Altera Pci Express

目录

[一、背景知识 1](#_Toc354062150)

[1.1 PCIE系统拓扑结构 1](#_Toc354062151)

[1.2 Altera PCIE硬核IP 1](#_Toc354062152)

[二、板级系统设计 3](#_Toc354062153)

[2.1 Pci Express-to-Avalon-MM地址转换 3](#_Toc354062154)

[2.2 Avalon-MM-to-Pci Express地址转换 4](#_Toc354062155)

[2.3 QSYS设计实例 5](#_Toc354062156)

[2.3.1 端口描述 5](#_Toc354062157)

[2.3.2 sgdma介绍 6](#_Toc354062158)

[2.3.3 系统DMA运行机制 6](#_Toc354062159)

[2.3.4 PCIE中断 7](#_Toc354062160)

[三、利用Windriver进行驱动应用开发 7](#_Toc354062161)

# 一、背景知识

## 1.1 PCIE系统拓扑结构

一个典型的PCIE组织拓扑结构如图1-1所示，图中展现了各种组件以及他们的连接关系。组件包括：一个Root Complex（根复合体）、一个Switch、多个Endpoint（IO/存储设备）、一个Pci Express to PCI/PCI-X桥。所有的连接都通过Pci Express链路，由此图可以看出，PCIE系统基于Point-to-Point连接，并且支持多级扩展。

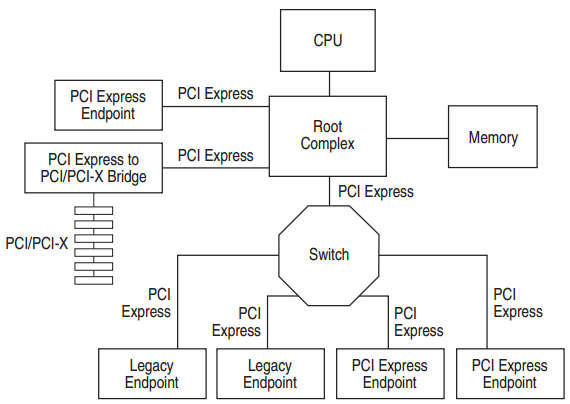


图1-1 典型的PCIE拓扑结构

## 1.2 Altera PCIE硬核IP

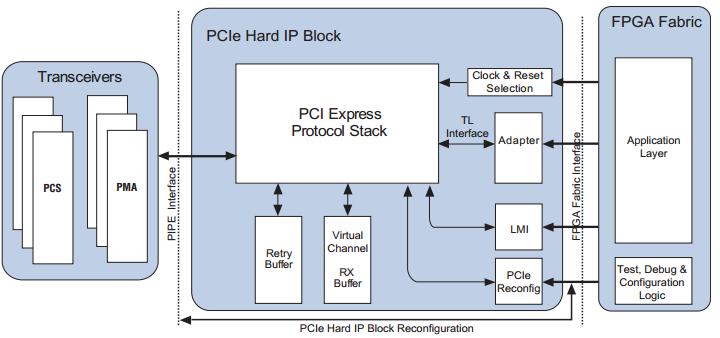


图1-2 Altera PCIE硬核结构

上图中所示，传输层与应用层的接口有：Clock&Reset、Adapter、LMI、PCIe Reconfig。其中Adapter在Mega-wizard设计流程中对应用层提供的是Avalon-ST接口，在SOPC/QSYS设计流程中提供Avalon-MM接口；LMI为应用程序提供访问配置空间的接口；PCIe Reconfig接口实现对PCIE硬核以及GX模块的重配置功能。

Altera PCIE硬核IP可以配置为root port(RP)或者endpoint(EP)模式，在我们的设计中使用EP模式。该硬核模块实现了所有PCIE协议要求的功能以及大部分可选功能。该模块实现的协议层有：Physical、Data link、Transaction。在Transaction layer中实现了设备的配置空间，收发缓冲，流控制credits。Data link layer除了管理包传输还负责流控制初始化及credits更新。Physical layer实现了速度匹配和lane(通道)数目协商。

图1-3展示一个应用层接口为Avalon-MM的协议层次图:

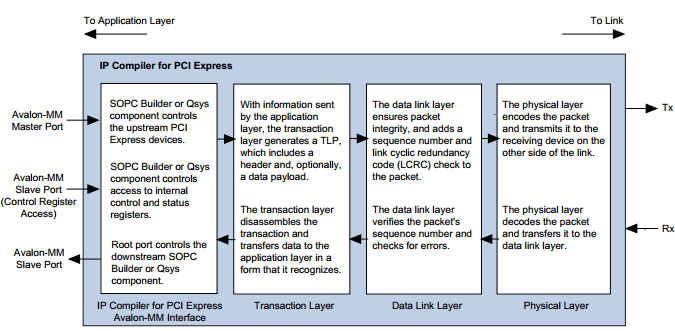


图1-3 协议层

# 二、板级系统设计



图2-1 系统框图

PC端驱动采用Windriver开发，驱动开发好之后编写应用程序可以实现对设备（Qsys Generated endpoint）的配置读/写操作和存储读/写操作。以上操作都是以PC机作为主设备，FPGA作为从设备进行的，这样的操作会消耗过多CPU资源，且速度慢。因此系统还提供了另一种操作方式DMA，由上图可以看出DMA操作时,FPGA作为主设备，FPGA通过发送读/写请求实现对PC机物理内存空间的读写。

## 2.1 Pci Express-to-Avalon-MM地址转换

PC应用软件通过发送读写请求包对设备进行读写，这些请求包中的地址为PCI地址，即映射到PC内存空间的地址。请求包到达FPGA后，需要先将该地址转换为Avalon-MM地址，然后才能发送到QSYS系统内部互联。

这种转换是通过“地址转换表”来实现的。请求包到达FPGA后，PCI地址的高几位（MSB bits）会被替换为地址转换表中的某个值，具体某个值由转换表入口（entry）决定，低几位（LSB bits）保持不变。需要被替换的MSB位数是由Avalon\_MM slave的容量大小决定的。在Altera Pcie硬核IP中支持6个地址转换表入口，每个入口对应一个BAR空间。

下图展现了Pci Express-to-Avalon-MM地址转换过程。

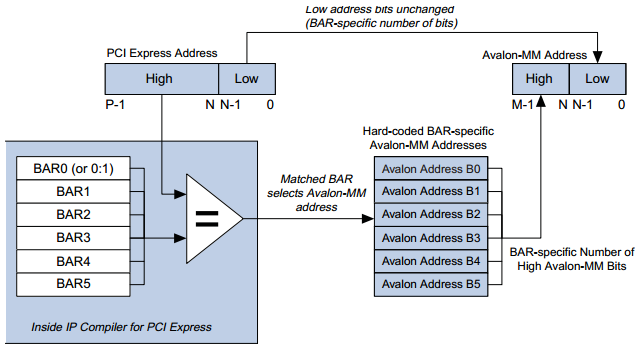


图2-2 Pci Express-to-Avalon-MM地址转换

## 2.2 Avalon-MM-to-Pci Express地址转换

当QSYS中的主设备（Avalon-MM Master比如DMA）要对PC内存进行读写时，需要将本地地址先转换为PCI地址，然后才能发送读写请求包给PC。这也是通过一个“地址转换表”实现的。Avalon-MM地址的高几位（MSB bits）会被替换为地址转换表中的某个值，具体某个值由转换表入口（entry）决定，低几位（LSB bits）保持不变。需要被替换的MSB位数是由PC中分配给FPGA访问的内存空间决定的。这个地址转换表最大支持512个入口，也就是PC能提供512片内存区域给FPGA访问。这个PCI地址支持32bit和64bit宽度。

应该说是Avalon-MM地址最高几位用来选择slave port，次高几位用来选择地址转换表入口，最低几位保持不变。所以最终传给PC的地址是地址转换表中的元素+Avalon-MM地址的LSBs。

下图展现了Avalon-MM-to-Pci Express地址转换过程。

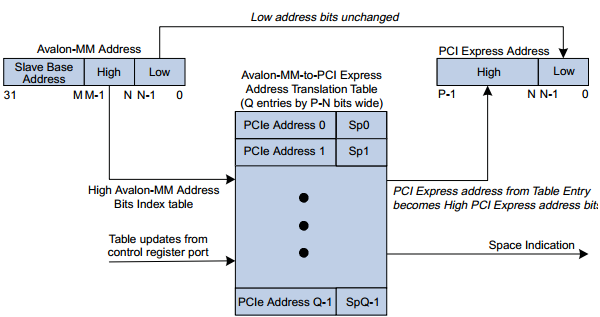


图2-3 Avalon-MM-to-Pci Express地址转换

## 2.3 QSYS设计实例

本设计实现Gen1 x8 PCIE接口，应用层时钟250Mhz，数据位宽64-bit。在该系统中加入了三个组件：Pcie硬核IP、on-chip memory、sgdma。

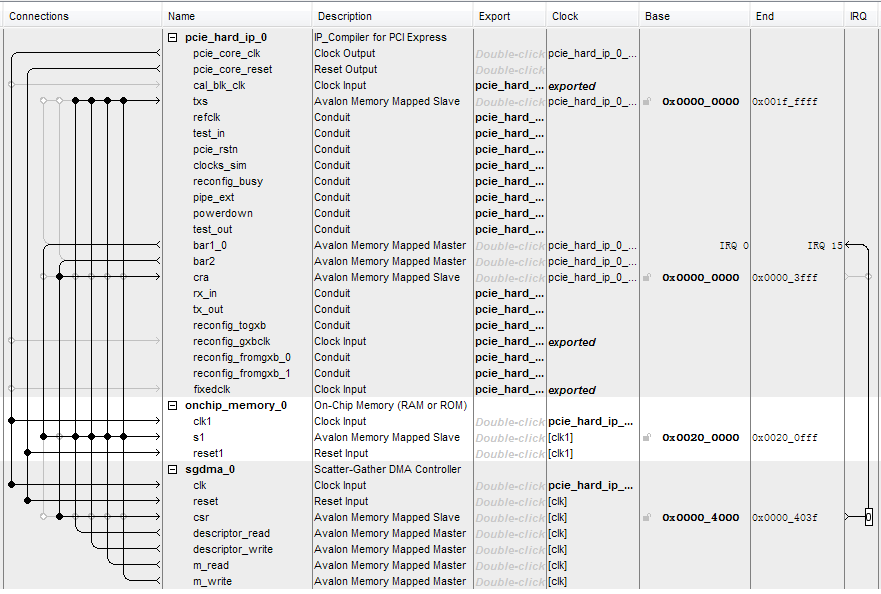


图2-4 QSYS系统目录

### 2.3.1 端口描述

（1）pcie应用层端口描述：

* bar1\_0 ：Avalon-mm master端口，PC软件通过此端口对on-chip memory进行存储器读写；
* bar2 : Avalon-mm master端口，PC软件通过此端口对pcie的cra端口或sgdma的csr端口进行寄存器读写；
* cra : Avalon-mm slave端口，提供pcie ip寄存器的访问接口，这里面也包括了pcie中断寄存器；
* txs : Avalon-mm slave端口，支持突发传输，通过该端口可以对PC内存空间进行读写操作。

（2）sgdma端口描述：

* csr : Avalon-mm slave端口，控制/状态寄存器访问端口；
* destriptor\_read : Avalon-mm master端口，获取描述符；
* destriptor\_write: Avalon-mm master端口，更新描述符；
* m\_read : Avalon-mm master端口，存储器读端口；
* m\_write : Avalon-mm master端口，存储器写端口。

### 2.3.2 sgdma介绍

sgdma(Scatter-Gather DMA)，实现两个组件之间的高速数据传输。与传统DMA不同的是：sgdma能实现从非连续的存储空间到一个连续的地址空间的数据传输。

sgdma是基于描述符(descriptor)实现的，每个描述符对应一个传输控制指令，sgdma能主动获取一连串的描述符，并依此进行数据搬运工作。这也大大减少了CPU资源的消耗。

（1）描述符结构：

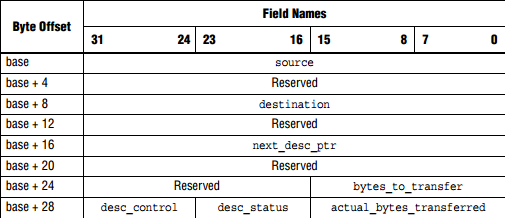


图2-5 sgdma描述符结构

以上结构中定义了数据搬运的源地址，目的地址，下一个描述符指针，传输长度，以及控制/状态字。

（2）寄存器描述：

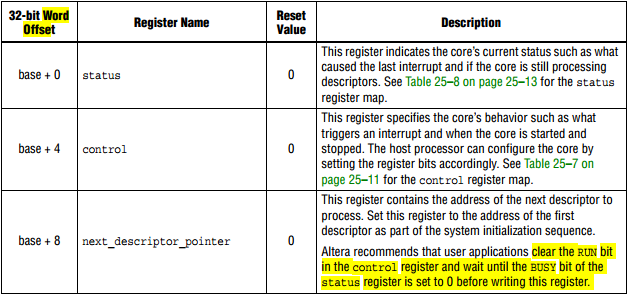


图2-6 sgdma寄存器

具体寄存器的描述请参看数据手册。

### 2.3.3 系统DMA运行机制

在PC应用软件中建立一个缓冲区存储一连串sgdma描述符。然后将第一个描述符的所在物理地址填写入sgdma寄存器next\_descriptor\_pointer中，最后通过sgdma控制寄存器启动dma。dma启动后sgdma会连续从PC内存获取所有的描述符并依次进行数据搬运工作。

sgdma每处理完一个描述符，还可以更新PC内存中被执行完的描述符的某些域，比如OWNED\_BY\_HW，desc\_status，actual\_bytes\_transfered。

### 2.3.4 PCIE中断

图2-4中可以看到系统中有一个中断源，该中断源由sgdma产生，传送给pcie。在pcie的qsys设计流程中，允许pcie有16种中断，每种中断通过一个独立的信号连接到pcie核，我们这里只是用了第0种。当pcie接收到中断使能信号时，会通过MSI机制发送消息给PC机。在PC机中MSI或者MSI-X都属于沿（edge-triggered）中断。

关于pcie中断寄存器的说明请参阅pcie数据手册。

# 三、利用Windriver进行驱动应用开发

使用Windriver能快速开发驱动，对板子进行测试，生成应用程序框架。用户可以根据需要自定义功能，最终发布稳定的驱动和应用程序。

安装驱动后Windriver提供一系列可供调用的High-Level、Low-Level API函数。关于Windriver的使用请参阅wd\_quickstart.pdf。

在这里介绍本设计中实现DMA和中断的几个主要函数：

* WDC\_WriteAddrBlock，WDC\_ReadAddrBlock ：系统库函数，实现对存储器、寄存器的读写操作；
* DMAOpen：自定义函数，打开DMA设备，在其中调DC\_DMAContigBufLock、MyDMAProgram函数；
* DMAContigBufLock：系统库函数，开辟一个缓冲区，返回一个WD\_DMA结构：缓冲区大小、用户地址、内核地址、物理地址等；
* MyDMAProgram：自定义函数，设置地址转换表，构建描述符表，以及初始化sgdma寄存器；
* MyDMAInterruptEnable：自定义函数，开启板级pcie中断；
* WDC\_IntEnable：系统库函数，注册中断服务函数；
* MyDMAStart：自定义函数，启动sgdma；
* interrupt\_handler：中断服务函数；
* WDC\_IntDisable：关闭Windriver的Pci中断；
* WDC\_DMASyncCpu，
* WDC\_DMASyncIo：系统库函数；
* WDC\_DMABufUnlock：系统库函数，释放缓冲区。