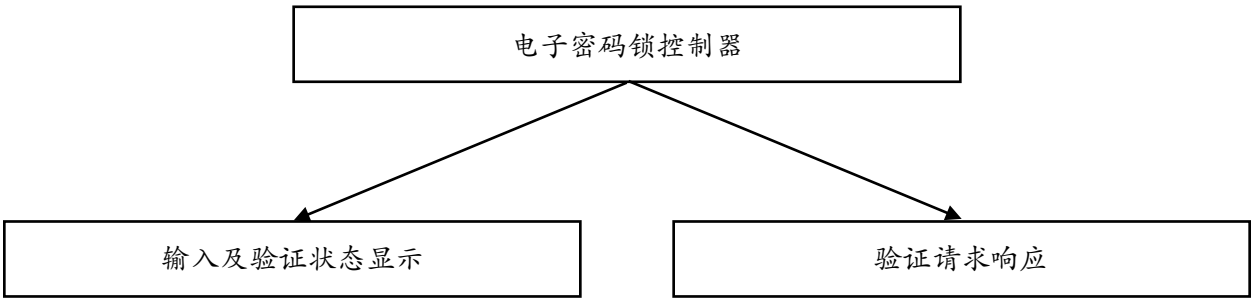
一、题目与设计目标

本次FPGA大作业旨在设计一个密码锁，以计时时钟分频后作为输入，通过拨码开关可以输入和设定密码，用微动开关触发相应操作，并能够显示输入密码的状态。即用8个拨码开关分别代表预设的密码和输入待验证的密码，输入密码的数值通过数码管实时显示。一个微动开关做为触发判断，判断结果通过数码管表示。

二、顶层逻辑设计图



三、设计方案与代码实现

由于需要使用数码管显示相关结果，因此需要完成一个用于转化设置7段码的底层模块；随后，我们定义了“设置密码”“密码清零”“验证密码”“重置显示”四个微动开关的功能，分别为——将当前拨码开关所代表的数值作为密码设定，并在数码管显示“SEt”；清空已设密码，预设为八位0，并在数码管显示“Clr”；验证当前拨码开关所代表的数值是否与已设密码相同，如相同则显示“Good”，反之则显示“Error”；重置数码管的显示状态，显示当前拨码开关所代表的数值。

除此之外，在触发微动开关之后，应保持其相应的数码管显示状态，而“重置显示”时更需要根据当前拨码开关的代表数值显示，并点亮其相应的LED灯，且以一定频率闪动。

```
1.  module locker_sub(
2.      input wire [4:0] num,
3.      output reg [6:0] a_to_g
4.      );
5.
6.      always@(*)
7.          case(num)
8.              0:  a_to_g = 7'b0000001;    //0  - '0'
9.              1:  a_to_g = 7'b1001111;    //1  - '1'
10.             2:  a_to_g = 7'b0010010;    //2  - '2'
11.             3:  a_to_g = 7'b0000110;    //3  - '3'
```

```
            4:  a_to_g = 7'b1001100;    //4  - '4'
            5:  a_to_g = 7'b0100100;    //5  - '5'
            6:  a_to_g = 7'b0100000;    //6  - '6'
            7:  a_to_g = 7'b0001111;    //7  - '7'
            8:  a_to_g = 7'b0000000;    //8  - '8'
            9:  a_to_g = 7'b0000100;    //9  - '9'
            10: a_to_g = 7'b0001000;    //10 - 'A'
            11: a_to_g = 7'b1100000;    //11 - 'B'
            12: a_to_g = 7'b0110001;    //12 - 'C'
            13: a_to_g = 7'b1000010;    //13 - 'D'
            14: a_to_g = 7'b0110000;    //14 - 'E'
            15: a_to_g = 7'b0111000;    //15 - 'F'
            20: a_to_g = 7'b1111010;    //20 - 'r'
            21: a_to_g = 7'b0110000;    //21 - 'E'
            22: a_to_g = 7'b0100100;    //22 - 'S'
            23: a_to_g = 7'b1110000;    //23 - 't'
            24: a_to_g = 7'b0100001;    //24 - 'G'
            25: a_to_g = 7'b1100010;    //25 - 'o'
            26: a_to_g = 7'b1000010;    //26 - 'd'
            27: a_to_g = 7'b0110001;    //27 - 'C'
            28: a_to_g = 7'b1110001;    //28 - 'L'
            29: a_to_g = 7'b1111111;    //29 - ''(null)
            default:
                a_to_g = 7'b1111110;    //df - '-'
        endcase
endmodule

module locker_top(
    input clk,
    input clr,
    input setting,
    input submit,
    input reset,
    input  wire [7:0] switch,
    output wire [3:0] an,
    output wire [6:0] a_to_g,
    output reg  [7:0] LED
    );

    reg [32:0]clk_cnt;
    reg [4:0] num;
    reg [7:0] password;
    reg [4:0] display_0;
    reg [4:0] display_1;
    reg [4:0] display_2;
    reg [4:0] display_3;
    reg [3:0] sw0;
    reg [3:0] sw1;
    reg [1:0] judge;
    reg [2:0] flag;
    wire control_0;
    wire control_1;

```

```
65.     always@(posedge clk)
66.     begin
67.         assign LED = switch;                    //用LED显示输入的密码
68.         assign sw0 = switch[3:0];               //低8位输入
69.         assign sw1 = switch[7:4];               //高8位输入
70.
71.         if(reset)                               //重置，数码管显示密码
72.         begin
73.             flag = 3'b000;
74.             clk_cnt = 0;
75.             case(sw0)
76.                 4'b0000: display_0 = 0;
77.                 4'b0001: display_0 = 1;
78.                 4'b0010: display_0 = 2;
79.                 4'b0011: display_0 = 3;
80.                 4'b0100: display_0 = 4;
81.                 4'b0101: display_0 = 5;
82.                 4'b0110: display_0 = 6;
83.                 4'b0111: display_0 = 7;
84.                 4'b1000: display_0 = 8;
85.                 4'b1001: display_0 = 9;
86.                 4'b1010: display_0 = 10;
87.                 4'b1011: display_0 = 11;
88.                 4'b1100: display_0 = 12;
89.                 4'b1101: display_0 = 13;
90.                 4'b1110: display_0 = 14;
91.                 4'b1111: display_0 = 15;
92.             endcase
93.
94.             case(sw1)
95.                 4'b0000: display_1 = 0;
96.                 4'b0001: display_1 = 1;
97.                 4'b0010: display_1 = 2;
98.                 4'b0011: display_1 = 3;
99.                 4'b0100: display_1 = 4;
100.                4'b0101: display_1 = 5;
101.                4'b0110: display_1 = 6;
102.                4'b0111: display_1 = 7;
103.                4'b1000: display_1 = 8;
104.                4'b1001: display_1 = 9;
105.                4'b1010: display_1 = 10;
106.                4'b1011: display_1 = 11;
107.                4'b1100: display_1 = 12;
108.                4'b1101: display_1 = 13;
109.                4'b1110: display_1 = 14;
110.                4'b1111: display_1 = 15;
111.            endcase
112.
113.        display_2 = 29;
114.        display_3 = 29;
115.        end
116.
```

```
117.        else if(setting)                        //设置密码，显示'SEt'
118.        begin
119.            flag = 3'b010;
120.            clk_cnt = 0;
121.            password = switch;
122.            display_0 = 23;
123.            display_1 = 21;
124.            display_2 = 22;
125.            display_3 = 29;
126.        end
127.
128.        else if(submit)                         //输入密码并验证
129.        begin
130.            clk_cnt = clk_cnt + 1;
131.            if(switch == password)              //正确，显示'Good'
132.            begin
133.                flag = 3'b100;
134.                display_0 = 26;
135.                display_1 = 25;
136.                display_2 = 25;
137.                display_3 = 24;
138.            end
139.
140.            else                                //错误，显示'Erro'
141.            begin
142.                flag = 3'b101;
143.                display_0 = 25;
144.                display_1 = 20;
145.                display_2 = 20;
146.                display_3 = 21;
147.            end
148.        end
149.
150.        else if(clr)                            //密码清零，显示'Clr'
151.        begin
152.            flag = 3’b110;
153.            LED = 8'b00000000;
154.            password = 8'b00000000;
155.            display_0 = 20;
156.            display_1 = 28;
157.            display_2 = 27;
158.            display_3 = 29;
159.        end
160.
161.        else                                    //不进行任何操作
162.        begin
163.            clk_cnt = clk_cnt + 1;
164.            judge = clk_cnt[25:24];
165.
166.            if(judge == 2'b11)
167.                LED = 8'b00000000;
```

```
168.
169.            if(flag == 3'b000)                //保留或更新数码管显示
170.            begin
171.                case(sw0)
172.                    4'b0000: display_0 = 0;
173.                    4'b0001: display_0 = 1;
174.                    4'b0010: display_0 = 2;
175.                    4'b0011: display_0 = 3;
176.                    4'b0100: display_0 = 4;
177.                    4'b0101: display_0 = 5;
178.                    4'b0110: display_0 = 6;
179.                    4'b0111: display_0 = 7;
180.                    4'b1000: display_0 = 8;
181.                    4'b1001: display_0 = 9;
182.                    4'b1010: display_0 = 10;
183.                    4'b1011: display_0 = 11;
184.                    4'b1100: display_0 = 12;
185.                    4'b1101: display_0 = 13;
186.                    4'b1110: display_0 = 14;
187.                    4'b1111: display_0 = 15;
188.                endcase
189.
190.                case(sw1)
191.                    4'b0000: display_1 = 0;
192.                    4'b0001: display_1 = 1;
193.                    4'b0010: display_1 = 2;
194.                    4'b0011: display_1 = 3;
195.                    4'b0100: display_1 = 4;
196.                    4'b0101: display_1 = 5;
197.                    4'b0110: display_1 = 6;
198.                    4'b0111: display_1 = 7;
199.                    4'b1000: display_1 = 8;
200.                    4'b1001: display_1 = 9;
201.                    4'b1010: display_1 = 10;
202.                    4'b1011: display_1 = 11;
203.                    4'b1100: display_1 = 12;
204.                    4'b1101: display_1 = 13;
205.                    4'b1110: display_1 = 14;
206.                    4'b1111: display_1 = 15;
207.                endcase
208.
209.                display_2 = 29;
210.                display_3 = 29;
211.            end
212.
213.            else if(flag == 3'b010)
214.            begin
215.                display_0 = 23;
216.                display_1 = 21;
217.                display_2 = 22;
218.                display_3 = 29;
219.            end
220.
```

```
            else if(flag == 3'b100)
            begin
                display_0 = 26;
                display_1 = 25;
                display_2 = 25;
                display_3 = 24;
            end

            else if(flag == 3'b101)
            begin
                    display_0 = 25;
                    display_1 = 20;
                    display_2 = 20;
                    display_3 = 21;
            end

            else if(flag == 3'b110)
            begin
                display_0 = 20;
                display_1 = 28;
                display_2 = 27;
                display_3 = 29;
            end

            else
                clk_cnt = clk_cnt;
        end
    end

    assign control_0 = clk_cnt[15];
    assign control_1 = clk_cnt[16];
    assign an[3] =  control_0 |  control_1;
    assign an[2] = ~control_0 |  control_1;
    assign an[1] =  control_0 | ~control_1;
    assign an[0] = ~control_0 | ~control_1;

    always@(*)
        case({control_1, control_0})
            3'b00: num = display_0;
            3'b01: num = display_1;
            3'b10: num = display_2;
            3'b11: num = display_3;
        endcase

    locker_sub sub(
        .num(num),
        .a_to_g(a_to_g)
    );

endmodule
```

四、实验中遇见的困难以及解决方法

1. 最开始发现当相同变量被放在同一触发条件或触发条件重叠的always语块中会报错。查阅资料后发现，每一个always语块类似于一个线程，是同步进行的，因此如果像上述那样使用，会导致同一数据同时反复被访问修改的错误。
2. 通过实践，我发现端口声明中被声明为input或inout型的端口，只能被定义为线网型（wire）；而被声明为output型的端口，则可以被定义为线网型（wire）或者寄存器型（reg）。在定义声明时如果不定义，则默认为线网型（wire）。
3. 尽管必做实验一的逻辑十分简单，但其最终作为了“重置显示”功能的基石。实际设计时，我首先完成了微动开关“设置密码”“密码清零”“验证密码”的功能，并实现了LED对拨码开关的响应，但是在实现动态显示输入数值功能的时候，遇到了很多问题。在添加这部分功能后，我发现数值虽然可以动态更新，但是“设置密码”“密码清零”“验证密码”功能的显示却出现了缺损现象。仔细研究代码逻辑后，我新增了flag寄存器变量，用于存储每一次微动开关操作后的显示状态，并增补“重置显示”功能，使得数值和响应的显示可以互不干扰。

五、实验心得

Verilog对我而言，是一门完全陌生的语言。尽管此前已经有过一定的软件编程基础，但是在刚开始接触Verilog这一硬件解释语言的时候，仍然一头雾水。之后，通过课上实现的代码以及一些参考资料，我逐步熟悉了一些常用的语句结构，并了解到了一些Verilog语言所特有的性质（即特殊的用法和执行顺序）。虽然硬件实现在写法上限制颇多，但其实现上的逻辑是异曲同工的，因此并不难适应。

不过，在功能的实现过程中切不可操之过急，不仅需要先确定一个明确的构想，还需要确定每一模块之间的相互关系和实现功能，不然混乱的逻辑会在debug的过程中带来不少麻烦，尤其是有些bug不是通过编译就能找出的。我在调试程序的过程中，就曾遇到过很多编译通过，烧入FPGA板后却不能正常实现功能的情况，清晰的逻辑能够十分有效地避免这些错误，同时提高修改和设计的效率。

非常高兴能够在这门实验课上接触到FPGA实验。硬件的编码对我来说，是一个崭新的领域。在实验过程中难免磕磕碰碰，因此也非常感谢老师与助教的耐心解释和悉心教导。