

# EDA2 基于FPGA的自动售货机

李端 软件92 2019013268

## EDA2 基于FPGA的自动售货机

- 一、实验目的
- 二、实验内容
- 三、设计思路
  - 1.核心运算逻辑部分
  - 2.显示逻辑部分
  - 3.消抖文件
  - 4.整体实现及PIN映射
- 四、实验中遇到的问题及解决方案

## 一、实验目的

1. 实践基于HDL设计和实现时序逻辑电路的流程和方法；
2. 掌握用HDL实现有限状态机的方法；
3. 实践利用FPGA解决实际问题的方法。

## 二、实验内容

利用实验板上的拨码开关和按键开关模拟投币、购物和退币输入，用发光二极管模拟各种提示信息，用数码管显示余额，实现一个自动售货机。要求满足如下规格：

1. 可接受 5 角、1 元和 5 元的投币，每次购买允许投入多种不同币值的钱币；用 3 只数码管显示当前投币金额，如 055 表示已投币 5.5 元；（提示：与实验三不同，此处需要实现 3 只数码管的循环扫描显示）
2. 可售出价格分别为 1.5 元和 2.5 元的商品，假设用户每次购买时只选择单件、一种商品；允许用户多次购买商品，每次购买后，可以进行补充投币；
3. 选择购买商品后，如果投币金额不足，则提醒；否则，售出相应的商品，并提醒用户取走商品；
4. 若用户选择退币，则退回余下的钱，并提醒用户取钱。

实验板上有 40MHz 的时钟信号，对应 FPGA 引脚号为 PIN\_152，自动售货机的工作时钟及数码管循环扫描显示的时钟可由该 40MHz 分频得到。

## 三、设计思路

### 1.核心运算逻辑部分

- “售货机”的本质就是对系统内部“钱”变量的交互，其中包括加、减、置零等等。
- 因此，本题的设计思路并不复杂，只需要实现“按下按钮时进行运算”这一个核心步骤即可。
- 个人觉得，问题难度在于实现的细节上，比如消抖等等。

```
1 module vending(sw,key,money,led);
2   input [4:0] sw;           //拨码开关
3   input key;               //按键
4   output [9:0] money;       //钱
5   output [1:0] led;         //LED提示信息
6
```

```

7
8 reg [9:0] money_r;
9 reg [1:0] led_r;
10
11
12 assign money=money_r;
13 assign led=led_r;
14
15 always @ (negedge key) //下降沿
16 begin
17     case(sw[4:0])
18         5'b10001: //投币5角
19             begin
20                 money_r<=money_r+5;
21                 led_r<=2'b11; //投币成功
22             end
23         5'b10010: //投币1元
24             begin
25                 money_r<=money_r+10;
26                 led_r<=2'b11; //投币成功
27             end
28         5'b10011: //投币5元
29             begin
30                 money_r<=money_r+50;
31                 led_r<=2'b11; //投币成功
32             end
33         5'b01001: //购买1.5元
34             begin
35                 if(money>=15)
36                     begin
37                         money_r<=money_r-15;
38                         led_r<=2'b11; //购买成功
39                     end
40                 else if(money_r<15)
41                     begin
42                         money_r<=money;
43                         led_r<=2'b10; //购买失败
44                     end
45                 end
46         5'b01010: //购买2.5元
47             begin
48                 if(money>=25)
49                     begin
50                         money_r<=money_r-25;
51                         led_r<=2'b11; //购买成功
52                     end
53                 else if(money_r<25)
54                     begin
55                         money_r<=money;
56                         led_r<=2'b10; //购买失败
57                     end
58                 end
59         5'b00100: //退款
60             begin
61                 money_r<=0;
62                 led_r<=2'b01; //退款提醒
63             end
64         default:

```

```

65         begin
66             money_r<=money;
67             led_r<=2'b00;
68         end
69     endcase
70 end
71
72 endmodule
73

```

- 这里采用了一个按键+五个拨码开关控制交互的模式。
- sw[4:0]的三个高位负责控制交互模式：投币、购买、退款，两个低位负责控制输入钱数。
- 由于按下开关时，按钮持续处于低电平状态，因此没办法用时钟的方式简单实现一次按键一次操作的功能，不管周期设成多少，总会出现按下一次按键多次投币的情况。
- 因此考虑用**下降沿触发**的方式控制。
- 单次按键理论上来讲只会产生一个下降沿，但是在实际使用中发现，会有很严重的多次投币现象。经过思考得知是电平抖动使得在一次按键周期中出现多个下降沿。因此考虑**消抖**
- （由于消抖主要在main模块的输入部分，因此后面再讲）

## 2.显示逻辑部分

- 基本与EDA1的十进制运算器时使用的扫描显示一致。因此就不赘述了

```

1  module show(clk,money,sig,seg,dig,led);
2      input clk;                //时钟
3      input [9:0] money;        //钱
4      input [1:0] sig;          //信号
5      output [7:0] led;         //LED
6      output [2:0] seg;         //控制哪个数码管
7      output [6:0] dig;        //数码管显示数字
8
9      reg [15:0] cnt;
10     reg [2:0] seg_r;
11     reg [6:0] dig_r;
12     reg [7:0] led_r;
13
14     reg [3:0] n1,n2,n3,nr;
15
16     assign seg=seg_r;
17     assign dig=dig_r;
18     assign led=led_r;
19
20
21
22     always @ (posedge clk)
23     begin
24         cnt<=cnt+1'b1;
25     end
26
27
28     always @ (money)
29     begin
30         n1<=money%10;          //第一位
31         n2<=money/10%10;       //第二位
32         n3<=money/100;        //第三位
33     end

```

```

34
35 always @ (n1 or n2 or n3 or cnt[14:13])
36 begin
37     case(cnt[14:13])
38     2'b00:
39     begin
40         nr<=n1;
41         seg_r<=3'b110;
42     end
43     2'b01:
44     begin
45         nr<=n2;
46         seg_r<=3'b101;
47     end
48     2'b10:
49     begin
50         nr<=n3;
51         seg_r<=3'b011;
52     end
53     default:seg_r<=3'b111;
54 endcase
55 end
56
57 always @ (nr)
58 begin
59     case(nr)
60     4'b0000: dig_r<= 7'b0000001;//0
61     4'b0001: dig_r<= 7'b1001111;//1
62     4'b0010: dig_r<= 7'b0010010;//2
63     4'b0011: dig_r<= 7'b0000110;//3
64     4'b0100: dig_r<= 7'b1001100;//4
65     4'b0101: dig_r<= 7'b0100100;//5
66     4'b0110: dig_r<= 7'b0100000;//6
67     4'b0111: dig_r<= 7'b0001111;//7
68     4'b1000: dig_r<= 7'b0000000;//8
69     4'b1001: dig_r<= 7'b0000100;//9
70     default: dig_r<= 7'b1111111;//不显示
71 endcase
72 end
73
74
75 always @ (sig)
76 begin
77     case(sig)
78     2'b00:led_r<=8'b00000000; //默认
79     2'b01:led_r<=8'b10101010; //提醒退款
80     2'b10:led_r<=8'b11110000; //购买失败
81     2'b11:led_r<=8'b01111110; //购买or投币成功
82 endcase
83 end
84 endmodule

```

### 3.消抖文件

- 本文件来自《数字电子技术基础2020秋》课程群中黄浩鹏同学的分享，非本人原创
- 注释为本人添加，属于个人理解。

```
1 module debounce(clk,key_i,key_o);
2     input  clk;           //时钟
3     input  key_i;         //按键输入
4     output key_o;         //按键输出
5
6     parameter NUMBER = 23'd10_000_000; //参数定义（原来是24'd10_000_000
7     parameter NBITS = 23;              //参数定义（原来是24
8                                         //因为在板子上试验时发现需要持续按住按键一
                                         //钟频率
                                         //段时间才能生效，可能是不同板子的时
                                         //钟频率
                                         //不一样，所以做了调整
9     reg [NBITS-1:0] count;             //计数
10    reg key_o_temp;                    //按键输出
11
12    reg key_m;                          //缓存上一时刻的按键信号
13    reg key_i_t1,key_i_t2;
14
15    assign key_o = key_o_temp;
16
17    always @ (posedge clk) begin
18        key_i_t1 <= key_i;
19        key_i_t2 <= key_i_t1;          //更新输入信号
20    end
21
22    always @ (posedge clk) begin
23        if (key_m!=key_i_t2) begin    //如果上一时刻和当前时刻信号不一致，则更新
24            key_m <= key_i_t2;        信号
25            count <= 0;                //计数归零
26        end
27        else if (count == NUMBER) begin //如果计数到达最大值，即长时间维持同一按键
28            key_o_temp <= key_m;      信号，是有效输入而不是微小抖动
29            //输出
30        end
31        else count <= count+1;        //记录上一个信号的持续时间
32    end
33 endmodule
34
```

### 4.整体实现及PIN映射

- main文件


























```
1 module main(clk,sw,key,seg,dig,led);
2     input clk;
3     input [4:0] sw;
4     input key;
5     output [2:0] seg;
6     output [6:0] dig;
```

```

7  output [7:0] led;
8
9  wire [9:0] money_o;
10 wire [1:0] sig_o;
11 wire key_o;
12 debounce debo1(clk,key,key_o);           //消抖
13 vending vender(sw,key_o,money_o,sig_o); //核心逻辑
14 show toshow(clk,money_o,sig_o,seg,dig,led); //显示
15
16 endmodule

```

#### • PIN映射

 clk	Location	PIN_152
 dig[6]	Location	PIN_56
 dig[5]	Location	PIN_68
 dig[4]	Location	PIN_65
 dig[3]	Location	PIN_55
 dig[2]	Location	PIN_70
 dig[1]	Location	PIN_52
 dig[0]	Location	PIN_63
 led[7]	Location	PIN_171
 led[6]	Location	PIN_169
 led[5]	Location	PIN_168
 led[4]	Location	PIN_167
 led[3]	Location	PIN_146
 led[2]	Location	PIN_145
 led[1]	Location	PIN_144
 led[0]	Location	PIN_143
 seg[0]	Location	PIN_49
 seg[1]	Location	PIN_50
 seg[2]	Location	PIN_51
 sw[2]	Location	PIN_166
 sw[1]	Location	PIN_161
 sw[0]	Location	PIN_160
 sw[3]	Location	PIN_164
 sw[4]	Location	PIN_174
 key	Location	PIN_131

## 四、实验中遇到的问题及解决方案

- 在运算部分，最初使用时钟方式进行交互时发现实现效果很差
  - 解决：考虑采用下降沿的方式解决。
- 最初使用下降沿的时候，是采用三个按键进行输入的，因此用三个按键的下降沿或起来，在always里判断是哪个按键被触发了，实际上是可行的逻辑，但是这时发现了电平抖动的问题，因此要考虑如何消抖，但是发现用消抖文件+这种输入模式并不能正常实现。
  - 解决：将输入改成单一按键输入，用拨动开关辅助控制输入，同时不需要在always语句里判断是哪个按键的下降沿。代码量也有一定程度上的减少。
- 在调整的过程中，本来main文件中有一个key[2:0]的输入，当把key改成单输入后，在PIN PLANNER中发现仍然保留着key[0]，并且是Input属性，但是没有出现key的可选项，本来我以为key[0]就是key，结果试了好长时间发现都不对。

- 解决：删除了key输入重新编译后发现key[0]变成Unknown属性，可以删除，再改回正常文件后，发现出现未分配的key，分配即可。