**北京邮电大学课程设计报告**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程设计**  **名称** | 计算机组成原理 | | **学院** | 计算机学院 | **指导教师** |  |
| **班级** | **班内序号** | **学号** | | **学生姓名** | **成绩** | |
| **2014211302** | 14 | 2014211168 | | 周尧棋 |  | |
| **2014211303** | 15 | 2014211201 | | 徐灿 |  | |
| **2014211302** | 18 | 2014211172 | | 邓精 |  | |
| **2014211303** | 18 | 2014211204 | | 李凡 |  | |
| **2014211303** | 16 | 2014211202 | | 廖竞鑫 |  | |
| **课**  **程**  **设**  **计**  **内**  **容** | 给定的数据格式和指令系统，在TEC-8设备上设计一台硬布线控制器控制的模型计算机，在设计好的模型计算机上进行测试、调试与改进，思考并完成中断功能的实现方案、测试方案。  分工：周尧棋、徐灿合作完成代码的编写与调试以及中断方案，廖竞鑫、李凡、邓精完成程序的测试与数据的记录。大家一起完成报告的撰写。 | | | | | |
| **学生**  **课程设计**  **报告**  （附页） | 见附页 | | | | | |
| **课**  **程**  **设**  **计**  **成**  **绩**  **评**  **定** | 遵照实践教学大纲并根据以下四方面综合评定成绩：  1、课程设计目的任务明确，选题符合教学要求，份量及难易程度  2、团队分工是否恰当与合理  3、综合运用所学知识，提高分析问题、解决问题及实践动手能力的效果  4、是否认真、独立完成属于自己的课程设计内容，课程设计报告是否思路清晰、文字通顺、书写规范  **评语**:  **成绩**:  指导教师签名：  年月日 | | | | | |

注：评语要体现每个学生的工作情况，可以加页。

**目录**

**1. 使用VHDL语言描述硬连线控制器的设计……………………………… 3**

**1.1 新设计CPU的指令系统………………………………………… 4**

**1.2 TEC-8实验台模拟计算机数据通路框图……………………… 5**

**1.3 硬连线控制器逻辑模块图…………………………………… 6**

**1.4 新设计CPU的硬连线控制器流程图…………………………… 7**

**1.5 硬连线控制器译码表………………………………………… 8**

**1.6 硬连线控制器VHDL源程序…………………………………… 11**

**2. 测试程序及数据………………………………………………………… 16**

**2.1 课本上的测试程序及运行结果 ……………………………… 17**

**2.2 老师给的测试程序及运行结果 ……………………………… 19**

**2.3 添加额外指令的测试程序及测试结果………………………21**

**3. 调试日志…………………………………………………………………23**

**4. 团队成员个人感想………………………………………………………26**

**5. 总结………………………………………………………………………28**

# 1使用VHDL硬件描述语言描述硬连线控制器的设计

**设计概述**：

此次设计是基于TEC-8实验台与EPM7128芯片的硬布线控制器设计。控制器应在控制台指令输入与机器指令输入后按照时序正确地产生硬件控制信号。于是我们采用了按照指令与时序顺序设计的原则，在代码中使用了选择结构来判断此时输入的控制台指令与机器指令内容，从而根据控制台指令（SWC SWB SWA）与机器指令（IR7——IR0）以及时序的判断来产生当前所需产生的控制信号。

TEC-8实验台模拟计算机数据通路中各寄存器的作用：

（1）**指令寄存器IR**：指令寄存器用来保存当前正在执行的一条指令。我们可以在实验台的IR7~IR0的小灯中观察到IR的值。

（2）**程序计数器PC**：PC的内容是从指存提取的第一条指令的地址，其保持的总是将要执行的下一条指令的地址。

（3）**数据地址寄存器AR**：数据地址寄存器用来保存当前CPU所访问的存储器中单元的地址。

（4）**通用寄存器R0~R3**：当算术逻辑单元ALU执行算数或逻辑运算时，为ALU提供一个工作区。

**ALU**：

根据74181ALU的功能表，M信号为1时，ALU做逻辑运算。M=0，Cin=1时，ALU做算术运算。运算的模式由S3~S0确定。

**时序**：

在TEC-8实验系统中，节拍脉冲信号Tk，（T1~T3）已经直接输送给数据通路。即：每个CPU周期分为三个节拍脉冲。在无其他信号的情况下，时序产生器只产生W1 W2两个节拍信号。如果SHORT信号有效，只产生W1，LONG信号有效，产生W1 W2 W3。SHORT LONG信号只对紧跟其后的第一个节拍点位的产生起作用。

## 1.1 新设计CPU的指令系统

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 汇编语言 | 功能 | 指令格式 | | |
| IR7 IR6 IR5 IR4 | IR3 IR2 | IR1 IR0 |
| 加法 | ADD Rd,Rs | Rd←Rd+Rs | 0001 | Rd | Rs |
| 减法 | SUB Rd,Rs | Rd←Rd-Rs | 0010 | Rd | Rs |
| 逻辑与 | AND Rd,Rs | Rd←Rd and Rs | 0011 | Rd | Rs |
| 加1 | INC Rd | Rd←Rd+1 | 0100 | Rd | XX |
| 取数 | LD Rd,[Rs] | Rd←[Rs] | 0101 | Rd | Rs |
| 存数 | ST Rs,[Rd] | Rs→[Rd] | 0110 | Rd | Rs |
| C条件转移 | JC addr | 如果C=1，则 PC←a+offset | 0111 | offset | |
| Z条件转移 | JZ addr | 如果Z=1，则 PC←a+offset | 1000 | offset | |
| 无条件转移 | JMP [Rd] | PC←Rd | 1001 | Rd | XX |
| 输出Rd | OUT Rd | SBUS←Rd | 1010 | Rd | XX |
| 异或运算 | XOR Rd，Rs | Rd←Rd xor Rs | 1011 | Rd | Rs |
| 同或运算 | XNOR Rd，Rs | Rd←Rd xnor Rs | 1100 | Rd | Rs |
| 逻辑或 | OR Rd,Rs | Rd←Rd or Rs | 1101 | Rd | Rs |
| 停机 | STOP | 暂停运行 | 1110 | XX | XX |
| 移数 | MOV Rd，Rs | Rd←Rs | 1111 | Rd | Rs |

表1 新设计CPU的指令系统

## 1.2 TEC-8实验台模拟计算机数据通路框图

这张图是我们此次实验所基于的数据通路。根据这张图上的各器件的控制信号描述，我们确定了控制信号产生的顺序。

图1 TEC-8实验台模拟计算机数据通路框图



## 1.3 硬连线控制器逻辑模块图

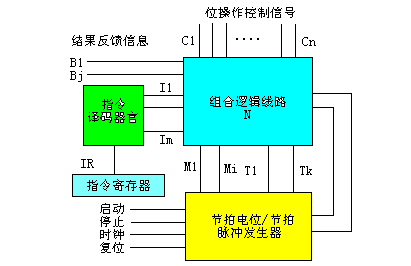


图2 硬连线控制器逻辑模块图

## 1.4 新设计CPU的硬连线控制器流程图

我们根据数据通路上各个器件的功能原理，以及各个操作对于器件的使用情况，确定了在各个CPU周期应产生的控制信号，制作了下面的时序流程图。

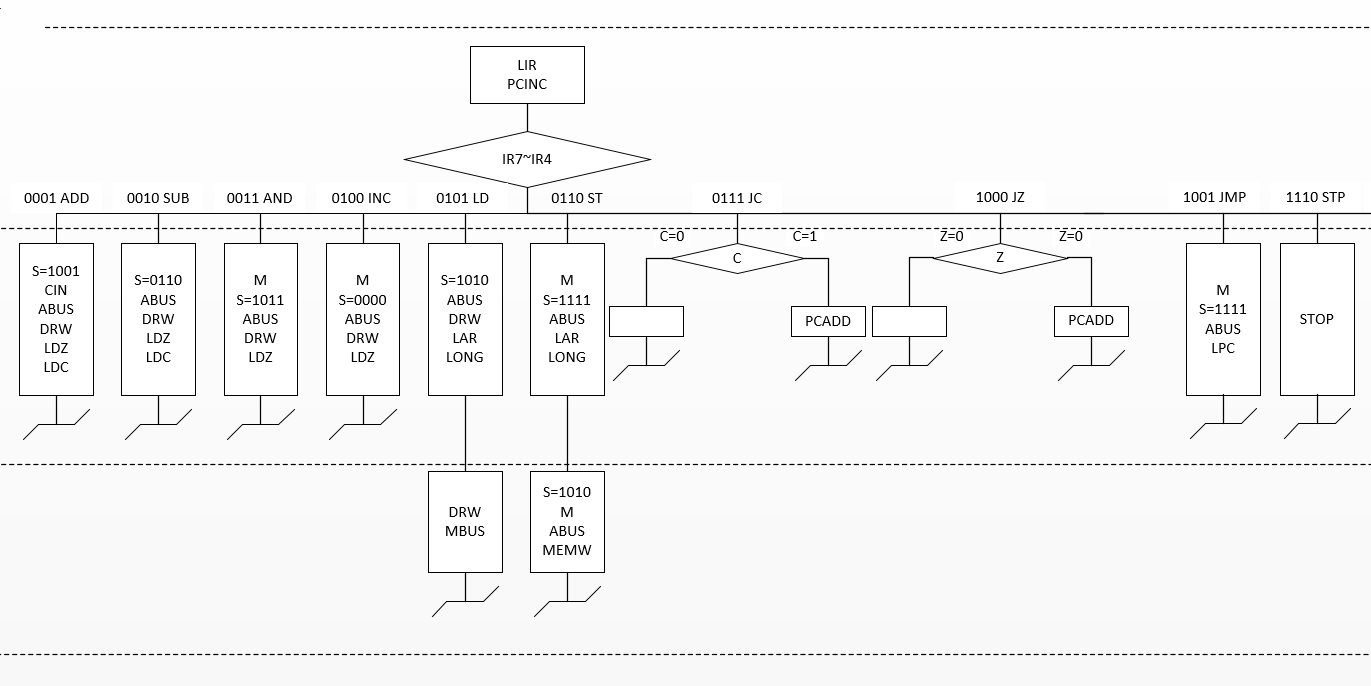
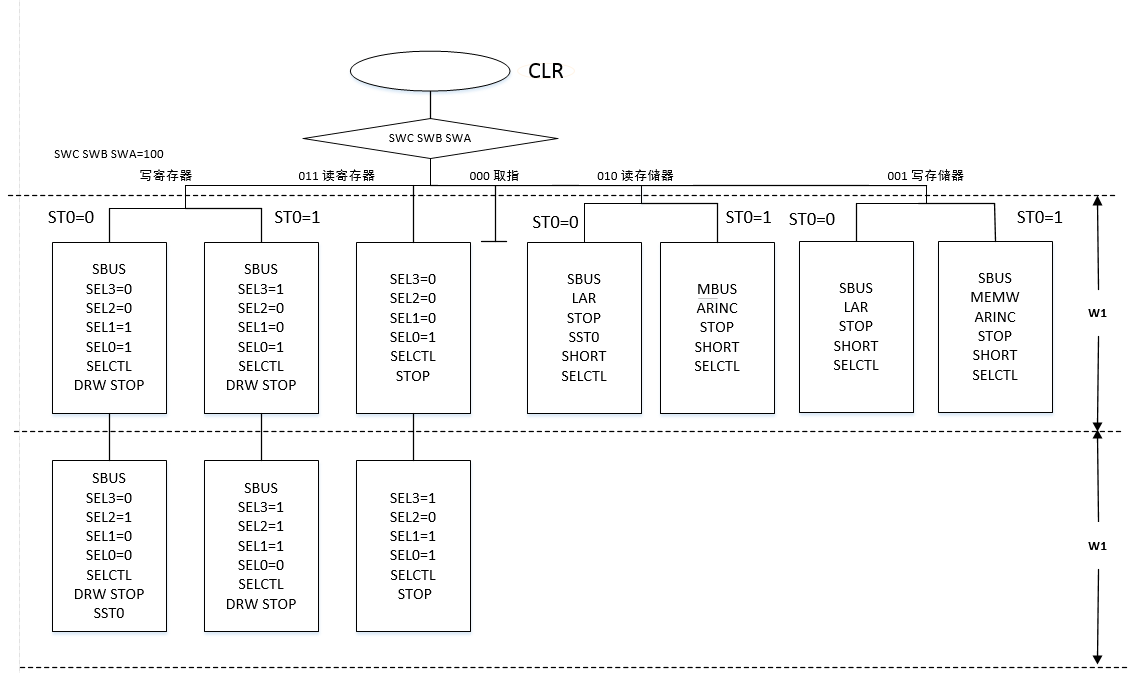
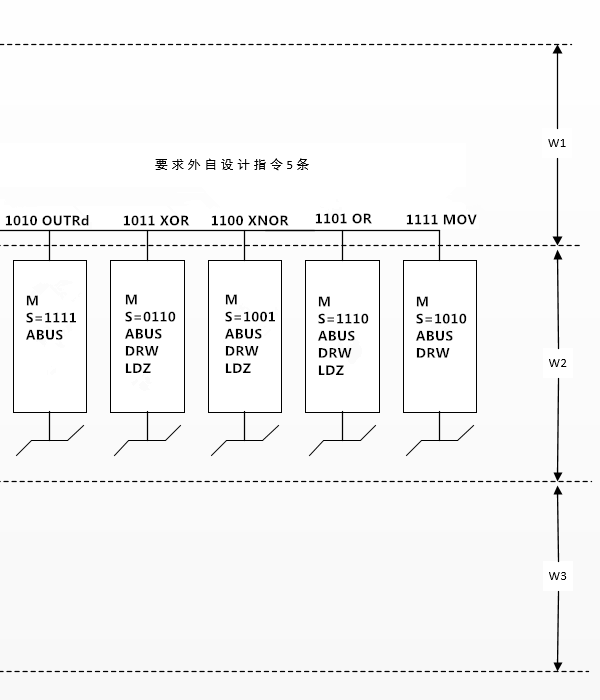


图3 新设计CPU的硬连线控制器流程图（基本要求）

根据数据通路的器件原理以及我们所扩展的操作对于器件的使用，我们总结出我们扩展的操作在各个时序产生的控制信号的内容，并画出了时序图。

图4新设计CPU的硬连线控制器流程图（自己设计指令）



## 1.5 硬连线控制器译码表

对于我们硬连线控制器产生的每个控制信号，都与当前的时序与输入指令有关。所以每个控制信号都可以使用时序与输入指令表示，我们根据这些关系，编辑了如下的译码表。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **100** | **011** | **010** | **001** |
|
| **DRW** | **W1+W2** |  |  |  |
| **PCINC** |  |  |  |  |
| **LPC** |  |  |  |  |
| **LAR** |  |  | **W1\*(NOT ST0)** | **W1\*(NOT ST0)** |
| **PCADD** |  |  |  |  |
| **ARINC** |  |  | **ST0\*W1** | **ST0\*W1** |
| **SELCTL** | **W1+W2** | **W1+W2** | **W1+W2** | **W1+W2** |
| **MEMW** |  |  |  | **ST0\*W1** |
| **STOP** | **W1+W2** | **W1+W2** | **W1** | **W1** |
| **LIR** |  |  |  |  |
| **LDZ** |  |  |  |  |
| **LDC** |  |  |  |  |
| **CIN** |  |  |  |  |
| **S3** |  |  |  |  |
| **S2** |  |  |  |  |
| **S1** |  |  |  |  |
| **S0** |  |  |  |  |
| **M** |  |  |  |  |
| **ABUS** |  |  |  |  |
| **SBUS** | **W1+W2** |  | **W1\*(NOT ST0)** | **W1** |
| **MBUS** |  |  | **W1\*ST0** |  |
| **SHORT** |  |  | **W1** | **W1** |
| **LONG** |  |  |  |  |
| **SEL3** | **ST0\*(W1+W2)** | **W2** |  |  |
| **SEL2** | **W2** |  |  |  |
| **SEL1** | **W1\*(NOT ST0)+W2\*ST0** | **W2** |  |  |
| **SEL0** | **W1** | **W1+W2** |  |  |

表2 硬连线控制器译码表1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **000** | | | | | | | |
| **0001 ADD** | **0010 SUB** | **0011 AND** | **0100 INC** | **0101 LD** | **0110 ST** | **0111 JC** | **1000 JZ** |
| **DRW** | **W2** | **W2** | **W2** | **W2** | **W3** |  |  |  |
| **PCINC** | **W1** | | | | | | | |
| **LPC** |  |  |  |  |  |  |  |  |
| **LAR** |  |  |  |  | **W2** | **W2** |  |  |
| **PCADD** |  |  |  |  |  |  | **W2\*C** | **W2\*Z** |
| **ARINC** |  |  |  |  |  |  |  |  |
| **SELCTL** |  |  |  |  |  |  |  |  |
| **MEMW** |  |  |  |  |  | **W3** |  |  |
| **STOP** |  |  |  |  |  |  |  |  |
| **LIR** | **W1** | | | | | | | |
| **LDZ** | **W2** | **W2** | **W2** | **W2** |  |  |  |  |
| **LDC** | **W2** | **W2** |  | **W2** |  |  |  |  |
| **CIN** | **W2** |  |  |  |  |  |  |  |
| **S3** | **W2** |  | **W2** |  | **W2** | **W2+W3** |  |  |
| **S2** |  | **W2** |  |  |  | **W2** |  |  |
| **S1** |  | **W2** | **W2** |  | **W2** | **W2+W3** |  |  |
| **S0** | **W2** |  | **W2** |  |  | **W2** |  |  |
| **M** |  |  | **W2** |  | **W2** | **W2+W3** |  |  |
| **ABUS** | **W2** | **W2** | **W2** | **W2** | **W2** | **W2+W3** |  |  |
| **SBUS** |  |  |  |  |  |  |  |  |
| **MBUS** |  |  |  |  | **W3** |  |  |  |
| **SHORT** |  |  |  |  |  |  |  |  |
| **LONG** |  |  |  |  | **W2** | **W2** |  |  |
| **SEL3** |  |  |  |  |  |  |  |  |
| **SEL2** |  |  |  |  |  |  |  |  |
| **SEL1** |  |  |  |  |  |  |  |  |
| **SEL0** |  |  |  |  |  |  |  |  |

表3 硬连线控制器译码表续1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **000** | | | | | | |
| **1001 JMP** | **1110 STP** | **1010**  **OUTRd** | **1011**  **XOR** | **1100**  **XNOR** | **1101**  **OR** | **1111 MOV** |
| **DRW** |  |  |  | **W2** | **W2** | **W2** | **W2** |
| **PCINC** | **W1** | | | | | | |
| **LPC** | **W2** |  |  |  |  |  |  |
| **LAR** |  |  |  |  |  |  |  |
| **PCADD** |  |  |  |  |  |  |  |
| **ARINC** |  |  |  |  |  |  |  |
| **SELCTL** |  |  |  |  |  |  |  |
| **MEMW** |  |  |  |  |  |  |  |
| **STOP** |  | **W2** |  |  |  |  |  |
| **LIR** | **W1** | | | | | | |
| **LDZ** |  |  |  | **W2** | **W2** | **W2** |  |
| **LDC** |  |  |  |  |  |  |  |
| **CIN** |  |  |  |  |  |  |  |
| **S3** | **W2** |  | **W2** |  | **W2** | **W2** | **W2** |
| **S2** | **W2** |  | **W2** | **W2** |  | **W2** |  |
| **S1** | **W2** |  | **W2** | **W2** |  | **W2** | **W2** |
| **S0** | **W2** |  | **W2** |  | **W2** |  |  |
| **M** | **W2** |  | **W2** | **W2** | **W2** | **W2** | **W2** |
| **ABUS** | **W2** |  | **W2** | **W2** | **W2** | **W2** | **W2** |
| **SBUS** |  |  |  |  |  |  |  |
| **MBUS** |  |  |  |  |  |  |  |
| **SHORT** |  |  |  |  |  |  |  |
| **LONG** |  |  |  |  |  |  |  |
| **SEL3** |  |  |  |  |  |  |  |
| **SEL2** |  |  |  |  |  |  |  |
| **SEL1** |  |  |  |  |  |  |  |
| **SEL0** |  |  |  |  |  |  |  |

表4 硬连线控制器译码表续2

## 1.6 硬连线控制器的VHDL源程序

LIBRARY IEEE ;

USE IEEE.STD\_LOGIC\_1164.ALL ;

USE IEEE.STD\_LOGIC\_UNSIGNED.ALL ;

ENTITY CPU IS

PORT (CLR, -- Clear

T3, -- Clock Pulse T3

SWA,SWB,SWC, -- Console Signal

IR7,IR6,IR5,IR4, -- Opcode

W3,W2,W1, -- Clock Level

C,Z: IN STD\_LOGIC ; -- Carry Bit or 0 Result Output

LDZ,LDC, -- Loading Instruction of C,Z

CIN, -- Output of Carry Bit

S3,S2,S1,S0, -- Operation Pattern of ALU

M, -- Arithmetic or Logic operation of ALU

ABUS,SBUS,MBUS, -- Switch of ABUS,SBUS,MBUS

DRW, -- Wrting Instruction of General Register

PCINC,PCADD,ARINC,-- Adding-1 Instrction of PC and AR, Adding Instruction of PC

LPC,LAR,LIR, -- Loading Instruction of PC,AR,IR

SELCTL, -- Experiment Instrument Signal

MEMW, -- Writing Instruction of RAM

STOP, -- Instruction of Stoping Machine

SHORT,LONG, -- Flag of 1 CPU Level or 3 CPU Level

SEL3,SEL2,SEL1,SEL0,-- Selection of General Register

O1,O2,O3,O4: OUT STD\_LOGIC) ;-- For Debugging

END CPU ;

ARCHITECTURE BEHAVE OF CPU IS

SIGNAL IR: STD\_LOGIC\_VECTOR (7 DOWNTO 4) ; -- Opcode Vector

SIGNAL SW: STD\_LOGIC\_VECTOR (2 DOWNTO 0) ; -- Console Signal Vector

SIGNAL ST0: STD\_LOGIC ; -- Flag of Instruction Period

SIGNAL SST0: STD\_LOGIC ; -- Flag of Instruction Period

SIGNAL FLAG: STD\_LOGIC ; -- Flag of Instruction Period

SIGNAL FFLAG: STD\_LOGIC ; -- Flag of Instruction Period

BEGIN

SW <= SWC & SWB & SWA ;

IR <= IR7 & IR6 & IR5 & IR4 ;

PROCESS (CLR,SW,IR,ST0,SST0,T3,W1,W2,W3,C,Z,FFLAG,FLAG)

BEGIN -- Initialization

LDZ <= '0' ;

LDC <= '0' ;

CIN <= '0' ;

S3 <= '0' ;

S2 <= '0' ;

S1 <= '0' ;

S0 <= '0' ;

M <= '0' ;

ABUS <= '0' ;

SBUS <= '0' ;

MBUS <= '0' ;

DRW <= '0' ;

PCINC <= '0' ;

PCADD <= '0' ;

ARINC <= '0' ;

LPC <= '0' ;

LAR <= '0' ;

LIR <= '0' ;

SELCTL <= '0' ;

MEMW <= '0' ;

STOP <= '0' ;

SHORT <= '0' ;

LONG <= '0' ;

SEL3 <= '0' ;

SEL2 <= '0' ;

SEL1 <= '0' ;

SEL0 <= '0' ;

IF (CLR = '0') THEN -- Response to CLR Signal

ST0 <= '0' ;

SST0 <= '0' ;

FFLAG <='0' ;

FLAG <='0' ;

ELSIF (SST0 = '1' AND falling\_edge(T3)) THEN -- Change to New CPU Levels

ST0 <= '1' ;

ELSIF (FFLAG = '1' AND falling\_edge(T3)) THEN -- Change to New CPU Levels

FLAG <= '1' ;

END IF ;

CASE SW IS

WHEN "100" => --Load Register

IF (ST0 = '0' AND W1 = '1' AND W2 = '0') THEN

SBUS <= '1' ;

SEL3 <= '0' ;

SEL2 <= '0' ;

SEL1 <= '1' ;

SEL0 <= '1' ;

SELCTL <= '1' ;

DRW <= '1' ;

STOP <= '1' ;

ELSIF (ST0 = '0' AND W1 = '0' AND W2 = '1') THEN

SBUS <= '1' ;

SEL3 <= '0' ;

SEL2 <= '1' ;

SEL1 <= '0' ;

SEL0 <= '0' ;

SELCTL <= '1' ;

STOP <= '1' ;

DRW <= '1' ;

SST0 <= '1' ;

ELSIF (ST0 = '1' AND W1 = '1' AND W2 = '0') THEN

SBUS <= '1' ;

SEL3 <= '1' ;

SEL2 <= '0' ;

SEL1 <= '0' ;

SEL0 <= '1' ;

SELCTL <= '1' ;

DRW <= '1' ;

STOP <= '1' ;

ELSIF (ST0 = '1' AND W1 = '0' AND W2 = '1') THEN

SBUS <= '1' ;

SEL3 <= '1' ;

SEL2 <= '1' ;

SEL1 <= '1' ;

SEL0 <= '0' ;

SELCTL <= '1' ;

DRW <= '1' ;

STOP <= '1' ;

END IF ;

WHEN "011" => -- Write Register

IF (W1 = '1' AND W2 = '0') THEN

SEL3 <= '0' ;

SEL2 <= '0' ;

SEL1 <= '0' ;

SEL0 <= '1' ;

SELCTL <= '1' ;

STOP <= '1' ;

ELSIF (W1 = '0' AND W2 = '1') THEN

SEL3 <= '1' ;

SEL2 <= '0' ;

SEL1 <= '1' ;

SEL0 <= '1' ;

SELCTL <= '1' ;

STOP <= '1' ;

END IF ;

WHEN "010" => -- Read Memory

IF (ST0 = '0' AND W1 = '1' ) THEN

SBUS <= '1' ;

LAR <= '1' ;

STOP <= '1' ;

SST0 <= '1' ;

SHORT <= '1' ;

SELCTL <= '1' ;

ELSIF (ST0 = '1' AND W1 = '1' ) THEN

MBUS <= '1' ;

ARINC <= '1' ;

STOP <= '1' ;

SHORT <= '1' ;

SELCTL <= '1' ;

END IF ;

WHEN "001" => -- Write Memory

IF (ST0 = '0' AND W1 = '1') THEN

SBUS <= '1' ;

LAR <= '1' ;

STOP <= '1' ;

SST0 <= '1' ;

SHORT <= '1' ;

SELCTL <= '1' ;

ELSIF (ST0 = '1' AND W1 = '1') THEN

SBUS <= '1' ;

MEMW <= '1' ;

ARINC <= '1' ;

STOP <= '1' ;

SHORT <= '1' ;

SELCTL <= '1' ;

END IF ;

WHEN "000" => -- Get Instructions

IF(FLAG <= '0' ) THEN -- Execute at Random Address

SBUS <= W1;

LPC <= W1;

STOP <= W1;

SHORT <= W1;

FFLAG <= W1;

ELSE

LIR <= W1 ;

PCINC <= W1 ;

END IF ;

CASE IR IS

WHEN "0001" => --ADD

S3 <= W2 ;

S0 <= W2 ;

CIN <= W2 ;

ABUS <= W2 ;

DRW <= W2 ;

LDZ <= W2 ;

LDC <= W2 ;

WHEN "0010" => --SUB

S2 <= W2 ;

S1 <= W2 ;

ABUS <= W2 ;

DRW <= W2 ;

LDZ <= W2 ;

LDC <= W2 ;

WHEN "0011" => --AND

M <= W2 ;

S3 <= W2 ;

S1 <= W2 ;

S0 <= W2 ;

ABUS <= W2 ;

DRW <= W2 ;

LDZ <= W2 ;

WHEN "0100" => --INC

ABUS <= W2 ;

DRW <= W2 ;

LDZ <= W2 ;

LDC <= W2 ;

WHEN "0101" => --LD

M <= W2 ;

S3 <= W2 ;

S1 <= W2 ;

ABUS <= W2 ;

LAR <= W2 ;

LONG <= W2 ;

DRW <= W3 ;

MBUS <= W3 ;

WHEN "0110" => --ST

M <= W2 OR W3 ;

S3 <= W2 OR W3 ;

S2 <= W2 ;

S1 <= W2 OR W3 ;

S0 <= W2 ;

ABUS <= W2 OR W3 ;

LAR <= W2 ;

LONG <= W2 ;

MEMW <= W3 ;

WHEN "0111" => --JC

IF (C = '1') THEN

PCADD <= W2 ;

END IF ;

WHEN "1000" => --JZ

IF (Z = '1') THEN

PCADD <= W2 ;

END IF ;

WHEN "1001" => --JMP

M <= W2 ;

S3 <= W2 ;

S2 <= W2 ;

S1 <= W2 ;

S0 <= W2 ;

ABUS <= W2 ;

LPC <= W2 ;

WHEN "1010" => --OUT Rd

M <= W2 ;

S3 <= W2 ;

S2 <= W2 ;

S1 <= W2 ;

S0 <= W2 ;

ABUS <= W2 ;

WHEN "1011" => --XOR

M <= W2 ;

S2 <= W2 ;

S1 <= W2 ;

ABUS <= W2 ;

DRW <= W2 ;

LDZ <= W2 ;

WHEN "1100" => --XNOR

M <= W2 ;

S3 <= W2 ;

S0 <= W2 ;

ABUS <= W2 ;

DRW <= W2 ;

LDZ <= W2 ;

WHEN "1101" => --OR

M <= W2 ;

S3 <= W2 ;

S2 <= W2 ;

S1 <= W2 ;

ABUS <= W2 ;

DRW <= W2 ;

LDZ <= W2 ;

WHEN "1110" => --STP

STOP <= W2 ;

WHEN OTHERS => --MOV

M <= W2;

S3 <= W2;

S1 <= W2;

ABUS <= W2;

DRW <= W2;

END CASE ;

WHEN OTHERS => NULL ;

END CASE ;

O1 <= IR(4) ;

O2 <= IR(5) ;

O3 <= IR(6) ;

O4 <= IR(7) ;

END PROCESS ;

END BEHAVE ;

# 测试程序

测试程序（1）：当时计组实验课本上的程序

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址 | 指令 | 机器代码 | 地址 | 指令 |
| 00H | LD R0,[R3] | 53H | 0FH | 85H |
| 01H | INC R3 | 4CH | 10H | 23H |
| 02H | LD R1,[R3] | 57H | 11H | EFH |
| 03H | SUB R0,R1 | 21H | 12H | 00H |
| 04H | JZ 0BH | 86H |  |  |
| 05H | ST R0,[R2] | 68H |  |  |
| 06H | INC R3 | 4CH |  |  |
| 07H | LD R0,[R3] | 53H |  |  |
| 08H | ADD R0,R1 | 11H |  |  |
| 09H | JC 0CH | 72H |  |  |
| 0AH | INC R2 | 48H |  |  |
| 0BH | ST R2,[R2] | 6AH |  |  |
| 0CH | AND R0,R1 | 31H | 寄存器 | 数据 |
| 0DH | OUT R2 | A2H | R2 | 12H |
| 0EH | STP | E0H | R3 | 0FH |

测试结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 始 | R2 00010010 | R3 00001111 |  |  |
|  |  |  |  |  |
|  | R0 | R1 | R2 | R3 |
| 00 | 10000101 |  | 00010010 | 00001111 |
| 01 | 10000101 |  | 00010010 | 00010000 |
| 02 | 10000101 | 00100011 | 00010010 | 00010000 |
| 03 | 01100010 | 00100011 | 00010010 | 00010000 |
| 04 | 01100010 | 00100011 | 00010010 | 00010000 |
| 05 | 此步之后地址12H中存放指令01100010 | | | |
| 06 | 01100010 | 00100011 | 00010010 | 00010001 |
| 07 | 11101111 | 00100011 | 00010010 | 00010001 |
| 08 | 00010010 | 00100011 | 00010010 | 00010001 |
| 09 | 前一步有进位，PC+2 | | | |
| 0C | 00000010 | 00100011 | 00010010 | 00010001 |
| 0D | 00000010 | 00100011 | 00010010 | 00010001 |
| 0E | STOP | | | |
|  |  | | | |
| 终 | R0 00000010 | R1 00100011 | R2 00010010 | R3 00010001 |
|  | 0FH 10000101 | 10H 00100011 | 11H 11101111 | 12H 01100010 |

测试程序（2）：老师给的程序

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址 | 指令 | 机器码 | 地址 | 指令 |
| 00H | LD RO,[R2] | 52H | 60H | 67H |
| 01H | INC R2 | 48H | 61H | 80H |
| 02H | LD R1,[R2] | 56H | 62H | FDH |
| 03H | ADD R0.R1 | 11H | 80H | 60H |
| 04H | JC 06H | 71H | FEH | 03H |
| 05H | AND R1,R0 | 34H | FFH | 03H |
| 06H | SUB R0,R2 | 22H |  |  |
| 07H | INC R1 | 44H |  |  |
| 08H | STA R0,[R1] | 64H |  |  |
| 09H | INC R3 | 4CH |  |  |
| 0AH | JZ 0DH | 82H |  |  |
| 0BH | LD R2,[R3] | 5BH |  |  |
| 0CH | JMP [R2] | 98H |  |  |
| 0DH | INC R3 | 4CH |  |  |
| 0EH | INC R3 | 4CH |  |  |
| 0FH | SUB R0,R2 | 22H |  |  |
| 10H | LD R2,[R0] | 58H |  |  |
| 11H | ADD R3,R2 | 1EH |  |  |
| 12H | LD R3,[R3] | 5FH | 寄存器 | 数据 |
| 13H | OUT R0 | A0H | R2 | 60H |
| 14H | STP | E0H | R3 | FDH |

测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 始 | R2 01100000 | R3 11111101 |  |  |
|  | 60H 01100111 | 61H 10000000 | 62H 11111101 |  |
|  | 80H 01100000 | FEH 00000011 | FFH 00000011 |  |
|  |  |  |  |  |
|  | R0 | R1 | R2 | R3 |
| 00H | 01100111 |  | 01100000 | 11111101 |
| 01H | 01100111 |  | 01100001 | 11111101 |
| 02H | 01100111 | 10000000 | 01100001 | 11111101 |
| 03H | 11100111 | 10000000 | 01100001 | 11111101 |
| 04H | 11100111 | 10000000 | 01100001 | 11111101 |
| 05H | 11100111 | 10000000 | 01100001 | 11111101 |
| 06H | 10000110 | 10000000 | 01100001 | 11111101 |
| 07H | 10000110 | 10000001 | 01100001 | 11111101 |
| 08H | 此步之后地址81H 存放指令10000110 | | | |
| 09H | 10000110 | 10000001 | 01100001 | 11111110 |
| 0AH | 10000110 | 10000001 | 01100001 | 11111110 |
| 0BH | 10000110 | 10000001 | 00000011 | 11111110 |
| 0CH | 无条件转移至R2 | | | |
| 03H | 00000111 | 10000001 | 00000011 | 11111110 |
| 04H | 前一步有进位，PC+1 | | | |
| 06H | 00000100 | 10000001 | 00000011 | 11111110 |
| 07H | 00000100 | 10000010 | 00000011 | 11111110 |
| 08H | 此步之后地址82H 存放指令00000100 | | | |
| 09H | 00000100 | 10000010 | 00000011 | 11111111 |
| 0AH | 00000100 | 10000010 | 00000011 | 11111111 |
| 0BH | 00000100 | 10000010 | 00000011 | 11111111 |
| 0CH | 无条件转移至R2 | | | |
| 03H | 10000110 | 10000010 | 00000011 | 11111111 |
| 04H | 10000110 | 10000010 | 00000011 | 11111111 |
| 05H | 10000110 | 10000010 | 00000011 | 11111111 |
| 06H | 10000011 | 10000010 | 00000011 | 11111111 |
| 07H | 10000011 | 10000011 | 00000011 | 11111111 |
| 08H | 此步之后地址83H 存放指令10000011 | | | |
| 09H | 10000011 | 10000011 | 00000011 | 00000000 |
| 0AH | 前一步结果为0，PC+2 | | | |
| 0DH | 10000011 | 10000011 | 00000011 | 00000001 |
| 0EH | 10000011 | 10000011 | 00000011 | 00000010 |
| 0FH | 10000000 | 10000011 | 00000011 | 00000010 |
| 10H | 10000000 | 10000011 | 01100000 | 00000010 |
| 11H | 10000000 | 10000011 | 01100000 | 01100010 |
| 12H | 10000000 | 10000011 | 01100000 | 11111101 |
| 13H | 10000000 | 10000011 | 01100000 | 11111101 |
| 14H | STOP | | | |
|  | | | | |
| 终 | R0 10000000 | R1 10000011 | R2 01100000 | R3 11111101 |
|  | 60H 01100111 | 61H 10000000 | 62H 11111101 |  |
|  | 80H 01100000 | 81H 10000110 | 82H 00000100 | 83H 10000011 |
|  |  | FEH 00000011 | FFH 00000011 |  |

测试程序（3）：自己设计的包含额外指令的程序

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 地址 | 指令 | 机器码 | 地址 | 指令 |
| 00H | LD RO,[R2] | 52H | 60H | 67H |
| 01H | INC R2 | 48H | 61H | 80H |
| 02H | LD R1,[R2] | 56H | 62H | FDH |
| 03H | ADD R0.R1 | 11H | 80H | 60H |
| 04H | JC 06H | 71H | FEH | 03H |
| 05H | AND R1,R0 | 34H | FFH | 03H |
| 06H | SUB R0,R2 | 22H |  |  |
| 07H | INC R1 | 44H |  |  |
| 08H | STA R0,[R1] | 64H |  |  |
| 09H | INC R3 | 4CH |  |  |
| 0AH | JZ 0DH | 82H |  |  |
| 0BH | LD R2,[R3] | 5BH |  |  |
| 0CH | JMP [R2] | 98H |  |  |
| 0DH | INC R3 | 4CH |  |  |
| 0EH | INC R3 | 4CH |  |  |
| 0FH | SUB R0,R2 | 22H |  |  |
| 10H | LD R2,[R0] | 58H |  |  |
| 11H | ADD R3,R2 | 1EH |  |  |
| 12H | LD R3,[R3] | 5FH | 寄存器 | 数据 |
| 13H | OUT R0 | A0H | R2 | 60H |
| 14H | XOR R0,R1 | B1H | R3 | FDH |
| 15H | XNOR R2,R3 | CBH |  |  |
| 16H | OR R0,R2 | D2H |  |  |
| 17H | MOV R1,R3 | F7H |  |  |
| 18H | STP | E0H |  |  |

测试结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 始 | R2 01100000 | R3 11111101 |  |  |
|  | 60H 01100111 | 61H 10000000 | 62H 11111101 |  |
|  | 80H 01100000 | FEH 00000011 | FFH 00000011 |  |
|  |  |  |  |  |
|  | R0 | R1 | R2 | R3 |
| 00H | 01100111 |  | 01100000 | 11111101 |
| 01H | 01100111 |  | 01100001 | 11111101 |
| 02H | 01100111 | 10000000 | 01100001 | 11111101 |
| 03H | 11100111 | 10000000 | 01100001 | 11111101 |
| 04H | 11100111 | 10000000 | 01100001 | 11111101 |
| 05H | 11100111 | 10000000 | 01100001 | 11111101 |
| 06H | 10000110 | 10000000 | 01100001 | 11111101 |
| 07H | 10000110 | 10000001 | 01100001 | 11111101 |
| 08H | 此步之后地址81H 存放指令10000110 | | | |
| 09H | 10000110 | 10000001 | 01100001 | 11111110 |
| 0AH | 10000110 | 10000001 | 01100001 | 11111110 |
| 0BH | 10000110 | 10000001 | 00000011 | 11111110 |
| 0CH | 无条件转移至R2 | | | |
| 03H | 00000111 | 10000001 | 00000011 | 11111110 |
| 04H | 前一步有进位，PC+1 | | | |
| 06H | 00000100 | 10000001 | 00000011 | 11111110 |
| 07H | 00000100 | 10000010 | 00000011 | 11111110 |
| 08H | 此步之后地址82H 存放指令00000100 | | | |
| 09H | 00000100 | 10000010 | 00000011 | 11111111 |
| 0AH | 00000100 | 10000010 | 00000011 | 11111111 |
| 0BH | 00000100 | 10000010 | 00000011 | 11111111 |
| 0CH | 无条件转移至R2 | | | |
| 03H | 10000110 | 10000010 | 00000011 | 11111111 |
| 04H | 10000110 | 10000010 | 00000011 | 11111111 |
| 05H | 10000110 | 10000010 | 00000011 | 11111111 |
| 06H | 10000011 | 10000010 | 00000011 | 11111111 |
| 07H | 10000011 | 10000011 | 00000011 | 11111111 |
| 08H | 此步之后地址83H 存放指令10000011 | | | |
| 09H | 10000011 | 10000011 | 00000011 | 00000000 |
| 0AH | 前一步结果为0，PC+2 | | | |
| 0DH | 10000011 | 10000011 | 00000011 | 00000001 |
| 0EH | 10000011 | 10000011 | 00000011 | 00000010 |
| 0FH | 10000000 | 10000011 | 00000011 | 00000010 |
| 10H | 10000000 | 10000011 | 01100000 | 00000010 |
| 11H | 10000000 | 10000011 | 01100000 | 01100010 |
| 12H | 10000000 | 10000011 | 01100000 | 11111101 |
| 13H | 10000000 | 10000011 | 01100000 | 11111101 |
| 14H | 00000011 | 10000011 | 01100000 | 11111101 |
| 15H | 00000011 | 10000011 | 01100010 | 11111101 |
| 16H | 01100011 | 10000011 | 01100010 | 11111101 |
| 17H | 01100011 | 11111101 | 01100010 | 11111101 |
| 18H | STOP | | | |
|  | | | | |
| 终 | R0 01100011 | R1 11111101 | R2 01100010 | R3 11111101 |
|  | 60H 01100111 | 61H 10000000 | 62H 11111101 |  |
|  | 80H 01100000 | 81H 10000110 | 82H 00000100 | 83H 10000011 |
|  |  | FEH 00000011 | FFH 00000011 |  |

# 3.调试日志

**7月5日**

小组开会明确课程设计任务，并一起复习了计算机组成原理课本上硬布线控制器的原理，并复习了VHDL语言，为后几天的工作打好基础。小组成员进行了明确的分工，也制定了后几天每天的任务安排，分工之后，我们又熟悉了TEC-8试验台环境和EPM7128SLG84-15芯片的引脚与实验台信号之间的连接，并检查了实验台是否有问题。

**问题**：

1、还是不知道具体要实现什么功能。

2、对于课本上的时序逻辑存在不理解。

**解决**：

1. 小组成员进行了激烈的讨论，又将书看了几遍后初步了解了具体要实现什么功能，
2. 查阅了一些资料后，发现了标志位的重要性，也初步理解了标志位与节拍电位是如何配合完成时序逻辑的，小组对整个VHDL代码的大致框架有了初步构想。

**7月6日**

今天一大早，小组成员就一起来到了实验室，准备进行此次硬布线控制器的VHDL硬件描述语言代码的编写。虽然有了第一天对VHDL语言的复习，但是真正开始编写的时候，还是发现了任务的艰巨与复杂。于是在上午，我们再次对VHDL语言进行了深入的复习，直到下午2点，才开始了代码的正式编写。经过了将近4个小时的不懈努力，代码终于雏形初见。然而意料之中的是，代码出现了很多bug以及棘手的错误。

**问题**：

1. 将输入初始化之后，比如CLR <= '0' ;T3 <= '0' ;之后，出现了报错。
2. 在条件判断的时候，由于长期使用C++语言，并且对VHDL语言不熟悉，导致总是忘记打单引号。
3. 在case语句的最后，忘记加when others的情况，导致一直报错。

**解决：**

按照编译报错报文，将错误逐个找到，通过互联网搜索引擎，与VHDL课本教程，找到了问题的解决方法，并将部分问题解决，但是代码整体仍然没有通过编译，无法运行，只能7号继续进行代码的debug。

**7月7日**

第一天的挫败并没有打击组员们的积极性。同样是8点30分，大家又来到了实验室，继续VHDL语言的编写工作。有了第一天的经验，今天的编写工作变得相对容易了起来。大家对VHDL语言的熟悉程度也在不断加深。一上午的努力之后，我们完成了对代码的编写。下午1点30.，我们开始了用一个简单的微程序，通过TEC-8实验台对代码进行测试。在测试期间出现了好几个问题，但是在下午5点之前，所有的问题都得到了圆满的解决。

**问题1**、将SWC,SWB,SWA设置为001，进行写存储器操作的时候，时序意外的进入了W2，在W1时指示灯全部不亮，出错！

**解决**：仔细看了下代码，发现本来应该是W2=‘0’的条件写成了W2=‘1’

**问题2**、上面改成W2=‘1’之后，W1时指示灯仍然不亮。

**解决**：发现是在工程创建时出现了问题，重新建立了一个工程并重新编译了之后，问题得到解决。

**问题3**、在写寄存器的控制台操作中，出现了只能写R0与R1，R2和R3无法写入。

**解决**：原来是我们忽略了PPT中提到的FLAG信号的设置问题，导致了我们只能在两个CPU周期中操作而无法在四个CPU周期中操作。于是我们加入了中间信号ST0与SST0。ST0的值代表了此时所经历的CPU周期是第1还是第2个主状态周期。SST0的值表示此时是否进入了第二个主状态周期。

**7月8日**

代码完成之后，我们便马不停蹄的开始了程序的测试，我们上午对本学期计组实验课上老师给出的一个微程序（同时也是书上的一个程序）进行了测试。这次测试出乎意料的顺利，在写存储器，写寄存器之后，读出来的数据和当初的数据一致，但是，我们仍然不能放心，正好昨天老师给了一个用来测试的微程序，于是我们就拿这个程序继续测试，但是在这个测试之前，却出现了工程崩溃的问题，同时引脚也莫名其妙地被清空。中午吃过了饭之后，花了一个小时解决了这个问题，然后仔细的把这个微程序输进了TEC-8试验台，最后经过手动验证，发现答案是正确的。不过出于严谨考虑，我们花了一定时间对每一步的答案都进行了验证，发现答案均无误。

**问题1**：在进行第二次测试之前，出现了工程崩溃引脚图清空的问题。

**解决**：处于对软件的不够了解，最终原因并没有搞懂，但是我们把原有工程删掉，重新创建了工程并且pin了引脚图之后，问题得到了解决。

**7月11日**

经过了两天的休整，成员们纷纷恢复了元气，同时通过前几天的努力，基本要求已经完成了，实验报告也完成了整体的框架，不过还有一个大块，就是额外指令的添加，还没有完成。经过讨论，我们最终决定添加的指令为：OUTRd，XOR，XNOR，OR和MOV指令。通过参照数据通路图，我们设计出了这5条指令的时序图，添加到了原有的基本时序图里面。接着就是设计一个新的包括了这个几条新指令的微程序了。一开始我们打算设计一个全新的微程序，这个做起来却发现了其困难之处，组员灵机一动，决定在第二个微程序的OUT指令和停机指令之间添加这几条新指令，再手动算出结果和机器运算的结果做对比就行。这大大节省了实验的时间，在设计出了这个新的程序之后，我们赶紧对它进行了测试，结果很圆满，与手算结果完全一致！

下午的时候任务基本已经完成，只剩下对文档的完善和中断实验报告的完成。因为在复习计组期末考试的时候，对中断方面的内容做了很多功课，所以中断实验报告完成的很顺利，下午过完的时候，中断实验报告就已经完成。同时另外的组员也把文档进行了最后的完善，最终在11号下午，完成了本次小学期的全部内容。

**问题1**：在设计包含额外指令的微程序的时候出现了困难

**解决**：在第二个微程序的OUT指令和停机指令之间添加这几条新指令，再手动算出结果和机器运算的结果做对比就行。

# 团队成员个人感想

**周尧棋：**

此次计算机组成原理设计实验我主要负责的是VHDL代码的编写和调试的部分。VHDL语言是在数字逻辑课上学习的，由于使用得不多而且很久未使用过导致起初我对这个语言有些不熟练，出现了一系列的编译错误，比如将值赋为0或者1时没有加单引号，IF、ELSE语句结束后没有加END IF语句等等，经过漫长的找错后终于将代码做到了无差错。

其次，这次实验代码的主要难点是不同长度的指令周期的设计，尤其以四个节拍电位的设计为最难的部分。起初我们还不理解老师提示的标志位的作用，后来经过小组内讨论以及查询资料大概理解了将四个节拍电位分成两个两个节拍电位的主要思想，编写完代码后在调试时却仍然出现了不少问题，比如明明该进行W1操作时W2的灯却是亮的等等。在对代码反复检查及思考后我发现了问题并最后实现了功能。另外一个难点是如何实现从任意的PC地址开始执行程序，我想到了通过执行指令一开始修改PC的值来实现此功能，但仍然要使用到标志位。做什么事都是想着简单，但是实际操作起来却不那么简单。这段短短的代码在我们调试了一下午后仍然出现了问题，SBUS和LPC的灯一直亮不起来。在经过一个深夜的思考后，我发现了是END IF语句的位置放错了，最后在早上到达实验室调试时终于解决了问题实现了该功能。

通过这次实验，我深深地体会到了硬布线控制器控制信号的原理。这次实验的成功的关键在于对于数据通路的理解和对时序逻辑的理解。如果对于数据通路不理解，那么建立在数据通路基础上的各种操作也就无法实现；如果对时序逻辑不了解，那么你便不知道指令到哪步执行到了哪个阶段。总之一个CPU控制器的设计，需要许许多多知识的掺入，而不仅仅是对于硬布线控制器的理解。

这次实验我也体会到团队合作的重要性，没有团队的一起配合是不能完成一项出色的任务的。唯有大家一起讨论一起思考，方能将大家各自的优点发挥到最大化，才能完成此次实验的任务。

**徐灿：**

这次我主要负责了代码的编写以及程序调试文档完善等等。此次任务一开始是显得艰巨的，一方面我们对整个计算机的整体原理把握还不够深刻，另一方面对VHDL语言也不够熟悉，因此在一开始讨论的时候，我们都觉得这次会很困难。在第一天的复习以及仔细研究之后，对任务的目标终于有了深刻的认识，并且对VHDL语言也更加熟悉了，2天时间，大家完成了代码的编写和调试。调试过程是一个特别锻炼能力和磨炼意志的过程，记得有一天工程突然崩溃，引脚也突然清空，重新创好之后又出现了一大堆莫名其妙的bug，通过一步一步耐心的调试，才终于把这个问题解决，要是稍微泄气一点， 我想进度一定会大大延缓。

一点一点完成整个报告是一个很神奇的过程，一开始只有数据通路图，时序图等等，慢慢的，代码贴上去了，指令集贴上去了，硬连线控制器译码表贴上去了，每天结束之后，还加上了每天的工作日志，几天过后，我们的实验报告终于完善了，看到大家一起劳动的结晶，心里还是很激动的。

特别想说一点就是那个加分项目，也就是要求取值不从默认的00H开始，而是可以从任意地址开始。我们一开始就想对了问题的方向，也就是在swcba=000时候添加一个LPC的case，不过一下午调试一直出问题，也是令大家都沮丧不已。结果晚上躺床上，突然就想到了也许是end if位置的问题，更神奇的是组员周尧棋也同时想到了这个问题，到了最后一天到实验室，把代码一改，结果真就成功了，也是让我惊喜不已。

总的来说这次实验难度并不大，不过团队合作的重要性也得到了充分的体现。没有团队的配合，我们的工作效率一定会大打折扣。同时整个实验让我对计算机的整体有了更深的把握，锻炼了自己的动手动脑能力，可以说是很宝贵的一次经历。

**李凡：**

这次在TEC-8实验平台上进行硬布线控制器的常规CPU设计实验，我对计算机组成原理与系统结构有了进一步了解加深，同时也加深了对CPU各模块工作原理及相互联系的认识。首先在QuartusⅡ上我们小组进行了VHDL的编程工作，在这个过程中，我对一个CPU的基本输入项，基本输出项以及节拍信号w1,w2等有了更加直观地认知，在添加管脚以及导入程序时心情是激动的，仿佛一个可以在电脑中运行的CPU被制作出来一样。在试验阶段，基本操作如同微程序控制器实验一般，基本按照课本流程即可。在这次实验过后，我增强了对CPU工作过程的认知，使之更加清晰与立体化，同时，也对微程序控制器与硬布线控制器的的异同有了更加清楚的对比。

**廖竞鑫：**

本次实验无疑是困难重重，虽然之前有学过计算机组成原理的实验，但是都是小实验，而这一次是一个多人合作的小组实验-硬布线控制器。一开始几乎是毫无头绪，但是在努力回忆实验课老师所传授的知识和计算机组成原理课本上的注释之后，慢慢摸索，竟有了一丝丝头绪，然后不断的进行小组讨论，每个人都在努力为实验做出贡献。每当有了BUG或者难题的时候，我们小组进行讨论，查阅各种资料，也慢慢有所进展。这次实验让我们对计算机也有了一点点更深刻的理解，有了一丝丝微不足道的见解。更加意识到了团队合作的重要性。

**邓精：**

通过本次试验，让我们对VHDL语言和硬布线控制器有了进一步的了解。起初我们拿到这个试验任务时有一点茫然，但是我们很快的明确了分工，首先我们开始回顾了之前学习的VHDL语言，之后我们先完成了编译工作，在过程遇到了一些困难，比如一些不太明白的编译错误，我们通过网上慢慢的查资料，最终完成了编译工作。在完成了编译工作之后，我们开始着手试验台TEC-8的操作，我们先回顾了一下之前微程序控制器的操作，之后开始硬布线控制器的操作，先明确了两种控制器的区别，过程比较顺利。之后我们测试了几个简单的程序。之后我们测试老师给的程序，过程也比较顺利。我们开始着手附加的功能，从任意PC值开始执行指令，我们先开始进行代码的编译，但遇到了一些问题，经过了一天的调整，最终实现了这个功能。

在此次实验中，让我们学会了团队合作，增强了动手能力。

# 5.总结

这次的TEC-8实验台模拟计算机硬连线控制器设计中，我们使用了分指令译码逻辑设计的方法，通过对于指令的译码，根据时序信号，产生相应的控制信号输出。

在实验中，我们小组团结合作，大家各有分工，而且都尽心尽力完成。在代码编写、调试中，我们遇到了很多问题，比如对于VHDL语言一些语法的疑惑以及对于实验台环境的不熟悉，我们都通过谈论或者是亲自操作实践解决了。

大家都收获颇丰，为以后的学习打下了基础。