16.反应测试器

16.1实验目标

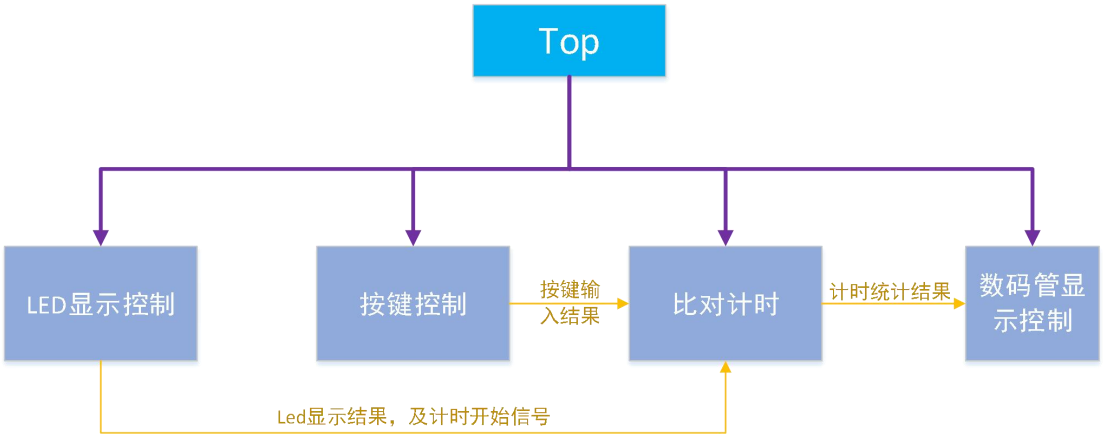
统计从人眼识别 LED 的点亮，到判断确认按下按键的时间；用于测量人体反应时间。

16.2实验要求

LD0-LD4 五个 LED 灯随机点亮一个，当看到灯亮后立刻按下相应的 S0~S4 按键。测量灯亮起到按键按下这段时间，然后将该时间以十进制的形式显示在数码管上，以 ms 为单位。

16.3实验原理

1、 产生一个随机的 led 状态，定义一个移位流水灯。触发开始后 3 秒取出一个固定的亮灯情况，并锁存住一组 led 显示状态，触发开始计时。
2、 以 ms 为单位进行计时，以触发开始信号复位计时器，计时器分个位，十位，百位，千位分别累加进行计数。
3、 数码管显示，重新测量信号触发归零，
4、 比对模块，对按键触发进行比较，比较正确退出。



16.4实验源码

顶层设计

对应外部的输入输出信号有时钟，按键，led 和数码管即可完成实验流程。

```
module lock_top(
    input         clk, //输入时钟 50MHz
    input  [4:0]   key,//5 个按键输入
    output [4:0]   led,//5 个 led 输出
    output [7:0]   smg,//8 位数码管段选
    output [3:0]   dig  //4 位数码管位选
);


//=====================================================================
    wire  [7:0] btn_deb;
    wire        restart;
    wire        det_start;
    wire        det_end;
    wire [15:0] ctrl;

//=====================================================================
    btn_deb#(       // 按键消抖
        .BTN_WIDTH     (  4'd5     ) //parameter  BTN_WIDTH = 4'd8 //按键位宽
    ) U_btn_deb
    (
        .clk           (  clk      ),//input  clk,//输入处理时钟 50MHz
        .btn_in        (  key      ),//input [BTN_WIDTH-1:0] btn_in,//输入按键信号
        .btn_deb       (  btn_deb  ) //output reg [BTN_WIDTH-1:0] btn_deb //输出按键消抖信号
    );

    assign restart = (~btn_deb[0])&(~btn_deb[4]);

    led_ctl led_ctl(
        .clk           (  clk        ),//input  clk,      //输入处理时钟 50MHz
        .restart       (  restart    ),//input restart,  //输入重新开始信号，高点平有效
        .det_start     (  det_start  ),//output reg det_start,//输出检测开始信号
        .led           (  led        ) //output reg [7:0] led //输出检测条件，以 LED 视觉触发
    );

    compare compare(
        .clk           (  clk       ),//input clk,      //输入处理时钟 50MHz
        .det_start     (  det_start  ),//input det_start,//计时开始信号，高电平有效
        .restart       (  restart    ),//input restart,  //输入重新开始信号，高点平有效
        .btn_deb       (  btn_deb    ),//input [ 7:0] btn_deb,  //输入按键信号
        .bit_sel       (  led        ),//input  [ 7:0] bit_sel,  //输入检测条件
        .det_end       (  det_end    ),//output det_end,  //输出计时结束信号
        .ctrl          (  ctrl       ) //output reg [15:0] ctrl //输出计时统计结果
    );

```

按键消抖

此模块对输入按键做消抖处理，可设置按键位宽，根据实际需求进行修改；前面章节有讲解此处不重复描述

LED 显示控制模块

模块内部有一个循环移位寄存器（1111_1110 为初始值），在开机后 3 秒触发将一个移位寄存器的值所存到 led 显示寄存器中作为反应测试的测试信号，并产生一个计时开始触发信号给到反应测试的比对计时模块；

```
1       module led_ctl(
2           input               clk,      //输入处理时钟 50MHz
3           input               restart,  //输入重新开始信号，高点平有效
4           output reg           det_start,//输出检测开始信号
5           output reg   [4:0]    led      //输出检测条件，以 LED 视觉触发
6       );
7
8
    //=============================================================================
9           reg [25:0] time_1s_cnt = 26'd0;  //0~49_999_999 1s 周期
10          always @(posedge clk)
11          begin
12              if(time_1s_cnt == 26'd4999_9999)
13                  time_1s_cnt <= `UD 26'd0;
14              else
15                  time_1s_cnt <= `UD time_1s_cnt + 26'd1;
16          end
17
18          reg [1:0] second_cnt = 2'd0;
19          always @(posedge clk)
20          begin
21              if(restart)
22                  second_cnt <= `UD 2'd0;
23              else if(time_1s_cnt == 26'd4999_9999)
24              begin
25                  if(second_cnt == 2'd3)
26                      second_cnt <= `UD second_cnt;
27                  else
28                      second_cnt <= `UD second_cnt + 2'd1;
29              end
30          end
```

```
32
    //=====================================================================
33          // 循环移位寄存器
34          reg [7:0]  led_temp = 5'b0_0001;
35          reg [9:0]  time_led_cnt=10'd0;
36          always @(posedge clk)
37          begin
38              if(time_led_cnt == 10'd579)
39                  time_led_cnt <= `UD 10'd0;
40              else
41                  time_led_cnt <= `UD time_led_cnt + 10'd1;
42          end
43
44          always @(posedge clk)
45          begin
46              if(time_led_cnt == 10'd579)
47                  led_temp <= `UD {led_temp[0],led_temp[4:1]};
48          end
49
50
    //=====================================================================
51          reg [8:0] time_cnt=9'd0;   //取出随机 led 状态
52          always @(posedge clk)
53          begin
54              time_cnt <= `UD time_cnt + 10'd1;
55          end
56
57
    //=====================================================================
58          reg   start_cnt=1'b0;  //start 计数
59          always @ (posedge clk)
60          begin
61              if(second_cnt == 2'd3 && start_cnt == 1'b0 && time_cnt == 10'd375)
62                  det_start <= `UD 1'b1;
63              else
64                  det_start <= `UD 1'b0;
65          end
66
67          always @(posedge clk)
68          begin
69              if(restart)
70                  start_cnt <= `UD 1'b0;
71              else if(second_cnt == 2'd3 && start_cnt == 1'b0 && time_cnt == 10'd375)
72                  start_cnt <= `UD start_cnt + 1'b1;
73          end
74
75          always @(posedge clk)
76          begin
77              if(restart)
78                  led <= `UD 5'b0_0000;
79              else if(second_cnt == 2'd3 && start_cnt == 1'b0 && time_cnt == 10'd375)
80                  led <= `UD led_temp;
81          end
```

比对计时模块

此模块功能为比对按键输入是否与测试信号一致并记录耗时多少 ms，由测试开始信号触发开始计时，按键检测匹配或者超时后结束 ms 的计数，输出 ms 计数结果，分个，十，百，千 4 个计数器，每个计数器范围为 0～9，到 9 后产生一个进位信号，触发更高位计数加一，同时注意计数溢出情况（超时，4 位 10 进制数超时为 9999ms），输出计时计数信号（取计时使能信号下降沿）；

```
1       module compare(
2           input               clk,      //输入处理时钟 50MHz
3           input               det_start,//输入计时开始信号，1 个时钟周期，高电平有
    效
4           input               restart,  //重新开始信号
5           input      [ 4:0]    btn_deb,  //输入按键信号
6           input      [ 4:0]    bit_sel,  //输入检测条件
7
8           output              det_end,  //输出计时结束信号
9           output reg [15:0]    ctrl      //输出计时统计结果
10      );
11
12      //=================================================================
    ===
13          //毫秒计数器
14          reg [15:0] time_ms_cnt= 16'd0;//0~49999   1ms 周期
15          always@(posedge clk)
16          begin
17              if(det_start)
18                  time_ms_cnt <= `UD 16'd0;
19              else if(time_ms_cnt == 16'd49999)
20                  time_ms_cnt <= `UD 16'd0;
21              else
22                  time_ms_cnt <= `UD time_ms_cnt + 16'd1;
23          end
24
25
    //=================================================================
26          reg    counter_en = 1'b0;
27          reg    flow = 1'b0;
28          reg    counter_en_1d = 1'b0;
29          always @(posedge clk)
30          begin
31              if(det_start)
32                  counter_en <= `UD 1'b1;
33              else if((btn_deb ==bit_sel) || flow || restart)
34                  counter_en <= `UD 1'b0;
35          end
36
37          always @(posedge clk)
38          begin
39              counter_en_1d <= `UD counter_en;
40          end
```

```
44      //=============================================================================
    =
45          //统计反应时间
46          wire [3:0] dec_single,dec_ten,dec_hundred,dec_thousand;
47          wire       dec_sigle_trg;
48          wire       carry1,carry2,carry3,carry4;//4 位进位
49
50          assign dec_sigle_trg = counter_en & (time_ms_cnt == 16'd49999);
51
52          always @(posedge clk)
53          begin
54              if(det_start)
55                  flow <= `UD 1'b0;
56              else if(carry4)
57                  flow <= `UD 1'b1;
58          end
59
60          assign ctrl ={dec_thousand,dec_hundred,dec_ten,dec_single};
61
62          dec_counter single(
63              .clk             (  clk          ),//input           clk,
64              .det_start       (  det_start    ),//input           det_start,
65              .trig            (  dec_sigle_trg ),//input          trig,
66              .flow            (  flow         ),//input           flow,
67              .carry           (  carry1       ),//output reg      carry,
68              .dec             (  dec_single   ) //output reg [3:0] dec
69          );
70
71          dec_counter ten(
72              .clk             (  clk          ),//input           clk,
73              .det_start       (  det_start    ),//input           det_start,
74              .trig            (  carry1       ),//input           trig,
75              .flow            (  flow         ),//input           flow,
76              .carry           (  carry2       ),//output reg      carry,
77              .dec             (  dec_ten      ) //output reg [3:0] dec
78          );
79
80          dec_counter hundred(
81              .clk             (  clk          ),//input           clk,
82              .det_start       (  det_start    ),//input           det_start,
83              .trig            (  carry2       ),//input           trig,
84              .flow            (  flow         ),//input           flow,
85              .carry           (  carry3       ),//output reg      carry,
86              .dec             (  dec_hundred  ) //output reg [3:0] dec
87          );
88
89          dec_counter thousand(
90              .clk             (  clk          ),//input           clk,
91              .det_start       (  det_start    ),//input           det_start,
92              .trig            (  carry3       ),//input           trig,
93              .flow            (  flow         ),//input           flow,
94              .carry           (  carry4       ),//output reg      carry,
95              .dec             (  dec_thousand ) //output reg [3:0] dec
96          );
97
98      endmodule
```

计时进位处理模块，当计数到 9 时，有触发信号到来，则产生进位信号作为下一位的触发信号，并将自身数值归零，当计数小于 9 时，有触发信号到来，则将自身数值加 1；

```
module dec_counter(
    input           clk,
    input           det_start,
    input           trig,
    input           flow,

    output reg      carry,
    output reg [3:0] dec
);

    always @(posedge clk)
    begin
        if(det_start)  //复位
            dec <= `UD 4'd0;
        else if(flow)  //溢出处理
            dec <= `UD 4'd10;
        else if(trig)
        begin
            if(dec == 4'd9)
                dec <= `UD 4'd0;
            else if(dec != 4'd10)
                dec <= `UD dec + 1'b1;
        end
    end

    always @(posedge clk)
    begin
        if(trig & dec == 4'd9)
            carry <= `UD 1'b1;
        else
            carry <= `UD 1'b0;
    end

endmodule

```

数码管显示模块

此模块功能显示计时结果，收到重新开始信号 4 个数码管显示 0000，收到计时结束信号显示对应位的计时结果，正常显示 0～9 数值，超时显示 HHHH。此模块在前面有多处用到，此处有不同的是置位和显示结果区别；

16.5实验现象

烧录固件后，第一次进入不需要触发，第二次以后进入检测需要同时按下 S0 和 S4；触发开始，开始后三秒 LD0-LD4 五个 LED 灯随机点亮一个，当看到灯

亮后立刻按下相应的按键。如果匹配 LED 灯的位置，将该时间以十进制的形式显示在数码管上，以 ms 为单位；若不匹配则不结束检测，直到计时超出 9999ms 时自动退出，数码管显示时间 9999；