文章编号:1671-4598(2020)04-0181-04 DOI:10.16526/j.cnki.11—4762/tp.2020.04.037 中图分类号:TP334.7 文献标识码:A

基于 Avalon 总线的双口 RAM 与 PCI 总线数据交换的设计

魏 珊,魏 丰,陈梦桐
(华中科技大学 人工智能与自动化学院,武汉 430074)

摘要:基于实验室的某同步数据采集卡项目要求,需要设计一个数据共享存储器,实现数据在 FPGA 和上位机之间高速有效地双向传输;为此在可编程逻辑器件 FPGA 的基础上设计了基于 Avalon 总线的单时钟真双端口模式的双口 RAM 共享存储器;其存储容量为 256 字节,数据线和地址线定制 8 位;为了避免数据丢包现象的出现,采用了奇偶交换页的思想,16 字节为一页;最后对所设计双口 RAM 进行验证测试,结果表明所设计的双口 RAM 实现了 12 字节报文信息在 FPGA 与上位机之间的高效传输;50 000 次的测试结果表明所设计的双口 RAM 实现了信息不丢包的实时传输;该设计充分利用了 FPCA 现有的存储资源,减少了电路设计的复杂性,同时达到数据的高效率传输。

关键词:双口 RAM;Avalon 总线;FPGA;奇偶交换页

Design and Implementation of Data Exchange between Dual Port RAM and PCI Bus Interface Based on Avalon Bus

Wei Shan, Wei Feng, Chen Mengtong
(School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan 430074, China)

Abstract: Based on the requirements of a synchronous data acquisition card project in the laboratory, we need to design a data shared memory to achieve high-speed and efficient bidirectional transmission of data between the FPGA and the host computer. To this end, we designed a dual-port RAM shared memory based on Avalon bus in single clock true dual port mode based on programmable logic device FPGA. Its storage capacity is 256 bytes, and data lines and address lines are customized to 8 bits. In order to avoid the phenomenon of data packet loss, the idea of parity switch page is adopted, 16 bytes is one page. Finally, the verification test of the designed dual-port RAM shows that the designed dual-port RAM realizes the efficient transmission of 12-byte message information between FPGA and computer. The test result of 50 000 times indicates that the designed dual-port RAM realizes real-time transmission of information without packet loss. This design takes full advantage of FPCA's existing storage resources, reducing the complexity of circuit design while achieving efficient data transfer.

Keywords: dual port RAM; Avalon bus; FPGA; parity exchange page

0 引言

实现数据的暂存和双向传输,一般会采用 FIFO 和双口 RAM 这两种模式作为数据交换和存储的共享存储器。FIFO (first input first output) 是数据从一端先写入然后在另一端优先读出来的存储器,一般用于异步时钟的数据交换系统中,充当两个不同时钟系统间的数据暂存器。所以称它为异步 FIFO。双口 RAM 是具有两个端口的静态存储器,两个端口分别有其自己的数据线,地址线和控制线,由于端口两边有独立的总线,所以两个 CPU 可以对双口 RAM 同时进行读写操作,或者一个 CPU 在一端对其读操作,另一个 CPU 在另一端对其写操作。双口 RAM 的此特性实现了 CPU 之间数据的高效灵活交换。实验室的同步数据采集卡项目要求上位机可以根据需要对存储器任意单元内的数据进行读写,这就对共享存储器的读写方式有了一定的要求。FIFO 存储器的读写严格的按照先进先出的规则,满足不了该项目的需求。双口 RAM 存储器的读写是灵活的,可以根据需求清楚地了解到双口 RAM 各个存储单元的信息,正好符合此项目的要求,因此本文选择双口 RAM 作为该项目的共享存储器[1]。

双口 RAM 存储器的实现既可以采用现成的双口 RAM 芯片,也可以通过 Verilog HDL 语言在 FPGA 内设计功能模块,然后在 Nios II 开发环境中封装成 IP 核来构造完成。现成的双口 RAM 芯片有 Integrated Device Technology 公司研发的 IDT7130,IDT7130 是一种 1K * 8 高速的双端口的静态 RAM[2],RAM 两个端口的数据,地址和控制线是独立且对称的,两个 CPU 通过 IDT7130 内部的硬件地址仲裁信号 BUSY 信号可以从任一端口对 IDT7130 进行完全异步的操作。此外 IDT7130 具有自动进入低功耗状态的特点,

收稿日期:2019-09-02; **修回日期**:2019-09-19。

作者简介:魏 珊(1994-),女,湖北孝感人,硕士研究生,主要从事检测装置与自动化仪表方向的研究。

因此 IDT7130 在民用和军用上都很受欢迎。为了充分利用 FPGA 上现有的存储器资源，同时简化该项目硬件设计的复杂程度，本文采用通过对 FPGA 内的存储器进行配置并将其封装成 IP 核的形式来构造双口 RAM。对其进行功能测试后结果表明本文设计的双口 RAM 实现了数字量和时标信息在 FPGA 与 PCI 总线接口之间高速有效且不丢包的实时传输。

1 Avalon 总线和 PCI 总线接口芯片

Altera 公司为了提高主从设备之间数据交换效率和速率，开发了一种新的总线结构 Avalon 总线。Avalon 总线是 Nios II 处理器内部模块和外围设备数据交换的一座桥梁，模块之间通过 Avalon 总线连接形成片上可编程系统 SOPC。Avalon 总线的特点是采用了分离的地址、数据和控制总线，没有信号选择电路，大大简化了数据传输的复杂性，提高了传输效率；Avalon 总线上的信号是高低电平，这样简单的信号可以在总线上高效地传输[3]。PCI 总线接口芯片 CH365 主要将高速复杂的 PCI 总线转换成八位并行的数据线和地址线[4]，上位机驱动程序通过总线接口芯片 CH365 对双口 RAM 共享存储器进行读写。实现数据在 FPGA 和上位机之间双向实时地传输。图 1 是 Avalon 总线和 PCI 总线接口芯片 CH365 通过共享存储器双口 RAM 进行数交换示意图。

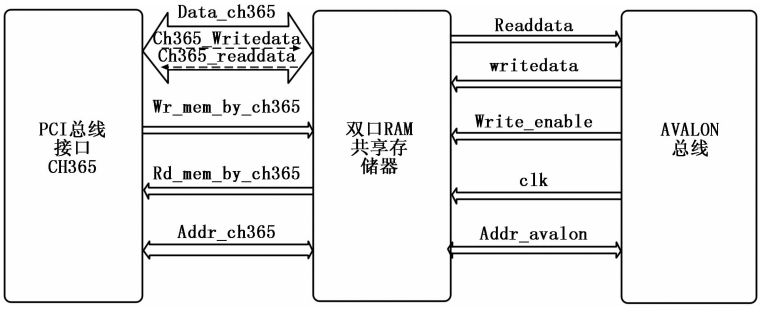

图 1 PCI 总线接口芯片 CH365 通过双口 RAM 与 Avalon 总线通信

说明：PCI 总线接口芯片 CH365 将复杂的 PCI 总线转换为八位数据线和地址线，图 1 可以看出接口芯片 CH365 对双口 RAM 的读写操作的总线是公用的，Avalon 总线的读写数据线则是分开的，为了充分使用 FPGA 现有的片上存储器资源，本文在设计单时钟真双端口 RAM 的时候需要对 CH365 的数据线进行一定的配置，使其符合 FPGA 自带的存储器的使用规则。这一特点也是设计双口 RAM 的初衷之一和难点之一。由图 1 可知整个系统只有一个时钟 clk，该时钟来自于 FPGA 的系统时钟，其作用是让两个 CPU 对双口 RAM 有效有序的读写操作。rd _ mem _ by _ ch365 和 Wr _ mem _ by _ ch365 分别是 CH365 对双口 RAM 的读写使能信号。Write _ enable 是 Avalon 总线的对双口 RAM 的写使能信号。

2 FPGA 中双口 RAM 的分类

图 2 将 FPGA 内的存储器做了一个清晰的分类。

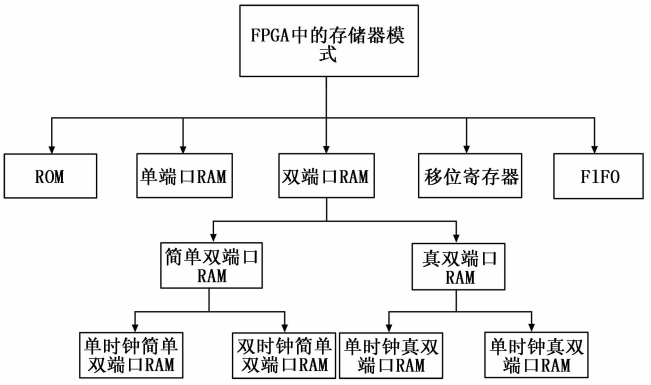

图 2 FPGA 内的存储器

Altera 公司开发的 Cyclone IV EP4CE6E22C8N 这款 FPGA 芯片具有嵌入式存储器结构，这一特点满足了该型号的 FPGA 对片上存储器的需求。嵌入式存储器的结构由一系列 M9K 存储器模块构成，对这些 M9K 存储器模块进行一定的配置，即可以实现各种各样的存储器功能，例如 RAM、移位寄存器、ROM 以及 FIFO 缓冲器。M9K 存储器具有很多特性，例如存储器的每一个端口都具有独立的读使能和写使能信号，在 Packed 模式下，M9K 存储器模块可被分成两个 4.5K 单端口 RAM，同时具有可变端口配置模式等。存储器根据读写端口的个数可以被配置成简单双端口和真双端口模式。

简单双端口模式：这种模式下的存储器一边只有一个读端口和另一边只有一个写端口，即这种模式支持不同位置的同时读写操作。

简单双端口 RAM 在读的这一端口只有与读相关的所有信号，例如读地址信号 wraddress []，读使能信号 wren，读时钟和读使能时钟。同理，在写的这一端口只有与写相关的所有信号。在此模式下，M9K 存储器模块支持独立读写使能信号 rden 和 wren。在没有进行读写操作时，一般讲 rden 信号保持在低电平（无效状态），从而降低功耗。相同地址上的 Read－during－write 操作能够在相应的位置上输出“New Data”数据，或者输出“Old Data”数据。

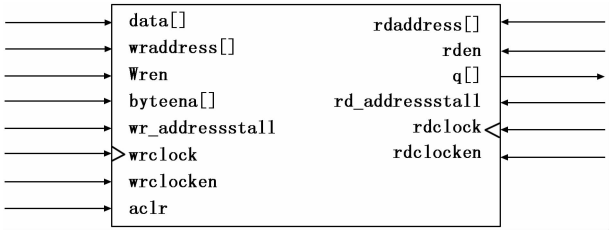

图 3 简单双端口 RAM

真双端口模式：这种模式下的存储器具有两个独立的写端口和两个独立的读端口，这种模式支持双端口操作的任何组合，例如在两个不同时钟频率上的两个读操作，两个写操作，或者一个端口的读操作和另一个端口的写操作[5]。

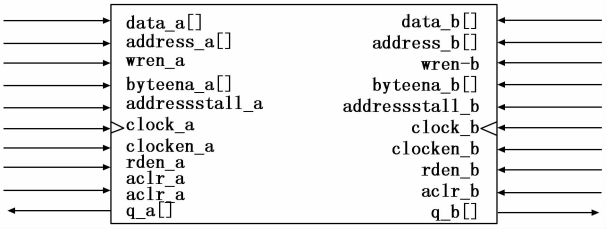

图 4　真双端口 RAM

相比于简单双端口模式，真双端口模式下的 RAM 两端口既有读信号也有写信号，因此在此模式下的 RAM 可以对存储器某一端口任意独立地进行读写操作，而不是固定哪个端口只能读或者只能写操作。对于存储器的操作更加灵活。当存储器两个端口对存储器同一位置进行读写操作时，可能会出现冲突，此时需要外部的冲裁电路来解决。

除此之外根据双端口 RAM 的时钟个数又可以细分为单时钟简单双端口 RAM、双时钟简单双端口 RAM、单时钟真双端口 RAM 和双时钟真双口 RAM。时钟是一个系统的脉搏，在所有系统中扮演着非常重要的角色，系统有条不紊的运行离不开时钟。存储器也一样，对存储器进行正确地读写操作离不开时标信号。观察 IDT7130 双端口静态 RAM 芯片的读写时序图，我们发现 IDT7130 芯片没有时钟信号，仔细阅读它的内部功能电路你会发现有 I/O 控制单元，地址译码器，存储器阵列以及仲裁逻辑控制的电路，在这些电路的基础上 IDT7130 采用中断方式交换信令的方法以及有效操作 BUSY 信号获得时标信号，从而对其进行高效有序的读写。FPGA 内部的存储器是直接采用时钟信号的形式来完成对存储器正确有序的读写操作，时钟信号控制着操作的先后次序，达到了数据交换的准确性和高效性。

根据图 1，该项目中的双端口共享存储器 RAM 只有一个时钟。基于该项目的需求，本文主要设计的是只有一个时钟的真双端口 RAM。

3　双口 RAM 在 FPGA 中的实现

本文采用自上而下的设计思想设计了共享存储器双口 RAM[6]。选择用 Altera 公司研发的的 Cyclone IV E 系列 EP4CE6E22C8N FPGA 器件，该系列器件具有 6K 的逻辑单元、270 Kbits 的嵌入式存储器、2 个通用的 PLL，最大用户 I/O 可以达到 179 个、其中 Nios II 的最高频率可以达到 170 MHz。该器件具有低成本高性价比的特性，被广泛使用。本文在这款 FPGA 器件内实现了 256 字节存储容量的单时钟真双端口 RAM 共享存储器。主要通过 Verilog HDL 语言来进行设计。首先我们选择的硬件开发环境是 QuartusII 第 11.0（64 位）版本，该软件主要通过硬件描述语言 Verilog HDL 实现 FPGA 的资源的分配利用[7]，Verilog HDL 编写 .V 文件，然后对其编译、综合，最后下载到 FPGA 内即可实现对 FPGA 资源的使用。该开发环境还可以对所设计的模型的时序进行分析，从而检测该模型是否符合设计需求[8]。选择 NiosII 的第 11.0 版本为软件开发环境。SOPC 的配置文件（后缀名 .sopcinfo）是连接软件和硬件资源的重要文件[9]。在开发环境 QuartusII 内部其实有很多 FPGA 内部资源的例程代码供参考，本文开发的单时钟真双端口 RAM 就是参考 QuartusII 内部的 RAM 例程来完成的。

本文采用模块化的程序设计，设计了两个模块，即两个 .V 文件。一个是 QuartusII 内部的单时钟真双端口 RAM 文件，另一个是根据项目中共享存储器两端口总线的特点设计了一个将 Avalon 总线和 CH365 接口总线连接起来的文件。对于第一个文件首先将 QuartusII 内部的单时钟真双端口 RAM 例程调出来，然后对其修改，具体步骤是：在 QuartusII 开发环境中新建一个 Verilog HDL File 文件，回到主界面点击菜单栏中的 Edit 下拉菜单中有一项是 Insert Template，出现的对话框有语言例程的分类，包括 AHDL 语言、VHDL 语言和 Verilog HDL 语言等，本文选择 Verilog HDL，里面包括 Full Design、Logic、Synthesis Attributes 等，本文选择 Full Design，内部包括 RAMs and ROMs，选中会发现有很多例程，根据本项目需求选择 True Dual Port RAM（single clock），将其例程代码导入到最初建立的 Verilog HDL File 文件中，对其做了一点修改，这样单时钟真双端口 RAM 的内部基本结构获得了，但是该基本结构缺少端口信号的具体设计。联系实际情况本文设计了第二个 Verilog HDL File 文件。

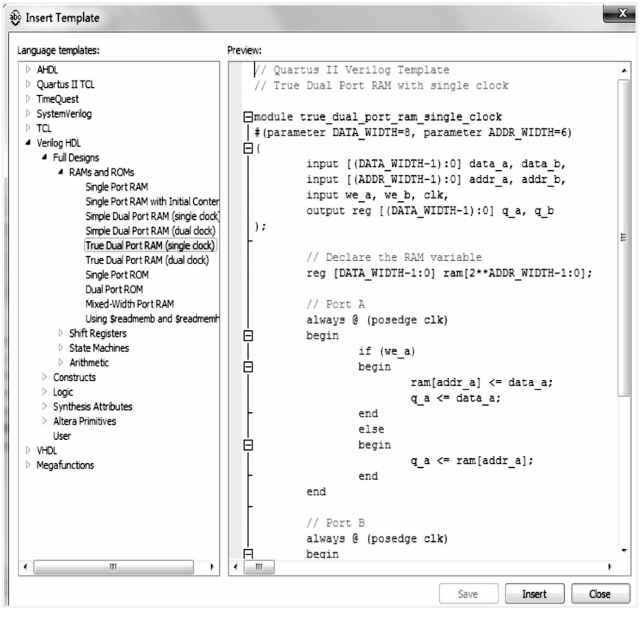

图 5　Quartus Ⅱ内部的单时钟真双端口 RAM 例程

在对第二个 .V 文件设计的时候，考虑到 FPGA 中的 SOPC 系统各个模块之间数据交换的总线标准是 Avalon 总线，本文设计的单时钟真双端口 RAM 最终要嵌入进 SOPC 系统中，而 CH365 接口总线的读写数据线和 Avalon 总线的读写数据线存在差异，CH365 接口总线的读写操作公用一条数据总线，Avalon 总线具有单独的读数据线和单独的些

数据线。考虑到这一点，我们来设计第二个 .V 文件[10]，主要解决第一个 .V 文件没有解决的问题以及配置 CH365 接口的读写数据总线。首先配置第一个 .V 文件定义的双口 RAM 的各个端口信号，主要包括系统时钟信号 clk，两端地址总线，数据总线的定义和读写使能信号的定义。然后合并 CH365 接口总线这边的数据线，实现的方法是程序中自定义了一个 CH365 数据总线输入输出控制信号 dir _ ch365，在系统时钟有效地前提下，当 365 的写双口 RAM 的写使能信号有效的时候控制信号 dir _ ch365 置 1，此时 CH365 的数据总线作为输出，向双口 RAM 写数据。反之，在系统时钟有效的前提下，当 365 的读双口 RAM 的读使能信号有效的时候控制信号 dir _ ch365 置 0，此时数据 CH365 的数据总线作为输入，读取双口 RAM 中的数据。

```
//ch365
always @(posedge clk)
    if(!wr_mem_by_ch365)
        dout_r_ch365 <= data_ch365;

//ch365
always @(posedge clk)
    if(!rd_mem_by_ch365)
        din_r_ch365 <= din_ch365;
    else
        din_r_ch365 <=  {DATA_WIDTH{1'bz}};

//ch365
always @(posedge clk)
    if(!wr_mem_by_ch365)
        dir_ch365 <= 1'b1;
    else
        dir_ch365 <= 1'b0;

dmodule
```

图 6 CH365 接口数据总线处理部分代码

4 将双口 RAM 封装成 IP 核

将自定基于 Avalon 总线的单时钟真双端口 RAM 模块封装成 IP 核嵌入到 SOPC 系统中需要在工具 SOPC Builder 中完成。首先在菜单栏 Project 中新建自定义组件点击 New component，出现图 7 的对话框。首先是组件编辑（component editor）的简介。第二步硬件描述语言文件（HDL Files）的加载，之前写好的两个 .V 文件就是要从这里加载进去，加载的同时系统会自动编译。第三步 .V 文件中自定义的端口信号与双口 RAM 内部信号之间一一对应的配置了，主要包括信号类型、信号宽度以及信号的方向。第四步信号线参数以及时序的配置。第五步数据总线和地址总线位宽的配置，数据总线和地址总线本在设计的都是 8 位。最后给配置完的 IP 核和命名，以及它属于的 IP 核组，方便使用者找到并使用。点击 Finish 后，系统首先进行编译最后生成自定义的 IP 核组件。这个 IP 核可以供使用者使用，同时还可以根据设计者的需求对其进行修改，方便灵活。

前文说过，双口 RAM 作为数据交换的共享存储器也有其不足之处。当两端的系统同时对双口 RAM 的同一单元读数据或者写数据的时候，会出现错读的现象，读到的数据不是使用者所需要的数据。针对这一现象，本文采用奇偶交换页的思想来访问双口 RAM。双口 RAM 的每 16 个字节单元为一页，分为奇数页和偶数页，偶数页作为双口 RAM

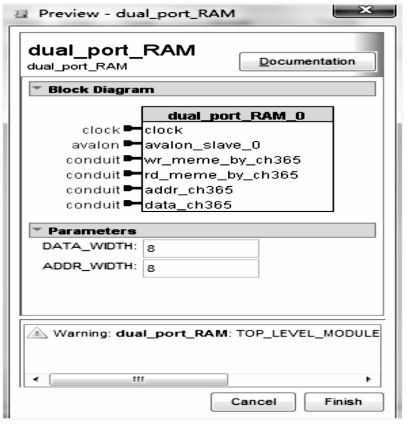

图 7 单时钟真双端口 RAM

的起始页[1]。当某一系统往奇数页写完或者读完数据后，将奇数页的标志位置 1。当系统操作完偶数页时，偶数页的标志位置 0。系统再次访问这些单元的时候先访问奇偶标志位，判断完成以后再访问存储单元里面的数据。这样的访问存储单元的思想有效地避免了数据错读的显小，提高了双口 RAM 双向通信的效率。

5 双口 RAM 的功能测试

为了验证本文设计的单时钟真双端口 RAM 的正确与否，本文首先在硬件开发环境 QuartusII 11.0 中设计设计硬件原理图构成 SOPC 硬件系统，然后在软件开发环境 NiosII 11.0 中编写测试双口 RAM 的代码。方法是向已经嵌入到 SOPC 中的双口 RAM 写入简单的数据，然后读出双口 RAM 中的数据，最终在显示终端打印出来。其测试结果如图 8 所示。为了验证本文设计的双口 RAM 是否可以通过 PCI 总线和上位机进行数据交换，本文还在 LINUX 环境下建立 PCI 字符设备，对所建立的 PCI 字符设备读写操作，测试双口 RAM 的功能，其测试结果如图 9 所示。最后在 NiosII 中编写软件将项目中设计的同步时钟采集卡通过 CPS 接收机采集到的时标信息传送到台式机，测试双口 RAM 能否在 FPGA 和上位机之间实现数据正确高效的双向传输。其结果如图 9 所示。

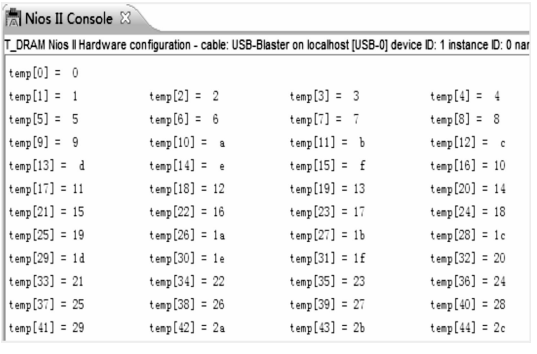

图 8 双口 RAM 与 NiosII 之间的通信测试结果

说明：图 8 的结果显示了数据可以正确地在双口 RAM 和 NiosII 之前传输，说明本文所设计的单时钟真双端口 RAM

（下转第 231 页）