EDA 大作业二　投币式手机充电仪

自 05 刘若涵 2020011126

一、 实验目的

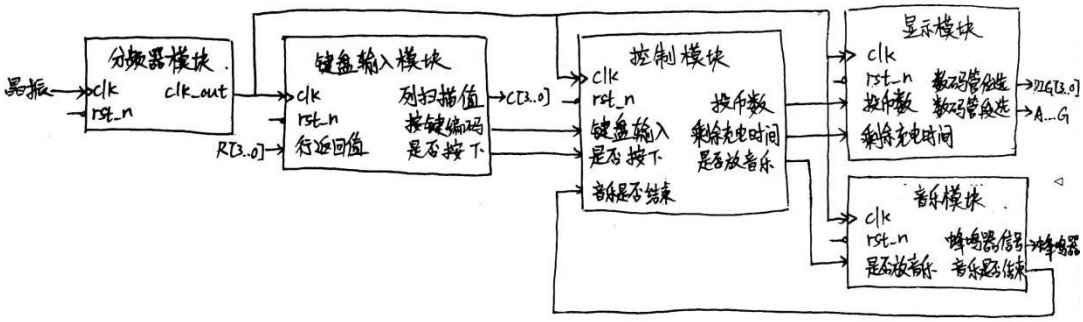
1. 学习自顶向下、分模块的数字系统分析、设计与调试方法；

2. 编写测试文件对设计电路进行仿真验证；

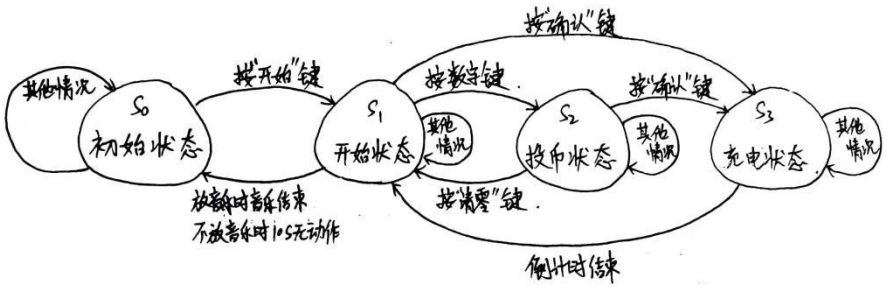
3. 掌握规范使用硬件描述语言描述状态机电路的方法。

二、 预习报告

1. 阅读并分析任务要求，画出电路的总体框图，注明各功能模块及其引脚



2. 根据任务要求画出控制电路的状态转换图



三、 设计思路

采用自顶而下的设计思路，根据功能将电路划分为三个主要模块，分别是键盘输入模块、控制模块和显示模块。另外还有两个辅助模块，分别是分频器模块和音乐模块。

随后分模块进行电路设计，每次将一个模块调好后，便将其封装，便于调用。下面将逐一阐述每个子模块的设计思路。

1. 分频器模块

分频器模块将来自晶振的 50MHz 时钟信号作为 clk 输入，输出分频后的 200Hz 信号 clk_out，为键盘输入、控制和显示模块提供时钟信号。内部使用 cnt 变量来计数，当 clk 上升沿到达的时候，如果 cnt 未达到 124999，则将 cnt 加 1；否则将 cnt 置零，并将 clk_out 翻转。

2. 键盘输入模块

键盘输入模块从矩阵键盘读入按键，并将按键状态信息传给控制模块。输出数据 key 为当前按下的键的编码，使能信号 EN 用来决定 key 是否有效。键盘输入模块通过对列扫描，依次将每个列线置零，并读取行线的值。如果 4 条行线不全为高电平，则扫描停止在该列。如果连续 32 个时钟周期行值不变，则确认按键按下，将 EN 置为 1，并根据当前的行列值为 key 赋值。如果 4 条行线变为 1111，则将 EN 置为 0。如果连续 32 个时钟周期行值全为 1，则确认按键松开，重新开始列线扫描。

3. 控制模块

控制模块采用三段式状态机编写，输入 200Hz 时钟信号 clk，键盘编码 key，使能信号 EN 和判断音乐是否结束信号 music_end，输出对应的投币数 MONEY、剩余充电时间 TIME 和控制音乐模块启动的信号 music_on。

（1）预处理

```
25  reg [3:0]  NUM;
26  reg [14:0] time_count;
27  reg [12:0] wait_count;
28  reg Istime_out;
29  reg Ismoney_out;
30  reg LAST_EN;
31  reg [3:0]  M0;
32  reg [7:0]  next_M;
33  wire RESET;
34  wire BEGIN;
35  wire CONFIRM;
36  wire KeyPress;
37
38  assign KeyPress = (!LAST_EN && EN);
39  assign RESET = (key == 4'b1011);
40  assign BEGIN = (key == 4'b1010);
41  assign CONFIRM = (key == 4'b1100);
```

首先对由键盘输入模块传来的信号做处理。为避免重复赋值，需判断按键是否是为新按下，用 LAST_EN 存下上一个 CLK 上升沿到来时 EN 的值，如果这一刻的 EN 为 1 而 LAST_EN 为 0，则表示在这一刻键盘被有效新按下，记 KeyPress = (!LAST_EN && EN)。同时用 BEGIN、CONFIRM、RESET 分别代表“开始”、“确认”、“清零”键状态，若按下则置为 1，若没有则置为 0。用 Istime_out 和 Ismoney_out 分别来判断是否要熄屏和充电倒计时是否结束，是为 1，否为 0。

```
always @ (*)
begin
    case (key)
    4'b0000:
        NUM <= 1;
    4'b0001:
        NUM <= 2;
    4'b0010:
        NUM <= 3;
    4'b0011:
        NUM <= 4;
    4'b0100:
        NUM <= 5;
    4'b0101:
        NUM <= 6;
    4'b0110:
        NUM <= 7;
    4'b0111:
        NUM <= 8;
    4'b1000:
        NUM <= 9;
    4'b1001:
        NUM <= 0;
    default :
        NUM <= 4'b1111;
    endcase
end
```

为方便后续判断与赋值，用 NUM 存下换算后的键盘编码，若按下的按键为数字键，则将对应数字赋给 NUM，若为其他按键，则将 4’b1111 赋给 NUM。

（2）状态转换

```
parameter ORIGIN = 2'b00;
parameter START  = 2'b01;
parameter INSERT = 2'b10;
parameter CHARGE = 2'b11;

reg [1:0] current_state;
reg [1:0] next_state;
```

状态机共有 ORIGIN（初始）、START（开始）、INSERT（投币）、CHARGE（充电）四个状态。

```
71  always @ (posedge clk, negedge rst_n)
72 begin
73    if (!rst_n)
74    begin
75      current_state <= ORIGIN;
76    end
77    else
78    begin
79      current_state <= next_state;
80    end
81  end
82
83  always @ (*)
84 begin
85    case (current_state)
86      ORIGIN:
87      begin
88      if(KeyPress)
89      begin
90        if (BEGIN)
91          next_state <= START;
92        else
93          next_state <= ORIGIN;
94      end
95      else
96          next_state <= ORIGIN;
97      end
98
```

上电时为 ORIGIN 状态，新按下“开始”键时转换为 START 状态，其他情况保持 ORIGIN 状态不变。

```
99      START:
00      begin
01      if(KeyPress)
02      begin
03        if (NUM < 10)
04          next_state <= INSERT;
05        else if(CONFIRM)
06          next_state <= CHARGE;
07        else
08          next_state <= START;
09      end
10      else
11      begin
12      if(Istime_out)
13          next_state <= ORIGIN;
14      else
15          next_state <= START;
16      end
17      end
```

START 状态下，若新按下数字键，即 NUM<10，则转换为 INSERT 状态，若新按下“确认”键，则转换为充电状态，若 Istime_out=1，则转换为 ORIGIN 状态，其他情况保持 START 状态。

```
    INSERT:
    begin
    if(KeyPress)
    begin
      if (RESET)
         next_state <= START;
      else if (CONFIRM)
         next_state <= CHARGE;
      else
         next_state <= INSERT;
    end
    else
         next_state <= INSERT;
    end
```

INSERT 状态下，若新按下“确认”键，则转换为 CHARGE 状态，若新按下“清零”键，则转换为 START 状态，其他情况下保持 INSERT 状态。

```
    CHARGE:
    begin
      if(Ismoney_out)
         next_state <= START;
      else
         next_state <= CHARGE;
    end
    default : next_state <= ORIGIN;
  endcase
end
```

CHARGE 状态下，若 Ismoney_out=1，则转换为 START 状态，其他情况下保持 CHARGE 状态。

（3）输出

```
 always @ (posedge clk, negedge rst_n)
begin
if (!rst_n)
begin
    MONEY <= 8'b11111111;
    TIME <= 8'b11111111;
end
else
begin
  case (next_state)
    ORIGIN:
    begin
    music_on <= 0;
    wait_count <= 0;
    Istime_out <= 0;
    MONEY <= 8'b11111111;
    TIME <= 8'b11111111;
    end
```

ORINGIN 状态下，输出 music_on=0，保证回到初始状态时音乐停止。并将 Istime_out 置为 0，方便下次使用。同时为方便显示模块译码，置 MONEY=4’b1111，TIME=4’b1111。

```
65      START:
66      begin
67      MONEY <= 0;
68      TIME <= 0;
69      Ismoney_out <= 0;
70      if(KeyPress)
71        wait_count <= 0;
72      else
73      begin
74      if(!music_on)
75      begin
76        if(wait_count < WAIT_TIME)
77           wait_count <= wait_count +1'b1;
78        else
79        begin
80           wait_count <= 0;
81           Istime_out <= 1;
82        end
83      end
84      else
85      begin
86        if(music_end)
87        begin
88           Istime_out <= 1;
89           music_on <= 0;
90        end
91      end
92      end
93      end
```

START 状态下，输出 MONEY=0，TIME=0。并将 Ismoney_out 置为 0，方便下次使用。当没有按键按下时，若在播放音乐，即 music_on=1，则在收到 music_end=1 的信号后将 Istime_out 置为 1，并将 music_on 置零，实现音乐结束后回到初始状态；若不在播放音乐，则用变量 wait_count 计数，若持续没有按键按下，则 wait_count 每个时钟周期加 1，达到 2000（对应 10s）后将 Istime_out 置为 1，实现 10s 无动作时回到初始状态，同时将 wait_count 清零，方便下次使用，若未达到 2000 前有按键按下则 wait_count 立即清零。

```
195     INSERT:
196     begin
197     music_on <= 0;
198     wait_count <= 0;
199     if(KeyPress)
200     begin
201          if(NUM < 10)
202          begin
203             M0 = MONEY % 10;
204             next_M = M0 * 10 + NUM;
205             if (next_M > 20)
206             begin
207                MONEY <= 20;
208                TIME <= 40;
209             end
210             else
211             begin
212                MONEY <= next_M;
213                TIME <= next_M * 2;
214             end
215          end
216     end
217     end
```

INSERT 状态下，输出 music_on=0，实现若有投币动作音乐停止。本控制电路可以输入多个数，但一旦输入大于两个，则取上一次显示的 MONEY 个位和新输入的数字作为有效输入，算出新的 MONEY。例如， 输入“1”，则数码管显示“0102”；再输入“1”，则数码管显示 “1122”，可重复输入同一数字；再输入“4”，则数码管显示 “1428”；再输入“7”，由于钱数 47 超过了 20 则数码管显示“2040”；再输入“9”，由于上一次显示的 MONEY 个位为 0，则数码管显示 “0918”。为了实现这个功能，使用 M0 来保存 MONEY 的个位，使用 next_M=M0*10+NUM 来判断下一时刻的 MONEY 值，M0 和 next_M 只有在确认数字键被新按下，即 KeyPress = (!LAST_EN && EN)=1 时，才会被赋值，保证了输入的有效性并避免了长按键的重复输入。若 next_M > 20，则输出 MONEY=20，TIME=40，若 next_M <= 20，则输出 MONEY=next_M，TIME=2*next_M。

```
    CHARGE:
    begin
    wait_count <= 0;
    if(TIME == 0)
    begin
      music_on <= 1;
      Ismoney_out <= 1;
    end
    else
    begin
      if(time_count < SEC)
      begin
        time_count <= time_count + 1'b1;
      end
      else
      begin
        time_count <= 0;
      end
      if(time_count == SEC - 1)
      begin
        if(TIME > 1 )
         TIME <= TIME - 1'b1;
        if(TIME == 1)
        begin
         music_on <= 1;
         Ismoney_out <= 1;
        end
      end
    end
    end
  endcase
  LAST_EN <= EN;
end
end
```

CHARGE 状态下，如果 MONEY 本身为 0，则直接将 music_on 和 Ismoney_out 置为 1，实现倒计时结束跳转为 START 状态并开始播放音乐。如果 MONEY 不为 0，则 MONEY 保持不变，用 time_count 计数，一个时钟周期加一，达到 200（对应 1s）后 TIME 减一，同时 time_count 清零重新开始计数。当 TIME 减到 1 后，下个 time_count 计满时，将 music_on 和 Ismoney_out 置为 1，实现倒计时结束跳转为 START 状态，TIME 和 MONEY 同时变 0，并开始播放音乐。

4. 显示模块

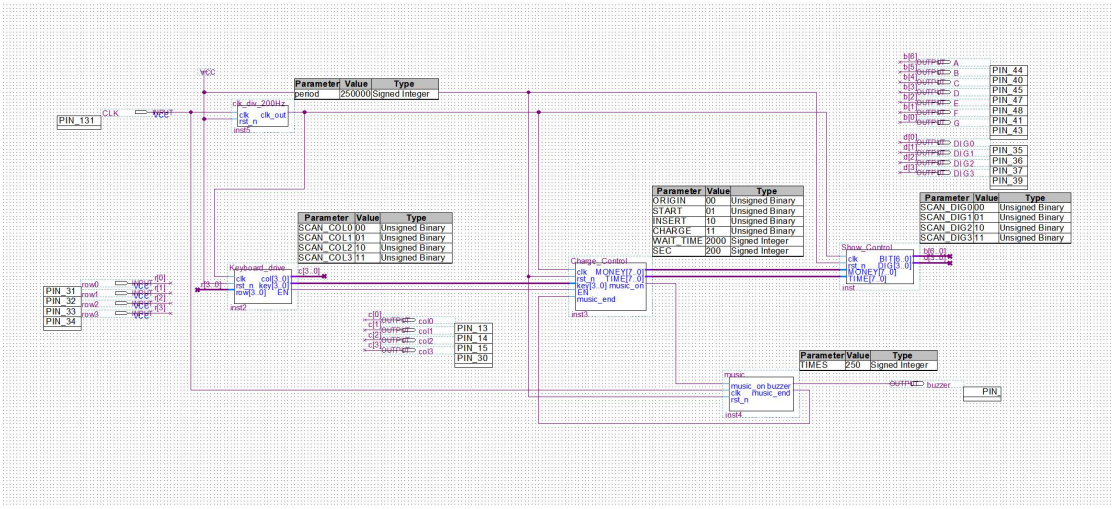
显示模块输入 200Hz 时钟信号 clk，投币数 MONEY 和剩余充电时间 TIME。输出数码管位选信号 DIG，数码管段选信号 BIT。DIG 实现对数码管的扫描，循环输出“0001”、“0010”、“0100”、“1000”四种取值，每个时钟周期变化一次。同时用变量 SHOW 记录对应数码管要显示的值，分别为 TIME/10、TIME%10、MONEY/10、MONEY%10。特别的，

若 MONEY 和 TIME 为 4’b1111，则代表数码管需要熄灭，SHOW=4’b1111。随后将 SHOW 译码，转化成数码管段选信号，参照 BCD-七段显示译码器译码方式进行。当 SHOW 为 0~9 时显示对应数字，SHOW > 9 时，七位段选均为 0，数码管全灭。

5. 音乐模块

音乐模块输入 50MHz 时钟信号 clk，音乐模块是否启动信号 music_on，输出驱动蜂鸣器信号 buzzer，判断音乐是否结束信号 music_end。首先根据各个音符的频率，计算出各个音符需要持续的时钟周期数，用 pre_set 记录。用 cnt0 来控制蜂鸣器响的频率，当 cnt0 < pre_set/2-1 时，buzzer=0，pre_set/2-1 < cnt0 < pre_set-1 时，buzzer=1，cnt0 = pre_set-1 时 cnt0 归零。用 cnt1 来计数一拍持续的时间，当 cnt0 计满 pre_set-1 时，cnt1 加 1，当 cnt1 计到设定的一拍时间 TIMES 时归零。用 cnt2 来计数播放到哪一拍，根据不同的拍为 pre_set 装填应播放的音符周期的值。当 cnt1 = TIMES-1 并且 cnt0 = pre_set-1 时，cnt2 加一，当 cnt2 计数到最后一拍（本设计中一共 32 拍）时，cnt2 归零，music_end 置为 1。需要注意的是，以上操作均在 music_on 为 1 的情况下进行，若 music_on 为 0，则 cnt0、cnt1、cnt2、music_end、buzzer 全部置 0。

四、 顶层电路图及各模块的功能



1. 分频器模块

将晶振输入的 50MHz 方波转化为 200Hz 方波输出。为键盘输入模块、控制模块和显示模块提供时钟信号。

2. 键盘输入模块

扫描键盘的列值，检测行值，读取键盘状态，若有键被按下，则对行列值进行编码输出，并输出判断按键是否有效按下的信号。

3. 控制模块

根据键盘模块和音乐模块的输入，完成状态的跳转，输出对应的投币数、剩余充电时间和控制音乐模块启动的信号。

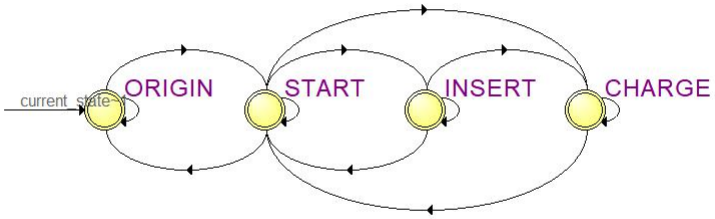
4. 显示模块

根据控制模块的输入，完成译码，并控制四位数码管的位选扫描，使得投币数和剩余充电时间同时显示在对应数码管上。

5. 音乐模块

根据控制模块的输入，启动蜂鸣器，播放设定好的音乐，音乐结束后向控制模块返回音乐结束信号。

五、 状态转换图及其说明



如图所示，控制模块状态机共有 ORIGIN（初始）、START（开始）、INSERT（投币）、CHARGE（充电）四个状态。

初始时为 ORIGIN 状态，新按下“开始”键时转换为 START 状态，其他情况下保持 ORIGIN 状态不变。

START 状态下，若新按下数字键，则转换为 INSERT 状态，若新按下“确认”键，则转换为充电状态。若 Istime_out 变为 1，则转换为 ORIGIN 状态，Istime_out 共有两种

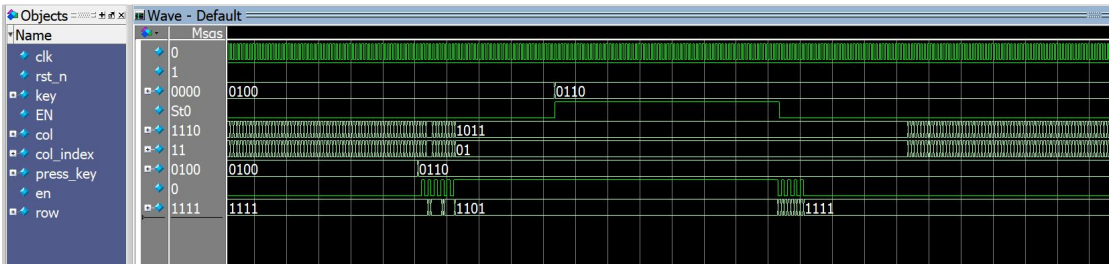
情况会置为 1：1）在播放音乐时若接收到音乐停止信号；2）不在播放音乐时，10s 无按键动作。其他情况下保持 START 状态。

INSERT 状态下，若新按下“确认”键，则转换为 CHARGE 状态，若新按下“清零”键，则转换为 START 状态，其他情况下保持 INSERT 状态。

CHARGE 状态下，若倒计时结束，则转换为 START 状态，其他情况下保持 CHARGE 状态。

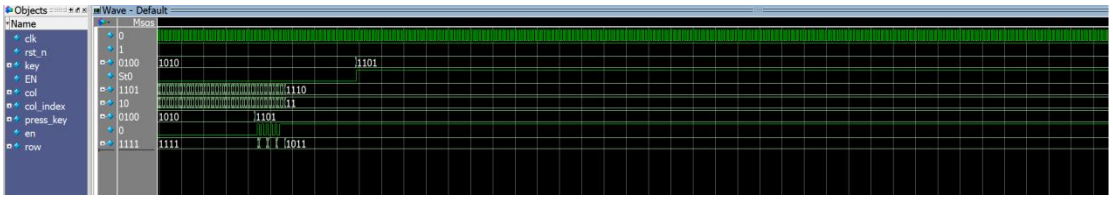
六、 仿真波形图及分析说明

1. 键盘输入模块



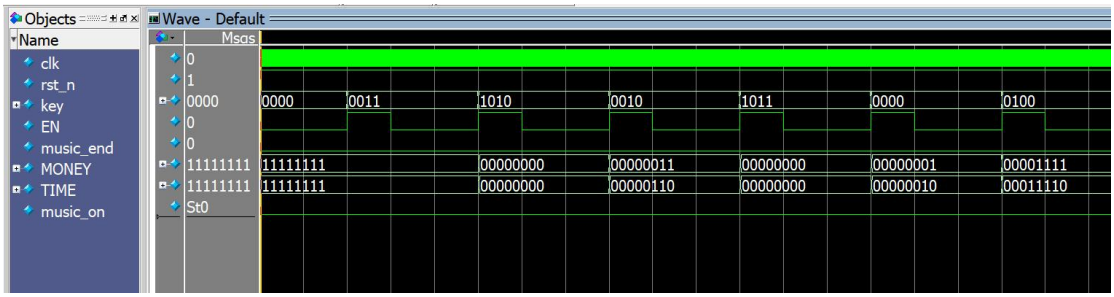
如图所示，key 表示输出的键盘编码，EN 表示输出数据是否有效，col 为列线扫描信号，col_index 用于表示当前扫描的列线线序号，row 为读入的行线数据。en 为测试文件中用于表示按键是否按下（用于控制抖动）的变量。press_key 为测试文件中的变量，表示当前按下的键的二进制编码。

由图可知，当没有按键按下时，row=1111，col 进行列线扫描，当 press_key=0110，即 6 号键（第二列第三行）被按下时，en 开始模拟按键抖动 10 个时钟周期（约 0.05s），此间 row 值变化，col 继续扫描。当抖动停止后，列线继续扫描到第二列时，行线读入 1101 不全为高电平，列线停止扫描，经过一段时间后，EN 由 0 变为 1，表示按键被有效按下，同时给 key 赋值 1010，与 press_key 编码一致，正确。按键被按下 200 个时钟周期（约 1s）后松开，en 模拟按键抖动 10 个时钟周期（约 0.05s）。当抖动开始时，row 值变化，col 仍保持不变，EN 立即由 1 变为 0，表示按键不再被有效按下，key 保持不变。抖动结束，row 保持 1111 一段时间后，col 重新开始列线扫描。



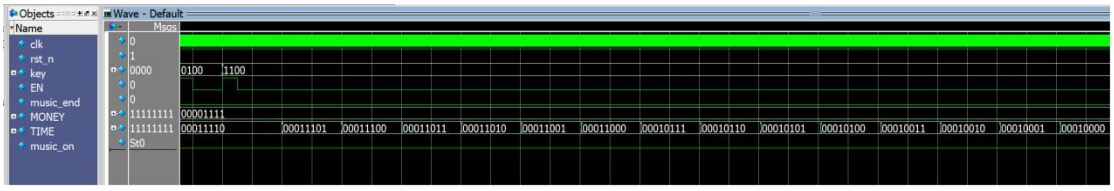
当长按键时，输出 key 只会被赋值一次，且保持不变，实现了长按键不重复赋值的功能。

2. 控制模块

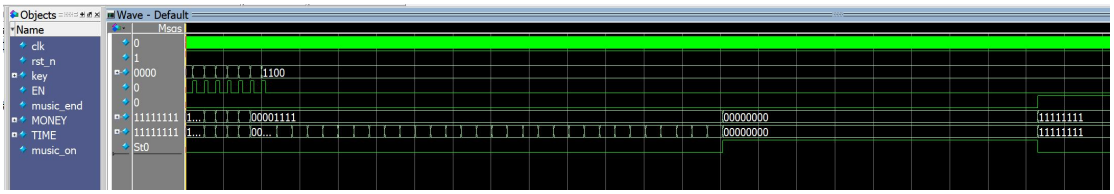


如图所示，key 为按键的编码，EN 用来判断按键是否有效按下，misic_end 用于判断音乐是否结束，MONEY 为输出投币数，TIME 为剩余充电时间，music_on 为控制音乐模块的开关。

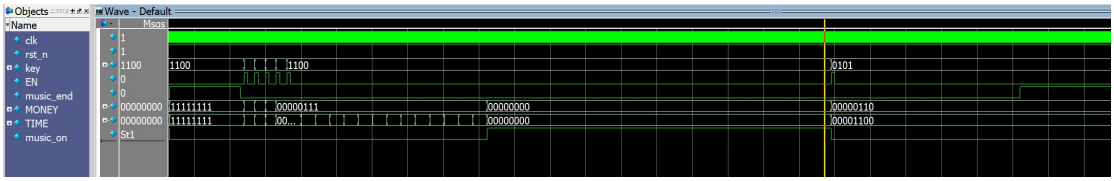
初始时为初始状态，MONEY=4'hff，TIME=4'hff，即代表数码管全灭，此时按下除“开始”键外的其他键（图中为“数字 4”键，编码 0011），仍保持 MONEY=4'hff，TIME=4'hff 不变。按下“开始”键（编码为 1010），跳转开始状态，MONEY=0，TIME=0。此后按下数字键开始投币，按下“数字 3”键（编码为 0010），MONEY=3，TIME=6，按下“清零”键（编码为 1011），MONEY=0，TIME=0，按下“数字 1”键（编码为 0000），MONEY=1，TIME=2，再按下“数字 5”键（编码为 0100），MONEY=15，TIME=30。



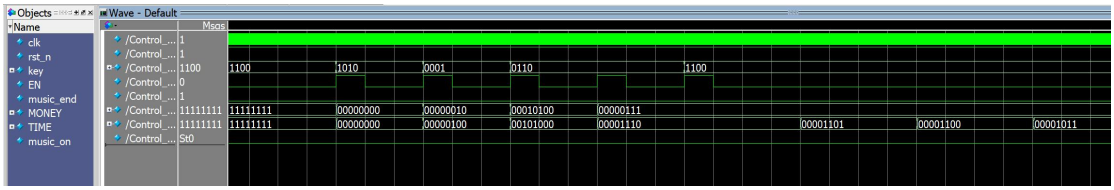
随后按下“确认”键（编码为 1100），跳转充电状态，开始倒计时。



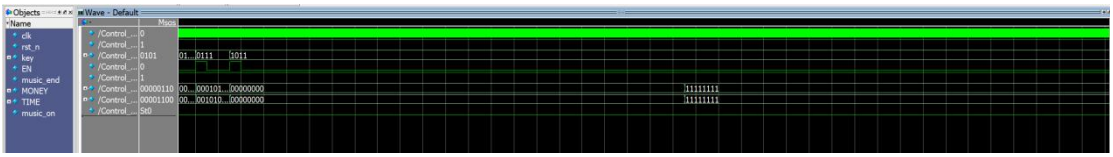
倒计时结束时，跳转开始状态，MONEY=0，TIME=0，music_on 变为 1，开始播放音乐，当 music_end 变为 1 时，即音乐结束时，跳转初始状态，MONEY=4'hff，TIME=4'hff，数码管全灭。



如图中黄线后所示，若在音乐结束前按下了数字键（图中为“数字 6”键，编码为 0101），则跳转为投币状态，MONEY=6，TIME=12，music_on 变为 0，音乐停止播放。

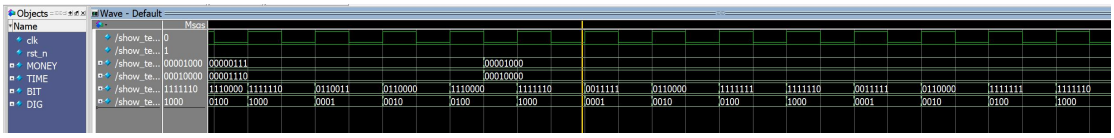


此外，若在投币状态输入的投币数超过 20，则 MONEY=20，TIME=40，若输入的数字超过两个，则取上一次 MONEY 的个位和新输入的数字为有效输入，算出新的 MONEY。图中先输入 2，MONEY=2，TIME=4，再输入 7，由于 27 大于 20，MONEY=20，TIME=40，再输入 7，MONEY=7，TIME=14，可重复输入同一数字，且取上一次 MONEY 的个位和新输入的数字为有效输入。



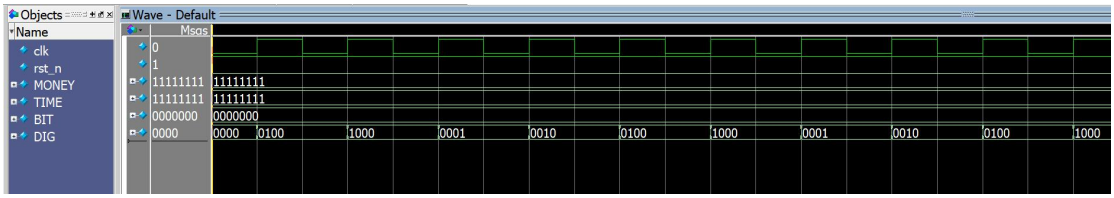
另若在开始状态 10s 无按键动作，则跳转为初始状态，MONEY=4’hff，TIME=4’hff，数码管全灭。

3. 显示模块



如图所示，MONEY 为投币数，TIME 为剩余充电时间，BIT 为数码管位选，DIG 为数码管段选。

当 MONEY 变为 8，TIME 变为 16，黄线后数码管显示变化，第四个的数码管（位选为 0001）显示 6，第三个数码管（位选为 0010）显示 1，第二个数码管（位选为 0100）显示 8，第一个数码管（位选为 1000）显示 0，四位数码管显示“0816”。



另外，若 MONEY=4’b1111，TIME=4’b1111，四位数码管均无显示，全灭。

七、 设计和调试中遇到的问题及解决方法

1. 仿真时，出现路径报错。原因可能是：testbench 文件里的顶层模块名和 testbench 文件名不一致；testbench 文件里引用的被测模块名与原文件的模块名不一致；被测模块未设置为顶层实体；仿真时 testbench 文件未更换为对应文件。

2. 仿真时输出变量一直为红线，显示 XXXX，在 testbench 文件中对输入变量进行初始化，再仿真即可解决这一问题。

3. 最开始设计的防抖方式为检测按键信号变化时刻及变化后 20Ms 的值，若相同，则为输出赋值，若不同则输出信号不变。这种方式经试验并不能有效防抖，键盘输入时经常显示其他列的值。更改为连续 32 个时钟周期按键信号不变，才给输出赋值。此种方法可以有效防抖。

4. 最开始长按同一键会重复赋值。引入使能信号 EN，只有在 EN 上升沿时刻才给变量赋值，从而解决了这一问题。