同济大学计算机系

数字逻辑课程综合实验报告

学　　号 1850000

姓　　名 一二三

专　　业 计算机类

授课老师 郭玉臣

一、实验内容

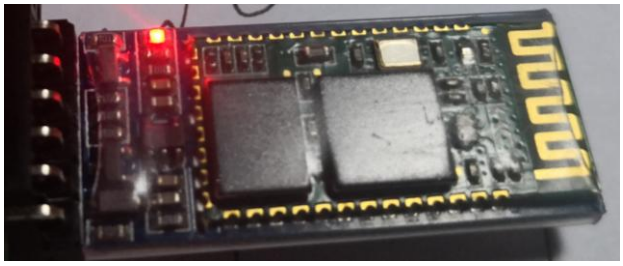
1. 项目内容概括

本项目主体是基于蓝牙模块通讯搭建起的遥控小车，在遥控端辅助以 OLED 显示屏与旋钮方便遥控操作，车载端则通过对发动机、舵机、测距仪等部件的控制实现车体的运动与测距。

2. 遥控操作说明

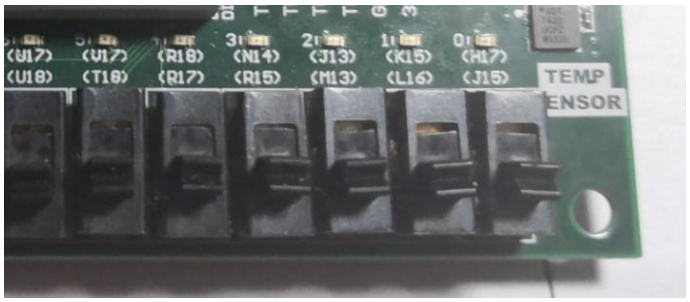
待蓝牙连接完成后，使用遥控端 FPGA 开发板右下侧的开关控制车体的前进、后退与其运动速度的控制；使用旋钮控制车体的左右旋转；同时可以通过 OLED 显示屏观察当前车体的旋转角度与前方障碍的距离。

蓝牙模块指示灯说明如下：



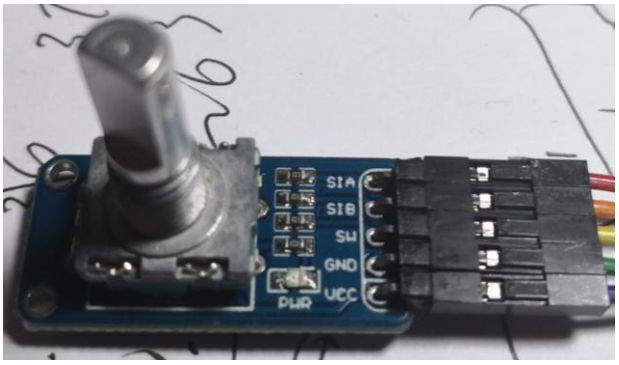
指示灯情况	说明
红色灯间断闪烁	尚未连接到遥控车，无法遥控
红色灯常亮	蓝牙已连接，可以遥控

开关控制按键说明如下：



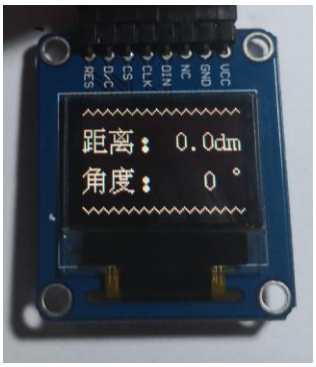
开关编号	用途	详细说明
J15	控制车体前进后退	置 1 表示前进 置 0 表示后退
R17、R15、M13、L16	控制车体移动速度	置 1 表示速度二进制编码中此位置 1； R17 为最高位，依次降低

旋钮控制说明如下：



旋钮操作	用途
旋钮顺时针转动一个档位	车头向右偏转 15°，最多偏至右 45°
旋钮逆时针转动一个档位	车头向左偏转 15°，最多偏至左 45°
按下按钮	角度清零，立刻摆正车头

OLED 显示说明如下：



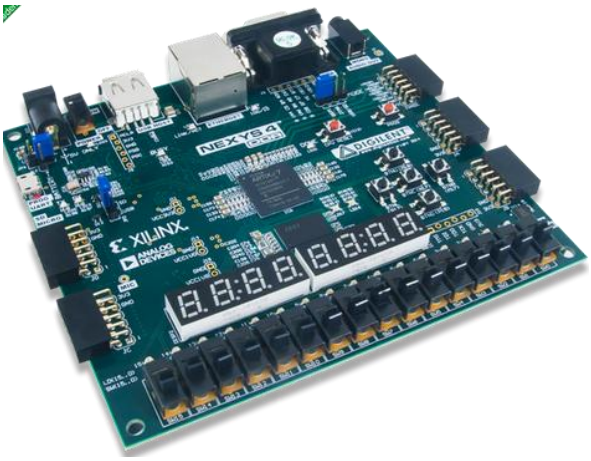
显示内容	详细说明
距离	显示遥控车距离前方障碍物的距离，以 0.1dm 为最小显示距离； 蓝牙未接通时常显 0.0dm
角度	显示遥控车车头的摆向角度，以 15°为最小显示角度； 蓝牙未接通时常显 0°

3. 使用器件说明

1）开发板扩展模块：

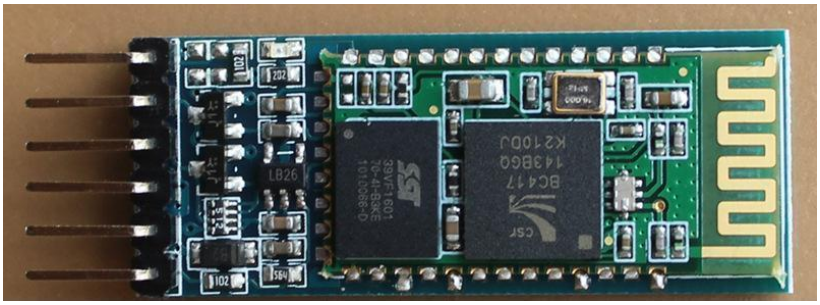
①Nexys 4 DDR Artix-7 FPGA Trainer Board：

由 Xilinx 公司开发出的一款可编程门阵列（FPGA）开发板。在本项目中用于遥控端和车载端的主体控制。



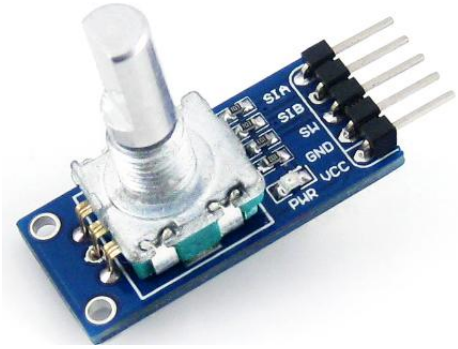
②JY-MCU HC-06 从机：

电源防反接，无线蓝牙串口透传模块，用于 HC-06 从机。在本项目中用于遥控端和车载端的双向通信。



③Rotation Sensor：

旋转编码器，可检测顺时针和逆时针旋转角度及旋转圈数，多用于工业中定位检测。在本项目中作为旋钮，用于控制车体的旋转角度。



④0.95inch RGB OLED：

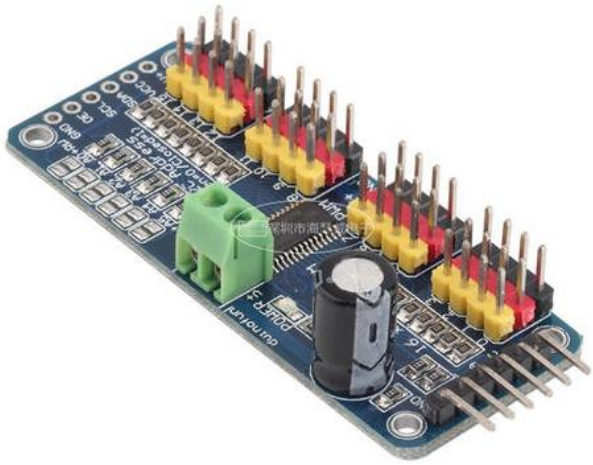
0.95 寸 RGB OLED 显示屏，由 SSD1331 驱动，65K 彩色，分辨率 96×64。在本项目中用于车辆情况的显示说明

⑤HC-SR04：

车载测距模块，采用 IO 口 TRIG 触发测距，使用较为简单。在本项目中用于在车载端监测距离前方障碍物的距离。

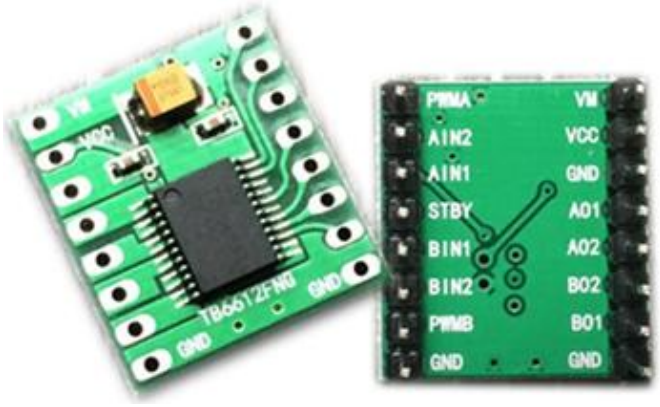
⑥PCA9685：

16 路 PWM 模块，使用 I2C 总线可以控制 16 路舵机。在本项目中用于车载端两个后置电机与前置舵机的驱动控制。



⑦TB6612FNG：

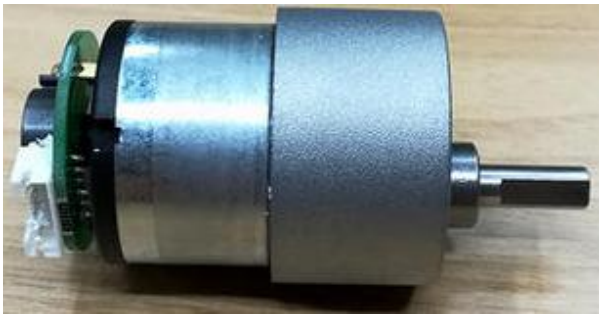
TB6612FNG 是东芝半导体公司生产的一块直流电机驱动器件，具有大电流 MOSFET-H 桥结构，双通道电路输出，可同时驱动 2 个电机。在本项目中用于两个后置轮的发动机驱动控制。



2）机械传动与外部供电部件：

①电机

常规重载 30 减速比电机，转速 500rpm，转速比 1:30，供电电压 7-13V。在本项目中用于两个后置轮的机械驱动。



②DS3119

数字舵机，舵机扭矩 20kg，供电电压 5V-7.4V，空载 100mA，堵转 2A。在本项目中用于前置轮的转向驱动。

③格氏 ace 2200 20C mAh 14.8V 航模电池

提供 14.8V 电压，放电倍率为 20C，在本项目中用于两个后置轮的驱动电力

提供。

④南孚 LR6 1.5V 电池

提供 4*1.5V 电压。在本项目中用于数字舵机的驱动电力提供。

⑤ROMOSS sense6 PH80 移动电源

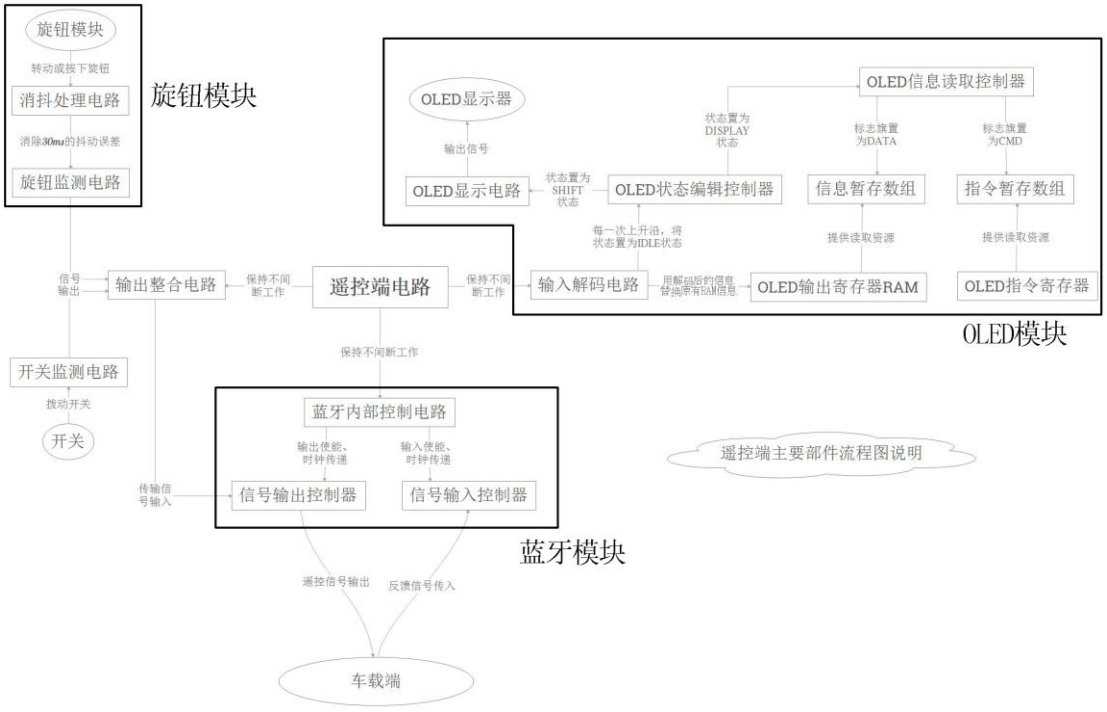
在本项目中，和绿联拓展坞一同为下板后的车载端 FPGA 供电。

二、框图说明

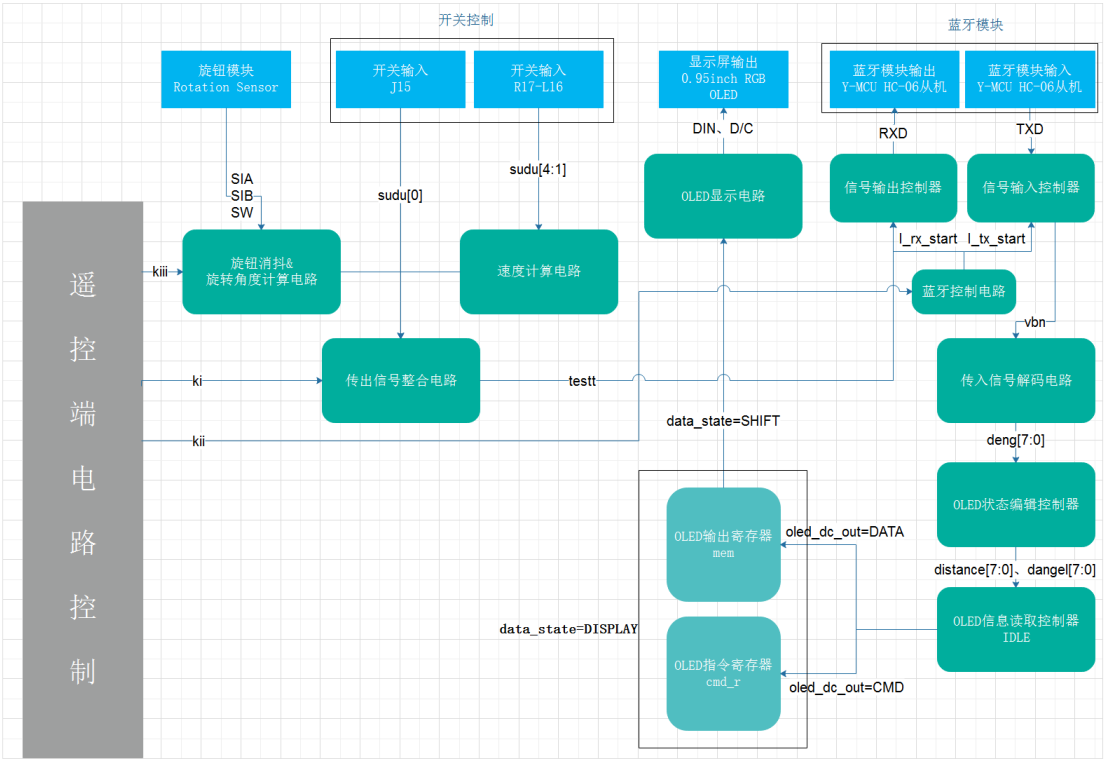
1）遥控端框图说明

遥控端的主要作用是将用户从旋转编码器与开关输入的，对遥控车的控制信号进行规整与编码，通过蓝牙模块传输到车载端 FPGA，实现控制；同时还要将从车载端接受到的反馈信息进行解码，获得车载端的当前角度和与前方障碍物的距离信息，并显示在 OLED 上。

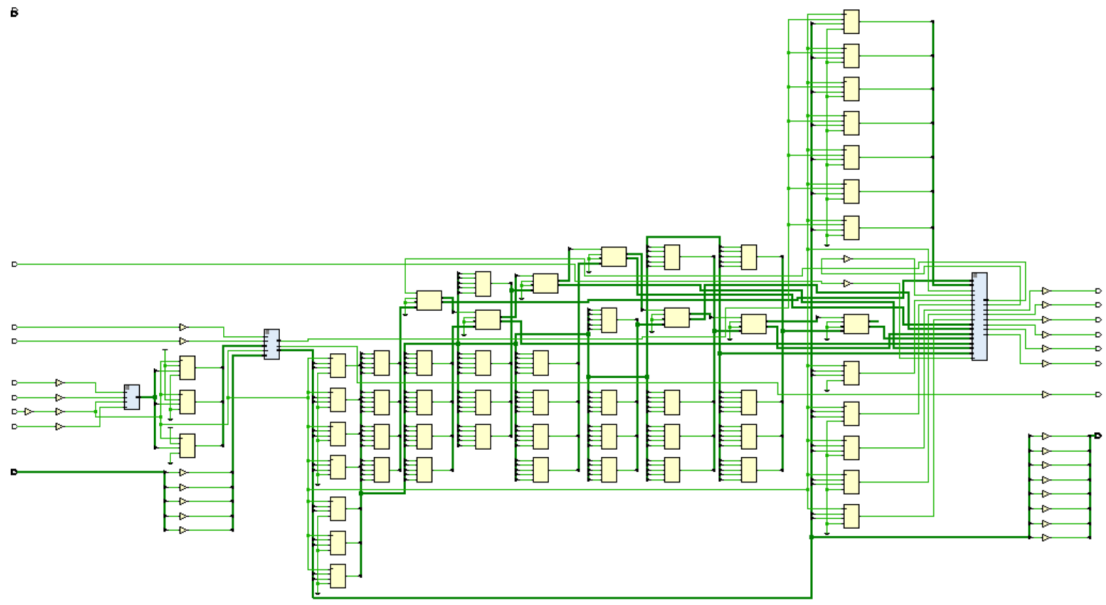
汇总起来，遥控端的流程图为：



其总框图为：



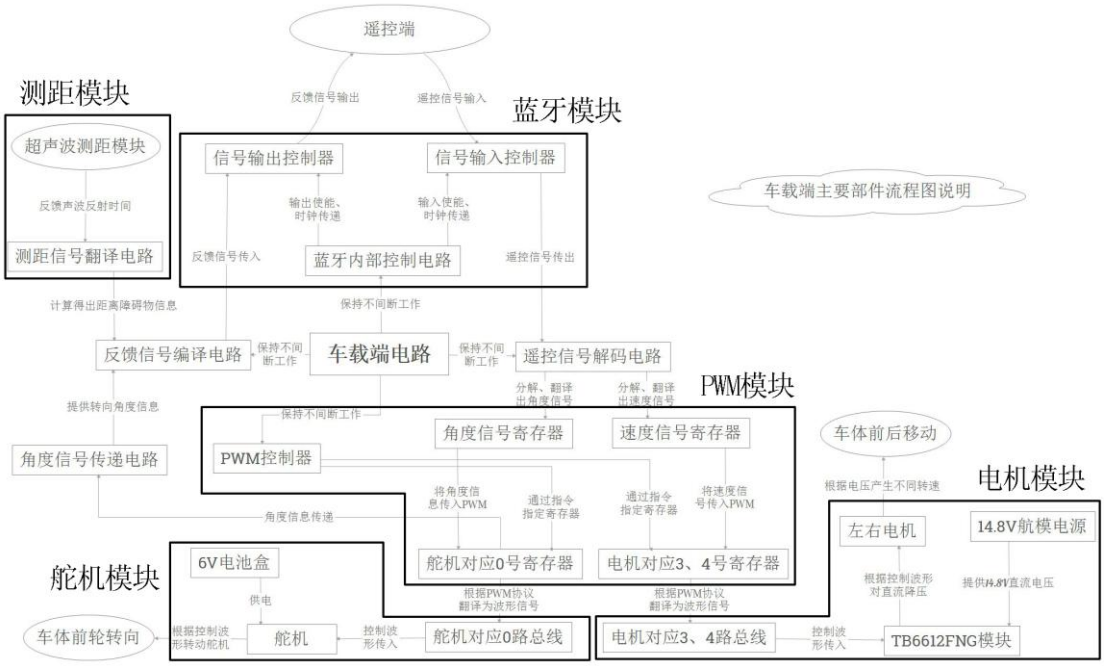
其 RTL 片图为：



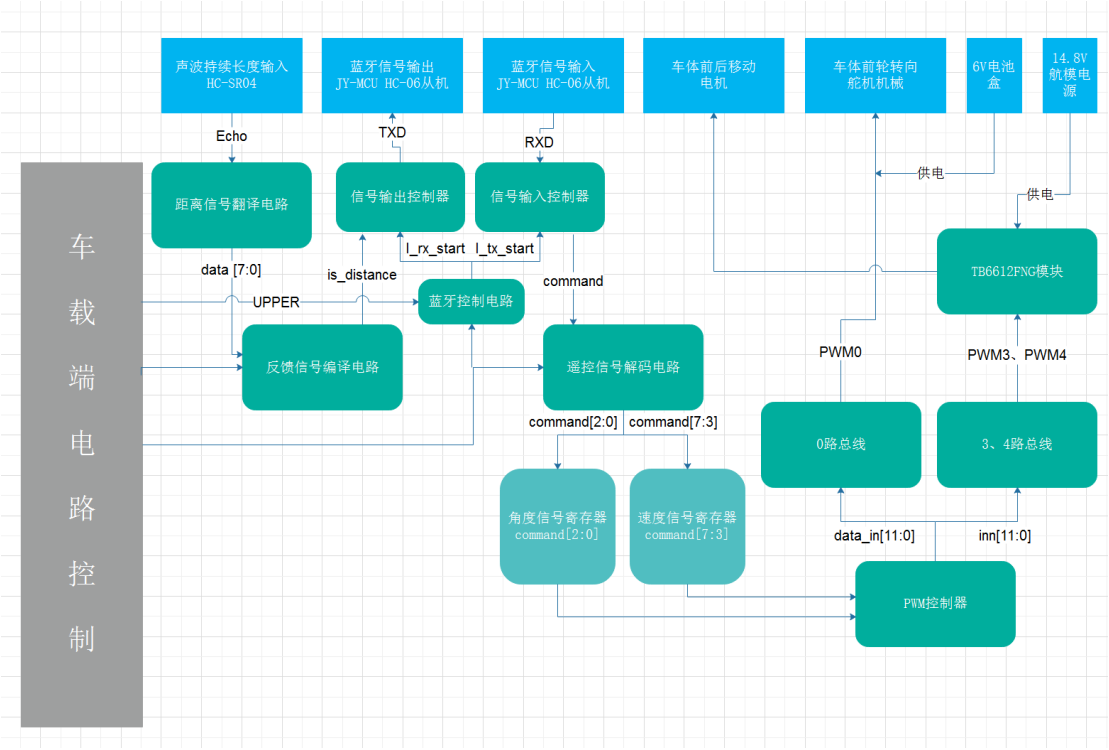
2）车载端

车载端的主要作用是将从蓝牙模块接受到的控制信号进行解码，并将相关控制指令下发到 PWM 模块，实现对电机与舵机的控制；同时从测距模块获得距离前方障碍物的距离信息，以及计算出当前前轮的偏转角度与方向，将这两组信息进行编码为反馈信号，从蓝牙模块反馈回遥控端。

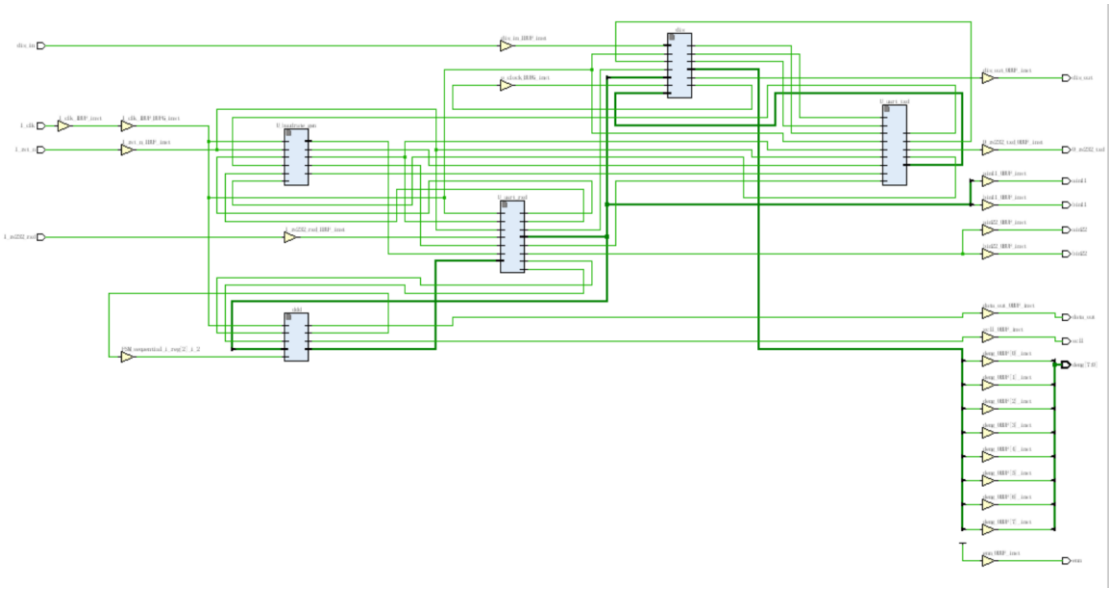
汇总起来，车载端的流程图为：



其总框图为：



其 RTL 片图为：

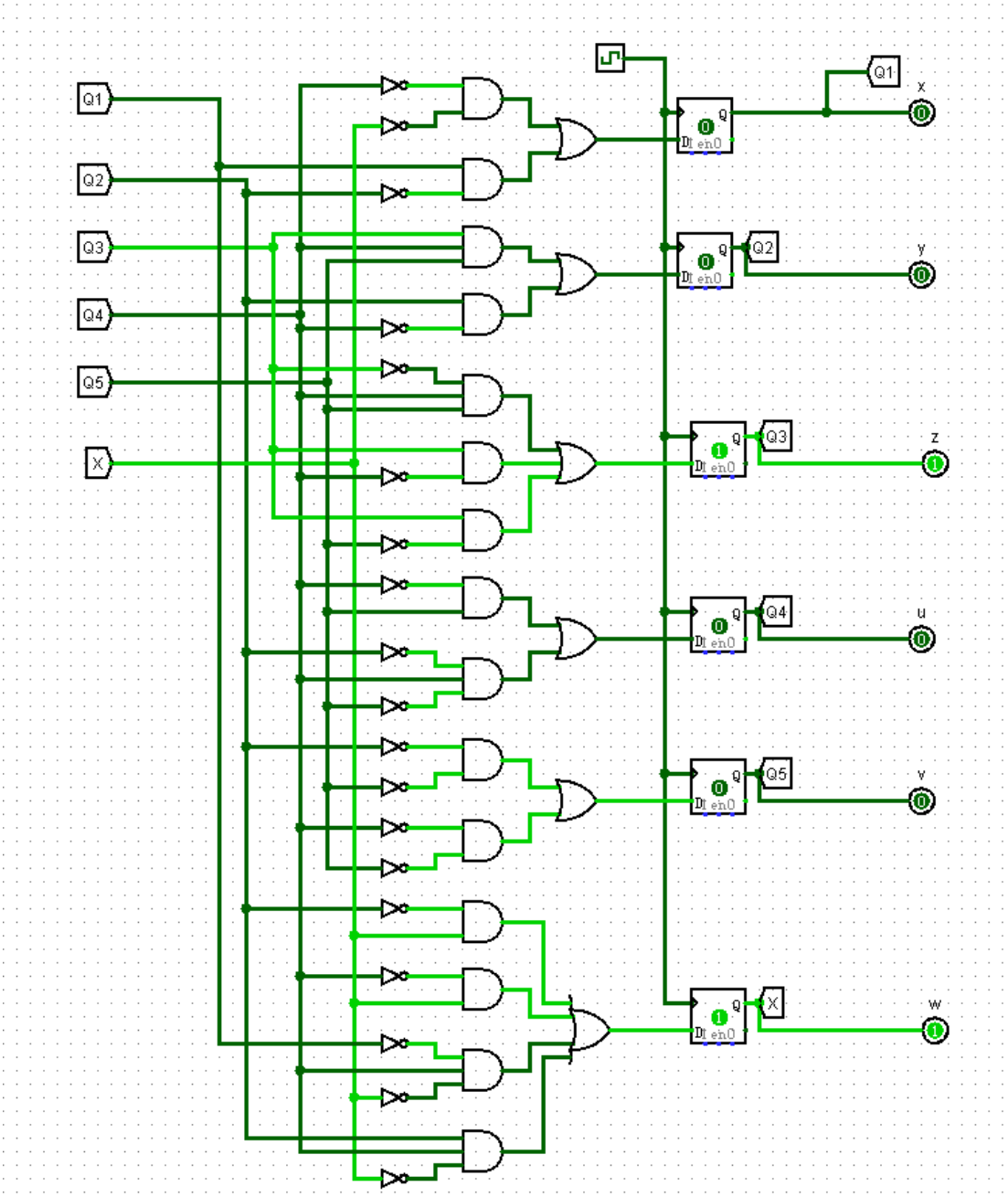


三、系统控制器设计

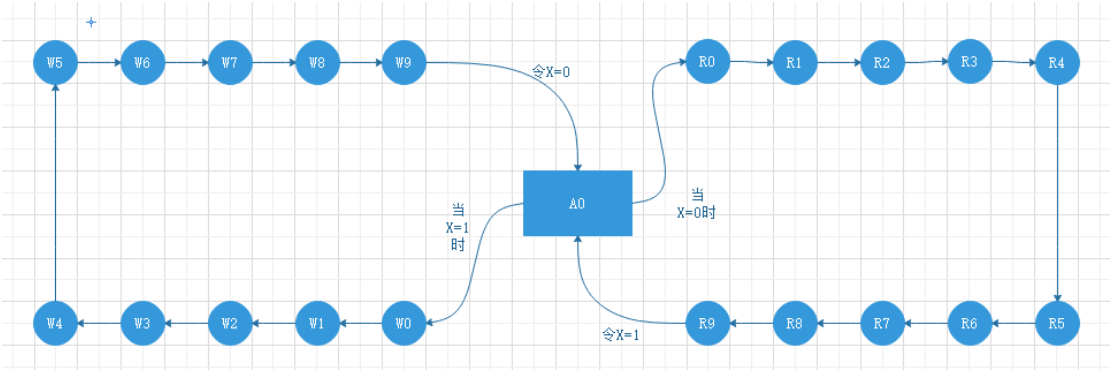
1）蓝牙发送模块的状态机说明：

现态						次态					
Q1	Q2	Q3	Q4	Q5	X	Q1	Q2	Q3	Q4	Q5	X
0	0	0	0	0	1	0	0	0	0	1	1
0	0	0	0	1	1	0	0	0	1	0	1
0	0	0	1	0	1	0	0	0	1	1	1
0	0	0	1	1	1	0	0	1	0	0	1
0	0	1	0	0	1	0	0	1	0	1	1
0	0	1	0	1	1	0	0	1	1	0	1
0	0	1	1	0	1	0	0	1	1	1	1
0	0	1	1	1	1	0	1	0	0	0	1
0	1	0	0	0	1	0	1	0	0	1	1
0	1	0	0	1	1	0	1	0	1	0	1
0	1	0	1	0	1	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	1	0
1	0	0	0	1	0	1	0	0	1	0	0
1	0	0	1	0	0	1	0	0	1	1	0
1	0	0	1	1	0	1	0	1	0	0	0
1	0	1	0	0	0	1	0	1	0	1	0
1	0	1	0	1	0	1	0	1	1	0	0
1	0	1	1	0	0	1	0	1	1	1	0
1	0	1	1	1	0	1	1	0	0	0	0
1	1	0	0	0	0	1	1	0	0	1	0
1	1	0	0	1	0	1	1	0	1	0	0
1	1	0	1	0	0	0	0	0	0	0	1

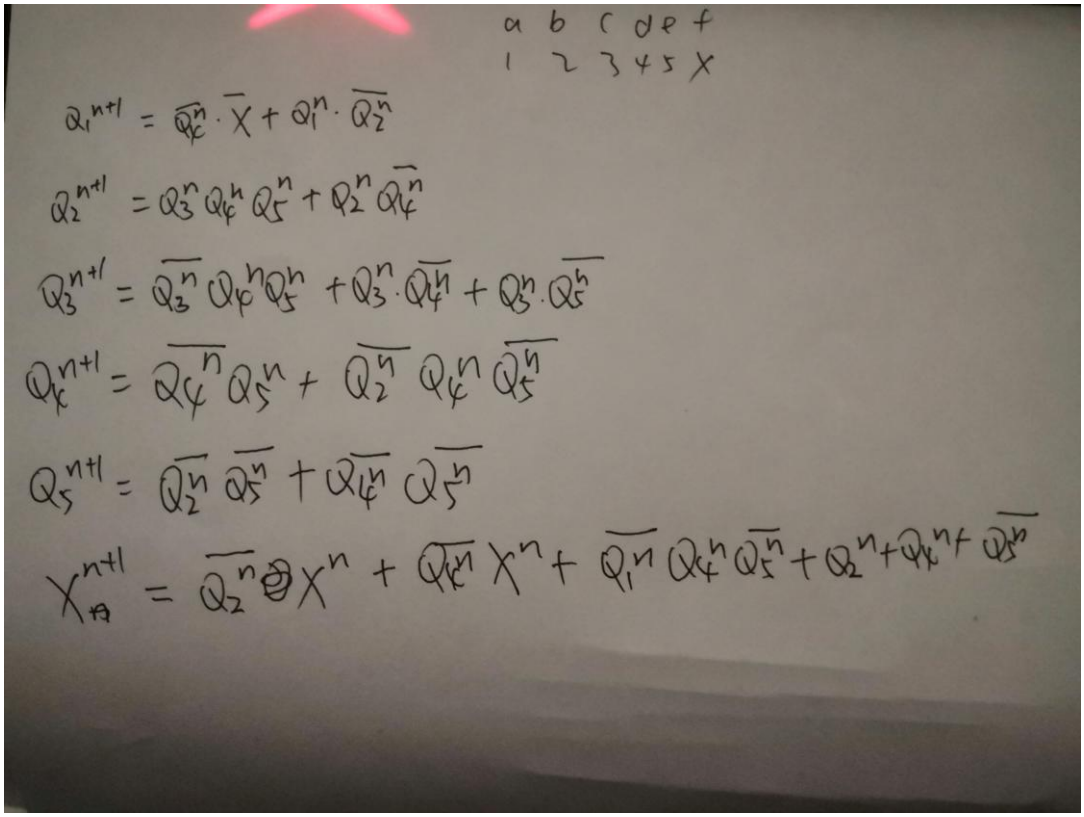
Logicsim 图:



状态转移图:



逻辑表达式：



四、子系统模块建模

1）遥控端子系统说明

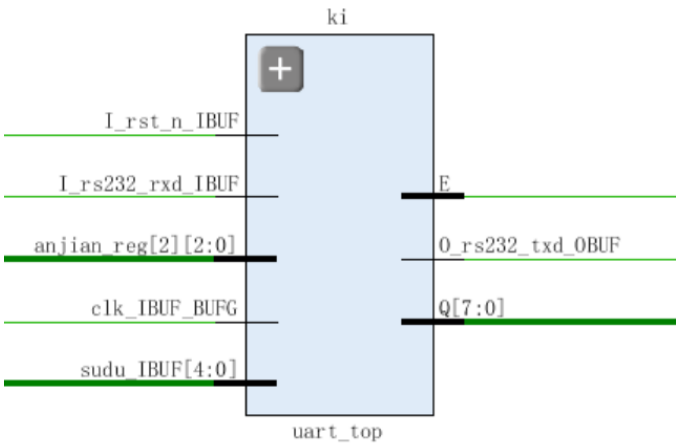
本项目中，遥控端共含有 3 个二级模块、5 个三级模块共计 8 个子模块，以下将逐一对这 8 个子模块进行说明：

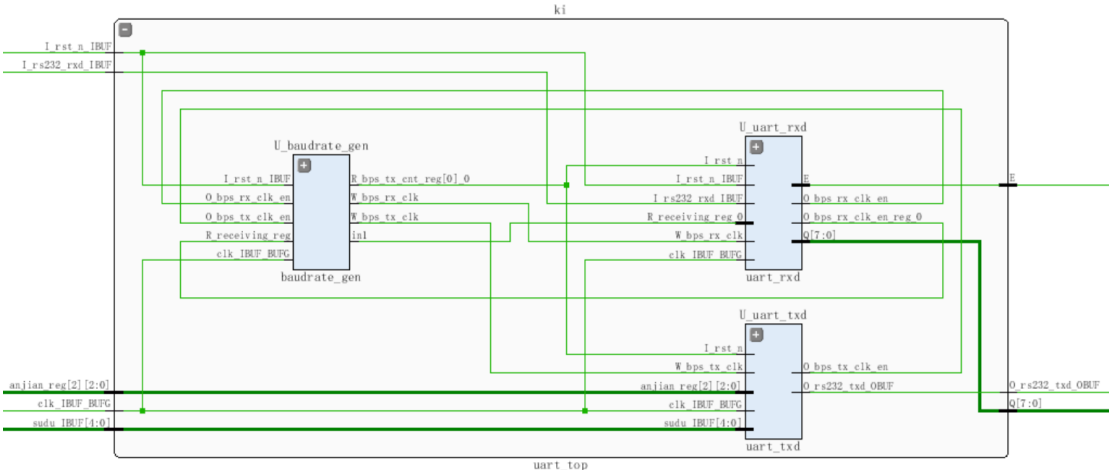
① uart_top 模块

1. 模块功能描述

该模块是蓝牙模块功能实现的顶层模块，其作用是通过实例化三个蓝牙功能三级模块的方式，实现两个蓝牙模块间互传信息的功能。

2. 功能框图





3. 接口信号说明

管脚名称	属性	具体描述
I_clk	input	输入 100MHz 的系统时钟
I_rst_n	input	用于系统的全局复位
I_rs232_rxd	input	接受要输入的串行数据
I_rs232_txd	output	发送要输出的串行数据
anjian	input [7:0]	以并行方式寄存要输出的数据
deng	output [7:0]	以并行方式寄存要输入的数据

4. verilog 实现

```
module uart_top
(
    input           I_clk          , //  系统 100MHz 时钟
    input           I_rst_n        , //  系统全局复位
    input           I_rs232_rxd    , //  接收的串行数据，在硬件上与串口相连
    output          O_rs232_txd    , //  发送的串行数据，在硬件上与串口相连
    input           [7:0]anjian,
    output          [7:0]deng
);

wire            W_bps_tx_clk                ;
wire            W_bps_tx_clk_en             ;
wire            W_bps_rx_clk                ;
wire            W_bps_rx_clk_en             ;
wire            W_rx_done                   ;
wire            W_tx_done                   ;
wire    [7:0]   W_para_data                 ;


baudrate_gen U_baudrate_gen
(
    .I_clk              (I_clk              ), //  系统 100MHz 时钟
    .I_rst_n            (I_rst_n            ), //  系统全局复位
    .I_bps_tx_clk_en    (W_bps_tx_clk_en    ), //  串口发送模块波特率时钟使能
信号
```

```
    .I_bps_rx_clk_en    (W_bps_rx_clk_en    ), // 串口接收模块波特率时钟使能
信号
    .O_bps_tx_clk       (W_bps_tx_clk       ), // 发送模块波特率产生时钟
    .O_bps_rx_clk       (W_bps_rx_clk       )  // 接收模块波特率产生时钟
);

uart_txd U_uart_txd
(
    .I_clk              (I_clk              ), // 系统 100MHz 时钟
    .I_rst_n            (I_rst_n            ), // 系统全局复位
    .I_tx_start         (W_rx_done          ), // 发送使能信号
    .I_bps_tx_clk       (W_bps_tx_clk       ), // 波特率时钟
    .I_para_data        (anjian             ), // 要发送的并行数据
    .O_rs232_txd        (O_rs232_txd        ), // 发送的串行数据，在硬件
上与串口相连
    .O_bps_tx_clk_en    (W_bps_tx_clk_en    ), // 波特率时钟使能信号
    .O_tx_done          (W_tx_done          )  // 发送完成的标志
);

uart_rxd U_uart_rxd
(
    .I_clk              (I_clk              ), // 系统 100MHz 时钟
    .I_rst_n            (I_rst_n            ), // 系统全局复位
    .I_rx_start         (1'b1               ), // 接收使能信号
    .I_bps_rx_clk       (W_bps_rx_clk       ), // 接收波特率时钟
    .I_rs232_rxd        (I_rs232_rxd        ), // 接收的串行数据，在硬件上与
串口相连
    .O_bps_rx_clk_en    (W_bps_rx_clk_en    ), // 波特率时钟使能信号
    .O_rx_done          (W_rx_done          ), // 接收完成标志
    .O_para_data        (deng               )  // 接收到的 8-bit 并行数据
);
endmodule
```

② baudrate_gen 模块

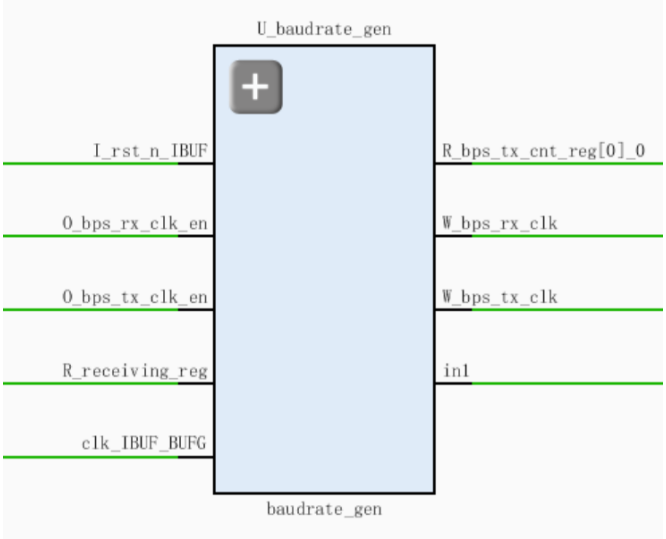
1. 模块功能描述

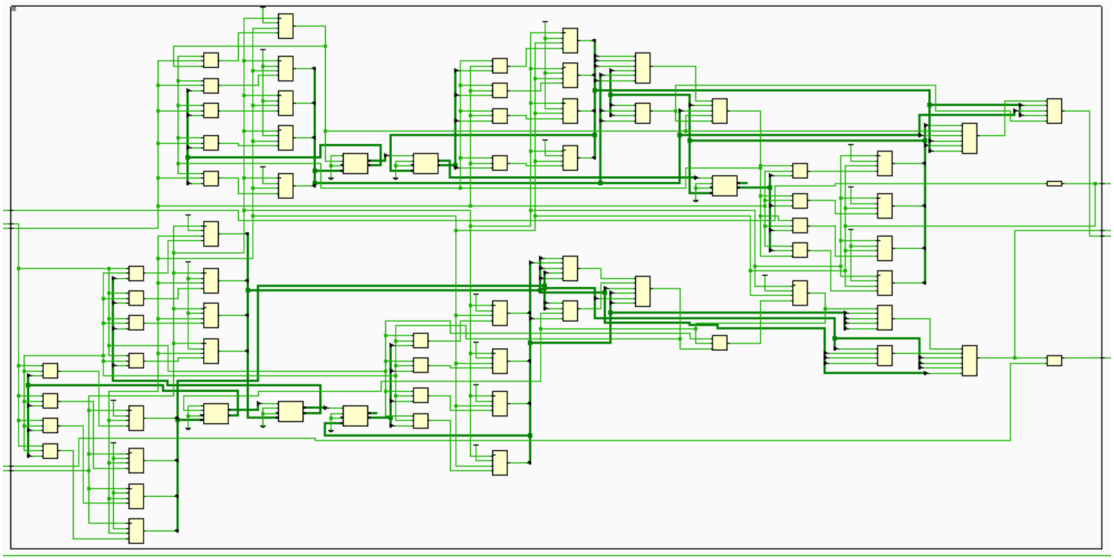
该模块用于在蓝牙模块功能的实现过程中，根据波特率调节系统时钟，使之契合蓝牙模块发射信息所需的模块频率。在这里，我们以串行数据的接收为例，对 baudrate_gen 模块的功能实现进行说明，其输出模块的原理相同。

当数据接收模块开始接受到蓝牙信号时，产生一个接收使能信号 I_bps_rx_clk_en 的高电位，说明此时开始接收串行数据。

当接收工作开始后，该模块会根据预先设定好的波特率 C_BPS_SELECT（在本项目中是 866），每隔波特率个时钟上升沿，就将接收模块时钟信号置高位一个系统时间长度。而这一位短暂的高电平尖峰，会成为数据接收模块执行功能所依赖的工作时钟，该模块的本质其实就是一个用于产生上升沿脉冲的分频器。

2. 功能框图





3. 接口信号说明

管脚名称	属性	具体描述
I_clk	input	输入 100MHz 的系统时钟
I_rst_n	input	用于系统的全局复位
I_bps_tx_clk_en	input	发送模块开始发送串行数据的时钟使能信号
I_bps_rx_clk_en	input	接收模块开始接收串行数据的时钟使能信号
O_bps_tx_clk	output	输出该模块产生的发送模块时钟信号
O_bps_tx_clk	output	输出该模块产生的接收模块时钟信号

4. verilog 实现

```
module baudrate_gen
(
    input    I_clk                ,// 系统 100MHz 时钟
    input    I_rst_n              ,// 系统全局复位
    input    I_bps_tx_clk_en      ,// 串口发送模块波特率时钟使能信号
    input    I_bps_rx_clk_en      ,// 串口接收模块波特率时钟使能信号
    output   O_bps_tx_clk         ,// 发送模块波特率产生时钟
```

```
    output    O_bps_rx_clk                  //  接收模块波特率产生时钟
);

parameter           C_BPS9600             = 5207            ,      //波特率为 9600bps
                    C_BPS19200            = 2603            ,      //波特率为 19200bps
                    C_BPS38400            = 1301            ,      //波特率为 38400bps
                    C_BPS57600            = 867             ,      //波特率为 57600bps
                    C_BPS115200           = 433             ;      //波特率为 115200bps

parameter           C_BPS_SELECT          = 866   ;      //波特率选择


reg [12:0]    R_bps_tx_cnt          ;
reg [12:0]    R_bps_rx_cnt          ;

////////////////////////////////////////////////////////////
// 功能：串口发送模块的波特率时钟产生逻辑
////////////////////////////////////////////////////////////
always @(posedge I_clk or negedge I_rst_n)
begin
    if(!I_rst_n)
        R_bps_tx_cnt <= 13'd0 ;
    else if(I_bps_tx_clk_en == 1'b1)
        begin
            if(R_bps_tx_cnt == C_BPS_SELECT)
                R_bps_tx_cnt <= 13'd0 ;
            else
                R_bps_tx_cnt <= R_bps_tx_cnt + 1'b1 ;
        end
    else
        R_bps_tx_cnt <= 13'd0 ;
end

assign O_bps_tx_clk = (R_bps_tx_cnt == 13'd1) ? 1'b1 : 1'b0 ;

////////////////////////////////////////////////////////////
// 功能：串口接收模块的波特率时钟产生逻辑
////////////////////////////////////////////////////////////
always @(posedge I_clk or negedge I_rst_n)
begin
    if(!I_rst_n)
        R_bps_rx_cnt <= 13'd0 ;
    else if(I_bps_rx_clk_en == 1'b1)
        begin
            if(R_bps_rx_cnt == C_BPS_SELECT)
                R_bps_rx_cnt <= 13'd0 ;
            else
                R_bps_rx_cnt <= R_bps_rx_cnt + 1'b1 ;
        end
    else
        R_bps_rx_cnt <= 13'd0 ;
end

assign O_bps_rx_clk = (R_bps_rx_cnt == C_BPS_SELECT >> 1'b1) ? 1'b1 : 1'b0 ;

endmodule
```

③ uart_txd 模块

1. 模块功能描述

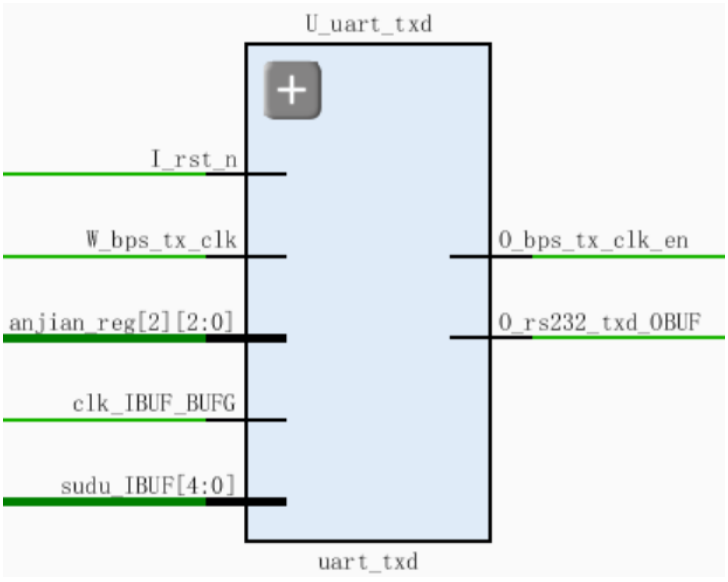
该模块用于在蓝牙模块功能的实现过程中，根据 baudrate_gen 模块所产生的波特时钟，将要发送的并行数据更改为串行进行发送。

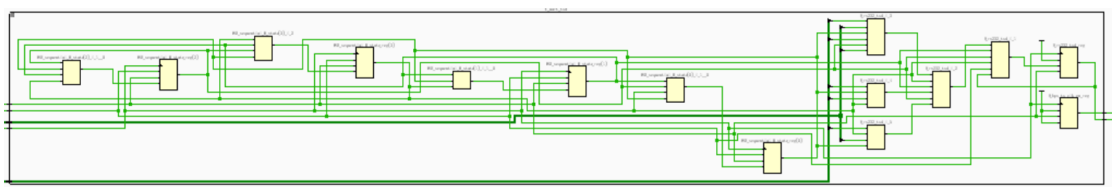
其实现逻辑在于，当发送标志位被置为高位时，开始进行数据的传输——首先将波特时钟的使能信号 O_bps_tx_clk_en 置高，之后开始根据 baudrate_gen 产生的波特时钟，每当时钟处于尖峰信号时，就向蓝牙模块传输一个比特的待发送信息，从而使得发送信息的频率与蓝牙传输的频率一致。

该模块在本项目中的实现依赖于计数器型状态机，其状态分配表如下：

现态	状态功能	次态
0000	发送一位信号 0，表示开始传输数据	0001
0001	发送待传输并行数据最低位 I_para_data[0]	0010
0010	发送高一位待传输并行数据 I_para_data[1]	0011
0011	发送高一位待传输并行数据 I_para_data[2]	0100
0100	发送高一位待传输并行数据 I_para_data[3]	0101
0101	发送高一位待传输并行数据 I_para_data[4]	0110
0110	发送高一位待传输并行数据 I_para_data[5]	0111
0111	发送高一位待传输并行数据 I_para_data[6]	1000
1000	发送最高位待传输并行数据 I_para_data[7]	1001
1001	发送终止信号 1，同时在这 8 位数据传输结束后，关闭发送波特钟使能信号 O_bps_tx_clk_en。	\

2. 功能框图





3. 接口信号说明

管脚名称	属性	具体描述
I_clk	input	输入 100MHz 的系统时钟
I_rst_n	input	用于系统的全局复位
I_tx_start	input	表示开始发送信息的使能信号
I_bps_tx_clk	input	接收到的发送波特钟信号
I_para_data	input [7:0]	待发送数据的并行存储单元
O_rs232_txd	output	发送到蓝牙模块的一位串行信息
O_bps_tx_clk_en	output	发送波特时钟的使能信号
O_tx_done	output	记录当前信息发送的结束与否

4. verilog 实现

```
module uart_txd
(
    input           I_clk           , //  系统 100MHz 时钟
    input           I_rst_n         , //  系统全局复位
    input           I_tx_start      , //  发送使能信号
    input           I_bps_tx_clk    , //  发送波特率时钟
    input   [7:0]   I_para_data     , //  要发送的并行数据
    output  reg     O_rs232_txd     , //  发送的串行数据，在硬件上与串口相连
    output  reg     O_bps_tx_clk_en , //  波特率时钟使能信号
    output  reg     O_tx_done       //  发送完成的标志
);

reg  [3:0]  R_state ;

reg R_transmiting ; //  数据正在发送标志

//////////////////////////////////////////////////////////////////////////////
//  产生发送  R_transmiting  标志位
//////////////////////////////////////////////////////////////////////////////
always @(posedge I_clk or negedge I_rst_n)
begin
    if(!I_rst_n)
        R_transmiting <= 1'b0 ;
    else if(O_tx_done)
        R_transmiting <= 1'b0 ;
    else if(I_tx_start)
        R_transmiting <= 1'b1 ;
end

//////////////////////////////////////////////////////////////////////////////
//  发送数据状态机
//////////////////////////////////////////////////////////////////////////////
always @(posedge I_clk or negedge I_rst_n)
begin
    if(!I_rst_n)
        begin
            R_state         <= 4'd0 ;
            O_rs232_txd     <= 1'b1 ;
            O_tx_done       <= 1'b0 ;
            O_bps_tx_clk_en <= 1'b0 ;   //  关掉波特率时钟使能信号
```

```
        end
    else if(1) // 检测发送标志被拉高，准备发送数据
        begin
            O_bps_tx_clk_en <= 1'b1 ;  // 发送数据前的第一件事就是打开波特率时钟使能信号
            if(I_bps_tx_clk) // 在波特率时钟的控制下把数据通过一个状态机发送出去，并产生发送完成信号
                begin
                    case(R_state)
                        4'd0  : // 发送起始位
                            begin
                                O_rs232_txd   <= 1'b0                ;
                                O_tx_done     <= 1'b0                ;
                                R_state       <= R_state + 1'b1   ;
                            end
                        4'd1  : // 发送 I_para_data[0]
                            begin
                                O_rs232_txd   <= I_para_data[0]   ;
                                O_tx_done     <= 1'b0                ;
                                R_state       <= R_state + 1'b1   ;
                            end
                        4'd2  : // 发送 I_para_data[1]
                            begin
                                O_rs232_txd   <= I_para_data[1]    ;
                                O_tx_done     <= 1'b0                ;
                                R_state       <= R_state + 1'b1    ;
                            end
                        4'd3  : // 发送 I_para_data[2]
                            begin
                                O_rs232_txd   <= I_para_data[2]    ;
                                O_tx_done     <= 1'b0                ;
                                R_state       <= R_state + 1'b1    ;
                            end
                        4'd4  : // 发送 I_para_data[3]
                            begin
                                O_rs232_txd   <= I_para_data[3]    ;
                                O_tx_done     <= 1'b0                ;
                                R_state       <= R_state + 1'b1    ;
                            end
                        4'd5  : // 发送 I_para_data[4]
                            begin
                                O_rs232_txd   <= I_para_data[4]    ;
                                O_tx_done     <= 1'b0                ;
                                R_state       <= R_state + 1'b1    ;
                            end
                        4'd6  : // 发送 I_para_data[5]
                            begin
                                O_rs232_txd   <= I_para_data[5]    ;
                                O_tx_done     <= 1'b0                ;
                                R_state       <= R_state + 1'b1    ;
                            end
                        4'd7  : // 发送 I_para_data[6]
                            begin
                                O_rs232_txd   <= I_para_data[6]    ;
                                O_tx_done     <= 1'b0                ;
                                R_state       <= R_state + 1'b1    ;
```

```
                    end
                4'd8   : // 发送 I_para_data[7]
                    begin
                        O_rs232_txd   <= I_para_data[7]    ;
                        O_tx_done     <= 1'b0              ;
                        R_state       <= R_state + 1'b1    ;
                    end
                4'd9   : // 发送 停止位
                    begin
                        O_rs232_txd   <= 1'b1              ;
                        O_tx_done     <= 1'b1              ;
                        R_state       <= 4'd0              ;
                    end
                default :R_state      <= 4'd0              ;
            endcase
        end
    end
else
    begin
        O_bps_tx_clk_en       <= 1'b0   ; // 一帧数据发送完毕以后就
关掉波特率时钟使能信号
        R_state<= 4'd0   ;
        O_tx_done<= 1'b0   ;
        O_rs232_txd<= 1'b1   ;
    end
end

endmodule
```

④ uart_rxd 模块

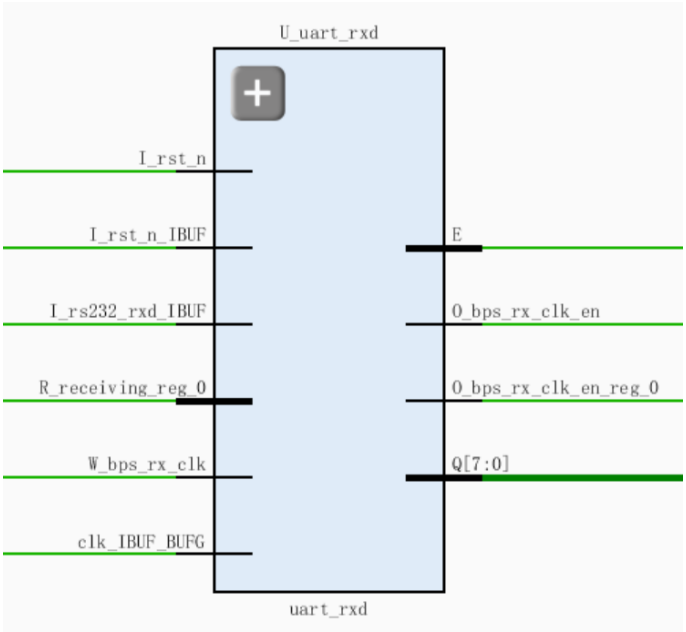
1. 模块功能描述

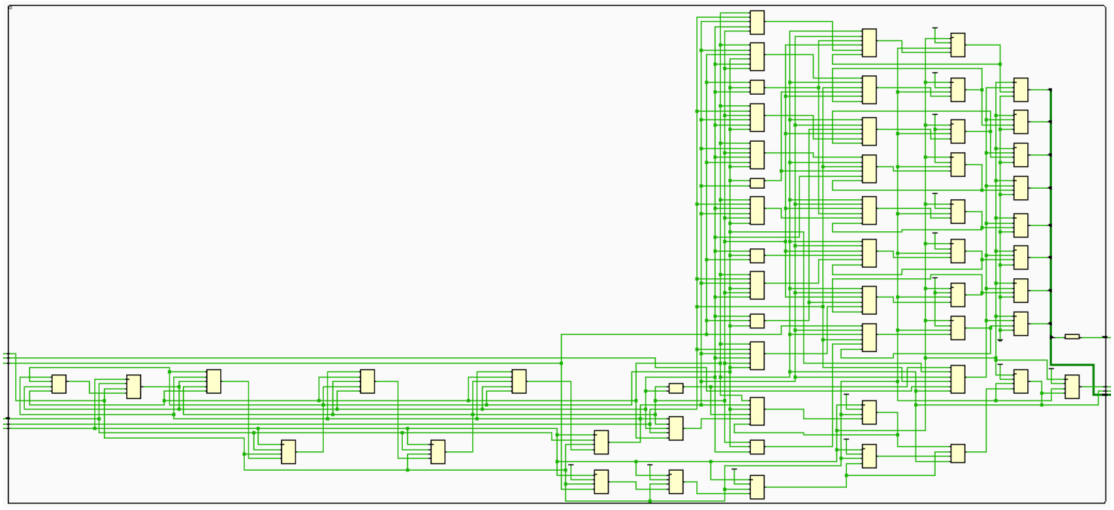
该模块用于在蓝牙模块功能的实现过程中，根据 baudrate_gen 模块所产生的波特时钟，将要发送的并行数据更改为串行进行发送。

其实现逻辑在于，首先判断复位信号是否为有效，如果复位状态有效，就将当前接受到的所有数据都清零；若不为有效状态，则开始进行数据的接受。其过程与 uart_txd 模块相仿，当准备开始接收数据的时候，就将接收波特时钟使能端 I_bps_rx_clk 置为高位。自此之后，根据波特时钟产生的尖峰下的状态机从蓝牙模块接收串行数据，再以并行方式存储即可，其状态分配表为：

现态	状态功能	次态
0000	接收数据 0 但不存储，将接收数据寄存器 R_para_data_reg 清零	0001
0001	接收一位数据并置于寄存器最低位 I_para_data[0]	0010
0010	接收一位数据并置于寄存器高一位 I_para_data[1]	0011
0011	接收一位数据并置于寄存器高一位 I_para_data[2]	0100
0100	接收一位数据并置于寄存器高一位 I_para_data[3]	0101
0101	接收一位数据并置于寄存器高一位 I_para_data[4]	0110
0110	接收一位数据并置于寄存器高一位 I_para_data[5]	0111
0111	接收一位数据并置于寄存器高一位 I_para_data[6]	1000
1000	接收一位数据并置于寄存器高一位 I_para_data[7]	1001
1001	接收终止信号 1 但不存储，将整个寄存器的值复制给模块输出端口 O_para_data	\

2. 功能框图





3. 接口信号说明

管脚名称	属性	具体描述
I_clk	input	输入 100MHz 的系统时钟
I_rst_n	input	用于系统的全局复位
I_rx_start	input	表示开始接收信息的使能信号
I_bps_rx_clk	input	接收到的接收波特钟信号
I_rs232_rxd	input	从蓝牙模块接收到的串行数据
O_bps_rs_clk_en	output	接收波特率时钟的使能信号
O_rx_done	output	记录当前信息接收的结束与否
O_para_data	output [7:0]	所接收到以并行方式存储的数据

4. verilog 实现

```
module uart_rxd
(
    input                           I_clk                   , //  系统 100MHz 时钟
    input                           I_rst_n                 , //  系统全局复位
    input                           I_rx_start              , //  接收使能信号
    input                           I_bps_rx_clk            , //  接收波特率时钟
    input                           I_rs232_rxd             , //  接收的串行数据，在硬
件上与串口相连
    output      reg                 O_bps_rx_clk_en         , //  波特率时钟使能信号
    output      reg                 O_rx_done               , //  接收完成标志
    output      reg     [7:0]       O_para_data             //  接收到的 8-bit 并行数
据
);

reg             R_rs232_rx_reg0 ;
reg             R_rs232_rx_reg1 ;
reg             R_rs232_rx_reg2 ;
reg             R_rs232_rx_reg3 ;

reg             R_receiving     ;

reg [3:0]   R_state             ;
reg [7:0]   R_para_data_reg ;

wire            W_rs232_rxd_neg ;

////////////////////////////////////////////////////////////////////////////
// 功能：把 I_rs232_rxd 打的前两拍，是为了消除亚稳态
//              把 I_rs232_rxd 打的后两拍，是为了产生下降沿标志位
////////////////////////////////////////////////////////////////////////////
always @(posedge I_clk or negedge I_rst_n)
begin
    if(!I_rst_n)
        begin
            R_rs232_rx_reg0 <= 1'b0 ;
            R_rs232_rx_reg1 <= 1'b0 ;
            R_rs232_rx_reg2 <= 1'b0 ;
            R_rs232_rx_reg3 <= 1'b0 ;
        end
    else
        begin
            R_rs232_rx_reg0 <= I_rs232_rxd      ;
            R_rs232_rx_reg1 <= R_rs232_rx_reg0  ;
            R_rs232_rx_reg2 <= R_rs232_rx_reg1  ;
            R_rs232_rx_reg3 <= R_rs232_rx_reg2  ;
        end
end
// 产生 I_rs232_rxd 信号的下降沿标志位
assign W_rs232_rxd_neg      =       (~R_rs232_rx_reg2) & R_rs232_rx_reg3 ;

////////////////////////////////////////////////////////////////////////////
// 功能：产生发送信号 R_receiving
////////////////////////////////////////////////////////////////////////////
always @(posedge I_clk or negedge I_rst_n)
begin
    if(!I_rst_n)
```

```
            R_receiving <= 1'b0 ;
        else if(O_rx_done)
            R_receiving <= 1'b0 ;
        else if(I_rx_start && W_rs232_rxd_neg)
            R_receiving <= 1'b1 ;
end

//////////////////////////////////////////////////////////////////////////////
// 功能：用状态机把串行的输入数据接收，并转化为并行数据输出
//////////////////////////////////////////////////////////////////////////////
always @(posedge I_clk or negedge I_rst_n)
begin
    if(!I_rst_n)
        begin
            O_rx_done           <= 1'b0 ;
            R_state             <= 4'd0 ;
            R_para_data_reg <= 8'd0 ;
            O_bps_rx_clk_en <= 1'b0 ;
        end
    else if(R_receiving)
        begin
            O_bps_rx_clk_en <= 1'b1 ; // 打开波特率时钟使能信号
            if(I_bps_rx_clk)
                begin
                    case(R_state)
                        4'd0  : // 接收起始位，但不保存
                            begin
                                R_para_data_reg      <= 8'd0                ;
                                O_rx_done              <= 1'b0                ;
                                R_state                <= R_state + 1'b1    ;
                            end
                        4'd1  : // 接收第 0 位，保存到 R_para_data_reg[0]
                            begin
                                R_para_data_reg[0]   <= I_rs232_rxd         ;
                                O_rx_done              <= 1'b0                ;
                                R_state                <= R_state + 1'b1    ;
                            end
                        4'd2  : // 接收第 1 位，保存到 R_para_data_reg[1]
                            begin
                                R_para_data_reg[1]   <= I_rs232_rxd         ;
                                O_rx_done              <= 1'b0                ;
                                R_state                <= R_state + 1'b1    ;
                            end
                        4'd3  : // 接收第 2 位，保存到 R_para_data_reg[2]
                            begin
                                R_para_data_reg[2]   <= I_rs232_rxd         ;
                                O_rx_done              <= 1'b0                ;
                                R_state                <= R_state + 1'b1    ;
                            end
                        4'd4  : // 接收第 3 位，保存到 R_para_data_reg[3]
                            begin
                                R_para_data_reg[3]   <= I_rs232_rxd         ;
                                O_rx_done              <= 1'b0                ;
                                R_state                <= R_state + 1'b1    ;
                            end
                        4'd5  : // 接收第 4 位，保存到 R_para_data_reg[4]
```

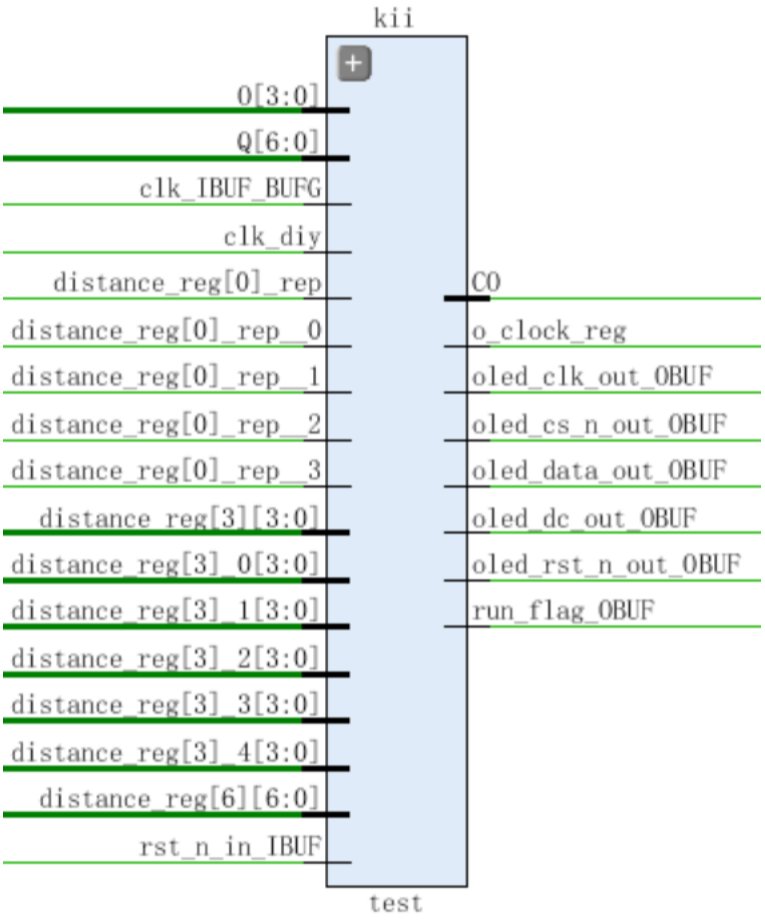
```
                    begin
                        R_para_data_reg[4]   <= I_rs232_rxd         ;
                        O_rx_done              <= 1'b0                ;
                        R_state                <= R_state + 1'b1      ;
                    end
                4'd6  : // 接收第 5 位，保存到 R_para_data_reg[5]
                    begin
                        R_para_data_reg[5]   <= I_rs232_rxd         ;
                        O_rx_done              <= 1'b0                ;
                        R_state                <= R_state + 1'b1      ;
                    end
                4'd7  :// 接收第 6 位，保存到 R_para_data_reg[6]
                    begin
                        R_para_data_reg[6]   <= I_rs232_rxd         ;
                        O_rx_done              <= 1'b0                ;
                        R_state                <= R_state + 1'b1      ;
                    end
                4'd8  : // 接收第 7 位，保存到 R_para_data_reg[7]
                    begin
                        R_para_data_reg[7]   <= I_rs232_rxd         ;
                        O_rx_done              <= 1'b0                ;
                        R_state                <= R_state + 1'b1      ;
                    end
                4'd9  : // 接收停止位，但不保存，并把 R_para_data_reg
给输出
                    begin
                        O_para_data          <= R_para_data_reg   ;
                        O_rx_done              <= 1'b1                ;
                        R_state                <= 4'd0                ;
                    end

                default:R_state <= 4'd0                                 ;
            endcase
        end
    end
else
    begin
        O_rx_done           <= 1'b0 ;
        R_state             <= 4'd0 ;
        R_para_data_reg     <= 8'd0 ;
        O_bps_rx_clk_en     <= 1'b0 ; // 接收完毕以后关闭波特率时钟使能
信号
    end
end

endmodule
```

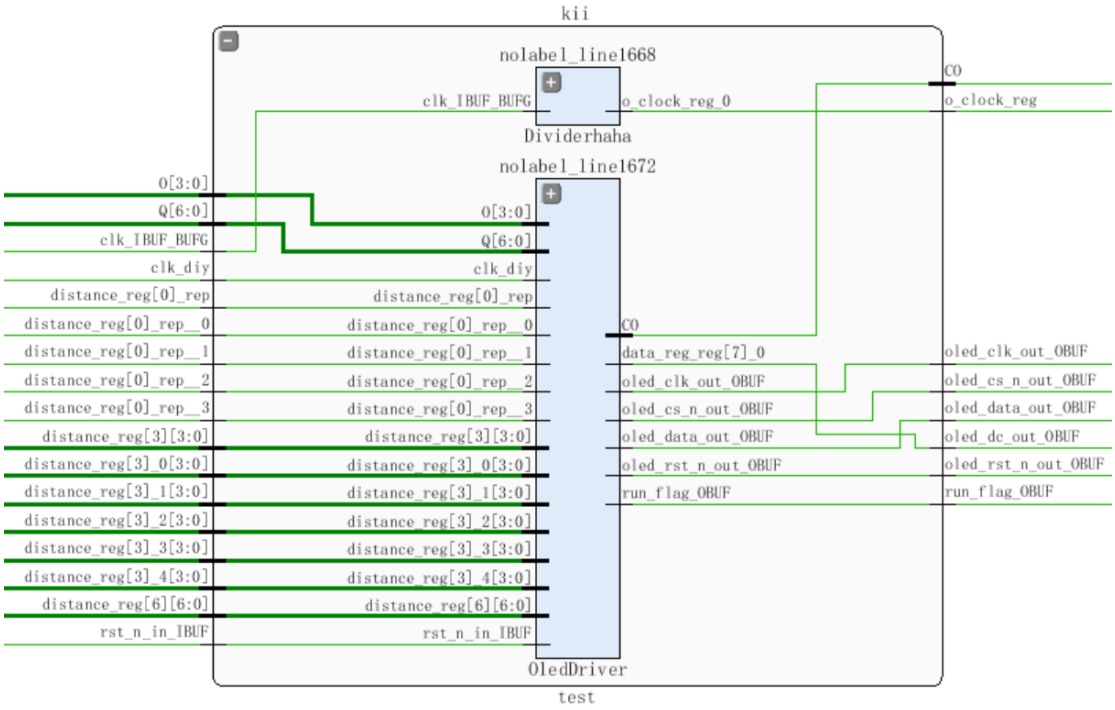
⑤ test 模块

1. 模块功能描述

该模块是 OLED 显示器驱动的顶层模块，通过实例化 Dividerhaha 模块与 OledDriver 模块的方式，完成 OLED 模块的功能实现。

2. 功能框图





3. 接口信号说明

管脚名称	属性	具体描述

distance	input [6:0]	由蓝牙模块传输而来的距离信息
dangel	input [6:0]	由蓝牙模块传输而来的角度信息
clk_in	input	输入系统时钟
rst_n_in	input	系统复位信号，低电平有效
run_flag	output	记录当前是否处于 OLED 信息写入过程
oled_rst_n_out	output	用于 OLED 屏幕信息的清空，低电平有效
oled_cs_n_out	output	片选信号，用于 OLED 屏幕的选定，低电平有效
oled_dc_out	output	模块命令信号，表示输入的信号是数据还是命令 低电平表示命令，高电平表示数据
oled_clk_out	output	向 OLED 外设模块输出时钟
oled_data_out	output	按照串行方式向 OLED 模块输出信号

4. verilog 实现

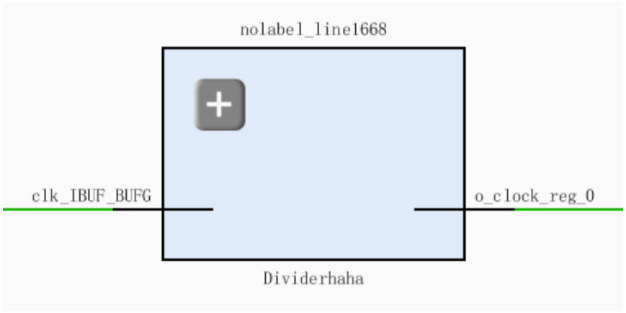
```
module test(
    input [6:0] distance,
    input [6:0] dangel,
    input clk_in,    //clk_in = 100mhz
    input rst_n_in,    //rst_n_in, active low
    output run_flag,
    output oled_rst_n_out,    //nokia5110 reset, active low
    output oled_cs_n_out,    //nokia5110 chip select, active low
    output oled_dc_out,    //nokia5110 data or command control
    output oled_clk_out,    //nokia5110 clock
    output oled_data_out    //nokia5110 data)
    );

wire clk_diy;
Dividerhaha(clk_in,0,clk_diy);


OledDriver
(
clk_diy,rst_n_in,    //rst_n_in, active low
distance,
dangel,
run_flag,
oled_rst_n_out,    //nokia5110 reset, active low
oled_cs_n_out,    //nokia5110 chip select, active low
oled_dc_out,    //nokia5110 data or command control
oled_clk_out,    //nokia5110 clock
oled_data_out    //nokia5110 data
);


endmodule
```

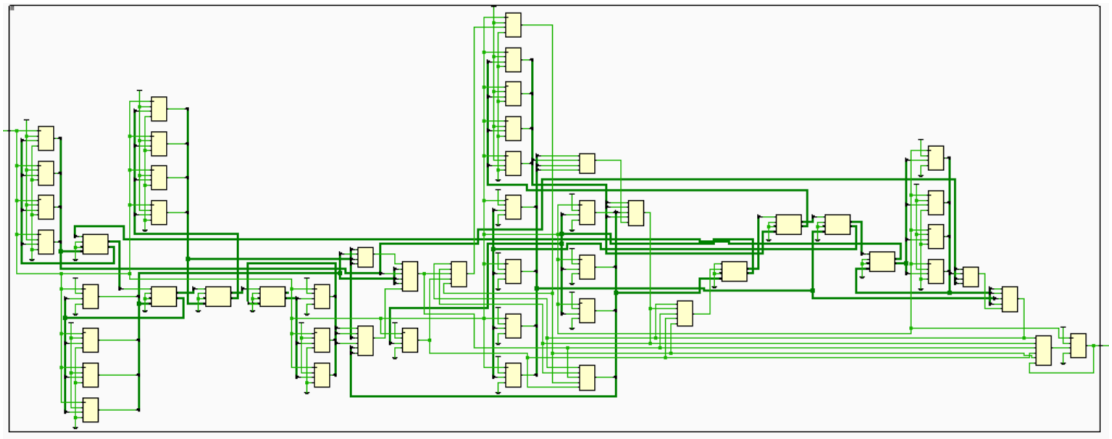
⑥ Dividerhaha 模块

1. 模块功能描述

该模块的功能是使用之前作业所写过的 Divider 模块，将开发板自带的时钟频率降到三分之一，以满足 OLED 屏幕刷新率的需要。

2. 功能框图





3. 接口信号说明

管脚名称	属性	具体描述
I_CLK	input	被降频的时钟，这里是系统时钟
rst	input	复位信号，高电平有效
O_CLK	input	降频后的输出时钟

4. verilog 实现

```
module Dividerhaha(
    input I_CLK,
    input rst,
    output O_CLK
);
    parameter N=3;
    reg o_clock;
    integer i;
    always @(posedge I_CLK)
    begin
    if(rst)
        begin
        o_clock=0;
        i=0;
        end
    else
        begin
        if(i==N)
        begin
        i=0;
        o_clock=~o_clock;
```

```
            end
        i=i+1;
        end
        end
        assign O_CLK=o_clock;
    endmodule
```

⑦ OledDriver 模块

1. 模块功能描述

该模块的功能是整体实现 OLED 显示屏的功能，包括指令信号的识别与读入、串行数据的识别与读入等。其功能的实现方式可简单概括如下：

首先定义好一个指令存储二维数组 cmd_r，该二维数组作为指令信号依次读入 OLED 可以实现 OLED 模块的初始化、频率设置等；此外还定义了一个与 OLED 屏幕等大的二维数组 mem，在其中存储一套 OLED 屏幕上对应位置的像素点信息。

而当我们需要显示屏显示的内容产生变化时，只需要编写对应的代码，更改对应位置的像素点信息。而 OLED 屏幕内容的输出与刷新则无需经过多余的更改，只需要每次从 mem 寄存器中取值，并逐一逐位输出，即可实现 OLED 显示内容的变化。

而关于 OLED 的遍历输出，本项目定义了 5 个 OLED 屏幕的输出状态：

IDLE：指令控制部分，该部分用于控制向 OLED 传输的信息类型、暂存地址等；

SHIFT：根据 IDLE 状态的指令，将工作寄存器 data_reg 中的并行数据进行串行输出，同时对其他几个引脚的值进行输出；

DISPLAY：根据 IDLE 状态的指令，将对应指令寄存器 cmd_r 或是 mem 中指定位置的信息存储到工作数组 data_reg 中；

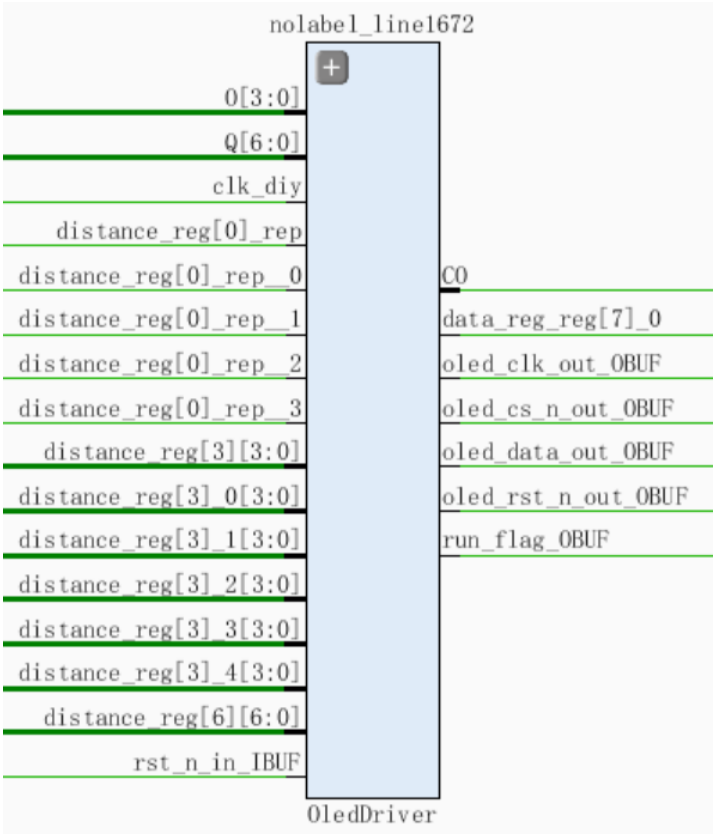
CLEAR：该状态用于将工作数组清零，同时将记录 DISPLAY 次数的计数器 temp_cnt 置零；

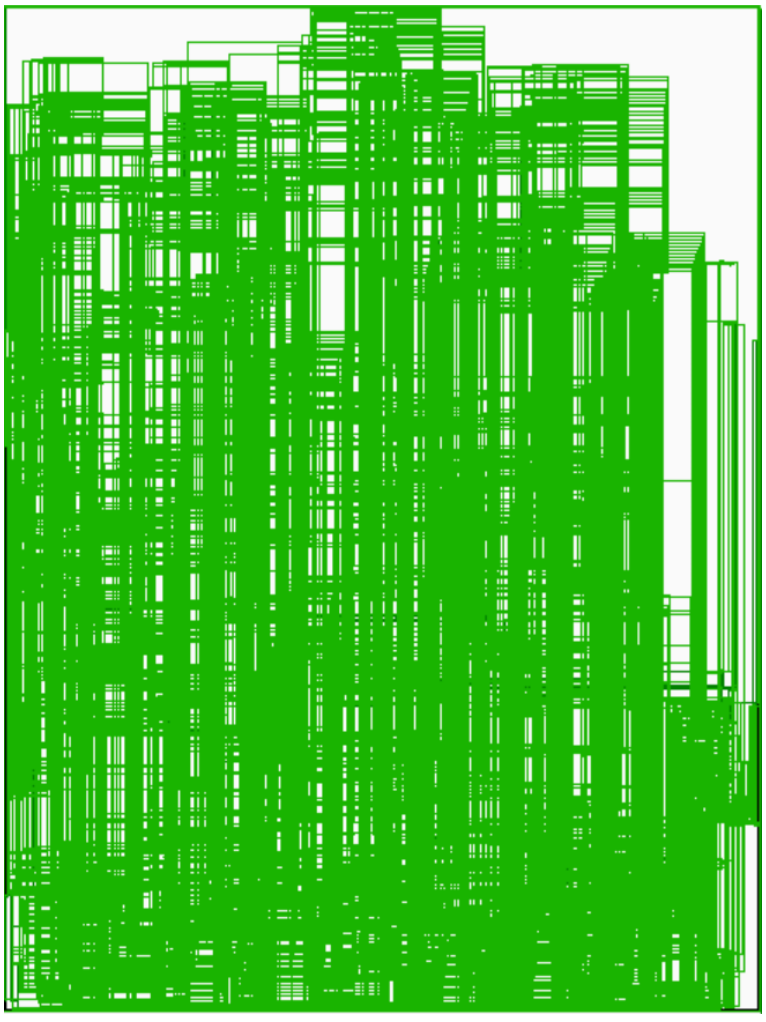
DELAY：该状态用于在每次 IDLE 状态内的状态机开始循环时调用，用于给出延迟缓冲时间，以防止冲突的发生。该状态不进行任何操作。

值得一提的是，为了保证数据传输的高效性，本项目采取以 8 个字节为一批，进行传输的方式，这样的话代码本身可能会显得较为冗长，但并行向串行的转换会变得较为迅速。而且工作数组每次只用传递少量的数据，避免了因为大批量数据高速赋值导致的卡顿现象。

在 IDLE 与 DISPLAY 状态中，因为需要一次对每八位的信息进行赋值与传输，所以在 IDLE 于 DISPLAY 内，本项目也采用了状态机进行编写。——即 OLED 持续不间断地从 cmd_r 与 mem 寄存器中读取指令或是信息，重复刷新 OLED 的显示页面。

2. 功能框图


nolabel_line1672
0[3:0]
Q[6:0]
clk_diy
distance_reg[0]_rep
distance_reg[0]_rep__0
distance_reg[0]_rep__1
distance_reg[0]_rep__2
distance_reg[0]_rep__3
distance_reg[3][3:0]
distance_reg[3]_0[3:0]
distance_reg[3]_1[3:0]
distance_reg[3]_2[3:0]
distance_reg[3]_3[3:0]
distance_reg[3]_4[3:0]
distance_reg[6][6:0]
rst_n_in_IBUF
CO
data_reg_reg[7]_0
oled_clk_out_OBUF
oled_cs_n_out_OBUF
oled_data_out_OBUF
oled_rst_n_out_OBUF
run_flag_OBUF
OledDriver



3. 接口信号说明

管脚名称	属性	具体描述
clk_in	input	输入系统时钟
rst_n_in	input	系统复位信号，低电平有效
distance	input [6:0]	需要输出的距离信息
angle	input [6:0]	需要输出的角度信息
run_flag	output	记录当前是否处于 OLED 信息写入过程
oled_rst_n_out	output	用于 OLED 屏幕信息的清空，低电平有效
oled_cs_n_out	output	片选信号，用于 OLED 屏幕的选定，低电平有效
oled_dc_out	output	模块命令信号，表示输入的信号是数据还是命令 低电平表示命令，高电平表示数据
oled_clk_out	output	向 OLED 外设模块输出时钟
oled_data_out	output	按照串行方式向 OLED 模块输出信号

4. verilog 实现

```
module OledDriver
(
input clk_in,    //clk_in = 25mhz
input rst_n_in,    //rst_n_in, active low
input [6:0] distance,
input [6:0] angle,
output reg run_flag,
output reg oled_rst_n_out,    //nokia5110 reset, active low
output reg oled_cs_n_out,    //nokia5110 chip select, active low
output reg oled_dc_out,    //nokia5110 data or command control
output oled_clk_out,    //nokia5110 clock
output reg oled_data_out    //nokia5110 data

);

parameter CLK_DIV_PERIOD=20; //related with clk_div's frequency
parameter DELAY_PERIOD=25000;    //related with delay time and refresh frequency

parameter CLK_L=2'd0;
parameter CLK_H=2'd1;
parameter CLK_RISING_DEGE=2'd2;
parameter CLK_FALLING_DEGE=2'd3;

parameter IDLE=3'd0;
parameter SHIFT=3'd1;
parameter CLEAR=3'd2;
parameter SETXY=3'd3;
parameter DISPLAY=3'd4;
parameter DELAY=3'd5;

parameter LOW =1'b0;
parameter HIGH =1'b1;
parameter CMD =1'b0;
parameter DATA =1'b1;
```

```
//assign oled_rst_n_out = 1;    //active low level, set 1 for normal
//assign oled_dc_out = 1;


//initial for memory register
reg [47:0] cmd_r [9:0];
initial
  begin
        cmd_r[0]= {8'hae, 8'ha0, 8'h76, 8'ha1, 8'h00, 8'ha2};      // command for initial
            cmd_r[1]= {8'h00, 8'ha4, 8'ha8, 8'h3f, 8'had, 8'h8e};      // command for initial
            cmd_r[2]= {8'hb0, 8'h0b, 8'hb1, 8'h31, 8'hb3,8'hf0};      // command for initial
            cmd_r[3]= {8'h8a, 8'h64, 8'h8b, 8'h78, 8'h8c,8'h64};      // command for initial
            cmd_r[4]= {8'hbb, 8'h3a, 8'hbe, 8'h3e, 8'h87,8'h06};      // command for initial
            cmd_r[5]= {8'h81, 8'h91, 8'h82, 8'h50, 8'h83,8'h7d};      // command for initial
            cmd_r[6]= {8'h15, 8'h00, 8'h5f, 8'h75, 8'h00,8'h3f};      // command for initial
            cmd_r[6]= {8'haf,8'h00, 8'h00, 8'h00, 8'h00, 8'h00};      // command for initial
  end

//initial for memory register
reg [1535:0] mem [63:0];
reg [1535:0] temp;
initial
  begin
      mem[0]= {16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000};
      mem[1]= {16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000};
      mem[2]= {16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'hffff};
```

```
        mem[3]= {16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000};
    //……这里代码重复度过高，所以不再展示
        mem[63]= {16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000};
end

//clk_div = clk_in/CLK_DIV_PERIOD, 50% is high voltage
integer distance1,distance2,distance3;
integer a1,a2;

wire [5:0]aaaa;
assign aaaa=angle[5:0];

always @(posedge clk_in)
begin
    if(clk_in)
    begin
    distance1=distance/100;
    distance2=distance/10-distance1*10;
    distance3=distance%10;
    a1=angle/64;
    a2=angle%64;
    case (distance1)
        0:
        begin
        mem[15]={{mem[15][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[15][639:0]}};
        mem[16]={{mem[16][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[16][639:0]}};
        mem[17]={{mem[17][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[17][639:0]}};
        mem[18]={{mem[18][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[18][639:0]}};
        mem[19]={{mem[19][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[19][639:0]}};
        mem[20]={{mem[20][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
```

```
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[20][639:0]}};
            mem[21]={{mem[21][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[21][639:0]}};
            mem[22]={{mem[22][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[22][639:0]}};
            mem[23]={{mem[23][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[23][639:0]}};
            mem[24]={{mem[24][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[24][639:0]}};
            mem[25]={{mem[25][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[25][639:0]}};
            end
            1:
            begin
            mem[15]={{mem[15][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[15][639:0]}};
            mem[16]={{mem[16][1535:768]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[16][639:0]}};
            mem[17]={{mem[17][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[17][639:0]}};
            mem[18]={{mem[18][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[18][639:0]}};
            mem[19]={{mem[19][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[19][639:0]}};
            mem[20]={{mem[20][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[20][639:0]}};
            mem[21]={{mem[21][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[21][639:0]}};
            mem[22]={{mem[22][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[22][639:0]}};
            mem[23]={{mem[23][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[23][639:0]}};
            mem[24]={{mem[24][1535:768]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[24][639:0]}};
            mem[25]={{mem[25][1535:768]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'hffff, 16'h0000,{mem[25][639:0]}};
            end
      endcase

      case (distance2)
            0:
            begin
            mem[15]={{mem[15][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[15][511:0]}};
            mem[16]={{mem[16][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[16][511:0]}};
            mem[17]={{mem[17][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[17][511:0]}};
            mem[18]={{mem[18][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[18][511:0]}};
            mem[19]={{mem[19][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[19][511:0]}};
            mem[20]={{mem[20][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[20][511:0]}};
            mem[21]={{mem[21][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[21][511:0]}};
            mem[22]={{mem[22][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
```

```
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][511:0]}};
            mem[23]={{mem[23][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][511:0]}};
            mem[24]={{mem[24][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[24][511:0]}};
            mem[25]={{mem[25][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[25][511:0]}};
            end

            1:
            begin
            mem[15]={{mem[15][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[15][511:0]}};
            mem[16]={{mem[16][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[16][511:0]}};
            mem[17]={{mem[17][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[17][511:0]}};
            mem[18]={{mem[18][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[18][511:0]}};
            mem[19]={{mem[19][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[19][511:0]}};
            mem[20]={{mem[20][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[20][511:0]}};
            mem[21]={{mem[21][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[21][511:0]}};
            mem[22]={{mem[22][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[22][511:0]}};
            mem[23]={{mem[23][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[23][511:0]}};
            mem[24]={{mem[24][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[24][511:0]}};
            mem[25]={{mem[25][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'hffff, 16'h0000,{mem[25][511:0]}};
            end

            2:
            begin
            mem[15]={{mem[15][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[15][511:0]}};
            mem[16]={{mem[16][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[16][511:0]}};
            mem[17]={{mem[17][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[17][511:0]}};
            mem[18]={{mem[18][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[18][511:0]}};
            mem[19]={{mem[19][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[19][511:0]}};
            mem[20]={{mem[20][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[20][511:0]}};
            mem[21]={{mem[21][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[21][511:0]}};
            mem[22]={{mem[22][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[22][511:0]}};
            mem[23]={{mem[23][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[23][511:0]}};
            mem[24]={{mem[24][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[24][511:0]}};
```

```
            mem[25]={{mem[25][1535:640]},16'h0000, 16'hffff, 16'hffff, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000,{mem[25][511:0]}};
            end

            3:
            begin
            mem[15]={{mem[15][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[15][511:0]}};
            mem[16]={{mem[16][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[16][511:0]}};
            mem[17]={{mem[17][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[17][511:0]}};
            mem[18]={{mem[18][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[18][511:0]}};
            mem[19]={{mem[19][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[19][511:0]}};
            mem[20]={{mem[20][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[20][511:0]}};
            mem[21]={{mem[21][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[21][511:0]}};
            mem[22]={{mem[22][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][511:0]}};
            mem[23]={{mem[23][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][511:0]}};
            mem[24]={{mem[24][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[24][511:0]}};
            mem[25]={{mem[25][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[25][511:0]}};
            end
            4:
            begin
            mem[15]={{mem[15][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[15][511:0]}};
            mem[16]={{mem[16][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[16][511:0]}};
            mem[17]={{mem[17][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[17][511:0]}};
            mem[18]={{mem[18][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[18][511:0]}};
            mem[19]={{mem[19][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[19][511:0]}};
            mem[20]={{mem[20][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[20][511:0]}};
            mem[21]={{mem[21][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[21][511:0]}};
            mem[22]={{mem[22][1535:640]},16'h0000, 16'hffff, 16'hffff, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'hffff,{mem[22][511:0]}};
            mem[23]={{mem[23][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[23][511:0]}};
            mem[24]={{mem[24][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[24][511:0]}};
            mem[25]={{mem[25][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'hffff, 16'hffff, 16'hffff,{mem[25][511:0]}};
            end
            5:
            begin
            mem[15]={{mem[15][1535:640]},16'h0000, 16'hffff, 16'hffff, 16'hffff, 16'hffff,
```

```
16'hffff, 16'hffff, 16'h0000,{mem[15][511:0]}};
            mem[16]={{mem[16][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[16][511:0]}};
            mem[17]={{mem[17][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[17][511:0]}};
            mem[18]={{mem[18][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[18][511:0]}};
            mem[19]={{mem[19][1535:640]},16'h0000, 16'hffff, 16'hffff, 16'hffff, 16'hffff,
16'h0000, 16'h0000, 16'h0000,{mem[19][511:0]}};
            mem[20]={{mem[20][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[20][511:0]}};
            mem[21]={{mem[21][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[21][511:0]}};
            mem[22]={{mem[22][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][511:0]}};
            mem[23]={{mem[23][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][511:0]}};
            mem[24]={{mem[24][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[24][511:0]}};
            mem[25]={{mem[25][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[25][511:0]}};
            end
            6:
            begin
            mem[15]={{mem[15][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[15][511:0]}};
            mem[16]={{mem[16][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[16][511:0]}};
            mem[17]={{mem[17][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[17][511:0]}};
            mem[18]={{mem[18][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[18][511:0]}};
            mem[19]={{mem[19][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[19][511:0]}};
            mem[20]={{mem[20][1535:640]},16'h0000, 16'hffff, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[20][511:0]}};
            mem[21]={{mem[21][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[21][511:0]}};
            mem[22]={{mem[22][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][511:0]}};
            mem[23]={{mem[23][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][511:0]}};
            mem[24]={{mem[24][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[24][511:0]}};
            mem[25]={{mem[25][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[25][511:0]}};
            end
            7:
            begin
            mem[15]={{mem[15][1535:640]},16'h0000, 16'hffff, 16'hffff, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000,{mem[15][511:0]}};
            mem[16]={{mem[16][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[16][511:0]}};
            mem[17]={{mem[17][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[17][511:0]}};
            mem[18]={{mem[18][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[18][511:0]}};
```

```
            mem[19]={{mem[19][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[19][511:0]}};
            mem[20]={{mem[20][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[20][511:0]}};
            mem[21]={{mem[21][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[21][511:0]}};
            mem[22]={{mem[22][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[22][511:0]}};
            mem[23]={{mem[23][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[23][511:0]}};
            mem[24]={{mem[24][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[24][511:0]}};
            mem[25]={{mem[25][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[25][511:0]}};
            end
            8:
            begin
            mem[15]={{mem[15][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[15][511:0]}};
            mem[16]={{mem[16][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[16][511:0]}};
            mem[17]={{mem[17][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[17][511:0]}};
            mem[18]={{mem[18][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[18][511:0]}};
            mem[19]={{mem[19][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[19][511:0]}};
            mem[20]={{mem[20][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[20][511:0]}};
            mem[21]={{mem[21][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[21][511:0]}};
            mem[22]={{mem[22][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][511:0]}};
            mem[23]={{mem[23][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][511:0]}};
            mem[24]={{mem[24][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[24][511:0]}};
            mem[25]={{mem[25][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[25][511:0]}};
            end
            9:
            begin
            mem[15]={{mem[15][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[15][511:0]}};
            mem[16]={{mem[16][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[16][511:0]}};
            mem[17]={{mem[17][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[17][511:0]}};
            mem[18]={{mem[18][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[18][511:0]}};
            mem[19]={{mem[19][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[19][511:0]}};
            mem[20]={{mem[20][1535:640]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'hffff, 16'h0000,{mem[20][511:0]}};
            mem[21]={{mem[21][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'hffff, 16'h0000,{mem[21][511:0]}};
            mem[22]={{mem[22][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
```

```
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][511:0]}};
            mem[23]={{mem[23][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][511:0]}};
            mem[24]={{mem[24][1535:640]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[24][511:0]}};
            mem[25]={{mem[25][1535:640]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[25][511:0]}};
            end
      endcase

      case (distance3)
                  0:
                  begin
                  mem[15]={{mem[15][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[15][255:0]}};
                  mem[16]={{mem[16][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[16][255:0]}};
                  mem[17]={{mem[17][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[17][255:0]}};
                  mem[18]={{mem[18][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[18][255:0]}};
                  mem[19]={{mem[19][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[19][255:0]}};
                  mem[20]={{mem[20][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[20][255:0]}};
                  mem[21]={{mem[21][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[21][255:0]}};
                  mem[22]={{mem[22][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][255:0]}};
                  mem[23]={{mem[23][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][255:0]}};
                  mem[24]={{mem[24][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[24][255:0]}};
                  mem[25]={{mem[25][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[25][255:0]}};
                  end

                  1:
                  begin
                  mem[15]={{mem[15][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[15][255:0]}};
                  mem[16]={{mem[16][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[16][255:0]}};
                  mem[17]={{mem[17][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[17][255:0]}};
                  mem[18]={{mem[18][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[18][255:0]}};
                  mem[19]={{mem[19][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[19][255:0]}};
                  mem[20]={{mem[20][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[20][255:0]}};
                  mem[21]={{mem[21][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[21][255:0]}};
                  mem[22]={{mem[22][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[22][255:0]}};
                  mem[23]={{mem[23][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[23][255:0]}};
```

```
                mem[24]={{mem[24][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[24][255:0]}};
                mem[25]={{mem[25][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'hffff, 16'h0000,{mem[25][255:0]}};
                end

                2:
                begin
                mem[15]={{mem[15][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[15][255:0]}};
                mem[16]={{mem[16][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[16][255:0]}};
                mem[17]={{mem[17][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[17][255:0]}};
                mem[18]={{mem[18][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[18][255:0]}};
                mem[19]={{mem[19][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[19][255:0]}};
                mem[20]={{mem[20][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[20][255:0]}};
                mem[21]={{mem[21][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[21][255:0]}};
                mem[22]={{mem[22][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[22][255:0]}};
                mem[23]={{mem[23][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[23][255:0]}};
                mem[24]={{mem[24][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[24][255:0]}};
                mem[25]={{mem[25][1535:384]},16'h0000, 16'hffff, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'hffff, 16'h0000,{mem[25][255:0]}};
                end

                3:
                begin
                mem[15]={{mem[15][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[15][255:0]}};
                mem[16]={{mem[16][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[16][255:0]}};
                mem[17]={{mem[17][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[17][255:0]}};
                mem[18]={{mem[18][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[18][255:0]}};
                mem[19]={{mem[19][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[19][255:0]}};
                mem[20]={{mem[20][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[20][255:0]}};
                mem[21]={{mem[21][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[21][255:0]}};
                mem[22]={{mem[22][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][255:0]}};
                mem[23]={{mem[23][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][255:0]}};
                mem[24]={{mem[24][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[24][255:0]}};
                mem[25]={{mem[25][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[25][255:0]}};
                end
```

```
                4:
                begin
                mem[15]={{mem[15][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[15][255:0]}};
                mem[16]={{mem[16][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[16][255:0]}};
                mem[17]={{mem[17][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[17][255:0]}};
                mem[18]={{mem[18][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[18][255:0]}};
                mem[19]={{mem[19][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[19][255:0]}};
                mem[20]={{mem[20][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[20][255:0]}};
                mem[21]={{mem[21][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[21][255:0]}};
                mem[22]={{mem[22][1535:384]},16'h0000, 16'hffff, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'hffff, 16'hffff,{mem[22][255:0]}};
                mem[23]={{mem[23][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[23][255:0]}};
                mem[24]={{mem[24][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[24][255:0]}};
                mem[25]={{mem[25][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'hffff, 16'hffff, 16'hffff,{mem[25][255:0]}};
                end
                5:
                begin
                mem[15]={{mem[15][1535:384]},16'h0000, 16'hffff, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'hffff, 16'h0000,{mem[15][255:0]}};
                mem[16]={{mem[16][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[16][255:0]}};
                mem[17]={{mem[17][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[17][255:0]}};
                mem[18]={{mem[18][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[18][255:0]}};
                mem[19]={{mem[19][1535:384]},16'h0000, 16'hffff, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[19][255:0]}};
                mem[20]={{mem[20][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[20][255:0]}};
                mem[21]={{mem[21][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[21][255:0]}};
                mem[22]={{mem[22][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][255:0]}};
                mem[23]={{mem[23][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][255:0]}};
                mem[24]={{mem[24][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[24][255:0]}};
                mem[25]={{mem[25][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[25][255:0]}};
                end
                6:
                begin
                mem[15]={{mem[15][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[15][255:0]}};
                mem[16]={{mem[16][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[16][255:0]}};
                mem[17]={{mem[17][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
```

```
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[17][255:0]}};
                mem[18]={{mem[18][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[18][255:0]}};
                mem[19]={{mem[19][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[19][255:0]}};
                mem[20]={{mem[20][1535:384]},16'h0000, 16'hffff, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[20][255:0]}};
                mem[21]={{mem[21][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[21][255:0]}};
                mem[22]={{mem[22][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][255:0]}};
                mem[23]={{mem[23][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][255:0]}};
                mem[24]={{mem[24][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[24][255:0]}};
                mem[25]={{mem[25][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[25][255:0]}};
                end
                7:
                begin
                mem[15]={{mem[15][1535:384]},16'h0000, 16'hffff, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'hffff, 16'h0000,{mem[15][255:0]}};
                mem[16]={{mem[16][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[16][255:0]}};
                mem[17]={{mem[17][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[17][255:0]}};
                mem[18]={{mem[18][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[18][255:0]}};
                mem[19]={{mem[19][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[19][255:0]}};
                mem[20]={{mem[20][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[20][255:0]}};
                mem[21]={{mem[21][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[21][255:0]}};
                mem[22]={{mem[22][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[22][255:0]}};
                mem[23]={{mem[23][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[23][255:0]}};
                mem[24]={{mem[24][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[24][255:0]}};
                mem[25]={{mem[25][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[25][255:0]}};
                end
                8:
                begin
                mem[15]={{mem[15][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[15][255:0]}};
                mem[16]={{mem[16][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[16][255:0]}};
                mem[17]={{mem[17][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[17][255:0]}};
                mem[18]={{mem[18][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[18][255:0]}};
                mem[19]={{mem[19][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[19][255:0]}};
                mem[20]={{mem[20][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[20][255:0]}};
```

```
                mem[21]={{mem[21][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[21][255:0]}};
                mem[22]={{mem[22][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][255:0]}};
                mem[23]={{mem[23][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][255:0]}};
                mem[24]={{mem[24][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[24][255:0]}};
                mem[25]={{mem[25][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[25][255:0]}};
                end
                9:
                begin
                mem[15]={{mem[15][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[15][255:0]}};
                mem[16]={{mem[16][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[16][255:0]}};
                mem[17]={{mem[17][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[17][255:0]}};
                mem[18]={{mem[18][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[18][255:0]}};
                mem[19]={{mem[19][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[19][255:0]}};
                mem[20]={{mem[20][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'hffff, 16'h0000,{mem[20][255:0]}};
                mem[21]={{mem[21][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'hffff, 16'h0000,{mem[21][255:0]}};
                mem[22]={{mem[22][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[22][255:0]}};
                mem[23]={{mem[23][1535:384]},16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[23][255:0]}};
                mem[24]={{mem[24][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[24][255:0]}};
                mem[25]={{mem[25][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[25][255:0]}};
                end
            endcase

        if(angle==7'b1000000 || angle==7'b0000000)
        begin

mem[36]={{mem[36][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[36][511:0]}}};

mem[37]={{mem[37][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[37][511:0]}}};

mem[38]={{mem[38][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[38][511:0]}}};

mem[39]={{mem[39][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[39][511:0]}}};
```

```
mem[40]={{mem[40][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[40][511:0]}}};

mem[41]={{mem[41][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[41][511:0]}}};

mem[42]={{mem[42][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[42][511:0]}}};

mem[43]={{mem[43][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[43][511:0]}}};

mem[44]={{mem[44][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[44][511:0]}}};

mem[45]={{mem[45][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[45][511:0]}}};

mem[46]={{mem[46][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[46][511:0]}}};

mem[47]={{mem[47][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[47][511:0]}}};

mem[48]={{mem[48][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[48][511:0]}}};

mem[49]={{mem[49][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[49][511:0]}}};

mem[50]={{mem[50][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[50][511:0]}}};

mem[51]={{mem[51][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[51][511:0]}}};
    end
    else if (a1==1)
    begin

mem[36]={{mem[36][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,
16'h0000,{mem[36][511:0]}}};

mem[37]={{mem[37][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,
```

16'h0000,{mem[37][511:0]}}};

mem[38]={{mem[38][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,
16'h0000,{mem[38][511:0]}}};

mem[39]={{mem[39][1535:768],16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,
16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'h0000,{mem[39][511:
0]}}};

mem[40]={{mem[40][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'hffff
,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,1
6'h0000,{mem[40][511:0]}}};

mem[41]={{mem[41][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'hffff
,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,1
6'h0000,{mem[41][511:0]}}};

mem[42]={{mem[42][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'hffff
,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,1
6'h0000,{mem[42][511:0]}}};

mem[43]={{mem[43][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000
,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,1
6'h0000,{mem[43][511:0]}}};

mem[44]={{mem[44][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'hffff,
16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'h0000,16'h0000,16'h0000,{me
m[44][511:0]}}};

mem[45]={{mem[45][1535:768],16'h0000,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000
,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'
h0000,{mem[45][511:0]}}};

mem[46]={{mem[46][1535:768],16'h0000,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000
,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'
h0000,{mem[46][511:0]}}};

mem[47]={{mem[47][1535:768],16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'h0000
,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'
h0000,{mem[47][511:0]}}};

mem[48]={{mem[48][1535:768],16'h0000,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000
,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'
h0000,{mem[48][511:0]}}};

mem[49]={{mem[49][1535:768],16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000
,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'
h0000,{mem[49][511:0]}}};

mem[50]={{mem[50][1535:768],16'h0000,16'h0000,16'hffff,16'hffff,16'hffff,16'hffff,16'h
ffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'h0000,{mem[50][5
11:0]}}};

mem[51]={{mem[51][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h000
0,16'h0000,{mem[51][511:0]}}};

```
    end
    else if(a1==0)
    begin

mem[36]={{mem[36][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,
16'h0000,{mem[36][511:0]}}};

mem[37]={{mem[37][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,
16'h0000,{mem[37][511:0]}}};

mem[38]={{mem[38][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h00
00,16'hffff,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,
16'h0000,{mem[38][511:0]}}};

mem[39]={{mem[39][1535:768],16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,
16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'h0000,{mem[39][511:
0]}}};

mem[40]={{mem[40][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'hffff
,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,1
6'h0000,{mem[40][511:0]}}};

mem[41]={{mem[41][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'hffff
,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,1
6'h0000,{mem[41][511:0]}}};

mem[42]={{mem[42][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000
,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,1
6'h0000,{mem[42][511:0]}}};

mem[43]={{mem[43][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000
,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,1
6'h0000,{mem[43][511:0]}}};

mem[44]={{mem[44][1535:768],16'h0000,16'h0000,16'h0000,16'hffff,16'hffff,16'hffff,16'
hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'h0000,16'h0000,16'h0000,{mem[
44][511:0]}}};

mem[45]={{mem[45][1535:768],16'h0000,16'h0000,16'hffff,16'h0000,16'hffff,16'h0000,
16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'
h0000,{mem[45][511:0]}}};

mem[46]={{mem[46][1535:768],16'h0000,16'hffff,16'h0000,16'h0000,16'hffff,16'h0000,
16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'
h0000,{mem[46][511:0]}}};

mem[47]={{mem[47][1535:768],16'hffff,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000,
16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'
h0000,{mem[47][511:0]}}};

mem[48]={{mem[48][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000
,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'
h0000,{mem[48][511:0]}}};

mem[49]={{mem[49][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000
```

```
,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'
h0000,{mem[49][511:0]}};

mem[50]={{mem[50][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'hffff,
16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'hffff,16'h0000,16'h0000,16'h0000,{me
m[50][511:0]}};

mem[51]={{mem[51][1535:768],16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000
,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'h0000,16'hffff,16'h0000,16'h0000,16'
h0000,{mem[51][511:0]}};
    end


    case(a2)
        0:
        begin
        mem[39]={{mem[39][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[39][383:0]}};
        mem[40]={{mem[40][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[40][383:0]}};
        mem[41]={{mem[41][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[41][383:0]}};
        mem[42]={{mem[42][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[42][383:0]}};
        mem[43]={{mem[43][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[43][383:0]}};
        mem[44]={{mem[44][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[44][383:0]}};
        mem[45]={{mem[45][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[45][383:0]}};
        mem[46]={{mem[46][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[46][383:0]}};
        mem[47]={{mem[47][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[47][383:0]}};
        mem[48]={{mem[48][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[48][383:0]}};
        mem[49]={{mem[49][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[49][383:0]}};

        mem[39]={{mem[39][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[39][255:0]}};
        mem[40]={{mem[40][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[40][255:0]}};
        mem[41]={{mem[41][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[41][255:0]}};
        mem[42]={{mem[42][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[42][255:0]}};
        mem[43]={{mem[43][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[43][255:0]}};
        mem[44]={{mem[44][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[44][255:0]}};
        mem[45]={{mem[45][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[45][255:0]}};
        mem[46]={{mem[46][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[46][255:0]}};
        mem[47]={{mem[47][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[47][255:0]}};
```

```
            mem[48]={{mem[48][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[48][255:0]}};
            mem[49]={{mem[49][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[49][255:0]}};
            end
            15:
            begin
            mem[39]={{mem[39][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[39][383:0]}};
            mem[40]={{mem[40][1535:512]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[40][383:0]}};
            mem[41]={{mem[41][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[41][383:0]}};
            mem[42]={{mem[42][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[42][383:0]}};
            mem[43]={{mem[43][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[43][383:0]}};
            mem[44]={{mem[44][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[44][383:0]}};
            mem[45]={{mem[45][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[45][383:0]}};
            mem[46]={{mem[46][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[46][383:0]}};
            mem[47]={{mem[47][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[47][383:0]}};
            mem[48]={{mem[48][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[48][383:0]}};
            mem[49]={{mem[49][1535:512]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'hffff, 16'h0000,{mem[49][383:0]}};

            mem[39]={{mem[39][1535:384]},16'h0000, 16'hffff, 16'hffff, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000,{mem[39][255:0]}};
            mem[40]={{mem[40][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[40][255:0]}};
            mem[41]={{mem[41][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[41][255:0]}};
            mem[42]={{mem[42][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[42][255:0]}};
            mem[43]={{mem[43][1535:384]},16'h0000, 16'hffff, 16'hffff, 16'hffff, 16'hffff,
16'h0000, 16'h0000, 16'h0000,{mem[43][255:0]}};
            mem[44]={{mem[44][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[44][255:0]}};
            mem[45]={{mem[45][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[45][255:0]}};
            mem[46]={{mem[46][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[46][255:0]}};
            mem[47]={{mem[47][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[47][255:0]}};
            mem[48]={{mem[48][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[48][255:0]}};
            mem[49]={{mem[49][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[49][255:0]}};
            end
            30:
            begin
            mem[39]={{mem[39][1535:512]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[39][383:0]}};
```

```
            mem[40]={{mem[40][1535:512]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[40][383:0]}};
            mem[41]={{mem[41][1535:512]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[41][383:0]}};
            mem[42]={{mem[42][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[42][383:0]}};
            mem[43]={{mem[43][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[43][383:0]}};
            mem[44]={{mem[44][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[44][383:0]}};
            mem[45]={{mem[45][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[45][383:0]}};
            mem[46]={{mem[46][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[46][383:0]}};
            mem[47]={{mem[47][1535:512]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[47][383:0]}};
            mem[48]={{mem[48][1535:512]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[48][383:0]}};
            mem[49]={{mem[49][1535:512]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[49][383:0]}};

            mem[39]={{mem[39][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[39][255:0]}};
            mem[40]={{mem[40][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[40][255:0]}};
            mem[41]={{mem[41][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[41][255:0]}};
            mem[42]={{mem[42][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[42][255:0]}};
            mem[43]={{mem[43][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[43][255:0]}};
            mem[44]={{mem[44][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[44][255:0]}};
            mem[45]={{mem[45][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[45][255:0]}};
            mem[46]={{mem[46][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[46][255:0]}};
            mem[47]={{mem[47][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[47][255:0]}};
            mem[48]={{mem[48][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[48][255:0]}};
            mem[49]={{mem[49][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[49][255:0]}};
            end
            45:
            begin
            mem[39]={{mem[39][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[39][383:0]}};
            mem[40]={{mem[40][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[40][383:0]}};
            mem[41]={{mem[41][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'hffff, 16'hffff, 16'h0000, 16'h0000,{mem[41][383:0]}};
            mem[42]={{mem[42][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[42][383:0]}};
            mem[43]={{mem[43][1535:512]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[43][383:0]}};
            mem[44]={{mem[44][1535:512]},16'h0000, 16'h0000, 16'hffff, 16'h0000,
```

```
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[44][383:0]}};
            mem[45]={{mem[45][1535:512]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[45][383:0]}};
            mem[46]={{mem[46][1535:512]},16'h0000, 16'hffff, 16'hffff, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'hffff,{mem[46][383:0]}};
            mem[47]={{mem[47][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[47][383:0]}};
            mem[48]={{mem[48][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[48][383:0]}};
            mem[49]={{mem[49][1535:512]},16'h0000, 16'h0000, 16'h0000, 16'hffff,
16'hffff, 16'hffff, 16'hffff, 16'hffff,{mem[49][383:0]}};

            mem[39]={{mem[39][1535:384]},16'h0000, 16'hffff, 16'hffff, 16'hffff, 16'hffff,
16'hffff, 16'hffff, 16'h0000,{mem[39][255:0]}};
            mem[40]={{mem[40][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[40][255:0]}};
            mem[41]={{mem[41][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[41][255:0]}};
            mem[42]={{mem[42][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'h0000, 16'h0000,{mem[42][255:0]}};
            mem[43]={{mem[43][1535:384]},16'h0000, 16'hffff, 16'hffff, 16'hffff, 16'hffff,
16'h0000, 16'h0000, 16'h0000,{mem[43][255:0]}};
            mem[44]={{mem[44][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[44][255:0]}};
            mem[45]={{mem[45][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[45][255:0]}};
            mem[46]={{mem[46][1535:384]},16'h0000, 16'h0000, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[46][255:0]}};
            mem[47]={{mem[47][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'h0000, 16'hffff, 16'h0000,{mem[47][255:0]}};
            mem[48]={{mem[48][1535:384]},16'h0000, 16'hffff, 16'h0000, 16'h0000,
16'h0000, 16'hffff, 16'h0000, 16'h0000,{mem[48][255:0]}};
            mem[49]={{mem[49][1535:384]},16'h0000, 16'h0000, 16'hffff, 16'hffff,
16'hffff, 16'h0000, 16'h0000, 16'h0000,{mem[49][255:0]}};
            end
      endcase
end
end


reg clk_div;
reg[15:0] clk_cnt=0;
always@(posedge clk_in or negedge rst_n_in)
begin
  if(!rst_n_in) clk_cnt<=0;
  else begin
        clk_cnt<=clk_cnt+1;
        if(clk_cnt==(CLK_DIV_PERIOD-1)) clk_cnt<=0;
        if(clk_cnt<(CLK_DIV_PERIOD/2)) clk_div<=0;
        else clk_div<=1;
  end
end
//divide clk_div 4 state, RISING and FALLING state is keeped one cycle of clk_in, like a
pulse.
reg[1:0] clk_div_state=CLK_L;
always@(posedge clk_in or negedge rst_n_in)
begin
```

```
  if(!rst_n_in) clk_div_state<=CLK_L;
      else
        case(clk_div_state)
              CLK_L: begin
                        if (clk_div) clk_div_state<=CLK_RISING_DEGE;
                        else clk_div_state<=CLK_L;
                    end
              CLK_RISING_DEGE :clk_div_state<=CLK_H;
              CLK_H:begin
                        if (!clk_div) clk_div_state<=CLK_FALLING_DEGE;
                        else clk_div_state<=CLK_H;
                    end
              CLK_FALLING_DEGE:clk_div_state<=CLK_L;
              default;
        endcase
end
reg shift_flag = 0;
reg[6:0] x_reg;
reg[2:0] y_reg;
reg[7:0] char_reg;
reg[8:0] temp_cnt;
reg[7:0] data_reg;
reg[2:0] data_state=IDLE;
reg[2:0] data_state_back;
reg[7:0] data_state_cnt=0;
reg[3:0] shift_cnt=0;
reg[25:0] delay_cnt=0;
//Finite State Machine,
always@(posedge clk_in or negedge rst_n_in)
begin
  if(!rst_n_in)
        begin
              data_state<=IDLE;
              run_flag <= 1;
              data_state_cnt<=0;
              shift_flag <= 0;
              oled_cs_n_out<=HIGH;
        end
      else begin
        case (data_state)
              IDLE: begin
                        oled_cs_n_out<=HIGH;
                        data_state_cnt<=data_state_cnt+1;
                        case(data_state_cnt)
                              0: oled_rst_n_out <= 0;
                              1: data_state<=DELAY;
                              2: oled_rst_n_out <= 1;
                              3: data_state<=DELAY;
                              //display initial
                              4: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=0;
end
                              5: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=1;
end
                              6: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=2;
```

```
end
                        7: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=3;
end
                        8: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=4;
end
                        //clear display
                        9: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=5;
end
                        //10: begin data_state<=CLEAR;data_state_back<=CLEAR;
end
                        10: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=6;
end
                        11: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=7;
end
                        /*12: begin data_state<=CLEAR;data_state_back<=CLEAR;
end
                        13: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=7;
end
                        14: begin data_state<=CLEAR;data_state_back<=CLEAR; end
                        15: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=8;
end
                        16: begin data_state<=CLEAR;data_state_back<=CLEAR; end
                        17: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=9;
end
                        18: begin data_state<=CLEAR;data_state_back<=CLEAR; end
                        19: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=10;
end
                        20: begin data_state<=CLEAR;data_state_back<=CLEAR; end
                        21: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=11;
end
                        22: begin data_state<=CLEAR;data_state_back<=CLEAR; end
                        23: begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=CMD;char_reg<=12;
end
                        24: begin data_state<=CLEAR;data_state_back<=CLEAR;
end*/
                        //set start point
                        25 : begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=DATA;char_reg<=0;
end
                            26 : begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=DATA;char_reg<=1;
end
                            27 : begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=DATA;char_reg<=2;
end
                            28 : begin
```

```
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=DATA;char_reg<=3;
end
                                29 : begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=DATA;char_reg<=4;
end
//……这里代码重复度较高，故不再展示
                                88 : begin
data_state<=DISPLAY;data_state_back<=DISPLAY;oled_dc_out<=DATA;char_reg<=63;
end
                                89:    begin data_state_cnt<=17;run_flag <= ~run_flag; end
                                default;
                            endcase
                    end

                SHIFT: begin
                            if(!shift_flag)
                                begin
                                    if (clk_div_state==CLK_FALLING_DEGE)
                                        begin
                                            if (shift_cnt==8)
                                                begin
                                                    shift_cnt<=0;
                                                    data_state<=data_state_back;
                                                end
                                            else begin
                                                    oled_cs_n_out<=LOW;
                                                    oled_data_out<=data_reg[7];
                                                    shift_flag <= 1;
                                                end
                                        end
                                end
                            else
                                begin
                                    if (clk_div_state==CLK_RISING_DEGE)
                                        begin
                                            data_reg<={data_reg[6:0], data_reg[7]};
                                            shift_cnt<=shift_cnt+1;
                                            shift_flag <= 0;
                                        end
                                end
                    end
                DISPLAY: begin
                                temp_cnt<=temp_cnt+1;
                                oled_cs_n_out<=HIGH;
                                if (temp_cnt==192)
                                    begin
                                        data_state<=IDLE;
                                        temp_cnt<=0;
                                    end
                                else
                                    begin
                                        temp = (oled_dc_out==CMD)?
cmd_r[char_reg]:mem[char_reg];
                                        case (temp_cnt)
                                          0 :    data_reg<=temp[1535:1528];
                                            1 :    data_reg<=temp[1527:1520];
                                            2 :    data_reg<=temp[1519:1512];
```

```
                                3 :    data_reg<=temp[1511:1504];
                                4 :    data_reg<=temp[1503:1496];
                            //……这里代码重复度较高，故不再展示
                                189 :    data_reg<=temp[23:16];
                                190 :    data_reg<=temp[15:8];
                                191 :    data_reg<=temp[7:0];
                                  default;
                            endcase
                            data_state<=SHIFT;
                        end
                end

            CLEAR: begin
                        data_reg<=8'h00;
                        temp_cnt<=temp_cnt+1;
                        oled_cs_n_out<=HIGH;
                        oled_dc_out<=DATA;
                        if (temp_cnt>=128)
                            begin
                                temp_cnt<=0;
                                data_state<=IDLE;
                            end
                        else data_state<=SHIFT;
                    end

            DELAY: begin
                        if(delay_cnt==DELAY_PERIOD)
                            begin
                                data_state<=IDLE;
                                delay_cnt<=0;
                            end
                        else delay_cnt<=delay_cnt+1;
                    end

                default;
            endcase
        end
    end
    assign oled_clk_out = oled_cs_n_out?0:clk_div;
    endmodule
```

⑧ xuanniu 模块

1. 模块功能描述

该模块用于识别旋钮的左右旋转，并由此进行车体前轮转动方向的控制，同时为了保证该模块在运行过程中能够较为平稳地识别旋钮的转动，还特别加入了防抖模块。

旋钮左右旋转方向的识别主要是通过对 SIA 与 SIB 信号哪一个先变化进行识别，然后其分别先后变化的差别区分其是向左转动还是向右转动。

值得特别指出的是，由于旋钮旋转过程中，两组输入数据可能会出现一定的抖动，所以本项目特别加入了旋钮的防抖设计。

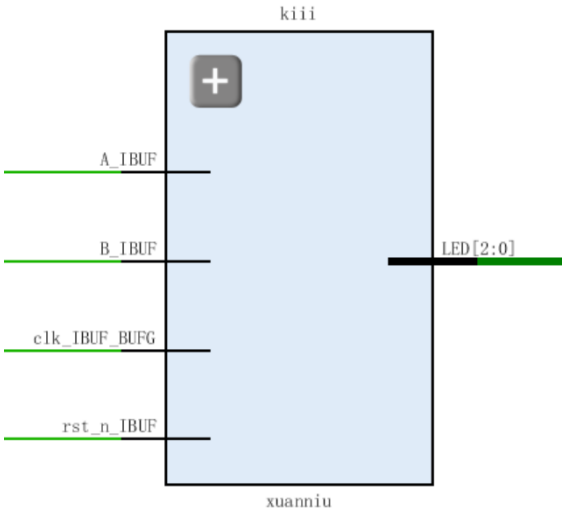
防抖的原理在于：首先使用分频器将时钟的频率降低，之后再连续定义 A_reg、A_reg0 寄存器，共同存储 SIA 接口输出的数据，前一个寄存器的值在下一个时刻会被转移到下一个寄存器，只有当两者寄存器的值相同的时候，才会将

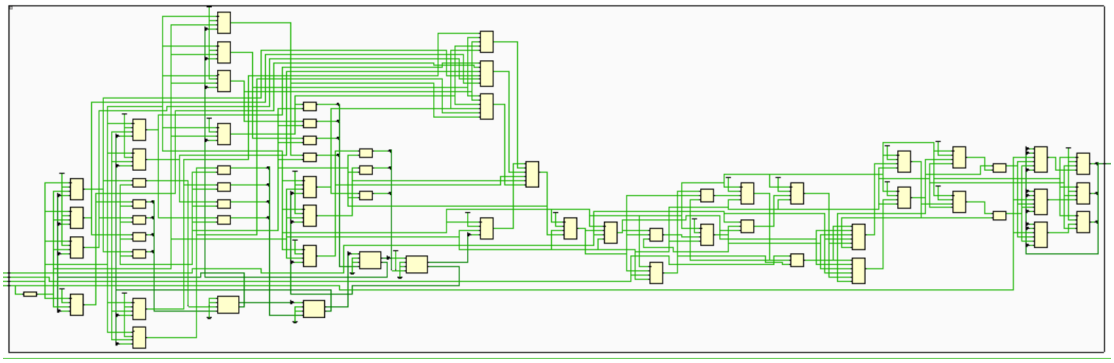
该值传递给存储 SIA 信号的寄存器 A_reg。

之后在处理 A_reg 内的值时，定义一个后继 A_Debounce 寄存器，该寄存器用于存储 A_reg 上一个时钟沿时 A_reg 的值，当二者不同时，可以得到当前 SIA 处于上升沿或是下降沿。

再检查当前时刻下，SIB 传递信息是否处于变化沿，便可得到旋钮当前的输入状态。

2. 功能框图





3. 接口信号说明

管脚名称	属性	具体描述
clk	input	传入系统的 100MHz 时钟
rst_n	input	复位信号，低电平有效
A	input	由 SIA 管脚输出的信号
B	input	由 SIB 管脚输出的信号
BTN	input	当旋钮被按下时，该位置 0
LED	output [2:0]	输出当前用户所控制的车体的偏转角度

4. verilog 实现

```
module xuanniu(
input clk,
input rst_n,
```

```
input A,
input B,
input BTN,
output [2:0] LED
);

reg clk_10ms;
reg [31:0]count;

reg A_reg,A_reg0;
reg B_reg,B_reg0;
reg BTN_reg,BTN_reg0;
wire A_Debounce;
wire B_Debounce;
wire BTN_Debounce;

reg A_Debounce_reg;
reg B_Debounce_reg;
wire A_pos,A_neg;
wire B_pos,B_neg;

reg rotary_right;
reg rotary_left;

reg rotary_right_reg,rotary_left_reg;
wire rotary_right_pos,rotary_left_pos;
wire rotary_event;

reg [2:0]shift_d;

always@(posedge clk,negedge rst_n)begin
if(!rst_n)begin
count <= 0;
clk_10ms <= 1'b0;
end
else begin
if(count < 32'd24_999)begin//10ms 消抖，25MCLK
count <= count + 1'b1;
clk_10ms <= 1'b0;
end
else begin
count <= 0;
clk_10ms <= 1'b1;
end
end
end

always@(posedge clk,negedge rst_n)begin
if(!rst_n)begin
A_reg <= 1'b1;
A_reg0 <= 1'b1;
B_reg <= 1'b1;
B_reg0 <= 1'b1;
BTN_reg <= 1'b0;
BTN_reg0 <= 1'b0;
end
else begin
```

```
if(clk_10ms)begin
A_reg <= A;
A_reg0 <= A_reg;
B_reg <= B;
B_reg0 <= B_reg;
BTN_reg <= BTN;
BTN_reg0 <= BTN_reg;
end
end
end

assign A_Debounce = A_reg0 && A_reg && A;
assign B_Debounce = B_reg0 && B_reg && B;
assign BTN_Debounce = BTN_reg0 && BTN_reg && BTN;//消抖后制作脉冲上升沿

always@(posedge clk,negedge rst_n)begin
if(!rst_n)begin
A_Debounce_reg <= 1'b1;
//B_Debounce_reg <= 1'b1;
end
else begin
A_Debounce_reg <= A_Debounce;
//B_Debounce_reg <= B_Debounce;
end
end

assign A_pos = !A_Debounce_reg && A_Debounce;
//assign B_pos = !B_Debounce_reg && B_Debounce;

assign A_neg = A_Debounce_reg && !A_Debounce;
//assign B_neg = B_Debounce_reg && !B_Debounce;

always@(posedge clk,negedge rst_n)begin
if(!rst_n)begin
rotary_right <= 1'b1;
rotary_left <= 1'b1;
end
else begin
if(A_pos && !B_Debounce)begin//A 的上升沿时候如果 B 为低电平，则旋转编码器是向右转
rotary_right <= 1'b1;
end

if(A_pos && B_Debounce)begin//A 上升沿时候如果 B 为低电平，则旋转编码器是向左转
rotary_left <= 1'b1;
end

if(A_neg && B_Debounce)begin//A 的下降沿 B 为高电平，则向右转结束
rotary_right <= 1'b0;
end

if(A_neg && !B_Debounce)begin//A 的下降沿 B 为低电平，则向左转结束
rotary_left <= 1'b0;
end
end
```

```
end

//assign Rotary_left = rotary_left;
//assign Rotary_right = rotary_right;
always@(posedge clk,negedge rst_n)begin
if(!rst_n)begin
rotary_right_reg <= 1'b1;
rotary_left_reg <= 1'b1;
end
else begin
rotary_right_reg <= rotary_right;
rotary_left_reg <= rotary_left;
end
end

assign rotary_right_pos = !rotary_right_reg && rotary_right;
assign rotary_left_pos = !rotary_left_reg && rotary_left;//消抖

assign rotary_event = rotary_right_pos || rotary_left_pos;//转动标志位

always@(posedge clk,negedge rst_n)begin
if(!rst_n)
shift_d <= 3'b100;
else if(rotary_event)begin
if(rotary_right_pos)
begin
      case(shift_d)
            3'b011:shift_d=3'b011;
            3'b010:shift_d=3'b011;
            3'b001:shift_d=3'b010;
            3'b100:shift_d=3'b001;
            3'b101:shift_d=3'b100;
            3'b110:shift_d=3'b101;
            3'b111:shift_d=3'b110;
       endcase
end
if(rotary_left_pos)
begin
      case(shift_d)
            3'b011:shift_d=3'b010;
            3'b010:shift_d=3'b001;
            3'b001:shift_d=3'b100;
            3'b100:shift_d=3'b101;
            3'b101:shift_d=3'b110;
            3'b110:shift_d=3'b111;
            3'b111:shift_d=3'b111;
       endcase
end
end
end

assign LED = shift_d;//灯的向左向右的测试模块

endmodule
```

2）车载端模块说明

在这里要说明的是，由于蓝牙模块的配置，遥控端与车载端的情况基本相似，所以不在此进行特别说明，仅在此针对车载端所特有的模块内容进行说明。

此外，本项目的车载端分别在 distance 子模块与 duoji 子模块下使用了类似于遥控端 Dividerhaha 分频器作为辅助构造模块，其模块名分别为 Divider（分频倍数为 1/1000）与 Divider1（分频倍数为 1/500）。考虑到这些模块的实现与 Dividerhaha 的实现并无区别，所以将不再在下文赘述。

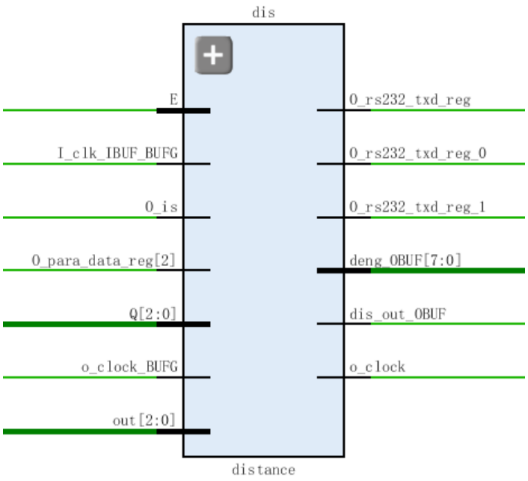
① distance 模块

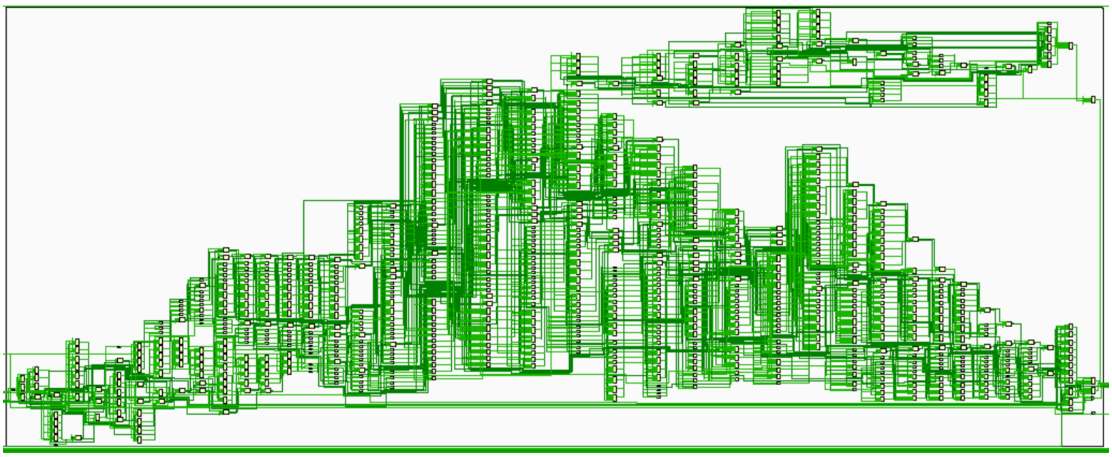
1. 模块功能描述

该模块是测距功能的主要实现模块，测距功能的实现主要在于 HC-SR04 部件会将其发出声波与接收到声波之间的时间长度，以高电平占比的形式传递给 distance 模块。

而在该模块中，我们通过记录每 60000 个时间刻内，处于高电平的时间刻的个数，之后计算高电平时间刻与总时间刻的比值，再除以光速以及单位间的换算比例，即可得到小车与障碍物之间相距的厘米数，并将之存于 data，之后再将 data 数据回传即可。

2. 功能框图





3. 接口信号说明

管脚名称	属性	具体描述
in	input	输入由 HC-SR04 部件生成的电位信号
ou	output	向 HC-SR04 部件发送开始输出使能信号
clk	input	输入开发板本身 100MHz 频率时钟
data	output [7:0]	输出换算后的障碍物距离

4. verilog 实现

```
module distance(
    input in,
    output reg ou,
    input clk,
    output [7:0] data
    );
    wire clk_;
    reg [32:0]kkk;
    Divider uutt (clk,0,clk_);

    integer i=0;
    always @(posedge clk_)
    begin
        if(i==0) ou=1;
        else ou=0;
        i=i+1;
        if(i==60000) i=0;
    end

    integer j=0;
    always @ (posedge clk_)
    begin
        if(in==1)
        begin
        j=j+1;
        kkk=j;
        end
        else       j=0;
    end

    assign data=kkk*10/58;

endmodule
```

② DEAL_DATA 模块

1. 模块功能描述

该模块的功能是对要从蓝牙模块发送出去的信息进行编码，将角度信息与距离的信息按照信息标志位的不同赋值给蓝牙发送信息的后七位，从而实现反馈信息的编译，并实现由车体向遥控端进行距离与角度信息的传递。

3. 接口信号说明

管脚名称	属性	具体描述
data_in_dis	input	输入距离障碍物的距离信号
data_in_round	output	输入转向的角度信号

clk	input	输入开发板本身 100MHz 时钟
data	output [7:0]	输出要进行蓝牙发送的反馈信号

4. verilog 实现

```
module DEAL_DATA(
    input [7:0]data_in_dis,
    input [7:0]data_in_round,
    input is_distance,
    output reg [7:0]data_out
    );
    always@(*)
    begin
        if(is_distance==1)
        begin
            data_out=data_in_dis;
            data_out[7]=1;
        end
        else
        begin
            case({data_in_round[1],data_in_round[2]})
            3'b00:data_out=0;
            3'b01:data_out=8'd15;
            3'b10:data_out=8'd30;
            3'b11:data_out=8'd45;
            endcase
            data_out[7]=0;
            data_out[6]=data_in_round[0];
        end
    end
endmodule
```

③ duoji 模块

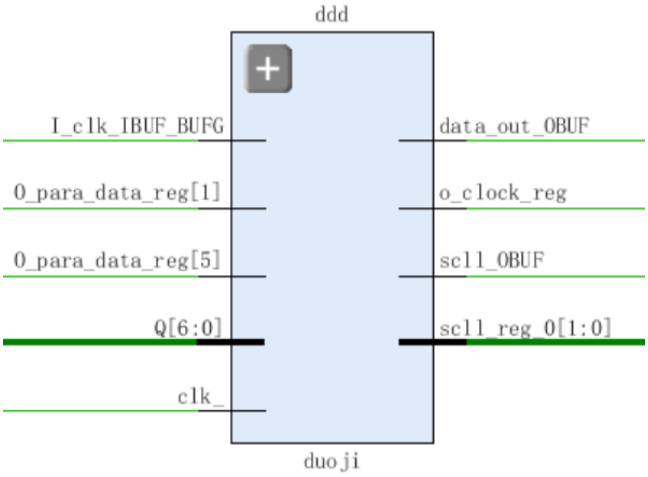
1. 模块功能描述

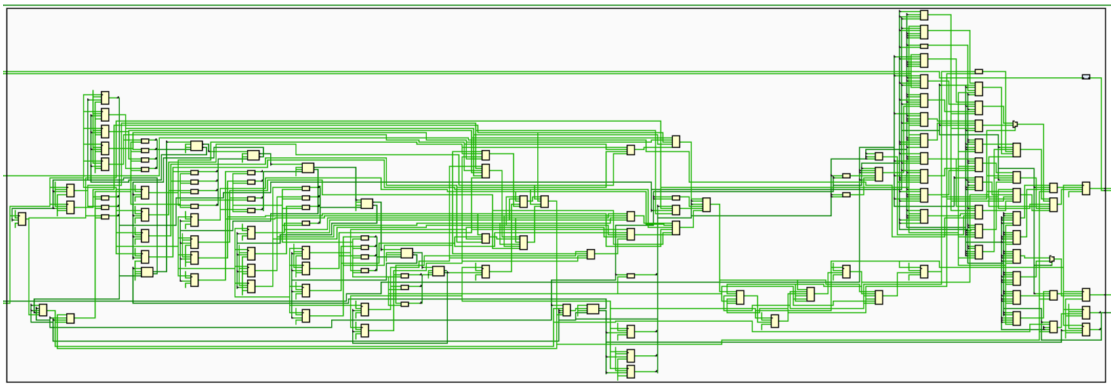
该模块用于 PWM 模块功能的具体实现，通过该模块可以同时实现对两个电机及一个舵机的控制，该控制的实现方式在于：

PWM 模块能够将对应的寄存器内的值根据 PWM 协议转化为对应占长比的波形信号，并在对应路的总线持续输出，直至寄存器的值被改变，开始输出新的占长比波形信号。

所以在这个模块中，为了实现对相应电机与舵机的电压控制，只需要根据 PWM 模块的协议，将待传入并行数据依次输入到对应的寄存器中，即可实现控制。在本项目中，我们拟设定 0 号寄存器与总线对应舵机的电压控制；3 号和 4 号寄存器与总线对应左右后轮驱动的电压控制。

2. 功能框图





3. 接口信号说明

管脚名称	属性	具体描述
inn	input [11:0]	输入要进行控制的速度指令
data_out	output	向 PWM 输出的控制/数据信号
clk	input	输入 100MHz
scll	output	向 PWM 输出的传输控制信号
ain11	output	向 TB6612FNG 输出一号电机的控制信号
ain22	output	协同 ain11 进行控制
bin11	output	向 TB6612FNG 输出二号电机的控制信号
bin22	output	协同 bin11 进行控制
enn	output	电机驱动使能端，低电平有效
iin1	input	输入对一号电机的控制信号
iin2	input	输入对二号电机的控制信号
data	input [2:0]	输入要进行控制的角度指令

4. verilog 实现

```
module duoji(
    input [11:0] inn,
    output reg data_out,
    input clk,
    output reg scll,
```

```
        output ain11,
        output ain22,
        output bin11,
        output bin22,
        output enn,
        input iin1,
        input iin2,
        input [2:0]data
);
        integer i=0,j=0;
        wire clk_;
        reg [11:0]data_in;
        reg [11:0]zhong=12'b0000000101000100;
        ////////////////////////////////////////////////////////////////////////////
        always@(*)
        begin
                case(data)
                3'b000:data_in=zhong;
                3'b100:data_in=zhong;
                3'b001:data_in=zhong+12'b0000000000010001;
                3'b010:data_in=zhong+12'b0000000000010001+12'b0000000000010001;

3'b011:data_in=zhong+12'b0000000000010001+12'b0000000000010001+12'b000000000
0010001;
                3'b101:data_in=zhong-12'b0000000000010001;
                3'b110:data_in=zhong-12'b0000000000010001-12'b0000000000010001;

3'b111:data_in=zhong-12'b0000000000010001-12'b0000000000010001-12'b0000000000
010001;
                endcase
        end
        assign ain11=iin1;
        assign bin11=iin1;
        assign ain22=iin2;
        assign bin22=iin2;

        assign enn=1'b1;

        Divider1 kkk (clk,1'b0,clk_);

        always @ (posedge clk_)
        begin
                case(i)
                        0:
                        begin
                        case(j)
                                0:begin data_out=1'b0; scll=1'b1; end
                                1:data_out=1'b0;
                                2:data_out=1'b0;
                                3:data_out=1'b0;
                                4:data_out=1'b0;
                                5:data_out=1'b1;
                                6:data_out=1'b0;
                                7:scll=1'b0;

                                8:data_out=1'b1;
                                9:scll=1'b1;
```

10:scll=1'b0;
11:data_out=1'b0;
12:scll=1'b1;
13:scll=1'b0;
14:data_out=1'b0;
15:scll=1'b1;
16:scll=1'b0;
17:data_out=1'b0;
18:scll=1'b1;
19:scll=1'b0;
20:data_out=1'b0;
21:scll=1'b1;
22:scll=1'b0;
23:data_out=1'b0;
24:scll=1'b1;
25:scll=1'b0;
26:data_out=1'b0;
27:scll=1'b1;
28:scll=1'b0;
29:data_out=1'b0;
30:scll=1'b1;
31:scll=1'b0;
33:scll=1'b1;
34:scll=1'b0;

35:data_out=1'b1;
36:scll=1'b1;
37:scll=1'b0;
38:data_out=1'b1;
39:scll=1'b1;
40:scll=1'b0;
41:data_out=1'b1;
42:scll=1'b1;
43:scll=1'b0;
44:data_out=1'b1;
45:scll=1'b1;
46:scll=1'b0;
47:data_out=1'b1;
48:scll=1'b1;
49:scll=1'b0;
50:data_out=1'b1;
51:scll=1'b1;
52:scll=1'b0;
53:data_out=1'b1;
54:scll=1'b1;
55:scll=1'b0;
56:data_out=1'b0;
57:scll=1'b1;
58:scll=1'b0;
60:scll=1'b1;
61:scll=1'b0;

62:data_out=1'b0;
63:scll=1'b1;
64:scll=1'b0;
65:data_out=1'b1;
66:scll=1'b1;

```
            67:scll=1'b0;
            68:data_out=1'b1;
            69:scll=1'b1;
            70:scll=1'b0;
            71:data_out=1'b1;
            72:scll=1'b1;
            73:scll=1'b0;
            74:data_out=1'b1;
            75:scll=1'b1;
            76:scll=1'b0;
            77:data_out=1'b0;
            78:scll=1'b1;
            79:scll=1'b0;
            80:data_out=1'b0;
            81:scll=1'b1;
            82:scll=1'b0;
            83:data_out=1'b1;
            84:scll=1'b1;
            85:scll=1'b0;
            86:scll=1'b1;
            87:scll=1'b0;

            88:data_out=1'b0;
            89:scll=1'b1;
            90:data_out=1'b1;
            91:data_out=1'b0;
            92:begin j=-1;i=1; end
        endcase
        j=j+1;
        end
1:
        begin
case(j)
        0:begin data_out=1'b0; scll=1'b1; end
        1:data_out=1'b0;
        2:data_out=1'b0;
        3:data_out=1'b0;
        4:data_out=1'b0;
        5:data_out=1'b1;
        6:data_out=1'b0;
        7:scll=1'b0;

        8:data_out=1'b1;
        9:scll=1'b1;
        10:scll=1'b0;
        11:data_out=1'b0;
        12:scll=1'b1;
        13:scll=1'b0;
        14:data_out=1'b0;
        15:scll=1'b1;
        16:scll=1'b0;
        17:data_out=1'b0;
        18:scll=1'b1;
        19:scll=1'b0;
        20:data_out=1'b0;
        21:scll=1'b1;
        22:scll=1'b0;
```

23:data_out=1'b0;
24:scll=1'b1;
25:scll=1'b0;
26:data_out=1'b0;
27:scll=1'b1;
28:scll=1'b0;
29:data_out=1'b0;
30:scll=1'b1;
31:scll=1'b0;
33:scll=1'b1;
34:scll=1'b0;

35:data_out=1'b0;
36:scll=1'b1;
37:scll=1'b0;
38:data_out=1'b0;
39:scll=1'b1;
40:scll=1'b0;
41:data_out=1'b0;
42:scll=1'b1;
43:scll=1'b0;
44:data_out=1'b0;
45:scll=1'b1;
46:scll=1'b0;
47:data_out=1'b0;
48:scll=1'b1;
49:scll=1'b0;
50:data_out=1'b0;
51:scll=1'b1;
52:scll=1'b0;
53:data_out=1'b0;
54:scll=1'b1;
55:scll=1'b0;
56:data_out=1'b0;
57:scll=1'b1;
58:scll=1'b0;
60:scll=1'b1;
61:scll=1'b0;

62:data_out=1'b0;
63:scll=1'b1;
64:scll=1'b0;
65:data_out=1'b0;
66:scll=1'b1;
67:scll=1'b0;
68:data_out=1'b0;
69:scll=1'b1;
70:scll=1'b0;
71:data_out=1'b0;
72:scll=1'b1;
73:scll=1'b0;
74:data_out=1'b0;
75:scll=1'b1;
76:scll=1'b0;
77:data_out=1'b0;
78:scll=1'b1;
79:scll=1'b0;

```
        80:data_out=1'b0;
        81:scll=1'b1;
        82:scll=1'b0;
        83:data_out=1'b1;
        84:scll=1'b1;
        85:scll=1'b0;
        86:scll=1'b1;
        87:scll=1'b0;

        88:data_out=1'b0;
        89:scll=1'b1;
        90:data_out=1'b1;
        91:data_out=1'b0;
        92:begin j=-1;i=2; end
endcase
j=j+1;
end
2:
        begin
case(j)
        0:begin data_out=1'b0; scll=1'b1; end
        1:data_out=1'b0;
        2:data_out=1'b0;
        3:data_out=1'b0;
        4:data_out=1'b0;
        5:data_out=1'b1;
        6:data_out=1'b0;
        7:scll=1'b0;

        8:data_out=1'b1;
        9:scll=1'b1;
        10:scll=1'b0;
        11:data_out=1'b0;
        12:scll=1'b1;
        13:scll=1'b0;
        14:data_out=1'b0;
        15:scll=1'b1;
        16:scll=1'b0;
        17:data_out=1'b0;
        18:scll=1'b1;
        19:scll=1'b0;
        20:data_out=1'b0;
        21:scll=1'b1;
        22:scll=1'b0;
        23:data_out=1'b0;
        24:scll=1'b1;
        25:scll=1'b0;
        26:data_out=1'b0;
        27:scll=1'b1;
        28:scll=1'b0;
        29:data_out=1'b0;
        30:scll=1'b1;
        31:scll=1'b0;
        33:scll=1'b1;
        34:scll=1'b0;

        35:data_out=1'b0;
```

36:scll=1'b1;
37:scll=1'b0;
38:data_out=1'b0;
39:scll=1'b1;
40:scll=1'b0;
41:data_out=1'b0;
42:scll=1'b1;
43:scll=1'b0;
44:data_out=1'b0;
45:scll=1'b1;
46:scll=1'b0;
47:data_out=1'b1;
48:scll=1'b1;
49:scll=1'b0;
50:data_out=1'b0;
51:scll=1'b1;
52:scll=1'b0;
53:data_out=1'b0;
54:scll=1'b1;
55:scll=1'b0;
56:data_out=1'b0;
57:scll=1'b1;
58:scll=1'b0;
60:scll=1'b1;
61:scll=1'b0;

62:data_out=data_in[7];
63:scll=1'b1;
64:scll=1'b0;
65:data_out=data_in[6];
66:scll=1'b1;
67:scll=1'b0;
68:data_out=data_in[5];
69:scll=1'b1;
70:scll=1'b0;
71:data_out=data_in[4];
72:scll=1'b1;
73:scll=1'b0;
74:data_out=data_in[3];
75:scll=1'b1;
76:scll=1'b0;
77:data_out=data_in[2];
78:scll=1'b1;
79:scll=1'b0;
80:data_out=data_in[1];
81:scll=1'b1;
82:scll=1'b0;
83:data_out=data_in[0];
84:scll=1'b1;
85:scll=1'b0;
86:scll=1'b1;
87:scll=1'b0;

88:data_out=1'b0;
89:scll=1'b1;
90:data_out=1'b1;
91:data_out=1'b0;

```
        92:begin j=-1;i=3; end
endcase
j=j+1;
end
3:
        begin
case(j)
        0:begin data_out=1'b0; scll=1'b1; end
        1:data_out=1'b0;
        2:data_out=1'b0;
        3:data_out=1'b0;
        4:data_out=1'b0;
        5:data_out=1'b1;
        6:data_out=1'b0;
        7:scll=1'b0;

        8:data_out=1'b1;
        9:scll=1'b1;
        10:scll=1'b0;
        11:data_out=1'b0;
        12:scll=1'b1;
        13:scll=1'b0;
        14:data_out=1'b0;
        15:scll=1'b1;
        16:scll=1'b0;
        17:data_out=1'b0;
        18:scll=1'b1;
        19:scll=1'b0;
        20:data_out=1'b0;
        21:scll=1'b1;
        22:scll=1'b0;
        23:data_out=1'b0;
        24:scll=1'b1;
        25:scll=1'b0;
        26:data_out=1'b0;
        27:scll=1'b1;
        28:scll=1'b0;
        29:data_out=1'b0;
        30:scll=1'b1;
        31:scll=1'b0;
        33:scll=1'b1;
        34:scll=1'b0;

        35:data_out=1'b0;
        36:scll=1'b1;
        37:scll=1'b0;
        38:data_out=1'b0;
        39:scll=1'b1;
        40:scll=1'b0;
        41:data_out=1'b0;
        42:scll=1'b1;
        43:scll=1'b0;
        44:data_out=1'b0;
        45:scll=1'b1;
        46:scll=1'b0;
        47:data_out=1'b1;
        48:scll=1'b1;
```

```
                49:scll=1'b0;
                50:data_out=1'b0;
                51:scll=1'b1;
                52:scll=1'b0;
                53:data_out=1'b0;
                54:scll=1'b1;
                55:scll=1'b0;
                56:data_out=1'b1;
                57:scll=1'b1;
                58:scll=1'b0;
                60:scll=1'b1;
                61:scll=1'b0;

                62:data_out=1'b0;
                63:scll=1'b1;
                64:scll=1'b0;
                65:data_out=1'b0;
                66:scll=1'b1;
                67:scll=1'b0;
                68:data_out=1'b0;
                69:scll=1'b1;
                70:scll=1'b0;
                71:data_out=1'b0;
                72:scll=1'b1;
                73:scll=1'b0;
                74:data_out=data_in[11];
                75:scll=1'b1;
                76:scll=1'b0;
                77:data_out=data_in[10];
                78:scll=1'b1;
                79:scll=1'b0;
                80:data_out=data_in[9];
                81:scll=1'b1;
                82:scll=1'b0;
                83:data_out=data_in[8];
                84:scll=1'b1;
                85:scll=1'b0;
                86:scll=1'b1;
                87:scll=1'b0;

                88:data_out=1'b0;
                89:scll=1'b1;
                90:data_out=1'b1;
                91:data_out=1'b0;
                92:begin j=-1;i=4; end
            endcase
            j=j+1;
            end
        4:
                begin
case(j)
        0:begin data_out=1'b0; scll=1'b1; end
        1:data_out=1'b0;
        2:data_out=1'b0;
        3:data_out=1'b0;
        4:data_out=1'b0;
        5:data_out=1'b1;
```

6:data_out=1'b0;
7:scll=1'b0;

8:data_out=1'b1;
9:scll=1'b1;
10:scll=1'b0;
11:data_out=1'b0;
12:scll=1'b1;
13:scll=1'b0;
14:data_out=1'b0;
15:scll=1'b1;
16:scll=1'b0;
17:data_out=1'b0;
18:scll=1'b1;
19:scll=1'b0;
20:data_out=1'b0;
21:scll=1'b1;
22:scll=1'b0;
23:data_out=1'b0;
24:scll=1'b1;
25:scll=1'b0;
26:data_out=1'b0;
27:scll=1'b1;
28:scll=1'b0;
29:data_out=1'b0;
30:scll=1'b1;
31:scll=1'b0;
33:scll=1'b1;
34:scll=1'b0;

35:data_out=1'b0;
36:scll=1'b1;
37:scll=1'b0;
38:data_out=1'b0;
39:scll=1'b1;
40:scll=1'b0;
41:data_out=1'b0;
42:scll=1'b1;
43:scll=1'b0;
44:data_out=1'b1;
45:scll=1'b1;
46:scll=1'b0;
47:data_out=1'b0;
48:scll=1'b1;
49:scll=1'b0;
50:data_out=1'b1;
51:scll=1'b1;
52:scll=1'b0;
53:data_out=1'b0;
54:scll=1'b1;
55:scll=1'b0;
56:data_out=1'b0;
57:scll=1'b1;
58:scll=1'b0;
60:scll=1'b1;
61:scll=1'b0;

```
        62:data_out=inn[7];
        63:scll=1'b1;
        64:scll=1'b0;
        65:data_out=inn[6];
        66:scll=1'b1;
        67:scll=1'b0;
        68:data_out=inn[5];
        69:scll=1'b1;
        70:scll=1'b0;
        71:data_out=inn[4];
        72:scll=1'b1;
        73:scll=1'b0;
        74:data_out=inn[3];
        75:scll=1'b1;
        76:scll=1'b0;
        77:data_out=inn[2];
        78:scll=1'b1;
        79:scll=1'b0;
        80:data_out=inn[1];
        81:scll=1'b1;
        82:scll=1'b0;
        83:data_out=inn[0];
        84:scll=1'b1;
        85:scll=1'b0;
        86:scll=1'b1;
        87:scll=1'b0;

        88:data_out=1'b0;
        89:scll=1'b1;
        90:data_out=1'b1;
        91:data_out=1'b0;
        92:begin j=-1;i=5; end
endcase
j=j+1;
end
        5:
                begin
        case(j)
            0:begin data_out=1'b0; scll=1'b1; end
            1:data_out=1'b0;
            2:data_out=1'b0;
            3:data_out=1'b0;
            4:data_out=1'b0;
            5:data_out=1'b1;
            6:data_out=1'b0;
            7:scll=1'b0;

            8:data_out=1'b1;
            9:scll=1'b1;
            10:scll=1'b0;
            11:data_out=1'b0;
            12:scll=1'b1;
            13:scll=1'b0;
            14:data_out=1'b0;
            15:scll=1'b1;
            16:scll=1'b0;
            17:data_out=1'b0;
```

18:scll=1'b1;
19:scll=1'b0;
20:data_out=1'b0;
21:scll=1'b1;
22:scll=1'b0;
23:data_out=1'b0;
24:scll=1'b1;
25:scll=1'b0;
26:data_out=1'b0;
27:scll=1'b1;
28:scll=1'b0;
29:data_out=1'b0;
30:scll=1'b1;
31:scll=1'b0;
33:scll=1'b1;
34:scll=1'b0;

35:data_out=1'b0;
36:scll=1'b1;
37:scll=1'b0;
38:data_out=1'b0;
39:scll=1'b1;
40:scll=1'b0;
41:data_out=1'b0;
42:scll=1'b1;
43:scll=1'b0;
44:data_out=1'b1;
45:scll=1'b1;
46:scll=1'b0;
47:data_out=1'b0;
48:scll=1'b1;
49:scll=1'b0;
50:data_out=1'b1;
51:scll=1'b1;
52:scll=1'b0;
53:data_out=1'b0;
54:scll=1'b1;
55:scll=1'b0;
56:data_out=1'b1;
57:scll=1'b1;
58:scll=1'b0;
60:scll=1'b1;
61:scll=1'b0;

62:data_out=1'b0;
63:scll=1'b1;
64:scll=1'b0;
65:data_out=1'b0;
66:scll=1'b1;
67:scll=1'b0;
68:data_out=1'b0;
69:scll=1'b1;
70:scll=1'b0;
71:data_out=1'b0;
72:scll=1'b1;
73:scll=1'b0;
74:data_out=inn[11];

```
        75:scll=1'b1;
        76:scll=1'b0;
        77:data_out=inn[10];
        78:scll=1'b1;
        79:scll=1'b0;
        80:data_out=inn[9];
        81:scll=1'b1;
        82:scll=1'b0;
        83:data_out=inn[8];
        84:scll=1'b1;
        85:scll=1'b0;
        86:scll=1'b1;
        87:scll=1'b0;

        88:data_out=1'b0;
        89:scll=1'b1;
        90:data_out=1'b1;
        91:data_out=1'b0;
        92:begin j=-1;i=6; end
endcase
j=j+1;
end
6:
              begin
case(j)
        0:begin data_out=1'b0; scll=1'b1; end
        1:data_out=1'b0;
        2:data_out=1'b0;
        3:data_out=1'b0;
        4:data_out=1'b0;
        5:data_out=1'b1;
        6:data_out=1'b0;
        7:scll=1'b0;

        8:data_out=1'b1;
        9:scll=1'b1;
        10:scll=1'b0;
        11:data_out=1'b0;
        12:scll=1'b1;
        13:scll=1'b0;
        14:data_out=1'b0;
        15:scll=1'b1;
        16:scll=1'b0;
        17:data_out=1'b0;
        18:scll=1'b1;
        19:scll=1'b0;
        20:data_out=1'b0;
        21:scll=1'b1;
        22:scll=1'b0;
        23:data_out=1'b0;
        24:scll=1'b1;
        25:scll=1'b0;
        26:data_out=1'b0;
        27:scll=1'b1;
        28:scll=1'b0;
        29:data_out=1'b0;
        30:scll=1'b1;
```

```
31:scll=1'b0;
33:scll=1'b1;
34:scll=1'b0;

35:data_out=1'b0;
36:scll=1'b1;
37:scll=1'b0;
38:data_out=1'b0;
39:scll=1'b1;
40:scll=1'b0;
41:data_out=1'b0;
42:scll=1'b1;
43:scll=1'b0;
44:data_out=1'b1;
45:scll=1'b1;
46:scll=1'b0;
47:data_out=1'b1;
48:scll=1'b1;
49:scll=1'b0;
50:data_out=1'b0;
51:scll=1'b1;
52:scll=1'b0;
53:data_out=1'b0;
54:scll=1'b1;
55:scll=1'b0;
56:data_out=1'b0;
57:scll=1'b1;
58:scll=1'b0;
60:scll=1'b1;
61:scll=1'b0;

62:data_out=inn[7];
63:scll=1'b1;
64:scll=1'b0;
65:data_out=inn[6];
66:scll=1'b1;
67:scll=1'b0;
68:data_out=inn[5];
69:scll=1'b1;
70:scll=1'b0;
71:data_out=inn[4];
72:scll=1'b1;
73:scll=1'b0;
74:data_out=inn[3];
75:scll=1'b1;
76:scll=1'b0;
77:data_out=inn[2];
78:scll=1'b1;
79:scll=1'b0;
80:data_out=inn[1];
81:scll=1'b1;
82:scll=1'b0;
83:data_out=inn[0];
84:scll=1'b1;
85:scll=1'b0;
86:scll=1'b1;
87:scll=1'b0;
```

```
        88:data_out=1'b0;
        89:scll=1'b1;
        90:data_out=1'b1;
        91:data_out=1'b0;
        92:begin j=-1;i=7; end
endcase
j=j+1;
end
7:
                begin
case(j)
        0:begin data_out=1'b0; scll=1'b1; end
1:data_out=1'b0;
2:data_out=1'b0;
3:data_out=1'b0;
4:data_out=1'b0;
5:data_out=1'b1;
6:data_out=1'b0;
7:scll=1'b0;

8:data_out=1'b1;
9:scll=1'b1;
10:scll=1'b0;
11:data_out=1'b0;
12:scll=1'b1;
13:scll=1'b0;
14:data_out=1'b0;
15:scll=1'b1;
16:scll=1'b0;
17:data_out=1'b0;
18:scll=1'b1;
19:scll=1'b0;
20:data_out=1'b0;
21:scll=1'b1;
22:scll=1'b0;
23:data_out=1'b0;
24:scll=1'b1;
25:scll=1'b0;
26:data_out=1'b0;
27:scll=1'b1;
28:scll=1'b0;
29:data_out=1'b0;
30:scll=1'b1;
31:scll=1'b0;
33:scll=1'b1;
34:scll=1'b0;

35:data_out=1'b0;
36:scll=1'b1;
37:scll=1'b0;
38:data_out=1'b0;
39:scll=1'b1;
40:scll=1'b0;
41:data_out=1'b0;
42:scll=1'b1;
43:scll=1'b0;
```

```
        44:data_out=1'b1;
        45:scll=1'b1;
        46:scll=1'b0;
        47:data_out=1'b1;
        48:scll=1'b1;
        49:scll=1'b0;
        50:data_out=1'b0;
        51:scll=1'b1;
        52:scll=1'b0;
        53:data_out=1'b0;
        54:scll=1'b1;
        55:scll=1'b0;
        56:data_out=1'b1;
        57:scll=1'b1;
        58:scll=1'b0;
        60:scll=1'b1;
        61:scll=1'b0;

        62:data_out=1'b0;
        63:scll=1'b1;
        64:scll=1'b0;
        65:data_out=1'b0;
        66:scll=1'b1;
        67:scll=1'b0;
        68:data_out=1'b0;
        69:scll=1'b1;
        70:scll=1'b0;
        71:data_out=1'b0;
        72:scll=1'b1;
        73:scll=1'b0;
        74:data_out=inn[11];
        75:scll=1'b1;
        76:scll=1'b0;
        77:data_out=inn[10];
        78:scll=1'b1;
        79:scll=1'b0;
        80:data_out=inn[9];
        81:scll=1'b1;
        82:scll=1'b0;
        83:data_out=inn[8];
        84:scll=1'b1;
        85:scll=1'b0;
        86:scll=1'b1;
        87:scll=1'b0;

        88:data_out=1'b0;
        89:scll=1'b1;
        90:data_out=1'b1;
        91:data_out=1'b0;
        92:begin j=-1;i=2; end
        endcase
        j=j+1;
        end
              endcase
        end
endmodule
```

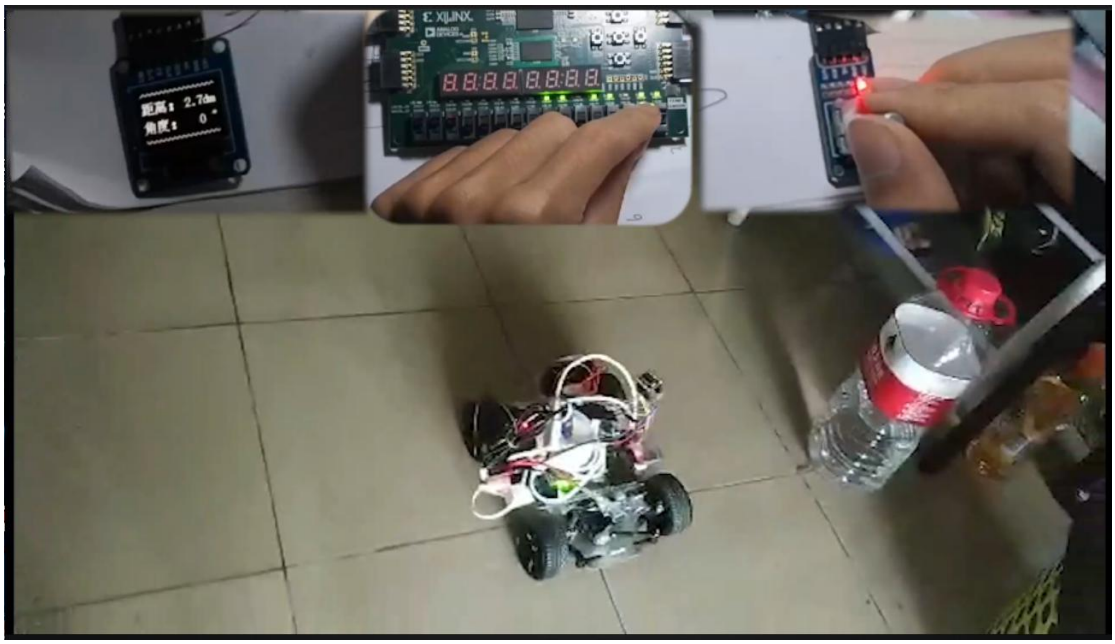
五、测试模块建模

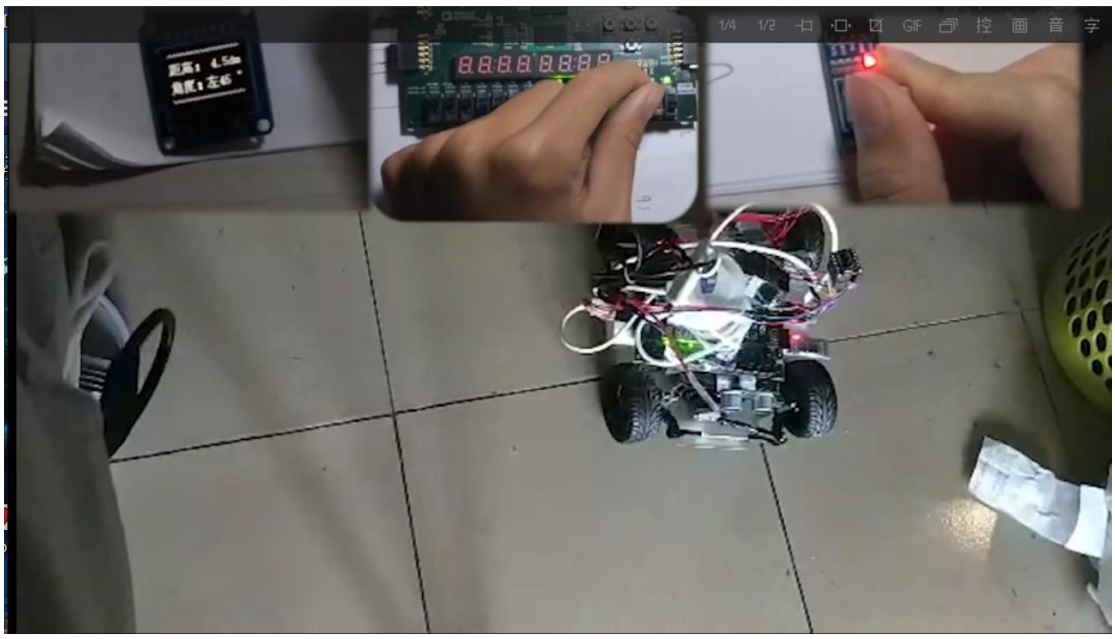
由于本项目的实现主要是基于不同部件之间协议的传递以及两块开发板之间信息的发送与接收，所以在调试过程中，并不需要进行测试模块的编写。

具体调试时，根据不同器件下板后的实际操作与现实，即可完成代码的试错与分析。所以在此并未编写测试模块。

六、实验结果

经实际下板测试，发现最终可以实现遥控端对于小车的实时控制，同时遥控端也可以接受到小车反馈回的信息，并成功显示在 OLED 显示屏上。具体实验结果视频截图如下，分别是测试小车的前进后退与转向功能：





七、结论

根据实际下板情况可以看出，小车的运行情况良好，各个子模块及外围部件都能平稳运行，达到预期的工作目标。由此可说明：

1）本项目成功实现了两块开发板之间的实时蓝牙互传通讯，同时还能根据我们自己设定的协议进行通讯信息的传递；

2）遥控端的 OLED 屏幕运行情况良好，能够实时刷新当前的距离和角度信息，对反馈信号的改变作出及时反应；

3）遥控端的旋转编码器可以有效克服旋转产生的抖动，实现对于车体转向的控制。

4）车载端的 PWM 模块能够顺利实现向对应寄存器和总线的控制，进而实现对舵机以及电机的电压控制；

5）测距模块功能运行平稳，能够实时反馈距离信息。

八、心得体会