基本CPU设计实验报告

计算机系统结构

姓名：沈思远　　学号：915106840433

姓名：孙维华　　学号：915106840434

姓名：吴宗泽　　学号：915106840439

指导老师：杜姗姗

2018年3月

一、实验目的

利用已有的计算机组成原理知识，以及对计算机系统结构的初步学习，设计一个包括指令系统、运算器、控制器和寄存器组等的完整CPU。

熟练掌握VHDL硬件描述语言，学会利用硬件设计工具软件对程序进行仿真和调试，并熟练掌握FPGA在开放式实验教学系统上的调试方法。

二、实验内容

完成一个完整CPU系统的设计，至少包含六条指令，传送类，运算类和跳转类都要有，并调试成功。

三、实验设备

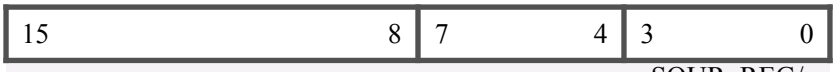
TEC-CA开放式CPU实验教学系统。

四、实验原理

1、指令系统设计

（1）指令格式分类

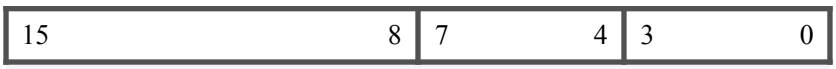
① 单字节指令



OPCODE	DEST_REG	SOUR_REG/ OFFSET

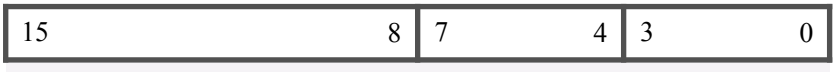
指令包括：DEC, INC, SHL, JRC, JRNC等。

② 双字节指令



OPCODE	DEST_REG	SOUR_REG

指令包括：ADD,SUB,AND,CMP,XOR,TEST,OR等。



OPCODE	DEST_REG	0000
DATA		

指令包括：MVRD等。

（2）指令的分组及节拍：

由于没有中断操作，本机指令的执行步骤可概括如下：

读取指令：地址寄存器<–指令地址，修改PC内容使其指向下一条将要执行的指令；

读内存，指令寄存器<–读出的内容。

分析指令

执行指令：通用寄存器之间的运算或传送，可1步完成；

读写内存，通常要两步完成。

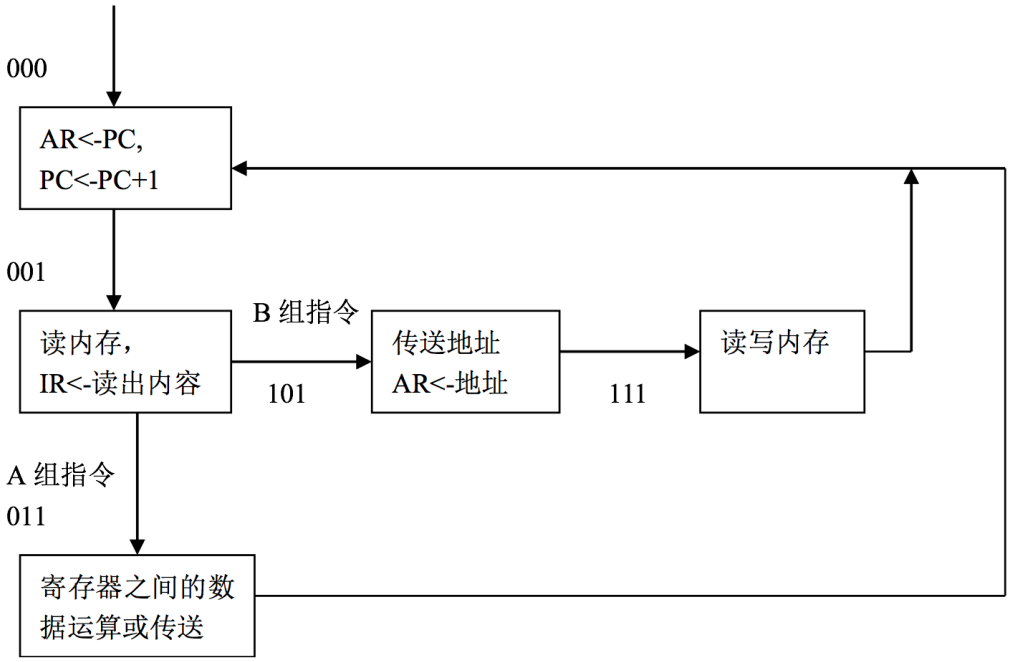
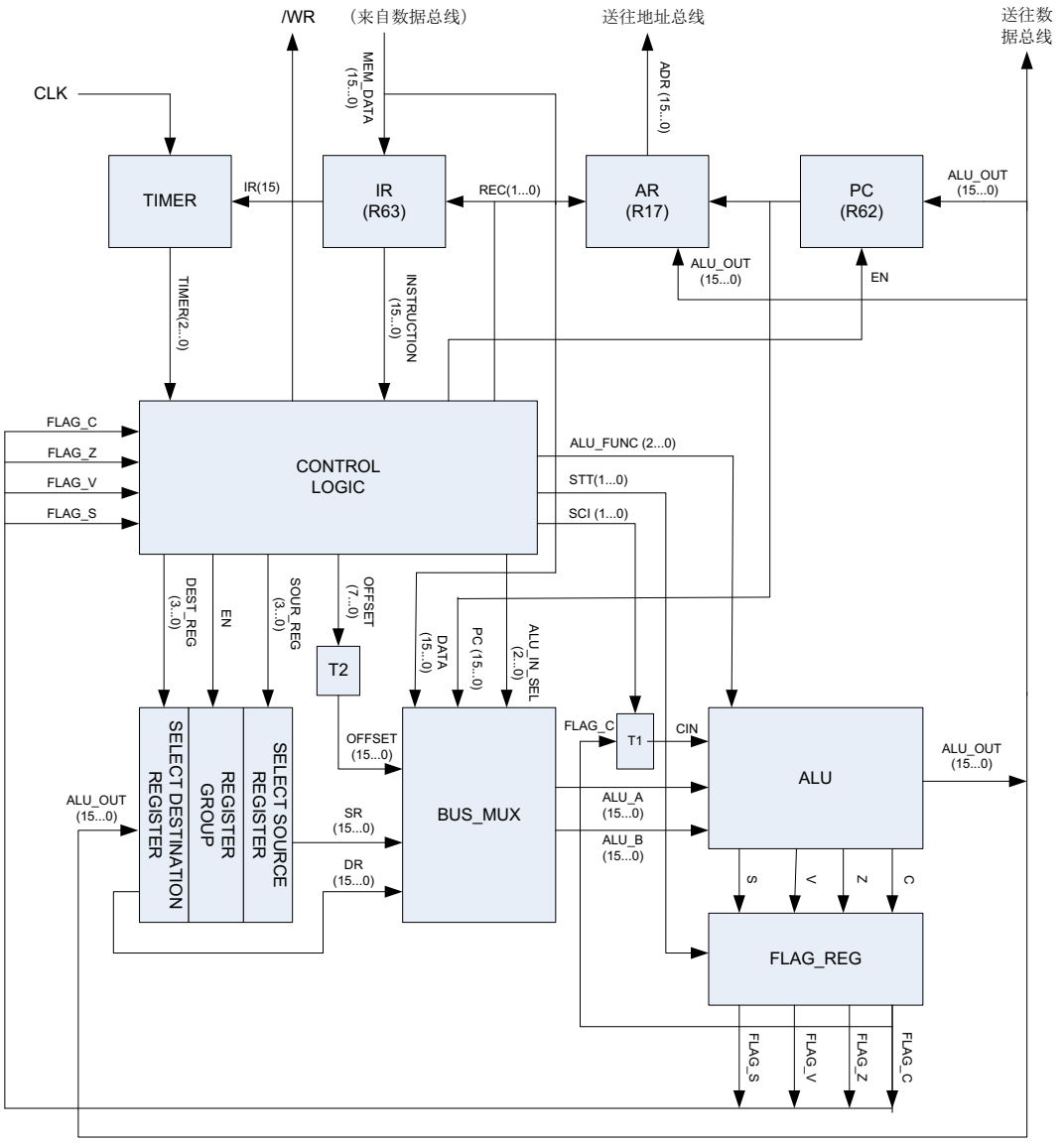


图1 控制器节拍示意图

2、结构图：



注：图中所有寄存器都与时钟信号相连，并当RESET信号到来时清零。

图2 CPU总体结构设计图

主要的部件就是算术逻辑单元ALU、控制逻辑、通用寄存器组、指令寄存器IR、地址寄存器AR、程序计数器PC、标志寄存器、节拍发生器以及一些数据选择器和译码电路。

3、组件

1、ALU

ALU的行为描述如下表所示：

I	功能
000	A+B+Cin
001	A-B-Cin
010	A与B

在这里我们发现源文件的代码逻辑比较复杂，进行了改写和简化，但由于后面没有连出电路图，改写的正确性不得而知。

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity alu is
port(cin:in std_logic;
     alu_a,alu_b:in std_logic_vector(15 downto 0);
     alu_func:in std_logic_vector(2 downto 0);
     alu_out:out std_logic_vector(15 downto 0);
     c,z,v,s:out std_logic);
end alu;

architecture behave of alu is
begin
     process(alu_a,alu_b,cin,alu_func)
     variable temp1,temp2,temp3 : std_logic_vector(15 downto 0) ;
     begin
            temp1 := "000000000000000"&cin;
            case alu_func is
                   when "000"=>
                   temp2 := alu_b+alu_a+temp1;

                   when "001"=>
                   temp2 := alu_b-alu_a-temp1;

                   when "010"=>
                   for I in 14 downto 0 loop
                   temp2(I+1):=alu_b(I);
                   end loop;
                   temp2(0):='0';

                   when others=>
                   temp2 := "0000000000000000";
            end case;
```

```
            alu_out <= temp2;
            if temp2 = "0000000000000000" then z<='1';
            else z<='0';
            end if;
            if temp2(15) = '1' then s<='1';
            else s<='0';
            end if;
            case alu_func is
                   when "000"=>
                   if (alu_a(15)= '1' and alu_b(15)= '1' and temp2(15) = '0') or
                      (alu_a(15)= '0' and alu_b(15)= '0' and temp2(15) = '1') then
                   v<='1';
                   else v<='0';
                   end if;
                   temp3 := "1111111111111111"-alu_b-temp1;
                   if temp3<alu_a then
                   c<='1';
                   else c<='0';
                   end if;

                   when "001"=>
                   if (alu_a(15)= '1' and alu_b(15)= '1' and temp2(15) = '0') or
                      (alu_a(15)= '0' and alu_b(15)= '0' and temp2(15) = '1') then
                   v<='1';
                   else v<='0';
                   end if;
                   if (alu_b<alu_a) then
                   c<='1';
                   else c<='0';
                   end if;

                   when "010"=>
                   c<=alu_b(15);
                   v<='0';

                   when others=>
                   v<='0';
            end case;
     end process;
end behave;
```

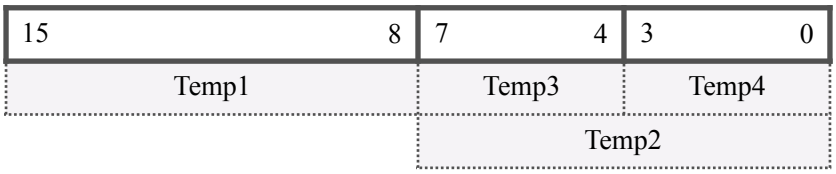
2、控制逻辑Controller

Controller根据IR指令内容和指令的执行步骤及标志寄存器的信号，形成并提供出计算机各部件当前时刻要用到的控制信号。

代码的约定如下：

字段	功能
c,z,v,s	标志位
offset	偏移量
dest_reg	目的寄存器
sour_reg	源寄存器
sst, sci, rec	控制信号
instruction	指令内容

Temp变量描述如下：



在本VHDL文件中，描述了ADD，SUB，JRC，MVRR，INC，SHL等规则。具体见代码注释处。

```
library ieee;
use ieee.std_logic_1164.all;

entity controller is
port(timer:                  in std_logic_vector(2 downto 0);
     instruction:            in std_logic_vector(15 downto 0);
     c,z,v,s:                in std_logic;
     dest_reg,sour_reg:      out std_logic_vector(3 downto 0);
     offset:                 out std_logic_vector(7 downto 0);
     sst,sci,rec:            out std_logic_vector(1 downto 0);
     alu_func,alu_in_sel:    out std_logic_vector(2 downto 0);
     en_reg,en_pc,wr:        out std_logic);
end controller;

architecture behave of controller is
begin
     process(timer,instruction,c,z,v,s)
     variable temp1,temp2 : std_logic_vector(7 downto 0) ;
     variable temp3,temp4 : std_logic_vector(3 downto 0) ;
     variable alu_out_sel: std_logic_vector(1 downto 0);
     begin
         for I in 7 downto 0 loop
                temp1(I):=instruction(I+8);
                temp2(I):=instruction(I);
         end loop;
         for I in 3 downto 0 loop
```

```
            temp3(I):=instruction(I+4);
            temp4(I):=instruction(I);
        end loop;
           case timer is
               when "100"=>
                         dest_reg<="0000";
                         sour_reg<="0000";
                         offset<="00000000";
                         sci<="00";
                         sst<="11";
                         alu_out_sel:="00";
                         alu_in_sel<="000";
                         alu_func<="000";
                         wr<='1';
                         rec<="00";
                  when "000"=>
                         dest_reg<="0000";
                         sour_reg<="0000";
                         offset<="00000000";
                         sci<="01";
                         sst<="11";
                         alu_out_sel:="10";
                         alu_in_sel<="100";
                         alu_func<="000";
                         wr<='1';
                         rec<="01";
                  when "001"=>
                         dest_reg<="0000";
                         sour_reg<="0000";
                         offset<="00000000";
                         sci<="00";
                         sst<="11";
                         alu_out_sel:="00";
                         alu_in_sel<="000";
                         alu_func<="000";
                         wr<='1';
                         rec<="10";
                  when "011"=>
                         wr<='1';
                         rec<="00";
                         case temp1 is
                                when "00000000"=> --ADD
                                dest_reg<=temp3;
                                sour_reg<=temp4;
                                offset<="00000000";
                                sci<="00";
                                sst<="00";
                                alu_out_sel:="01";
                                alu_in_sel<="000";
                                alu_func<="000";
```

```
when "00000001"=> --SUB
dest_reg<=temp3;
sour_reg<=temp4;
offset<="00000000";
sci<="00";
sst<="00";
alu_out_sel:="01";
alu_in_sel<="000";
alu_func<="001";

when "01000000"=> --JRC
dest_reg<="0000";
sour_reg<="0000";
offset<=temp2;
sci<="00";
sst<="11";
alu_out_sel:=c&"0";
alu_in_sel<="011";
alu_func<="000";

when "00000111"=> --MVRR
dest_reg<=temp3;
sour_reg<=temp4;
offset<="00000000";
sci<="00";
sst<="11";
alu_out_sel:="01";
alu_in_sel<="001";
alu_func<="000";

when "00001001"=> --INC
dest_reg<=temp3;
sour_reg<=temp4;
offset<="00000000";
sci<="01";
sst<="00";
alu_out_sel:="01";
alu_in_sel<="010";
alu_func<="000";

when "00001010"=> --SHL
dest_reg<=temp3;
sour_reg<=temp4;
offset<="00000000";
sci<="00";
sst<="00";
alu_out_sel:="01";
alu_in_sel<="010";
alu_func<="010";

when others=>
```

```
                                null;
                        end case;
                when "101"=>
                        alu_func<="000";
                        wr<='1';
                        sst<="11";
                        dest_reg<=temp3;
                        sour_reg<=temp4;
                        offset<="00000000";
                        case temp1 is
                                when "10000000" | "10000001"=>
                                sci<="01";
                                alu_out_sel:="10";
                                alu_in_sel<="100";
                                rec<="01";
                                when others=>
                                null;
                        end case;
                when "111"=>
                        dest_reg<=temp3;
                        sour_reg<=temp4;
                        offset<="00000000";
                        sci<="00";
                        sst<="11";
                        alu_func<="000";
                        rec<="00";
                        case temp1 is
                                when "10000001"=>
                                alu_out_sel:="01";
                                alu_in_sel<="101";
                                wr<='1';
                                when "10000000"=>
                                alu_out_sel:="10";
                                alu_in_sel<="101";
                                wr<='1';
                                when others=>
                                null;
                        end case;
                when others=>
                null;
            end case;
            en_reg<=alu_out_sel(0);
            en_pc<=alu_out_sel(1);
     end process;
end behave;
```

3、其他组件

部件类型	名称	作用
组合逻辑部件	数据选择器BUS_MUX	通过三位控制信号控制ALU模块A、B端的输入。可选择的输入有源寄存器数据，目标寄存器数据，带符号位扩展的偏移地址，PC，以及从内存读取的立即数、跳转地址等数据。
组合逻辑部件	T1	两位控制信号选择产生ALU的进位输入CIN
时序逻辑部件	标志寄存器FLAG_REG	接收ALU的标志位输出，在控制信号SST的控制下输出实际需要的标志位
组合逻辑部件	T2	将8位OFFSET(来自指令的低8位)带符号位拓展到16位
时序逻辑部件	程序计数器PC	在控制信号pc_en的控制下接收ALU的运算结果，结果送往地址寄存器或者数据选择器
时序逻辑部件	地址寄存器AR/指令寄存器IR	AR存放要读写的内存地址单元，IR存放当前执行指令的内容。两者共用REC两位信号控制
组合逻辑部件	寄存器选择器REG_MUX	输出选定寄存器的内容以及对应寄存器的写使能信号
时序逻辑部件	节拍发生器	用多位触发器的输出信号的不同组合状态标识每条指令的执行步骤
组合逻辑部件	控制器CONTROLLER	根据指令内容(IR)和指令的执行步骤以及其他条件信号形成并提供计算机各部件的控制信号
组合逻辑部件	T3	双向门，解决数据总线的冲突问题，读时高阻，写时输出ALU运算结果
组合逻辑部件	REG_OUT	对外输出寄存器内容的译码电路。根据外部输入的选择信号输出指定寄存器内容

五、实验步骤

1、打开Quartus Ⅱ，选择File→New Project Wizard，输入目标路径，工程名及顶层设计实体名，区分大小写。

2、输入工程中包含的设计文件

3、确定设计使用的器件，Cyclone →EP1C12Q240C8

4、选择EDA工具：综合、仿真和时序分析

5、检查工程中的各项设置

6、建立新文件。File →New，选择VHDL，输入程序，保存

7、File →Creat/Update →Creat Symbol Files For Current File，产生一个类型为电原理图的新文件。

8、完成原理图绘制。

9、分配引脚。执行Assignments →Pins命令，启动分配引脚功能.

10、编译源文件，Processing →Start Compilation

11、编写规则文件

12、编写测试程序

13、将模式开关REGSEL,CLKSEL,FDSEL设置为101，即单片机控制FPGA-CPU调试运行模式

14、用下载电缆将PC机和CPLD的下载电路连接起来，执行Tool→Programmer命令，在框中选择默认的JTAG下载方式，Add File将生成的sof文件添加进来，启动Start按钮完成下载

15、打开Debugcontroller软件，按单片机复位按钮，打开规则文件，打开测试程序

16、执行Build→Compile Code命令，将汇编文件译为十六进制文件

17、执行Build→Upload BIN命令，将十六进制代码文件传送到存储器中

18、开始调试，执行Debug→Begin Debug命令

19、使用Build→Download RAM命令，可以查看实验台上存储器的内容。

20、执行Debug→End Debug命令结束调试

六、引脚规定

存储器地址线	引脚	实验台上的指示灯
A0	41	A0
A7	48	A7
A8	57	A8
A15	64	A15

存储器数据线	引脚	实验台上的指示灯
D0	200	D0
D3	203	D3
D4	214	D4

存储器数据线	引脚	实验台上的指示灯
D7	217	D7
D8	223	D8
D11	226	D11
D12	234	D12
D16	237	D16

标志寄存器数据	引脚	实验台上的指示灯
C	86	C
Z	85	Z
V	84	V
S	83	S

寄存器地址	引脚	实验台上的指示灯
REGSEL0	12	RS0
REGSEL5	17	RS5

寄存器数据	引脚	实验台上的指示灯
REG0	158	R0
REG7	165	R7
REG8	173	R8
REG10	175	R10
REG11	177	R11
REG16	181	R16

七、测试报告

根据规定的逻辑，写出的规则如下：

```
SUB dr,sr "00000001[u4][u4]",dr,sr
ADD dr,sr "00000000[u4][u4]",dr,sr
INC dr "00001001[u4]xxxx",dr
JRC addr "01000100[8]",addr-@-1
MVRD dr,data "10000001[u4]xxxx[u16]",dr,data
JMPA addr "1000000000000000[u16]",addr
R0 = 0
R1 = 1
```

写出以下汇编代码进行功能测试：

```
MAIN: MVRD R0,11
      MVRD R1,23
      ADD R0,R1
      INC R1
      SUB R0,R1
T1:   MVRD R1,0xFFFF
      ADD R0,R1 ;创造进位
      JRC MAIN ;跳转
      MVRD R0,0
```

首次测试过程中，发现MVRD的规则有误。改正后，汇编代码能够正确运行。可能是由于没有流水线的设计，或是教学器件的原因，代码的执行较慢，有一定延迟。

八、小结

本次实验的难点主要在：

1、结构图较复杂，连线耗时；

2、VHDL知识空白，需要在短时间内学会改写代码。

3、结构图信息不全，比如输入输出(clk等)的接法，还有寄存器组互相的连线，自己很难领会，需要一些参考材料。

我们小组互相讨论，克服了以上的大部分难点，这让我们感到兴奋和满足。但由于时间有限，未能将连线全部完成，这也是这次实验的遗憾。