文章编号:1671-4598(2023)08-0273-07 DOI:10.16526/j.cnki.11—4762/tp.2023.08.041 中图分类号:TP752 文献标识码:A

基于 FPGA+GPU 的图像采集处理系统设计

蒋俊伦，丰大强，徐新瑞，程 坤，常中坤，王 桢
（山东航天电子技术研究，山东 烟台 255000）

摘要：随着嵌入式图像处理系统的快速发展，对于前端图像采集模块的需求越来越高；图像采集的速度、分辨率、可靠性以及集成度对后续设计的准确度由极大的影响；通过对数字图像采集系统进行研究，设计出了基于 FPGA 和 GPU 架构的图像采集处理系统，重点研究了图像采集处理系统的硬件设计过程和软件设计过程；在基于 FPGA+GPU 的图像采集处理系统中，让具有强大运算处理能力的 GPU 专注于数据存储、用户交互以及后续的图像处理；系统中，FPGA 则负责图像的采集、外设控制、任务调度；GPU 与 FPGA 之间通过高速 PCIE 总线进行通信，分别设计编写基于 Linux 系统的驱动程序和 FPGA 端 PCIE 程序；实验结果表明，所设计基于 FPGA+GPU 的图像采集处理系统可实现 437.5 Mbps 的实时图像采集存储速度，传输过程实时稳定，数据传输完整。

关键词：FPGA；GPU；图像采集；数据存储；任务调度

Design of Image Acquisition and Processing System Based on FPGA and GPU

JIANG Junlun, FENG Daqiang, XU Xinrui, CHENG Kun, CHANG Zhongkun, WANG Zhen
(Shandong Institute of Space Electronic Technology, Yantai 255000, China)

Abstract: With the rapid development of embedded image processing systems, the demand for front-end image acquisition modules is increasing. The speed, resolution, reliability, and integration of image acquisition greatly affect the accuracy of subsequent image processing. By studying the digital image acquisition system, an image acquisition and processing system based on FPGA and GPU architecture is designed, with focusing on the hardware and software design process of the image acquisition and processing system. In the image acquisition and processing system based on FPGA and GPU, the GPU with powerful computing and processing capabilities is focused on the data storage, user interaction, and subsequent image processing. In the system, FPGA is responsible for the image acquisition, peripheral control, and task scheduling. The communication between GPU and FPGA is carried out through the high-speed PCIE bus, and the driver programs based on Linux system and FPGA PCIE programs are designed separately. The experimental results show that the designed image acquisition and processing system based on FPGA and GPU can achieve a real-time image acquisition and storage speed of 437.5 Mbps. The transmission process is real-time and stable, and the transmitted data is complete.

Keywords: FPGA; GPU; image data acquisition

0 引言

近年来，地面扫描观测设备得到快速发展，扫描观测需要实时采集存储数字图像，对图像采集技术的要求也越来越高。图像采集模块采集到的图像的质量对后续设计的准确性有极大的影响。作为图像信号处理系统的重要组成部分，图像采集模块的速度、分辨率、可靠性以及集成性等方面受到了越来越多的关注[1]。目前，图像采集系统大多采用电脑端使用的图像采集卡，这种图像采集卡采集速度快、可以保证较高的采集精度，但体积大、功耗高，而且只能在电脑端使用，无法在便携式产品中使用。从适应性角度考虑要求图像采集系统性价比高，环境的适应能力要强，并且要易安装和使用。嵌入式图像采集处理系统为图像采集提供了新的实现途径，但单纯应用 ARM 或 DSP 等嵌入式平台由于其存储容量有限、处理速度低、扩展性较差的特点，不能满足当前图像采集系统的带宽需求[2]。

图像采集前端模块是整个图像采集系统的核心部件，会直接影响到整个系统的工作执行效率。考虑到目前扫描观测设备的图像数据分辨率高、帧频快，如何使用恰当的方法采集、传输以及存储高质量的图像是一个非常重要的问题。图像采集的整体过程相对比较简单，但是图像采集时，图像的数据量非常的大，系统对实时性要求非常高[3]，图像采集的实时性是仅靠软件是无法实现的，开发人员要考虑使用软件与高速硬件系统相结合的方式来完成对图像的高速实时采集[4]。

文献［5］中，采用 MSP430 单片机作为主控芯片，通过 USB 数据采集技术实现对监控现场视频数据的采集和传输，但该方案最终达到的数据传输速率仅为 8～10 Mbps。对于目前相机来说，常规的相机参数可达到 640×512 像素，单像素 12 位采样，帧频 100 Hz，数据量可达到 375 Mbps。该系统难以满足实际应用需求。

文献［6］中，提出了一种基于现场可编程逻辑门阵列（FPGA，field programmable gate array）的高速图像采集系

收稿日期:2023-03-01； **修回日期**:2023-04-06。

作者简介:蒋俊伦(1992-)，男，山东荷泽人，硕士，工程师，主要从事图像处理以及电子信息处理方向的研究。

引用格式:蒋俊伦，丰大强，徐新瑞，等. 基于 FPGA+GPU 的图像采集处理系统设计[J]. 计算机测量与控制，2023，31(8)：273-279，305.

统硬件方案。采用 FPGA 作为图像采集主控芯片，大大简化了系统硬件结构，并提高了系统可靠性，同时可以应用于高速图像采集中。但 FPGA 操作起来流程复杂，应用门槛较高。

同时为了保证采集到的图像的完整性和连续性，不仅要对图像数据进行采集，而且要对原始数据进行缓存。一般的图像采集模块主要包括存储器单元、CCD 或者 CMOS 相机接口以及总线接口等。系统中设计了一个图像采集处理系统，采用 Xilinx 公司的 XC7K410T FPGA 进行相机图像采集以及与其他设备如转台之间的信息交互。采用英伟达公司的 Jetson TX2 GPU 作为图像数据存储处理的平台。XC7K410T 通过 LVDS 接口采集相机数据，采集到数据后通过 PCIE 总线将图像数据传输给 TX2，TX2 再将数据存储到固态硬盘中，以进行后续的处理。

1 系统结构及原理

对相机图像进行采集存储需要一个可靠性好、携带便捷以及实时性高的完整系统。该系统由电源模块、外设模块、FPGA、GPU、固态硬盘组成。其中：

电源模块为系统中 FPGA、GPU、外围电路器件提供供电，典型电压包括：5 V、3.3 V、1.8 V、1.2 V 和 1.0 V。外设模块实现与相机等外部设备的通信，包括 RS422 接口、LVDS 接口。GPU 模块使用 TX2 模组，TX2 使用 Linux 操作系统作为底层系统，可以充当上位机的功能。与此同时 TX2 实现通过 PCIE 从 FPGA 接收传送的相机数据，将接收图像数据通过 SATA 接口存储到硬盘中，并执行相应的图像处理算法，将处理出的结果进行转发。FPGA 模块通过 16 路 LVDS 接收到图像数据并通过 PCIE 发送到 TX2 模块中。系统的原理如图 1 所示。

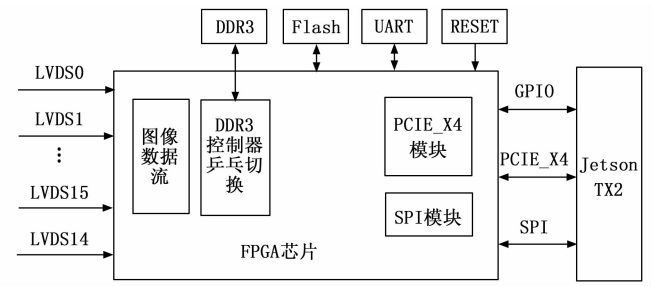

图 1 图像采集系统原理图

2 系统硬件设计

2.1 Jetson TX2 模块

图形处理器（GPU，graphic processing unit）是一种专门进行图形处理的硬件设备，经过近年来的快速发展，GPU 已经从最初的图形处理领域扩展到机器学习、深度学习、三维重建等并行计算领域，其架构从专用于图像处理的固定架构发展到了用于通用计算的并行计算架构[7-8]。相比于 CPU，GPU 在并行计算、分布计算和浮点运算方面的运算能力可以高出数十倍乃至上百倍。在嵌入式硬件方面，英伟达公司在 2013 年推出 Jeston TX2 模组，TX2 是一款高性能、低功耗的 AI 单模块超级计算机，采用 GPU＋CPU 架构设计，其 GPU 采用新一代 Pascal 架构设计，具有 256 个 CUDA 核；CPU 使用集成了四核的 ARM® Cortex™-A57 处理器（PS-processing system）。外围电路方面该模组还包括 10/100/1000BASE-T 以太网络接口，同时配备了 8 GB 的 DDR4 内存和 32 GB eMMC5.1 Flash 存储空间。另外具备 HDMI、CSI、SATA 等外部接口，其浮点计算能力可以达到 1.5 Tflops。因此选用 TX2 作为图像采集处理的核心模块十分方便快捷，其强大的数据处理能力可以将图像数据快速的写入到固态硬盘中，另外 TX2 还可以通过图形化界面与用户交互，容易操作。

在本系统中 TX2 的作用包括两部分：一是利用 SPI 模块发送消息通知 FPGA 开始进行图像数据采集；二是通过 PCIE 模块将图像数据从 FPGA 接收数据，并将数据通过 SATA 接口写入到固态硬盘中。

2.2 FPGA 模块

FPGA 是一种可以通过编程来改变内部结构的“特殊芯片”，自 Xilinx 公司于 1984 年研发了世界上的首款型号为 XC2064 的 FPGA 以来，FPGA 的发展经历了多个阶段[9]。目前最新的 FPGA 逻辑门的数量已经达到亿级以上，工艺尺寸也达到了 16 nm 左右，无论是功耗还是性能，相比于 XC2064 都得到了极大的提升[10]。与普通的集成电路芯片相比，FPGA 具有许多十分突出的特点，FPGA 的可编程特性使它不仅可以用于 ASIC（专用集成电路）的设计，同时可以用来完成 ASIC 芯片的原型验证。FPGA 常采用高速的 CMOS 工艺进行设计，具有低功耗的优势，而且其内部包含丰富的触发器和可编程 I/O 资源，可以用于实现较大规模的集成电路设计。FPGA 在体系结构上有很大的并行度，这使其十分擅长并行数学计算[11]。

FPGA 模块采用 Xilinx 的 XC7K410T 芯片，通过 16 路 LVDS 接口接收相机数据，其中 14 位数据，1 位 GATE，1 位选通。1 路 RS422 作为相机控制接口，1 路 RS422 作为遥测遥控接口，一路 RS232 作为转台控制接口。数据采集转发 FPGA 具体型号为 XC7K410T-2FFG900I，该芯片的主要性能如下[12]：

具有 48 万逻辑单元；

具有 37 Mbit 内部 RAM；

具有 2 800 个 DSP48E 模块；

具有 32 个 GTX 接口模块；

具有 14 个 CMT 模块；

具有 700 个 HPIO 管脚。

由于该 FPGA 是基于 SRAM 结构的 FPGA，具有设备断电后数据丢失的缺点。需要为 XC7K410T-2FFG900I 专门配备一个程序存储器用来存储配置数据，在设备上电初始化时完成配置数据的装载。K7 系列的 FPGA 支持×1，×2，×4 三种 SPI 加载方式，Flash 选择 S29GL512P11TFI010，该芯片＋3.3 V 供电，容量 512 Mb。

由于相机输出 14 位宽的数据，帧频 100 Hz，相机像素

分辨率 640×512，由此将产生 14 bit×100 MHz×640＊512（约为 437.5 Mbps）的数据量，考虑到 TX2 使用的是 Linux 系统，Linux 是一种非实时响应的操作系统，这么快速的数据量即所谓的迸发数据流没有办法直接通过 DMA 传送给 TX2，需要将图像数据临时存储在内存条（DDR3 SDRAM）中。DDR3 存储器是一种采用时钟双沿工作的高速存储器，是处理器常用的片外存储器。与静态随机访问存储器（SRAM，static random access memory）所采用的 CMOS 工艺不同，DDR3 存储器采用动态随机访问存储器（DRAM，dynamic random access memory）动态电路工艺，多采用电容储值，读写之前必须先对数据线进行预充电；读是破坏性的，读后必须写回；漏电流的存在使得 DDR3 必须保持定期刷新（读出放大后再写回）[13]。

FPGA 通过对 DDR3 SDRAM 的访问存储器控制，实现大容量迸发数据的读、写、刷新等时序操作。当 FPGA 收到图像数据后将数据缓存到 DDR3 内，当收满一副图像后 FPGA 将数据放到 DMA 内通过 PCIE 总线发送给 TX2。

2.3　电源模块

电源模块为整个系统供电，采用＋28 V 电源作为输入电压，将直流＋28 V 供电转化成设备各组件工作所需的＋12 V、＋5 V 电源等，其中 TX2 模块供电模块与 FPGA 供电模块分别采用不同的电源模块，实现 TX2 模块与 FPGA 模块的相互独立。

＋12 V 为 TX2 模块供电，＋5 V 为 FPGA 模块供电。直流 28 V 输入电源后，首先经过滤波后供给 DC/DC 转换模块，转换成设备所需的工作电压，滤波后输出给各用电设备组件。供电模块原理如图 2 所示。

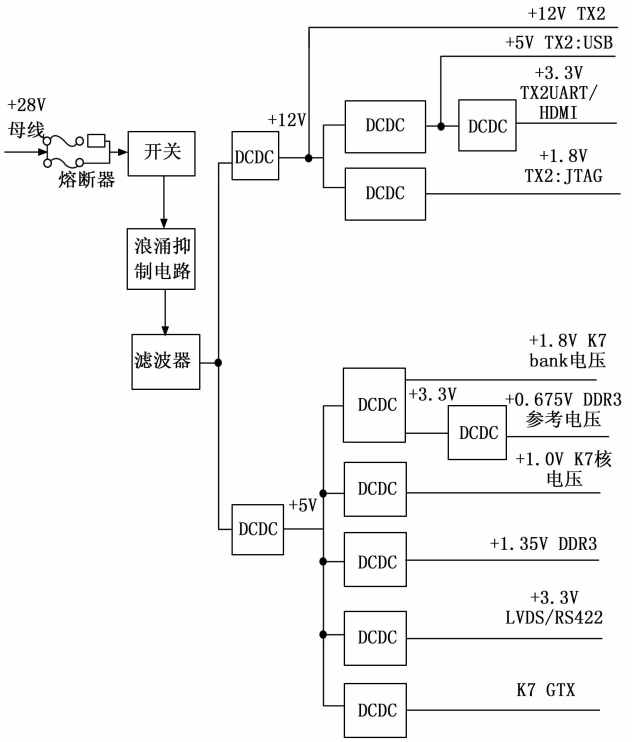

图 2　供电模块原理框图

2.4　外设模块

本系统中外设模块主要包括低电压差分信号（lowvoltage differential signaling，LVDS）接口、RS422 接口，LVDS 共 16 路，其中 14 路为数据、1 路为选通、1 路为时钟，时钟速率为 31.25 Mbps。LVDS 接口用于保证图像采集和传输，可以将图像高速数据实时的进行传输存储。RS422 接口用于相机控制、转台控制等。

3　系统软件设计

基于 FPGA＋GPU 的图像采集处理系统软件程序主要包括 FPGA 程序设计、TX2 驱动程序设计和 TX2 应用程序设计。FPGA 开发基于 Xilinx 公司的 Vivado 集成开发仿真环境。TX2 本身自带 Linux 操作系统，可以在 Linux 上安装 VScode 软件，使用 VScode 对 TX2 驱动程序和应用程序进行编写编译。

本系统中 TX2 是系统软件的运行主体，当系统启动后，操作人员通过 TX2 上位机界面，发送图像采集指令给 FPGA，FPGA 接收到指令后对指令进行解析处理，读取图像数据，并将数据发送给 TX2。系统软件的程序流程如图 3 所示。

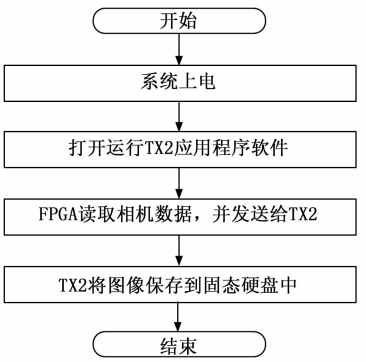

图 3　系统流程图

3.1　软件设计思路和编程方法

3.1.1　FPGA 程序设计

FPGA 内部逻辑结构如图 4 所示，包括 PCIE 模块，PCIE 协议处理相关模块，DDR3 缓存模块和 SPI 及串口模块。系统上电后进入复位状态，复位完成后接收 TX2 通过 PCIE 发送的指令进入相应的工作模式，包括空闲模式和采集模式及系统复位模式。TX2 通过 PCIE 接口配置相关的寄存器来实现对设备工作模式的修改，实现 TX2 对相关外设的控制。

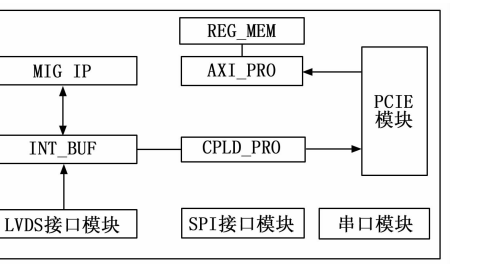

图 4　FPGA 逻辑框图

1）DDR3 控制器：DDR3 控制器使用 Xilinx 官方 IP 核，该 IP 核为免费 IP，其内部包含了基本的时钟和复位处理模块、物理层接口处理模块、延时控制单元、读写控制器、数据缓存单元及用户接口等模块。其中时钟和复位处理模块通过内部的 PLL 生成 IP 核所需的时钟网络和相应的复位信号。物理层接口模块负责完成和物理芯片的交互，对 DDR3 物理接口的高速 DDR 时序进行处理，通过延时控制单元完成数据相位的延时和相位对齐等操作。此外 IP 核内部还利用 FPGA 内部的缓存器实现对物理层数据的跨时钟域缓存，以实现数据位宽的转换以及物理层时钟向用户层时钟的跨时钟域设计。用户接口模块负责对用户的指令进行译码，缓存用户数据以及对用户的数据实现流控。设计中采用的 DDR3 硬件数据位宽为 16 位，容量为 2 Gbit，理论读写速率可达 50 Gbps，考虑在实际控制时的带宽损耗，即使是在带宽效率为 50%时带宽也可达到 25 Gbps，而实际应用时相机的速率为 2.5 Gbps，满足传输速率要求。

设计中的 INT _ BUF 模块负责对 DDR3 用户数据的打包和缓存，实现时采用片上 FIFO 进行缓存及实现跨时钟域的处理，根据设计指标，片上 FIFO 的读写位宽为 128 位，深度为一帧图像中的三行数据。

系统上电后，进入复位转台，复位完成后状态机为 IDLE 状态，在该状态下，等待上位机发送的读写指令。在空闲状态下，图像数据虽然实时采集，但是不进入 DDR3 缓存，当收到采集指令时通过 LVDS 协议区分数据格式的开头和结尾，对数据进行读取和传输。

FPGA 与 TX2 之间大容量数据传输设计原理如下：当图像数据采集开始，FPGA 将接收到的图像数据写入 DDR3 中，并开始将 DDR3 中的数据转存到 DMA 空间中，等待 TX2 发起读数据请求。由于 DDR3 和 FPGA 的速度极高，而 TX2 是非实时操作系统，所以在图像数据的转存系统设计过程中，要着重考虑两者之间数据传输的同步问题。在进行图像数据传输过程中，FPGA 应时刻检测当前周期内从 DMA 中读出的数据是否已经被 TX2 接收完毕，若未完成接收，则不能继续将数据从 DDR3 写入到 DMA 中，直到 TX2 完成数据接收存储，才进行 DDR3 下一周期的数据读取。

2）PCIE 模块：PCIE 总线从 PCI 总线演化而来，PCI 总线使用层次式体系结构，通过 PCI 桥将父总线和子总线连接起来。PCIE 总线是一种全新的串行总线技术，可以实现全双工点对点的传输。它彻底变革了 PCI 总线的并行技术，克服了 PCI 总线在系统带宽、传输速度等方面的固有缺陷[14]。PCIE 总线的发展也是高速 IO 总线的过程，PCIE 总线相比 PCI 总线具有以下特点：PCIE 总线在数据传输模式上，采用双通道全双工串行传输。一条 PCIE 通道包括一条发送通道，一条接收通道，速率可达到 2.5 GB/s。同时针对不同通信带宽的设备，PCIE 可根据具体需求配置成×1、×2、×4、×8 的并行通道。在软件方面 PCIE 总线完全兼容 PCI 总线，驱动程序可直接一直到 PCIE 系统中。

目前 PCIE 总线凭着传输速率快、质量高等特点，逐渐取代 PCI 总线占据高速数据总线的主导地位，在数据采集系统传输中发挥了极大的作用。

PCIE 总线层次结构如图 5 所示，从图中可以看到数据在 PCIE 总线接口传输过程。

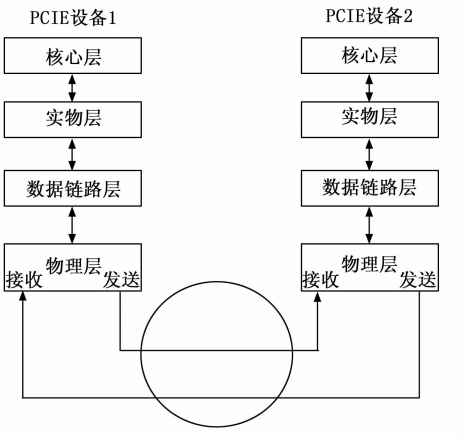

图 5 PCIE 总线的层次结构图

FPGA 代码设计过程中，常用的代码设计技巧包括以下几种：乒乓操作、串行信号并行化设计、流水线操作设计。合理的使用上述技巧可以提高代码的可行性，节省 FPGA 资源，保证系统的鲁棒性，此外还可以提高代码的执行效率以及稳定定。

乒乓操作是一种用来对高速数据流进行处理的设计思想，它的基本思想就是一种以面积换取速度，从某种层面来看，乒乓操作可以看成一种比较特殊的串并转换方式。乒乓操作实现的关键部分在于要保证不同通路间的数据处理是互斥的，使得输入的数据流能够按照一定的节拍，相互配合进行切换，从而实现数据的无缝缓冲和处理[15]。

本系统中 FPGA 通过 GTX 端口与 TX2 的 PCIE 接口进行连接，GTX 单个端口速率最高可达 12.5 Gbps[16]。PCIE 模块采用 Xilinx 的 IP 核来实现 PCIE 物理层和传输层等相关的 PCIE 协议。根据协议，PCIE 的节点分为 Leagcy、Endpoint 以及 Root complex。根据设计，在 FPGA 中实现的应该为 Endpoint 类型，所以设置 IP 核工作在 Endpoint 模式[2]。上电后 TX2 要完成对 EP 端的遍历和配置，根据协议规范，在 EP 模式下，配置空间为 type0 模式，新的 PCIE 规定配置空间的大小为 4 KB，前 256 字节保持与以往 PCIE 协议兼容。

上电后 TX2 完成对作为 EP 端的 FPGA PCIE 外设的内存映射，为该节点分配相应大小的内存。配置 IP 核的所需空间为 128 MB，该信息被记录在 BAR0 寄存器中，上电后 TX2 遍历找到该 EP 后读取该 EP 内的 BAR0 寄存器，译码后获得所需的内存的 128 MB，然后 TX2 为其分配相应的存储空间。设计中分配对应的起始地址开始的 128 个连续地址为控制寄存器区，TX2 通过向该地址区域写入数据来配置寄存器的值，更改相关的工作模式。数据传输时采用 DMA 传输机制配合乒乓操作进行图像数据传输，首先在

PCIE 空间开辟两块内存，分别是内存块 1 和内存块 2，当第一帧图像传输时，TX2 配置对应地址寄存器，告知 FPGA 本次所需传输数据的大小以及要写入的内存块 1 起始地址后，FPGA 自动进行数据打包和传输；当第二帧图像传输时，TX2 配置对应地址寄存器，告知 FPGA 本次所需传输数据的大小以及要写入的内存块 2 起始地址后，FPGA 自动进行数据打包和传输。将传输完成后进入等待模式。

3）SPI 模块：设备预留有 SPI 接口，通过该接口对外设进行控制，SPI 接口的时序如图 6 所示。

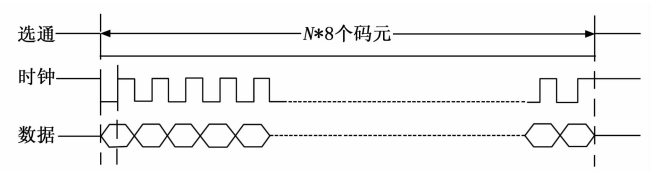

图 6　SPI 时序图

SPI 接口的码速率为 1 Mbps，时钟上升沿对准数据中间，数据高位在前低位在后，数据长度为 16 bit。

4）LVDS 模块：LVDS 是一种低摆幅的差分信号技术，采用一对差分线实现高速信号传输，使信号能在差分 PCB 线对或平衡电缆上以几百 MB/s 的速率传输[17]。设计中采用了低至 400 mV 的电压摆幅和大约 3.5 mA 的低电流驱动输出，加之差分对线的共模抑制功能，LVDS 具有低噪声和低功耗的优点[18]。

LVDS 数据发送和接收均采用三线制同步接口，包括三种信号：门控、时钟和数据，时钟频率可设置，且时钟连续不间断。发送端门控和数据信号的边沿均与时钟信号的上升沿对齐，接收方在时钟的下降沿读取数据。时序图和时间关系如图 7 所示，时钟上升沿与数据之间的延时为 4 ns。接口特点为：

1）门控信号低电平有效，一个有效门控内的数据定义为一帧；

2）门控信号下降沿到来时开始传送数据；高位先输出、低位后输出；

3）时钟占空比：45%～55%。

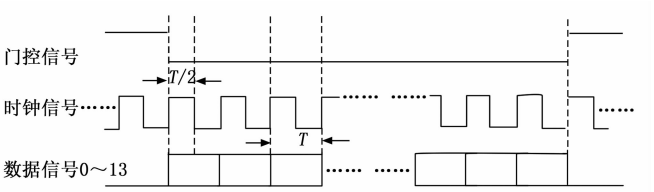

图 7　LVDS 时序图

3.1.2　TX2 驱动程序设计

TX2 自带的 Linux 系统中已经编写了 SPI 驱动和 UART 驱动，因此只需对 PCIE 驱动程序进行设计。

PCIE 驱动程序需要对 PCIE 配置空间进行读写操作。驱动程序通过 PCIE 的配置空间与硬件进行通信，通信的过程中一般需要读取设备 ID、厂商 ID 以及 PCI 类设备三个寄存器，完成匹配后再对基地址寄存器 0（BAR0）进行操作。PCIE 配置空间可以将 PCI 配置空间完全兼容，PCI 配置空间如图 8 所示[19]，包含由 16 个双字组成的 PCI 头标区，和由 18 个双字组成的 PCI 设备寄存器。PCIE 空间在 PCI 空间的基础上增加了 960 双字的配置空间，并在 PCI 的基础上增加了 PCIE 的高级错误报告、虚通道、电源预算以及设备序列号等功能[20]。

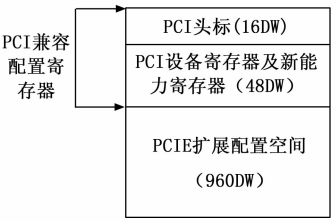

图 8　PCIE 配置空间示意图

系统中 TX2 操作系统使用 Linux 操作系统，首先在内存中开辟两块 DMA 缓冲区用来完成底层硬件内存与 TX2 应用程序之间数据的传输，DMA 缓冲区在 PCIE 底层硬件 TX2 应用程序之间相当于桥梁。FPGA 将数据发送到 TX2 的 PCIE 接口，PCIE 驱动程序将数据存放到 DMA 缓冲区，通过 MSI（message signaled interrupt）中断机制通知 TX2 应用程序对数据进行操作。PCIE 驱动程序的总体流程如图 9 所示，系统硬件启动后，TX2 在加载驱动程序时，首先会在驱动程序入口对 PCIE 硬件检测，检测到 PCIE 设备后，对驱动程序进行初始化，主要包括以下步骤：使能 PCIE 设备、使能 PCIE 总线主设备、读取 EP 设备号、申请 IO 内存区域、虚拟内存映射、使能 MSI 中断、申请 DMA 缓冲区以及注册字符设备等[4]。在上述步骤完成后，还需要对中断进行处理，以及在 IO 操作中实现对 DMA 缓冲区的读写。完成数据传输后，需要应在卸载驱动前将申请的内存空间进行释放。

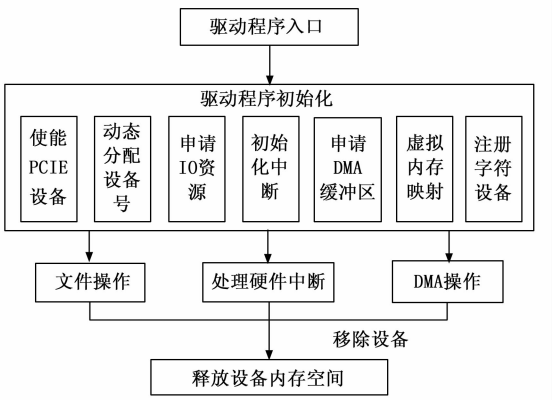

图 9　PCIE 驱动程序流程图

PCIE 驱动程序的具体操作流程如下：

1）打开以及关闭 PCI 设备。驱动程序中首先要做的是使能 PCI 设备，只有完成该步骤后，驱动程序才能够访问设备 IO 内存和中断。调用 pci _ enable _ device（）函数，

该函数会将 PCI 配置空间中命令区域中最低两位置为 1。当 PCIE 驱动程序完成任务后，会调用关闭 PCI 设备函数 pci _ disable _ device ()。该函数会将 PCI 配置空间中命令区域中最低两位置为 0。

2）申请 PCIE IO 资源和内存资源，并在程序使用完成后进行资源释放。

首先获取 IO 资源的基地址以及长度。

```
sc -> bar0_addr = pci_resource_start(pdev, 0); //获取 PCIE BAR0 空间基地址
sc -> bar0_len = pci_resource_len(pdev, 0); //获取 PCIE BAR0 内存大小
```

将驱动程序中申请的 IO 资源通过 ioremap 函数将 IO 资源映射到虚拟地址。

```
error = pci_request_regions(pdev, sc -> name); //申请 IO 内存区域
sc->bar0 = ioremap(sc ->bar0_addr, sc -> bar0_len); //将 BAR0 空间的 IO 内存映射为内核可用的虚拟内存
pci_release_regions(pdev); //程序最后释放 IO 资源
iounmap(sc->bar0); //取消 IO 资源映射
```

3）将 TX2 端设置为总线主设备模式。将 PCI 命令寄存器中的总线主设备位设置为 1，在该模式下，设备可实时获得总线占有权，处于 DMA 模式下。

```
pci_set_master(pdev); //设置 PCIE 总线主设备模式
```

4）使能、请求 MSI 中断以及禁止、释放 MSI 中断[7]。PCIE 中通 MSI 包进行中断响应。调用 pci _ enable _ msi (dev) 函数使能中断。然后调用 request _ irq (dev -> irq, intrpt _ handler, IRQF _ SHARED, sc -> name, sc) 函数绑定中断响应函数。

当驱动程序释放时关闭 MSI 中断，并将中断释放。

```
pci_disable_msi(pdev); //关闭 MSI 中断
free_irq(pdev ->irq, NULL); //释放中断
```

5）申请与释放 DMA 缓冲区[6]。调用 Linux 系统中的内核函数 pci _ alloc _ consistent 申请 DMA 缓冲区，通过这种方式获得的 DMA 缓冲区可以保证数据的一致性。主设备申请 DMA 缓冲区后，主设备和从设备均可以访问 DMA 缓冲区。

```
nzy_read_vir = pci_alloc_consistent(pdev, BUF_SIZE, &(nzy_read_phy)); //分配 DMA 缓冲区
pci_free_consistent(pdev, BUF_SIZE, sc -> dma_read_vir, sc -> dma_read_phy); //释放 DMA 缓冲区
```

6）文件操作：完成上述 PCIE 驱动相关工作后，通过 file _ operation 结构体设置字符设备驱动程序设计的主体内容，包括 open ()、write ()、read ()、close () 等函数。open () 函数用于打开字符设备；write () /read () 函数实现对字符设备读/写操作；ioctl () 控制字符设备函数，对 DMA 寄存器读写操作；release () 函数用于关闭设备。

```
static const struct file_operations fpga_fops = {
.owner = THIS_MODULE,
.unlocked_ioctl = fpga_ioctl,
.fasync = fpga_fasync, //向应用提供驱动异步通知响应
.mmap = my_mmap //将物理地址映射到应用空间虚拟地址
};
```

7）DMA 操作：本系统中传输的图像数据量大、传输速度快，为了实现数据的完整性，采用 DMA 的方式对图像数据进行传输。DMA 传输的过程中，处理器可以进行其他操作，提高了系统的性能。此外 DMA 是直接内存读写操作，相比于处理器读写速度更快。

DMA 读操作流程如图 10 所示。DMA 传输时需要将 DMA 块的起始地址、要传输的数据块长度以及数据传输启动指令发送给 FPGA 端。FPGA 收到启动指令后，将相机数据存放到 DMA 缓冲区内，当传输完成后，FPGA 发送 MSI 中断信息，TX2 收到中断信号后进去到驱动中的中断函数，中断函数中 SIGIO 信号触发应用程序中断，进而应用程序对数据进行存储。

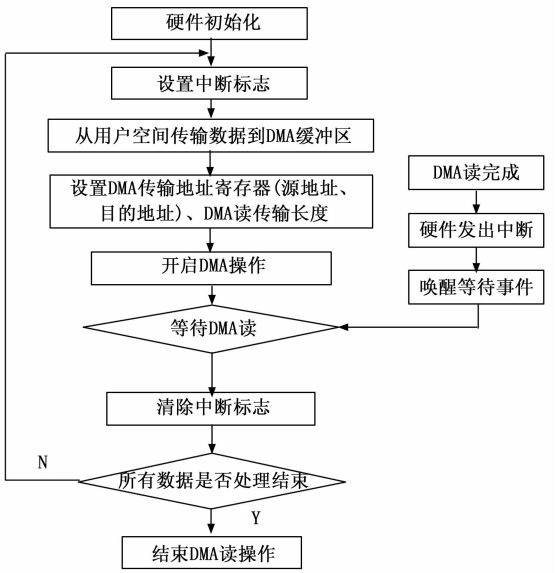

图 10 DMA 操作流程图

部分代码如下：

```
write_reg(fpgas[io.id], DMA_READ_LOW, ((fpgas[io.id] -> dma_read_phy) + BUF_SIZE/2)); //将地址写入 DMA 写的低 32 位地址
write_reg(fpgas[io.id], DMA_READ_HIGH, 0); //DMA 写的高 32 位不使用
write_reg(fpgas[io.id], DMA_READ_SIZE, io.len); //DMA 要写的大小写到寄存器中
write_reg(fpgas[io.id], DMA_READ_ENABLE, 1);
static irqreturn_t intrpt_handler(int irq, void *dev_id) //中断处理函数
{
  kill_fasync(&dma_async, SIGIO, POLL_IN); //read mem data
return IRQ_HANDLED;
}
```

8）处理硬件中断：在中断函数中，通过 kill _ fasync 函数将中断信号发送给应用程序。

```
kill_fasync(&dma_async, SIGIO, POLL_IN);
```

3.1.3　TX2 应用程序设计

1）主程序：主程序中通过 open（）、read（）、write（）等函数接口去调用内核空间的相应驱动函数对设备进行操作。设计中要求使用 PCIE 进行大数据传输，在应用程序中需要指定要读写的数据量 RW _ TIME。以接收为例，首先打开要写入的文件，判断当前写次数的奇数次还是偶数次，分别不同的内存块中将数据写入文件。应用程序部分代码如下：

```
FILE* fp = fopen(tSaveFileInfo.LocalFile, "a+"); //打开存盘数据文件
fpga = (fpga_t *)malloc(sizeof(fpga_t)); //分配内存
fpga->fd = open("/dev/"DEVICE_NAME, O_RDWR | O_SYNC); //打开设备文件
ioctl(fpga -> fd, IOCTL_FILL_COPY, 0xBB); //启动 DMA 数据传输
fseek(fp, 0, SEEK_END);close(fd); //关闭设备
fclose(fp);//关闭存盘文件
```

在本系统中，应用程序与驱动程序之间的通信主要包括以下几个方面：

（1）在应用程序中使用 open（）函数打开设备文件，对应到底层驱动是打开 PCIE 设备。

（2）通过 ioctl（）函数调用驱动层的 ioctl（）函数实现与 PCIE 的数据读写。

（3）通过 close（）函数关闭设备文件。

2）中断程序：在应用程序中，使用 Signal 函数将驱动层中断信号传输到应用层。

```
signal(SIGIO, vSigHandleFunc);
```

在中断处理函数中首先判断当前接收中断的次数，如果是奇数次则从 DMA 缓存的上半区读取数据，如果是偶数次则从 DMA 缓存的下班去读取数据。并将读取到的数据写入到文件中

```
ioctl(fpga -> fd, IOCTL_FILL_COPY, 0xBB); //从 DMA 缓存的上半区读取数据
fseek(fp, 0, SEEK_END);
fwrite(pRead, 1, VIR_ADDR_SIZE/2, fp);
```

4　实验结果与分析

测试环境及条件：

1）烧写固化 FPGA 程序；

2）TX2 预先安装好 Ubuntu 操作系统；

3）将系统通过 LVDS 接口连接相机，相机帧频 100 Hz，像素数为 640×512，单像素数采样为 14 位将系统 SATA 接口连接固态硬盘；

测试步骤及测试项目：

1）给板卡上电后，电源指示灯点亮，运行 sudo insmod *.ko 加载 PCIE 驱动，驱动正常加载后终端输出如图 11 所示，并加载 PCIE 驱动程序；

2）运行 TX2 端应用程序；

3）打开相机，系统接收相机数据并将数据存储到固态硬盘中；

4）读取固态硬盘中存储的相机数据并对图像帧计数进行判断有无丢失数据情况。

图 11　驱动加载

通过测试发现，当相机以 437.5 Mbps 的速率传输数据时，系统可以完整接收相机数据，数据完整无丢失情况，

5　结束语

本项目采用了新型的处理架构进行图像采集存储，并开展了相应的软硬件设计。该系统采用 FPGA＋GPU 的形式，采用 PCIE 总线实现 FPGA 与 GPU 之间的高速通信，实现了高速数据传输。系统体积小，可应用多种平台。为后续的图像智能处理提供了硬件以及底层基础。

参考文献：

[1] 吴　晴，周　建. 嵌入式图像采集系统的设计与实现 [J]. 电子测量技术，2007，30（2）：90－92.

[2] 胡　锦，谢立红. 基于低功耗 SoC 的微型图像采集系统设计 [J]. 湖南大学学报（自然科学版），2019，46（2）：86－91.

[3] 林文森，李钟慎，洪　健. 基于 ARM 和 CMOS 的图像采集系统设计 [J]. 电子测量，2008，5（5）：12－16.

[4] 高亚男. 基于 STM32 的掌纹图像采集模块的研究 [D]. 北京：北京信息科技大学，2015.

[5] 商婷婷. 基于 MSP430 的网络化视频监控系统的设计 [J]. 电子设计应用，2007（9）：110－111.

[6] 周　泉，马　俊，董亚男，等. 基于低功耗微控制器的图像采集系统设计 [J]. 电子测量技术，2014，37（3）：77－81.

[7] 韩思旭，李　勇，陈卫营，等. 基于 cUDA 并行计算的大地电磁二维有限元数值模拟研究 [J]. 地球物理学进展，2016，31（3）：1095－1102.

[8] 王泽寰，王　鹏. GPU 并行计算编程技术介绍. 科研信息化技术与应用 [J]. 2013，4（1）：81－87.

[9] BLAKE J，MAGUIRE L，MCGINNITY T，et al. Using Xilinx FPGAs to implement neural networks and fuzzy systems [J]. IET Colloquium on Neural and Fuzzy Systems，2015，5（22）：1－2.

（下转第 305 页）