目录

1 总体概述

1.1 设计任务描述

概述：本次课程设计的任务是在TEC-8实验平台提供的设备资源上开发一个硬连线控制器，该控制器能够完成基本控制台操作，并能够执行规定的指令。

具体描述：设计一个硬连线控制器，和TEC-8模型计算机的数据通路结合在一起，构成一个完整的CPU，该CPU要求能够完成控制台操作：启动程序运行、读存储器、写存储器、读寄存器和写寄存器。并且能够执行下图中的指令，完成规定的指令功能。

名称	助记符	功 能	指令格式		
			IR(7-4)	IR(3-2)	IR(1-0)
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←@ + offset	0111	offset	
Z条件转移	JZ addr	如果Z=1，则 PC←@ + offset	1000	offset	
无条件转移	JMP [Rd]	PC←Rd	1001	Rd	XX
停机	STOP	暂停运行	1110	XX	XX

表13.1中，XX代表随意值。Rs代表源寄存器号，Rd代表目的寄存器号。在条件转移指令中，@代表当前PC的值，offset是一个4位的有符号数，第3位是符号位，0代表正数，1代表负数。**注意：@不是当前指令的PC值，是当前指令的PC值加1。**

在Quartus Ⅱ下对硬连线控制器对设计方案进行编程和编译将编译后的硬连线控制器下载到TEC-8实验台上的ISP器件EPM7128中去，使EPM7128成为一个硬连线控制器。根据指令系统，编写检测硬连线控制器正确性的测试程序，并用测试程序对硬布线控制器在单拍方式下进行调试，直到成功。

在基础要求之外，实验还需要对指令集进行扩充，至少三条指令。

1.2 实验环境描述

本次实验使用的是Altera公司开发的软件Quartus Ⅱ 9.0，这款EDA简化了控制器的设计，加快了整体开发速度，为本次实验提供了便利。

具体实验环境描述：

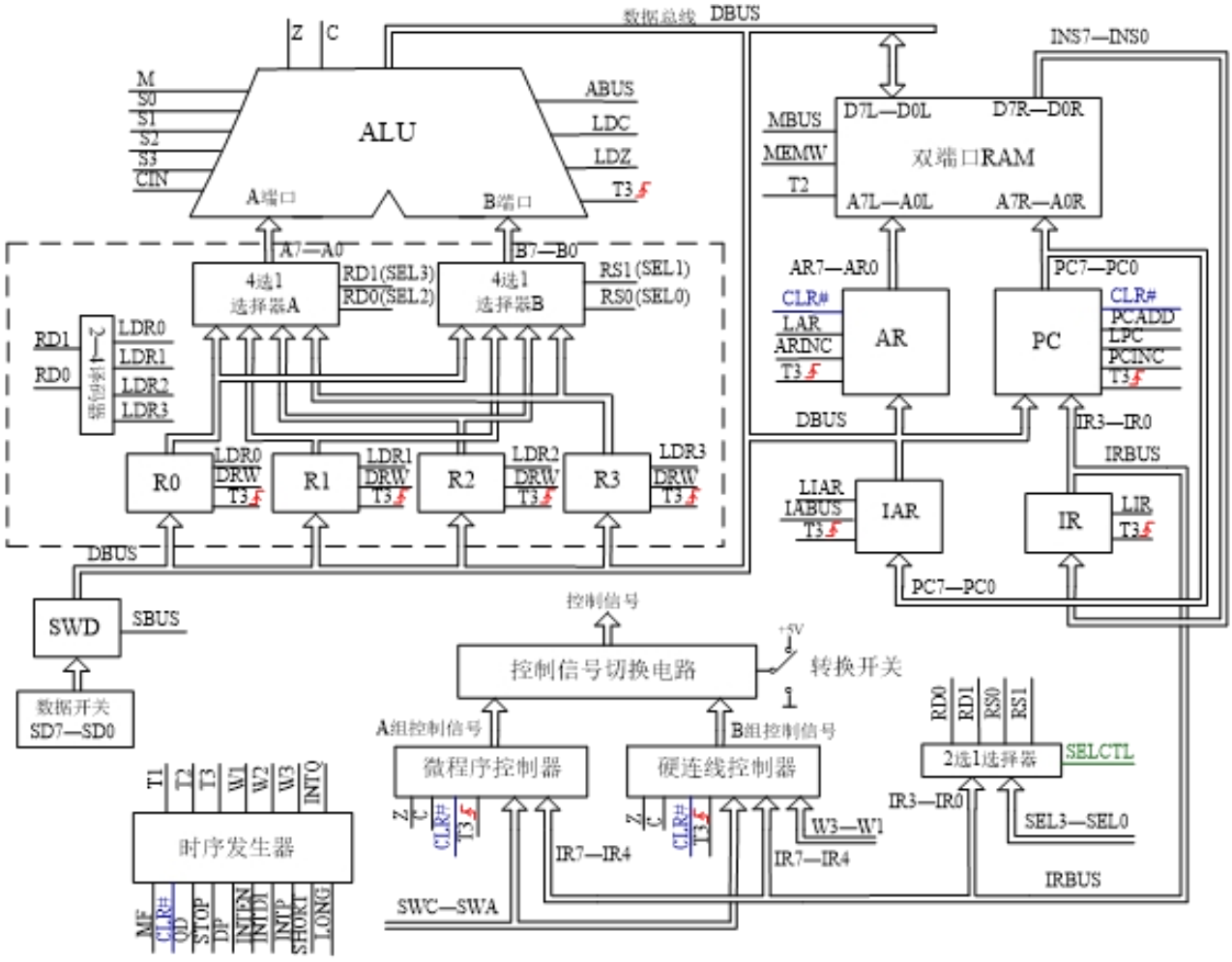
① 开发环境：Win10系统，Quartus Ⅱ 9.0 的PC机。

② 调试工具：TEC-8实验板。

③ 实验平台：Quartus Ⅱ 9.0，TEC-8实验板。

由于课程设计依赖TEC-8实验系统，故给出TEC-8试验系统的数据通路。

TEC-8数据通路如下图所示：

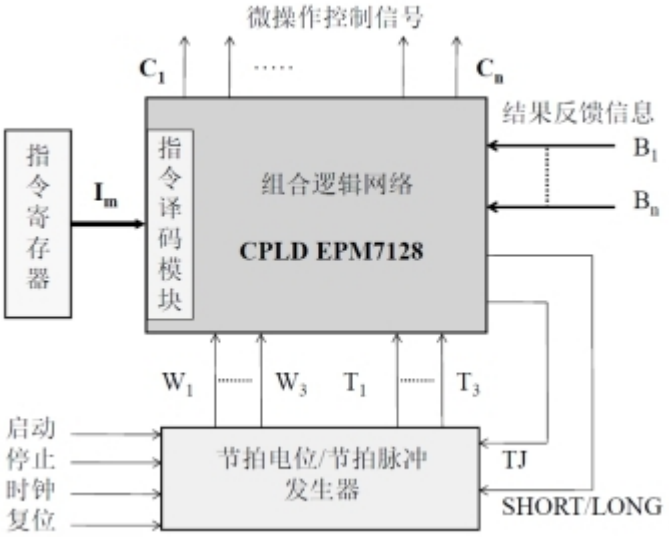


2 详细设计描述

2.1 硬连线控制器原理介绍

2.1.1 硬连线控制器原理

首先给出硬连线控制器的逻辑模块图:



逻辑网络的输入信号来源有三个:

(1)来自指令操码译码器的输出Im

(2)来自执行部件的反馈信息Bj

(3)来自时序产生器的时序信号，包括节拍电位信号M和节拍脉冲信号T

在硬布线控制器中，某一微操作控制信号由布尔代数表达式描述的输出函数产生。针对每一个控制信号Cz，其函数表达式为:

$$C_x = f(I_m,\ M_i,\ T_k,\ B_j)$$

由于TEC-8实验系统有控制台操作，而控制台操作可以看作是具有特殊功能的复杂指令，因此SWC、SWB、SWA也可当作Im的一部分。Mi是时序发生器产生的节拍电位信号W1~W3，我们可利用SHORT、LONG控制信号来调整并产生不同操作所需要的节拍数；Bj包括ALU所产生的进位信号C，以及结果为0的判断信号Z。

2.1.2 流水工作原理

模型机在非流水顺序执行指令时，会在执行完一条指令之后再继续从存储器中取出下一条指令，而经过我们小组的讨论与研究，在流水的情况下，在一条指令的执行周期中，实际上可以同时将下一条指令加载到IR中，并将PC加1。这一原理体现在流程图中便是所有指令均没有第一个用于取指的节拍，而是在每条指令的执行周期中都加入LIR和PCINC信号，将双端口存储器中读出的指令打入指令寄存器，同时将PC指向下一条指令，完成取指的工作。

在不使用流水方式的系统中，执行一条指令所需要的的周期数为2或3，这在流水方式中也是一样，只不过我们将不同指令中可以同时操作的部分放在了一个周期里进行。也就是说，每一条指令都是在上一条指令执行的最后一个周期取指令，分析指令。之后PC指针+1。上图中“-1”周期是抽象等价出来的，因为按下CLR之后，IR会变为0000，这也就相当于在整个程序运行前自带一个NOP指令，其作用是分析第一条指令，之后PC指向第二条指令。

2.2 含有流水线及指令扩展及优化的硬连线控制器

2.2.1扩展功能讲解

根据题目要求，以及扩指要求（我组扩指共四条：NOP空指令，OUT输出，DEC减1，SHL左移），我组本次实验的完整指令格式如下所示：

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←@+offset	0111	offset	
Z条件转移	JZ addr	如果Z=1，则 PC←@+offset	1000	offset	
无条件转移	JMP [Rd]	PC←Rd	1001	Rd	XX
输出	OUT Rs	DBUS←Rs	1010	XX	Rs
空操作	NOP	--	0000	XX	XX
左移	SHL Rd	Rd←Rd<<1	1100	Rd	XX
减1	DEC Rd	Rd←Rd-1	1011	Rd	XX
停机	STP	停止运行	1110	XX	XX

（标红为扩充指令集）

我组在指令扩展以外，还实现了更改PC指针功能（具体可见2.2.2流程图）

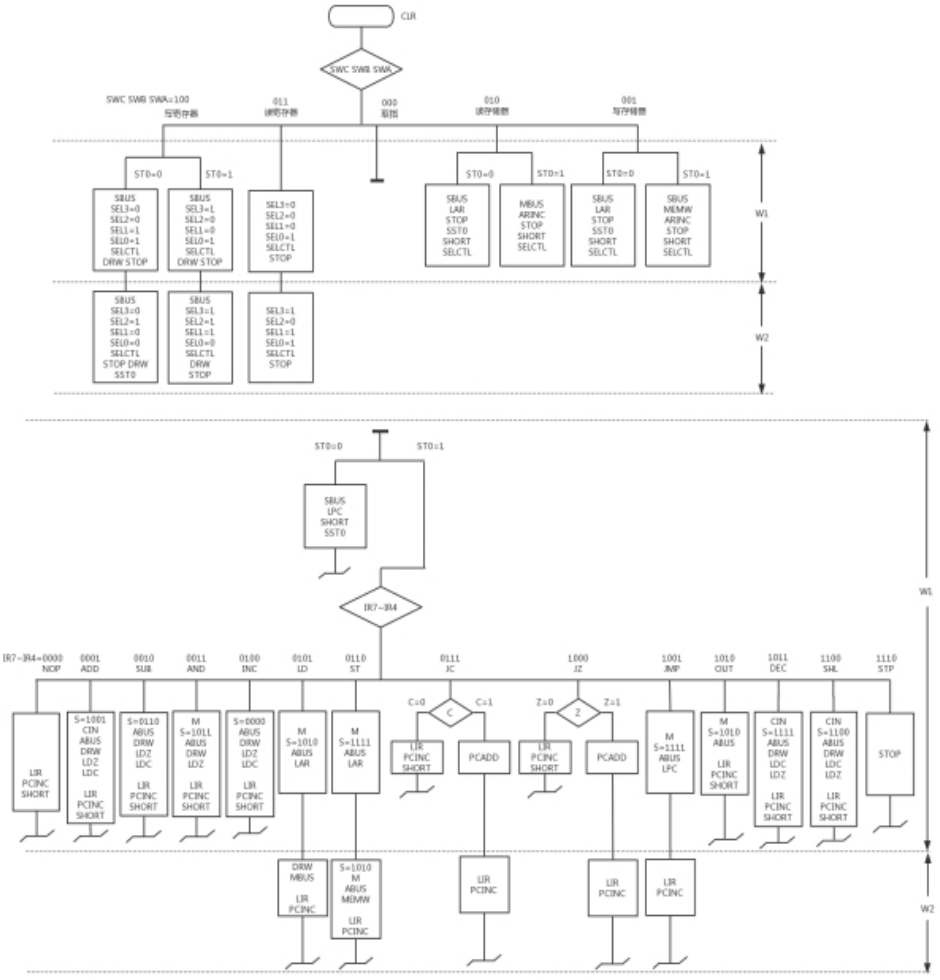
PC指针更改功能的实现依赖于中间信号ST0与SST0，在进入控制台执行指令的功能之前，ST0的初始值为0；在进入执行指令（000）功能后，数据开关的数值将通过DBUS被打入PC，修改PC指针的值，从而达到指定程序入口的功能。

不难发现，PC指针更改功能其原理基本上与读写存储器一致。且SST0的值被修改为1；而在进入程序入口后，ST0的值将被修改为1，此后机器将会进入正常的指令周期进行循环，不会产生别的影响。

2.2.2流程图

硬连线控制器以W1，W2，W3节拍电位为时间单位。

下图为我组设计的指令周期流程图:



2.2.4组合逻辑译码表

根据周期流程图，可写出如下对应组合逻辑译码表:

指令IR	ADD	SUB	AND	INC	LD	ST	JC	JZ	JMP	STP	DEC	SHL	OUT	NOP
LIR	W1	W1	W1	W1	W2	W2	W2+(~C)*W1	W2+(~Z)*W1	W2		W1	W1	W1	W1*ST0
PCINC	W1	W1	W1	W1	W2	W2	W2+(~C)*W1	W2+(~Z)*W1	W2		W1	W1	W1	W1*ST0
S3	W1		W1		W1	W1+W2			W1		W1	W1	W1	
S2		W1				W1			W1		W1	W1		
S1		W1	W1		W1	W1+W2			W1		W1		W1	
S0	W1		W1			W1			W1		W1			
CIN	W1										W1	W1		
ABUS	W1	W1	W1	W1	W1	W1+W2			W1		W1	W1	W1	
DRW	W1	W1	W1	W1	W2						W1	W1		
LDZ	W1	W1	W1	W1							W1	W1		
LDC	W1	W1		W1							W1	W1		
M			W1		W1	W1+W2			W1				W1	
LAR					W1	W1								
LONG					W2									
MBUS					W2									
MEMW						W2								
PCADD							C*W1	Z*W1						
LPC									W1					
STOP										W1				
SHORT	W1	W1	W1	W1			(~C)*W1	(~Z)*W1		W1	W1	W1	W1	W1

2.2.5信号逻辑表达式

根据组合逻辑译码表可写出各个信号对应的逻辑表达式:

LIR=PCINC=W1*(ADD+SUB+AND+INC+(~C)*JC+(~Z)*JZ+DEC+SHL+OUT)+W2*(ST+JC+JZ+JMP+ LD)+W1*NOP*ST0

S3 = W1*(ADD+AND+LD+ST+JMP+DEC+SHL+OUT)+W2*ST

S2 = W1*(SUB+JMP+ST+DEC+SHL)

S1 = W1*(SUB+AND+LD+ST+JMP+DEC+OUT)+W2*ST

S0 = W1*(ADD+AND+ST+JMP+DEC)

CIN = W1*(ADD+DEC+SHL)

ABUS = W1*(ADD+SUB+AND+INC+LD+ST+JMP+DEC+SHL+OUT)+W2*ST

DRW = W1*(ADD+SUB+AND+INC+DEC+SHL)+W2*LD

LDZ = W1*(ADD+SUB+AND+INC+DEC+SHL)

LDC = W1*(ADD+SUB+INC+DEC+SHL)

M = W1*(AND+LD+ST+JMP+OUT)+W2*ST

LAR = W1*(LD+ST)

LONG = MBUS=W2*LD

MEMW = W2*ST

PCADD = W1*(C*JC+Z*JZ)

LPC = W1*JMP

STOP = W1*STP

SHORT=W1*(NOP+ADD+SUB+AND+INC+(~C)*JC+(~Z)*JZ+STP+DEC+SHL+OUT)

2.2.6卡诺图优化示例

本次课程设计的硬连线控制器的VHDL代码中使用的是CASE分支语句来控制机器判断并执行不同的指令。因此代码的执行速度会受到一定的影响。

基于此原因，我们将原本用CASE条件分支实现的指令判断选择，改为使用与或逻辑表达式直接进行表示，并借助卡诺图进行了化简，进一步提升了代码运行速度。

下面以S3信号逻辑表达式的化简过程为例。

由上图可得：

S3 = W1*(~IR6*IR4+~IR7*~IR5*IR4+IR7*IR6*~IR4)+W2*(~IR7*IR6*IR5*~IR4)

3 调试过程记录

3.1测试程序

我组设计的程序如下表所示：

地址	程序指令	机器码	16进制
00H	NOP	0000 0000	00
01H	LD R0,[R2]	0101 0010	52
02H	INC R2	0100 1000	48
03H	LD R1,[R2]	0101 0110	56
04H	ADD R0,R1	0001 0001	11
05H	JC 07H	0111 0001	71
06H	STP	1110 0000	E0

07H	STA R0,[R3]	0110 1100	6A
08H	JMP [R0]	1001 0000	90

寄存器初值设定：R0:00H R1:00H R2:40H R3:42H

存储器初值设定：[40H] = A0H [41H] = 64H

测试后寄存器结果：R0:68H R1:64H R2:41H R3:42H

测试后存储器结果：[42H] = 04H

3.2调试问题记录及解决

1.【JC,JZ,JMP的周期问题】

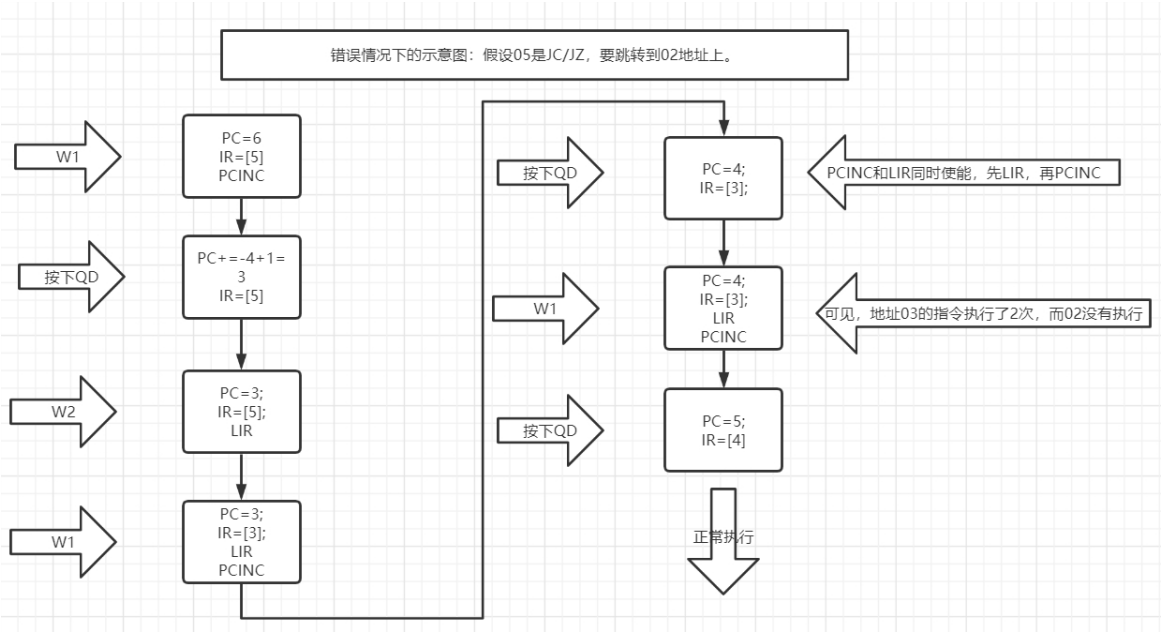
首在JC,JZ,JMP中，由于不存在DBUS冲突现象，所以在这三个指令是单周期还是双周期的问题产生的讨论。经过分析与测试，我们发现这些指令不能直接使用一个节拍完成功能。其中转移指令JMP、JC、JZ在跳转时需使用两个节拍来完成，第一个节拍实现分支转移的功能，第二个节拍实现IR的装载与PC的自增，在不跳转时仅需使用一个节拍即可完成。

2.【吞指令问题】：

首先吞指令问题有发生在LD,ST和JC及JZ指令上。在我们的调试过程中，LD及ST并没有出现吞指令的现象，只出现了JC和JZ指令上。但是原理上，LD,ST,JC,JZ吞指的原因是一样的，由于这四个指令都是双周期指令，有W1和W2两个周期，其中关键就是必须在最后一个周期实行IR装载和PC自增，所以LD，ST的吞指令是因为PC和IR在W1即使能，而JC,JZ,JMP的吞指令解决必须根据跳转结果判断，使得在不跳转时在W1完成IR和PC的改变，在跳转时则在W2自增PC和装载IR。其中的原理我们猜测如下:

流水线执行JC和JZ最多需要两个周期W1，W2，最开始设置的时候为PCINC<=W1.但是实际应该写为PCINC<=W2.因为将偏移量写入PC是在W1周期，如果此时PC就+1了，会导致PC指向跳转后指令的下一条指令。这样的话，在W2的时候LIR就会将指向指令的下一条指令写入IR中，如果这个时候没有PCINC会导致跳转后的指令的下一条指令执行两次而不执行本应执行的指令。有PCINC的话也是从跳转后指令的下一条指令开始执行。这是因为流水方式下LIR和PCINC应同时执行。除了刚开始执行的指令，所有指令执行中IR中指令应该是PC-1对应指令，如果LIR之后PC不+1，那么会导致当前指令执行两次。正确的解:

PCINC<=W2,LIR<=W2.这样先跳转，之后加载指令，之后PCINC，结果就正确了。（PCINC<=W1的情况流程如下，不执行将要跳转的指令，反而执行两次将要跳转指令的下一条指令）



3.【ST0及SST0理解问题】：

只SST0和ST0都是用来指示阶段的信号，比如写内存以及修改PC指针+执行程序这两个步骤。先送地址再存内存/执行指令，可以说当进入这个阶段后，先执行一个阶段1再重复若干个阶段2以达到目的，而ST0及SST0则是区分这些阶段信号

4.【NOP指令驱动程序问题】：

不需要NOP指令，经过大量的测试得知，在TEC-8实验系统中，当按下CLR键后，IR的内容将会被复位为00H，故不论运行什么程序，第一条被执行的机器指令总是NOP，因此不用考虑在程序的起始位置是否需要通过额外添加NOP指令来驱动流水线的运作。

4 实验总结

4.1优点及待改进点

4.1.1我组硬连线控制器优点

1.解决了JCJZ的吞指令问题，运行准确率很高。能够保证所有符合本控制器指令的程序都可以在本控制器上完整正常运行，不会出现执行错误的情况，保证执行结果执行顺序并不会吞指令和重复执行。

2.使用卡诺图来优化指令执行的微信号输出，加快了运行速度。在作出when-case这种易于理解便于看懂执行逻辑的代码后，我们继续进行优化，将各个微信号的组合逻辑时序逻辑表达式求出，加快了控制器的执行速度，也更加符合硬布线控制器的要求。

3.采用了流水加快执行速度。由于双端口存储器的存在，可以同时取指令和存取读取内存，给了实现流水的条件，于是采用了流水线技术，近似加速比为2，加快了程序的执行速度。

4.增加了数条指令以及提供修改PC指针的功能，便于适应各种程序以及修改程序在内存中的存储位置，利于程序的移植和内存空间的分配。

4.1.2我组控制器不足及待改进之处

1. 增加指令困难，这是硬布线逻辑控制器的先天性不足，一旦要加入新指令，需要重写很多信号的逻辑表达式，相对于微程序控制器非常麻烦。

2.指令种类少，没有中断等重要过程指令，受到指令格式位数限制，我们只能设计16条指令，而由于硬件设备的原因，无法实现中断这一常用操作。