西安交通大學

XI'AN JIAOTONG UNIVERSITY

计算机组成与结构
实验报告

班级：软件 42
姓名：陈铮
学号：2141601026

2017 年 1 月 4 日

目录

插图

表格

代码索引

概览

- 实验地点

  西安交通大学软件学院机房组成实验室

- 实验教材

  现代计算机组成原理实验讲义

- 实验环境

  Quartus II 9.0

- 实验设备

  GW48-CP+ 型现代计算机组成原理实验开发系统（杭州康芯电子有限公司）

  Cyclone EP1C6Q240C8

- 致谢

  - 李晨老师

    作为指导老师，提供了大量参考资料可供学习。

  - 柳亚辉、张柳

    同小组实验成员，一起做实验，特别是绘制了本报告中大多数的图。

- 开源

  本报告及代码根据 CC-BY-NC 协议开源。

  公开于 GitHub: https://github.com/zccz14/Computer-Organization-Exp-Report

  - 问题 & 勘误

    请提一个 Issue: https://github.com/zccz14/Computer-Organization-Exp-Report/issues

  - 参与贡献

    Fork & Create Pull Request

    Welcome your contributions!

第一章　节拍脉冲发生器时序电路实验

1.1　实验内容

本实验分为三个子实验：

- 连续节拍发生电路设计
- 单步节拍发生电路设计
- 单步/连续节拍发生电路设计

要求先单独设计出连续、单步节拍发生电路，然后将它们组合到一个电路里，并设计控制切换的电路。

1.2　实验原理

1.2.1　连续节拍发生电路

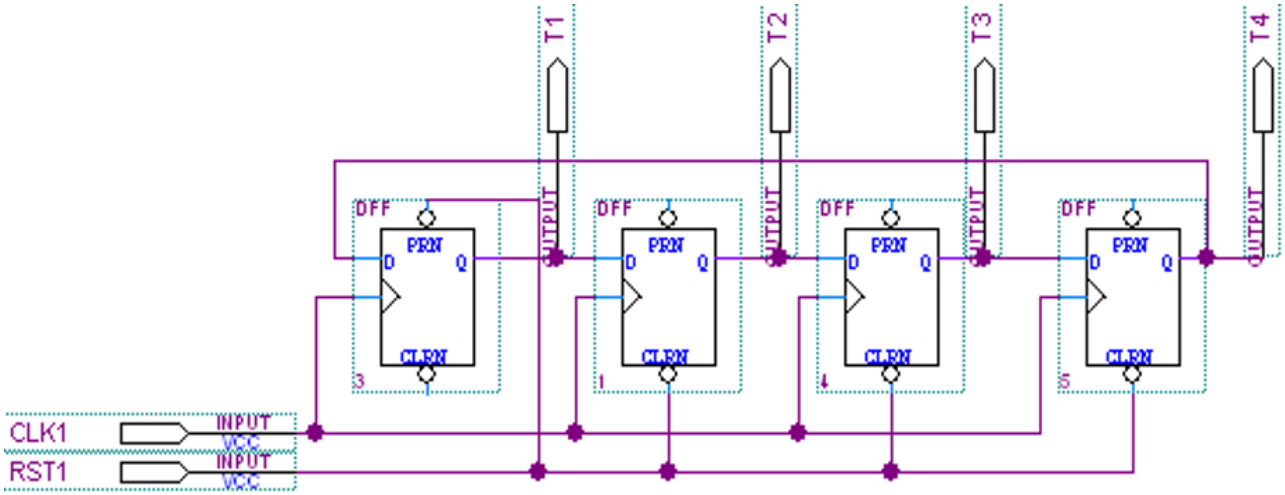


图 1.1: 连续节拍发生电路

如图1.1，由四个 D 触发器组成，可产生四个等间隔的时序信号 T1, T2, T3, T4。

输入信号

- CLK1

  时钟信号。稳定的等间隔方波信号。

- RST1

  重置信号，连续节拍发生器的开关。

  可由外部控制的输入信号。

输出信号

- T1

  第一个输出信号。

- T2

  第二个输出信号。

- T3

  第三个输出信号。

- T4

  第四个输出信号。

详细说明

1. RST1 = 0

   第一个 D 触发器的 PRN = 0 (低有效)，强制 Q = 1 因此 T1 = 1；

   其他 D 触发器的 CLRN = 0 (低有效)，强制 Q = 0 因此 T2 = T3 = T4 = 1。

2. RST1 = 1

   这个时候实际上 RST1 对电路是没有其他影响的，触发器的 PRN, CLRN 都处于无效态。

   对于任意一个 D 触发器，将在下一个 CLK1 上升沿使得 Q := D。由于四个 D 触发器的 Q, D 首尾相连，实际上它们在每个 CLK1 上升沿时，将前一个 D 触发器的输出电位“传导”到下一个 D 触发器输出电位。周而复始，形成连续节拍发生器。

   其周期为 CLK1 周期的 4 倍。

1.2.2　单步节拍发生电路

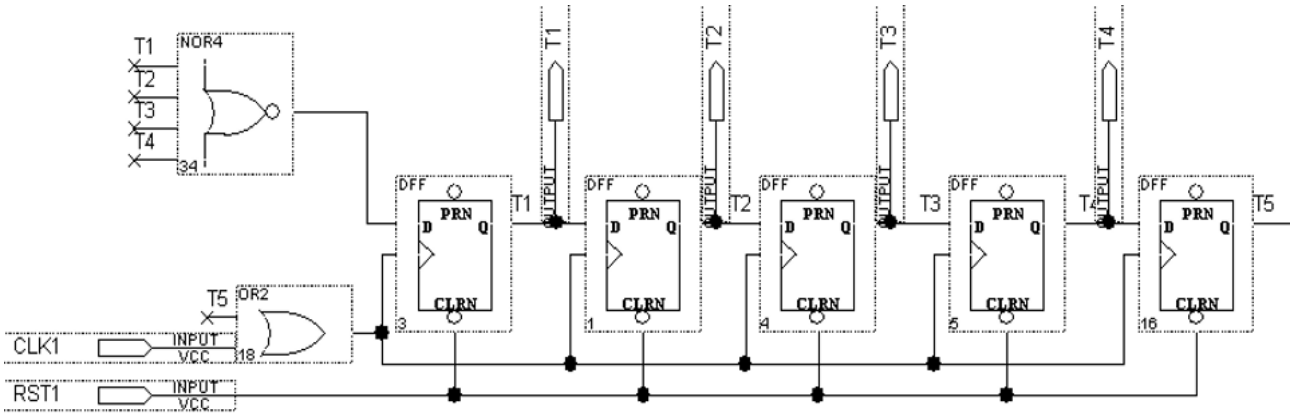


图 1.2: 单步节拍发生电路

如图1.2，由五个 D 触发器、一个 4 路或非门、一个 2 路或门组成，可产生四个等间隔的时序信号 T1, T2, T3, T4。

输入信号

- CLK1

  时钟信号。稳定的等间隔方波信号。

- RST1

  重置信号，每个上升产生一个单步信号。

  是可由外部控制的输入信号。

输出信号

- T1

  第一个输出信号。

- T2

  第二个输出信号。

- T3

  第三个输出信号。

- T4

  第四个输出信号。

详细说明

1. RST1 = 0

   所有 D 触发器 CLRN = 0 (低有效)，T1 = T2 = T3 = T4 = T5 = Q = 0。

   注意此时第一个 D 触发器的输入是 D = NOR(T1, T2, T3, T4) = NOR(0, 0, 0, 0) = 1，这相当于一个单步时序的初始化。

   T5 = 0 使得 OR(CLK1, T5) = CLK1，此时 2 路或门没有作用。

2. RST1 = 1

   所有 D 触发器的 PRN, CLRN 都处于无效态。

   在 RST1 = 1 后的第一个 CLK1 上升沿，第一个 D 触发器的 T1 = Q := D = 1，T1 = 1 时 4 路或非门将输出 0，因此第一个 D 触发器的 D 端将马上被置为 0，这样就能保证只产生一个信号。

   在第二、三、四个 CLK1 上升沿，同连续时序发生器，电位在触发器之间被“传导”，T2, T3, T4 相继输出 1，然后复位。

   在第五个 CLK1 上升沿，T5 := 1，2 路或门的输出保持为 1，屏蔽时钟信号对节拍发生器的影响。

1.2.3　单步/连续节拍发生电路

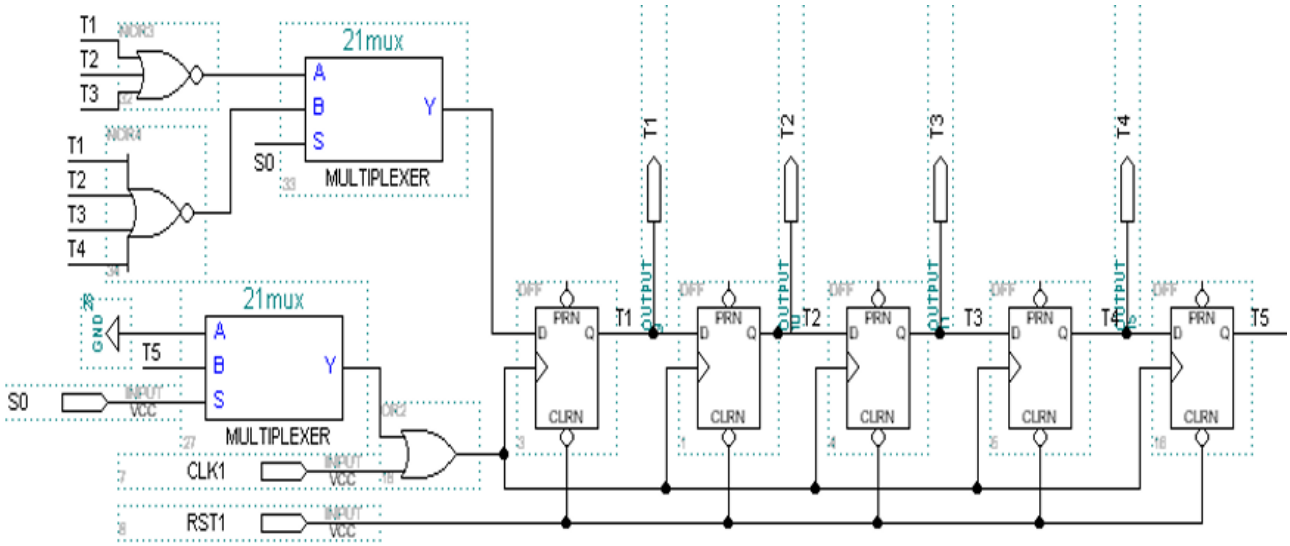


图 1.3: 单步/连续节拍发生电路

如图1.3，由五个 D 触发器、一个 4 路或非门、一个 3 路或非门、一个 2 路或门、两个 2 选 1 选择器组成，可产生四个等间隔的时序信号 T1, T2, T3, T4。

输入信号

- CLK1

  时钟信号。稳定的等间隔方波信号。

- RST1

  重置信号，每个上升产生一个单步信号。

  是可由外部控制的输入信号。

- S0

  选择信号，0：单步；1：连续。

  是可由外部控制的输入信号。

输出信号

- T1

  第一个输出信号。

- T2

  第二个输出信号。

- T3

  第三个输出信号。

- T4

  第四个输出信号。

详细说明

1. S0 = 0

   选择器 B 端导通，图 1.3 可化简为图 1.2，即单步时序发生器。

2. S0 = 1

   选择器 A 端导通，图 1.3 可化简：

   - 2 路或门的一个输入接地，相当于或门无作用，可消去。
   - T5 断路，连带第五个 D 触发器的输出断路，因此该触发器无效，可消去。

   此时类似单步时序发生器的原理，在 RST1 = 0 时由 3 路或非门给出第一个高电平信号；在 RST1 = 1 时“传导”该信号，当 T4 = 1 时，可知 T1 = T2 = T3 = 0, 3 路或非门又将产生一个新的高电平信号。周而复始，构成连续时序发生器。

1.3　实验任务与实验步骤

1.3.1　连续节拍发生电路设计

1. 按照原理图 1.1 连接电路图，然后编译。
2. 将输入输出器件绑定到对应的引脚上，然后重新编译。
   - CLK1 PIN_28 (时钟信号)
   - RST1 PIN_173 (Key 8)
   - T1 PIN_137 (发光管 1)
   - T2 PIN_138 (发光管 2)
   - T3 PIN_139 (发光管 3)
   - T4 PIN_140 (发光管 4)
3. 下载到实验设备上。
4. 调整为工作模式 1。
5. 观察实验现象。
   - Key 8
     (重) 启动连续节拍发生。
6. 绘制仿真波形图。

1.3.2　单步节拍发生电路设计

1. 按照原理图 1.2 连接电路图，然后编译。
2. 将输入输出器件绑定到对应的引脚上，然后重新编译。
   - CLK1 PIN_28 (时钟信号)
   - RST1 PIN_173 (Key 8)

- T1 PIN_137 (发光管 1)
- T2 PIN_138 (发光管 2)
- T3 PIN_139 (发光管 3)
- T4 PIN_140 (发光管 4)

3. 下载到实验设备上。
4. 调整为工作模式 1。
5. 观察实验现象。
   - Key 8
     启动下一个单步节拍发生。
6. 绘制仿真波形图。

1.3.3　单步/连续节拍发生电路设计

1. 按照原理图 1.3 连接电路图，然后编译。
2. 将输入输出器件绑定到对应的引脚上，然后重新编译。
   - CLK1 PIN_28 (时钟信号)
   - RST1 PIN_173 (Key 8)
   - S0 PIN_169 (Key 7)
   - T1 PIN_137 (发光管 1)
   - T2 PIN_138 (发光管 2)
   - T3 PIN_139 (发光管 3)
   - T4 PIN_140 (发光管 4)
3. 下载到实验设备上。
4. 调整为工作模式 1。
5. 观察实验现象。
   - Key 7
     切换连续/单步模式。
   - Key 8
     产生新的节拍。
6. 绘制仿真波形图。

1.4　实验结果分析

1.4.1　实验电路图

根据原理图 1.1 绘制实验电路图 1.4。

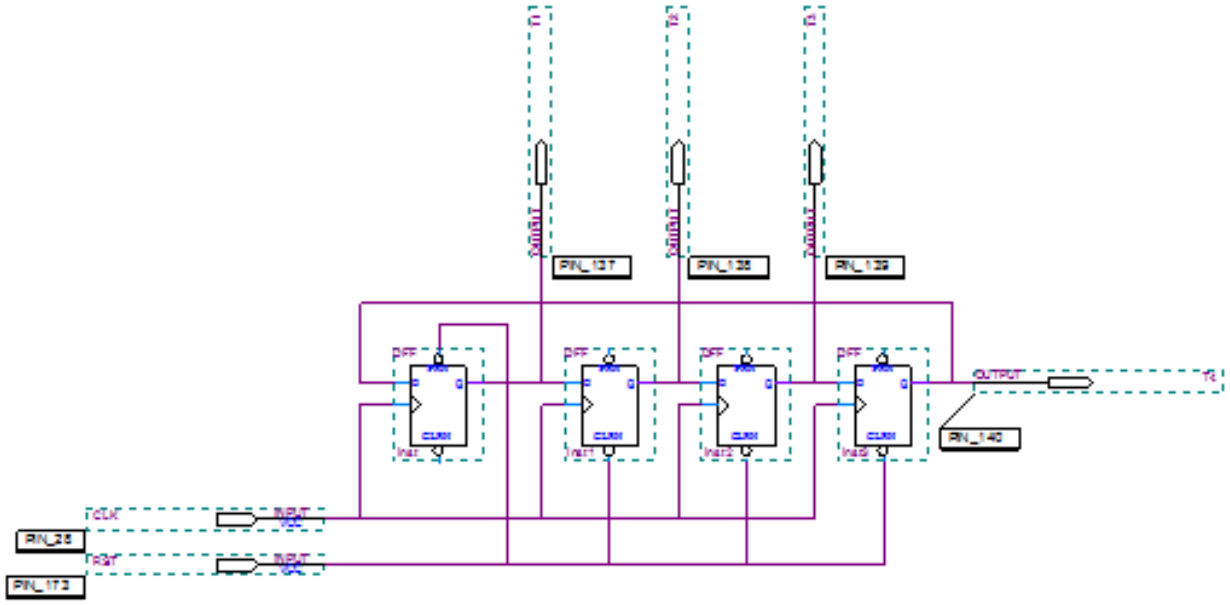


图 1.4: 连续节拍发生电路图

根据原理图 1.2 绘制实验电路图 1.5。

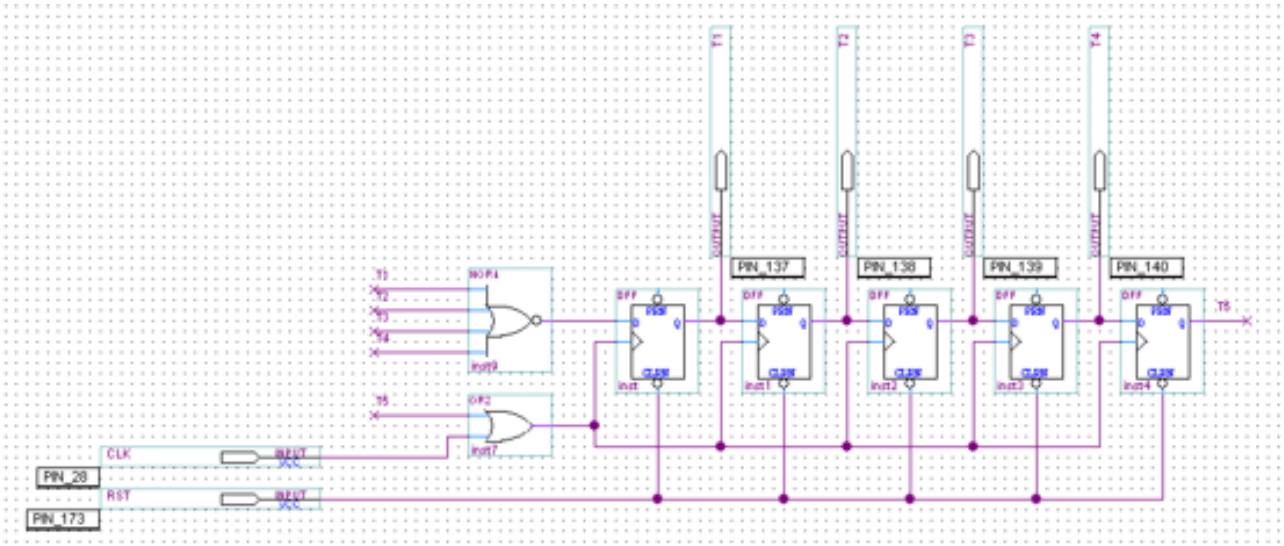


图 1.5: 单步节拍发生电路图

根据原理图 1.3 绘制实验电路图 1.6。

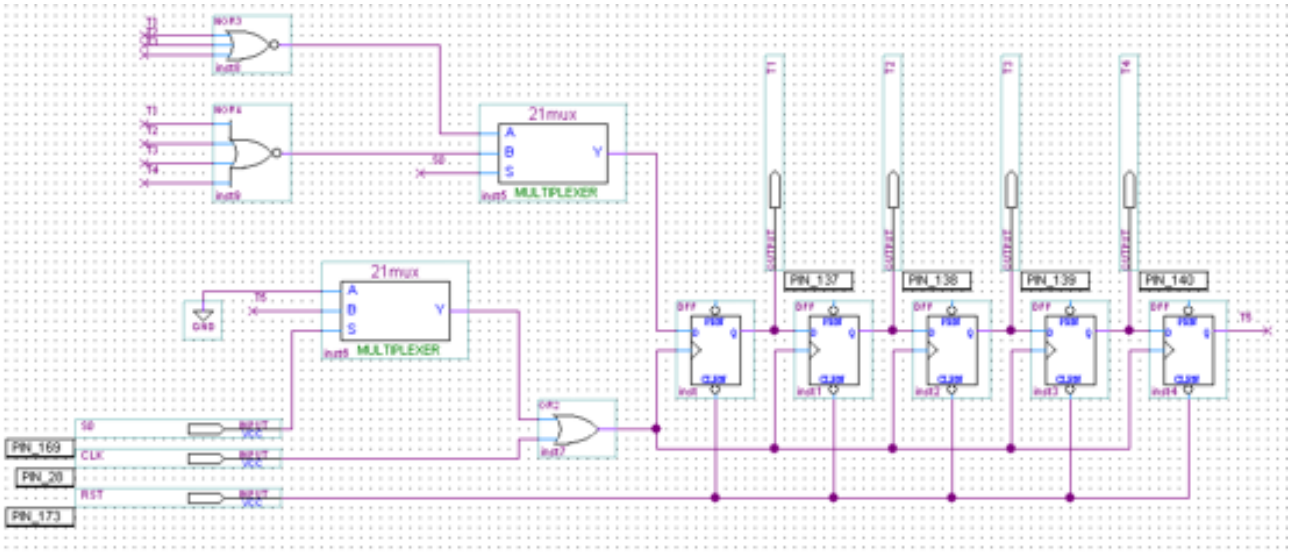


图 1.6: 连续/单步节拍发生电路图

1.4.2　仿真波形图

利用 Quartus II 产生仿真波形图 1.7。

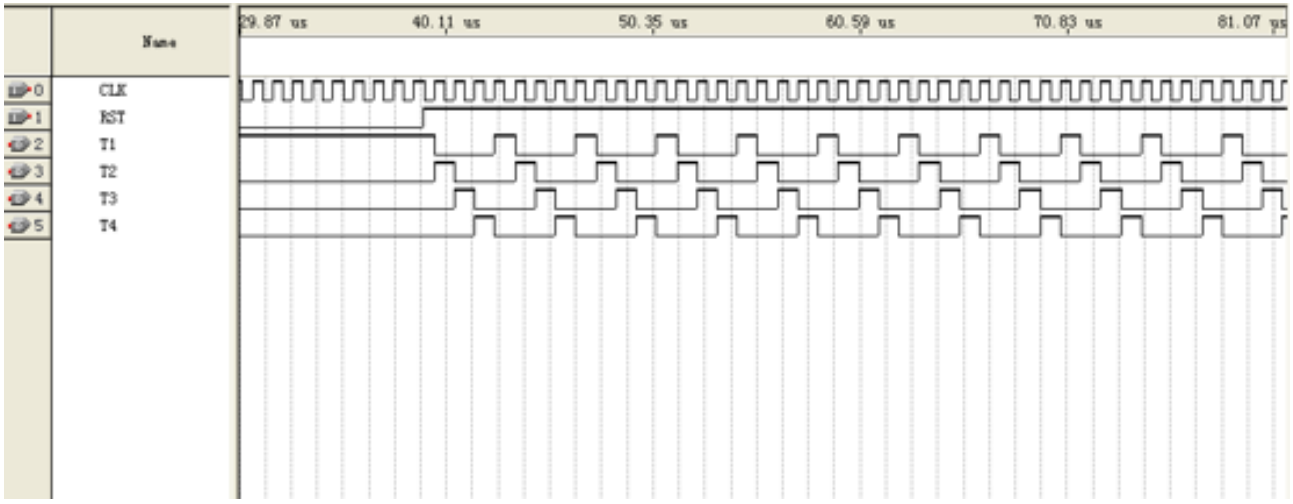


图 1.7: 连续节拍发生电路仿真波形图

利用 Quartus II 产生仿真波形图 1.8。

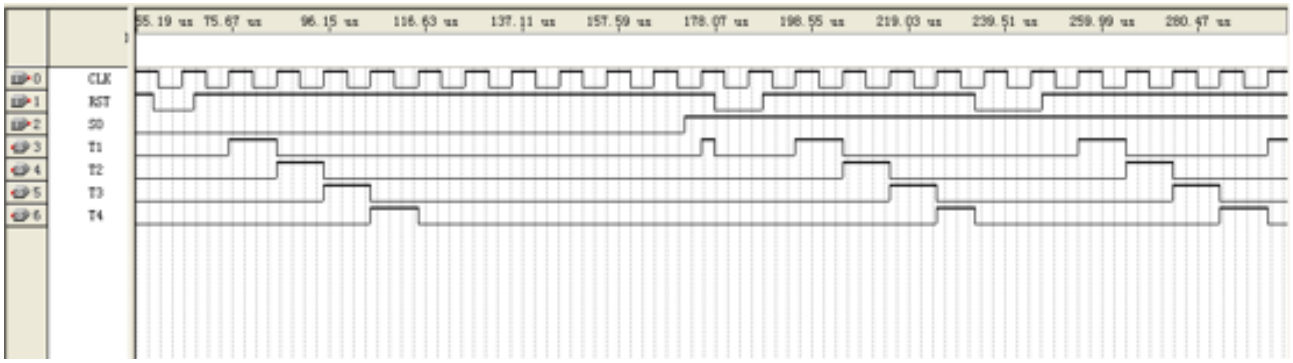


图 1.8: 单步节拍发生电路仿真波形图

利用 Quartus II 产生仿真波形图 1.9。

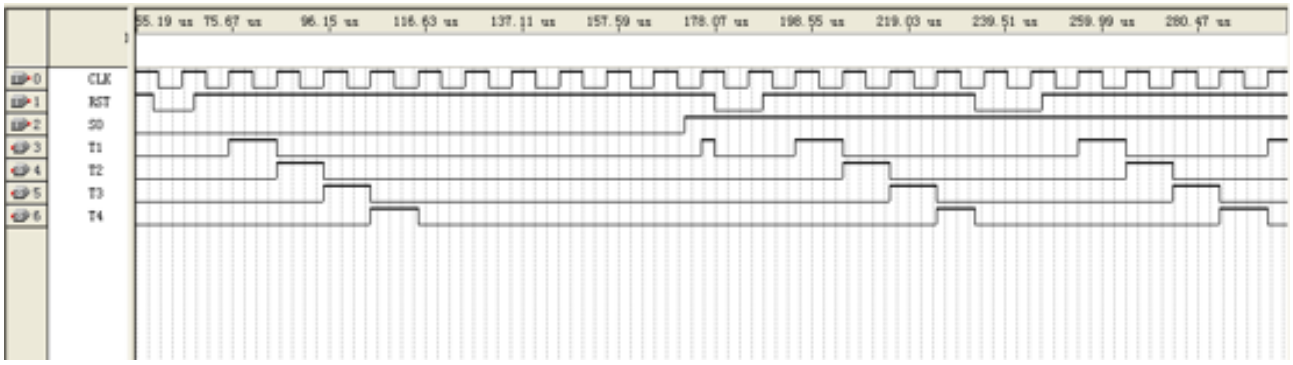


图 1.9: 连续/单步节拍发生电路仿真波形图

1.4.3　附加题

1. 为何能产生所需节拍?

   参考本章**实验原理**节的对应的**详细说明**小节。

2. 对于单步节拍发生电路，特别要对比没有 T5 输入时的仿真波形图，借此说明 T5 的作用。

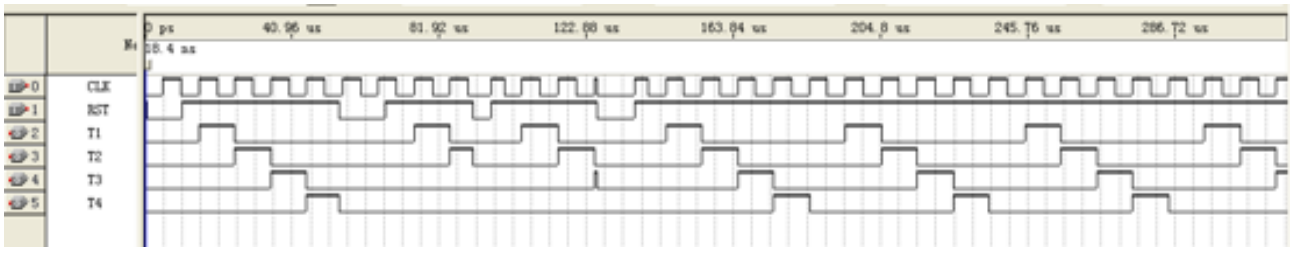


图 1.10: 没有 T5 输入时的仿真波形图

与仿真波形图 1.8 对比可以发现单步失控了。

若 2 路或门没有 T5 输入而是 GND 输入，或门失效，可消去。

当高电位“传导”至 T5 时，T1 = T2 = T3 = T4 = 0, 4 路或非门将输出 1，产生新的高电平。

这样就化为连续节拍发生器了。

由此说明，T5 输入的作用在于屏蔽时钟信号，阻止下一节拍的发生。

由此引发思考：如果直接在单步节拍发生电路的基础上，将 T5 输入改为 AND(NOT(S0), T5)，就产生了单步/连续节拍发生器。

这与原理图 1.3 的区别在于连续时序发生器的周期不同，在高电位传导至 T5 时没有输出 T5，相当于产生了一个空节拍。

重新思考原理图 1.3 还是能发现一些冗余、可以优化的地方，例如 3 路或非门可以通过将“先选择，后或非”的方式合并到 4 路或非门中。

1.4.4　思考题

1. 单步运行与连续运行有何区别，它们各自的使用环境怎样?

   答：正如字面意思，对于每次按键，单步运行只产生一个节拍，而连续运行会不断产生节拍。

   单步运行通常在**程序调试**中使用；连续运行通常在**程序运行**中使用。

2. 如何实现单步/连续运行工作方式的切换?

   答：关键在于 D 触发器的输出将会产生什么影响。

   如果每个 D 触发器的输出都能使得下一个“传导”暂停，那么就是单步模式；

   否则如果最后一个 D 触发器的输出使得第一个 D 触发器得到新的输入信号，那么就是连续模式；

   通过逻辑电路，引入新的输入（S0），实现模式的切换。

第二章　LPM_ROM & LPM_RAM 实验

2.1　实验内容

本实验分为两个子实验：

- LPM_ROM 实验
- LPM_RAM 实验

要求会初始化/读/写存储器。

2.2　实验原理

2.2.1　LPM_ROM

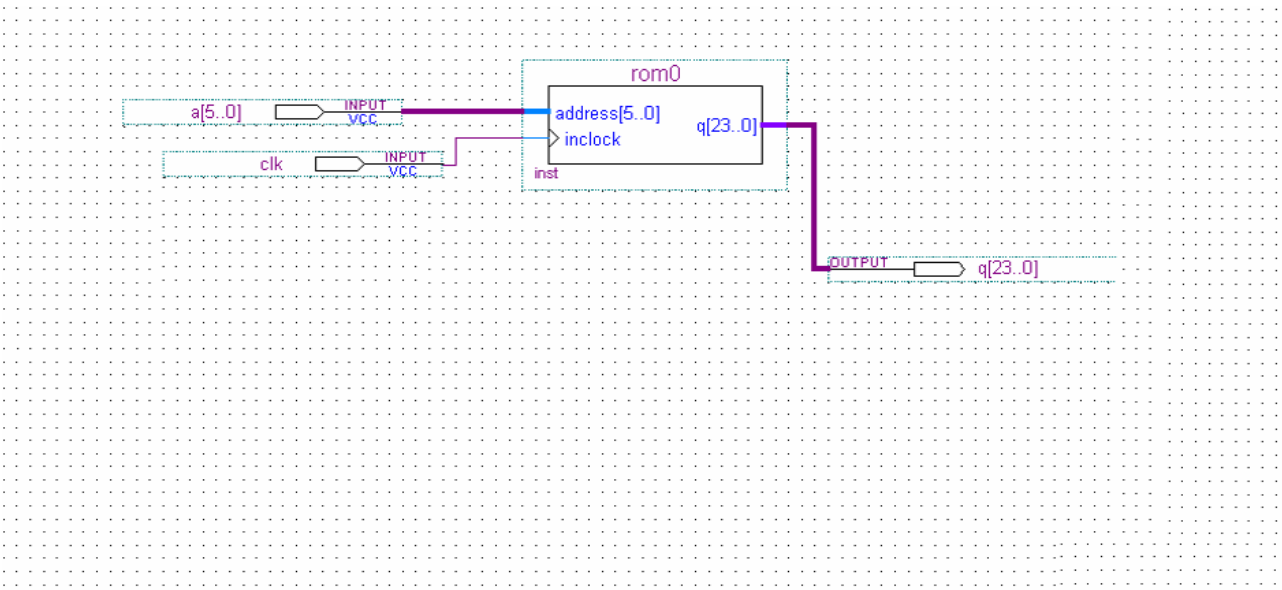


图 2.1: LPM_ROM

如图2.1，由 LPM_ROM 组成，地址线 6 位，数据线 24 位。

输入信号

- clk

  时钟信号。

  可由外部控制的输入信号。

- a[5..0]

  地址信号，6 位，用于存储器寻址。

  可由外部控制的输入信号。

输出信号

- q[23..0]

  数据信号，24 位，存储器输出的数据信号。

2.2.2 LPM_RAM

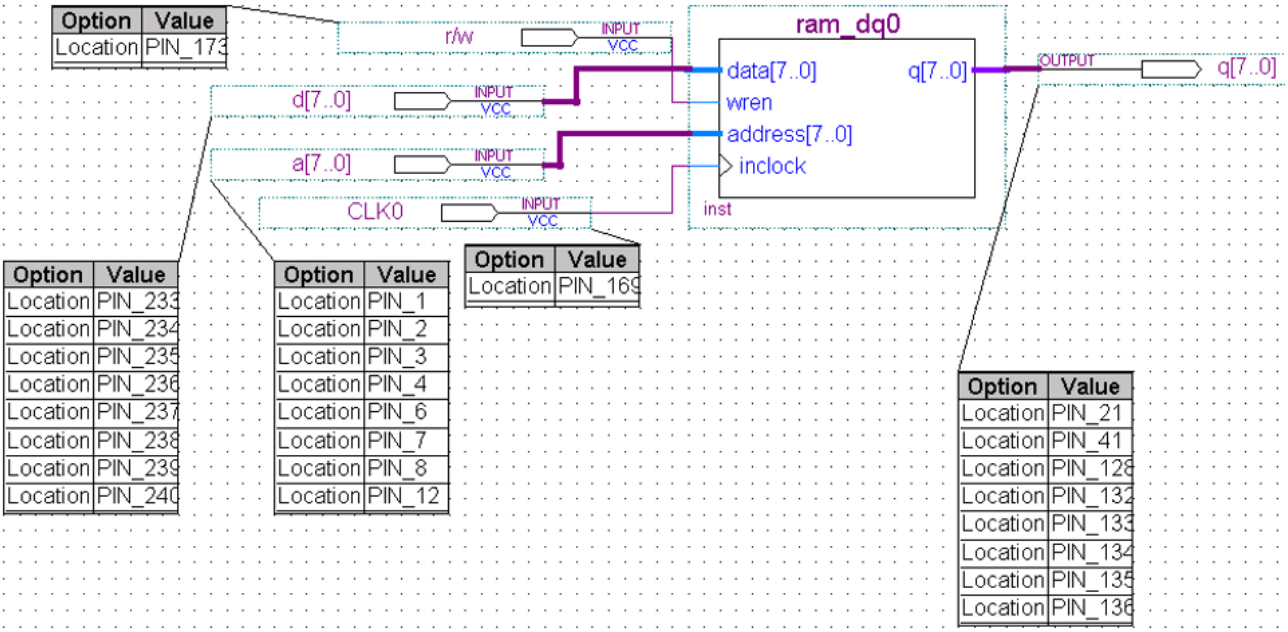


图 2.2: LPM_RAM

如图2.2，由 LPM_RAM 组成，地址线 8 位，数据线 8 位。

输入信号

- CLK0

  时钟信号。

  可由外部控制的输入信号。

- r/w

  读写控制信号，控制存储器当前是读模式还是写模式。

- a[7..0]

  地址信号，8 位，用于存储器寻址。

  可由外部控制的输入信号。

- d[7..0]

  数据信号，8 位，用于写存储器时输入数据。

  可由外部控制的输入信号。

输出信号

- q[7..0]

  数据信号，8 位，存储器输出的数据信号。

2.3　实验任务与实验步骤

2.3.1　LPM_ROM

1. 按照原理图 2.1 连接电路图，然后编译。

   (a) 配置 LPM_ROM

   - 输出数据字长：24 位。
   - 存储器可寻址单元数：64 字。
   - 禁用输出端口。
   - 创建对应大小的 ROM_5.mif 文件初始化 ROM。

ROM_5.MIF*

Addr	+0	+1	+2	+3	+4	+5	+6	+7
00	018110	01ED82	00C048	00E004	00B005	01A206	919A01	00E00D
10	001001	01ED83	01ED87	01ED8E	01ED96	038201	00E00F	00A015
20	01ED92	01ED94	00A018	018001	002017	010A01	00D181	038A11
30	010A10	000000	000000	000000	000000	000000	000000	000000
40	000000	000000	000000	000000	000000	000000	000000	000000
50	000000	000000	000000	000000	000000	000000	000000	000000
60	000000	000000	000000	000000	000000	000000	000000	000000
70	000000	000000	000000	000000	000000	000000	000000	000000

图 2.3: LPM_ROM 初始化数据

   - 启用内建的储存器内容编辑器 [1]，实例名为 rom3。

2. 将输入输出器件绑定到对应的引脚上，然后重新编译。

[1] Allow In-System Memory Content Editor to capture and update content independently of the system clock: 允许系统内建的储存器内容编辑器异步地跟踪与更新储存器内容

名称	引脚号	备注
clk	240	Key 8
a[0]	1	Key 1
a[1]	2	Key 1
a[2]	3	Key 1
a[3]	4	Key 1
a[4]	6	Key 2
a[5]	7	Key 2
q[0]	21	数码 3
q[1]	41	数码 3
q[2]	128	数码 3
q[3]	132	数码 3
q[4]	133	数码 4
q[5]	134	数码 4
q[6]	135	数码 4
q[7]	136	数码 4
q[8]	137	数码 5
q[9]	138	数码 5
q[10]	139	数码 5
q[11]	140	数码 5
q[12]	141	数码 6
q[13]	158	数码 6
q[14]	159	数码 6
q[15]	160	数码 6
q[16]	161	数码 7
q[17]	162	数码 7
q[18]	163	数码 7
q[19]	164	数码 7
q[20]	165	数码 8
q[21]	166	数码 8
q[22]	167	数码 8
q[23]	168	数码 8

表 2.1: LPM_ROM 实验引脚表

3. 下载到实验设备上。
4. 调整为工作模式 0。
5. 观察实验现象。
   - Key 1, 2 (D1 - D8)
     控制地址，低 6 位 (D1 - D6) 有效。
   - Key 8
     控制时钟。

- 数码管 3 - 8
  输出的数据，24 位全有效。

6. 绘制仿真波形图。

2.3.2 LPM_RAM

1. 按照原理图 2.2 连接电路图，然后编译。

   (a) 配置 LPM_RAM

   - 输出数据字长：8 位。
   - 存储器可寻址单元数：256 字。
   - 禁用输出端口。
   - 创建对应大小的 5_RAM.mif 文件初始化 RAM。

Addr	+0	+1	+2	+3	+4	+5	+6	+7
00	00	10	0A	20	0B	30	0B	40
08	00	00	34	00	00	00	00	00
10	00	00	00	00	00	00	00	00
18	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00
28	00	00	00	00	00	00	00	00
30	00	00	00	00	00	00	00	00
38	00	00	00	00	00	00	00	00
40	00	00	00	00	00	00	00	00
48	00	00	00	00	00	00	00	00
50	00	00	00	00	00	00	00	00
58	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	00	00
68	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00
78	00	00	00	00	00	00	00	00
80	00	00	00	00	00	00	00	00
88	00	00	00	00	00	00	00	00
90	00	00	00	00	00	00	00	00
98	00	00	00	00	00	00	00	00
a0	00	00	00	00	00	00	00	00
a8	00	00	00	00	00	00	00	00
b0	00	00	00	00	00	00	00	00
b8	00	00	00	00	00	00	00	00

图 2.4: LPM_RAM 初始化数据

   - 启用内建的储存器内容编辑器，实例名为 ram1。

2. 将输入输出器件绑定到对应的引脚上，然后重新编译。

名称	引脚号	备注
CLK0	169	Key 7
r/w	173	Key 8
a[0]	1	Key 3
a[1]	2	Key 3
a[2]	3	Key 3
a[3]	4	Key 3
a[4]	6	Key 4
a[5]	7	Key 4
a[6]	8	Key 4
a[7]	12	Key 4
d[0]	233	Key 2
d[1]	234	Key 2
d[2]	235	Key 2
d[3]	236	Key 2
d[4]	237	Key 2
d[5]	238	Key 2
d[6]	239	Key 2
d[7]	240	Key 2
q[0]	21	数码 7
q[1]	41	数码 7
q[2]	128	数码 7
q[3]	132	数码 7
q[4]	133	数码 8
q[5]	134	数码 8
q[6]	135	数码 8
q[7]	136	数码 8

表 2.2: LPM_ROM 实验引脚表

3. 下载到实验设备上。

4. 调整为工作模式 1。

5. 观察实验现象。

- Key 1, 2 (数码 1, 2)
  控制输入数据，8 位全有效。
- Key 3, 4 (数码 3, 4)
  控制地址，8 位全有效。
- Key 7 (D15)
  控制时钟。
- Key 8 (D16)
  控制 r/w 读写开关。

- 数码管 7, 8
  输出的数据，8 位全有效。

6. 绘制仿真波形图。

2.4　实验结果分析

2.4.1　实验电路图

根据原理图 2.1 绘制实验电路图 2.5。

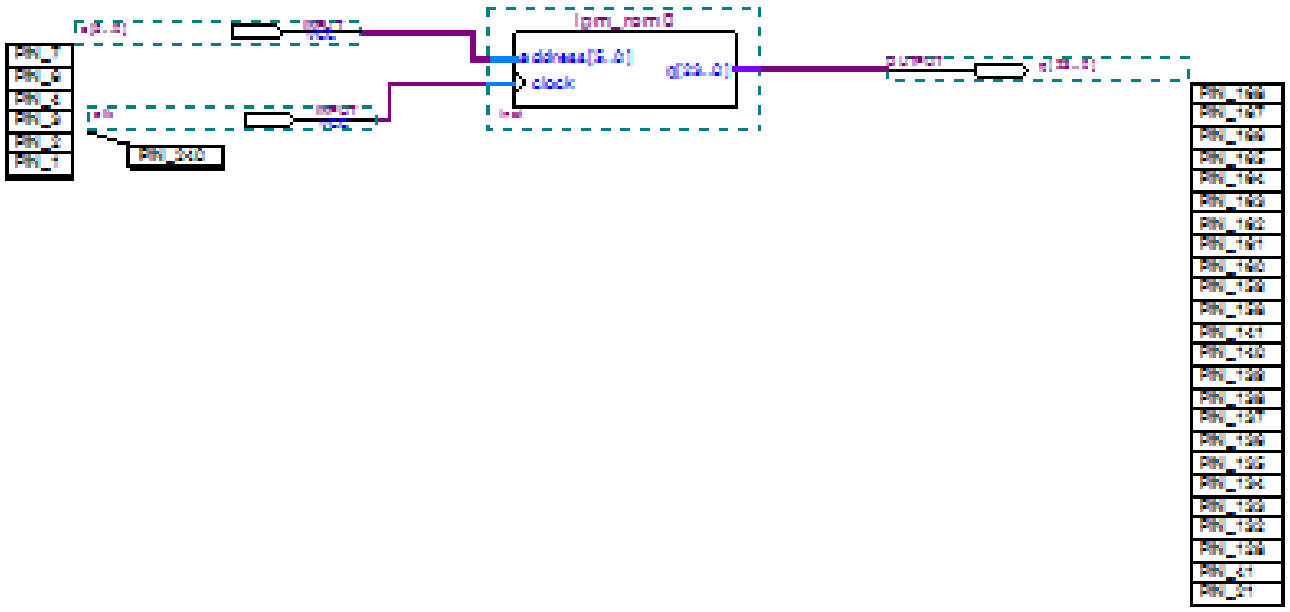


图 2.5: LPM_ROM 实验电路图

根据原理图 2.2 绘制实验电路图 2.6。

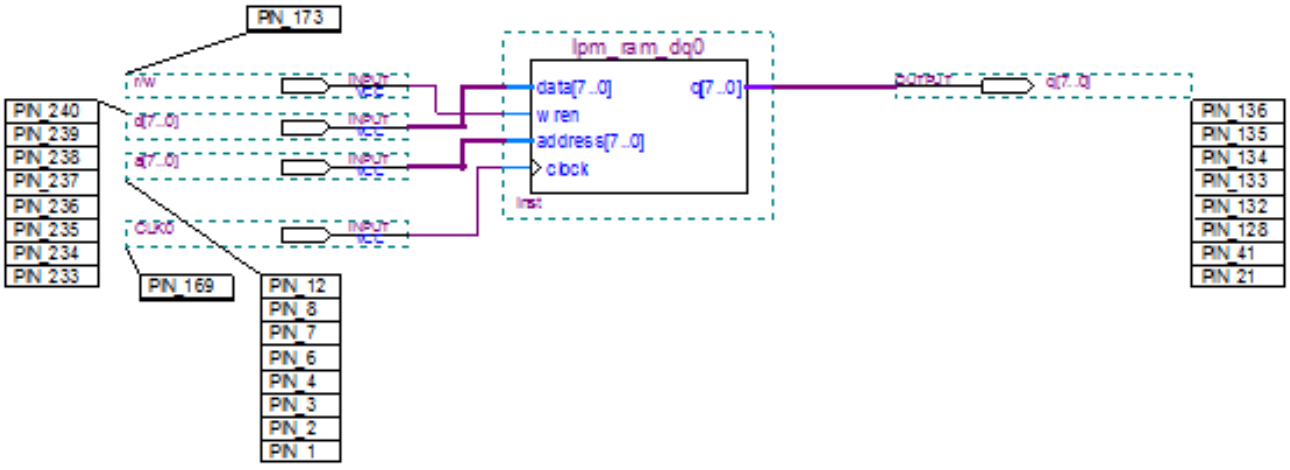


图 2.6: LPM_RAM 实验电路图

2.4.2　仿真波形图

利用 Quartus II 产生仿真波形图 2.7。

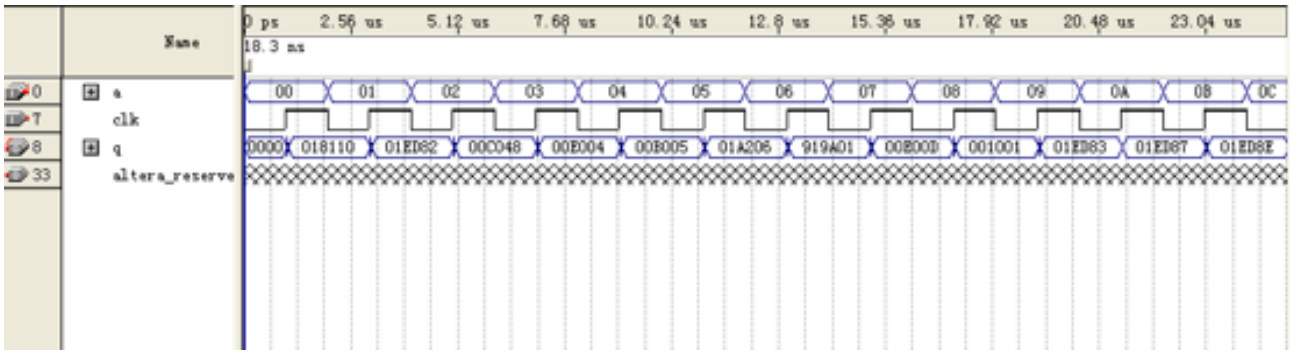


图 2.7: LPM_ROM 实验仿真波形图

利用 Quartus II 产生仿真波形图 2.8。

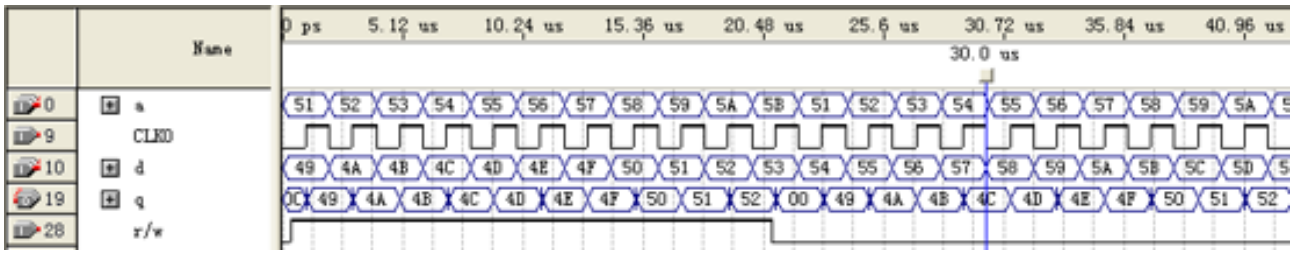


图 2.8: LPM_RAM 实验仿真波形图

2.4.3　思考题

1. 如何建立 lpm_ram_dq 的数据初始化?

   答：需要创建一个初始化文件并在储存器中配置该初始化文件。

   (a) 创建 Hexadecimal File (Intel-Format) (.MIF) 文件。

   (b) 设定大小，然后进行初始化赋值。

   (c) 在储存器的属性中配置该初始化文件。

第三章　算术逻辑运算单元 ALU 设计实验

3.1　实验内容

设计算术逻辑运算单元 ALU。

3.2　实验原理

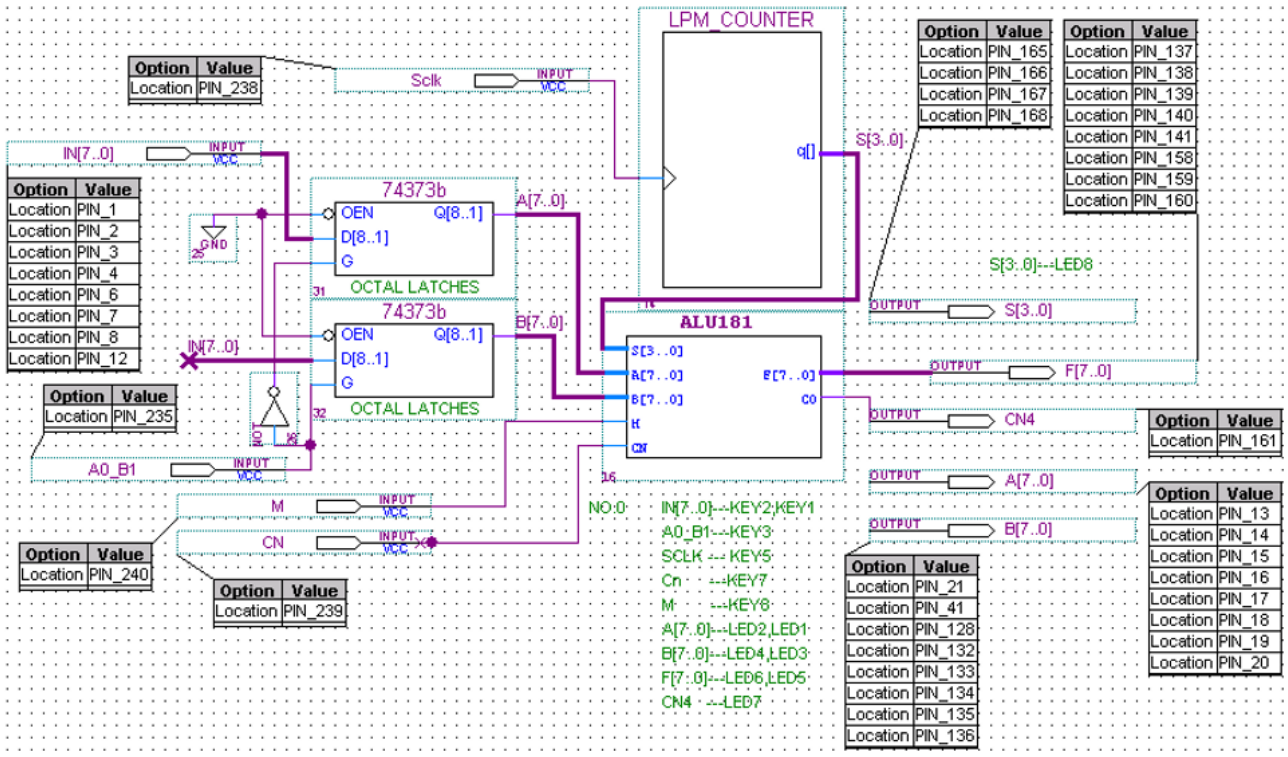


图 3.1: ALU 实验原理图

如图3.1，由 74LS181 (ALU 181), LPM_COUNTER, 两个 74383b 锁存器, 非门组成，数据线 8 位。

输入信号

- Sclk

  时钟信号。

  可由外部控制的输入信号。

- IN[7..0]

数据信号，8 位，用于输入源操作数。

可由外部控制的输入信号。

- A0_B1

  片选信号，控制源操作数进入不同的锁存器。

  可由外部控制的输入信号。

- M

  运算模式控制。0: 算术运算, 1: 逻辑运算。

  可由外部控制的输入信号。

- CN

  运算模式控制。0: 无进位, 1: 带进位。

  可由外部控制的输入信号。

输出信号

- S[3..0]

  ALU 功能码，4 位。

- F[7..0]

  ALU 输出，8 位。

- CN4

  进位标记。

- A[7..0]

  输出源操作数 A 的值，8 位。

- B[7..0]

  输出源操作数 B 的值，8 位。

3.3　实验任务与实验步骤

1. 按照原理图 3.1 连接电路图，然后编译。

   (a) 用 VDHL 语言对 74LS181 编程。

   代码 3.1: alu181.vhd

```
1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3 USE IEEE.STD_LOGIC_UNSIGNED.ALL;
4 ENTITY ALU181 IS
5 PORT ( S  : IN  STD_LOGIC_VECTOR(3 DOWNTO 0 );
6    A, B  : IN  STD_LOGIC_VECTOR(7 DOWNTO 0);
7    F  : OUT STD_LOGIC_VECTOR(7 DOWNTO 0);
8    COUT : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
```

```
    M, CN  : IN  STD_LOGIC;
    CO, FZ : OUT STD_LOGIC  );
END ALU181;
ARCHITECTURE behav OF ALU181 IS
SIGNAL A9, B9, F9 : STD_LOGIC_VECTOR(8 DOWNTO 0);
BEGIN
A9 <= '0' & A ;  B9 <= '0' & B ;
PROCESS(M,CN,A9,B9)
BEGIN
CASE S  IS
WHEN "0000"=>IF M='0' THEN F9<=A9 + CN ; ELSE  F9<=NOT A9; END IF;
WHEN "0001"=>IF M='0' THEN F9<=(A9 OR B9)+CN; ELSE  F9<=NOT(A9 OR B9); END IF;
WHEN "0010"=>IF M='0' THEN F9<=(A9 OR (NOT B9))+CN;
ELSE F9<=(NOT A9) AND B9; END IF;
WHEN "0011"=>IF M='0' THEN F9<= "000000000"-CN ; ELSE F9<="000000000"; END IF;
WHEN "0100"=>IF M='0' THEN F9<=A9+(A9 AND NOT B9)+CN ;
 ELSE F9<=NOT (A9 AND B9); END IF;
WHEN "0101"=>IF M='0' THEN F9<=(A9 OR B9)+(A9 AND NOT B9)+CN ;
 ELSE  F9<=NOT B9; END IF;
 WHEN "0110"=>IF M='0' THEN F9<=A9 -B9 - CN   ; ELSE F9<=A9 XOR B9; END IF;
WHEN "0111"=>IF M='0' THEN F9<=(A9 OR (NOT B9))-CN ;
 ELSE F9<=A9 AND (NOT B9); END IF;
WHEN "1000" =>IF M='0' THEN F9<=A9 + (A9 AND B9)+CN;
 ELSE  F9<=(NOT A9) OR B9;END IF;
WHEN "1001" =>IF M='0' THEN F9<=A9 + B9 + CN; ELSE  F9<=NOT(A9 XOR B9);
END IF;
WHEN "1010" =>IF M='0' THEN F9<=(A9 OR (NOT B9))+(A9 AND B9)+CN ;
 ELSE  F9<=B9; END IF;
WHEN "1011" =>IF M='0' THEN F9<=(A9 AND B9)- CN ; ELSE  F9<=A9 AND B9; END IF;
WHEN "1100" =>IF M='0' THEN F9<=A9 + A9 + CN; ELSE  F9<= "000000001"; END IF;
WHEN "1101" =>IF M='0' THEN F9<=(A9 OR B9)+A9 + CN ;
ELSE  F9<=A9 OR (NOT B9); END IF;
WHEN "1110" =>IF M='0' THEN F9<=(A9 OR(NOT B9))+A9+CN; ELSE F9<=A9 OR B9;END I
WHEN "1111" =>IF M='0' THEN F9<=A9-CN; ELSE  F9<=A9 ; END IF;
WHEN OTHERS  =>F9<= "000000000" ;
END CASE;
IF (A9= B9) THEN FZ <= '0';END IF;
END PROCESS;
F<= F9(7 DOWNTO 0) ;   CO <= F9(8) ;
COUT <= "0000" WHEN F9(8) = '0'   ELSE  "0001" ;
END behav;
```

2. 将输入输出器件绑定到对应的引脚上，然后重新编译。

名称	引脚号	备注
A0_B1	235	Key 3
Sclk	238	Key 6
CN	239	Key 7
M	240	Key 8
IN[0]	1	Key 1
IN[1]	2	Key 1
IN[2]	3	Key 1
IN[3]	4	Key 1
IN[4]	6	Key 2
IN[5]	7	Key 2
IN[6]	8	Key 2
IN[7]	12	Key 2
A[0]	13	数码 1
A[1]	14	数码 1
A[2]	15	数码 1
A[3]	16	数码 1
A[4]	17	数码 2
A[5]	18	数码 2
A[6]	19	数码 2
A[7]	20	数码 2
B[0]	21	数码 3
B[1]	41	数码 3
B[2]	128	数码 3
B[3]	132	数码 3
B[4]	133	数码 4
B[5]	134	数码 4
B[6]	135	数码 4
B[7]	136	数码 4
F[0]	137	数码 5
F[1]	138	数码 5
F[2]	139	数码 5
F[3]	140	数码 5
F[4]	141	数码 6
F[5]	158	数码 6
F[6]	159	数码 6
F[7]	160	数码 6
CN	161	数码 7
S[0]	165	数码 8
S[1]	166	数码 8
S[2]	167	数码 8
S[3]	168	数码 8

表 3.1: ALU 实验引脚表

3. 下载到实验设备上。
4. 调整为工作模式 0。
5. 观察实验现象。
    - Key 1, 2 (D1 - D8)
      控制输入数据，8 位全有效，同时根据 A0_B1 的值显示在数码 1 - 2 或数码 3 - 4 上。
    - Key 3 (D11)
      控制当前输入的是第一个源操作数还是第二个源操作数，锁定 A0_B1 的值。
    - Key 6 (D14)
      控制时钟。其值通过 LPM_COUNTER 累加器反映到数码 8 上。
    - Key 7
      控制 CN 输入，影响 ALU，切换 (带/无) 进位运算模式。
    - Key 8
      控制 M 输入，影响 ALU，切换 (逻辑/算数) 运算模式。
    - 数码管 1, 2
      第一个源操作数，8 位全有效。
    - 数码管 3, 4
      第二个源操作数，8 位全有效。
    - 数码管 5, 6
      ALU 计算结果，8 位全有效。
    - 数码管 7
      ALU 计算结果的进位标记位，最低位有效。
    - 数码管 8
      ALU 功能码，4 位全有效。
6. 绘制仿真波形图。

3.4　实验结果分析

3.4.1　实验电路图

根据原理图 3.1 绘制实验电路图 3.2。

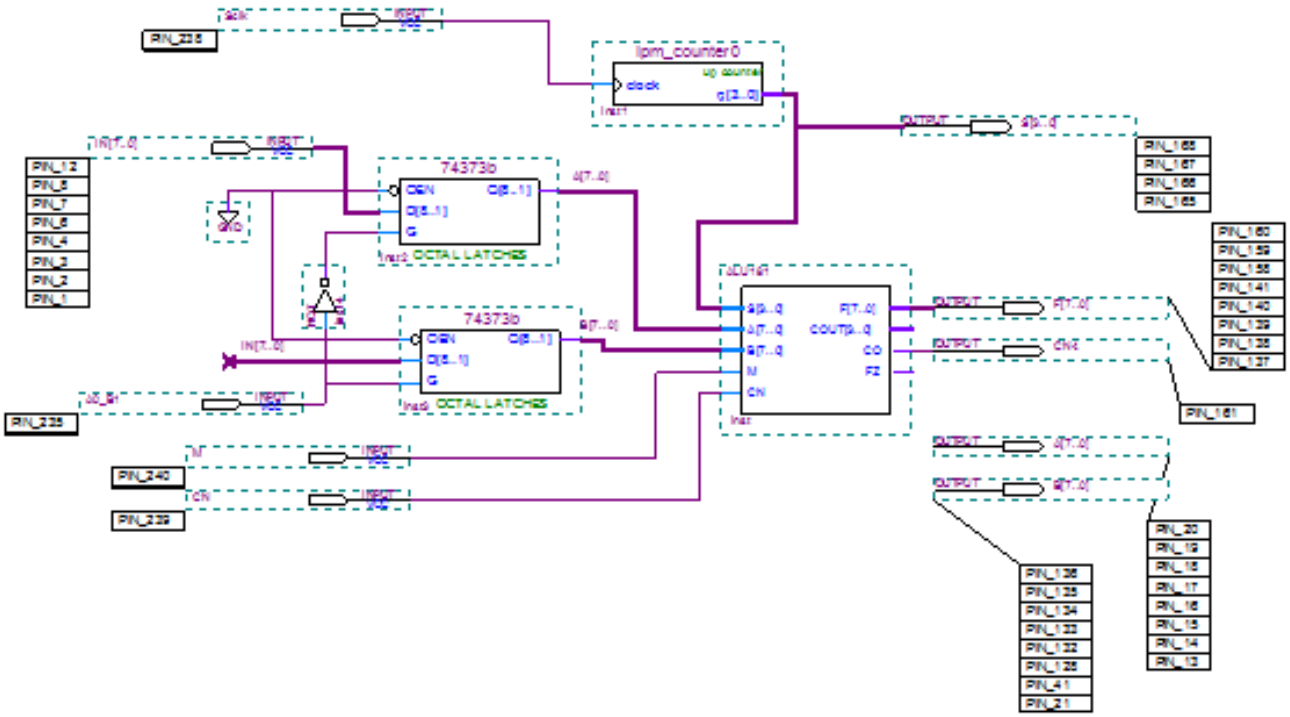


图 3.2: ALU 实验电路图

3.4.2　仿真波形图

利用 Quartus II 产生仿真波形图 3.3，3.4。

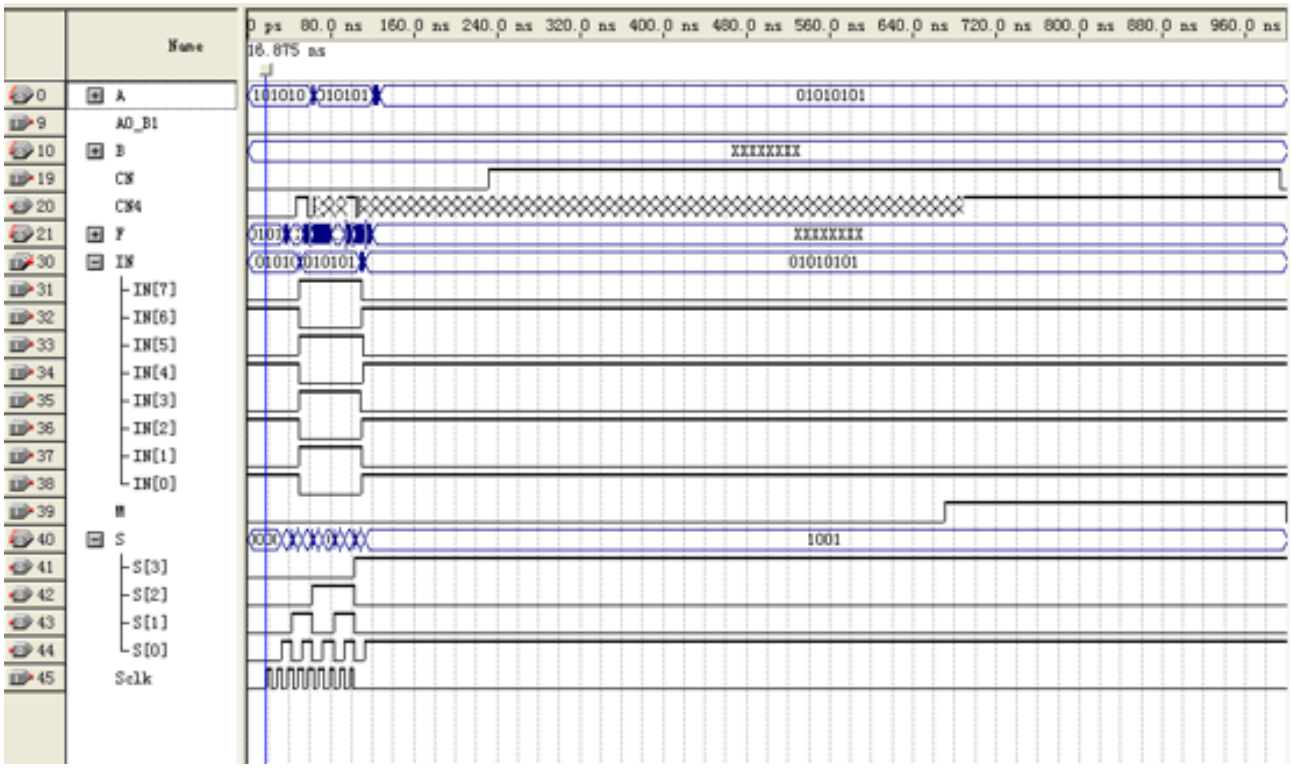


图 3.3: ALU 实验仿真波形图 1

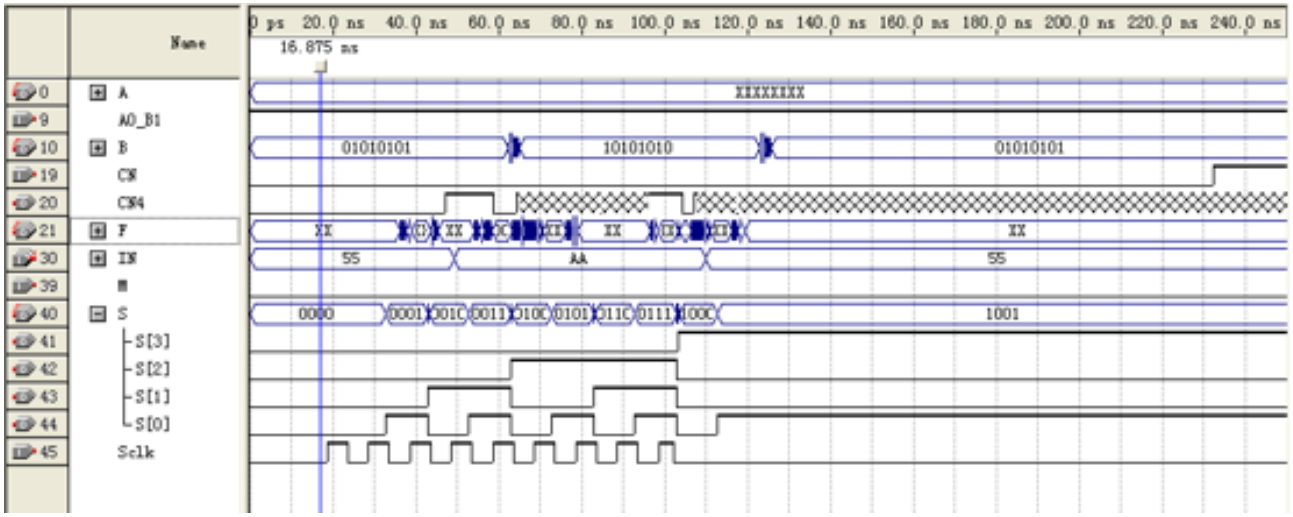


图 3.4: ALU 实验仿真波形图 2

3.4.3　附加题

1. 完成真值表:

   答:

   实验所得真值表如下:

S[4..0]				A[7..0]		B[7..0]		算术运算 M = 0		逻辑运算 M = 1
S3	S2	S1	S0	AH	AL	BH	BL	CN = 0 (无进位)	CN = 1 (有进位)	
0	0	0	0	A	A	5	5	0AA	0AB	155
0	0	0	1	A	A	5	5	0FF	100	100
0	0	1	0	A	A	5	5	1AA	1AB	055
0	0	1	1	A	A	5	5	0FF	1FF	000
0	1	0	0	F	F	0	1	1FD	1FE	1FE
0	1	0	1	F	F	0	1	1FD	1FE	1FE
0	1	1	0	F	F	0	1	0FE	0FD	0FE
0	1	1	1	F	F	0	1	1FF	1FE	0FE
1	0	0	0	F	F	0	1	100	101	101
1	0	0	1	F	F	0	1	100	101	101
1	0	1	0	F	F	0	1	000	001	001
1	0	1	1	F	F	0	1	001	000	000
1	1	0	0	F	F	0	1	1FE	1FF	001
1	1	0	1	5	5	0	1	0AA	0AB	1FF
1	1	1	0	5	5	0	1	054	055	055
1	1	1	1	5	5	0	1	055	054	055

表 3.2: ALU 实验值

理论值表计算如下，两个表中标记黄色的单元格为实验值与理论值不同之处。

S[4..0]				A[7..0]		B[7..0]		算术运算 M = 0		逻辑运算 M = 1
S3	S2	S1	S0	AH	AL	BH	BL	CN = 0 (无进位)	CN = 1 (有进位)	
0	0	0	0	A	A	5	5	0AA	0AB	055
0	0	0	1	A	A	5	5	0FF	100	000
0	0	1	0	A	A	5	5	0AA	0AB	055
0	0	1	1	A	A	5	5	0FF	0FF	000
0	1	0	0	F	F	0	1	1FD	1FE	0FE
0	1	0	1	F	F	0	1	1FD	1FE	0FE
0	1	1	0	F	F	0	1	0FE	0FD	0FE
0	1	1	1	F	F	0	1	0FF	0FE	0FE
1	0	0	0	F	F	0	1	100	101	001
1	0	0	1	F	F	0	1	100	101	001
1	0	1	0	F	F	0	1	100	101	001
1	0	1	1	F	F	0	1	001	000	000
1	1	0	0	F	F	0	1	1FE	1FF	001
1	1	0	1	5	5	0	1	0AA	0AB	0FF
1	1	1	0	5	5	0	1	154	155	055
1	1	1	1	5	5	0	1	055	054	055

表 3.3: ALU 理论值

发现偏差都表现在进位位上，分析原因发现 VHDL 代码 3.1 中关于进位位的实现有问题。

第四章　程序计数器 PC 与地址寄存器 AR 实验

4.1　实验内容

设计 PC 与 AR，采用两种不同的连线方式。

1. 总线开关式
2. 三态门式

4.2　实验原理

4.2.1　多路总线开关式

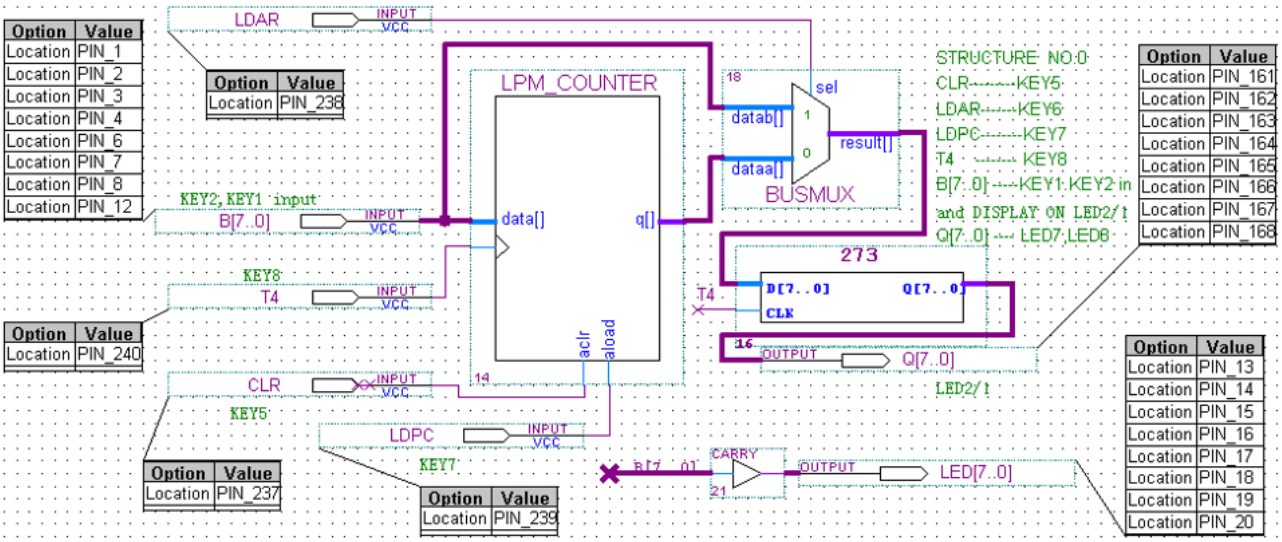


图 4.1: PC & AR 实验总线开关式原理图

如图4.1，由 LPM_COUNTER, BUSMUX 2 路总线开关，273 寄存器 (AR) 组成，数据线 8 位。

输入信号

- T4

  时钟脉冲信号。

  可由外部控制的输入信号。

- B[7..0]

数据总线信号，8 位，用于输入指令。

可由外部控制的输入信号。

- LDAR

  控制信号，是否启用 B 信号。0: 禁用，1: 启用。

  可由外部控制的输入信号。

- CLR

  控制信号，清空 LPM_COUNTER 的值。0: 不清零，1: 清零。

  可由外部控制的输入信号。

- LDPC

  控制信号，控制 LPM_COUNTER 是否从 data 端装入数据。0: 不装入，1: 装入。

  可由外部控制的输入信号。

输出信号

- Q[7..0]

  显示地址寄存器输出的数据，8 位。

- LED[7..0]

  显示 B[7..0] 输入信号，8 位。

4.2.2　三态门式

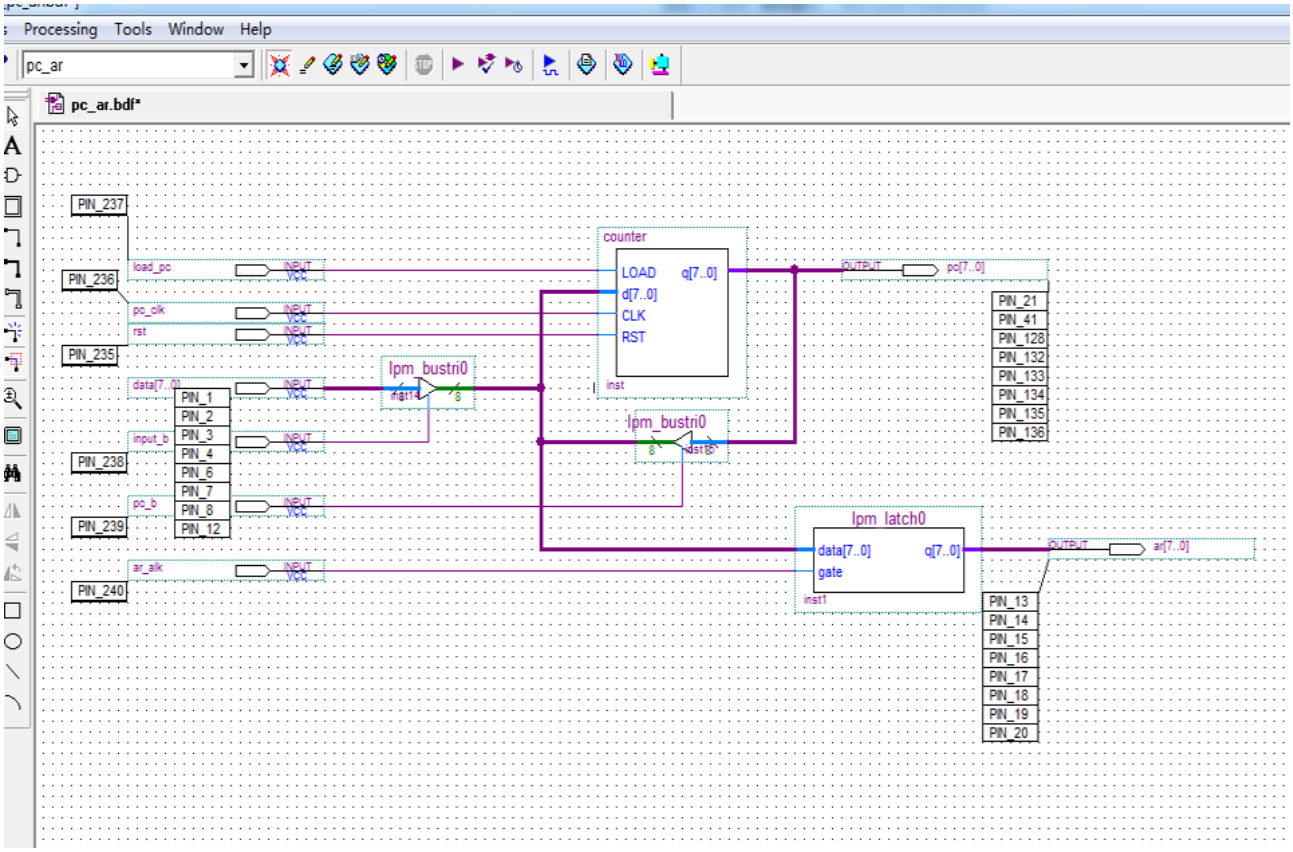


图 4.2: PC & AR 实验三态门式原理图

如图4.2，由 LPM_COUNTER, lpm_bustri0 三态门，latch 锁存器组成，数据线 8 位。

输入信号

- pc_clk

  时钟脉冲信号，其上升沿时计数器自增 1。

  可由外部控制的输入信号。

- data[7..0]

  数据总线信号，8 位，用于输入指令。

  可由外部控制的输入信号。

- load_pc

  控制信号，控制计数器从其 d 端加载数据。0: 不加载，1: 加载。

  可由外部控制的输入信号。

- rst

  控制信号，控制计数器清零。0: 不清零，1: 清零。

  可由外部控制的输入信号。

- input_b

  控制信号，控制左侧三态门的开闭。0: 闭，1: 开。

  可由外部控制的输入信号。

- ar_clk

  时钟脉冲信号，其上升沿时锁存器从其 data 端加载数据。

  可由外部控制的输入信号。

- pc_b

  控制信号，控制右侧三态门的开闭。0: 闭，1: 开。

  可由外部控制的输入信号。

输出信号

- pc[7..0]

  显示计数器输出的 PC 数据，8 位。

- ar[7..0]

  显示锁存器输出的 AR 数据，8 位。

4.3　实验任务与实验步骤

4.3.1　总线开关式

1. 按照原理图 4.1 连接电路图，然后编译。

   (a) 创建 LPM_COUNTER 计数器

   - 启用异步清除
   - 启用异步加载
   - 禁用异步置位
   - 禁用同步清除
   - 禁用同步加载
   - 禁用同步置位

2. 将输入输出器件绑定到对应的引脚上，然后重新编译。

名称	引脚号	备注
CLR	237	Key 5
LDAR	238	Key 6
LDPC	239	Key 7
T4	240	Key 8
B[0]	1	Key 1
B[1]	2	Key 1
B[2]	3	Key 1
B[3]	4	Key 1
B[4]	6	Key 2
B[5]	7	Key 2
B[6]	8	Key 2
B[7]	12	Key 2
LED[0]	13	数码 1
LED[1]	14	数码 1
LED[2]	15	数码 1
LED[3]	16	数码 1
LED[4]	17	数码 2
LED[5]	18	数码 2
LED[6]	19	数码 2
LED[7]	20	数码 2
Q[0]	161	数码 7
Q[1]	162	数码 7
Q[2]	163	数码 7
Q[3]	164	数码 7
Q[4]	165	数码 8
Q[5]	166	数码 8
Q[6]	167	数码 8
Q[7]	168	数码 8

表 4.1: PC & AR 实验：总线开关式引脚表

3. 下载到实验设备上。

4. 调整为工作模式 0。

5. 观察实验现象。

   - Key 1, 2 (D1 - D8)
     控制输入数据 B，8 位全有效，同时显示在数码 1 - 2 上。
   - Key 5 (D13)
     控制是否清空计数器。
   - Key 6 (D14)
     控制是否加载 AR。

- Key 7 (D15)
  控制是否加载 PC。
- Key 8 (D16)
  控制时钟。
- 数码管 1, 2
  显示输入数据 B 的值，8 位全有效。
- 数码管 7, 8
  地址寄存器 AR 输出的值，8 位全有效。

6. 绘制仿真波形图。

4.3.2　三态门式

1. 按照原理图 4.2 连接电路图，然后编译。

   (a) 创建 LPM_COUNTER 计数器

   - 启用异步清除
   - 禁用异步加载
   - 禁用异步置位
   - 禁用同步清除
   - 启用同步加载
   - 禁用同步置位

2. 将输入输出器件绑定到对应的引脚上，然后重新编译。

名称	引脚号	备注
rst	235	Key 3
pc_clk	236	Key 4
load_pc	237	Key 5
input_b	238	Key 6
pc_b	239	Key 7
ar_clk	240	Key 8
data[0]	1	Key 1
data[1]	2	Key 1
data[2]	3	Key 1
data[3]	4	Key 1
data[4]	6	Key 2
data[5]	7	Key 2
data[6]	8	Key 2
data[7]	12	Key 2
ar[0]	13	数码 1
ar[1]	14	数码 1
ar[2]	15	数码 1
ar[3]	16	数码 1
ar[4]	17	数码 2
ar[5]	18	数码 2
ar[6]	19	数码 2
ar[7]	20	数码 2
pc[0]	21	数码 3
pc[1]	41	数码 3
pc[2]	128	数码 3
pc[3]	132	数码 3
pc[4]	133	数码 4
pc[5]	134	数码 4
pc[6]	135	数码 4
pc[7]	136	数码 4

表 4.2: PC & AR 实验：三态门式引脚表

3. 下载到实验设备上。
4. 调整为工作模式 0。
5. 观察实验现象。
   - Key 1, 2 (D1 - D8)
     控制输入数据 B，8 位全有效。
   - Key 3 (D11)
     控制是否清空计数器。
   - Key 4 (D12)

控制计数器自增 1。

- Key 5 (D13)
  控制 PC 预置值加载。
- Key 6 (D14)
  控制输入数据 data 对总线的控制权。
  把控左侧三态门的开闭。
- Key 7 (D15)
  控制计数器的输出对总线的控制权。
  把控右侧三态门的开闭。
- Key 8 (D16)
  控制锁存器的时钟。
- 数码管 1, 2
  显示锁存器输出的 AR 的数据，8 位全有效。
- 数码管 3, 4
  显示计数器输出的 PC 的数据，8 位全有效。

6. 绘制仿真波形图。

4.4　实验结果分析

4.4.1　实验电路图

根据原理图 4.1 绘制实验电路图 4.3。

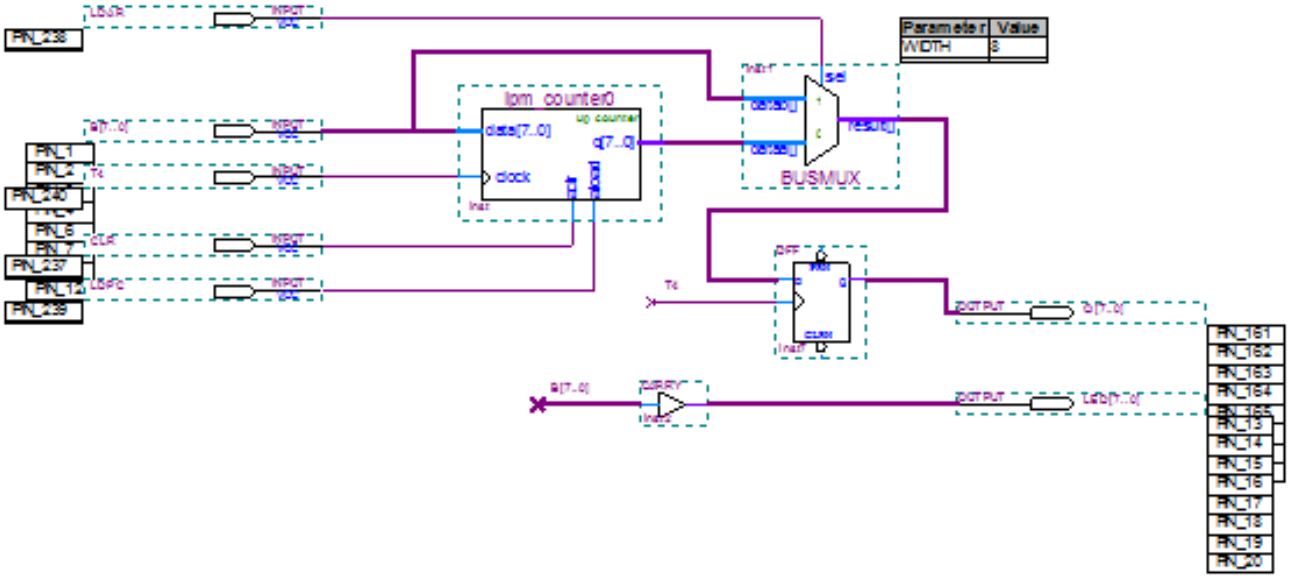


图 4.3: PC & AR 实验总线开关式电路图

根据原理图 4.2 绘制实验电路图 4.4。

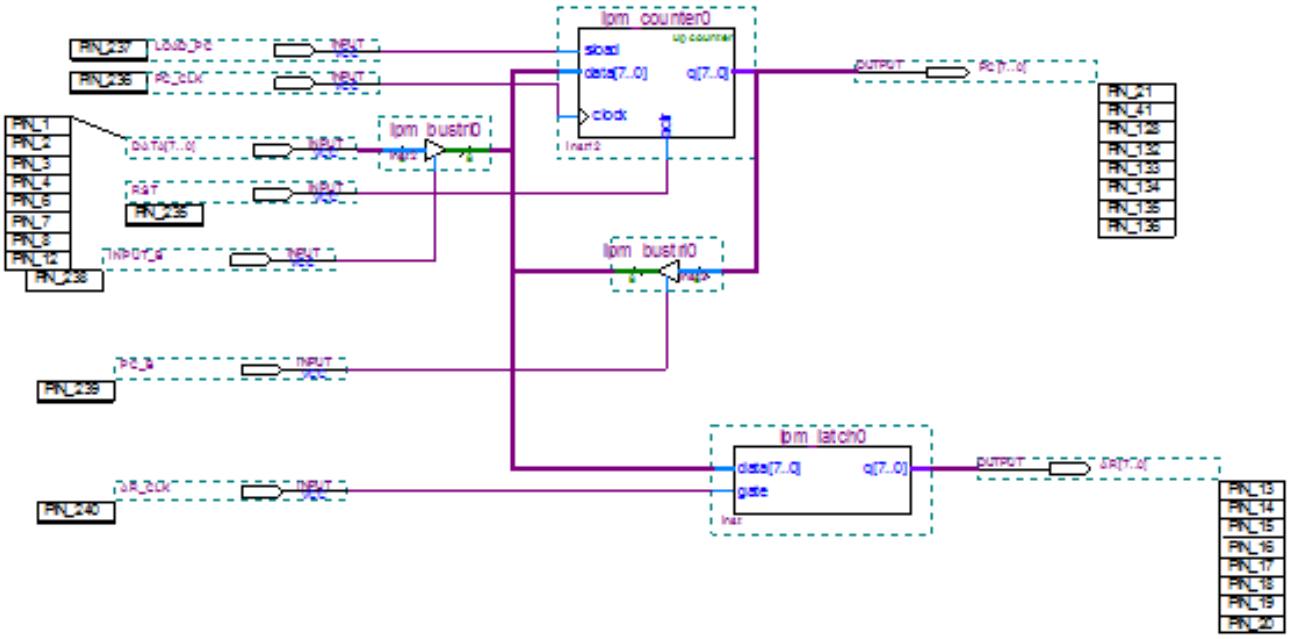


图 4.4: PC & AR 实验三态门式电路图

4.4.2　仿真波形图

利用 Quartus II 产生仿真波形图 4.5。

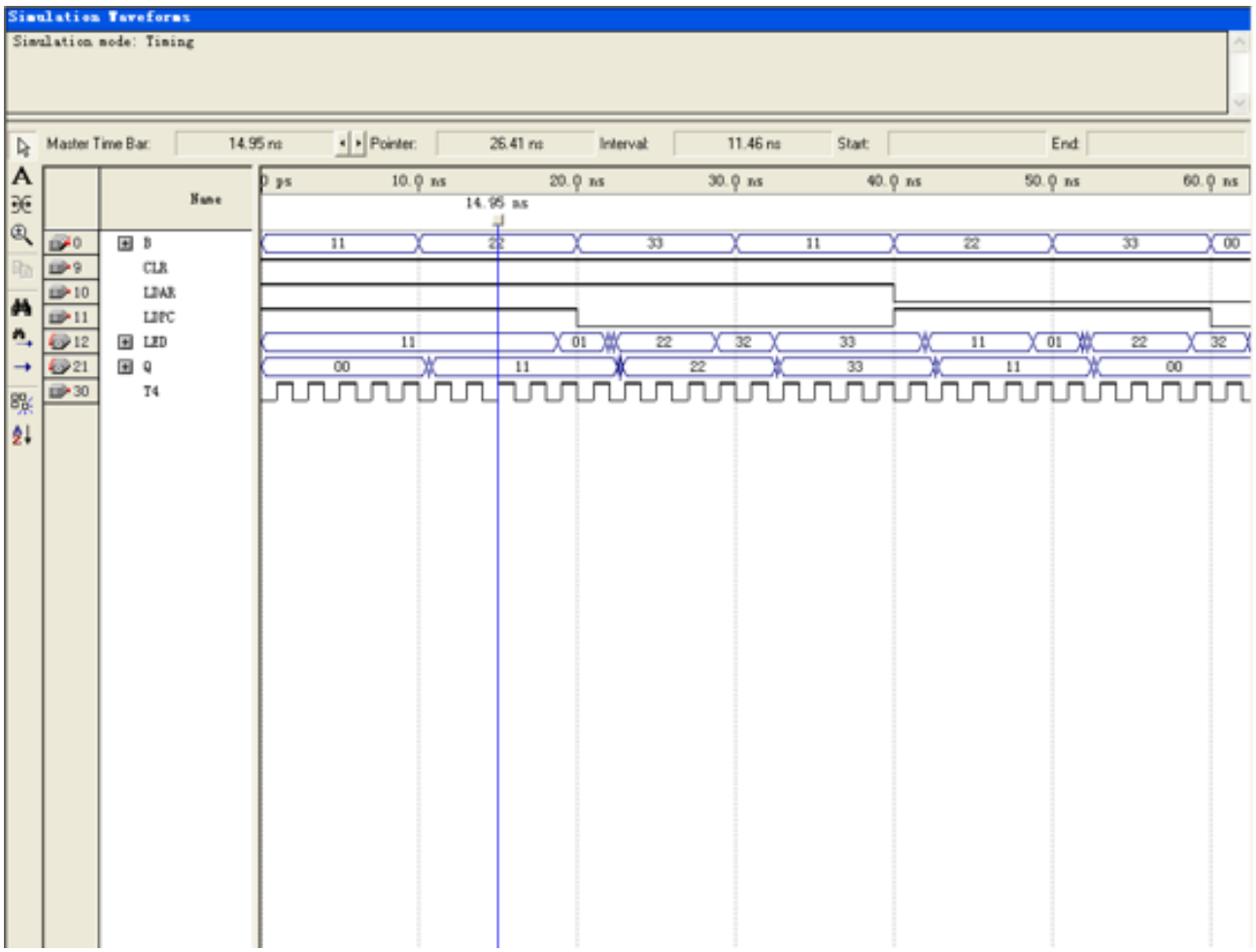


图 4.5: PC & AR 实验总线开关式仿真波形图

利用 Quartus II 产生仿真波形图 4.6。

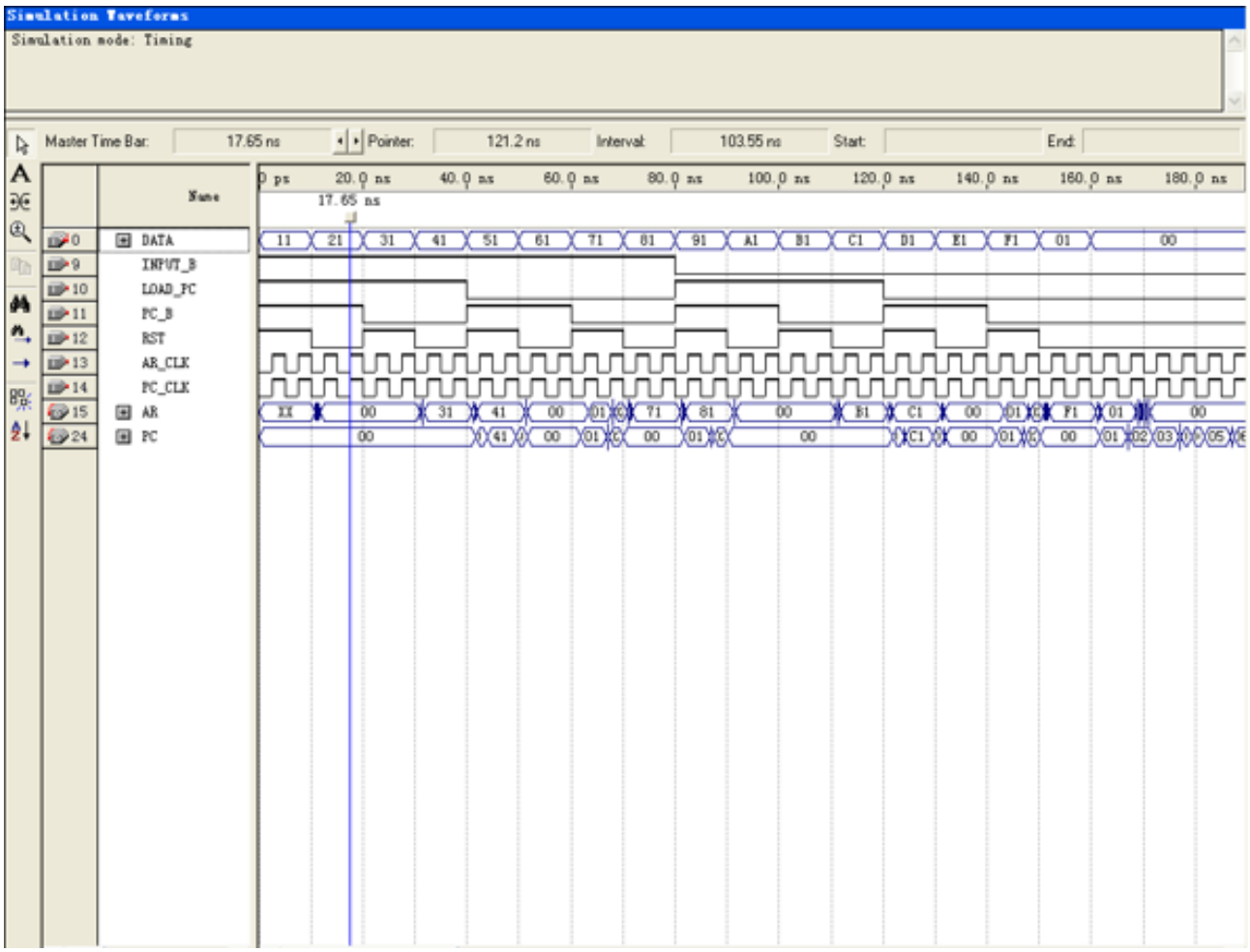


图 4.6: PC & AR 实验三态门式仿真波形图

4.4.3　思考题

声明：下述题目的回答默认参考三态门式原理图 4.2：

1. 说明顺序执行程序时，将 PC 值送 AR，从 AR 所指向的 RAM 地址单元取出指令的操作。

   答：

   不包含跳转指令时，PC 的改变始终来源自增 1 的时钟正脉冲信号，全程关闭左侧三态门。

   input_b := 0 (Key 6)

   (a) 打开右侧三态门，PC 进入内部总线

   pc_b := 1 (Key 7)

   (b) 发出锁存器时钟正脉冲，PC → AR

   ar_clk := 1 (Key 8)

   ar_clk := 0 (Key 8)

   (c) AR → Address Bus

   这是外部过程。

   (d) Address Bus → RAM

   这是外部过程。

(e) 发出计数器时钟正脉冲，PC := PC + 1
pc_clk := 1 (Key 4)
pc_clk := 0 (Key 4)

(f) 重复上述操作。

2. 执行分支/转移程序与执行顺序程序时，对地址单元的操作有何区别?

答:

两个三态门的开闭状态不同。

(a) 分支/转移
打开左侧三态门，关闭右侧三态门。

(b) 顺序执行
关闭左侧三态门，打开右侧三态门。

3. 请说明实现 PC 值自动加 1，指向下一个地址单元的操作过程。

答:

给 pc_clk 接上指令结束的外部同步脉冲信号 [1] 即可实现 PC 的自动自增 1。

4. 请说明在教材中图 4-65 [2] 电路中，在实行程序转移时，从 DATA[7..0] 输入转移地址送 PC 和 AR，AR 输出新的转移地址的操作过程。

答:

(a) 调整内部总线的控制权

- 关闭计数器输出对内部总线的控制权开关 PC_B
  PC_B := 0
- 打开数据线对内部总线的控制权开关 LOAD_PC
  LOAD_PC := 1

以上两项可并发，均执行完毕后进入下一步。

(b) 发送时钟正脉冲信号

- 锁存器时钟
  AR_CLK := 0 AR_CLK := 1
  执行完毕后 AR 应已输出。
- 计数器时钟
  PC_CLK := 0 PC_CLK := 1
  执行完毕后 PC 应已输出。

以上两组脉冲可并发。

5. 要实现程序的分支转移，需对教材中图 4-63 [3] 中的程序计数器 PC 和地址寄存器 AR 做怎样的操作? 应改变那些控制信号?

答:

需要将数据线的值送 PC，然后将 PC 的值送 AR，同时 PC 自增 1。

调整执行顺序如下，可以提高并发性。

[1] 这里指 CPU 完成指令后发送的指令结束正脉冲信号
[2] 教材图 4-63 为三态门式原理图 4.2。
[3] 教材图 4-63 为总线控制式原理图 4.1。

(a) 以下两个操作具有并发性

- 送 AR：调整总线选择器 LDAR
  LDAR := 1
- 送 PC：发送时钟正脉冲信号 LDPC
  LDPC := 1
  defer [4] LDPC := 0
  此时 PC 应已收到数据，但没有输出。

(b) 送 AR：发送时钟正脉冲信号 T4
T4 := 1
T4 := 0
此时 AR 已收到数据并输出到 Q，PC 自增 1。

6. 从存储器读取运算数据和执行取指令操作时，地址控制单元完成的操作有何不同?

答:

- 从储存器读取运算数据
  读操作数发生在指令译码之后，并没有执行完一条指令，不改变 PC 的值。
  DATA 总线传入数据地址，应将数据存入 AR，但不改变 PC 的值。
  (a) 调整内部总线控制权
    – 启用数据总线对内部总线的控制权 INPUT_B
      INPUT_B := 1
    – 禁用计数器 PC 对内部总线的控制权 PC_B
      PC_B := 0
    以上两个操作可并发。
  (b) 送 AR：发送时钟正脉冲信号 AR_CLK
    AR_CLK := 1
    AR_CLK := 0
- 执行取指令操作 (FI)
  取指令操作发生在指令译码之前，从 PC 中读取下一条指令的地址，之后 PC 应自增 1。
  DATA 总线上的数据是无效的。
  (a) 调整内部总线控制权
    – 禁用数据总线对内部总线的控制权 INPUT_B
      INPUT_B := 0
    – 启用计数器 PC 对内部总线的控制权 PC_B
      PC_B := 1
    以上两个操作可并发。
  (b) PC → AR：发送时钟正脉冲信号 AR_CLK
    AR_CLK := 1
    AR_CLK := 0
  (c) PC := PC + 1：发送时钟正脉冲信号 PC_CLK
    PC_CLK := 1
    PC_CLK := 0

[4] defer: 可延迟执行

第五章　微控制器组成实验

5.1　实验内容

本实验分为三个子实验：

- 微指令控制电路
- 微指令寄存器电路
- 数据寄存器译码控制电路

5.2　实验原理

5.2.1　微指令控制电路实验

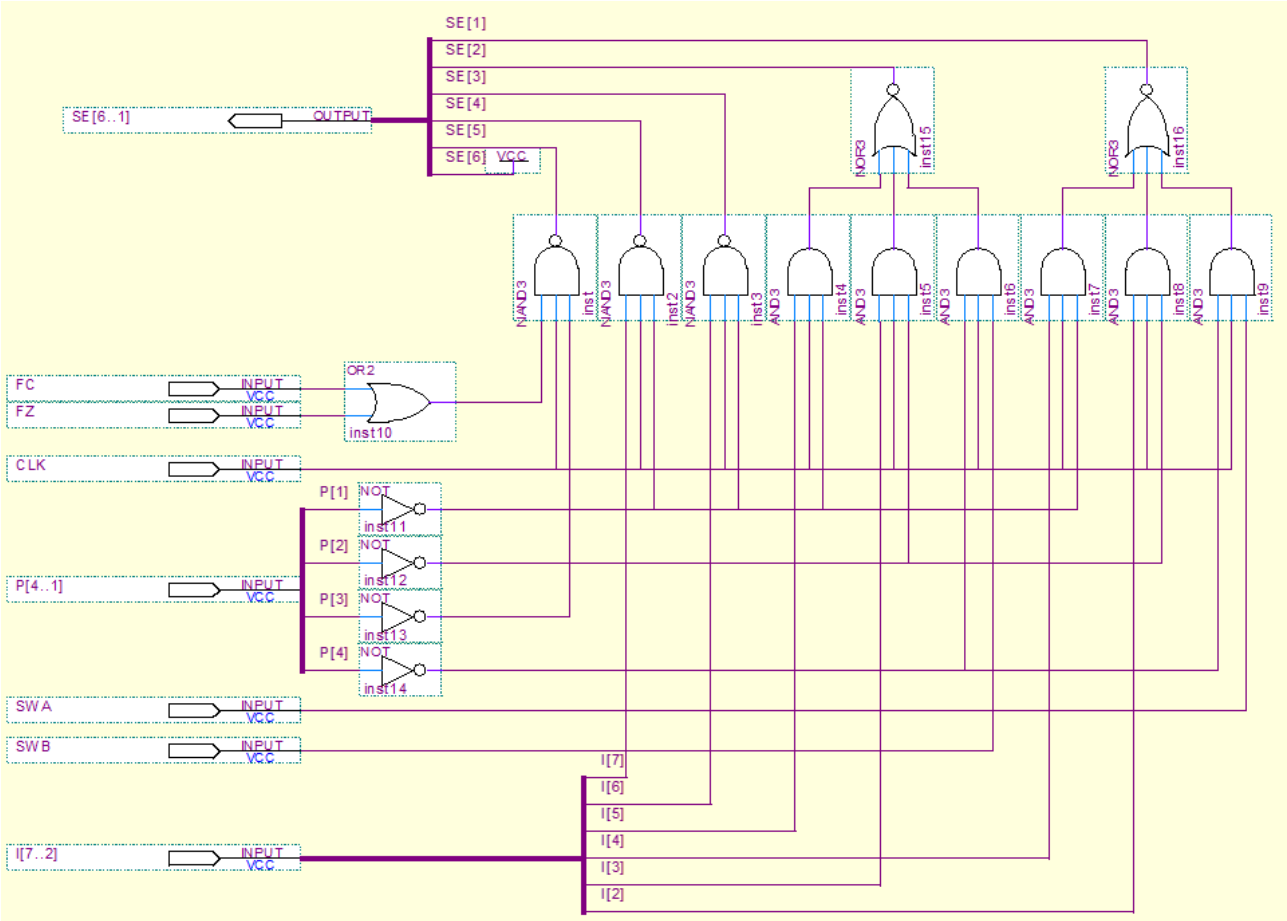


图 5.1: 微指令控制电路

如图5.1，由各种逻辑门电路组合而成。

输入信号

- FC, FZ

  标志信号。

- CLK

  时钟信号。

- P[4..1]

  分支控制信号，4 位。

  重置信号，连续节拍发生器的开关。

- SWA, SWB

  控制台控制信号。

- I[7..2]

  指令数据，6 位。

输出信号

- SE[6..1]

  输出地址控制信号，6 位。

5.2.2　微指令寄存器电路

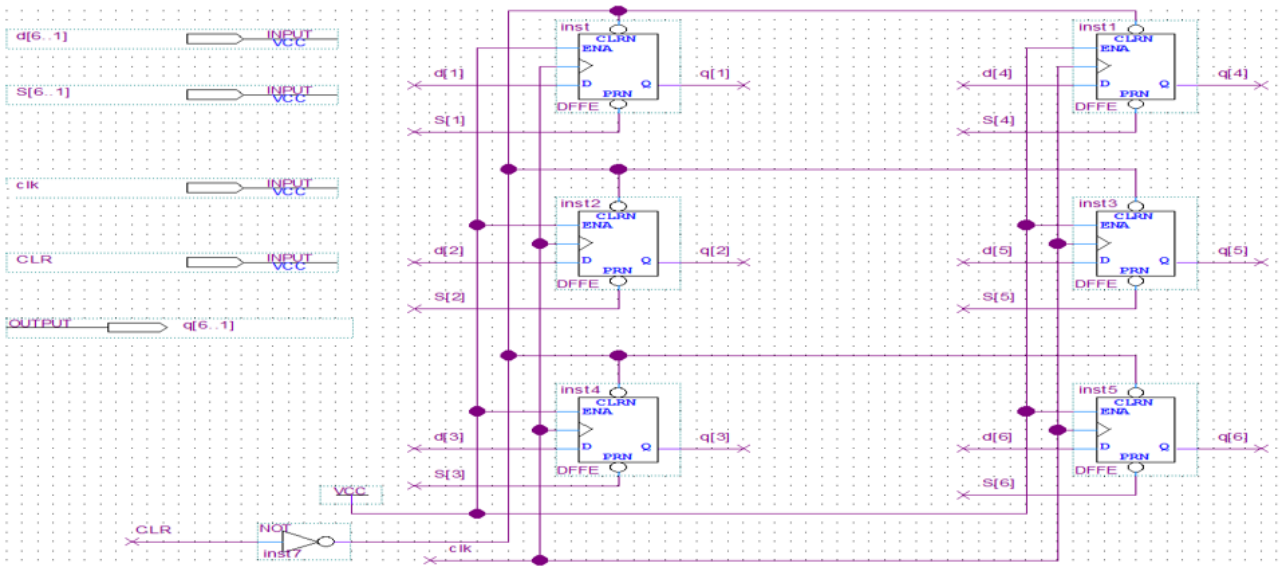


图 5.2: 微指令寄存器电路

如图5.2，由六个 D 触发器、一个非门，形成一个 6 位寄存器，用于寄存微指令控制电路输出的 6 位 SE 地址信号。

输入信号

- S[6..1]

  地址总线信号，6 位。

  来自微指令控制电路的输出。

- clk

  时钟信号，每个正脉冲更新所有 D 触发器的输出。

- CLR

  控制信号，控制所有 D 触发器的清空。0: 清空，1: 不清空。

- d[6..1]

  数据信号，D 触发器的默认值数据，6 位。

  可通过这个数据信号设置地址掩码 (mask code)。

输出信号

- q[6..1]

  输出地址信号，6 位。

5.2.3　数据寄存器译码控制电路

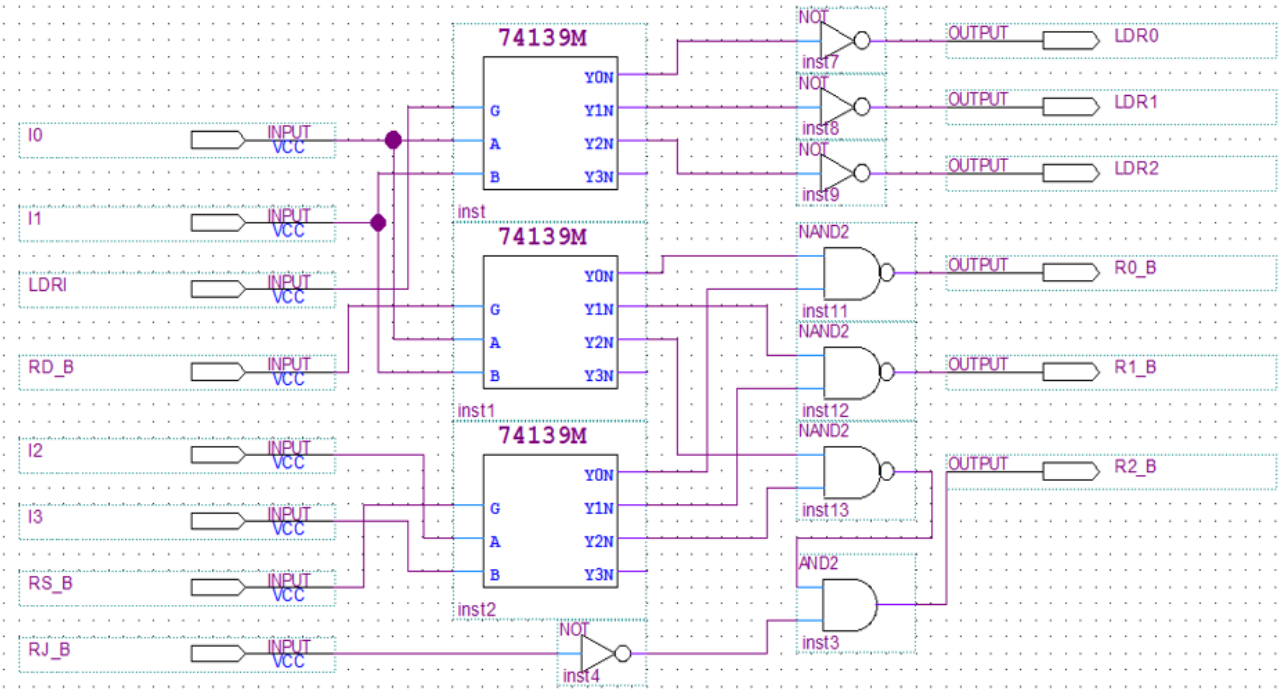


图 5.3: 数据寄存器译码控制电路

如图5.3，由三个 74139M 二四译码器、若干逻辑门组成，形成一个译码电路。

输入信号

- I0, I1, I2, I3, LDRI, RD_B, RS_B, RJ_B

  不可描述的谜之输入信号。

输出信号

- LDR0, LDR1, LDR2, R0_B, R1_B, R2_B

  不可描述的谜之输出信号。

5.3　实验任务与实验步骤

5.3.1　微指令控制电路

1. 按照原理图 5.1 连接电路图，然后编译。
2. 将输入输出器件绑定到对应的引脚上，然后重新编译。

名称	引脚号	备注
CLK	173	Key 8
I[2]	233	Key 1
I[3]	234	Key 1
I[4]	235	Key 1
I[5]	236	Key 1
I[6]	237	Key 2
I[7]	238	Key 2
FC	239	Key 2
FZ	240	Key 2
P[1]	1	Key 3
P[2]	2	Key 3
P[3]	3	Key 3
P[4]	4	Key 3
SWA	6	Key 4
SWB	7	Key 4
SE[1]	13	数码 5
SE[2]	14	数码 5
SE[3]	15	数码 5
SE[4]	16	数码 5
SE[5]	17	数码 6
SE[6]	18	数码 6

表 5.1: 微指令控制电路实验：引脚表

3. 下载到实验设备上。
4. 调整为工作模式 1。
5. 观察实验现象。

- Key 1, 2 (数码 1, 2)
  输入 I 数据（低 6 位）
  输入 FC （最高位）
  输入 FZ （次高位）
- Key 3 (数码 3)
  输入 P 数据，4 位全有效。
- Key 4 (数码 4)
  输入 SWA (最低位)
  输入 SWB (次低位)
- Key 8 (D16)
  时钟信号。
- 数码 5, 6
  输出的 SE 信号，低 6 位有效。

6. 绘制仿真波形图。

5.3.2　微指令寄存器电路

1. 按照原理图 5.2 连接电路图，然后编译。
2. 将输入输出器件绑定到对应的引脚上，然后重新编译。

名称	引脚号	备注
CLR	169	Key 7
clk	173	Key 8
d[1]	233	Key 1
d[2]	234	Key 1
d[3]	235	Key 1
d[4]	236	Key 1
d[5]	237	Key 2
d[6]	238	Key 2
S[1]	1	Key 3
S[2]	2	Key 3
S[3]	3	Key 3
S[4]	4	Key 3
S[5]	6	Key 4
S[6]	7	Key 4
q[1]	21	数码 7
q[2]	41	数码 7
q[3]	128	数码 7
q[4]	132	数码 7
q[5]	133	数码 8
q[6]	134	数码 8

表 5.2: 微指令寄存器电路实验：引脚表

3. 下载到实验设备上。
4. 调整为工作模式 1。
5. 观察实验现象。
   - Key 1, 2 (数码 1, 2)
     输入 d 数据，低 6 位有效。
   - Key 3, 4 (数码 3, 4)
     输入 S 数据，低 6 位有效。
   - Key 7 (D15)
     寄存器清空信号，低电平有效。
   - Key 8 (D16)
     时钟信号。
   - 数码 7, 8
     输出的 q 数据，低 6 位有效。
6. 绘制仿真波形图。

5.3.3　数据寄存器译码控制电路

1. 按照原理图 5.3 连接电路图，然后编译。
2. 将输入输出器件绑定到对应的引脚上，然后重新编译。

名称	引脚号	备注
I0	233	Key 1 (D9)
I1	234	Key 2 (D10)
I2	235	Key 3 (D11)
I3	236	Key 4 (D12)
LDRI	237	Key 5 (D13)
I2	238	Key 6 (D14)
I3	239	Key 7 (D15)
LDRI	240	Key 8 (D16)
LDR0	1	D1
LDR1	2	D2
LDR2	3	D3
R0_B	7	D6
R1_B	8	D7
R2_B	12	D8

表 5.3: 数据寄存器译码控制电路实验：引脚表

3. 下载到实验设备上。
4. 调整为工作模式 5。

5. 观察实验现象。

   参照引脚表 5.3，按键 1 - 8，观察 LED 灯 1 - 3, 6 - 16 的值。

6. 绘制仿真波形图。

5.4　实验结果分析

5.4.1　实验电路图

根据原理图 5.1 绘制实验电路图 5.4。

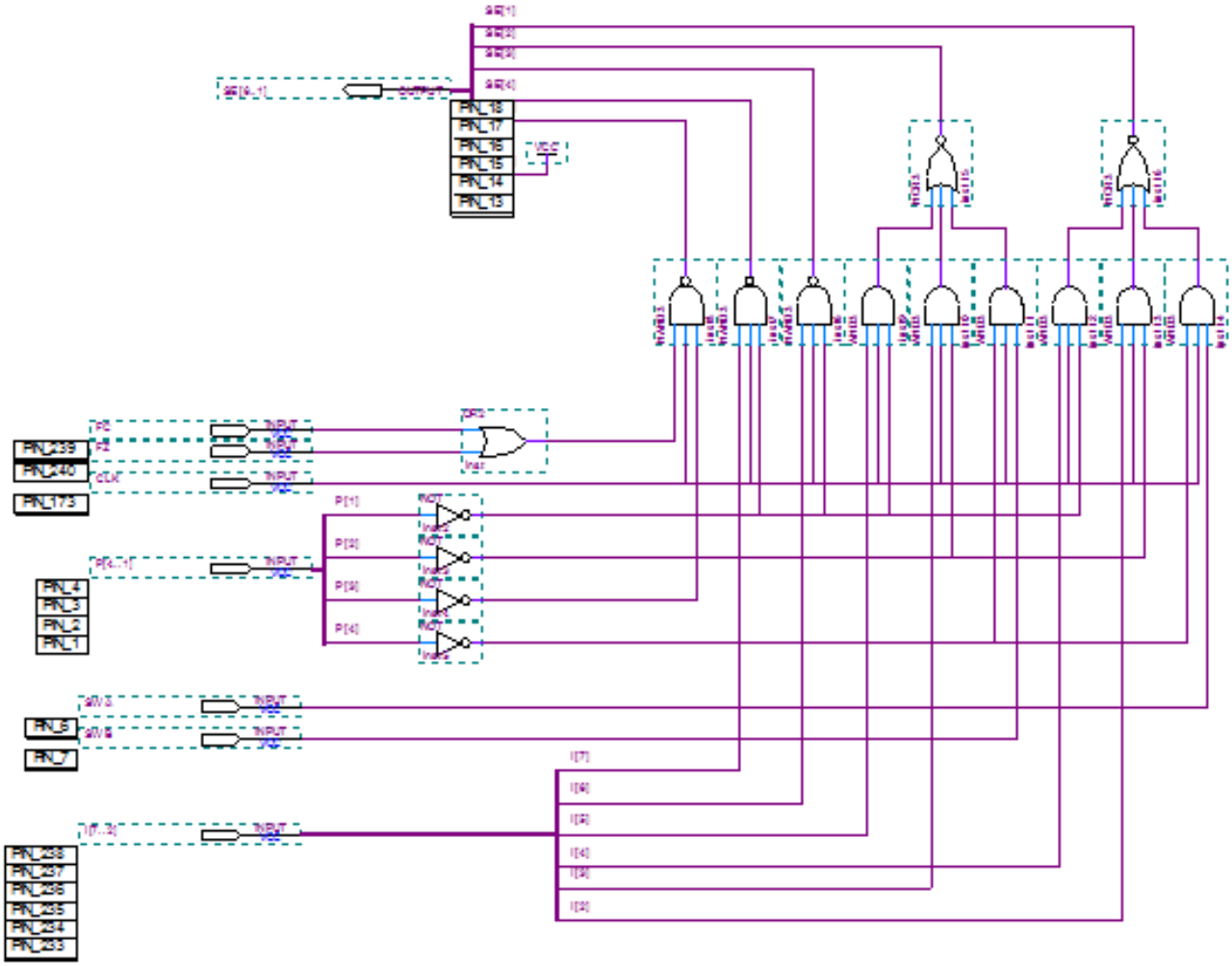


图 5.4: 微指令控制电路图

根据原理图 5.2 绘制实验电路图 5.5。

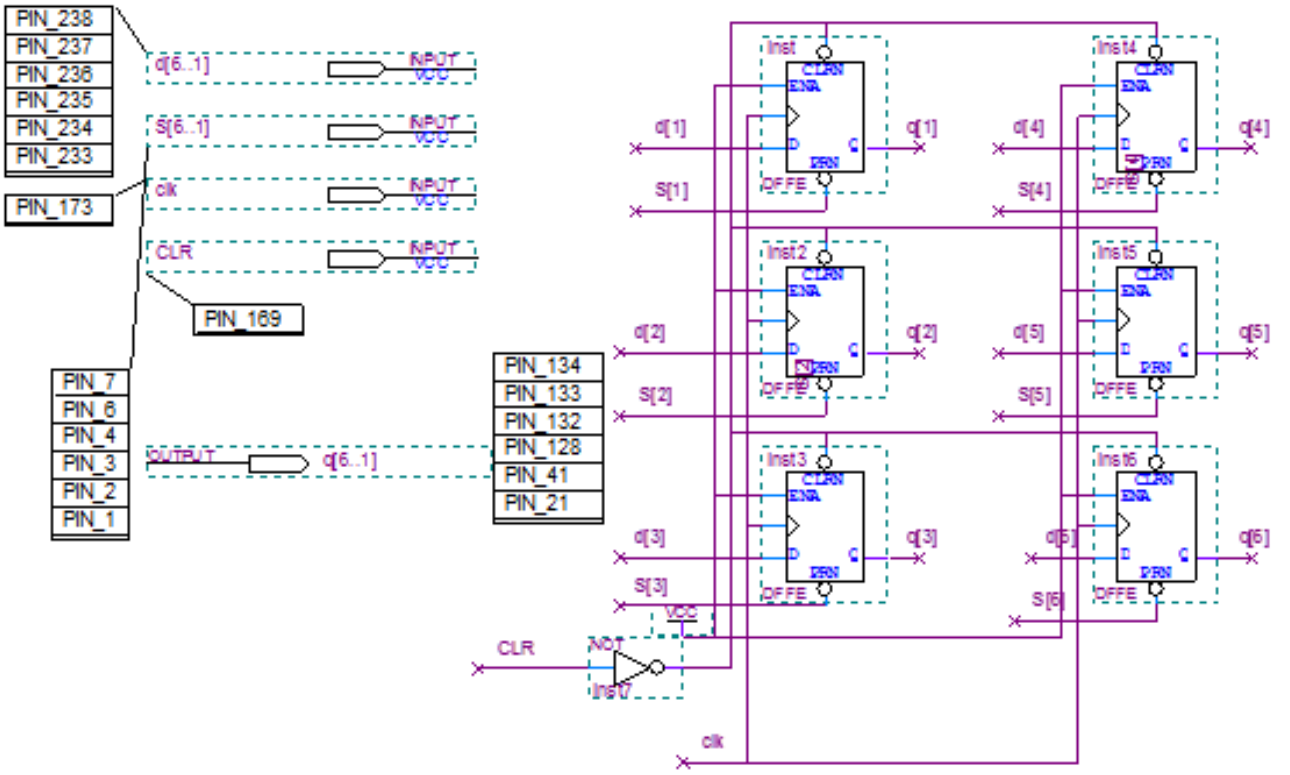


图 5.5: 微指令寄存器电路图

根据原理图 5.3 绘制实验电路图 5.6。

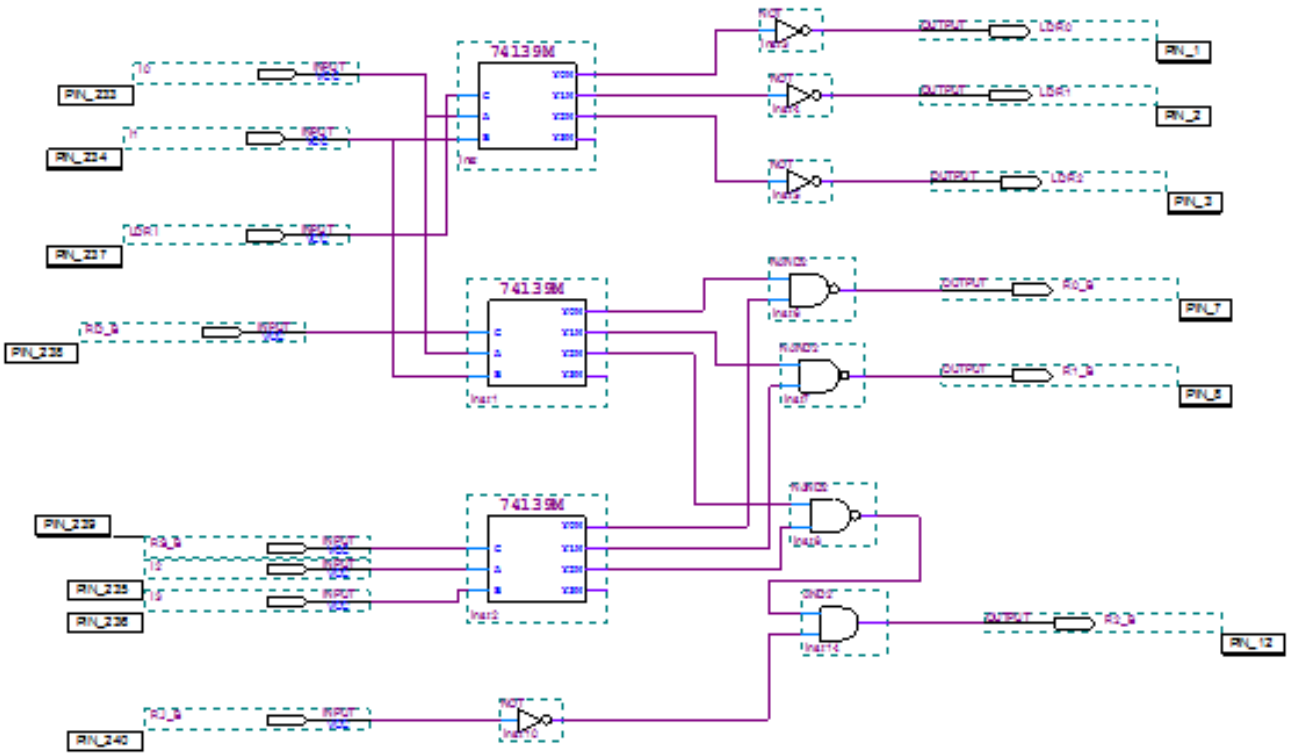


图 5.6: 数据寄存器译码控制电路图

5.4.2　仿真波形图

利用 Quartus II 产生仿真波形图 5.7。

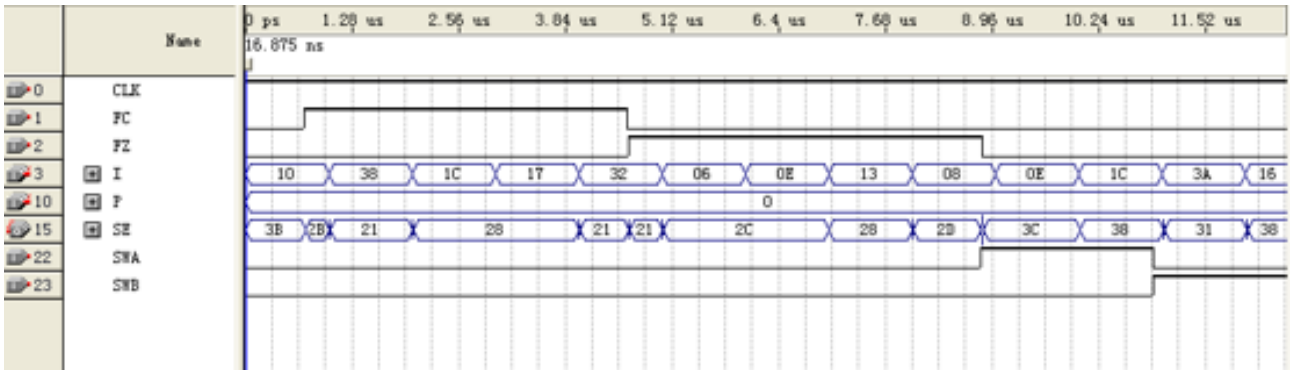


图 5.7: 微指令控制电路仿真波形图

利用 Quartus II 产生仿真波形图 5.8。

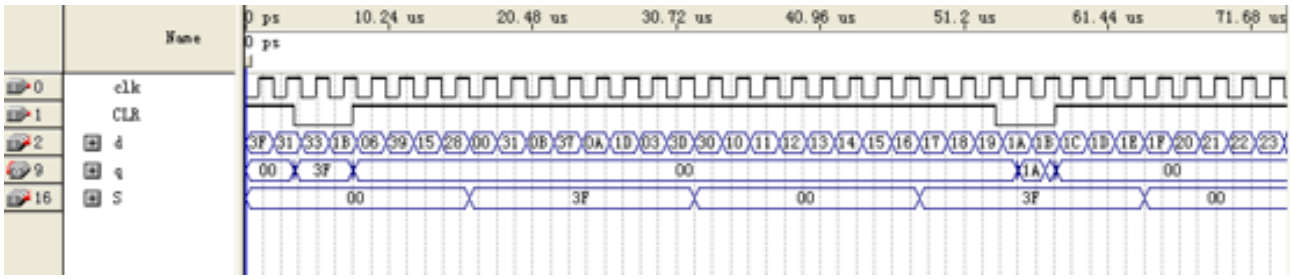


图 5.8: 微指令寄存器电路仿真波形图

利用 Quartus II 产生仿真波形图 5.9。

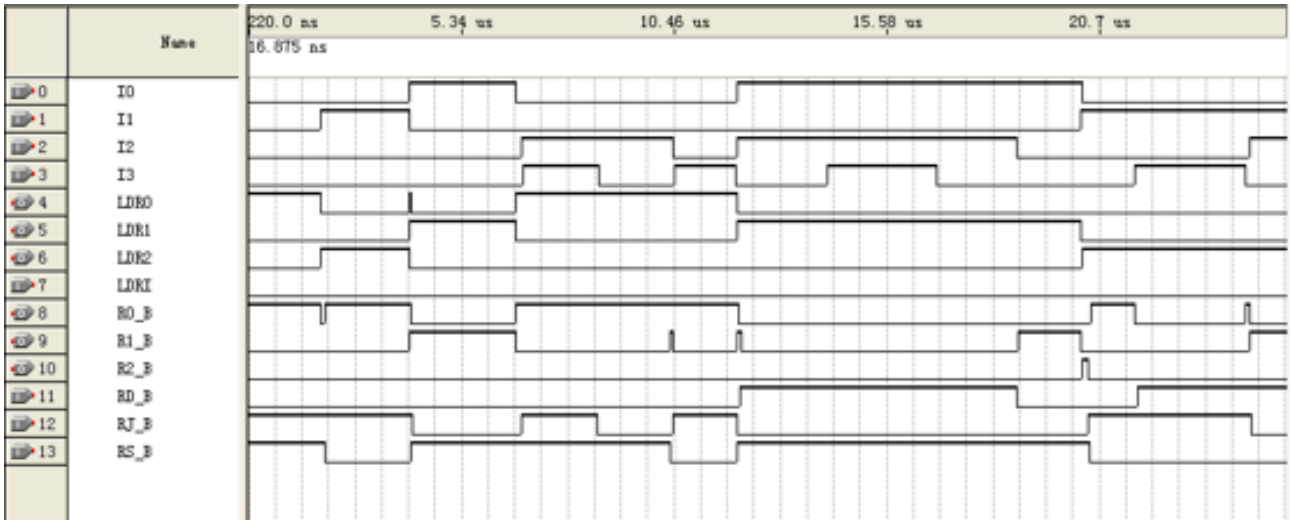


图 5.9: 数据寄存器译码控制电路仿真波形图

5.4.3 思考题

1. 在微指令控制电路中，当 FC 或 FZ 有效时，对其输出 SE[6..1] 有何影响? 对微地址寄存器的输出会有何影响? 如何实现对程序的控制/转移功能?

   答:

   讨论 FC 或 FZ 有效时:

   对输出 SE 的影响：等同于 FC, FZ 直接接地

   首先 FC, FZ 只可能对 SE[5] 有影响，对 SE 其他位无影响（等同于接地）。

   且仅当 FC = FZ = 0 时影响 S[5] = 1。

   那么当 FC 或 FZ 有效时，其对 SE 无影响。

   对微地址寄存器的输出的影响：等同于 FC, FZ 直接接地

   微地址寄存器的输入来源于微指令控制电路的输出，由上题得无影响（等同于接地）。

   结束 FC, FZ 的讨论。

   讨论如何实现程序的控制/转移：输入恰当的 P 值

   P 值是微程序功能码，会被译码成指令。某个 P 对应的是控制/转移指令，就可以实现控制/转移。

2. 说明 P[4..1] 信号分别有效时，对微指令控制电路中输出 SE[6..1] 有何影响?

   答:

   - P[1] = 1 时
     - 直接影响 SE[3] = SE[4] = 1
     - 屏蔽了 I[5] 对 SE[2] 的影响
     - 屏蔽了 I[4] 对 SE[1] 的影响
   - P[2] = 1 时
     - 屏蔽了 I[3] 对 SE[2] 的影响
     - 屏蔽了 I[2] 对 SE[1] 的影响
   - P[3] = 1 时
     - 直接影响 SE[5] = 1
   - P[4] = 1 时
     - 屏蔽了 SWB 对 SE[2] 的影响
     - 屏蔽了 SWA 对 SE[1] 的影响

3. 当控制信号 SWA、SWB 取不同的值时，对微指令控制电路中输出 SE[6..1] 有何影响?

   答:

   SWA 影响 SE[1], SWB 影响 SE[2]，互相独立。

   当 SWx = 0 时，能屏蔽 P[4] 对上述对应的 SE[y] 的影响。

第六章　基本模型计算机设计实验

6.1　实验内容

设计 8 位 CISC 模型计算机。

1. 深入理解基本模型计算机的功能、组成知识；
2. 深入学习计算机各类典型指令的执行流程；
3. 学习微程序控制器的设计过程和相关技术，掌握 LPM_ROM 的配置方法；
4. 在掌握部件单元电路实验的基础上，进一步将单元电路组成系统，构造一台基本模型计算机；
5. 定义五条机器指令，并编写相应的微程序，上机调试，掌握计算机整机概念；
6. 掌握微程序的设计方法，学会编写二进制微指令代码表；
7. 通过熟悉较完整的计算机的设计，全面了解并掌握微程序控制方式计算机的设计方法。

6.2　实验原理

在微过程的控制下自动产生各部件单元控制信号，实现特定的功能。试验中，计算机数据通路的控制将由微过程控制器来完成，CPU 从内存中取出一条机器指令对应一个微程序。本实验采用五条机器指令：IN（输入）,ADD（二进制加法）,STA（存数据）,OUT（输出）,JMP（无条件跳转）。步骤如下：先根据 pc 的值读取内存中指定的指令（操作码，操作数的地址，操作结果的存储地址，下一条指令的地址）放入寄存器中，数据的前几位是操作码，也就是 CPU 的指令码，将该指令码送往 ALU（运算器），在解析出指令中的操作数等信息，对相应的数据进行相应的操作，完成特定的功能。微指令（微程序控制的计算机中，将由同时发出的控制信号所执行的一组微操作）。微操作码字段，又称操作控制字段，该字段指出微指令执行的微操作；微地址码字段，又称顺序控制字段，指出下一条要执行的微指令的地址。

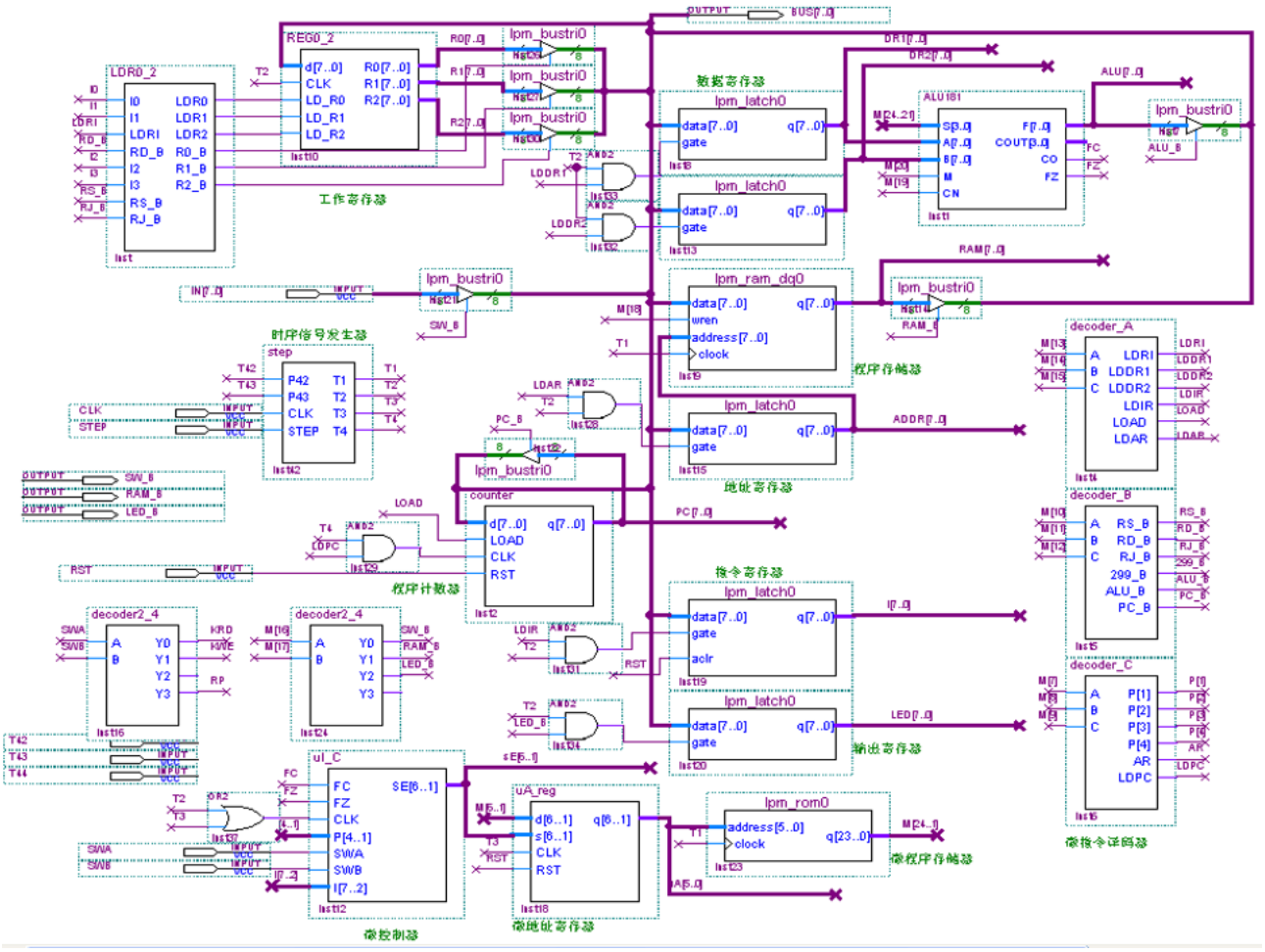


图 6.1: 基本模型计算机 CPU 顶层设计原理图

24	23	22	21	20	19	18	17	16	15 14 13	12 11 10	9 8 7	6	5	4	3	2	1
S3	S2	S1	S0	M	Cn	WE	A9	A8	A	B	C	uA5	uA4	uA3	uA2	uA1	uA0
操作控制信号									译码器	译码器	译码器	下地址字段					

图 6.2: 微指令指令格式

A字段				B字段				C字段			
15	14	13	选择	12	11	10	选择	9	8	7	选择
0	0	0		0	0	0		0	0	0	
0	0	1	LDRi	0	0	1	RS-B	0	0	1	P (1)
0	1	0	LDDR1	0	1	0	RD_B	0	1	0	P (2)
0	1	1	LDDR2	0	1	1	RJ_B	0	1	1	P (3)
1	0	0	LDIR	1	0	0	SFT_B	1	0	0	P (4)
1	0	1	LOAD	1	0	1	ALU-B	1	0	1	AR
1	1	0	LDAR	1	1	0	PC-B	1	1	0	LDPC

图 6.3: 微指令指令格式：图 6.2 中 A, B, C 字段详情

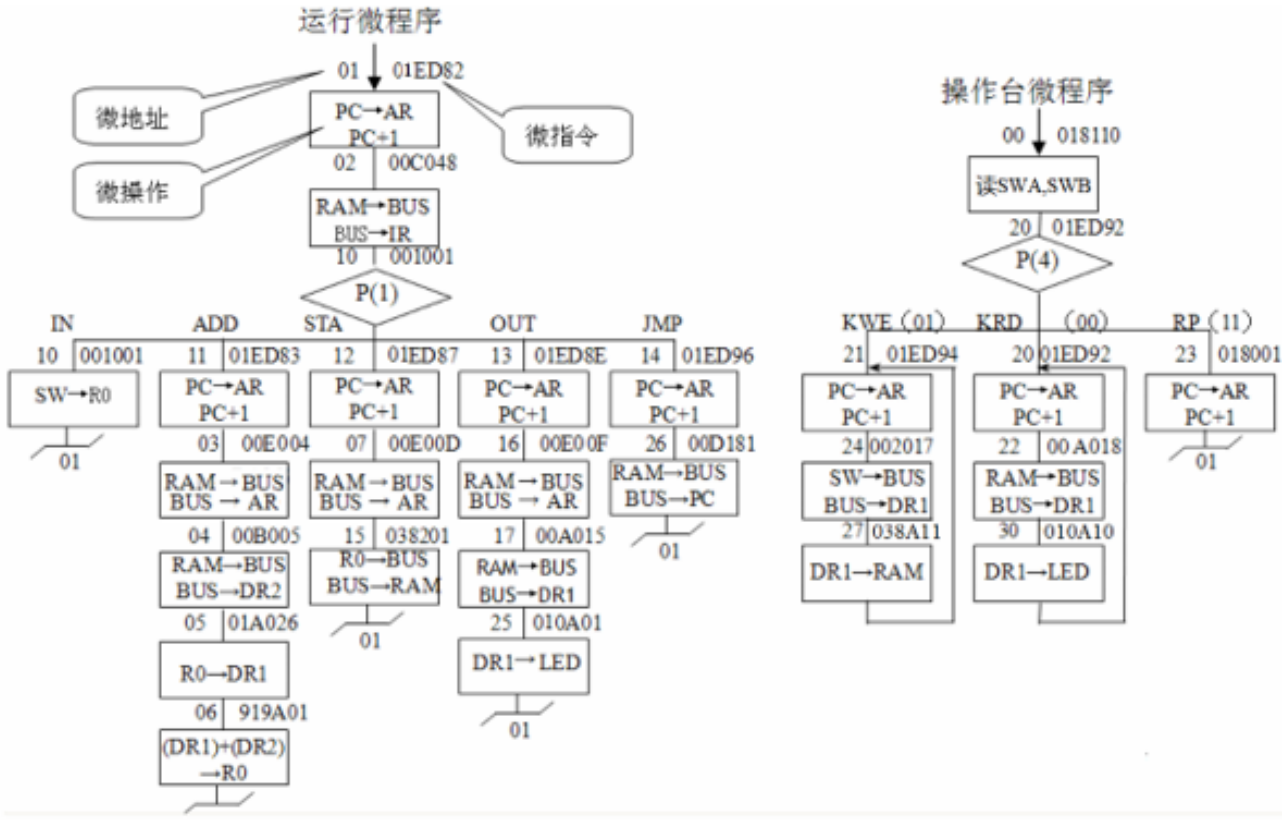


图 6.4: 微指令流程图

6.3　实验任务与实验步骤

1. 用图形编辑工具设计模型 CPU 的顶层电路原理图。

2. 根据微程序的微操作，对于所需的控制信号，确定微指令，并确定微地址。

3. 微程序流程图按微指令格式转化为“二进制微代码表”。

4. 设计控制存储器 LPM_ROM。
5. 对模型 CPU 的整机硬件电路进行编译、波形仿真和调试。
6. 根据仿真波形，查找故障原因，排除故障，重新编译。
7. 将编译通过的电路和应用程序下载到实验台上的 FPGA 中，在实验台上单步跟踪微程序的执行过程。
8. 最终完成模型 CPU 的硬件电路设计和应用程序及微程序的设计和调试。

6.4　实验结果分析

6.4.1　实验电路图

根据原理图 6.1 绘制实验电路图 6.5。

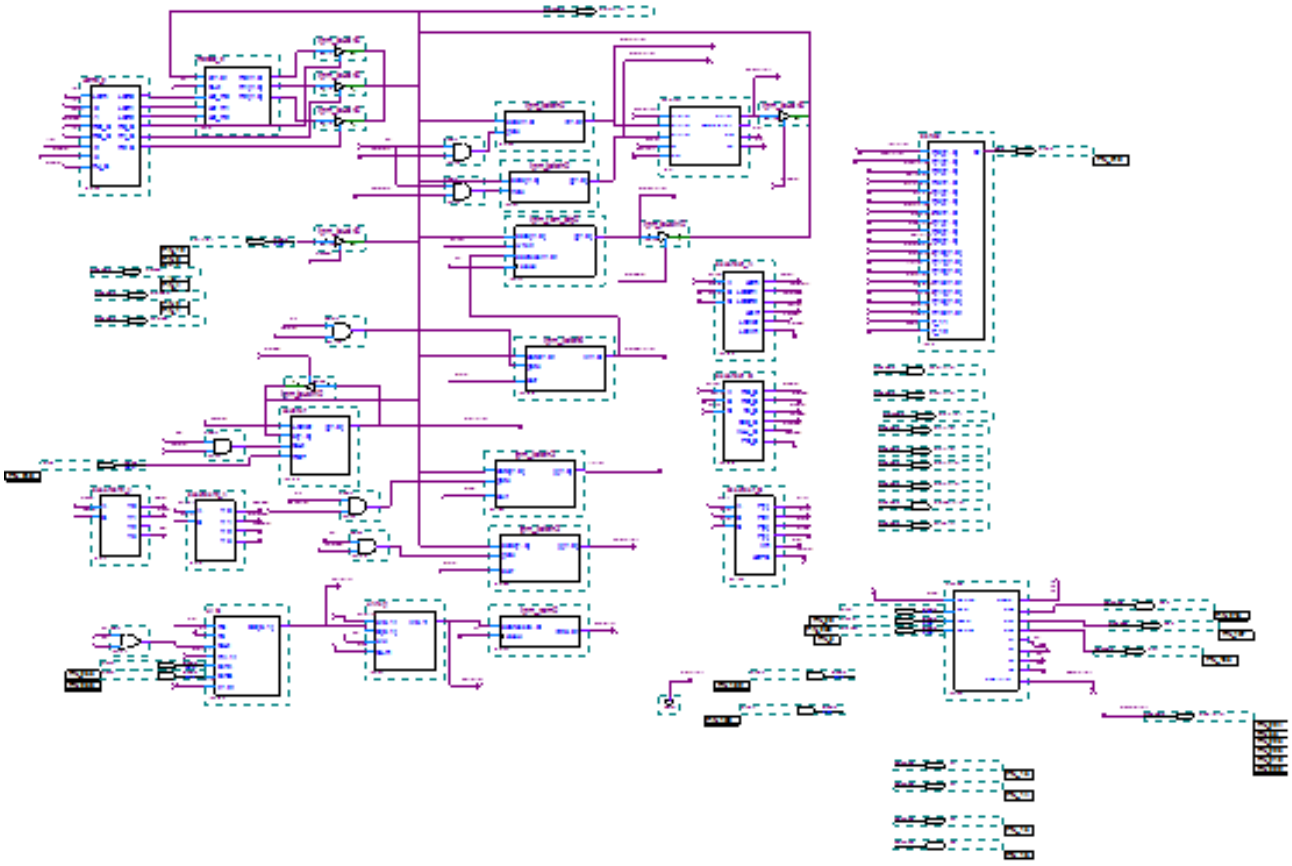


图 6.5: 基本模型计算机实验电路图

6.4.2　储存器快照

利用内存编辑器查看 ROM, RAM 快照 6.6。

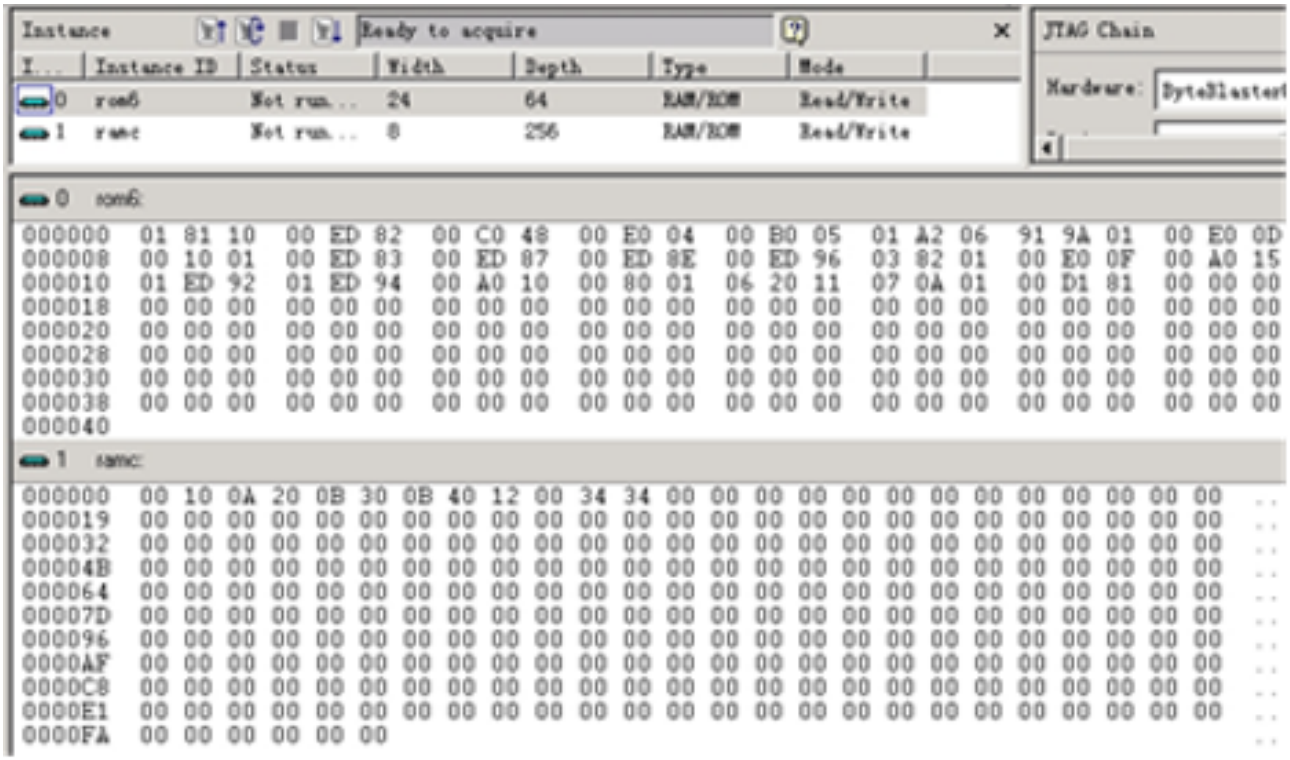


图 6.6: 基本模型计算机实验 ROM RAM 快照

6.4.3　仿真波形图

利用 Quartus II 产生仿真波形图 6.7。

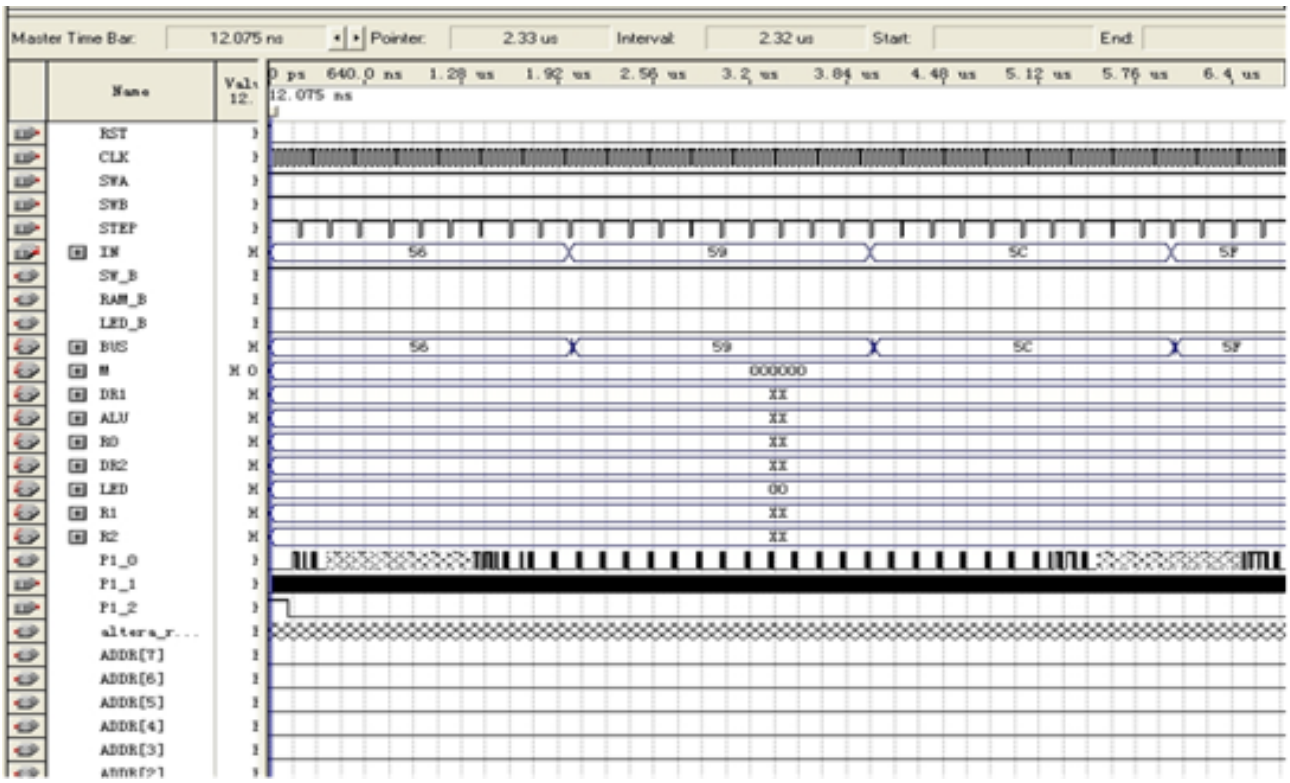


图 6.7: 基本模型计算机实验仿真波形图

6.4.4　思考题

1. 除了已有的 IN,ADD,STA,OUT,JMP 指令外，再设计减法 SUB，带进位加法 ADC，逻辑与 AND，逻辑或 OR 和异或 XOR 指令，共 10 条指令，编写相应微程序流程图，写出微指令代码表。[1]

   答:

[1] 从我能获取的参考资料上已不能找到完整的题目，参考他人的报告得到完整题目，如有错误，请指出。

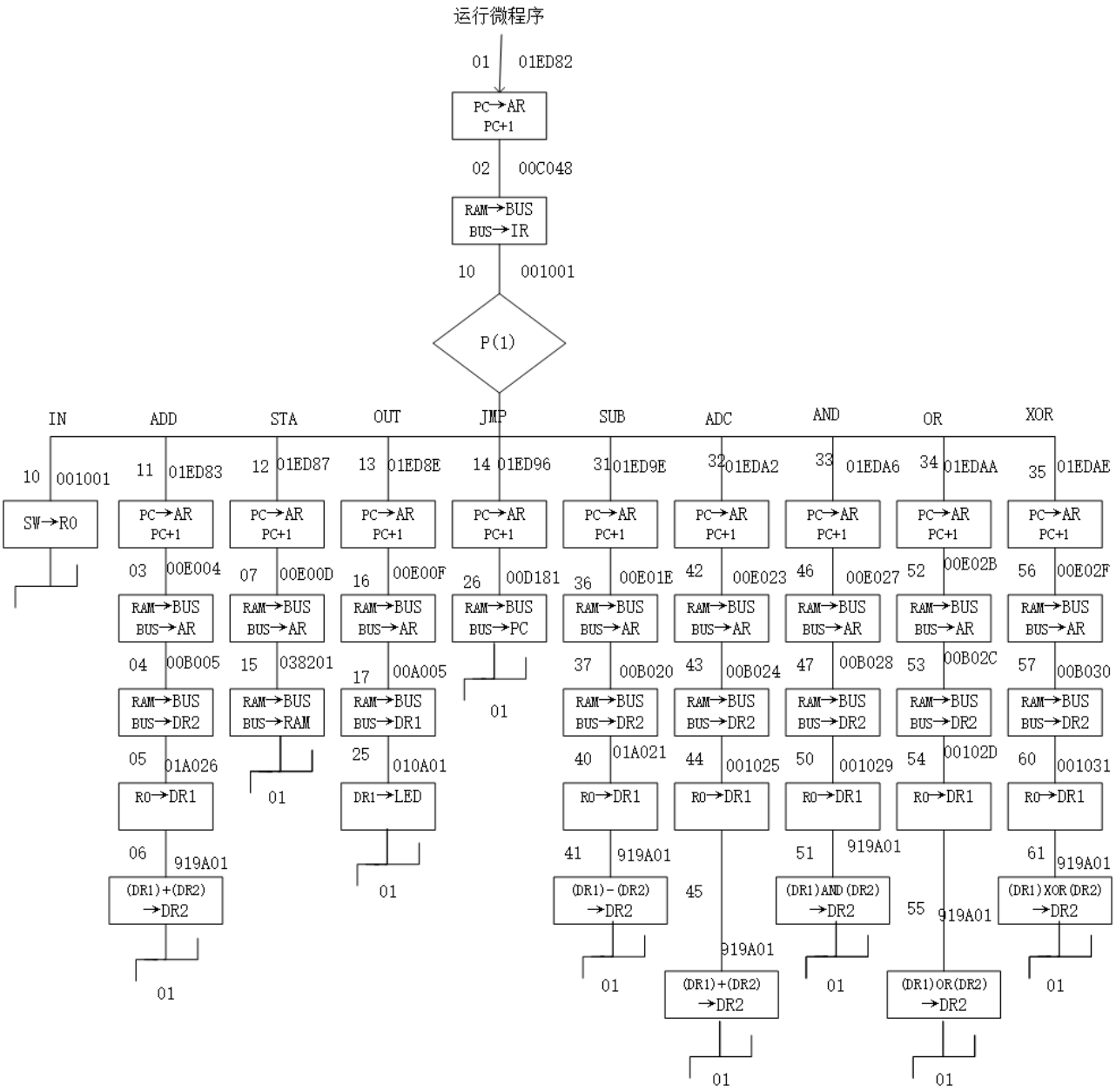


图 6.8: 思考题答案微程序流程图

微指令代码表如下:

微地址 (OCT)	助记符	微指令 (HEX)
10	IN	001001
11	ADD	01ED83
12	STA	01ED87
13	OUT	01ED8E
14	JMP	01ED96
31	SUB	01ED9E
31	ADC	01EDA2
33	AND	01EDA6
34	OR	01EDAA
35	XOR	01EDAE

表 6.1: 思考题答案微指令代码表

第七章　K8051 单片机核的构建和相关设计实验

7.1　具体应用描述

在本次实验中，我们使用 K8051 单片机核进行音乐播放器的构建和测试。
设计时设定 K8051 晶振频率为 12 MHz，依此计算延迟所需的指令周期。
实现了音乐的顺序播放与暂停功能，有瑕疵地[1]完成了音乐切换的功能。

7.2　Quartus 下硬件设计原理图、模式及引脚说明

7.2.1　硬件设计原理图

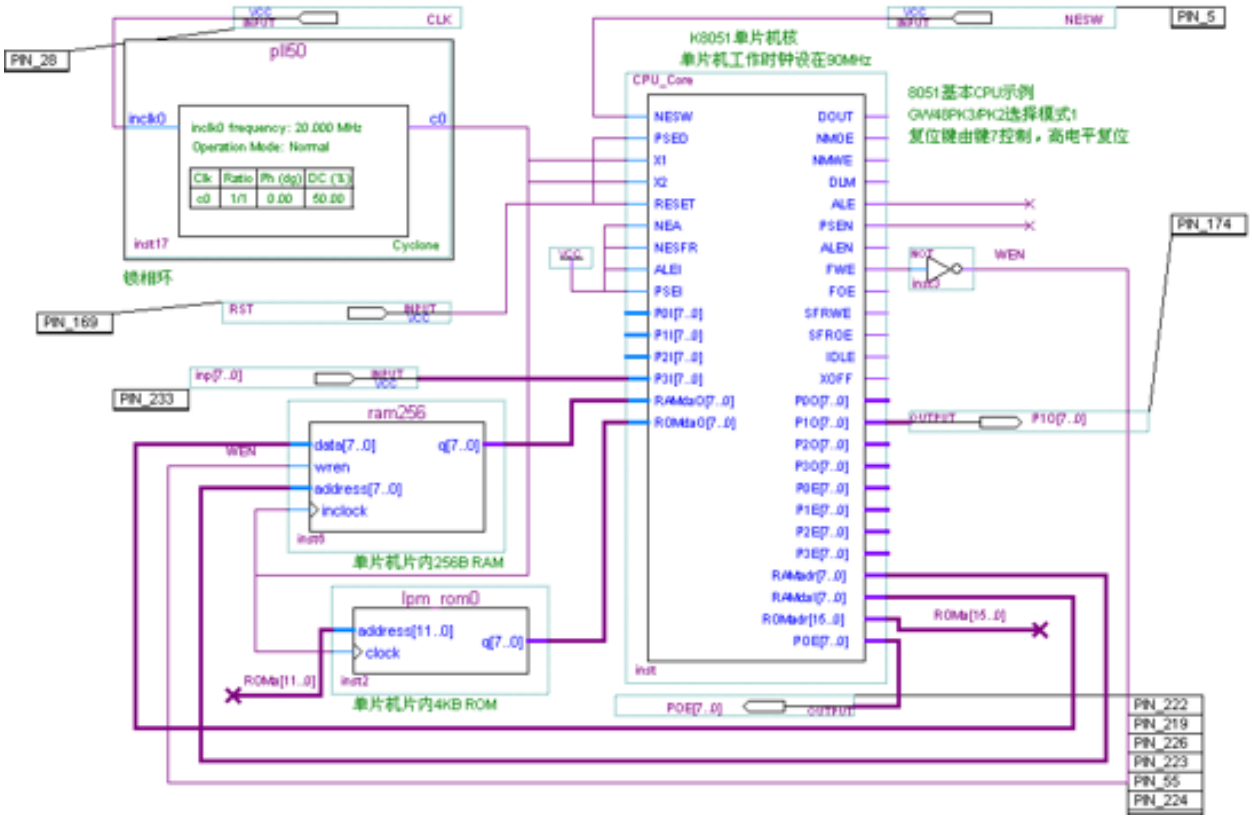


图 7.1: 硬件设计原理图

7.2.2　模式及引脚说明

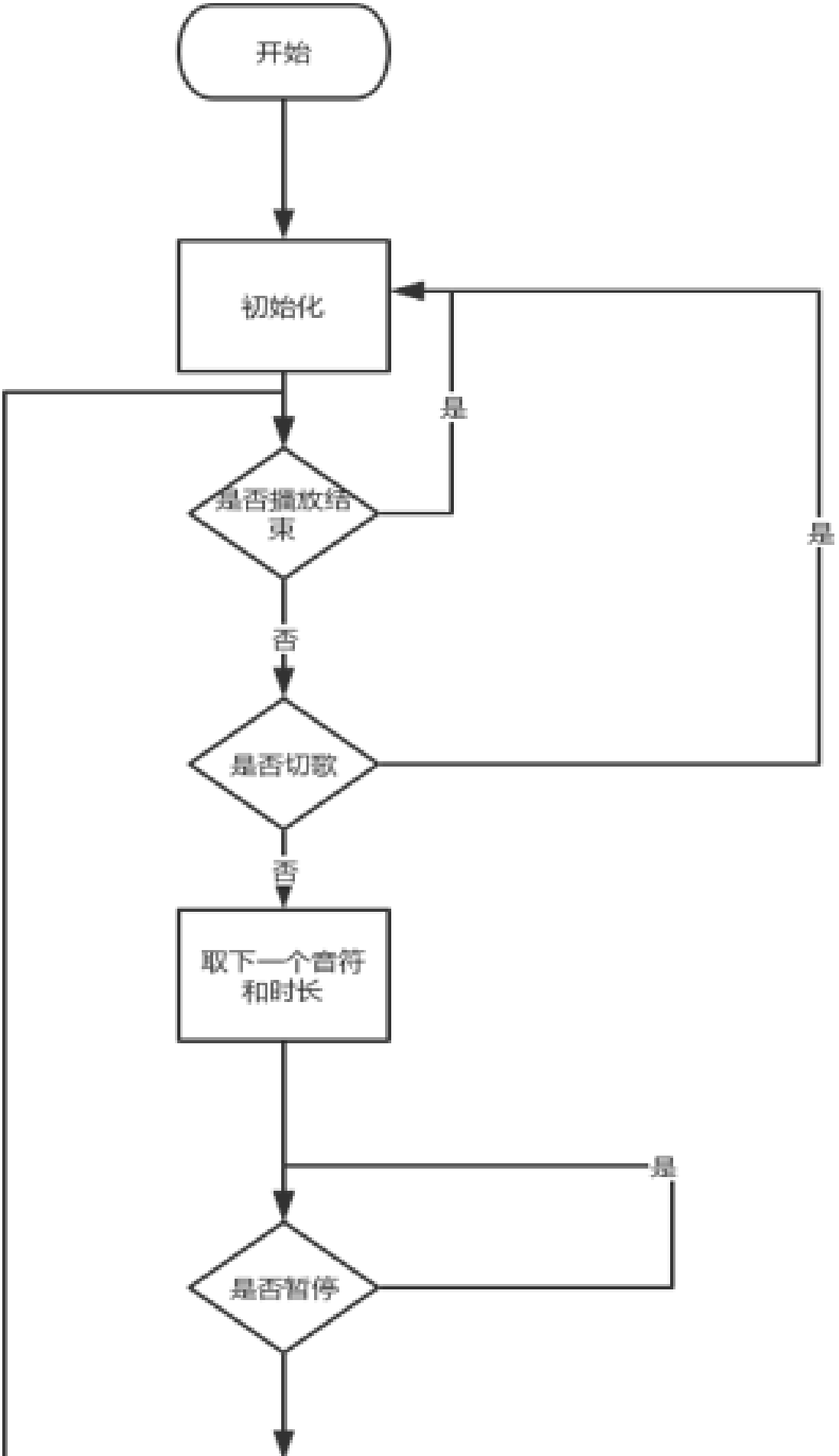
工作模式：1

[1]因此代码中不包含音乐切换的功能

描述	引脚	备注
时钟	28	CLOCK0
复位键	169	Key 7 (D15)
暂停键	233	Key 1 (数码 1)
切歌键	237	Key 2 (数码 2)
蜂鸣器	174	SPK

表 7.1: 音乐播放器：引脚表

7.2.3　软件设计流程图及相关描述



7.2.4　延时计算公式

已知 8051 晶振频率为 F (Hz)，每个指令需要 12 个机器周期，因此指令频率为 $\frac{F}{12}$
对于一个音符，其有频率 f (Hz) 与节拍数 p (单位是 1/8 节拍)。
需要事先设定一类音符的两个值：每个 1/8 拍中含有的方波个数 s 与方波周期参数 t。
根据代码统计分析（具体见代码批注）得如下方程

$$6f(48pst + 40ps + 36p + 9) = psF$$

忽略左边的小量 $54f$ 后可以消去 p 得到

$$24f(12st + 10s + 9) = sF$$

再忽略左边的小量 $216f$ 后可以消去 s 得到

$$48f(6t + 5) = F$$

这样就得到了 t 的近似计算公式：

$$t = \frac{\frac{F}{48f} - 5}{6} \tag{1}$$

例如对于 $F = 12,000,000$，$f = 440$，可以计算得 $t = 94$。

7.3　汇编源代码

代码 7.1: music_8051.asm

```
1  ; @ref: music90/ASM/Music1.ASM
2  ; @analyst: zccz14
3  ; @note:
4  ; CPU_FREQUENCY = 12MHz (8051 晶振频率)
5  ; 每个机器周期(Cycle) 消耗 12 个时钟周期
6  ; 真实时间消耗 T = 12 * Cycles / CPU_FREQUENCY
7  ; 太不清真了这个代码，亟需重构。
8  ; 出现了很多"参数穿透多层过程使用"的现象。
9  ; 子过程划分逻辑混乱，不合语义。
10 ; @diff:
11 ;   - 修改了注释
12 ;   - 汇编助记符改为小写
13 ;   - 在二元运算符(+, *) 两端加了空格，在逗号后面加了空格
14 ;   - 调整了缩进
15 ;   - 添加了暂停功能（通过键 0 控制）
16
17 ; OUTPUT
18 SPK EQU P1.0; SPK: PIN_174 (SPKER)
19 ; INPUT
20 PAUSE EQU P3.0; PAUSE: PIN_233 (Key 0)
21
```

```
22 org 0000H; code segment align
23 jmp START; goto START => 2 cycles
24
25 ; @name: START
26 ; @desc: 主程序
27 ; @param: TABLE {Array} - 音符对应以下两表的偏移量表
28 ; @param: S_PARA {Array} - 方波个数表
29 ; @param: DELAY_T {Array} - 方波周期参数表
30 ; @write: R0, R1, R2, R3, R4, R5, R7, A
31 ; @cycle: Infinity
32 START:
33   mov R3, #00H; => 1 cycle
34   NEXT:
35     mov A, R3; => 1 cycle
36     mov DPTR, #TABLE; => 2 cycles
37     movc A, @A + DPTR; => 2 cycles
38     jz START; => 2 cycles
39     mov R7, A; => 1 cycle
40     inc R3; => 1 cycle
41
42     mov A, R3; => 1 cycle
43     movc A, @A + DPTR; => 2 cycles
44     mov R2, A; => 1 cycle
45
46     acall SONG; => ... @see {SONG}
47     inc R3; => 1 cycle
48     sjmp NEXT; => 2 cycles
49
50 ; @name: SONG
51 ; @desc: 产生一个音符的信号
52 ; @param: R2 - 此音长为 R2 个八分之一拍 (实际上还有蜜汁乘2)
53 ; @param: DELAY_T {Array} - 方波周期参数表
54 ; @param: S_PARA {Array} - 方波个数表
55 ; @param: R7 - 表偏移量
56 ; @write: R0, R1, R2, R4, R5, A
57 ; @alias:
58 ;   - [0]: S_PARA[R7]
59 ;   - [1]: DELAY_T[R7]
60 ; @cycle: R2 * [0] * [1] * 48 + R2 * [0] * 40 + R2 * 36 + 9 (if R2 != 0)
61 ; @cycle: [0] * [1] * 24 + [0] * 20 + 28 (only if R2 = 0)
62 ; @note: 音符时长 t = 12 * @cycle / CPU_FREQUENCY
63 ; @note: 音符频率 f = (R2 * 2 * [0]) / t
64 ; @note: 这个明明叫 SONG 的过程竟然只能播放一个音符？
65 ; @note: 可以看到这里把 R2 放到 SONG 的内部处理很不清真，失去了语义
```

```
66  SONG:; call this => 2 cycles
67    mov A, R2; 1 cycle
68    rl A; 1 cycle
69    jnz KEEP; 2 cycles
70      mov A, #01H; 1 cycle
71    KEEP:; branch => 2 cycles (+1 only if R2 = 0)
72    mov R2, A; 1 cycle
73    REPEAT:; [0] * [1] * 24 + [0] * 20 + 18 cycles for each, hit R2 * 2 (1 only if R2 =
74      acall EIGHTH; => [0] * [1] * 24 + [0] * 20 + 16 cycles
75      djnz R2, REPEAT; => 2 cycles
76    ret; return => 2 cycles
77
78  ; @name: EIGHTH
79  ; @desc: 产生 1/8 拍的方波信号
80  ; @param: DELAY_T {Array} - 方波周期参数表
81  ; @param: S_PARA {Array} - 方波个数表
82  ; @param: R7 - 表偏移量
83  ; @write: R0, R1, R4, R5, A
84  ; @cycle: S_PARA[R7] * DELAY_T[R7] * 24 + S_PARA[R7] * 20 + 16
85  ; @note: 方波个数 n = S_PARA[R7]
86  ; @note: 八分之一拍时长 t = 12 * @cycle / CPU_FREQUENCY
87  ; @note: 频率 f = n / t
88  ; @note: 其实我个人认为在这个过程中查表很不清真
89  EIGHTH:; call this => 2 cycles
90    ; R4 := DELAY_T[R7]
91    mov A, R7; => 1 cycle
92    mov DPTR, #DELAY_T; => 2 cycles
93    movc A, @A + DPTR; => 2 cycles
94    mov R4, A; 1 cycle
95    ; R5 := S_PARA[R7]
96    mov A, R7; => 1 cycle
97    mov DPTR, #S_PARA; => 2 cycles
98    movc A, @A + DPTR; => 2 cycles
99    mov R5, A; => 1 cycle
100   NEXTCYC:; R4 * 24 + 20 cycles each loop, hit R5 times => R5 * R4 * 24 + R5 * 20 cycl
101     jb PAUSE, $; while (!PAUSE);
102     acall SOUND; => R4 * 24 + 18 cycles
103     djnz R5, NEXTCYC; => 2 cycles
104   ret; return => 2 cycles
105
106 ; @name: SOUND
107 ; @desc: 产生一个方波脉冲信号
108 ; @param: R4 - 正线性相关于方波周期
109 ; @write: R0, R1, A
```

```
110 ; @cycle: R4 * 24 + 18 (方波周期)
111 ; @note: 方波占空比为 50 %
112 ; @note: 上升、衰减时间为 1 cycle
113 SOUND:; => call this => 2 cycles
114   setb SPK; => 1 cycle
115   acall SDELAY; => R4 * 12 + 6 cycles
116   clr SPK; => 1 cycle
117   acall SDELAY; => R4 * 12 + 6 cycles
118   ret; return => 2 cycles
119
120 ; @name: SDELEY
121 ; @desc: Soft Delay (软延时)
122 ; @param: R4 - 正线性相关与延时时长
123 ; @write: R0, R1, A
124 ; @cycle: R4 * 12 + 6
125 SDELAY:; call this => 2 cycles
126   mov A, R4; => 1 cycle
127   mov R0, A; => 1 cycle
128   XL2:; 12 cycles each loop, hit R4 times => R4 * 12 cycles
129     mov R1, #03H; => 1 cycle
130     DL1:; 3 cycles each loop, hit 3 times => 9 cycles
131       nop; => 1 cycle
132       djnz R1, DL1; => 2 cycles
133     djnz R0, XL2; => 2 cycles
134   ret; return => 2 cycles
135
136 ; @name: S_PARA
137 ; @desc: 八分之一拍中含有的方波个数表
138 ; @note: 表中有一些空值(FFH, 00H) 是无效值, 实际上也没有用到
139 S_PARA:
140 ds 1DH
141 db 15H, 16H, 00H
142 db 19H, 00H, 1CH, 00H, 1FH, 21H, 00H, 25H
143 db 00H, 29H, 2CH, 00H, 31H, 34H, 37H, 00H
144 db 3EH, 41H, 00H, 49H, 00H, 52H, 57H, 00H
145 db 62H
146
147 ; @name: DELAY_T
148 ; @desc: 方波周期表
149 ; @note: 表中有一些空值(FFH, 00H) 是无效值, 实际上也没有用到
150 DELAY_T:
151 ds 1DH
152 db 7EH, 77H, 00H
153 db 6AH, 00H, 5EH, 00H, 54H, 4FH, 00H, 46H
```

```
154 db 00H, 3FH, 3BH, 00H, 35H, 32H, 2FH, 00H
155 db 2AH, 27H, 00H, 23H, 00H, 1FH, 1DH, 0C0H
156 db 1AH
157
158
159 ; @name: TABLE
160 ; @desc: 《龙卷风》 的谱子编码
161 ; @note: 每个字对应一个音符
162 ; @note: 每个字的高字节对应了 S_PARA 与 DELAY_T 中的偏移量
163 ; @note: 每个字的低字节对应了音符的节拍长度
164 TABLE:
165 DW 2008H, 2708H, 2704H, 2504h, 2908H
166 DW 2008H, 2708H, 2704H, 2504H, 2908H
167 DW 2208H, 2908H, 2904H, 2704H, 2A08H
168 DW 2208H, 2906H, 2A02H, 2904H, 2704H, 2504H, 2202H, 2002H
169 DW 2008H, 2708H, 2704H, 2504H, 2908H
170 DW 2008H, 2708H, 2704H, 2504H, 2908H
171 DW 2208H, 2908H, 2904H, 2704H, 2A08H
172 DW 2208H, 2906H, 2A02H, 2904H, 2704H, 2504H, 2C02H, 2D02H
173 DW 2D04H, 2904H, 2904H, 2504H, 2504H, 2204H, 2504H, 2C04H
174 DW 2C04H, 2704H, 2704H, 2404H, 2404H, 2004H, 2404H, 2504H
175 DW 2208H, 2504H, 2204H, 2504H, 2202H, 2502H, 2502H, 2702H, 2902H, 2A02H
176 DW 2A02H, 2704H, 2708H, 2908H, 2A08H
177
178 DW 2C03H, 2A03H, 2903H, 2903H, 2A03H, 2C03H, 2C03H, 2A03H, 2903H, 2903H, 2502H
179
180 DW 2C03H, 2A03H, 2903H, 2903H, 2A03H, 2C03H, 2C03H, 2A03H, 2903H, 2903H, 2501H, 2701H
181
182 DW 2904H, 2C04H, 2C04H, 2502H, 2702H, 2904H, 2C04H, 2C04H, 2502H, 2702H
183 DW 2504H, 2704H, 2504H, 2704H, 2904H, 2702H, 2902H, 2A04H, 2902H, 2A02H
184
185 DW 2C03H, 2A03H, 2903H, 2903H, 2A03H, 2C03H, 2C03H, 2A03H, 2903H, 2903H, 2502H
186
187 DW 2C03H, 2A03H, 2903H, 2903H, 2A03H, 2C03H, 2C03H, 2A03H, 2903H, 2903H, 2501H, 2701H
188 DW 2904H, 2C04H, 2C04H, 2502H, 2702H, 2904H, 2C04H, 2C04H, 2502H, 2902H
189 DW 2504H, 2704H, 2504H, 2704H, 2904H, 2A04H, 2C04H, 3004H
190
191 DW 3020H
192 DW 0000H
193
194 end
```

7.4　调试总结

调试期间出现了以下问题：

- 时间紧迫，速成 8051 汇编
  - 对指令作用理解有偏差
  - 不恰当的代码设计模式
- 引脚绑定错误

  迅速被调通。
- 程序中汇编指令逻辑错误

  这是常有的事情，让人充分认识到结构化程序设计里程碑式的发展。