浙江大学

本科实验报告

课程名称：	计算机组成
姓　　名：	姜雨童
学　　院：	计算机科学与技术学院
专　　业：	计算机科学与技术
邮　　箱：	3220103450@zju.edu.cn
QQ　　号：	1369218489
电　　话：	19550103468
指导教师：	马德
报告日期：	2024 年 4 月 6 日

浙江大学实验报告

课程名称：计算机组成 实验类型：综合

实验项目名称：Lab2：IP 核集成 SOC 设计

学生姓名：姜雨童 学号：33220103450 同组学生姓名：/

实验地点：紫金港东四 509 室 实验日期：2024 年 3 月 6 日

一、操作方法与实验步骤

实验任务：

通过第三方 IP 和已有 IP 模块建立 CPU 测试环境（SOC 系统的集成实现）------参考逻辑原理图采用 verilog 例化调用子模块的方式实现。

实验步骤：

SOC 系统分解成 11 个子模块：

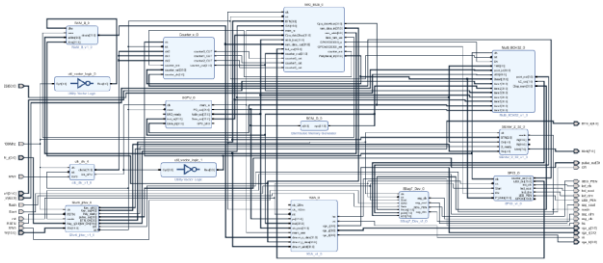
集成实现SOC

- U1： CPU -SCPU
- U2： ROM -ROM_D
- U3： RAM -RAM_B
- U4： 总线(含外设3~4) -MIO_BUS
- U5： 七段显示接口 -Multi_8CH32
- U6： 外设1- 七段显示设备 -Seg7_Dev
- U7： 外设2-GPIO接口及LED -PIO
- U8： 辅助模块一，通用分频模块 -Clk_div
- U9： 辅助模块二，机械去抖模块 -Anti_jitter
- U10：通用计数器 -Counter_x
- U11: VGA显示接口 -VGA

实验二搭建平台时，子模块基本都已经给出，只需要将.v 和.edf 文件导入项目工程中即可。其中包含两个**特例**：U2—ROM 需要在工程中重新生成（具体操作方法已在 lab 0 中熟悉），U6—Seg7_Dev_0 需要以 IP 核的形式导入。

随后根据顶层逻辑图写出 Verilog 代码，完成各模块的调用。

SOC顶层逻辑图



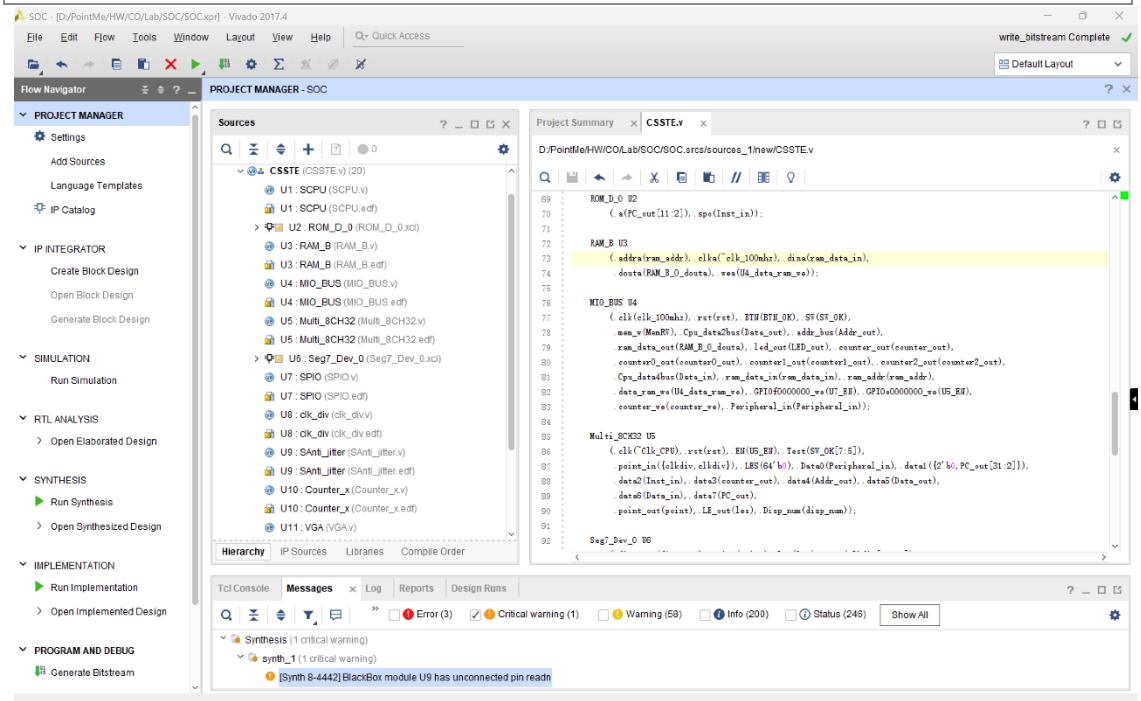
本图仅供概览，详细连接图请参见完整版pdf文档
（参考逻辑图完成模块调用）

Verilog 代码如下：

```
module CSSTE(
    input clk_100mhz,
    input RSTN,
    input [3:0] BTN_y,
    input [15:0] SW,
    output [3:0] Blue,
    output [3:0] Green,
    output [3:0] Red,
    output HSYNC,
    output VSYNC,
    output [15:0] LED_out,
    output [7:0] AN,
    output [7:0] segment
    );
    wire [31:0] clkdiv;
    wire [31:0] PC_out;
    wire [15:0] SW_OK;
    wire [31:0] Addr_out;
    wire [31:0] Data_in;
    wire [31:0] Data_out;
    wire MemRW;
    wire Clk_CPU;
    wire [31:0] Inst_in;
    wire rst;
    wire [9:0] ram_addr;
    wire [31:0] ram_data_in;
    wire [31:0] RAM_B_0_douta;
    wire U4_data_ram_we;
    wire [3:0] BTN_OK;
    wire [31:0] counter_out;
    wire counter0_out;
    wire counter1_out;
    wire counter2_out;
```

```
    wire U7_EN;
    wire U5_EN;
    wire counter_we;
    wire [31:0] Peripheral_in;
    wire [63:0] point;
    wire [7:0] les;
    wire [7:0] disp_num;
    wire [2:0] counter_set;

    SCPU U1
        (.Addr_out(Addr_out),.Data_in(Data_in),.Data_out(Data_out),
         .MIO_ready(1'b0),.MemRW(MemRW),.PC_out(PC_out),
         .clk(Clk_CPU),.inst_in(Inst_in),.rst(rst));

    ROM_D_0 U2
        (.a(PC_out[11:2]),.spo(Inst_in));

    RAM_B U3
        (.addra(ram_addr),.clka(~clk_100mhz),.dina(ram_data_in),
         .douta(RAM_B_0_douta),.wea(U4_data_ram_we));

    MIO_BUS U4
        (.clk(clk_100mhz),.rst(rst),.BTN(BTN_OK),.SW(SW_OK),
         .mem_w(MemRW),.Cpu_data2bus(Data_out),.addr_bus(Addr_out),
         .ram_data_out(RAM_B_0_douta),.led_out(LED_out),.counter_out(counter_out),
         .counter0_out(counter0_out),.counter1_out(counter1_out),.counter2_out(counter2_o
ut),
         .Cpu_data4bus(Data_in),.ram_data_in(ram_data_in),.ram_addr(ram_addr),
         .data_ram_we(U4_data_ram_we),.GPIOf0000000_we(U7_EN),.GPIOe0000000_we(U5_EN),
         .counter_we(counter_we),.Peripheral_in(Peripheral_in));

    Multi_8CH32 U5
        (.clk(~Clk_CPU),.rst(rst),.EN(U5_EN),.Test(SW_OK[7:5]),
         .point_in({clkdiv,clkdiv}),.LES(64'b0),.Data0(Peripheral_in),.data1({2'b0,PC_out
[31:2]}),
         .data2(Inst_in),.data3(counter_out),.data4(Addr_out),.data5(Data_out),
         .data6(Data_in),.data7(PC_out),
         .point_out(point),.LE_out(les),.Disp_num(disp_num));

    Seg7_Dev_0 U6
        (.disp_num(disp_num),.point(point),.les(les),.scan(clkdiv[18:16]),
         .AN(AN),.segment(segment));

    SPIO U7
```

```
        (.clk(~Clk_CPU),.rst(rst),.Start(clkdiv[20]),.EN(U7_EN),.P_Data(Peripheral_in),
         .counter_set(counter_set),.LED_out(LED_out));

    clk_div U8
        (.clk(clk_100mhz),.rst(rst),.SW2(SW_OK[2]),.SW8(SW_OK[8]),.STEP(SW_OK[10]),
         .clkdiv(clkdiv),.Clk_CPU(Clk_CPU));

    SAnti_jitter U9
        (.clk(clk_100mhz),.RSTN(RSTN),.Key_y(BTN_y),.SW(SW),
         .rst(rst),.BTN_OK(BTN_OK),.SW_OK(SW_OK));

    Counter_x U10
        (.clk(~Clk_CPU),.rst(rst),.clk0(clkdiv[6]),.clk1(clkdiv[9]),.clk2(clkdiv[11]),
         .counter_we(counter_we),.counter_val(Peripheral_in),.counter_ch(counter_set),
         .counter0_OUT(counter0_out),.counter1_OUT(counter1_out),.counter2_OUT(counter2_o
ut),
         .counter_out(counter_out));

    VGA U11
        (.clk_25m(clkdiv[1]),.clk_100m(clk_100mhz),.rst(rst),
         .pc(PC_out),.inst(Inst_in),.alu_res(Addr_out),.mem_wen(MemRW),
         .dmem_o_data(RAM_B_0_douta),.dmem_i_data(ram_data_in),.dmem_addr(Addr_out),
         .hs(HSYNC),.vs(VSYNC),.vga_r(Red),.vga_g(Green),.vga_b(Blue));

endmodule
```

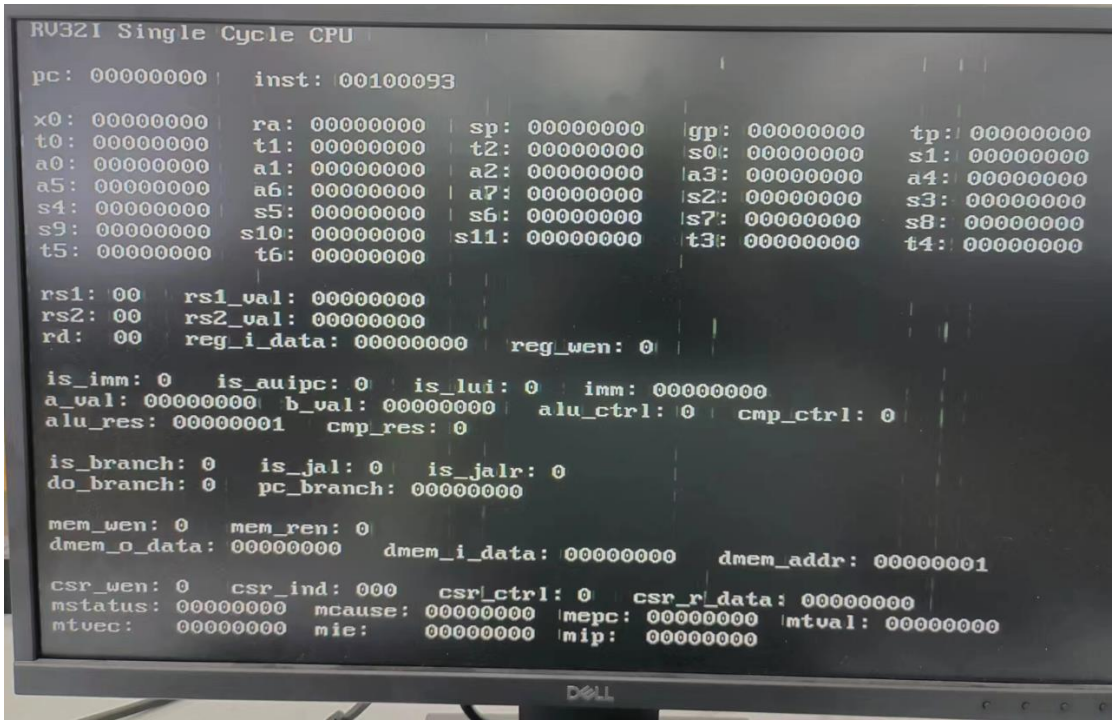


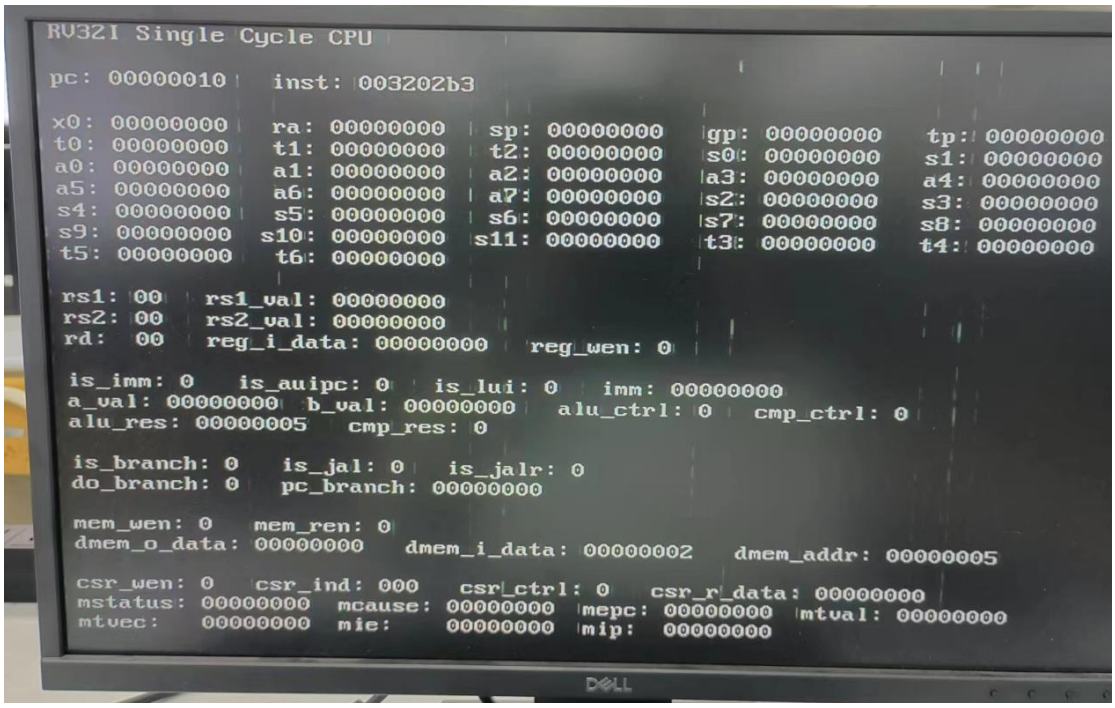
（此处报错在报告第三部分—讨论与心得模块说明。）

最后加入引脚约束文件（已给出），生成 bit 文件后上板验证。

二、实验结果与分析

1、将 vga 线连上显示屏后，屏幕出现如下信息。执行几条指令后，显示信息如下图二所示。





下面给出 sword 板上对应数据显示（上下关系均对应）：

SW[7:5] = 001 时，七段数码管显示 CPU 指令字地址 PC_out：


SW[7:5] = 010 时，七段数码管显示 ROM 指令输出 Inst_in：


SW[7:5] = 011 时，七段数码管显示 counter 值：

SW[7:5] = 100 时，七段数码管显示 CPU 数据存储地址 addr_bus (ALU)：

SW[7:5] = 101 时，七段数码管显示 CPU 数据输入 Cpu_data2bus (寄存器 B)：

SW[7:5] = 110 时，七段数码管显示 CPU 数据输出 Cpu_data4bus (RAM 输出)：

SW[7:5] = 111 时，七段数码管显示 CPU 指令字节地址 PC_out：

2、控制 SW[8]、SW[2]可以选择 CPU 自动/手动单步时钟，由于时钟动效不好以图片形式表示，这里只截取连续几帧作为表示。

开关	位置	功能
SW[8]SW[2]	00	CPU全速时钟 100MHZ
SW[8]SW[2]	01	CPU自动单步时钟(2*24分频)
SW[8]SW[2]	1X	CPU手动单步时钟
SW[10]	0~1	CPU手动单步时钟(开关SW[10]从0到1)
SW[7:5]	001	CPU指令字地址PC_out[31:2]
SW[7:5]	010	ROM指令输出Inst_in
SW[7:5]	100	CPU数据存储地址addr_bus(ALU输出)
SW[7:5]	101	CPU数据输出Cpu_data2bus(寄存器B)
SW[7:5]	110	CPU数据输入Cpu_data4bus(RAM输出)
SW[7:5]	111	CPU指令字节地址PC_out

三、讨论、心得

感受：

个人 SOC 平台的搭建最麻烦的地方在于对着逻辑图写顶层 verilog。因为图很大，连线很多，想要看清楚对应的线条和接口就必须放大图片，然后拖动图片改变其可见可见范围。在这个过程中很容易不小心把线看错了，导致接口没接对。（在这个过程中，vivado 有一个不错的地方就是，使用到的接口名字如果没有在上面定义过/被重复定义了，都会以荧光高亮的形式显示出来，减少了很多不必要的校对。）

遇到的报错：

1、 最开始写 verilog 的时候把 output 接口落了，所以上板没有任何反应；

2、 修改了顶层.v 文件，重新生成 bit 文件后，上板发现还是没有反应。检查引脚约束文件的时候发现，虽然第一次上板时使用的是助教给出的完整的引脚约束文件，但是可能因为第一次没有写 output 接口，引脚文件自动把多余的引脚删除了。因此，此时的引脚约束文件内并不含输出接口的引脚约束，需要修改。

3、 最开始导入子模块 IP 的时候，顺便把同路径（非 D 盘）下.mem 文件导入了，但是没有修改 vga 模块的对应接口并重新封装。后续导入 D 盘路径下的.mem 文件后，不再报错。

四、个人生活照片