组成原理实验课程第_5_次实验报告

实验名称	存储器实现			班级	李涛老师
学生姓名	孙蕗	学号	2112060	指导老师	董前琨
实验地点	A308		实验时间	2023.5.23	

1、 实验目的

（1） 了解只读存储器 ROM 和随机存取存储器 RAM 的原理。

（2） 理解 ROM 读取数据及 RAM 读取、写入数据的过程。

（3） 理解计算机中存储器地址编址和数据索引方法。

（4） 理解同步 RAM 和异步 RAM 的区别。

（5） 掌握调用 xilinx 库 IP 实例化 RAM 的设计方法。

（6） 熟悉并运用 verilog 语言进行电路设计。

（7） 为后续设计 cpu 的实验打下基础。

2、 实验任务说明

（1） 学习存储器的设计及原理，如：ROM 读地址索引读取数据过程及时序，RAM 读写时序，同步和异步的区别等。

（2） 学习计算机中内存地址编址和数据索引方法。

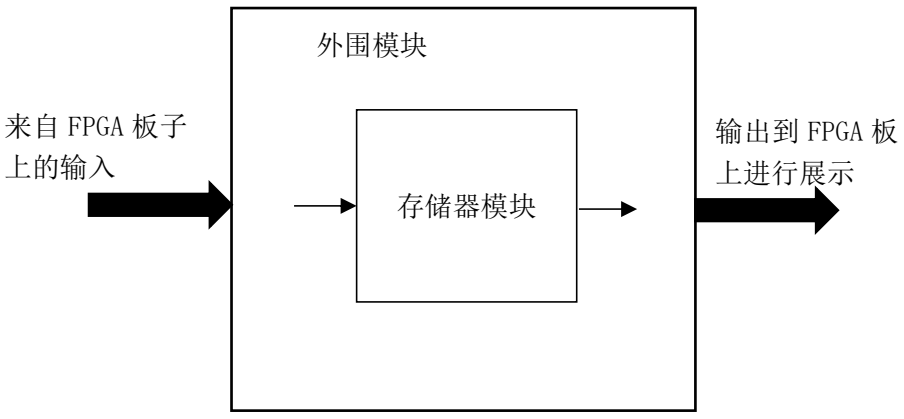
（3） 自行设计本次实验的方案，画出结构框图，详细标出输入输出端口，确定存储器宽度、深度和写使能位数。

（4） 学习 ISE 工具中调用库 IP 的方法。

（5） 本次实验要求调用 xilinx 库 IP 实例化一块 RAM。实例化的 RAM 选择为同步 RAM。本次实验的 RAM 建议设置为两个端口，一个端口用来正常的读写，另一个端口作为调试端口只使用读功能用于观察存储器内部数据。

（6） 调用 xilinx 库 IP 实例化一块 RAM，并进行仿真，得到正确的波形图。

（7） 将以上设计作为一个单独的模块，设计一个外围模块去调用该模块，见图 6.1。外围模块中需调用封装好的 LCD 触摸屏模块，显示 RAM 的正常端口的地址、待写入的数据和读出的数据，显示调试端口的地址和读出的数据。并且需要利用触摸功能输入正常端口的地址和写数据，以及调试端口的地址。



（8） 将编写的代码进行综合布局布线，并下载到实验箱中的 FPGA 板子上进行演示。

（9） 注意：存储器深度不要过大，避免耗费过多的 FPGA 上的资源。本次实验要求实现同步的存储器。而异步存储器的搭建方法同寄存器堆的搭建，但不同的是，寄存器堆中读写端口是分开的，但对于异步 RAM 要求读写共用一个端口，只是会增加一个写使能信号。

可以自行尝试搭建异步的 ROM 和 RAM，在单周期 CPU 实验中会用到异步的 ROM 作为指令存储器，而异步 RAM 作为数据存储器。

3、 实验要求

（1） 做好预习：

1） 掌握存储器的工作原理，明白 ROM 和 RAM，同步和异步的区别；

2） 学习并掌握调用 xilinx 库 IP 进行设计的方法；

3） 确定存储器的输入输出端口及宽度、深度和写使能设计；

4） 如果对 FPGA 板了解的话，可确定设计中与 FPGA 板上交互的接口，画出包含外围模块的整体设计框图，即补充完善图 6.1。

（2） 实验实施：

1） 确认存储器的设计方案的正确性；

2） 编写 verilog 代码；

3） 对该模块进行仿真，得出正确的波形，截图作为实验报告结果一项的材料；

4） 完成调用存储器模块的外围模块的设计，并编写代码；

5） 对代码进行综合布局布线下载到实验箱里 FPGA 板上，进行上板验证。

3. 实验检查：

1） 完成上板验证后，让指导老师或助教进行检查，进行现场演示，按照检查人员的要求，对特定存储器单元读/写，可对演示结果进行拍照作为实验报告结果一项的材料。

4. 实验报告的撰写：

1） 实验结束后，需按照规定的格式完成实验报告的撰写。

4、 本次实验要求

（1） 建立四个工程分别完成同步、异步的 ROM 和 RAM 存储器实验，在实验箱报告中每个工程至少用一组上箱照片和介绍性文字总结验证功能。

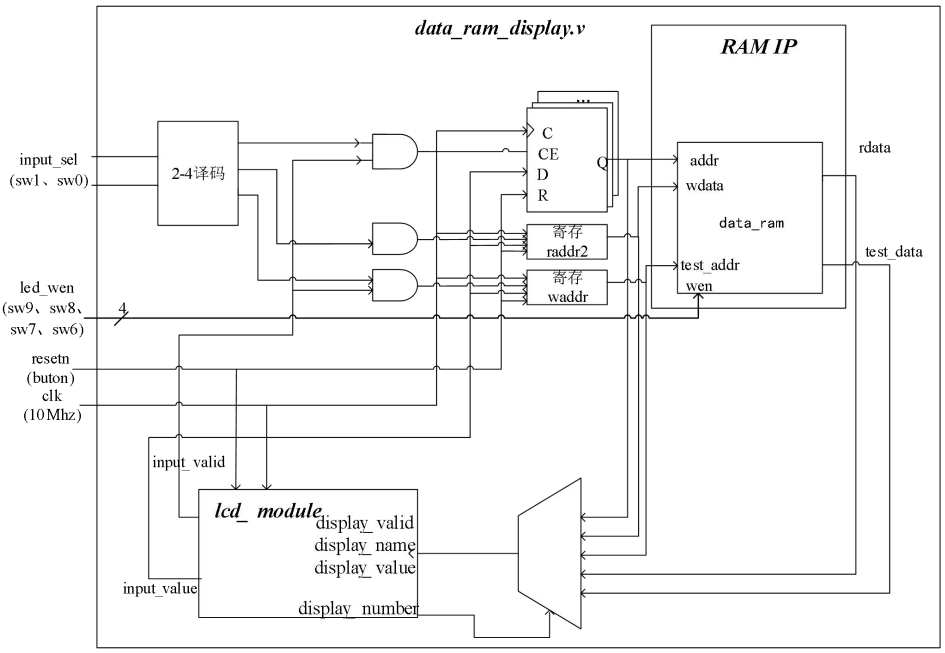
（2） 实验原理图使用实验指导书的图 6.13 和 6.14 即可，无需修改。

（3） 在实验总结中用自己的话总结回答下面几个问题：

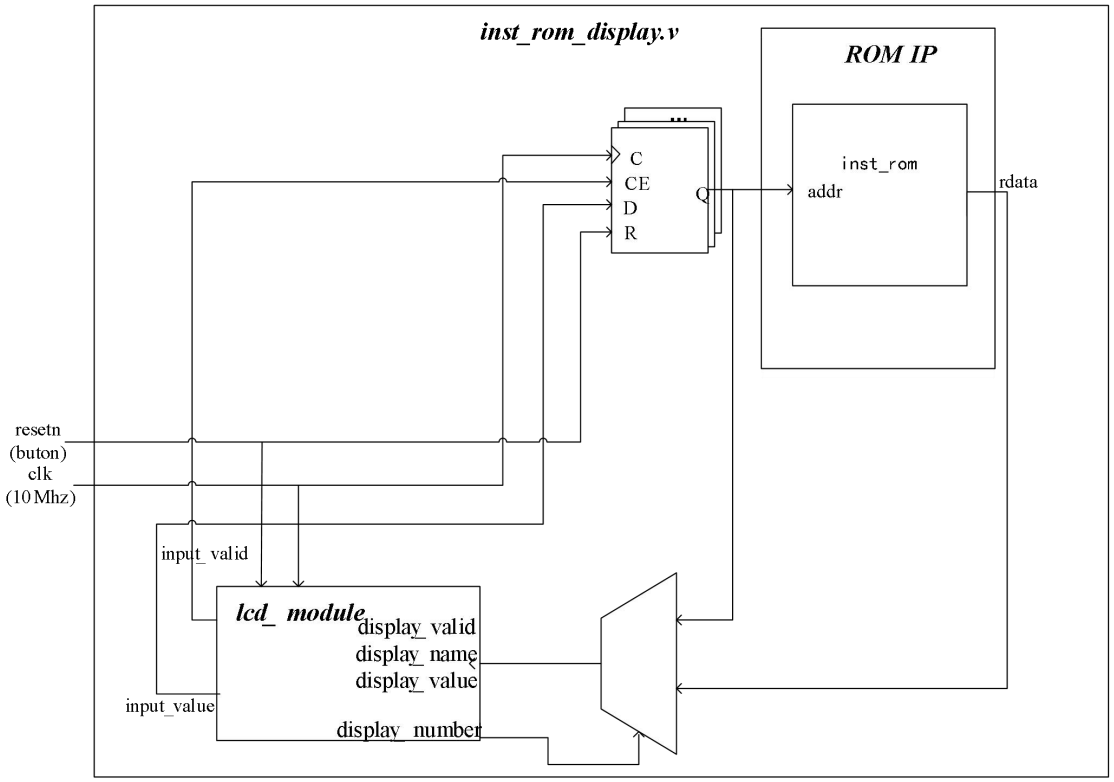
1） 总结一下 ROM 和 RAM 的区别；

2） 分析一下同步存储器和异步存储器的特点，思考说明一下何时需要使用同步存储器，何时需要使用异步存储器。

5、 实验原理图



同步 ram 参考设计的顶层展示原理块框图



inst_rom 参考设计的顶层模块框图

6、 实验步骤

5.1 同步 RAM

5.1.1 生成 IP 核 RAM

新建工程 data_ram，IP Catalog->Memories and Storage Elements->RAMs & ROMs & BRAM->Block Memory Generator。Basic 页面，Component Name 输入"data_ram"，选择 Memory 类型为“True Dual Port RAM”，一个端口作为正常的读写端口，一个端口作为调试端口。勾选“Write Enable”下的“Byte Write Enable”，“Byte Size”选 8bits。Port A options 页面，RAM 宽度设置为 32 位，深度为 256，因为后续 CPU 实验是基于 32 位数据运算的。"Enalbe Port Type" 页面选择"Always Enabled"。在“Port B options”选项卡下进行同样的设置。接点击“OK”，在弹出的窗口中点击“Generate”生成 IP 核即可。

5.1.2 同步 RAM

data_ram_display.v 源代码：

```
1   `timescale 1ns / 1ps
2   //*************************************************************
    **
3   //   > 文件名: data_ram_display.v
4   //   > 描述   : 数据存储器模块显示模块，调用 FPGA 板上的 IO 接口和触摸屏
5   //   > 作者   : LOONGSON
6   //   > 日期   : 2016-04-14
7   //*************************************************************
    **
8   module data_ram_display(
9       //时钟与复位信号
```

```
10      input clk,
11      input resetn,    //后缀"n"代表低电平有效
12
13      //拨码开关，用于产生写使能和选择输入数
14      input [3:0] wen,
15      input [1:0] input_sel,
16
17      //led灯，用于指示写使能信号，和正在输入什么数据
18      output [3:0] led_wen,
19      output led_addr,     //指示输入读写地址
20      output led_wdata,    //指示输入写数据
21      output led_test_addr, //指示输入test地址
22
23      //触摸屏相关接口，不需要更改
24      output lcd_rst,
25      output lcd_cs,
26      output lcd_rs,
27      output lcd_wr,
28      output lcd_rd,
29      inout[15:0] lcd_data_io,
30      output lcd_bl_ctr,
31      inout ct_int,
32      inout ct_sda,
33      output ct_scl,
34      output ct_rstn
35      );
36  //-----{LED显示}begin
37      assign led_wen      = wen;
38      assign led_addr     = (input_sel==2'd0);
39      assign led_wdata    = (input_sel==2'd1);
40      assign led_test_addr = (input_sel==2'd2);
41  //-----{LED显示}end
42  //-----{调用数据储存器模块}begin
43      //数据存储器多增加一个读端口，用于读出特定内存地址显示在触摸屏上
44      reg  [31:0] addr;
45      reg  [31:0] wdata;
46      wire [31:0] rdata;
47      reg  [31:0] test_addr;
48      wire [31:0] test_data;
49
50      data_ram data_ram_module(
51          .clka  (clk         ),
52          .wea   (wen         ),
53          .addra (addr[9:2]    ),
```

```
54          .dina  (wdata        ),
55          .douta (rdata        ),
56          .clkb  (clk          ),
57          .web   (4'd0         ),
58          .addrb (test_addr[9:2]),
59          .doutb (test_data    ),
60          .dinb  (32'd0        )
61      );
62  //-----{调用寄存器堆模块}end
63
64  //---------------------{调用触摸屏模块}begin--------------------//
65  //-----{实例化触摸屏}begin
66  //此小节不需要更改
67      reg        display_valid;
68      reg  [39:0] display_name;
69      reg  [31:0] display_value;
70      wire [5 :0] display_number;
71      wire       input_valid;
72      wire [31:0] input_value;
73
74      lcd_module lcd_module(
75          .clk          (clk         ),   //10Mhz
76          .resetn        (resetn       ),
77
78          //调用触摸屏的接口
79          .display_valid  (display_valid ),
80          .display_name   (display_name  ),
81          .display_value  (display_value ),
82          .display_number (display_number),
83          .input_valid    (input_valid   ),
84          .input_value    (input_value   ),
85
86          //lcd 触摸屏相关接口，不需要更改
87          .lcd_rst       (lcd_rst      ),
88          .lcd_cs        (lcd_cs       ),
89          .lcd_rs        (lcd_rs       ),
90          .lcd_wr        (lcd_wr       ),
91          .lcd_rd        (lcd_rd       ),
92          .lcd_data_io    (lcd_data_io   ),
93          .lcd_bl_ctr     (lcd_bl_ctr    ),
94          .ct_int        (ct_int       ),
95          .ct_sda        (ct_sda       ),
96          .ct_scl        (ct_scl       ),
97          .ct_rstn       (ct_rstn      )
```

```
98      );
99  //-----{实例化触摸屏}end
100
101 //-----{从触摸屏获取输入}begin
102 //根据实际需要输入的数修改此小节,
103 //建议对每一个数的输入，编写单独一个 always 块
104     //当 input_sel 为 2'b00 时，表示输入数为读写地址，即 addr
105     always @(posedge clk)
106     begin
107         if (!resetn)
108         begin
109             addr <= 32'd0;
110         end
111         else if (input_valid &&  input_sel==2'd0)
112         begin
113             addr[31:2] <= input_value[31:2];
114         end
115     end
116
117     //当 input_sel 为 2'b01 时，表示输入数为写数据，即 wdata
118     always @(posedge clk)
119     begin
120         if (!resetn)
121         begin
122             wdata <= 32'd0;
123         end
124         else if (input_valid && input_sel==2'd1)
125         begin
126             wdata <= input_value;
127         end
128     end
129
130     //当 input_sel 为 2'b10 时，表示输入数为 test 地址，即 test_addr
131     always @(posedge clk)
132     begin
133         if (!resetn)
134         begin
135             test_addr  <= 32'd0;
136         end
137         else if (input_valid && input_sel==2'd2)
138         begin
139             test_addr[31:2] <= input_value[31:2];
140         end
141     end
```

```
142   //-----{从触摸屏获取输入}end
143
144   //-----{输出到触摸屏显示}begin
145   //根据需要显示的数修改此小节，
146   //触摸屏上共有 44 块显示区域，可显示 44 组 32 位数据
147   //44 块显示区域从 1 开始编号，编号为 1~44，
148      always @(posedge clk)
149      begin
150         case(display_number)
151            6'd1:
152            begin
153               display_valid <= 1'b1;
154               display_name  <= "ADDR ";
155               display_value <= addr;
156            end
157            6'd2:
158            begin
159               display_valid <= 1'b1;
160               display_name  <= "WDATA";
161               display_value <= wdata;
162            end
163            6'd3:
164            begin
165               display_valid <= 1'b1;
166               display_name  <= "RDATA";
167               display_value <= rdata;
168            end
169            6'd5:
170            begin
171               display_valid <= 1'b1;
172               display_name  <= "T_ADD";
173               display_value <= test_addr;
174            end
175            6'd6:
176            begin
177               display_valid <= 1'b1;
178               display_name  <= "T_DAT";
179               display_value <= test_data;
180            end
181            default :
182            begin
183               display_valid <= 1'b0;
184               display_name  <= 40'd0;
185               display_value <= 32'd0;
```

```
            end
        endcase
    end
//-----{输出到触摸屏显示}end
//----------------------{调用触摸屏模块}end--------------------//
endmodule
```

约束文件 data_ram.ucf 源代码：

```
#时钟信号连接
set_property PACKAGE_PIN AC19 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]

#脉冲开关，用于输入作为复位信号，低电平有效
set_property PACKAGE_PIN Y3 [get_ports resetn]
set_property IOSTANDARD LVCMOS33 [get_ports resetn]

#led 灯连接，用于输出
set_property PACKAGE_PIN H7 [get_ports led_wen[3]]
set_property PACKAGE_PIN D5 [get_ports led_wen[1]]
set_property PACKAGE_PIN A3 [get_ports led_wen[0]]
set_property PACKAGE_PIN A5 [get_ports led_addr]
set_property PACKAGE_PIN A4 [get_ports led_wdata]
set_property PACKAGE_PIN F7 [get_ports led_test_addr]
set_property IOSTANDARD LVCMOS33 [get_ports led_wen[3]]
set_property IOSTANDARD LVCMOS33 [get_ports led_wen[2]]
set_property IOSTANDARD LVCMOS33 [get_ports led_wen[1]]
set_property IOSTANDARD LVCMOS33 [get_ports led_wen[0]]
set_property IOSTANDARD LVCMOS33 [get_ports led_addr]
set_property IOSTANDARD LVCMOS33 [get_ports led_wdata]
set_property IOSTANDARD LVCMOS33 [get_ports led_test_addr]

#拨码开关连接，用于输入
set_property PACKAGE_PIN AC21 [get_ports input_sel[1]]
set_property PACKAGE_PIN AD24 [get_ports input_sel[0]]
set_property PACKAGE_PIN AC22 [get_ports wen[3]]
set_property PACKAGE_PIN AC23 [get_ports wen[2]]
set_property PACKAGE_PIN AB6    [get_ports wen[1]]
set_property PACKAGE_PIN W6     [get_ports wen[0]]
set_property IOSTANDARD LVCMOS33 [get_ports input_sel[1]]
set_property IOSTANDARD LVCMOS33 [get_ports input_sel[0]]
set_property IOSTANDARD LVCMOS33 [get_ports wen[3]]
set_property IOSTANDARD LVCMOS33 [get_ports wen[2]]
set_property IOSTANDARD LVCMOS33 [get_ports wen[1]]
set_property IOSTANDARD LVCMOS33 [get_ports wen[0]]

#触摸屏引脚连接
set_property PACKAGE_PIN J25 [get_ports lcd_rst]
set_property PACKAGE_PIN H18 [get_ports lcd_cs]
```

```
41  set_property PACKAGE_PIN K16 [get_ports lcd_rs]
42  set_property PACKAGE_PIN L8 [get_ports lcd_wr]
43  set_property PACKAGE_PIN K8 [get_ports lcd_rd]
44  set_property PACKAGE_PIN J15 [get_ports lcd_bl_ctr]
45  set_property PACKAGE_PIN H9 [get_ports {lcd_data_io[0]}]
46  set_property PACKAGE_PIN K17 [get_ports {lcd_data_io[1]}]
47  set_property PACKAGE_PIN J20 [get_ports {lcd_data_io[2]}]
48  set_property PACKAGE_PIN M17 [get_ports {lcd_data_io[3]}]
49  set_property PACKAGE_PIN L17 [get_ports {lcd_data_io[4]}]
50  set_property PACKAGE_PIN L18 [get_ports {lcd_data_io[5]}]
51  set_property PACKAGE_PIN L15 [get_ports {lcd_data_io[6]}]
52  set_property PACKAGE_PIN M15 [get_ports {lcd_data_io[7]}]
53  set_property PACKAGE_PIN M16 [get_ports {lcd_data_io[8]}]
54  set_property PACKAGE_PIN L14 [get_ports {lcd_data_io[9]}]
55  set_property PACKAGE_PIN M14 [get_ports {lcd_data_io[10]}]
56  set_property PACKAGE_PIN F22 [get_ports {lcd_data_io[11]}]
57  set_property PACKAGE_PIN G22 [get_ports {lcd_data_io[12]}]
58  set_property PACKAGE_PIN G21 [get_ports {lcd_data_io[13]}]
59  set_property PACKAGE_PIN H24 [get_ports {lcd_data_io[14]}]
60  set_property PACKAGE_PIN J16 [get_ports {lcd_data_io[15]}]
61  set_property PACKAGE_PIN L19 [get_ports ct_int]
62  set_property PACKAGE_PIN J24 [get_ports ct_sda]
63  set_property PACKAGE_PIN H21 [get_ports ct_scl]
64  set_property PACKAGE_PIN G24 [get_ports ct_rstn]
65
66  set_property IOSTANDARD LVCMOS33 [get_ports lcd_rst]
67  set_property IOSTANDARD LVCMOS33 [get_ports lcd_cs]
68  set_property IOSTANDARD LVCMOS33 [get_ports lcd_rs]
69  set_property IOSTANDARD LVCMOS33 [get_ports lcd_wr]
70  set_property IOSTANDARD LVCMOS33 [get_ports lcd_rd]
71  set_property IOSTANDARD LVCMOS33 [get_ports lcd_bl_ctr]
72  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[0]}]
73  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[1]}]
74  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[2]}]
75  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[3]}]
76  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[4]}]
77  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[5]}]
78  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[6]}]
79  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[7]}]
80  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[8]}]
81  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[9]}]
82  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[10]}]
83  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[11]}]
84  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[12]}]
85  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[13]}]
86  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[14]}]
87  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[15]}]
88  set_property IOSTANDARD LVCMOS33 [get_ports ct_int]
```

```
89    set_property IOSTANDARD LVCMOS33 [get_ports ct_sda]
90    set_property IOSTANDARD LVCMOS33 [get_ports ct_scl]
91    set_property IOSTANDARD LVCMOS33 [get_ports ct_rstn]
```

5.2 同步 ROM

5.2.1 生成 IP 核 ROM

新建工程 inst_rom，后续步骤同 5.1 所述，但在图 6.5 那步时，Memory 类型需要选择为“Single Port ROM”。点击“Port A Options”设置宽度和深度，"Enalbe Port Type"选择"Always Enabled"，”Other Options”勾选“Load Init File”,并选中需要装载的初始化文件(.coe 文件)。

5.2.2 同步 rom

inst_rom_display.v 源代码：

```
1   `timescale 1ns / 1ps
2   //****************************************************************
    **
3   //   > 文件名: inst_rom_display.v
4   //   > 描述   : 异步指令存储器显示模块，调用 FPGA 板上的 IO 接口和触摸屏
5   //   > 作者   : LOONGSON
6   //   > 日期   : 2016-04-14
7   //****************************************************************
    **
8   module inst_rom_display(
9       //时钟与复位信号
10      input clk,
11      input resetn,    //后缀"n"代表低电平有效
12
13      //触摸屏相关接口，不需要更改
14      output lcd_rst,
15      output lcd_cs,
16      output lcd_rs,
17      output lcd_wr,
18      output lcd_rd,
19      inout[15:0] lcd_data_io,
20      output lcd_bl_ctr,
21      inout ct_int,
22      inout ct_sda,
23      output ct_scl,
24      output ct_rstn
25      );
26  //-----{调用数据储存器模块}begin
27      //数据存储器多增加一个读端口，用于读出特定内存地址显示在触摸屏上
28      reg  [31:0] addr;
29      wire [31:0] inst;
30
31      inst_rom inst_rom_module(
32          .clka   (clk      ),
```

```
33          .addra  (addr[9:2] ),
34          .douta  (inst     )
35      );
36  //-----{调用寄存器堆模块}end
37
38  //---------------------{调用触摸屏模块}begin--------------------//
39  //-----{实例化触摸屏}begin
40  //此小节不需要更改
41      reg        display_valid;
42      reg  [39:0] display_name;
43      reg  [31:0] display_value;
44      wire [5 :0] display_number;
45      wire       input_valid;
46      wire [31:0] input_value;
47
48      lcd_module lcd_module(
49          .clk          (clk         ),   //10Mhz
50          .resetn        (resetn       ),
51
52          //调用触摸屏的接口
53          .display_valid  (display_valid ),
54          .display_name   (display_name  ),
55          .display_value  (display_value ),
56          .display_number (display_number),
57          .input_valid    (input_valid   ),
58          .input_value    (input_value   ),
59
60          //lcd 触摸屏相关接口，不需要更改
61          .lcd_rst       (lcd_rst      ),
62          .lcd_cs        (lcd_cs       ),
63          .lcd_rs        (lcd_rs       ),
64          .lcd_wr        (lcd_wr       ),
65          .lcd_rd        (lcd_rd       ),
66          .lcd_data_io    (lcd_data_io   ),
67          .lcd_bl_ctr     (lcd_bl_ctr    ),
68          .ct_int        (ct_int       ),
69          .ct_sda        (ct_sda       ),
70          .ct_scl        (ct_scl       ),
71          .ct_rstn       (ct_rstn      )
72      );
73  //-----{实例化触摸屏}end
74
75  //-----{从触摸屏获取输入}begin
76  //根据实际需要输入的数修改此小节，
```

```
77  //建议对每一个数的输入，编写单独一个 always 块
78      always @(posedge clk)
79      begin
80          if (!resetn)
81          begin
82              addr <= 32'd0;
83          end
84          else if (input_valid)
85          begin
86              addr[31:2] <= input_value[31:2];
87          end
88      end
89  //-----{从触摸屏获取输入}end
90
91  //-----{输出到触摸屏显示}begin
92  //根据需要显示的数修改此小节，
93  //触摸屏上共有 44 块显示区域，可显示 44 组 32 位数据
94  //44 块显示区域从 1 开始编号，编号为 1~44，
95      always @(posedge clk)
96      begin
97         case(display_number)
98             6'd1:
99             begin
100                display_valid <= 1'b1;
101                display_name  <= "ADDR ";
102                display_value <= addr;
103            end
104            6'd2:
105            begin
106                display_valid <= 1'b1;
107                display_name  <= "INST ";
108                display_value <= inst;
109            end
110            default :
111            begin
112                display_valid <= 1'b0;
113                display_name  <= 40'd0;
114                display_value <= 32'd0;
115            end
116        endcase
117     end
118 //-----{输出到触摸屏显示}end
119 //----------------------{调用触摸屏模块}end--------------------//
120 endmodule
```

约束文件 inst_rom.ucf 源代码：

```
#时钟信号连接
set_property PACKAGE_PIN AC19 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]

#脉冲开关，用于输入作为复位信号，低电平有效
set_property PACKAGE_PIN Y3 [get_ports resetn]
set_property IOSTANDARD LVCMOS33 [get_ports resetn]

#触摸屏引脚连接
set_property PACKAGE_PIN J25 [get_ports lcd_rst]
set_property PACKAGE_PIN H18 [get_ports lcd_cs]
set_property PACKAGE_PIN K16 [get_ports lcd_rs]
set_property PACKAGE_PIN L8 [get_ports lcd_wr]
set_property PACKAGE_PIN K8 [get_ports lcd_rd]
set_property PACKAGE_PIN J15 [get_ports lcd_bl_ctr]
set_property PACKAGE_PIN H9 [get_ports {lcd_data_io[0]}]
set_property PACKAGE_PIN K17 [get_ports {lcd_data_io[1]}]
set_property PACKAGE_PIN J20 [get_ports {lcd_data_io[2]}]
set_property PACKAGE_PIN M17 [get_ports {lcd_data_io[3]}]
set_property PACKAGE_PIN L17 [get_ports {lcd_data_io[4]}]
set_property PACKAGE_PIN L18 [get_ports {lcd_data_io[5]}]
set_property PACKAGE_PIN L15 [get_ports {lcd_data_io[6]}]
set_property PACKAGE_PIN M15 [get_ports {lcd_data_io[7]}]
set_property PACKAGE_PIN M16 [get_ports {lcd_data_io[8]}]
set_property PACKAGE_PIN L14 [get_ports {lcd_data_io[9]}]
set_property PACKAGE_PIN M14 [get_ports {lcd_data_io[10]}]
set_property PACKAGE_PIN F22 [get_ports {lcd_data_io[11]}]
set_property PACKAGE_PIN G22 [get_ports {lcd_data_io[12]}]
set_property PACKAGE_PIN G21 [get_ports {lcd_data_io[13]}]
set_property PACKAGE_PIN H24 [get_ports {lcd_data_io[14]}]
set_property PACKAGE_PIN J16 [get_ports {lcd_data_io[15]}]
set_property PACKAGE_PIN L19 [get_ports ct_int]
set_property PACKAGE_PIN J24 [get_ports ct_sda]
set_property PACKAGE_PIN H21 [get_ports ct_scl]
set_property PACKAGE_PIN G24 [get_ports ct_rstn]

set_property IOSTANDARD LVCMOS33 [get_ports lcd_rst]
set_property IOSTANDARD LVCMOS33 [get_ports lcd_cs]
set_property IOSTANDARD LVCMOS33 [get_ports lcd_rs]
set_property IOSTANDARD LVCMOS33 [get_ports lcd_wr]
set_property IOSTANDARD LVCMOS33 [get_ports lcd_rd]
set_property IOSTANDARD LVCMOS33 [get_ports lcd_bl_ctr]
set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[4]}]
```

```
48  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[5]}]
49  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[6]}]
50  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[7]}]
51  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[8]}]
52  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[9]}]
53  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[10]}]
54  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[11]}]
55  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[12]}]
56  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[13]}]
57  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[14]}]
58  set_property IOSTANDARD LVCMOS33 [get_ports {lcd_data_io[15]}]
59  set_property IOSTANDARD LVCMOS33 [get_ports ct_int]
60  set_property IOSTANDARD LVCMOS33 [get_ports ct_sda]
61  set_property IOSTANDARD LVCMOS33 [get_ports ct_scl]
62  set_property IOSTANDARD LVCMOS33 [get_ports ct_rstn]
```

5.3 异步 ram

data_ram.v 源代码：

```
1   `timescale 1ns / 1ps
2   //*************************************************************
    *
3   //   > 文件名: data_mem.v
4   //   > 描述   : 异步数据存储器模块，采用寄存器搭建而成，类似寄存器堆
5   //   >          同步写，异步读
6   //   > 作者   : LOONGSON
7   //   > 日期   : 2016-04-14
8   //*************************************************************
    *
9   module data_ram(
10      input         clk,          // 时钟
11      input  [3:0]  wen,          // 字节写使能
12      input  [4:0] addr,          // 地址
13      input  [31:0] wdata,        // 写数据
14      output reg [31:0] rdata,        // 读数据
15
16      //调试端口，用于读出数据显示
17      input  [4 :0] test_addr,
18      output reg [31:0] test_data
19  );
20      reg [31:0] DM[31:0];//数据存储器，字节地址 7'b000_0000~7'b111_1111
21
22      //写数据
23      always @(posedge clk)    // 当写控制信号为 1，数据写入内存
24      begin
25          if (wen[3])
26          begin
```

```
27              DM[addr][31:24] <= wdata[31:24];
28          end
29      end
30      always @(posedge clk)
31      begin
32          if (wen[2])
33          begin
34              DM[addr][23:16] <= wdata[23:16];
35          end
36      end
37      always @(posedge clk)
38      begin
39          if (wen[1])
40          begin
41              DM[addr][15: 8] <= wdata[15: 8];
42          end
43      end
44      always @(posedge clk)
45      begin
46          if (wen[0])
47          begin
48              DM[addr][7 : 0] <= wdata[7 : 0];
49          end
50      end
51
52      //读数据,取 4 字节
53      always @(*)
54      begin
55          case (addr)
56              5'd0 : rdata <= DM[0 ];
57              5'd1 : rdata <= DM[1 ];
58              5'd2 : rdata <= DM[2 ];
59              5'd3 : rdata <= DM[3 ];
60              5'd4 : rdata <= DM[4 ];
61              5'd5 : rdata <= DM[5 ];
62              5'd6 : rdata <= DM[6 ];
63              5'd7 : rdata <= DM[7 ];
64              5'd8 : rdata <= DM[8 ];
65              5'd9 : rdata <= DM[9 ];
66              5'd10: rdata <= DM[10];
67              5'd11: rdata <= DM[11];
68              5'd12: rdata <= DM[12];
69              5'd13: rdata <= DM[13];
70              5'd14: rdata <= DM[14];
```

```
 71              5'd15: rdata <= DM[15];
 72              5'd16: rdata <= DM[16];
 73              5'd17: rdata <= DM[17];
 74              5'd18: rdata <= DM[18];
 75              5'd19: rdata <= DM[19];
 76              5'd20: rdata <= DM[20];
 77              5'd21: rdata <= DM[21];
 78              5'd22: rdata <= DM[22];
 79              5'd23: rdata <= DM[23];
 80              5'd24: rdata <= DM[24];
 81              5'd25: rdata <= DM[25];
 82              5'd26: rdata <= DM[26];
 83              5'd27: rdata <= DM[27];
 84              5'd28: rdata <= DM[28];
 85              5'd29: rdata <= DM[29];
 86              5'd30: rdata <= DM[30];
 87              5'd31: rdata <= DM[31];
 88          endcase
 89      end
 90      //调试端口，读出特定内存的数据
 91      always @(*)
 92      begin
 93          case (test_addr)
 94              5'd0 : test_data <= DM[0 ];
 95              5'd1 : test_data <= DM[1 ];
 96              5'd2 : test_data <= DM[2 ];
 97              5'd3 : test_data <= DM[3 ];
 98              5'd4 : test_data <= DM[4 ];
 99              5'd5 : test_data <= DM[5 ];
100              5'd6 : test_data <= DM[6 ];
101              5'd7 : test_data <= DM[7 ];
102              5'd8 : test_data <= DM[8 ];
103              5'd9 : test_data <= DM[9 ];
104              5'd10: test_data <= DM[10];
105              5'd11: test_data <= DM[11];
106              5'd12: test_data <= DM[12];
107              5'd13: test_data <= DM[13];
108              5'd14: test_data <= DM[14];
109              5'd15: test_data <= DM[15];
110              5'd16: test_data <= DM[16];
111              5'd17: test_data <= DM[17];
112              5'd18: test_data <= DM[18];
113              5'd19: test_data <= DM[19];
114              5'd20: test_data <= DM[20];
```

```
            5'd21: test_data <= DM[21];
            5'd22: test_data <= DM[22];
            5'd23: test_data <= DM[23];
            5'd24: test_data <= DM[24];
            5'd25: test_data <= DM[25];
            5'd26: test_data <= DM[26];
            5'd27: test_data <= DM[27];
            5'd28: test_data <= DM[28];
            5'd29: test_data <= DM[29];
            5'd30: test_data <= DM[30];
            5'd31: test_data <= DM[31];
        endcase
    end
endmodule
```

data_ram_display.v 源代码：

```
`timescale 1ns / 1ps
//*************************************************************************
//   > 文件名: data_ram_display.v
//   > 描述   : 数据存储器模块显示模块，调用 FPGA 板上的 IO 接口和触摸屏
//   > 作者   : LOONGSON
//   > 日期   : 2016-04-14
//*************************************************************************
module data_ram_display(
    //时钟与复位信号
     input clk,
    input resetn,    //后缀"n"代表低电平有效

    //拨码开关，用于产生写使能和选择输入数
    input [3:0] wen,
    input [1:0] input_sel,

    //led 灯，用于指示写使能信号，和正在输入什么数据
    output [3:0] led_wen,
    output led_addr,      //指示输入读写地址
    output led_wdata,     //指示输入写数据
    output led_test_addr, //指示输入 test 地址

    //触摸屏相关接口，不需要更改
    output lcd_rst,
    output lcd_cs,
    output lcd_rs,
    output lcd_wr,
```

```
28      output lcd_rd,
29      inout[15:0] lcd_data_io,
30      output lcd_bl_ctr,
31      inout ct_int,
32      inout ct_sda,
33      output ct_scl,
34      output ct_rstn
35      );
36  //-----{LED显示}begin
37      assign led_wen      = wen;
38      assign led_addr     = (input_sel==2'd0);
39      assign led_wdata    = (input_sel==2'd1);
40      assign led_test_addr = (input_sel==2'd2);
41  //-----{LED显示}end
42  //-----{调用数据储存器模块}begin
43      //数据存储器多增加一个读端口，用于读出特定内存地址显示在触摸屏上
44      reg  [31:0] addr;
45      reg  [31:0] wdata;
46      wire [31:0] rdata;
47      reg  [31:0] test_addr;
48      wire [31:0] test_data;
49
50      data_ram data_ram_module(
51          .clk   (clk   ),
52          .wen   (wen   ),
53          .addr  (addr[6:2]),
54          .wdata (wdata ),
55          .rdata (rdata ),
56          .test_addr(test_addr[6:2]),
57          .test_data(test_data)
58      );
59  //-----{调用寄存器堆模块}end
60
61  //----------------------{调用触摸屏模块}begin--------------------//
62  //-----{实例化触摸屏}begin
63  //此小节不需要更改
64      reg        display_valid;
65      reg  [39:0] display_name;
66      reg  [31:0] display_value;
67      wire [5 :0] display_number;
68      wire        input_valid;
69      wire [31:0] input_value;
70
71      lcd_module lcd_module(
```

```
72          .clk           (clk          ),   //10Mhz
73          .resetn        (resetn       ),
74
75          //调用触摸屏的接口
76          .display_valid  (display_valid ),
77          .display_name   (display_name  ),
78          .display_value  (display_value ),
79          .display_number (display_number),
80          .input_valid    (input_valid   ),
81          .input_value    (input_value   ),
82
83          //lcd 触摸屏相关接口，不需要更改
84          .lcd_rst       (lcd_rst      ),
85          .lcd_cs        (lcd_cs       ),
86          .lcd_rs        (lcd_rs       ),
87          .lcd_wr        (lcd_wr       ),
88          .lcd_rd        (lcd_rd       ),
89          .lcd_data_io    (lcd_data_io   ),
90          .lcd_bl_ctr     (lcd_bl_ctr    ),
91          .ct_int        (ct_int       ),
92          .ct_sda        (ct_sda       ),
93          .ct_scl        (ct_scl       ),
94          .ct_rstn       (ct_rstn      )
95      );
96  //-----{实例化触摸屏}end
97
98  //-----{从触摸屏获取输入}begin
99  //根据实际需要输入的数修改此小节，
100 //建议对每一个数的输入，编写单独一个 always 块
101     //当 input_sel 为 2'b00 时，表示输入数为读写地址，即 addr
102     always @(posedge clk)
103     begin
104         if (!resetn)
105         begin
106             addr <= 32'd0;
107         end
108         else if (input_valid && input_sel==2'd0)
109         begin
110             addr[31:2] <= input_value[31:2];
111         end
112     end
113
114     //当 input_sel 为 2'b01 时，表示输入数为写数据，即 wdata
115     always @(posedge clk)
```

```
116     begin
117         if (!resetn)
118         begin
119             wdata <= 32'd0;
120         end
121         else if (input_valid && input_sel==2'd1)
122         begin
123             wdata <= input_value;
124         end
125     end
126 
127     //当 input_sel 为 2'b10 时，表示输入数为 test 地址，即 test_addr
128     always @(posedge clk)
129     begin
130         if (!resetn)
131         begin
132             test_addr  <= 32'd0;
133         end
134         else if (input_valid && input_sel==2'd2)
135         begin
136             test_addr[31:2] <= input_value[31:2];
137         end
138     end
139 //-----{从触摸屏获取输入}end
140 
141 //-----{输出到触摸屏显示}begin
142 //根据需要显示的数修改此小节，
143 //触摸屏上共有 44 块显示区域，可显示 44 组 32 位数据
144 //44 块显示区域从 1 开始编号，编号为 1~44，
145     always @(posedge clk)
146     begin
147       case(display_number)
148           6'd1:
149           begin
150               display_valid <= 1'b1;
151               display_name  <= "ADDR ";
152               display_value <= addr;
153           end
154           6'd2:
155           begin
156               display_valid <= 1'b1;
157               display_name  <= "WDATA";
158               display_value <= wdata;
159           end
```

```
160             6'd3:
161             begin
162                 display_valid <= 1'b1;
163                 display_name  <= "RDATA";
164                 display_value <= rdata;
165             end
166             6'd5:
167             begin
168                 display_valid <= 1'b1;
169                 display_name  <= "T_ADD";
170                 display_value <= test_addr;
171             end
172             6'd6:
173             begin
174                 display_valid <= 1'b1;
175                 display_name  <= "T_DAT";
176                 display_value <= test_data;
177             end
178             default :
179             begin
180                 display_valid <= 1'b0;
181                 display_name  <= 40'd0;
182                 display_value <= 32'd0;
183             end
184         endcase
185     end
186 //-----{输出到触摸屏显示}end
187 //----------------------{调用触摸屏模块}end---------------------//
188 endmodule
```

异步 data_ram 在烧写到板上展示时需要的约束文件和同步 data_ram 的一样。

5.4 异步 ROM

inst_rom.v 源代码：

```
1   `timescale 1ns / 1ps
2   //*************************************************************
    **
3   //   > 文件名: inst_rom.v
4   //   > 描述   : 异步指令存储器模块，采用寄存器搭建而成，类似寄存器堆
5   //   >          内嵌好指令，只读，异步读
6   //   > 作者   : LOONGSON
7   //   > 日期   : 2016-04-14
8   //*************************************************************
    **
9   module inst_rom(
10      input     [4 :0] addr, // 指令地址
```

```
11      output reg [31:0] inst      // 指令
12      );
13
14      wire  [31:0]  inst_rom[19:0];    //  指 令 存 储 器 ， 字 节 地 址
    7'b000_0000~7'b111_1111
15      //------------ 指令编码 ----------|指令地址|---- 汇编指令 ----|- 指令结果 -----//
16      assign inst_rom[ 0] = 32'h24010001;//00H: addiu $1 ,$0,#1  | $1 = 0000_0001H
17      assign inst_rom[ 1] = 32'h00011100;//04H: sll   $2 ,$1,#4  | $2 = 0000_0010H
18      assign inst_rom[ 2] = 32'h00411821;//08H: addu  $3 ,$2,$1  | $3 = 0000_0011H
19      assign inst_rom[ 3] = 32'h00022082;//0CH: srl   $4 ,$2,#2  | $4 = 0000_0004H
20      assign inst_rom[ 4] = 32'h00642823;//10H: subu  $5 ,$3,$4  | $5 = 0000_000DH
21      assign inst_rom[ 5] = 32'hAC250013;//14H: sw    $5 ,#19($1)| Mem[14H] = DH
22      assign inst_rom[ 6] = 32'h00A23027;//18H: nor   $6 ,$5,$2  | $6 = FFFF_FFE2H
23      assign inst_rom[ 7] = 32'h00C33825;//1CH: or    $7 ,$6,$3  | $7 = FFFF_FFF3H
24      assign inst_rom[ 8] = 32'h00E64026;//20H: xor   $8 ,$7,$6  | $8 = 0000_0011H
25      assign inst_rom[ 9] = 32'hAC08001C;//24H: sw    $8 ,#28($0)| Mem[1CH] = 11H
26      assign inst_rom[10] = 32'h00C7482A;//28H: slt   $9 ,$6,$7  | $9 = 0000_0001H
27      assign inst_rom[11] = 32'h11210002;//2CH: beq   $9 ,$1,#2  | 跳转到指令 34H
28      assign inst_rom[12] = 32'h24010004;//30H: addiu $1 ,$0,#4  | 不执行
29      assign inst_rom[13] = 32'h8C2A0013;//34H: lw    $10,#19($1)| $10 = 0000_000DH
30      assign inst_rom[14] = 32'h15450003;//38H: bne   $10,$5,#3  | 不跳转
31      assign inst_rom[15] = 32'h00415824;//3CH: and   $11,$2,$1  | $11 = 0000_0000H
32      assign inst_rom[16] = 32'hAC0B001C;//40H: sw    $11,#28($0)| Men[1CH] = 0H
33      assign inst_rom[17] = 32'hAC040010;//44H: sw    $4 ,#16($0)| Mem[10H] = 4H
34      assign inst_rom[18] = 32'h3C0C000C;//48H: lui   $12,#12    | [R12] = C_0000H
35      assign inst_rom[19] = 32'h08000000;//4CH: j     00H       | 跳转指令 00H
36
37      //读指令,取 4 字节
38      always @(*)
39      begin
40          case (addr)
41              5'd0 : inst <= inst_rom[0 ];
42              5'd1 : inst <= inst_rom[1 ];
43              5'd2 : inst <= inst_rom[2 ];
44              5'd3 : inst <= inst_rom[3 ];
45              5'd4 : inst <= inst_rom[4 ];
46              5'd5 : inst <= inst_rom[5 ];
47              5'd6 : inst <= inst_rom[6 ];
48              5'd7 : inst <= inst_rom[7 ];
49              5'd8 : inst <= inst_rom[8 ];
50              5'd9 : inst <= inst_rom[9 ];
51              5'd10: inst <= inst_rom[10];
52              5'd11: inst <= inst_rom[11];
53              5'd12: inst <= inst_rom[12];
```

```
            5'd13: inst <= inst_rom[13];
            5'd14: inst <= inst_rom[14];
            5'd15: inst <= inst_rom[15];
            5'd16: inst <= inst_rom[16];
            5'd17: inst <= inst_rom[17];
            5'd18: inst <= inst_rom[18];
            5'd19: inst <= inst_rom[19];
            default: inst <= 32'd0;
        endcase
    end
endmodule
```

inst_rom_display.v 源代码：

```
`timescale 1ns / 1ps
//*************************************************************
*
//   > 文件名: inst_rom_display.v
//   > 描述   : 异步指令存储器显示模块，调用 FPGA 板上的 IO 接口和触摸屏
//   > 作者   : LOONGSON
//   > 日期   : 2016-04-14
//*************************************************************
*
module inst_rom_display(
    //时钟与复位信号
    input clk,
    input resetn,    //后缀"n"代表低电平有效

    //触摸屏相关接口，不需要更改
    output lcd_rst,
    output lcd_cs,
    output lcd_rs,
    output lcd_wr,
    output lcd_rd,
    inout[15:0] lcd_data_io,
    output lcd_bl_ctr,
    inout ct_int,
    inout ct_sda,
    output ct_scl,
    output ct_rstn
    );
//-----{调用数据储存器模块}begin
    //数据存储器多增加一个读端口，用于读出特定内存地址显示在触摸屏上
    reg  [31:0] addr;
    wire [31:0] inst;

```

```
31      inst_rom inst_rom_module(
32          .addr  (addr[31:2]),
33          .inst  (inst     )
34      );
35  //-----{调用寄存器堆模块}end
36
37  //---------------------{调用触摸屏模块}begin--------------------//
38  //-----{实例化触摸屏}begin
39  //此小节不需要更改
40      reg        display_valid;
41      reg  [39:0] display_name;
42      reg  [31:0] display_value;
43      wire [5 :0] display_number;
44      wire       input_valid;
45      wire [31:0] input_value;
46
47      lcd_module lcd_module(
48          .clk          (clk         ),   //10Mhz
49          .resetn       (resetn      ),
50
51          //调用触摸屏的接口
52          .display_valid  (display_valid ),
53          .display_name   (display_name  ),
54          .display_value  (display_value ),
55          .display_number (display_number),
56          .input_valid    (input_valid   ),
57          .input_value    (input_value   ),
58
59          //lcd 触摸屏相关接口，不需要更改
60          .lcd_rst       (lcd_rst     ),
61          .lcd_cs        (lcd_cs      ),
62          .lcd_rs        (lcd_rs      ),
63          .lcd_wr        (lcd_wr      ),
64          .lcd_rd        (lcd_rd      ),
65          .lcd_data_io    (lcd_data_io  ),
66          .lcd_bl_ctr     (lcd_bl_ctr   ),
67          .ct_int        (ct_int      ),
68          .ct_sda        (ct_sda      ),
69          .ct_scl        (ct_scl      ),
70          .ct_rstn       (ct_rstn     )
71      );
72  //-----{实例化触摸屏}end
73
74  //-----{从触摸屏获取输入}begin
```

```
75  //根据实际需要输入的数修改此小节，
76  //建议对每一个数的输入，编写单独一个 always 块
77      always @(posedge clk)
78      begin
79          if (!resetn)
80          begin
81              addr <= 32'd0;
82          end
83          else if (input_valid)
84          begin
85              addr[31:2] <= input_value[31:2];
86          end
87      end
88  //-----{从触摸屏获取输入}end
89
90  //-----{输出到触摸屏显示}begin
91  //根据需要显示的数修改此小节，
92  //触摸屏上共有 44 块显示区域，可显示 44 组 32 位数据
93  //44 块显示区域从 1 开始编号，编号为 1~44，
94      always @(posedge clk)
95      begin
96         case(display_number)
97             6'd1:
98             begin
99                 display_valid <= 1'b1;
100                display_name  <= "ADDR ";
101                display_value <= addr;
102            end
103            6'd2:
104            begin
105                display_valid <= 1'b1;
106                display_name  <= "INST ";
107                display_value <= inst;
108            end
109            default :
110            begin
111                display_valid <= 1'b0;
112                display_name  <= 40'd0;
113                display_value <= 32'd0;
114            end
115        endcase
116     end
117 //-----{输出到触摸屏显示}end
118 //----------------------{调用触摸屏模块}end--------------------//
```

```
119   endmodule
```

异步 inst_rom 在烧写到板上展示时需要的约束文件和同步 inst_rom 的一样。

7、 实验结果分析

00 表示输入数为读写地址，即 addr，

01 表示输入数为写数据，即 wdata

10 表示输入数为 test 地址，即 test_addr

（1） 同步 ram

根据 input_sel 的值选择相应的输入数据类型。

当 input_sel 为 2'b00 时，表示输入数为读写地址，即 addr。在每个时钟上升沿，如果 input_valid 为 1 且 input_sel 等于 2'b00，则将 input_value 的高 30 位赋值给 addr 的高 30 位。

当 input_sel 为 2'b01 时，表示输入数为写数据，即 wdata。在每个时钟上升沿，如果 input_valid 为 1 且 input_sel 等于 2'b01，则将 input_value 赋值给 wdata。

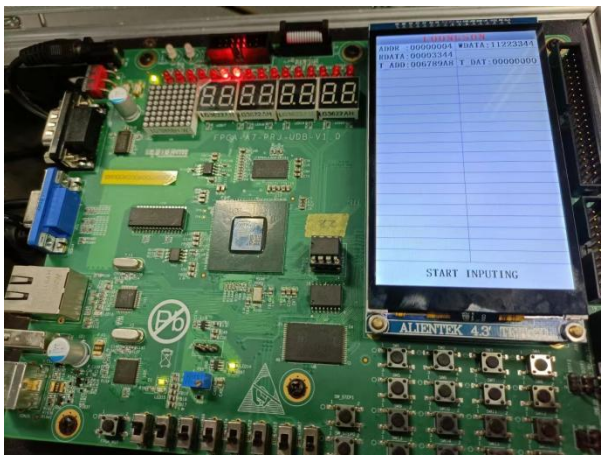
当 input_sel 为 2'b10 时，表示输入数为 test 地址，即 test_addr。在每个时钟上升沿，如果 input_valid 为 1 且 input_sel 等于 2'b10，则将 input_value 的高 30 位赋值给 test_addr 的高 30 位。

addr=12345678，wdata=12ABCDEF，t_add=167995A8，rdata=0000CDEF。结果正确。

（2） 同步 rom

当输入有效(input_valid 为高电平)时，将 input_value 的高 30 位存储在 addr 的高 30 位。addr=00000014，inst=07210010，结果正确。

（3） 异步 ram



输入信号：clk（时钟），resetn（低电平有效的复位信号）

输入信号：

wen（字节写使能，4 位）：用于控制写操作的使能信号。

addr（地址，5 位）：写入或读取数据的地址。

wdata（写数据，32 位）：要写入 RAM 的数据。

input_sel（输入选择，2 位）：用于选择输入数据的类型。

输出信号：

rdata（读数据，32 位）：从 RAM 读取的数据。

led_wen（写使能指示灯，4 位）：用于指示写使能信号。

led_addr（地址指示灯）：用于指示输入的读写地址。

led_wdata（写数据指示灯）：用于指示输入的写数据。

led_test_addr（测试地址指示灯）：用于指示输入的测试地址。

内部信号：DM（数据存储器）：存储数据的寄存器堆，大小为 32 个 32 位寄存器。

写数据：

使用 always 块根据 wen 信号和地址进行写操作，将 wdata 的不同字节写入对应地址的寄存器。分别使用 wen[3]、wen[2]、wen[1]、wen[0]控制写入高字节、次高字节、次低字节、低字节。

读数据：

使用 always @(*)块根据地址选择对应的寄存器中的数据，并将其赋值给 rdata。

通过 case 语句根据 addr 的值选择对应的 DM 寄存器，并将其值赋给 rdata。

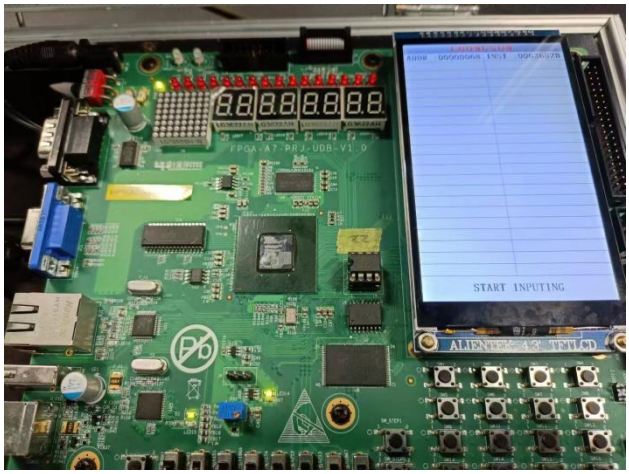
调试端口：

使用 always @(*)块根据 test_addr 选择对应的寄存器中的数据，并将其赋值给 test_data。

异步 RAM 模块具有 32 个 32 位寄存器，可以进行异步读取和同步写入操作。通过 wen 信号控制写使能，通过 addr 信号选择读取的地址，通过 wdata 写入数据，通过 rdata 读取数据。通过调试端口和触摸屏接口，可以方便地显示和输入数据。

addr=00000004，wdata=11223344，rdara=00003344，t_add=006789A8，结果正确。

（4） 异步 rom



addr：用于存储输入的指令地址

inst：用于存储从异步指令存储器中读取的指令内容

Addr=00000068，inst=0063682B，结果正确。

8、 思考题

（1） ROM 和 RAM 的区别；

1） RAM

a) RAM 是一种可读写的存储器，允许数据的读取和写入操作。RAM 中的数据可以根据需要进行写入和修改，可以在运行时写入和修改数据。

b) RAM 支持随机访问，可以通过内部地址直接访问存储单元，读写速度较快。

c) RAM 主要用于临时存储数据和程序，它提供了计算机系统的工作区域。

d) RAM 是一种易失性存储器，当电源关闭时，其中存储的数据将丢失。

2） ROM

a) ROM 是一种只读存储器，其中的数据在制造过程中被预先编程，并且无法被修改。

b) ROM 支持顺序访问，数据的读取是按照顺序逐个字节或字进行的。

c) ROM 中存储的数据是持久的，即使断电也不会丢失。

d) ROM 中的数据是在制造过程中被固化的，无法被用户修改，因此通常用于存储固定的程序指令、固件和数据表。

（2） 分析一下同步存储器和异步存储器的特点，思考说明一下何时需要使用同步存储器，何时需要使用异步存储器。

1） 同步存储器的特点

a) 时钟同步：同步存储器的读写操作是与时钟信号同步的，读写操作在时钟的上升沿或下降沿进行。

b) 数据稳定性：同步存储器在时钟信号稳定的边沿进行读写操作，保证数据的稳定性和一致性。

c) 同步性能：同步存储器具有较高的同步性能，适用于需要高速读写和频繁的数据交换操作的场景。

d) 控制复杂：同步存储器的控制逻辑相对复杂，需要考虑时序和同步问题。

2） 异步存储器的特点

a) 无需时钟同步：异步存储器的读写操作不依赖于时钟信号，可以根据特定的信号来进行读写操作。

b) 简单控制：异步存储器的控制逻辑相对简单，不需要考虑时钟同步和复杂

的时序问题。

c) 异步性能：异步存储器的读写性能相对较低，访问速度较慢，适用于对速度要求不高或者访问频率较低的场景。

d) 数据不稳定：由于异步存储器没有时钟同步，读取的数据可能存在不稳定性和不一致性。

3） 何时需要使用同步存储器

a) 高速读写要求：当需要频繁进行高速读写操作时，同步存储器是更好的选择，因为它具有较高的同步性能和稳定性。

b) 复杂控制需求：如果存储器系统的控制逻辑相对复杂，需要考虑时序和同步问题时，同步存储器更适合。

c) 数据一致性要求：当需要保证读取数据的稳定性和一致性时，同步存储器能够提供可靠的数据输出。

4） 何时需要使用异步存储器

a) 低速读写要求：当读写操作的速度要求不高或者访问频率较低时，异步存储器可以满足需求，并且具有更简单的控制逻辑。

b) 简单控制需求：对于控制逻辑较简单的存储器系统，不需要考虑时序和同步问题时，异步存储器是更合适的选择。

c) 经济成本因素：异步存储器通常比同步存储器成本更高，因为同步存储器需要额外的时钟信号和复杂的控制逻辑，而异步存储器的设计相对简单，因此在经济成本方面，异步存储器可能更具有优势。

9、 总结感想

通过实验深入了解了存储器的基本原理和组成结构。了解到存储器可以分为随机存储器（RAM）和只读存储器（ROM），并学会了它们的工作原理和特点。通过编写代码和搭建电路，我实际上实现了一个存储器模块。这个过程让我更好地理解了存储器的设计和实现过程，包括地址线、数据线、读写控制信号等的连接和使用。学习了同步存储器和异步存储器的特点和区别。同步存储器需要时钟信号和复杂的控制逻辑，适用于高性能和精确同步的应用；而异步存储器相对简单，适用于低功耗和简单控制的应用。存储器是计算机系统中重要的组成部分，对系统性能和功能起着至关重要的作用。通过这次实验，我深刻认识到了存储器在系统设计中的重要性，以及在不同场景下选择适合的存储器类型的重要性。