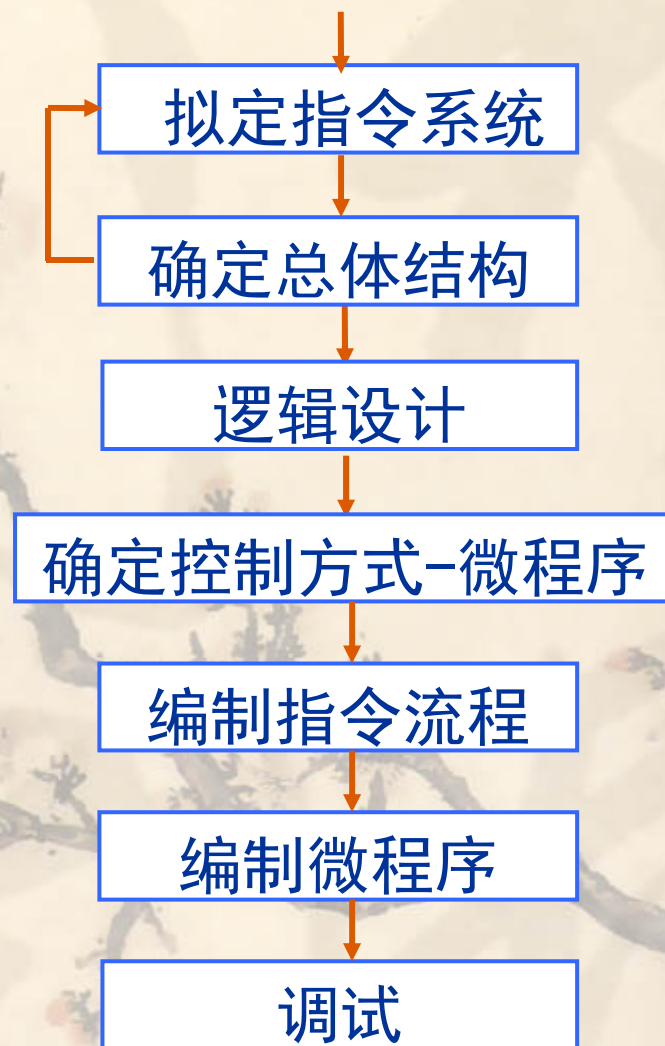


第3讲—简单模型机的设计 (微程序实现)

计算机学院

张瑞华

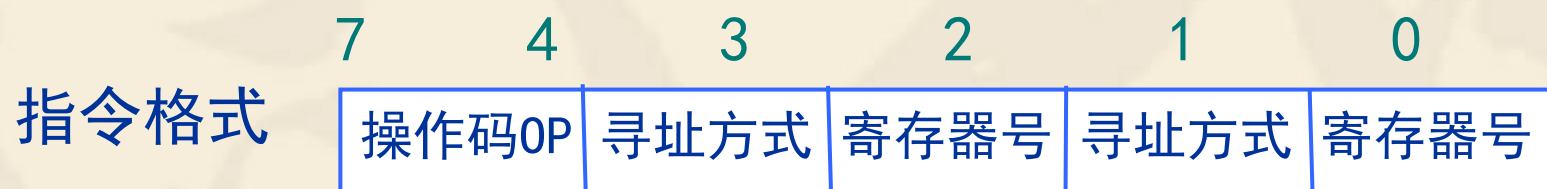
模型机设计步骤



1、拟定指令系统

- ❖ 拟定指令系统将涉及到基本字长、指令格式、指令种类、寻址方式等内容。这些内容的确定又和总体结构密切相关。
- ❖ 基本字长
 - ∞ 存储器容量为 256×8 ，基本字长定为8位
- ❖ 指令格式
 - ∞ 指令格式可有单字长指令和双字长指令两种
 - ∞ 在双字长格式中，第二字节一般定义为操作数或操作数地址。

基本字长 8位



源操作数

目的操作数

❖ 指令类型

- ❧ 模型机有单操作数指令、双操作数指令和无操作数指令。
- ❧ 操作码OP共4位，最多可定义16条指令。

❖ 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，相应的寄存器号位为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，
 - ❖ 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。

2、确定总体结构

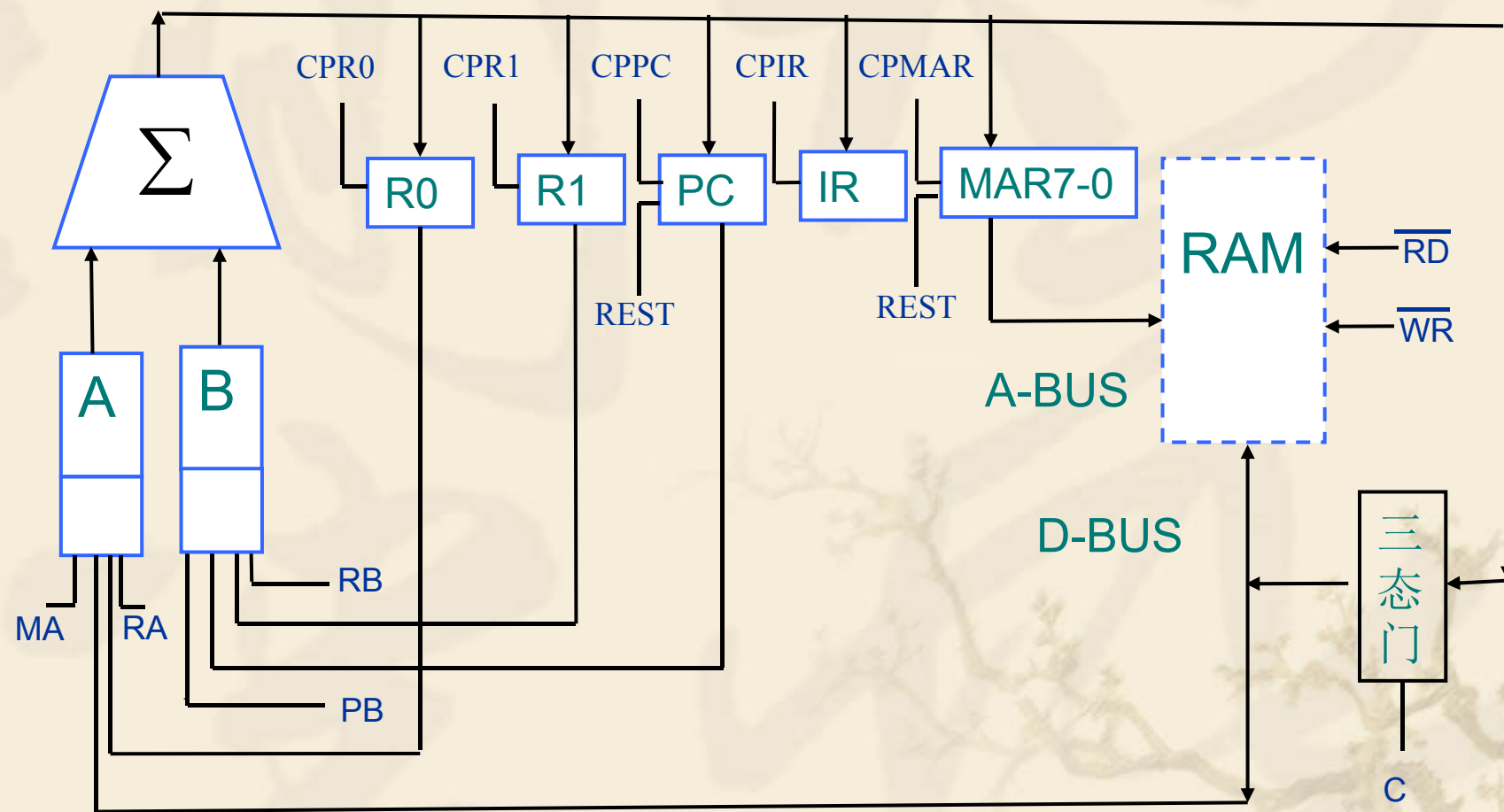


图1

❖ 寄存器组的设置

- ❧ R_0 、 R_1 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。

❖ 加法器ALU的设置

- ❧ 采用74181、74182实现

❖ 选择器的设置

- ❧ 连入A选择器的数据来源是RAM的读出数据和 R_0 寄存器的数据。
- ❧ 连入B选择器的数据来源是PC的数据和 R_1 的数据。

❖ 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。

❖ 取指令:

❧ MA A直传 CPIR

❧ $RAM \rightarrow \text{选择器A} \rightarrow \Sigma \rightarrow \text{Bus} \rightarrow \text{IR}$

❖ 送指令地址

❧ PB B直传 CPMAR

❧ $PC \rightarrow \text{选择器B} \rightarrow \Sigma \rightarrow \text{Bus} \rightarrow \text{MAR}$

❖ 指令计数器+1

❧ PB A加B加1 (A为0) cppc

❧ $PC \rightarrow \text{选择器B} \rightarrow \Sigma \rightarrow \text{Bus} \rightarrow PC$

❖ $R_0 \rightarrow R_1$

∞

RA

A直传

CPR1

∞ $R_0 \rightarrow \text{选择器A} \rightarrow \Sigma \rightarrow \text{Bus} \rightarrow R_1$

❖ $R_1 \rightarrow \text{RAM}$

∞

RB

B直传

C

WR

∞ $R_1 \rightarrow \text{选择器B} \rightarrow \Sigma \rightarrow \text{Bus} \rightarrow \text{RAM}$

3、 逻辑设计

- ❖ 总体结构中，虚线框内的RAM是FPGA之外预先配置好的。
- ❖ （1）ALU的逻辑设计
 - ∞ 由74181和74182组成。

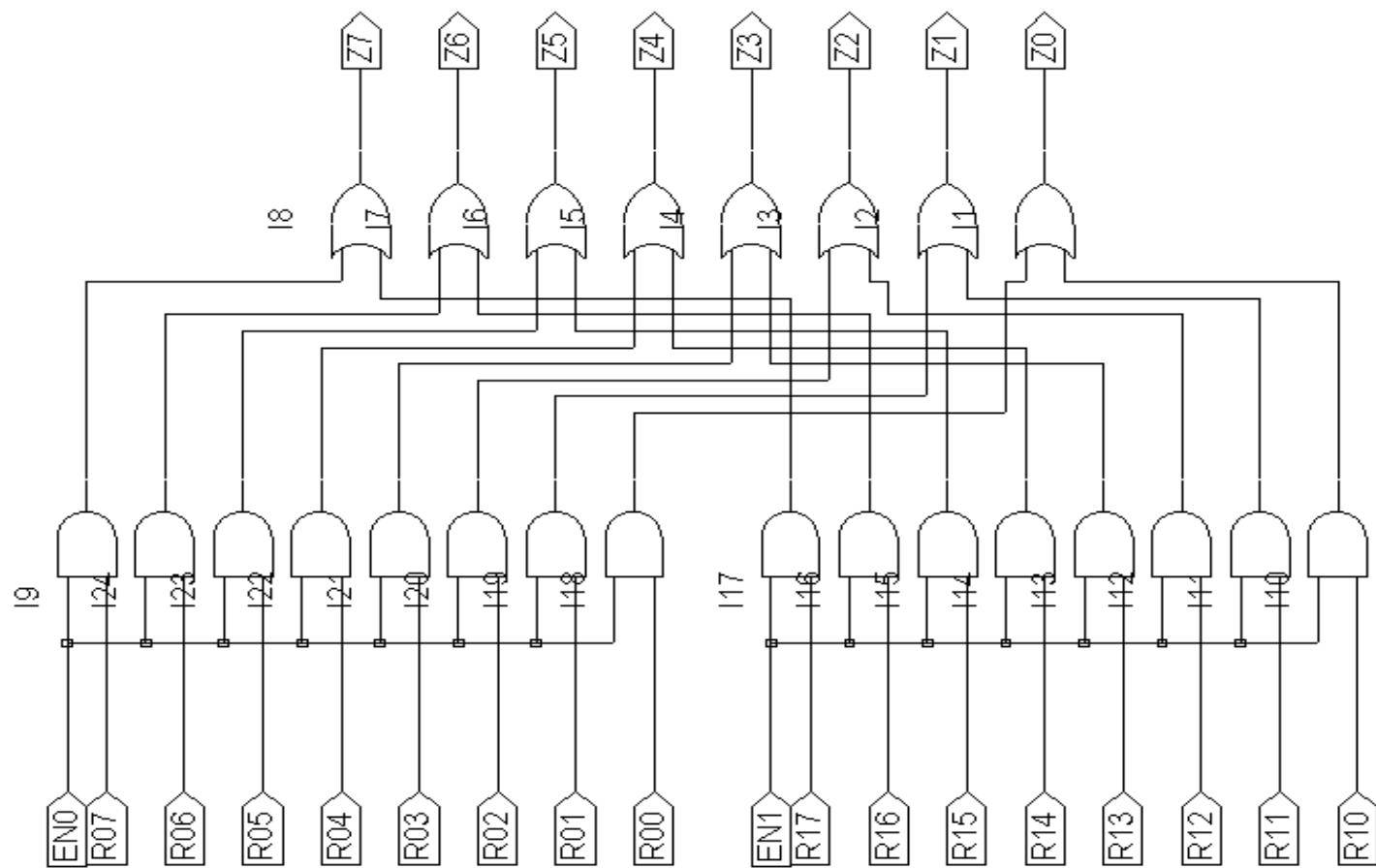


图2 选择器设计

❖ (2) 寄存器的设计

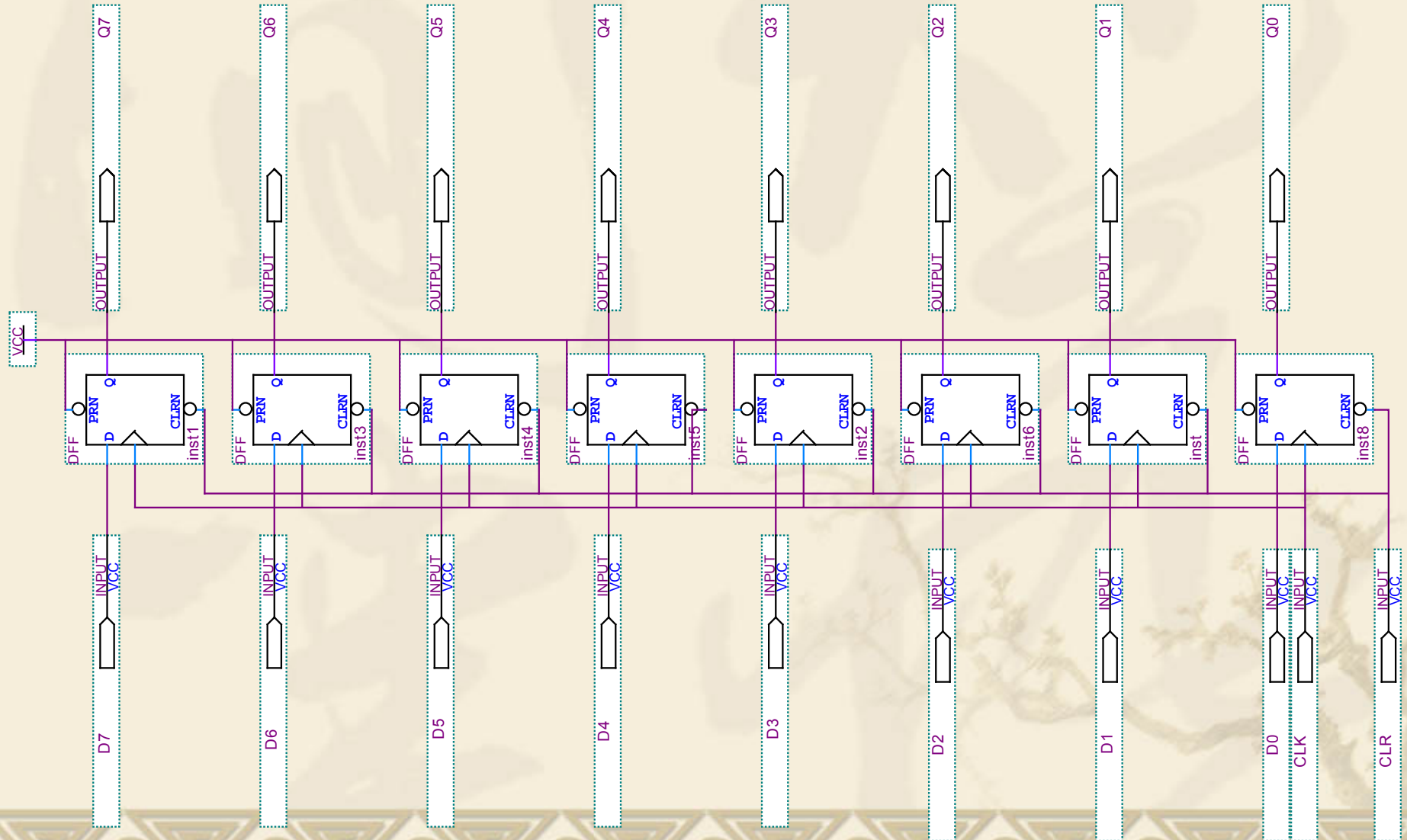
❖ 不带复位的寄存器

- ❧ 结构中 R_0 、 R_1 通用寄存器，可存放操作数或结果、中间结果，每个寄存器均由8个D触发器构成。
- ❧ 在 CPR_i 的作用下接收总线的数据送入寄存器，输出连入选择器。
- ❧ 指令寄存器IR其结构同通用寄存器。

❖ 带复位的寄存器

- ❧ 结构中MAR地址寄存器是一个带复位的寄存器，带复位是指当有复位信号时，MAR清零。
- ❧ 逻辑图如图3所示。

图3 带复位的八位寄存器逻辑图



❖ 程序计数器PC的设计

- ❧ 程序计数器结构如上图3所示，是有复位信号的8位寄存器。PC加1是通过加法器实现的。
- ❧ 复位信号RET 的作用是有复位信号时，计数器PC清零。

❖ (3) 三态门的设计

- ❧ 利用8个三态门TRI实现：
- ❧ C=L时，三态；C=H时，Y=A

❖ (4) 部件之间的连接

- ❧ 由系统结构图（图1）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。
- ❧ 加法器的输出通过总线**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**。
- ❧ 连线图如图4所示。

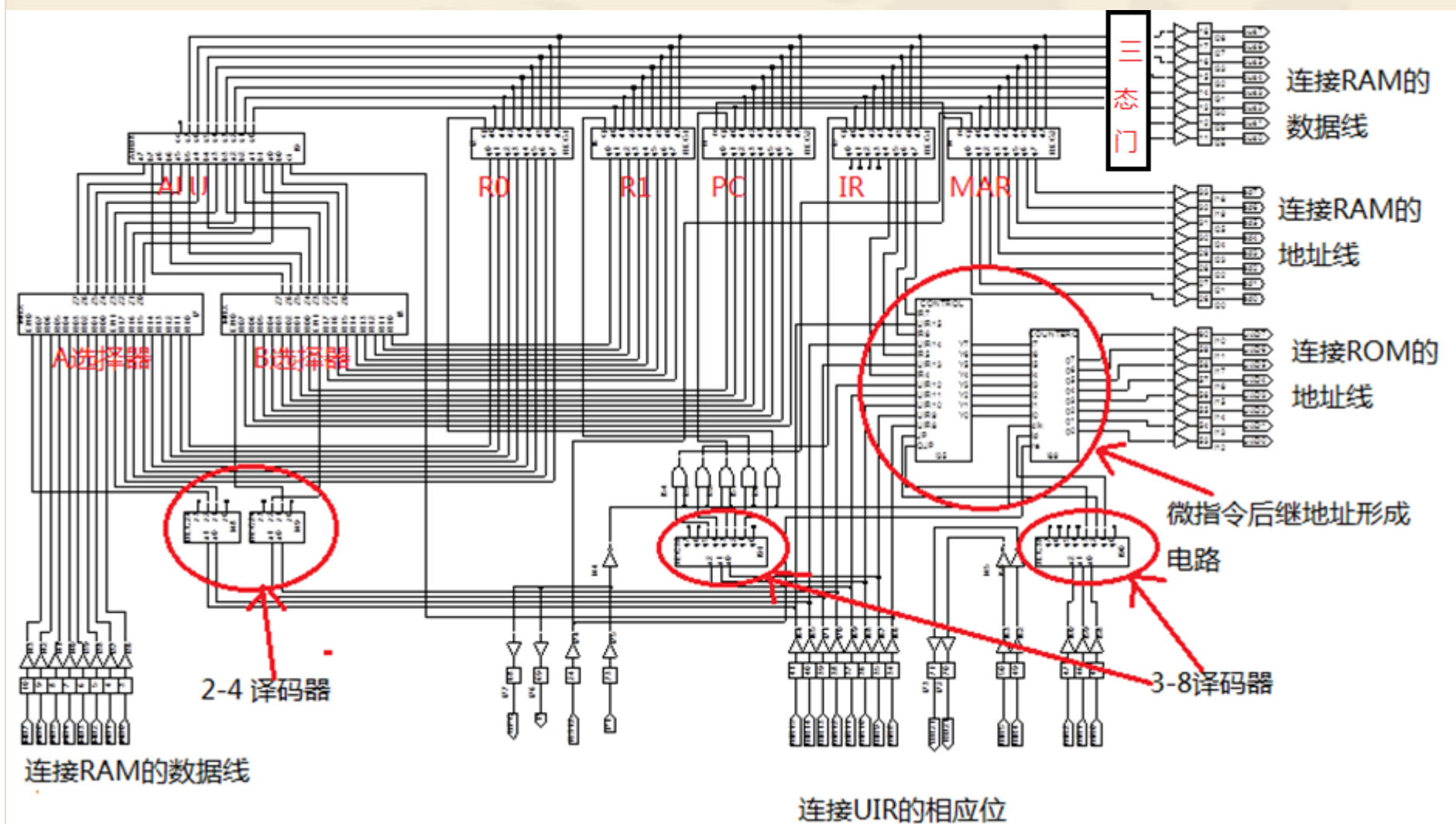
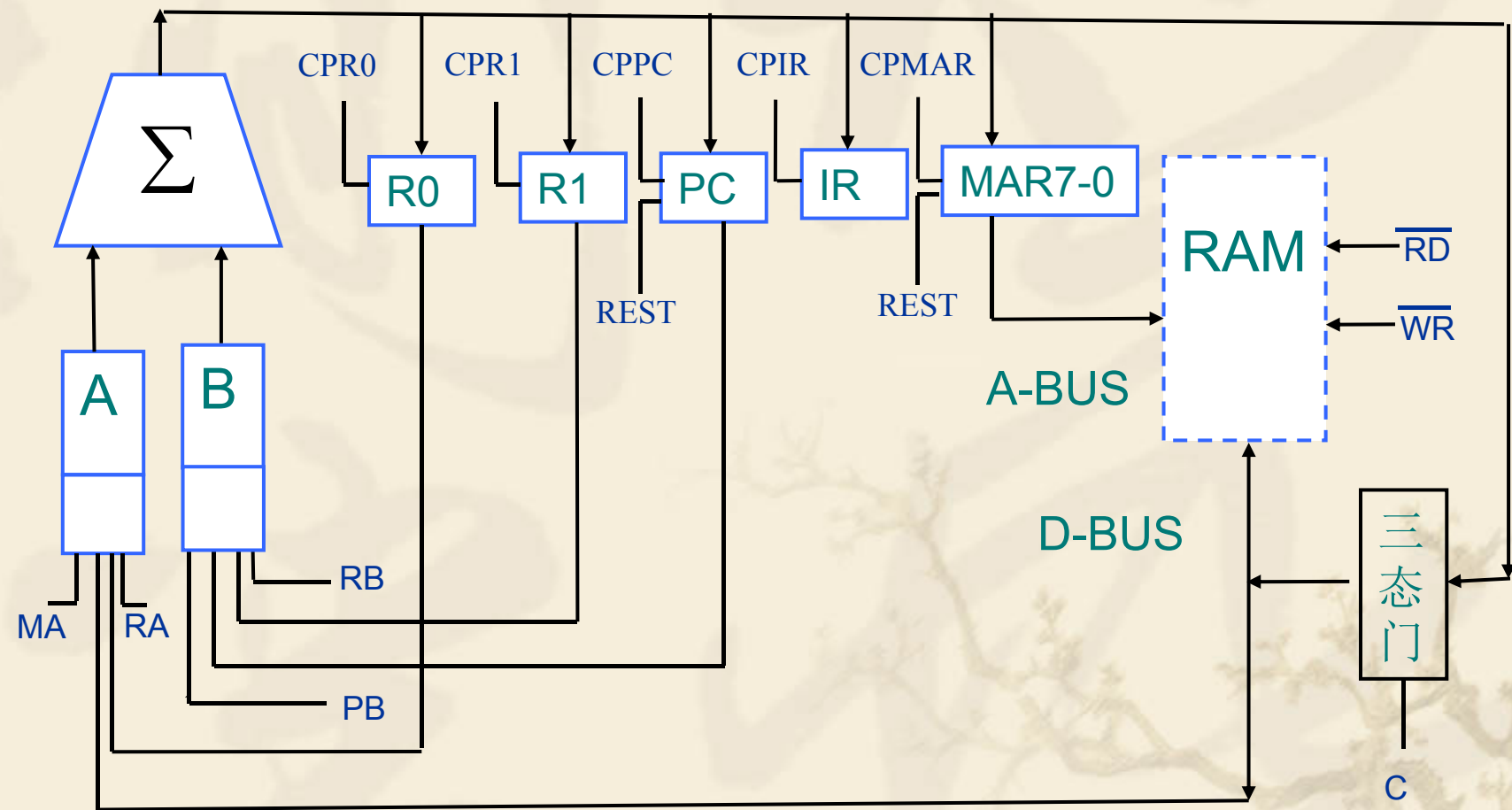


图4

4、确定控制方式

- ❖ 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。
- ❖ 架构图图1中，涉及到了许多控制命令例如 CPR_0 、CPMAR、MA、RB等等，这些命令如何产生？
- ❖ 通常有两种方式，即硬布线逻辑电路方式和微程序方式。本章模型机采用微程序方式。
- ❖ 微程序的执行方式采用增量、垂直方式。

确定总体结构



❖ 微程序控制器的结构

∞ 如图5所示，主要由控制存储器CROM（256X24）、微指令寄存器 UIR（24位）、RAM（256X8）构成。

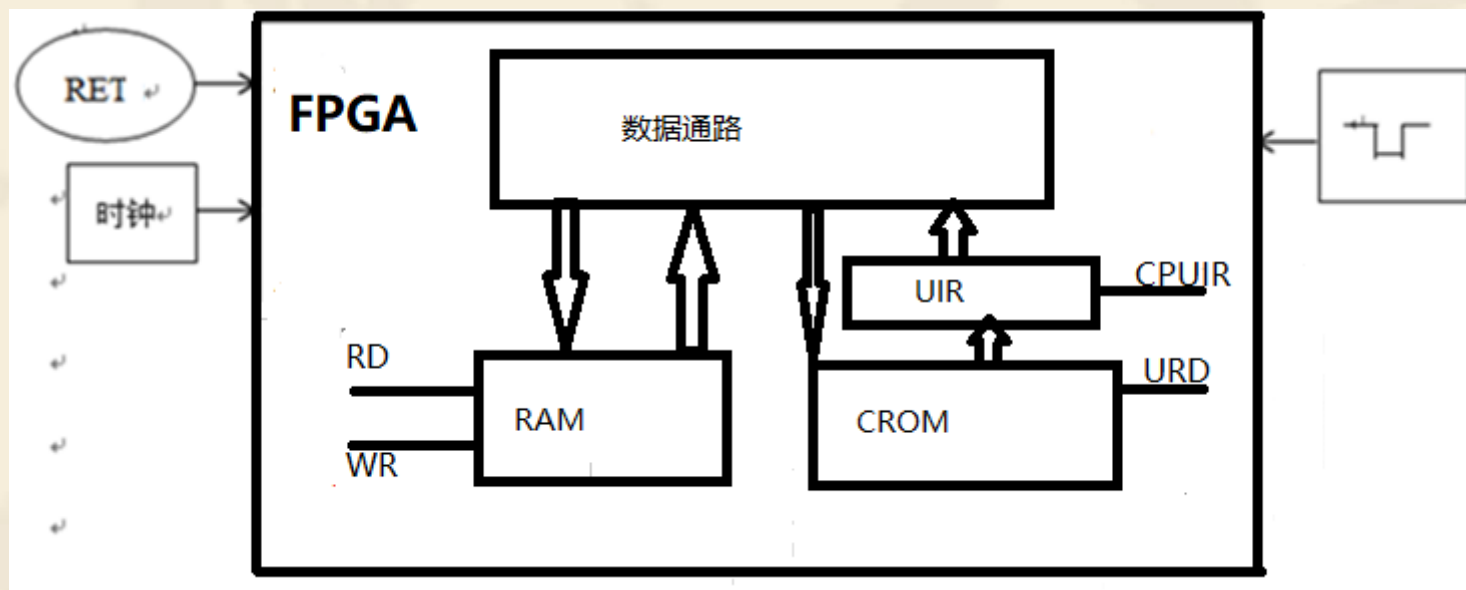
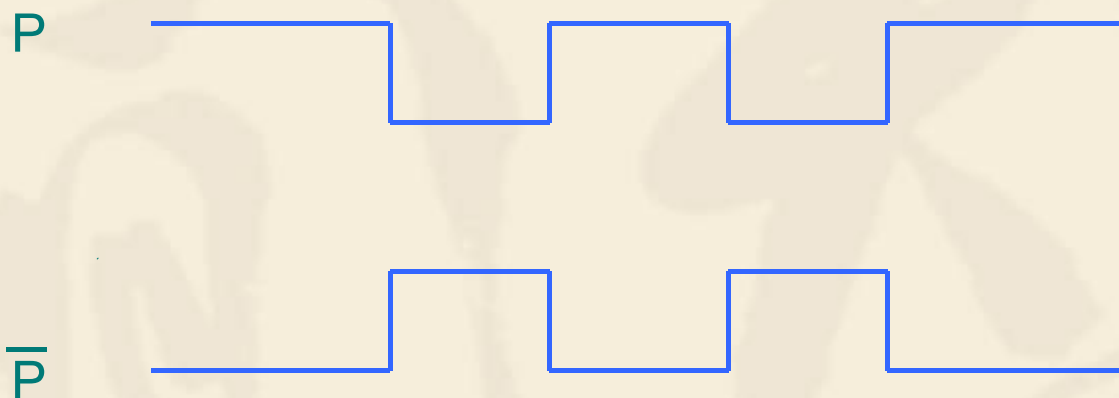


图5



微程序控制器时序

P脉冲的低电平用做控制存储器读命令 $\overline{\mu RD}$

P脉冲的上升边沿将读出的微指令送 μIR

脉冲的上升边沿将形成的后继地址送微程序计数器

μPC ，同时将运算结果（总线的数据）送指定的寄存器。

❖ 3、微指令格式

❖ 微指令字长24位即 $\mu IR_{23} \sim \mu IR_0$ 。

❖ (1) 微指令字段定义

❖ ALU控制: $\mu IR_{21} \cdot \mu IR_{20} \mu IR_{19} \cdot \mu IR_{18} \mu IR_{17} \cdot \mu IR_{16}$

❖ M S3 S2 S1 S0 C0

❖ 三态门控制: μIR_6

❖ 0 高阻态 使C=1

❖ 1 三态门使能 使C=0

❖ 停机控制: μIR_3

❖ 0 G=0, 运行

❖ 1 G=1, 停机



❖ A选择器控制: $\mu IR_{15} \cdot \mu IR_{14}$

⌘	0	0	备用
⌘	0	1	RA
⌘	1	0	MA
⌘	1	1	备用

2-4
译码器，
互斥

❖ B选择器控制: $\mu IR_{13} \cdot \mu IR_{12}$

⌘	0	0	备用
⌘	0	1	PB
⌘	1	0	RB
⌘	1	1	备用

2-4
译码器，
互斥

3-8
译码器，
互斥

❖ 输出分配: $\mu IR_{11} \cdot \mu IR_{10} \cdot \mu IR_9$

∞	0	0	0	备用
∞	0	0	1	CPR ₀
∞	0	1	0	CPR ₁
∞	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: $\mu IR_5 \cdot \mu IR_4$

∞	1	0	\overline{RD}
∞	0	1	\overline{WR}

❖ 后继微地址形成方式:

❖ $\mu IR_2 \cdot \mu IR_1 \cdot \mu IR_0$

3-8 译码器，互斥

❖ 0 0 0 备用

❖ 0 0 1 $\mu PC + 1$ 顺序执行

❖ 0 1 0 JP无条件转移，地址由 IR_{15-8} 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

- ❖ (2) 微命令形成逻辑
- ❖ 微命令形成逻辑电路如图6所示。
- ❖ 图中二—四译码器逻辑原理如图7所示。
- ❖ 三—八译码器逻辑原理如图8所示。

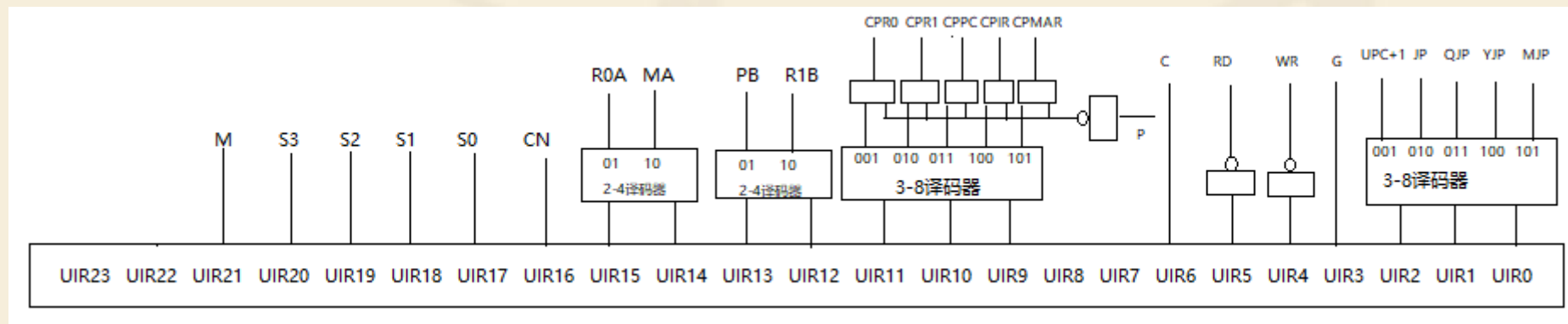
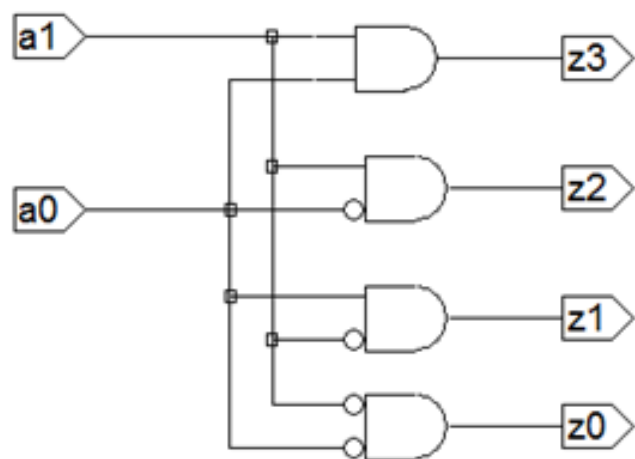
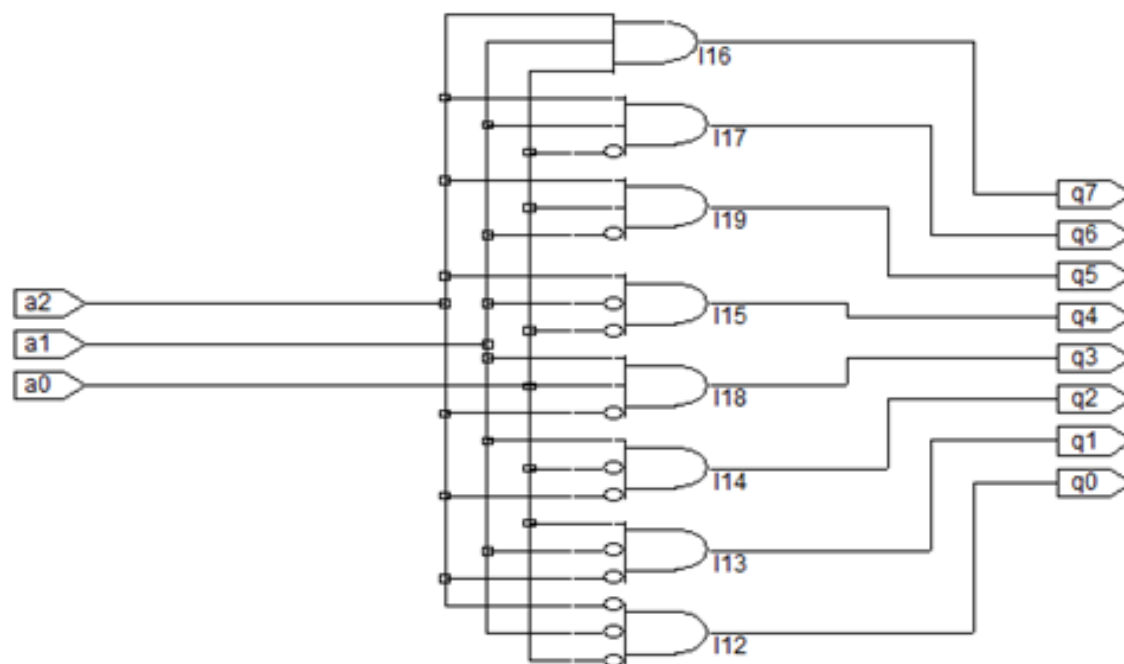


图6



2—4 译码器逻辑图



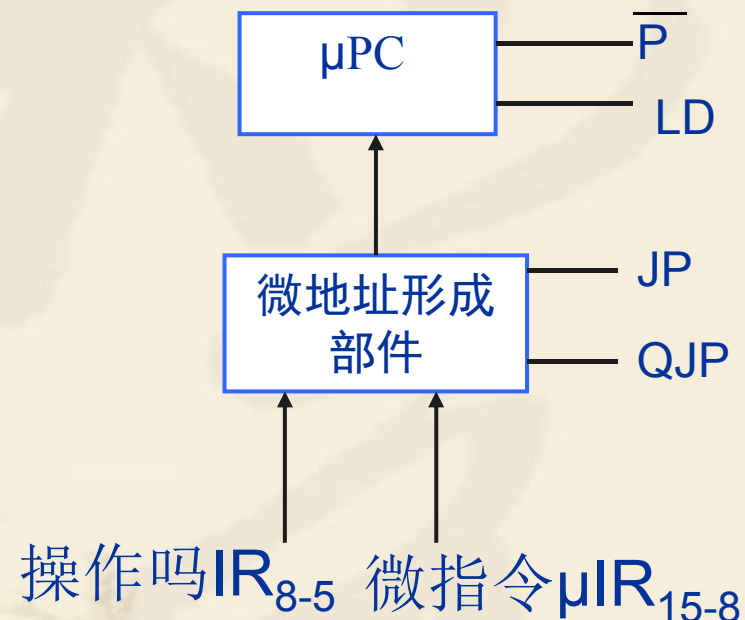
附图 8 3——8 译码器逻辑图

(3) 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。

其结构框图如图9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。

图9 后继地址形成部件是多路选择器

❖ μ PC的设计

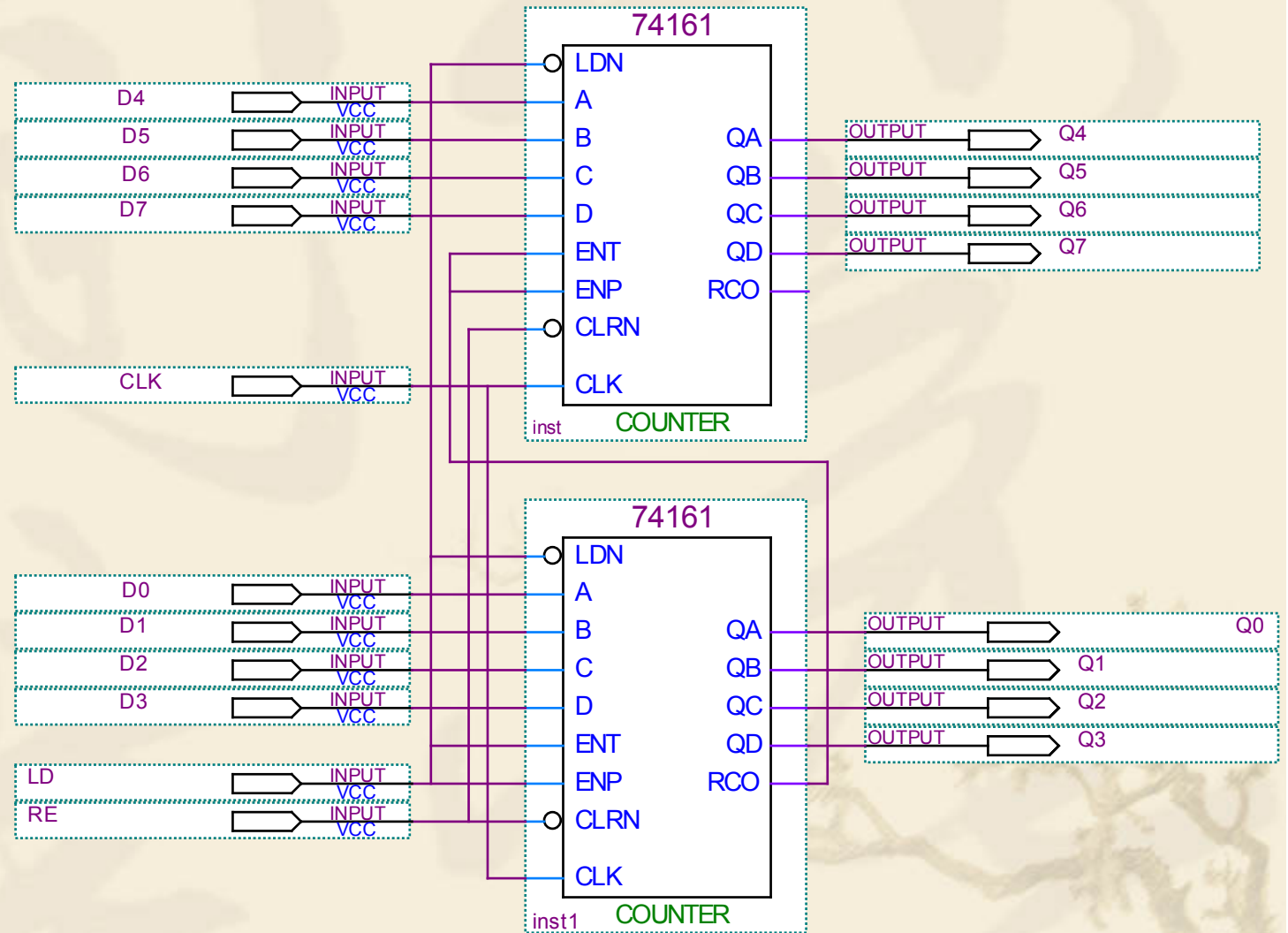
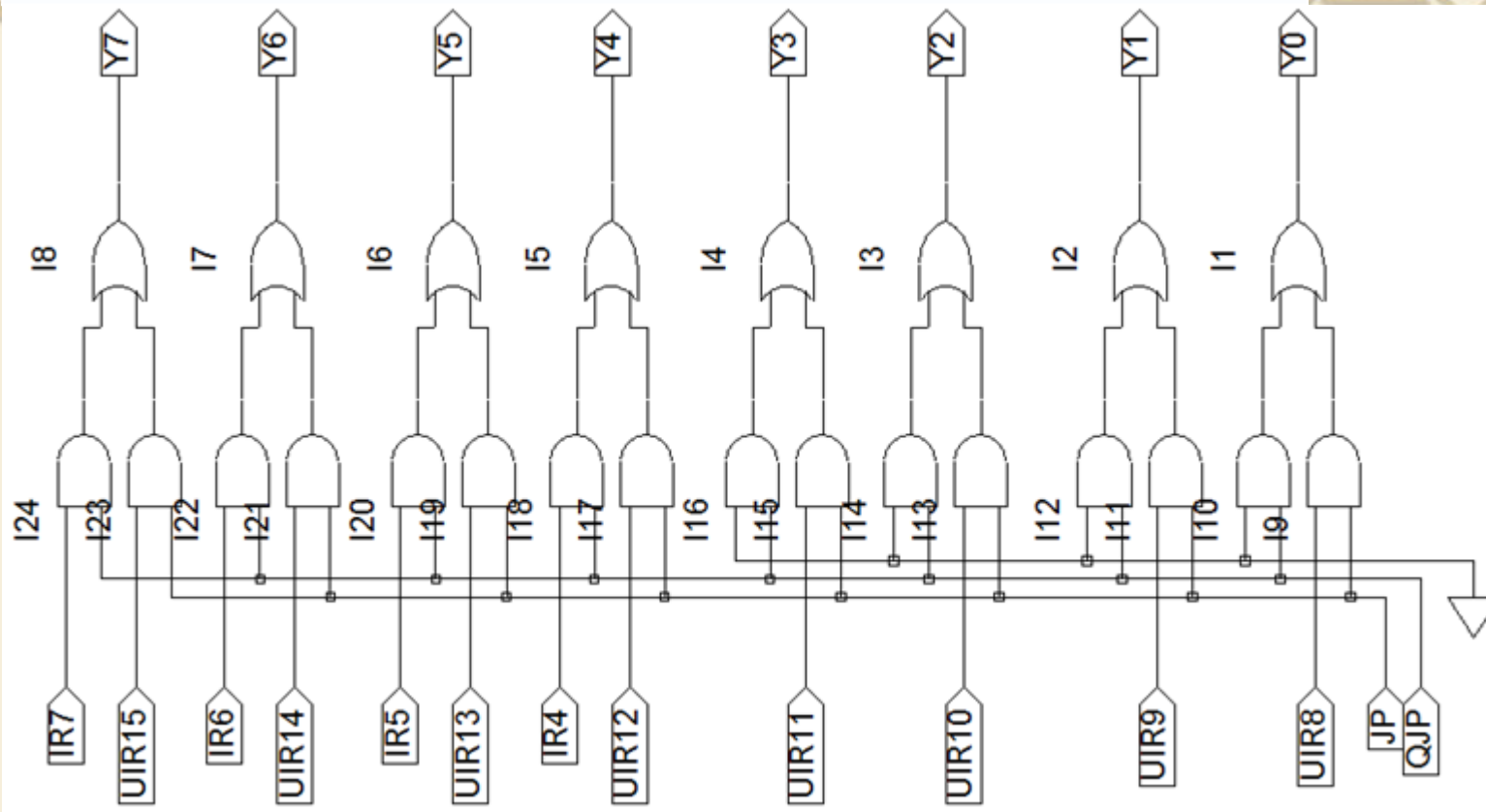


图9中的 后继地址形成电路



❖ 功能：多路选择器

⌘ 当JP=1, QJP=0时, $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0 = \mu IR_{15} \mu IR_{14} \mu IR_{13} \mu IR_{12} \mu IR_{11} \mu IR_{10} \mu IR_9 \mu IR_8$

⌘ 当JP=0, QJP=1时, $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0 = IR_7 IR_6 IR_5 IR_4 0000$

❖ 链接时, $Y_7Y_6Y_5Y_4Y_3Y_2Y_1Y_0$ 连接μPC的D7~D0, μPC的RE接高电平vcc。

4、微程序编写

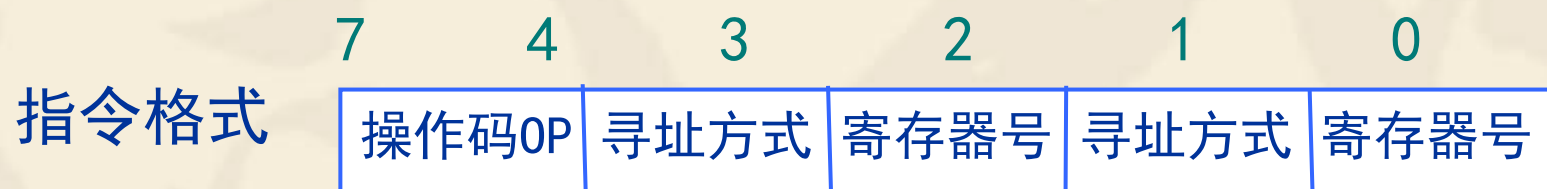
❖ 编写程序

- ❧ MOV1 05#, R₀
- ❧ MOV2 01#, R₁
- ❧ ADD R₀, R₁
- ❧ MOV3 R₁, (FA#)
- ❧ HALT

❖ (2) 操作码二进制代码

- ❧ MOV1: 0001
- ❧ MOV2: 0010
- ❧ ADD : 0011
- ❧ MOV3: 0100
- ❧ HALT: 0101

基本字长 8位



源操作数

目的操作数

❖ 指令类型

- ❧ 模型机有单操作数指令、双操作数指令和无操作数指令。
- ❧ 操作码OP共4位，最多可定义16条指令。

❖ 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，相应的寄存器号位为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，
 - ❖ 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。

❖ 程序代码:

地址	代码	地址	内容
0	0001 10 00	1	0000 0101
2	0010 10 01	3	0000 0001
4	0011 00 01		
5	0100 01 11	6	1111 0101
7	0101 00 00		

❖ (3) 微程序入口（十六进制代码）

❧ 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

❧ HALT执行周期微指令 入口：50H

❖ (4) 指令执行流程

❖ 00

❖ ↓

❖ RAM → IR

❖ ↓

❖ PC+1 → PC

❖ 10 ↓

❖ PC → MAR

❖ ↓

❖ PC+1 → PC

❖ ↓

❖ RAM → R₀

❖ ↓

❖ PC → MAR

❖ ↓

❖ JP

❖ 20 ↓

❖ PC → MAR

❖ ↓

❖ PC+1 → PC

❖ ↓

❖ RAM → R₁

❖ ↓

❖ PC → MAR

❖ ↓

❖ JP

❖ 30 ↓

❖ R₀ + R₁ → R₁

❖ ↓

❖ PC → MAR

❖ ↓

❖ JP

❖ 40 ↓

❖ PC → MAR

❖ ↓

❖ PC+1 → PC

❖ ↓

❖ RAM → MAR

❖ ↓

❖ R₁ → RAM

❖ ↓

❖ PC → MAR

❖ ↓

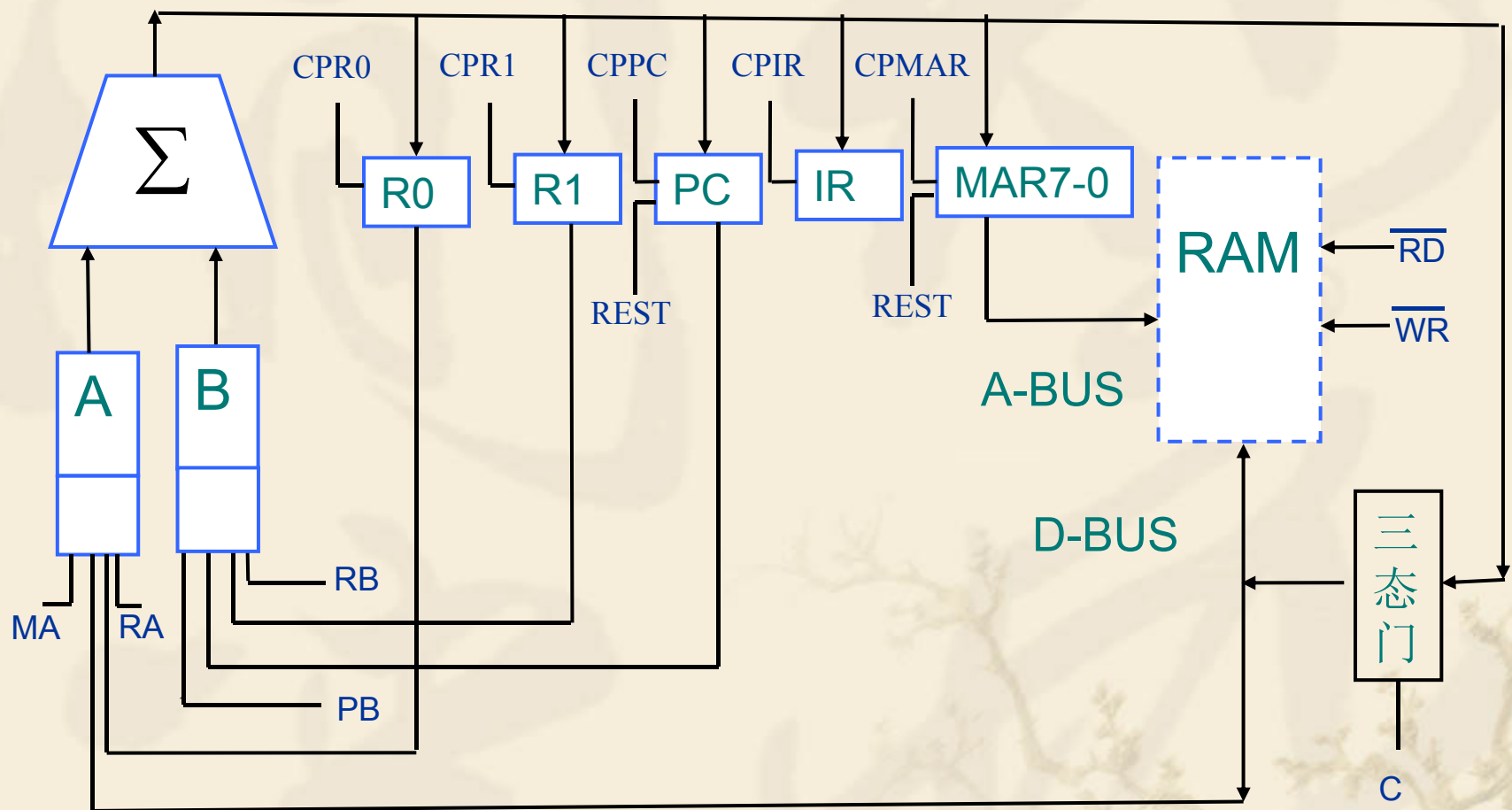
❖ JP

❖ ↓

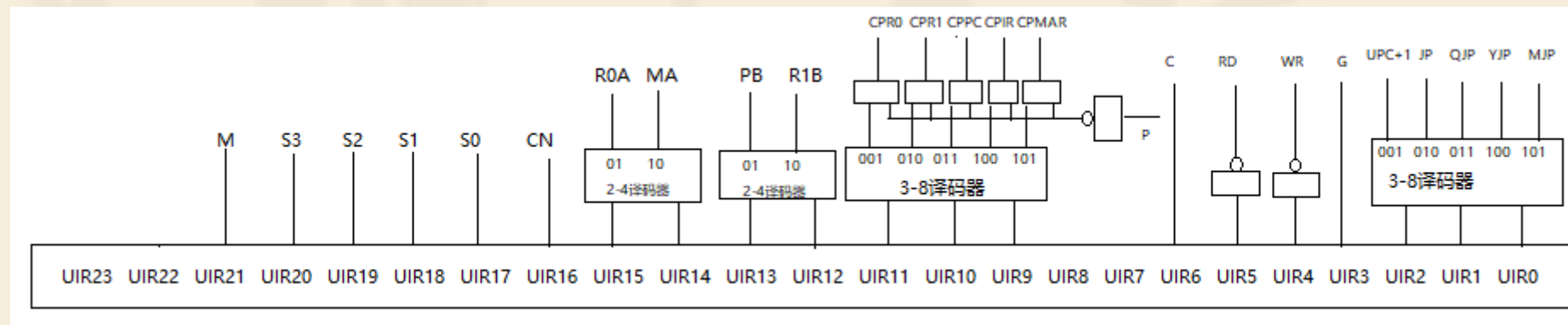
❖ G=1

(5) 编制微程序

- ❖ 根据指令流程和微指令格式就可以开始编制微程序。
- ❖ 指令流程中每一个流程对应一条微指令，结合总体结构框图图1，写出这个流程所对应的数据通路的控制命令。
- ❖ 例RAM→IR所需的控制命令是MA, \overline{RD} , CPIR并在表3-1中的相应位置填写上“1”，不需要的命令填写“0”。
- ❖ 另外每一条微指令都要确定下条微指令地址的生成方式。



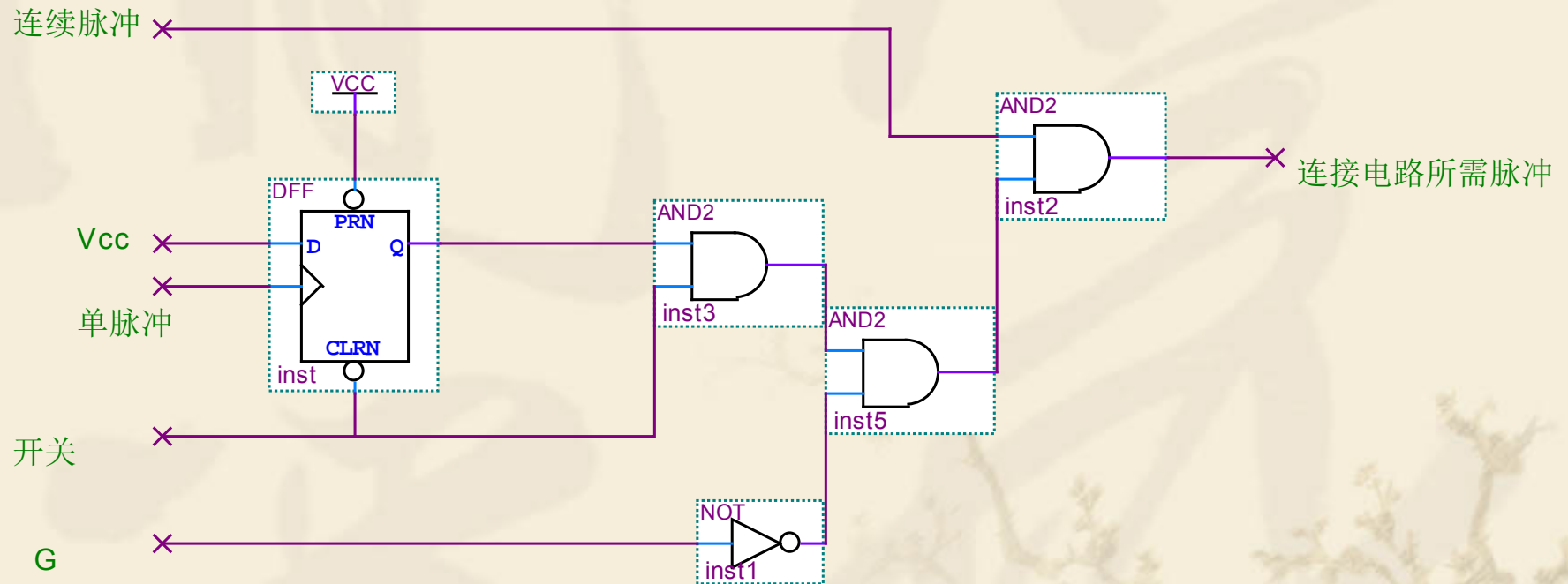
总体结构



❖ 微代码:

微操作	微地址	$\mu IR_{23}\mu IR_{22}$	$\mu IR_{21}\mu IR_{20}\mu IR_{19}\mu IR_{18}\mu IR_{17}\mu IR_{16}$	$\mu IR_{15}\mu IR_{14}$	$\mu IR_{13}\mu IR_{12}$	$\mu IR_{11}\mu IR_{10}\mu IR_9$	μIR_8	μIR_7	μIR_6	$\mu IR_5\mu IR_4$	$\mu IR_3\mu IR_2\mu IR_1\mu IR_0$	十六进制代码			
			M S3 S2 S1 S0 C0	A选择	B选择	输出分配			C	RD WR		转移方式	ROM#3	ROM#2	ROM#1
RAM→IR	00	00	1 1111 0	10	00	100	0	0	0	10	0	001	3E	88	21
PC+1→PC	01	00	0 1001 0	00	01	011	0	0	0	00	0	001	12	16	01
QJP	02	00	0 0000 0	00	00	000	0	0	0	00	0	011	00	00	03
PC→MAR	10	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
PC+1→PC	11	00	0 1001 0	00	01	011	0	0	0	00	0	001	12	16	01
RAM→R0	12	00	1 1111 0	10	00	001	0	0	0	10	0	001	3E	82	21
PC→MAR	13	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	14	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
PC→MAR	20	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
PC+1→PC	21	00	0 1001 0	00	01	011	0	0	0	00	0	001	12	16	01
RAM→R1	22	00	1 1111 0	10	00	010	0	0	0	10	0	001	3E	84	21
PC→MAR	23	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	24	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
R0+R1→R1	30	00	0 1001 1	01	10	010	0	0	0	00	0	001	13	64	01
PC→MAR	31	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	32	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
PC→MAR	40	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
PC+1→PC	41	00	0 1001 0	00	01	011	0	0	0	00	0	001	12	16	01
RAM→MAR	42	00	1 1111 0	10	00	101	0	0	0	10	0	001	3E	8A	21
R1→RAM	43	00	1 1010 0	00	10	000	0	0	1	01	0	001	34	20	51
PC→MAR	44	00	1 1010 0	00	01	101	0	0	0	00	0	001	34	1A	01
JP	45	00	0 0000 0	00	00	000	0	0	0	00	0	010	00	00	02
G=1	50	00	0 0000 0	00	00	000	0	0	0	00	1	000	00	00	08

- ❖ 在验收时，为使用连续脉冲，增加下列启动/停止电路。



❖ 5 调试

❖ 1、FPGA系统

- ❧ 平台上的所有开关和发光二极管（除L 之外）均随意编程用作数据输入和状态显示。
- ❧ 选择系统结构中典型部件进行功能测试看是否满足要求，若有错改之。
- ❧ 典型部件如下：选择器A
 - ❧ 带复位的寄存器MAR
 - ❧ 不带复位的寄存器R₀
 - ❧ 程序计数器PC

❖ 在部件设计无错、连线无错、FPGA的管脚定义无错时可生成下载文件下载到FPGA中。

❖ 2、单片机系统

❧ 微程序经过检查无误后以初始化文件的形式写入CROM的相应单元中。然后再读出检查看是否正确，有错改之。

❧ 汇编程序以十六进制代码从0单元开始写入RAM的相应单元中。

❖ 6统调

❖ 具体步骤如下：

❖ 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。

❧使微程序计数器UPC清0，保证从CROM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 $\overline{\mu\text{RD}}$), 将 CROM 0号单元的24位微指令代码读出, 用 $\overline{\mu\text{RD}}$ 的上升沿将微指令送入 μIR_{23-0} , 看是否正确。
- ❧ 第一条微指令产生的命令是: MA 、 $\overline{\text{RD}}$ 、 CPIR , 后继微地址产生方式为 $\mu\text{PC}+1$, 其操作是: $\overline{\text{RD}}$ 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。此时, 总线上的内容点亮 LR15-8 , 查看是否正确。
- ❧ 注意的是: 在没有按下次脉冲键前, 数据通路的内容一直不变。

- ❖ (2)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 $\mu PC+1$ 。该负脉冲的低电平用以读出 μPC 指示的第二条微指令。
- ❖ 要求：实现更多的指令，如减法、逻辑运算等。