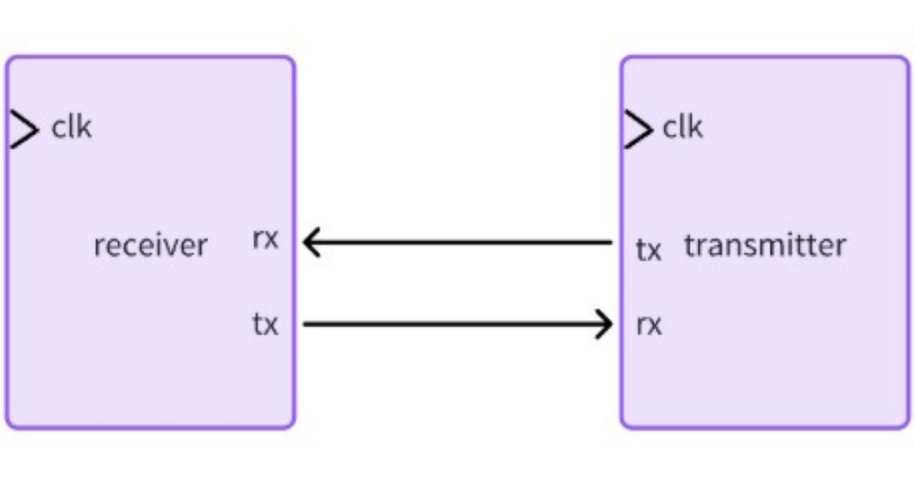
实验原理

1、UART介绍

UART全称是 universal asynchronous receiver-transmitter，通用异步收发器，是一种异步的串行收发协议，数据格式和传输速率都可配置。广义上来说，采用串行接口进行数据通信的接口都可以称为串口，如 SPI 接口等，但通常所说的串口一般是指 UART。

UART收发两端各只需要两根信号，一根发送一根接收，能同时实现收发。



1.2 数据帧格式

UART串行收发需要解决两个问题：

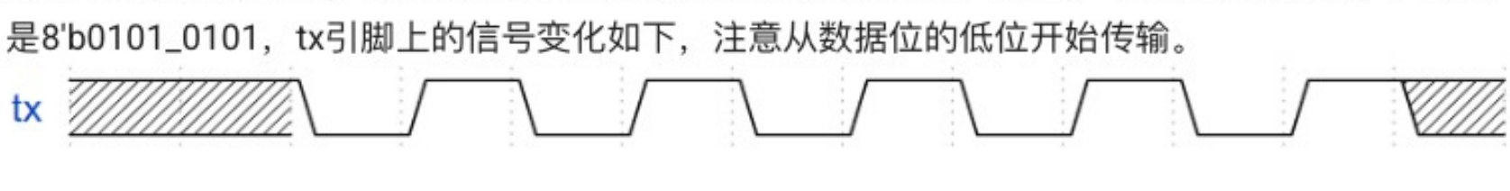
- UART收发两端使用各自的时钟，时钟频率一般不同，怎么解决数据接收端解析出来的数据与发送端一致？通过波特率约定每一位数据占据的时间。收发两端各自以该约定的周期去处理数据。
- 通信双方不会一直有数据发送，怎么判断对端什么时候发送，什么时候结束？发送序列是一串长长的 0-1 串，需要约定固定的格式用于识别。

UART收发两端采用相同的数据格式，即数据帧，空闲时始终是高电平，一个UART数据帧包含以下内容：

- 起始位：Start bit，有且只有1位，低电平；
- 数据位：Data bits，5-9位，实验当中选择8位，先发送低位，再发送高位
- 奇偶校验位：Parity bit，判断数据位中是奇数个1还是偶数个1，用作校验数据发送过程中是否变化，提升可靠性，可选，实验当中不设置；
- 停止位：Stop，1位或者2位，高电平。

空闲时的数据也叫空闲帧。以上数据位、奇偶校验位、停止位是可配置的，根据需要而定，实验要求实现1位起始位、8位数据位、1位停止位的数据帧格式，即8N1。每一数据帧都由「起始位 + 数据位 + 停止位」三部分组成，相邻两个数据帧之间可以插入任意长度的高电平的空闲帧，传输效率最高是（8/(1+8+1)）= 80%。

比如发送ASCII字符U，查ASCII表可知对应的ASCII值（十进制）是85，十六进制是8'h55，二进制是8'b0101_0101，tx引脚上的信号变化如下，注意从数据位的低位开始传输。



1.3 波特率

baud，波特率也叫符号率，即每秒发送的符号个数。容易混淆的另一个概念是比特率，比特率是每秒发送的比特数量，当一个符号只包含两种可能，则波特率等于比特率。

UART常用的波特率有9600，19200，115200等。波特率9600，指的是一秒钟最多传输9600个符号，一个符号就是1个数据位，所以每一位持续的时间是（1/9600）秒。

1.3 Minisys开发板UART接口

开发板上用了 CP2102 芯片，将 UART 转换成 USB 接口，再通过microUSB或者TypeC的接口连接电脑的 USB 口，电脑会将这个接口识别为串行器件。UART口的收发引脚如下：

信号名称	FPGA管脚	信号描述
UART_RX	Y19	FPGA接收串口信息管脚
UART_TX	V18	FPGA发送串口信息管脚

2、回环测试

先通过回环测试掌握下UART收发操作。回环，即将发送端直接接到输出端，形成一个环路，电脑端的串口软件发送数据过来，FPGA接收到之后原样发送回去。回环测试已经提供了完整代码，新建Vivado项目，添加以下功能代码和约束代码，生成比特流即可测试。也可点击uart_loopback.bit下载生成好的比特流文件。

```
module uart_loopback (
    input   uart_rx,
    output  uart_tx
);

    assign uart_tx = uart_rx;

endmodule
```

约束文件

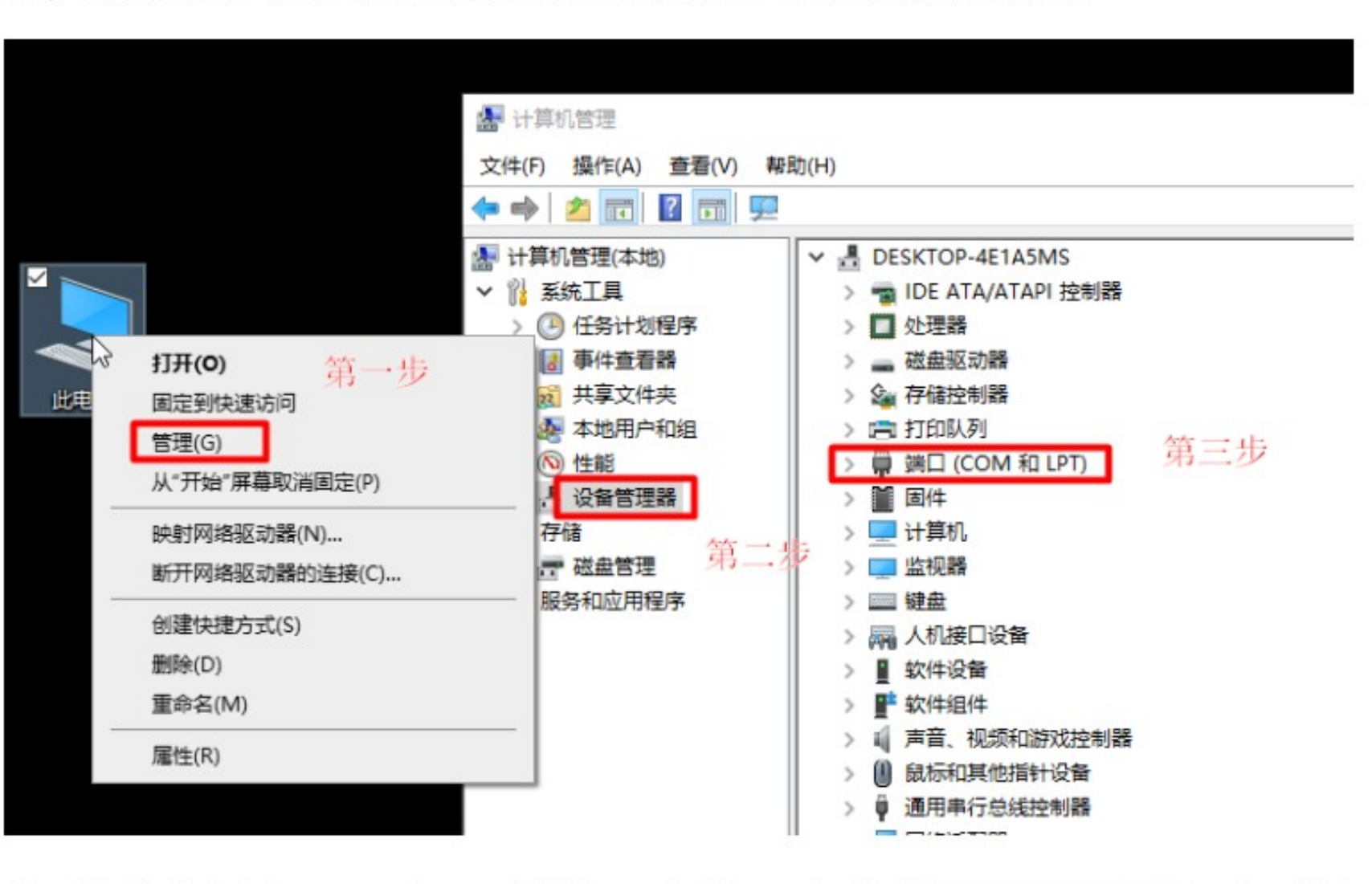
```
set_property -dict { PACKAGE_PIN Y19 IOSTANDARD LVCMOS33 } [get_ports  uart_rx];
set_property -dict { PACKAGE_PIN V18 IOSTANDARD LVCMOS33 } [get_ports  uart_tx];
```

注意

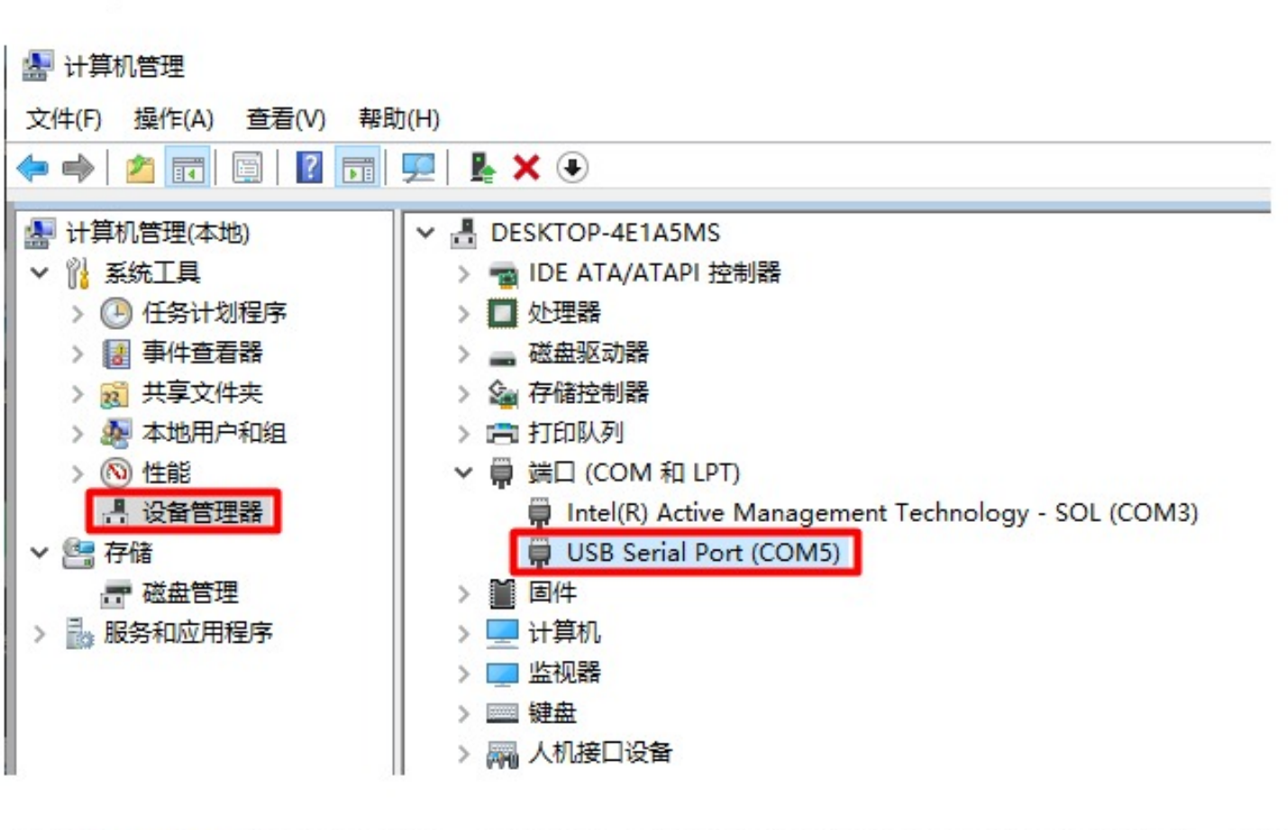
回环测试非必做，是帮助大家理解串口功能和工具使用，约束文件可以参考，操作步骤上板测试需要用到。

开发板V2和V3的TypeC口直接支持UART，开发板V1还需要另外接一条Micro USB的线，请T2615上课使用V1开发板的同学，如果microUSB线不够，课上再找老师拿一条。如下图所示接到右边的Micro USB的口，接口旁边的黄灯会亮。

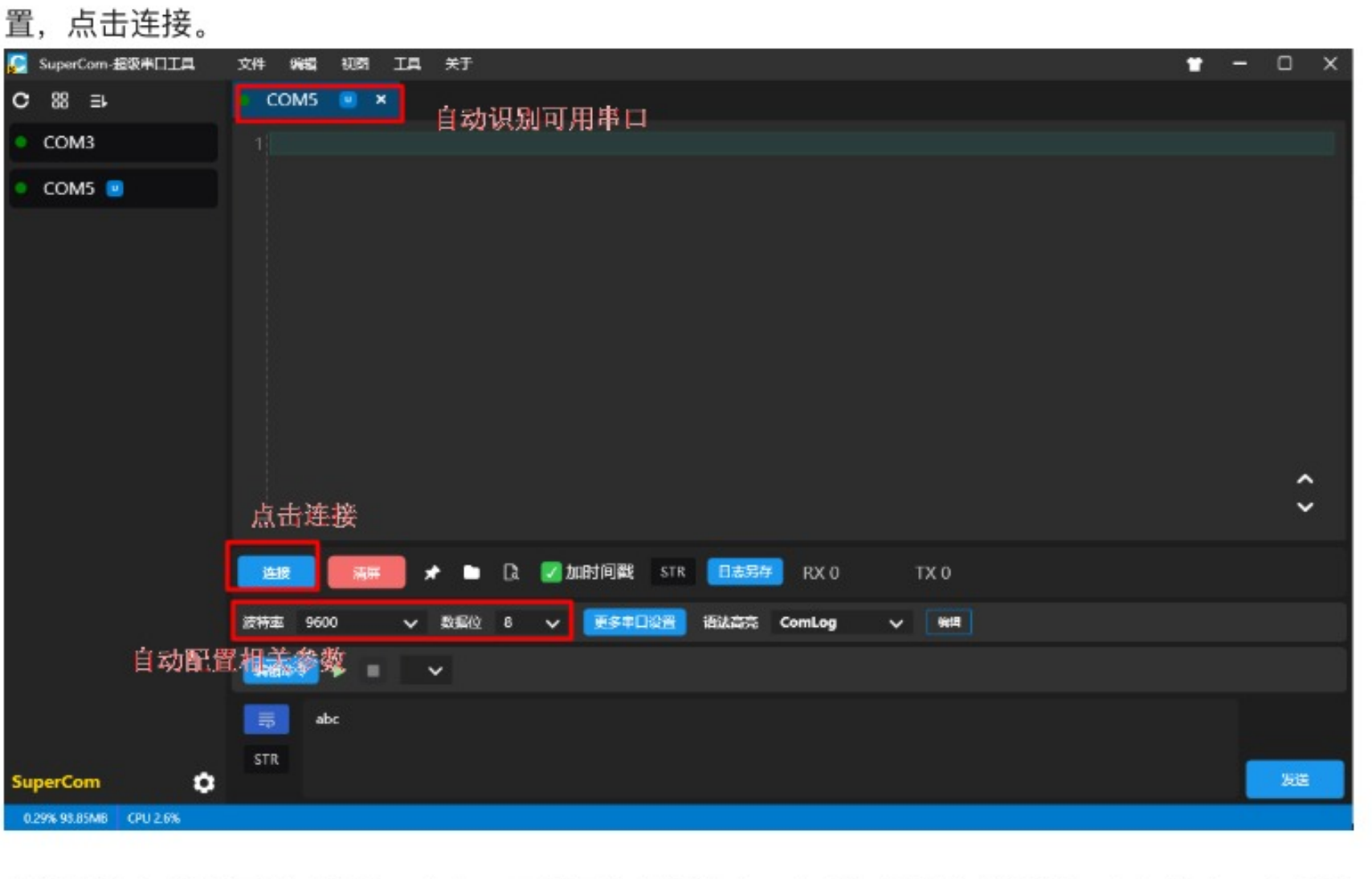
接下来是电脑端的串口操作，按下图所示，先右键"此电脑"打开"管理"功能，再打开"设备管理器"，再打开端口。也可直接在windows的搜索框中搜索"设备管理器"进行打开。



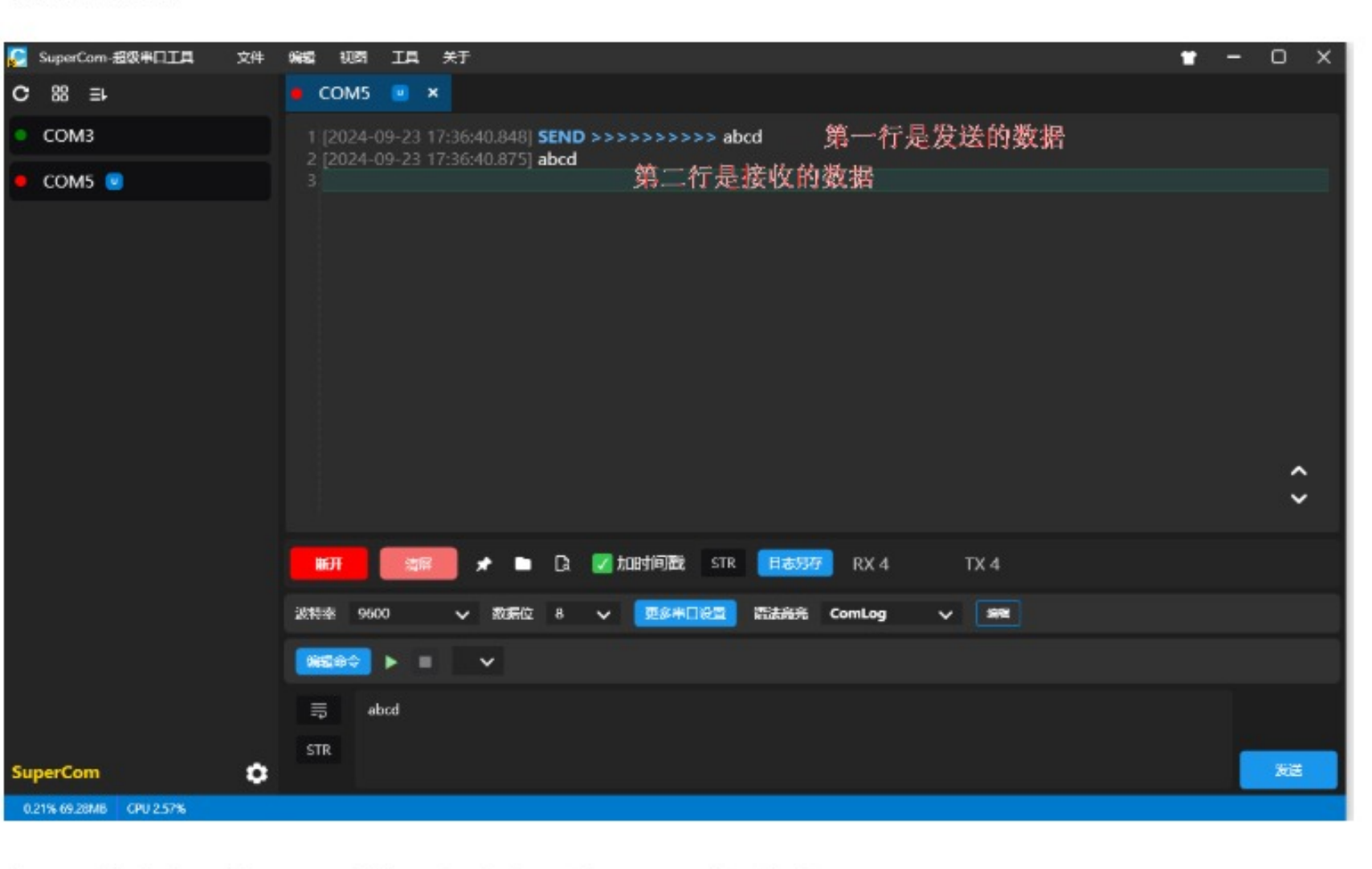
将开发板连接到电脑USB口，打开开发板电源，在"端口"下方会新增 `USB Serial Port` 项（开发板V1是显示 `Silicon Labs CP210x USB to UART Bridge`），记住后边的COM号，比如这里是COM5。



点击SuperCom下载串口软件，若实验室电脑D盘或E盘已有该软件无需再下载。该软件是开源的项目，托管在github。解压后双击 `SuperCom.exe` 打开，会自动识别到可用的端口号和配置，左侧若显示多个COM口，有蓝色图标"U"表示可用，确认COM口的编号和上图中的一致，确认波特率的设置，点击连接。

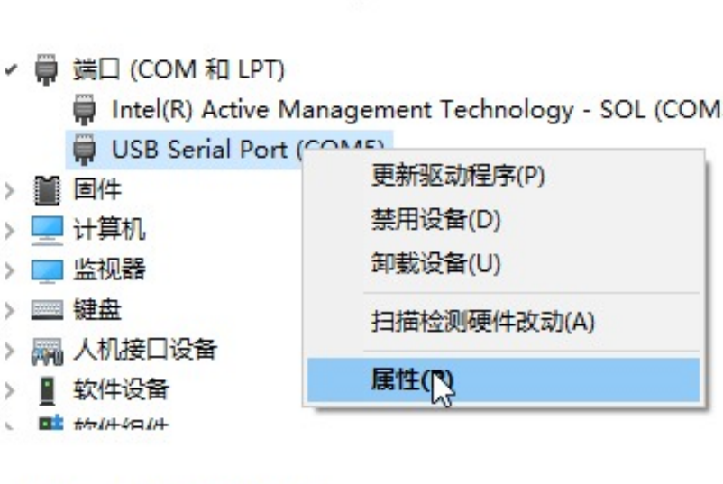


将比特流文件烧写到开发板，在串口工具下方的发送串口中输入任意字符发送，上方的串口会显示收到的字符。



```
1 [2024-09-23 17:36:40.848] SEND >>>>>>>>>> abcd
2 [2024-09-23 17:36:40.875] abcd
3
```

如果要修改串口的配置，按如下操作打开端口设置进行修改



USB Serial Port (COM5) 属性

每秒位数 (B): 9600
数据位 (D): 8
奇偶校验 (P): 无
停止位 (S): 1
流控制 (F): 无

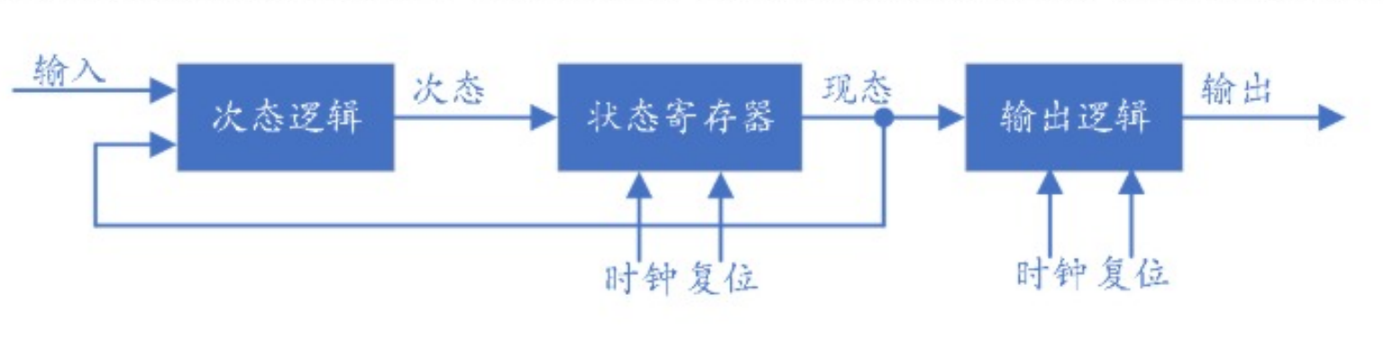
3、状态机

状态机的本质就是对具有逻辑顺序或时序规律事件的一种描述方法。状态机能够根据控制信号按照预先设定的状态进行状态转移，是协调相关信号动作、完成特定操作的控制中心。根据输出是否与当前输入有关，可将状态机分为Moore型状态机和Mealy型状态机。结构上看，两种状态机都包含以下3部分。

- 次态逻辑：组合电路
- 状态寄存器：时序电路
- 输出逻辑：可以是组合电路，也可以是时序电路，时序电路输出比较稳定

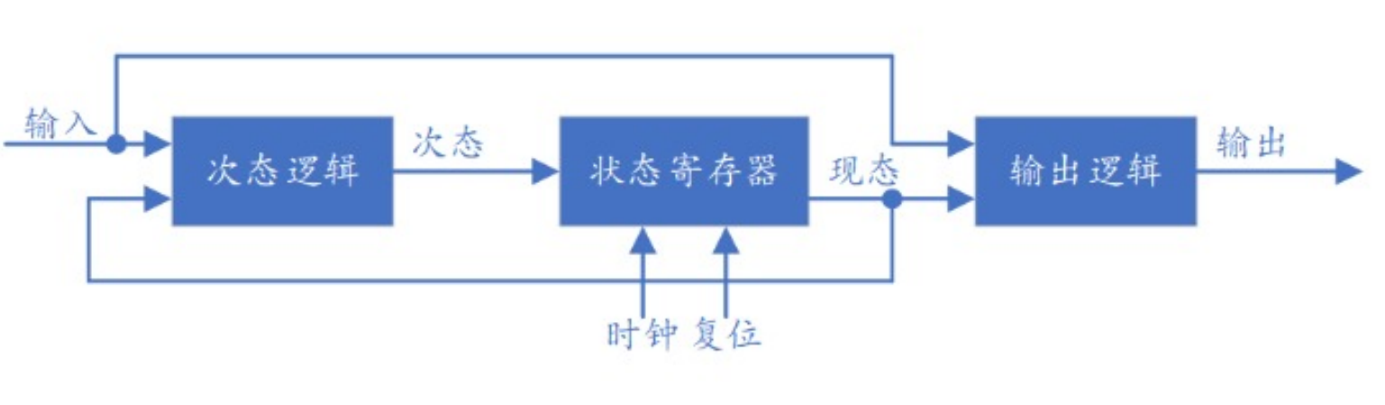
3.1 Moore型状态机

若状态机的输出只和现态有关而与当前输入无关，则称其为Moore型状态机，其原理如下图所示。



3.2 Mealy型状态机

输出不仅和现态有关而且和当前输入有关



3.3 状态机Verilog描述

状态机Verilog实现重点是建模：状态转移条件（次态逻辑）、状态寄存器、输出逻辑，最常见的有三种描述方式：

- 一段式：整个状态机写到一个always块里面；
- 二段式：用两个always块来描述，一个always块采用同步时序描述次态到现态的转移，一个always块采用组合逻辑描述状态转移条件判断以及输出；
- 三段式：使用三个always块，一个always块采用同步时序描述状态转移，一个always块采用组合逻辑判断状态转移条件，另一个always块描述状态对应的输出(可以用组合电路输出，也可以时序电路输出)。

always块的数量并不是严格的1个，2个，3个。如果输出比较复杂，按照Verilog编码规范，每个always块只进行一个变量的赋值，用到多个always块，则always块数量明显多于3个，但仍然属于3段式。

3.4 三段式描述方法示例模板

3段式的示例如下

```
parameter IDLE = 3'b001;
parameter S1   = 3'b010;
parameter S2   = 3'b100;

reg [2:0] current_state;
reg [2:0] next_state;
reg [1:0] out ;

// 第1个always块，描述次态迁移到现态
always @(posedge clk or posedge rst) begin
    if(rst)  current_state <= IDLE;
    else     current_state <= next_state;
end

// 第2个always块，描述状态转移条件判断
always @(*) begin
    case (current_state)
        IDLE: if(...) next_state = S1;
              else    next_state = IDLE;
        S1: if(...)   next_state = S2;
            else      next_state = S1;
        S2: if(...)   next_state = IDLE;
            else      next_state = S2;
        default :     next_state = IDLE;
end

// 第3个always块，描述输出逻辑，也可以用next_state作判断，对时序不敏感的电路两者都可以。
always @(posedge clk or posedge rst) begin
    if(rst) out <= 2'b00;
    else begin
        case(current_state)
            S1 :      out <= 2'b01;
            S2 :      out <= 2'b10;
            default : out <=2'b00;
    end
end
```

4、UART发送的状态机实现

4.1 模块接口和状态定义

串口发送模块，将 8 位数据转换成符合串口协议的数据帧通过tx引脚发送出去即可。发送功能比较简单，有多种实现方式，实验要求串口发送的核心功能按如下接口用三段式状态机实现。根据数据帧格式，定义了4个状态，请画出对应的状态图，并完成功能代码实现。

```
module uart_send(
    input         clk,
    input         rst,
    input         valid,      // 为1表明接下来的8位data有效，只维持一个时钟周期
    input [7:0]   data,       // 待发送的8位数据
    output reg    dout        // 发送信号
);

localparam IDLE  = 2'b00;   // 空闲态，发送高电平
localparam START = 2'b01;   // 起始态，发送起始位
localparam DATA  = 2'b10;   // 数据态，将8位数据位发送出去
localparam STOP  = 2'b11;   // 停止态，发送停止位

endmodule
```

注意valid表明输入的数据data是否有效，高电平代表有效，只维持一个时钟周期。该信号用来控制UART模块是否往外发数据，本实验虽不断的发送数据，但每个字符之间仍需停顿。

4.2 波特率计算

波特率由计数器控制，发送时每一位持续的时间约：(1/波特率)秒，波特率是9600，则每一个位持续（1/9600）= 0.000104秒，持续时间通过计数器实现，计数器计数次数公式如下：

$$\text{计数次数} = \frac{\text{时钟频率}}{\text{波特率}}$$

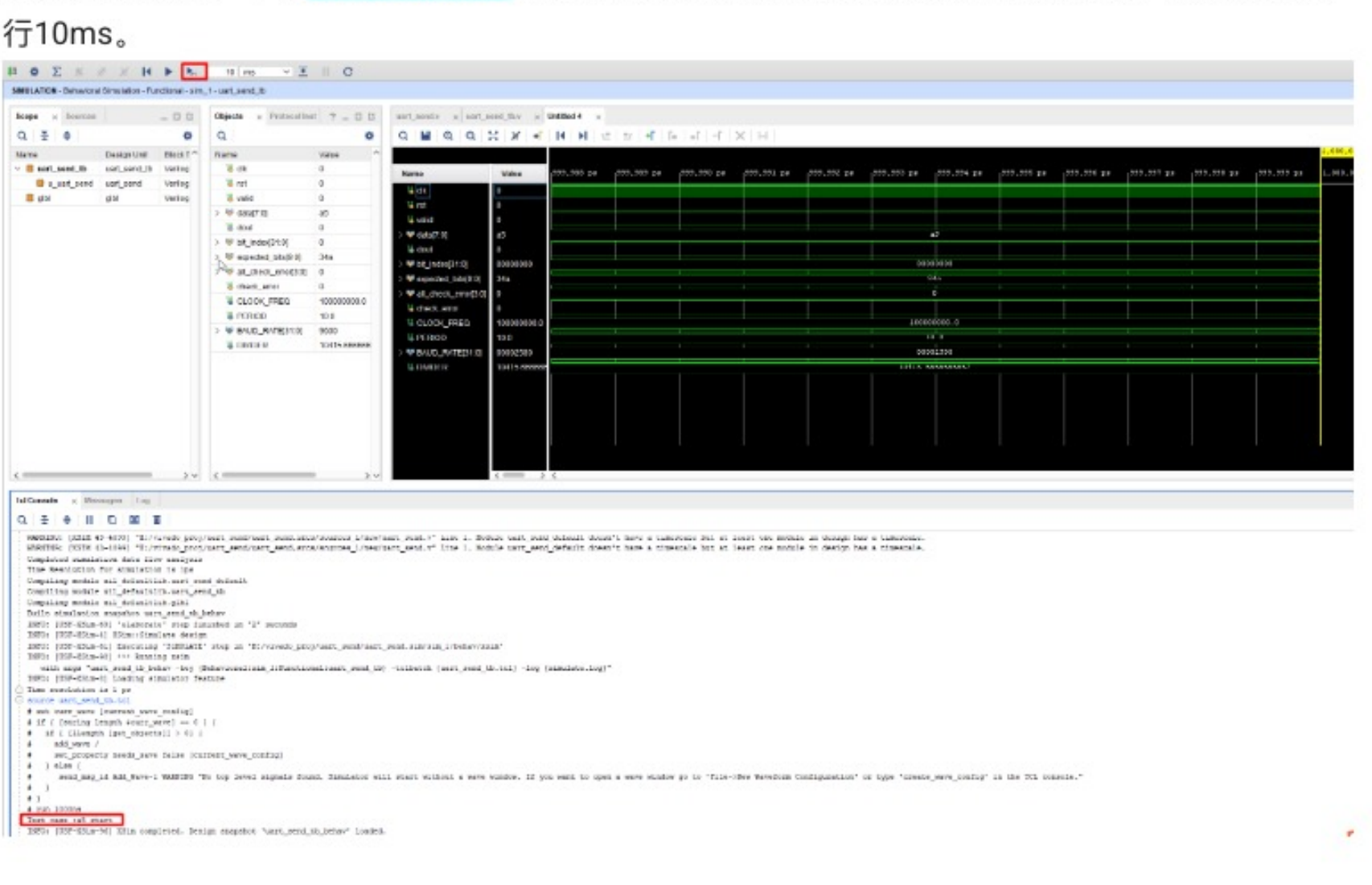
开发板的时钟频率是100MHz，以9600波特率为例，计数次数应为：

$$\frac{100MHz}{9600} \approx 10416.7 \approx 10416$$

如果收发两端的波特率不匹配，接收端会收到错误的数据或乱码。

4.3 发送模块仿真验证

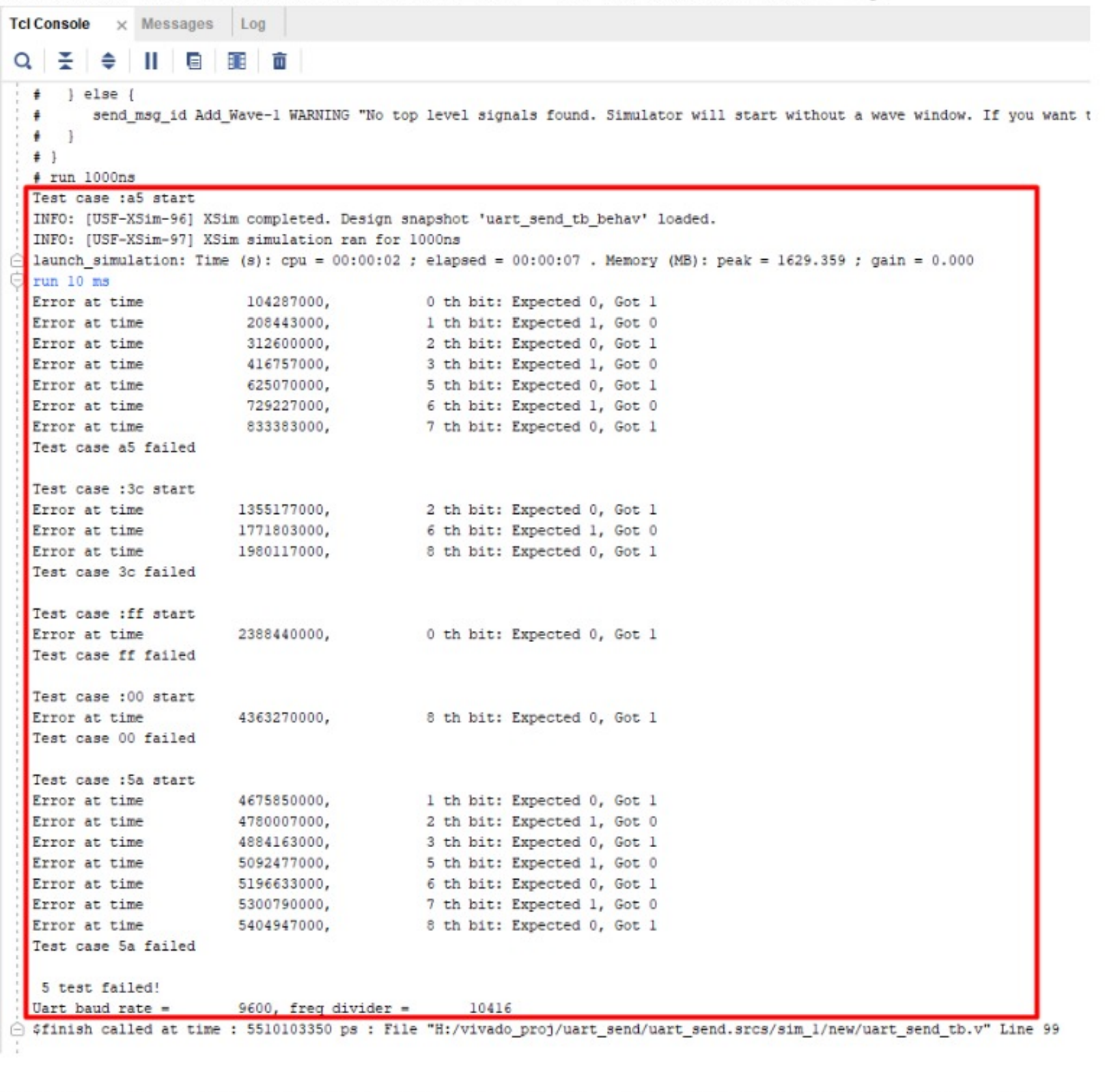
完成RTL代码后，下载uart_send_tb.v添加到工程进行仿真验证。默认仿真时间不够，需要继续运行10ms。



结束后点击 `Vivado` 下方的 `Tcl Console` 窗口，可以看到校验结果日志信息，总共测试了5个数据，如果实现正确，会提示测试通过。

```
Test case :a5 start
INFO: [USF-XSim-96] XSim completed. Design snapshot 'uart_send_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:04 ; elapsed = 00:00:07 . Memory (MB): peak = 1629.359 ; gain = 0.000
run 10 ms
Test case a5 success

Test case :3c start
Test case 3c success

Test case :ff start
Test case ff success

Test case :00 start
Test case 00 success

Test case :5a start
Test case 5a success

All tests passed successfully!
Uart baud rate =        9600, freq divider =       10416
$finish called at time : 5510103350 ps : File "H:/vivado_proj/uart_send/uart_send.srcs/sim_1/new/uart_send_tb.v" Line 99
```

如果功能不正确，会有错误提示，提示哪些位不一致，需要结合波形自行debug。



```
#   } else {
#       send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want t
#   }
# }
# run 1000ns
Test case :a5 start
INFO: [USF-XSim-96] XSim completed. Design snapshot 'uart_send_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:02 ; elapsed = 00:00:07 . Memory (MB): peak = 1629.359 ; gain = 0.000
run 10 ms
Error at time          104287000,          0 th bit: Expected 0, Got 1
Error at time          208443000,          1 th bit: Expected 1, Got 0
Error at time          312600000,          2 th bit: Expected 0, Got 1
Error at time          416757000,          3 th bit: Expected 1, Got 0
Error at time          625070000,          5 th bit: Expected 0, Got 1
Error at time          729227000,          6 th bit: Expected 1, Got 0
Error at time          833383000,          7 th bit: Expected 0, Got 1
Test case a5 failed

Test case :3c start
Error at time         1355177000,          2 th bit: Expected 0, Got 1
Error at time         1771803000,          6 th bit: Expected 1, Got 0
Error at time         1980117000,          8 th bit: Expected 0, Got 1
Test case 3c failed

Test case :ff start
Error at time         2388440000,          0 th bit: Expected 0, Got 1
Test case ff failed

Test case :00 start
Error at time         4363270000,          8 th bit: Expected 0, Got 1
Test case 00 failed

Test case :5a start
Error at time         4675850000,          1 th bit: Expected 0, Got 1
Error at time         4780007000,          2 th bit: Expected 1, Got 0
Error at time         4884163000,          3 th bit: Expected 0, Got 1
Error at time         5092477000,          5 th bit: Expected 1, Got 0
Error at time         5196633000,          6 th bit: Expected 0, Got 1
Error at time         5300790000,          7 th bit: Expected 1, Got 0
Error at time         5404947000,          8 th bit: Expected 0, Got 1
Test case 5a failed

 5 test failed!
Uart baud rate =        9600, freq divider =       10416
$finish called at time : 5510103350 ps : File "H:/vivado_proj/uart_send/uart_send.srcs/sim_1/new/uart_send_tb.v" Line 99
```

所提供的testbench是在每位的末尾进行校验，要求每一位都维持与波特率相对应的时间。仿真里面用的 `DIVIDER = CLOCK_FREQ / BAUD_RATE -1;` ，vivado的除法算出来多了1，所以减去1，需要保证9600的波特率维持的时钟周期数是10416。如果因为dout错开了几个时钟周期，导致仿真不过，大家重点检查下计数器计数值对不对。如果计数值差一点，会出现上板正常，仿真过不了。

4.4 添加顶层模块

上述的 `uart_send` 模块写完后不能直接添加约束文件就上板， `valid` 和 `data` 不是直接从IO管脚输入的，需要再添加顶层模块，在顶层模块中生成对应的valid和data信号，并例化 `uart_send` ，将个人的学号不断循环往复的往外发。

接收端的串口软件按照ASCII字符表解析，参考：ASCII字符表。注意ASCII表很多字符是控制字符，是不可见的，串口工具不会显示。根据ASCII表找到自己学号对应的ASCII值，将对应的数值发送出去即可。

顶层模块实现后，再添加约束文件生成比特流上板测试。

5、常见问题

- uart_send.v必须按照指定接口实现，其他文件不做限制。**不能** 在uart_send.v中实现发送学号的逻辑。
- 接收端是否自动换行看字符发送的速率以及是否勾选了"加时间戳"，如果不勾选"加时间戳"，不会自动换行；如果勾选"加时间戳"，两个相邻的字符间隔比较短，也会连在一起不会换行。
- testbench中间有个测试用例输入的数据只维持一个时钟周期，该测试用例需要重点关注。testbench只做了基本的功能验证，有些情况没有覆盖，比如连续发送。
- 最后一个测试用例不过，把相关计数变量添加到仿真波形，检查每位持续的时间是不是10416个时钟周期。

5.1 仿真通过下板不过

如果仿真通过，下板串口软件没收到数据，或者收到的是乱码，或者多接收了数据，大概率代码不规范导致仿真和综合不一致。可以排查以下原因：

- valid信号是否正确设置；
- 功能做减法，改为只发送单个字符比如A，看串口软件收到什么内容，可以点击"加时间戳"旁边的STR切换位HEX（十六进制）模式查看接收到的原始数据；
- 注意发送可见ASCII字符，对应ASCII值是33-126；
- 检查串口软件波特率、数据位宽的配置是否与发送端一致；
- 检查Messages窗口，重点查看提示具体代码行数的warnings，处理代码规范性问题。

5.2 随机出现分行

少部分同学尽管按时间间隔要求发送，上板后仍出现字符串随机分行，解决方式有如下两种：

- 方式1：将串口软件分包超时调大比如100ms。感谢卢皓霖、乔印程两位同学的反馈。

编码 ASMO-708
校验位 None
停止位 One
读超时 2000 ms
写超时 1000 ms
分包超时 100 ms
握手 None

- 方式2：串口设置中调低bm选项延迟计时器，默认是16ms，可以调低到1ms。感谢文以勒同学反馈。