同济大学计算机系

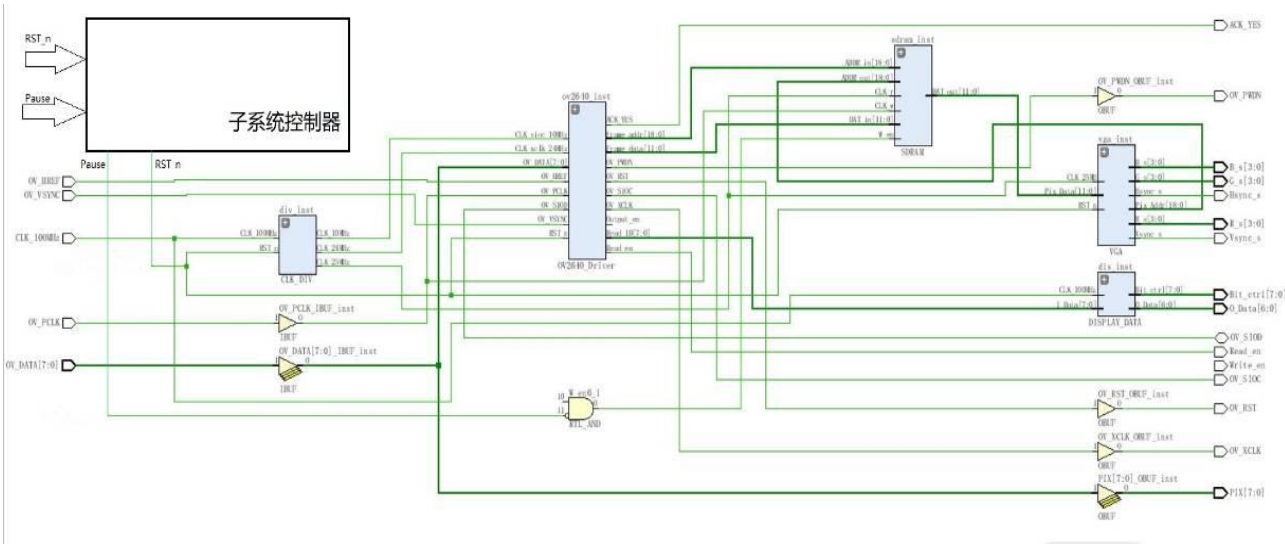
数字逻辑课程综合实验报告

一、实验内容

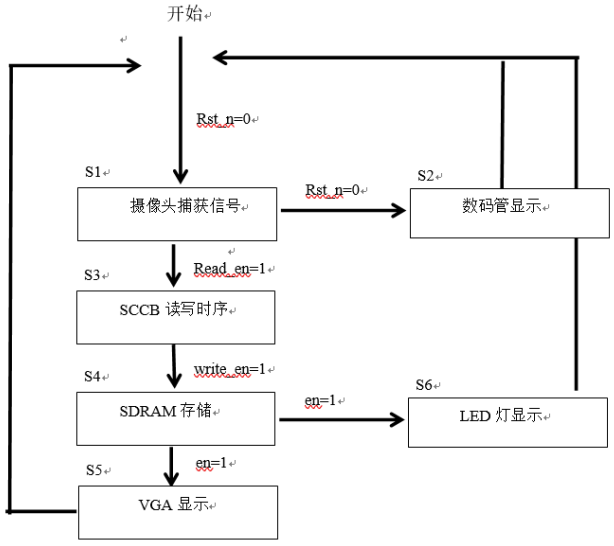
基于 OV2640 摄像头模块的视频显示系统。

使用说明：该视频显示系统的使用方法十分简单。首先将摄像头模块与 VGA 显示器与开发板连接好，在计算机 VIVADO 上生成比特流成功后，利用 USB 数据线将计算机与 FPGA 开发板连接起来，上电，显示器也同时上电，将比特流写入开发板后，初始为待机状态，拨动开发板上的拨动开关 SW[0]（右边第一个），即可在显示器上观察到摄像头的画面，并且七段数码管上显示着摄像头的 ID 地址，可以看到左侧 8 个 LED 灯亮暗有变化，也就是闪烁，表示视频数据流在正常流动，按下方向键的中键（BTNC）可以让显示的画面暂停，松开该中键，画面恢复实时运动。

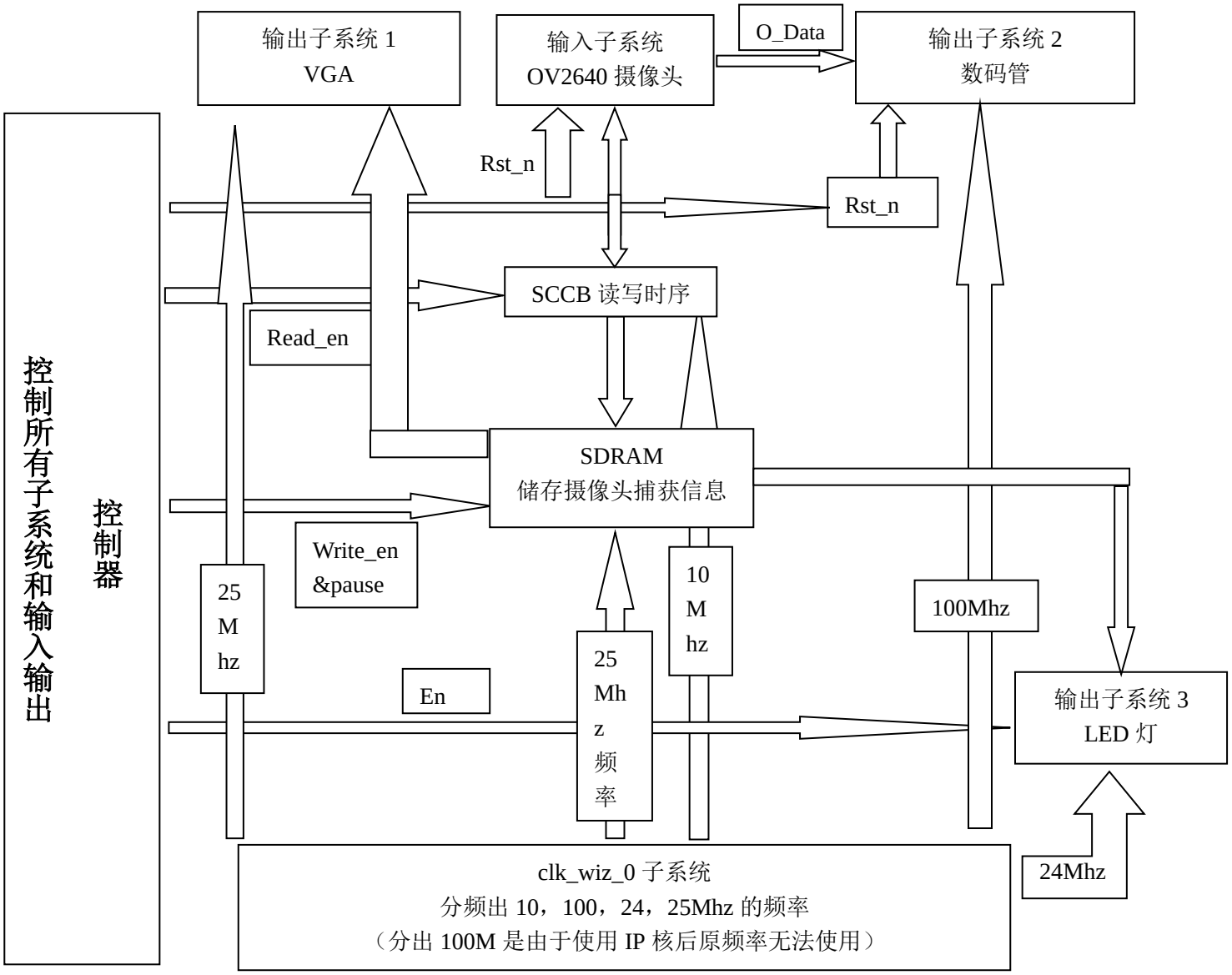
二、摄像头视频显示系统总框图



三、ASM 流程图



四、系统控制器设计



2. 状态转移真值表：

输入		输出
RST_n	Pause	Result
1	1	RGB565_Capture module = 0, Other module = 1
	0	All module = 1
0	x	All module = 0

3. 次态函数表达式：

（本模块只涉及复位开关以及图像暂停按键，只有组合控制电路，没有时序控制电路）

RGB565_Capture module = RST_n | Pause，Other module = RST_n.

4. 控制命令逻辑表达式：

RGB565_Capture module = RST_n | Pause，Other module = RST_n.

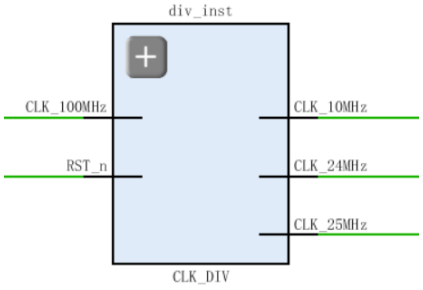
五、子系统模块建模

本系统共划分了 5 个子系统模块，共包括：时钟分频模块、OV2640 驱动模块、暂存数据 RAM 模块、VGA 显示模块、七段数码管显示模块。下面将分别具体介绍之。

1. 时钟分频模块（CLK_DIV）

（1）描述：本人利用 VIVADO 自带的 IP 核：Clocking Wizard，实例化生成了一个将开发板 100MHz 的系统时钟分频成多种频率时钟的 PLL 时钟分频器。其中，10MHz 的时钟分配给摄像头驱动模块，用于摄像头的 SCCB 读写协议；24MHz 的时钟分配给摄像头驱动模块，向摄像头 XCLK 接口输入该时钟；25.175MHz 的时钟分配给 VGA 显示模块，用于驱动 VGA 显示器。

（2）功能框图：



（3）接口信号定义：

接口名称	接口属性	接口描述
CLK_100MHz	input	开发板系统时钟，100MHz
RST_n	input	复位信号，低电平有效
CLK_10MHz	output	SCCB 协议时钟，10MHz
CLK_24MHz	output	摄像头模块 XCLK 时钟，24MHz
CLK_25MHz	output	VGA 驱动显示时钟，25.175MHz

（4）模块 verilog 代码：（该模块为实例化 IP 核的模块）

```
CLK_DIV
div_inst(.CLK_100MHz(CLK_100MHz), .CLK_10MHz(w_clk_10MHz), .CLK_24MHz(w_clk_24MHz), .CLK_25MHz(w_clk_25MHz), .RST_n(RST_n));
```

2. OV2640 驱动模块（OV2640_Driver. v）

（1）描述：本人借助网上资料以及对 SCCB 以及数据采集的理解，编写此模块，该模块是整个摄像头驱动，乃至整个系统的核心关键模块，重点以及难点，一是利用 SCCB 协议读写 OV2640 摄像头的内部寄存器，以进行初始化配置，二是在配置摄像头成功后，对其返回的数据作采集、处理、缓存与输出。

该子系统包括 3 个子模块，第一个是 SCCB 时序控制模块（SCCB_Timing_Control. v），第二个是 SCCB 配置信息模块（RGB565_Config. v），第三个是摄像头输出信号捕获处理模块（RGB565_Capture. v）。

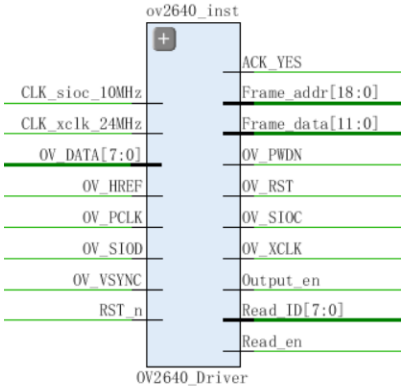
①SCCB 时序控制模块是一大难点，核心是根据生成的 SIOC 时钟，利用 SCCB 协议配

置状态机，对摄像头模块进行配置信息的写入与读取其 ID 信息，通讯过程，数据发送方将并行数据转为串行数据，借助时钟向摄像头模块输入，每输入完一相的数据就会等待摄像头的应答，根据应答情况决定是否继续对下一条数据进行写入。本人对状态机进行了一个巧妙的处理，由于只有一串信息写入成功后，才会进行下一条信息的写入，故本人将读取摄像头 ID 的信息放至最后一条，只有当所有的配置信息写入成功后，才会有摄像头 ID 信息的显示，不仅可以验证信息写入完毕，还可以得到摄像头 ID 序号，判断摄像头是否工作正常。而对于返回来的串行 ID 数据，在转换成并行数据后显示在七段数码管上。另外，可以读取的 ID 序号有四种，都可以进行读取。

②SCCB 配置信息模块主要是顺序存储着需要配置的信息，相当于一个 ROM，可以将根据配置信息的序号看做为地址，通过该序号即可取得对应的配置信息。

③摄像头输出信号捕获处理模块是另一大难点，由于配置摄像头输出的数据格式为 RGB565 格式，并且由于接口限制，每个像素周期只输出半个像素，因此需要将两个像素时钟周期的两个像素拼接成一个完整的像素，而由于开发板上的 RGB 每个颜色都是 4 位，并且受限于开发板的空间大小，需要对拼接好的像素进行截断，从 16 位至 12 位，再缓存至 RAM 里，同时，缓存的数据要与 VGA 显示的数据要都能一一对应，不然显示效果就会受到严重的影响（比如雪花、万花筒等“奇特”的效果），利用双端 RAM 缓存，可以利用地址线对此进行控制。

（2）功能框图：



（3）接口信号定义：

接口名称	接口属性	接口描述
OV_SIOC	output	摄像头 SIOC 接口，SCCB 时钟
OV_SIOD	inout	摄像头 SIOD 接口，SCCB 数据传输通道
OV_VSYNC	input	摄像头场同步信号接口
OV_HREF	input	摄像头行同步信号接口
OV_PCLK	input	摄像头输出视频数据时钟
OV_XCLK	output	提供摄像头工作的外部时钟
OV_DATA	input	摄像头输出的 8 位像素数据口
OV_RST	output	摄像头复位接口
OV_PWDN	output	摄像头掉电模式使能接口
Frame_data	output	输出捕获并处理后的单个像素数据
Frame_addr	output	输出对应像素数据需要存放的 RAM 地址
Output_en	output	标志摄像头开始输出数据的信号

RST_n	input	复位信号，低电平有效
CLK_sioc_10MHz	input	SCCB 协议时钟，10MHz
CLK_xclk_24MHz	input	摄像头模块 XCLK 时钟，24MHz
ACK_YES	output	VGA 驱动显示时钟，25.175MHz
Read_en	output	用于 tb 中控制三态门读写的信号线
Read_ID	output	摄像头 ID 序列

（4）模块 verilog 代码：

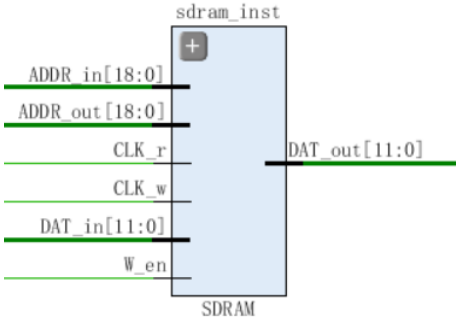
```
module OV2640_Driver(OV_SIOC, OV_SIOD, OV_VSYNC, OV_HREF, OV_PCLK, OV_XCLK, OV_DATA, OV_RST, OV_PWDN, Frame_data, Frame_addr, Output_en, RST_n, CLK_sioc_10MHz, CLK_xclk_24MHz, ACK_YES, Read_en, Read_ID);//clk_pll, , Capture_en
    //摄像头接口
    output OV_SIOC;//SCCB 时钟
    inout OV_SIOD;//SCCB 数据，双向
    input OV_VSYNC;//场同步信号
    input OV_HREF;//行有效信号输出
    input OV_PCLK;//像素时钟
    output OV_XCLK;//输入主时钟
    input [7:0]OV_DATA;//8bit 数据
    output OV_RST;//复位
    output OV_PWDN;//省电
    //与 FIFO 接口
    output [11:0]Frame_data;
    output [18:0]Frame_addr;
    output Output_en;//配置完成信号
    //本模块内部全局接口
    input RST_n;//复位
    input CLK_sioc_10MHz;//SCCB 协议使用时钟，10MHz。
    input CLK_xclk_24MHz;//摄像头 XCLK 使用时钟，24MHz
    output ACK_YES;//摄像头车应答信号（低电平为有应答）
    output Read_en;//TBd 三态门读取信号控制
    output [7:0]Read_ID;//读取寄存器的数据

    //宏定义参数
    parameter p_rIDaddr = 8'h60;//写寄存器的 ID 地址
    //模块内部连线
    wire [7:0]w_cfg_size;
    wire [7:0]w_cfg_index;
    wire [15:0]write_data;//读取配置数据线
    wire w_xclk, w_sioc;//时钟连线
    wire w_cfg_done;//配置完成信号

    ////固定赋值
    assign OV_RST   = 1;//正常使用，拉高电平
    assign OV_PWDN = 0;//正常使用，拉低电平

    //SCCB 时序读写控制
    SCCB_Timing_Control timing_inst(
        .CLK(CLK_sioc_10MHz),
        .RST_n(RST_n),
        .SCCB_CLK(OV_SIOC),
        .SCCB_DATA(OV_SIOD),
        .CFG_size(w_cfg_size), //w_cfg_size
        .CFG_index(w_cfg_index), //w_cfg_index
        .CFG_data({p_rIDaddr, write_data[15:0]}), //r_read_ID
        .CFG_done(w_cfg_done), //
        .CFG_rdata(Read_ID),
        .ACK(ACK_YES),
        .Read_en(Read_en)
    );

    //配置信息模块
    RGB565_Config config_inst(.LUT_INDEX(w_cfg_index), .LUT_DATA(write_data), .LUT_SIZE(w_cfg_size));

    //捕获输出数据模块
    RGB565_Capture capture_inst(
        .CLK(CLK_xclk_24MHz),
        .RST_n(RST_n),
        .CFG_done(w_cfg_done),
        .OV_pclk(OV_PCLK),
        .OV_xclk(OV_XCLK),
        .OV_vsync(OV_VSYNC),
        .OV_href(OV_HREF),
        .OV_din(OV_DATA),
        .Out_frame_data(Frame_data),
        .Out_data_en(Output_en),
        .Out_data_addr(Frame_addr)
    );
endmodule
```

3. 暂存数据 RAM 模块（SDRAM. v）

（1）描述：该模块在上述有提到，就是用来模拟 SDRAM 的功能，对处理完成的显示

数据像素进行缓存。该模块为二次封装了实例化的 VIVADO 自带的 IP 核：Block Memory Generator。一端为写专用接口，根据写时钟、写使能、写地址，来进行数据的写入，另一端为读专用接口，根据读时钟、读地址来进行数据的读取。读写时钟异步，相互干扰较少。

（2）功能框图：



（3）接口信号定义：

接口名称	接口属性	接口描述
W_en	input	写入数据使能信号
CLK_w	input	写操作时钟
CLK_r	input	读操作时钟
ADDR_in	input	写入数据的地址
ADDR_out	input	读取数据的地址
DAT_in	input	写入的数据
DAT_out	output	读取的数据

（4）模块 verilog 代码：

```
module SDRAM(W_en, CLK_w, CLK_r, ADDR_in, ADDR_out, DAT_in, DAT_out);//, Write_en
    input W_en;//写使能，高电平有效
    input CLK_w;//写入数据时钟
    input CLK_r;//读取数据时钟
    input [18:0]ADDR_in;//地址写入
    input [18:0]ADDR_out;//地址写出
    input [11:0]DAT_in;//12 位像素写入
    output [11:0]DAT_out;//12 位像素读出

    blk_mem_gen_0 sdram_inst (
        .clka(CLK_w),       // 写时钟
        .wea(W_en),         // 使能
        .addra(ADDR_in),    // 写地址   input wire [18 : 0] addra
        .dina(DAT_in),      // 写数据   input wire [11 : 0] dina
        .clkb(CLK_r),       // 读时钟   input wire clkb
        .addrb(ADDR_out),   // 读地址   input wire [18 : 0] addrb
        .doutb(DAT_out)     // 读数据   output wire [11 : 0] doutb
    );

endmodule
```

4. VGA 显示模块（VGA. v）

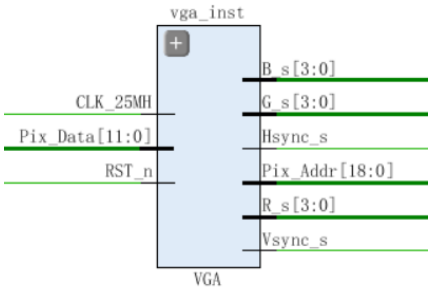
（1）描述：该模块为根据输入的像素数据、行场驱动与时钟驱动，来进行对数据的 VGA 显示。由于本人之前曾经写过 VGA 显示 ROM 里的静态图片，而本系统中需要 VGA 从 RAM 里读取像素数据，差别并不大，因此只需对之前的模块做稍许改动，便可以在 VGA 显示屏显示出图像乃至视频。根据资料的参考，该子系统包含两个子模块，一个是行列同步控制模块（SYNC. v），另一个是 VGA 数据显示控制模块（CONTROL. v）。

①行列同步控制模块，主要负责对显示坐标的控制、显示使能信号以及行列同步信号的处理。其中，坐标对应着显示屏上的像素坐标，以显示屏左上角为坐标原点，水平向右为 x

正半轴方向，垂直向下为 y 正半轴方向，显示使能信号有效时，也就是在显示屏可以显示的区间范围是，才能够将像素数据显示。行列同步信号的功能与之相似。

②VGA 数据显示控制模块，则关系到像素数据的具体显示位置以及颜色的解码，像素数据已经统一为 12 位，适应开发板的要求，并且也规定 12 位像素中，高 4 位为 R，中 4 位为 G，低 4 位为 B。

（2）功能框图：



（3）接口信号定义：

接口名称	接口属性	接口描述
CLK_25MHz	input	输入的驱动时钟，25MH
RST_n	input	复位信号，低电平有效
Pix_Data	input	需要显示的像素数据
Pix_Addr	output	像素数据对应的 RAM 地址
Vsync_s	output	行同步信号
Hsync_s	output	列同步信号
R_s	output	RGB 红颜色信号
G_s	output	RGB 绿颜色信号
B_s	output	RGB 蓝颜色信号

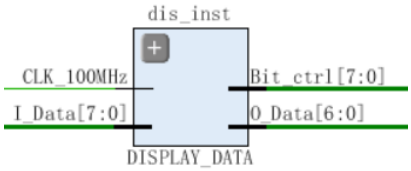
（4）模块 verilog 代码：

```
module VGA(CLK_25MH, RST_n, Pix_Data, Pix_Addr, Vsync_s, Hsync_s, R_s, G_s, B_s);
    input CLK_25MH;//时钟信号。25MHz，上升沿有效
    input RST_n;//复位信号，低电平有效
    input [11:0]Pix_Data;//图片的像素数据，12 位
    output [18:0]Pix_Addr;//像素地址
    output Vsync_s;//行同步信号
    output Hsync_s;//列同步信号
    output [3:0]R_s;//红颜色信号
    output [3:0]G_s;//绿颜色信号
    output [3:0]B_s;//蓝颜色信号

    //各个模块之间的连线
    wire ready_s;//有效区域显示线
    wire [10:0]row;//纵坐标
    wire [10:0]col;//横坐标

    //实例化同步控制模块
    SYNC sync_inst(
        .CLK(CLK_25MH),
        .RST_n(RST_n),
        .Vsync_s(Vsync_s),
        .Hsync_s(Hsync_s),
        .Ready_s(ready_s),
        .Col_s(col),
        .Row_s(row)
    );
    //实例化 VGA 控制模块
    CONTROL control_inst(
        .CLK(CLK_25MH),
        .RST_n(RST_n),
        .Ready_s(ready_s),
        .Col_s(col),
        .Row_s(row),
        .Rom_addr(Pix_Addr),
        .Rom_data(Pix_Data),
        .R_s(R_s),
        .G_s(G_s),
        .B_s(B_s)
    );

endmodule
```

5. 七段数码管显示模块（DISPLAY_DATA. v）

（1）描述：该模块是由之前理论课做过的七段数码管显示模块添加位控功能改装的，其自带分频，可以直接接入开发板的 100MHz 系统时钟，为了适应系统要求，即显示出摄像头返回的 ID 序号（8 位数据），因此该模块显示数据标准为 16 进制，并能显示 8 位的数据，需要用到 2 个七段数码管，其余未使用到的数码管均不亮。该模块在顶层模块实现位控，而具体某位的控制则使用到以前的显示模块，并且添加了 16 进制数的 a、b、c、d、e、f 显示情况。

（2）功能框图：



（3）接口信号定义：

（4）模块 verilog 代码：

```
module      DISPLAY_DATA(CLK_100MHz, I_Data, O_Data, Bit_ctrl);
    input CLK_100MHz;//时钟，上升沿有效
    input  [7:0]I_Data;//需要显示的数据//默认八位
    output [6:0]O_Data;
    output reg[7:0]Bit_ctrl;//位控

    wire w_clk;//分频后的时钟
    reg [3:0]temp_disp;//暂时显示
    reg flag_set = 0;

    //位控
    always @ (posedge w_clk)
    begin
        if (flag_set)
            begin
            flag_set <= 0;
            Bit_ctrl = 8'b1111_1101;//显示[1]位
            temp_disp <= {I_Data[7:4]};
            end
        else
            begin
            flag_set <= 1;
            Bit_ctrl = 8'b1111_1110;//显示[0]位
            temp_disp <= {I_Data[3:0]};
            end
    end

    //显示
    display7 DIS_inst(.iData(temp_disp), .oData(O_Data));
    //分频时钟
    divider DIV_inst(.I_CLK(CLK_100MHz), .rst(0), .O_CLK(w_clk));
endmodule
```

六、测试模块建模

注：以下 tb 模块代码均为在各个子系统模块未连接时的单独测试模块功能的代码，各个模块连接综合后的总系统暂无 tb 模块。

1. OV2640_Driver. v（OV2640 驱动模块）中 SCCB 时序验证的 tb 代码

```
module OV_ID_tb();
    reg CLK;
    reg RST_n;
    wire SIOC;
    wire SIOD;
    reg OV_VSYNC;
    reg OV_HREF;
    reg OV_PCLK;
    wire OV_XCLK;
    reg [7:0]OV_DATA;
```

```
    wire OV_RST;
    wire OV_PWDN;

    wire Frame_vsync;
    wire Frame_href;
    wire [15:0]Frame_data;
    wire Frame_clken;
    wire [7:0]Fps_rate;

    wire read_en;
    wire sccb_out;
    reg sccb_in;
    wire yes;
    wire [6:0]oData;
    wire [7:0]bit_ctrl;
    wire done;

    initial
        CLK = 0;
    always
        #1 CLK = ~CLK;

    initial
        begin
        RST_n = 0;
        #3 RST_n = 1;
        end

    initial
        sccb_in = 0;

    initial
        OV_PCLK = 0;
    always
        #1 OV_PCLK = ~OV_PCLK;

    initial
        OV_DATA = 0;
    always
        #2 OV_DATA = OV_DATA + 1'b1;

    initial
        begin
        OV_VSYNC = 1;
        OV_HREF = 1;
        end

    assign SIOD = (read_en) ? sccb_in : 1'bz;//向模块写入数据
    assign sccb_out = (~read_en) ? SIOD : 1'bz;//从模块读出数据

    Show_ID uut(
        .OV_SIOC(SIOC),
        .OV_SIOD(SIOD),
        .OV_VSYNC(OV_VSYNC),
        .OV_HREF(OV_HREF),
        .OV_PCLK(OV_PCLK),
        .OV_XCLK(OV_XCLK),
        .OV_DATA(OV_DATA),
        .OV_RST(OV_RST),
        .OV_PWDN(OV_PWDN),
        .Frame_vsync(Frame_vsync),
        .Frame_href(Frame_href),
        .Frame_data(Frame_data),
        .Frame_clken(Frame_clken),
        .Fps_rate(Fps_rate),

        .oData(oData),
        .bit_ctrl(bit_ctrl),

        .YES(yes),

        .RST_n(RST_n),
        .CLK(CLK),
        .Read_en(read_en),
        .done(done)
    );
endmodule
```

2. VGA. v（VGA 驱动模块）中输出结果的 tb 代码

```
module VGA_tb();
    reg CLK;//时钟信号，上升沿有效
    reg RST_n;//复位信号，低电平有效
    wire Vsync_s;//行同步信号
    wire Hsync_s;//列同步信号
    wire [3:0]R_s;//红颜色信号
    wire [3:0]G_s;//绿颜色信号
    wire [3:0]B_s;//蓝颜色信号

    initial//时钟
        CLK = 0;
    always
        #1 CLK = ~CLK;

    initial//复位信号
    begin
        RST_n = 0;
        #1
        RST_n = 1;
    end
    //实例化测试模块
    VGA
uut(.CLK(CLK), .RST_n(RST_n), .Vsync_s(Vsync_s), .Hsync_s(Hsync_s), .R_s(R_s), .G_s(G_s), .B_s(B_s));
endmodule
```

3. DISPLAY_DATA. v（七段数码管显示模块）中输出结果的 tb 代码

```
module display7_tb();
    reg CLK;//时钟，上升沿有效
    reg [7:0]iData;
    wire [6:0]oData;
    wire [7:0]bit_ctrl;
```

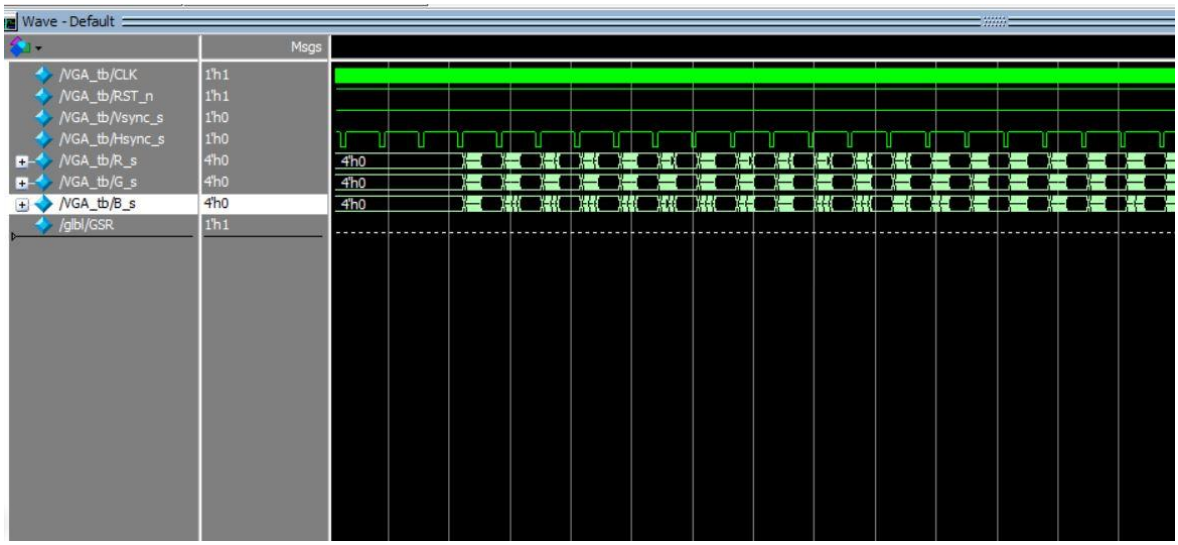
```
initial
        CLK = 0;
always
        #1 CLK = ~CLK;

initial
begin
                //    0
        iData = 8'b00010000;
        #20 //    2
        iData = 8'b00110010;
        #20 //    4
        iData = 8'b01010100;
        #20 //    6
        iData = 8'b01110110;
        #20 //    8
        iData = 8'b10011000;
    end

    //实例化
    display_data uut(
        .CLK(CLK),
        .iData(iData),
        .oData(oData),
        .bit_ctrl(bit_ctrl)
    );
endmodule
```

七、实验结果

注：实验结果包括子模块的独立测试结果，以及整个系统的总体测试结果。

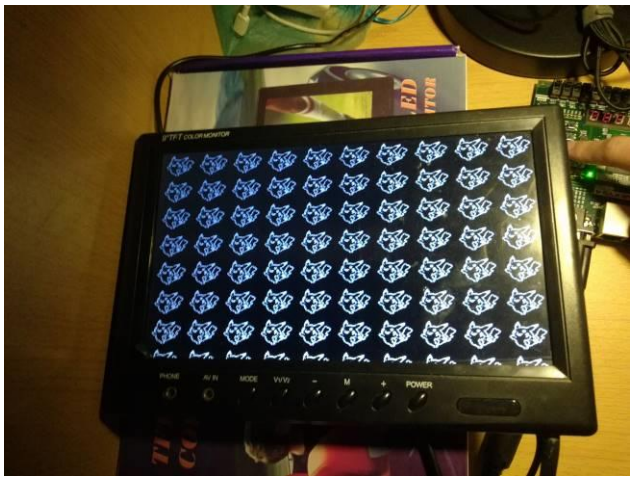
1. VGA 显示模块实验结果

（1）modelsim 仿真波形图

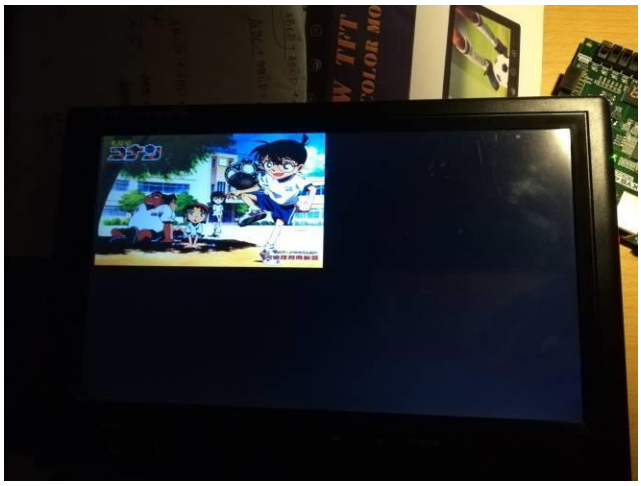


（2）综合下板后显示结果

①黑白简单图片的循环显示

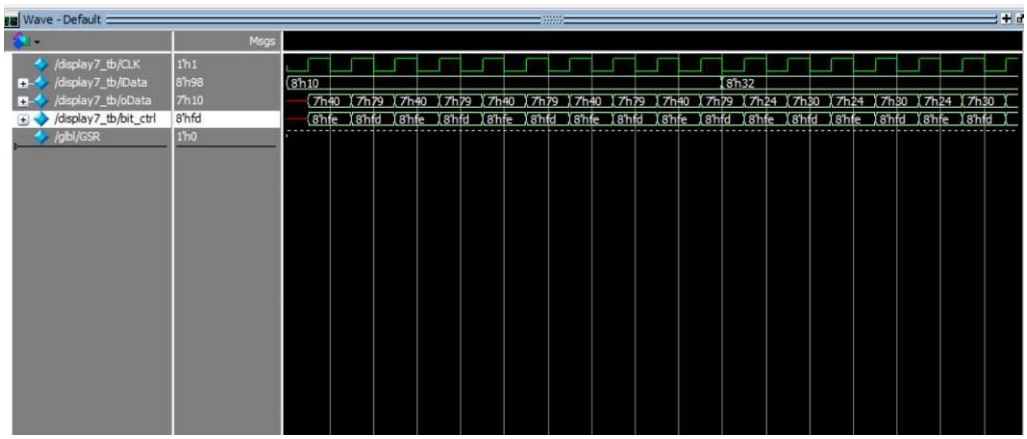


②彩色复杂图片的显示

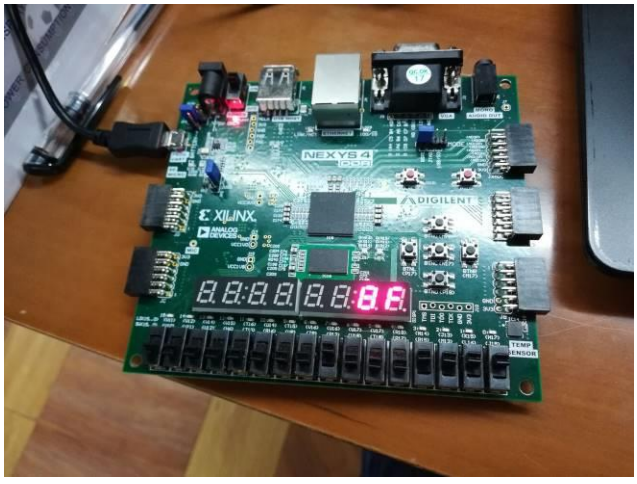


2. 七段数码管显示模块实验结果

（1）modelsim 仿真波形图



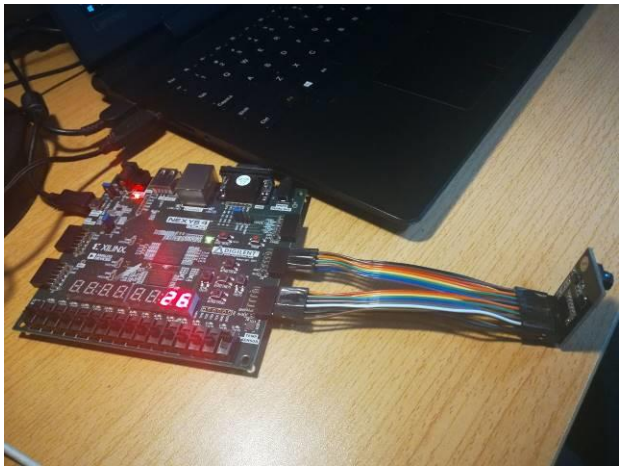
（2）综合下板后显示结果（通过开关产生数据）



3. OV2640 驱动中 SCCB 协议数据传输实验结果

（1）modelsim 仿真波形图

（2）综合下板后显示结果（读摄像头取 ID）

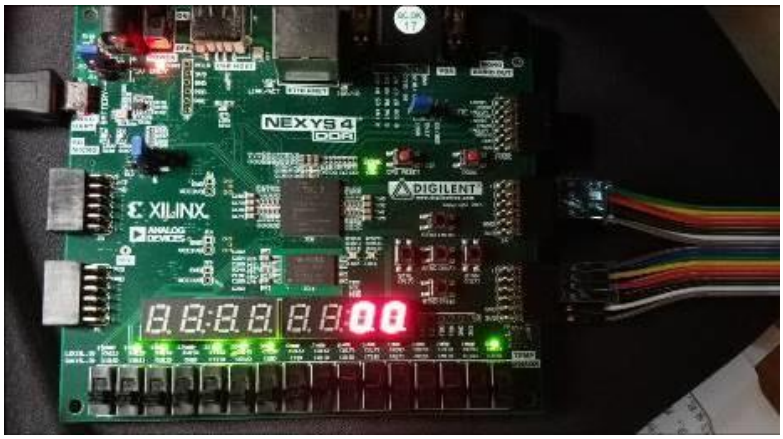


4. OV2640 显示总系统的实验结果

综合下板后显示结果

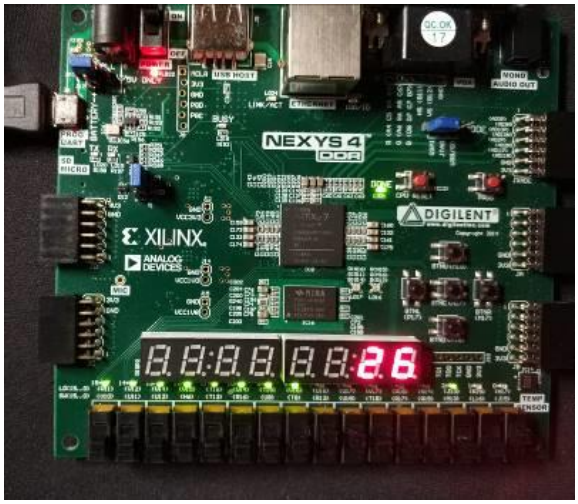
（1） 初始状态：

数码管显示 00，左下角 8 个 LED 灯全亮。



（2） 开始工作：

数码管显示 26，为产品 ID，此时摄像头无遮挡正常采光，左下角 LED 灯会随输入信号亮暗发生变化，当前较亮，因此 LED 灯较亮。



（3） 当遮挡住摄像头时，左下角 LED 灯变暗。

（4） 将摄像头对准水杯，此时 VGA 上显示水杯上的标签，说明采集图像并显示成功，同时可以看到 LED 灯较亮，ID 显示正常。

（5） 暂停屏幕画面：

八、结论

1. 对比于 OV7670、OV7725 等其他系列的摄像头，OV2640 的像素高，功能更多更全面，配置寄存器更多，因而实现起来更加困难，在配置寄存器以及兼容显示器方面就有很多更高的要求，因而难度更大，但并不能说无法通过 FPGA 开发出来，只是需要对其传输协议（SCCB 协议）有深层次的理解，能正确配置出其时序信号，保证 FPGA 开发板能正常与摄像头模块进行通信，，对于其产生的数据流，巧妙运用开发板上的资源对其进行处理与缓存，同样能在较低分辨率上显示出来。

2. 目前，本人对 OV2640 摄像头模块的开发程度还不够，由于摄像头寄存器之间关联

很大，资料介绍也不是很详细，十分容易导致配置不当而无法输出图像信息，因此现在也只是仅仅只能按照固有的配置信息来配置摄像头，无法进行更多的个性化选择，并且，由于没有开发 microSD 卡卡槽，还不能让摄像头拍照并将照片储存至 microSD 卡中，甚至让摄像头也实现录像并储存在 microSD 卡中，这些还需要对摄像头模块寄存器有更深的理解以及掌握 microSD 卡的传输协议。限于时间与精力，本人目前只研究到显示图像这一层面。