

基于 PCIE 非透明桥的嵌入式异构平台设计

徐 健^{1,2}, 张建泉^{1,2}, 张 健^{1,2}

(1 中国科学院 空间应用工程与技术中心, 北京 100094; 2 中国科学院 太空应用重点实验室, 北京 100094)

摘 要: PCI Express(PCIE)总线因其具有传输带宽高、全新的点点互连架构和对 PCI 总线高度兼容等优点已在计算机平台中获得广泛应用. 在通过 PCIE 桥接两个以上处理器系统的场合, PCIE 非透明桥(NTB)有效地隔离的桥两端系统的 PCIE 总线域, 防止桥两端寻址冲突. 本文使用 PCIE 交换芯片(PCIE Switch)以非透明桥的方式连接 CPU 和双 GPU, 搭建高性能嵌入式异构计算平台. 在软件层设计上, 本文基于 Linux 操作系统, 完成了 PCIE 总线驱动和 PCIE Switch 的驱动设计, 实现了多个 Root 之间的高速通信.

关键词: PCIE; 非透明桥; 交换芯片; 嵌入式; 异构平台

中图分类号: TN61-1123

文献标识码: A

文章编号: 1000-7180(2018)01-0026-05

An Application of PCIE Non-transparent Bridge on A Heterogeneous Platform

XU Jian^{1,2}, ZHANG Jian-quan^{1,2}, ZHANG Jian^{1,2}

(1 Technology and Engineering Center for Space Utilization, Chinese Academy of Science, Beijing 100094, China ;
Key Laboratory of Space Applications, Chinese Academy of Science, Beijing 100094, China)

Abstract: PCI Express (PCIE) bus has been widely used in the computer platform because of the high bandwidth, the point-to-point interconnect architecture, highly compatible to PCI bus and so on. When more than two processor systems are connected by PCIE bridge, PCIE non-transparent bridge (NTB) effectively isolates the PCIE bus domain of both ends of the bridge, preventing conflict when addressing. In the paper, PCIE switch connects the CPU and double GPU by way of a non-transparent bridge to build the embedded heterogeneous computing platform of high-performance. In the design of software layers, the PCIE bus driver and PCIE Switch driver design is completed based on the Linux operation system. At last, the paper achieves a high-speed communication between a plurality of Root.

Key words: PCIE; non-transparent bridge; switching chip; embedded

1 引言

对于嵌入式异构平台而言, 相比于 CPU+单 GPU(两个 Root)架构, CPU+多 GPU(多个 Root)架构会有更高的计算能力和更大的存储带宽. 因此高性能的嵌入式异构计算一般使用 CPU+多 GPU 的架构, 即系统由 CPU 和多个协处理器 GPU 组成. 相比于 CPU+多 Endpoint 系统, 当多个 GPU 挂载到总线上后, 总线的传输压力增大, 这直接会引

起数据传输延时增大. 特别是在有些算法中, 需要 CPU 和 GPU 之间频繁进行数据交换, 数据传输延时将会更为严重. 因此总线的性能对嵌入式异构平台的性能起到决定性的作用.

PCI Express 是继 PCI 和 PCI-X 总线后的第三代 I/O 互连技术, 无论是在速度、性能、功能, 还是可扩展性和兼容性等各方面, 与之前相比都有了显著改进和提高, 由于其高可靠性、低延时和高传输速率, PCI Express 广泛用于嵌入式异构平台上. 与

PCI 总线不同的是,PCI Express 采用基于报文交换的点对点串行传输技术,为每个设备分配独立通道,所有设备均通过各自独立的通道发送和接受数据^[1].正因为这种特性,PCI Express 很容易实现两个 Root 之间的通信,对于多个 Root 之间的通信,需要通过具体硬件,实现 PCIe 交换开关的功能.本文的系统采用采用 CPU+双 GPU 的架构,使用 IDT 公司提供的 PCIe 交换芯片 PES32NT8AG2 实现多个 Root 端的互联,并设计 PCIe 交换芯片的驱动,完成了非透明桥的功能,实现了多个 Root 之间的高速通信^[2].

2 系统总体架构

系统架构如图 1 所示, PES32NT8AG2 为 IDT 公司提供的 8 个端口交换芯片(PCIe Switch),每个端口可以复用为上游端口(Upstream)和下游端口(Dowstream). Port0 接的是 Xilinx 公司提供的 ZC7045,这是一个可编程片上系统(All Programmable Soc). ZC7045 由 PS(Processing System)和 PL(Programmmable Logic)组成. PCIe Switch 的 Port2 和 Port4 接 GPU,GPU 为 Nvida 公司提供的 Tegra K1.

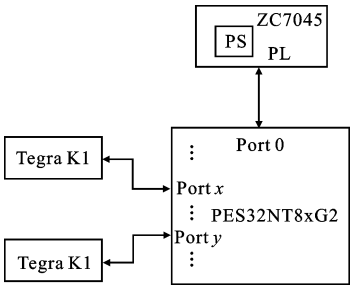


图 1 系统架构图

3 非透明桥

3.1 非透明桥的功能

在一个处理器系统中,处理器和 PCIe 设备可以通过 PCIe 桥形成总线拓扑结构. 32 位的 PCIe 总线能提供 4 GB 地址空间,对于多处理器系统,4 GB 的地址空间往往会全部被处理器的存储空间占用,而没有空间分配给 PCIe 设备,并且多个处理器在访问 PCIe 的地址空间时可能会发生地址冲突. PCI Express 非透明桥的出现,为多个处理器之间(或者异构平台多个 Root 之间)的高速通信提供了良好的解决方案.

两个处理器系统可以通过非透明 PCIe 桥

(NTB)进行连接. 如图 2 所示,非透明桥两端可以连接两条对等且独立的 PCIe 总线,两个处理器系统的 PCIe 总线域地址空间被隔离开来. 非透明桥解决了多个处理器在访问 PCIe 的地址空间时的地址域冲突问题,两个处理器通过非透明互相隔离,处理器之间进行 I/O 数据通信时,其数据传输地址通过非透明桥的地址翻译功能进行相互转换. 在通信过程中,处理器间的状态信息通过非透明桥特殊寄存器实现^[1].

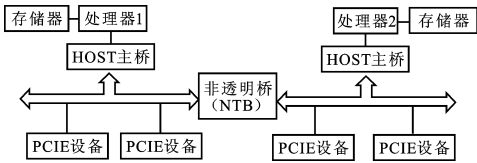


图 2 非透明桥连接处理器系统

3.2 PCIe 数据传输流程

ZC7045 的 PL 部分主要实现的是 HOST 主桥,其中包含了数据链路层和事务层的功能,其架构如图所示,架构体现两个半桥分别处理来自不同方向的事务请求^[3]. 即 Slave Bridge 封装来自应用端的 TLP,并接收该事务反馈回的完成(complete)报文. Master Bridge 解析来自设备端的 TLP,并可返回完成报文.

如图 3 所示, TLP 报文由 PL 端发出,通过 ID 路由,根据总线号、设备号、功能号^[4]找到 Port0;通过地址路由确定该 TLP 包要寻址的地址空间是否属于 Port0 的 Bar 所涵盖的范围,以决定接受还是拒收该包. TLP Header 中包含了 Bar Address 和 Requester ID,在 TLP 经过非透明桥的过程中,其 Request ID 和 Bar Address 均进行了转换. 通过地址翻译的方式将 TLP 携带的 Bar Address 翻译成对端 Bar 空间的地址.

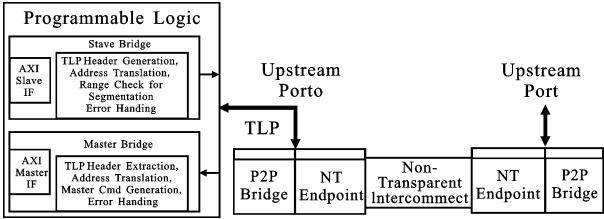


图 3 TLP 传输流程图

4 系统关键程序设计

软件层主要实现了数据通过 PCIe 链路的收发工作. 由于 Linux 操作系统良好的可裁剪、可移植

性, 因此软件层采用 Linux 系统, 其的核心是 PCIE 总线驱动和 PCIE Switch 驱动.

4.1 PCIE 总线驱动

PCIE 总线驱动是在 PCI 总线驱动的基础上, 加上 PCIE 平台驱动实现的. Linux 系统会把 PCIE 设备进行编号, 以 0000:00:00.0 为例, 前 32 位是 PCIE 设备的域名, 一般情况下为零; 接下来的 8 位表示 PCIE 设备的总线号, 系统最多支持 256 条总线, 因此总线号的范围为 0~255; 后面 8 位表示设备号, 在一条给定的总线上最多支持 32 个设备, 因此设备号的范围为 0~31; 最后 4 位表示功能号, PCIE 设备支持 8 种不同的功能, 因此功能号的范围为 0~7.

启动 Linux 操作系统, uboot 先扫描 PCIE 设备, 但是扫描的结果不能直接使用. 然后启动 Linux 内核, 加载 PCIE 平台驱动, 完成连接 PCIE 链路、设置中断节点、PCIE 主桥的挂接后, 然后 Linux 内核开始进行总线扫描. 扫描分两步, 扫描设备和分配资源^[5]. 扫描设备会从主桥开始, 按照深度优先方式扫描所有设备^[6]. 在扫描过程中如果遇到一个桥设备, 则系统扩展一个总线号, 并对这条总线进行扫描. 扫描获取的信息主要有 PCIE 总线拓扑信息和各个设备所需要的资源信息. 分配资源也从主桥开始, 每扫描完一个设备, 便按需为设备分配资源并更新配置信息. 桥根据其下设备所分配的资源更新配置范围, 以供 PCIE 总线路由. Linux PCIE 扫描流程如图 4 所示.

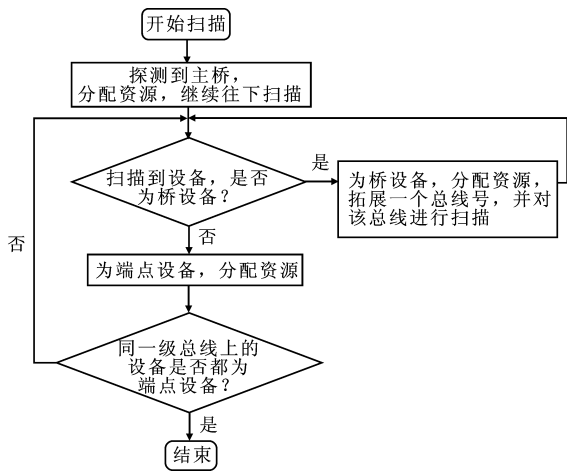


图 4 PCIE 扫描流程图

4.2 PCIE Switch 驱动设计

PCIE Switch 驱动包含了一系列的 Linux 内核模块, 主要实现了功能层(function layer)和基础层(base layer)两部分功能. 功能层所提供的是插入了

Linux TCP/IP 协议栈的以太网驱动, 能够给用户 提供通信的功能接口. 驱动的各个模块关系如图 5 所示.

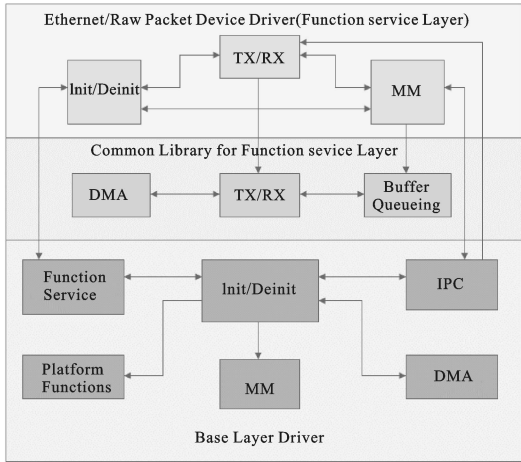


图 5 驱动架构图

Init/De-init 模块: Init/De-init 模块包括模块初始化(ntb_init)、模块析构(ntb_exit)、PCI 驱动相关例程、设备初始化(dev_init)、设备析构(dev_exit)五个部分. ntb_init 调用 pci_register_driver 函数来向系统注册驱动.

中断服务(ISR)程序: ISR 通过 dev_init 注册, 当远程系统发送门铃过来时, 便调用 ISR 程序. 输入门铃寄存器基于分区号来接收门铃. 输入门铃寄存器一共有四种门铃状态: 0 - Valid, 1 - Done, 2 - Function service, 3 - Data packet. 接收到 IPC_VALID 门铃时, 系统分配 IPC 结构, 用消息寄存器数据对结构进行填充. 获取消息的 ID 号. 接收到 IPC_DONE 门铃时, 检查是否本地的 IPC_DONE 消息计数器与远程的计数器相等, 如果不相等, 将本地的计数器自增.

Tx/Rx 模块: 这个子模块包括收/发功能的主要函数. 这些函数用来传输 Function service 下所需要的数据和接收 PostQueue 中的数据. 传输包: 在成功动态分配 ntb_frame 数据结构的内存后, 功能层调用 frame_send 来传输数据. 如果其返回为负值, 包会丢失, 内存被释放. 如果为正值, 功能层驱动需要通知应用层暂停数据传输, 设置一个定时器来传输现在的包. 否则, 传输完成. 当数据传输完成, tx_cb 函数被调用来向远程端点的 PostQueue 插入数据包, 并向端点发送门铃, 释放 ntb_frame 数据结构内存.

接收包: 当 IPC 状态是 STATE_CONN_EST, 并且本地端点接收到门铃, 这意味着远程端点已经

传输完一个包. 源分区的 tasklet 调用 frame_receive 来决定在本地 PostQueue 中是否有任何的未定的接收缓冲区. 如果有,调用 frame_rx 来设置接收包中的 ntb_frame 结构体. 功能层驱动需要请求一个目的缓冲并将其保存在 ntb_frame 结构体中,之后通过 DMA 或内存拷贝开始数据传输. 数据传输完成时,回调函数来释放内存和目标缓冲区.

MM(内存管理)模块:该子模块负责数据传输的本地缓冲区的初始化. 在初始化例程中,dev_init 调用 mem_init 函数来提取内存缓冲区.

IPC 事务:IPC 在端点与端点之间初始化,每个端点的基础层通过发送 IPC_CMD_START 命令来建立连接. 之后基础层发起 IPC_CMD_MAP,从而实现端点之间的地址交互. 完成之后,基础层告诉功能层映射完成,功能层通过发起 IPC_CMD_HELLO IPC 来与其他端点建立连接,数据交互便实现了.

IPC 状态机:在整个的配置流程中,每一个阶段的地址处理方式是不一样的,下面描述了连接和地址映射阶段的 IPC 状态机.

连接阶段:这个部分描述了系统之间的状态连接事务. 对于每一个分区,在数据传输连接完全建立之前,状态改变如图 6 所示. 当基础层的驱动加载完成,模块初始化完成后,驱动通过向 NTB 另一端点发送 IPC_CMD_START 命令来初始化 IPC 通道. 在收到 NTB 另一端的响应后,该端点等待另一端发送 IPC_CMD_START 命令. 如果两端都完成了握手,它们便准备地址映射.

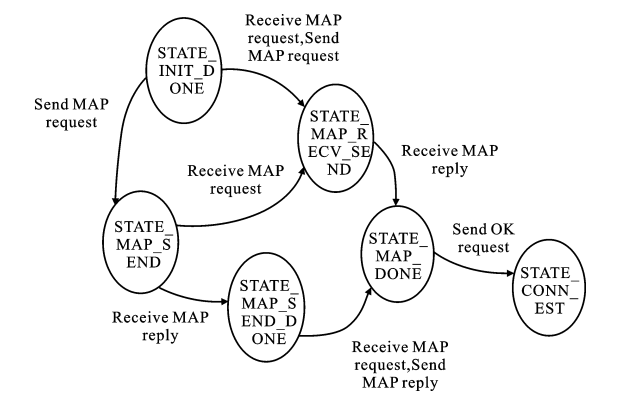


图 6 状态转换图

4.3 建立连接后 IPC 设置

这个部分描述了信息交互阶段的状态转移. 在 init Done 状态下,每个系统会通过发送 IPC_CMD_MAP 来询问远程系统的内存地址.

哪一个 NTB 端点先加载驱动是无所谓的,因

为它们在 STATE_INIT 状态下通过发送 IPC_CMD_START 命令实现同步. 在完成所有 IPC_CMD_MAP 交换之后,基础层驱动调用功能层中回调函数来通知所有功能层驱动. 一旦功能层驱动接收到消息,便开始 IPC HELLO 消息交互,实现了功能层中本地端点基地址的交互.

在接收到 IPC HELLO 消息反馈后,基础层向功能层发送 EVENT_REMOTE_READY. 远程队列结构被设置,基础层准备好向对端发送数据事务.

5 系统数据传输测试

数据传输测试是在 Linux 应用层实现的. 通过编写应用程序,调用 PCIE Switch 驱动完成数据传输. 应用程序所传递的参数有 Partion ID, System ID, Data Transfer Size, 将实参传递进去后,在完成规定包数(1 000 包)数据接收后通过数据统计信息与系统时间进行计算,得到当前总线传输速率,并显示结果.

实验完成了 3 个 Root 之间的数据传输测试. 例如,在 PS 端 Linux 系统中,将 Partion ID 设置为 1, System ID 设置为 0,就能完成 PS 与 GPU1 之间的数据传输. 同理,在 GPU1 端 Linux 系统中改变目标 ID 号,可以完成 GPU1 与另外两个 Root 的通信.

改变数据传输大小,进行多次测试,分别验证了系统在不同数据传输大小下,多个 Root 之间的传输速率. 通过多次测量,记录相关测试数据,通过 ORIGIN 软件对数据进行拟合,得到的统计数据关系曲线如图 7 所示.

可以看出,数据传输大小与总线传输速率开始呈现正比例关系,即每次数据传输规模越大,总线传输速率越大. 当数据传输规模大于 1 024 MB,总线传输速率增长的越来越慢. 在 PS 和 GPU 之间传输速率可以达到 107 MB/s, 两个 GPU 之间传输速率能达到 197 MB/s.

传输性能分析:PCIE 2x 的理论带宽可以达到 1.0 GB/s. 可以看出实验数据和理论带宽之间存在很大的差别. 原因分析如下:

(1)分区之间的传输性能有着较大的差异,从实验数据可以看出,分区 0 和其他两个分区间的传输速率要小于分区 1 和分区 2 之间的传输速率.

(2)通过 memcpy 传输数据的过程如下:本地 memory—>本地 Bar 空间—>远程 Bar 空间. 通过非透明桥需要更多的时间.

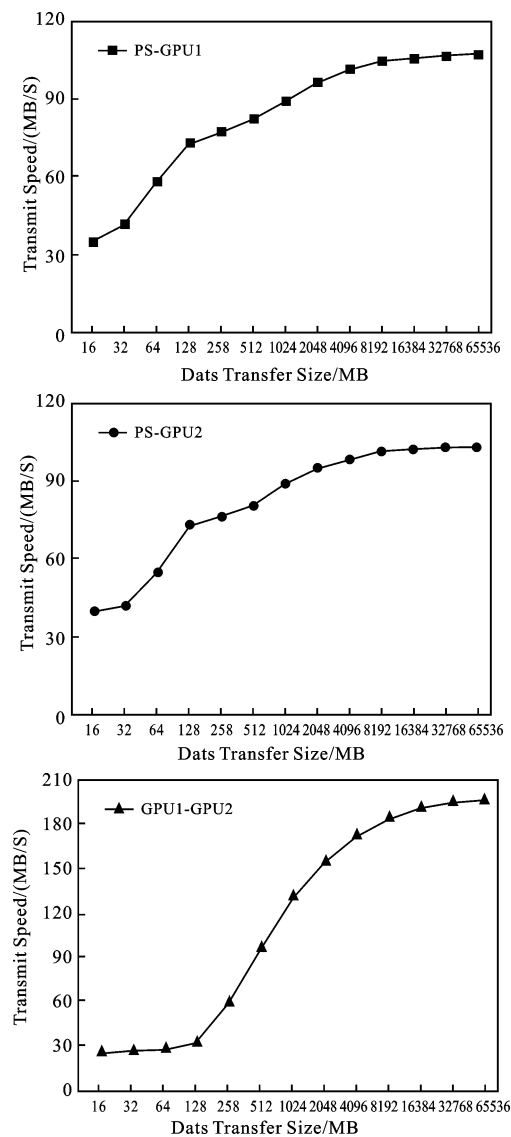


图 7 数据关系曲线

(3) 由于在两个 GPU 之间没有 DMA 通道, PS-GPU 的 DMA 模式并没有应用于实际的工程中, 在 DMA 模式下带宽预计可以达到 700~800 MB/s.

6 结束语

本文在搭建 CPU+双 GPU 的异构平台的基础上, 主要完成了嵌入式异构平台上的软件设计, 实现了多个 Root 通过非透明 PCIE 桥进行高速通信. PCIE 非透明桥防止了系统中多处理器之间的访问冲突, 并且具有高可靠性和高数据传输速率, 使其将成为 PCIE 系统结构中不可或缺的一环. 基于本文的异构平台, 可以拓展成更加复杂的 PCIE 系统拓扑结构, 对高速分布式计算奠定了良好的基础.

参考文献:

[1] 王齐. PCI Express 体系结构导读[M]. 北京: 机械工业出版社, 2010: 965-972.

[2] Jack Regula. Using non-transparent bridging in PCI express systems [M] USA: PLX Technology, 2004: 56-58.

[3] IDT 89HPES32NT8xG2 PCI Express Switch User Manual [EB/OL]. (2013-01-15). [2017-03-20]. www.idt.com; Integrated Device Technology.

[4] Ho K J, Chen M H, Chu H M. Method for configuring a peripheral component interconnect express (PCIE) [J]. US Patent 7,506,087, 2009: 12-15.

[5] 李才华. PCI-Express 非透明桥在智能系统中的应用设计[J]. 电子元件应用, 2009, 27(4): 62-65.

[6] 杨庆宇. Linux 操作系统下的 PCI 驱动开发[J]. 电子元件应用, 2010, 12(6): 121-128.

作者简介:

徐 健 男, (1992-), 硕士研究生. 研究方向为航天综合电子系统. E-mail: xujian14@mails.ucas.ac.cn.

张建泉 男, (1981-), 高级工程师. 研究方向为航天综合电子系统.

张 健 男, (1971-), 研究员. 研究方向为航天综合电子系统.