文章编号:1671-4598(2020)06-0125-04 DOI:10.16526/j.cnki.11—4762/tp.2020.06.026 中图分类号:TN913.8 文献标识码:A

基于 FPGA 的串行多节点数据采集传输系统的应用研究

纳杰斯，岳 雷，马 雪，郭春福
（昆明船舶设备研究试验中心，昆明 650051）

摘要：多节点分布式传感器的数据采集与传输系统有着广泛的应用，其物理层实现主要分为串行及并行两种方式；针对特定的应用场景，选用串行传输的实现方式，设计以 FPGA 芯片、RS485 芯片及 AD 芯片为核心的硬件平台，使用 Verilog 语言设计 FPGA 的 RTL（register transfer level）逻辑代码，并在 Modelsim 中完成了功能仿真验证；最终实现了物理层为串行传输、传输层为总线模型的数据采集传输系统，该系统 RTL 代码设计简单，维护性强，可靠性高，占用芯片资源少，具有一定的工程应用价值。

关键词：多节点数据传输；数据采集系统；FPGA 设计

Application Research of Serial Multi—node Data Acquisition and Transmission System Based on FPGA

Na Jiesi，Yue Lei，Ma Xue，Guo Chunfu
（Kunming Ship Equipment Research Test Center，Kunming 650051，China）

Abstract：The data acquisition and transmission system of multi—node distributed sensors has a wide range of applications，and its physical layer implementation is mainly divided into serial and parallel modes. The serial transmission is adopted for this project，and the hardware platform with FPGA chip，RS485 chip and AD chip is designed. The RTL（register transfer level）code of is designed by Verilog and functional verification is complete in Modelsim. The RTL code of the system is simple in design，strong in maintenance，high in reliability，and occupies less chip resources，and has certain engineering application value.

Keywords：multi—node data transmission；data acquisition system；FPGA design

0 引言

多节点传感器的数据采集传输系统有着广泛的应用，在具体的实现技术上，物理层实现方式主要分为并行传输和串行传输[1]。如图 1 所示，图（a）中每个节点都有独占的有线信道与主机直接相连，并行传输的数据传输速率只与节点本身的传输速率有关，与节点数无关；节点控制程序设计简单，各节点间相对独立[2]。图（b）中每个节点与下级节点相连，各节点共用信道与主机相连，在总线带宽相同的情况下，串行传输速率与节点数量负相关，节点间有数据或命令交互。

相比并行传输，串行传输的优势主要在于电缆长度较短、直径较小、重量较轻。式（1）、式（2）分别为并行传输与串行传输方式下，电缆长度与节点数量的关系为：

$$D_p = \sum_{n=0}^{N} d_0 + n * \Delta d \tag{1}$$

$$D_s = d_0 + N * \Delta d \tag{2}$$

式（1）～（2）中，N 为节点数量；$n=1，2，\cdots，N$，为每个节点序号；Δd 为节点间距，d_0 为主机与最近一个节点间距。

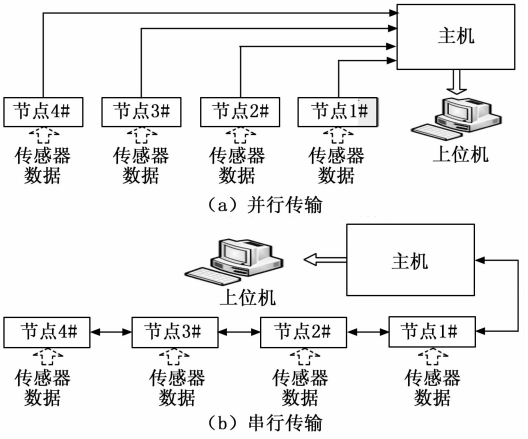

图 1 多节点数据传输物理层实现方式

在某些应用场景中，对电缆的重量及直径较为敏感[3]。针对这些应用场景，本文选用串行传输的方式，使用 Verilog 语言实现了水下拖缆多节点传感器数据同步采集与传输的 RTL 代码设计，并在 Modelsim 中进行了功能仿真验证。

1 串行传输物理层设计

在串行传输方式的物理层设计上，为了进一步减小电缆重量及直径，节点间采用半双工的工作模式，为保证较高的传输信噪比，节点间使用双绞线连接，数据以差分信

收稿日期：2019-10-11； 修回日期：2019-11-12。

作者简介：纳杰斯(1988-)，男，云南玉溪人，硕士研究生，工程师，主要从事数字电路设计、水声信号处理方向的研究。

号的形式传输。具体各节点使用 Intel 公司的 Cyclone 系列 EP4CE15 FPGA 为主控芯片，主机使用了硬件资源更丰富的 EP4CE75 FPGA 作为主控芯片，FPGA 控制 TI sn65hvd23 芯片将 3.3 V TTL 单端逻辑电平转换为符合 TIA/EIA－485 协议的差分信号，驱动节点间的双绞线进行通讯，sn65hvd23 芯片最高支持 100 m 传输距离内高达 25 Mbps 的传输速率。图 2 展示了各节点与主机的连接关系，图中只画出了两个节点，实际项目中使用了 1 个主机和 4 个节点，主机与最近节点间距 $d_0 \approx 18$ m，各节点间距 $\Delta d \approx 17$ m。以主机为起始，编号为 0＃，各节点由近及远，分别编号为 1＃、2＃、3＃、4＃。

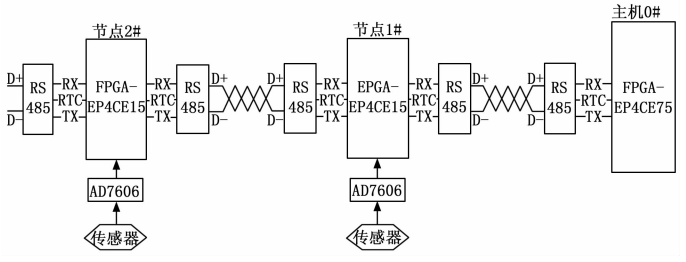

图 2 各节点与主机间的连接关系

2 串行传输链路层设计

2.1 逐级交付模型

不同的链路层传输方式，对应着不同的链路层传输模型，逐级交付模型是最常见的一种。逐级交递模型中，数据传输的单位是帧，一帧数据为单个节点单次采集的数据，本项目中，每个节点的 AD 单次采集得到的数据总长度为 48 bit，48 bit 即为一个数据帧。

逐级交递模型传输方法是，从 4＃节点开始，将本节点数据帧传输到 3＃节点，3＃节点将 4＃节点的数据域与本节点数据一起打包，再往 2＃节点传输。以此类推，直到所有数据交付给主机。整个过程中，从 4＃节点到 1＃节点，节点间传输的数据越来越多，每个节点包含该节点及该节点之前所有节点的数据[4]。图 3 展示了逐级交付传输模型的工作流程，图中为了简明只画了 3 个节点，Dx 代表第 x 个节点采集的数据。

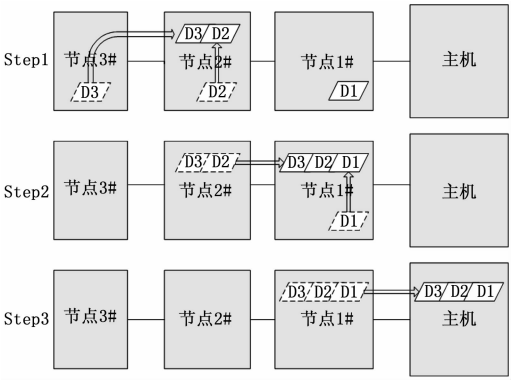

图 3 逐级交付传输模型

2.2 流水线模型

在流水线模型中，传输数据的单位是帧，帧的定义与上文 2.1 节定义相同。流水线模型传输方法是，在第一个时刻，各节点将本节点采集到的数据装帧传输到下一节点，各节点在传输本节点数据帧的同时，接收并缓存上一节点发送来的数据帧；在第二个时刻，每个节点将上一时刻接收到的数据帧传输到下一节点；在第三个时刻，各节点重复第二时刻动作，直到上一节点无数据交付，且本节点已将缓存区数据交付到下一节点，则当前节点停止工作。当所有数据交付到主机时，整个过程中所有节点并行执行接收发送的操作，就像工厂的流水线作业。图 4 展示了流水线传输模型的工作流程。

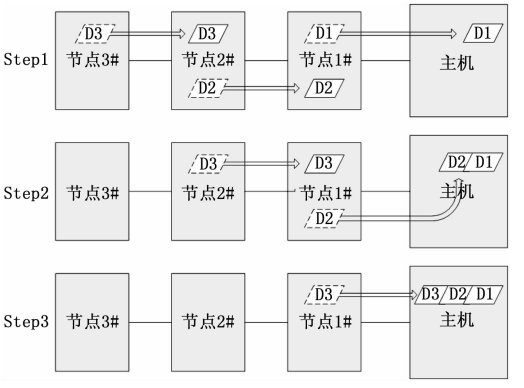

图 4 流水线传输模型

2.3 总线模型

在总线模型中，数据传输的单位是比特。总线模型传输方法是，首先在每个数据采样周期内，为每个节点分配时间片，各节点在自己的时间片内才可传输数据，不在自己时间片内的节点，使用寄存器锁存端口信号并直接转发到下级节点。时间片内的节点数据在其他节点中逐级转发，形成数据直达主机的信号通路，数据以比特流的形式直接交付给主机。总线模型传输的方法是，在第一个时间片内，4＃节点首先占用信道，3＃、2＃、1＃节点使用寄存器在节点间形成类似物理上的连通，使得 4＃节点数据可直接交付给主机；在第二个时间片内，已完成数据传输的 4＃节点停止工作，3＃节点开始占用信道并将数据交付给主机；以此类推，直到所有节点都将数据交付给主机。这种传输模式类似于在一条总线上挂了 1 个主机和 4 个节点，各节点按顺序分时占用总线资源与主机通讯。图 5 展示了总线传输模型的工作流程。

2.4 传输模型对比

三种模型均可实现多节点数据传输的功能需求，表 1 描述了各链路层传输模型的优缺点。

3 节点同步

多节点的数据传输还需要考虑节点间的同步问题，同步是保证各节点协调工作的基础[5]。本文中，各节点采用误差为 20 ppm（part per million）的晶振作为系统时钟源，各节点的数据采样率（AD 采样周期）为 20 kHz，若节点间

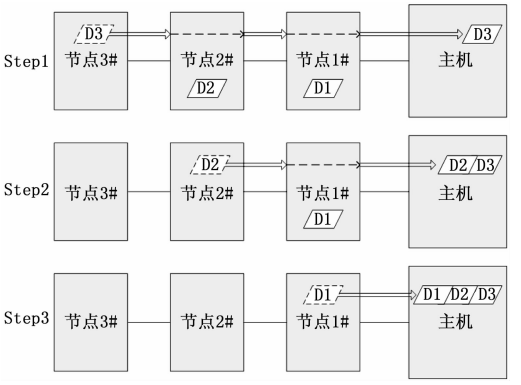

图 5　总线传输模型

表 1　各链路层传输模型对比

实现方式	优点	缺点
逐级交付模型	逻辑清晰，代码设计简单	随节点数量的增多，传输数据量及传输时间急剧增加，靠近主机的节点占用资源急剧增加
流水线模型	随节点数量的增多，传输数据量及传输时间线性增加	同一节点并行处理接收与发送，控制逻辑相对复杂
总线模型	逻辑清晰，代码设计简单；随节点数量的增多，传输数据量及传输时间线性增加；占用器件存储资源少	无明显缺点

能接受的最大时钟偏差为一个数据采样周期，考虑最坏情况，系统每运行 $T=\frac{1/(20\times10^3)}{20\times10^{-6}}=2.5$ s 后，节点间就会相差一个采样周期，从数据上看即表现为某些节点数据丢失[6]。更恶劣的情况是，如果采用总线模型的传输方式，当时钟偏差超过一个时间片时，则两个节点会同时占用同一个时间片，导致系统无法正常工作。

为了避免增加额外的同步信号线，同步信号与数据传输共用物理信道。具体方法是，将某个节点或主机设置为同步源，负责为其他节点发送同步命令，节点在收到同步命令后，同时启动 AD 采集，多节点数据传输完成后，再等待下一次同步命令。同步源在节点内实现称为节点同步源，在主机内实现称为主机同步源。

节点同步源是将同步源放置在数据传输的起始节点内，本项目中在 4＃节点内以 20 kHz 的频率发送同步命令。该方法同步命令与数据传输的方向相同，在半双工的工作模式中，无需考虑数据传输方向的切换控制逻辑，代码实现简单。但是存在可靠性较低的问题，如果同步源节点损坏，系统将无法正常工作。另外，同步源逻辑在节点内实现，功能划分模糊，因为其他节点依赖同步源节点的同步命令工作，同步源节点与其他节点形成一种主从关系，而非原本的等同关系。

主机同步源的方法是将同步源放置在主机内，这种方法克服了节点同步源的可靠性低、功能划分模糊的缺点，但是同步命令与数据传输的方向相反，各节点及主机需要额外的半双工传输控制逻辑切换传输方向。表 2 是这两种实现方式的对比。

表 2　同步实现方式对比表

同步方式	优点	缺点
节点同步源	代码设计简单，传输线路为单向传输工作模式	可靠性较差，同步源节点故障导致全系统故障；功能划分模糊，各节点间存在主从关系
主机同步源	可靠性较高，个别末端节点故障并不影响系统工作；功能划分清晰，各节点功能完全相同	代码设计稍微复杂，需要设计半双工传输控制逻辑

4　工作流程

本文选用“总线模型－主机同步”的方式实现多节点数据采集与传输系统，系统由 4 个节点和 1 个主机组成，各节点及主机使用 PLL（phase locking loop）生成 100 MHz 的系统时钟，数据采样率为 20 kHz，每个采样周期对应 5 000个系统时钟，令每个节点时间片占用 1 000 个系统时钟，另外 1 000 个系统时钟分配给主机用于发送同步命令等工作。图 6 为各节点工作流程图，图中" 等待路径延迟" 是为了等待所有节点都收到同步命令后再同时开始 AD 采集，路径延迟时间为：

$$T_{path}=(N_{total}-N_{self})\times T_{node}$$

T_{path} 的单位为系统时钟个数。其中 $N_{total}=4$，表示总节点数；$N_{self}=1$，2，3，4 表示各节点序号；$T_{node}=4$ 表示节点之间数据传输的延迟，因为使用了两级寄存器消除亚稳态，再加上接收端与发送端寄存器，共有 4 个周期的时钟延迟。

“等待所有节点传输完毕”是为了各节点能同时转换数据传输方向为接收，等待时间为：

$$T_w=(N_{self}-1)\times T_{slice}$$

T_w 单位为系统时钟个数。$T_{slice}=1\ 000$，表示时间片占多少个系统时钟。

主机工作流程与节点工作流程类似，篇幅有限，不在赘述。

5　代码设计及仿真

本文使用 Verilog 硬件描述语言实现系统代码设计。所有节点功能相同，采用宏定义及条件编译等手段保证各节点代码的高复用性，不同节点的代码仅仅是节点号的宏定义不同，这样极大的提高了系统的维护性和扩展性。主机数据接收模块与节点的数据接收模块相同，另外还负责同步源的实现及同步命令的产生，需要单独设计。为了增加代码可靠性，在设计中还有以下几种手段。

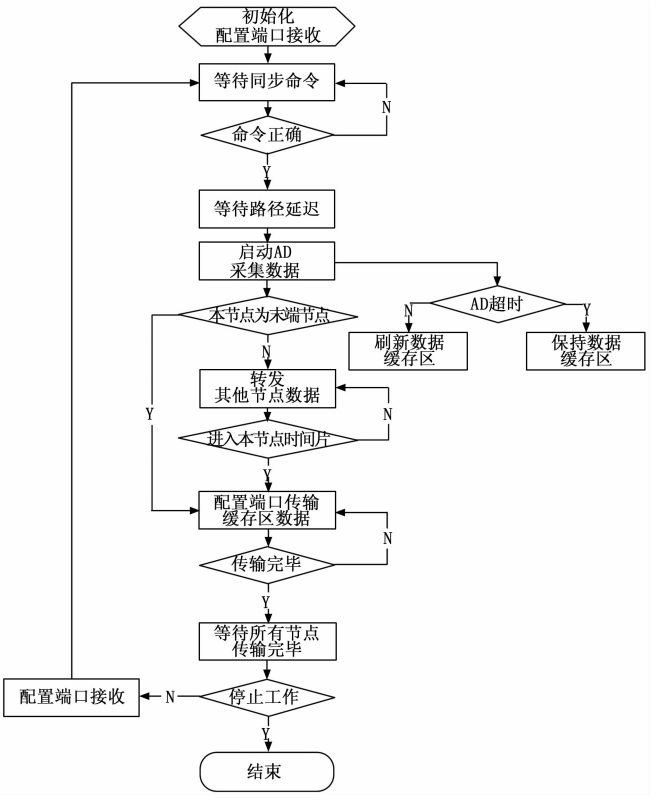

图 6 节点工作流程图

5.1 亚稳态消除

数字电路设计中，除了逻辑设计外，还要考虑时序设计，在时序分析满足的情况下，另一个较重要的问题是亚稳态。上文所述的节点同步只是保证各节点同步启动 AD 采集。在微观上，各节点的时钟信号不可能严格同步，属于不同的时钟域，当节点间数据传输信号跳变沿刚好处于采样时钟沿时，会导致采样信号的建立保持时间不满足，此时用于锁存数据的触发器输出端会在比较长时间内处于不确定状态，这个状态称为亚稳态。亚稳态将导致逻辑误判，严重情况下输出的不稳定导致下一级寄存器也产生亚稳态，称为亚稳态传播，使得故障面扩大。使用两级寄存器采样，可有效减少亚稳态传播的概率[7]。图 7 为两级寄存器采样的代码，图中 p1 _ rx 为接收端口的异步信号，经两级采样后得到同步信号 p1 _ rx _ syn。

```
always@(posedge clk_pll_100m or negedg
begin
  if(!rst_n)
  begin
    p1_rx_tmp0 <= 1'b1;
    p1_rx_tmp1 <= 1'b1;
    p1_rx_syn <= 1'b1;
  end
  else
  begin
    p1_rx_tmp0  = p1_rx;
    p1_rx_tmp1  = p1_rx_tmp0;
    p1_rx_syn  = p1_rx_tmp1;
  end
end
```

图 7 亚稳态消除代码

另外，AD 芯片使用 ad _ busy 信号表示转换状态，FPGA 在每个时钟沿读取该信号的值判断 AD 是否转换完成，该信号的跳变时间由器件的转换时间特性决定，对于 FPGA 而言是异步信号，同样需要考虑亚稳态问题。与图 7 相同，在 FPGA 中使用两级寄存器采样 ad _ busy 信号，得到同步信号 ad _ busy _ syn，消除亚稳态。

5.2 数据位检测

根据应用场景，使用 10 Mbps 的数据传输速率可满足应用需求，FPGA 内部使用 100 MHz 系统时钟，则每个数据位占用 10 个系统时钟。拖缆中，除了本系统的数据传输线缆外，还有其他用于高功率信号传输的线缆，与本系统线缆编制在同一条主电缆内，线缆间可能相互干扰引起的信号电平误动。为了增加传输可靠性，选取数据位传输的 10 个系统时钟中部的 5 个时钟做为数据位检测点，统计 5 个检测点的电平，若高电平占多数，则该信号为逻辑“1”，若低电平占多数，则该信号为逻辑“0”，这种方法避免了单点采样因扰动引起的数据检测错误。图 8 的 RTL 代码实现了 5 个采样点统计和的数据位检测，图中 clk _ cnt 为系统时钟计数器，p2 _ rx _ syn 为时钟域同步后的端口信号。

```
if(clk_cnt > 4'd2 && clk_cnt < 4'd8)
begin
  if(p2_rx_syn)
  begin
    check_cnt <= check_cnt + 4'd1;
  end
end
```

图 8 数据位检测代码

5.3 超时处理

为了防止系统进入异常锁死状态，在等待判断等关键位置处设置超时处理逻辑。例如在 AD 采样流程中，启动 AD 后，代码逻辑一直处于等待状态，直到“ad _ busy”信号为逻辑低，表示 AD 转换完成，代码才会进入下一步执行。在此处设置超时处理逻辑，时间上限为 AD 使用手册标注的转换时间，文本使用的 AD7606 芯片转换时间为 4 μs，超时发生时，将状态机强制转换至 AD 转换完成状态，并设置超时标志位。

5.4 保持总线驱动

使用半双工的工作方式，在设计时，需要注意传输方向切换时的总线状态，在总线无驱动的状态下，总线极易受到外界干扰。为增强总线抗干扰能力，一种方式是硬件层片，在 RS485 芯片端口处设计上拉或下拉电阻，在总线无驱动的状态下保持稳定的电平。另一种方法是代码逻辑设计层面，设计代码使得总线总是处于有驱动的状态。

5.5 仿真验证

代码设计完成后，使用 Modelsim 进行功能验证，也称为前仿真。图 9 为系统代码在 Modelsim 中的仿真波形，图中 m0p1 _ tx 表示主机向节点 1＃发送的信号，p1p2 _ tx 表示节点 1＃向节点 2＃发送的信号。m0p1 _ rx 表示主机接收节点 1＃发送的信号，p1p2 _ rx 表示节点 1＃接收节点 2＃发送的信号，其他信号以此类推。图中可以看出个各节点首先逐级转发主机发出的同步命令，之后各节点逐次在自己的时间片内占用总线向主机发送数据，某节点发送数据时，

（下转第 139 页）