1. 实验要求

设计 25 分钟倒计时器，输入为 key1 和 key2，初始状态为数码管显示 2500。
通过 key1 控制切换计数与暂停状态。当 key1 按下时，开始倒计时，再次按下，暂停计时。
通过 key2 控制置初值。暂停状态下，当 key2 按下时，回到初始状态，数码管显示 2500。
设计暂不考虑倒计时到 0000 时的处理，要做处理需要增加输入（计数到零时的标志位）和状态（归零状态）。
提示：上节课完成的 25 分钟倒计时可以增加两个输入，一个为计数使能端，一个为置数端。
使用课前预习的状态机模块，控制 25 分钟倒计时模块。
调试时，可先使用开关作为状态机的按键输入，测试没有问题后，再加入按键控制。

2. 源代码

1) counter25_top.v

```
`timescale 1ns / 1ps
module counter25_top(
input wire clk,
input wire key1,
input wire key2,
output wire[6:0]a_to_g,
output wire[3:0]an
    );
wire jinwei;
wire a;
wire b;
wire anjian1,anjian2;

Sec_pule_generator U0(
    .clk(clk),
    .sec_pule(jinwei)
);
ButtonEdgeDetect U3(
    .clk(clk),
    .button(key1),
    .rise(anjian1)
    );
ButtonEdgeDetect U4(
    .clk(clk),
    .button(key2),
    .rise(anjian2)
    );
State U1(
    .clk(clk),
```

```
        .key1(anjian1),
        .key2(anjian2),
        .cnt_en(a),
        .load(b)
);
counter25 U2(
        .clk_sec(jinwei),
        .clk(clk),
        .cnt_en(a),
        .load(b),
        .a_to_g(a_to_g),
        .an(an)
         );
Endmodule
```

2) ButtonEdgeDetect.v

```
module ButtonEdgeDetect(
        input clk,
        input button,
        output reg rise
        );
        reg[7:0] samp;

        initial
        begin
                rise=0;

                samp=8'b0000_0000;
        end

        always @(posedge clk)
                samp<={samp[6:0],button};

        always @(posedge clk)
                if(samp==8'b0000_0001)
                        rise<=1'b1;
                else
                        rise<=1'b0;

//      always @(posedge clk)
//              if(samp==8'b1111_1110)
//                      fall<=1'b1;
//              else
//                      fall<=1'b0;
```

```
Endmodule
```

3) Sec_pul_genterator.v

```
`timescale 1ns / 1ps
module Sec_pule_generator(
input clk,
output reg sec_pule
    );
parameter M=24;
reg [M:0] cnt;
initial cnt=25'b0;
always@(posedge clk)
begin
if(cnt==10000000)
cnt<=0;
else
cnt<=cnt+1;
end
always@(posedge clk)
begin
if(cnt==10000000)
sec_pule<=1;
else
sec_pule<=0;
end
endmodule
```

4) counter25.v

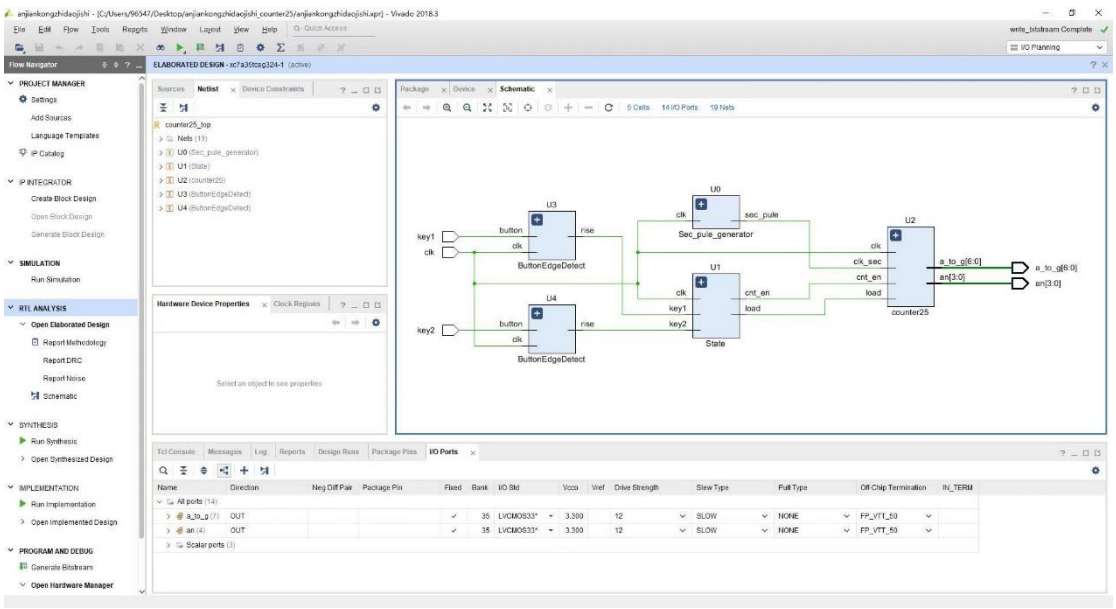
```
`timescale 1ns / 1ps
module counter25(
input wire clk,
input wire clk_sec,
input wire cnt_en,
input wire load,
output reg[6:0]a_to_g,
output reg[3:0]an
    );
reg [15:0] cnt25;
reg carry;
initial begin
    cnt25[3:0]=0;
    cnt25[7:4]=0;
    cnt25[11:8]=5;
    cnt25[15:12]=2;
end
always@(posedge clk_sec)
```

```
begin
if(cnt_en==1&&load==0)
  begin
        carry<=0;
    if(cnt25[3:0]==0&&cnt25[7:4]==0&&cnt25[11:8]==5&&cnt25[15:12]==2)
        begin
                cnt25[3:0]=9;
                cnt25[7:4]=5;
                cnt25[11:8]=4;
                cnt25[15:12]=2;
        end
      else begin
        cnt25[3:0]<=cnt25[3:0]-1;
        if(cnt25[3:0]==0)
            begin
               cnt25[3:0]<=9;
               cnt25[7:4]<=cnt25[7:4]-1;
            end
        if(cnt25[7:4]==0&&cnt25[3:0]==0)
        begin
                cnt25[7:4]<=5;
                cnt25[11:8]<=cnt25[11:8]-1;
        end
        if(cnt25[11:8]==0&&cnt25[7:4]==0&&cnt25[3:0]==0)
        begin
                cnt25[11:8]<=9;
                cnt25[15:12]<=cnt25[15:12]-1;
        end
        if(cnt25[15:12]==0&&cnt25[11:8]==0&&cnt25[7:4]==0&&cnt25[3:0]==0)
            begin
            cnt25[3:0]<=0;
            cnt25[7:4]<=0;
            cnt25[11:8]<=5;
            cnt25[15:12]<=2;
            carry<=1;
            end
        end
    end
else if(cnt_en==0&&load==0)
  begin
            cnt25[3:0]<=cnt25[3:0];
            cnt25[7:4]<=cnt25[7:4];
            cnt25[11:8]<=cnt25[11:8];
            cnt25[15:12]<=cnt25[15:12];
```

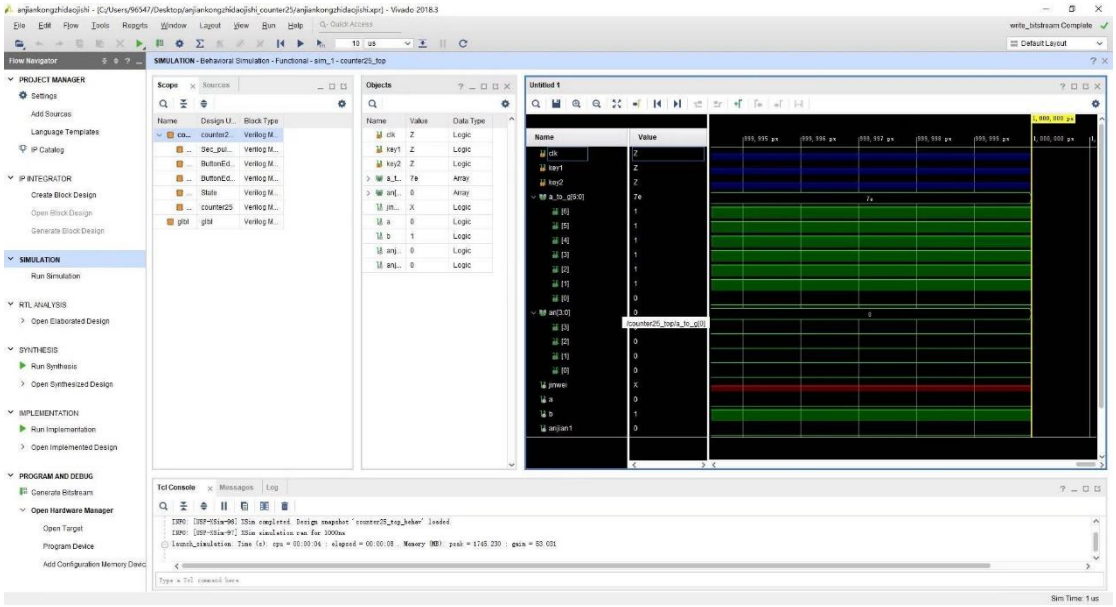
```
  end
else if(cnt_en==0&&load==1)
  begin
              cnt25[3:0]<=0;
              cnt25[7:4]<=0;
              cnt25[11:8]<=5;
              cnt25[15:12]<=2;
  end
end
wire rst_n=1'b1;
wire [15:0]x;
assign x[3:0]=cnt25[3:0];
assign x[7:4]=cnt25[7:4];
assign x[11:8]=cnt25[11:8];
assign x[15:12]=cnt25[15:12];
reg[18:0]clkdiv;
always @(posedge clk or negedge rst_n)begin
     if(!rst_n)
          clkdiv<=19'd0;
       else
          clkdiv<=clkdiv+1;
end
wire [1:0]bitcnt;
assign bitcnt=clkdiv[18:17];
always @* begin
       if(!rst_n)
            an=4'd0;
         else
            an=4'd0;
            an[bitcnt]=1;
end
reg [3:0]digit;
always@(*)begin
     if(!rst_n)
          digit=4'd0;
     else
          case(bitcnt)
             2'd0:digit=x[3:0];
             2'd1:digit=x[7:4];
             2'd2:digit=x[11:8];
             2'd3:digit=x[15:12];
          default:digit=4'd0;
          endcase
end
```

```
always @(*) begin
        if(!rst_n)
            a_to_g=7'b1111111;
        else
            case(digit)
            0:a_to_g=7'b1111110;
            1:a_to_g=7'b0110000;
            2:a_to_g=7'b1101101;
            3:a_to_g=7'b1111001;
            4:a_to_g=7'b0110011;
            5:a_to_g=7'b1011011;
            6:a_to_g=7'b1011111;
            7:a_to_g=7'b1110000;
            8:a_to_g=7'b1111111;
            9:a_to_g=7'b1111011;
            4'b1111:a_to_g=7'b0000000;
            default a_to_g=7'b1111110;
            endcase
end
endmodule
```

3. RTL 分析



4. 仿真波形



5. 实验结果

实验演示见附件中的视频