计算机组成原理实验报告

人工智能与计算机学院　计科 1803 班　　学号：1033180311　　姓名：何元梅
实验日期：2020 年 12 月 03 日　同组同学：毛萍兰、孙橙橙、陈智瑞　指导老师：刘登峰

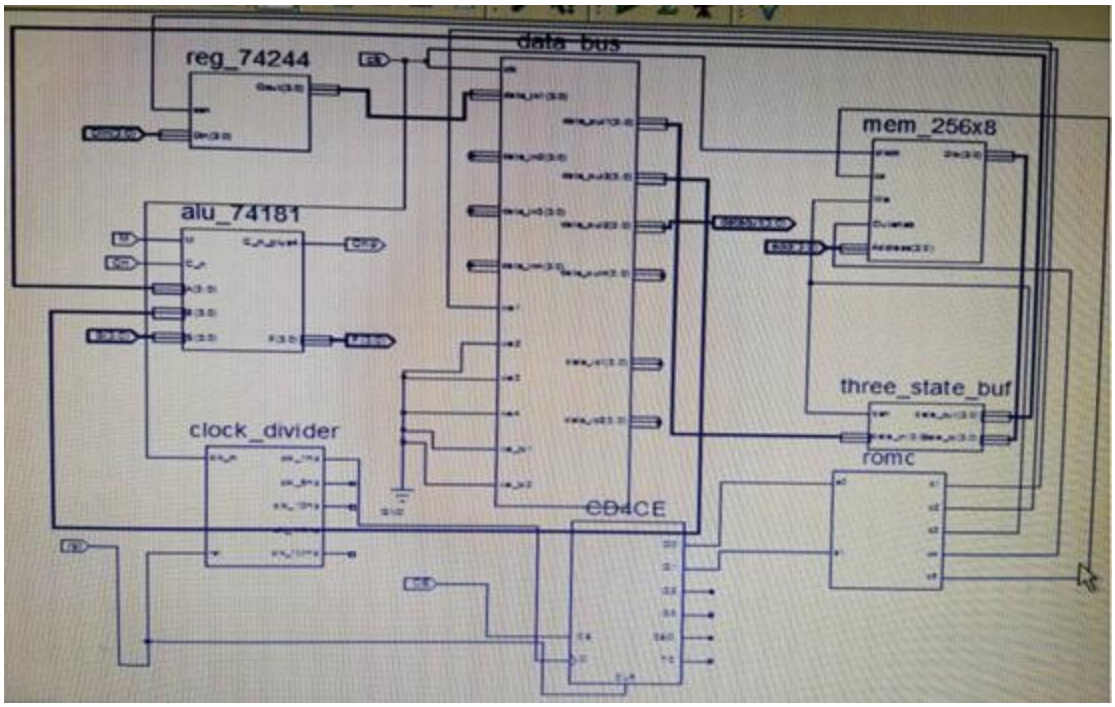
实验名称：用微程序控制器实现加法器

实验目的：

1. 掌握微程序控制器的工作原理。
2. 学会用微程序控制器对进行控制实现数据在总线上传输存入寄存器或存储器再进入加法器进行计算。

实验步骤

（一） 实验原理图



（二） 实验步骤

1. 建立工程文件，添加实验模块，完成原理图设计

（一）建立工程文件

（1）点击桌面 Xilinx ISE 软件

（2）选择 File/New Project，输入工程名为 sample

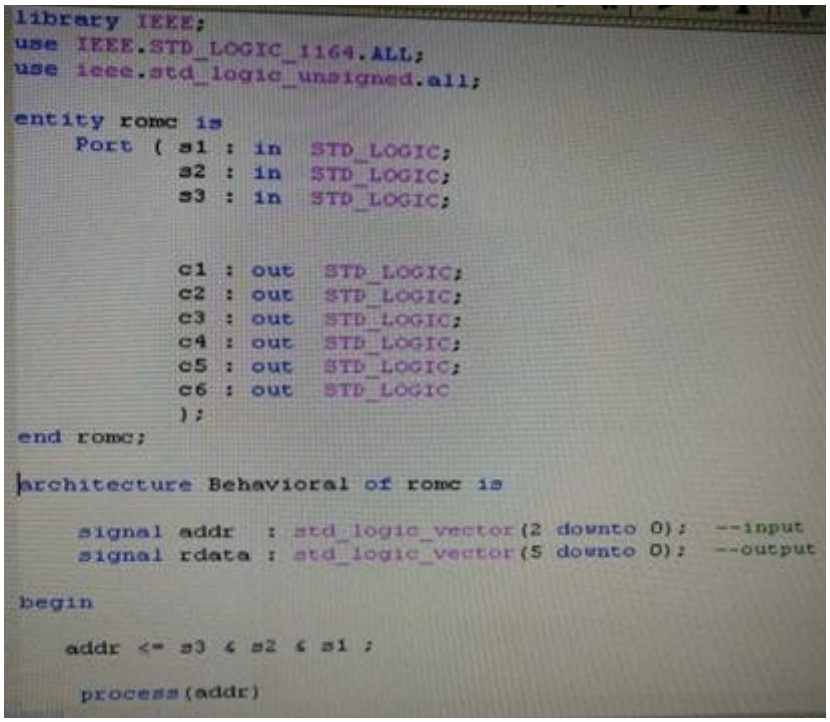
（3）在 Hierarchy 框中，右击鼠标，选择 New Source，选择 Schematic，输入文件名 test

（二）添加实验模块

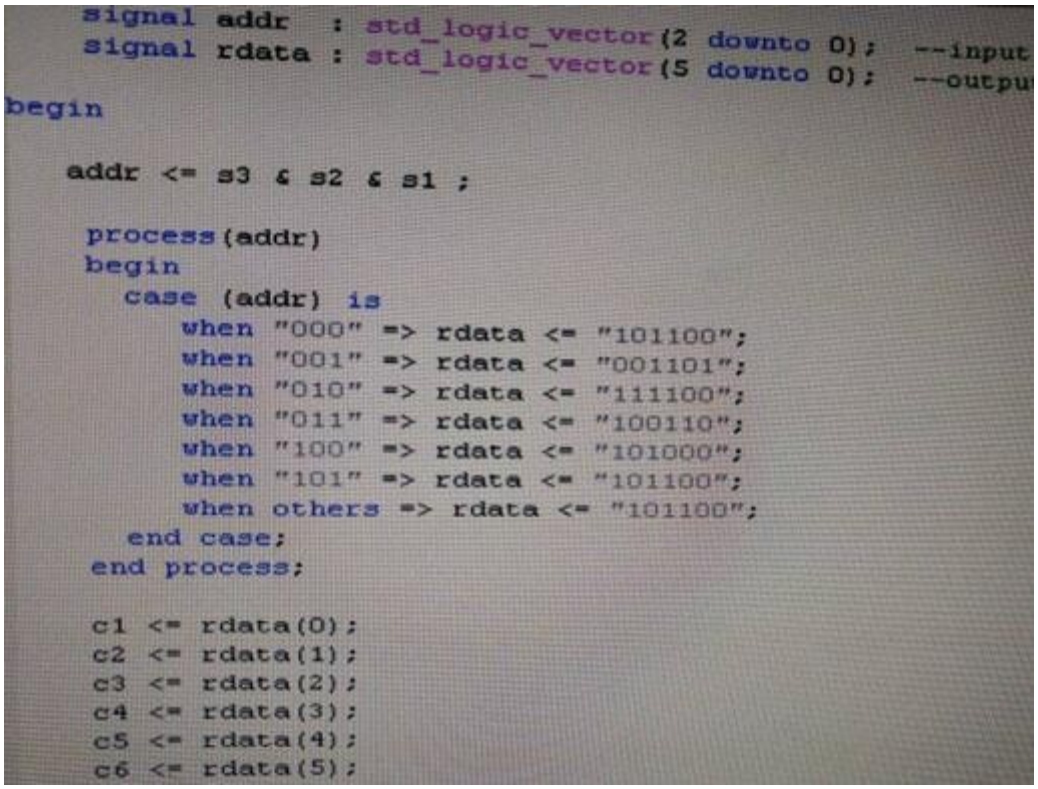
（1）在桌面左下方选择 Design 栏，在 Hierarchy 框中，右击鼠标，选择 Add Copy of Source

（2）在 D：/jan_lab_source 中，选择所用模块的．vhd 文件，点击打开

（3）再 Hierarchy 框中，双击各实验模块，打开控制器文件，根据实验中需要用到的实验步数，确定输入端口，根据实验原理图上所用的控制步数，确定输入端口，根据自己设计的实验原理图的所用的实验模块确定输入端口，修改控制器的 VHDL 语言程序，完成后点击保存。



```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;

entity romc is
    Port ( s1 : in  STD_LOGIC;
           s2 : in  STD_LOGIC;
           s3 : in  STD_LOGIC;

           c1 : out  STD_LOGIC;
           c2 : out  STD_LOGIC;
           c3 : out  STD_LOGIC;
           c4 : out  STD_LOGIC;
           c5 : out  STD_LOGIC;
           c6 : out  STD_LOGIC
           );
end romc;

architecture Behavioral of romc is

    signal addr  : std_logic_vector(2 downto 0);  --input
    signal rdata : std_logic_vector(5 downto 0);  --output

begin

   addr <= s3 & s2 & s1 ;

    process(addr)
```



```
    signal addr  : std_logic_vector(2 downto 0);  --input
    signal rdata : std_logic_vector(5 downto 0);  --outpu
begin

   addr <= s3 & s2 & s1 ;

    process(addr)
    begin
      case (addr) is
          when "000" => rdata <= "101100";
          when "001" => rdata <= "001101";
          when "010" => rdata <= "111100";
          when "011" => rdata <= "100110";
          when "100" => rdata <= "101000";
          when "101" => rdata <= "101100";
          when others => rdata <= "101100";
      end case;
    end process;

    c1 <= rdata(0);
    c2 <= rdata(1);
    c3 <= rdata(2);
    c4 <= rdata(3);
    c5 <= rdata(4);
    c6 <= rdata(5);
```

（4）在 Design 栏的 Processes 框中展开 Design Utilities,双击 Create Schematic Symbol 生成新的实验模块

（5）在桌面左下方选择 Symbols 栏，在 Symbols 框中，选择所用实验模块，点击拖动到桌面右面的原理图编辑框中

（三）原理图设计

（1）选择原理图编辑框左侧 Add I/O Marker，在实验模块的所用引脚端口建立端口符号

（2）右击所用端口符号，选择 Rename Port，选择 Rename the Branch，对端口符号进行命名

（3）选择原理图编辑框左侧 Add wire，可在实验模块间画线

（4）原理图设计完毕，点击保存

2. 修改用户约束文件，建立端口名与实验箱上拨动开关及 LED 灯对应联系，注意数据排列时的高低位顺序。

（四）修改用户约束文件

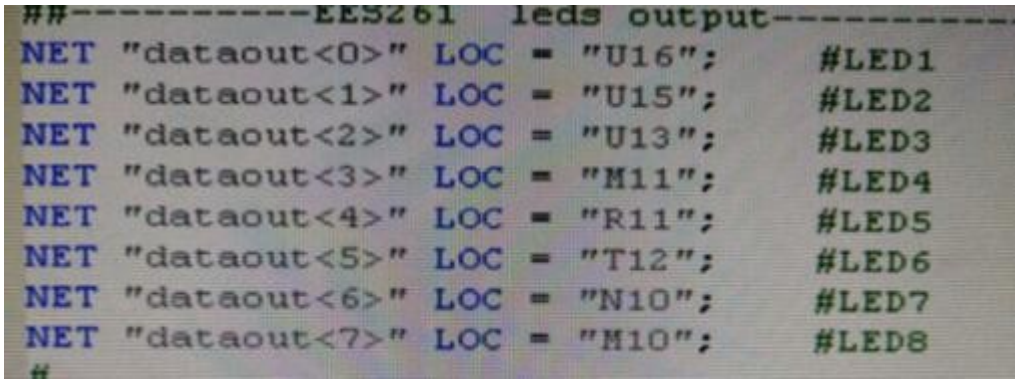
（1）在桌面左下方选择 Design 栏，在 Hierarchy 框中，点击鼠标，选择 Add Copy of Source

（2）在 D：/jan_lab_source 中选择 Myucf 文件，点击打开

（3）在 Hierarchy 框中，展开品字形符号栏，双击 Myucf

（4）用所命名的端口名修改 Myucf 文件中的语句，修改后程序如下所示：

```
###------------CLOCK-----------
NET "clk"    LOC = "L15";
#
###------------Atlys led output--------
NET "Qout[0]"    LOC = U18;   #Atlys LD0
NET "Qout[1]"    LOC = M14;   #Atlys LD1
NET "Qout[2]"    LOC = N14;   #Atlys LD2
NET "Qout[3]"    LOC = L14;   #Atlys LD3
NET "Qout[4]"    LOC = M13;   #Atlys LD4
NET "Qout[5]"    LOC = D4;    #Atlys LD5
NET "Qout[6]"    LOC = P16;   #Atlys LD6
NET "Qout[7]"    LOC = N12;   #Atlys LD7
#
###-----------Atlys Switch input--------
#NET "Qout[0]"     LOC = A10;   # Atlys sw
#NET "Qout[1]"     LOC = D14;   # Atlys sw
#NET "Qout[2]"     LOC = C14;   # Atlys sw
#NET "Qout[3]"     LOC = P15;   # Atlys sw
#NET "Qout[4]"     LOC = P12;   # Atlys sw
#NET "Qout[5]"     LOC = R5;    # Atlys sw
#NET "Qout[6]"     LOC = T5;    # Atlys sw
#NET "Qout[7]"     LOC = E4;    # Atlys sw
#
###------------EES261 switch input-------
NET "CE" LOC = "U11";     #SW20
NET "rst" LOC = "R10";     #SW19
#NET "swt[17]" LOC = "U10";    #SW18
#NET "swt[16]" LOC = "R8";    #SW17
#
```



（5）修改完毕，点击保存

3. 编译，下载

（五）编译

（1）在桌面左下方选择 Design 栏，在 Hierarchy 框中，选中所建立的 .Sch 文件

（2）在 Processes 框中，双击 Generate Programing File，观察编译后的提示信息

（六）下载

（1）打开实验箱电源

（2）在桌面上选择 开始 / 程序 / Digilent / Adept / Adept

（3）点击 Browse，选择 C : / Documents and Settings / 工程名 / 文件名.bit，点击打开

（4）点击 Program，完成下载

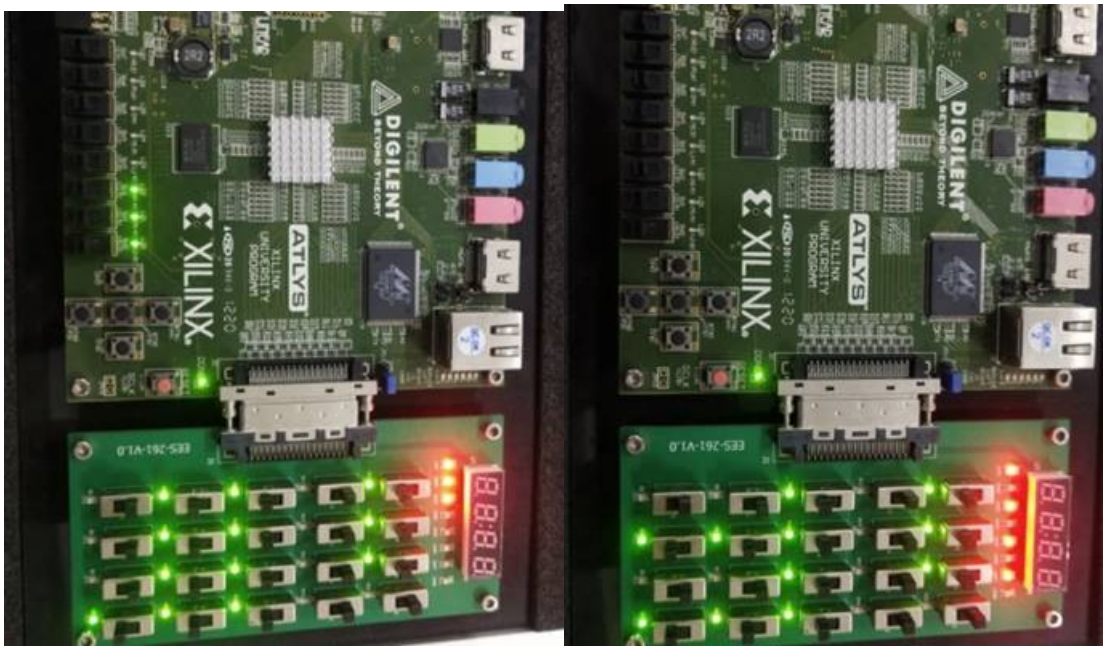
4.设定输入数据，操作每个实验模块的控制端开关，是数据在总线上进行传输，注意向总线传输数据时，一次只允许一个实验模块输出，因此在操作时应先将其它实验模块的控制端设在无效状态。

5. 将两个不同的数据分别写入二个 74373。

5.将两个 74373 的数据进行交换，然后读出验证是否正确

实验结果

下载完成后实验箱照片如下所示：



注释：以上实验截图及结果均为手动控制总线传输实验改写成用微程序控制的总线传输实验。

附加题的实验结果及分析如下：

（三） 实验感想

实验解决方法分析：

我们小组在上节课将微程序控制器控制数据在总线上传输的实验做出来了，做这个实验得原理和那个类似，就是器件变多了。根据数据从输入到在总线上传输，进入存储器，在进入到加法器，需要控制的参与器件输入端口的各阶段的控制状态确定步数，从而推理出 romc 的输入与输出端口的数目，然后按照

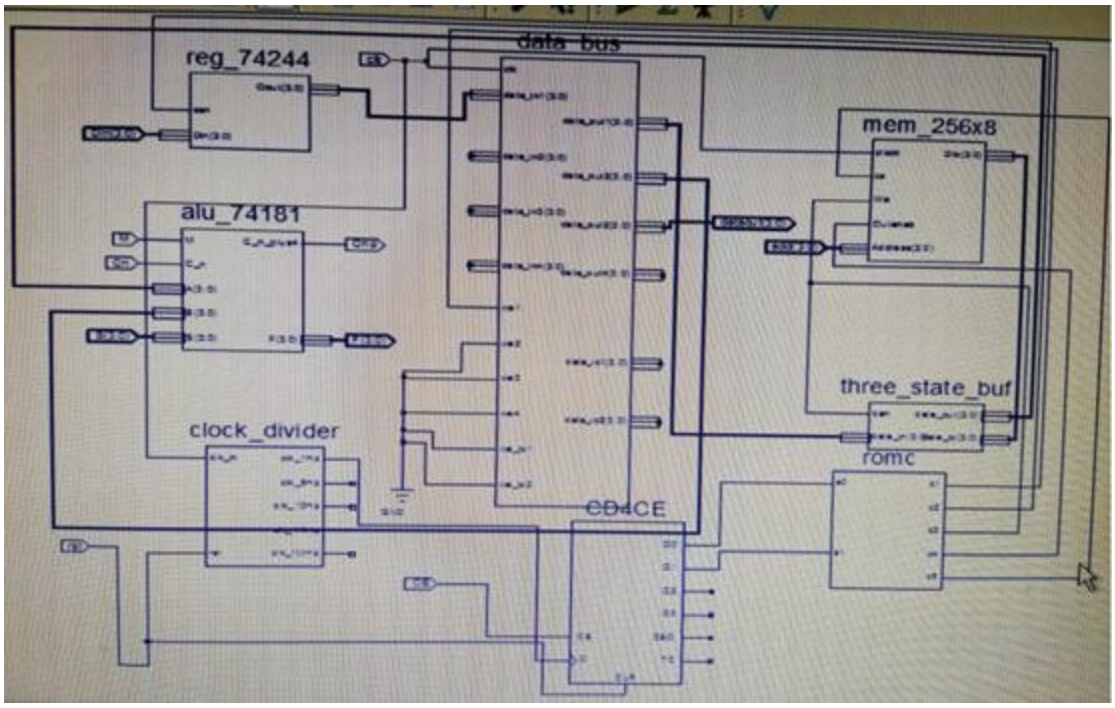
设计好的电路图进行连线即可。我们电路的设计思路是先由 reg_74244 输入数据并写入总线，然后从总线传输到 mem_256x8+three_state_buf 构成的存储器里面，然后从存储器输出到 alu_74181 加法器进行运算，romc 就控制 reg_74244 的 oen、，data_bus 的 we1、mem_256X8 的 cs、we 和 Outenab,所以 romc 有五个输出端口，各端口的状态表如下：

控制端口 步骤	Oen（低有效）	we1（高有效）	cs（高有效）	we（高有效）	outenab（高有效）
写入数据 1 并传到总线和存储器	0	1	1	1	0
数据 1 传到 alu_74181 的 A 输入端口	0	0	1	0	1
写入数据 2 并传到总线和存储器	0	1	1	1	0
数据 2 传	0	0	1	0	0

到 alu_74181 的 B 输入端口					

所以控制步数为 4。

原理图如下：



实现情况分析：

这个实验最后我们没有做出来，因为要实现四位的加法器，有很多控制器件的 VHDL 程序语言需要修改，连接好线路之后有很多的发现有些器件的修改错了，然后就不得不删除掉重新再来一次，来来回回折腾了好几次，最后 mem_256x8+three_state_buf 构成的存储器那块的出了些不知名的问题，导致下载编译的程序总是出现错误，时间转眼过了大半，已经快要四点了，于是我们去询问有些完成了该实验的同学，他们竟然没有用

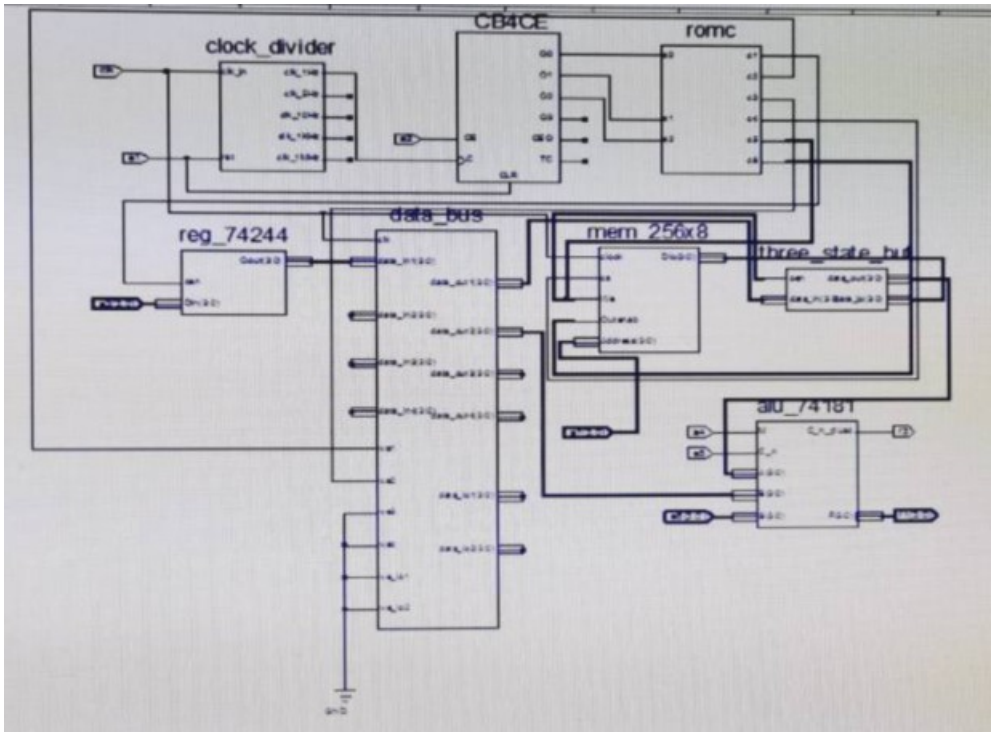
mem_256x8+three_state_buf 构成的存储器这个实验，他们两个数据居然是用两个 reg_74373 寄存器，我们理解有点偏差，我们以为要用原来的之前存储器的那个实验的模块构成，但是按道理我们这个应该也可以，就是地址控制有点麻烦，还得用几灯来控制。理论上没啥毛病。那么按照后一种方法应该是这样的：

由 reg_74244 输入数据并写入总线，然后从总线传输到 reg_74373 寄存器中，然后从两个寄存器读出数据输入到 aul_74181 的 A、B 端口进行运算操作，计算的结果可通过另一个 reg_74373 存储然后送到总线上输出。Romc 的输出端需要控制 reg_74244 的 oen、data_bus 的 we1、第一个 reg_74373 的 gwe1、cen_n1、第二个 reg_74373 的 gwe2、cen_n2 端口，所以 romc 有六个输出端口，各端口的状态表如下：

控制端口 步骤	Oen	We1	We2	Gwe1	Cen_n1	Gwe2	Cen_n2	Gwe3	Cen_n3
写入数据 1 并传到总线和写入存储器 1	0	1	0	1	1	0	1	0	1
写入数据 2 并传到总线和写入存储器 2	0	1	0	0	1	1	1	0	1
将两个数据传入到 alu 的 A、B 端	1	0	0	0	0	0	0	0	1
将 alu 的计算结果传入存储器 3 并写	1	0	1	0	1	0	1	1	1

入总线输出									

最后我们重做了一下。可惜没时间了，现连接的半成品的原理图如下：



总结：这次实验不算很难，原理也基本理清楚了，就是做实验的用错了实验模块，增加实验难度，而且做了很久之后才发现，最后就只出了一个半成品的实验电路图，就这吧，尽力了。