文章编号:1671-4598(2017)08-0255-06 DOI:10.16526/j.cnki.11—4762/tp.2017.08.066 中图分类号:TN91 文献标识码:A

FPGA 状态机在多路异步串口通信处理的实时优势

张建军，安海佳，谭可仕，王 锋，韩怀宇
（中国空间技术研究院，北京 100094）

摘要：首先简要介绍了异步串口板的通常设计方法，并且提出了这些方法的不足之处，重点阐述了基于 FPGA 状态机和片上总线的新设计方案，以及该方案的技术优势，随后公布了基于该方案的异步串口板达到的性能指标；通过比较有关应答延迟的试验数据，提出了基于 FPGA 状态机和基于 DSP 处理器的异步串口板卡存在明显的处理速度差异问题，并基于两种设计方案，解释了形成差异的原因；最后提出了 FPGA 状态机对外部总线存储器或端口的访问管理性能大幅超越了任何一款 DSP 处理器的观点，并对同行提出了类似研发项目的设计建议。

关键词：FPGA；状态机；异步串口；实时

FPGA State—machine's Realtime Advantage at Multiple UART Communication Processing

Zhang Jianjun, An Haijia, Tan Keshi, Wang Feng, Han Huaiyu
(China Academy of Space Technology, Beijing 100094, China)

Abstract: Firstly, usual design methods of asynchronous serial interface board are introduced briefly, and defects of the methods are put forwarded, then the original design method and it's technical superiorities based on FPGA state—machine and bus—on—chip are expounded emphatically, then the performances of asynchronous serial interface board based on above design method are clarified. By comparing test data about time of response delay, obvious difference on processing rate between asynchronous serial interface boards based on FPGA state—machine and DSP processor are presented, and by comparing above two design methods, cause for the difference on processing rate are paraphrased. In the end, the opinion that FPGA state—machine is superior over any DSP processor evidently in performance to manage memorizer and port on exterior bus, and a suggestion of design method for similar equipment or system are put forward.

Keywords: FPGA; state—machine; asynchronous serial interface; realtime

0 引言

某设备的测试平台必须达到 36 路通道、11 种通讯协议、波特率 4/19.2/38.4/57.6/115.2（Kbps）、小于协议分析和数据处理时间小于 1 μs、通讯模式可配置等技术要求，因此测试平台内异步串口板的设计方案要面向上述技术要求而制定。目前异步串口板通常采用嵌入式处理器或者基于 FPGA 的，非总线式的，异步串口收发和处理逻辑。前者易于管理总线，分析协议和处理数据，但处理速度过慢，后者处理速度快，但无法在线配置通讯模式，不能利用总线进行全局管理，故需要一种新的多路异步串口板设计方案，即能够实现管理总线，分析协议和处理数据，又具有强实时性能，以达到测试平台的技术要求。

1 背景技术

目前异步串口板一般采用以下两类方法实现。

采用独立的嵌入式处理器作为数据处理单元。异步串口单元要么使用嵌入式处理器自身的 2 到 3 个异步串口，要么使用连接到 FPGA 片内总线的通用异步收发器或异步收发逻辑，从而建立起一主多从式总线结构。

采用 FPGA 芯片集成了若干独立的异步串口通路，每个通路均有一对处理逻辑和收发逻辑，其中收发逻辑实现了一路串行数据的接受、发送和并串转换，处理逻辑实现了一路串行数据的读取、处理和存储。

上述两类方法存在不足之处。传统的异步串口板，都是采用单片机或嵌入式处理器作为处理单元，通过访问通用异步收发器来读取或者发送数据，数据存储器为一片双口 SDRAM，通过信号灯控制协议，总线上位机和板卡处理单元轮换来访问数据，板卡处理单元直接接受和应答数据。通常情况下，串口应答有响应时间的要求，这使处理单元底层软件的开发难度加大，任何操作都需要处理单元的软件来控制。用嵌入式处理器读取、处理和存储数据，速度慢，特别是无法满足高速、多路、实时接收和应答的要求。高速嵌入式处理器从串口数据帧接收完毕开始，到串行数据帧开始发送为止，即处理延迟至少 10 μs 级。如果高波特率、多串口通道、独立通讯时，就可能出现串口应答滞后或数据帧不完整的问题。

使用 FPGA 芯片集成串口收发和处理逻辑的技术手段，解决了多通路、高通讯速率和高频率下通讯可靠性问题，可以保证各串行通路发送数据帧的完整。但依然存在不足，一是收发逻辑对处理逻辑是一一对应的，而不是基于片内总线实现一主对多从的模式，造成 FPGA 资源的浪费，异步串口板可扩

收稿日期:2017-03-01； **修回日期**:2017-04-24。

作者简介:张建军(1973-)，男，河南省人，硕士，高级工程师，主要从事企业管理方面的工作。

展性差。二是在目前应用中，串口收发逻辑仅实现了串并转换，与通用异步串行收发器相比，功能不足，不具有通用性，如不支持总线的通讯模式设置、工作状态查询、通讯错误类型查询、处理器中断报警和波特率设置等，使得兼容通用异步串行收发器的异步串口板驱动程序不兼容 FPGA 芯片集成串并转换电路的板卡。

2 设计方案

2.1 设计思路

为便于性能比较，分别用 TMS320C6416 DSP 处理器和 FPGA 状态机作处理单元，进行多路通用异步串口管理，协议分析和数据处理，最终实现两块功能相同的异步串口板（两者系统结构、程序流程、通讯协议和软硬件接口均相同），以便性能比较。每块均有 1 个 FPGA 芯片 XC3S2000－4 FG456，1 个 PCI 总线控制器 PCI9052，1 个双口 SDRAM IDT7132。只是前者采用通常的方法，在板卡上多了 TMS320C6416 DSP 处理器，而后者没有 DSP 处理器，在 FPGA 芯片上多了 WB 总线处理逻辑。两者在 FPGA 上均集成了 WB 片上总线，36 个通用异步串行收发器软核、1 个双口 SDRAM 接口逻辑、地址译码器、配置状态寄存器区，以及串口接收滤波逻辑等。下文重点介绍 FPGA 状态机作处理单元的实现方法，性能指标，并对两者的处理速度进行了比较和分析。

2.2 硬件板卡设计

基于 FPGA 状态机和 WISHBONE 片上总线（下文简称为 WB 总线）的 36 通路异步串口板系统架构如图 1 所示。

2.2.1 PCI9052 总线控制器

实现了总线到 9052 LOCAL 总线协议适配，LOCAL 总线配置为 8 位数据线，1 个 2K 字节的 RAM 空间，1 个 IO 空间，1 个高电平有效的中断源。

2.2.2 双口 SDRAM

采用 IDT7132 芯片作为数据缓冲区，容量 2 k * 8 bit，读写周期均为 20 ns。为了避免上位机正在更新某数据区，而与该数据区对应的异步串口要求发送该数据区的情况，在设计中将双口 SDRAM 的存储空间分为两部分，即上位机可写访问的一级缓冲区和 WB 总线处理逻辑可写访问的二级缓冲区。对应 36 通路，一级和二级缓冲区均分成了 72 个子区，每路异步串口对应着一级缓冲区内的一对发送子区和接收子区，以及二级缓冲区内的一对发送子区和接收子区。上位机可读写一级发送子区，但只能读一级接收子区；WB 总线处理逻辑只能读一级发送子区，可读写一级接收子区和二级所有子区。

2.2.3 FPGA 芯片

FPGA 芯片为串口板核心器件，选用 XC3S2000－4 FG456 芯片，支持 200 MHz 时钟，IO 口上升和下降速度均达到 5 ns。

1）WB 片内总线：即 WB 片内总线，是一种应用普遍的、具有灵活性的 IP 核互联接口。

2）异步收发器：异步收发器为 IP 软核，来源于 opencores 开源组织，支持 WB 接口，与通用异步收发器兼容，每个可实现一路全双工异步串口通讯。

3）双口 SDRAM 总线接口逻辑：片内有两个独立的双口 SDRAM 总线接口逻辑，分别实现了双口 SDRAM 与 WB 总线和 LOCAL 总线的逻辑连接。

4）LOCAL 总线寄存器区：为了方便上位机对各异步串口独立灵活配置、全面监控工作状态，必须设置可供上位机访问的若干配置寄存器和状态寄存器，而且 LOCAL 总线寄存器区逻辑是上位机与片内 WB 总线处理逻辑之间的通讯桥梁。

5）地址译码逻辑：片内有两个独立的地址译码逻辑，分别根据 LOCAL 总线地址和 WB 总线地址，片选总线从逻辑。

6）串口接收滤波逻辑：片内 36 个独立的串口接收滤波逻辑，每个对应一路异步收发器的接收端，可滤除正负跳变宽度小于 1 μs 的毛刺，但会造成 1 μs 的信号延迟。

7）WB 总线处理逻辑：实质是 WB 总线状态机，实现了访问 WB 总线、数据处理、数据存储和各串口通讯协议的功能。主要任务是根据上位机设置的 LOCAL 总线配置寄存器区，设置各异步收发器的工作模式，接收串口数据，遵循通讯

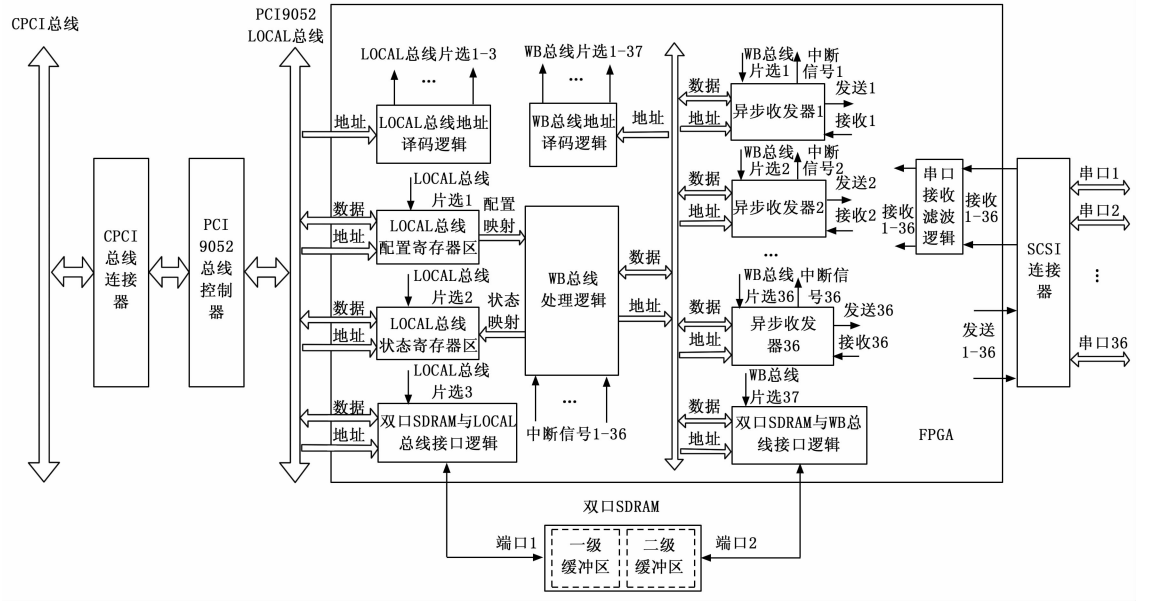

图 1 基于 FPGA 状态机和 WB 片上总线的异步串口板系统架构

协议，发送串口数据，完成一级、二级缓冲区数据更新或复制，并将各串口通道的工作状态标识到 LOCAL 总线状态寄存器区，供上位机查询。

2.2.4　FPGA 资源占用情况

布局布线后，FPGA 资源占用情况如下。

Target Device : xc3s2000

Target Speed : -4

Target Package : fg456

----------Design Summary----

Slice Flip Flops: 13117 out of 40,960 32%

4 input LUTs : 26679 out of 40,960 65%

Occupied Slices : 17264 out of 20,480 84%

Bonded IOBs : 211 out of 333 63%

Total equivalent gate count for design: 373677

2.3　FPGA 程序设计

FPGA 程序主要模块包括，WB 总线处理逻辑、PCI9052 总线控制器与双口 SDRAM 的接口逻辑、WB 片内总线与双口 SDRAM 的接口逻辑、串口接收滤波逻辑、地址译码器和配置状态寄存器区等，其中 WB 总线处理逻辑实现了多路通用异步串口管理、协议分析和数据处理，重点介绍 WB 总线处理逻辑（状态机）的实现方法。因为串口板通道数量多，通信协议繁杂，版面所限，仅介绍其中 1 路通道的编程方法，其它 35 路通道的编程方法相同，也由此逻辑统一管理。

该路通信协议，波特率 115.2 k，每个字包括 1 个起始位，8 个数据位，1 个效验位和 1 个停止位。通信帧包括命令段、数据段和效验段，结构如下表：

通信帧	命令段	数据段	效验段
接收帧(从串口到上位机)	1 个字	8 个字	1 个字
发送帧(从上位机到串口)	1 个字	10 个字	1 个字

WB 总线处理逻辑状态机的处理流程简述如下。如果上位机请求更新一级缓冲区的某子区，上位机查询“FPGA 访问双口 SDRAM 一级缓冲区标志”，若为真，则等待；否则置位“上位机访问双口 SDRAM 一级缓冲区标志”，设置“更新发送区 ID 寄存器”和“发送帧长度寄存器”，更新一级缓冲区的发送子区，完毕后复位“上位机访问双口 SDRAM 一级缓冲区标志”，产生“更新二级缓冲区的中断请求”。WB 总线处理逻辑响应该中断，查询“上位机访问双口 SDRAM 一级缓冲区标志”，若为真，则等待；否则置位“FPGA 访问双口 SDRAM 一级缓冲区标志”，依据“更新发送区 ID 寄存器”内容，读访问一级缓冲区的相应发送子区，并存入二级缓冲区的相应发送子区，完毕后复位“FPGA 访问双口 SDRAM 一级缓冲区标志”。在 WB 总线处理逻辑更新二级缓冲区期间，暂不响应异步串口交易中断。每次 FPGA 读写访问 IDT7132，需 20 ns。在处理任务最重条件下，即 WB 总线处理逻辑在更新二级缓存时，串口恰好要求传送 12 个字节上限的数据，发送只延迟 20 * 12 * 2=480 ns，仍满足小于 1 μs 的试验要求。

如果某异步收发器存在接收或发送任务，即串口通信中断请求存在，WB 总线处理逻辑读访问该异步收发器的中断状态寄存器，如果是发送中断请求，WB 总线处理逻辑从二级缓冲区某发送子区读取发送数据，写入该异步收发器的“发送数据寄存器”中，每发送一个字节，异步收发器均会再请求中断，WB 总线处理逻辑循环响应，直至发送的字节数等于对应的“发送帧长度寄存器”预设值，发送完毕，清相应中断；如果是接收中断请求，读访问接收数据寄存器，并将数据存入二级缓冲区对应接收子区。每接收一个字节后，在“接收帧超时结束门限寄存器”设置的时间内未出现新的接收数据中断，则认为已收到完整数据帧。随后根据接收数据帧的内容，判断是否存在错误，以及何种错误，并标识在对应的“接收帧诊断结果寄存器”和“接收帧长度寄存器”，并设置“二级缓冲区接收子区覆盖一级缓冲区接收子区”的中断请求。如果上位机未访问一级缓冲区，则 WB 总线处理逻辑响应上述中断，执行二级至一级的某接收子区“数据复制”后，清相应中断，置位 LOCAL 总线接收中断请求，请求上位机从一级缓冲区读取数据；如果上位机正在访问一级缓冲区，则等待时机再复制数据。

WB 总线处理逻辑状态机的实现方法如下。

```
module wb_master
(
输入输出端口声明
);
always@(posedge WB 总线复位 or posedge WB 总线时钟)
begin
if (WB 总线复位)
复位"二级缓冲区接收子区覆盖一级缓冲区接收子区"中断请求;
else if(接收帧超时结束门限寄存器设置的时间内未出现新的接收数据中断请求)
认为已收到一个数据帧,设置"二级缓冲区接收子区覆盖一级缓冲区接收子区"中断请求;
else if(二级缓冲区覆盖一级相应子区完毕)
复位"二级缓冲区接收子区覆盖一级缓冲区接收子区"中断请求;
end
always@(posedge WB 总线复位 or posedge WB 总线时钟)
begin
if (WB 总线复位)
接收帧诊断结果寄存器和接收帧长度寄存器复位;
else if(接收帧超时结束门限寄存器设置的时间内未出现新的接收数据中断请求)
begin
随后根据接收数据帧的内容,判断是否存在错误,以及何种错误,并标识在对应的
"接收帧诊断结果寄存器"和"接收帧长度寄存器";
end
end
always@(posedge WB 总线复位 or posedge WB 总线时钟) //WB 总线处理逻辑状态机
begin
if (WB 总线复位)
寄存器初始化;
else
begin
case(WB 总线处理逻辑工作状态)
清 UART 中断寄存器 :
启动 WB 读时序,读 UART 中断寄存器;
等待状态 :
begin
```

```
if(UART 通信中断请求)
WB 总线处理逻辑工作状态 = 读中断状态寄存器;
else if(延迟发送定时器溢出)
begin
复位延迟发送定时器、延迟发送定时器溢出信号;
if(发送帧长度为 0)
   WB 总线处理逻辑工作状态 = 等待状态;
   else
   begin
if(如果接收帧完整正确)
WB 总线处理逻辑工作状态 = 读二级缓冲区发送子区;
else
WB 总线处理逻辑工作状态 = 发送错误提示码;
end
end
else if(更新二级缓冲区中断请求 & ~上位机访问双口 SDRAM 一级缓冲区标志 & 相应串口不正在发送数据)
    begin
     WB 总线处理逻辑工作状态 = 读一级缓冲区的延迟发送时间寄存器;
     置位 FPGA 访问双口 SDRAM 一级缓冲区标志;
    end
    else if(二级缓冲区接收子区覆盖一级缓冲区接收子区中断请求 & ~上位机访问双口 SDRAM 一级缓冲区标志 & 相应串口不正在接收数据)
    begin
WB 总线处理逻辑工作状态 = 读二级缓冲区接收子区数据;
置位 FPGA 访问双口 SDRAM 一级缓冲区标志;
  end
  else
  begin
   WB 总线处理逻辑工作状态 = 等待状态;
   复位 FPGA 访问双口 SDRAM 一级缓冲区标志;
  end
end
读一级缓冲区的延迟发送时间寄存器 :
begin
启动 WB 总线读时序,读一级缓冲区的延迟发送时间寄存器;
WB 总线处理逻辑工作状态 = 写二级缓冲区的延迟发送时间寄存器;
end
写二级缓冲区的延迟发送时间寄存器 :
begin
启动 WB 总线写时序,写二级缓冲区的延迟发送时间寄存器;
WB 总线处理逻辑工作状态 = 读一级缓冲区的更新发送区 ID 寄存器;
end
读一级缓冲区的更新发送区 ID 寄存器 :
begin
启动 WB 总线读时序,读一级缓冲区的更新发送区 ID 寄存器;
根据一级缓冲区的更新发送区 ID 寄存器,确定更新长度寄存器;
WB 总线处理逻辑工作状态 = 写二级缓冲区的更新发送区 ID 寄存器;
End
module wb_master
(
输入输出端口声明
);
always@(posedge WB 总线复位 or posedge WB 总线时钟)
begin
if (WB 总线复位)
复位"二级缓冲区接收子区覆盖一级缓冲区接收子区"中断请求;
else if(接收帧超时结束门限寄存器设置的时间内未出现新的接收数据中断请求)
认为已收到一个数据帧,设置"二级缓冲区接收子区覆盖一级缓冲区接收子区"中断请求;
else if(二级缓冲区覆盖一级相应子区完毕)
复位"二级缓冲区接收子区覆盖一级缓冲区接收子区"中断请求;
end
always@(posedge WB 总线复位 or posedge WB 总线时钟)
begin
if (WB 总线复位)
接收帧诊断结果寄存器和接收帧长度寄存器复位;
else if(接收帧超时结束门限寄存器设置的时间内未出现新的接收数据中断请求)
begin
随后根据接收数据帧的内容,判断是否存在错误,以及何种错误,并标识在对应的
"接收帧诊断结果寄存器"和"接收帧长度寄存器";
end
end
always@(posedge WB 总线复位 or posedge WB 总线时钟) //WB 总线处理逻辑状态机
begin
if (WB 总线复位)
寄存器初始化;
else
begin
case(WB 总线处理逻辑工作状态)
清 UART 中断寄存器 :
启动 WB 读时序,读 UART 中断寄存器;
等待状态 :
begin
if(UART 通信中断请求)
WB 总线处理逻辑工作状态 = 读中断状态寄存器;
else if(延迟发送定时器溢出)
begin
复位延迟发送定时器、延迟发送定时器溢出信号;
if(发送帧长度为 0)
   WB 总线处理逻辑工作状态 = 等待状态;
   else
   begin
if(如果接收帧完整正确)
WB 总线处理逻辑工作状态 = 读二级缓冲区发送子区;
else
WB 总线处理逻辑工作状态 = 发送错误提示码;
end
end
else if(更新二级缓冲区中断请求 & ~上位机访问双口 SDRAM 一级缓冲区标志 & 相应串口不正在发送数据)
    begin
     WB 总线处理逻辑工作状态 = 读一级缓冲区的延迟发送时间寄
```

```
存器；
        置位 FPGA 访问双口 SDRAM 一级缓冲区标志；
      end
      else if(二级缓冲区接收子区覆盖一级缓冲区接收子区中断请求
&. ～上位机访问双口 SDRAM 一级缓冲区标志 &. 相应串口不正在接
收数据)
       begin
    WB 总线处理逻辑工作状态 = 读二级缓冲区接收子区数据；
    置位 FPGA 访问双口 SDRAM 一级缓冲区标志；
      end
      else
      begin
       WB 总线处理逻辑工作状态 = 等待状态；
       复位 FPGA 访问双口 SDRAM 一级缓冲区标志；
      end
    end
    读一级缓冲区的延迟发送时间寄存器 ：
    begin
    启动 WB 总线读时序，读一级缓冲区的延迟发送时间寄存器；
    WB 总线处理逻辑工作状态 = 写二级缓冲区的延迟发送时间寄
存器；
    end
    写二级缓冲区的延迟发送时间寄存器 ：
    begin
    启动 WB 总线写时序，写二级缓冲区的延迟发送时间寄存器；
    WB 总线处理逻辑工作状态 = 读一级缓冲区的更新发送区 ID 寄
存器；
    end
    读一级缓冲区的更新发送区 ID 寄存器 ：
    begin
    启动 WB 总线读时序，读一级缓冲区的更新发送区 ID 寄存器；
    根据一级缓冲区的更新发送区 ID 寄存器，确定更新长度寄存器；
    WB 总线处理逻辑工作状态 = 写二级缓冲区的更新发送区 ID 寄
存器；
    end
    else if(接收字节计数器 < 接收帧长度寄存器)
    WB 总线处理逻辑工作状态 = 写二级缓冲区接收子区；
    else
    WB 总线处理逻辑工作状态 = 等待状态；
    if(接收字节计数器 < 接收帧长度寄存器 - 1)
    效验码累加寄存器 = 效验码累加寄存器 + 接收数据寄存器；
    else if(接收字节计数器 == 接收帧长度寄存器 - 1)
    begin
    效验码累加寄存器 = 效验码累加寄存器 ^ 接收数据寄存器；
    if(效验码累加寄存器 == 0)
    复位效验码错误标识；
    else
    置位效验码错误标识；
    end
    end
    写二级缓冲区接收子区 ：
    begin
    启动 WB 总线写时序，将接收数据写入二级缓冲区接收子区；
    if(接收字节计数器 < 接收帧长度寄存器)
    接收字节计数器 = 接收字节计数器 + 1；
    if(接收字节计数器 < 接收帧长度寄存器)
    WB 总线处理逻辑工作状态 = 等待状态；
    else if(接收字节计数器 == 接收帧长度寄存器)
    WB 总线处理逻辑工作状态 = 读二级缓冲区的发送延迟时间寄
存器；
    end
    读二级缓冲区的发送延迟时间寄存器 ： //读二级缓冲区的延迟
发送时间寄存器，打开延迟发送计时器
    begin
    启动 WB 总线读时序，读二级缓冲区的延迟发送时间寄存器；
    设置延迟发送时间寄存器；
    启动延迟发送定时器；
    WB 总线处理逻辑工作状态 = 等待状态；
    end
    读二级缓冲区发送子区 ：
    begin
    根据命令码寄存器和发送字节计数器，确定发送数据在二级缓冲区
发送子区的地址；
    启动 WB 总线读时序，读二级缓冲区发送子区数据；
      WB 总线处理逻辑工作状态 = 发送二级缓冲区发送子区数据；
    end
    发送二级缓冲区发送子区数据 ：
    begin
    启动 WB 总线写时序，将二级缓冲区发送子区数据写入发送数据寄
存器；
    发送字节计数器 = 发送字节计数器 + 1；
    WB 总线处理逻辑工作状态 = 等待状态；
    end
    发送错误提示码 ： //当接收帧错误时，发送错误提示码
    begin
    启动 WB 总线写时序，将错误提示码写入发送数据寄存器；
    发送字节计数器 = 发送字节计数器 + 1；
    WB 总线处理逻辑工作状态 = 等待状态；
    end
    读二级缓冲区接收子区数据 ：
    begin
    根据 UART 地址和更新字节计数器，确定新数据在二级缓冲区接
收子区的地址；
    启动 WB 总线读时序，读二级缓冲区接收子区新数据；
    WB 总线处理逻辑工作状态 = 写一级缓冲区接收子区数据；
    end
    写一级缓冲区接收子区数据 ：
    begin
    根据 UART 地址和更新字节计数器，确定一级缓冲区接收子区的
地址；
    启动 WB 总线写时序，将新数据写入一级缓冲区接收子区；
    更新字节计数器 = 更新字节计数器 + 1；
    if(更新字节计数器 == 接收帧长度寄存器)
    begin
    WB 总线处理逻辑工作状态 = 等待状态；
    更新字节计数器 = 8'h0；
    end
    else if(更新字节计数器 < 接收帧长度寄存器)
    WB 总线处理逻辑工作状态 = 读二级缓冲区接收子区数据；
    end
    default ：
```

表 2 应答延迟时间统计

波特率(Kbps)	对应通道数量	对应通讯协议序号	数据量(接收帧字数/发送帧字数/每字位数)	固有延迟(μs)(信号滤波延迟+接收帧结束半位延迟)	TMS320C6416 DSP 处理器作处理单元		XC3S2000 FPGA 状态机作处理单元	
					实测延迟(μs)	处理延迟(μs)(实测延迟-固有延迟)	实测延迟(μs)	处理延迟(μs)(实测延迟-固有延迟)
4	2	11	12 / 0 / 11	/	/	/	/	/
19.2	9	4,5,6,10	1 / 5 / 12	1+26=27	37.6	37.6-27=10.6	27.6	27.6-27=0.6
38.4	17	1,2,3,9	12 / 5 / 12	1+13=14	n1; n1∈{25,26}	n2=n1-14; n2∈{11,12}	14.6	14.6-14=0.7
57.6	6	8	6 / 9 / 11	1+8.6=9.6	19.8	19.8-9.6=10.2	10.2	10.2-9.6=0.6
115.2	2	7	10 / 12 / 11	1+4.3=5.3	15.4	15.4-5.3=10.1	5.9	5.9-5.3=0.6
同时测试 36 通道,每通道遵循各自波特率、数据格式和通讯协议				各通道固有延迟同上	各通道固有延迟+n3; n3∈{11,14}	n3; n3∈{11,14}	各通道固有延迟+0.8;	0.8

```
    WB 总线处理逻辑工作状态 = 等待状态;
  endcase
  end
  end
  endmodule
```

2.5 性能指标

3U 的异步串口板实现了 36 通道的全双工通用异步收发器，11 种通讯协议，每种通讯协议对应 1 种波特率、1 种数据格式和若干通道。通讯周期 200 ms，每通道通讯间隔 5 ms；每路串口通讯最高速率不低于 2 Mbps；每路串口通道 12 个字节的发送缓冲区和 12 个字节的接收缓冲区；处理时间小于 1 μs。上位机可分别配置各通道波特率、数据格式、通讯协议，并查询工作状态，完全满足某重要设备测试平台的试验要求。

3 试验数据分析

在联机试验中，测试平台先后使用了由 TMS320C6416 DSP 处理器和 XC3S2000 FPGA 状态机作处理单元的异步串口板，分组测试了相同波特率的所有通道，及全部通道。试验重点考察数据之一是应答延迟时间，即接收帧停止位和发送帧起始位之间的延迟，其由两部分组成，分别是固有延迟和处理延迟。其中固有延迟由为滤波而设置的 1 μs 滤波延迟和为判断接收帧结束而设置的半位延迟组成，半位延迟与波特率成反比。处理延迟由处理单元访问总线端口、数据处理和实现通讯协议所耗时间组成，DSP 处理器和 FPGA 状态机作处理单元时，处理延迟有明显差别，实测应答延迟时间统计如表 2 所示。

从统计表得出如下结论：

1）无论采用何种处理单元，处理延迟与波特率和通讯数据量无关。因为上位机已将数据写入各通道的数据发送子区，依据接收帧命令参数，DSP 处理器和 FPGA 状态机只需进行简单逻辑和算法运算即可获得发送数据帧，对于 4 — 115.2 kbps 之间的波特率和 12 字节的数据量，有充分时间裕量，不会出现处理瓶颈。

2）TMS320C6416 DSP 处理器作处理单元时，处理延迟及其变化范围较大，而且通道数越多，处理延迟及其变化范围越大，反之则越小。因为 DSP 处理器 IO 端口数量有限，当串口通道数量较多时，部分重要信号只能通过输入输出端口成组传送，所以 DSP 处理器不仅要通过片上总线访问异步收发器和 SDRAM，还要通过片上总线访问相当数量的输入输出端口，以便和上位机、FPGA 交换信号。为了及时传输这些信号，DSP 处理器还需要以一定的频率巡检这些信号。另外 DSP 处理器所有处理过程都是顺序执行的，处理时间与程序语句数量成正比关系。

3）XC3S2000 FPGA 状态机作处理单元时，处理延迟及其变化范围很小，而且与通道数量无关。因为 FPGA 集成片上系统后，其状态机与其它片上逻辑之间的信号传输通过片内布线完成，而布线资源几乎不受限，不仅简化了处理单元的处理任务，而且保证了信号的实时传输，不必巡检端口。另外 FPGA 状态机不仅能够通过状态转移完成时序功能，而且能够通过并行处理完成算法功能，所以全部处理时间基本由访问总线端口的程序语句数量决定，与算法和通道数量几乎无关。

4）FPGA 状态机对外部总线或端口的访问管理性能大幅超越了 TMS320C6416 DSP 处理器，片上逻辑之间的信号实时传输能力也大幅超越了后者。

4 结论和建议

基于 FPGA 状态机和片上总线的多通道异步串口板，性能大幅超越了基于 DSP 处理器的串口板。这表明在算法相对简单、功能单一的条件下，FPGA 状态机对外部总线存储器或端口的访问管理性能大幅超越了任何一款 DSP 处理器，另外 FPGA 状态机的信号实时传输能力也大幅超越了 DSP 处理器。虽然开发周期相对较长，但一旦定型，性能突出，可靠性高。而高性能 DSP 处理器仅在内存中运行程序时速度快，但在管理外部总线存储器或端口时，其优势无法发挥。

建议同行在研制“具有高时序性能的、多外设接口的、功能单一的设备”时，参考上述基于 FPGA 状态机和片上总线的设计方案。

参考文献：

[1] Xilinx，Inc. Spartan — 3 FPGA Family：Complete Datasheet [M]. 2004.

[2] Opencores. WISHBONE System—on—Chip (SoC) Interconnection Architecture for Portable IP Cores [M]. Revision：B. 3，September 7，2002.

[3] Jacob Gorban. UART IP Core [M]. Revision：0. 6 August 11，2002.

[4] 孙进平，王 俊，李 伟，等. DSP/FPGA 嵌入式实时处理技术及应用 [M]. 北京：航空航天大学出版社，2011.