《计算机组成与设计》
课程设计报告

简单模型机设计

学堂计机 18 | 张芮睿 | 201800301072 | 2020.5.13

目录

总述 2

实验目的 2

实验环境 2

实现指令组 2

简单模型机设计—微程序实现 3

设计目的 3

总体设计流程 3

拟定指令系统 3

确定总体结构 4

关键部件设计： 6

控制方式 6

编制指令流程及书写微程序 9

调试 11

模型及设计—硬布线实现 12

设计目标 12

总体结构 12

关键部件设计： 13

各部件的设计： 13

RAM 中编写代码 15

调试 16

问题分析 17

实验感悟 17

附录 17

微程序实现的模型机： 17

硬布线实现的模型机： 21

总述

实验目的

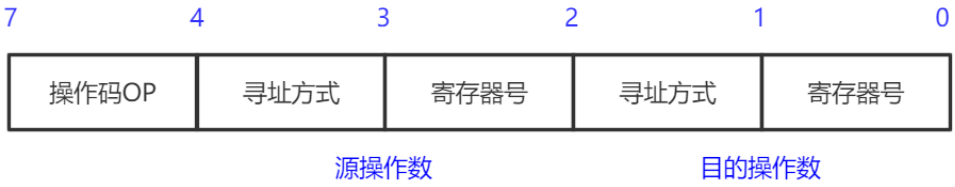
通过该课程设计的学习，总结计算机组成原理课程的学习内容，运用计算机原理知识，设计一台模型机，从而巩固课堂知识、深化学习内容、完成教学大纲要求，学好这门专业基础课。

实验环境

Quartus II 13.1, Win10 系统，FPGA 主板

实现指令组

简单模型机设计实现了基本的加减，逻辑乘，逻辑加，直传，加一，乘等指令。



指令：	对应的代码（16 进制）
MOV1 05#,R0	18 05
MOV2 03#,R1	29 03
MOV3 R1,(F0#)	47 F0

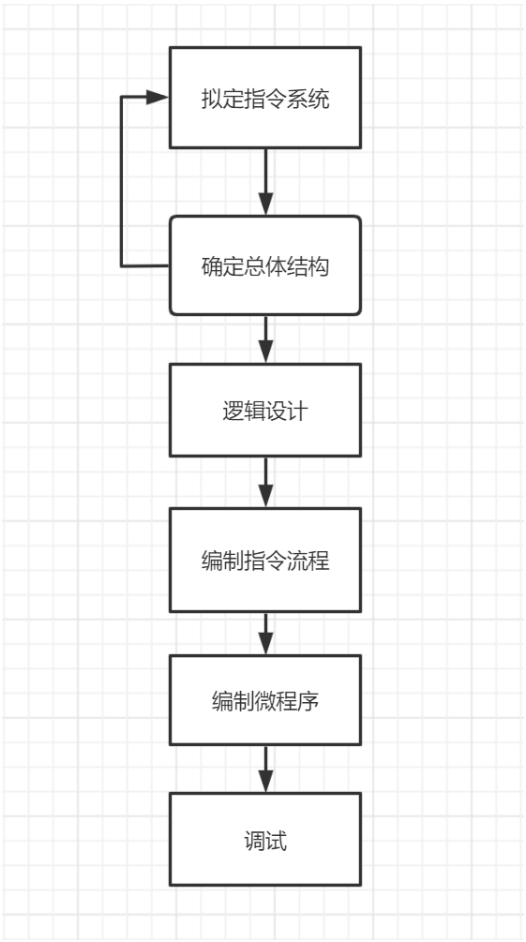
其他的实际操作在后续的微程序和硬布线实现中具体说明。（这上面的三条指令是下面两种实现方式通用的不会变。具体的寻址方式在下面的微程序实现部分中说明。）

简单模型机设计—微程序实现

设计目的

本次设计要完成一个基于微程序设计的简单模型机。其中 ALU 部分由 74181 芯片和 74182 芯片来设计。

总体设计流程



拟定指令系统

1. 基本字长

   模型机基本字长定位 8 位，存储容量为 256*8 位。

   指令格式：

2. **指令**类型

   指令类型有三类：单操作数指令、**双操作数指令和无操作数指令。操作**码 OP **共四位，最多可以定**义 16 **条指令**。

3. 寻址方式

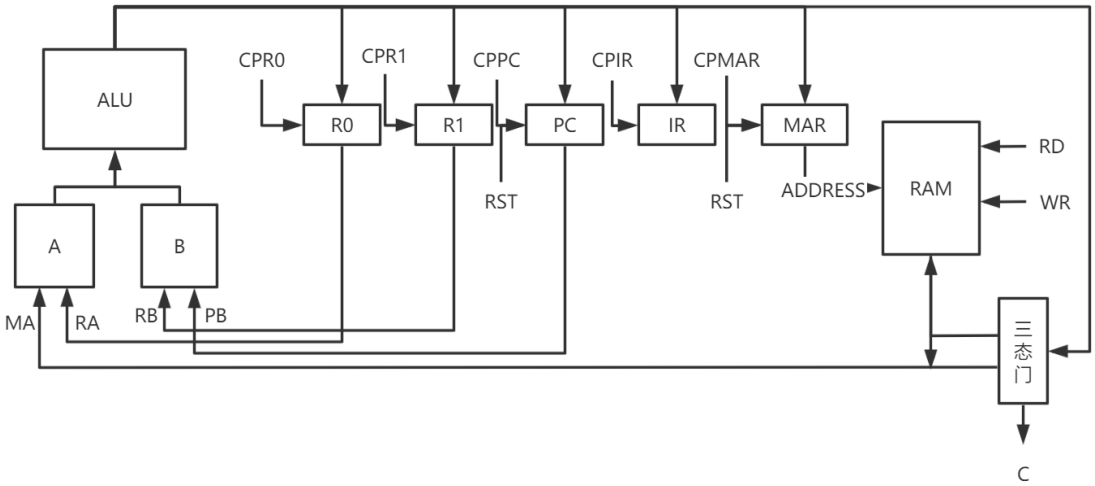
   当寻址方式位为 0，**是寄存器**寻址，**操作数在指定的寄存器中，相**应的寄存器号为 0 **是** R0，为 1 **是寄存器** R1；

   当寻址方式位为 1 时，寻址方式位和寄存器号位组合，**其中**：

   10：**是立即数**寻址，**操作数在指令的下一个**单元；

   11：**是直接**寻址，**操作数地址在指令的下一个**单元。

确定总体结构



1．**寄存器**组设置：

   R0、R1 为通用寄存器，8 **位**；

   IR 为指令寄存器，8 **位**；

PC 程序计数器，8 位；

MAR 为地址寄存器，8 位。

2．运算器 ALU 设置

加法、减法、逻辑运算采用两片 74181 芯片实现。

乘法操作采用一个 74284 芯片和一个 74285 芯片组合实现。

3．选择器设置

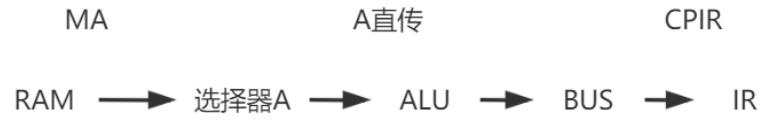
连入 A 选择器的数据来源是 RAM 的读出数据和 Ro 寄存器的数据。

连入 B 选择器的数据来源是 PC 的数据和 R1 的数据。

4．数据通路

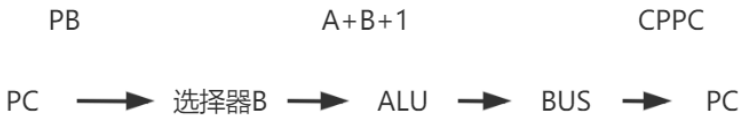
模型机的数据通路是以总线为基础，以 CPU 为核心构成的。

5．取指令



6．送指令地址

7．指令计数器+1



8．Ro→R1

9．R1→RAM

关键部件设计：

具体设计电路见附录。

总体结构中，RAM 是 Quartus II 中直接设计好的。

1. ALU 的逻辑设计

   加减乘除逻辑运算由两片 74181 组成。扩充的乘法实现是由采用一个 74284 芯片和一个 74285 芯片组合实现。

2. 寄存器的设计
   a) 结构中 R0、R1 通用寄存器，可存放操作数或结果、中间结果。
   b) 在 CPRi 的作用下接收总线的数据送入寄存器，输出连入选择器。
   c) 指令寄存器 IR 结构同通用寄存器
   d) 结构中 MAR 地址寄存器是一个带复位信号的八位寄存器，PC 加 1 是通过加法器实现的。复位信号 RST 的作用是有复位信号时，计数器 PC 清零。
3. 三态门的设计

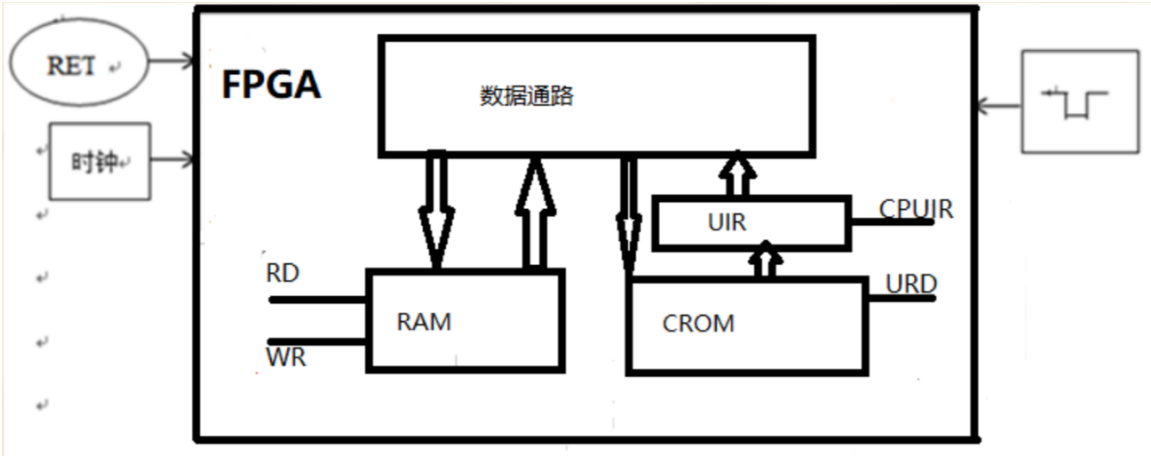
   采用 8 个 tri 原件实现，当控制输入端为高电平是直传，低电平时为高阻态。

4. 部件之间的连接

   由系统结构图可看出，部件之间的连接是采用以 CPU 为中心的总线连接方式。运算器的输出通过总线连接到所有寄存器和存储器的输入端，除指令寄存器 IR 和地址寄存器 MAR 的输出端外，其他部件的输出端分别送入选择器 A 和选择器 B.

控制方式

控制方式有两种，硬布线逻辑电路方式和微程序方式。先考虑微程序实现方式。控制命令是确定信息的流向，不同的数据通路需要不同的控制命令。涉及到了许多控制命令例如 CPR0、CPMAR、MA、RB 等等，这些控制命令由以下内容，主要由控制存储器 ROM3#、 ROM2#、ROM1#、微指令寄存器 IR23-16、 IR15-8、 IR7-0 构成。



P 脉冲的低电平用做控制存储器读命令μRD

P 脉冲的上升边沿将读出的微指令送 μIR

CU 设计原理图见附录。

1. 微指令格式

微指令字长 24 位即 μIR23 ~ μIR0。

2. 微指令字段定义

ALU 控制：

μIR22·μIR21·μIR20μIR19·μIR18μIR17·μIR16

Multiply M S3 S2 S1 S0 C0

三态门控制：

μIR6

0 高阻态 使 C = 1

1 三态门使能 使 C = 0

停机控制：

μIR3

0 G=0，运行

1 G=1，停机

A 选择器控制：

μIR15·μ IR14

0 0 备用

0 1 RA

1 0 MA

1 1 备用

B 选择器控制：

μIR13· μIR12

0 0 备用

0 1 PB
1 0 RB
1 1 备用

输出分配：

μIR11· μIR10· μIR9

0 0 0 备用
0 0 1 CPR0
0 1 0 CPR1
0 1 1 CPPC
1 0 0 CPIR
1 0 1 CPMAR
1 1 0 备用
1 1 1 备用

存储器读写控制： μIR5· μIR4

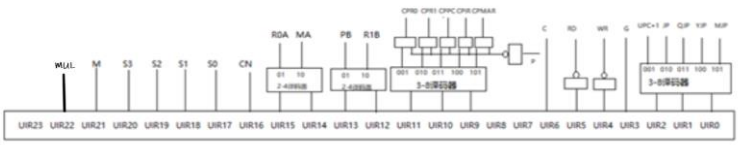
1 0 RD
0 1 WR

后继微地址形成方式：

μIR2· μIR1· μIR0

0 0 0 备用
0 0 1 μ PC + 1 顺序执行
0 1 0 JP 无条件转移，地址由 IR15-8 提供。
0 1 1 QJP 高四位按操码转移，低 4 位为 0。
1 0 0 YJP 给定高 4 位低 4 位按源寻址方式转移。
1 0 1 MJP 给定高 4 位低 4 位按目寻址方式转移。
1 1 0 备用
1 1 1 备用

3. 微命令形成逻辑

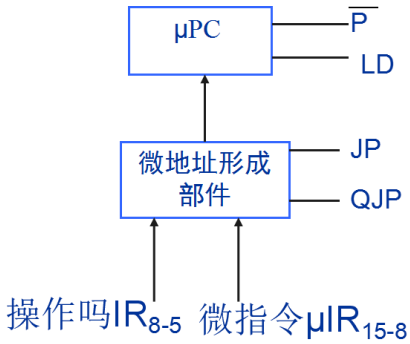


4. 后继微地址产生逻辑

为简单起见只选三种后继微地址生成方式

即增量方式、无条件转移方式、按操作码转移方式。

其结构框图如下图所示。



当 LD = 1 时，微程序计数 μPC 执行加 1 操作。

当 LD = 0 时且 JP = 1 时，无条件转移，有微指令的中八位提供转移地址。

当 LD = 0 时且 QJP = 1 时，按操作码转移。

功能：多路选择器

当 JP=1，QJP=0 时，Y7Y6Y5Y4Y3Y2Y1Y0 = μIR15 μIR14 μIR13 μIR12 μIR11 μIR10 μIR9 μIR8

当 JP=0，QJP=1 时，Y7Y6Y5Y4Y3Y2Y1Y0 = IR7 IR6 IR5 IR4 0000

链接时， Y7Y6Y5Y4Y3Y2Y1Y0 连接 μPC 的 D7~D0, μPC 的 RE 接高电平 vcc。

实现过程中，μPC 和微地址形成部件的具体电路图如附录图中所示。

编制指令流程及书写微程序

编写程序	代码(机器指令，16 进制)
MOV1 05#,R0	18 05
MOV2 03#,R1	29 03
ADD R0,R1	31
MOV3 R1,(F1#)	47 F1
SUB R0,R1	61
MOV3 R1,(F2#)	47 F2
AND R0,R1	71
MOV3 R1,(F3#)	47 F3
OR R0,R1	81
MOV3 R1,(F4#)	47 F4
XOR R0,R1	91
MOV3 R1,(F5#)	47 F5

MUL R0,R1	A1
MOV3 R1,(F6#)	47 F6
A+1 R0,R1	B1
MOV3 R1,(F7#)	47 F7
A-1 R0,R1	C1
MOV3 R1,(F8#)	47 F8
A+AB+1 R0,R1	D1
MOV3 R1,(F9#)	47 F9
A+B+1 R0,R1	E1
MOV3 R1,(FA#)	47 FA
HALT	08

操作码二进制代码：

MOV1: 0001

MOV2: 0010

ADD: 0011

MOV3: 0100

SUB: 0110

AND: 0111

OR: 1000

XOR: 1001

MUL:1010

A+1: 1011

A-1: 1100

A+AB+1: 1101

A+B+1: 1110

微程序入口（16 进制代码）

取指周期微指令 **入口**：00H

MOV1 执行周期微指令　　入口：10H

MOV2 执行周期微指令　　入口：20H

ADD 执行周期微指令　　入口：30H

MOV3 执行周期微指令　　入口：40H

SUB 执行周期微指令　　入口：60H

AND 执行周期微指令　　入口：70H

OR 执行周期微指令　　入口：80H

XOR 执行周期微指令　　入口：90H

MUL 执行周期微指令　　入口：A0H

A+1 执行周期微指令　　入口：B0H

A-1 执行周期微指令　　入口：C0H

A+AB+1 执行周期微指令　　入口：D0H

A+B+1 执行周期微指令　　入口：E0H

HALT 执行周期微指令　　入口：50H

调试

1、微程序经过检查无误后通过"计算机组成原理与系统结构安装软件"以十六进制写入 3#ROM2#ROM 和 1#ROM 的相应单元中去。

按复位键 RET 使 MAR 清 0、指令计数器 PC 清 0，保证从存储器 0 号单元取指令。使微程序计数器 UPC 清 0，保证从 3#RAM、2#ROM、1# ROM 的 0#单元取出取指令微程序的第一条微指令。

2、 执行微程序

按复位键后, μPC,PC,MAR 为 0。

按一次脉冲键产生一负脉冲(作为 μRD)，将 ROM1# 、 ROM2# 0 号单元的

16 位微指令代码读出，用 的上升沿将微指令送入 μIR15-0，看是否正确。第一条微指令产生的命令是： MA、RD2 、CPIR，后继微地址产生方式为 μPC+1，其操作是： RD2 读 RAM，单元地址为 0，即读 0 号单元的内容。 0 号单元的内容是一条指令，指令代码读出后，在 MA 的作用下，进入加法器至总线。按一次脉冲键又产生一负脉冲。该负脉冲反相后的上升沿产生 CPIR,将上条微指令读出的指令代码送 IR，同时上升沿还将 μPC+1。该负脉冲的低电平用以读出 μPC 指示的第二条微指令。

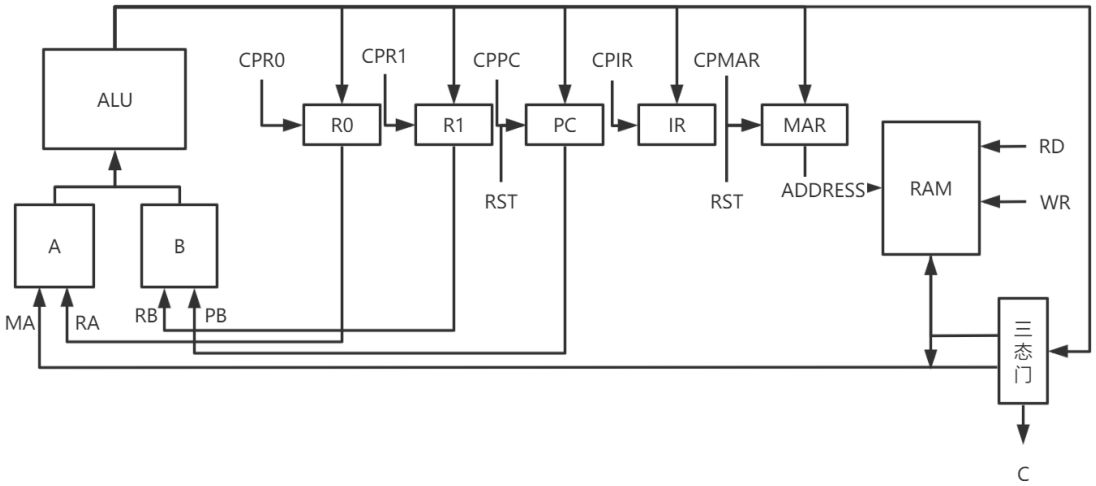
模型及设计—硬布线实现

设计目标

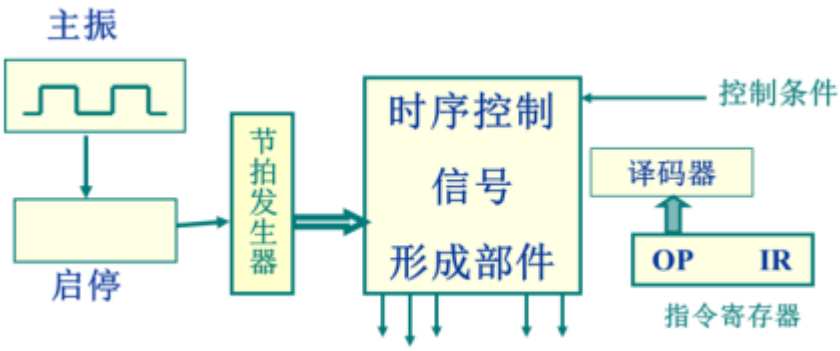
本次设计要实现一个硬布线控制的模型机， ALU 部分用 74181 芯片与 74182 芯片来设计。本模型机所要实现的功能包括：加法、减法、增 1、减 1、两数之和增一、逻辑与、或、异或、直传、非和停机。 并对该模型机进行一定的优化，使其节省节拍信号数，以获得更快的执行速度。

字长采用八位字长， 指令格式与微程序设计中相同。

总体结构



硬布线逻辑电路控制器的结构如下图所示：



关键部件设计：

具体设计电路图见附录。

ALU，二选一选择器，三态门，寄存器组的设计与微程序实现的模型机一致。

部件之间的连接是采用以 CPU 为中心的总线连接方式。 ALU 的输出通过总线 BUS 连接到所有寄存器和存储器的输入端，除指令寄存器 IR 和地址寄存器 MAR 的输出端外，其它部件的输出端分别送入选择器 A 和选择器 B。

各部件的设计：

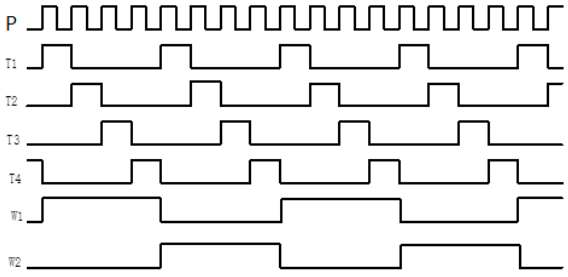
1. 主振和启停电路的设计

   在微程序控制的模型机中我们已经实现了可以控制的连续脉冲的启停电路。微程序控制的启停电路在硬布线实现的模型机中也可以使用，但发现运算指令的执行周期所需的时钟周期数最少，可优化，因而选择对该种指令进行优化。只需要当期执行周期的第一个时钟周期执行完成后，对节拍发生器中的模四增一计数器进行清零即可。

2. 节拍发生器的设计与实现

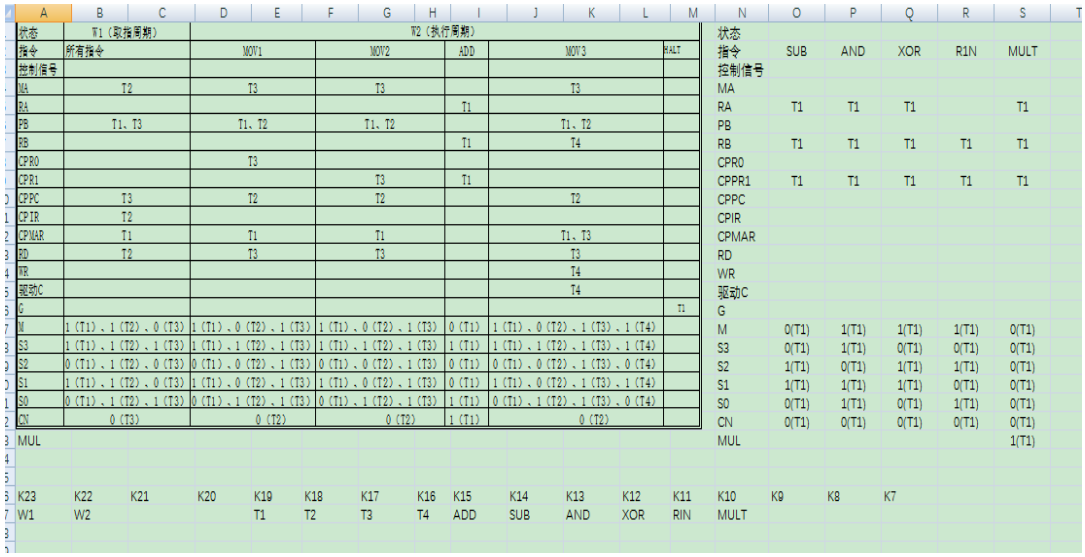
   根据指令执行流程，设计时序为：分取指周期和执行周期，每个周期为 4 节拍，波形图如下：

   W1 高电平时是取指周期的 4 节拍，W2 高电平时是执行周期的 4 节拍。



3. 时序控制信号产生部件的设计

各个小部件的详细设计电路图见附录图，**采用的是每一个指令的分解都由一个小部件形成**。



状态	W1（取指周期）		W2（执行周期）																	
指令	所有指令		MOV1		MOV2		ADD	MOV3		HALT										
控制信号																				
MA	T2		T3		T3			T3												
RA							T1													
PB	T1、T3		T1、T2		T1、T2			T1、T2												
RB							T1	T4												
CPR0			T3																	
CPR1					T3		T1													
CPPC	T3		T2		T2			T2												
CPIR	T2																			
CPMAR	T1		T1		T1			T1、T3												
RD	T2		T3		T3			T3												
WR								T4												
驱动C								T4												
G										T1										
M	1（T1）、1（T2）、0（T3）		1（T1）、0（T2）、1（T3）		1（T1）、0（T2）、1（T3）		0（T1）	1（T1）、0（T2）、1（T3）、1（T4）												
S3	1（T1）、1（T2）、1（T3）		1（T1）、1（T2）、1（T3）		1（T1）、1（T2）、1（T3）		1（T1）	1（T1）、1（T2）、1（T3）、1（T4）												
S2	0（T1）、1（T2）、0（T3）		0（T1）、0（T2）、1（T3）		0（T1）、0（T2）、1（T3）		0（T1）	0（T1）、0（T2）、1（T3）、0（T4）												
S1	1（T1）、1（T2）、0（T3）		1（T1）、0（T2）、1（T3）		1（T1）、0（T2）、1（T3）		0（T1）	1（T1）、0（T2）、1（T3）、1（T4）												
S0	0（T1）、1（T2）、1（T3）		0（T1）、1（T2）、1（T3）		0（T1）、1（T2）、1（T3）		1（T1）	0（T1）、1（T2）、1（T3）、0（T4）												
CN	0（T3）		0（T2）		0（T2）		1（T1）	0（T2）												
MUL																				

状态						
指令	SUB	AND	XOR	R1N	MULT	
控制信号						
MA						
RA	T1	T1	T1		T1	
PB						
RB	T1	T1	T1	T1	T1	
CPR0						
CPPR1	T1	T1	T1	T1	T1	
CPPC						
CPIR						
CPMAR						
RD						
WR						
驱动C						
G						
M	0(T1)	1(T1)	1(T1)	1(T1)	0(T1)	
S3	0(T1)	1(T1)	0(T1)	0(T1)	0(T1)	
S2	1(T1)	0(T1)	1(T1)	1(T1)	0(T1)	
S1	1(T1)	1(T1)	1(T1)	0(T1)	0(T1)	
S0	0(T1)	1(T1)	0(T1)	1(T1)	0(T1)	
CN	0(T1)	0(T1)	0(T1)	0(T1)	0(T1)	
MUL					1(T1)	

K23	K22	K21	K20	K19	K18	K17	K16	K15	K14	K13	K12	K11	K10	K9	K8	K7
W1	W2			T1	T2	T3	T4	ADD	SUB	AND	XOR	RIN	MULT			

MA = W1*T2 + W2*T3* (MOV1 +MOV2+MOV3)

RA = W2*T1*(ADD + SUB+ AND + XOR +MULT)

PB = W1*(T1+T3) + W2*(T1+T2)*(MOV1 + MOV2 + MOV3)

RB = W2*T1*(ADD+SUB+AND+XOR+R1N+MULT) + W2*T4*MOV3

CPR0 = W2*T3*MOV1

CPR1 = W2*T1*(ADD+SUB+AND+XOR+R1N+MULT) + W2*T3*MOV2

CPPC = W1*T3 + W2*T2*(MOV1 + MOV2 + MOV3)

CPIR = W1*T2

CPMAR = W1*T1 + W2*T1*(MOV1 + MOV2 + MOV3) + W2*T3*MOV3

RD = W1*T2 + W2*T3*(MOV1+MOV2+MOV3)

WR = W2*T4*MOV3

驱动 C = W2*T4*MOV3

G = W2*T1*HALT

M

= W1*(T1 + T2 + ~T3) + W2*(MOV1*(T1 + ~T2 + T3) + MOV2*(T1 + ~T2 + T3) + MOV3*(T1 + ~T2 + T3 + T4)) + W2*T1*(AND+ XOR + R1N) + W2*~T1*(ADD + SUB + MULT)

= W1*(T1 + T2 + ~T3) + W2*(MOV1+MOV2+MOV3)*(T1 + ~T2 + T3) + W2*MOV3* T4 + W2*T1*(AND+ XOR + R1N) + W2*~T1*(ADD + SUB + MULT)

S3 = W1*(T1 + T2 + T3) + W2*(T1+T2+T3)*(MOV1+MOV2+MOV3) + W2*T4*MOV3 + W2*T1*(ADD+AND) + W2*~T1*(SUB + XOR + R1N + MULT)

S2=W1*(~(T1+T3)+T2)+W2*(~(T1+T2)+T3)*(MOV1+MOV2)+W2*(~(T1+T2+T4)+T3)*MOV3+W2*T1*(SUB+XOR+R1N) + W2*~T1*(ADD+AND + MULT)

S1 = W1*(T1 + T2 + ~T3) + W2*(T1 + ~T2 + T3)*(MOV1+MOV2+MOV3) + W2*T4*MOV3 + W2*T1*(SUB+AND+XOR) + W2*~T1*(ADD+ R1N + MULT)

S0=W1*(~T1+T2+T3)+W2*(MOV1+MOV2)*(~T1+T2+T3)+W2*MOV3*(~(T1+T4)+T2+T3)+ W2*T1*(ADD + AND + R1N) + W2*~T1*(SUB + XOR + MULT)

CN = W1* ~T3 + W2*~T2*(MOV1+MOV2+MOV3) + W2*(T1*ADD + ~T1*(SUB + AND + XOR + R1N + MUTL))

MUL = W2*T1*MULT

RAM 中编写代码

RAM 指令集如下：

地址:

00

数据:

18 05 29 03 31 47 F1 加

地址:

07

数据:

18 05 29 03 51 47 F2 减

地址:

0E

数据:

18 05 29 03 61 47 F3 自增一

地址:

15

数据:

18 05 29 03 71 47 F4 与

地址:

1C

数据:

18 05 29 03 81 47 F5 或

地址:

23

数据:

18 05 29 03 91 47 F6 异或

地址:

2A

数据:

18 05 29 03 A1 47 F7 直传

地址:

31

数据:

18 05 29 03 B1 47 F8 非

地址:

38

数据:

18 05 29 03 D1 47 F9 乘

（具体的指令结构同上面的微指令实现）

调试

1、按复位键 RET

使 MAR 清 0、指令计数器 PC 清 0，保证从存储器 0 号单元取指令。

2、按单脉冲键，启动程序执行。

问题分析

1. 做硬布线的模型机的时候，出现了一系列的问题。经过排查原因主要有两个，一个是由于脉冲波形的问题，需要在 CPR0、 CPR1、CPPC、 CPIR、 CPMAR 信号的 P 输入端加上一个非门。

2. 乘法器

   因为这次实验采用的是用两片 74285 实现乘法器，输入是两个四位，输出是两个八位，而且乘法器的操作并不涉及到 M、S0-S3，所以要把乘法器和其他 ALU 部件放在一个层次上，用一个选择信号控制，即 UIR22。

3. 做两个实现的模型机的时候，需要对 RAM 中的数据和总线中的数据进行筛选。

实验感悟

微指令实现模型机的操作的关键在于整个模型机的运行是按步骤实现的，一步步执行整个程序，对于下地址形成部件，选择器用来提取 IR 中高四位来确定微程序的入口，uPC 是用来执行确定执行下一步微指令。逻辑设计其实是将具体操作划分为指令，指令又划分为微指令，微指令中含有 uIR 的各种端口来响应操作。

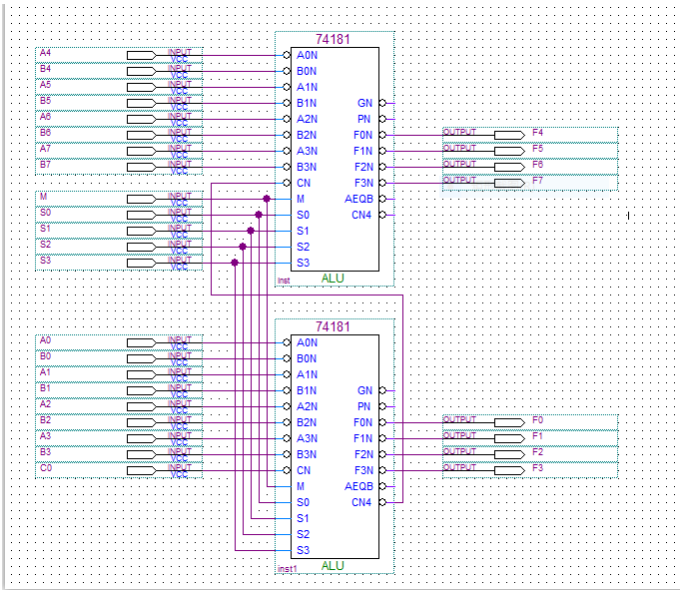
硬布线电路不用 ROM 存储器，直接通过相应的硬布线电路来设计这些控制信号，不同于微程序模型机通过微指令信号中的 uIR23—uIR0 控制。

无论是底层基础元件的设计，还是整体系统的连接都是不小的挑战，工作极为繁琐需要个人的细致和耐心，对于脉冲，节拍和指令有了进一步的理解，继修读了《计算机组成原理》后算是真正明白了一台基本计算机的执行过程，对计算机整体有了进一步的理解。

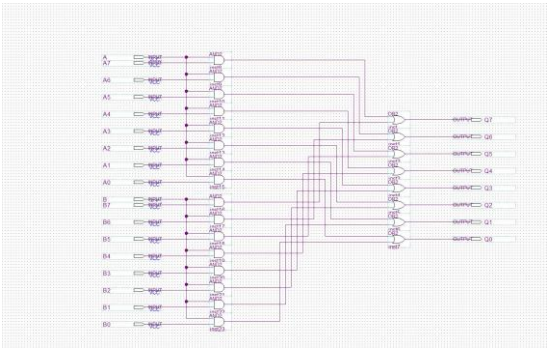
附录

微程序实现的模型机：

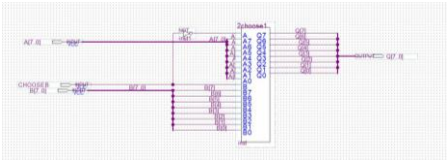
ALU



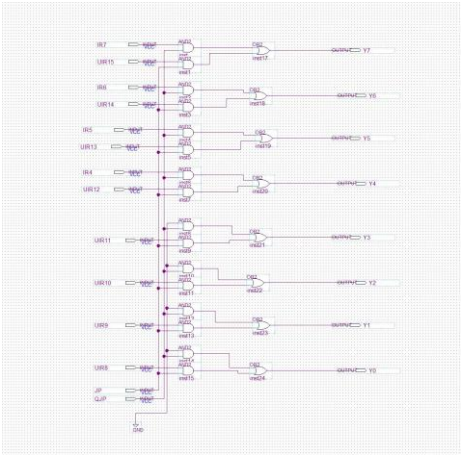
二选一选择器：



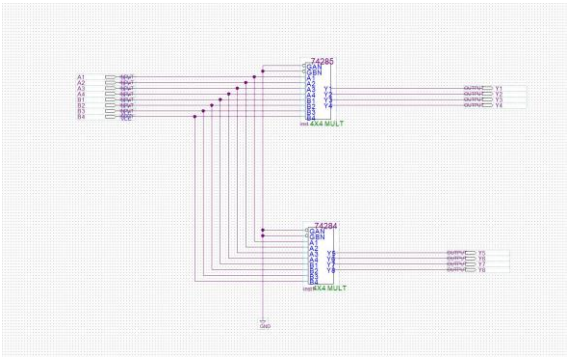
RAM 选择器：



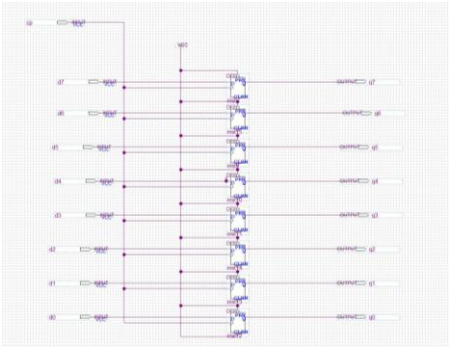
后继形成：

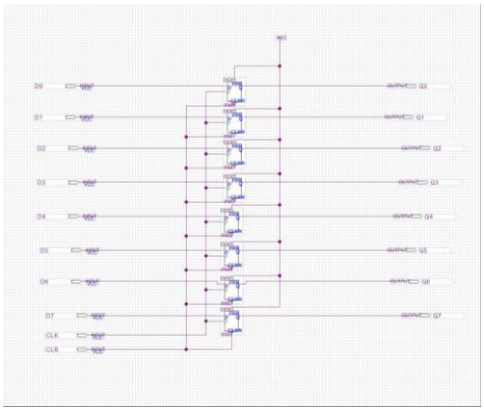


乘法器：

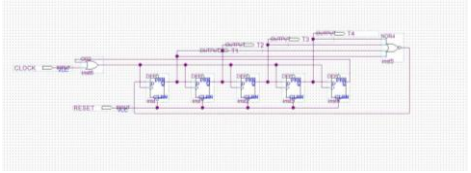


寄存器：

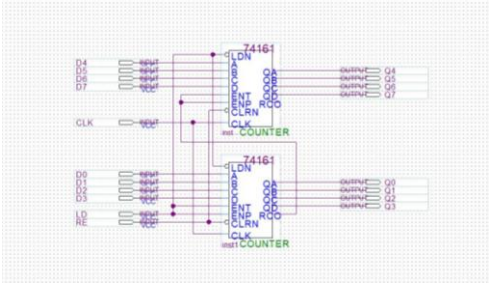




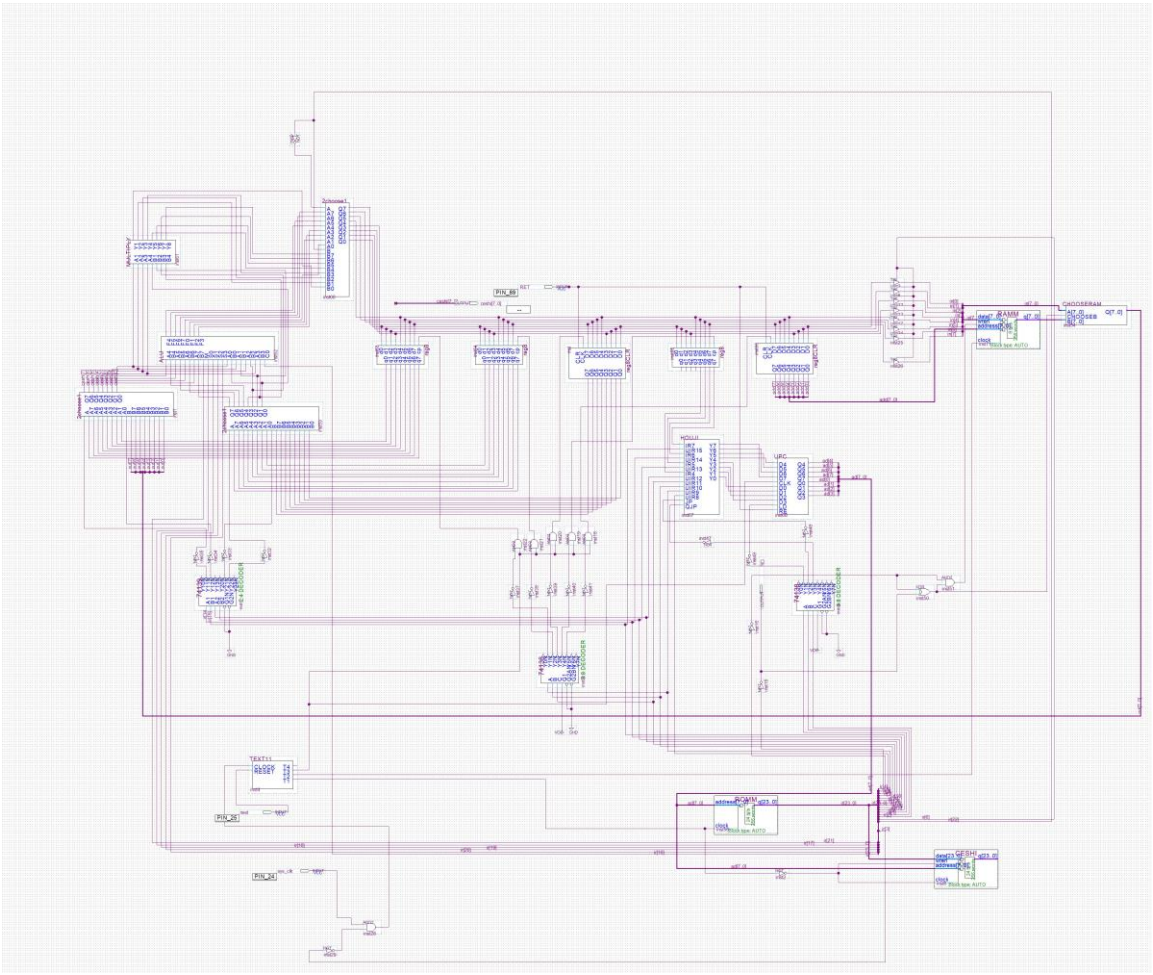
时钟：



Upc：



总体结构图：

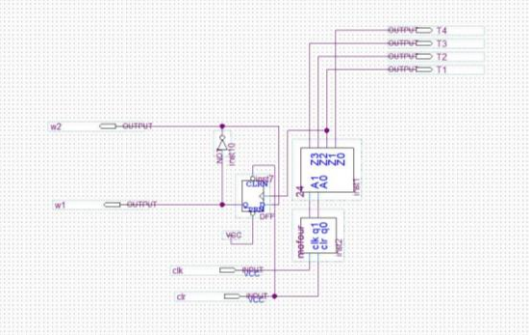


硬布线实现的模型机：

ALU **部分与上面相同**

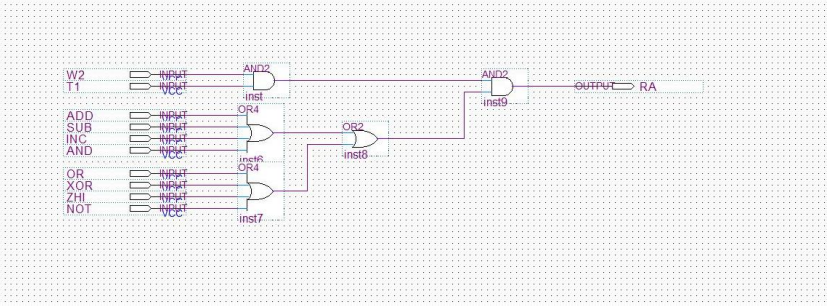
硬布线实现的时序：

节拍器：

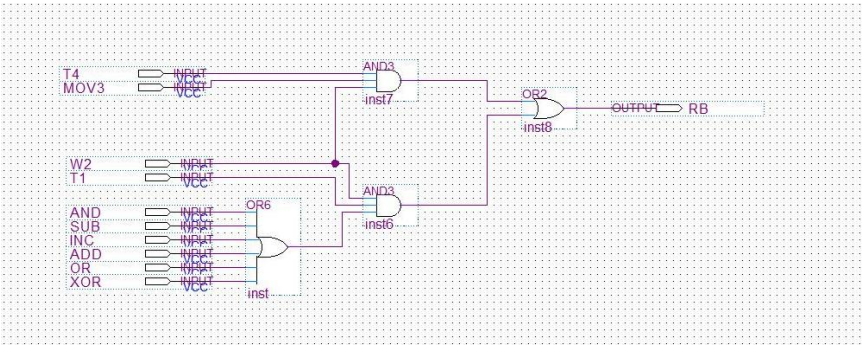


具体指令实现：

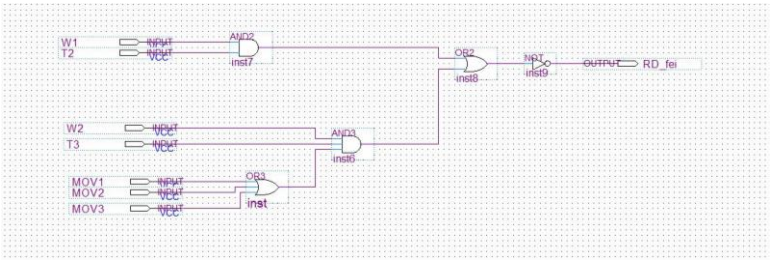
RA:



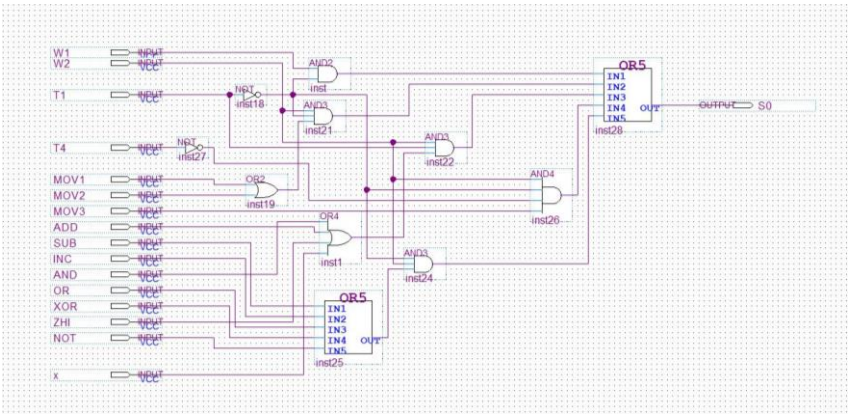
RB:



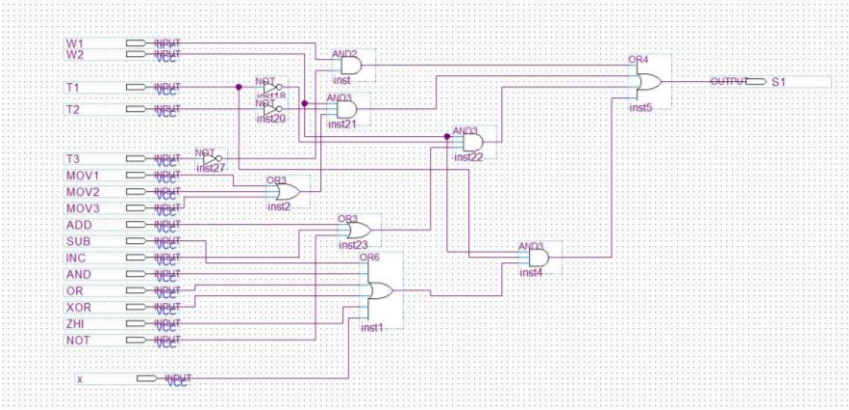
RD:


W1
T2
W2
T3
MOV1
MOV2
MOV3
AND2
inst7
OR2
inst8
NOT
inst9
OUTPUT
RD_fei
AND3
inst6
OR3
inst

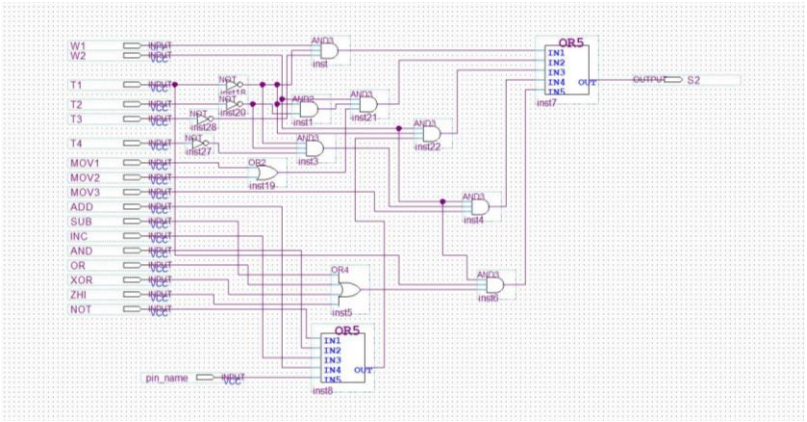
So:


W1
W2
T1
T4
MOV1
MOV2
MOV3
ADD
SUB
INC
AND
OR
XOR
ZHI
NOT
x
NOT
inst18
inst27
AND2
inst
AND3
inst21
OR2
inst19
OR4
inst1
OR5
IN1
IN2
IN3
IN4
IN5
OUT
inst25
AND3
inst22
AND4
inst26
AND3
inst24
OR5
inst28
OUTPUT
S0

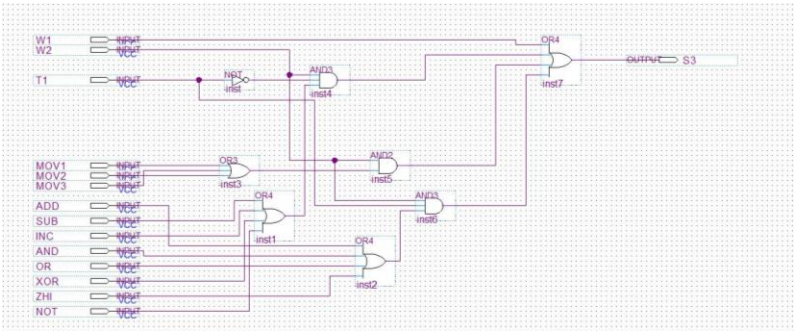
S1:


W1
W2
T1
T2
T3
MOV1
MOV2
MOV3
ADD
SUB
INC
AND
OR
XOR
ZHI
NOT
x
NOT
inst18
NOT
inst20
NOT
inst27
AND2
inst
AND3
inst21
OR3
inst2
OR3
inst23
OR6
inst1
AND3
inst22
AND3
inst4
OR4
inst5
OUTPUT
S1

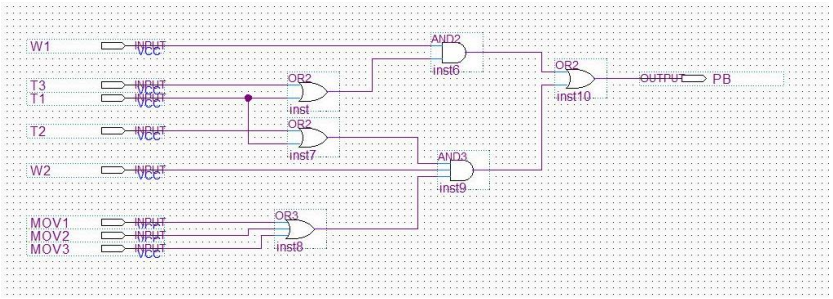
S2:


W1
W2
T1
T2
T3
T4
MOV1
MOV2
MOV3
ADD
SUB
INC
AND
OR
XOR
ZHI
NOT
pin_name
INPUT
VCC
AND3
OR5
IN1
IN2
IN3
IN4
IN5
OUT
OUTPUT
S2
NOT
OR2
OR4
inst
inst1
inst2
inst3
inst4
inst5
inst6
inst7
inst8
inst19
inst20
inst21
inst22
inst27
inst28

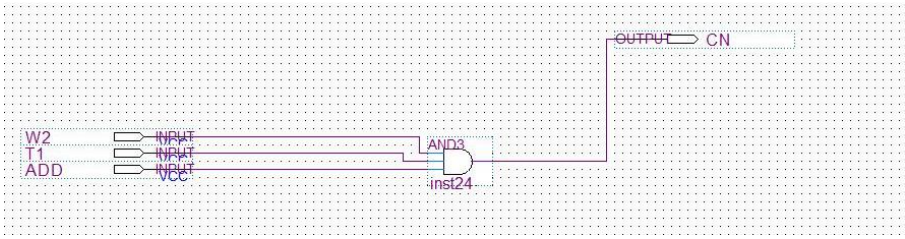
S3:


W1
W2
T1
MOV1
MOV2
MOV3
ADD
SUB
INC
AND
OR
XOR
ZHI
NOT
INPUT
VCC
NOT
AND3
AND2
OR3
OR4
OUTPUT
S3
inst
inst1
inst2
inst3
inst4
inst5
inst6
inst7

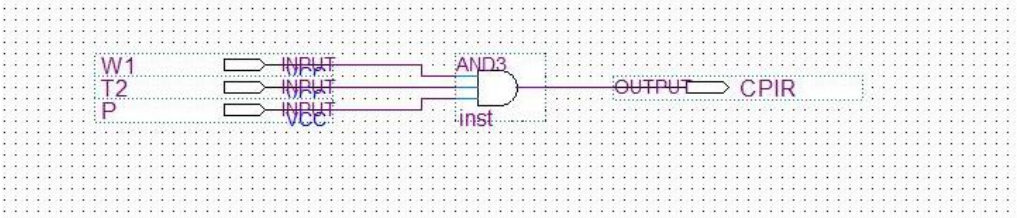
PB:


W1
T3
T1
T2
W2
MOV1
MOV2
MOV3
INPUT
VCC
AND2
OR2
AND3
OR3
OUTPUT
PB
inst
inst6
inst7
inst8
inst9
inst10

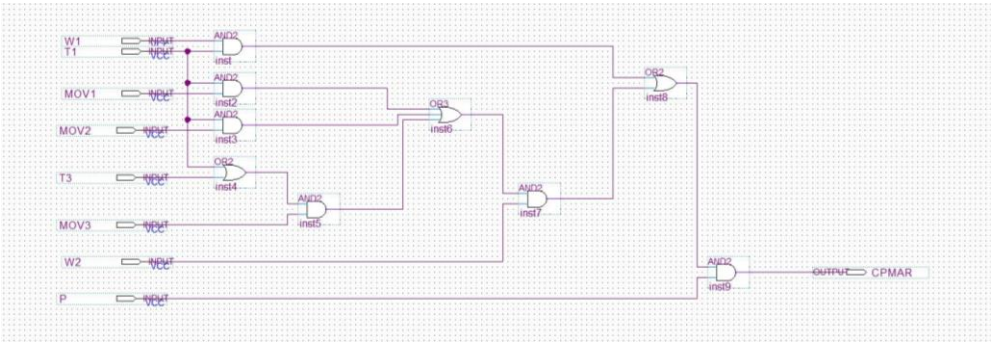
CN：



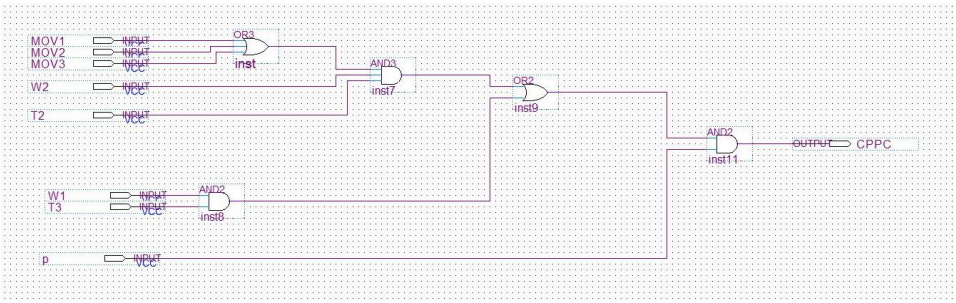
CPIR:



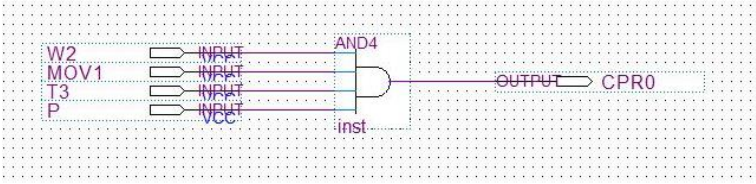
CPMAR:



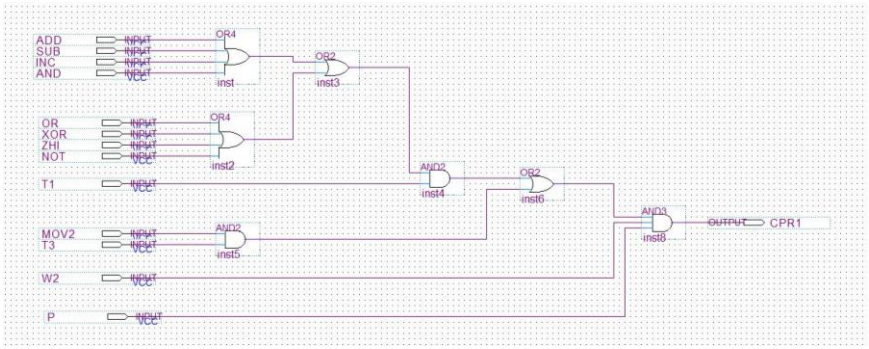
CPPC:



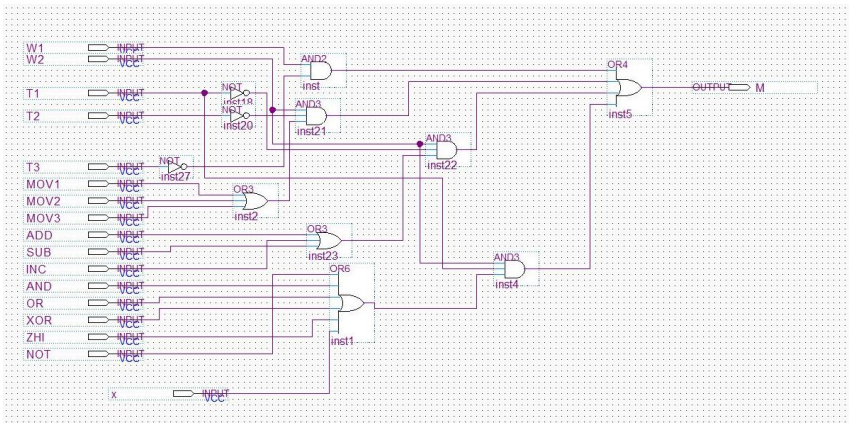
CPR0:



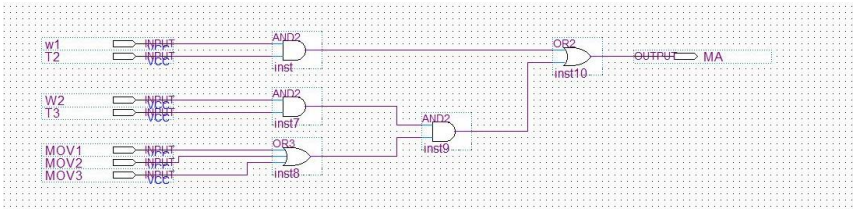
CPR1:



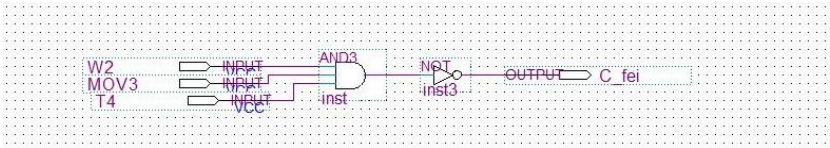
M:



MA:



C_fei：



总体电路图：