《UART 的 FPGA 实现》

--“MCU 外设设计”开源项目

项名	内容	备注
题目	UART 的 FPGA 实现	
创建者	Adeko	创造力电子开发网主办者
参与者	Adeko	文档创建者
	期待您的加入……	您的主要贡献……
当前版本	V1.00	
	本版本实现了基本的 UART 通信功能，模仿了 C51 和 AVR 单片机 UART 外设了大部分功能，通过外部总线，可以像使用内部 UART 外设一样地使用本版本的 FPGA 设计的 UART	没有实现多机通信功能
历史版本		
概述	本项目的初级目标是 UART 通信功能的实现，最终目标为大家集合各厂家 UART 外设的特点， 再发挥自己的创造力，设计一个功能强大的 UART 设备	
维护和声明	创造力电子开发网承担项目的维护和控制等责任，网址：www.edaok.net；创造力电子开发网具有和保留整个项目资料（包括且不仅包括项目的源代码和文档）的所有发布权	

★ **特别注意：**

当前文档于 2009 年 5 月中旬发布，可能已有更新，请访问创造力电子开发网检查更新。

目 录

第1章 UART 相关背景

1.1 知识背景

1.1.1 UART 概述

UART 的全称是通用异步收发器（Universal Asynchronous Receiver/Transmitter），是实现设备之间低速数据通信的标准协议。“异步”指不需要额外的时钟线进行数据的同步传输，是一种串行总线接口，只需占用两根线就可以完成数据的收发（一根接收数据，一根发送数据），常用的标准通信波特率有 9600bps、115200bps 等。

1.1.2 帧格式

UART一帧由起始位、数据位、校验位和停止位组成。数据逐位传输，示意图如图 1.1所示。

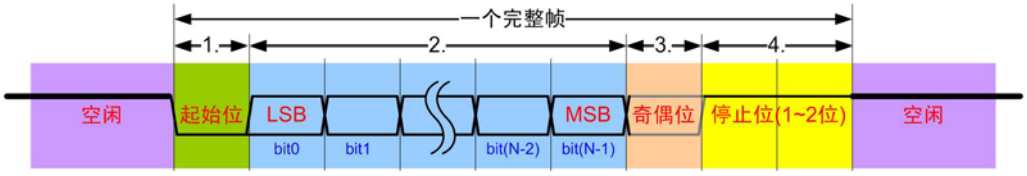

图 1.1 UART 帧格式示意图

1. 起始位

UART 空闲时（没有数据传输），总线为高电平（逻辑 1），当需要数据传输时，首先发送一个“起始位”，起始位为一个低电平“逻辑 0”。

- 为什么需要起始位？

因为 UART 没有控制线，要让接收方知道什么时候开始接收数据，需要一些手段，在 UART，数据的传输只有一根线，所以在发送数据之前，先发一位逻辑“0”作为数据发送的起始标志，接收方在空闲时，当检测到有一个低电平，则开始接逐位接收数据。

2. 数据位

如图 1.1的“2.”所示，紧挨着“起始位”的是数据位，它可以是 5、6、7 或 8 位，收/发双方在数据开始传输前，需要对双方数据位位数作一致的定义，否则会导致数据的传输错误；数据位的发送采用低位（LSB）先发送。

- 为什么数据位可变？

因为 UART 是一种低速总线，每多发一位都占用不少的时间（由传输波特波决定），所以可以传输数据的特点，采用不同位数据的波特率以节约时间。如在 Modbus 协议中，Modbus-ASCII 是以 ASCII 码传输的，因为 ASCII 是不会大于 0x7F，所以使用 7 位的数据位，而 Modbus-RTU 则采用 8 位的数据位。

注：Modbus是一种工业现场常用的通信协议，基于UART总线（常用RS-485）的Modbus协议有Modbus-ASCII和Modbus-RTU。关于Modbus的更多，请参考www.modbus.org。

3. 校验位

UART的校验位紧挨着数据位，采用奇/偶位校验方式，可有可无，是为了为了验证数据传输的安全性而设置的，在收/发双方进行数据传输前要预设好是否需要校验位，如果需要则是奇校验还是偶校验。如图 1.1的“3.”所示。

- 为什么需要校验?

UART 长距离数据传输时，使用 RS-232、RS-485 传输，一般使用较长的导线连接收/发设备，其工作现场因存在不同程序的干扰，可能会导致数据的传输错误，加个校验位可作初步的校验，如果校验出错，说明数据是不可靠的，直接丢弃；如果校验通过，则数据有一定的可靠性，可进入下一级的校验，通常加入通信协议，如 Modbus-RTU 引入了 CRC 校验。

- 什么是奇偶校验？

奇 / 偶校验是对数据进行逐位同或 / 异或运算。如公式（1）和公式（2）所示。

$$D_{EVEN} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \quad \text{公式（1）}$$

$$D_{ODD} = \sim D_{EVEN} \quad \text{公式（2）}$$

如对 8 位数据 0x55（01010101b）作偶校验操作，得到的结果为“0”；作奇校验得到的结果为“1”。简单而言，偶校验增加一位“0”或“1”，使数据位加上校验位后“1”的个数为偶数；而奇校验则是使“1”的个数为奇数。

如图 1.2所示为奇偶校验RTL（寄存器传输级）示意图。使用移位寄存器进行逐位操作。当“LOAD”为逻辑“1”时，加载移位寄存器的数据和初始化校验结果寄存器为逻辑“0”；之后在“CLK”的每个上升沿进行逐位“异或”运算。

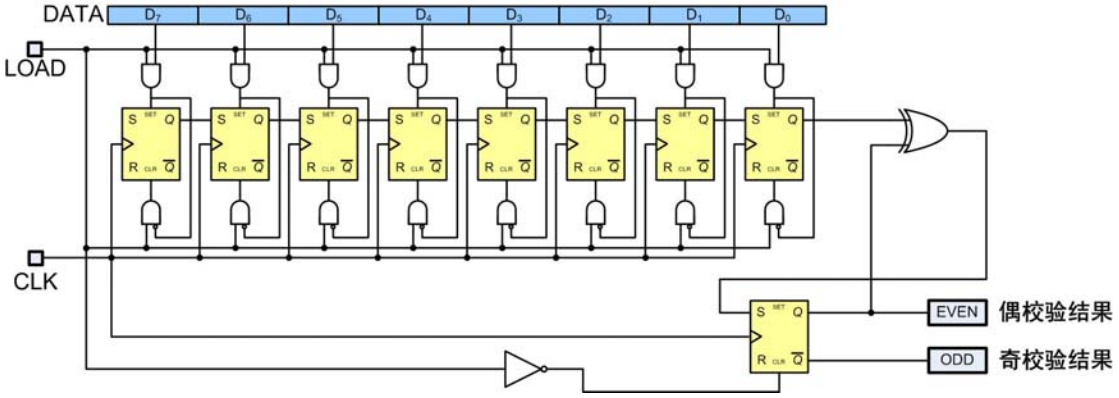

图 1.2 奇偶校验 RTL 示意图

4. 停止位

UART 的帧以停止位作为停止标志，是在数据位（没有校验位）和校验位（有校验位）之后发送 1~2 位的逻辑“1”。停止为可以为 1 位、1.5 位和 2 位。当发送完停止位之后，UART 总线进入空闲。

5. 空闲

空闲指 UART 总线上没有数据进行传输，表现为发送方输出逻辑“1”，在空闲时，接收方时刻监视 UART 总线上电平变化，当发现起始化，则进入数据接收状态，直至接收完一帧数据，如果最后没有检测到停止位，则标志帧错误。

6. 波特率（Baudrate）

由于 UART 没有同步时钟线，收/发双方如果需要进行正确的数据传输，则要在收/发双方定义一致的位时钟，位时钟可以理解为 UART 总线一个位所占用的时间，即“波特率”。在定义上，收/发双方的波特率可以是随意的，只需要保持一致，如双方都是 1000bps，但是，这不能兼容现有常用的设备，兼容性差。所以在工程应用中，常用一些特定的波特率真，如 4800bps、9600bps 或 115200bps 等。

- 如果波特率是 9600bps，则传输速度最大是多少?

波特率的单位是 bps（位每秒），是指发送一位所占用的时间。如果 UART 定义 8 位数数位、无校验、1 位停止位。则一帧数据的位数 N 如公式（3）所示：

N = 1 位起始位+8 位数据位+0 位校验位+1 位停止位=10bits 公式（3）

所以最大数据传输率为：9600bps / N = 960BPS（字节每秒）。

即当波特率为 9600rps，UART 帧格式为 8 位数据位、无校验、1 位停止位时，最大的数据传输率为 0.96KBPS。

1.1.3 示例

因为 UART 的帧格式是可变，以下以几个示例，形象地描述 UART 的不同设置下的帧格式。

1. 常用帧格式

如图 1.3所示为 8 位数据位、无校验位、1 位停止位的帧格式示意图，一帧共有 10 位。发送的数据为 0xA5。

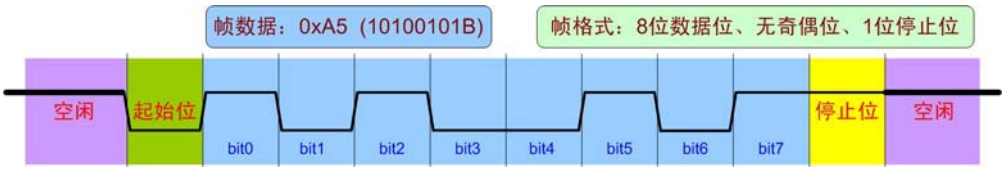

图 1.3 常用帧格式

2. 偶校验

如图 1.4所示为 8 位数据位、偶校验、1 位停止位的帧格式示意图。每帧共有 11 位。发送的数据为 0xA5，则偶校验结果为“0”。当加个校验位后，可以看出，“1”的个数为偶数。

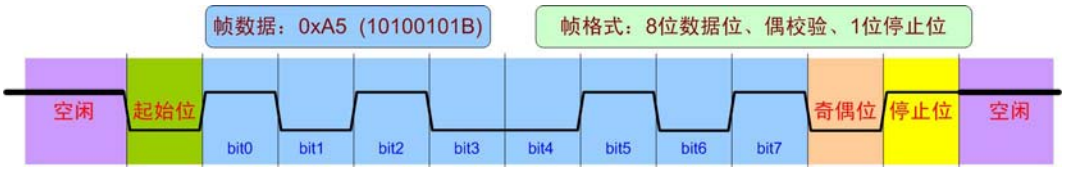

图 1.4 带偶校验帧格式

3. 奇校验

如图 1.5所示为 8 位数据位、奇校验、1 位停止位的帧格式示意图。每帧共有 11 位。发送的数据为 0xA5，则奇校验结果为“1”。当加个校验位后，可以看出，“1”的个数为奇数。

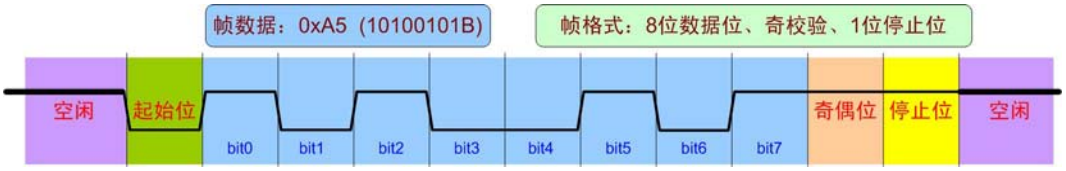

图 1.5 带奇校验帧格式

1.1.4 物理接口

UART 的物理接口可以是 RS-232、RS-485 和 IrDA 红外线等。其中 RS-232 是最为大家容易接触的，在 2006 年之前是大部分 PC 机的标准接口，早期电话线上网就是利用 RS-232 连接数 PC 机和 Modem。

1. RS-232 连接器

RS-232 的连接器常用的是DB-9，其连接器示意图如图 1.6所示，为九针连接器。对于常用的简单应用，使用到的有三根线：RxD、TxD和GND。

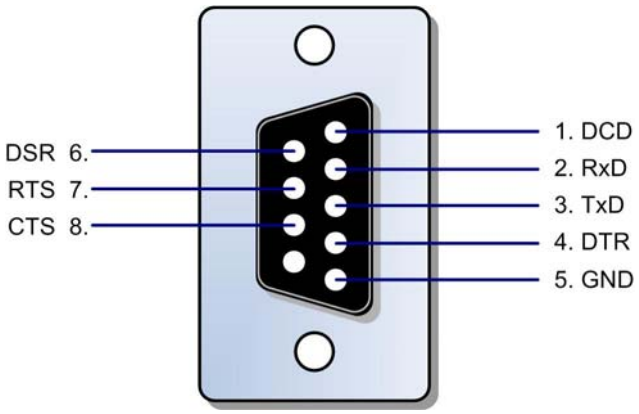

图 1.6 DB-9 连接器示意图

实物图如图 1.7所示。

图 1.7 DB-9 实物图

2. RS-232 电气特性

RS-232 规定说明了 RS-323C 标准对逻辑电平的定义。对于数据（信息码）：逻辑“1”（传号）的电平低于-3V，逻辑“0”（空号）的电平高于+3V；对于控制信号；接通状态（ON）即信号有效的电平高于+3V，断开状态(OFF)即信号无效的电平低于-3V，也就是当传输电平的绝对值大于 3V 时，电路可以有效地检查出来，介于-3～+3V 之间的电压无意义，低于-15V 或高于+15V 的电压也认为无意义，因此，实际工作时，应保证电平在±(3～15)V 之间。

在 TxD 和 RxD 上：

- 逻辑“1”为-3V～-15V；
- 逻辑“0”为+3～＋15V。

在 RTS、CTS、DSR、DTR 和 DCD 等控制线上：

- 信号有效（接通，ON 状态，正电压）＝ +3V～+15V ；
- 信号无效（断开，OFF 状态，负电压) ＝ -3V～-15V 。

3. RS-232 电平转换电路

由于 RS-232 的电气特性和 MCU 等输出的电气特性不一致，所以对于 TTL 电平的 MCU，使用 RS-232 连接器（如和 PC 机通信），需要使用电平转换电路，通常使用集成电路(IC)完成电平转换，常用的 IC 有 MAX232 等，当然还有其它公司生产的 IC，如 SP232 等，常以 232 作为标识。

如图 1.8所示，是使用MAX232 作为转换IC的电路图。

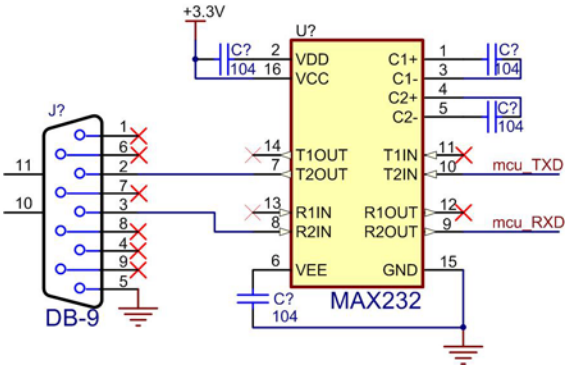

图 1.8 电平转换电路

4. 实际应用

- RS-232

RS-232 在实际应用一般用于点对点的数据传输，当然在硬件和软件上加上一些手段，可以用于多机通信，但应用不多。在通信速率低于 20kb/s 时，RS-232C 所直接连接的最大物理距离为 15m。

- RS-485

RS-485 在工业应用现场应用比较广泛，常使用单工通信组成一主多从的通信网络，使用双绞线连接多台设备，使用差分信号传输数据，所以抗共模干扰能力比 RS-232 强，共通信距离可以超过 1Km。

1.2 小结

UART 是现在 MCU 常用外设，它具有应用简单（只有几个寄存器），应用广泛，接线简单等优点，对电平进行转换后，通信距离较长。

UART 由于其简单易用，常用是调试嵌入式系统的好工具，可以方便地把程序的运行状态和信息（如 Lunux 的引导信息）实时地打印到 PC 机的屏幕（如使用 Windos 系统自带的超级终端。打开方法参考为“开始”→“程序”→“附件”→“通迅” →“超级终端”），以方便掌握程序的运行。出于这个原因，以至很多性能较高的 MCU 增加了一个专门用于调试程序的 UART，使用起来更方便，更节约 CPU 资源，如一部分 ARM7 芯片和 ARM9 芯片。

UART 工作于 OSI 模型的数据链路层，实际应用中，可以增加一层应用层，方便设备间的通信。Modbus 协议是应用于电子控制器上的一种通用语言，其物理层一般使用 RS-485，使用 UART 发收数据。

预告：日后，基于Mobus协议的广泛应用，创造力电子开发网将会创建一个应用FPGA实现Modbus协议的开源项目，期待大家对这个项目的关注和支持，如果您有任何的见解、建议或疑问，欢迎访问我们的网站（http://www.edaok.net）或联系E-mail：edaok@vip.163.com，谢谢。

1.3 下一章概述

下一章，将会介绍 UART 的设计过程，提供设计的 Verilog HDL 源代码，Testbench 仿真测试源代码，单片机测试源代码，使用 Altera 的 Cyclone 系列 FPGA 及 Quartus 设计软件，仿真测试使用专业的仿真软件 ModelSim……

设计的关注点：

EBI（外部总线接口）、同步设计、状态机设计、分层设计等。

第2章 UART 的 FPGA 实现

2.1 概况

本章意图介绍 UART 的 FPGA 的实现过程，除了实现了 UART 基本的收/发能力，作为单片机的外设，综合(synthesis)的外部总路接搭到单片机的外部总线，完全作为 MCU 的一个 UART 外设使用。

如图 2.1所示为本设计的系统总框图，FPGA引出的EBI总线接到单片机的外部总线上，共享系统时钟（CLK）和复位（nRST）。从宏观上看，FPGA综合出的UART完全作为单片机的外设，提供中断输出。

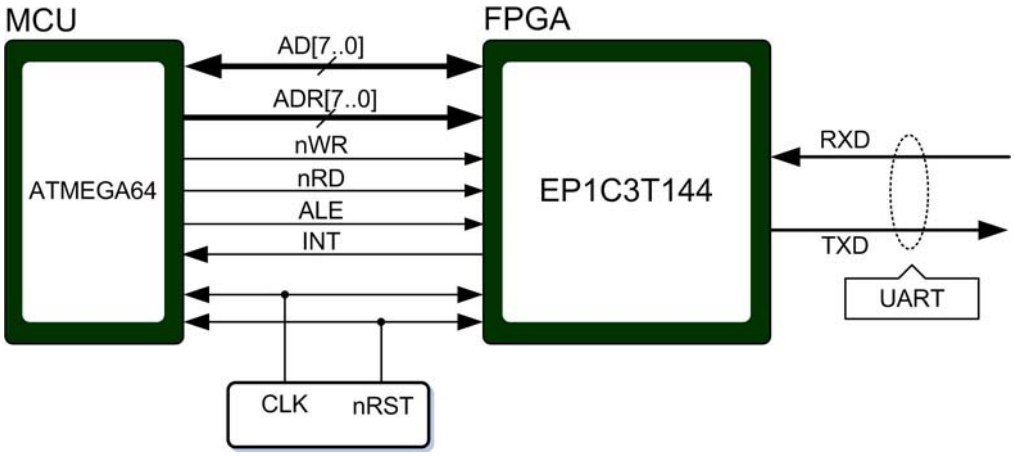

图 2.1 系统总框图

2.1.1 已实现功能

当前 UART 外设设计开源项目实现了了以下功能：

- UART 收/发器；
- 支持 5-8 位数据位、1-2 位停止位、1 位奇偶校验位；
- 帧格式错误、奇偶校验结果和过速（Overrun）错误检测能力；
- 波特率（Buarrate）可调；
- 支持接收数据完成、发送数据完成中断，中断可屏蔽；
- 1 级数据缓冲；
- 外部总路设计，数字、操作寄存器影射。

以上为创造力开源项目——MCU 外设设计的 UART 的 FPGA 实现的完成状态，提供完整的工程文件（VerilogHDL 硬件源码、VerilogHDL 仿真激励输入文件、MCU 测试 IAR 工程）。

2.1.2 工作环境参考

如表 2.1所示为工作环境参考表，硬件测试使用Altera的FPGA和Atmel的ATMEGA64，由创造力电子开发网（http://www.edaok.net）制作的“外设设计测试验证板——F1C3M64”可以快速完成该设计的硬件工作配置。

关于本设计使用的工作软件只供参考，由于已分享出设计所有源代码，你可以使用自己钟爱的软件验证、改进设计。当然，如果你想快速地使用本设计，你可以参考表 2.1配置你的工作环境。

本文档内容以表 2.1所示的工作硬件、软件环境展开编写。

表 2.1 工作环境参考表

软件工具	作用	备注
Quartus II V7.2	FPGA 综合、下载等	EP1C3T144C8 FPGA
IAR 5.11 For AVR	编译、链接、调试单片机测试程序	ATMEGA64L-8AU 单片机
ModelSim 6.2b	仿真 UART 硬件源码	
SSCOM3.2(丁丁)	硬件测试 UART，通过串口收发数据	串口调试助手
gVim	编辑源代码(*.c、*.v、……)	非常好用的代码编辑器
F1C3M64 V1.00c	硬件测试 UART，内嵌以总线方式连接的单片机和 FPGA、基本的测试器件和接插件	外设设计测试验证板

注：关于以上软件的使用技巧等方面欢迎访问“创造力电子开发网”学习或分享。

2.2 快速使用

“快速使用”本意是让大家通过验证本设计，快速溶入本设计的思路，改进、升级设计。

“快速使用”的工作软件环境配置请参考表 2.1。硬件可以使用由“创造力电子开发网”制作的“*外设设计测试验证板——F1C3M64*”，也可以自己搭电路或改装现有线路板。

步骤 1：

打开FPGA的工程目录（./uart/fpga/v0p10），如图 2.2所示；使用Quartus II V7.2 打开本目录下的uart.qpf工作文件，本工程的已配置为“外设设计测试验证板－F1C3M64”，即目标芯片为“EP1C3T144C8”。

关于Quartus II软件的使用、设置等可以访问创造力电子开发网（http://www.edaok.net）。

注：如果自选设置或重新设置工程参数，一定要为使能综合器的“安全状态机（Safe State Machine）”为有效“ON”状态，否则硬件测试可能会失败。

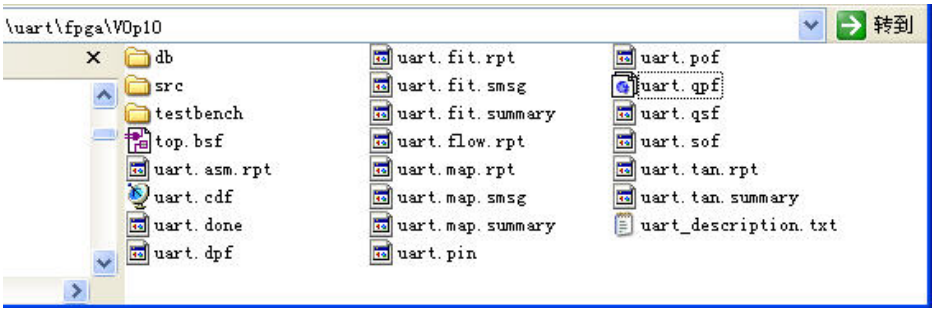

图 2.2 FPGA 工程目录

步骤 2：

打开MCU的工程目录（./uart/mcu/），如图 2.3所示。使用IAR 5.11 For AVR打开本本目录下的UartTest.eww工程文件。本工程的调试器设置为“JTAGICE”，你可以使用你钟爱的调试器，甚至使用其它调试工具（如AVRStatio）。

关于 IAR 软件的使用、设置等可以访问创造力电子开发网（http://www.edaok.net）。

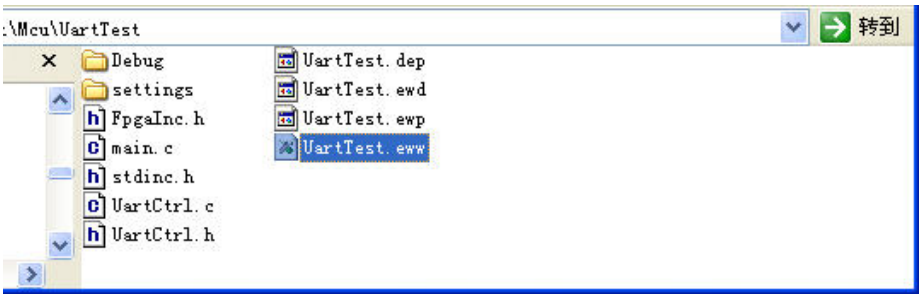

图 2.3 MCU 工程目录

步骤 3：

连接硬件，如图 2.4所示，“外设设计测试验证板－F1C3M64”本身已完成了MCU和FPGA的连接等，只需要如图所示连接单片机调试器、FPGA的JTAG下载器和RS-232 串口线。

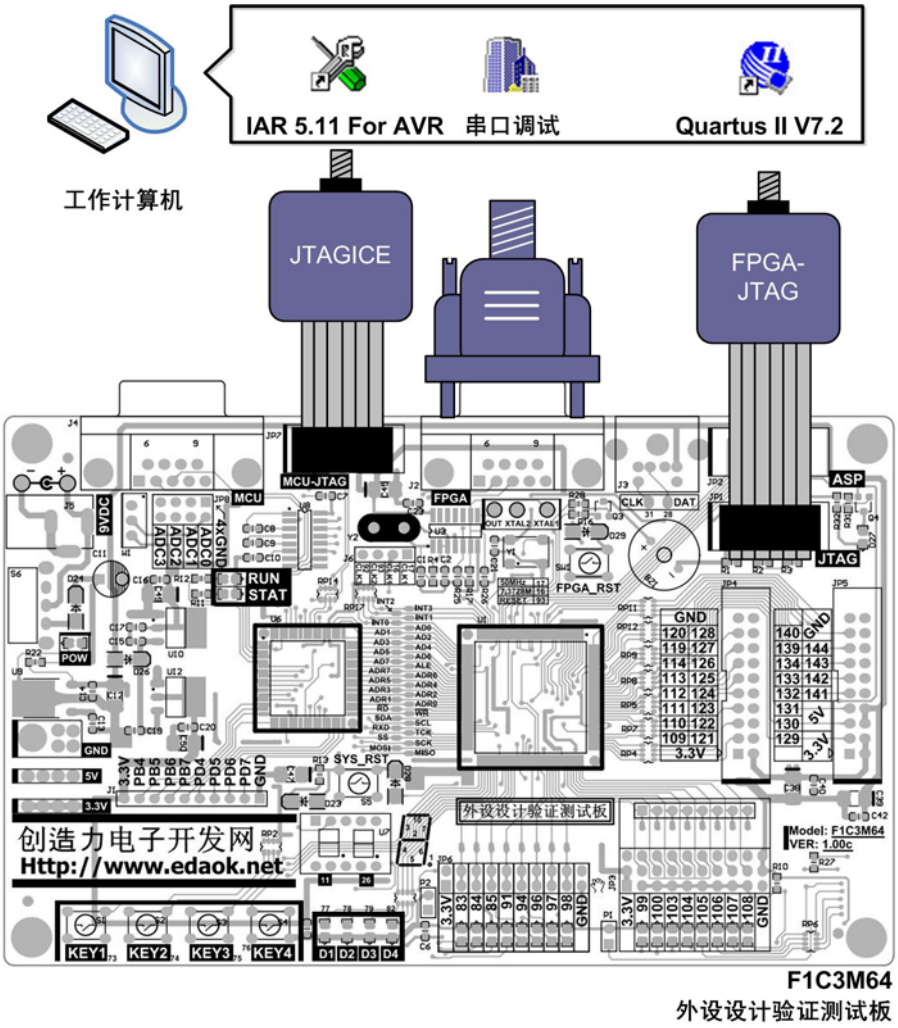

图 2.4 使用 F1C3M64 验证板的 UART 测试连接图

完成以上的工作后，下载单片机程序和配置 FPGA，打开串口调试助手，就可以验证本设计目前的成果，同时也搭过好了工作环境，可以修改源代码，改进和升级 UART 的设计。

注：如果对“快速使用”有疑问、建议等，欢迎通过 E-mail：support.edaok@163.com 联系我或访问创造力电子开发网（http://www.edaok.net），谢谢！

2.3 测试环境

设计包含了仿真测试和硬件测试。

- 仿真测试

仿真测试软件使用 ModelSim 专业的仿真测试软件。激励输入文件应用 VerilogHDL 编写，模仿单片机的读写时序。

- 硬件

硬件测试借助单片机，FPGA 配置的 UART 作为外设，用 PC 机通过 RS-232 不断发数据，当单片机接收到数据就往 PC 机发送刚接收到的数据。

2.3.1 ModelSim 仿真测试

如图 2.5所示为使用ModelSim软件的仿真结果图，激励文件模拟了外部总线的行为，控制使用FPGA实现的UART外设，

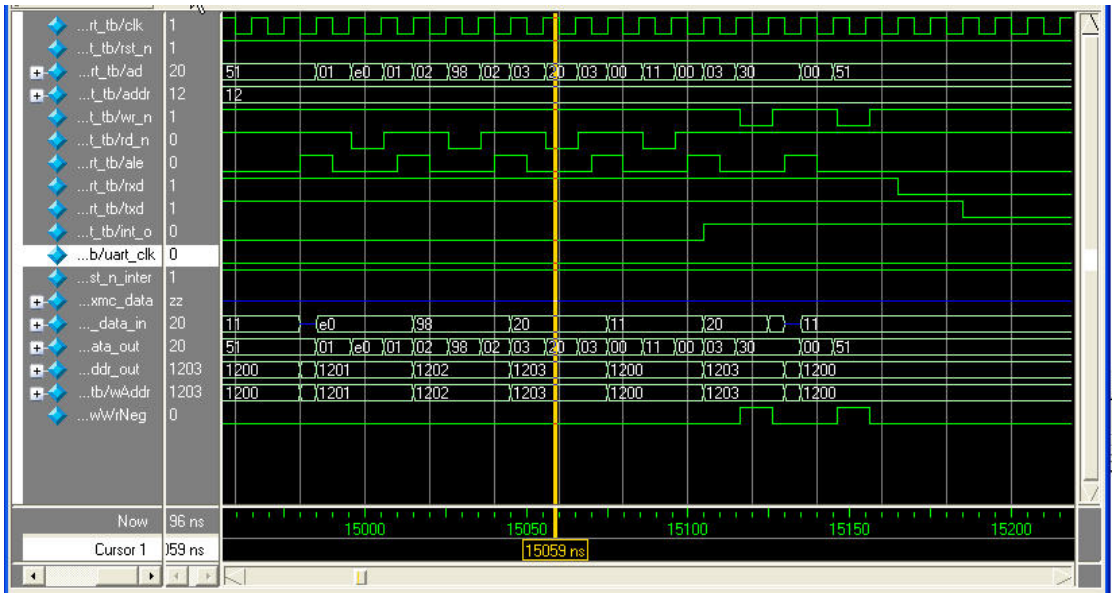

图 2.5 ModelSim 仿真截图

如程序清单 2.1所示是使用Verilog HDL编写的EBI（外部总线接口）的时序行为描述代码，模拟了单片机读写外部总线的时序。

程序清单 2.1 仿真文件的 EBI 读写时序模拟

```
/*
 *  根据设定的地址, 读取 EBI 的数据, 模拟 C51 外部总线的时序, 低 8 位的地址和数据线
 *  复用, 使用 ALE 外 Address Lock Enable
 */
task ebiReadDat;
input   [15:0]  a;
output  [7:0]   return;

reg     [7:0]   return;
begin
```

```
   @(negedge clk)
   ale    = 1'b1;
   @(posedge clk)
   rAD    = a[7:0];
   addr   = a[15:8];
   @(negedge clk)
   ale    = 1'b0;
   @(posedge clk)
   rd_n   = #1 1'b0;

   @(posedge clk)
   return  = wAD;
   rd_n   = #1 1'b1;
end
endtask


/*
 *  往EBI(Extenl Bus Interface)写数据, 模拟C51的外部总线的时序, 输入为地址和数
 *  据
 */
task ebiWriteDat;
input   [15:0]     addr_i;
input   [7:0]      data_i;

begin
   @(negedge clk)
   ale    = 1'b1;

   @(posedge clk)
   rAD    = addr_i[7:0];
   addr   = addr_i[15:8];

   @(negedge clk)
   ale    = 1'b0;

   @(posedge clk)
   rAD    = data_i;
   wr_n   = #1 1'b0;

   @(posedge clk)
   wr_n   = #1 1'b1;

end
```

```
endtask
```

注：整个 testbench 文件请打 s 开 FPGA 工程目录下的 testbench 文件夹下的 top_tb.v 文件，有关仿真软件 ModelSim 的使用请访问创造力电子开发网（http://www.edaok.net）。

2.3.2 单片机硬件测试

本设计不会只停留在软件仿真上，而是全部设计通过硬件测试。

单片机选用国内应用广泛的 ARV 单片机——ATMEGA64L-8AU，该款单片机使用 8 位 AVR 内核，开放外部总线，开发过程清晰明了，即使未用过该系列单片机的开发环境，通过短时间的学习，也可以快速应用。

PC 机使用串口调试软件不断地发送数据，当 FPGA 配置的 UART 从 RXD 线上接收数据，将通过中断方式通知单片机，单片机在中断服务程序中往 PC 机发送刚接收到的数据，最终从串口调试软件得到的结果是收/发个数应该是一至的（设计没有错误的前提下）。通过这样完成硬件测试。

如图 2.6所示为硬件测试结果，可以看出收/发数据个数一致。

图 2.6 PC 测试过程

如程序清单 2.2所示为FPGA配置的UART的寄存器定义，本设计的UART寄存器和 ATMEGA64 内置UART寄存器定义兼容，所以可以设计编绎开关FPGA_UART_EN选择使用哪个UART，如程序清单 2.2的（1）所示。

所以，想快速使用本设计的 UART，可以参考 ATMEGA64 数据手册的 UART 寄存器描述。

注：本设计 UART 的中断输出是低电平有效，默认配置下使用了 ATMEGA64 的“INT4”外部中断线，所以使用本设计的 UART 的中断时，需要使能单片机的外部中断“INT4”，并配置为低有效触发。

程序清单 2.2 单片机 UART 配置定义

```
/*
 *  定义 FPGA 配置的 UART 寄存器地址映射
 */
#define FPGA_UDR   (*(volatile unsigned char *)0x1200)
#define FPGA_UCSRA  (*(volatile unsigned char *)0x1201)
#define FPGA_UCSRB  (*(volatile unsigned char *)0x1202)
#define FPGA_UCSRC  (*(volatile unsigned char *)0x1203)
#define FPGA_UBRRH  (*(volatile unsigned char *)0x1204)
#define FPGA_UBRRL  (*(volatile unsigned char *)0x1205)


#define FPGA_UART_EN   1                                                    (1)

/*
 *  根据代码配置, 选择 FPGA 配置的 UART 或单片机内部 UART
 */
#ifndef FPGA_UART_EN
#error Have not define the macro FPGA_UART_EN
#endif
#if FPGA_UART_EN > 0
#define X_UDR       FPGA_UDR
#define X_UCSRA     FPGA_UCSRA
#define X_UCSRB     FPGA_UCSRB
#define X_UCSRC     FPGA_UCSRC
#define X_UBRRH     FPGA_UBRRH
#define X_UBRRL     FPGA_UBRRL
#else
#define X_UDR       UDR0
#define X_UCSRA     UCSR0A
#define X_UCSRB     UCSR0B
#define X_UCSRC     UCSR0C
#define X_UBRRH     UBRR0H
#define X_UBRRL     UBRR0L
#endif
```

2.4 FPGA 设计

2.4.1 顶层图

设计采用经典的自上以下的分层次结构，如图 2.7所示为UART的顶层框图，对于熟悉AVR单片机的朋友，可能很快地从图看出，框图的寄存器定义很像AVR单片机内置UART的寄存器定义。的确，本设计是仿照AVR单片机UART而设计，如前面所述，通过单片机的外部总线（EBI）接口，控制本设计配置的UART和控制AVR单片机内置的UART是没有太大差别的。

所以，想更了解本设计的寄存器定义，可以参考 AVR 单片机的数据手册。

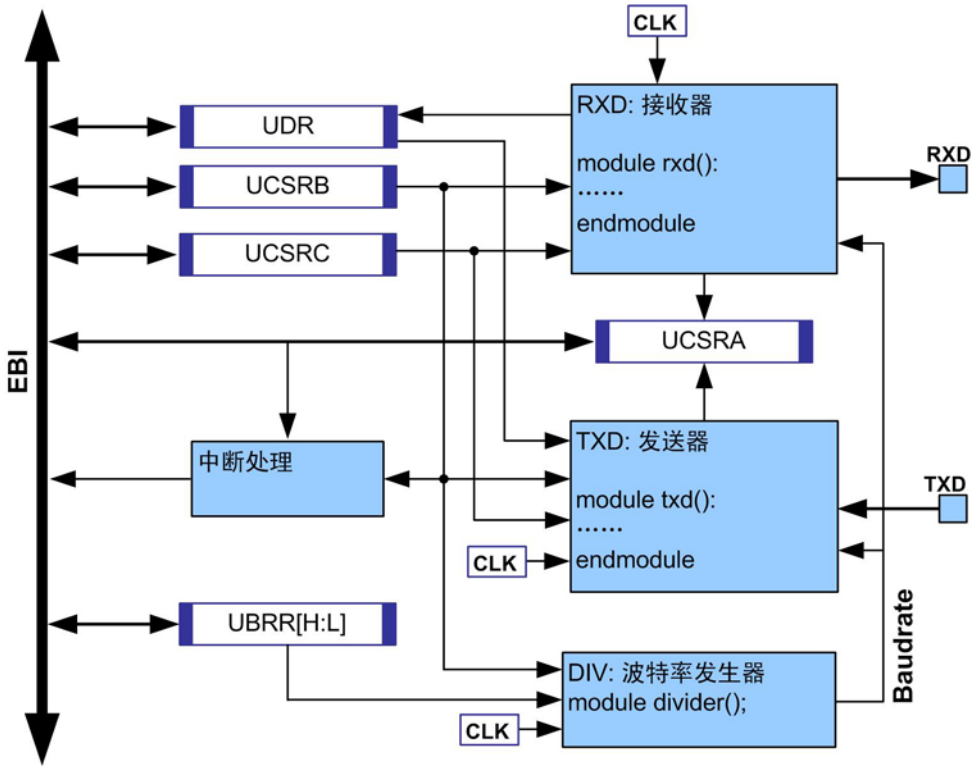

图 2.7 UART 顶层框图

由图 2.7的顶层框图获知，本设计的UART分为以下三个模块（module）：

- 波特率发生器——divider

这个模块用于为 UART 的收/发器提供时钟，这个时钟为 UART 的波特率的 16 倍频，即如果波特率要求为 9600bps 时，该模块输出的时钟为 9600Hz×16=153600Hz。输出时钟需要可配置（UBRRH、UBRRL）。

- 接收器——rxd

接收器时刻监视 UART 总线的 RXD 线的电平，当检测到起始位，接收器启动接收状态机，根据寄存器（UCSRB、UCSRC）的设定，解析 RXD 线的电平，当完成一个字节接收，输出接收到的数据和线状态。

接收器的设计主要是围绕“有限状态机”而进行。

- 发送器——txd

发送器是监视 EBI 总线，当检测需要发送一个字节时，发送器启动发送状态机，同样时，发送器会根据寄存器（UCSRB、UCSRC）的设定，逐位往 UART 总线的 TXD 线发送数据。当发送完成一个字节，发送器输出发送状态。

和接收器相似地，发送器的设计也是围绕“有限状态机”而进行。

- UART 顶层设计

UART 的顶层设计是使用和整合以上三个模块，以寄存器的方式接到 EBI，控制收/发器模块。

顶层主要是完成寄存器的实现，使 FPGA 配置的 UART 可以以外设方式接受单片机的监控。

以下内容介绍 UART 核心的设计，先介绍一些公用模块的实现。

2.4.2 奇偶校验位设计

如前面的文字可以了解到，奇 / 偶校验是对数据进行逐位同或 / 异或运算。可以概括成如公式（1）和公式（2）所示。

$$D_{EVEN} = D_7 \oplus D_6 \oplus D_5 \oplus D_4 \oplus D_3 \oplus D_2 \oplus D_1 \oplus D_0 \quad \text{公式（1）}$$

$$D_{ODD} = \sim D_{EVEN} \quad \text{公式（2）}$$

对于串行UART，数据是逐位传输，奇偶检验显得很简单，如图 2.8所示为串行数据奇偶校验示意图，可以看出，只需要一个异或门和一个D触发器，D触发器用于寄存当前异结果，异或门监视串行数据，检测到下一个串行数据到来时，把这位的串行数据和D触发器寄存的结果作异或运算，D触发器再把这个异或结果寄存起来。

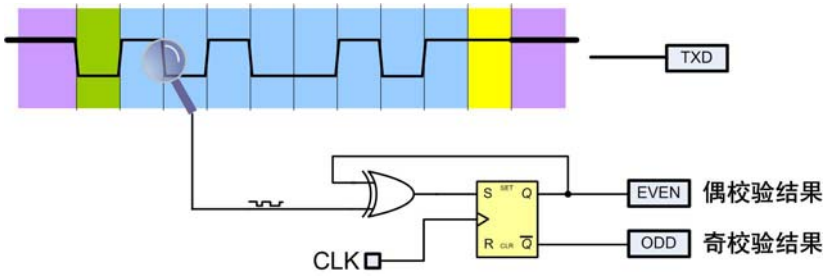

图 2.8 串行数据奇偶校验

UART 收/发器分别包含一个奇偶校验，发送器的奇偶校验用于对将要发送的数据产生检验值，发送到 TXD 线上；接收器的奇偶校验用于对接收的数据产生校验值，再和从 RXD 线上接收到的校验值比较，产生校验结果。

代码实现请见源低码。

2.4.3 有限状态机设计

有限状态机——Finite State Machine（FSM）的设计在 FPGA 设计中很经常用到，在本设计状态机也是采用常用的三步式结构——更新状态机当前状态、判断状态机下一个状态和根据状态机当前状态决定输出。

如程序清单 2.3为FMS三步式结构伪代码，本设计UART的接收器和发送器的状态机是按照这种代码结构编写。

首先，先定义状态机的状态值和编码方式（如 Gray、One-Hot 等编码），按规范编写的状态机可以被综合器发现，可以在综合器选项中选择状态机的编码方式，所以代码中状态机的状态值设定不是太关键，综合器是会自动分配，但从代码的可读性角度考虑，建议认真定义各个状态。

程序清单 2.3 FMS 三步式结构伪代码

```
/*
 *  定义状态机的状态
 */
`define STAT_WIDTH     3
parameter
   STAT_IDLE   =   1,
   STAT_ONE    =   2,
```

```
    STAT_TWO    =   3,
    STAT_THREE  =   4;

reg [`STAT_WIDTH-1] rStatCur;
reg [`STAT_WIDTH-1] rStatNext;

/*
 *  更新状态机的当前状态
 */
always @(posedge clk or negedge rst_n)
begin
    if (~rst_n) begin
        rStatCur    <= STAT_IDLE;
    end
    else begin
        rStatCur    <= rStatNext;
    end
end


/*
 *  根据当前状态机的状态和判断条件, 决定状态机的下一个状态,
 *  综合成组合逻辑
 */
always @(
    rStatCur or
    ... or
    ...
)
begin
    case (rStatCur)

    STAT_IDLR: begin
    if (...)
        rStatNext   <= STAT_ONE;
    else
        rStatNext   <= STAT_IDLE;
    end

    STAT_ONE: begin
    ...;
    ...;
    end
```

```
   STAT_TWO: begin
   ...;
   ...;
   end

   STAT_THREE: begin
   ...;
   ...;
   end

   default: begin
      rStatNext   <= STAT_IDLE;
   end

   endcase
end


/*
 *  根据当前状态机的状态, 决定输出
 */
always @(posedge clk or negedge rst_n)
begin
   if (~rst_n) begin

   end
   else begin
      case (rStatCur)

      STAT_IDLR: begin
      ...;
      ...;
      end

      STAT_ONE: begin
      ...;
      ...;
      end

      STAT_TWO: begin
      ...;
      ...;
      end
```

```
        STAT_THREE: begin
        ...;
        ...;
        end

        default: begin
        ...;
        ...;
        end

        endcase
    end
end
```

状态机的设计在 FPGA 设计中很重要，同时会有一此注意事项，如在本设计中的状态机需要使用“安全状态机”，否则状态机可能会死掉。

关于状态机的设计，请关注创造力电子开发网（http://www.edaok.net）的专题文档。

2.4.4 UART 接收器设计

UART接收器的框图如图 2.9所示。由框图看出，UART接收器的设计主要是围绕“接收状态机”进行。RXD线需经滤波器处理再进入状态机处理，同步时钟是有用于使UART接收器的波特率时钟和发送器同步，波特率计数器产生波特率，控制状态机的数据接收。

整个接收器的工作由 RXD 线上的有效起始位启动。

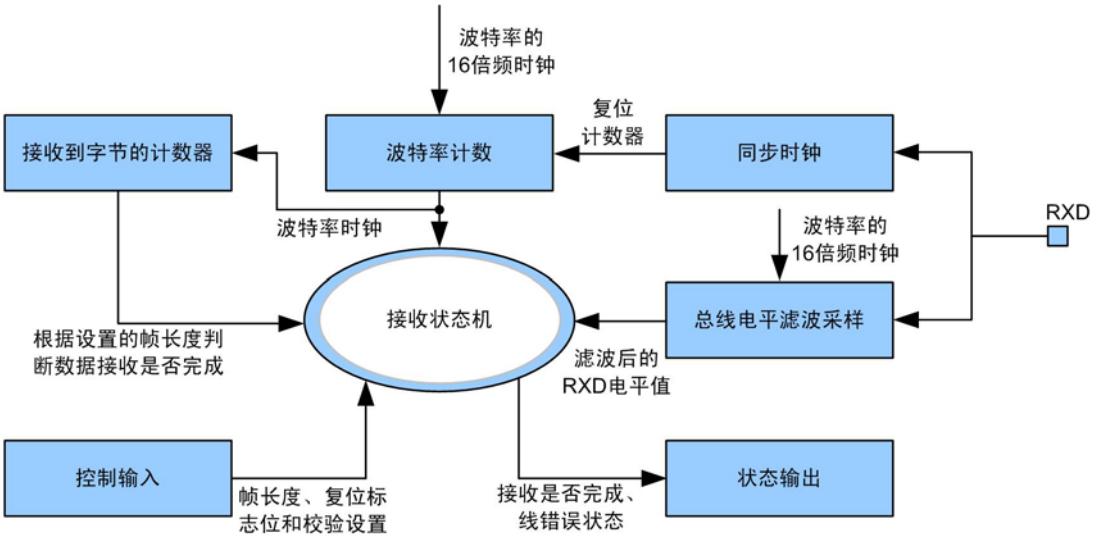

图 2.9 UART 接收器框图

1. 总线电平滤波采样

由于使用环境的干扰等，RXD 线可能出现毛刺，所示需要 RXD 的输入需经滤波处理，处理方法是在一位数据中间取三个点的电平值，少于 2 个点为电平“1”，则把该位判为电平“0”，否则判为电平“1”，这种方法称为“多数表决法”。

如图 2.10所求为这种方法的原理和实现示意图，表 2.2所示为对应的真值表。

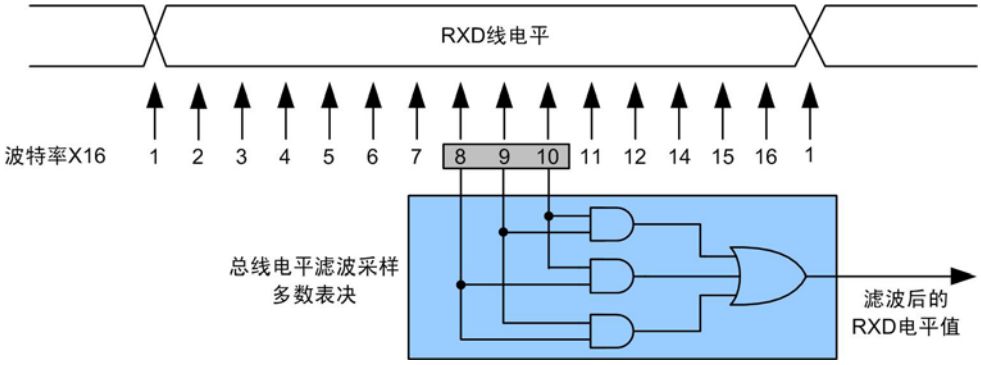

图 2.10 RXD 滤波（多数表决）

表 2.2 RXD 滤波真值表

RXD 线			
D8	D9	D10	输出
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

对 RXD 线上的滤波适用于起始位的检测、数据位接收、校验位接收和停止位的判别。

2. 同步时钟

由于 UART 的收/发两个设备的时钟可以存在差异，如果接收器在接收时和发送器的时钟不同步，增加了传输错误的机率，所以，接收器需要在检测到起始位的起始边沿时，同步自身的时钟，这样将减小因双方时钟上的差异而导致数据传输错误的机率。

该功能的实现是在接收器空闲时，当检测到 RXD 线的电平从高到低的变化时，复位接收器的滤特率计数器，即滤特率计数器在检测到以上条件时才开始计数。

3. 停止位处理

停止位是发送器明确告诉接收器一帧的结束，发送器发送完停止位后可以立即发送下一帧数据的起始位。

如上小节所述，检测一位的电平是取中间三点电平作多数表决运算，这个法规同样对停止位的接收有效。如果多数表决为“0”电平，则置位“帧错误”标志位。

但有别于其它位的接收，如图 2.11所示，完成停止位的接收不是在一个波特位结束处，而是在取完三点电平作多数表决运算后立取结束停止位的接收，之后接收器马上重新监视RXD线电平的下降沿，开始检测下一帧的起始位。

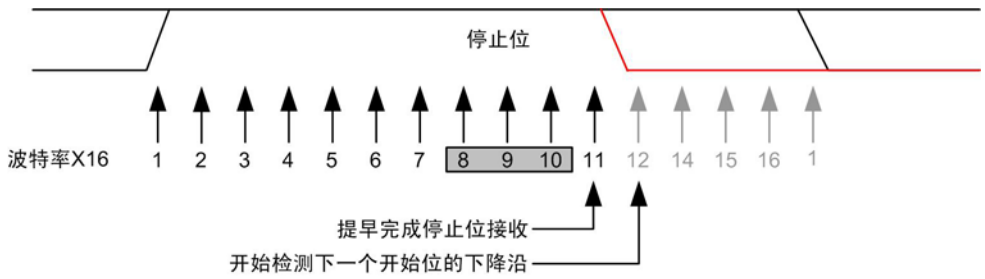

图 2.11 接收器停止位处理

停止位处理可以降低 UART 收/发两个设备因时钟的不一致而导致的传输失败率。

4. 奇偶校验处理

奇偶校验是用于标记状态寄存器的“校验错误”标志位。如2.4.2小节所述的奇偶校验设计，无论是否使能奇偶校验，接收器每接收到一个数据位就运算一次校验，完成数据位的接收后得到刚接收到数据的校验值，如果禁止了奇偶校验，这个校验值被放弃；如果使能了校验，这个校验值会和接发到的校验位作比较，根据比较结果标志相关的状态位。

5. 接收器状态机

图 2.12所示为接收器状态机的状态图，状态机的设计在2.4.3 小节已有所述，本模块的状态机是按照其结构编写。

状态机在空闲状态时，RXD 为高电平时，由状态图看出，空闲状态下，RXD 的下降沿将启动状态机，这个下降沿是 UART 通信协议的起始位。

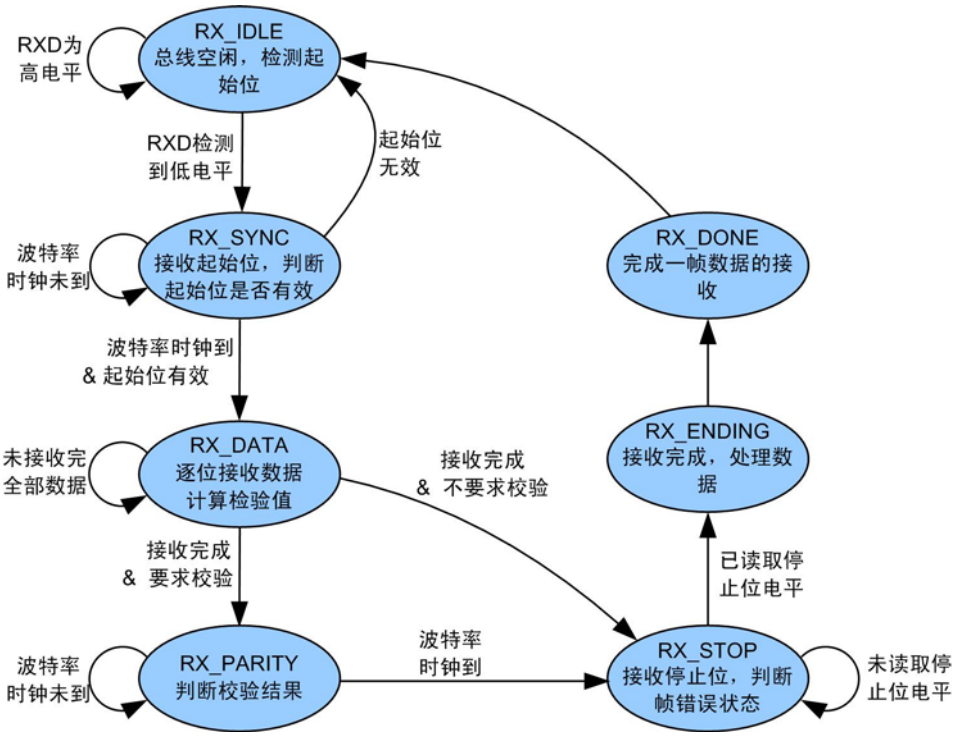

图 2.12 接收器状态机

使用状态机处理接收，可以使设计思路清晰，有利于设计的维护和升级。

6. 模块接口

接收器模块的接口定义如程序清单 2.4所示。

程序清单 2.4 接收器模块接口

```
/*********************************************************************
 **                        All right reserve 2008-2009(C)
 **                    Created &maintain by http://www.edaok.net
 **=====================================================================
 ** 模 块 名:   rxd
 ** 描    述:   实现 UART 的接收功能, 可能选择数据位, 检查奇偶检验, 帧错误检查功能, 实
 **             现了 ARV 单片机的异步串口(UART)
 **
 ** 原 作 者:   Adeko (http://www.edaok.net)
 ** 参 与 者:   (...welcome you join in)
 **
 **=====================================================================
 *********************************************************************/
module rxd (
   clk,
   rst_n,

   clk_en,
   data_o,

   rxd_xi,

   ctrl_i,
   frame_bits_i,
   stat_o,

   enable,

   debug_o
);

input            clk;                          //  全局时钟线
input            rst_n;                        //  全局复位线, 低电平有效

input            clk_en;
output  [7:0]     data_o;                      //  数据的输出口(8bits)

input            rxd_xi;                       //  串口接收引脚

input   [2:0]     ctrl_i;                      //  控制信号输入
input   [3:0]     frame_bits_i;                //  帧位数参数输入
output  [3:0]     stat_o;                      //  状态信号输出
```

```
input             enable;                          //  模块使能线

output  [7:0]      debug_o;
......
......
endmodule
```

2.4.5 UART 发送器设计

发送器的设计相对接收器简单很多，因为发送器只需根据控制输入，往 TXD 逐位输出数据，在这里省去了发送器的设计过程，请自行参考源代码，代码设计和接收器相似。

如程序清单 2.5所示为发送器模块接口。

程序清单 2.5 发送器模块接口

```
/*********************************************************************
 **                            All right reserve 2008-2009(C)
 **                         Created & maintain by http://www.edaok.net
 **=====================================================================
 ** 模 块 名:   uart.v
 ** 描    述:   实现 UART 的发送功能, 数据帧字节长度可设置, 支持奇偶检验功能; 实现了 AVR
 **             单片机的通用异步串口的大部分功能, (没有多机通信, 同步发送方式)
 **
 ** 原 作 者:   Adeko (http://www.edaok.net)
 ** 参 与 者:   (...welcome you join in)
 **
 **=====================================================================
 *********************************************************************/

module txd (
   clk,
   rst_n,

   clk_en,
   data_i,
   enable,

   txd_xo,

   ctrl_i,
   frame_bits_i,
   stat_o,

   busy_o
);
```

```
input             clk;                          //  全局时钟时
input             rst_n;                        //  全局复位时，低有效

input             clk_en;
input   [7:0]      data_i;                      //  将要发送到 UART 总线上的数据
input             enable;                       //  模块使能线

output            txd_xo;                       //  UART 数据发送端口

input   [4:0]      ctrl_i;                      //  控制信号输入
input   [3:0]      frame_bits_i;                //  帧数据位设置输入
output  [1:0]      stat_o;                      //  UART 发送器状态输出

output            busy_o;                       //  发送器正在工作，非空闲
......
......
endmodule
```

2.5 扩展参考

本文档描述了 UART 基本功能的 FPGA 实现，常温安静环境条件下经过测试（时间大于 6 小时），传输没有出现错误。

本设计作为创造力电子开发网的开源项目，希望目前的成果可以敲起大家的兴趣，升级本设计，比如加入 FIFO，解析简单的协议等。

欢迎访问创造力电子开发网（http://www.edaok.net）。

★ **特别注意：**

当前文档于 2009 年 5 月中旬发布，可能已有更新，请访问创造力电子开发网检查更新。