|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **课程设计**  **名称** | 硬布线控制器的CPU设计 | | **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** | |
| **课**  **程**  **设**  **计**  **内**  **容** | 简要介绍课程设计的主要内容，包括课程设计教学目的、基本内容、实验方法和团队分工等   1. 教学目的 2. 融会贯通计算机组成原理课程各章教学内容，通过知识的综合运用，加深对CPU各模块工作原理及相互联系的认识 3. 掌握硬连线控制器的设计方法 4. 学习运用大容量可编程器件开发技术，掌握设计和调试的基本步骤和方法，体会ISP技术的优点 5. 培养科学研究能力，取得设计与调试的实践经验。   .   1. 基本实验内容   完成由硬布线控制器控制的简单指令（CLA，ADD，STA，JMP，NOP）的实现。计算机的数据通路的控制器由硬联线控制器来完成，CPU从存储器中取出一条指令到指令结束为一个指令周期，由于各种指令的操作功能不同，有的指令简单，有的指令复杂，所以各种指令的指令周期是不相同的。一个指令周期由若干个机器周期组成，每个机器周期又由若干个时钟周期组成。   1. 实验操作方式   画出控制台指令及机器指令流程图。根据流程图作出微操作控制信号的译码与时序分布表，然后用逻辑表达式表示出每个信号。由逻辑表达式写出VHDL语言源代码。对程序进行编译，下载到芯片。连线，调试。   1. 小组成员相应任务 | | | | | |
| **学生**  **课程设计**  **报告**  （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：  2010年 月 日 | | | | | |

# 实验设计报告

1. **实验设备及环境**

TEC-5计算机组成原理试验系统 一台

Pentium 3以上微型计算机 一台

双踪示波器 一台

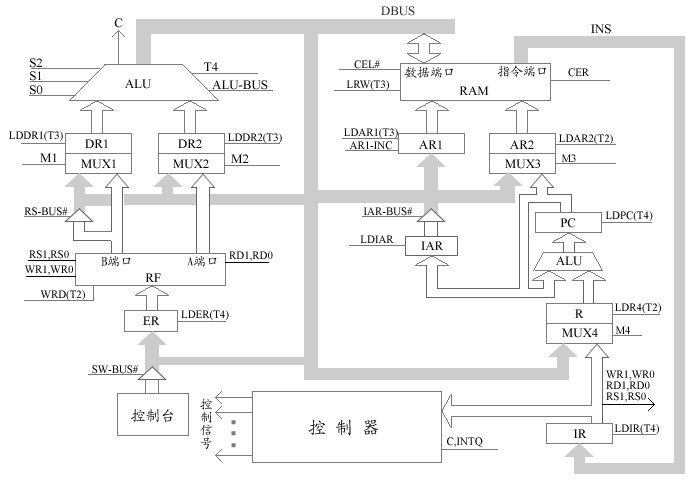
逻辑测试笔 一支

实验环境：Windows 下 Isp vmSystem、 Isp Level软件;

1. **设计原理**

（1）指令系统和数据通路

采用教学实验用模型计算机相同的指令系统。数据通路和微程序控制器方案相同。



（2）硬布线控制器的基本原理

硬布线控制器的基本原理是，每个微操作控制信号S是一系列输入量的逻辑函数，即用组合辑电路来实现，

S = f（Im，Mi，Tk，Bj）

其中Im是机器指令操作码译码器的输出信号，Mi是节拍电位信号，Tk是节拍脉冲信号，Bj是状态条件判断信号。

在TEC-5试验系统中，节拍电位信号Tk（T1 -- T4）已经直接输送给数据通路；因机器指令系统比较简单，操作码只有4位，省去操作码译码器，用Im直接作为操作码译码器输出，即指令寄存器的IR7—IR4信号。Mi是时序模块的节拍电位信号，例如W3—W1。Bj的信号包括：来自数据通路中运算器ALU的进位信号C；来自控制台的开关信号SWC、SWB、SWA；其他信号。

其中C、SWC、SWB和SWA信号在微程序控制器中同样存在。

每个控制信号都是上述输入信号的逻辑表达式，因此可以用组合逻辑结构电路。只要对所有控制信号都设计出逻辑函数表达式，这个硬布线控制器的方案也就确定了。

（3）指令周期流程图设计

微程序控制器的控制信号以微指令周期为时间单位，硬布线控制器以节拍电位（CPU周期）为时间单位，两者本质上是一样的，1个节拍电位时间和1个微指令周期都是从节拍脉冲T1的上升沿到T4的下降沿的一段时间。在微程序控制器流程图中，1个执行框代表1个微指令周期，而在硬布线控制流程图中，1个执行框就代表1个节拍电位时间。

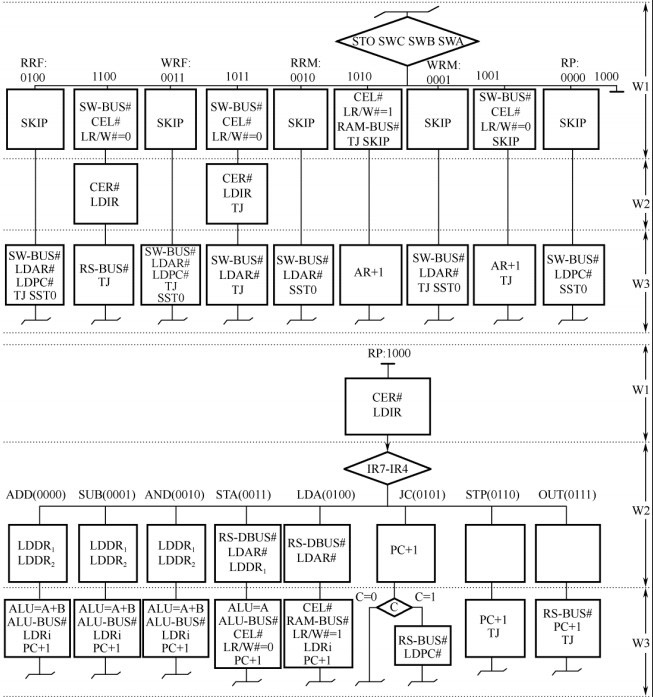
（4）执行一条机器指令的节拍电位数

在本实验中，选用3个节拍电位对大多数指令就够用了，所以节拍电位发生器产生3个电位信号（W1—W3）。

对于所需节拍电位时间较多的指令，将其执行化为占用两条（或者更多）机器指令的节拍。例如执行一条指令可以占用W1、W2、W3、W1、W2、W3六拍时间。为了区分一条指令的两个不同阶段，可以用某些特殊的寄存器标志将其区分，例如FLAG=0时，表示该指令执行第一个W1、W2、W3；FLAG=1时，表示该指令执行第二个W1、W2、W3。上文中提到的Bj包括其他信号，FLAG就可以认为是一个其他信号，由于某些控制台指令（例如读寄存器RRF）只需要4拍，占用2条机器指令周期（6拍）则浪费了时间。为了减少浪费，在时序电路中加入了一个控制信号SKIP

的输入，该信号的作用是使节拍发生器在任意下直接跳到最后一拍（W3）。这样，设计控制器流程时，在所需节拍较少的指令流程适当的位置使SKIP控制信号有效，多余的节拍就可以跳过，从而提高了性能。

在硬布线控制器中，控制台指令的流程图设计与机器指令的流程图设计相类似。

****

（5）组合逻辑译码表

设计出了硬布线指令流程图，就可以据此设计出译码逻辑电路。先根据流程图列出译码表，作为逻辑设计的依据。与微程序表的设计相似，译码表的内容也包括横向设计与纵向设计。流程图中横向为一拍（W1、W2、W3等），纵向为一条指令，而译码逻辑是针对每一个控制信号的，因此在译码表中，横向变成了一个信号。每行中的内容表示某个控制信号在个指令中有效的条件，主要是节拍电位和节拍脉冲信号指令操作码译码器输出、执行结果标志信号等。

根据译码表可以写出每个控制信号的逻辑表达式，这个表达式就是它所在的行各乘积项相加（逻辑或）。使用可编程逻辑器件（ISP、FPGA），只要将表达式直接写成VHDL源程序，编译软件会自动完成电路优化的工作。（译表如下）

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 状  态  指  令 | RRF1 | RRF2 | WRF1 | WRF2 | RRM1 | RRM2 | WRM1 | WRM2 | RP1 | RP2 |
| S3 |  |  |  |  |  |  |  |  |  |  |
| S2 |  |  |  |  |  |  |  |  |  |  |
| S1 |  |  |  |  |  |  |  |  |  |  |
| S0 |  |  |  |  |  |  |  |  |  |  |
| M |  |  |  |  |  |  |  |  |  |  |
| Cn\_0 |  |  |  |  |  |  |  |  |  |  |
| LR\_W |  |  |  |  |  | W1 |  |  |  |  |
| CEL\_0 |  | W1 |  | W1 |  | W1 |  | W1 |  |  |
| CER\_0 |  | W2 |  | W2 |  |  |  |  |  | W1 |
| RAM\_BUS\_0 |  |  |  |  |  | W1 |  |  |  |  |
| ALU\_BUS\_0 |  |  |  |  |  |  |  |  |  |  |
| RS\_BUS\_0 |  | W3 |  |  |  |  |  |  |  |  |
| SW\_BUS\_0 | W3 | W1 | W3 | W1,W3 | W3 |  | W3 | W1 | W3 |  |
| LDRi |  |  |  |  |  |  |  |  |  |  |
| LDDR2 |  |  |  |  |  |  |  |  |  |  |
| LDDR1 |  |  |  |  |  |  |  |  |  |  |
| LDAR\_0 | W3 |  | W3 | W3 | W3 |  | W3 |  |  |  |
| AR\_ADD |  |  |  |  |  | W3 |  | W3 |  |  |
| LDPC\_0 | W3 |  | W3 |  |  |  |  |  | W3 |  |
| PC\_ADD |  |  |  |  |  |  |  |  |  |  |
| LDIR |  | W2 |  | W2 |  |  |  |  |  | W1 |
| TJ | W3 | W3 | W3 | W2 |  | W1 | W3 | W3 |  |  |
| SKIP | W1 |  | W1 |  | W1 | W1 | W1 | W1 | W1 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 状  态  指  令 | RP2 | | | | | | | |
| ADD | SUB | AND | STA | LDA | JC | STP | OUT |
| S3 | W3 |  | W3 |  |  |  |  |  |
| S2 |  | W3 |  |  |  |  |  |  |
| S1 |  | W3 | W3 |  |  |  |  |  |
| S0 | W3 |  | W3 |  |  |  |  |  |
| M |  |  | W3 |  |  |  |  |  |
| Cn\_0 |  | W3 |  |  |  |  |  |  |
| LR\_W |  |  |  |  | W3 |  |  |  |
| CEL\_0 |  |  |  | W3 | W3 |  |  |  |
| CER\_0 |  |  |  |  |  |  |  |  |
| RAM\_BUS\_0 |  |  |  |  | W3 |  |  |  |
| ALU\_BUS\_0 | W3 | W3 | W3 | W3 |  |  |  |  |
| RS\_BUS\_0 |  |  |  | W2 | W2 | W3(C=1) |  | W3 |
| SW\_BUS\_0 |  |  |  |  |  |  |  |  |
| LDRi | W3 | W3 | W3 |  | W3 |  |  |  |
| LDDR2 | W2 | W2 | W2 |  |  |  |  |  |
| LDDR1 | W2 | W2 | W2 | W2 |  |  |  |  |
| LDAR\_0 |  |  |  | W2 | W2 |  |  |  |
| AR\_ADD |  |  |  |  |  |  |  |  |
| LDPC\_0 |  |  |  |  |  | W3(C=1) |  |  |
| PC\_ADD | W3 | W3 | W3 | W3 | W3 |  | W3 | W3 |
| LDIR |  |  |  |  |  |  |  |  |
| TJ |  |  |  |  |  |  | W3 | W3 |
| SKIP |  |  |  |  |  |  |  |  |

（6）调试与总测试

ISP计数设计的硬布线控制器，其分调试完全是软件模拟的向量测试。但应注意，向量测试方程的设计应全面，尽量覆盖所有的可能性，避免将错漏带到总调试中。另外要注意两个细节问题：

1、测试软件要求测试状态连续，即上一方程的终结状态作为下一方程的初始状态，不能中断；

2、如果方程的输入向量组中有某些影响输出向量的项缺失，测试仍能进行，但缺失项将会以随机值影响输出。

分调试完成后，可将控制器与数据通路相连接，进行全机总调试。

总调试的第一步是检查硬布线控制流程，以单拍（DP）方式执行指令。进行的顺序也是先执行控制台指令，然后执行机器指令。当全部控制流程图检查完毕后，如果数据通路的执行部件（运算器、存储器等）功能正确，就算总调试第一步完成。

第二步在内存中装入包括有全部指令系统的一段程序和有关数据，进一步可采用单步（DP）方式或连续方式执行，以验证机器执行指令的正确性。

第三步编写一段表演程序，令机器运行。

第四步运验收程序。如果通过，则设计和调试就告完成。

1. **硬布线控制器VHDL代码**

LIBRARY IEEE;

USE IEEE.STD\_LOGIC\_1164.ALL;

ENTITY hardwired\_controller IS

PORT(CLR,W1,W2,W3,IR4,IR5,IR6,IR7,SWC,SWB,SWA,C: IN STD\_LOGIC;

S3,S2,S1,S0,M,Cn,LR\_W,CEL,RAM\_BUS,ALU\_BUS,RS\_BUS,CER,LDRi,

SW\_BUS,LDDR1,LDDR2,LDAR,AR\_ADD,LDPC,PC\_ADD,LDIR,TJ,SKIP: OUT STD\_LOGIC);

attribute LOC:string;

ATTRIBUTE LOC OF CEL :SIGNAL IS "P3";

ATTRIBUTE LOC OF IR4 :SIGNAL IS "P4";

ATTRIBUTE LOC OF S1 :SIGNAL IS "P5";

ATTRIBUTE LOC OF LR\_W :SIGNAL IS "P6";

ATTRIBUTE LOC OF RS\_BUS :SIGNAL IS "P7";

ATTRIBUTE LOC OF RAM\_BUS :SIGNAL IS "P8";

ATTRIBUTE LOC OF W3 :SIGNAL IS "P9";

ATTRIBUTE LOC OF LDDR2 :SIGNAL IS "P10";

ATTRIBUTE LOC OF LDPC :SIGNAL IS "P12";

ATTRIBUTE LOC OF PC\_ADD :SIGNAL IS "P13";

ATTRIBUTE LOC OF SWC :SIGNAL IS "P15";

ATTRIBUTE LOC OF W2 :SIGNAL IS "P16";

ATTRIBUTE LOC OF IR5 :SIGNAL IS "P18";

ATTRIBUTE LOC OF CER :SIGNAL IS "P26";

ATTRIBUTE LOC OF TJ :SIGNAL IS "P27";

ATTRIBUTE LOC OF LDIR :SIGNAL IS "P28";

ATTRIBUTE LOC OF LDAR :SIGNAL IS "P29";

ATTRIBUTE LOC OF S3 :SIGNAL IS "P30";

ATTRIBUTE LOC OF SKIP :SIGNAL IS "P31";

ATTRIBUTE LOC OF LDDR1 :SIGNAL IS "P32";

ATTRIBUTE LOC OF IR7 :SIGNAL IS "P33";

ATTRIBUTE LOC OF S0 :SIGNAL IS "P34";

ATTRIBUTE LOC OF IR6 :SIGNAL IS "P35";

ATTRIBUTE LOC OF ALU\_BUS :SIGNAL IS "P37";

ATTRIBUTE LOC OF C :SIGNAL IS "P51";

ATTRIBUTE LOC OF SWB :SIGNAL IS "P57";

ATTRIBUTE LOC OF CLR :SIGNAL IS "P59";

ATTRIBUTE LOC OF W1 :SIGNAL IS "P60";

ATTRIBUTE LOC OF S2 :SIGNAL IS "P68";

ATTRIBUTE LOC OF SWA :SIGNAL IS "P69";

ATTRIBUTE LOC OF SW\_BUS :SIGNAL IS "P70";

ATTRIBUTE LOC OF AR\_ADD :SIGNAL IS "P71";

ATTRIBUTE LOC OF LDRI :SIGNAL IS "P73";

ATTRIBUTE LOC OF M:SIGNAL IS "P74";

ATTRIBUTE LOC OF CN :SIGNAL IS "P75";

END hardwired\_controller;

ARCHITECTURE behavior OF hardwired\_controller IS

SIGNAL RRF1,WRF1,RRM1,WRM1,RP1,RRF2,WRF2,RRM2,WRM2,RP2: STD\_LOGIC;

SIGNAL ADD,SUB,AND1,STA,LDA,JC,STP,OUT1: STD\_LOGIC;

SIGNAL STO:STD\_LOGIC:='0';

SIGNAL SSTO:STD\_LOGIC:='0';

BEGIN

PROCESS(CLR,W1,W3,SSTO)

BEGIN

IF(CLR='1') THEN --清零

SSTO<='0';

STO<='0';

ELSIF((SSTO='0') AND (W3='1') AND (W3'EVENT)) THEN --第一个周期ok

SSTO<='1';

ElSIF((W1'EVENT AND W1='1') AND SSTO='1') THEN --第二个周期ok

STO<='1';

END IF;

END PROCESS;

RRF1 <= (NOT STO) AND SWC AND (NOT SWB) AND (NOT SWA);

RRF2 <= STO AND SWC AND (NOT SWB)AND (NOT SWA);

WRF1 <= (NOT STO) AND (NOT SWC) AND SWB AND SWA;

WRF2 <= STO AND (NOT SWC) AND SWB AND SWA;

RRM1 <= (NOT STO) AND (NOT SWC) AND SWB AND (NOT SWA);

RRM2 <= STO AND (NOT SWC) AND SWB AND (NOT SWA);

WRM1 <= (NOT STO) AND(NOT SWC) AND (NOT SWB) AND SWA;

WRM2 <= STO AND(NOT SWC) AND (NOT SWB) AND SWA;

RP1 <= (NOT STO) AND (NOT SWC) AND (NOT SWB) AND (NOT SWA);

RP2 <= STO AND (NOT SWC) AND (NOT SWB) AND (NOT SWA);

ADD <= RP2 AND (NOT IR7) AND (NOT IR6) AND (NOT IR5) AND (NOT IR4);

SUB <= RP2 AND (NOT IR7) AND (NOT IR6) AND (NOT IR5) AND (IR4);

AND1 <= RP2 AND (NOT IR7) AND (NOT IR6) AND (IR5) AND (NOT IR4);

STA <= RP2 AND (NOT IR7) AND (NOT IR6) AND (IR5) AND (IR4);

LDA <= RP2 AND (NOT IR7) AND (IR6) AND (NOT IR5) AND (NOT IR4);

JC <= RP2 AND (NOT IR7) AND (IR6) AND (NOT IR5) AND (IR4);

STP <= RP2 AND (NOT IR7) AND (IR6) AND (IR5) AND (NOT IR4);

OUT1 <= RP2 AND (NOT IR7) AND (IR6) AND (IR5) AND (IR4);

SKIP <= (W1 AND (NOT STO)) OR (W1 AND (RRM2 OR WRM2));

LDAR <= NOT ((W3 AND (NOT STO) AND (NOT RP1)) OR (W2 AND (STA OR LDA)));

LDPC <= NOT ((W3 AND (RRF1 OR WRF1 OR RP1)) OR (C AND W3 AND JC));

TJ <= (W3 AND (RRF1 OR WRF1 OR WRM1)) OR ((W3 AND RRF2) OR ((W2 OR W3) AND WRF2) OR (W1 AND RRM2) OR (W3 AND WRM2))OR (W3 AND (STP OR OUT1));

CEL <= NOT ((W1 AND (RRF2 OR WRF2 OR RRM2 OR WRM2)) OR (W3 AND (STA OR LDA)));

LR\_W <= (W1 AND RRM2) OR (W3 AND LDA);

CER <= NOT ((W2 AND (RRF2 OR WRF2)) OR (W1 AND RP2));

LDIR <= (W2 AND (RRF2 OR WRF2)) OR (W1 AND RP2);

LDRi <= W3 AND (WRF2 OR ADD OR SUB OR AND1 OR LDA);

LDDR1 <= W2 AND (ADD OR SUB OR AND1 OR STA);

LDDR2 <= W2 AND (ADD OR SUB OR AND1);

RS\_BUS <= NOT ((W3 AND RRF2) OR (W3 AND (OUT1 OR (C AND JC))) OR (W2 AND (STA OR LDA)));

SW\_BUS <= NOT ((W3 AND (NOT STO)) OR ((W1 AND RRF2) OR ((W1 OR W3) AND WRF2) OR (W1 AND WRM2)));

RAM\_BUS <= NOT ((W1 AND RRM2) OR (W3 AND LDA));

ALU\_BUS <= NOT (W3 AND (ADD OR SUB OR AND1 OR STA));

AR\_ADD <= W3 AND (RRM2 OR WRM2);

PC\_ADD <= (W3 AND (ADD OR SUB OR AND1 OR STA OR LDA OR STP OR OUT1)) OR (W2 AND JC);

M <= AND1 AND W3;

S0 <= W3 AND (AND1 OR ADD);

S1 <= W3 AND (AND1 OR SUB);

S2 <= W3 AND SUB;

S3 <= W3 AND (AND1 OR ADD);

Cn <= NOT (SUB AND W3);

附：

操作控制台工作方式：

|  |  |  |  |
| --- | --- | --- | --- |
| SWC | SWB | SWA | 操作 |
| 0 | 0 | 0 | 启动程序（PR） |
| 0 | 0 | 1 | 写存储器（WRM） |
| 0 | 1 | 0 | 读存储器（RRM） |
| 0 | 1 | 1 | 写寄存器（WRF） |
| 1 | 0 | 0 | 读寄存器（RRF） |

机器指令系统：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 名称 | 助记符 | 功能 | 指令格式 | | |
| IR7 IR6 IR5 IR4 | IR3 IR2 | IR1 IR0 |
| 加法 | ADD Rd,Rs | Rd+Rs🡪Rd | 0 0 0 0 | RS1、RS0 | RD1、RD0 |
| 减法 | SUB Rd,Rs | Rd-Rs🡪Rd | 0 0 0 1 | RS1、RS0 | RD1、RD0 |
| 逻辑与 | AND Rd,Rs | Rd & Rs🡪Rd | 0 0 1 0 | RS1、RS0 | RD1、RD0 |
| 存数 | STA Rd,[Rs] | Rd🡪[Rs] | 0 0 1 1 | RS1、RS0 | RD1、RD0 |
| 取数 | LDA Rd,[Rs] | [Rs]🡪Rd | 0 1 0 0 | RS1、RS0 | RD1、RD0 |
| 条件转移 | JC R3 | 若C=1则R3🡪PC | 0 1 0 1 | 1 1 | × × |
| 停机 | STP | 暂停执行 | 0 1 1 0 | × × | × × |
| 输出 | OUT Rs | Rs🡪DBUS | 0 1 1 1 | RS1、RS0 | × × |

（2）输入、输出和其他信号的确定

输入信号：

IR7,R6,IR5,IR4 --操作码译码输出

W3,W2,W1 --时序模块的节拍电位信号

SWC,SWB,SWA --控制台开关信号

C --ALU进位信号

CLR --清零信号

输出信号：

S3,S2,S1,S0 --选择ALU的运算类型

M --选择ALU的运算模式

Cn# --ALU最低位的+1信号

LR\_W# --双端口存储器左端口读写控制信号

CEL# --双端口存储器左端口使能信号

CER# --双端口存储器右端口使能信号

RAM\_BUS# --存储器数据送数据总线DBUS信号

ALU\_BUS# --ALU输出三态门使能信号

RS\_BUS# --通用寄存器右端口三态门使能信号

SW\_BUS# --控制台输出三态门使能信号

LDRi --双端口寄存器堆写入信号

LDDR2 --对操作数寄存器DR2进行加载的控制信号

LDDR1 --对操作数寄存器DR1进行加载的控制信号

LDAR# --对地址寄存器AR进行加载的控制信号

AR\_ADD --对AR进行加1操作的电位控制信号

LDPC# --对程序计数器PC进行加载的控制信号

PC\_ADD --对PC进行加1操作的电位控制信号

LDIR --对指令寄存器进行加载的控制信号

TJ --停机命令,关闭时序信号

SKIP --跳转命令,在任意状态下直接跳转到最后1拍

控制器内部信号：

RRF1,RRF2,WRF1,WRF2,RRM1,RRM2,WRM1,WRM2,RP1,RP2 --表示不同的操作，后面的数字表示不同的周期

ADD,SUB,AND,STA,LDA,JC,STP,OUT --表示不同的命令

FLAG; --标志第二周期开始

SSTO; --标志第一周期结束

1. **调试数据及结果**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RAM地址 | 程序 | 机器码 | RAM地址 | 数据1 |
| 00H | LDA R0，[R2] | 0100 10 00 | 60H | 24H |
| 01H | LDA R1，[R3] | 0100 11 01 | 61H | 83H |
| 02H | ADD R0，R1 | 0000 01 00 |  |  |
| 03H | JC R2 | 0101 10 00 | 寄存器 | 数据1 |
| 04H | AND R1，R0 | 0010 00 01 | R2 | 60H |
| 05H | SUB R0，R3 | 0001 11 00 | R3 | 61H |
| 06H | STA R0，[R1] | 0011 01 00 |  |  |
| 07H | OUT R0 | 0111 00 00 |  |  |
| 08H | OUT R1 | 0111 01 00 |  |  |
| 09H | STP | 0110 00 00 |  |  |

运算结果：

R0= 46 H ； R1=83H ； R2=60H ；R3= 61H ； RAM[R1]= 46H

实验程序及测试数据

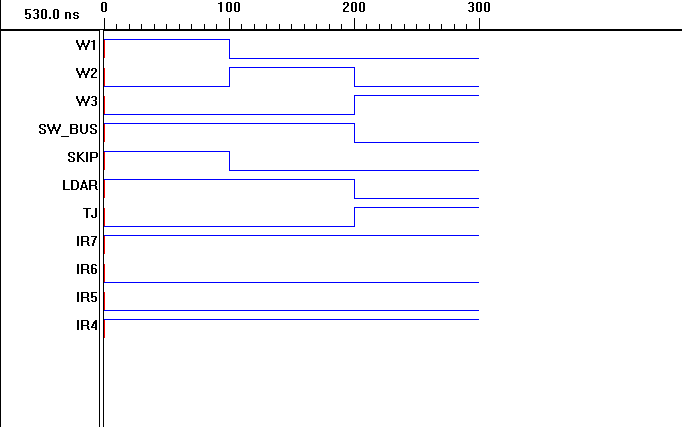
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RAM地址 | 程序 | 机器码 | RAM地址 | 数据2 |
| 00H | LDA R0，[R2] | 0100 10 00 | 60H | 60H |
| 01H | LDA R1，[R3] | 0100 11 01 | 61H | F8H |
| 02H | ADD R0，R1 | 0000 01 00 |  |  |
| 03H | JC R2 | 0101 10 00 | 寄存器 | 数据2 |
| 04H | AND R1，R0 | 0010 00 01 | R2 | 60H |
| 05H | SUB R0，R3 | 0001 11 00 | R3 | 61H |
| 06H | STA R0，[R1] | 0011 01 00 |  |  |
| 07H | OUT R0 | 0111 00 00 |  |  |
| 08H | OUT R1 | 0111 01 00 |  |  |
| 09H | STP | 0110 00 00 |  |  |

运算结果：

R0=58H ； R1=F8H ； R2= 60H ；R3=61H ； RAM[R1]= 0H

1. **波形图及相应代码**

本图为WRM（0001）指令所对应时序下的相应波形图



相关代码：

module hardwired\_controller;

c,l,m,n,x,y,z = .c.,.x.,.x.,.x.,.x.,.x.,.x.;

IR7,IR6,IR5,IR4,SWC,SWB,SWA,W1,W2,W3,CEL,SW\_BUS,LDAR,AR\_ADD,TJ,SKIP PIN;

TEST\_VECTORS

([IR7,IR6,IR5,IR4,SWC,SWB,SWA,W1,W2,W3]->[CEL,SW\_BUS,LDAR,AR\_ADD,TJ,SKIP])

[ 1,0,0,1,0,0,1,1,0,0]->[ l,m,n,x,y,z ];

[ 1,0,0,1,0,0,1,0,1,0]->[ l,m,n,x,y,z ];

[ 1,0,0,1,0,0,1,0,0,1]->[ l,m,n,x,y,z ];

END

1. **实验心得及相关问题的思考**

邓舒姗——

从最开始的无从下手到最后的全部细节都很清楚，我在这次计组实验中收获了许多，最开始我的认识就是跟计组小实验最后一个微程序的CPU差不多，一步一步地走程序，后来到了实验室，我们经过讨论，一步一步地列真值表，编VHDL程序，在编程过程中，查阅了许多资料，从来没有编过计组的程序，而距离VHDL的学习也过了很久了，大家都有些遗忘，看了一些书上的例子，终于出来了结果，编译，烧到芯片里面，全英文的软件用起来的确有点头晕，很多参数不知道是什么意思，还好有大家而不是自己。然后就开始了伟大的接线和查错，这个过程非常痛苦，真的不知道是哪里错了，接线的时候一根线一根线的试，接好以后发现数据不对，又拿逻辑笔一点一点地测高低电平，我们不光是小组讨论，还跟其他组同学学习，问老师问题，终于，我们做出了正确结果，作为我们702这个实验室第一个做出正确结果的小组，真的十分开心，但是我们并不止于这一点点成就，而是想对程序继续进行改进，我们又拿isp软件画了波形，对这个过程有了更直观的理解，从VHDL到试验台结果到波形，一步一步地努力，是我们小组四个人所取得的小小成功，感谢这次计组小学期的锻炼，不仅学到知识，增进了大家的友谊，也让我们看到了集体的强大*。*

文黎力——

这次课程设计，我们对硬件设计的步骤有了深刻地了解。当我们拿到任务之后，运用我们掌握的知识对这个任务进行分析，然后划分几个阶段并明确每个阶段的任务如下：首先理 解题目，画出流程图；其次进行状态编码并予以化简；再次进行编程，下载；然后是调试，查错，验收；最后完成实验报告。

实验的过程并不是一帆风顺的，主要在于这个实验的关键，就是流程图的设计。但是，我们并没有气馁，而是向老师请教，并且和同学讨论，还应用了一些以前实验中没有用过 的控制信号和方法，最终解决了实验中出现的种种问题，也从其中获得的不少从课本学习和日常实验中得不到的东西。通过这次实验我们巩固了《计算机组成原理》课程中重要部分的知识，对于计算机系统中的指令执行过程和指令周期的概念有了更深入的了解，编程的过程还巩固了我们在《数字逻辑与数字系统》课程中学习的可编程语言部分的知识。同时课程设计也提高了我们的动手能力，亲自调试自己的程序，为了检验错误存在的原因用逻辑笔不厌其烦地一个管脚一个管脚的测试，每进行一次修改就要进行一次甚至更多次的数据输入来验证实验结果。

随着实验的不断深入同学们的配合越来越默契，共同探讨，互补合作，使得大家在知识和能力上都有了很大的提高。总而言之，经过几天的试验我们收获颇丰，同时我们还要感谢实验老师对我们的悉心指导，帮我们分析解决了许多棘手的问题，也为我们拓展思路提供了很好的指导。

鄢舒源——

这次小学期，我们学习了如何设计硬布线控制器。在这次实验中，我们遇到了很多问题，最初不熟悉软件，导致我们下载程序到控制器失败，经过学习发现我们没有没有合理设置设备，选择了错误类型的设备。在这次实验中，我主要负责的是管脚的接线与实际的操作，小组一起参加讨论设计控制器的程序，在实际操作过程中，我们遇到了这次实验困扰我们比较久的问题，在第一次连线后，我们能够成功的读写RAM,但是当我们写寄存器时，遇到了问题，写入寄存器和读出寄存器的值是不一样的。我们小组讨论，首先根据操作的流程图，根据W1,W2,W3的操作控制指令，逐个检查信号电平，经过测试，在每个脉冲的信号都是正确的，我们很迷惑，无法找到问题所在。我们小组经过讨论，决定更换寄存器号，检验是否开始选择的寄存器是坏的，我们选择了4个不同的数据，分别打入了0，1，2，3号寄存器，接着分别读出0，1，2，3号寄存器里的值，发现寄存器中0和1的值总是相同，2和3的值总是相同，根据二进制编码，我们猜测可能是00和01，10和11中的地位线电平不正确。

但是当我们用逻辑笔测试的时候，却与我们想的不一样。无奈之下，我们只能将线全部拆下，然后重新连接。为了确保我们的线是没问题的，我们将每一根线都做了测试，发现其中有两根线是接触不良的，我们换了两根线之后，我们猜测可能是检查IR0和IR1信号时，由于用逻辑笔给了压力，使得导线接触好了，从而测试电平时的信号是正确的。吸取了第一次连线的经验，将线理的比较清楚。但是在连好线以后，发现还是读写不正确，这时候，我们发现我们在连好线后的操作中，脉冲信号打错了一拍，导致写入不正确，经改正，我 们成功的实现了寄存器读写和RAM的读写，之后的操作都比较顺利。

在这次实验中，我们学到了很多东西，学会了如何设计简单的硬布线控制器，学会了如果分析问题和解决问题，学会了如果作为一个团队去完成一个实验，当我们解决问题的那 一刻，我们都很开心，大家都很享受发现问题并开动脑筋去猜想问题，分析问题，解决问题的过程。

张译恬——

回顾刚看到小学期这个硬布线控制器的任务的时候,感觉这个实验既熟悉又陌生,它非常像我们曾经做过的微程序控制器,但这次实验却不涉及实验台上的控制器部分，给了我们很大的自由发挥空间。

***实验过程遇到的困难和解决方法***

VHDL程序是根据译码表写出来的，主要都是信号之间的逻辑关系。刚开始的时候我们仅仅把这些逻辑关系变为VHDL代码，很快问题就浮现了，由于ispLEVER软件分配管脚是随机的，每当程序有了修改，就不得不大规模地拔线重插，十分耗时，此时我们学习了attribute语句，固定管脚，效果十分不错。

当程序已经下载到芯片中时，我们的插线工作开始，为了确保每根线的质量，每根用在实验室的线都经过三重检验——第一，连在开关和指示灯之间能确保指示灯亮；第二，轻拍两端和线，确保灯不闪（即线接触良好）；第三，电线连接到电路板上，用逻辑测试笔测两端电平是否一致(在电路板插孔正常).

为了查出为什么寄存器一直写与读不一致，我们花费了大量时间排查。先是对照流程图，用逻辑测试笔分别测量在对应时序下，控制信号是否有效，为此我们将W1、W2、W3分别接到指示灯上，可以直观看出运行到哪个CPU周期。确认无误后，我们怀疑寄存器堆内部有问题，于是分别给R0~R3写入不同值，试验结果是R0与R1同，R3与R4同。从这样的结果，我们推测WR0永远为高电平。但是，只给R0赋值却又正确了。硬件的障碍排除了，一下子排错工作陷入困境。此时，我和同学交换角色，由我看着流程图操作，竟然成功读入寄存器！经过反复比对，发现我们对SKIP的理解有出入，SKIP应该是占一个周期且在该周期数据通路上信号并没有打入。

寄存器读写问题解决后，一切豁然开朗，试验进行得异常顺利，把老师给的验收程序运行一遍便成功了，那一瞬间真的很有成就感，觉得自己总往实验室跑得时间值了，大家几天的理解这么快变成了成果，这就是团队的力量吧！

由于我们组在该实验室里完成算是最顺利的，我们纷纷到各个小组帮同学排错，别的小组问题可以归纳为两种，寄存器写入总为零，或是CPU时序总停留在W1、W2，导致这些问题也可以归于两种，一个是接线的问题，另一个是程序问题，要么程序设置高电平清零，但接CLR#，要么是在信号的逻辑表达式中没有加入时序因素。

***实验思考***

最后我们小组摸索着做时序的波形，一开始我们按着教程，做的是逻辑表达式的波形，幸亏老师及时纠正，我们就改成CPU周期作为自定义输入信号，将控制信号作为输出观察，终于得到与流程图相符的波形图。

老师建议我们改变书上的设计图,我认为对于从数据通路上重新打开SW\_BUS有困难,按照书上设计,数据是在不同的CPU周期输入的,直接并拍有可能会引起数据冲突。转而参考微程序控制器，但是并没有找到可以让硬布线控制器的流程进行循环的资料，因而不太可行。所以初步的想法是RRF由三拍组成，第一拍执行SW-BUS#、LDAR#、LDPC#、TJ、SST0，第二拍执行SW-BUS#、CEL#、LR/W#=0，第三拍执行CER#、LDIR、RS-BUS、TJ；同理，WRF、 RRM、WRM均可以撤掉SKIP操作，将无数据输入的操作与下一个有数据输入输出操作合并。 至于RP，本来就是由三拍组成，可以保留。有了新的流程图，画出译码表，后续操作基本 上一样。

通过这次实验，我控制器和数据通路又有了更深刻地认识，CPU的神秘面纱也在这次操作中揭开，与同学们交流与合作的过程也是十分惬意。同时也十分感谢老师的指点和学习资料，确保实验完成的质量，也驱除许多关于操作上的疑云。

相关问题的思考：

Q:一个W里面包含了几个t1～t4？

A:4个

Q:W是怎么产生的？

A：由时钟脉冲产生