版本历史

文档更新记录	文档名:	Lab01_同步存储器
	版本号	V0.1
	创建人:	计算机体系结构研讨课教学组
	创建日期 :	2017-09-07

更新历史

序号	更新日期	更新人	版本号	更新内容
1	2017/09/01	邢金璋	V0.1	初版。

文档信息反馈: xingjinzhang@loongson.cn

1 实验一 同步存储器实现与替换

在学习并尝试本章节前，你需要具有以下环境和能力：

(1) 装有 Vivado2017.1 的电脑一台。

由于 Vivado2017.1 下载、安装繁琐，本次实验不要求一定要使用 Vivado2017.1，但希望后续大家逐步转移到 Vivado2017.1 平台上。

(2) 熟悉 Vivado2017.1，并能初步使用。

如果对 Vivado 不熟悉，请参考文档“A04_Vivado2017.1 使用说明”。

(3) 熟悉龙芯体系结构实验箱（Artix-7）。

请确保，你有阅读过文档“A01_龙芯体系结构教学实验箱（Artix-7）介绍”。

通过本章节的学习，你将获得：

(1) Vivado2017.1 生成 RAM IP 的方法。

(2) FPGA 开发的实践经验，如 testbench 编写、仿真、综合实现等等。

1.1 实验目的

1. 了解随机存取存储器 RAM 的原理。
2. 理解 RAM 读取、写入数据的过程。
3. 理解同步 RAM 和异步 RAM 的区别。
4. 掌握调用 Xilinx 库 IP 实例化 RAM 的设计方法。
5. 熟悉并运用 verilog 语言进行电路设计。

1.2 实验设备

1. 装有 Xilinx Vivado 的计算机一台。
2. 龙芯体系结构教学实验箱（Artix-7）一套。

1.3 实验任务

1. 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。
2. 调用 Xilinx 库 IP 实例化一块 RAM，并进行仿真，得到正确的波形图。要求自己编写一个 RAM 初始化的文件，在生成 IP 时加载进入 RAM 中，初始化内容自定义。
3. 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 1-1。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。
4. 以 display 模块为顶层进行综合布局布线，并下载到实验箱中的 FPGA 板子上进行演示。
5. 本次实验不需要提交实验报告，以现场查看仿真和上板运行情况为评分标准。

(1) 仿真检查：只对 RAM IP 仿真，要求自己编写仿真顶层(testbench)，设置输入激励。如果不了解仿真顶层的编写和输入激励的设置，可以参考文档“A04_Vivado2017.1 使用说明”第 1.3 节 功能仿真。

(2) 上板检查：要求对 display 模块进行综合实现，display 模块会调用 RAM IP 和 LCD 模块。现场演示，按照检查人员的要求，对特定存储器单元读/写。

6. 附加任务：在图 2-5 中自学“Port A Optional Output Registers”选项下的各项的意思，并通过仿真查看其区别。

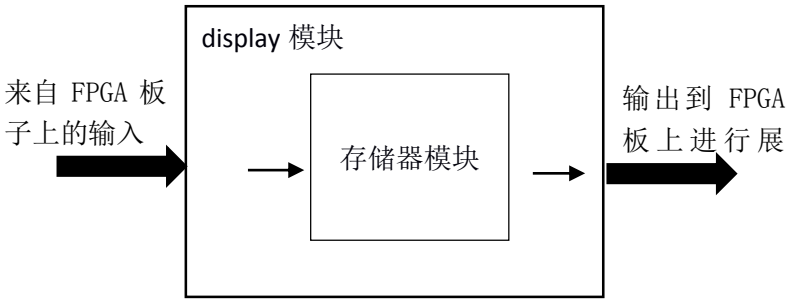

图 1-1 存储器设计实验的顶层模块大致框图

1.4 实验环境

以下红色部分，为本次实验需要大家自行实现的。黑色部分已发布到课程网站上 lab1 目录下。

|--testbench.v	仿真顶层，需自行编写。可参考“A04_Vivado2017.1 使用说明”第 1.3 节。
| |--ram ip	本次实验的重点
| |--ram.coe	Ram 初始化文件，需自行编写。参考本章第 2.4 节。
|	
|--ram_display.v	综合顶层，已提供。
| |--ram ip	同 testbench 下调用的 ram ip
| |--lcd_module.dcp	LCD 屏硬件驱动模块，已提供，直接添加到本实验工程中即可。
|--ram.xdc	综合实现时，约束文件，已提供。

2 调用 Xilinx 库 IP 的方法

本部分以生成同步 RAM 为例说明调用 xilinx 库 IP 的方法，以下示例使用的是 Vivado2015.3，未能及时更新到 Vivado2017.1，但步骤是基本一致的。

2.1 新建工程

新建一个工程 data_ram，参考文档“A04_Vivado2017.1 使用说明”：

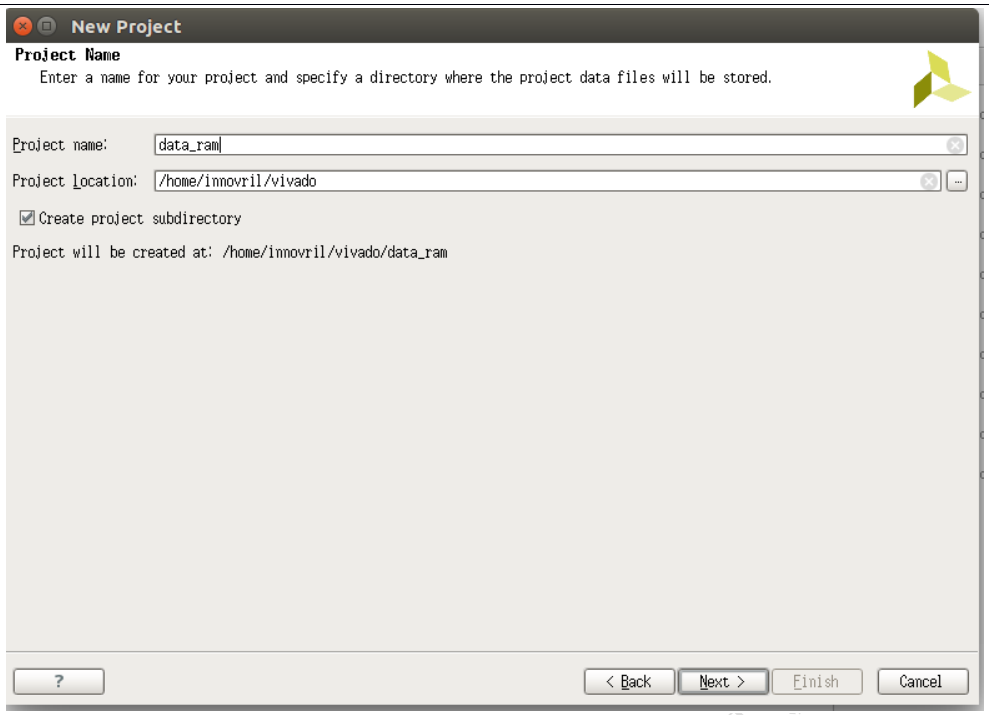

图 2-1 新建工程 data_ram

2.2 新建 IP

点击“IP Catalog”：

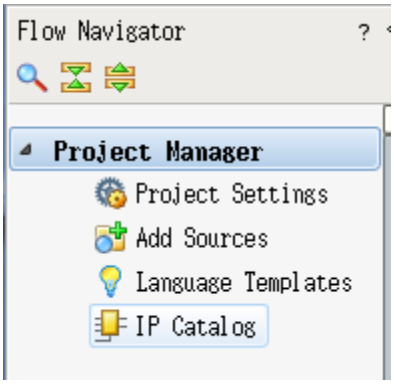

图 2-2 打开 IP 目录

在右侧列表中双击选择"Memories and Storage Elements"->"RAMs & ROMs & BRAM"中的"Block Memory Generator"：

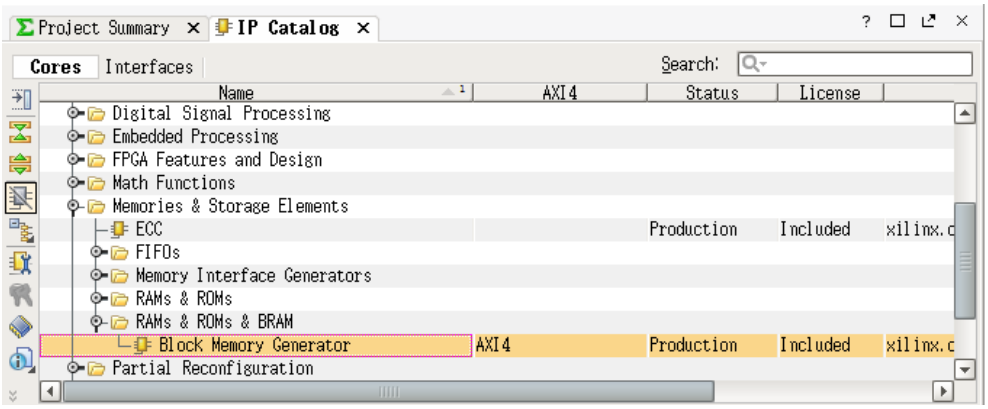

图 2-3 选择 IP 类型

2.3 设置 RAM 参数

完成第二步后，会出现如下界面，需要依次选择 Memory 的参数：

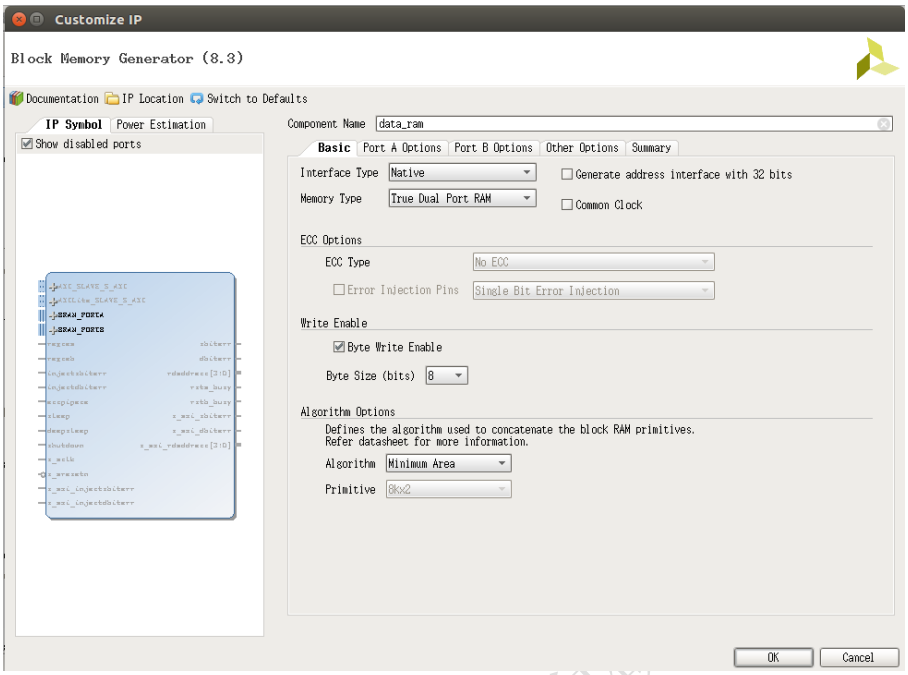

图 2-4 Memory 定制界面

在上图中，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，因为后续 CPU 实验中存在写一个字节的 store 指令，故需要数据 RAM 为字节写使能。"Basic"部分设置完成后击“Port A options”选项卡：

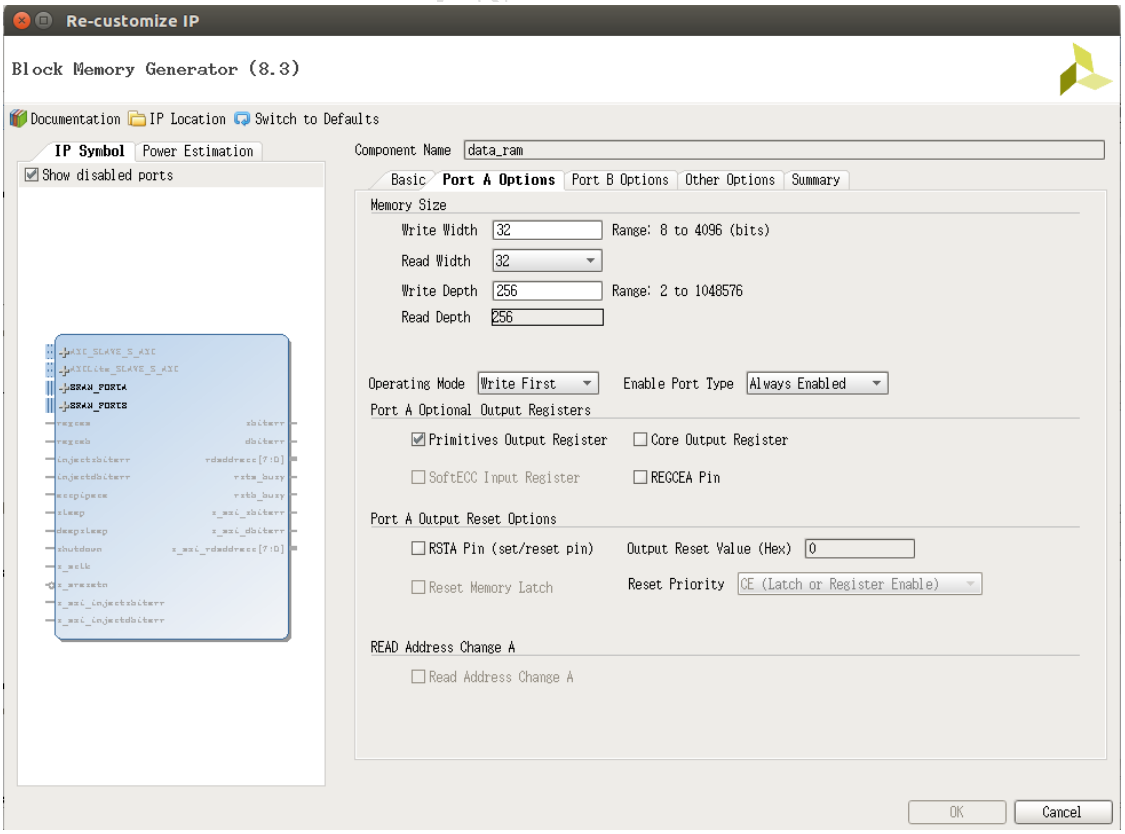

图 2-5 Memory Port A 参数设定

在上图中，RAM 宽度设置为 32 位，深度为 256。"Enalbe Port Type"选择"Always Enabled"，则表示不生成片选

信号。“Port A Optional Output Registers”请自学三个选项的意思，并尝试勾选，生成 IP 进行仿真，观察区别。选择在“Port B options”选项卡下进行同样的设置。

对于一般的 RAM 生成，后续的步骤都不需要了，故此处直接点击“OK”，在弹出的窗口中点击“Generate”生成 IP 核即可。

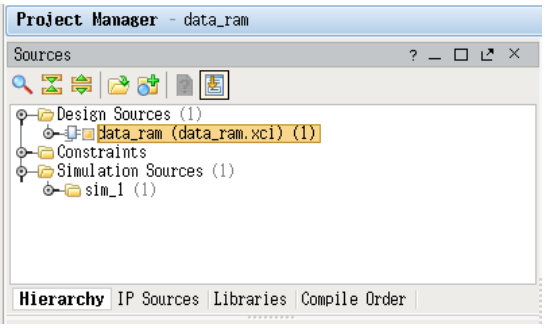

图 2-6 生成 RAM 成功

生成成功的工程管理界面如上图，.xci 文件为存放 IP 核配置信息的文件，以后需要更改直接双击该文件即可重新配置 IP 核。

2.4 设置初始化数据

RAM IP 中是可以设置初始化数据，在图 2-5 中，点击“Other Options”选项卡：

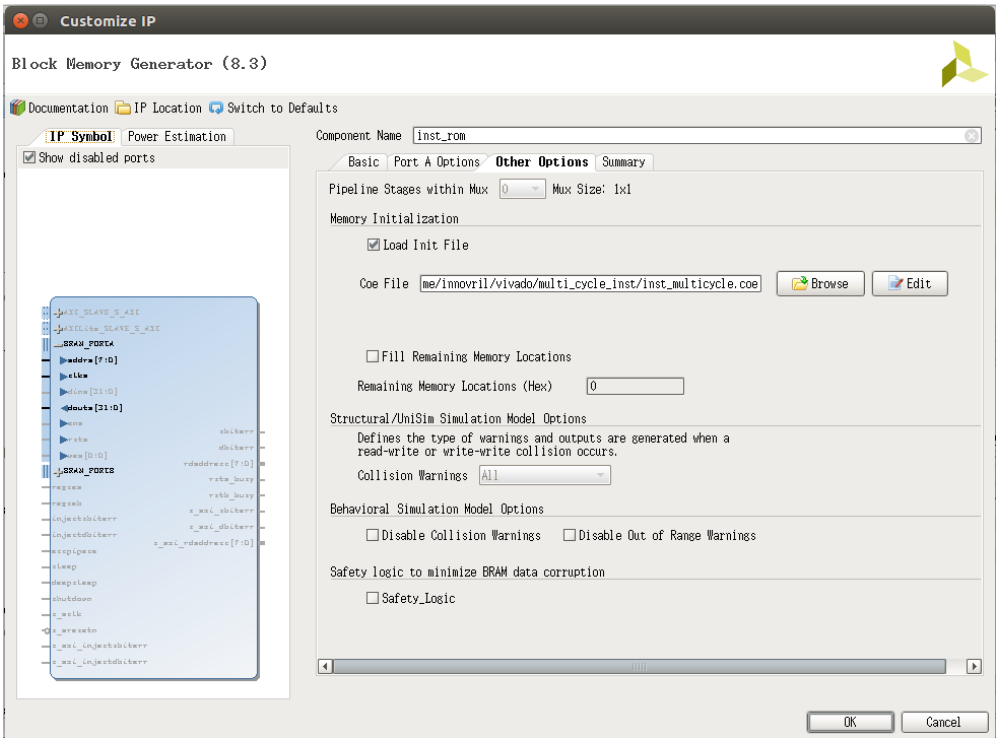

图 2-7 装载初始数据

在上图中，需要勾选“Load Init File”,并选中需要装载的初始化文件(.coe 文件)。.coe 文件为 Vivado 中存储器初始化文件，其格式如下：

```
1  memory_initialization_radix = 16;
2  memory_initialization_vector =
3  24010001
4  00011100
5  ……
```

第一行指定了初始化数据格式，此处为 16 进制，也可以设置为 2 进制。第二行说明从第三行开始为初始化的

数据向量，由于宽度为 32 位，故一个初始化向量为 32 位数据。初始化向量之间必须用空格或换行符隔开，此处使用换行符，故一行为一个初始化向量。初始化数据会从 RAM 中的 0 地址处开始依次填充。当初始化数据格式设置为 2 进制时，后续的初始化向量需要用二进制编写。

2.5 coe 文件和 mif 文件的区别

经过以上步骤，已经生成一个 ram 的 IP，假设上述建立的工程名为 A，则生成的 RAM IP 位于 A/A.srcssources_1/ip/目录下，其目录结构类似下图：

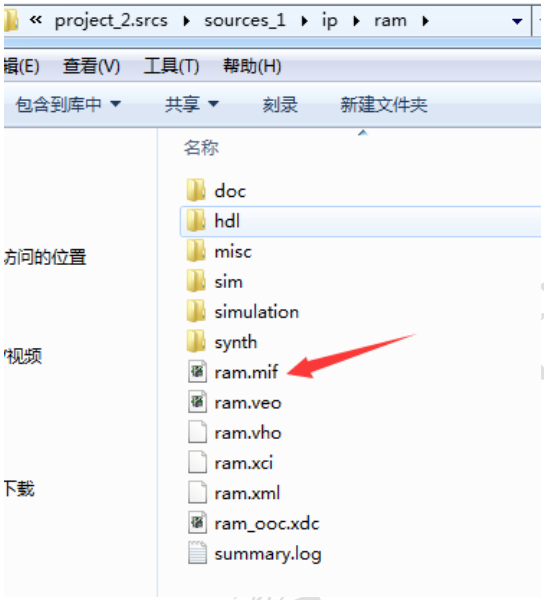

图 2-8 生成的 ram ip 目录

可以看到有一个后缀为 mif 的文件，该文件是由初始化文件 coe 文件转换得到的。也就是在定制 RAM IP 时，如果加载了初始化文件 coe，则会自动在生成的 IP 目录下产生一个 mif 文件，该文件存储了一行行的二进制数据，对应 coe 文件的第 3 行以后的数据，如下图。

```
memory_initialization_radix = 16;
memory_initialization_vector =
3c044f00
24020000
001f6021
00008021
00009021
3c081234
35085678
3c091234
……
```

(a) coe 文件

```
00111100000001000100111100000000
00100100000000100000000000000000
00000000000111110110000000100001
00000000000000001000000000100001
00000000000000001001000000100001
00111100000010000001001000110100
00110101000010000101011001111000
00111100000010010001001000110100
……
```

(b) mif 文件

图 2-9 coe 文件与对应 mif 文件的对比

如果点开 Vivado 工程里生成的 ram 的仿真模型库文件（每个 Xilinx IP 都有对应的仿真模型库，用于仿真），如下图：

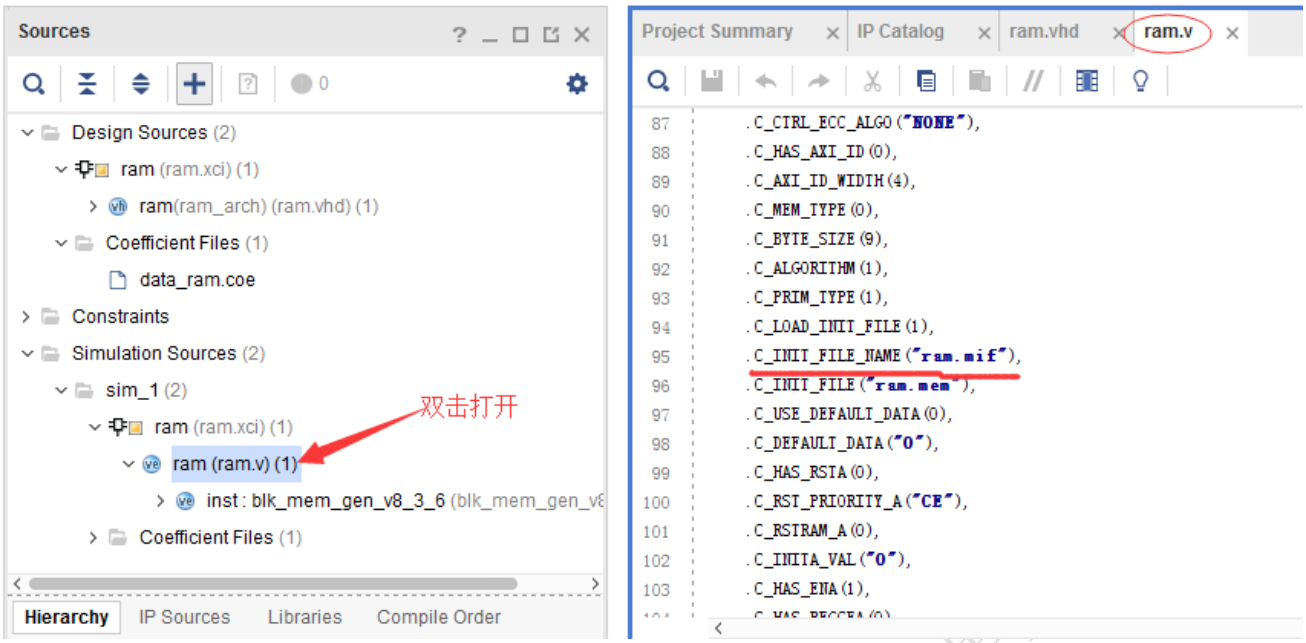

图 2-10 打开 RAM IP 的仿真库文件

可以看到源码里有一行“.C_INIT_FILE_NAME(“ram.mif”),”，从字面意思能看出了，这是仿真里载入的初始化文件。

因此，生成RAM IP时，附带生成的mif文件其存储的数据同coe文件一样，但是其是用于仿真时读入的文件。那综合实现时 RAM 中的初始化数据又记录在哪呢？其实综合时的初始化文件已经被包含在 RAM 的网表里了。也就是说，如果我们要修改仿真时 RAM 里的初始化文件，可以只用修改 mif 文件。但如果我们要修改综合实现时 RAM 里的初始化文件，目前只能重新加载 coe 文件。

3 LCD 触摸屏模块调用方法

正如前面所述，ram_display.v 作为综合实现的顶层模块，里面实例化了 RAM IP，同时也调用了 lcd_module 用来上板演示。此小节着重讲解 ram_display.v 里调用 LCD 触摸屏的方法。

3.1 LCD 触摸屏调用接口

lcd_module.v 的代码：

```
//*************************************************************
//   > 文件名: lcd_module.v
//   > 描述   : lcd 触摸屏模块，为黑盒文件
//   > 作者   : LOONGSON
//   > 日期   : 2016-04-14
//*************************************************************
//synthesis attribute box_type <lcd_module> "black_box"
module lcd_module(
    input  clk,         //10Mhz
    input  resetn,    //低使能

    //调用触摸屏的接口
    input         display_valid,
    input  [39:0] display_name,
    input  [31:0] display_value,
    output [ 5:0] display_number,
    output        input_valid,
    output [31:0] input_value,

    //lcd 触摸屏相关接口，不需要更改
    output reg   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
    );
endmodule
```

可以看到该模块为黑盒文件，其中时钟和复位信号以及 LCD 触摸屏引脚接口不需要关注，下面介绍调用触摸屏的 6 个接口。

3.2 调用 LCD 触摸屏显示

```
    input         display_valid,
    input  [39:0] display_name,
    input  [31:0] display_value,
    output [ 5:0] display_number,
```

display 的 4 个接口用于在屏上显示数据。LCD 屏用于显示的区域块共 f 分为 2 列，22 行，故共可显示 44 组数据。

display_number 就是输出到外部说明当前需要显示的区域块为第几块，有效编号从 1~44，指示 44 块显示区域块。

每块显示区域块可显示 14 个字符，其中头五个字符为块名，指示当前块显示的数据的意义，由 display_name

输入指定。display_name 输入的为要显示字符的 ASCII 码，5 个 ASCII 码共 40 位。块名可显示字符为大写的 26 个字母，下划线”_”，0~9 数字以及空格。

每块显示区域块的第 6 个字符为冒号，用于区分块名和块数据段。后 8 个字符显示该块的数值，显示的为 32 位 16 进制数，故占用 8 个字符。该段由 display_value 输入指定，display_value 输入的为 32 位 2 进制数，lcd_module 内部会自动转换为 8 个字符显示。

最后还有 display_valid 输入，用于指示是否需要在当前显示区域块(由 display_number)显示数据，为 1 有效。

在本例中，顶层模块 ram_display.v 调用 LCD 触摸屏显示功能的代码如下：

```
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----------------------//
```

可以看到 RAM 模块只需要区域块 1~3、5、6 用于显示，其他块(default 分支)的 display_valid 为 0。另外，display_name 接口赋值 5 位字符组成的字符串即可，字符串里应当只出现大写的 26 个字母，下划线"_"，0~9 数字或空格，其他字符在 LCD 触摸屏上暂不支持显示。而 display_value 直接赋值 32 位的数据即可。

3.3 调用 LCD 触摸屏输入

```
    output         input_valid,
    output [31:0] input_value,
```

input 的两个接口用于使用触摸屏的输入功能。当需要使用输入功能时，触摸屏底部的"START INPUTING"栏即可进入输入模式。点击屏小键盘上的 OK 键完成输入，会退出输入模式，同时 lcd_module 会拉高 input_valid 信号一拍，表示有数据要输出，而输出数据 input_value 会依据之前的输入确定，当输入不足 32 位时，会自动高位补 0，比如：只输入了 123，就按 OK 键，则最终 input_~~valid~~ 的值为 0x00000123。当输入有误时，可以按 BACK 键回退一格。 input_value

在本例中，顶层模块 ram_display.v 调用 LCD 触摸屏输入功能的代码如下：

```
 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
```

可以看到 RAM 模块需要使用触摸屏输入 3 个数：写数据、写地址和测试端口的地址。故需要 2 个拨码开关用于选择输入的数为哪个数。理论上，触摸屏加上外部的选择信号，可以给任意信号输入 32 位的数，比如可以输入一条 32 位的指令，可以输入内存的 32 位地址等等。当需要使用触摸屏输入多个数据时，建议每个数据单独用一个 always 块采集输入，这样的写法更接近电路实际情况，示例中的代码就是如此写的。

从 ram_display.v 源码中可以看到，本次实验使用到了实验箱上的 1 个时钟引脚、1 个复位键、4 个 led 灯、6 个拨码开关和 LCD 屏，如下图。（注意，拨码开关是拨上为 0，拨下为 1；LED 灯为接 0 亮，接 1 灭。）

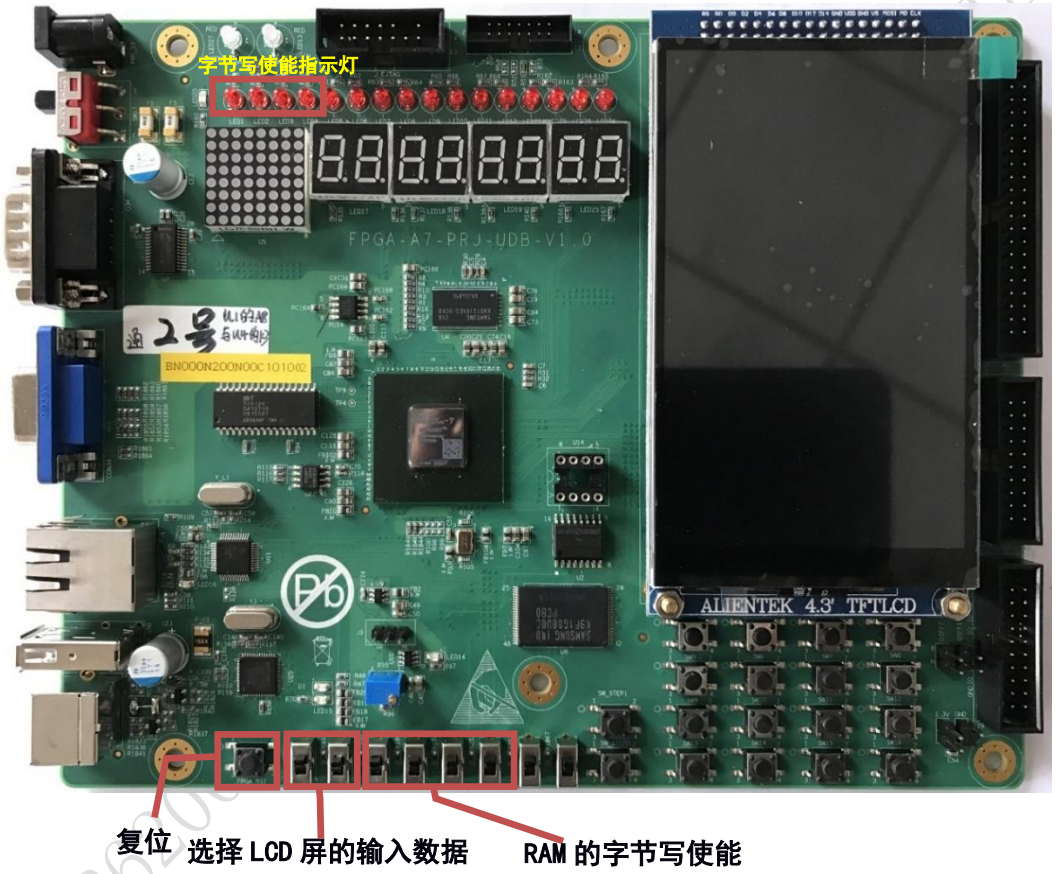

图 3-1 实验一使用到的实验箱上的 IO 引脚