

实验 3-Verilog 综合实验：串口 uart

一、 实验准备

由于新的 FPGA 实验班没有七段数码管，然而后续实验中我们需要增加一种实验手段来输出执行结果。因此，本次实验需要完成串口通讯实验，完成计算机（PC 电脑）与 FPGA 芯片之间的通讯交互，为后面的 MIPS CPU 设计提供测试交互接口，输出打印 CPU 的计算结果。本次实验在后续的实验中陆续会用到，本实验手册提供了所有实验的操作步骤，包括串口收发模块，FIFO 模块，以及模块之间的互联。在进行本次实验前，你需要具备以下基础能力：

- 1.熟悉 Vivado 的仿真功能 (行为仿真)
- 2.理解串口协议

本次实验会提供现成的串口模块，你需要自行根据提供的波形图，自行编写程序将串口模块进行连接，以实现回环功能并进行测试。

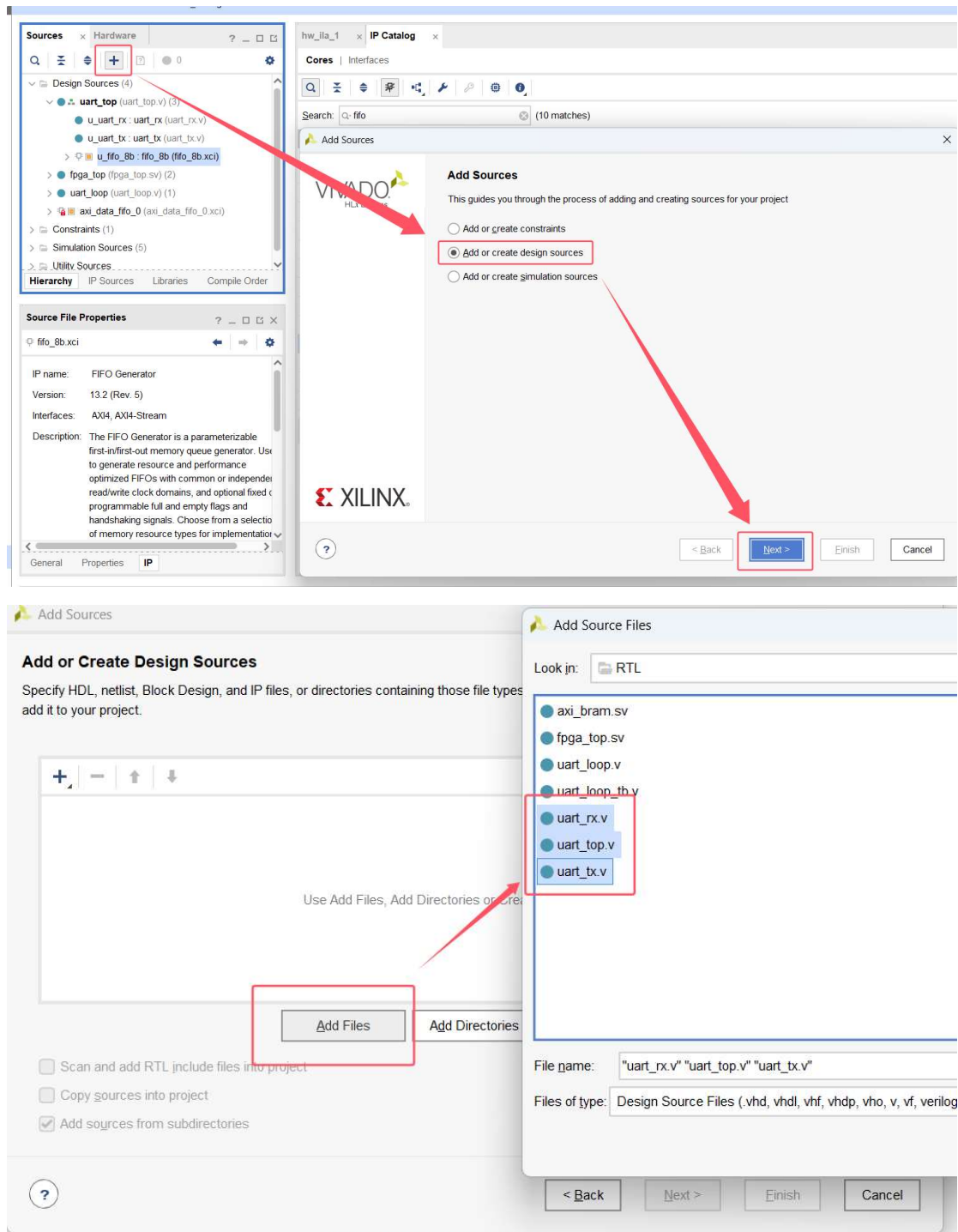
二、 实验目的

1. 掌握串口原理和时序协议标准
2. 读懂串口 Verilog 代码
3. 掌握测试串口通讯的方法。

三、 实验步骤

本章节将会介绍如何进行本次实验。

1. 添加样例代码



2. 添加 IP 核

The screenshot illustrates the process of adding a FIFO Generator IP core to a Xilinx project.

Top Panel: IP Catalog

- The **IP Catalog** window is open, showing search results for **fifo** (10 matches).
- The **FIFO Generator** core is highlighted in the search results.

Bottom Panel: FIFO Generator (13.2) Configuration

- The **Component Name** is set to **fifo_8b**.
- The **Native Ports** tab is selected.
- Read Mode**: **Standard FIFO** is selected.
- Data Port Parameters**:
 - Write Width**: 8
 - Write Depth**: 1024 (Actual Write Depth: 1024)
 - Read Width**: 8
 - Read Depth**: 1024 (Actual Read Depth: 1024)
- ECC, Output Register and Power Gating Options**:
 - Output Registers** is checked.
- Initialization**:
 - Reset Pin** is checked.
 - Reset Type**: Synchronous Reset
 - Full Flags Reset Value**: 0
 - Dout Reset Value**: 0 (Hex)
 - Read Latency**: 1

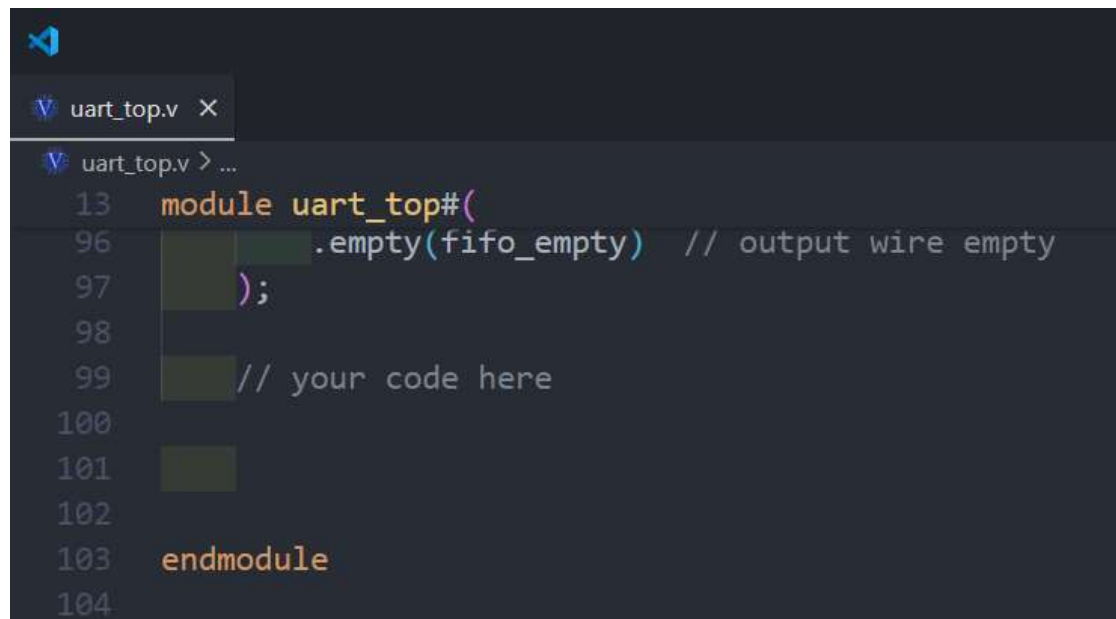
Port Diagram

The port diagram shows the following connections:

- FIFO_WRITE** (Input)
- FIFO_READ** (Output)
- clk** (Clock Input)
- srst** (Reset Input)

3. 开始编写代码

打开 uart_top.v, 定位到第 100 行, 开始编写代码。

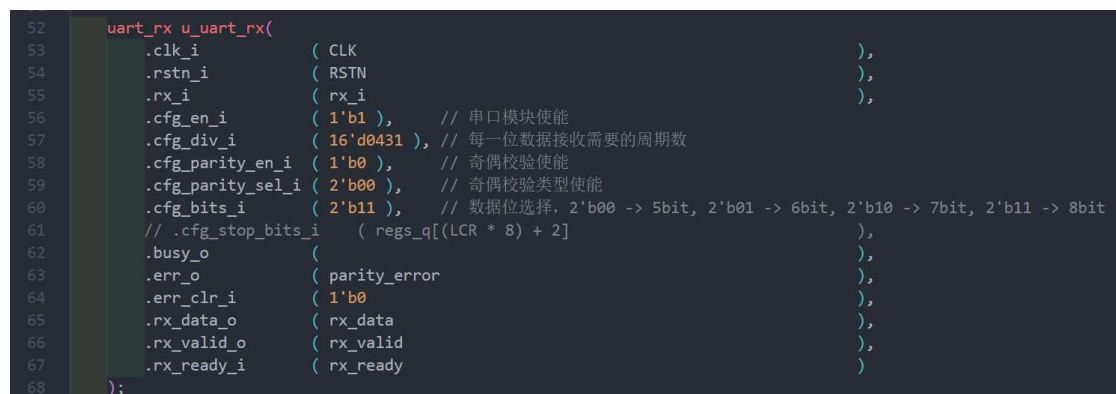


```
13 module uart_top#(
96     .empty(fifo_empty) // output wire empty
97 );
98
99 // your code here
100
101
102
103 endmodule
104
```

四、 实验原理

本章节将会给出编写回环接口所需要的时序图, 同学们可以按照时序图编写代码。

首先关注串口接收模块的端口:



```
52 uart_rx u_uart_rx(
53     .clk_i      ( CLK
54     .rstn_i     ( RSTN
55     .rx_i       ( rx_i
56     .cfg_en_i   ( 1'b1 ), // 串口模块使能
57     .cfg_div_i  ( 16'd0431 ), // 每一位数据接收需要的周期数
58     .cfg_parity_en_i ( 1'b0 ), // 奇偶校验使能
59     .cfg_parity_sel_i ( 2'b00 ), // 奇偶校验类型使能
60     .cfg_bits_i ( 2'b11 ), // 数据位选择, 2'b00 -> 5bit, 2'b01 -> 6bit, 2'b10 -> 7bit, 2'b11 -> 8bit
61     // .cfg_stop_bits_i ( regs_q[(LCR * 8) + 2]
62     .busy_o     (
63     .err_o      ( parity_error
64     .err_clr_i  ( 1'b0
65     .rx_data_o  ( rx_data
66     .rx_valid_o ( rx_valid
67     .rx_ready_i ( rx_ready
68 );
```

其中以 cfg 开头的端口为串口相关的配置, 具体数值已经写好不需要更改, 有兴趣可以自行研究。我们只需要关注如何获取从 RX 端口获得的数据, 注意最后三行, 这是一个很简单的流量控制端口, 其中 rx_data 为从 RX 上恢复的数据, rx_valid 为上游数据发送有效, rx_ready 为下游数据接收有效。当 rx_valid 与 rx_ready 同时拉高时, 表示这一拍的 rx_data 已经被下游取走。具体的时序图如下:



这里的 rx_valid 由 uart_rx 内部控制，我们需要控制的是 rx_ready。

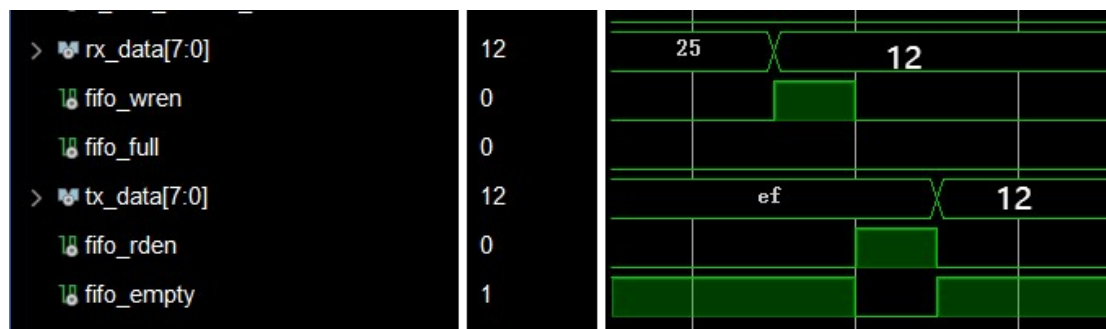
同理，串口发送模块也是一样的接口，下面给出发送模块的时序图：



需要注意，当 tx_valid 与 tx_ready 同时拉高时，uart_tx 内部才会将这一拍的 tx_data 发送出去。tx_ready 由 uart_tx 内部控制，我们只需要控制 tx_valid。

现在的实现脉络就比较明显了，我们需要控制 rx_ready 告诉 uart_rx 我们已经取走了 rx_data，把数据放到 tx_data 上，并控制 tx_valid 让 uart_tx 将数据发送出去。

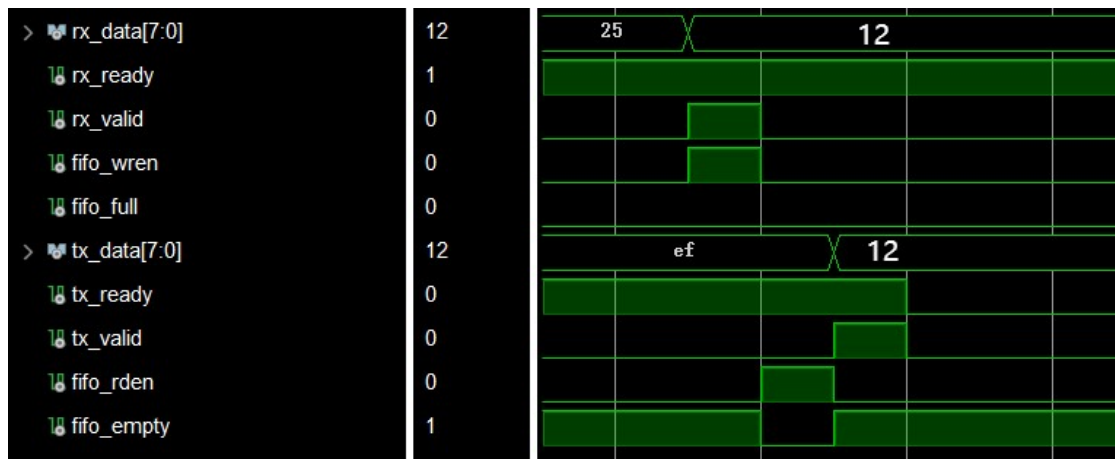
接下来我们需要用 FIFO 将 rx_data 与 tx_data 隔离开。FIFO 拥有一个写端口和一个读端口，当写端口的写使能拉高时，就会把输入数据端口的 rx_data 写入 FIFO 中；当读端口的读使能拉高后，在下一拍会将最早进入 FIFO 的数据放置在输出数据端口的 tx_data。FIFO 的接口时序图如下所示：



FIFO 还有空信号和满信号来指示当前 FIFO 的状态。简单来说，当满信号拉高后，FIFO 内就无法写入数据了；当空信号拉高时，拉高读使能也无法读出数据。因此我们可以利用空满信号来控制 rx_ready 与 tx_valid：为了能保证 rx_data 能正常写入，我们需要保证 rx_ready 拉高时 FIFO 不是满状态；而 tx_valid 拉高必然是 FIFO 处于非空状态。

注意：FIFO 读使能拉高到数据读出有一拍的间隔，而 tx_valid 与 tx_ready 同时拉高时数据就会被 uart_tx 发送出去，需要自己解决打拍问题。

最后的时序图应该与下图是一致的，具体的 rx_data 与 tx_data 由你使用串口调试助手输入的数据为准：

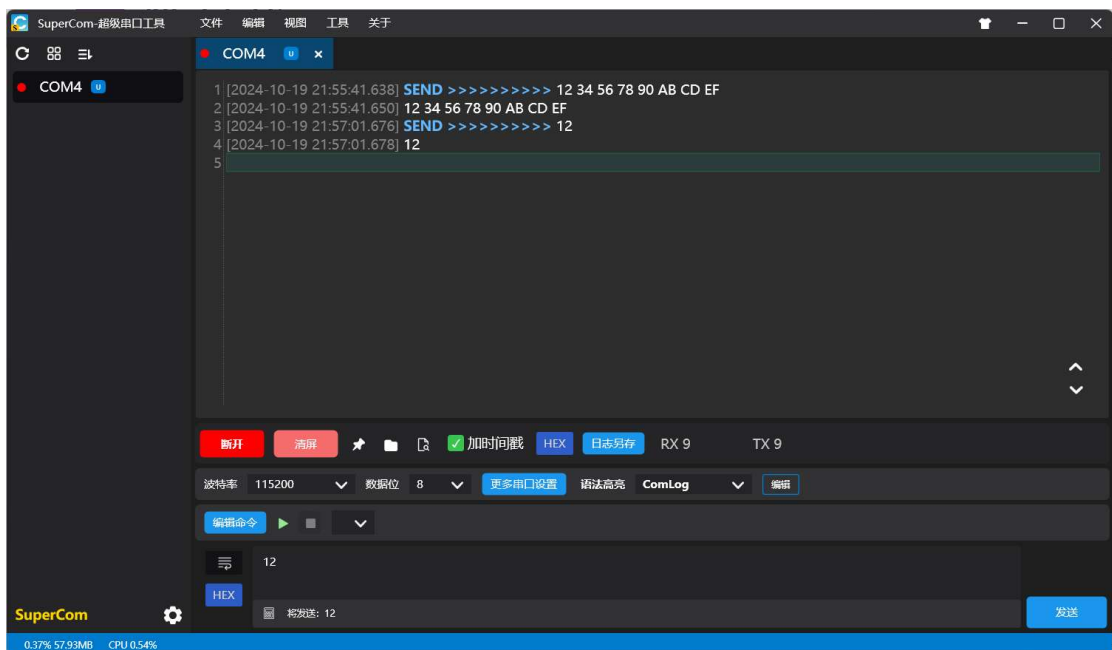


我们可以自己先编写一个 testbench，保证编写的回环部分代码波形图是正确的再进行上机测试，其中 rx_data, rx_valid, tx_ready 可以根据上面的波形图在 tb 中进行控制。Testbench 样例你可以在附录 1 中找到。

使用串口调试助手时需要注意，发送数据和接收数据请务必选择 HEX(16 进制)模式，否则发送的数据会被转为 ASCII 码发送，同时波特率需要选择 115200，数据位 8 位，1 位停止位，无奇偶校验。

如果是第一次接触串口调试助手，可以前往附录 2 学习具体的操作方法。

注意：你需要控制的信号只有 tx_data, tx_valid, tx_ready, rx_data, rx_valid, rx_ready, fifo_wren, fifo_full, fifo_rden, fifo_empty, 其他信号都不应该出现在你编写的代码里(除了时钟与复位信号)!



五、实验报告要求

1. 根据实验要求编写环回代码，并编写 testbench 验证功能正确性，附上仿真波形图；
2. 将代码烧录上板，使用串口调试助手分别发送 "1234567890ABCDEF" 和你的学号这两

条字符串，查看回传的数据是否一直。将整个工具的界面截图附在实验报告中。如果功能正确，串口调试助手中显示的数据应与下图类似，其中第二条“23111111”为你的学号：

```
1 [2024-10-21 01:25:37.253] SEND >>>>>>>>> 12 34 56 78 90 AB CD EF
2 [2024-10-21 01:25:37.256] 12 34 56 78 90 AB CD EF
3 [2024-10-21 01:26:04.109] SEND >>>>>>>>> 23 11 11 11
4 [2024-10-21 01:26:04.111] 23 11 11 11
5
```

“1234567890ABCDEF”的空格可不添加，本处仅为美观。

附录 1 Testbench 样例

下面给出本次实验需要使用的 testbench 模板，你需要在 your code here 处开始编写你的仿真代码来模拟串口模块行为，并在 loop code here 复制粘贴你的环回部分代码，从而测试你的环回模块。只要你的仿真波形与第四章中的波形图一致，则环回接口一般都能正常运行。

```
`timescale 1ns / 1ps

module uart_loop_tb;

    // Internal signals
    parameter clk_freq = 10;

    reg        clk;
    reg        rst_n;

    reg        tx_release;

    reg  [7:0]  rx_data;
    reg        rx_valid;
    wire       rx_ready;
    wire  [7:0] tx_data;
    reg        tx_valid;
    reg        tx_ready;

    wire       fifo_wren;
    wire       fifo_rden;
    wire       fifo_full;
    wire       fifo_empty;

    // Clock process to generate clock signals
    always begin
        # (clk_freq / 2) clk = ~clk;
    end

    // Reset signals
    initial begin
        clk = 1'b1;
        rst_n = 1'b0;
        #50 rst_n = 1'b1;
    end
end
```



```

// DUT instantiation
fifo_8b u_fifo_8b (
    .clk(clk),    // input wire clk
    .rst(~rst_n), // input wire rst
    .din(rx_data), // input wire [7 : 0] din
    .wr_en(fifo_wren), // input wire wr_en
    .rd_en(fifo_rden), // input wire rd_en
    .dout(tx_data), // output wire [7 : 0] dout
    .full(fifo_full), // output wire full
    .empty(fifo_empty) // output wire empty
);

// simulation code
initial begin
    rx_data = 8'h25;
    rx_valid = 1'b0;
    tx_ready = 1'b1;
    tx_release = 1'b0;
    // oper after rst release
    #80 tx_release = 1'b0;

    // your code here

    // your code end

    // this signal used for release tx_ready
    #80 tx_release = 1'b1;
    #10 tx_release = 1'b0;
end

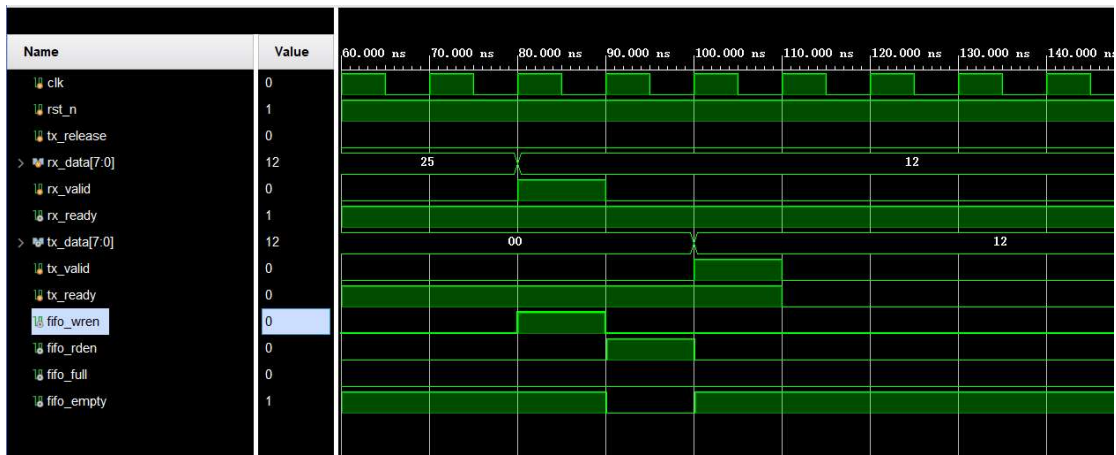
always @(posedge clk) begin
    if(tx_valid)
        tx_ready <= 1'b0;
    else if (tx_release)
        tx_ready <= 1'b1;
end

// loop code here

endmodule

```

同时给出正确的仿真波形图：



如果发现在仿真起始时有一个时钟周期的 X 状态，这是 FIFO 处于复位状态导致的，属于正常情况。但如果存在较长的 X 状态就是代码编写有问题。

附录 2 串口调试助手使用方法

本章节为第一次接触串口调试助手的同学提供教学，使用的软件为正点原子出品的 ATK-XCOM，该软件经过大量市场检验，工作稳定。我们不强制要求使用哪款串口调试工具，如果有自己习惯的工具也可以使用。

ATK-XCOM 下载地址：<http://www.openedv.com/docs/tool/ruanjian/ATK-XCOM.html>

由于目前个人 PC 已经取消了 UART 接口，我们只能使用 USB 转换为 UART，所需要的驱动下载地址为：https://www.wch.cn/downloads/CH341SER_EXE.html

启动后界面如下所示



我们需要做的是修改波特率并打开串口，启动 16 进制发送显示，启动时间戳，关闭发送新行，修改后界面如下：



接下来鼠标点到下方的输入框中，输入测试数据“1234”，点击发送。如果环回功能正常，我们可以在上方的显示框中看到回传的 1234。

