EDA 大作业二 投币式手机充电仪

实验报告

姓名：门恺文

班级：自 83

学号：2018011501

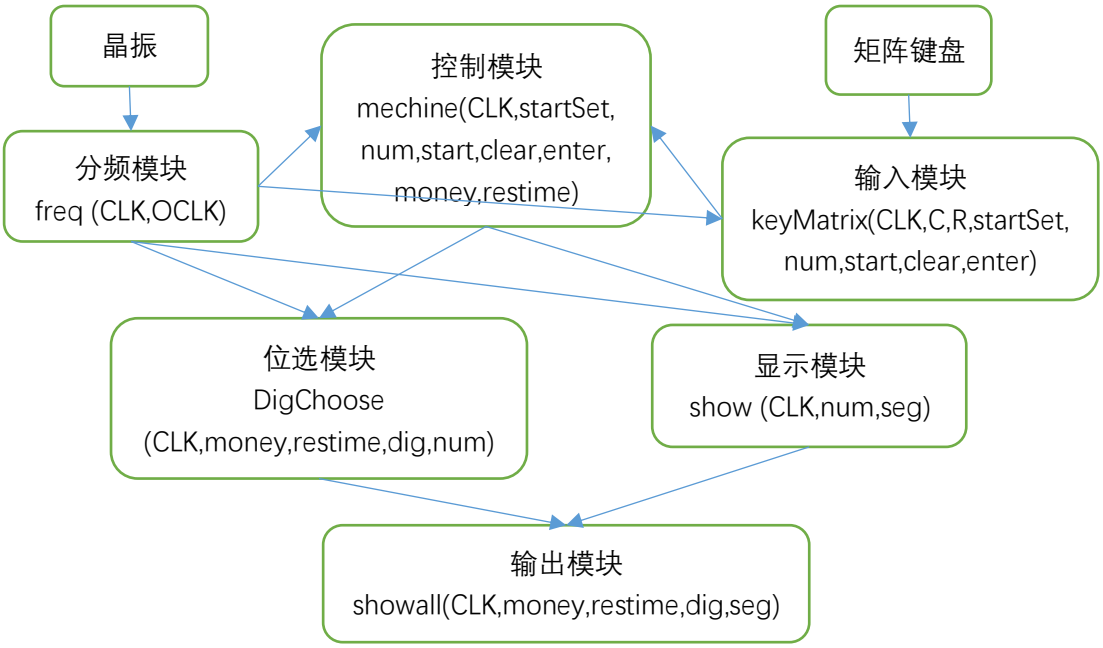
日期：2019.12.19

一、实验目的

1. 学习自顶向下、分模块的数字系统分析、设计与调试方法。
2. 编写测试文件对设计电路进行仿真验证。
3. 掌握规范使用硬件描述语言描述状态机电路的方法。

二、预习任务

1.总体框图及各模块引脚标注



（1）分频模块 freq：将晶振的 50MHz 的频率降低到 25kHz，便于之后的使用。输入为 CLK，输出为 OCLK。

（2）输入模块 keyMatrix：将键盘上的按键转化为对应的编码，同时通过这些编码转化为具体对应的输入信息。输入为 CLK,C,输出为 R, startSet, num, start, clear, enter。

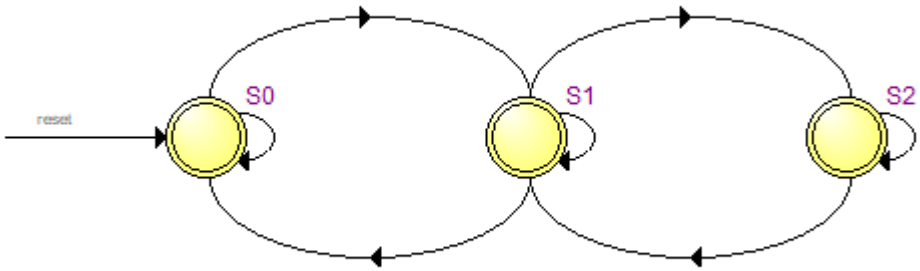
（3）控制模块 mechine：实现电路的具体功能。输入为 CLK, startSet, num, start, clear, enter，输出为 money,restime。

（4）显示模块 showall：包括两个模块，位选模块 DigChoose 和显示模块 show。位选模块选择数码管进行显示，显示模块将具体数字显示在相应数码管上。输入为 CLK,money,restime，输出为 dig,seg。

2.状态转换图

状态转换图如下。S0 为初始状态，也是上电后最开始的状态；S1 为按下开始键

后到按下确认键之前的输入状态；S2 为按下确认键后的倒计时状态。



三、设计思路

本次设计为自顶向下、分模块的数字系统设计，主要划分为四个模块，各模块之间关系及输入输出见预习任务，各模块设计思路如下：

1. 分频模块

为便于后级模块使用同步时钟，将晶振分频至合理值。本次设计将晶振时钟分频为 25kHz，即每接收到 1000 个晶振时钟信号翻转一次。代码实现如下：

```
1   module freq (CLK,OCLK);
2      input CLK;
3      output OCLK;
4      reg OCLK;
5      reg [12:0]cnt;
6
7      initial
8         begin
9         cnt<=0;
10        OCLK <= 0;
11        end
12
13     always @ (posedge CLK)
14        begin
15           cnt <= cnt + 1;
16           if (cnt == 1000)
17              begin
18              OCLK <= ~OCLK;
19              cnt <= 0;
20              end
21        end
22
23  endmodule
```

2. 输入模块

（1）输入列信号，在没有按键且列信号为 4’b1111 时遍历行信号。

```
always @ (posedge CLK)
   begin
   if (!isPush)
   begin
      if (C==4'b1111)
      begin
         startSet<=0;
         posCnt<=0;
         case (round)
         0: R=4'b0111;
         1: R=4'b1011;
         2: R=4'b1101;
         3: R=4'b1110;
         endcase
         round=round+1;
      end
      else isPush<=1;
   end
```

（2）通过声明 negCnt 和 posCnt 分别对于按键前后时间长短的检查，以及每次确定按键后对于 negCnt 和 posCnt 的清零以便重新计数，实现防抖功能。

```
begin
   if (C==4'b1111)
   begin
      negCnt<=0;
      if(posCnt<4000) posCnt=posCnt+1;
      else isPush<=0;
   end
   else if (negCnt<4000) negCnt=negCnt+1;
   else
   begin
      startSet=1;
      negCnt<=0;
      posCnt<=0; //防抖
   end
end
```

（3）确定按键后，通过确定按键的上升沿信号进行键盘的编码，与防抖相配合，可以解决长按键问题，即即使一直按着，在合理扰动范围内，编码不会改变。

```
always @ (posedge startSet)  //完全解决长按键问题
```

（4）通过列 C 与行 R 的状态，将矩阵键盘按下键进行编码得到 key。

```
case(R)
   4'b1110:key=0;
   4'b1101:key=4;
   4'b1011:key=8;
   4'b0111:key=12;
endcase
case(C)
   4'b1110:key=key+1;
   4'b1101:key=key+2;
   4'b1011:key=key+3;
   4'b0111:key=key+4;
endcase
```

（5）通过得到的编码，转化为对应的数字或者 start，clear，enter 信号，对应关系如下：

```
case(key)
    1:num<=1;
    2:num<=2;
    3:num<=3;
    4:num<=4;
    5:num<=5;
    6:num<=6;
    7:num<=7;
    8:num<=8;
    9:num<=9;
    10:num<=0;
    11:start<=1;
    12:clear<=1;
    13:enter<=1;
    default:
        begin
        num<=20;
        start<=0;
        clear<=0;
        enter<=0;
        end
endcase
```

3.控制模块

（1）状态转换：初始状态为 S0，按下开始键后进入 S1 状态，其他按键均保留在 S0 状态；进入 S1 状态之后，若 money 和 restime 均为 0，10 秒钟无动作后回到 S0 状态，若 money 和 restime 不为 0，且按下了 enter 键，进入 S2 状态，其他按键均停留在 S1 状态；进入 S2 状态后，倒计时未结束前，一直在 S2 状态，倒计时结束后，进入 S1 状态。

（2）有效按键：检测到按键后，需要确定按下状态的前一个状态是未按下，该按下信息才有效，且与之前的防抖配合，以解决长按键问题。

```
wire set = ((last_set != startSet) && startSet);

-------
last_set <= startSet;
```

（3）S0 状态：初始状态，除了按下 start 键以外均无效。

```
S0:
begin
   if (set)
   begin
      if (start)
      begin
         state <= S1;
         money <= 0;
         restime <= 0;
         timer <= 1;
      end
      else
         state <= S0;
   end
   else
   begin
      state <= S0;
      money <= 8'hff;
      restime <= 8'hff;
   end
end
```

（4）S1 状态：若按下 clear，则清零并重设时钟，开始 10 秒倒计时，若是从初始状态进入 S1 状态而 money 为 0，则也重设时钟进行 10 秒钟倒计时。

```
if (clear || start)
begin
   if (clear)
   begin
      money <= 0;
      restime <= 0;
      reset = 1;
      timer <= 1;
      pause_count <= 0;
   end
   else
   begin
      if (money == 8'hff && restime == 8'hff)
      begin
         money <= 0;
         restime <= 0;
      end
      if (money == 0)
      begin
         reset = 1;
         timer <= 1;
         pause_count <= 0;
      end
   end
end
```

如果输入的 num 为有效值（即 0-9），则关闭 10 秒钟倒计时，并开始进行金额和时间的设置。先将下个金额数阻塞性赋值到 next_money，若小于 20，则同时赋值给 money 和 restime，否则分别将 20 和 40 赋值给 money 和 restime。

```
else if (num < 10)
begin
   timer <= 0;
   pause_count <= 0;
   next_money = (money * 10 + num) % 100;
   if (next_money > 20)
   begin
      money <= 20;
      restime <= 40;
   end
   else
   begin
      money <= next_money;
      restime <= next_money * 2;
   end
end
```

（5）10 秒钟倒计时设置：如果时钟信号开启且没有重设时钟信号，则进行十秒钟倒计时。pause 设置为正常的十秒钟，即 250000 个时钟信号。经过十秒钟后，回到 S0 状态。

```
if (timer && (!reset))
begin
    if (pause_count < pause)
       pause_count <= pause_count + 1;
    else
    begin
       pause_count <= pause + 1;
       state <= S0;
       timer <= 0;
       pause_count <= 0;
    end
end
```

（6）S2 状态：进入充电倒计时，second 设置为实际的 1 秒钟，即 25000 个时钟信号。time_count 进行 25000 次计数后倒计时一秒。restime 为 1 时，让下一个状态的 money 为 0，这样时间和金额可以同时归零。当 money 为 0 后，状态回到 S1 并开始 10 秒钟倒计时。

```
S2:
begin
   if (restime == 0)
      money <= 0;
   else
   begin
      if (time_count < second)
         time_count <= time_count + 1;
      else
         time_count <= 0;
      if (time_count == second - 1)
      begin
         if (restime != 0)
            restime <= restime - 1;
         if (restime == 1)
            money <= 0;
      end
   end
   if (money == 0)
   begin
      state <= S1;
      timer <= 1;
      pause_count <= 0;
   end
end
```

4.输出模块

包括两个模块：数码管位选模块及相应的输出，数码管显示模块。

（1）数码管位选模块及对应输出：若 money 为设定的无效值，则仅对应 DIG3，且将 num 设为无效值

```
if (money == 8'hff)
begin
   dig <= 4'b0001;
   num <= 4'b1111;
end
```

若 money 为有效值，则轮流显示四个数码管，且每个数码管显示对应要显示的数。由于是同步电路，所以数值为下一状态显示数码管对应的数值。如果没有对应的数码管位选信号，则将 num 设为无效值。

```
case(dig)
   4'b0001:
   begin
      dig <= 4'b0010;
      num <= money % 10;
   end
   4'b0010:
   begin
      dig <= 4'b0100;
      num <= money / 10;
   end
   4'b0100:
   begin
      dig <= 4'b1000;
      num <= restime % 10;
   end
   4'b1000:
   begin
      dig <= 4'b0001;
      num <= restime / 10;
   end
   default
   begin
      dig = 4'b0001;
      num = 4'b1111;
   end
endcase
```

（2）数码管显示模块：将 num 对应到七段显示译码管的编码，同时对于无效值，应使七段显示译码管全灭。

```
case (num)
    4'd0 :
        seg <= 7'b0111111;
    4'd1 :
        seg <= 7'b0000110;
    4'd2 :
        seg <= 7'b1011011;
    4'd3 :
        seg <= 7'b1001111;
    4'd4 :
        seg <= 7'b1100110;
    4'd5 :
        seg <= 7'b1101101;
    4'd6 :
        seg <= 7'b1111101;
    4'd7 :
        seg <= 7'b0000111;
    4'd8 :
        seg <= 7'b1111111;
    4'd9 :
        seg <= 7'b1101111;
    default :
        seg <= 7'b0000000;
endcase
```

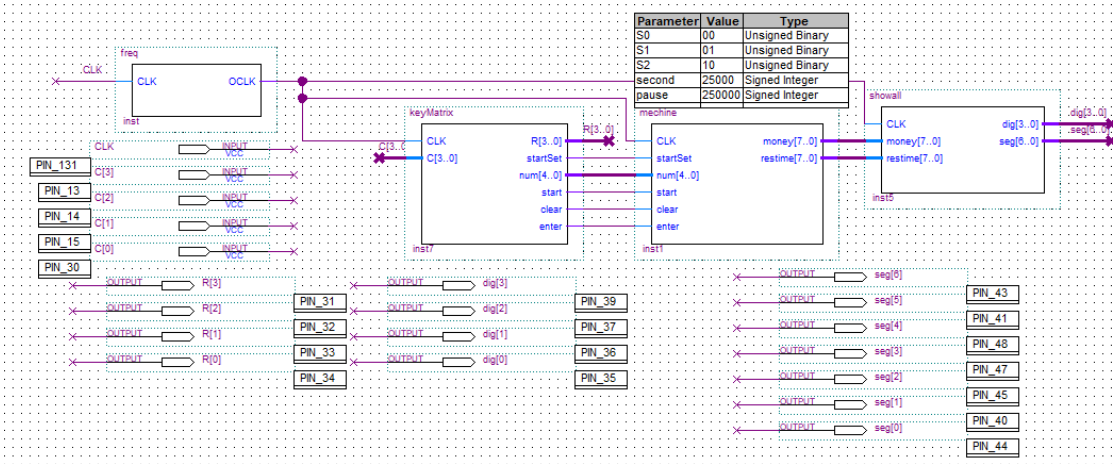
（3）合并：直接将两模块合并，即得输出模块。

```
module showall(CLK,money,restime,dig,seg);
    input CLK;
    input [7:0] money;
    input [7:0] restime;
    output [3:0] dig;
    output [6:0] seg;
    wire [3:0] num;

    DigChoose dc(CLK,money,restime,dig,num);
    show sh(CLK,num,seg);

endmodule
```

四、顶层电路图及各模块功能



顶层电路图如上图，各模块设计思路中已阐述功能，此处进行简要对应如下：

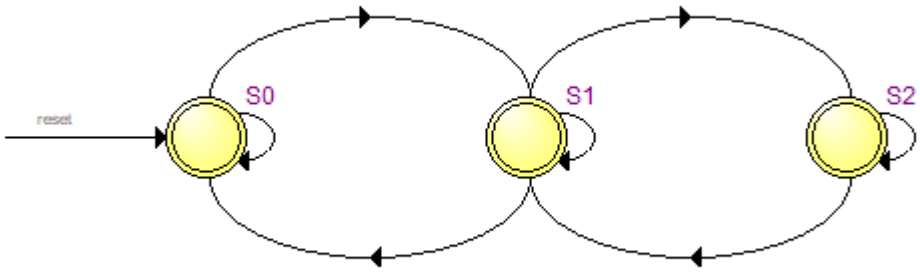
1.freq：分频；

2.keyMatrix：将矩阵键盘的输入转化为数字，开始信号，清零信号，确认信号；

3.mechine：状态机主逻辑，输出金额、时间信息；

4.showall：在四位数码管上显示。

五、状态转换图及说明

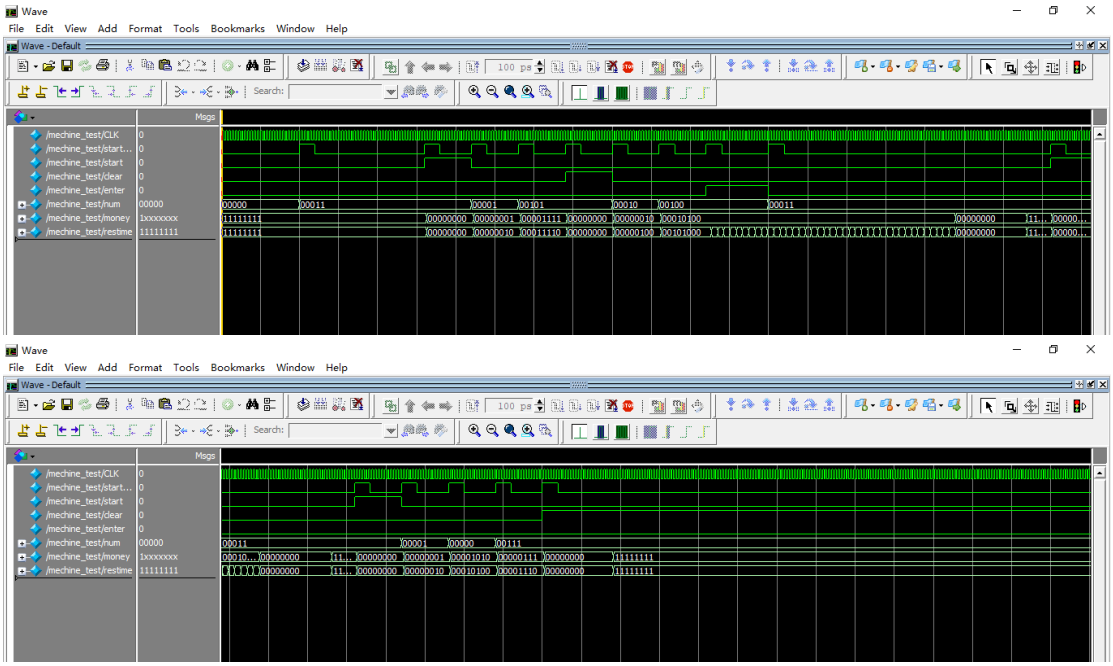


状态转换图的说明已在设计思路中阐明，即初始状态为 S0，按下开始键后进入 S1 状态，其他按键均保留在 S0 状态；进入 S1 状态之后，若 money 和 restime 均为 0，10 秒钟无动作后回到 S0 状态，若 money 和 restime 不为 0，且按下了 enter 键，进入 S2 状态，其他按键均停留在 S1 状态；进入 S2 状态后，倒计时未结束前，一直在 S2 状态，倒计时结束后，进入 S1 状态。

六、仿真波形图及其说明

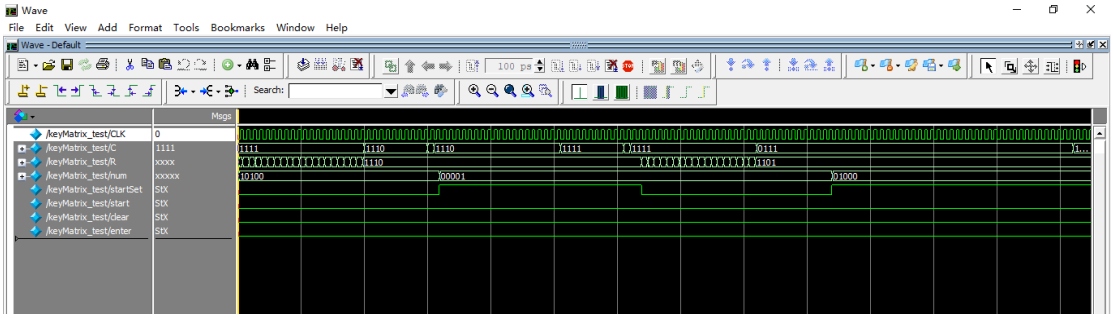
1.状态机电路仿真

仿真波形如下，由于波形图比较长，放了两张图。第一行表示时钟信号，第二行表示有无按键，第三行为开始信号，第四行为清零信号，第五行为确认信号，第六行为读入的数字，第七行为金额，第八行为剩余时间。由仿真波形可看出，处于 S0 状态时按除开始建以外的键无反应，按下开始键后，金额和时间都为 0；之后输入数字显示金额和时间，金额大于 20 时按 20 显示；按下清零时金额与时间清零；再次输入后，按下确认，开始倒计时，且时间与金额同时归零。十秒之后回到初始状态。之后再次按下开始，进行输入并清零后，十秒钟之后也回到初始状态。



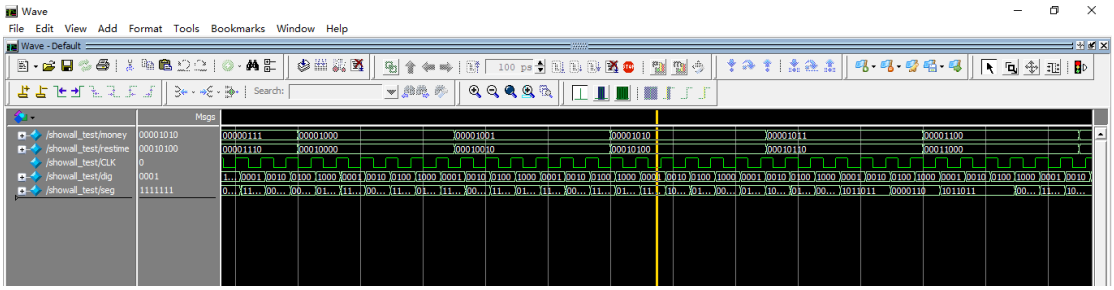
2. 矩阵键盘仿真

仿真中表现了矩阵键盘的防抖和防止长按键功能。从仿真可看到，按下和抬起都有抖动，但是不影响数字的设置；之后长按键，数字也没有改变。



3. 数码管显示电路仿真

如下图，在 money 和 restime 变化过程中，随着时钟信号位选信息和显示信息变化如下：



七、设计和调试中遇到的问题和解决方法

1. 在刚开始不够明晰非阻塞赋值与阻塞赋值的区别，有时会对同一个变量混用，

导致报错。

解决方法：对同一个变量统一。

2.误用非阻塞赋值。在同一次时钟信号下，如果使用非阻塞赋值，再次使用赋值后的值时，还是使用前一状态的值。

解决方法：引入另一个变量进行阻塞赋值，再将该值赋给需要的变量。

3.使用 modelism 仿真时找不到对应文件：编写测试文件时把要测试的文件名写错了。

解决方法：纠正文件名。

4.仿真时发现在输入有效数字时输出不符合预期。分析发现是输出取值范围太小，导致有时暂时赋值较大而越界，

解决方法：扩大取值范围。

5.按下 clear 清零后不能倒计时回到初始状态。发现是将时钟信号置 1 后没有重设时钟和倒计时。

解决方法：在按下 clear 之后执行重设倒计时，并在每次接收到时钟信号后将时钟重设，重新判断是否要执行倒计时。