时序电路

时序电路

任意时刻的输出不仅取决于当时的输入信号，而且还取决于电路原来的状态，或者说，还与以前的输入有关。

时序电路特点：

1. 通常包含组合电路和存储电路，存储电路必不可少
2. 存储电路的输出状态必须反馈到组合电路的输入端，与输入信号一起，共同决定逻辑电路的输出。

时序电路的存储电路

触发器——具有记忆功能的基本逻辑单元。

时序电路划分：

1. Mealy型——输出信号不仅取决于存储电路的状态，而且还取决于输入变量。
2. Moore型——输出信号仅取决于存储电路的状态。

时序电路中的关键信号

- CLK信号：描述CLK信号的两种模式

1. 时钟信号作为进程的敏感信号
   时钟信号显式地出现在进程语句PROCESS后的敏感信号列表中
2. 利用WAIT ON语句等待时钟变化

程序举例—触发器（最简单的**D触发器**）

```
Flipflop1 : process(clk)
begin
if clk='1' and clk'event then
Q <= D;
end if;
end process;
```

注意：进程敏感表只包括时钟信号

```
Flipflop1 : PROCESS
  BEGIN
 wait until clk'event and clk='1';
     Q<=D;
END PROCESS;
```

时序电路中的关键信号

- 复位信号：

1.同步复位

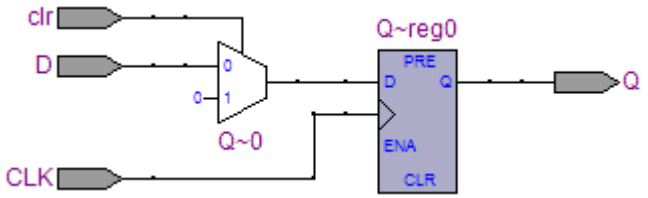
- 用if语句说明复位条件

2.异步复位

- 敏感信号中含复位信号reset ；
- 利用if语句描述复位条件；
- 利用elsif段来描述时钟信号的边沿条件;

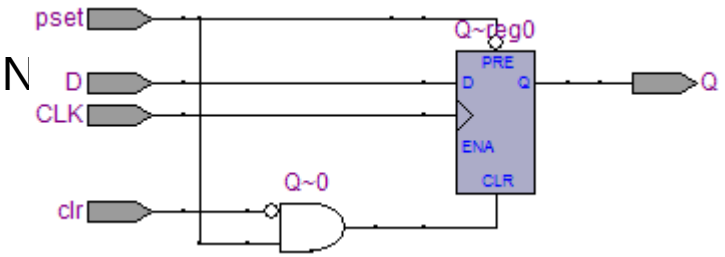
同步复位的D触发器

```
PROCESS(CLK)
 BEGIN
  IF (CLK'EVENT AND CLK='1')THEN
   IF (clr='1')THEN
     Q <= '0' ;
   ELSE
              Q <= D ;
   END IF;
  END IF;
  END PROCESS;
```



异步复位的D触发器

```
PROCESS(CLK,PSET,CLR) IS
BEGIN
IF(PSET='0')THEN     --置位信号为1,则触发器被置位
    Q<='1';
ELSIF(CLR='0')THEN --复位信号为1,则触发器被复位
    Q<='0';
ELSIF(CLK'EVENT AND CLK='1')THEN
     Q<=D;
  END IF;
  END PROCESS;
```



状 态 机 设 计

时序电路也称为状态机。

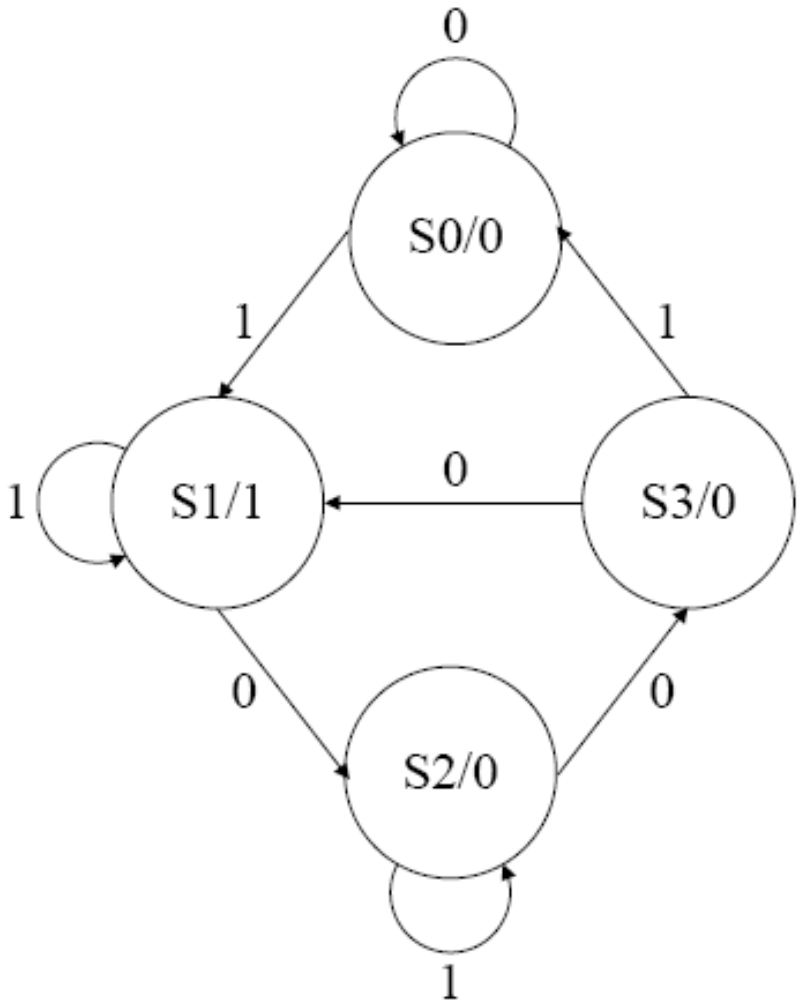
- Moore和Mealy状态机的特点：

– Moore状态机：输出会在一个完整的时钟周期内保持其稳定值，既使在该时钟周期内输入信号有变化，输出也不会发生变化。输入对输出的影响要到下一个时钟周期才能反映出来。把输入与输出隔离开来，是Moore状态机的一个重要特点。

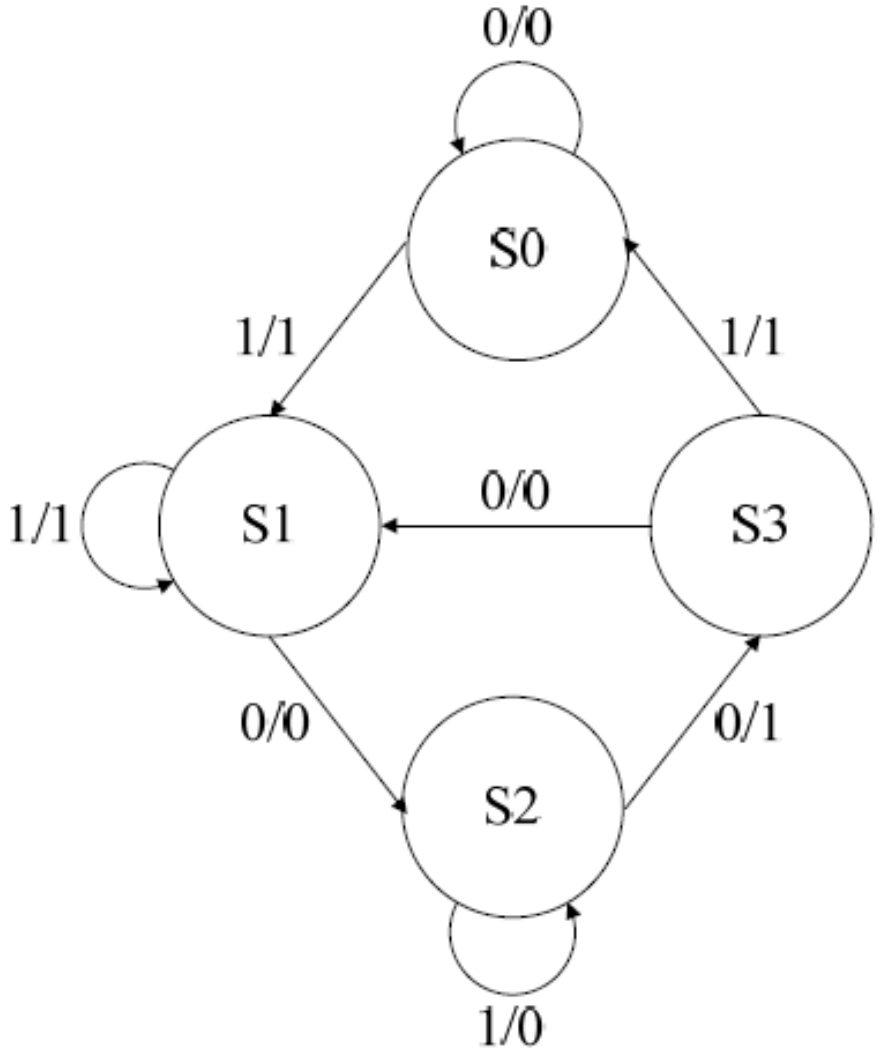
– Mealy状态机：输出直接受输入影响，由于输入变化可能出现在时钟周期内的任何时刻，使得Mealy状态机对输入的响应可以比Moore状态机对输入的响应早一个时钟周期。输入信号中的噪声可能出现在输出端。

– 实现同样的功能，Moore状态机所需的状态个数可能比Mealy状态机多

状态转换图



Moore 状态机



Mealy 状态机

描述状态机方法

- 方法1：

  缺点：看不见状态转换图。

```
process(clk)
begin

    if clk'event and clk='1' then
        case state is:
            when "000" =>
                If 条件1 then
                    state <= "001";
                end if;
            when ……
        end case;
    end if;

end process
```

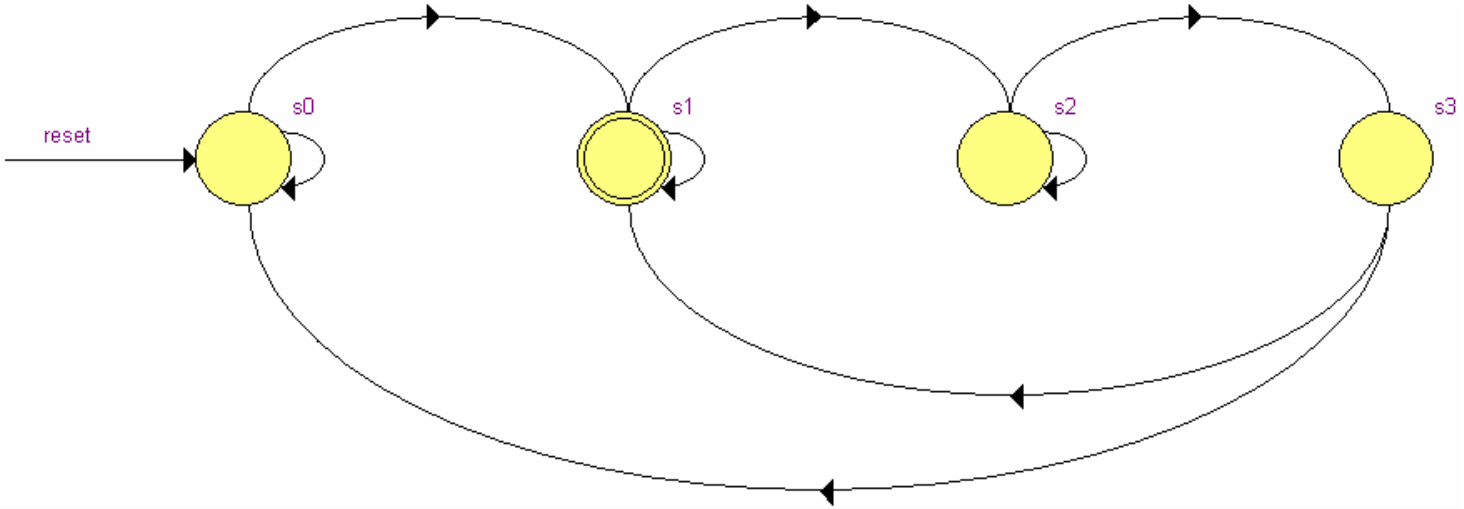
描述状态机方法

- 方法2：使用TYPE STATE IS (xx, xx，……);

SIGNAL PresentState : STATE;
SIGNAL NextState : STATE;

描述状态机。

优点：结构清晰，可看出状态转换图。



	Source State	Destination State	Condition
1	s1	s1	(DIN)
2	s1	s2	(!DIN)
3	s2	s2	(DIN)
4	s2	s3	(!DIN)
5	s3	s1	(!DIN)
6	s3	s0	(DIN)
7	s0	s1	(DIN)
8	s0	s0	(!DIN)

TYPE STATE IS的三种描述方式：

- **单进程描述方式：**

  - 用一个进程描述有限状态机中的次态逻辑、状态寄存器和输出逻辑。

- **双进程描述方式：**

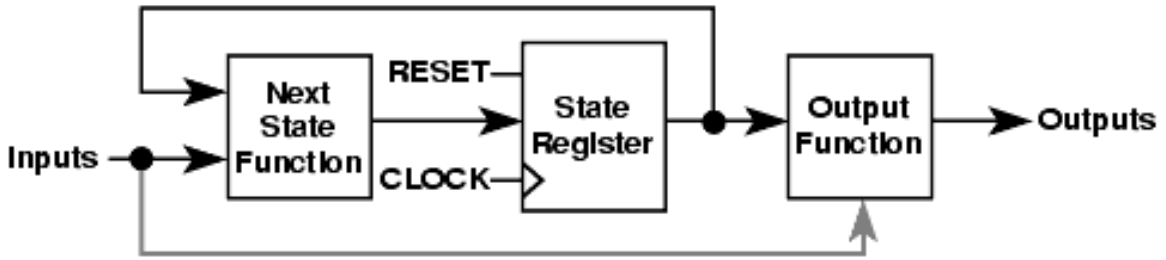
  - 进程1描述次态逻辑、状态寄存器和输出逻辑中的任何两个
  - 进程2描述剩余的功能。

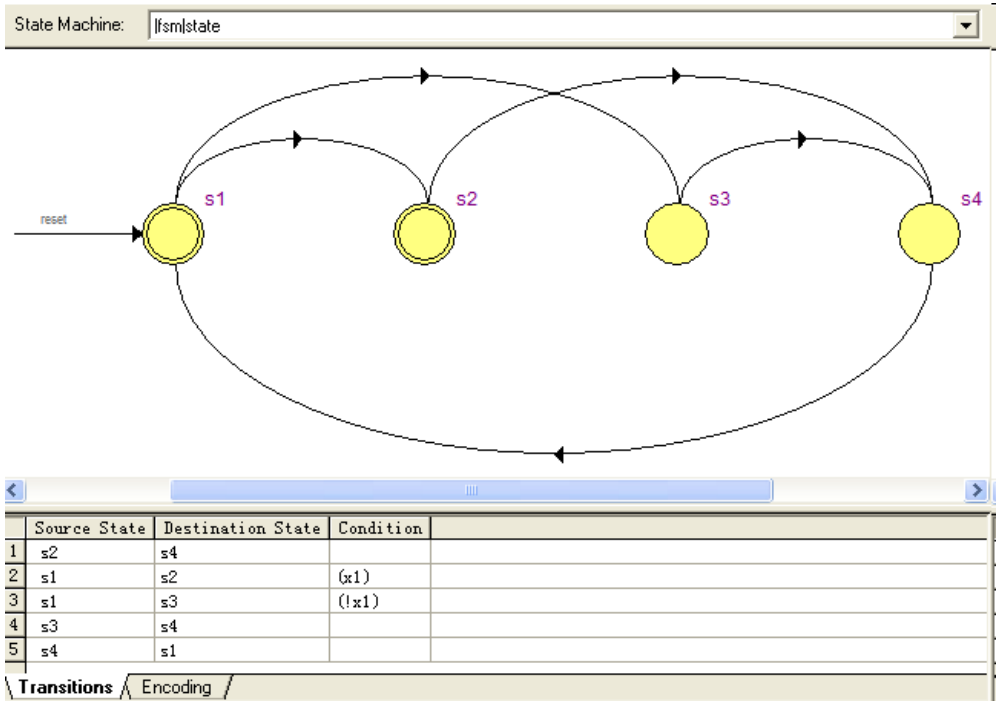
- **三进程描述方式：**

  - 进程1描述次态逻辑
  - 进程2描述状态寄存器
  - 进程3描述输出逻辑

程序举例—单进程描述

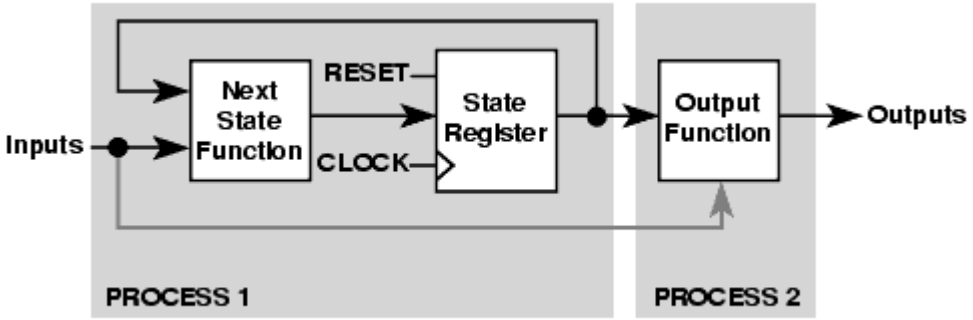
```
library IEEE;
use IEEE.std_logic_1164.all;
entity fsm is
  port ( clk, reset, x1 : IN std_logic;
                 outp : OUT std_logic);
end entity;
architecture beh1 of fsm is
  type state_type is (s1,s2,s3,s4);
  signal state: state_type ;
begin
  process (clk,reset)
  begin
    if (reset ='1') then
      state <=s1; outp<='1';
    elsif (clk='1' and clk'event) then
      case state is
        when s1 => if x1='1' then state <= s2;
                   else          state <= s3;
                   end if;
                   outp <= '1';
        when s2 => state <= s4; outp <= '1';
        when s3 => state <= s4; outp <= '0';
        when s4 => state <= s1; outp <= '0';
      end case;
    end if;
  end process;
end beh1;
```



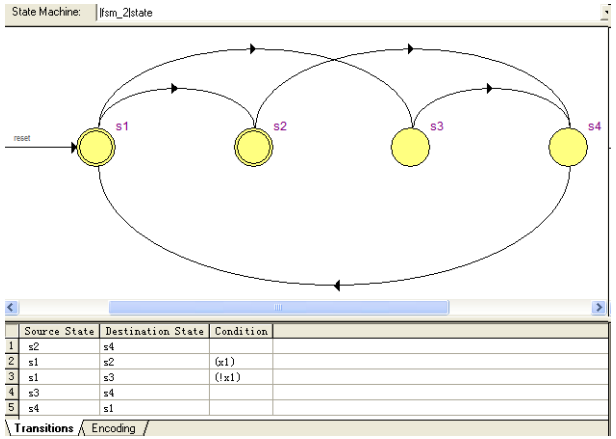


程序举例—双进程描述

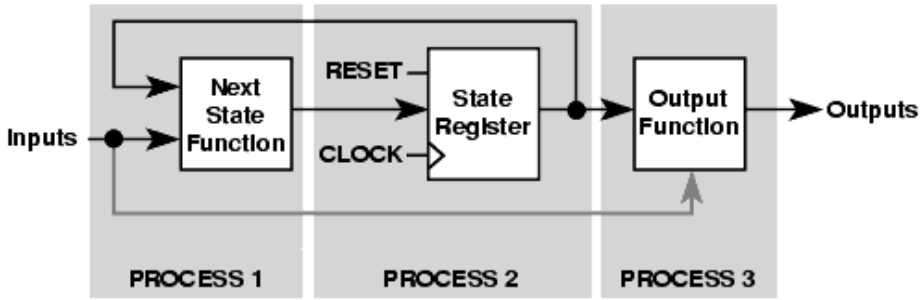
```
library IEEE;
use IEEE.std_logic_1164.all;

entity fsm is
  port ( clk, reset, x1 : IN std_logic;
                 outp : OUT std_logic);
end entity;
architecture beh1 of fsm is
  type state_type is (s1,s2,s3,s4);
  signal state: state_type ;
begin
  process1: process (clk,reset)
  begin
    if (reset ='1') then state <=s1;
    elsif (clk='1' and clk'Event) then
      case state is
        when s1 => if x1='1' then state <= s2;
                   else           state <= s3;
                   end if;
        when s2 => state <= s4;
        when s3 => state <= s4;
        when s4 => state <= s1;
      end case;
    end if;
  end process process1;
```




```
process2 : process (state)
  begin
    case state is
      when s1 => outp <= '1';
      when s2 => outp <= '1';
      when s3 => outp <= '0';
      when s4 => outp <= '0';
    end case;
end process process2;
end beh1;
```



程序举例—三进程描述




```
library IEEE;
use IEEE.std_logic_1164.all;

entity fsm is
  port ( clk, reset, x1 : IN std_logic;
                   outp : OUT std_logic);
end entity;
architecture beh1 of fsm is
  type state_type is (s1,s2,s3,s4);
  signal state, next_state: state_type ;
begin
  process1: process (clk,reset)
  begin
    if (reset ='1') then
      state <=s1;
    elsif (clk='1' and clk'Event) then
      state <= next_state;
    end if;
  end process process1;
```

```
process2 : process (state, x1)
  begin
    case state is
        when s1 => if x1='1' then
                     next_state <= s2;
                   else
                     next_state <= s3;
                   end if;
        when s2 => next_state <= s4;
        when s3 => next_state <= s4;
        when s4 => next_state <= s1;
    end case;
end process process2;

process3 : process (state)
  begin
      case state is
        when s1 => outp <= '1';
        when s2 => outp <= '1';
        when s3 => outp <= '0';
        when s4 => outp <= '0';
      end case;
end process process3;
end beh1;
```

作业：投币式手机充电仪

基于FPGA实验板设计一台投币式手机充电仪的控制电路，可以实现投币、实时显示投币数额和充电时间等功能。

要求使用硬件描述语言设计底层各功能模块电路，其中控制电路必须使用状态机设计；顶层电路设计方式不限，即语言或原理图方式均可。

1	5	9	确认
2	6	0	
3	7	开始	
4	8	清零	

矩阵键盘示意图

具体操作过程如下：

1. 刚上电即“初始状态”，数码管显示全灭。
2. 按“开始”键后进入准备投币状态，数码管显示“0000”。
3. 矩阵键盘可直接输入投币数额 1～20 角，并实时显示在左侧 2 位数码管上。

例如：输入 5 角（按下键 5）时，数码管显示 05；输入 15 角（先后按下键 1、键 5）时，数码管上先后显示 01、15。键盘中各键所代表的数字和功能如上图所示。

4. 投币（按下数字键）时，2 倍于投币数额的允许充电时间实时显示在右侧 2 位数码管上。
5. 输入大于 20 角，均显示 20；则充电时间最多显示 40。

例如：先后输入 4、5，金额先后显示 04、20，时间先后显示 08、40。

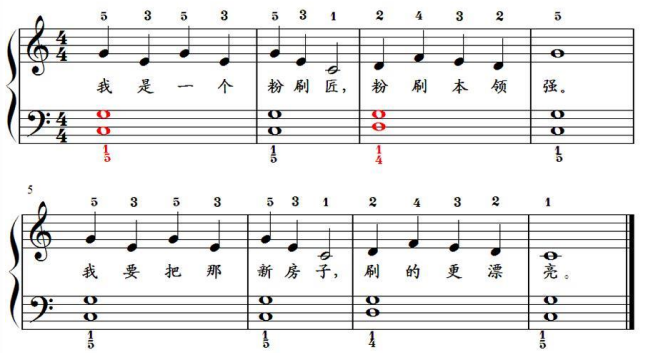
6. 未确认充电之前可随时“清零”。清零回至“开始状态”，10 秒无动作回到“初始状态”。
7. 确认充电后，充电时间（最多从 40）开始倒计时，此时投币数额仍保持显示；当时间计至 0 时，投币数额也同时归 0，回到“开始状态”，10 秒后回到“初始状态”。
8. 按照正常的输入，不会先输入 0。若先后输入 1、2、3，可认定输入为 12 或是 23，由设计者确定。
9. 其他未明确说明的要求处理原则一致、合理。

设计一个蜂鸣器驱动电路，可以播放一段音乐。要求：充电结束倒计时归零时播放音乐，期间若有投币动作音乐停止进入“投币状态”，若无动作音乐结束后回到“初始状态”。

C 调各音符频率对照表

音符	频率 Hz	周期 μs
低 1Do	262	3816
低 2Re	294	3401
低 3Mi	330	3030
低 4Fa	349	2865
低 5So	392	2551
低 6La	440	2272
低 7Si	494	2024
中 1Do	523	1912
中 2Re	587	1703
中 3Mi	659	1517
中 4Fa	698	1432
中 5So	784	1275
中 6La	880	1136
中 7Si	988	1012
高 1Do	1047	955
高 2Re	1175	851
高 3Mi	1319	758
高 4Fa	1397	751
高 5So	1568	637
高 6La	1760	568
高 7Si	1967	508

粉刷匠



作业：投币式手机充电仪

- 分解任务(扫描显示和扫描输入)

1.矩阵键盘的连续输入

2. 数码管的移位显示功能

3. 控制电路设计

……

- 关注细节

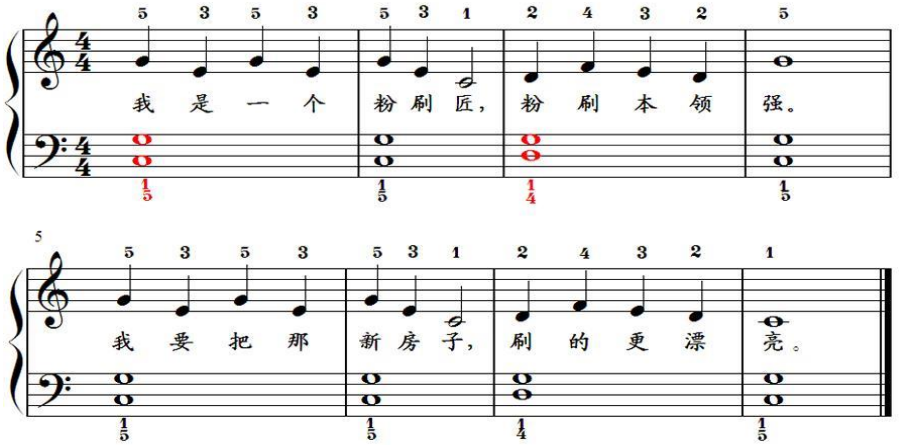
例如：键盘、数码管的扫描频率；时序电路中clk的频率选取等。

音乐铃声设计

C 调各音符频率对照表

音符	频率 Hz	周期 μs
低 1Do	262	3816
低 2Re	294	3401
低 3Mi	330	3030
低 4Fa	349	2865
低 5So	392	2551
低 6La	440	2272
低 7Si	494	2024
中 1Do	523	1912
中 2Re	587	1703
中 3Mi	659	1517
中 4Fa	698	1432
中 5So	784	1275
中 6La	880	1136
中 7Si	988	1012
高 1Do	1047	955
高 2Re	1175	851
高 3Mi	1319	758
高 4Fa	1397	751
高 5So	1568	637
高 6La	1760	568
高 7Si	1967	508

粉刷匠



意大利文	释义	每分钟拍数
grave	庄板	40左右
largo	广板	46左右
lento	慢板	52左右
larghetto	小广板	56左右
adagio	柔板	60左右
adagietto	小柔板	66左右
andante	行板	72左右
andantino	小行板	80左右
maestoso	庄严的	88左右
moderato	中板	96左右
allegretto	小快板	108左右
animato	活跃的	120左右
allegro	快板	132左右
vivace	极快板	160左右
presto	急板	184左右
prestissimo	最急板	208左右

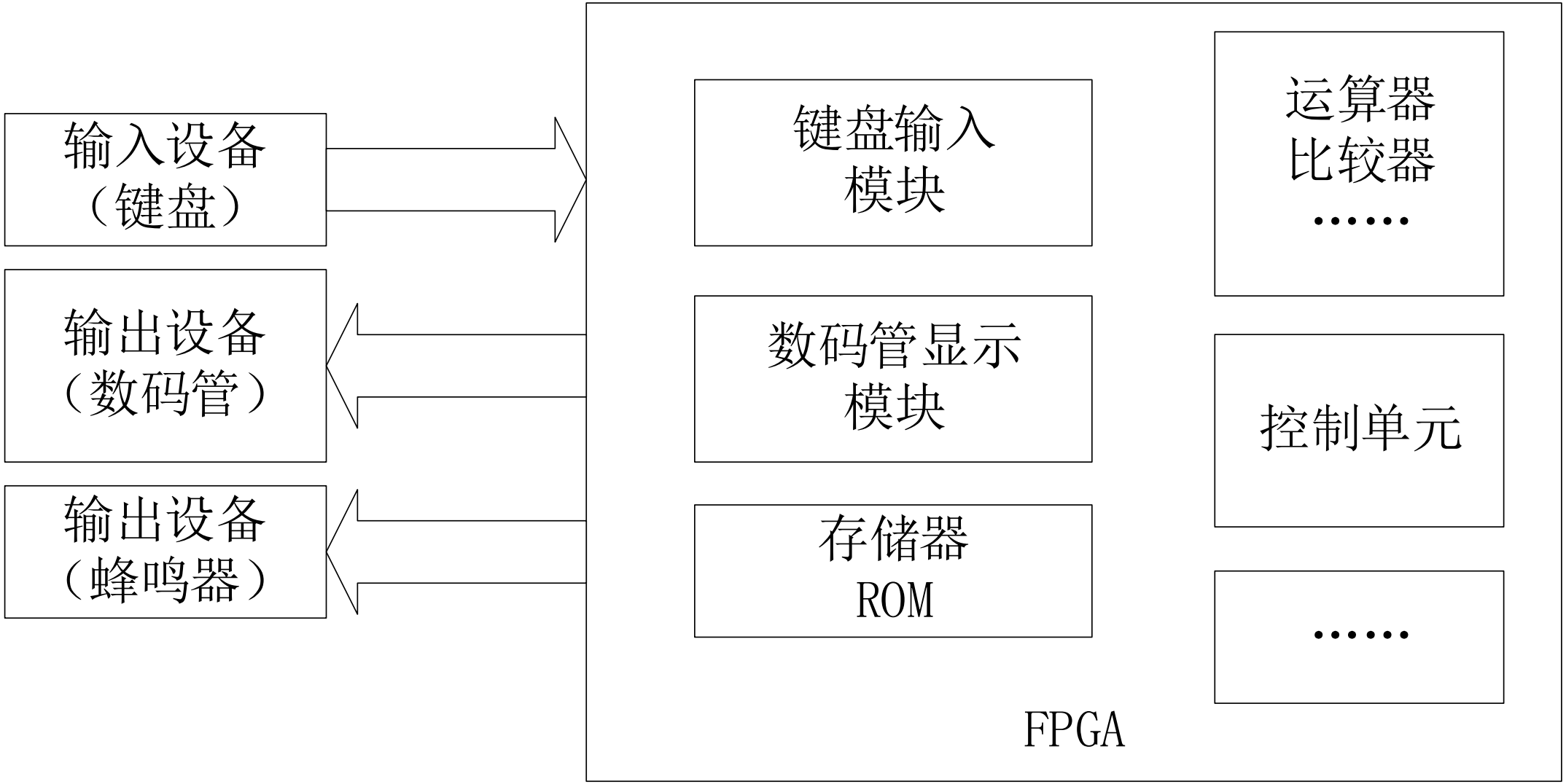
在音乐中，时间被分成均等的基本单位，每个单位叫做一个“拍子”或 称一拍。拍子的时值是以音符的时值来表示的，一拍的时值可以是四分音符（即以四分音符为一拍），也可以是二分音符（以二分音符为一拍）或八分音符（以八分音符为一拍）。拍子的时值是一个相对的时间概念，比如当乐曲的规定速度为每分钟 60 拍时，每拍占用的时间是一秒，半拍是二分之一 秒； 当规定速度为每分钟 120 拍时，每拍的时间是半秒，半拍就是四分之一 秒，依此类推。拍子的基本时值确定之后，各种时值的音符就与拍子联系在一起。例如，当以四分音符为一拍时，一个全音符相当于四拍，一个二分音符相当于两拍， 八分音符相当于半拍，十六分音符相当于四分之一拍；如果以八分音符做为 一拍，则全音符相当于八拍，二分音符是四拍，四分音符是两拍，十六分音 符是半拍。

音符时值的长度是相对的，只有倍数关系，没有绝对时间。比如稍快的曲子，全音符大概两秒，二分音符就是一秒，四分音符就是半秒。稍慢的曲子，全音符大概四秒，二分音符就是两秒，四分音符就是一秒。

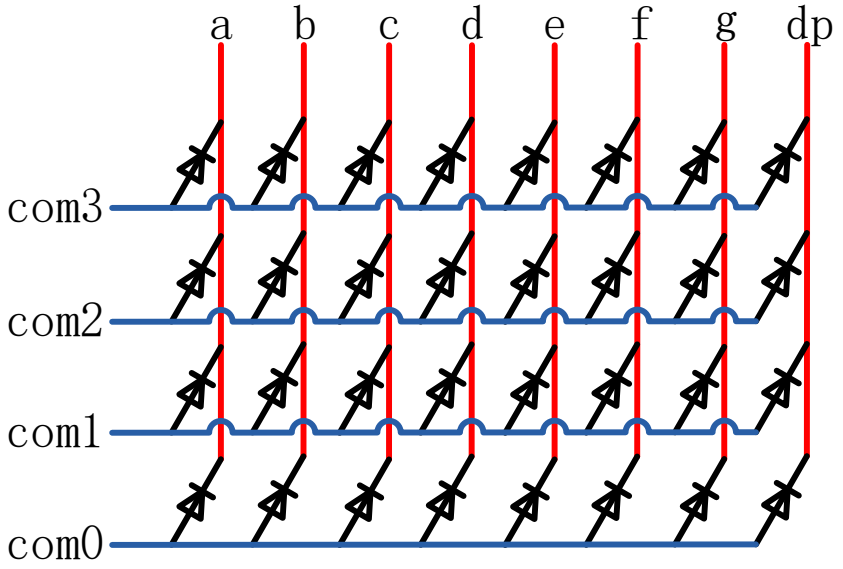
一般的简谱前面写着1=4/4，有的还写着♩=76，就是指以四分音符为一拍，每分钟76拍，那么算一下就知道一个四分音符时长就是0.789秒了。另外二分音符是四分音符时长的二倍。

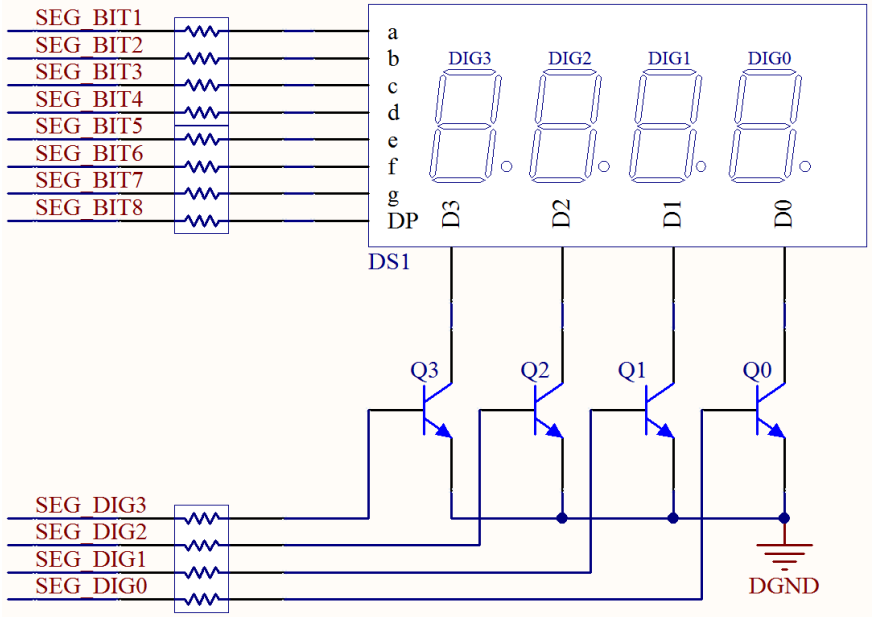
Addr	+0	+1	+2	+3	+4	+5	+6	+7	ASCII
0	5	3	5	3	5	3	1	2	.....
8	4	3	2	5	5	5	5	5	.....
16	3	5	3	5	3	1	2	4	.....
24	3	2	1	1	1	1	0	0	.....
32	0	0	0	0	0	0	0	0	.....
40	0	0	0	0	0	0	0	0	.....
48	0	0	0	0	0	0	0	0	.....

作业：投币式手机充电仪

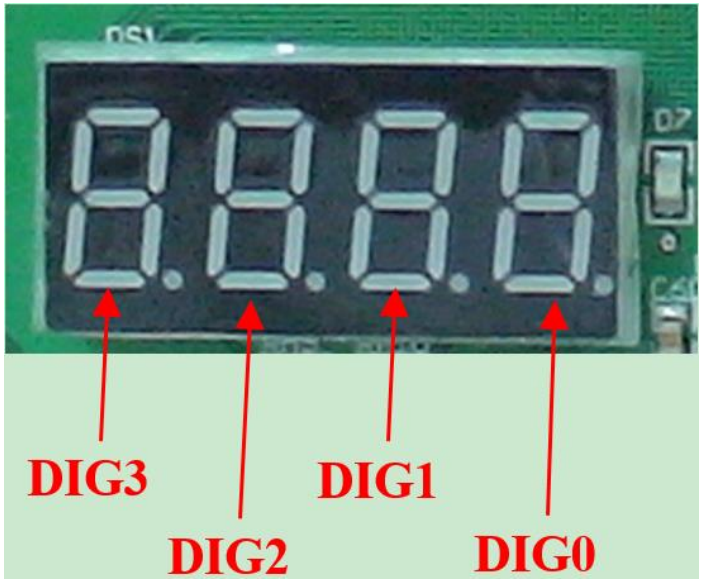

输入设备
（键盘）
输出设备
（数码管）
输出设备
（蜂鸣器）
键盘输入
模块
数码管显示
模块
存储器
ROM
运算器
比较器
……
控制单元
……
FPGA

4位扫描显示数码管

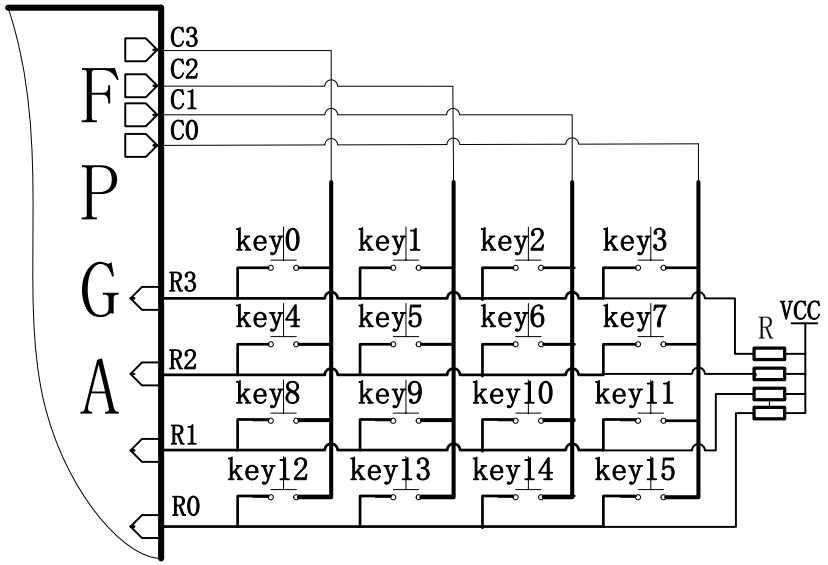


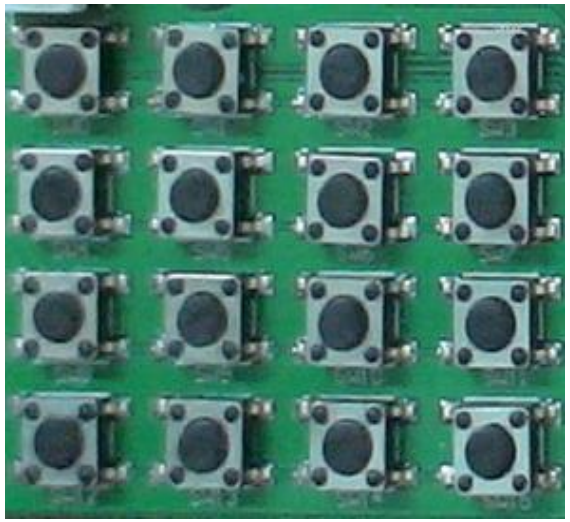


数码管	DIG3	PIN_39	高电平有效
	DIG2	PIN_37	
	DIG1	PIN_36	
	DIG0	PIN_35	
	DP	PIN_46	高电平有效
	G	PIN_43	
	F	PIN_41	
	E	PIN_48	
	D	PIN_47	
	C	PIN_45	
	B	PIN_40	
	A	PIN_44	



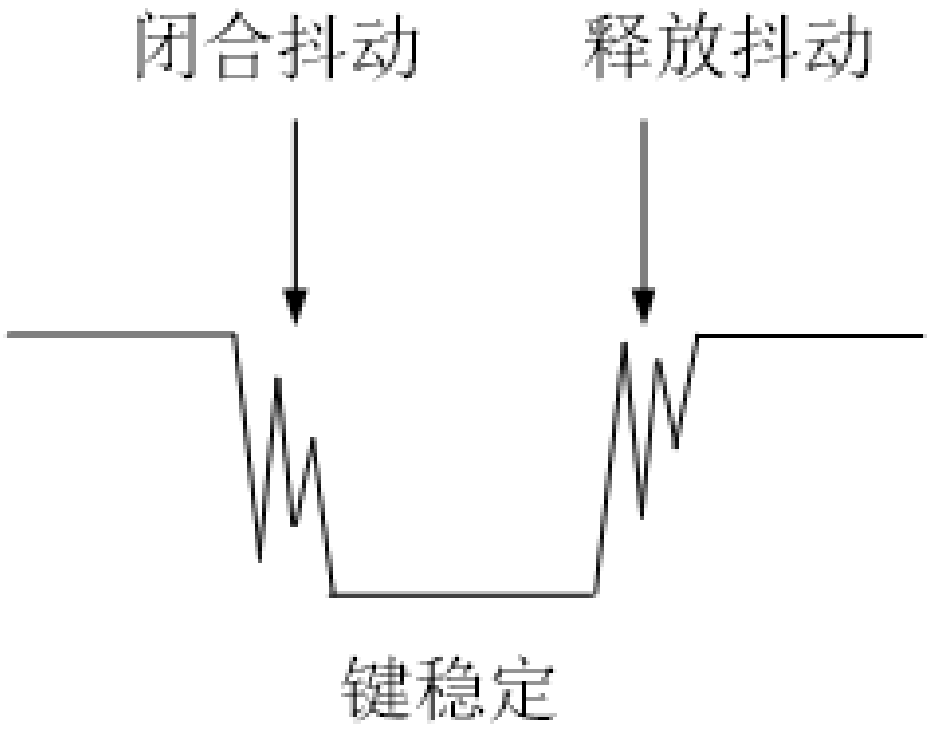
4*4矩阵键盘





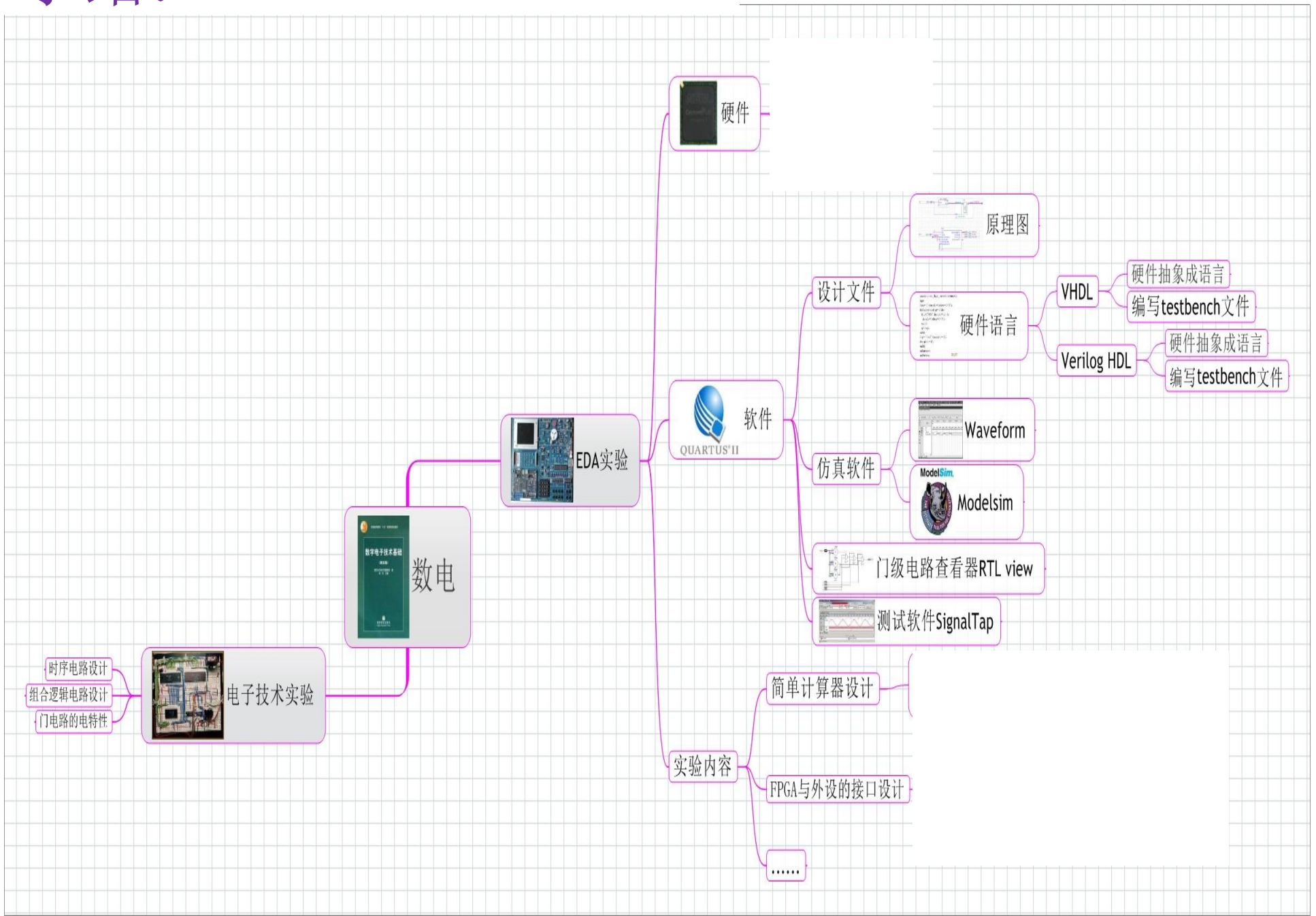
矩阵键盘	C3	PIN_13	列(Column line) 输出→ 低电平有效
	C2	PIN_14	
	C1	PIN_15	
	C0	PIN_30	
	R3	PIN_31	行(Row line) 输入← 低电平有效
	R2	PIN_32	
	R1	PIN_33	
	R0	PIN_34	

按键防抖



抖动时间一般为1~10ms，建议防抖

小结：


数电
电子技术实验
时序电路设计
组合逻辑电路设计
门电路的电特性
EDA实验
硬件
软件
QUARTUS II
实验内容
简单计算器设计
FPGA与外设的接口设计
.....
设计文件
原理图
硬件语言
VHDL
硬件抽象成语言
编写testbench文件
Verilog HDL
硬件抽象成语言
编写testbench文件
仿真软件
Waveform
ModelSim
Modelsim
门级电路查看器RTL view
测试软件SignalTap

谢　谢