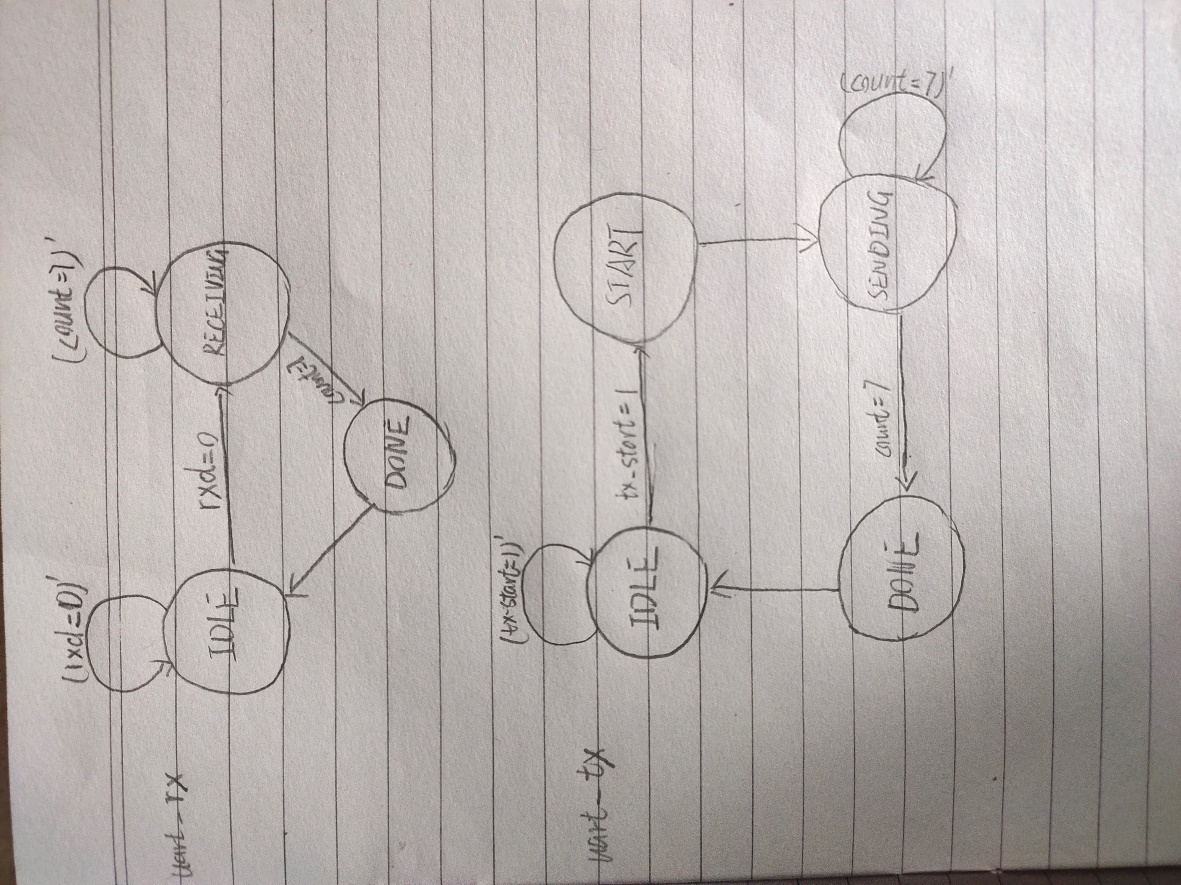


**1）The state diagram of the transmission module and receiving module**

****

**2) the corresponding Verilog code.**

**传输模块：**

module uart\_tx (

    input clk,

    input [7:0] tx\_data,

    input tx\_start,

    output tx\_busy,

    output reg txd

);

    localparam IDLE = 0;

    localparam START = 1;

    localparam SENDING = 2;

    localparam DONE = 3;

    reg [2:0] state=IDLE, next\_state=IDLE;

    reg [2:0] count;

    always @(posedge clk ) begin

        state <= next\_state;

    end

    always @(\*) begin

        next\_state = state;

        case (state)

            IDLE: begin

                if (tx\_start) begin

                    next\_state = START;

                end

            end

            START: begin

                next\_state = SENDING;

            end

            SENDING: begin

                if (count == 7) begin

                    next\_state = DONE;

                end else begin

                    next\_state = SENDING;

                end

            end

            DONE: begin

                next\_state = IDLE;

            end

        endcase

    end

    always @(posedge clk ) begin

        if (state == SENDING) begin

            count <= count + 1;

        end else if (state == DONE||state == IDLE) begin

            count <= 0;

        end

        end

    always @(posedge clk ) begin

        if (state == START) begin

            txd <= 0;

        end else if (state == SENDING) begin

            txd <= tx\_data[count];

        end else if (state == DONE|state==IDLE) begin

            txd <= 1;

        end

        end

    assign tx\_busy = (state == SENDING);

endmodule

**接收模块：**

module uart\_rx (

    input clk,

    input rxd,

    output  data\_rdy,

    output  [7:0] data

);

localparam IDLE = 0;

localparam RECEIVING = 1;

localparam DONE = 2;

reg [2:0] state=0, next\_state=0;

reg [2:0] count;

reg [7:0] rx\_data=0;

reg rx\_data\_rdy;

always @(posedge clk ) begin

    state <= next\_state;

end

always @(\*) begin

    next\_state = state;

    case (state)

        IDLE: begin

            if (rxd == 0) begin

                next\_state = RECEIVING;

            end

        end

        RECEIVING: begin

            if (count == 7) begin

                next\_state = DONE;

            end else begin

                next\_state = RECEIVING;

            end

        end

        DONE: begin

            next\_state = IDLE;

        end

    endcase

end

always @(posedge clk ) begin

    if (state == IDLE) begin

        count <= 0;

    end else if (state == RECEIVING) begin

        count <= count + 1;

    end else if (state == DONE) begin

        count <= 0;

    end

end

always @(posedge clk ) begin

    if (state == RECEIVING) begin

        rx\_data[count] <= rxd;

    end

    end

assign data\_rdy = (state == DONE);  //

assign data = rx\_data;  //

endmodule

顶层模块：

`include "uart\_tx.v"

`include "uart\_rx.v"

module top (

    input clk,

    input rxd,

    output txd

);

    wire [7:0] data;

    wire dete\_busy;

    uart\_tx  u\_uart\_tx (

        .clk                     ( clk             ),

        .tx\_data                 ( data      [7:0] ),

        .tx\_start                ( dete\_rdy        ),

        .tx\_busy                 ( dete\_busy       ),

        .txd                     ( txd             )

    );

    uart\_rx  u\_uart\_rx (

        .clk                     ( clk             ),

        .rxd                     ( rxd             ),

        .data\_rdy                ( dete\_rdy        ),

        .data                    ( data      [7:0] )

    );

endmodule

**3）Simulate the transmission module to see check the timing of the output TxD signal is correct.**

**仿真文件代码：**

//~ `New testbench

`timescale  1ns / 1ps

`include "uart\_tx.v"

module tb\_uart\_tx;

// uart\_tx Parameters

parameter PERIOD  = 10;

// uart\_tx Inputs

reg   clk                                  = 0 ;

reg   [7:0]  tx\_data                       = 0 ;

reg   tx\_start                             = 0 ;

// uart\_tx Outputs

wire  tx\_busy                              ;

wire  txd                                  ;

initial

begin

    forever #(PERIOD/2)  clk=~clk;

end

uart\_tx  u\_uart\_tx (

    .clk                     ( clk             ),

    .tx\_data                 ( tx\_data   [7:0] ),

    .tx\_start                ( tx\_start        ),

    .tx\_busy                 ( tx\_busy         ),

    .txd                     ( txd             )

);

initial

begin

$dumpfile("tb\_uart\_tx.vcd");

$dumpvars(0,tb\_uart\_tx);

tx\_data = 8'b10011000;

tx\_start = 1;

#20

tx\_start = 0;

#200;

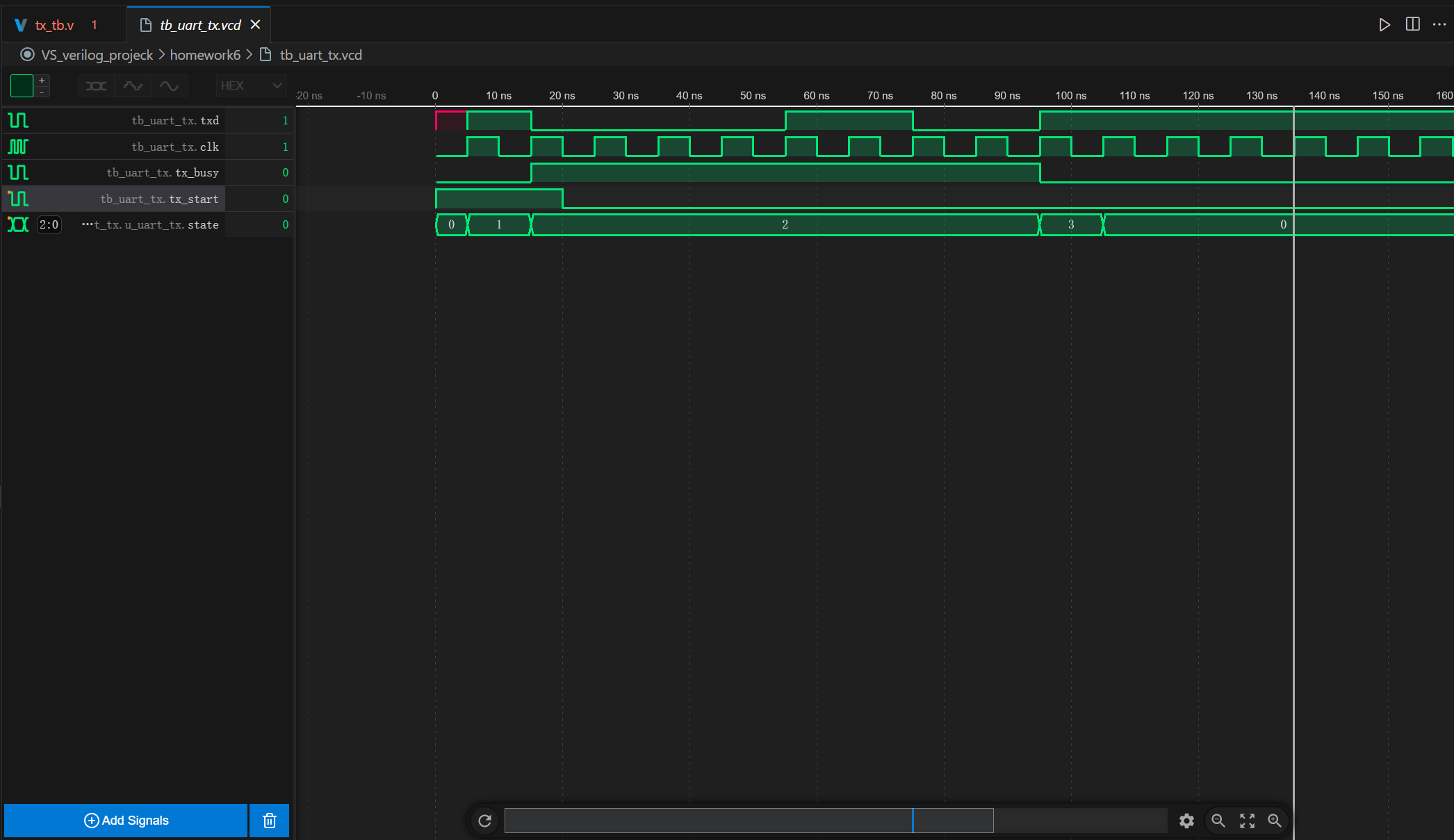
    $finish;

end

endmodule

**仿真波形如下：**

**结果正确，传输模块功能正常**

****

**4）Connect the transmission module to the receiving module to test whether the receiving module can work correctly. Copy and draw the simulation results to show the correctness of your design.**

**仿真代码如下：**

//~ `New testbench

`include "top.v"

`timescale  1ns / 1ps

module tb\_top;

// top Parameters

parameter PERIOD  = 10;

// top Inputs

reg   clk                                  = 0 ;

reg   rxd                                  = 0 ;

// top Outputs

wire  txd                                  ;

initial

begin

    forever #(PERIOD/2)  clk=~clk;

end

top  u\_top (

    .clk                     ( clk   ),

    .rxd                     ( rxd   ),

    .txd                     ( txd   )

);

initial

begin

     $dumpfile("tb\_top.vcd");

        $dumpvars(0,tb\_top);

    rxd = 1;

    #100

    rxd = 0;

    #10

    rxd = 1;

    #10

    rxd = 0;

    #10

    rxd = 1;

    #10

    rxd = 0;

    #10

    rxd = 0;

    #10

    rxd = 1;

    #10

    rxd = 1;

    #10

    rxd = 0;

    #10

    rxd = 1;

    #200

    $finish;

end

endmodule

**仿真测试波形如下：**

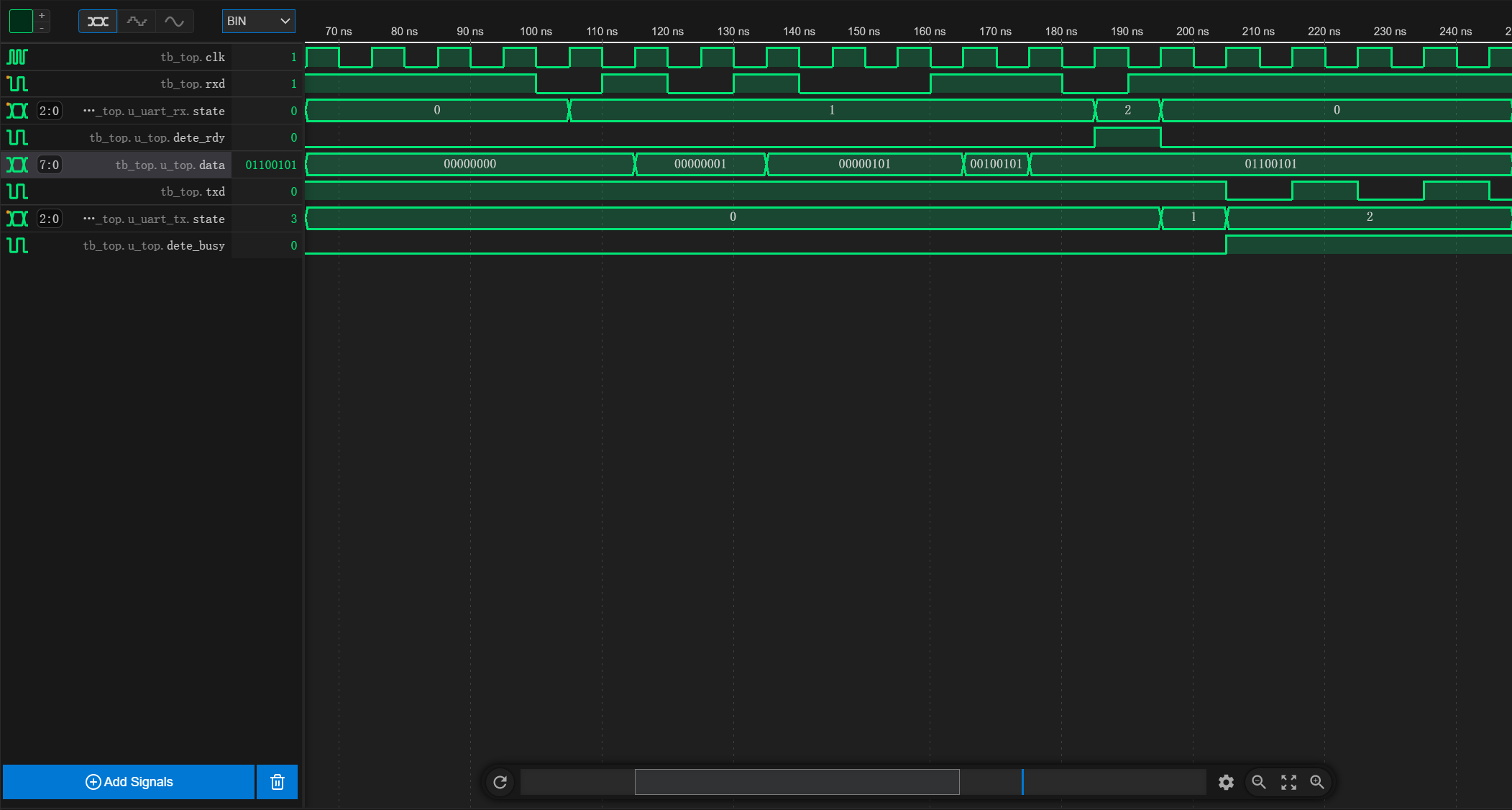
**首先，仿真输入10表示信号开始接受，随后输入10100110 最后输入1表示结束。**

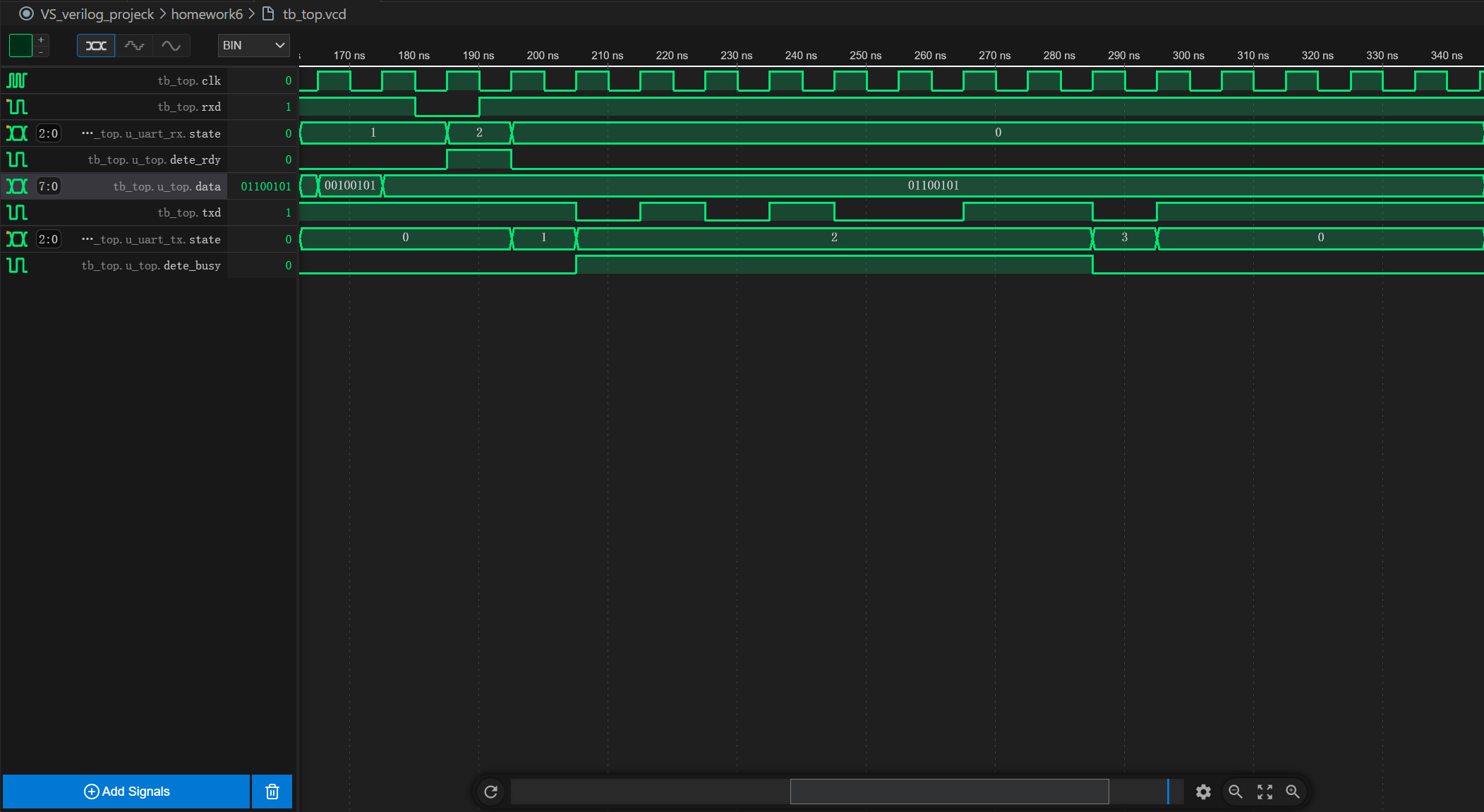
**从tx模块仿真信号看，rxd接受数据正常，数据成功从低到高位存入寄存器data中。**

**随后data\_rdy信号拉高，传输模块开始按顺序传输数据，txd数据传输正常。**

**从tx和rx的state信号看，状态机状态跳转正常。**

**综上， uart\_tx 和 uart\_tx 模块功能正常。**

****

****