,56(1):58-68.

[5] 杨亚涛 , 张松涛 , 李子臣 , 等 . 基于 Zynq 平台 PCIe 高速数据接口的设计与实现 [J]. 电子科技大学学报 , 2017, 46(3): 522-528.

[6] 吴良顺 , 张斌 , 应忍冬 . 实时响应的嵌入式系统虚拟化微内核架构 [J]. 自动化与仪器仪表 ,2023(7):219-221+239.

[7] 张象羽 , 施慧莉 . 基于以太网和 PCIe 的多核 DSP 开发平台 [J]. 计算机工程与科学 ,2019,41(10):1731-1737.

[8] 鲍中 , 徐嵩皓 , 鲍广建 .Linux 下 PCI-Express 驱动研究 [J]. 电子质量 ,2023(8):16-21.

[9] 何景波 . 基于 Linux 的嵌入式应用系统技术研究 [D]. 太原 : 中北大学 ,2009.

[10] 蒋丰 , 张春元 , 王巧云 .Linux 下实现设备驱动程序的物理内存静态分配 [J]. 计算机应用研究 ,2001(7):147-148.

【作者简介】

田浩（1995—），男，陕西富平人，硕士，助理工程师，研究方向：嵌入式软件。

马超（1987—），男，安徽肥东人，硕士，高级工程师，研究方向：嵌入式软件。

王晨（1997—），男，陕西渭南人，硕士，助理工程师，研究方向：计算机应用。

（收稿日期：2024-01-10）