

计算机组成原理

实验指导书

PROTEUS 版

计算机组成原理课题组
南通大学计算机科学与技术学院

VERSION I

目 录

实验 1 实验平台 Proteus 的使用	1
实验 2 运算器组成实验	4
实验 3 半导体存储器原理实验	8
实验 4 存储器系统组成实验	9
实验 5 简单数据通路的组成与故障分析实验	14
实验 6 指令与微程序控制器	16
课程设计	24
附 录	1
Proteus 的说明	1
自制器件参考实例	3
器件说明	11

实验1 实验平台Proteus的使用

一、实验目的

- 1.掌握 Proteus 的基本使用方法。
- 2.了解 74138 (3: 8) 译码器、74244b、74273b 的功能。
- 3.利用 Proteus 验证 74138 (3: 8) 译码器、74244b、74273b 的功能。
- 4.掌握器件封装及自制器件

二、实验任务

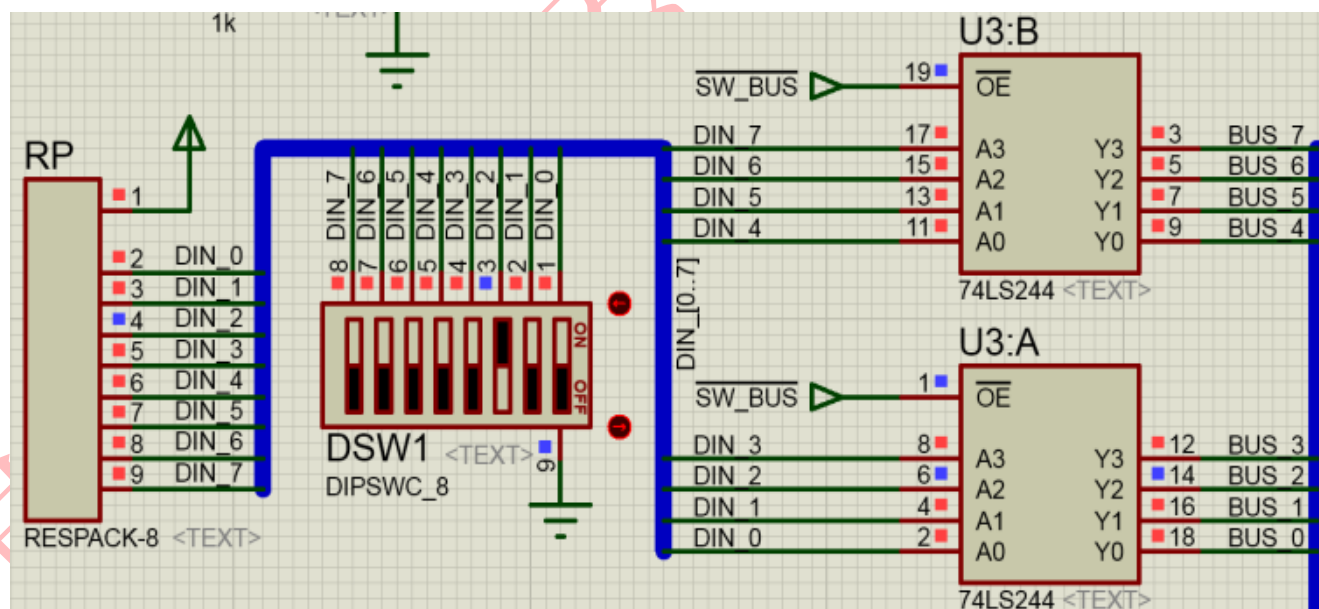
- 1.熟悉 Proteus 中管理项目、输入原理图以及仿真的设计方法与流程。
- 2.设计 74138、74244b 和 74273b 三种器件的功能验证电路图。
- 3.新建项目，利用原理图编辑方式输入电路，依照各器件功能表（见附录）分别进行仿真，验证这三种器件的功能。

三、实验要求

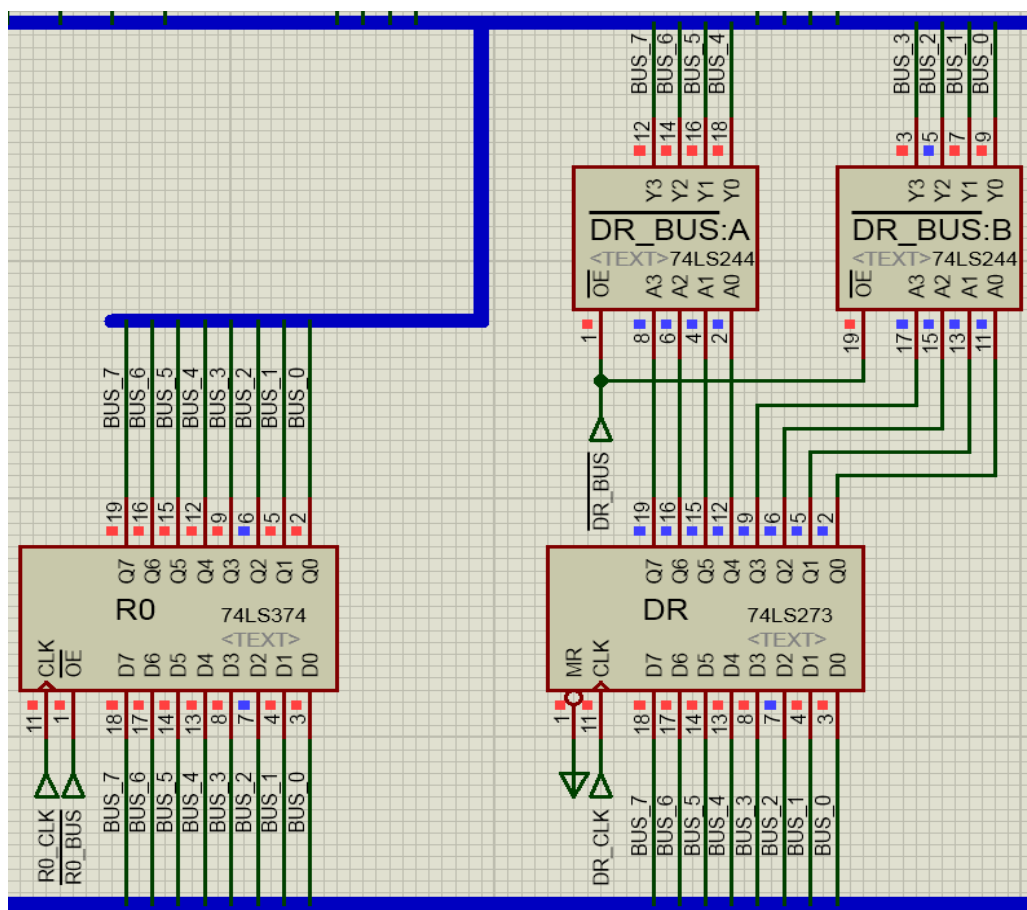
- 1.做好实验预习，掌握 74138、74244b、74273b 的功能特性。
- 2.写出实验报告，内容如下：
 - ①实验目的。
 - ②实验电路图。
 - ③完整的实验步骤。
 - ④74138、74244b 和 74273b 的功能仿真，有关输入输出信号要标注清楚。
 - ⑤仿真分析方法、分析过程和分析结果。

四、参考电路

- 1、利用拨码盘和 244 芯片构造总线



- 2、利用锁存器 273/373 构造数据寄存器，利用触发器 374 构造寄存器 R0。

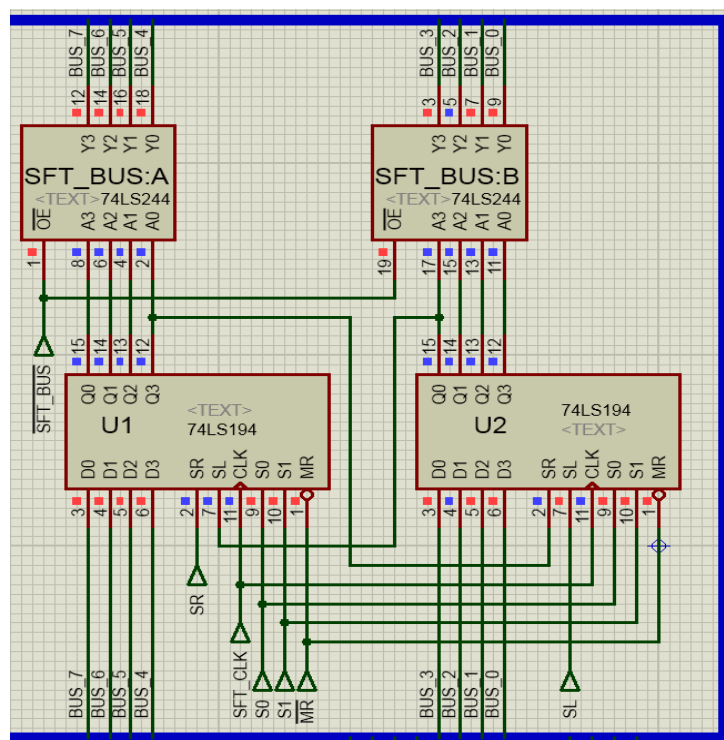


五、实验步骤

- 1) 令 $\#R0_BUS = \#DR_BUS = 1$; $\#SW_BUS = 0$, 启动仿真, 三态门 74LS244 导通, 手动拨码开关输入数据 0xAA 到总线, 观察此时寄存器 74LS374 和 74LS273 输出端的状态。
- 2) 令寄存器 R0 (74LS374) 的 R0_CLK 端上升沿跳变, 把总线上的数据 0xAA 存入 R0。
- 3) 令 $\#SW_BUS = 1$, 三态门 74LS244 阻断, 观察总线 BUS 的状态。
- 4) 令 $\#R0_BUS = 0$, 74LS374 输出选通, 观察总线 BUS 的状态。
- 5) 令寄存器 DR (74LS273) 的 DR_CLK 端上升沿跳变, 把总线上的 0xAA 数据存入 DR。观察寄存器 74LS273 的输出端。
- 6) 再令 $\#R0_BUS = 1$; 观察寄存器 74LS374 的输出端, 请比较器件 74LS244、74LS273 和 74LS374 的异同。
- 7) 手动拨码开关输入新数据 0x55 到总线 BUS ($\#SW_BUS = 0$)。此时, 新的数据会冲掉 R0 寄存器保存的原有数据 0xAA 么? 若再令 $\#R0_BUS = 0$, 会出现什么情况?
- 8) 假设手动拨码开关分别打入数据 0xAA 和 0x55 到 R0 寄存器 (74LS374) 和 DR 寄存器 (74LS273), 并且同时令 $\#R0_BUS = 0$ 和 $\#DR_BUS = 0$, 会出现什么情况? 在总线上可以同时选择多个寄存器输出 (导通输出端三态门) 么?

六、思考

1、移位寄存器构造



实验2 运算器组成实验

一、实验目的

1. 掌握算术逻辑运算单元（ALU）的工作原理。
2. 熟悉简单运算器的数据传送通路。
3. 掌握 8 位补码加/减法运算器的设计方法。
4. 掌握运算器电路的仿真测试方法。

二、实验说明

实验参考电路如下图所示，下图（a）是 1 位全加器的电路原理图，图（b）是由 1 位全加器采用行波进位方法设计的多位补码加/减法运算器。

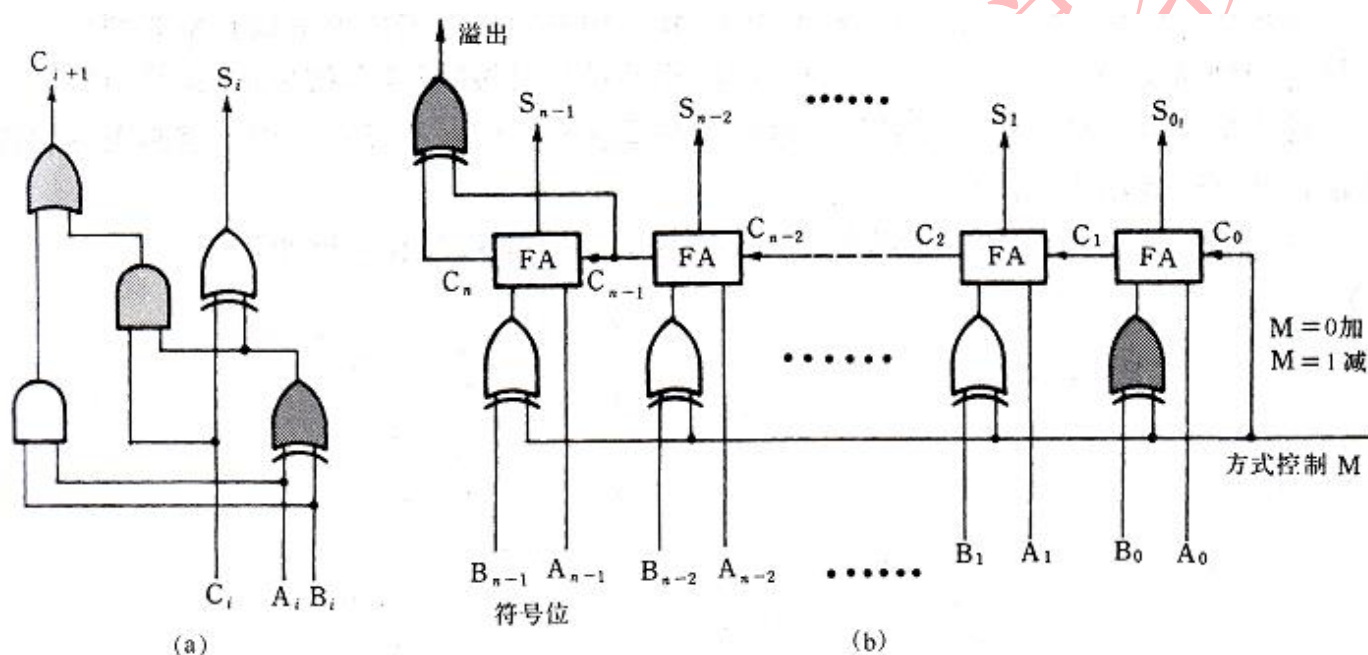


图 2.1 补码加/减法运算器

三、实验任务

1. 设计一个补码加/减法运算器
 - (1) 在 PROTEUS 里输入原理图，设计一个补码加/减法运算器。
 - (2) 对该补码加/减法运算器进行功能仿真测试。
 - (3) 测试通过后，封装成一个芯片。

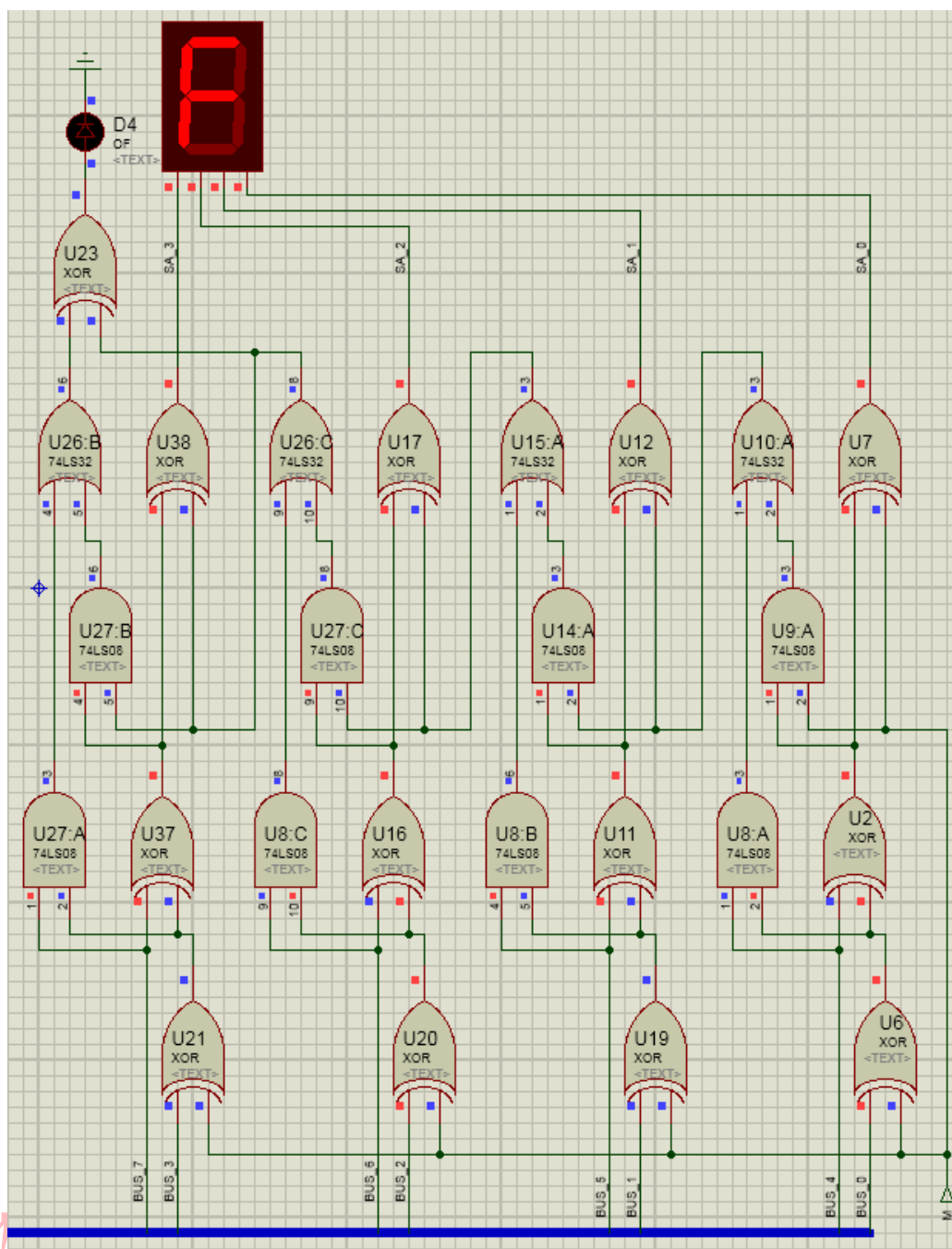


图 2.2 参考 4 位串行加法器

2. 设计运算器通路电路

结合实验一的数据总线，利用第 1 步设计的补码加/减法运算器芯片建立运算器通路。

3. 利用仿真，测试数据通路的正确性。

- 启动仿真，令 $BUS_{[7..4]}=0101$ ， $BUS_{[3..0]}=0010$ ， $M=0$ ，记录运算结果，是否溢出？如果改为 $BUS_{[3..0]}=0011$ ，结果如何？
- 令 $BUS_{[7..4]}=0101$ ， $BUS_{[3..0]}=0011$ ， $M=1$ ，记录运算结果，是否溢出？如果运算器输入改为 $BUS_{[7..4]}=0011$ ， $BUS_{[3..0]}=0101$ ， $M=1$ 不变，结果如何？
- 令 $BUS_{[7..4]}=1101$ ， $BUS_{[3..0]}=0011$ ， $M=0$ ，记录运算结果。是否溢出？如果改为 $M=1$ ，结果如何？

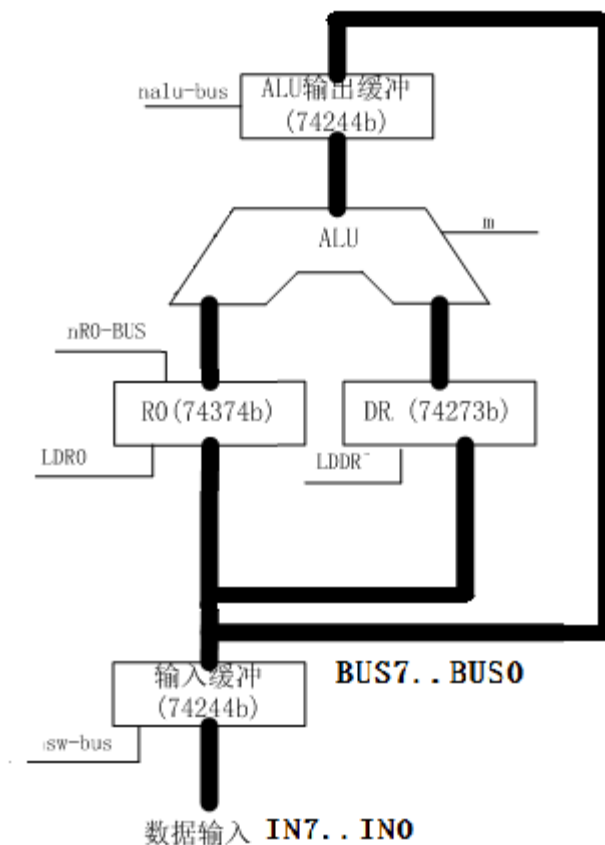


图 2.3 数据通路参考

四、实验要求

- (1) 做好实验预习，掌握运算器的数据传送通路和 ALU 的功能特性。
- (2) 实验完毕，写出实验报告，内容如下：
 - ①实验目的。
 - ②实验电路图。
 - ③按实验任务 3 的要求，参考下表，记录各控制信号的值及时序关系。表中的序号表示各控制信号之间的时序关系。要求表项自行设计，一个控制任务填一张表，并可用文字对有关内容进行说明。

序号	sw-bus	m	nalu-bus	IN7~IN0	BUS7~BUS0

- ④仿真结果的分析方法、分析过程和分析结果。
- ⑤实验体会与小结。

设计一个并行 8 位补码加/减法运算器

设计一个并行 8 位补码加/减法运算器



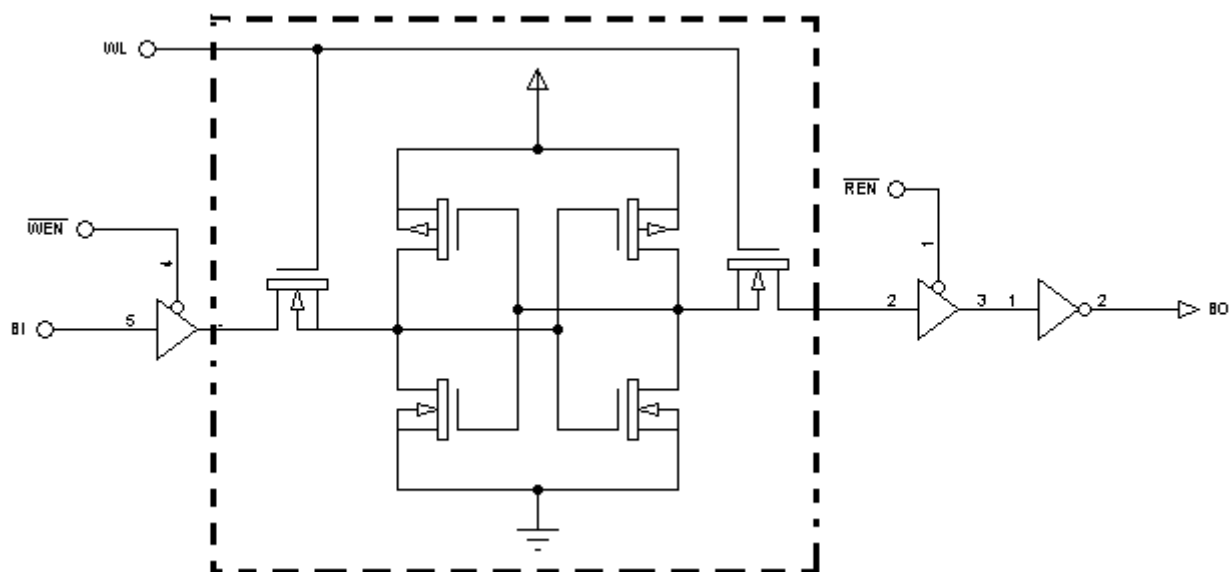
实验3 半导体存储器原理实验

一、实验目的

1. 熟悉存储器单元构造方法（RAM 为例）；
2. 熟悉半导体存储器存储和读出数据的过程；
3. 了解使用半导体存储器电路时的定时要求。

二、实验电路

参考实验电路如下：



三、实验任务

1. 利用 Proteus 器件库提供的 NMOS 管设计一个 1bit 的存储单元电路，并进行测试。
2. 对电路进行封装，构造自制芯片。
3. 设计一个 8×8 位的 RAM 芯片

四、实验要求

- (1) 做好实验预习，了解 ROM 和 RAM 存储器的功能特性和使用方法。
- (2) 写出实验报告，内容是：
 - ① 实验目的。
 - ② 实验所用的实验电路图。
 - ③ 实验仿真，仿真结果的分析方法、分析过程和分析结果。
 - ④ 实验体会与小结。

实验4 存储器系统组成实验

一、实验目的

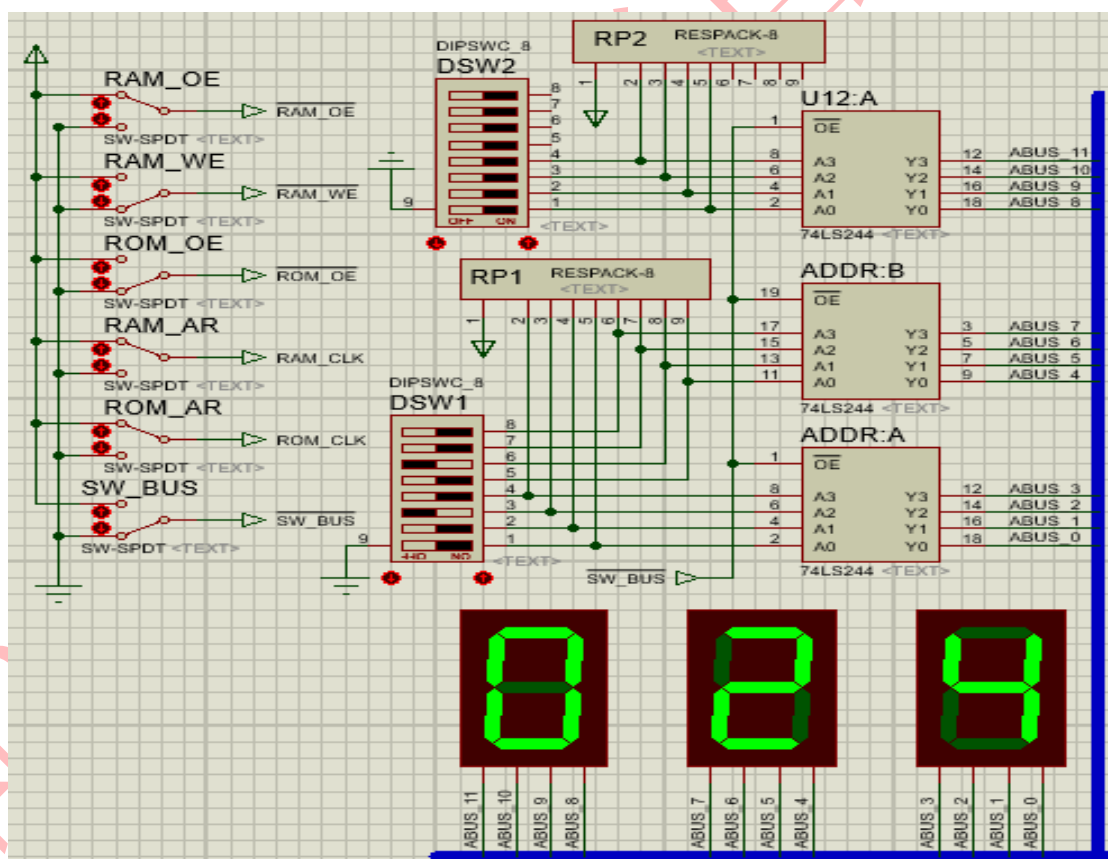
- 1.熟悉静态随机存储器 RAM 和只读存储器 ROM 的工作特性和使用方法；
- 2.熟悉半导体存储器存储和读出数据的过程；
- 3.了解使用半导体存储器电路时的定时要求。

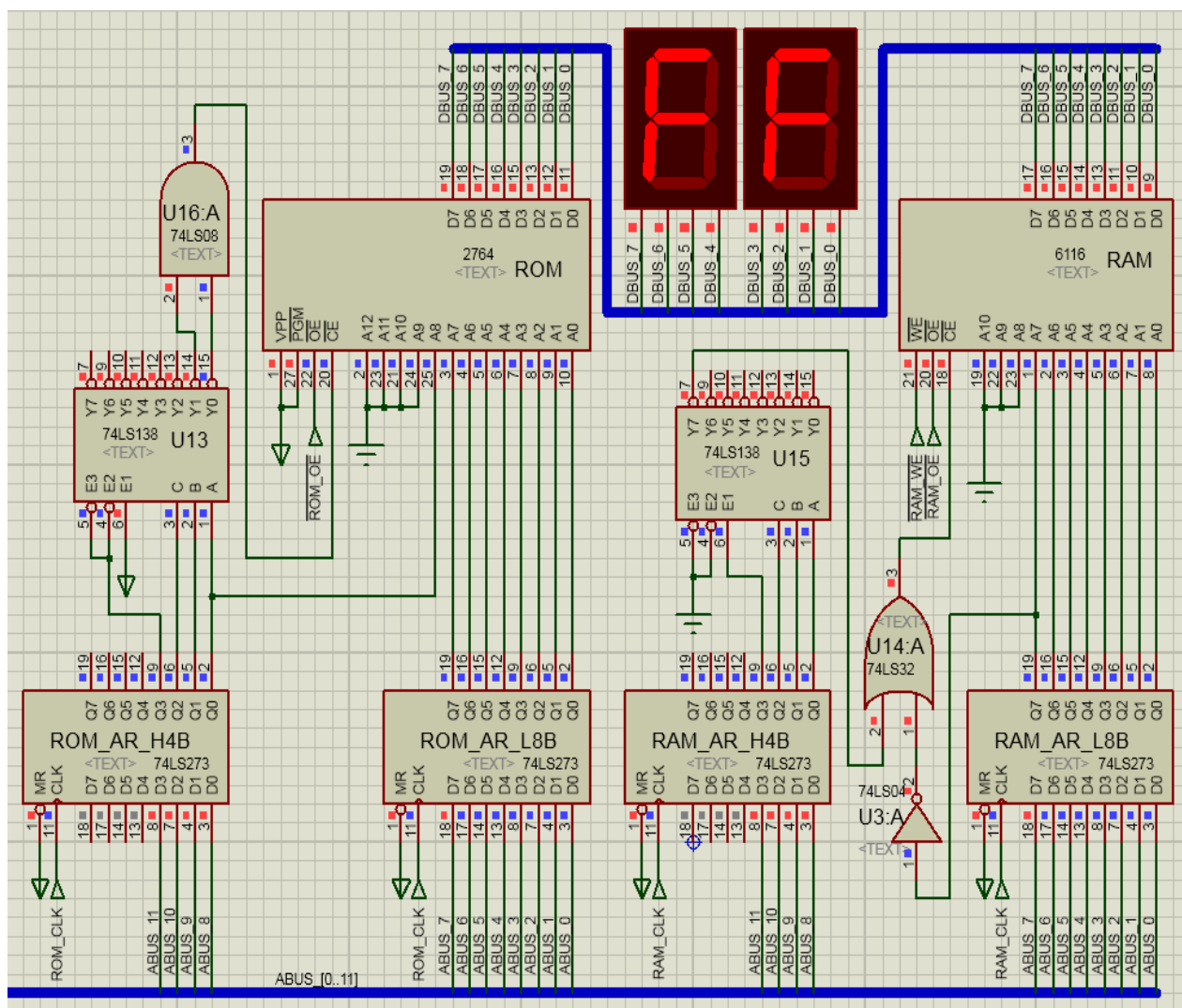
二、实验任务

- 1.利用 Proteus 器件库提供的 2114RAM 芯片设计一个由 $2K \times 8$ 位的 RAM（地址空间：200H 开始）构成的 RAM 存储器系统。
- 2.利用 Proteus 器件库提供的 2716ROM 芯片设计一个由 $1K \times 8$ 位的 ROM（地址空间：00H 开始）构成的 ROM 存储器系统。
 - (1) 设计实验电路图，在 Proteus 的编辑环境下，进行原理图的输入和编辑工作。
 - (2) 对 ROM 的存储单元 00H~05H 进行初始化。（注）
 - (3) 通过仿真，检查 ROM 工作及数据烧写的正确性。记录仿真波形、分析方法、分析过程和分析结果。

三、实验电路

参考实验电路如下：





四、实验要求

- (1) 做好实验预习，了解 ROM 和 RAM 存储器的功能特性和使用方法。
- (2) 写出实验报告，内容是：
 - ①实验目的。
 - ②实验所用的实验电路图。
 - ③实验仿真结果的分析方法、分析过程和分析结果。
 - ④实验体会与小结。

五、ROM 批量导入数据的技巧（注）

如何把一组程序或数据一次性批量导入 ROM 中，属于单片机应用中的关键方法，本课程加以借鉴利用，从而使 ROM 在往后的实验中可以充当程序存储器或数据存储器的角色。

本教程将借用 proteus 的 8051 汇编器中的伪汇编指令来实现上述功能。

1、新建文本文件，按照下列格式输入数据（注：16 进制 A~F 前面要加 0），保存为 Program.txt（注：文件名区分大小写），然后把.txt 后缀改名为.asm 后缀。

```
Program - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
ORG      0000H
DB       01010101B
DB       01010101B
DB       01010101B
DB       01010101B

DB       10101010B
DB       10101010B
DB       10101010B
DB       10101010B

ORG      0024H
DB       0AAH
DB       0AAH
DB       0AAH
DB       0AAH

DB       55H
DB       55H
DB       55H
DB       55H

END
```

解释:

“ORG xxxxH”规定 ORG 语句后所跟数组存储的首地址，数组末尾必须以其他 ORG 语句或“END”作为结束。在 asm 文件中可以使用多个“ORG”语句来规定在存储器的不同位置存放不同长度的数组。

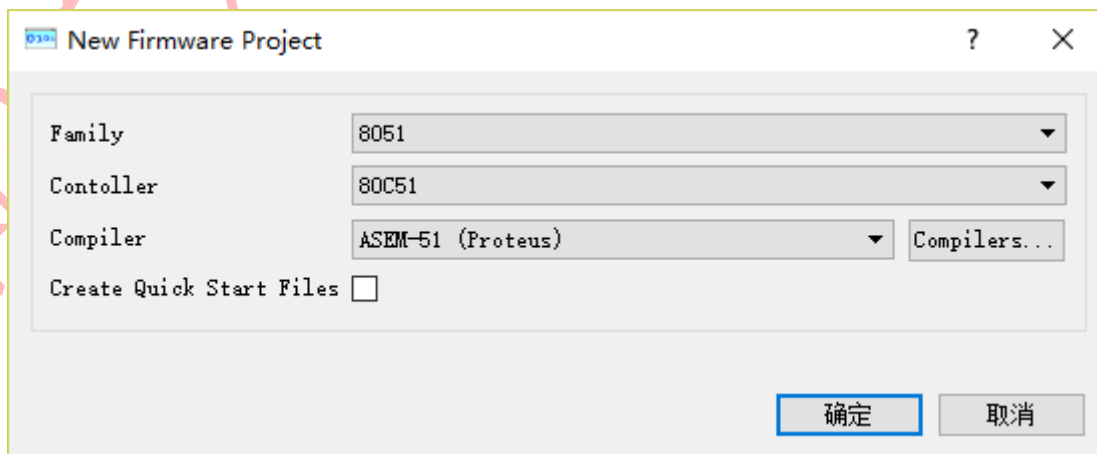
在“ORG”语句后的“DB xxxxxxxxB”或“DB xxH”语句都表示一个存储单元存放的 8 位数据，前者是二进制，x 表示 0 或 1；后者是十六进制，x 表示 0~9 和 A~F（若 x=A~F，则要写成 0AxH~0FxH）。

注：如果已经通过 ORG 语句定义的数据段需要清零，需要写一段全零的 ORG 语句来覆盖，否则数据永远存在。

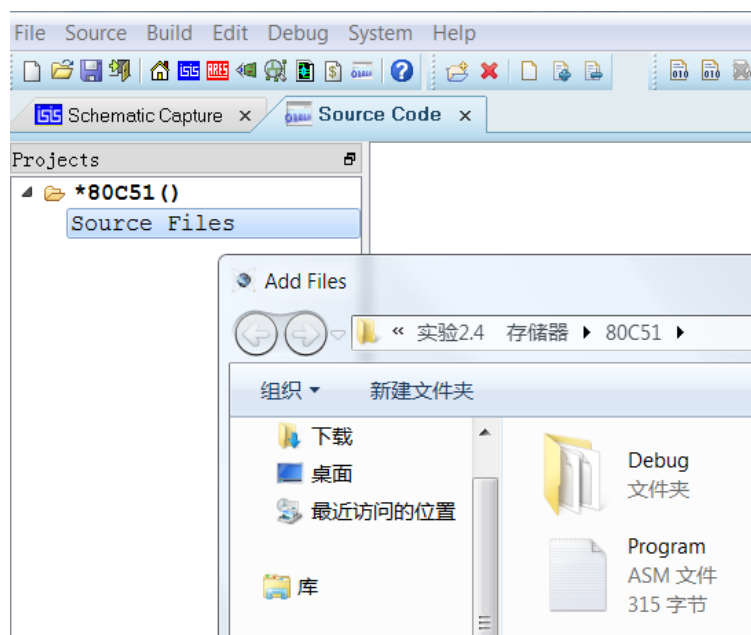
2、在快捷菜单栏里打开 Source Code 面板:



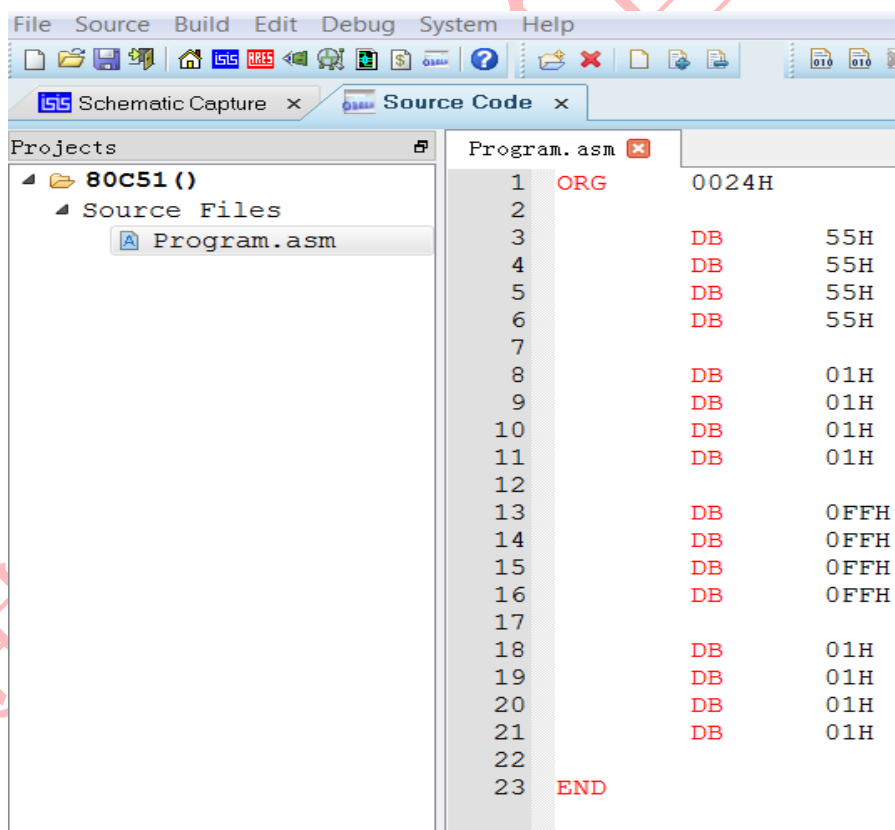
3、在菜单栏里找到 Source->Create Project; 弹出的对话框中，如图选择：（注：Creat Quick Start Files 不勾）



4、标签页上会出现“*80C51()”的新 project 后，右键单击下方的“Source Files”，选择 Add files （或 Import Existing File ）选项，在弹出的视窗中选择所需的 asm 文件，按“确定”就添加到当前的 project 中



5、在 project 中已经添加的 asm 文件上双击，右侧打开 asm 文本内容。若要更换 asm 文件，可以右键单击 asm 文件，菜单列表中选择“Remove file”删除当前 asm 文件，然后再选择新的 asm 文件添加到 project 中。



若第一次编译 asm 文件，则须右键单击 asm 文件，在弹出菜单中点击 Project Settings，弹出框中 controller 选项选择“80C51”，注意：要去掉勾选 Embed files（或 Attach files）！点击“OK”选项。然后右键单击 asm 文件，在弹出框内点击 Build Project 执行编译。

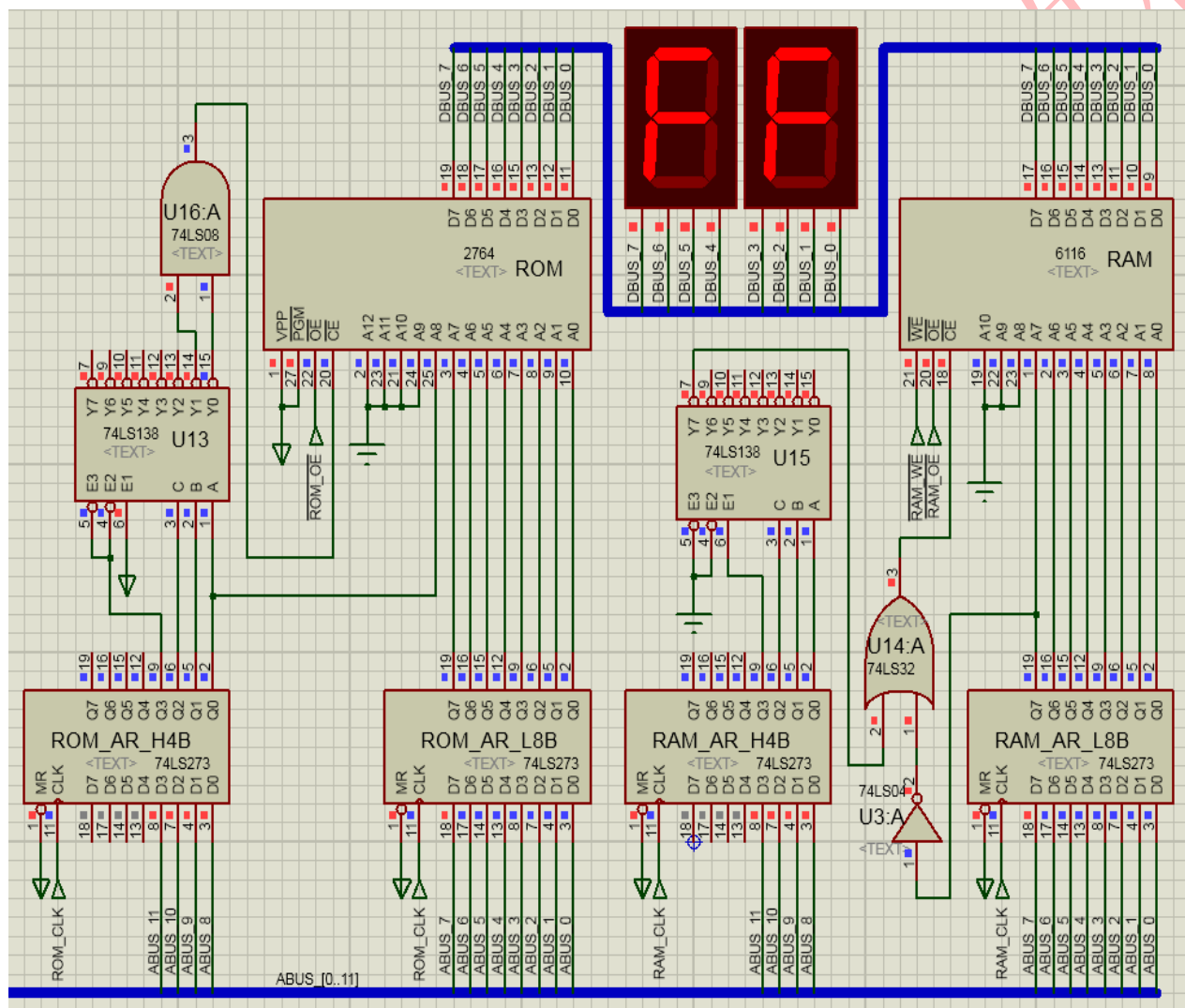
实验5 简单数据通路的组成与故障分析实验

一、实验目的

- (1) 了解数据通路的连接与工作时序；
- (2) 进一步熟悉计算机的数据通路；
- (3) 分析问题与解决问题的能力，学会在出现故障的情况下，独立分析故障现象并排除故障。

二、实验电路

整个电路总线结构的形式自行设计。参考电路原理图见下。



三、实验任务

利用 Proteus 器件库提供的 6116RAM 芯片，2716ROM 芯片设计一个由 $1K \times 8$ 位的 ROM（地址空间：00H 开始）和 $1K \times 8$ 位的 RAM（地址空间：F00H 开始）构成的存储器系统。

- (1) 设计实验电路图，在 Proteus 的编辑环境下，进行原理图的输入和编辑工作。
- (2) 对 ROM 的存储单元进行初始化。
- (3) 通过仿真，检查 ROM 工作及数据烧写的正确性。记录仿真波形、分析方法、分析过程和分析结果。

四、实验操作步骤：

- 1) 依照“附录：ROM 批量导入数据技巧”，加载 project.asm 文件编译的 hex 二进制文件到 ROM 芯片 2716，并且查看 ROM 烧写的数据段是否正确。

2) 启动仿真前, 令 $\#ROM_OE = \#RAM_OE = \#RAM_WE = 1$; 启动仿真后, 令 $\#SW_BUS = 0$, 手动拨码开关输入 024H 到地址总线 $ABUS_{[0..11]}$ (绿色数码管显示)。

3) 令地址锁存信号 ROM_CLK 上升沿跳变 “0→1”, 将地址总线上的 024H 打入地址锁存器 ROM_AR ; 令 $\#ROM_OE=0$, 使能 ROM 存储器 2764 输出, 在数据总线 $DBUS_{[0..7]}$ (红色数码管显示) 上查看存储单元[024H]读出的内容。

4) 手动拨码开关, 向地址锁存器 RAM_AR 打入地址 F80H; 令 $\#RAM_WE = 0$, 使能 RAM 存储器 6116 输入, 把存储单元[024H]的内容写入存储单元[F80H]。(打开 Memory Contents 进行验证)

5) 令 $\#ROM_OE = 1$ (禁止 ROM 存储器 2764 输出) 且 $\#RAM_OE = 0$ (允许 RAM 存储器 6116 输出), 在数据总线 $DBUS_{[0..7]}$ 上观察存储单元[F80H]写入内容是否正确。

6) 按照上述操作, 把 ROM 存储器单元[024H]、[028H]、[02CH]、[030H]的内容依次写入 RAM 存储器单元[F80H]、[F81H]、[F82H]、[F83H], 查看写入 RAM 数据是否正确。

五、实验要求

(1) 做好实验前的预习工作, 掌握实验电路的数据通路特点和集成电路的功能特性。

(2) 写出实验报告, 内容为:

①实验目的。

②实验电路图。

③实验数据记录 (仿真波形及仿真结果的分析方法、分析过程和分析结果)。

④写出本次实验的心得, 讨论实验中遇到的问题。

六、思考

设计实验电路图, 把前面进行的运算器实验模块与存储器实验模块两部分电路连接在一起。RAM 和 ROM 的输出应能送至寄存器 DR 作为运算器的输入, 而运算器的结果应既可以送入 R0 暂存, 又可以送入 RAM 的指定单元。

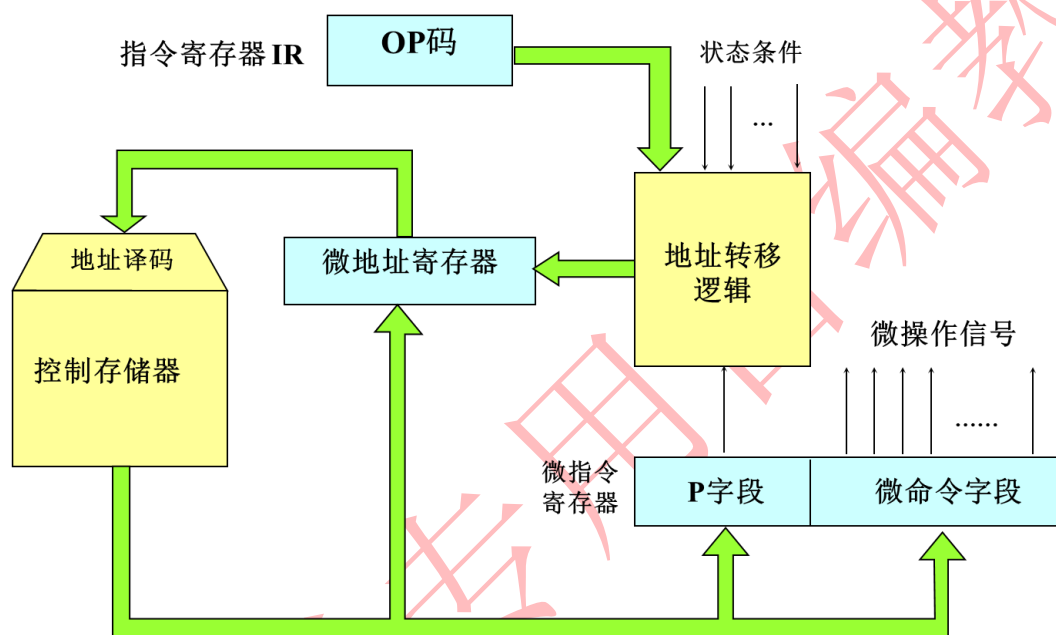
实验6 指令与微程序控制器

一、实验目的

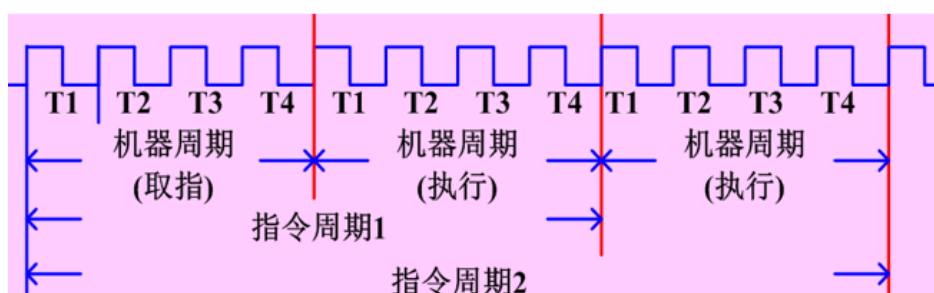
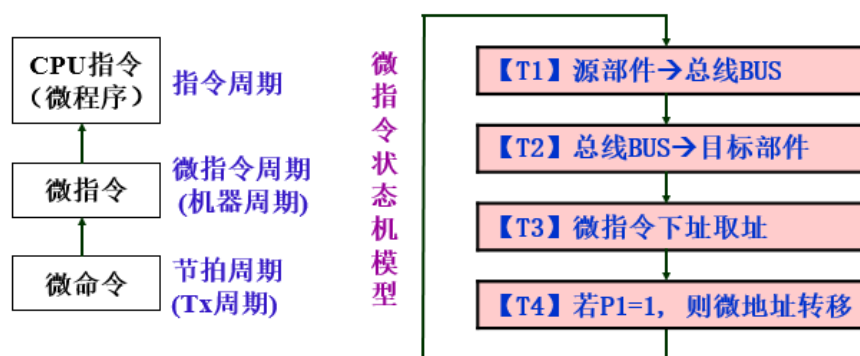
- (1) 理解“微程序”思想和“指令-微指令-微命令”微程序结构。
- (2) 掌握微程序控制器的结构和设计方法。
- (3) 分析问题与解决问题的能力，学会在出现故障的情况下，独立分析故障现象并排除故障。

二、实验说明

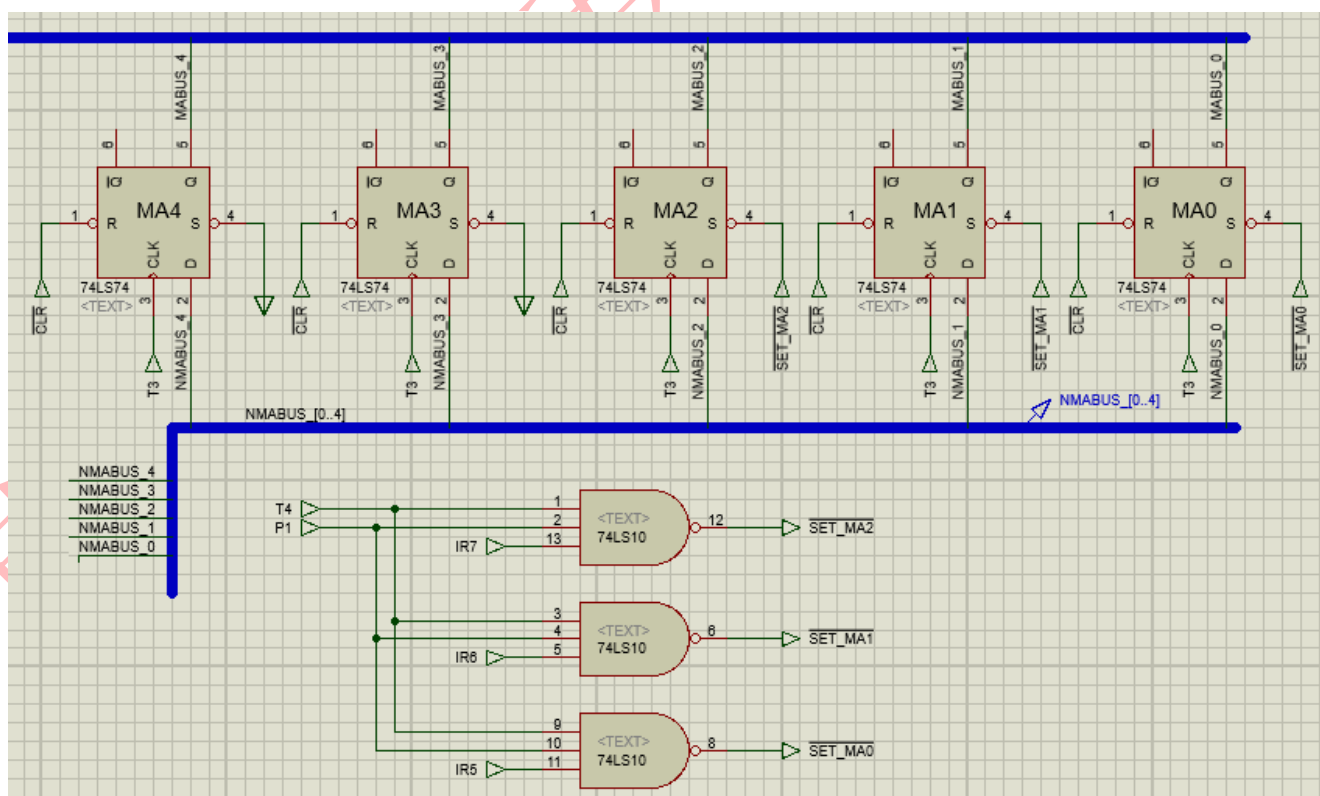
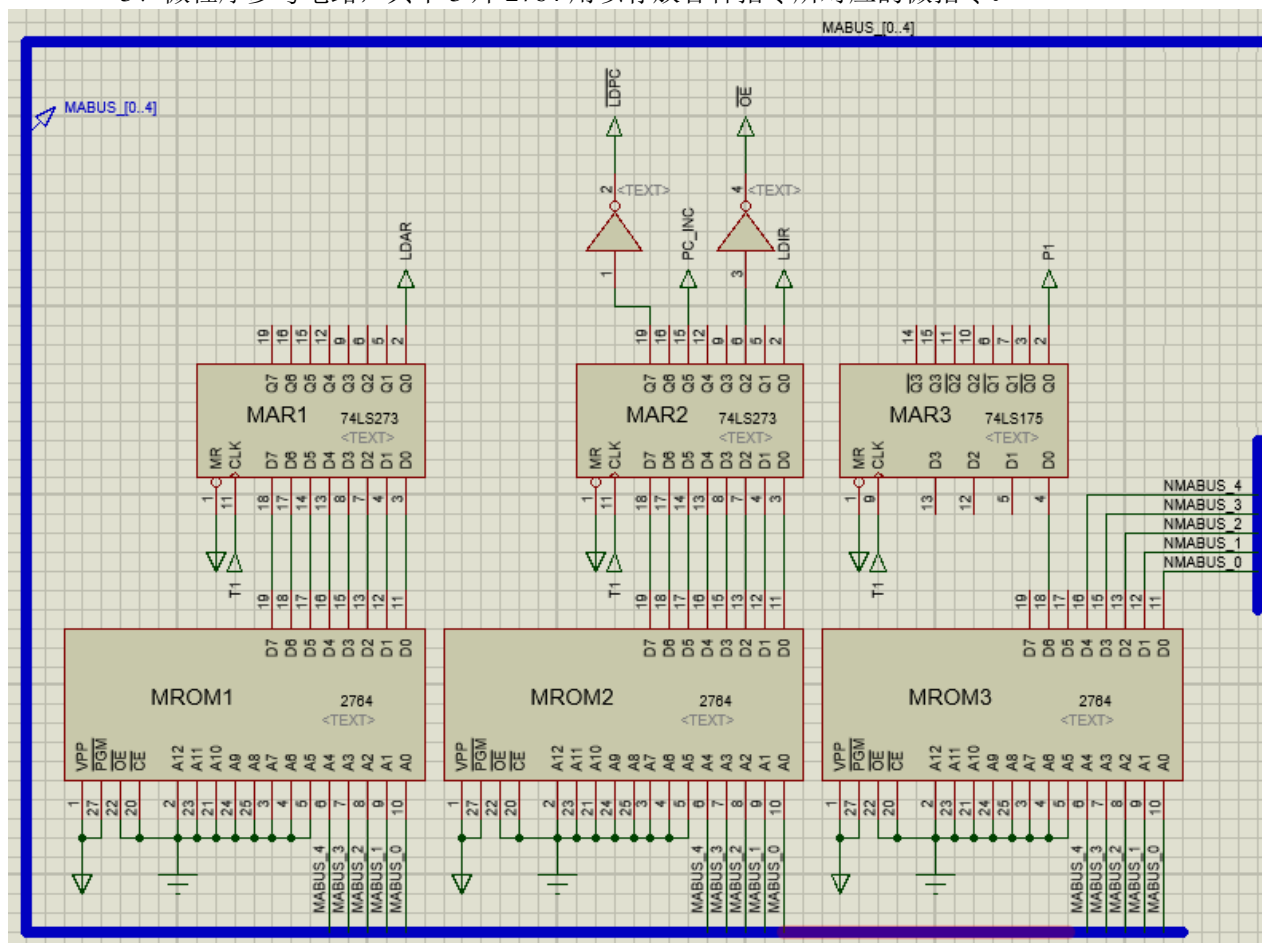
1、微程序结构示意图



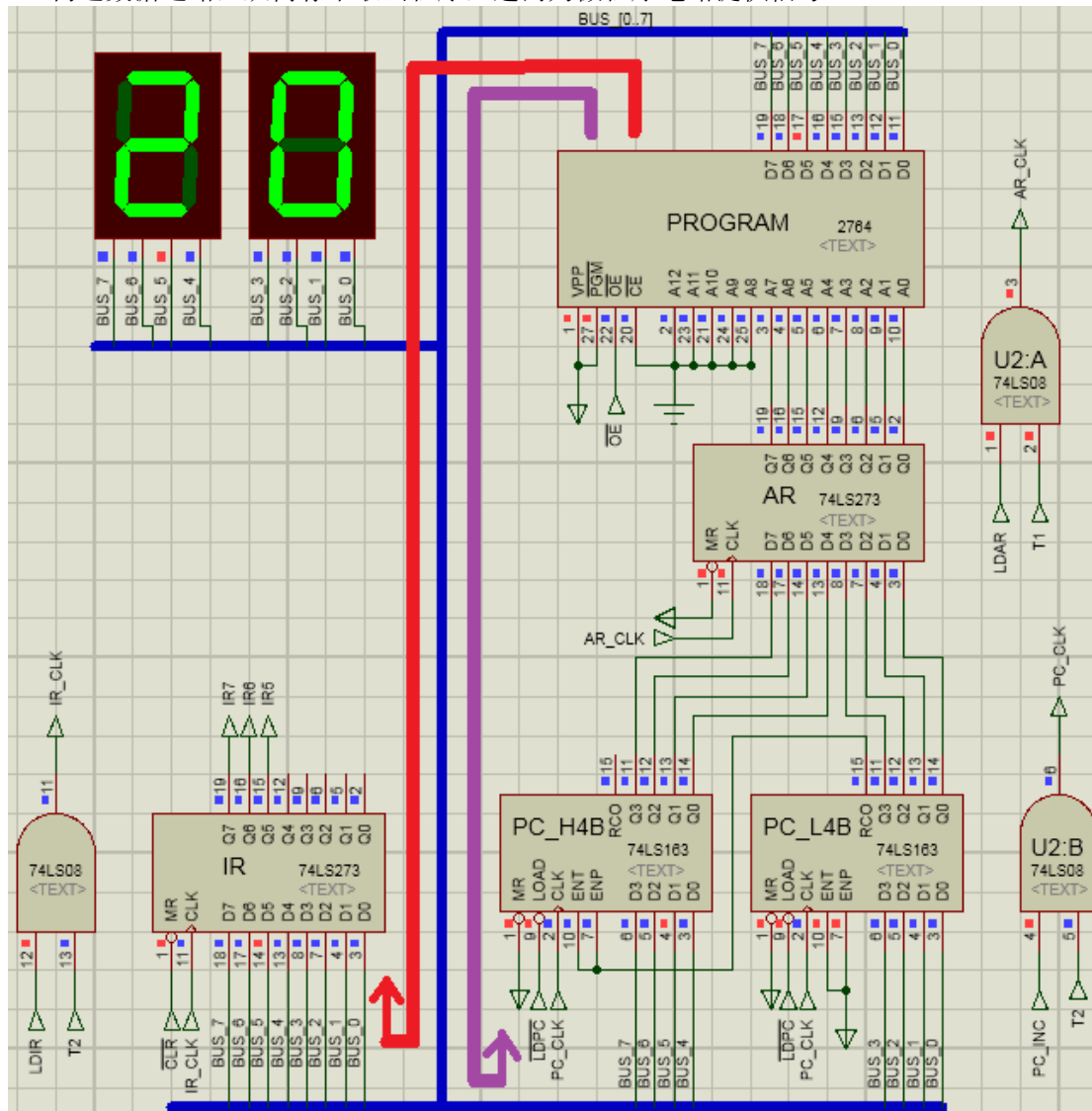
2、指令的“微程序”架构



3、微程序参考电路，其中 3 片 2764 用以存放各种指令所对应的微指令。



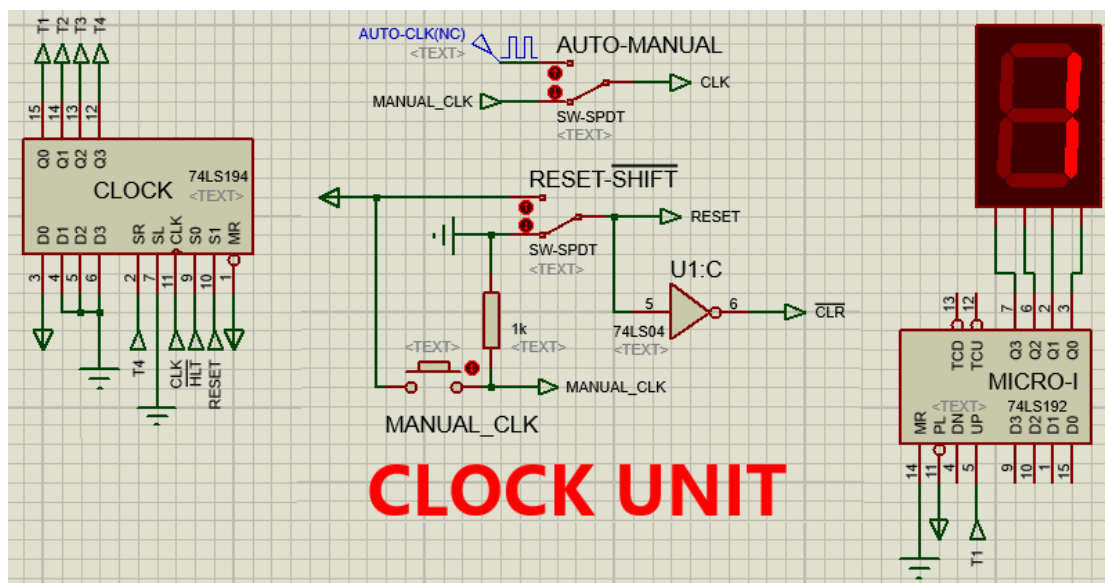
4、构造数据通路，从内存中取出程序，进而为微程序电路提供信号



四、实验步骤:

设计一个“最简版本”的 CPU 模型机：利用时序发生器来产生 CPU 的预定时序，通过微程序控制器的自动控制，在数据通路中完成唯一的 CPU 功能——程序跳转。

1) 构造时钟发生器。



2) 设计所需实现的机器指令及其微指令。应有以下步骤:

A、整理所需实现的指令

CPU指令格式

NOP 0000 XX XX

HLT 1110 XX XX

“断点”：硬件停机

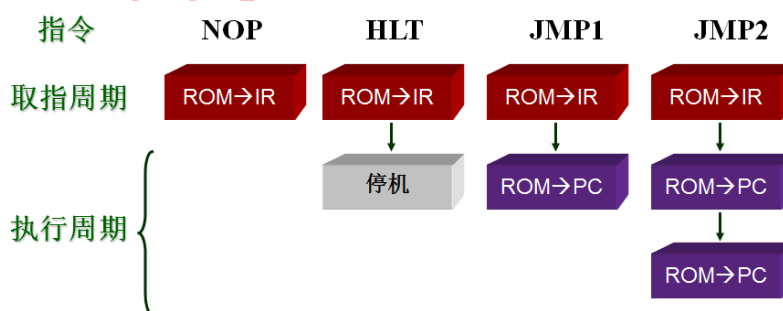
JMP1 0010 XX XX
addr1

直接寻址: addr1 → PC

JMP2 0100 XX XX
addr1

间接寻址: [addr1] = addr2, addr2 → PC

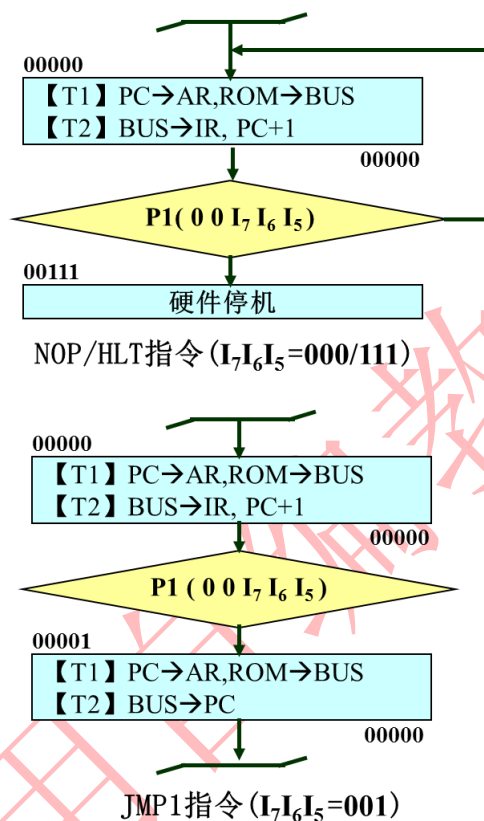
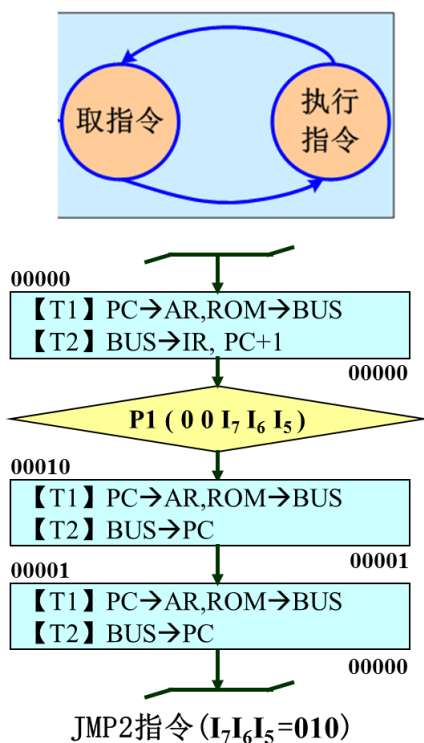
B、分析指令实现的操作



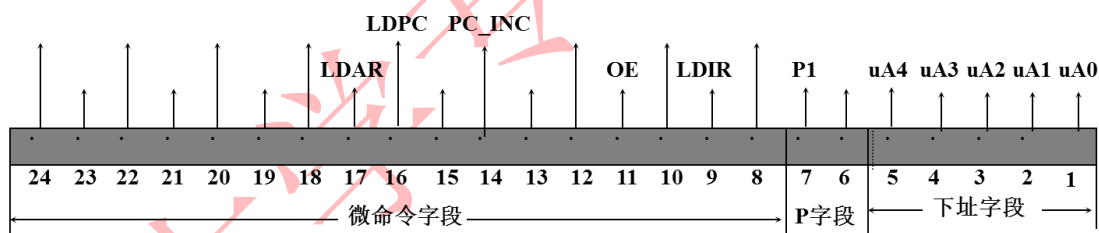
		有效的微操作信号	功能
指令流	T1	#OE, AR_CLK (LDAR)	PC → AR, ROM → BUS
	T2	#OE, IR_CLK (LDIR), PC_CLK (PC_INC)	BUS → IR, PC+1
数据流	T1	#OE, #LDPC, AR_CLK (LDAR)	PC → AR, ROM → BUS
	T2	#OE, #LDPC, PC_CLK (PC_INC)	BUS → PC

C、解析指令执行的微程序流程

微程序流程图



D、设计微指令



微指令结构图

微命令字段中每一位表示一个微命令：第N位 = $\begin{cases} 1 & \text{有微操作} \\ 0 & \text{无微操作} \end{cases}$

微指令代码表

Addr	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
00000	0	0	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0
00001	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0
00010	0	0	0	0	0	0	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	1
00111	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

3) 将设计成功的微指令，分别烧写至相应存储器。

EPROM1烧写内容	EPROM2烧写内容	EPROM3烧写内容
ORG 0000H	ORG 0000H	ORG 0000H
DB 00000001B	DB 00100101B	DB 01000000B
DB 00000001B	DB 10100100B	DB 00000000B
DB 00000001B	DB 10100100B	DB 00000001B
DB 00000000B	DB 00000000B	DB 00000000B
DB 00000000B	DB 00000000B	DB 00000000B
DB 00000000B	DB 00000000B	DB 00000000B
DB 00000000B	DB 00000000B	DB 00000000B
DB 00000000B	DB 00000000B	DB 00000000B
END	END	END

4) 利用机器指令编写程序，烧写至程序存储器 PROGRAM。

汇编助记符	注释	(M地址: 机器指令)
JMP1, 06H	程序跳转到地址06H执行 06H→PC	00H:00100000
		01H:00000110
HLT	停机	02H:11101010
NOP/Addr	空/【地址】	03H:00001010
NOP	空	04H:00000000
NOP	空	05H:00000000
NOP/Addr	空/【地址】	06H:00000010
HLT	停机	07H:11100001
JMP2, [06H]	程序跳转到地址[06H]执行 [06H]=02H, 02H→PC	08H:01000000
		09H:00000110
	未定义	0AH:11111111
		0BH:11111111

```
J3.asm - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

ORG    0000H

        DB      00100000B ; JMP1, 06H
        DB      00000110B
        DB      11101010B ; HLT
        DB      00001010B ; NOP/Addr

        DB      00000000B ; NOP
        DB      00000000B ; NOP
        DB      00000010B ; NOP/Addr
        DB      11100001B ; HLT

        DB      01000000B ; JMP2, [06H]
        DB      00000110B
        DB      11100000B ; HLT
        DB      00000011B ; NOP/Addr

END|
```

5) 仿真观测并记录

准备仿真前,时钟信号 CLK 接在 MANUAL_CLK 端;启动仿真,复位信号 RESET=1,然后手动按钮 MANUAL_CLK 一次,令时钟信号 CLK 上升沿跳变,初始化节拍 {T1,T2,T3,T4} = {1,0,0,0};最后,令复位信号 RESET=0,初始化过程完成。

手动按钮 MANUAL_CLK,单步执行上述机器语言程序。在 JMP1 或 JMP2 指令的指令周期中,对照微程序流程图,观察每一条微指令的作用,以及单步执行的结果(例如寄存器 AR、IR、PC 及总线 BUS 上的数据)。

五、实验要求

(1) 做好实验前的预习工作,掌握实验电路的数据通路特点和集成电路的功能特性。

(2) 写出实验报告,内容为:

- ① 实验目的。
- ② 实验电路图。
- ③ 实验数据记录(仿真波形及仿真结果的分析方法、分析过程和分析结果)。
- ④ 写出本次实验的心得,讨论实验中遇到的问题。

六、思考

1、增加一条二次间接寻址的跳转指令 **JMP3**。请补充微程序流程图及微指令代码表（新增微指令地址[00011]），实现如上所示 **JMP3** 指令的功能。

JMP3	0110	XX	XX
	addr1		

二次间址：[addr1]=addr2, [addr2]=addr3, addr3→PC

2、此版本 CPU 最多有多少条微指令？最多有多少条 CPU 指令？微指令和 CPU 指令的容量分别由什么因素限定？

3、请问微程序控制器“状态机”可否提升效率，减少到三个状态{T1,T2,T3}？即微指令周期可否减少到只用 T1、T2、T3 三个节拍即可完成一条微指令从取指到执行的全过程？

4、在本实验的 CPU 模型机上增加两个 74LS173 寄存器 R1 和 R0，扩展 CPU 指令集，增加下述 MOV/SET 指令及相应的微指令。注：IMM 是 8 位立即数；RA 和 RB 是在指令“功能”描述中的逻辑寄存器，可以对应 R0 或 R1 寄存器。

汇编助记符	功能	I ₇ I ₆ I ₅ I ₄	I ₃ I ₂	I ₁ I ₀
MOV RA, RB;	(RB)→RA	0110	RA	RB
SET RA, IMM;	IMM→RA	0011	RA	x/x
		IMM		

5、在上述思考题的电路基础上，参考上述“2.5 运算器实验”，再增加运算器电路，扩展 CPU 指令集，增加下述 ADD/SUB/AND/OR/XOR 指令及相应的微指令：

汇编助记符	功能	I ₇ I ₆ I ₅ I ₄	I ₃ I ₂	I ₁ I ₀
ADD RA, RB;	(RA) + (RB) → RA	1101	RA	RB
SUB RA, RB;	(RA) - (RB) → RA	1100	RA	RB
AND RA, RB;	(RA) ∧ (RB) → RA	1110	RA	RB
OR RA, RB;	(RA) ∨ (RB) → RA	1111	RA	RB
XOR RA, RB;	(RA) ⊕ (RB) → RA	1011	RA	RB

课程设计

一、设计目的

1. 融会贯通教材各章的内容，通过知识的综合运用，加深对计算机系统各模块的工作原理及相互联系的认识，加深计算机工作中“时间-空间”概念的理解，从而清晰地建立计算机的整机概念。
2. 学习设计和调试计算机的基本步骤和方法，培养科学研究的独立工作能力，取得工程设计和调试的实践和经验。

二、设计任务

1. 根据给定的数据格式和指令系统，设计一台微程序控制的模型计算机。
2. 根据设计图，在 PROTEUS 环境下仿真调试成功。
3. 在调试成功的基础上，整理出设计图纸和相关文件，包括：
 - (1) 总框图（数据通路图）；
 - (2) 微程序控制器逻辑图；
 - (3) 微程序流程图；
 - (4) 微程序代码表；
 - (5) 设计说明书及工作小结。

三、设计内容与步骤

1. 数据格式

数据字规定采用定点整数补码表示法，字长 8 位，其中最高位为符号位，其格式如下：

7	6	5	4	3	2	1
符号位		尾数				

2. 指令格式

本实验设计使用 5 条机器指令，其格式与功能说明如下：

	7	6	5	4	3	2	1	0
IN	0	0	1	0	0	0	0	0
ADD	0	1	0	0	0	0	0	A
STA	0	1	1	0	0	0	0	A
OUT	1	0	0	0	0	0	0	A
JMP	1	0	1	0	0	0	0	A

IN 指令为单字长（字长为 8bits）指令，其功能是将数据开关的 8 位数据输入到 R0 寄存器。

ADD 指令为双字长指令，第一个字为操作码，第二个字为操作数地址，其功能是将 R0 寄存器的内容与内存中地址为 A 的数相加，结果存放在 R0 寄存器中。

STA 指令为双字长指令，第一个字为操作码，第二个字为操作数地址，其功能是将 R0 寄存器中的内容存储到以第二个字为地址的内存单元中。

OUT 指令为双字长指令，第一个字为操作码，第二个字为操作数地址，其功能是将内存中以第二个字为地址的内存单元中的数据读出到数据总线，显示之。

JMP 指令为双字长指令，第一个字为操作码，第二个字为操作数地址，其功能是程序无条件转移到第二个字指定的内存单元地址。

数据通路图一旦确定，指令流与数据流的通路也就确定了。图 1 中各功能器件上标

注的控制点及控制信号，就是微程序控制器设计的依据。

微指令格式建议采用水平型微指令格式，后继微地址采用断定方式。微指令格式如图 2 所示。

微指令长度为 22 位，据此可以确定控制存储器的字长也应为 22 位。微指令格式确定后，微程序的横向设计在于正确选择数据通路，纵向设计在于确定后继微指令地址。纵向设计的通常做法是先确定微程序分支处的微地址，因为微程序分支处需要进行判别测试，这些微地址确定后，就可以在“微地址表”中把相应的微地址单元填进去，以免后面的设计中重复使用，以致造成设计错误。

当拟定“取指”微指令时，该微指令的判别测试字段应指明 P(1)测试。“取指”微指令是所有微程序都使用的公用微指令，P(1)测试的结果导致微程序出现多路分支。在本模型机中，只拟设计 5 条机器指令，故用指令寄存器的前 3 位（IR7-IR5）作为测试条件，微程序可以实现 8 路转移，但我们只用到前 5 路。分支后的微地址分别定为 01001B-01101B。

微程序控制器的原理框图如图 3 所示。

3. 设计步骤

(1) 对指令系统中的各条指令进行分析，得出所需要的占领周期与操作序列，以便确定各器件的类型和数量；

(2) 设计总框图草图，进行各逻辑部件之间的互相连接，即初步确定数据通路，使得由指令系统所要求的数据通路都能实现，并满足技术指标的要求；

(3) 检查全部指令周期的操作序列，确定所需要的控制点和控制信号；

(4) 检查所设计的数据通路，尽可能降低成本，简化线路，优化性能。

以上过程可以反复进行，以便得到一个较好的方案。

图 1 给出了一个参考方案，数据通路的设计和器件的选择应同时进行，**接入总线的器件都要有三态输出**，以便与总线连接。图 1 中所示的方案采用单总线结构，使用的许多器件都是三态输出，这种方案便于总线的连接和扩展。

四、课程设计报告内容

- 1、数据通路逻辑电路图；
- 2、微程序控制器逻辑电路图；
- 3、微程序流程图；
- 4、微程序代码表；
- 5、设计说明及工作小结。

五、课程设计考核要求

- | | |
|----------|-----|
| 1、平时考勤 | 20% |
| 2、仿真结果 | 30% |
| 3、课程设计报告 | 50% |

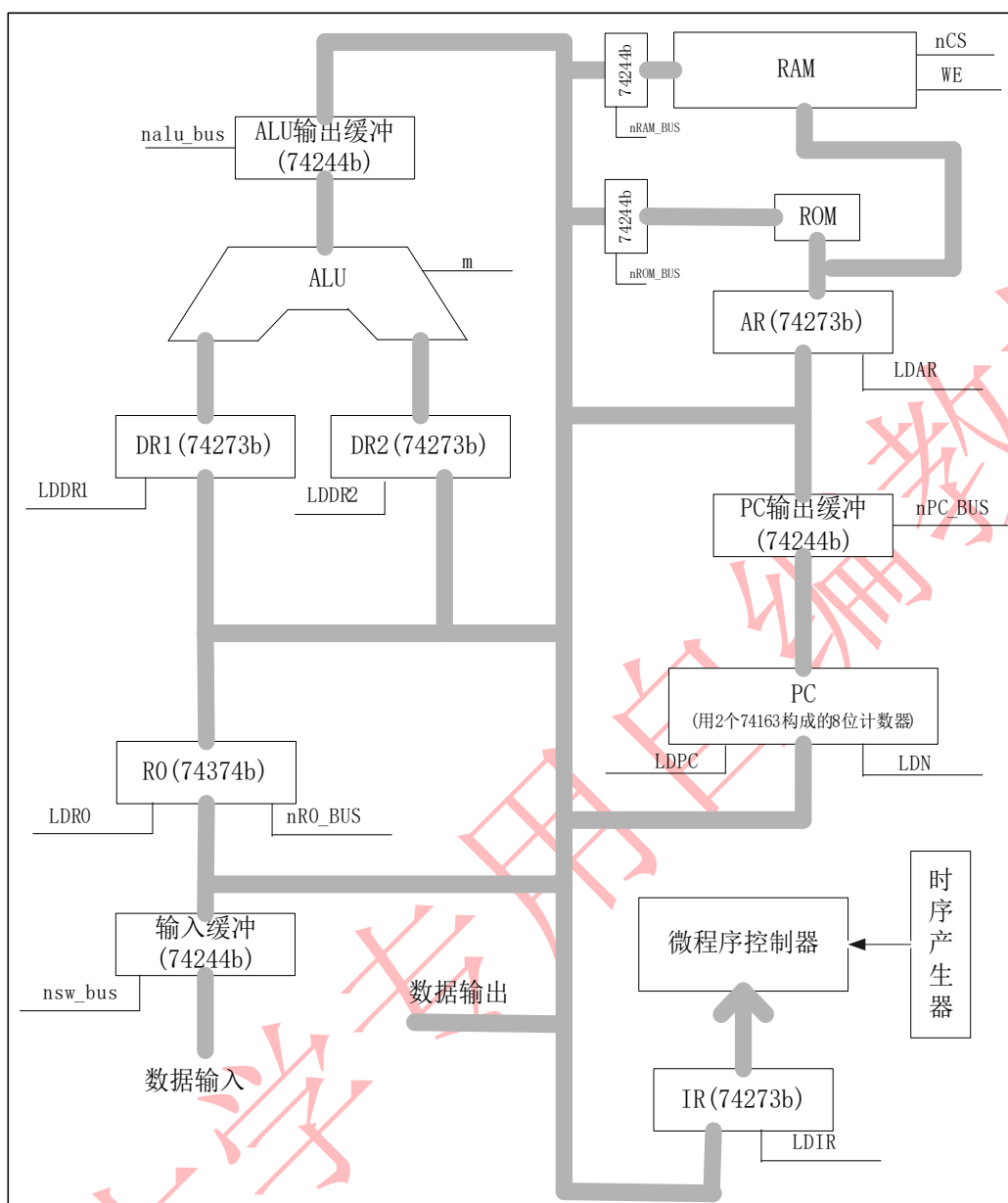


图 1 模型机的数据通路

●图 1 微控制信号说明：

- ①nROM_BUS: ROM 数据输出到总线控制信号，低电平有效。
- ②nRAM_BUS: RAM 数据输出到总线控制信号，低电平有效。
- ③m: 加、减法选择控制信号，为 1 做加法，为 0 做减法。
- ④nSW_BUS: 数据输入到总线控制信号，低电平有效。
- ⑤LDN: PC 置数控制信号，低电平有效。
- ⑥nCS: RAM 片选信号。（此信号的有效值根据所用 RAM 器件特性来定）
- ⑦WE: RAM 写信号，高电平时做写操作。
- ⑧LDRO: 数据打入 R0 锁存控制信号，脉冲上升沿有效。
- ⑨LDDR1: 数据打入 R1 锁存控制信号，脉冲上升沿有效。
- ⑩LDDR2: 数据打入 R2 锁存控制信号，脉冲上升沿有效。
- ⑪DIR: 数据打入 IR 锁存控制信号，脉冲上升沿有效。
- ⑫DPC: 数据打入 PC 锁存控制信号，脉冲上升沿有效。
- ⑬DAR: 数据打入 AR 控制信号，脉冲上升沿有效。
- ⑭ALU_BUS: 运算器 ALU 结果输出到总线控制信号，低电平有效。
- ⑮PC_BUS: 低电平有效。

⑩R0_BUS, 低电平有效。

nROM_BUS	nRAM_BUS	\overline{M}	nSW_BUS	LDN	nCS	\overline{WE}	LDR0	LDDR1	LDDR2	LDIR	LDPC	LDAR	nALU_BUS	nPC_BUS	nR0_BUS	P(1)	ad4	ad3	ad2	ad1	ad0
↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑	↑
21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

图2 微指令格式

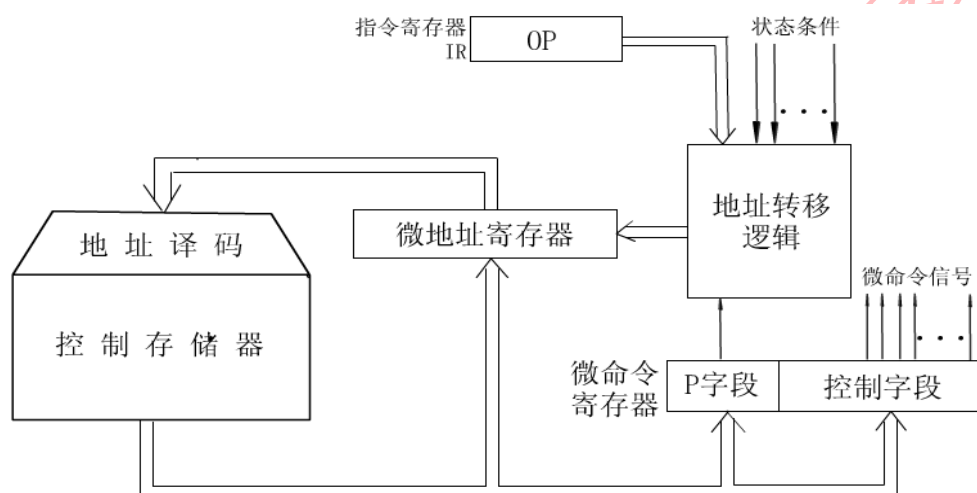


图3 微程序控制器原理图

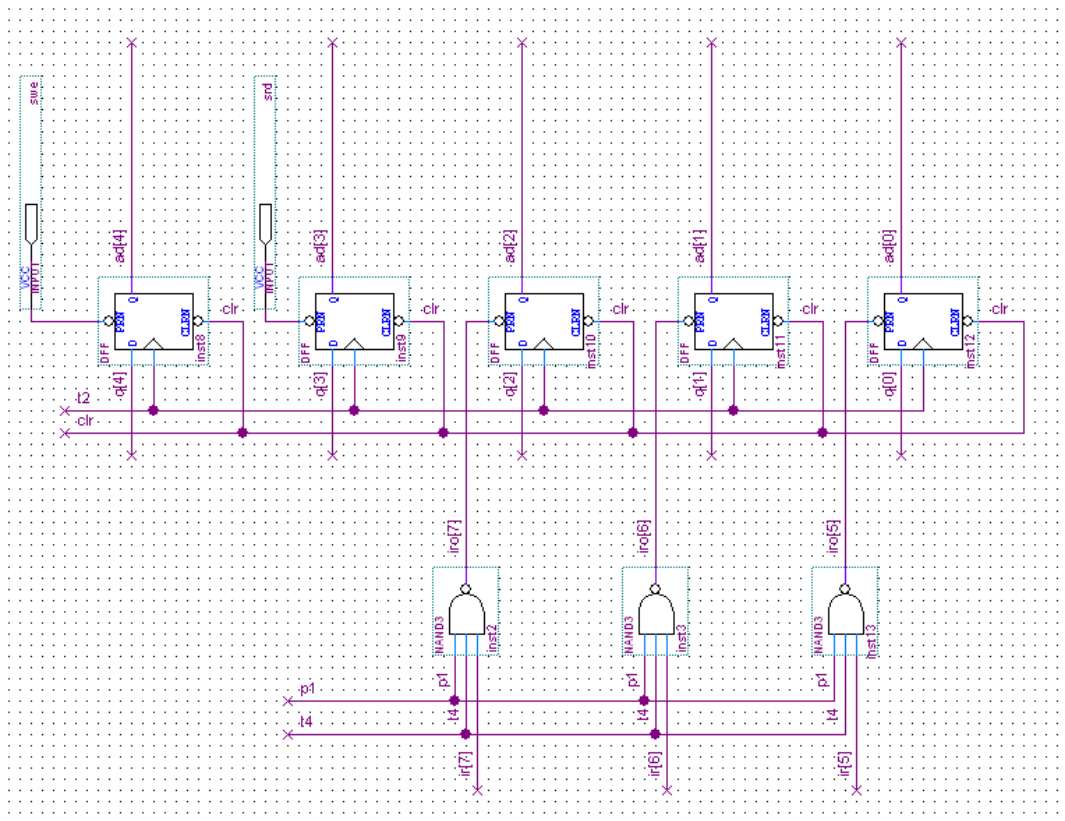


图4 地址转移逻辑电路图

●图4 信号说明:

- ①swe 信号: 总清信号
- ②srd 信号: 总清信号
- ③p1 信号: 译码判别位
- ④ad[4]...ad[0]信号: 微地址(控制存储器地址)
- ⑤ir[7]...ir[5]信号: 操作码
- ⑥clr 信号: 总清信号
- ⑦t2、t4 信号: 节拍脉冲

附 录

Proteus 的说明

Proteus 是英国 Labcenter Electronics 公司开发的电子设计 EDA 软件，可进行原理图和 PCB 版图设计，还可进行电路仿真，其中 MCU 与外围模数混合电路的协同仿真最具特色。为了实现上述功能，Proteus 软件附带了大量的元器件，许多常用元件都能找到。但即使如此，在实际的设计中仍会遇到很多库中没有的元器件，这时就要用户自己去制作和添加。

一、元器件及模型的分类：

电路是由元器件用导线连接而成的，元器件是组成电路的基本单元。在 Proteus 中，元器件分为原理图元件、PCB 封装和原理图仿真模型三大类。

1. 原理图元件：

Proteus 软件分为 ISIS 原理图编辑和 ARES 印刷电路板 (PCB) 绘制两部分。为了电路逻辑清晰和分析方便，电子产品一般都会先画原理图，从功能上确认后再实际制作产品。而 Proteus 的 ISIS 部分就是提供了电气原理图的编辑和分析界面，在这里要使用的是原理图元件。

Proteus 原理图元件有 30 多个库共 8000 多个元器件，包括常用的电阻、电容、电感、二极管、三极管、可控硅、光电显示器件、开关继电器、温度压力传感器、马达、电子管和集成电路等，集成电路 (IC) 可分为模拟集成电路 (如放大器、比较器、滤波器、模拟开关、稳压电源等) 和数字集成电路两大类，数字集成电路有 74 系列、4000 系列、ECL10000 系列和处理器及外围器件，74 系列又可细分为 S、LS、AS、ALS、F、HC、HCT 等系列。处理器主要是 51 系列、AVR 系列、PIC 系列、MSP430 系列、MC68HC11 和 ARM7 (LPC21XX 系列)，处理器外围器件包括 SRAM、DRAM、EPROM、EEPROM、ADC、DAC、显示驱动、实时时钟、电压监测、数字电位器、RS232/485 收发器、遥控编解码、接口扩展及其他 MCU 外围接口器件。原理图元件大多使用国际上标准的电路图形符号，任何国家的用户都能很方便地辨识和使用。

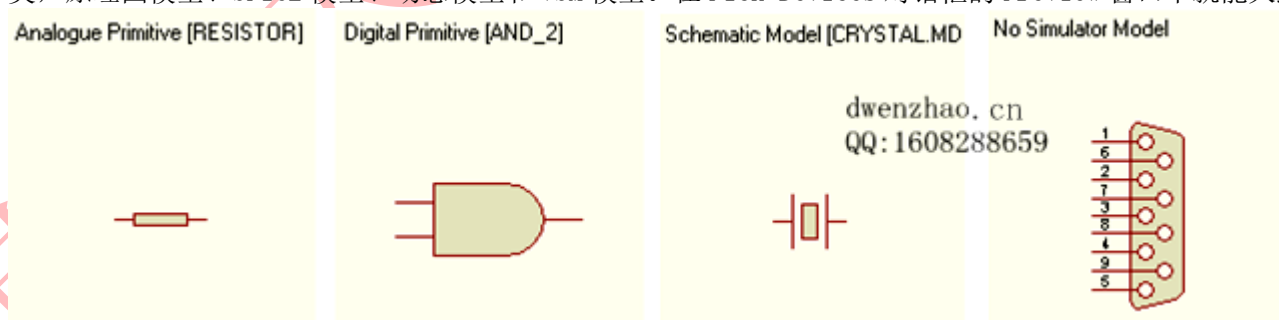
2. PCB 封装：

现在，印刷电路板 (PCB) 是最常用的实际电路组装方式。为了实现这种组装。每个原理图中的元器件都要有一个对应的封装 (Package)，其大小对应着实际元器件的外形尺寸，关键是包含了 PCB 组装用的焊盘大小和位置，制造时能使实际元件固定并使电路接触良好。

Proteus 附带的封装包括连接器、分立元件、集成电路及其他一些特殊元件等，包括了插孔式 (Through Hole) 和表面贴装 (Surface Mount) 两大类的常用封装，各种集成电路的封装形式最多也较复杂。

3. 仿真模型：

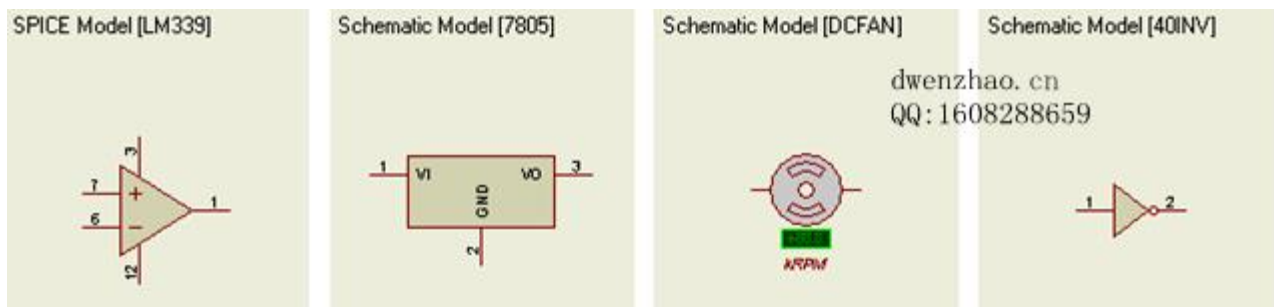
作为一种电路 EDA 软件，Proteus 很重要的一点是具有在原理图上进行仿真分析的能力，为了实现这种功能，原理图元件就不能只有一个外形和管脚，还要有相应的仿真模型。不具有仿真功能的元器件被称为绘图模型 (Graphical Model)，具有仿真功能的元器件称为电气模型 (Electrical Model)。Proteus 的电气模型分为 4 类，原理图模型、SPICE 模型、动态模型和 VSM 模型。在 Pick Devices 对话框的 Preview 窗口中就能大致分辨。



(Primitive 为仿真原型，是构建其他仿真模型的基本“砖块”，分模拟和数字两类)

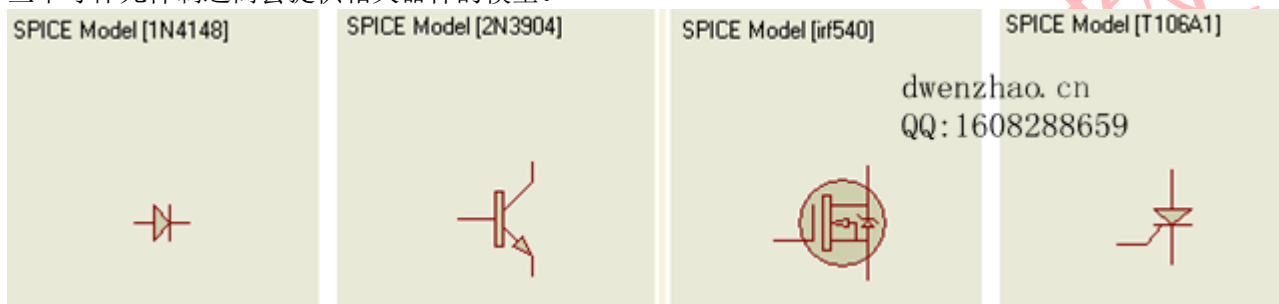
1) 原理图模型 (Schematic Models):

由仿真原型 (Simulator Primitives) 构建，与实际元器件有相同等效电路性能。它的并不是按实际器件的内电路搭建，而只是外特性与实际器件等效。主要包括 Modelling Primitives、Simulator Primitives 库中的模型或使用其中的模型构建的模型。



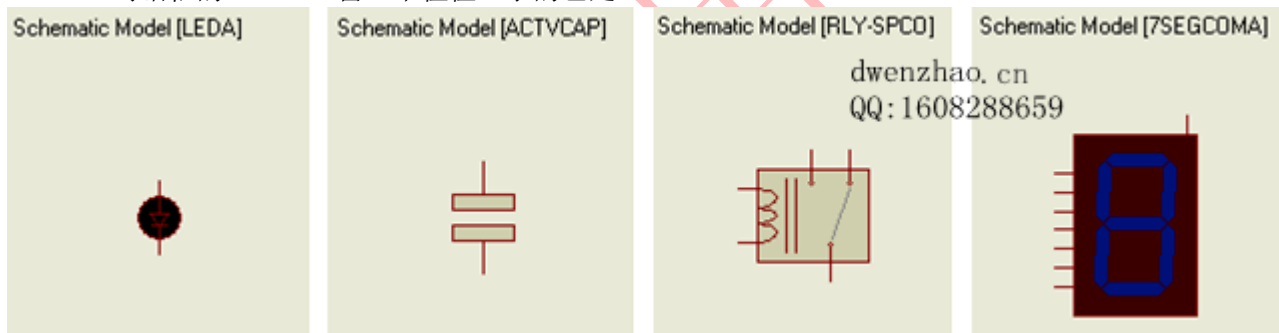
2) SPICE 模型 (SPICE Model):

使用符合 SPICE3F5 规范的 SPICE 文件或库设计的仿真元器件，主要为二极管、三极管等分立半导体元件。SPICE 是一种业界普遍使用的电路级模拟程序，它通过半导体器件的内部结构和参数建立起相关的分析模型和方法，一些半导体元件制造商会提供相关器件的模型。



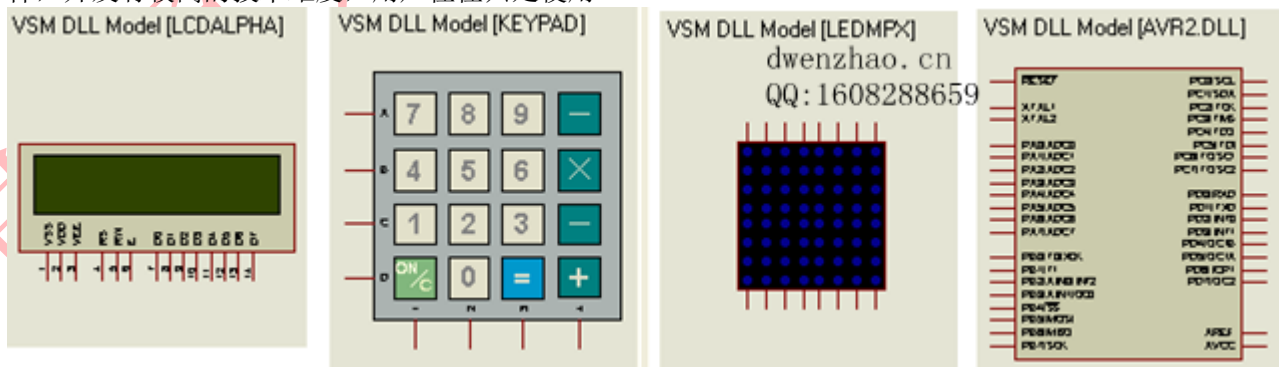
3) 动态模型 (Active Components):

具有动画效果的模型，如继电器、灯、LED 数码显示等。通过动画模仿器件的动作过程，很直观很形象。但 Pick Devices 对话框的 Preview 窗口中往往显示的也是 Schematic Models。



4) VSM 模型 (VSM Models):

是基于动态链接库 (DLL) 的仿真模型。DLL 是利用 Labcenter 提供的 VSM SDK (软件开发包) 用 C++ 编写的，用以描述器件的电气行为，这是 Proteus 独特的部分。VSM 模型主要包括处理器、液晶模块、传感器等复杂的元器件，开发有较高的技术难度，用户往往只是使用。




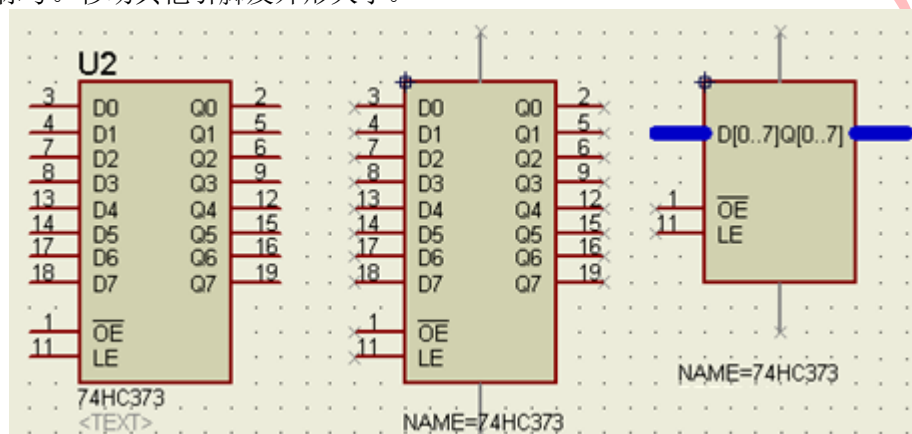
自制器件参考实例

A、原理图元件的修改:

个人制作元件比较繁琐，特别是制作具有仿真功能的元件更是如此，如果是在库中原有元件的基础上修改，会比较简单一些。示例，把 74HC373 改为总线方式。

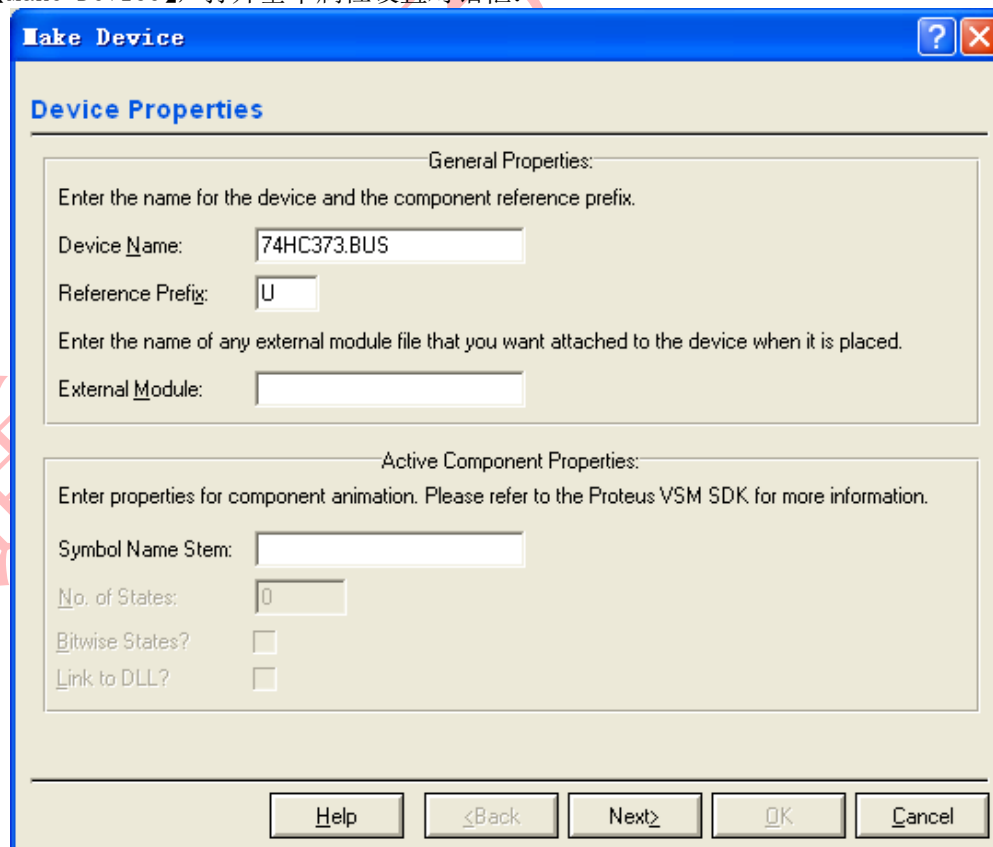
1. 元件外形修改:

把 74HC373 放入编辑界面中，点选，使用工具栏中 （或鼠标右键菜单中 Decompose），把元件分解而进入可编辑状态。把 Q0~Q7 及 D0~D7 引脚删除，添加总线引脚（见上节）并加标号。移动其他引脚及外形大小。

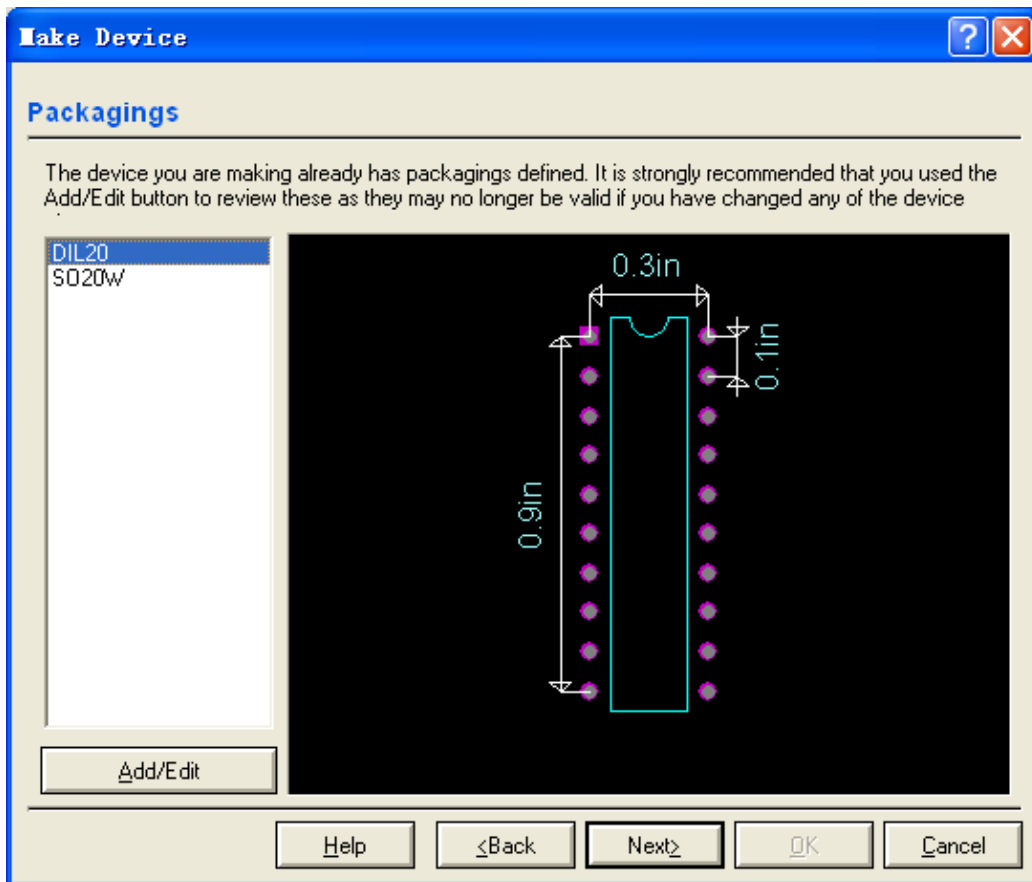


2. 元件入库:

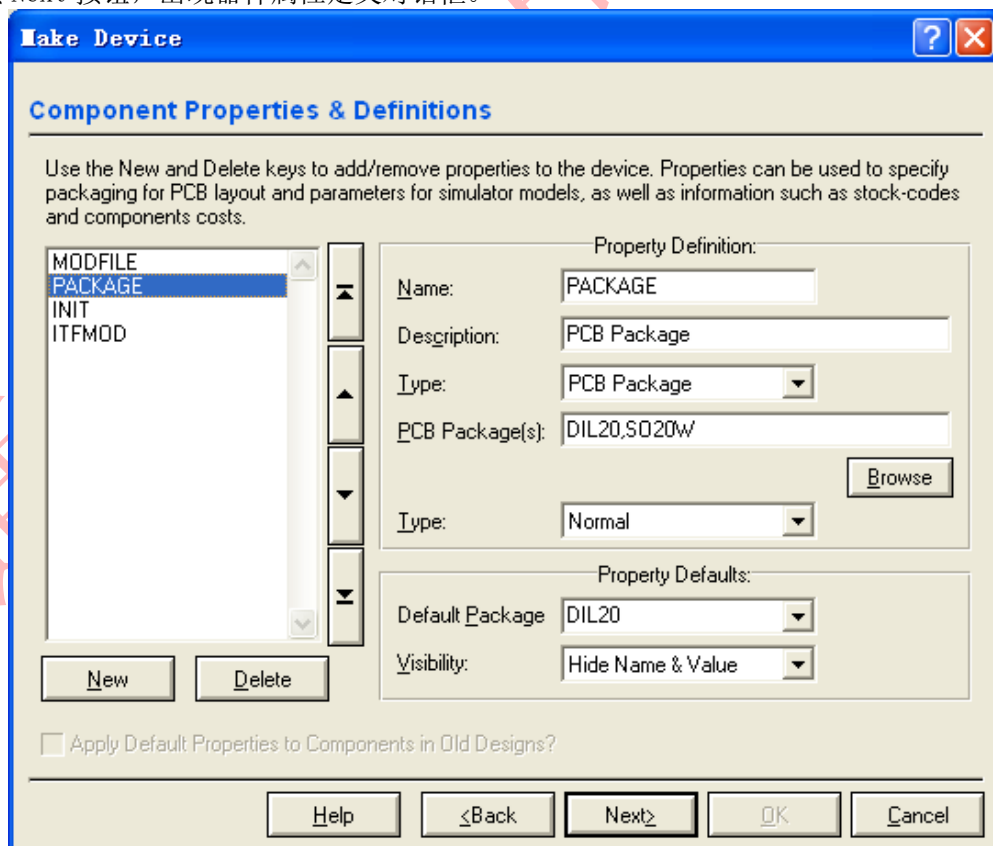
用鼠标拖选整个元件（包括下面的三行 Decompose 后产生的字符串），菜单【Library】→【Make Device】，打开基本属性设置对话框:



只需要把名称后面加.BUS 后缀即可，因为原有器件的各种属性都保留着。按 Next 按钮，显示的是封装对话框，不用修改。



按 Next 按钮，出现器件属性定义对话框。



可以看到，左侧列表包括了 MODFILE、PACKAGE、INIT 和 ITFMOD 等 4 个属性，分别有对应的设置页面，如果是新建元件就要花很大精力去设置，这里因为只是修改，都可以沿

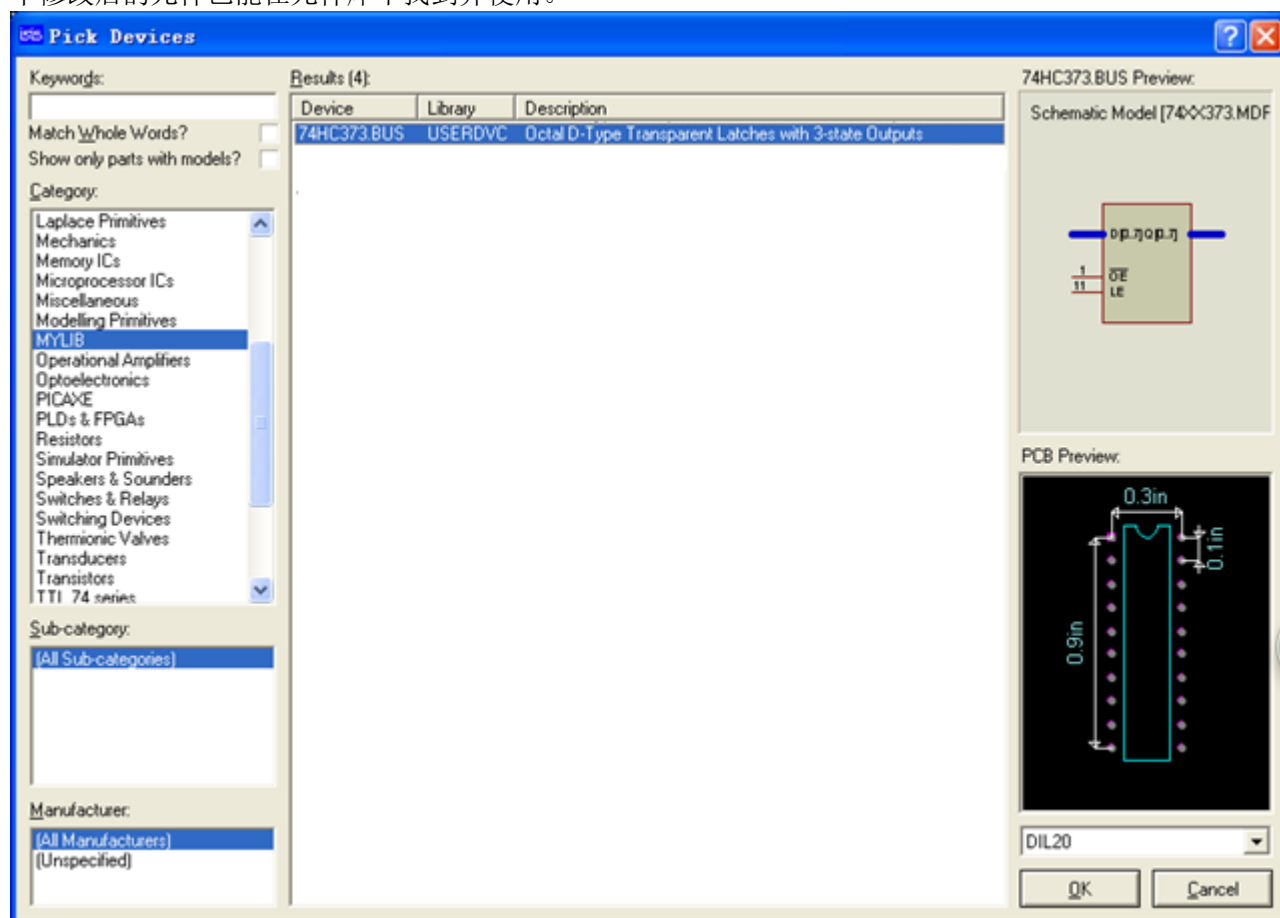
用原来的，感兴趣的可以去看看每个界面的设置情况。继续 Next，出现的是 Datasheet 界面。

The screenshot shows the 'Make Device' dialog box with the 'Device Data Sheet & Help File' tab selected. The dialog has a blue title bar and standard Windows window controls. The main area is light beige. At the top, it says 'Device Data Sheet & Help File'. Below that, a text block explains: 'You can link your device to a data sheet (Acrobat .PDF file) and/or a help file. These can then be accessed via special buttons on the 'Edit Component' dialogue form.' There are two main sections: 'Data Sheet:' and 'Help Topic:'. The 'Data Sheet:' section contains fields for 'Data Sheet Filename:' (with 'mm74hc373.pdf' entered), 'Download Server:', 'Download Path:', 'Download User Id:', 'Download Password:', 'CD Title:' (with 'Proteus CD' entered), and 'CD Path:' (with 'pdfs\digital' entered). The 'Help Topic:' section contains a 'Help File:' field with a file icon button and a 'Context Number:' field with '0' entered. At the bottom, there are five buttons: 'Help', '<Back', 'Next>', 'OK', and 'Cancel'.

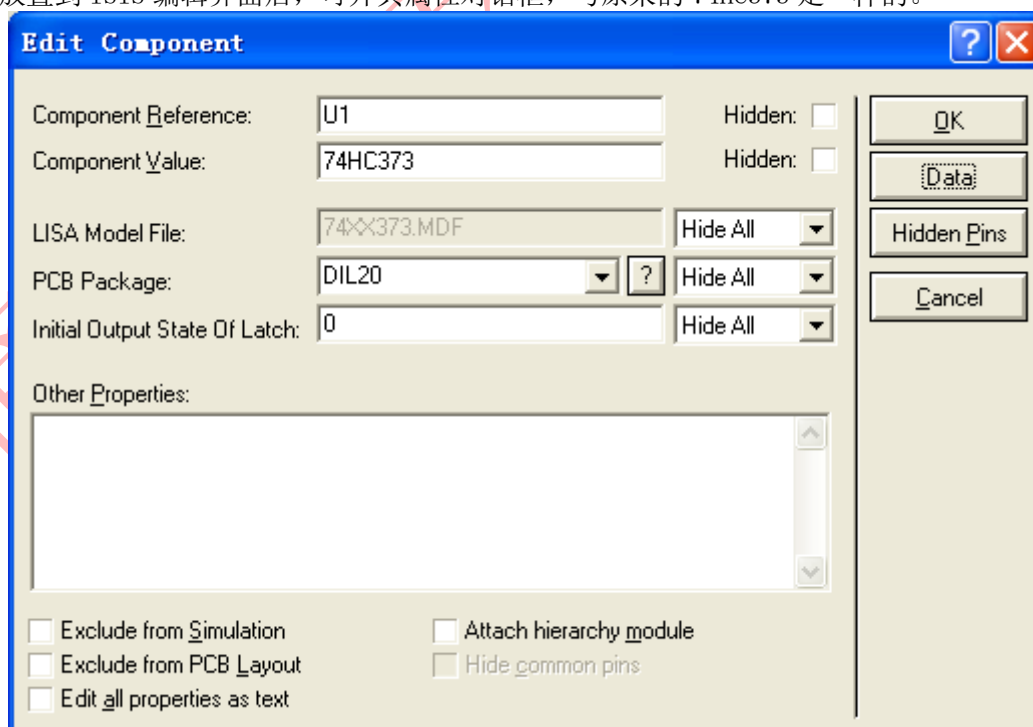
也可以保持原状，继续 Next，出现元件库选择界面，可以选 MYLIB。

The screenshot shows the 'Make Device' dialog box with the 'Indexing and Library Selection' tab selected. The dialog has a blue title bar and standard Windows window controls. The main area is light beige. At the top, it says 'Indexing and Library Selection'. There are two main sections. The left section contains fields for 'Device Category:' (with 'MYLIB' selected in a dropdown), 'Device Sub-category:' (with '(None)' selected in a dropdown), 'Device Manufacturer:' (with '(None)' selected in a dropdown), 'Stock/Order Code:' (an empty text field), 'Device Description:' (with 'Octal D-Type Transparent Latches with 3-state Ou' entered), an unchecked checkbox for 'Advanced Mode (Edit Fields Manually)', and a 'Device Notes:' text area. The right section is titled 'Save Device To Library:' and contains a list box with 'USERDVC' selected. At the bottom, there are five buttons: 'Help', '<Back', 'Next>', 'OK', and 'Cancel'.

按 OK 按钮，就完成了保持了仿真性能的元器件的修改，需要设置的内容很少。这个修改后的元件已能在元件库中找到并使用。





放置到 ISIS 编辑界面后，可开其属性对话框，与原来的 74HC373 是一样的。

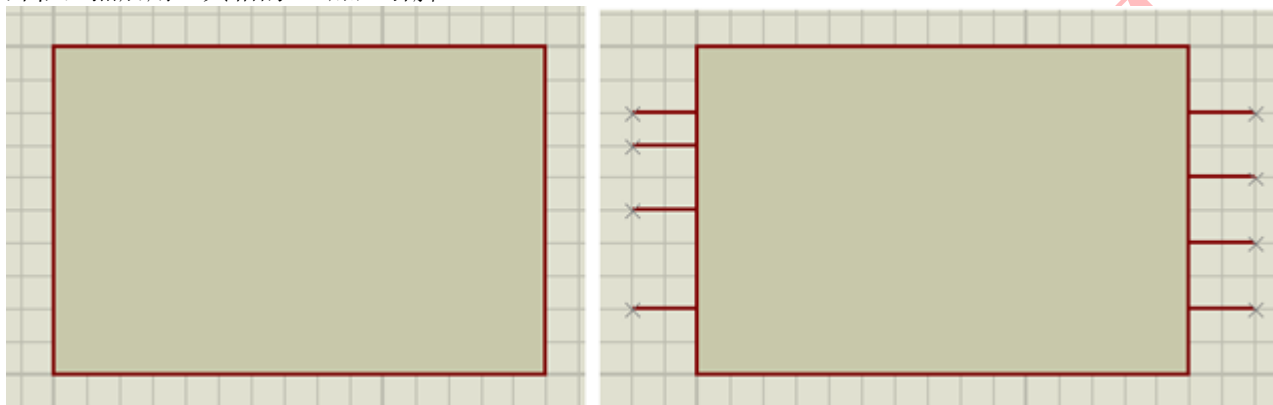



B、模块元器件的制作：

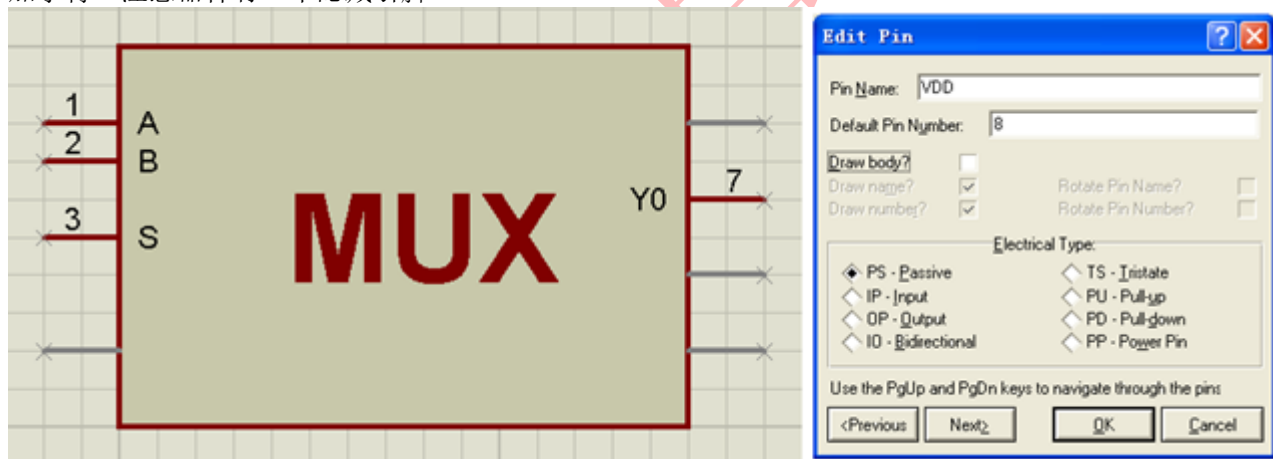
PROTEUS ISIS 原理图设计中，往往会遇到比较复杂的电路，为简化设计，可以引入模块元器件，形成一种层次结构。模块元器件可以像其他元器件一样放置和使用，如果内电路用可仿真的元器件组成还可以进行整体仿真，这种方式制作的模块元器件也是一种原理图模型。

1. 模块元器件的绘制：

像制作原理图元器件一样，先点击 2D 工具箱中的 ，在编辑窗口拖动，先画出模块的外框，然后用工具箱的  加入引脚。

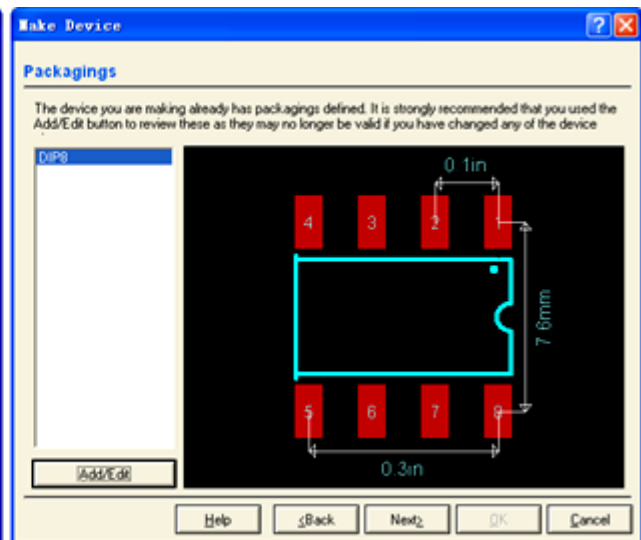
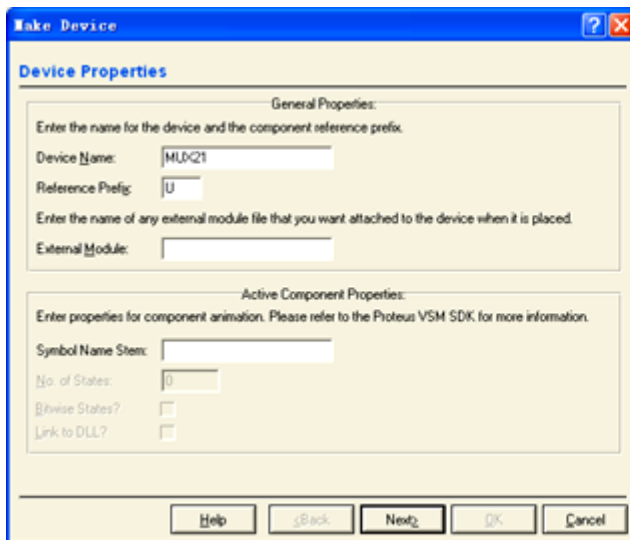


然后为每个引脚添加名称及序号并设置引脚属性，还可以用工具箱中的 ，为模块添加字符。注意器件有 4 个隐藏引脚。

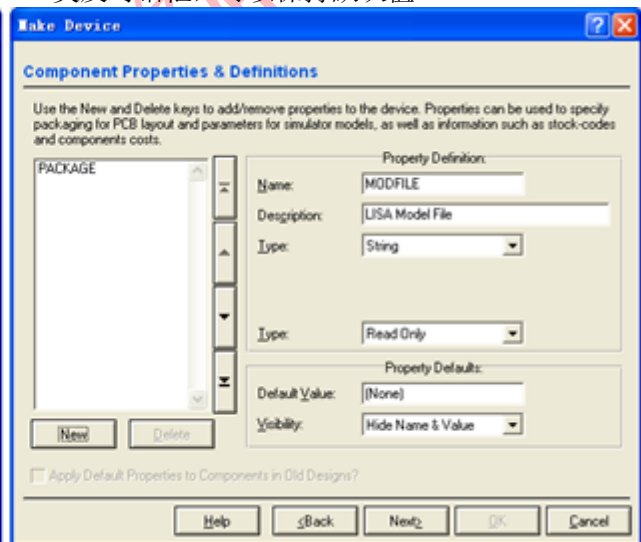
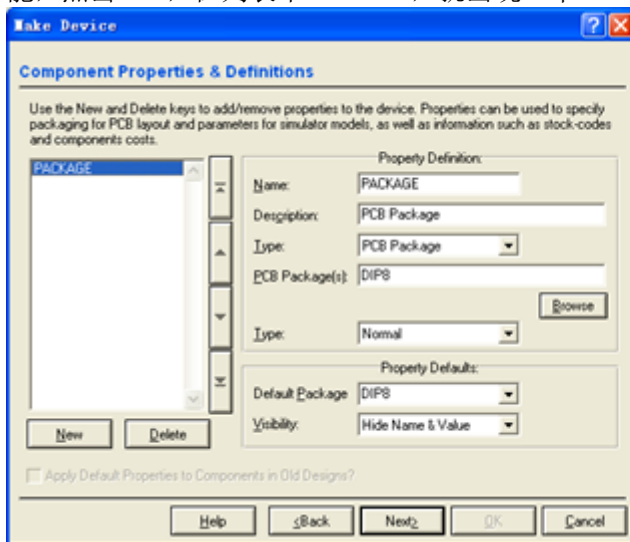


2. 模块入库：

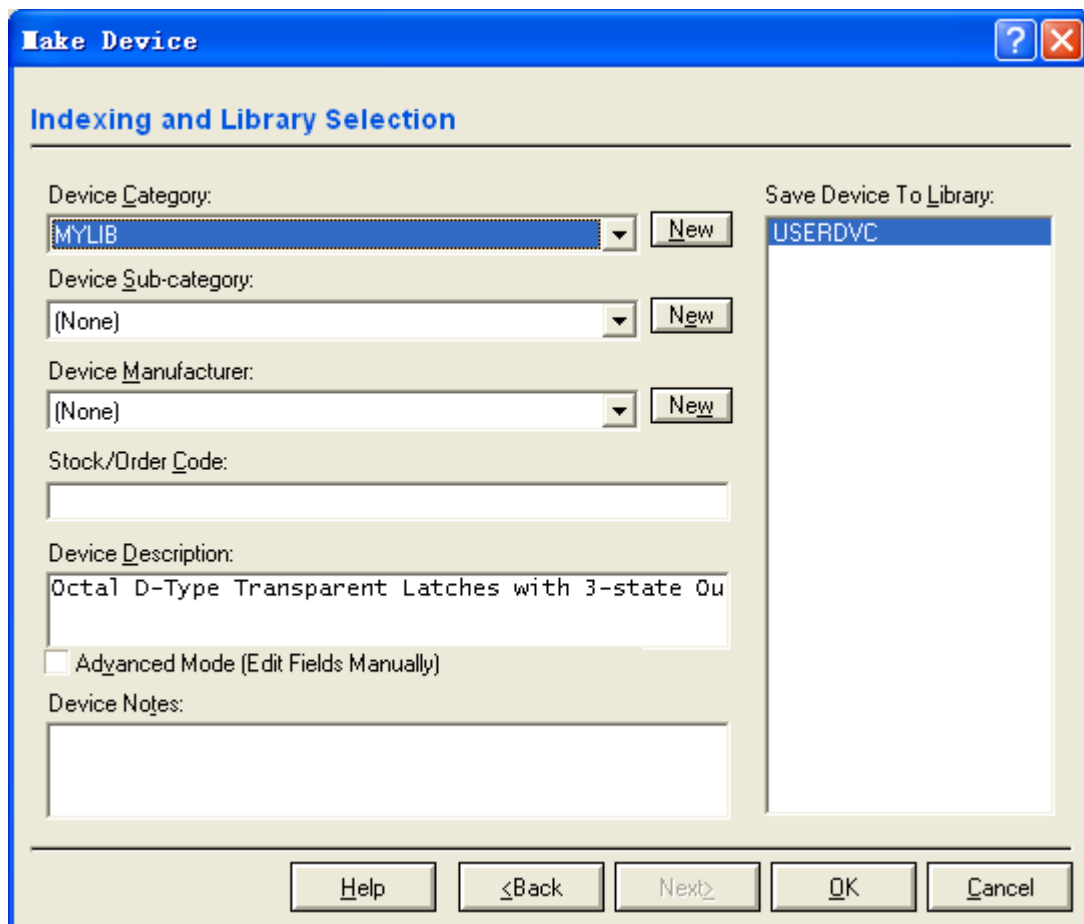
同制作原理图元器件一样，用鼠标拖选整个元件，菜单【Library】→【Make Device】（或鼠标右键菜单中的 Make Device），打开基本属性设置对话框，填入模块的名称及类型前缀。然后点击 Next 按钮，出现的是封装对话框，点击 Add/Edit 按钮，在出现的界面中再点 Add 按钮，出现 Pick Package 界面，在其中选择适合的封装，这里为器件选择了 8PIN 的 SMD 封装（也可以选择其他适合的 8PIN 的 Package）。这个封装与原理图中定义的序号对应。



点 Next 按钮，在定义元器件属性的对话框中，就有了 PACKAGE 类，因为要使用仿真功能，点击 NEW，在列表中 MODFILE，就出现一个 MODFILE 类及对话框，可以保持默认值。



再点 Next 按钮，略过出现的 Datasheet 说明文件对话框，点击 Next 按钮，出现的是库选择对话框，一般把自己制作的元件都放入单独的库中。

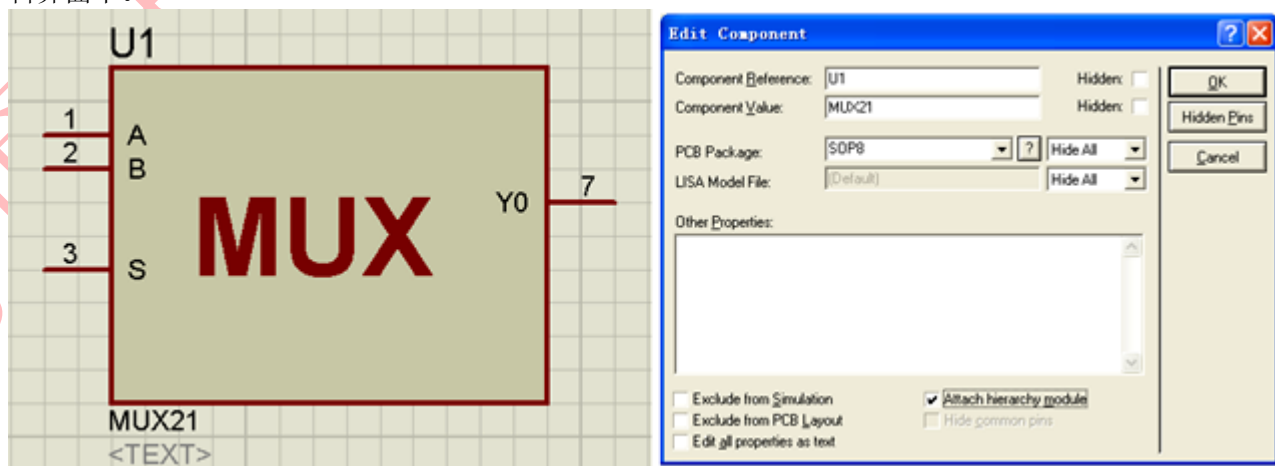


点击 Device Category 右侧的 New 按钮创建新库，或从列表选择一个自建的元件库，如 MYLIB，点击 OK 按钮完成。

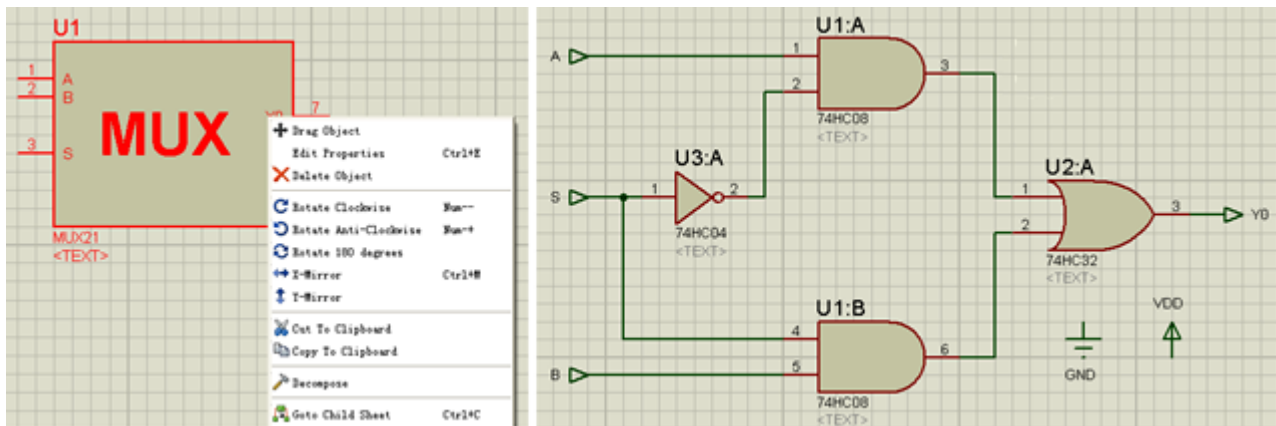
3. 建立层次结构：

以上过程与创建新的原理图元器件的步骤基本一样，但此模块元器件的内容是空的，只是一个外观符号，是一个绘图模型（Graphical Model），并没有任何仿真功能。

菜单【Library】→【Pick Device/Symbol】，打开元器件拾取对话框，在 MYLIB 库中找到上面创建的模块元器件，也可以直接在 Keywords 输入名称 MUX21 查找。点选模块元器件，按钮 OK，就可添加到设计文档的元器件列表中，然后在编辑区点击鼠标，放入编辑界面中。



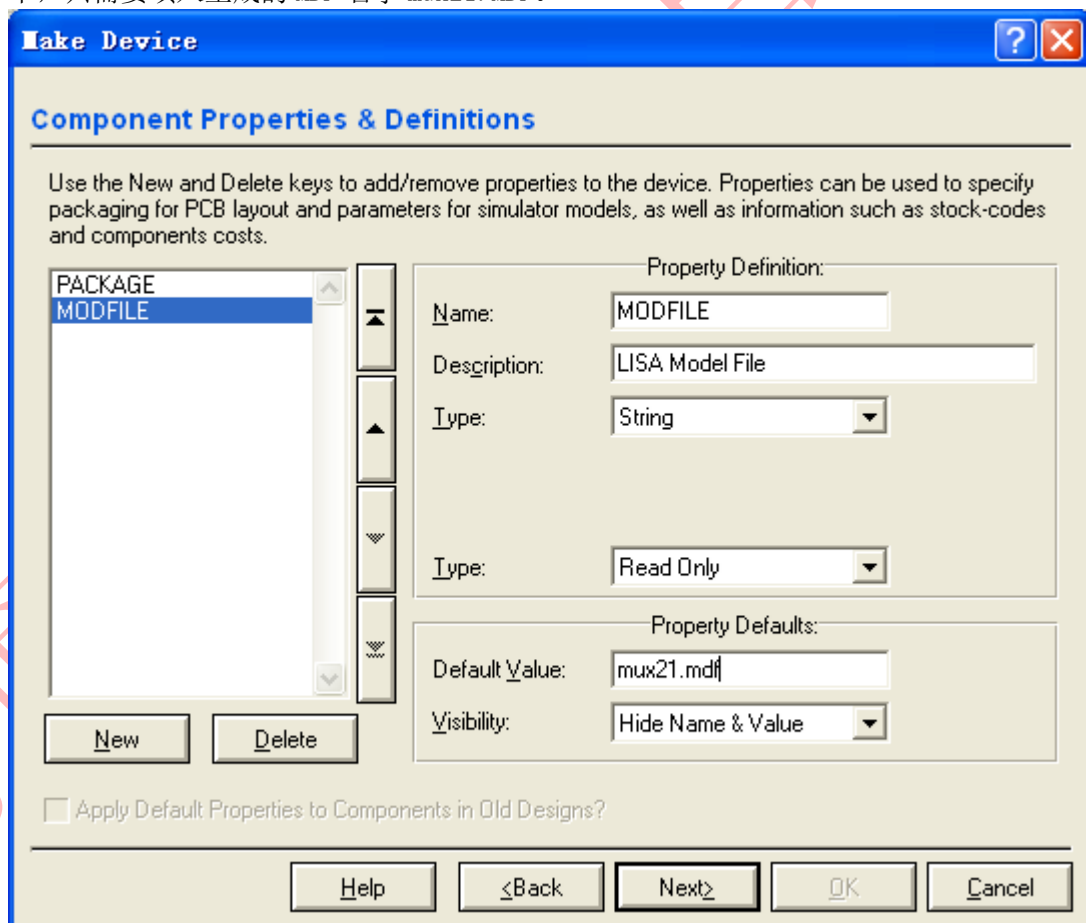
双击，打开器件属性对话框，点选下面的 Attach hierarchy module（附加层次模块）选项，点击 OK 确认。点选器件，鼠标右键菜单 Goto Child Sheet，转入子页面。



在子页面中如图绘制内电路原理图，注意信号输入/输出引脚要与模块元件的引脚名称一致，因模块元件有隐藏电源引脚，所以也要加入。完成后，菜单【Design】→【Previous Sheet】返回。可以对电路施加激励信号验证其性能。

4. 生成模板文件：

返回子电路，菜单【Tools】→【Model Compiler】，生成一个 mux21.MDF 文件，保存到 MODELS 文件夹中（或另外指定文件夹），这个文件夹中存放着用于仿真的很多模型文件。返回上一层，鼠标右键菜单 Make Device，两次点击 Next 按钮，到元器件属性对话框，在 MODFILE 界面中填入刚才生成的模型文件的名称和路径。因为存放在默认的 MODELS 中，只需要填入生成的 MDF 名字 mux21.MDF。



再两次按 Next 按钮，然后点击 OK，软件会出现一个提示框，询问是否替换已存在的 mux21 器件（这是用新加入的有模型的器件替换原有的符号），点击 OK。这样，一个完整的具有仿真性能的模块器件就制作完成了。如果熟练掌握就可以自己制作一些模块器件用于仿真。（有时使用的子电路中会有一些变量，在使用模块器件时需要在属性框中进行相应设置。）

全用自編教材

一、译码器 74138 真值表

* $G2 = G2A + G2B$
H = High Level, L = Low Level, X = Don't Care

二、缓冲器 74244b 真值表

L = Low Logic Level
H = High Logic Level
X = Either Low or High Logic Level
Z = High Impedance

三、寄存器 74273b 真值表

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial

四、寄存器 74374 真值表

OEN	Inputs CLK	D	Outputs Q
H	X	X	Z
L	X	X	X
L	J	L	L
L	J	H	H
L	L	X	Q ₀

五、计数器 74163 真值表

CLK	Inputs					D	C	B	A		Outputs				
	LDN	CLRN	ENP	ENT							QD	QC	QB	QA	RCO
1	X	L	X	X							L	L	L	L	L
2	L	H	X	X	d	c	b	a			d	c	b	a	*
3	H	H	L	H							QD	QC	QB	QA	
4	H	H	H	L							QD	QC	QB	QA	L
5	H	H	H	H							L	L	L	L	L
6	H	H	H	H							L	L	L	H	L
7	H	H	H	H							L	L	H	L	L
8	H	H	H	H							L	L	H	H	L
9	H	H	H	H							L	H	L	L	L
10	H	H	H	H							L	H	H	L	L
11	H	H	H	H							L	H	H	H	L
12	H	H	H	H							L	H	H	H	L
13	H	H	H	H							H	L	L	H	L
14	H	H	H	H							H	L	H	L	L
15	H	H	H	H							H	L	H	H	L
16	H	H	H	H							H	H	L	H	L
17	H	H	H	H							H	H	H	L	L
18	H	H	H	H							H	H	H	L	H

* RCO = QD & QC & QB & QA & ENT