7. 序列检测器

7.1 实验目的

在连续信号中，检测是否包含特定序列，例如检测“1101”中是否包含“01”

7.2 实验要求

1、拨码开关 SW7-SW4 作为序列信号输入；

2、S1-S0 按键作为特定信号输入序列，按键按下后对应的 LED 灯会亮起，表示对应位为 1，再按一下会熄灭，表示对应位为 0；

3、S2 为序列检测开始和序列检测结束按键，初次按下 S2，开始检测，此时 LED2 也会被点亮，显示当前状态，再按一下停止检测，LED2 熄灭；结束后序列串中出现特定序列的次数显示在数码管上。

7.3 实验原理

SW7～SW4 的状态为检测序列；

LED1～LED0 为特定序列；

左侧数码管显示的结果为 LED[1:0]在 SW[7:4]中出现的次数；

7.4 实验源码（完整源码查看 demo 源文件）

7.4.1 方案设计

从实验目的分析此实验的实现需要有三个功能模块：

1、按键 LED 模块；

按键调整特定序列，由按键 S[1:0]控制特定序列值；S2 控制是否检测；输出用 LED 来显示及保存特定序列，同时也将特定序列与检测使能信号传递给检测模块；

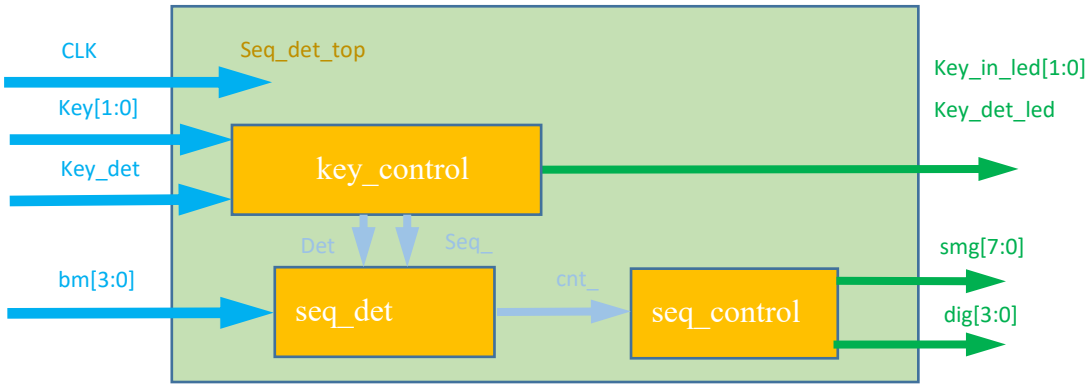
2、序列对比模块；

由拨码开关提供待检测序列，接收按键控制模块传递过来的特定序列与检测使能信号控制与待检测序列进行比较；比较结果输出给到数码管显示模块进行显示；

3、数码管控制模块

数码管显示模块的目标是将统计结果显示出来，用动态数码管显示的方式即可；

对应模块之间的连线如下框图：



7.4.2 顶层模块（含数码管显示模块）设计

```
`timescale 1ns / 1ps
`define UD #1
module top_seq_det
(
    input           clk,            // input clock    50M
    input           key_det,        // S2 input control detected enable
    input [1:0] key_in,              // S[1:0] input control detected sequence
    input [3:0] bm,                 // DIP Switch[7:4] input used to detecting
    output          key_det_led,    // display the detecte status
    output [1:0]      key_in_led, // display the detected sequence
    output reg [3:0]      dig,      // output Digital tube Bit selection
    output reg [7:0]      smg       // output Digital tube segment selection
);

    /*=============================================================
                            按键消抖及状态标记
    =============================================================*/
    wire [1:0]seq_data;
    key_control key_control
    (
        .clk                ( clk               ),
        .key_det            ( key_det           ),
        .key_in             ( key_in            ),
        .key_det_led        ( key_det_led       ),
        .key_in_led         ( key_in_led[1:0]   ),
        .seq_data           ( seq_data          )
    );

    /*=============================================================
```

```
                                序列检测
==========================================================*/
wire [3:0]data;

seq_det seq_det (
      .clk                ( clk              ),
      .key_det_led        ( key_det_led ),//检测状态标记
      .key_in_led         ( seq_data         ),//待检测序列
      .bm                 ( bm                 ),//输入序列
      .data               ( data             )
);
/*==========================================================
                                时钟分频
==========================================================*/
wire clk_1khz;
div_clk div_clk(
      .clk                ( clk              ),
      .clk_1khz           ( clk_1khz         )
);
/*==========================================================
                                数码管显示
==========================================================*/
reg   [1:0]sel=0;
wire [3:0]dig0;
wire [7:0]smg0;

always @(posedge clk_1khz)
begin
      sel <= `UD sel+1'b1;
end
// Digital Tube 0 output
seq_control seq_control_0(
      .sel(sel),
      .key(data),
      .dig(dig0),
      .smg(smg0)
);
// Digital Tube 1 output
wire [3:0]dig1;
wire [7:0]smg1;
seq_control seq_control_1
(
      .sel(sel),
      .key(4'd0),
```

```
        .dig(dig1),
        .smg(smg1)
    );
    // Digital Tube 2 output
    wire [3:0]dig2;
    wire [7:0]smg2;
    seq_control seq_control_2
    (
        .sel(sel),
        .key(4'd0),
        .dig(dig2),
        .smg(smg2)
    );
    wire [3:0]dig3;
    wire [7:0]smg3; // Digital Tube 3 output
    seq_control seq_control_3
    (
        .sel(sel),
        .key(4'd0),
        .dig(dig3),
        .smg(smg3)
    );
    // display by Digital Tube
    always @(posedge clk_1khz)
    begin
        if(sel==2'b00)
            dig <= `UD dig0;
        else if(sel==2'b01)
            dig <= `UD dig1;
        else if(sel==2'b10)
            dig <= `UD dig2;
        else if(sel==2'b11)
            dig <= `UD dig3;
    end

    always @(posedge clk_1khz)
    begin
        if(sel==2'b00)
            smg <= `UD smg0;
        else if(sel==2'b01)
            smg <= `UD smg1;
        else if(sel==2'b10)
            smg <= `UD smg2;
        else if(sel==2'b11)
```

```
                smg <= `UD smg3;
        end

    endmodule
```

7.4.3 按键 LED 控制模块

```
    `timescale 1ns / 1ps
    `define UD #1
    module key_control
    (
        input               clk,       // input clock
        input               key_det,        // S2 input
        input [1:0] key_in,       //S[2:0] input
        output reg    key_det_led,    // LED2 control sigal
        output reg [1:0]       key_in_led,// LED[1:0] control signal
        output reg [1:0]       seq_data      // Sequence to be detected
    );

    //==按键消斗=========================================
        wire [1:0] key_out;
        wire           key_det_out;
        btn_deb#(
            .BTN_WIDTH     (   4'd2                    ) ,
            .MS_20         (   20'd1_000_000           )
        ) btn_deb_key
        (
            .clk           (   clk                     ),
            .btn_in        (   key_in                  ),
            .btn_deb       (   key_out                 )
        );

        btn_deb#(
            .BTN_WIDTH     (   4'd1                    ) ,
            .MS_20         (   20'd1_000_000           )
        ) btn_deb_det
        (
            .clk           (   clk                     ),
            .btn_in        (   key_det                 ),

            .btn_deb       (   key_det_out             )
        );
```

```
reg [1:0]key_out_reg;
reg key_det_out_reg;

always @(posedge clk)
begin
    key_out_reg <= `UD key_out;
    key_det_out_reg<= `UD key_det_out;
    key_det_led <= `UD key_8_flag;
end

reg key_8_flag = 1'b0;
always @(posedge clk)
begin
    if(!key_det_out && key_det_out_reg)
        key_8_flag <= `UD ~key_8_flag;
    else
        key_8_flag <= `UD key_8_flag;
end

reg [1:0]key_flag=2'b00;
always @(posedge clk)
begin
    if(~(~key_det_led || key_8_flag)) //检测结束，序列复位
        key_flag[0] <= `UD 1'b0;
    else if(!key_out[0] && key_out_reg[0])
        key_flag[0] <= `UD ~key_flag[0];
    else
        key_flag[0] <= `UD key_flag[0];
end

always @(posedge clk)
begin
    if(~(~key_det_led || key_8_flag))//检测结束，序列复位
        key_flag[1] <= `UD 1'b0;
    else if(!key_out[1] && key_out_reg[1])
        key_flag[1] <= `UD ~key_flag[1];
    else
        key_flag[1] <= `UD key_flag[1];
end

always @(posedge clk)
begin
    key_in_led <= `UD key_flag;
```

```
        end

        //seq_data
        always @(posedge clk)




        begin
            if(key_8_flag)
                seq_data <= `UD key_in_led;
        end

    endmodule
```

注意：按键输入信号需作消抖处理，详细代码请参考按键消抖实验

7.4.4 序列检测模块设计

```
    `timescale 1ns / 1ps
    `define UD #1
    module seq_det
    (
        input                clk,
        input                key_det_led,//检测状态标记
        input [1:0]          key_in_led,//待检测序列
        input [3:0]          bm,//输入序列
        output reg [3:0]     data
    );

        // 4bit data detecte 2 bit sequence，we need compare three numbers
        reg [2:0]flag=0;
        reg          det_en=0;

        reg key_det_led_1d=0;
        always @(posedge clk)
        begin
            key_det_led_1d <= `UD key_det_led;
        end

        always @(posedge clk)
        begin
            if(~key_det_led_1d && key_det_led) //检测按键触发后才进入
```

```
                det_en <= `UD 1'b1;
        end

        always @(posedge clk)
        begin
            if(key_det_led && det_en)
            begin
                flag[0] <= `UD (bm[3:2]==key_in_led);
                flag[1] <= `UD (bm[2:1]==key_in_led);
                flag[2] <= `UD (bm[1:0]==key_in_led);
            end
        end

        always @(posedge clk)
        begin
            if(~key_det_led && key_det_led_1d)//检测结束统计结果
                data <= `UD flag[2] + flag[1] + flag[0];
        end

    endmodule
```

7.5 实验现象

实验步骤：

1、按下轻触按键 S2，进入检测状态；

2、调整输入序列，更改拨码开关的输入值（SW[7：4]）；

3、调整固定序列，通过轻触按键改变 LED 状态（LED[1:0]）；

4、按下轻触按键 S2，退出检测，查看数码管显示的统计结果，重新执行前面三个步骤；

实验现象举例：

当 SW[7:4]=4’b1010;LED[1:0]=2’b01 时，按下 S2 后数码管显示数字 1；

当 SW[7:4]=4’b1010;LED[1:0]=2’b10 时，按下 S2 后数码管显示数字 2；