矩阵键盘扫描控制电路设计

徐震 PB20051102

实验内容

生成随时钟周期选择每一列的列选信号作为输出，键盘上的行选信号作为输入，用来实现按键数字在数码管上的显示

设计分析

生成随时钟周期选择每一列的列选信号作为输出，键盘上的行选信号作为输入，通过确认行和列的电平来判断是哪个按钮被按下

源码

```
module keyboard_xz(SW_R3,SW_R2,SW_R1,SW_R0,CLK,a,b,c,d,e,f,g,COM,SW_CO3,SW_CO2,SW_CO1,SW_CO0);
//顶层实体
   input SW_R3,SW_R2,SW_R1,SW_R0,CLK;
   output a,b,c,d,e,f,g,COM,SW_CO3,SW_CO2,SW_CO1,SW_CO0;
   wire [1:0]colum_chose,saved_colum;
   wire CLKD6;
   wire [3:0]data;
   wire [2:0]row_CHS;

   DF6 U0(CLK,CLKD6);
   counter U1(CLKD6,colum_chose);
   colum U2(colum_chose,SW_CO3,SW_CO2,SW_CO1,SW_CO0);
   row_CHS U3(SW_R3,SW_R2,SW_R1,SW_R0,row_CHS);
   Q_save_colum U4(row_CHS,CLKD6,colum_chose,saved_colum);
   judge U5(row_CHS,saved_colum,data);
   FOUR_SEVEN_DECODE U6(data,a,b,c,d,e,f,g);

   assign COM=1'b1;
endmodule
```

```
module DF6(clk,clk6d);
//分频器
   input clk;
   output clk6d;
   reg clk6d;
   reg [19:0] cnt;
   always@(posedge clk)begin
   if(cnt==20'b11111111111111111111)begin cnt<=20'b00000000000000000000;clk6d=~clk6d;end
   else cnt<=cnt+20'b00000000000000000001;
   end
endmodule
```

```
module counter(clk,colum_chose);
//计数器
   input clk;
   output [1:0]colum_chose;
   reg [1:0]colum_chose=2'b00;

   always@(posedge clk)begin
   if(colum_chose==2'b11)colum_chose<=2'b00;
   else colum_chose<= colum_chose+2'b01;
   end
endmodule
```

```
module colum(colum_chose,SW_CO3,SW_CO2,SW_CO1,SW_CO0);
//生成列选信号
   input [1:0]colum_chose;
   output SW_CO3,SW_CO2,SW_CO1,SW_CO0;
   reg SW_CO3,SW_CO2,SW_CO1,SW_CO0;

   always@(colum_chose)begin
      case(colum_chose)
         2'b00: begin SW_CO3<=1'b1;SW_CO2<=1'b1;SW_CO1<=1'b1;SW_CO0<=1'b0;end
         2'b01: begin SW_CO3<=1'b1;SW_CO2<=1'b1;SW_CO1<=1'b0;SW_CO0<=1'b1;end
         2'b10: begin SW_CO3<=1'b1;SW_CO2<=1'b0;SW_CO1<=1'b1;SW_CO0<=1'b1;end
         2'b11: begin SW_CO3<=1'b0;SW_CO2<=1'b1;SW_CO1<=1'b1;SW_CO0<=1'b1;end
      endcase
   end
endmodule
```

```
module row_CHS(SW_R3,SW_R2,SW_R1,SW_R0,row_chose);
//编码器，编码行选信号
   input SW_R3,SW_R2,SW_R1,SW_R0;
   output [2:0]row_chose;
   reg [2:0]row_chose;

   always@(SW_R3 or SW_R2 or SW_R1 or SW_R0)begin
      case({SW_R3,SW_R2,SW_R1,SW_R0})
         4'b1111: begin row_chose<=3'b100;end
         4'b1110: begin row_chose<=3'b000;end
         4'b1101: begin row_chose<=3'b001;end
         4'b1011: begin row_chose<=3'b010;end
         4'b0111: begin row_chose<=3'b011;end
      endcase
   end
endmodule
```

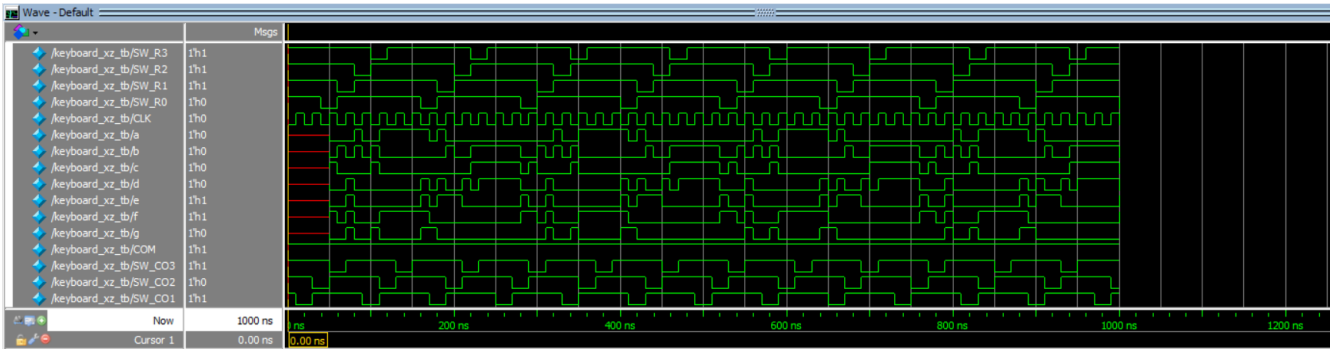
```
module Q_save_colum(row_chose,clk6d,colum_chose,saved_colum);
//边缘触发寄存器，保留状态使输出稳定
   input [2:0]row_chose;
   input [1:0]colum_chose;
   input clk6d;
   output [1:0]saved_colum;
   reg [1:0]saved_colum;

   always@(posedge clk6d)begin
      if(row_chose!=3'b100)saved_colum<=colum_chose;
   end
endmodule
```

```
module judge(row_chose,colum_chose,data);
//通过列选信号和行选信号判断按下的按键，输出是二进制编码的数字
   input [2:0]row_chose;
   input [1:0]colum_chose;
   output [3:0]data;
   reg [3:0]data;

   always@(row_chose or colum_chose)
      if(row_chose!=3'b100)begin
         if(row_chose==3'b000)
            case(colum_chose)
               2'b00: begin data<=4'b0000;end
               2'b01: begin data<=4'b0001;end
               2'b10: begin data<=4'b0010;end
               2'b11: begin data<=4'b0011;end
            endcase
         else if(row_chose==3'b001)
            case(colum_chose)
               2'b00: begin data<=4'b0100;end
               2'b01: begin data<=4'b0101;end
               2'b10: begin data<=4'b0110;end
               2'b11: begin data<=4'b0111;end
            endcase
         else if(row_chose==3'b010)
            case(colum_chose)
               2'b00: begin data<=4'b1000;end
               2'b01: begin data<=4'b1001;end
               2'b10: begin data<=4'b1010;end
               2'b11: begin data<=4'b1011;end
            endcase

         else if(row_chose==3'b011)
            case(colum_chose)
               2'b00: begin data<=4'b1100;end
               2'b01: begin data<=4'b1101;end
               2'b10: begin data<=4'b1110;end
               2'b11: begin data<=4'b1111;end
            endcase
      end
endmodule
```

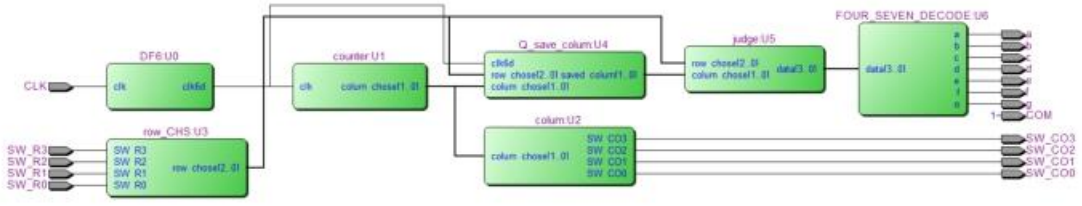
```
module FOUR_SEVEN_DECODE(data,a,b,c,d,e,f,g);
//47译码器，把二进制编码译制为数码管的控制信号
   input [3:0]data;
   output a,b,c,d,e,f,g;
   reg [6:0]final_data;

   always@(data)begin
      case(data)
         4'b0000: final_data<=7'b0000001;
         4'b0001: final_data<=7'b1001111;
         4'b0010: final_data<=7'b0010010;
         4'b0011: final_data<=7'b0000110;
         4'b0100: final_data<=7'b1001100;
         4'b0101: final_data<=7'b0100100;
         4'b0110: final_data<=7'b0100000;
         4'b0111: final_data<=7'b0001111;
         4'b1000: final_data<=7'b0000000;
         4'b1001: final_data<=7'b0000100;
         4'b1010: final_data<=7'b0001000;
         4'b1011: final_data<=7'b1100000;
         4'b1100: final_data<=7'b0110001;
         4'b1101: final_data<=7'b1000010;
         4'b1110: final_data<=7'b0110000;
         4'b1111: final_data<=7'b0111000;
      endcase
   end
   assign a=final_data[6];
   assign b=final_data[5];
   assign c=final_data[4];
   assign d=final_data[3];
   assign e=final_data[2];
   assign f=final_data[1];
   assign g=final_data[0];
endmodule
```

仿真

```
1  `timescale 10ns/1ns
2  module keyboard_xz_tb();
3     reg SW_R3,SW_R2,SW_R1,SW_R0,CLK;
4     wire a,b,c,d,e,f,g,COM,SW_CO3,SW_CO2,SW_CO1,SW_CO0;
5
6     initial CLK=0;
7     always #1 CLK=~CLK;
8
9     initial begin
10    #0 SW_R3<=1'b1;#0 SW_R2<=1'b1;#0 SW_R1<=1'b1;#0 SW_R0<=1'b1;
11    end
12    always begin
13    #4 SW_R3<=1'b1;#0 SW_R2<=1'b1;#0 SW_R1<=1'b1;#0 SW_R0<=1'b0;
14    #2 SW_R3<=1'b1;#0 SW_R2<=1'b1;#0 SW_R1<=1'b0;#0 SW_R0<=1'b1;
15    #2 SW_R3<=1'b1;#0 SW_R2<=1'b0;#0 SW_R1<=1'b1;#0 SW_R0<=1'b1;
16    #2 SW_R3<=1'b0;#0 SW_R2<=1'b1;#0 SW_R1<=1'b1;#0 SW_R0<=1'b1;
17    #2 SW_R3<=1'b1;#0 SW_R2<=1'b1;#0 SW_R1<=1'b1;#0 SW_R0<=1'b1;
18    end
19
20    keyboard_xz U0(SW_R3,SW_R2,SW_R1,SW_R0,CLK,a,b,c,d,e,f,g,COM,SW_CO3,SW_CO2,SW_CO1,SW_CO0);
21 endmodule
```



中间结果



Flow Summary	
Flow Status	Successful - Wed Nov 01 19:55:35 2023
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Full Version
Revision Name	keyboard_xz
Top-level Entity Name	keyboard_xz
Family	Cyclone V
Device	5CEFA2F23C8
Timing Models	Final
Logic utilization (in ALMs)	27 / 9,430 (< 1 %)
Total registers	27
Total pins	17 / 224 (8 %)
Total virtual pins	0
Total block memory bits	0 / 1,802,240 (0 %)
Total DSP Blocks	0 / 25 (0 %)
Total HSSI RX PCSs	0
Total HSSI PMA RX Deserializers	0
Total HSSI PMA RX ATT Deserializers	0
Total HSSI TX PCSs	0
Total HSSI PMA TX Serializers	0
Total HSSI PMA TX ATT Serializers	0
Total PLLs	0 / 4 (0 %)
Total DLLs	0 / 4 (0 %)

FPGA 验证

依次按下数字键盘，均显示按下的数字，没有抖动

Node Name	Direction	Location	I/O Bank	VREF Group	Fitter Location	I/O Standard	Reserved	Current Strength	Slew Rate
CLK	Input	PIN_W16	4A	B4A_N0	PIN_W16	2.5 V (default)		12mA (default)	
COM	Output	PIN_C11	7A	B7A_N0	PIN_C11	2.5 V (default)		12mA (default)	1 (default)
SW_CO0	Output	PIN_Y17	4A	B4A_N0	PIN_Y17	2.5 V (default)		12mA (default)	1 (default)
SW_CO1	Output	PIN_AA20	4A	B4A_N0	PIN_AA20	2.5 V (default)		12mA (default)	1 (default)
SW_CO2	Output	PIN_AB20	4A	B4A_N0	PIN_AB20	2.5 V (default)		12mA (default)	1 (default)
SW_CO3	Output	PIN_AB22	4A	B4A_N0	PIN_AB22	2.5 V (default)		12mA (default)	1 (default)
SW_R0	Input	PIN_Y21	4A	B4A_N0	PIN_Y21	2.5 V (default)		12mA (default)	
SW_R1	Input	PIN_Y19	4A	B4A_N0	PIN_Y19	2.5 V (default)		12mA (default)	
SW_R2	Input	PIN_W21	4A	B4A_N0	PIN_W21	2.5 V (default)		12mA (default)	
SW_R3	Input	PIN_V21	4A	B4A_N0	PIN_V21	2.5 V (default)		12mA (default)	
a	Output	PIN_K19	7A	B7A_N0	PIN_K19	2.5 V (default)		12mA (default)	1 (default)
b	Output	PIN_H18	7A	B7A_N0	PIN_H18	2.5 V (default)		12mA (default)	1 (default)
c	Output	PIN_K20	7A	B7A_N0	PIN_K20	2.5 V (default)		12mA (default)	1 (default)
d	Output	PIN_M18	5B	B5B_N0	PIN_M18	2.5 V (default)		12mA (default)	1 (default)
e	Output	PIN_E16	7A	B7A_N0	PIN_E16	2.5 V (default)		12mA (default)	1 (default)
f	Output	PIN_G13	7A	B7A_N0	PIN_G13	2.5 V (default)		12mA (default)	1 (default)
g	Output	PIN_G17	7A	B7A_N0	PIN_G17	2.5 V (default)		12mA (default)	1 (default)

实验总结

注意判断按键位置时应该加入边缘触发的触发器维持电路稳定