实验 4：矩阵键盘扫描控制电路设计

林莉淇 PB22051128

1 实验内容

编写 verilog 程序实现矩阵键盘扫描控制电路：四个列控制信号 C0-C3 循环输出“1110、1101、1011、0111”来驱动键盘阵列，每输出一个列序列，紧接着就读取相应的 4 个行信号。通过读取的数据或状态来判断 16 个按键中有没有按键被按下、哪个键被按下，并对其状态做编码输出。接着编写 Test Bench 仿真文件实现仿真。接着进行管脚分配，最后按动矩阵键盘按键检验数码管上的输出是否正确。

2 设计分析

需要针对序列产生器、七段数码管、分频器、按键检测器编写不同模块。思路上，序列产生器用有限状态机的方法编写。先将序列产生器与按键检测器相连，再接到七段数码管上，得到输出。其中，CLK 信号由分频器分频。

具体模块如下：

- 列扫描信号产生器
  - 输入：使复位信号 reset、时钟信号 clk（实际输入是经过分频后的 clkD）；
  - 输出：数据序列；
- 按键检测器
  - 输入：时钟信号 clk（实际输入是经过分频后的 clkD）、行信号 row、列信号 col；
  - 输出：按键编码；
- 数码管显示器
  - 输入：4 位二进制信号 q；
  - 输出：段码信号 A,B,C,D,E,F,G；
- 分频器
  - 输入：原始时钟 clk、复位信号 reset;
  - 输出：分频后 clkD;

3 Verilog 源代码

```
module scanout_llq(clk,reset,Moore);
       input clk,reset;
       output [3:0] Moore;
       reg [3:0] Moore;

       reg [1:0] st;
       parameter S0=0,S1=1,S2=2,S3=3;

       always @(posedge clk or negedge reset) begin
              if (!reset) begin
                     st <= S0;
              end
```

```
13              else begin
14                      case (st)
15                              S0:begin st <= S1;end
16                              S1:begin st <= S2;end
17                              S2:begin st <= S3;end
18                              S3:begin st <= S0;end
19                              default st <= S0;
20                      endcase
21              end
22      end
23
24      always @(st) begin
25              case (st)
26                      S0:Moore <= 4'b1110;
27                      S1:Moore <= 4'b1101;
28                      S2:Moore <= 4'b1011;
29                      S3:Moore <= 4'b0111;
30                      default Moore <= 4'b1110;
31              endcase
32      end
33  endmodule
```

代码 1: 列扫描信号产生器

```
1   module get_key_llq(clk,col,row,Moore);
2       input clk;
3       input [3:0] col;
4       input [3:0] row;
5       output [3:0] Moore;
6       reg [3:0] Moore;
7
8       always @(posedge clk) begin
9               case(row)
10                      4'b1110:
11                              case(col)
12                                      4'b1110:Moore<=4'b0001;
13                                      4'b1101:Moore<=4'b0010;
14                                      4'b1011:Moore<=4'b0011;
15                                      4'b0111:Moore<=4'b1010;
16                              endcase
17                      4'b1101:
18                              case(col)
19                                      4'b1110:Moore<=4'b0100;
20                                      4'b1101:Moore<=4'b0101;
21                                      4'b1011:Moore<=4'b0110;
22                                      4'b0111:Moore<=4'b1011;
23                              endcase
24                      4'b1011:
25                              case(col)
26                                      4'b1110:Moore<=4'b0111;
27                                      4'b1101:Moore<=4'b1000;
28                                      4'b1011:Moore<=4'b1001;
29                                      4'b0111:Moore<=4'b1100;
30                              endcase
31                      4'b0111:
32                              case(col)
33                                      4'b1110:Moore<=4'b1110;
34                                      4'b1101:Moore<=4'b0000;
35                                      4'b1011:Moore<=4'b1111;
36                                      4'b0111:Moore<=4'b1101;
37                              endcase
38              endcase
39      end
40  endmodule
```

代码 2: 读取按键模块

```
1   module Seg7Led(q,A,B,C,D,E,F,G);
2       input [3:0] q;
3       output A,B,C,D,E,F,G;
4       reg [6:0] ATOG;
5       always@(q)
6               begin
7                       ATOG = 7'b0000000;
8                       case(q)
9                               4'b0000:ATOG = 7'b0000001;
10                              4'b0001:ATOG = 7'b1001111;
11                              4'b0010:ATOG = 7'b0010010;
```

```
                        4'b0011:ATOG = 7'b0000110;
                        4'b0100:ATOG = 7'b1001100;
                        4'b0101:ATOG = 7'b0100100;
                        4'b0110:ATOG = 7'b0100000;
                        4'b0111:ATOG = 7'b0001111;
                        4'b1000:ATOG = 7'b0000000;
                        4'b1001:ATOG = 7'b0001000;
                        4'b1010:ATOG = 7'b0001000;
                        4'b1011:ATOG = 7'b1100000;
                        4'b1100:ATOG = 7'b0110001;
                        4'b1101:ATOG = 7'b1000010;
                        4'b1110:ATOG = 7'b0110000;
                        4'b1111:ATOG = 7'b0111000;
                        //default:ATOG = 7'b1111111;
                endcase
        end

        assign A = ATOG[6];
        assign B = ATOG[5];
        assign C = ATOG[4];
        assign D = ATOG[3];
        assign E = ATOG[2];
        assign F = ATOG[1];
        assign G = ATOG[0];

endmodule
```

代码 3: 数码管显示

```
module FD_llq(clk,clkD,reset);
        input clk;
        input reset;
        output clkD;
        reg [20:0] cnt = 0;
        reg clkD;
        always @(posedge clk) begin
                if(reset == 1'b0) begin
                        cnt <= 20'd0;
                        clkD <=0;
                end
                else begin
                        if(cnt==20'd100000) begin
                                cnt <=20'd0;
                                clkD <=~clkD;
                        end
                        else begin
                                cnt <=cnt+1;
                        end
                end
        end
endmodule
```

代码 4: 分频器

```
module FPGA_EXP4(clk,reset,col,row,A,B,C,D,E,F,G,LED);
        input clk,reset;
        input [3:0] row;
        output [3:0] col;
        output A,B,C,D,E,F,G,LED;

        wire [3:0] data;
        wire clkD;

        FD_llq FD(clk,clkD,reset);
        scanout_llq scanout(clkD,reset,col);
        get_key_llq get_key(clkD,col,row,data);
        Seg7Led Seg7Led(data,A,B,C,D,E,F,G);

        assign LED=1;
endmodule
```

代码 5: 顶层函数

```
`timescale 10ns/1ns
module FPGA_EXP4_tb();
        reg clk,reset;
        reg [3:0] row;
```

```
        wire [3:0] col;
        wire [3:0] data;

        initial #0 clk=0;
        initial #0 reset=0;
        initial #2 clk=0;
        initial #2 reset=1;

        always #2 clk=~clk;

        always begin
                #5 row=4'b1110; #2 row=4'b1111;
                #2 row=4'b1110; #2 row=4'b1111;
                #2 row=4'b1110; #2 row=4'b1111;
                #2 row=4'b1110; #2 row=4'b1111;

                #2 row=4'b1101; #2 row=4'b1111;
                #2 row=4'b1101; #2 row=4'b1111;
                #2 row=4'b1101; #2 row=4'b1111;
                #2 row=4'b1101; #2 row=4'b1111;

                #2 row=4'b1011; #2 row=4'b1111;
                #2 row=4'b1011; #2 row=4'b1111;
                #2 row=4'b1011; #2 row=4'b1111;
                #2 row=4'b1011; #2 row=4'b1111;

                #2 row=4'b0111; #2 row=4'b1111;
                #2 row=4'b0111; #2 row=4'b1111;
                #2 row=4'b0111; #2 row=4'b1111;
                #2 row=4'b0111; #2 row=4'b1111;

                #2 row=4'b1110; #2 row=4'b1111;
                #2 row=4'b1110; #2 row=4'b1111;
                #2 row=4'b1110; #2 row=4'b1111;
                #2 row=4'b1110; #2 row=4'b1111;
                #2 $stop;
        end
        scanout_llq scanout(clk,reset,col);
        get_key_llq get_key(clk,col,row,data);
endmodule
```

代码 6: 仿真代码

仿真了 FPGA_EXP3 模块 (去掉分频器)。0-20ns，reset=0，统一复位；20ns 之后，reset=1，使能列扫描信号产生器。

4 仿真结果记录

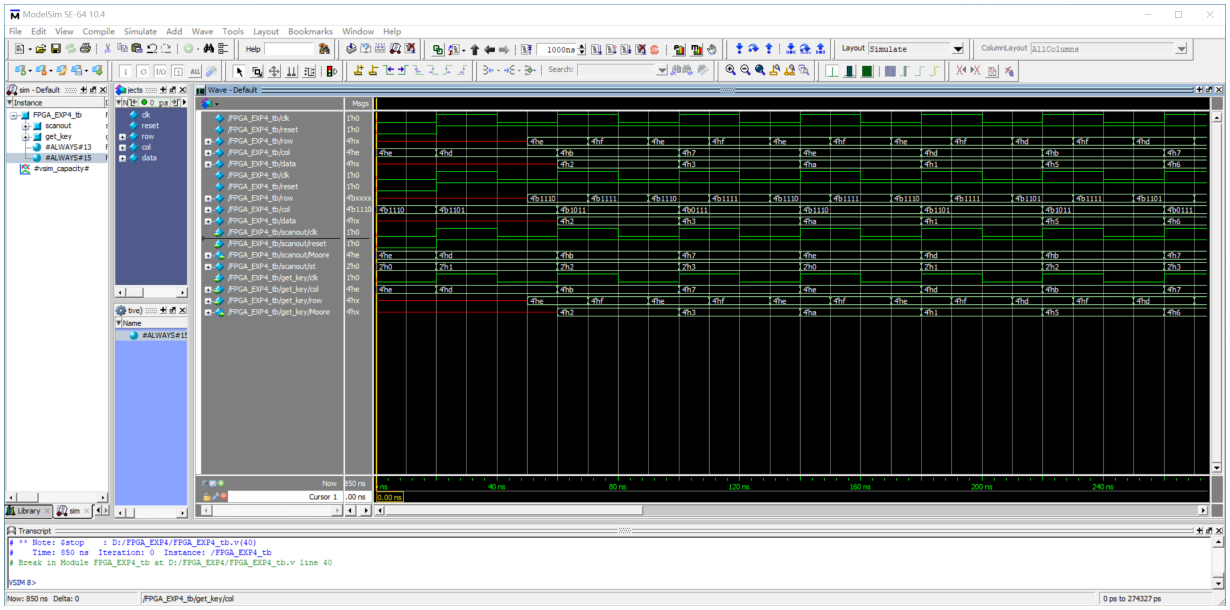

图 1: 仿真结果

由图可见：

- 行信号为 1110 时
  - 列信号为 1101，选中按键 B2;
  - 列信号为 1011，选中按键 B3;
  - ……

5 RTL 结构图

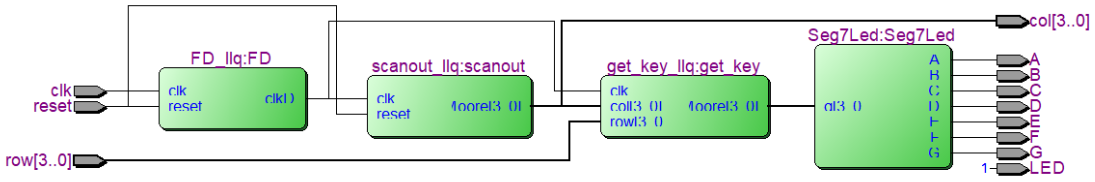

图 2: 总程序

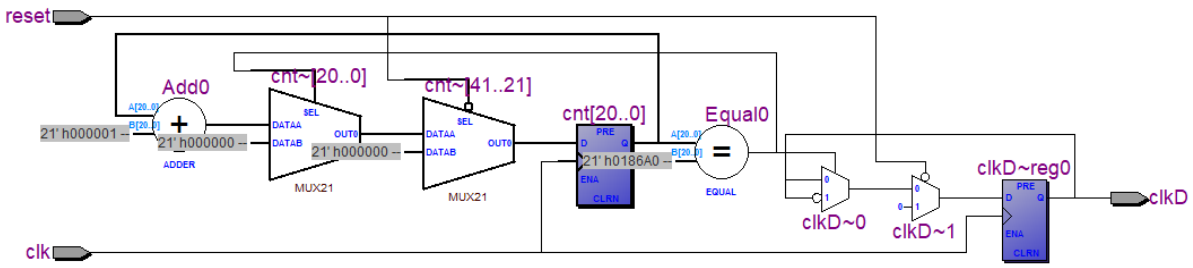

图 3: 分频器

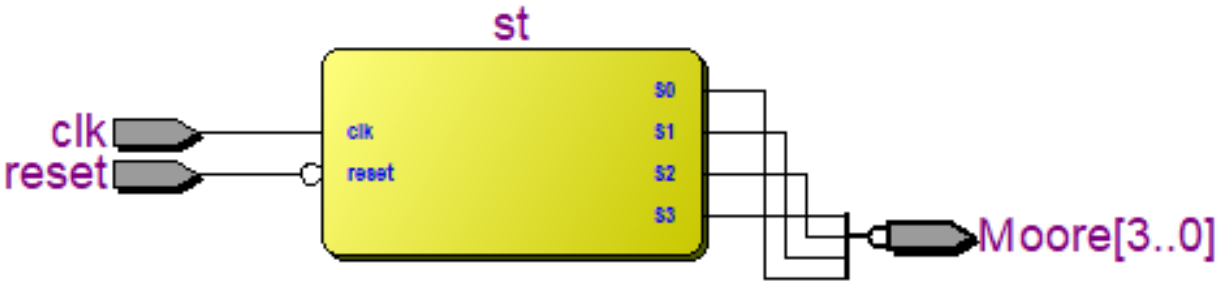

图 4: 列扫描信号产生器

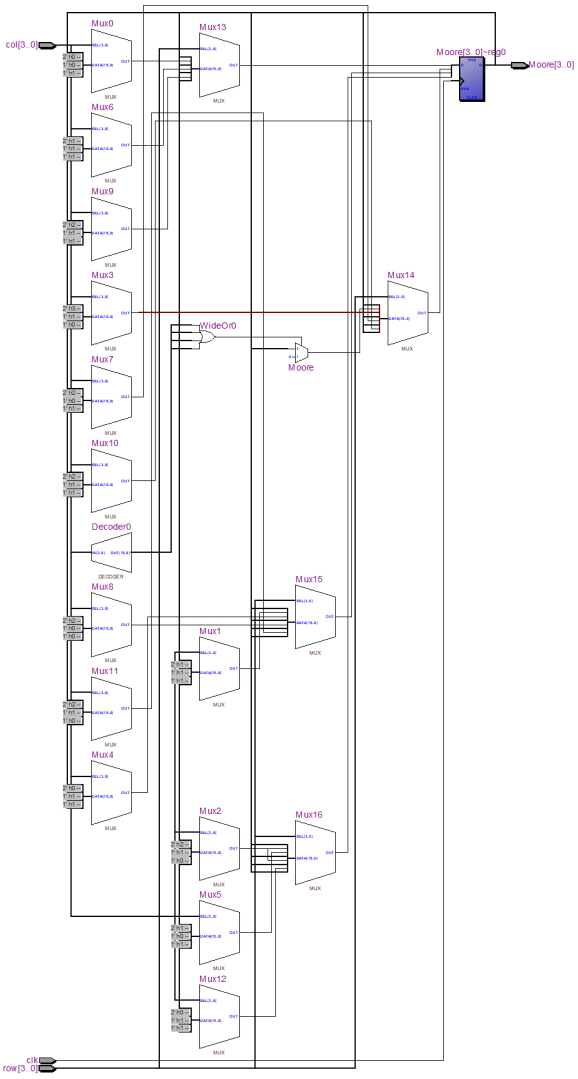

图 5: 获取按键模块

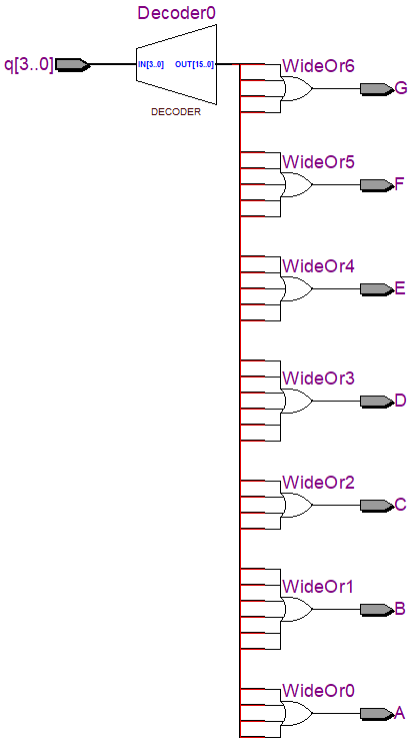

图 6: 七段数码管

6 FPGA 验证结果记录

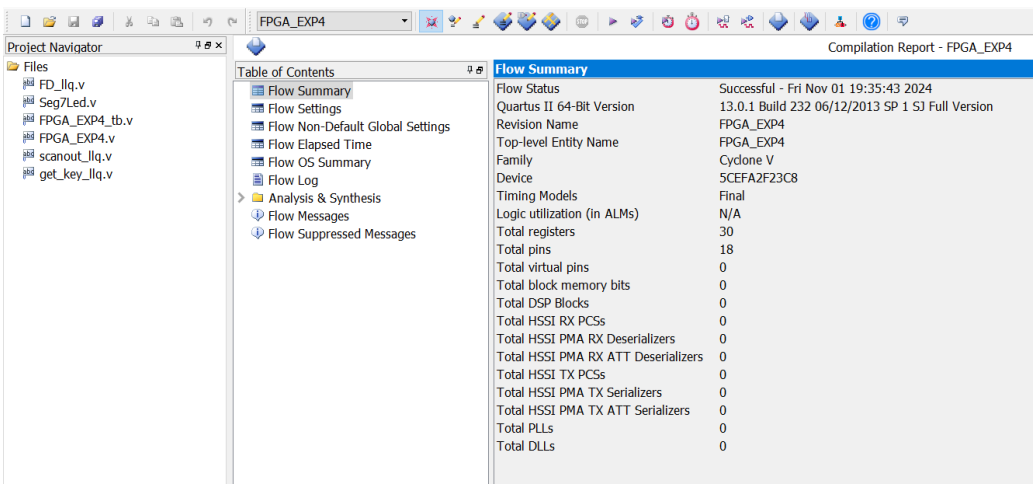

图 7: 资源占用

Node Name	Direction	Location	I/O Bank	REF Grou	O Standa
clk	Unknown	PIN_W16	4A	B4A_N0	2.5 ...ult)
A	Unknown	PIN_K19	7A	B7A_N0	2.5 ...ult)
B	Unknown	PIN_H18	7A	B7A_N0	2.5 ...ult)
C	Unknown	PIN_K20	7A	B7A_N0	2.5 ...ult)
D	Unknown	PIN_M18	5B	B5B_N0	2.5 ...ult)
E	Unknown	PIN_E16	7A	B7A_N0	2.5 ...ult)
F	Unknown	PIN_G13	7A	B7A_N0	2.5 ...ult)
G	Unknown	PIN_G17	7A	B7A_N0	2.5 ...ult)
LED	Unknown	PIN_E12	7A	B7A_N0	2.5 ...ult)
col[3]	Unknown	PIN_AB22	4A	B4A_N0	2.5 ...ult)
col[2]	Unknown	PIN_AB20	4A	B4A_N0	2.5 ...ult)
col[1]	Unknown	PIN_AA20	4A	B4A_N0	2.5 ...ult)
col[0]	Unknown	PIN_Y17	4A	B4A_N0	2.5 ...ult)
reset	Unknown	PIN_T19	5A	B5A_N0	2.5 ...ult)
row[3]	Unknown	PIN_V21	4A	B4A_N0	2.5 ...ult)
row[2]	Unknown	PIN_W21	4A	B4A_N0	2.5 ...ult)
row[1]	Unknown	PIN_Y19	4A	B4A_N0	2.5 ...ult)
row[0]	Unknown	PIN_Y21	4A	B4A_N0	2.5 ...ult)

图 8: 管脚锁定情况

动态结果的文字描述：先拨动 DIP15=0，复位。再拨动 DIP15=1 并保持不变，使能列扫描信号产生器。

按下矩阵键盘，第一个数码管会显示按下的数字。

7 实验总结

通过这次实验，我明白了分模块调用的重要性。针对不同器件写不同模块再进行调用，可以使代码结构更加清晰。

其中，我在设计保持数码管上的数字这部分代码时遇到了困难，后来发现只要在设计译码器时，不设置 default 的值，不按按键时的结果便不会记录，从而实现了数码管上数字的保持。

在仿真的时候，出现了“2，3，b，1，5，6，c，4”这样的序列，经检查调试，发现是因为行信号与列信号一起产生，输出信号读取延迟的原因，因此我把行信号延迟产生，即成功实现“2，3，a，1，5，6，b，4”这样的正确输出。

我还碰到上板时按下按键，数码管一直没有反应的情况。经检查，分频器过度分频，在按下的时候没遇到上升沿，检测不到，需要长按或修改分频系数。

在这次实验中，我收获良多。下次实验会更注意让代码结构更清晰，仿真更加全面，以便于最后的硬件验证。