



**数字集成电路设计实验报告**

|  |  |
| --- | --- |
| 实验题目： | 实验三 AHB UART 外设设计与实现 |
|  |  |
| 学生学号： | 35320212200396 |
| 学生姓名： | 黄腾熙 |
| 院 系： | 电子科学与技术学院 |
| 专 业： | 集成电路设计与集成系统 |

# 实验三 AHB UART 外设设计与实现

# 一、实验原理

本实验旨在设计并实现基于AHB总线的UART外设，用于实现FPGA与PC的通信。硬件部分包括设计UART模块及其与AHB-Lite总线的接口，并在FPGA上进行验证；软件部分需编写Cortex-M3处理器的代码，通过UART实现数据的收发。此外，通过调试工具分析硬件信号和软件行为，扩展功能包括支持波特率配置和奇偶校验等，进一步提升UART的灵活性和可靠性。

# 二、实验步骤

1、通过修改程序，使PC的串口终端能够换行

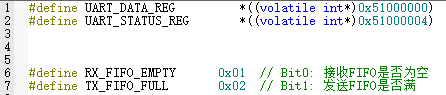
2、增加控制寄存器UART\_CONTROL\_REG，使用bit [1:0]来设置串口的波特率。波特率可设置为以下几个值：0：19200bps（上电默认），1：57600bps，2：115200bps，3:230400bps

3、串口通讯增加奇偶校验位功能，使用控制寄存器UART\_CONTROL\_REG的bit2来控制。若bit2=0，禁止奇偶校验（上电默认）；若bit2=1,使能奇校验（或偶校验）

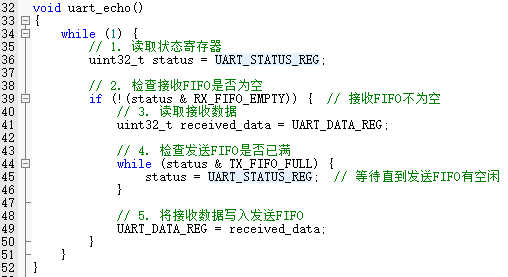
# 三、实验结果分析

**（一）通过修改程序，使PC的串口终端能够换行**

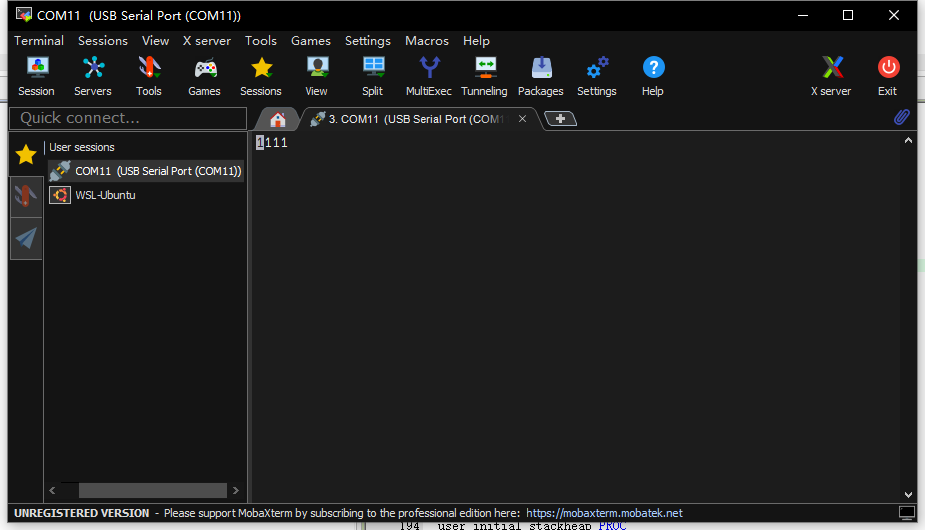
1、首先将Verilog代码烧录至FPGA后，根据所给地址编写对应的宏定义



2、编写串口回环函数



但以上回送函数在输入回车后，会出现以下现象，只进行了回车操作（光标回到第一位），并未进行换行



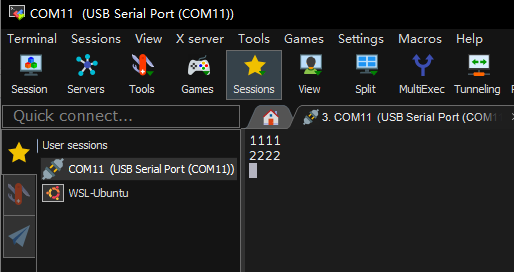
这是因为windows系统的问题，在按下回车键时，仅发送一个回车符而不发送换行符，并未进行换行操作，因此需要对回送函数的逻辑进行相应的修改

3、在回送函数中添加关于回车符的处理逻辑



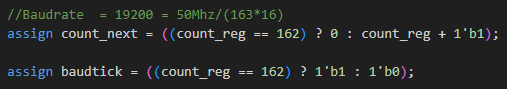
4、经过修改后，再次测试

此时，按下回车键就可以正常进行回车并换行



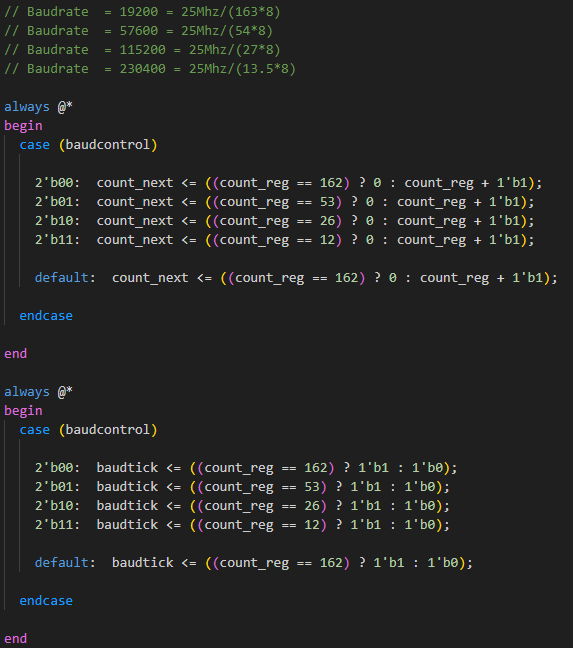
**（二）增加控制寄存器UART\_CONTROL\_REG，使用bit [1:0]来设置串口的波特率。波特率可设置为以下几个值：0：19200bps（上电默认），1：57600bps，2：115200bps，3:230400bps**

1、阅读Verilog代码后发现，波特率主要由baudgen模块控制

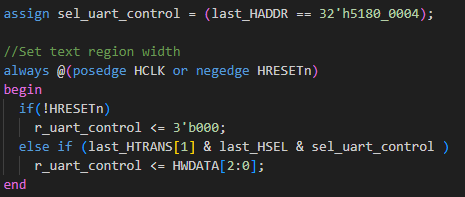


当计数到设定数值后，便发送一次tick，而在RX和TX模块中，会对tick进行计数，当tick次数到达设定数值后，便发送一位数据

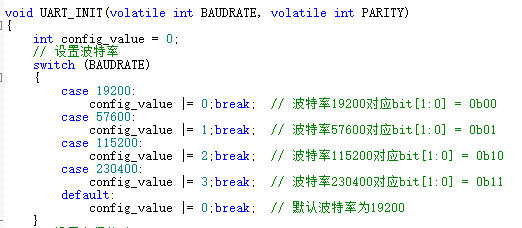
由于在做VGA模块时，为实现高分辨率的显示，我将核心主频设置为25MHz，因此为了实现要求中的高波特率，我将一位数据发送所需的tick改成了8个，并在计算参数时向下取整，对Verilog代码进行如下修改



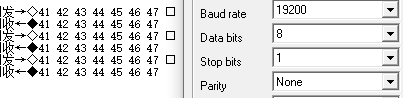
2、添加波特率的设置逻辑后，需要给UART\_CONTROL\_REG分配地址，使得总线上的数据能设置波特率，此处分配的地址为0x5180\_0004

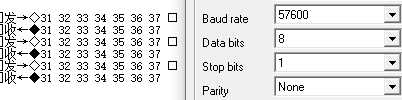


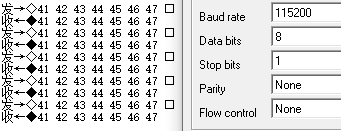
3、编写对应的软件代码

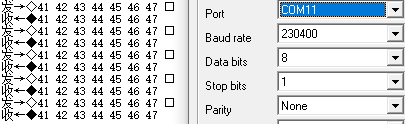


4、测试后在不同波特率下均能正常收发数据



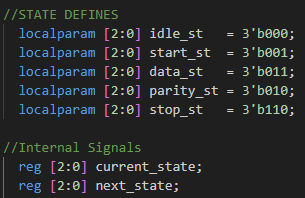




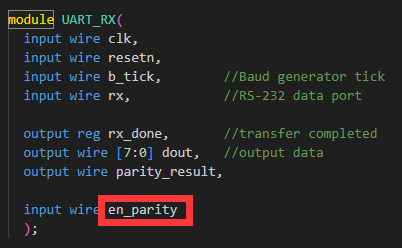


**（三）串口通讯增加奇偶校验位功能，使用控制寄存器UART\_CONTROL\_REG的bit2来控制。若bit2=0，禁止奇偶校验（上电默认）；若bit2=1,使能奇校验（或偶校验）**

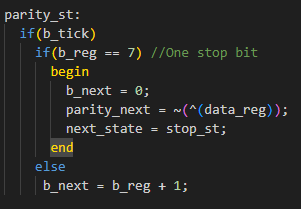
1、要增加奇偶校验，首先需要在RX和TX的状态机中增加校验状态，这样也便于控制



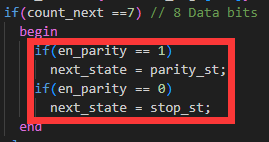
2、增加en\_partiy端口，以控制是否进入校验状态，同时预留partiy\_result接口



3、本次实验只使用奇校验

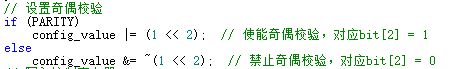


4、同时增加控制部分，当UART\_CONTROL\_REG的bit2为1时就发送校验位，为0则发送停止位



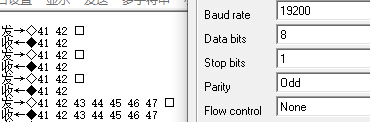
5、对TX部分进行类似的修改

6、编写对应的如那间

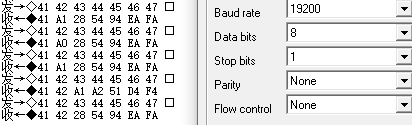


7、测试

设置串口软件奇校验，并发送测试，可以发现串口回送正常



关闭校验则数据开始出现错误，可以说明校验位有效



**（一）写testbench实现soc的仿真验证，要求截取写LED寄存器和读开关寄存器操作时AHB总线上的各个关键信号的波形图**

1、增加时钟RESET信号，确保时钟能正常运行

屏幕的截图

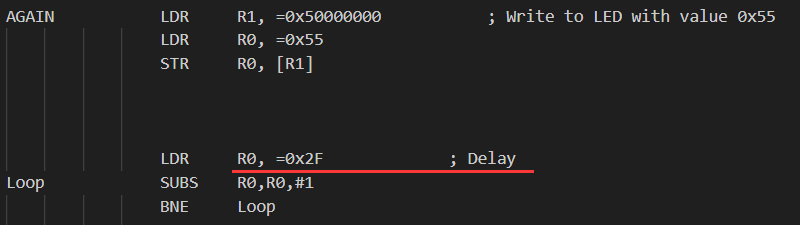
描述已自动生成

2、将需要在SoC上执行的代码路径加入文件

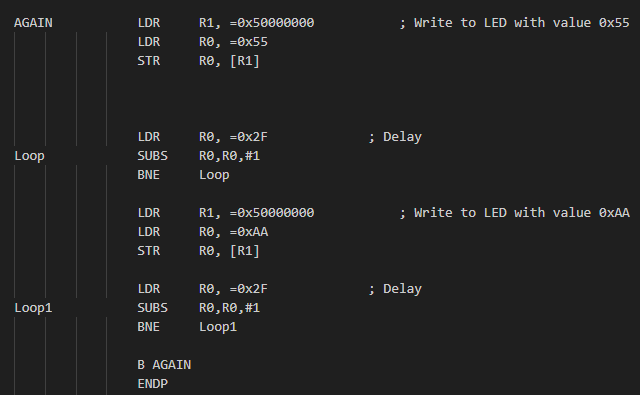
文本

描述已自动生成

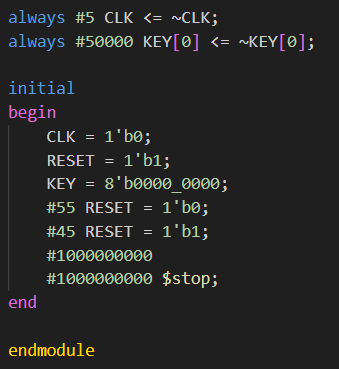
3、调整汇编中的延时，以便于查看仿真波形



4、完整汇编如下

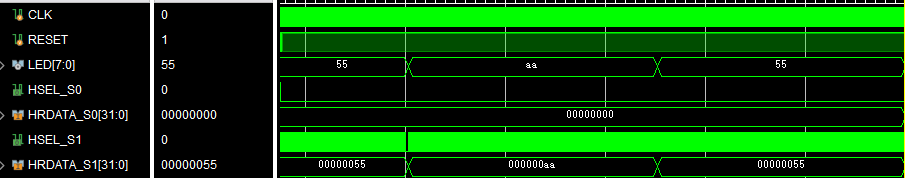


5、编写testbench

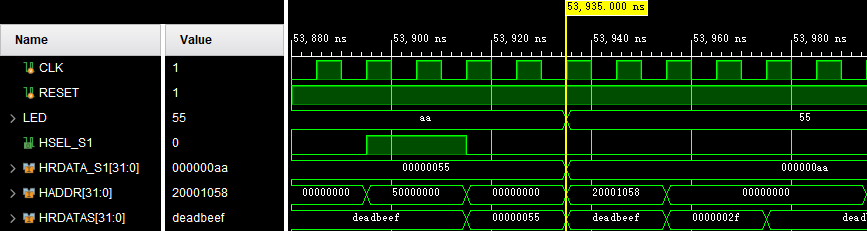


6、运行仿真

可以观察到LED状态随时间变化



仔细观察仿真波形，可以观察到，当地址总线上的地址指向LED寄存器（0x5000\_0000）时，HSEL拉高，同时向寄存器内写入数据总线上的当前值，随后LED输出随之变化。

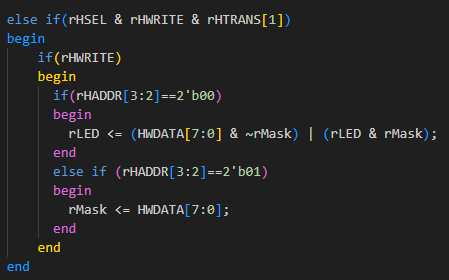


**（二）增加LED的mask寄存器，实现写LED灯时可以屏蔽某些bit**

1、首先为mask定义寄存器



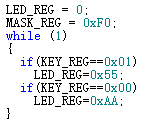
2、编写Verilog代码，为mask寄存器分配地址，此处分配的mask地址即为0x5000\_0004，其中对应的LED寄存器写入逻辑即可实现LED屏蔽



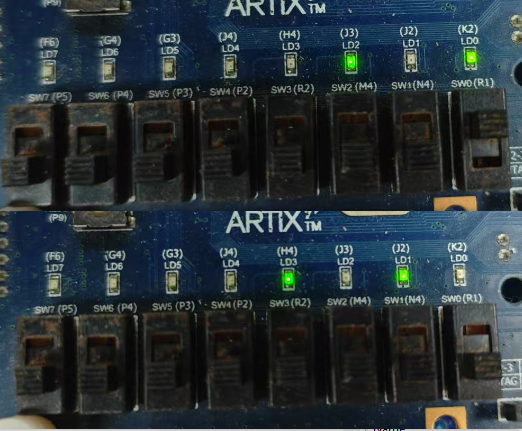
3、编写相应程序

向MASK寄存器中写入一个值



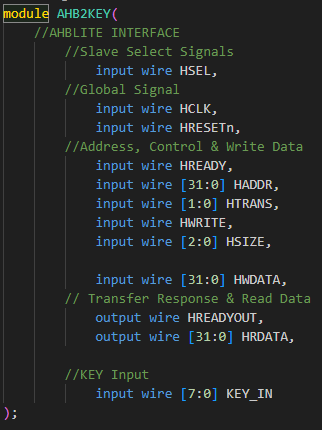


4、烧录并测试，开关时即可实现将高4位LED屏蔽

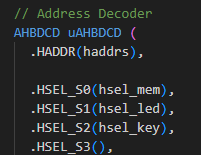


**（三）在AHB上增加挂载switch开关寄存器，实现arm程序读入硬件开关的状态，并结合LED输出寄存器实现拨动开关控制LED灯的亮灭**

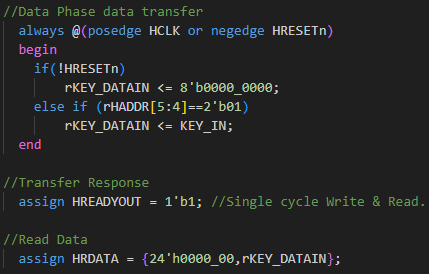
1、仿照LED模块编写KEY模块



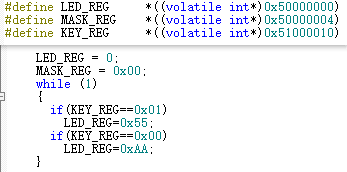
2、在TOP模块中增加KEY外设，并分配基地址0x5100\_0000



3、同时在KEY外设中，分配对应的KEY寄存器地址，以实现KEY值的读取



4、编写如下软件程序并烧录，拨动开关SW0时，应该出现全部LED状态的切换

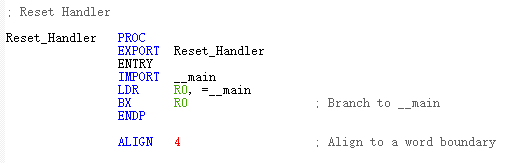


5、实际现象如下，拨动SW0，LED状态变化，实现了拨动开关控制LED灯的亮灭

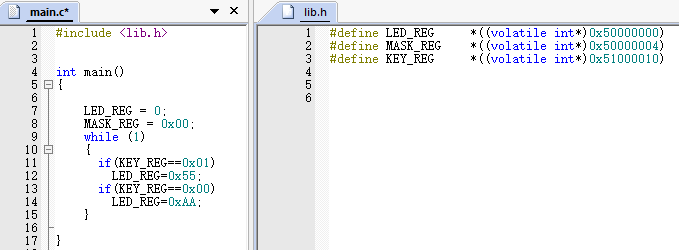


**（四）将arm汇编程序改成使用C语言实现**

1、在汇编程序中添加以下代码，即可实现RESET后跳转至C语言程序中的main函数中



2、编写C语言程序，其中的宏定义与Verilog硬件对应



3、烧录后运行效果与汇编程序相同