实 验 报 告

课程名称 CPU 设计

实验题目 模型计算机功能模块设计

指导教师

实验日期 2014.06.09~2014.06.30

系 别 计算机科学与技术

专 业 计算机科学与技术

班 级

姓名/学号

姓名/学号

姓名/学号

成 绩

模型计算机功能模块设计

——微程序控制器逻辑

一、实验目的

综合运用 QUARTUSⅡ图形编辑器、文本编辑器、仿真器等设计输入与仿真手段，通过本实验的设计输入与设计处理，充分应用层次化、模块化设计方法，掌握自顶向下、自底向上及混合式设计方法及图形、文本文件的编译、适配和仿真操作，并通过选定 PLD 芯片、指定引脚、编程（下载），完成**微程序控制器逻辑**的全部设计与实现。

二、实验器材

微机系统，QUARTUSⅡ 6.0 开发工具软件，计算机组成原理实验装置。

三、实验要求

1. 提前预习、实验前作好设计准备，合理有效安排每次实验应完成的内容，由实验指导老师登记备案。
2. 最后一次实验时，应已完成所有实验内容，并演示说明设计要点及结果，并提交本综合实验报告。

四、实验内容

1. 图形设计输入
   （1）**以《计算机组成原理实验指导教程》（简为“教程”）为参考**，在理解微程序控制器工作原理的基础上，参照教程中 P41 页图 2-10、P44 页图 2-11、P46 页图 2-12 的逻辑描述，对微程序控制器（计算机组成原理实验装置的 C 板）进行功能模块划分；
   （2）参照上述各功能模块的逻辑描述，建立各功能模块的图形输入模块文件，并编译通过；
2. 功能仿真设计
   （1）分别对各功能模块进行功能仿真，确定各功能模块的输入输出逻辑关系；
   （2）必要时，可以重新划分功能模块，以形成相对功能完整的逻辑关系和仿真结果；
3. VHDL 文本设计
   （1）对各功能模块的逻辑关系用 VHDL 进行文本设计描述，建立各功能模块的 VHDL 文本文件，并编译通过；
   （2）分别对各 VHDL 功能模块进行功能仿真，验证功能符合预期；
4. 顶层文件设计
   （1）分别用图形和文本设计方法，建立顶层设计文件的两种形式；
   （2）分别对两种形式的顶层文件进行功能仿真，确定设计无误，建立顶层设计输入输出逻辑关系。
5. 编程下载
   （1）指定芯片和引脚
   （2）编程下载

6. 验证

将存有微程序的 E^2PROM 芯片插入新 C 板，与 A 板和 B 板相连接，通过运行一个相对完整的程序，对比原 C 板同样的程序运行结果，判断、验证所设计的结果是否正确。

五、实验设计

（一） 总体功能理解的叙述和功能模块划分设计说明

本次实验主要实现了模型计算机 CPU 微程序控制器的相关功能。包括两个项目：CPLD1 主要实现了微指令译码的工作CPLD2 主要实现了通用寄存器的控制信号的生成的工作。CPLD1 和CPLD2 分别对应两个 MAX II EPM240T100C5 芯片，所以对应两张原理图：如图 4-1、图 4-2 和图 4-3：

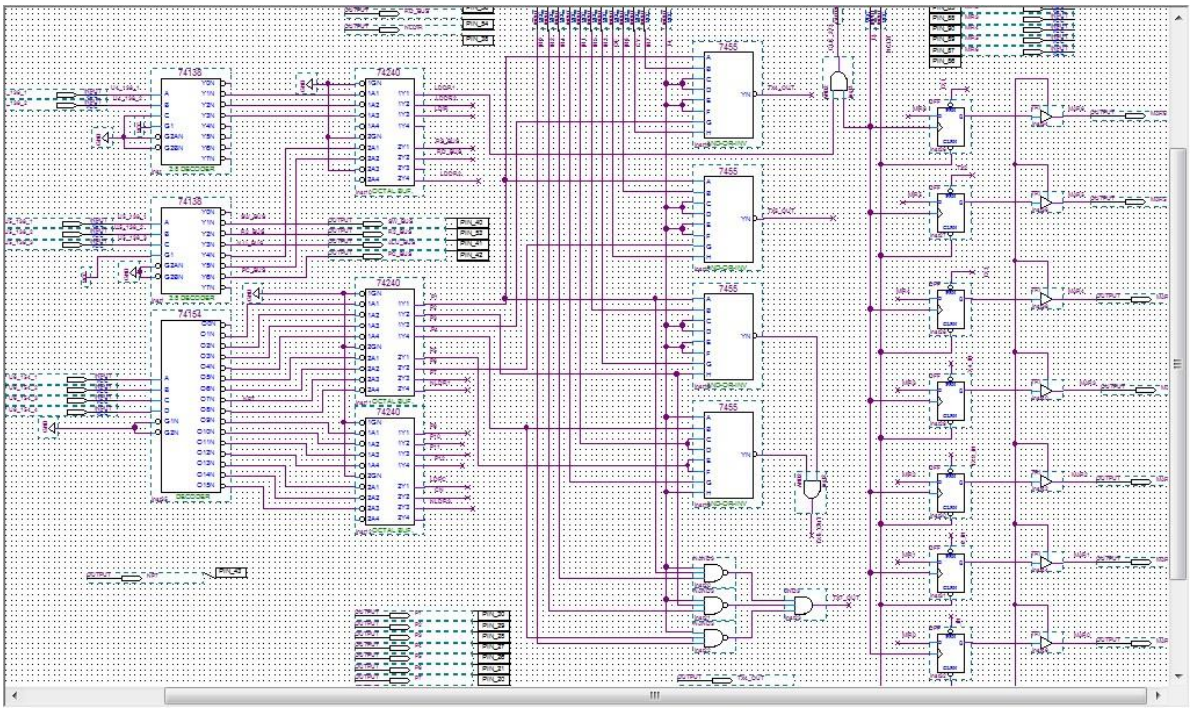

图 4-1

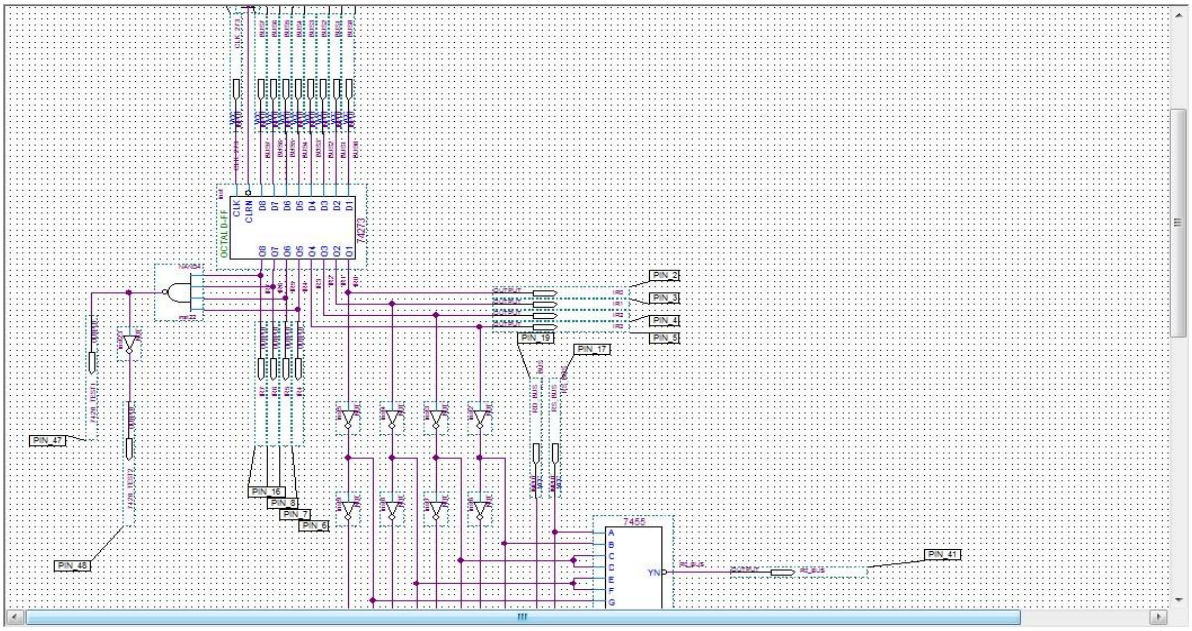

图 4-2

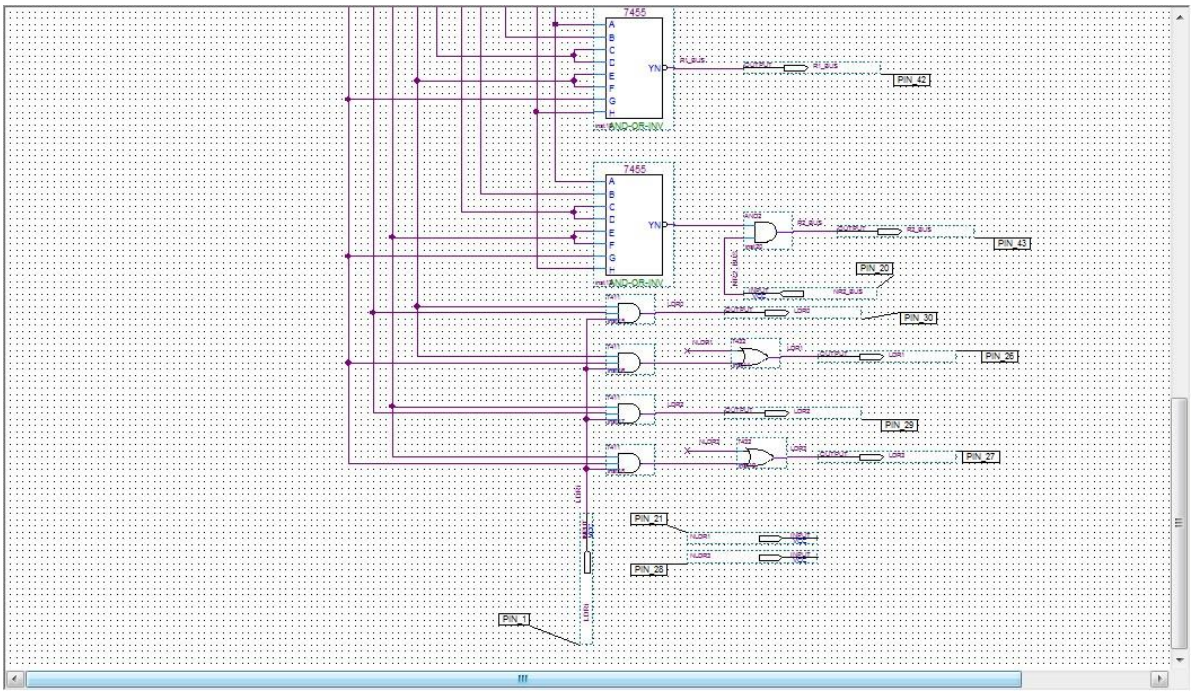

图 4-3

计算机的 CPU 包括两大部分——运算器和控制器。控制器的作用是将指令码从存储器中读出，并将该指令码翻译成为机器能够识别、安排操作的时序，并且向计算机的相关部件发出一系列执行该指令所需要的微操作控制信号，从而使得指令得到执行。所以是控制器是计算机有序工作的关键部件。

微程序控制器主要有控制存储器(CM)、微指令寄存器(MIR)、微地址寄存器(MAR)和地址转移逻辑等部分组成。其中控制存储器(CM)和微指令寄存器(MIR)在新的 C 板上已经使用相关的硬件器件做好了，不需要我们再去做了。我们只需从微指令寄存器(MIR)中引出相关的信号经过译码电路产生控制信号，并且将下一条指令的地址输出。相关的寄存器的控制也是通过指令的不同字段进行控制的。所以从 IR 接收到指令后，对其进行译码。最后通用寄存器输出控制信号。具体流程如图 4-3 所示。

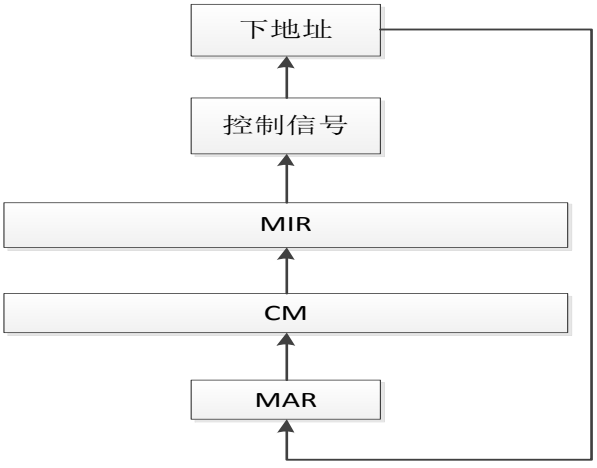

图 4-3

（二） 控制信号模块设计

控制信号模块实现了控制器中对数据传送信号或是判别测试字段的相关信号的生成。由于本

模型机字长为 32 位，其编码方式为直接表示和字段直接译码相结合的混合编码方式。这样大大节约了指令字的长度。在 32 位微指令中，23~22 位为 A 字段、21~19 位为 B 字段 11~8 位为 P 字段。直接表示的信号已经在 C 板中直接从 74273 芯片接出来信号。而需要译码的 A、B、P 字段则是我们的主要工作。在获取到 23~22 位后，在芯片中设置的节点名为 U4_138_1 和 U4_138_2 分别代表了第 23 位和第 22 位。他们产生的 A 字段的编码信号为如下表 4-1 所示。

U4_138_1(23)	U4_138_2(22)	选择
0	0	NOP
0	1	LDDR1
1	0	LDDR2
1	1	LDIR

表 4-1

这样我们可以直接使用 74138 译码器，将其组合所代表的控制信号输出。可以使用 7139 芯片，但由于 74139 芯片是双 2 线-4 线译码器，会造成资源的浪费，故我们使用 74138 的两个输入端口即可。由于 LDDR1、LDDR2 和 LDIR 都是高电平有效的控制信号，而经过 74138 输出后的信号为低电平。所以使用 74240 反向缓冲器进行反向。同理我们使用 74138 译码器对 B 字段控制信号进行译码表示。其中 21、20、19 位的接入信号为 U5_138_1、U5_138_2 和 U5_138_13。B 字段的编码如下表 4-2 所示：

U5_138_1(21)	U5_138_1(20)	U5_138_1(19)	选择
0	0	0	NOP
0	0	1	SW→BUS
0	1	0	R_2→BUS
0	1	1	ALU→BUS
1	0	0	R_s→BUS
1	0	1	R_d→BUS
1	1	0	PC→BUS

表 4-2

要注意的是 B 字段中并非所有的信号都是低电平有效，R_s→BUS 和 R_d→BUS 为高电平有效。所以将他们接在 74240 上进行反向输出。接着 P 字段总共有 16 中信号，由 4 为组成，所以我使用了 74154 4 线-16 线译码器。74154 的输入信号名为 U6_154_1、U6_154_2、U6_154_3 和 U6_154_4 分别对应第 11、10、9、8 位。P 字段的编码明细如表 4-3 所示：

U6_154_1(11)	U6_154_2(10)	U6_154_3(9)	U6_154_4(8)	选择
0	0	0	0	NOP
0	0	0	1	P(1)
0	0	1	0	P(2)
0	0	1	1	P(3)

0	1	0	0	P(4)
0	1	0	1	P(5)
0	1	1	0	P(6)
0	1	1	1	P(7)
1	0	0	0	LDR1’
1	0	0	1	P(9)
1	0	1	0	P(10)
1	0	1	1	P(11)
1	1	0	0	P(12)
1	1	0	1	LDPC
1	1	1	0	CN
1	1	1	1	LDR3’

表 4-3

上面所有的信号都是高电平有效的，所有上面的信号从 74154 出来后都接 74240 反向输出。具体电路图如图 4-3 所示：

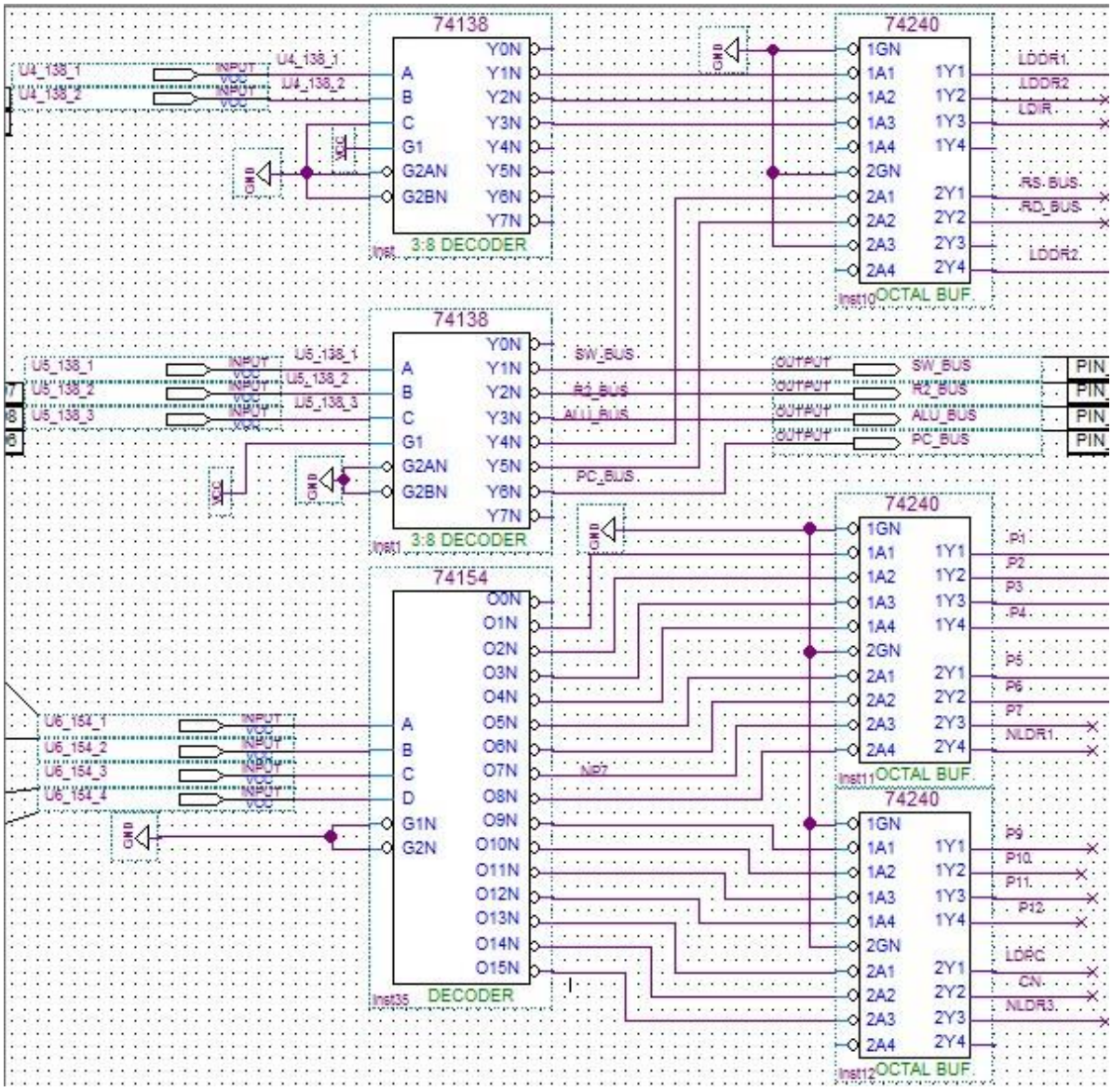

图 4-3

至此我们完成了 CPLD1 中一半的工作量。另一边就是地址转移逻辑的形成。它直接控制下地址的形成。本模型机的地址转移逻辑只对 MAR3~MAR0 即低四位地址有效其各位表达式如下：

$$\boldsymbol{MS_3 = \overline{P(1) \cdot IR_7 \cdot T_4 + P(3) \cdot Cy \cdot T_4}}$$

$$\boldsymbol{MS_2 = \overline{P(1) \cdot IR_6 \cdot T_4 + P(6) \cdot OF \cdot T_4}}$$

$$\boldsymbol{MS_1 = \overline{P(1) \cdot IR_5 \cdot T_4 + P(2) \cdot IR_3 \cdot T_4} \cdot \overline{P(4) \cdot IR_1 \cdot T_4 + P(5) \cdot IR_{1-L} \cdot T_4}}$$

$$\boldsymbol{MS_0 = \overline{P(1) \cdot IR_4 \cdot T_4} \cdot \overline{P(2) \cdot IR_2 \cdot T_4} \cdot \overline{P(4) \cdot IR_0 \cdot T_4}}$$

上述 MS3、MS2、MS1、MS0 在图中表示为跳线的输出。首先在 T_3时刻现将下一条指令的地址读到一个 8 位计数器中，在这里用 8 个 D 触发器代替。当低四位需要改变的时候，通过 D 触发器的异步置 1 功能，将该位置为 1，上述跳线的输出再使用 4 个输出分别输入到第四位的 D 触发器的异步置 1 端。最后通过三态缓冲器将其真实的下一条指令的地址输出。具体逻辑电路如图 4-4 所示：

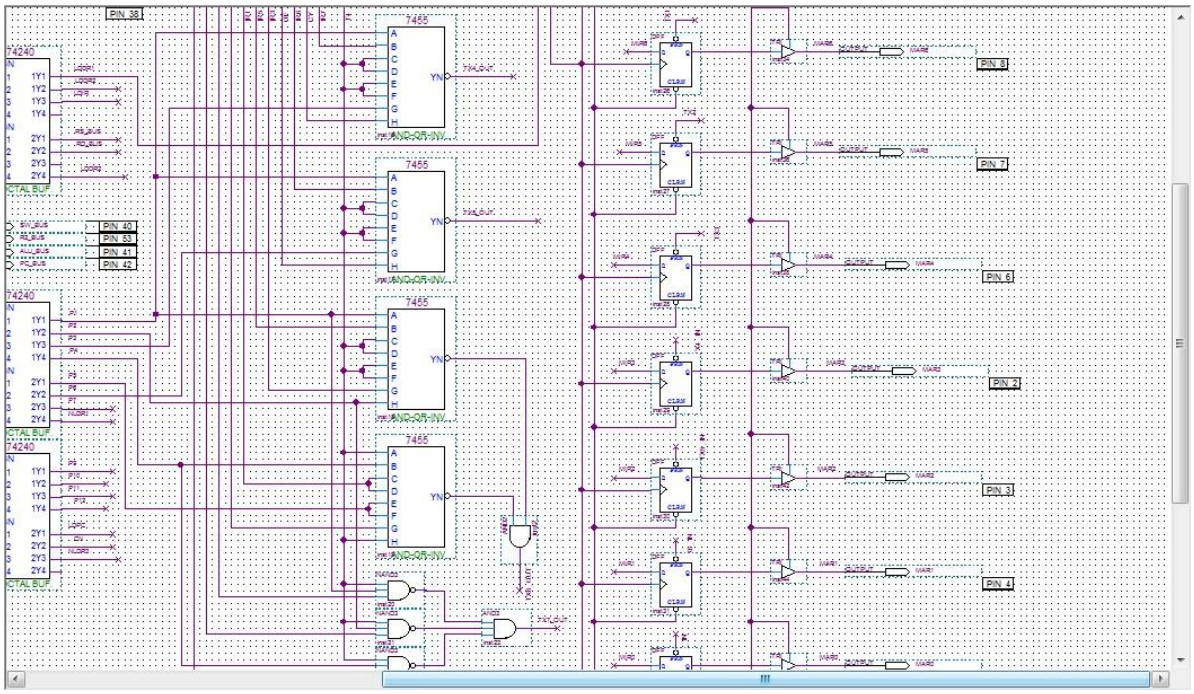

图 4-4

还有一个 CLK_273 的输出即 742738 位数据锁存器的时钟信号，该信号要由 LDIR 和 T_3控制加一个与门将其输出即可。至此完成 CPLD1 中所有的功能。

在使用逻辑电路完成了上述实验要求后，我们考虑使用 VHDL 语言重写上述功能。在清楚了工作的目的后，使用 VHDL 语言进行描述上述芯片的功能反而会更加容易。该芯片一共有 37 个输入端口，37 个输出端口。在实体中声明端口。如图 4-5 所示。

```
ENTITY CPLD1 IS
    port
    (
        U4_138_1 :  IN  STD_LOGIC;
        U4_138_2 :  IN  STD_LOGIC;
        U5_138_1 :  IN  STD_LOGIC;
        U5_138_2 :  IN  STD_LOGIC;
        U5_138_3 :  IN  STD_LOGIC;
        T4 :  IN  STD_LOGIC;
        IR0 :  IN  STD_LOGIC;
        IR1 :  IN  STD_LOGIC;
        IR2 :  IN  STD_LOGIC;
        IR3 :  IN  STD_LOGIC;
        IR4 :  IN  STD_LOGIC;
        IR5 :  IN  STD_LOGIC;
        IR6 :  IN  STD_LOGIC;
        IR7 :  IN  STD_LOGIC;
        OFG :  IN  STD_LOGIC;
        CY :  IN  STD_LOGIC;
        T3 :  IN  STD_LOGIC;
        NCLR :  IN  STD_LOGIC;
        MIR0 :  IN  STD_LOGIC;
        MIR1 :  IN  STD_LOGIC;
        MIR2 :  IN  STD_LOGIC;
        MIR3 :  IN  STD_LOGIC;
        MIR4 :  IN  STD_LOGIC;
        MIR5 :  IN  STD_LOGIC;
        MIR6 :  IN  STD_LOGIC;
        TX1 :  IN  STD_LOGIC;
        TX2 :  IN  STD_LOGIC;
        TX3 :  IN  STD_LOGIC;
        RIL :  IN  STD_LOGIC;
        TX4_IN :  IN  STD_LOGIC;
        TX5_IN :  IN  STD_LOGIC;
        TX6_IN :  IN  STD_LOGIC;
        TX7_IN :  IN  STD_LOGIC;
```

图 4-5

在对其功能进行描述，首先是译码器，通过相关的信号的输入的组合。输出相关的控制信号。我们可以使用 case…when 语句进行这样的控制。然后在地址转移逻辑的地方，直接使用公式进行计

算即可。如图 4-6 所示。完整的 VHDL 文本见附录。

```
499 end if;
500 end process;
501
502 process(T3,NCLR,TX3)
503 begin
504 if (NCLR = '0') then
505     DFF_inst28 <= '0';
506 elsif (TX3 = '0') then
507     DFF_inst28 <= '1';
508 elsif (rising_edge(T3)) then
509     DFF_inst28 <= MIR4;
510 end if;
511 end process;
512
513 process(T3,NCLR,TX4_IN)
514 begin
515 if (NCLR = '0') then
516     DFF_inst29 <= '0';
517 elsif (TX4_IN = '0') then
518     DFF_inst29 <= '1';
519 elsif (rising_edge(T3)) then
520     DFF_inst29 <= MIR3;
521 end if;
522 end process;
523
524 CLK_273 <= LDIR AND T3;
525
526 process(T3,NCLR,TX5_IN)
527 begin
528 if (NCLR = '0') then
529     DFF_inst30 <= '0';
530 elsif (TX5_IN = '0') then
531     DFF_inst30 <= '1';
532 elsif (rising_edge(T3)) then
533     DFF_inst30 <= MIR2;
534 end if;
535 end process;
```

图 4-6

（三） 通用寄存器控制模块设计

通用寄存器控制模块是 CPLD2 的工作，该模块，逻辑结构相对比较简单，控制信号较少，比较容易实现。根据模型机的指令系统的格式，我们知道，指令的操作数如果为寄存器的话，其地址通常会存在低四位。通用寄存器控制逻辑就是通过从每次输入的指令从指令寄存器 IR 中读取数据到 BUS，然后判断其低 4 位的与相关的 R_s 控制信号和 R_d 控制信号的逻辑。有关通用寄存器控制逻辑的表达式如下：

$$\overline{\boldsymbol{R_0}\rightarrow\mathrm{BUS}} = \overline{(\boldsymbol{R_s}\rightarrow\mathrm{BUS})\cdot\overline{\boldsymbol{IR_3}}\cdot\overline{\boldsymbol{IR_2}}\cdot+(\boldsymbol{R_d}\rightarrow\mathrm{BUS})\cdot\overline{\boldsymbol{IR_1}}\cdot\overline{\boldsymbol{IR_0}}}$$

$$\overline{\boldsymbol{R_1}\rightarrow\mathrm{BUS}} = \overline{(\boldsymbol{R_s}\rightarrow\mathrm{BUS})\cdot\overline{\boldsymbol{IR_3}}\cdot\boldsymbol{IR_2}\cdot+(\boldsymbol{R_d}\rightarrow\mathrm{BUS})\cdot\overline{\boldsymbol{IR_1}}\cdot\boldsymbol{IR_0}}$$

$$\overline{\boldsymbol{R_2}\rightarrow\mathrm{BUS}} = \overline{(\boldsymbol{R_s}\rightarrow\mathrm{BUS})\cdot\boldsymbol{IR_3}\cdot\overline{\boldsymbol{IR_2}}\cdot+(\boldsymbol{R_d}\rightarrow\mathrm{BUS})\cdot\boldsymbol{IR_1}\cdot\overline{\boldsymbol{IR_0}}}$$

$$\boldsymbol{LDR_0} = \boldsymbol{LDR_i}\cdot\overline{\boldsymbol{IR_1}}\cdot\overline{\boldsymbol{IR_0}}$$

$$\boldsymbol{LDR_0} = \boldsymbol{LDR_i}\cdot\overline{\boldsymbol{IR_1}}\cdot\boldsymbol{IR_0} + \boldsymbol{LDR_1}'$$

$$\boldsymbol{LDR_0} = \boldsymbol{LDR_i}\cdot\boldsymbol{IR_1}\cdot\overline{\boldsymbol{IR_0}}$$

$$\boldsymbol{LDR_0} = \boldsymbol{LDR_i}\cdot\boldsymbol{IR_1}\cdot\boldsymbol{IR_0} + \boldsymbol{LDR_3}'$$

其中的 $\boldsymbol{IR_0}$~$\boldsymbol{IR_3}$是从 *IR* 中获取并在 T_3 时刻读取到 BUS 上的。然后通过接反相器导出其反信号，为了从一定程度上消除竞争冒险现象，我们通过填加冗余项的方法来实现电路的稳定。从寄存器到 BUS 总线的控制信号如图 4-7 所示：

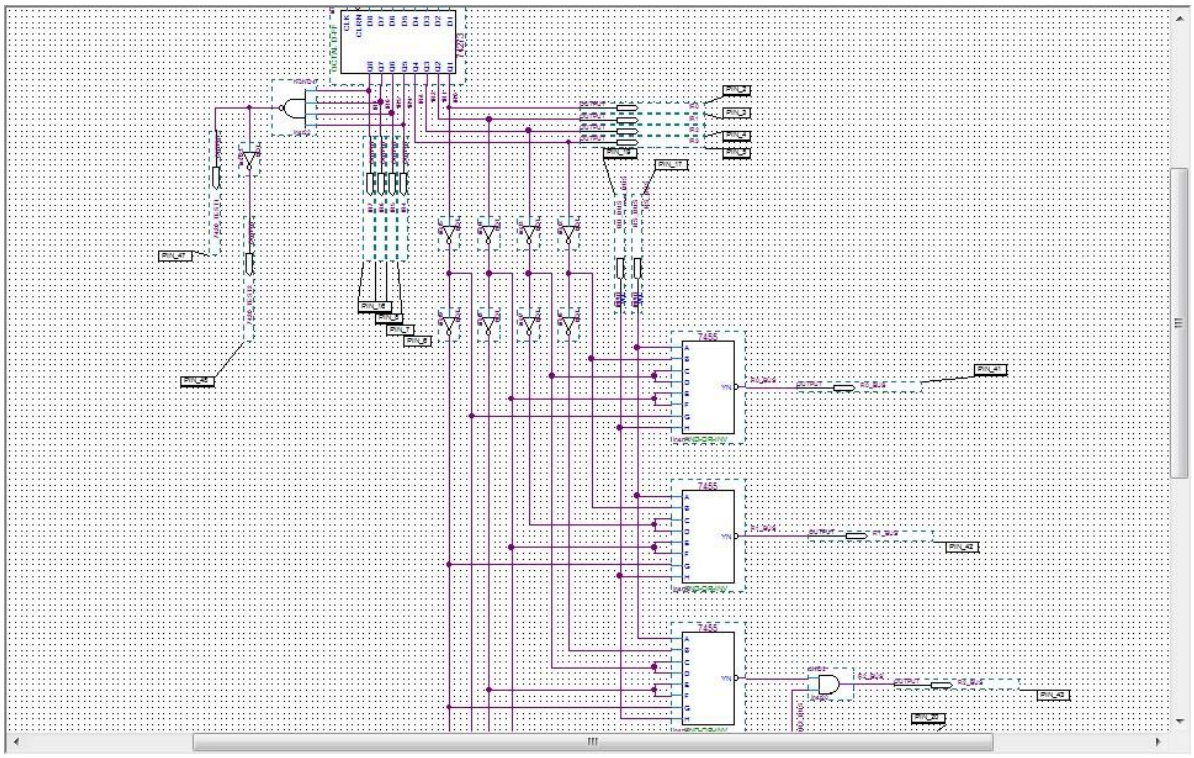

图 4-7

下半部分从总线到寄存器的控制信号逻辑如图 4-8 所示：

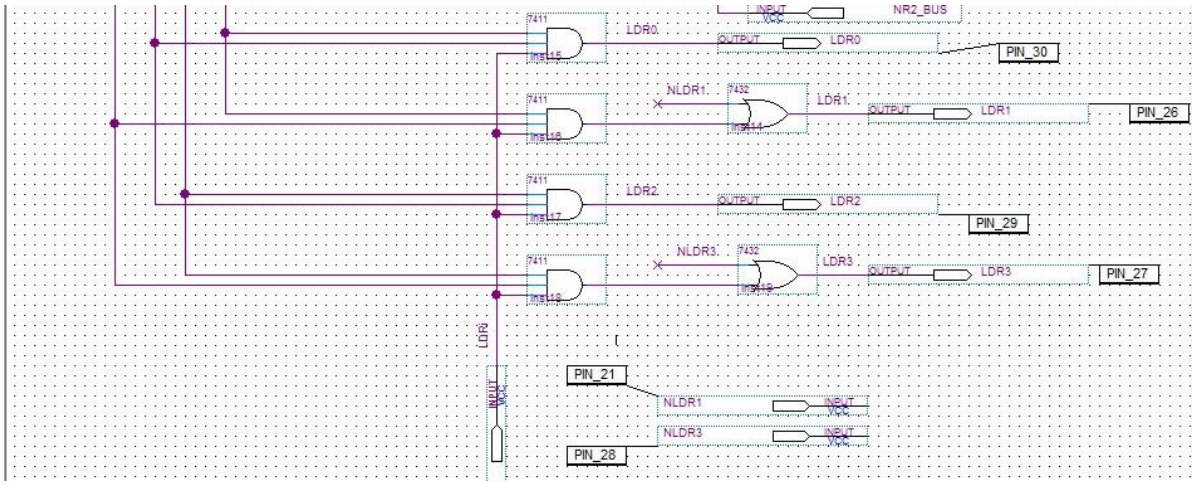

图 4-8

通过上述逻辑，我们完成了对特定寄存器的输入和输出的控制。

是使用 VHDL 语言编写通用寄存器控制逻辑比控制信号的生成更简单。只需要将相关的表达是进行计算就可以了。如图 4-9 所示：

```
299 SYNTHESIZED_WIRE_35 <= NOT(IR_ALTERA_SYNTHESIZED3);
300
301
302 R2_BUS <= SYNTHESIZED_WIRE_29 AND NR2_BUS;
303
304 SYNTHESIZED_WIRE_30 <= NOT(IR_ALTERA_SYNTHESIZED7 AND IR_ALTERA_SYNTHESIZED4 AND IR_ALTERA_SYNTHESIZED6 AND IR_ALTERA_SYN
305
306 7420_TEST2 <= NOT(SYNTHESIZED_WIRE_30);
307
308
309 SYNTHESIZED_WIRE_39 <= NOT(IR_ALTERA_SYNTHESIZED2);
310
311
312 SYNTHESIZED_WIRE_37 <= NOT(IR_ALTERA_SYNTHESIZED1);
313
314
315 SYNTHESIZED_WIRE_40 <= NOT(IR_ALTERA_SYNTHESIZED0);
316
317
318 SYNTHESIZED_WIRE_13 <= NOT(SYNTHESIZED_WIRE_35);
319
320
321 SYNTHESIZED_WIRE_36 <= NOT(SYNTHESIZED_WIRE_39);
322
323
324 SYNTHESIZED_WIRE_41 <= NOT(SYNTHESIZED_WIRE_37);
325
326
327 SYNTHESIZED_WIRE_38 <= NOT(SYNTHESIZED_WIRE_40);
328
329 IR4 <= IR_ALTERA_SYNTHESIZED4;
330 IR5 <= IR_ALTERA_SYNTHESIZED5;
```

图 4-9

（四） 芯片和引脚说明

1. 此次实验中 CPLD1 模块和 CPLD2 都是用了 MAX II EPM240T100C5 芯片，该种芯片有 80 个引脚，内部逻辑单元有 240 个。这两个芯片已经在我们的 C 板上做好了，外部的硬件电路也已经连接好了。我们需要做的就是将我们做好的逻辑写到芯片，并且按照实验板的生产厂家提供的引脚位置，指定引脚。
2. 引脚说明

   我们按照厂家提供的引脚分配文件，将引脚进行分配。附录。

（五） 实验结果说明

1. 本次实验我们没有在将这两个逻辑功能进行更加细致的功能区分，所以每个工程只有一张顶层原理图。如上图 4-1 和图 4-2 所示。
2. 本次实验实现了对模型机的 C 板的控制器的制作。即从获取指令后对控制信号的产生，和对数据在通用寄存器中和总线上的传输。以及地址转移逻辑的控制等。完成了上述工作，我们就制作了一个能够使用的 C 板，即微程序控制板。同时，使用我们的微程序可以正常的运行我们自己编写的程序。我们现在对 C 板进行测试。测试步骤如下：

（1） 按下 CLR 总清键

（2） 置 SW=0000 0011，然后 KQD，进入写内存指令

（3） 置 SW=0000 0000，然后 KQD

（4） KQD，把 PC 送 AR

（5） 置 SW=0100 0001（LDI 01H, R1），然后 KQD

（6） KQD，把数据送入 RAM，并且 PC+1

（7） KQD，返回第（5）步 SW=0000 0001

（8） 按下 CLR 总清键

（9） 由于执行程序的控制台指令也为 00 并且我写的内存地址为 00，故一直按下 KQD 即可。

（10） 观察 C 板和 A 板上的指示灯。如图 4-10 所示：

图 4-10 地址转移逻辑的指示灯

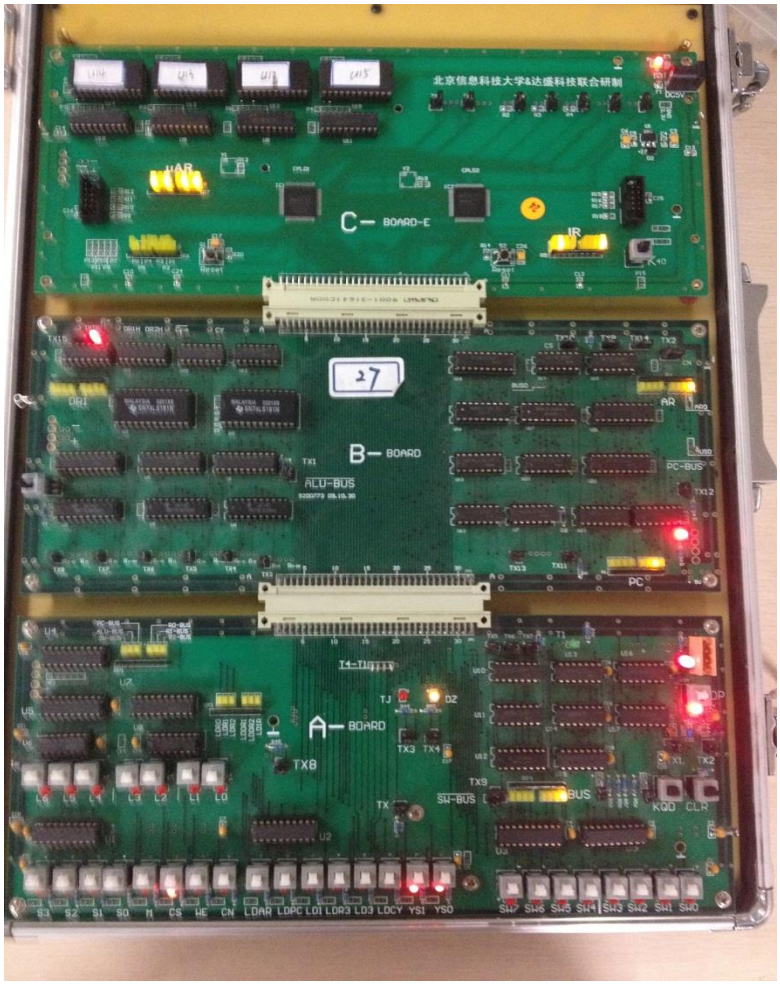

图 4-11 整机信号灯显示

六、实验总结

1. 本次实验为CPU设计课程EDA阶段综合性最强的一次实验，涉及了从实验原理的分析到芯片的选择，到实验原理图的绘制，对工程顶层文件的设置，编译，对芯片的管脚的指定以及下载和对VHDL文本语言进行逻辑描述的过程和生成符号文件的过程和方法。涵盖了EDA设计的绝大多数方法和技巧。
2. 本次实验可为一波三折，开始的时候，我们对本次实验的具体要求不甚清楚。对相关的原理不甚理解。导致我们只能对照课本，照葫芦画瓢，一点不差的完成课本生的逻辑图。由于课本的图上没有具体的引脚和相关的输入输出端口。我们结合C板制作厂家提供的原理图进行绘制，但还是麻烦不断，错误连连。而且根本不知道错误何在。直到最后还是没能实现应有的功能。我们并没有接着进行无谓的工作而是暂时停下了，转而攻克实验的原理。终于，我们明白了自己要做的工作就是产生控制信号、地址转移逻辑和通用寄存器控制信号。在明白了这些原理后，我们对照32位微程序的每位的含义，完全自己设计电路图，通过加译码电路产生直接编码的信号，通过将逻辑函数化为逻辑图的形式完成了CPLD1和CPLD2两张实验原理图。
3. 此次实验，我们综合运用了Quartus II的图形编辑器仿真器以及文本编辑器，产生了全部微程序控制器逻辑原理图和VHDL语言描述的程序。充分应用了层次话和模块化的设计方法，深入理解并掌握了自顶向下和自底向上混合设计的思路及方法。更加熟悉了对引脚指定编程下载的操作流程。实现了微程序控制器逻辑的全部功能。
4. 通过此次实验，我们最终实现了C板的微程序控制逻辑，但在开始的时候，我们不了解原理，只是盲目的去做，都白费了功夫现在看来，在做每个实验之前，都应该深入的理解实验的原理，弄清实验的逻辑，熟悉实验的步骤和过程，提前掌握实验的方法。这样，才能成功的做出实验并得到实验带给我们的好处。如果原理不清，方法不明，思路不通，则即便巧合出了结果，也是毫无意义，白费功夫。其次，此次实验是小组实验，我们小组工作细致，分工明确，组员和睦，互帮互助，着实减轻了个人很大的压力。