地址总线

数据总线

地址寄存器

寄存器

输出单元

存储器

**第二章 计算机组成原理实验**

**2.1 总线与寄存器实验**

**一、实验目的**

1. 掌握总线以及数据通路的概念及传输特性。
2. 理解锁存器、通用寄存器及移位寄存器的组成和功能。

**二、实验内容**

构建一条 8 位单总线的寄存器数据通路，将若干寄存器连接起来。

通 过拨码开 关输入原始数据到某 个寄存 器； 或者 从一个 寄存 器向另 一个 寄存 器赋 值。 同时，利用移位寄存器实现数据的置数、左移、右移等功能 。

比 较 以下 各 组 器 件 之 间 的 异 同 ： 触 发器 74LS74 和 74LS175 ； 触 发 器 74LS175 和 寄 存 器 74LS273；寄存器 74LS273 和 74LS374；寄存器 74LS273 和移位寄存器 74LS194。

**三、实验器件**

1. D 触发器（74LS74、74LS175）、三态缓冲器（74LS244）。
2. 寄存器（74LS273、74LS374 ）和移位寄存器（74LS194）

**四、实验原理**

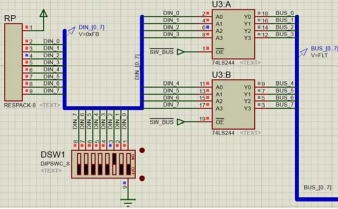
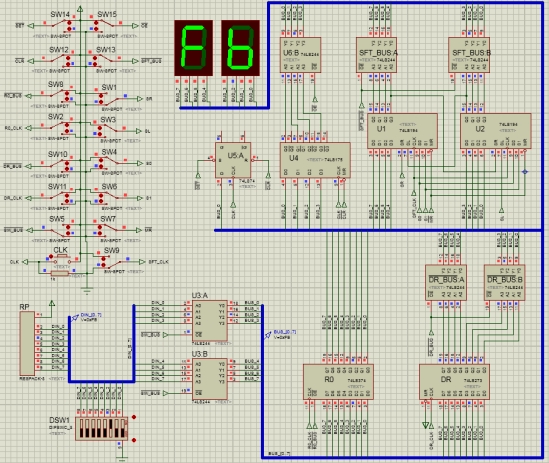
总线是多个系统部件之间进行数据传送的公共通路，是构成计算机系统的骨架。借助总 线连接，计算机在系统各部件之间实现传送地址、数据和控制信息的操作。因此，所谓总线 就是指能为多个功能部件服务的一组公用信息线。

总线示意图如图 2-1 所示：输入单元、输出单元、寄存器、存储器及其地址寄存器等不 同的设备挂在同一条总线上。这些设备都需要有三态输出控制，保证任何时刻总线上只有唯 一的数据存在。

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 输入单元 | | |

图 2-1 总线示意图

本 实 验 的寄存 器 数 据通 路 如 图 2-2 所 示， 本 通 路 包 含 了输入单 元 （ 拨码 开 关）、输 出 单 元 （数 码 显 示 管 ） 和 寄 存 器 堆 。 输 入 单元 如图 2-3 所 示 ，本 身 就 是 一个 8 位 的 输 入 总 线



DIN\_[0..7]（如表 2-1 所示）。DIN 总线上挂靠的拨码开关和上拉电阻用来设置输入数据。低 电平有效的开关 SW\_BUS\ 控制三态门 74LS244，实现总线 DIN\_[0..7]与数据总线 BUS\_[0..7] 的连通。三态门 74LS244 逻辑功能如表 2-2 所示。输出单元即是一对数码管，用来显示数据 总线 BUS\_[0..7]上的当前数据。

图 2-2 寄存器数据通路图

表 2-1 总线逻辑

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DIN\_7 | DIN\_6 | DIN\_5 | DIN\_4 | DIN\_3 | DIN\_2 | DIN\_1 | DIN\_0 | 总线数据  (16 进制) |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0xFB H |

表 2-2 74LS244 逻辑功能表

图 2-3 输入单元

|  |  |  |
| --- | --- | --- |
| 输 入 | | 输 出 |
| A | OE | Y |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| × | 1 | Z |

寄 存 器 堆 由 D 触 发 器 （ 74LS74 、74LS175 ）构 成 的 锁 存器 、 通 用 寄 存 器 R0 （74LS374 ） 和 DR（74LS273）、移位寄存器（74LS194）构成。其中除了 74LS374 器件自带三态门结构外， 其余锁存器和寄存器的输出都经过三态门（ 74LS244）和数据总线 BUS\_[0..7]相连，以确保 按照信息传输的要求恰当有序的控制锁存器和寄存器的数据输出到总线，任何时刻总线数据 都不会发生冲突。

图 2-2 中已将本实验需要连接的控制开关标明在左上方：其中除 CLK、R0\_CLK、DR\_CLK 和 SFT\_CLK 为上升沿控制信号，其余开关均为电平控制信号。

上 升 沿 有 效 的 开 关 CLK(74LS74 和 74LS175 共 用 ) 、 R0\_CLK(74LS374 使 用 ) 、 DR\_CLK(74LS273 使用)和 SFT\_CLK（74LS194 使用），负责把总线数据打入各自的器件；

低 电 平 有 效的 开 关 OE （ 74LS74 、 74LS175 的 三 态 门 控制 ）、R0\_BUS （74LS374 输 出 控 制 ）、DR\_BUS （74LS273 的 三 态门控 制 ） 和 SFT\_BUS （74LS194 的 三 态 门 控 制 ），负责 各 个 器 件输出所保存的数据到数据总线。

低 电 平 有 效 的 开 关 SET 是 74LS74 的 置 1 开 关 ， 低 电 平 有 效 的 开 关 CLR （ 74LS74 和 74LS175 共 用 ）、 MR （74LS194 使 用 ） 是 各 个 器 件 相 应 的 清 零 开 关 。 注 意 的 是 ： 寄 存 器 74LS374 是 没有 清零功 能的 ，74LS273 有 置 0 端 MR ， 可以设 置 清零 功能。 但 是，在 本实验 中 MR 接高电平，实际上 74LS273 取消了清零功能。

高 电平有效 的开 关 SL、SR 、S0、S1 是 移位寄存器 74LS194 的 专属开关， 负责其置数、 移位等功能的设置，具体详细见下文移位寄存器 74LS194 的功能列表。

**维持-阻塞型 D 触发器：**

D 触发器逻辑功能如表 2-3： Sd

和 R d

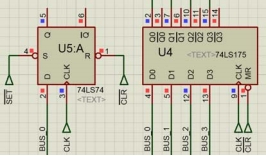
端分别为异步置 1 端和置 0 端，CLK 为时钟脉冲端。

CLK 脉冲上升沿触发。触发器的状态只取决于时钟到来前 D 端的状态，其输出状态的更新发 生在 CLK 脉冲的上升沿。 D 触发器可用作数字信号的寄存，移位寄存，分频和波形发生等。 本 实 验 使用 的 是 单位 D 触 发 器 74LS74 和 四位 D 触 发 器 74LS175 ， 如 图 2-3 所 示。 **74LS175 相当于四个并行的 74LS74 组合，但是省略了置 1 端，保留了置 0 端用以清零。**

表 2-3 D 触发器逻辑功能表

图 2-4 D 触发器

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 输 入 | | | | 输 出 | |
| S  d | R  d | CLK | D | Qn＋1 | Q n＋1 |
| 0 | 1 | × | × | 1 | 0 |
| 1 | 0 | × | × | 0 | 1 |
| 0 | 0 | × | × | φ\* | φ\* |
| 1 | 1 | ↑ | 1 | 1 | 0 |
| 1 | 1 | ↑ | 0 | 0 | 1 |
| 1 | 1 | ↓ | × | Qn | Q n |



注 ： \* 这 种 情 况 下 是 不 稳 定 的 ， 即 当 Sd

和 R d

端 回

到高电平时，输出状态将不能保持。

**通用寄存器：**

本实验中使用了 74LS273 和 74LS374 两款芯片作为通用寄存器，如图 2-5 所示。两者实 现的功能基本相同：D0-D7 为并行输入端，Q0-Q7 为并行输出端，CLK 端为时钟脉冲（上升沿触

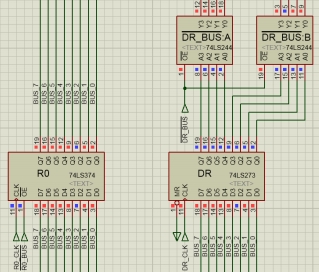
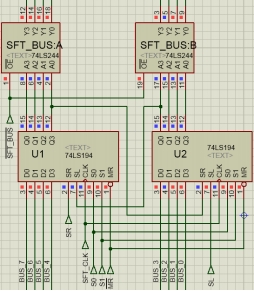
发）；输出端 Q

X

的状态只取 决于 CLK 端时钟脉冲到来时刻输入端 D

X

的状态，输出状态的更新



Q

Q

Q

Q

Q

发 生 在 CLK 端 脉 冲的 上 升沿 。 **74LS374 自 带 三 态门 输 出控 制结 构， 比 74LS273 多 了 一 个 OE 端 输 出 使 能 ， 相 当 于 “ 74LS273 + 74LS244” 组合。74LS273 本 身没有三 态 门 输 出 控 制 ，输 出 端必须 经过 74LS244 接 到 数据总线 BUS 。 但是 ， 74LS273 比 74LS374 多 了 一个 MR 端 输 出 清零。**两者的逻辑功能如表 2-4 所示：

表 2-4 74LS273 & 374 逻辑 功能表 图 2-5 通用寄存器

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MR | OE | CLK | D | n+1 |
| 0 | 0 | × | × | 0 |
| 1 | 0 | ↑ | 0 | 0 |
| 1 | 0 | ↑ | 1 | 1 |
| 1 | 0 | 0 | × | Qn |
| 1 | 0 | 1 | × | Qn |
| × | 1 | × | × | Z |

**移位寄存器：**

在数字系统中能寄存二进制信息，并进行移位的逻辑部件称为移位寄存器。

本实验采用四位双向通用移位寄存器 74LSl94，如图 2-6 所示：D

0

、D

1

、D

2

、D3

为并行输

入端； Q0 、Q1 、Q2 、Q3 为并行输 出端；SR 为右 移串 行输入 端；SL 为左 移串 行输入 端；S1 、S0 为 操作 模式控制端；MR 为直接无条件清零端 （注： 74LSl94 寄存或移位数据时，必须 MR=1）； CLK 为时钟 输入端。寄存 器有 四种 不同操作模 式：①并 行寄存；②右 移( 方 向由 Q0 ->Q3)； ③

左移(方向由 Q3

->Q0)；④保持。74LS194 的逻辑功能表如表 3\_5 所示。

表 2-5 74LS194 逻辑功能表

图 2-6 移位寄存器

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CLK | MR\ | S  1 | S  0 | 功能 | Q Q Q Q  3 |
| × | O | × | × | 清除 | Q Q Q Q =0 |
| ↑ | 1 | l | 1 | 送数 | Q0 Q1 Q2 Q3 = D0 D1 D2 D3 |
| ↑ | 1 | O | 1 | 右移 | Q0 Q1 Q2 Q3 = SR Q0 Q1 Q2 |
| ↑ | l | l | 0 | 左移 | Q Q1 Q2 Q =  Q Q2 Q3 SL |
| ↑ | l | 0 | 0 | 保持 | D D D D =  Q0 Q1 Q2 Q3 |

Q

**五、实验步骤**

**实验 1：拨码开关输入数据至总线**

● SW\_BUS = R0\_BUS = DR\_BUS = SFT\_BUS = 1 ； 手 动 操 作总线 DIN 上 的 拨 码 开 关 ， 在总线 DIN 上置位数据 0x55，缓冲器 244 阻断。比较总线 DIN 与 BUS 状态的异同。

● SW\_BUS=0， 比较总线 DIN 与 BUS 状态的异同，记录 BUS 总线的数据：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| BUS\_7 | BUS\_6 | BUS\_5 | BUS\_4 | BUS\_3 | BUS\_2 | BUS\_1 | BUS\_0 | BUS 总线 |
|  |  |  |  |  |  |  |  |  |

**实验 2：D 触发器数据锁存实验**

● SW\_BUS = 0 ， R0\_BUS = DR\_BUS = SFT\_BUS =1 ； 通 过 拨 码 开 关 改 变 74LS74 的 D

端 （即 BUS 总线的 BUS\_0 ） 的状态，按 照下表置位 74LS74 的 Sd 端、 R d

\_

端上升沿 、下降沿跳变时刻 Q 端、Q端的状态，填观测结果于表中。

端 ，观 察并记录 CLK

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sd R d | CLK | D | Qn | Qn+1 | \_  n+1 |
| 0 1 | X | X | 0 |  |  |
| 1 |  |  |
| 1 0 | X | X | 0 |  |  |
| 1 |  |  |
| 1 1 | ↑ | 0 | 0 |  |  |
| 1 |  |  |
| 1 1 | ↑ | 1 | 0 |  |  |
| 1 |  |  |
| 1 1 | 0（1） | X | 0 |  |  |
| 1 |  |  |

● 74LS175 的 三 态 门 244 阻 断 （OE=1 ），拨码开关置位 BUS 总 线 数 据 ，使 74LS175 的 D

\_

端分别接高，低电平，观察并记录当 CLK 上升沿 、下降沿跳变时 Q 端、Q端的状态。

\_ \_

● 观 察 74LS175 的 Q 端、Q端和 74LS74 的 Q 端、Q端的异同，观察当 74LS175 的 MR 端置

\_

0 后（CLR = 0），输出 Q 端、Q端的变化。

**实验 3：通用寄存器实验**

● SW\_BUS = 0 ，R0\_BUS = DR\_BUS = SFT\_BUS = 1 ；操作拨 码开 关输入数 据 0xAA 到

总线，观测此时 74LS374 和 74LS273 输出端 Q

X

的各自状态。

● 74LS374 的 CLK 端 R0\_CLK 上 升 沿 跳 变 把 总 线 上 的 0xAA 数 据 存 入 R0 寄 存 器 （74LS374）。

● 拨码开关的三态门 244 阻断（SW\_BUS = 1），观察此时总线 BUS 上的状态。

● 74LS374 的输出选通（R0\_BUS = 0），观测总线 BUS 的状态。

● 74LS273 的 CLK 端 DR\_CLK 上 升 沿 跳 变 把 总 线 上 的 0xAA 数 据 存 入 DR 寄 存 器

（74LS273）。观察 74LS374 和 74LS273 输出端 Q

X

的各自状态。

**实验 4：移位寄存器实验**

● SW\_BUS = 0，R0\_BUS = DR\_BUS = SFT\_BUS = 1 ；通过拨码开关送入总线 BUS 任意

八 位 二 进 制 数，赋 值 74LS194 的输 入端 D

0

D1

D2

D

3

。 按 照 下表 置位 74LS194 的 MR 端 、S

1

端 、 S

0

端、SL 端、SR 端 ，观察并记录 CLK 端上升沿 、下降沿跳变时 刻输出端 Q0Q1 Q2Q3 的状态，填观

测结果于表中，并填写功能总结。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 清除 | 模式 | | 时钟 | 串行 | | 输入 | 输出 | 功能总结 |
| MR | S1 | S0 | CLK | SL | SR | D0D1D2D  3 | Q0Q 1Q2Q3 |  |
| 0 | × | × | × | × | × | ×××× |  |  |
| 1 | 1 | 1 | ↑ | × | × | abc d |  |  |
| l | 0 | 1 | ↑ | × | 0 | ×××× |  |  |
| 1 | 0 | 1 | ↑ | × | l | ×××× |  |  |
| l | 0 | 1 | ↑ | × | 0 | ×××× |  |  |
| 1 | 0 | 1 | ↑ | × | 0 | ×××× |  |  |
| 1 | 1 | 0 | ↑ | 1 | × | ×××× |  |  |
| 1 | l | 0 | ↑ | l | × | ×××× |  |  |
| 1 | 1 | 0 | ↑ | 1 | × | ×××× |  |  |
| l | 1 | 0 | ↑ | l | × | ×××× |  |  |
| 1 | 0 | 0 | ↑ | × | × | ×××× |  |  |

**六、思考题**

\_

1. 把 74LS175 的Q端 接 三态门 244， 拨码开关把 数 据 0xAA 打入 74LS175 。假 设 OE=0， SW\_BUS=0，会出现什么情况？为什么？
2. 拨 码 开 关 先 打 入 数 据 0xAA 到 R0 寄 存器 （74LS374 ），R0\_BUS\=1 ； 拨码 开关 再输 入 新 的 数 据 0x55 到 总 线 BUS ， 会 冲 掉 R0 寄 存 器 里 保 存 的 0xAA 么 ？ 假 设 此 时 令 R0\_BUS=0，会出现什么情况？为什么？
3. 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 ），假 设 此时 令 R0\_BUS = DR\_BUS = 0 ， 会出现 什么 情况？ 可 以同时 打开多 个寄 存器输出到总线的三态门控制么？

4 、 74LS194 的 S

L

端 、 S

R

端 数 据 究 竟 是提 供 D0

D1

D

2D

3

端 移 入 数 据 还 是 保 存 D0

D1

D

2D

3

端 移 出

数据？假设要保存 74LS194 移位时 D

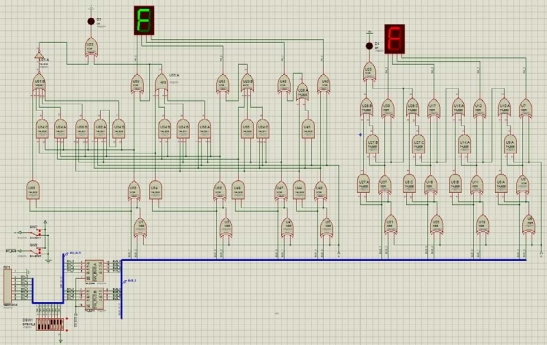
0

D

1D2

D3

端移出的数据，该怎么修改寄存器电路？



**2.2 进位加法器实验**

**一、实验目的**

1. 了解半加和全加运算器的电路结构。
2. 掌握串行进位加法器和并行进位加法器的原理及设计方法。

**二、实验内容**

设 计拥有 共同 输入 端的 4 位 带符号位的 串 行加法 器和 并行 加法器， 并比较两者 的 运算 输出结果。

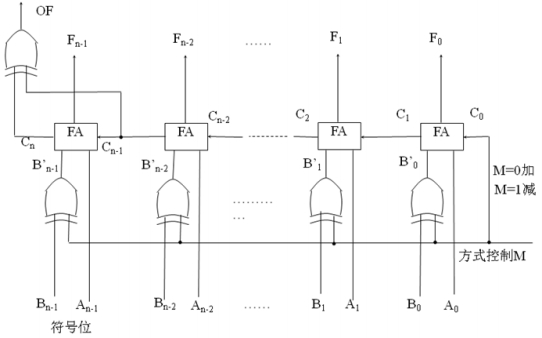
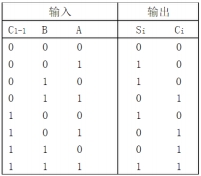
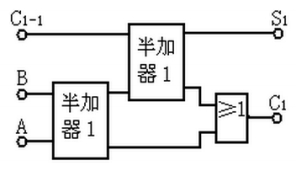
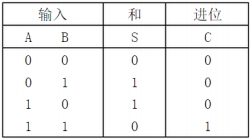
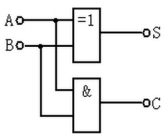
**三、实验器件**

1、2/3/4 与门(74LS08/11/21)、非门(74LS04)、或门(74LS32)、异或门(XOR)等逻辑门。 2、三态门（74LS244）、LED 指示灯及数码显示管。

**四、实验原理**

实 验 中 的加法 运 算 器电 路 如 图 2-7 所 示。 本 电 路 包 含 了输入单 元 （ 拨码 开 关）、输 出 单 元（数码显示管）和两个四位全加器。输入单元如图 2-3 所示，跟上述实验相同：拨码开关 用 来 设 置 输 入 数 据 ， 低 电 平 有 效 的 开 关 SW\_BUS 控 制 输 入 总 线 DIN\_[0..7] 与 数 据 总 线 BUS\_[0..7] 是 否 连 通 。 数 据 总 线 BUS 同 时 为 两 个 运 算 器 提 供 相 同 的 输 入 端 ： BUS\_[0..3] 和 BUS\_[4..7] 。输出 单元是一对数码显示管， 绿色显示管 是并 行进位 加法器的输 出端，红色显 示管是串行进位加法器的输出端

图 2-7 进位加法器电路图



加法 器是执行二进 制加减法 运算 的逻辑部 件（减法 可以通过 补码相加 实现 ），分 为半加 器和 全加器（FA），不 考虑低位的 进位，只考虑两个 二进制数相加 ，得到和以 及向高位进 位 的加法器为半加器，如图 2-8 所示；而全加器(FA)是在半加器的基础上又考虑了低位过来的 进位信号，如图 2-9 所示。

图 2-8 一位半加器逻辑电路图

半加器逻辑公式： Si

= Ai

⊕B

i

且

C

i+1

= A

i

·Bi

图 2-9 一位全加器（FA）逻辑电路图

全加器逻辑公式： Si

= Ai

⊕B

i

⊕C

i

且

C

i+1=Ai

·B

i

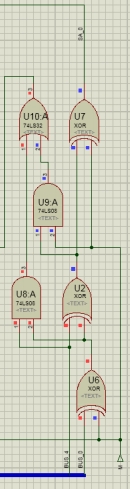
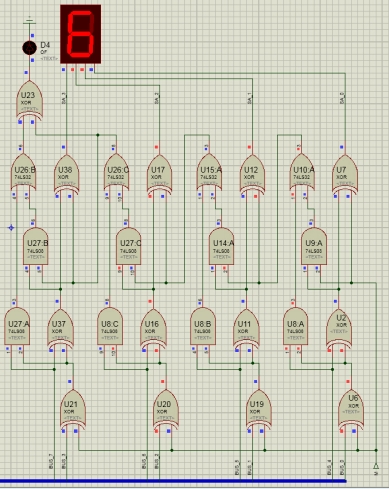
+ (A

i

⊕Bi

)·Ci

图 2-10 串行加法器（行波进位加法器）逻辑电路图



本实验的串行加法器又称为行波进位加法器，如图 2-10 所示：由若干位全加器 FA 串行 相连得到，其中低位 FA 的进位输出直接与相邻的高位 FA 的进位输入相连。因为高位进位产 生依赖于相邻低位 FA 的进位，所以各位运算依次完成，延迟较长，效率不高。

图 2-11 所示是四位串行加法器仿真图，由四个图 2-12 所示的一位全加器串行形成， 最高位是符号位，数字有效位是三位。M 端控制器件做加法(M=0)或减法(M=1)，因为“任意 两数之差的补码等于被减数的补码与减数相反数的补码之和”，所以 M 端工作原理基于“[Y]

补

→[-Y]补”求补运算原则：取反再加 1。图 2-11 中执行的运算是：“101+001=110”。

图 2-11 串行加法器

图 2-12 一位全加器

本实验的并行加法器又称为超前进位加法器，如图 2-13 所示：也是由若干位全加器 FA 组成，但是每一个 FA 所需的低位进位都不依赖其他 FA，而是根据最低进位 C0 及各个位的加

数、被加数即可同时计算所有进位 C

n

。因此称之为“并行”加法器，其推导公式如下：

Ci+1=Ai·Bi + (Ai⊕Bi)·Ci 令 Yn= An·Bn Xn= (An⊕Bn)，则有：

Cn+1=Yn+Xn·C

n

故 ：

C1 = Y0+X0·C0

C2

= Y

1

+ X

1·C

1

= Y1

+ X

1

·Y0

+ X1

·X

0

·C0

C3 = Y2 + X2·C2 = Y2 + X2·Y1 + X2·X1·Y0 + X2·X1·X0·C0

C4

= Y

3

+ X

3·C

3

= Y3

+ X

3

·Y2

+ X3

·X

2

·Y1

+ X3

·X

2·X

1·Y

0

+ X

3·X

2·X1

·X0

·C

0



由图 2-13 中可以看出，每一个 FA 所需的低位进位都只跟 An、Bn 和 C0 相关，与相邻 FA 产生的进位无关。从右到左，随着位数越高，进位 Cn 生成电路越复杂。图 2-13 中执行的运 算是：“101+011=1000”；因为进位影响到符号位改变，所以导致溢出，数码管旁边的 OF 指示灯点亮。

图 2-13 并行加法器

**五、实验步骤**

● 请制作半加器和全加器的电路，并验证其输入输出是否与图 2-8 和图 2-9 一致。

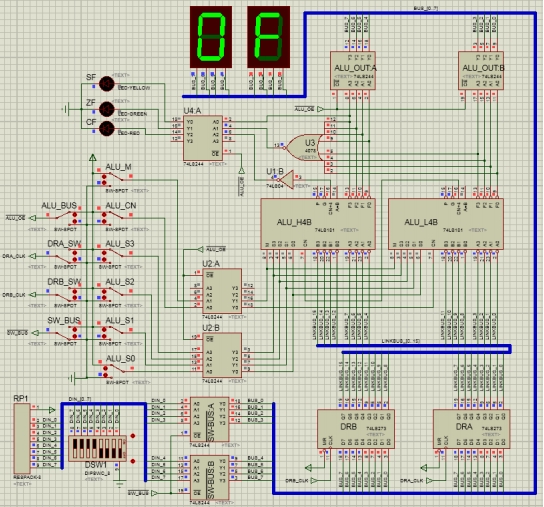
● 令 BUS\_[7..4]=0101 ， BUS\_[3..0]=0010 ， M=0 ， 写 出 该 运算 的 真 值 表 达 式 ， 记 录 并 比 较串行与并行加法器的运算结果，是否有溢出？如果改为 BUS\_[3..0]=0011 ，结果如何呢？

● 令 BUS\_[7..4]=0101 ， BUS\_[3..0]=0011 ， M=1 ，写 出该 运 算 的 真值 表 达式 ， 记 录 并比 较 串 行 与 并 行加 法 器 的运 算 结果，是否有溢 出 ？如果 运 算器 的 输 入 改为 BUS\_[7..4] =0011 ， BUS\_[3..0]=0101，M=1 不变，结果如何呢？

● 令 BUS\_[7..4]=1101 ， BUS\_[3..0]=0011 ， M=0 ， 写 出该 运算的 真 值 表达式 ， 记 录 并比 较串行与并行加法器的运算结果。是否有溢出？如果改为 M=1，输入不变，结果如何呢？

**六、思考题**

1. 请问本实验运算器电路是补码运算器还是原码运算器？假设需要把运算器电路改为 四位无符号数的原码加法器，请问电路怎么修改？
2. 本实验中，四位补码的表数范围是多少？若要修改为五位补码运算器（符号位一位， 数值位四位），其表数范围又是多少？请问电路怎么修改？



**2.3 运算器实验**

**一、实验目的**

1. 了解算术逻辑运算器（74LS181）的组成和功能。
2. 掌握基本算术和逻辑运算的实现方法。

**二、实验内容**

运用算术逻辑运算器 74LS181 进行有符号数/无符号数的算术运算和逻辑运算。 **三、实验器件**

1. 算术逻辑运算器（74LS181）。
2. 三态门（74LS244）及寄存器（74LS273、74LS373）。

**四、实验原理**

图 2-14 运算器通路

本 实 验 的算 术 逻 辑运算器 电 路 如 图 2-14 所 示 ： 输入和 输 出 单 元跟 上 述实 验 相 同 ：拨码

开关用来给出参与运算的数据，并经过三态门 74LS244 和数据总线 BUS 相连；一对数码显示 管用来显示数据总线 BUS 的内容。

运 算器则 由两 个 74LS181 以 串行 进 位形 式构成 8 位字长 的 ALU ：ALU\_L4B 为低 4 位 运算 芯 片 ， ALU\_H4B 为高 4 位 运算 芯 片 。 ALU\_L4B 的 进 位 输 出端 CN+4 与 ALU\_H4B 的 进 位 输 入 端 CN 相连，使低 4 位 运算 产生 的进 位 送进高 4 位 运算 中 。ALU\_L4B 的进位 输入 端 CN 连 接到 外 来进位端开关 ALU\_CN，ALU\_H4B 的进位输出端 CN+4 经过反相器 74LS04，通过三态门接到溢 出标志位 CF 指示灯（CF=1，即 ALU 运算结果溢出）。

ALU 除了 溢出 标志位 CF 外 ， 还有两个 标志位 ：零标志位 ZF （ZF=1，即 ALU 运算结 果为 0）和符号标志位 SF（SF=1，即运算结果为负数；SF=0 即运算结果为正数或 0）。

ALU 的 工作方式可通过设 置两个 74181 芯片 的控制信号 （ALU\_S0 、 S1、 S2 、 S3 、M 及 CN） 来实现 , 其逻辑功能表由表 2-6 给出，表中“A”和“B”分别表示参与运算的两个数 ， “+”表示逻辑或，“加”表示算术求和。

表 2-6 74LS181 逻辑功能表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| S3 | S2 | S1 | S0 | M=0（算术运算） | | M=1  （逻辑运算） |
| CN=1 无进位 | CN=0 有进位 |
| 0 | 0 | 0 | 0 | F= *A* | F=A 加 1 | F= *A* |
| 0 | 0 | 0 | 1 | F= *A* + *B* | F=（ *A* + *B* ）加 1 | F= *A* + *B* |
| 0 | 0 | 1 | 0 | F= *A* + *B* | F=（ *A* + *B* ）加 1 | F= *AB* |
| 0 | 0 | 1 | 1 | F=0 减 1 | F=0 | F= 0 |
| 0 | 1 | 0 | 0 | F= *A* 加 *A B* | F= *A* 加 *A B* 加 1 | F= *AB* |
| 0 | 1 | 0 | 1 | F=（ *A* + *B* ）加 *A B* | F=（ *A* + *B* ）加 *A B* 加 1 | F= *B* |
| 0 | 1 | 1 | 0 | F= *A* 减 *B* 减 1 | F= *A* 减 *B* | F= *A* ⊕ *B* |
| 0 | 1 | 1 | 1 | F= *A B* 减 1 | F= *A B* | F= *A B* |
| 1 | 0 | 0 | 0 | F= *A* 加 *AB* | F= *A* 加 *AB* 加 1 | F= *A* + *B* |
| 1 | 0 | 0 | 1 | F= *A* 加 *B* | F= *A* 加 *B* 加 1 | F= *A* ⊕ *B* |
| 1 | 0 | 1 | 0 | F=（ *A* + *B* ）加 *AB* | F=（ *A* + *B* ）加 *AB* 加 1 | F= *B* |
| 1 | 0 | 1 | 1 | F= *AB* 减 1 | F= *AB* | F= *AB* |
| 1 | 1 | 0 | 0 | F= *A* 加 *A* | F= *A* 加 *A* 加 1 | F=1 |
| 1 | 1 | 0 | 1 | F=（ *A* + *B* ）加 *A* | F=（ *A* + *B* ）加 *A* 加 1 | F= *A* + *B* |
| 1 | 1 | 1 | 0 | F=（ *A* + *B* ）加 *A* | F=（ *A* + *B* ）加 *A* 加 1 | F= *A* + *B* |
| 1 | 1 | 1 | 1 | F= *A* 减 1 | F= *A* | F= *A* |

运 算器 ALU 的输出 经 过三 态门（74LS244 ）和数据 总线 BUS 相连 。当 运算器 使 能开关 低 电平有效（ ALU\_OE=0）的时 候，运算器三个部位的三态门 244 状态为直通： 74181 的控制信 号 （ S0~S3、M、CN ） 全 部连 通 ；74181 的 运 算 标 志 位（ CF 、ZF 和 SF ） 的指 示 灯 全 部连通； 以 及 74181 的 运 算结 果 输 出 到 数 据 总 线 BUS。 当 ALU\_OE=1 的 时 候 ，74181 停止工作， 此 时 74181 的输出端数据为无效数据，与数据总线 BUS 隔断。

运 算 器 ALU 的 两 个数 据 输 入 端 分 别由 两 个 数 据 暂 存 器 （ 74LS273 ） DRA 、 DRB 锁 存 ， 74LS181 将 DRA 、DRB 内的数据作为上述表 2-6 中参与运算的数 A 和 B 。由于 DRA、DRB 已 经 把 数 据 锁 存 ， 只 要 74LS181 的 控 制 信 号 不 变 ， 那 么 74LS181 的输 出 数 据 也 不 会 发 生 改 变。数据暂存器 DRA 、DRB 的输入连至数据总线 BUS，在 DRA\_CLK、DRB\_CLK 端开关出现上

升沿跳变的时候，总线 BUS 的数据分别打入 DRA、DRB 锁存。

**五、实验步骤**

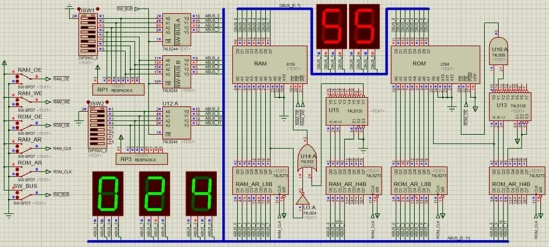
● 令 图 2-14 各 个 开 关 的初 始 状 态 为 ： DRA\_CLK=DRB\_CLK=0 ，SW\_BUS = ALU\_OE = 1 ， (S3,S2,S1,S0,M,CN)=(1,1,1,1,1,1) 。 操 作 拨 码 开 关 ， 向 数 据 暂 存 器 DRA 写 入 AAH ， DRB 写 入 55H （即 A=0xAAH ， B=0x55H ）。改变运 算 器的 控 制信 号 (S3,S2,S1,S0,M,CN)的 组 合，运 算 器使 能（ALU\_OE=0 ），观 察 运 算器 的输 出和标志 位， 并填 入下 表中，与 理论 值 比较，验证 74LS181 的功能。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DRA | DRB | S3 | S2 | S1 | S0 | M=0（算术运算） | | 标志位  CF/ZF/SF | M=1  逻辑运算 |
| CN=1 无进位 | CN=0 有进位 |
|  |  | 0 | 0 | 0 | 0 | F= | F= |  | F= |
|  |  | 0 | 0 | 0 | 1 | F= | F= |  | F= |
|  |  | 0 | 0 | 1 | 0 | F= | F= |  | F= |
|  |  | 0 | 0 | 1 | 1 | F= | F= |  | F= |
|  |  | 0 | 1 | 0 | 0 | F= | F= |  | F= |
|  |  | 0 | 1 | 0 | 1 | F= | F= |  | F= |
|  |  | **0** | **1** | **1** | **0** | **F=** | **F=** |  | F= |
|  |  | 0 | 1 | 1 | 1 | F= | F= |  | F= |
|  |  | 1 | 0 | 0 | 0 | F= | F= |  | F= |
|  |  | **1** | **0** | **0** | **1** | **F=** | **F=** |  | F= |
|  |  | 1 | 0 | 1 | 0 | F= | F= |  | F= |
|  |  | 1 | 0 | 1 | 1 | F= | F= |  | F= |
|  |  | 1 | 1 | 0 | 0 | F= | F= |  | F= |
|  |  | 1 | 1 | 0 | 1 | F= | F= |  | F= |
|  |  | 1 | 1 | 1 | 0 | F= | F= |  | F= |
|  |  | 1 | 1 | 1 | 1 | F= | F= |  | F= |

● 操作拨码开关，向数据暂存器 DRA、DRB 分别打入有符号数 +7AH，-75H（即 A= + 0x7A H ， B= - 0x75 H ）。改 变运 算 器 的 控 制 信 号 (S3,S2,S1,S0,M,CN) 的 组 合 ， 运 算 器 使 能 （ ALU\_OE=0 ）， 观 察 运 算 器 的 输 出 和 标 志 位 ， 并 填 入 上 表 中 ， 与 理 论 值 比 较 ， 验 证 74LS181 的功能。

**六、思考题**

1. 74181 组成的运 算 器通 路，可以 区分 有符 号数运 算 和无 符号数运算 么？两 者的运 算 过程有不同么？两者的 数值表示范围有不同么？
2. 当 74181 进 行无符号 数 运 算 的 时 候 ， 运 算 结 果 的 标志 位 SF 有 无 意 义 ？在有 符 号 数 运算和无符号数运算过程中，标志位 CF 、ZF 和 SF 的含义都是一样的么？



**2.4 存储器实验**

**一、实验目的**

1. 了解静态随机存储器 RAM 和只读存储器 ROM 的工作特性及读写方法。
2. 掌握存储器与总线的连接及存储器地址空间映射的原理。

**二、实验内容**

设 计 一 个 8 位 字 长 的 存 储 器 系 统 ， 包 括 ROM 和 RAM 两 个 地 址 相 互 独 立 的 存 储 器 ， 实现对 ROM 和 RAM 存储器的数据读写操作及数据成批导入 ROM 的操作。

**三、实验器件**

1. 只读存储器 2764 及静态随机存储器 6116。
2. 三态门（74LS244）、寄存器（74LS273）及 3-8 译码器(74LS138)。

**四、实验原理**

本 实 验 的存 储 器 通路如图 2-14 所 示 ，由 地 址 输 入单 元、存 储 器 及地 址选 择 电 路 组成 。 存 储 器 通 路 共 有 两 条 总 线 ： 12 位 地址 总 线 ABUS\_[0..11] 和 8 位 数 据 总 线 DBUS\_[0..7] 。 下 图左 边是拨码开关 构成的 12 位地 址输入端 连在 地址总线上 ，通 过三 个绿 色数 码管输出显 示 12 位地址信息 。右 边则是 存储 器 ROM 、RAM 及其地 址选 择电路。ROM 和 RAM 存储器内部 有三 态门结构，其数据输出端直接连在数据总线上，通过两个红色数码管显示 8 位数据信息。

图 2-15 存储器通路

存储器是用来存储信息的部件，是计算机的重要组成部分，常见的半导体存储器类型主 要 有 ROM 和 RAM ： ROM 是 Read Only Memory （ 只 读 存储器 ） 的 缩 写 ， RAM 是 Random Access Memory（随机存取存储器）的缩写。ROM 价格较低，一般容量较大，在系统停止供 电的时候仍然可以保持数据；而 RAM 价格较高，一般容量较小，在掉电之后就丢失数据， 典 型 的 RAM 就 是 计 算 机的 内 存 。 本 实 验 中 使 用 的 ROM 是 2764 （ 8K × 8 位 ），RAM 是 6116（2K×8 位），如图 2-16 所示。

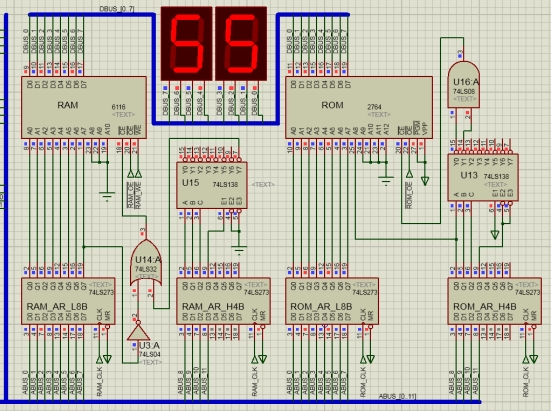


图 2-16 存储器及地址选择电路

ROM 芯 片 2764 的数 据线 D0~D7 接 到 数 据 总线， 地 址 线 A0~A8 由 地 址 锁 存 器 74LS273 给 出，用来对 ROM 片内存储单元寻址。其余地址线 A9~A12 接地，所以其理论容量为 8K 字节。 2764 有两个控制端：CE（片选）、OE（读）。

RAM 芯 片 6116 的数 据线 D0~D7 接 到 数 据 总线， 地 址 线 A0~A7 由 地 址 锁 存 器 74LS273 给 出，用来对 RAM 片内存储单元寻址。其余地址线 A8~A10 接地，所以其理论容量为 2K 字节。 6116 有三个控制端：CE（片选）、OE（读）、WE（写）。

存储器通路控制信号的逻辑功能见表 2-7。值得注意的是：在对 ROM 或 RAM 读写的时候， 首 先必 须 在 存储 器 的 片 选有 效 （ CE=0 ） 前 提 下 ， 才能 对 相应 的 存 储器 读 （ OE=0 ） 或 写 （WE=0）。例如：对 ROM 芯片 2764 进行读操作，必须使能 ROM\_CE=0 且 ROM\_OE=0。存储器 片 选信号 ROM\_CE 和 RAM\_CE 是由地址信号的高 4 位 ABUS\_8~11 经过片选逻辑电路自动形成， 不需要拨码开关控制。

表 2-7 存储器通路控制信号说明

|  |  |  |
| --- | --- | --- |
| **信号名称** | **作 用** | **有效电平** |
| ROM\_CLK | 2764 地址的锁存脉冲信号 | 上升沿跳变有效（开关手动） |
| RAM\_CLK | 6116 地址的锁存脉冲信号 | 上升沿跳变有效（开关手动） |
| ROM\_CE\ | 2764 的片选有效信号 | 低电平有效 （地址驱动） |
| RAM\_CE\ | 6116 的片选有效信号 | 低电平有效 （地址驱动） |
| ROM\_OE\ | 2764 的读允许信号 | 低电平有效（开关手动） |
| RAM\_OE\ | 6116 的读允许信号 | 低电平有效（开关手动） |
| RAM\_WE\ | 6116 的写允许信号 | 低电平有效（开关手动） |
| SW\_BUS\ | 地址总线输入允许信号 | 低电平有效（开关手动） |

其 次 ，必须 在 地址 锁 存 器 （74LS273 ） ROM\_AR 、 RAM\_AR 锁 存 地址信 号 ，才 能 选中 存 储 器 片 内 相 应 的 单 元 。 地 址 锁 存器 ROM\_AR 和 RAM\_AR 的 输 入 都 连 接 至 地 址 总 线 ABUS\_0~7 ， 在 其 CLK 端 开 关 出 现 上 升 沿 跳 变 的 时 候 ， 地 址 总 线 ABUS\_0~7 的 数 据 打 入 ROM\_AR 或 RAM\_AR 锁 存 。 锁存后无 论 地址总线 ABUS 如 何 变化 ， 选中的存 储 单元 也 不 会发生改变，可以进行稳定的读写操作（存储器数据端输入或输出）。

存储器通路设计的最重要环节是存储器与地址总线的连接，因为连接方式决定了存储器 地 址 空 间的映 射 关系， 即 决 定了 每 个 存储器 芯 片 在整个 存 储空间 中 的 地 址 范 围。12 位地 址 总 线 的 理 论 地 址 空 间 为 4K （ 000H~FFFH ）， 本 实 验 分 配 其 中 最 低 的 512 地 址 为 ROM 区 (000H~1FFH)，最高的 128 地址为 RAM 区（F80H~FFFH），其余留空，如表 2-8 所示。

表 2-8 存储器与地址总线连接范围及选片

|  |  |  |  |
| --- | --- | --- | --- |
| **A11 A10 A9 A8** | **A7 A6 A5 A4 A3 A2 A1 A0** | **地址范围** | **选片** |
| 0 0 0 0  ………….  0 0 0 1 | 0 0 0 0 0 0 0 0 …………………..  1 1 1 1 1 1 1 1 | 000H~1FFH | ROM 芯片 2764  （实际容量 512 字节） |
| ………… | ………………. | 200H~F7FH | （空） |
| 1 1 1 1  ………..  1 1 1 1 | 1 0 0 0 0 0 0 0  ………………..  1 1 1 1 1 1 1 1 | F80H~FFFH | RAM 芯片 6116  （实际容量 128 字节） |

存储器通路设计一般将地址总线区分为低位地址线和高位地址线两部分：低位地址线直 接和存储器芯片的地址信号连接作为片内地址译码，而高位地址线的连接主要用来产生片选 信号（称为片间地址译码），以决定每个芯片在整个存储系统中的地址范围。

在 本 实 验 中 ， 12 位 理 论 地 址 空 间 分 为 低 8 位 地 址 线 和 高 4 位 地 址 线 。 低 8 位 地 址 线 ABUS\_0~7 分 别与 ROM 和 RAM 芯 片的地址 线 A0~A7 共 用； 高 4 位地 址 线 ABUS\_8~11 则 通 过 两 个 3-8 译码器把地址空间分为 16 个部分，如图 2-16 所示。ROM 芯片的片选由低位 3-8 译码 器的最低 2 个部分片选信号的片选电路形成， RAM 芯片的片选由高位 3-8 译码器的最高 1 个 部 分片 选信 号与 RAM 的 A7 地 址线的片选电 路形成。值 得注意的 是：片选 电路的 逻辑 组合 不 是唯一的，可以有多种实现形式。

**五、实验步骤**

**实验 1：二进制数据批量导入 ROM**

● 新建文本文件（.txt），按照下列格式输入数据。然后，把.txt 后缀改名为.asm 后缀： ORG 0024H

DB 55H

DB 55H

DB 55H

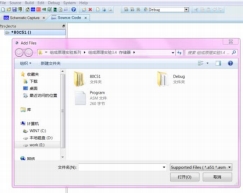
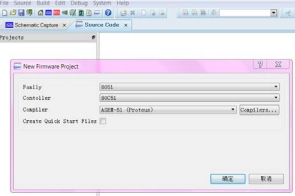
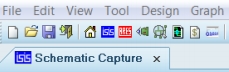
DB 55H

DB 01H

DB 01H

DB 01H

DB 01H



DB 0FFH

DB 0FFH

DB 0FFH

DB 0FFH

DB 01H

DB 01H

DB 01H

DB 01H

END

注：上述代码中 “ORG xxxxH”规定存储数据的首地址，可以从 0000H 开始，也可以 从 任 意地 址 开 始。每 一 行“DB xxH ” 是一 个 存 储单元 存 放 的数据（ 注 ：16 进制 A~F 前面 要加 0），从 首地 址开始， 按顺序 存放。中间 的 空行 仅仅为阅读 方便，没有实 际意义。文件 必须以“END”作为结束

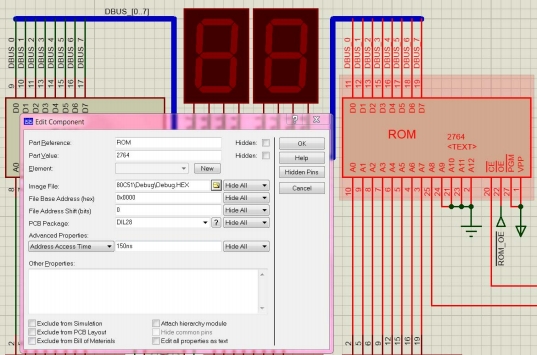
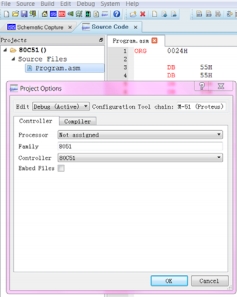
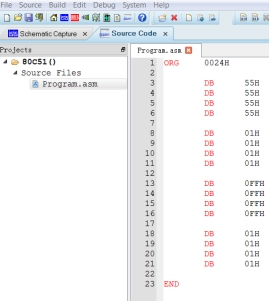
● 在 proteus 开 发 环 境 中 ， 菜 单 栏 里 点 击 Source Code 面 板 （ 左 下 图 的 最 右 边 ， Graph 下面那个图标），然后会弹出 Source Code 标签页，如右下图所示：

● 如 果 该 项 目 没 有 打 开 过 asm 文 件 ， 则 打 开 的 标 签 页 为 空 ， 需 要 在 菜 单 栏 里 选 择 Source->Create Project，在 弹出 的对话 框里如左下 图所示勾选（注： Creat Quick Start Files 选 项 不 勾），再按 “确 定 ” 按钮 ； 在 标签 页 上会 出 现 一个“\*80C51() ” 的新 project， 右 键单 击 “ \*80C51()”，在 弹出 的 菜单列表中 选择 Add files（ 或 Import Existing File ） 选项， 在弹出的 视窗内选中所需的 asm 文件，按确定就添加进 project。如右下图所示。

● 在已经添加的 asm 文件上双击，在右侧会打开 asm 文件的文本内容，如左下图所示。 如 果 一 个 proteus 的 项 目 中 已 经 建 立 了 project 且 打 过 asm 文 件 ， 则 弹 出 的 Source Code 标 签 页 会 直 接 显 示 左 下 图 。 在 左 下 图 的 右 侧 ， 可 以 直 接 修 改 asm 文 件 的 内 容 ， 并 保 存 。 如 果 要 更 换 其 他 asm 文 件 ， 可以 右 键 单击 不 要的 asm 文 件 ， 菜 单列 表 中选 择 “Remove file ” 删 除 当前 asm 文件，然后遵循前述操作，重新添加新的 asm 文件进入 project。

● 如 果 asm 文 件 是 第一 次 编 译 ，则 须 右 键单 击 asm 文 件 ， 在 弹 出的 菜 单 列 表 中 选 择点 击 Project Settings ， 弹 出 对 话 框 中 controller 选 择 ” 80C51 ”， 注 意 Embed files （ 或 Attach files ）要去掉选中！才能确保编译后生成的二进制 hex 文件在项目当前的目录下。最后，点 击“OK”选项，如右下图所示。

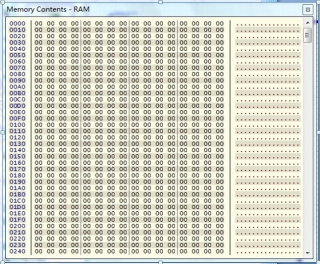
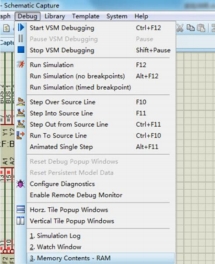
● 右键单击 asm 文件，右键单击 asm 文件点击 Build Project 执行编译。编译成功后，在 下 方的 显示 栏里会出现“Compiled successfully！”字 样。此时， 在项 目当 前文件夹的子 文件



夹 80C51 的 孙 文 件 夹 Debug 里 面 ， 就 有 编 译 后 生 成 的 hex 二 进 制 文 件 （ 请 注 意 看 文 件 的 修 改 日 期 ， 确 认是最近 编 译 的 文 件）。该 hex 文 件 默 认 编 译后 的 文 件名为“ Debug.hex ”。最 好 把 Debug.hex 文件重新命名，放置在固定的目录里保存。否则下一次编译其他 asm 源程序的 时候，会在相同路径生成重名的 Debug.hex，把之前编译的 hex 文件冲掉。

● 双击 ROM 芯片，弹出如下图的对话框：在 Image File 中选择所需烧写的 hex 文件， 点 击 OK ， ROM 加 载 成 功 。 最 好 不 要 直 接 加 载 上 述 直 接 编 译 生 成 的\80C51\Debug 路 径里 面 的 Debug.hex ， 应加 载重命名在 固定 目录 下的 hex 文件。 否则 ，当编译其他 asm 源 程序 后， Debug.hex 自动刷新，容易造成 ROM 错误加载本应烧写到其他 ROM 里面的 hex 文件。

● 假设仿真出现以下报错，可能是项目路径里面原来加载 hex 文件的路径被取消，或是 被移动到不同路径文件夹，或是文件名变了。特别是在项目里面有多个 ROM 需要加载的时 候，要留意在移动或修改项目的时候不要破坏原有的 ROM 加载路径。



**实验 2：查看项目中存储器 ROM 和 RAM 的内容**

● 在 proteus 项目的仿真过程下按 “暂停”，在点击菜单栏里的 Debug，在弹出的菜单列 表 最下 方有 Memory Contents-ROM/RAM ， 如左 下图 所示 。项 目里 有多 少个 存储 器，就 有多 少 个 Memory Contents 可 供选择 。点击 所 需 查看存储 器对 应 的 Memory Contents- 选 项 ，则 弹 出如右下图所示的当前存储器内容，其中蓝色为地址，黑色为存储单元中的数据，按顺序从 左到右，从上到下排列显示。值得注意的是，右下图显示的存储器地址是存储器芯片地址线 A0~A7 所 指定的地 址 ， 即实验 原 理 所述 的 低 位“ 片内地 址”。而 “ 片 内地址 ” 加上 存储 器片 选信号译码所用的高位“片间地址”，才是地址空间的真实地址。

**实验 3：存储器 ROM 和 RAM 的读写**

● 依照上述实验 1 步骤，加载 project.asm 文件编译的 hex 二进制文件到 ROM 芯片 2764。 依照上述实验 2 方法，查看 ROM 烧写是否正确。

● 令 图 2-16 各 个 开 关 的 初 始 状 态 为 ： ROM\_OE=1 ， RAM\_OE=1 ， RAM\_WE=1 ， SW\_BUS=0。允许拨码开关给地址总线 ABUS\_[0..11]赋值

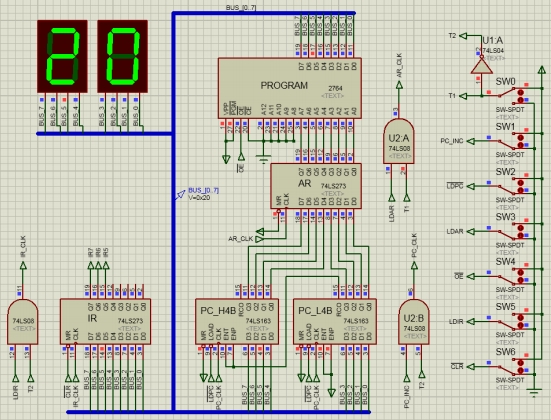
● 操 作 拨 码 开 关 ， 向 地 址 锁 存 器 ROM\_AR 打 入 地 址 024H ， 然 后 2764 输 出 使 能 （ROM\_OE=0）, 在数据总线 DBUS\_[0..7]的红色数码显示管上查看存储单元 024H 读出的内 容 。 再 操 作 拨 码 开 关 ， 向 地 址 锁 存 器 RAM\_AR 打 入 地 址 F80H ， 6116 输 入 使 能 （RAM\_WE=0）把存储单元 024H 的内容写入存储单元 F80H。

● 2764 输出失效 （ROM\_OE=1 ） , 6116 输出 使能（RAM\_OE=0 ） , 在数据总 线的 红 色 数 码显示管上查看存储单元 F80H 写入的内容是否正确。

● 按 照 上 述 操 作 ， 把 ROM 地 址 024H 、028H 、 02CH 、 030H 的 内 容 依 次 写 入 RAM 地 址 F80H、F81H、F82H、F83H。依照上述实验 2 方法，查看 RAM 的写入是否正确。

**六、思考题**

1. 是不是 烧写入 ROM 的 ASM 文件里面定义的所有数据都可以被访问到？假设把实 验 1 中的 ASM 文件开头改为“ORG 0224H ”，请 问 烧 写 进 去 的 数 据 还能 被 读 出 么？ 如 果 不能，ROM 的片选电路要如何修改？
2. 请给出 RAM 片选电路（F80H~FFFH）的第二种逻辑组合实现形式。假如 RAM 的范 围改为 800H~874FH，请问片选电路的逻辑组合形式是怎样的？
3. 为 何 ROM 和 RAM 需 要 使 用 两 个独 立 的 3-8 译码器？若 RAM 的 片 选 电 路 与 ROM 的片选电路共用一个 3-8 译码器，即 ROM 所在 3-8 译码器的最低 2 个端口给 ROM 使用，最 高 1 个端口给 RAM 使用。此时 ROM 和 RAM 的地址范围各自为多少？



**2.5 微程序控制器实验**

**一、实验目的**

1. 了解微程序控制器的组成原理及工作方式。
2. 掌握微程序的编制及写入，观察微程序的运行。

**二、实验内容**

设计一个可自动运行的 CPU 电路，利用时序电路来产生指令执行所需的时间先后次序， 通过微程序控制器来代替手动拨码开关的人工操作，在数据通路中完成程序的跳转重定向。

**三、实验器件**

1. 存储器 2764 和计数器（74LS163、74LS192）。
2. D 触发器（74LS74）、寄存器（74LS273）及移位寄存器(74LS194)。

**四、实验原理**

如 图 2-17 所 示 ， 指 令 寄 存 器 IR （ 74LS273 ）、 程 序 计 数 器 PC （ 74LS163 ）、 存 储 器 PROGRAM(2764) 及 其 地 址 寄 存 器 AR （74LS273 ）组 成 了 一 个 “ 手 动 版 ” 的 数 据 通 路 。 该通路 是单总线结构（所有部件共用 8 位总线 BUS，通过两个绿色数码管显示 8 位总线信息）和单 存储器结构（即程序和数据共享存储器 PROGRAM）。

图 2-17 数据通路



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MR | L  D | ET | E  P | 功能 | Q0 Q1 Q2 Q3 |
| O | × | × | × | 清除 | 0 0 0 0 |
| 1 | 0 | × | × | 加载 | Q0 Q1 Q2 Q3 = D0 D1 D2 D3 |
| 1 | 1 | O | × | 保持 | Q Q Q Q  3 |
| l | l | × | 0 | 保持 | Q Q Q Q  3 |
| l | l | 1 | 1 | 自加 1 | {Q0 Q1 Q2 Q3}状态码+1 |

在图 2-17 所示的数据通路中，程序计数器 PC 采用两个同步计数器 74LSl63 串联形成八

位递增计数器，74LSl63 的逻辑功能表如表 2-9 所示：D0

、D1、D

2

、D

3

为并行输入端；Q0

、Q

1、

Q2、Q3 为并行输出端；ENT、ENP 为递增使能端；LOAD 为置数端；MR 为直接无条件清零端；

CLK 为 时 钟 输 入 端 ； RCO 为 进 位 输 出 端 （ 当 Q3

-Q0

输 出 端 递 增 溢 出 ， 则 RCO=1 ）。值得 注 意 的

是：表 2-9 中所列 74LS163 全部功能都必须在 CLK 端上升沿跳变后才能实现。

表 2-9 74LS163 逻辑功能表

数 据

通 路 的 运

行 包 括 了

两个阶段： 取 指 周 期

（ 取 出 指

令） 和执行周期（ 执行取指周 期取出的指令）。而 总线 BUS 上的 信息 流动 路径 也相 应分为指 令 流 和 数据 流 。 指 令 流：取 指 周期 ， 指 令 信息 从 存 储 器流 向 指 令寄 存 器（即 ROM->IR ），实 现 指令 译码 ；数 据流 ：执 行周 期， 数据 信息 从存 储器 流 向唯一的 指令执行部件 PC（ 即 ROM- >PC），实现程序的重定向。值得注意的是，若有多个指令执行部件，则数据流还有不同的分 支路径

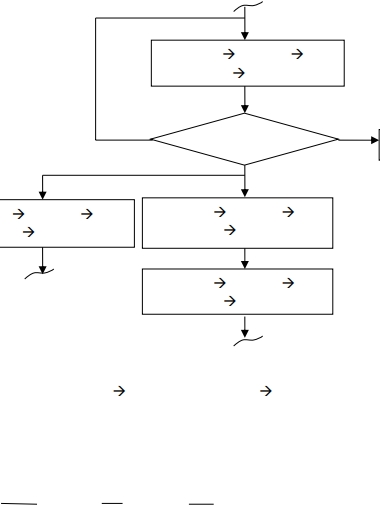
因为数据通路只有 PC 作为唯一的执行部件，用以验证指令跳转的功能。所以如表 2-10 所 示 ， 本实 验 设 计了 空 指令 NOP 、 停 机 指 令 HLT 和 两 条 不同 类 型 的跳 转 指令： 相 对 寻 址 的 JMP1 指令和直接寻址的 JMP2 指令。

表 2-10 微程序控制器实验的指令表

|  |  |  |  |
| --- | --- | --- | --- |
| 汇编助记符 | [OP 码]  I7 I6 I5 | 机器语言程序 | 指令说明 |
| JMP1,addr | 0 0 1 | 00100000; JMP1  xxxxxxxx; addr | 相对寻址：程序最终跳转到地址 xxH 执行 PC->xxH |
| JMP2,[addr] | 0 1 0 | 01000000; JMP2  xxxxxxxx;[addr] | 直接寻址：程序最终跳转到地址 yyH 执行 PC->yyH，yyH=[xxH] |
| HLT | 1 1 1 | 11100000; HLT | 硬件停机 |
| NOP | 0 0 0 | 00000000; NOP | “空”指令 ，不执行任何操作 |

图 2-18 列 出 了上 述 四 条 指 令 的逻 辑 流 程图 ， 其 中 方框 代 表 了 从某 个源 部 件 经过 总线 BUS 到 达另一 个目 标部件的 信 息路 径 ， 菱形代 表 了 指 令 译码结 果 （决 定不同指 令 的 跳 转）。 菱形之上的方框属于所有指令都必须经历的取指周期，而菱形之下各个分支代表了不同指令 的 执 行 周 期 。 为 了 信 息 的 稳 定 可 靠 传 输 ， 每 一 个 方 框 都 包 含 两 个 等 长 的 节 拍 T1 和 T2 。 T1 时 刻 ， 源 部 件 的 信 息 输 出 到 总 线 BUS ， 所 以 AR\_CLK 上 升 沿 有 效， 使 得 AR 地 址 对 应 的 ROM 存 储单 元 稳 定输 出 信息 到 总 线 BUS ； T2 时 刻，总 线 BUS 上 的 信息 输入 到 目 标部件， 所以 PC\_CLK 和 IR\_CLK 上升沿有效，使得总线 BUS 上的信息稳定打入到 PC（重定向指令 地址），或稳定打入到 IR（ 同时 PC 自加 1）。

通 过图 2-18 可 以发现，所有跳转 指令的逻辑流 程图实 际上仅需要由 两种方 框组成：在 取指 周期中出现的 指令流｛ROM IR ｝ 和在执 行周期中 出现的数 据流｛ROM PC ｝。JMP1 和 JMP2 指 令 的 区 别 仅 仅 在 于 JMP2 指 令 做 了 递 归 操 作 ， 执 行 了 两 次 相 同 的 数 据 流



【T1】PC AR,ROM BUS 【T2】BUS IR, PC+1

【T1】PC AR,ROM BUS 【T2】BUS PC

硬件停机

【T1】PC AR,ROM BUS 【T2】BUS PC

【T1】PC AR,ROM BUS 【T2】BUS PC

｛ ROM PC ｝ 操作。而 NOP 指 令 和 HLT 指 令 比 较 特 殊 ， 只 有 取 指 周 期 ，没 有 执 行 周 期 。 指令译码后，NOP 指令直接返回下一个取指周期，而 HLT 指令直接硬件停机。

00 000

00 000

NOP

P1 (00 I

7I

6I

5)

HLT

00 111

JMP1

00 001

00 000

JMP2

00 010

00 001

00 000

图 2-18 指令逻辑流程图

为 了 完 成 指 令流｛ROM IR ｝ 和 数据 流 ｛ROM PC ｝，数据 通 路 中需 要 根 据一 定 的 时 序安排一系列的微操作信号来确保信息从某个部件输出到总线，再从总线输入到另一个部件。 数 据 通路 的操 作信 号 说 明如表 2-11 所 示 。值得 注 意的 是 ： 程序计 数 器 PC 、 地址 寄存 器 AR 以 及 指 令寄 存 器 IR 的 各 自 锁 存 脉 冲 信 号 PC\_CLK 、 AR\_CLK 及 IR\_CLK 分 别 由 PC\_INC 、 LDAR 及 LDIR 信号在 T1 或 T2 节拍形成的。所以，数据通路中实际需要的微操作信号只有： PC\_INC、LDPC、LDAR、OE、LDIR 和 CLR 信号。

表 2-11 数据通路的微操作信号表

|  |  |  |
| --- | --- | --- |
| **信号名称** | **作 用** | **有效电平** |
| PC\_CLK | 程序计数器 PC 的锁存脉冲信号 | 上升沿跳变有效 |
| PC\_INC | T2 节拍，PC\_INC 形成 PC\_CLK | 高电平有效 |
| LDPC\ | 程序计数器 PC 的加载信号 | 低电平有效 |
| AR\_CLK | 地址寄存器 AR 的锁存脉冲信号 | 上升沿跳变有效 |
| LDAR | T1 节拍，LDAR 形成 AR\_CLK | 高电平有效 |
| OE\ | 存储器 PROGRAM 的读允许信号 | 低电平有效 |
| IR\_CLK | 指令寄存器 IR 的锁存脉冲信号 | 上升沿跳变有效 |
| LDIR | T2 节拍，LDIR 形成 IR\_CLK | 高电平有效 |
| CLR\ | 指令寄存器 IR 的清零信号 | 低电平有效 |

指令流｛ROM IR｝和数据流｛ROM PC｝各自所需的微操作信号及其信息路径如 表 2-12 所示，每个周期内微操作信号再与(T1,T2)节拍逻辑组合。如果把表 2-11 中的微操 作信号用人工拨码开关来实现的话，实验者可以根据表 2-12 手动执行微操作信号，以 T 节 拍为基准时钟，顺序依次实现指令逻辑流程图 2-18 中的各个方框，完成一条机器指令的执 行。其中，实验者在菱形 P1 时刻，可以通过观测指令寄存器 IR 的{IR7,IR6,IR5}是否与某 个指令的 OP 码相同，来判断不同指令所需的执行周期（即朝哪个分支继续执行）。

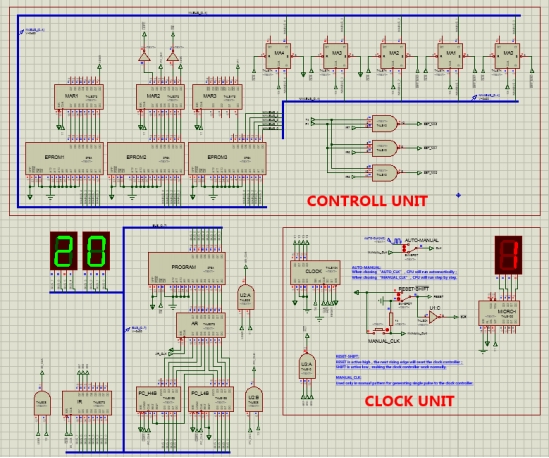


表 2-12 指令流和数据流的微操作信号表

|  |  |  |
| --- | --- | --- |
|  | 微操作信号（工作节拍） | 功能 |
| 指令流  ROM IR | LDAR(T1), OE (T1&T2),  LDIR(T2), PC\_INC(T2) | 【T1】：PC AR,ROM BUS 【T2】：BUS IR, PC+1 |
| 数据流  ROM PC | LDAR(T1), OE (T1&T2), LDPC (T1&T2), PC\_INC(T2) | 【T1】：PC AR,ROM BUS 【T2】：BUS PC |

进而，我们可以将图 2-17 的“手动版 ”数据通路升级为“自动版”的 CPU 电路，如下 图 2-19 所示。“自动版”CPU 由数据通路、时序电路及微程序控制器三大部分组成：下方左 边的数据通路保持不变，而上方红色框内的微程序控制器(CONTROL UNIT)取代了拨码开关及 人 工操 作 者；下方右边 红色 框 内的 时序 电路(CLOCK UNIT) 则 负责 提 供数据通路和微 程序控 制器依序运行所需的节拍序列｛T1,T2,T3,T4｝。

图 2-19 微程序控制器实验电路

微程序控制器(CONTROL UNIT)的主要任务是完成当前指令的翻译和执行，即将当前存 储 器 PROGRAM 读 出 的 指 令 转 换成 硬 件 逻辑部 件 工 作 的微 操作 信号 序列 ，完 成信息 传 送 和 各 种 处 理操作 。 因此，微 程序控制 器将 表 2-11 中 的 一条微 操作信号 视 为 一 个微命 令， 而所 有微命令的集合以并行二进制序列的形式构成微指令，如表 2-13 所示。

表 2-13 微指令结构图

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 微 代 码 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
| 微 命 令 | LDAR | LDPC | \ | PC\_INC | \ | \ | OE | \ | LDIR | \ | P1 | \ | uA4 | uA3 | uA2 | uA1 | uA0 |



上 表 中 的 微 指 令 字 长 24 位 ， 采 用 下 址 转 移 方 式 确 定 后 续 运 行 的 微 指 令 ： 微 指 令的 1-5 位表示顺序下一条微指令地址 [uA4-uA0]；而 6-24 位表示微命令集合（由于微指令 18-24 位 没 有 对 应 的 微 命 令 ， 所 以 表 2-13 只 显 示 了 微 指 令 1-17 位 ）。所有微 指令 都存 储在 专 用 的 控 制 存 储 器 (CM) 中 ， 如图 2-20 所 示 ： 控 制 存 储器 24 位 ， 由 3 个 ROM 芯 片 2764 组 成 。 微 命 令寄存器 18 位，由两个寄存器 74LS273 和一个 4 位触发器 74LS175 组成。

图 2-20 微程序控制器的控制存储器（CM）电路

在 指 令 逻辑 流 程 图 2-18 中 ， 一个 方 框 对 应一 条 特 定的 微 指 令， 方框 右上 方 是 当 前微 指 令 在控制 存 储器 CM 中的 地址， 右 下方 是顺序下 一条 微 指 令地 址[uA4-uA0] 。若 考 虑 方框 右 上 方 和 右下方 微 指令地 址的差异 ，在图 2-18 中 总共有 四 种不 同 的 方框， 所以本 实 验的 微 指 令代码表 2-14 所示。每条微指令的 1-5 位即下址字段[uA4-uA0]，而 6-24 位是控制字段。

在 T1 节拍，上图 2-20 中的控制存储器把当前微指令打入微命令寄存器，微指令中所有 标 识 为“ 1 ” 的位 表 示执 行 了相 应 的 微 操作（ 且 数 据通 路 自 动生成 LDAR AR\_CLK ）， 从 而实 现对应方框 内的信 息路径；反之，标识为“0” 的位 则代表没有执行对 应的微 操作。在 T2 节拍，数据通路自动生成了 LDIR IR\_CLK 和 PC\_INC PC\_CLK 的后续操作。

表 2-14 微指令代码表

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 微地址 | 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 |

一条机器指令所经历的所有微指令（方框）组成的序列称为该机器指令的微程序，每条

微指令包含了相同的取指周期微指令和各异的执行周期微指令序列。取指周期微指令地址必 须是控制存储器 CM 首地址 00000（因为系统启动时从首地址开始运行）。而菱形框 P1 (00

I7I

6I

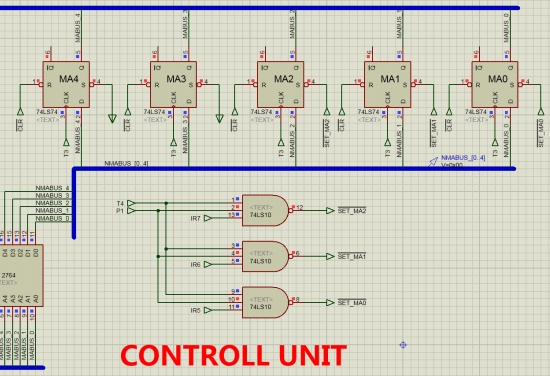
5)是地址转移逻辑电路，当取指周期结束的时候，根据机器指令 OP 码的 I

7I

6I

5

位进行译



码，转入该机器指令对应的执行周期微指令序列的入口地址[00I7

I6

I5

]。指令｛NOP, HLT,

JMP1, JMP2｝的执行周期微指令序列的入口地址分别为｛00000, 00111,00001,00010｝。

在所有指令执行周期的微指令序列末尾，最后一条微指令的[uA4-uA0]都必须是取指微 指令地址 00000，即一条机器指令结束后必须返回公共的取指微指令，以便于取出下一条指 令。而 NOP 指令因为取指后译码的执行指令入口地址仍为 00000，直接指向下一条指令的取 指周期微指令；所以 NOP 指令在取指后实际没有执行操作。

本实验微程序控制器的地址转移电路如图 2-21 所示，微指令地址寄存器五位(MA4- MA0)，由触发器 74LS74 组成，均自带清“0”端和预置端。地址转移逻辑需要增加 T3、 T4 节拍，T3 时刻，当前微指令的后续微指令地址[uA4-uA0]打入微地址寄存器。T4 时刻， 若当前微指令不是取指周期微指令（即 P1=0），则不执行任何操作，当前微地址转为顺序 下一条微指令地址[uA4-uA0]；若当前微指令是取指周期微指令（即 P1=1），则图 2-21 中的

地址转移逻辑电路根据指令寄存器 IR 的 I7

I6

I5

位强制置位寄存器 MA2-MA0，即修改当前微

地址[uA2-uA0]位，转向该指令的执行周期微指令序列的入口地址[00I7

I6

I5

]。

图 2-21 微程序控制器的地址转移电路

综上所述，一条微指令包括了 4 个节拍（T1、T2、T3、T4），其作用分别是： T1：当前微指令的微操作信号使能；信息从源部件输出到总线 BUS

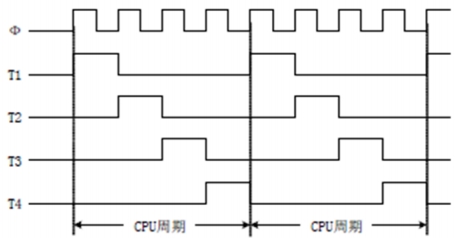
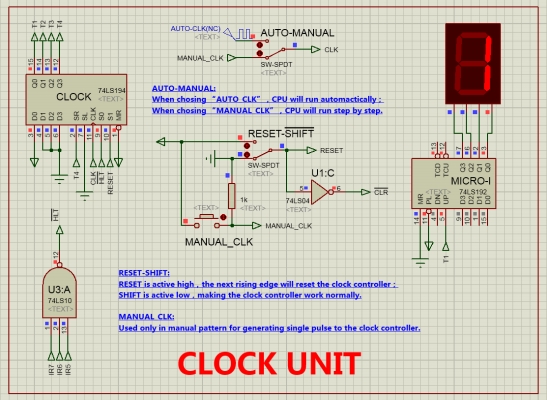
T2：信息从总线 BUS 打入目的部件；程序计数器 PC+1（取指周期微指令专用） T3：微指令下址取址

T4：根据指令寄存器 IR 所存的指令 OP 码修改微指令下址 00I7I

6I5

（取指周期微指令专用）

微程序控制器必须严格按照节拍时序来工作，所以时序电路的设计是非常重要的，本实 验 的 时 序电 路 如 图 2‐23 所 示 。 CLK 为 全 系统的基 准 时钟 信 号 ， 可以 由 方 波 信号 源 AUTO\_CLK 提供（双击信号源可以自行选择方波信号频率），或者通过人工操作按键 MANUAL\_CLK 手动步



进 ； 移 位 寄 存 器 74LS194 用 来 作 为 节 拍 生 成 器 ：当 HLT=1 且 RESET=0 ，74LS194 的 状 态

{S0,S1}={1,0} ，工 作 模 式为右移。而 S

R=T4 使得 74LS194 处于循环 右 移 模 式， 以 CLK 为基

准时钟，周而复始地发送 4 个基准时钟长度的时序节拍 T1-T4（T1->T4 的一次循环即为一个 微指令的运行周期，称为 CPU 周期），如图 2-22 所示：

图 2-22 基于 74LS194 的节拍时序状态图

为了便于观测微程序的运行，时序电路提供了微指令显示功能：一个由 T1 上升沿驱动 的升 序计数 器 74LS192，通 过数码管显 示当前运行第几条微 指令。时序电路还提供了软件停

机的“断点”功能（便于调试 ），即 HLT 指 令：当指令寄存器 IR 的 OP 码 I7

I6I

5=111 的时候，

硬 件 逻 辑 生成 HLT=0 ， 此 时 74LS194 的 状 态 {S0,S1}={0,0}， 工作 模 式为 保 持 ， 节 拍 时序 状 态静止在｛T1,T2,T3,T4｝={0,1,0,0}，即停机在 HLT 指令的取指周期 T2 节拍上。

图 2-23 微程序控制器实验的时序电路

此 外 ， 时 序 电 路 设 置 了 RESET 电 路 ， 用 来 初 始 化 时 序 发 生 器 74LS194 ，以 及 HLT 指令 停机后跳出“断点”。其“重启”的操作步骤如下：

1. RESET 置 1， 则 CLR=0 ，清 零 指令 寄存 器 IR（ 指令 OP 码 归零 且 HLT=1 ）和 微地址 寄存器 MA4-MA0 （ 微程 序 控制 器 从 取 指 周 期 开始 运行）。此时 ， 74LS194 的状 态 {S0,S1}={1,1}, 工作模式为送数。
2. 手动按一次 CLK，将｛T1,T2,T3,T4｝置位初始化值{1,0,0,0}。
3. RESET 置 0，74LS194 恢复循环右移模式，进入第一条指令的取指周期节拍时序。

（注：若是 HLT 指令“断点”后操作，则进入 HLT 指令后续下一条指令的取指周期） **五、实验步骤**

**实验 1：手动版程序通路运行**

●在数据通路（手动版）的存储器 PROGRAM 中，烧写进去如下所示的机器语言程序文 本（烧写 ROM 方法参见 2.4 存储器实验）：

ORG 0000H

DB 00100000B; JMP1, 06H

DB 00000110B

DB 11101010B; HLT

DB 00001010B;

DB 00000000B

DB 00000000B

DB 00000010B; NOP/[ADDR]

DB 11100001B; HLT

DB 01000000B; JMP2, 06H->02H

DB 00000110B

END

● 在 图 2-17 所示 的 数据 通路 （ 手动版 ）上，依 据指令逻 辑流 程图 2-18， 手动 拨码开关 执行上述程序，观察每一步手工操作的结果（寄存器 AR、IR、PC 及总线 BUS 信息）。

**实验 2：基于 JMP1 和 JMP2 指令的汇编程序**

● 根 据表 2-14 微指 令 代码表， 将 下列 微程 序分别 烧 写入 2-20 所 示 微程序 控制 器的控制 存储器 EPROM1、EPROM2 及 EPROM3（切记勿写错存储器！）。

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

● 启动仿真图 2-19 所示的微程序控制器实验电路，首先 RESET=1，然后手动 CLK 一次， 初 始 化 74LS194 节 拍 发 生 器 ｛ T1,T2,T3,T4 ｝={1,0,0,0} ， 最 后 RESET=0， 恢 复 正 常 节拍 时 序。开始手动 CLK 单步执行上述实验 1 的机器语言程序。观察每一次单步执行的结果，例如 寄 存 器 AR 、 IR 、 PC 及 总线 BUS 信 息。（注 意 程 序进 入 HLT 指 令“断 点 ” 后 需 要重 复 上 述 初始化操作才能跳出“断点”）

● 依 照 上述 启 动 初 始化方法 ， 在 RESET=0 恢 复 正 常 运 行 后， 转 为 AUTO\_CLK 方 波 信 号源自动 CLK。当程序通路陷入 “断点” 时刻（HLT 指令），查看 寄存器 AR 、IR、 PC 、总 线 BUS 信息及微指令周期数指示。

**实验 3：增加新指令 JMP3**

● 增加一条间接寻址的跳转指令 JMP3，如下所示。请补充指令逻辑流程图 2-18 及微指 令代码表 2-14（新增的微指令地址定为 00011 ），以实现 JMP3 指令。

|  |  |  |  |
| --- | --- | --- | --- |
| 汇编助记符 | OP 码 | 机器语言程序 | 指令说明 |
| JMP3,[[addr]] | 01100000 | 01100000; JMP3  xxxxxxxx;[[addr]] | 间 接 寻址：程序最终跳转到 地 址 zzH ，  PC->zzH, zzH=[yyH], yyH=[xxH] |

● 在 图 2-19 所 示 微程序控 制 器实 验电 路的 存储 器 PROGRAM 中 ，烧 写 进去 如下所 示 的 机器语言程序文本（烧写 ROM 方法参见 2.4 存储器实验）：

ORG 0000H

DB 00100000B; JMP1, 06H

DB 00000110B

DB 11101010B; HLT

DB 00001010B; NOP/[ADDR]

DB 01100000B; JMP3, 0BH->03H->0AH

DB 00001011B

DB 00000010B; NOP/[ADDR]

DB 11100001B ; HLT

DB 01000000B; JMP2, 06H->02H

DB 00000110B

DB 11100000B; HLT

DB 00000011B; NOP/[ADDR]

END

● 根据 实验 2 所述 方法，分别手动 单步执行或 AUTO\_CLK 自动执行实 验 3 新的 机器语 言 程序文本，观察单步执行或“断点”时刻的寄存器 AR、IR、PC 及总线 BUS 信息。

**六、思考题**

1. 如图 2-19 所示的 微程序控制器实验电路中，如果采用微命令来实现 HLT 指令（例 如安排控制存储器中一位作为 HLT 信号）是否可行？请修改电路试验。
2. 在实验 2 和实验 3 的机器语言文本中，有部分地址标示 “NOP/[ADDR]”，为何相同 位 置会有不 同的执行 效果 ？在什么情 况下 执 行到该处 是 NOP 指令 （顺序 滑 过）？ 什 么情 况 下执行到该处是打入 PC 的地址（跳转）？
3. 本实验的控制存储器 CM 最多可容纳多少条微指令？而指令体系最多可容纳多少条 机器指令？微指令容量和指令容量各自是由于什么因素限定的？
4. 假 设 只 用 两 个 2764 芯 片 作为控 制存储 器 ， 节拍 脉 冲 精 简 到三个 脉冲（T1-T3 ），请 问微程序怎么改？机器语言程序需要改么？请修改电路试验。

**2.6 硬布线控制器实验**

**一、实验目的**

1. 掌握硬布线控制器的组成原理及设计方法。
2. 了解单周期和多周期硬布线控制器的各自特点。

**二、实验内容**

分 别设计 单周 期和 多 周 期硬 布线版 本 的 CPU 电路， 两 者 在功 能上 皆完全 兼 容前述 微 程 序 版 本 的 CPU ： 数据通 路 相 同 ， 指令体 系 相 同 ， 仅用硬布 线 逻 辑取代微 程 序 控制 器来产 生 各时序阶段的微操作信号。

**三、实验器件**

1. 存储器 2764 和计数器（74LS163、74LS192）。
2. 寄存器（74LS273）及移位寄存器(74LS194)。

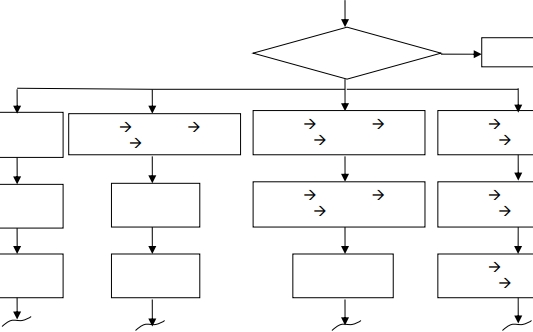
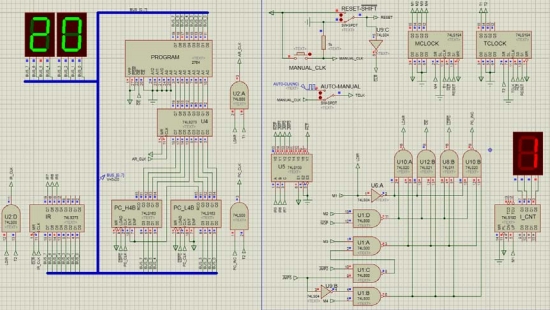
**四、实验原理**

硬布线控制器的数据通路和机器指令集都与微程序控制器版本保持一致，不同之处是实 现各指令功能所需的微操作信号不是由微指令的方式产生，而是由微操作信号出现的逻辑条 件（机器指令）及相应时序的逻辑组合形成。如表 2‐15 所示。本实验共定义了空指令 NOP、 停 机 指令 HLT 和 三 条不 同 类 型 的 跳 转 指 令 JMPx ：相对 寻 址 的 JMP1 指 令 、直接 寻 址的 JMP2 指令和间接寻址的 JMP3 指令。

表 2-15 硬布线控制器实验的指令表

|  |  |  |  |
| --- | --- | --- | --- |
| 汇编助记符 | [OP 码]  I7 I6 I5 | 机器语言程序 | 指令说明 |
| JMP1,addr | 0 0 1 | 00100000; JMP1  xxxxxxxx; addr | 相对寻址：程序跳转到地址 xxH 执行 PC->xxH |
| JMP2,[addr] | 0 1 0 | 01000000; JMP2  xxxxxxxx;[addr] | 直接寻址：程序跳转到地址 yyH 执行 PC->yyH，yyH=[xxH] |
| JMP3,[[addr]] | 0 1 1 | 01100000; JMP3  xxxxxxxx;[[addr]] | 间接寻址：程序跳转到地址 zzH 执行 PC->zzH, zzH=[yyH], yyH=[xxH] |
| HLT | 1 1 1 | 11100000; HLT | 硬件停机 |
| NOP | 0 0 0 | 00000000; NOP | “空”指令 ，不执行任何操作 |

硬布线控制器有两种设计方法：单周期和多周期硬布线控制器。单周期硬布线控制器的 定义是所有指令均在一个等长的时钟周期内完成，这个时钟周期是由执行时间最长的指令决 定的。 在微程序控制器实验的指令逻辑流程图 2-18 中的方框代表了一个 CPU 周期，又代表 了该周期内执行的一条微指令。在硬布线控制器实验中，虽然没有微指令结构及微程序编程， 但是指令的逻辑流程是保持一致的，图 2-18 中的一个方框依旧表示一个 CPU 周期，同时代 表了硬布线逻辑在此 CPU 周期的状态。因为微程序控制器实验中，JMP3 指令执行时间最长 （ 4 个 CPU 周 期 ） ，所 以 在 单周期硬 布 线 控制 器 实验 中 ，除了 HLT 指 令在取 指 周 期 M1 末 尾 硬 件 停 机 。 其 他 指令都 必 须 经 历四 个 CPU 周 期 M1-M4 。 所 有 指 令 的状 态 机 如 图 2-24 所



T1: PC AR,ROM BUS

T2: BUS IR, PC+1

T1: PC AR,ROM BUS

T2: BUS PC

T1: PC AR,ROM BUS

T2: BUS PC

T1: PC AR,ROM BUS

T2: BUS PC

T1: PC AR,ROM BUS

T2: BUS PC

T1: PC AR,ROM BUS

T2: BUS PC

T1: PC AR,ROM BUS

T2: BUS PC

硬件停机

T1:

T2:

T1:

T2:

T1:

T2:

T1:

T2:

T1:

T2:

T1:

T2:

示，与前述微控制器版本的图 2-18 非常类似，方框右上方是当前 CPU 周期标识 Mx。M1 是 取指周期，M2、M3 和 M4 是执行周期。

【M1】

OP 码{I

7

I6

I5

}

HLT

NOP

【M2】

JMP1

【M2】

JMP2

【M2】

JMP3

【M2】

【M3】

【M4】

【M3】

【M4】

【M3】

【M4】

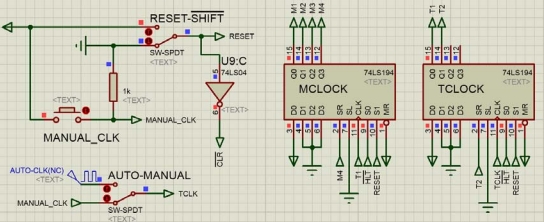
【M3】

【M4】

图 2-24 单周期硬布线控制器的指令状态机

图 2-25 单周期硬布线控制器

单 周 期 硬布线 控 制器实验 的 电路 图 如 上图 2‐25 所 示 ， 由左边的 数据通路 和 右 边 的时序 电路及硬布线控制逻辑组成，左边的数据通路部分与微程序版本完全相同。而右边的时序电 路如图 2‐26 所示，有两级时序发生器：节拍 T1‐T2 和 CPU 周期 M1‐M4。基准时钟信号 TCLK



驱 动 第 一 级 移 位 寄 存 器 TCLOCK ， 循 环 发 送 节 拍 T1->T2->T1->T2->T1……, 每 一 次 {T1,T2} 时 序 循环 即 为一个 CPU 周期 M； 而节拍 T1 驱 动第 二级 移位 寄 存器 MCLOCK， 循环发 送 4 个 等长 的 CPU 周期 M1->M4 ，每 一 次 循环 即 为一个 机 器 指 令 周期 。 所 以， 单 周期硬 布 线 控 制 器可 以 周 而 复 始的 产 生 时 序：M1{T1,T2}, M2{T1,T2}, M3{T1,T2}, M4{T1,T2}, M1{T1,T2},…… 。 此外 ， 下 图 2‐26 中 的 基 准 时 钟 电 路 ， RESET 复 位 电 路 ， 以 及 用 以 系 统 初 始 化 和 跳 出 HLT 指 令 “ 断 点”的“重启”操作步骤均与微程序版电路保持一致。

图 2-26 单周期硬布线控制器的时序电路

硬布线控制器的设计思想是：控制器发出的微操作信号是时间因素和空间因素的函数。

各条机 器指令的 OP 码｛I7

I6I5

｝经过 译码产生 的不同指令 信号 构成 了空间特征 ，而 CPU 周期

时序信号 Mx 构成了时间特征，硬布线控制器就是把输入的时间特征和空间特征逻辑组合， 输出具有定时特点的微控制信号。而微操作信号的组合逻辑如下表 2‐16 所示：

表 2-16 微操作信号的组合逻辑

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | M1 | M2 | M3 | M4 |
| LDIR | NOP/HLT/ JMP1/JMP2/JMP3 |  |  |  |
| LDAR | NOP/HLT/ JMP1/JMP2/JMP3 | JMP1/JMP2/JMP3 | JMP2/JMP3 | JMP3 |
| OE\ | NOP/HLT/ JMP1/JMP2/JMP3 | JMP1/JMP2/JMP3 | JMP2/JMP3 | JMP3 |
| LDPC\ |  | JMP1/JMP2/JMP3 | JMP2/JMP3 | JMP3 |
| PC\_INC | NOP/HLT/ JMP1/JMP2/JMP3 | JMP1/JMP2/JMP3 | JMP2/JMP3 | JMP3 |

根据上表 2-16，可以设计出如图 2-27 中间所示的由指令信号和时序信号 Mx 构成的微 操作信号的组合逻辑，其逻辑表达式如下（注：HLT 指令只有 M1 周期）：

LDIR=M1 （注：因为所有指令在 M1 都用到 LDIR，所以简化为 M1）

LDAR=M1+NOP·M2+(JMP2+JMP3)·M3+JMP3·M4（注：NOP=JMP1+JMP2+JMP3） OE= M1+NOP ·M2+(JMP2+JMP3)·M3+JMP3·M4

LDPC= NOP·M2+(JMP2+JMP3)·M3+JMP3·M4

PC\_INC= M1+NOP·M2+(JMP2+JMP3)·M3+JMP3·M4

上述时序信号 Mx 由图 2-26 的时序发生器 MCLK 提供，而每条机器指令唯一对应的指

令信号则从指令寄存器 IR 输出的 OP 码｛I7

I6

I5

｝经过下图 2-27 左边的 3-8 译码器 74LS138

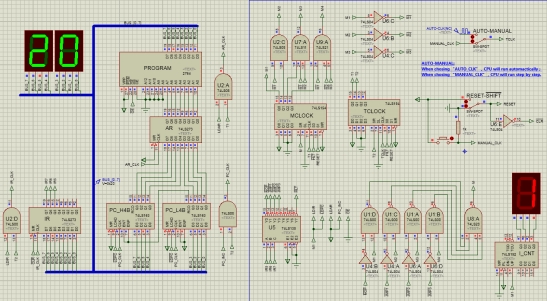
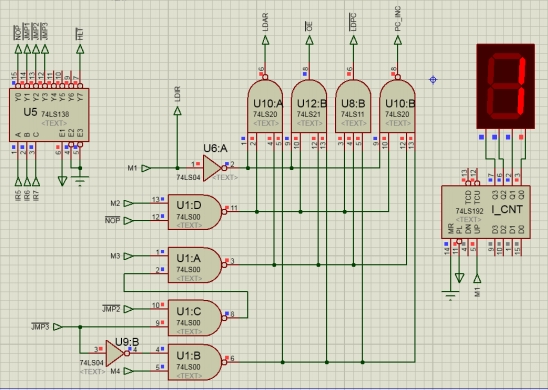
转换成。值得注意的是，当指令寄存器 IR 的 OP 码 I

7I

6I

5=111 的时候，3-8 译码器产生的指令

信号 HLT\把上图 2-26 中的两级时序发生器 MCLK 和 TCLK 的工作模式改为保持，即整个 系统停机在 M1 取指周期的 T2 节拍上。



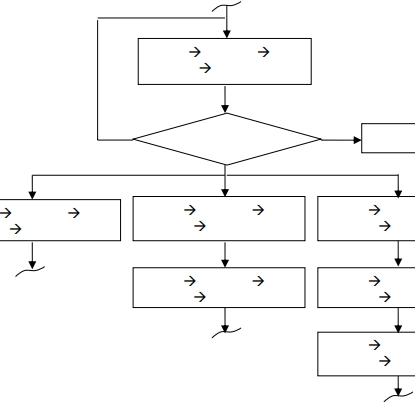
此外，为了便于观测程序的运行，本实验提供了指令显示功能：下图 2-27 右边是一个 由 M1 上升沿驱动的升序计数器 74LS192，通过数码管显示当前运行是第几条指令。

图 2-27 单周期硬布线控制器的硬布线控制逻辑

单周期硬布线控制器虽然直观简单，但是存在运行效率不高的问题： 所有指令的周期 都是一致的，由执行时间最长的指令 JMP3 决定。因此，如图 2-24 所示，对于其他执行时 间较短的指令来说，存在着一个或多个空闲的 CPU 周期，造成了 CPU 性能的闲置浪费。

图 2-28 多周期硬布线控制器

硬布线控制器的另一种较为普遍的设计方法是多周期硬布线控制器，其定义是不同指令



T1: PC AR,ROM BUS

T2: BUS IR, PC+1

T1: PC AR,ROM BUS

T2: BUS PC

T1: PC AR,ROM BUS

T2: BUS PC

T1: PC AR,ROM BUS

T2: BUS PC

T1: PC AR,ROM BUS

T2: BUS PC

T1: PC AR,ROM BUS

T2: BUS PC

T1: PC AR,ROM BUS

T2: BUS PC

硬件停机

依据各自的功能需要，可以有不同数目的 CPU 周期。多周期硬布线控制器的实验电路图如 上图 2-28 所示，左边的数据通路部分依旧与微程序版本保持一致，而右边的时序电路和硬 布线逻辑则实现了如下图 2-29 所示的多周期硬布线版本的指令状态机。与单周期硬布线版 本的指令状态机图 2-24 比较，多周期硬布线版本的指令的 CPU 周期数都是按照指令功能所 需定制，没有空闲的 CPU 周期，因此系统的运行效率大大提高。

【M1】

NOP

OP {I7I

6I

5I

4}

HLT

JMP1

【M2】

JMP2

【M2】

JMP3

【M2】

【M3】

图 2-29 多周期硬布线控制器的状态机描述

【M3】

【M4】

如下图 2-30 所示，时序电路依旧有两级时序：节拍 T1-T2 和 CPU 周期 M1-M4。基准 时钟信号 TCLK 驱动第一级移位寄存器 TCLOCK，循环发送节拍 T1->T2->T1->T2->T1……, 每一次{ T1,T2}时序循环即为一个 CPU 周期，与单周期版本保持一致。节拍 T1 同样驱动第 二级移位寄存器 MCLOCK，但是 MCLOCK 的循环周期不是统一的 4 个 CPU 周期了，而是由移位

寄存器的右移串行输入端 S

R

的 M 值决定。根据上图 2-29 的指令状态机，可以推出如下表 2-

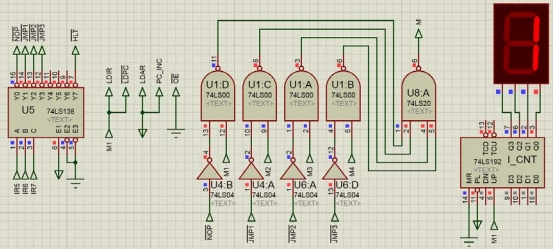
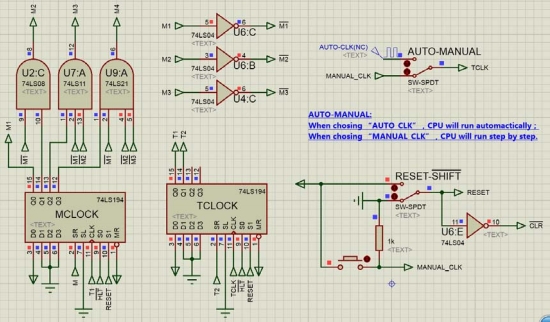
17 所示的各个指令所需的 CPU 周期：

表 2-17 多周期硬布线控制器的指令周期组成

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 指令 | NOP/HLT/ | JMP1 | JMP2 | JMP3 |
| CPU 周期 | M1 | M1+ M2 | M1+ M2+M3 | M1+ M2+M3+M4 |

根据上表可以推导出 M 的逻辑表达式如下（具体电路如图 2-31 中间的组合逻辑所示）： M = NOP· M1+ JMP1·M2+JMP2·M3+JMP3·M4

因为从指令寄存器 IR 输出的 OP 码｛I7I6I5｝经过下图 2-31 左边的 3-8 译码器 74LS138， 只可能输出唯一的指令信号，所以任何情况下上述表达式中只有一个分项是有效的。假设当 前执行的是指令 JMP1，则根据上式可知 M=M2，MCLOCK 输出的 Q0Q1Q2Q3 时序：1000、 0100、1010、0101、1000…为了使得 MCLOCK 状态机保持“M1 M2 M1 ….”，下图 2- 30 中 MCLOCK 的 Q1 Q2Q3 端外接了限制项，只有前述移位 Qx-i 都为 0 的状态下，Qx=1 才成立；



否则，Qx 清零。因此，在 M=M2 的条件下，M1-M4 的时序最终为：1000、0100、1000、 0100、1000…….即 MCLOCK 状态机保持为“M1 M2 M1 ….”状态。

图 2-30 多周期硬布线控制器的时序电路

此外，上图 2-30 中的基准时钟电路、RESET 复位电路，以及用以初始化和跳出 HLT 指 令“断点”的“重启”操作步骤都与单周期硬布线版本完全相同。下图 2-31 中的指令译码 器 74LS138 电路和指令显示电路亦与单周期硬布线版本保持一致。指令信号 HLT\的出现同 样把图 2-30 中的两级时序发生器 MCLK 和 TCLK 的工作模式改为保持，让系统停机在 M1 取指周期的 T2 节拍上。根据表 2-16 和上表 2-17，可以将数据通路的微操作信号的组合逻 辑简化如下（电路如下图 2-31 中指令译码器 74LS138 右边所示）：

LDIR = M1

LDPC = M1

LDAR = OE = PC\_INC = 1

图 2-31 多周期硬布线控制器的微操作信号硬布线逻辑

**五、实验步骤**

●在图 2-25 和图 2-28 所示的硬布线控制器电路的存储器 PROGRAM 中，烧写进去如下 所示的机器语言程序文本（烧写 ROM 方法参见 2.4 存储器实验）：

ORG 0000H

DB 00100000B; JMP1, 06H

DB 00000110B

DB 11101010B; HLT

DB 00001010B; NOP/[ADDR]

DB 01100000B; JMP3, 0BH->03H->0AH

DB 00001011B

DB 00000010B; NOP/[ADDR]

DB 11100001B ; HLT

DB 01000000B; JMP2, 06H->02H

DB 00000110B

DB 11100000B; HLT

DB 00000011B; NOP/[ADDR]

END

● 分 别 启 动 仿 真 图 2-25 所 示 的 单 周 期 硬 布 线 控 制 器 和 图 2-28 所 示 的 多 周 期 硬 布 线 控 制 器 ： 首 先 RESET=1 ， 然 后 手 动 CLK 一 次 ，初 始 化 74LS194 节拍 发 生 器 ｛ T1,T2,T3,T4 ｝ ={1,0,0,0}，最后 RESET=0，恢复正常节拍时序。开始手动 CLK 单步执行上述实验 1 的机器 语言程序。观察每一次单步执行的结果，例如寄存器 AR 、IR、PC 及总线 BUS 信息。（注意 程序进入 HLT 指令“断点”后需要重复上述初始化操作跳出“断点”）

● 依 照 上述 启 动 初 始化方法 ， 在 RESET=0 恢 复 正 常 运 行 后， 转 为 AUTO\_CLK 方 波 信 号 源自 动 CLK 运 行单周期和多周 期硬布线控制 器。当程序通路 陷入“断 点” 时 刻（HLT 指 令），查看寄存器 AR、IR、PC、总线 BUS 信息及微指令周期数指示。

**六、思考题**

1. 在上述机器语言文本中，有部分地址标示“NOP/[ADDR]”，为何相同位置会有不同 的 执行效果 ？在什么 情况 下执行到该 处是 NOP 指 令（顺 序滑过 ）？ 什么情 况下 执行到该处 是打入 PC 的地址（跳转）？
2. 假 设上 述机器语言 文 本的 倒 数第 二行不是 HLT 指 令，而 是 修改为 NOP 指 令。则系 统启动仿真时会出现什么情况？系统会跑飞么？当系统运行到超过机器语言文本最后一行代 码后会出现什么情况？会自动停机么？是什么原因导致自动停机？
3. 假设 只有一 级时序 发生器 MCLOCK ， 节拍脉冲 T 简 化到只有一个 CLK ， 请问硬布 线电路怎么改？数据通路需要修改么？机器语言程序需要改么？请修改电路试验。