# 5.序列检测器实验例程

## 5.1MES22GP 开发板简介

MES22GP 扩展底板提供了一个四位八段的共阳极数码管。（详情请查看“MES22GP 开发板硬件使用手册”）。

## 5.2实验目的

在连续信号中，检测是否包含特定序列，例如检测“11011000 “中是否包含”101”。

1、拨码开关 SW0-SW7 作为序列信号输入；

2、KEY1-KEY3 作为特定信号输入序列，KEY 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、K8 为序列检测开始和序列检测结束按键，初次按下 KEY8，开始检测，此时 LED8 也会被点亮，显示当前状态，再按一下停止检测，LED8 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。

## 5.3 实验原理

SW0～SW7 的状态为检测序列；

LED1～LED3 为特定序列；

数码管显示的结果为 LED[3:1]在 SW[7:0]中出现的次数。

## 5.4 实验源码设计

### 5.4.1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 KEY[2:0]控制特定序列值；KEY8 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；

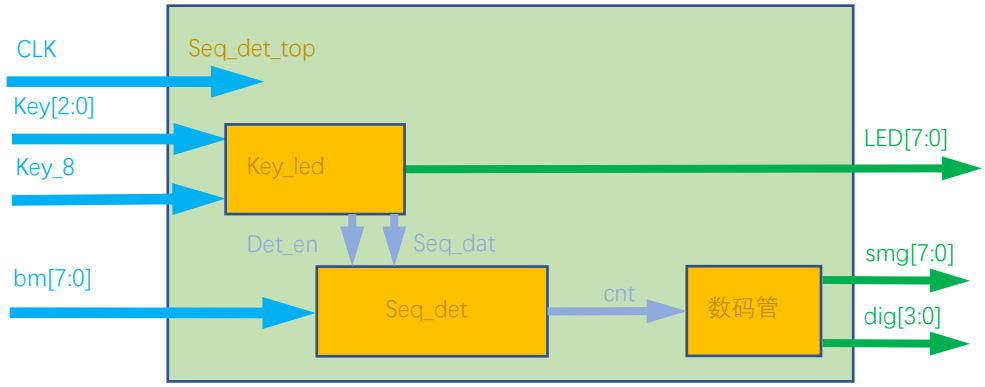
2、序列对比模块；

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代检测序列进行比较；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：



## 5.4.2顶层模块（含数码管显示模块）设计

```
1   `timescale 1ns / 1ps
2   `define UD #1
3   module top_seq_det
4   (
5       input            clk,         // input clock
6       input            key_8,       // KEY 8 input control detected enable
7       input [2:0]      key_in,      // KEY[2:0] input control detected sequence
8       input [7:0]      bm,          // DIP Switch[7:0] input used to detecting
9       output           key8_led,    // display the detecte status
10      output [6:0]     key_in_led, // display the detected sequence
11      output reg [3:0] dig,         // output Digital tube Bit selection
12      output reg [7:0] smg          // output Digital tube segment selection
13  );
14
15  /*================================================
16                      按键消斗及状态标记
17  ================================================*/
18  wire [2:0]seq_data;
19  key_control key_control
20  (
21      .clk             (  clk             ),
22      .key_8           (  key_8           ),
23      .key_in          (  key_in          ),
24      .key8_led        (  key8_led        ),
25      .key_in_led      (  key_in_led[2:0] ),
26      .seq_data        (  seq_data        )
27  );
28
29  assign key_in_led[6:3] = 4'd0;
30
31  /*================================================
32                      序列检测
33  ================================================*/
34  wire [3:0]data;
35
```

```
36  seq_det seq_det
37  (
38      .clk              (  clk        ),
39      .key8_led         (  key8_led   ),//检测状态标记
40      .key_in_led       (  seq_data   ),//待检测序列
41      .bm               (  bm         ),//输入序列
42      .data             (  data       )
43  );
44  /*=================================================
45                      时钟分频
46  =================================================*/
47  wire clk_100khz;
48  div_clk div_clk
49  (
50      .clk            (  clk       ),
51      .clk_100khz     (  clk_100khz)
52  );
53  /*=================================================
54                      数码管显示
55  =================================================*/
56  reg  [1:0]sel=0;
57  wire [3:0]dig0;
58  wire [7:0]smg0;
59
60  always @(posedge clk_100khz)
61  begin
62      sel <= `UD sel+1'b1;
63  end
64  // Digital Tube 0 output
65  seq_control seq_control_0
66  (
67      .sel(sel),
68      .key(data),
69      .dig(dig0),
70      .smg(smg0)
71  );
72  // Digital Tube 1 output
73  wire [3:0]dig1;
74  wire [7:0]smg1;
75  seq_control seq_control_1
76  (
77      .sel(sel),
78      .key(4'd0),
79      .dig(dig1),
80      .smg(smg1)
81  );
82  // Digital Tube 2 output
83  wire [3:0]dig2;
84  wire [7:0]smg2;
85  seq_control seq_control_2
86  (
87      .sel(sel),
88      .key(4'd0),
89      .dig(dig2),
90      .smg(smg2)
91  );
```

```
93  wire [3:0]dig3;
94  wire [7:0]smg3; // Digital Tube 3 output
95  seq_control seq_control_3
96  (
97      .sel(sel),
98      .key(4'd0),
99      .dig(dig3),
100     .smg(smg3)
101 );
102 // display by Digital Tube
103 always @(posedge clk_100khz)
104 begin
105     if(sel==2'b00)
106         dig <= `UD dig0;
107     else if(sel==2'b01)
108         dig <= `UD dig1;
109     else if(sel==2'b10)
110         dig <= `UD dig2;
111     else if(sel==2'b11)
112         dig <= `UD dig3;
113 end
114
115 always @(posedge clk_100khz)
116 begin
117     if(sel==2'b00)
118         smg <= `UD smg0;
119     else if(sel==2'b01)
120         smg <= `UD smg1;
121     else if(sel==2'b10)
122         smg <= `UD smg2;
123     else if(sel==2'b11)
124         smg <= `UD smg3;
125 end
126
127 endmodule
128
```

## 5.4.3按键 LED 控制模块

```
1   `timescale 1ns / 1ps
2   `define UD #1
3   module key_control
4   (
5       input               clk,       // input clock
6       input               key_8,     // KEY 8 input
7       input [2:0]         key_in,    // KEY[2:0] input
8       output reg          key8_led,  // LED8 control sigal
9       output reg [2:0]    key_in_led,// LED[2:0] control signal
10      output reg [2:0]    seq_data   // Sequence to be detected
11  );
12
```

```
13  /*==================================================
14                      按键消斗
15  ==================================================*/
16  wire [2:0]key_out;
17  btn_deb #(
18     .BT_WIDTH(4'd3)
19  ) u_btn_deb_key1 (
20      .clk(clk),
21      .btn_in(key_in),
22      .btn_out(key_out)
23  );
24
25  btn_deb #(
26     .BT_WIDTH(4'd1)
27  ) u_btn_deb_key8 (
28      .clk(clk),
29      .btn_in(key_8),
30      .btn_out(key_8_out)
31  );
32
33  /*==================================================
34  ==================================================*/
35  reg [2:0]key_out_reg;
36  reg key_8_out_reg;
37
38  always @(posedge clk)
39  begin
40      key_out_reg <= `UD key_out;
41      key_8_out_reg<= `UD key_8_out;
42  end
43
44  reg key_8_flag=0;
45  always @(posedge clk)
46  begin
47      if(!key_8_out && key_8_out_reg)
48          key_8_flag <= `UD ~key_8_flag;
49      else
50          key_8_flag <= `UD key_8_flag;
51  end
52
53  always @(posedge clk)
54  begin
55      key8_led <= `UD key_8_flag;
56  end
57
58  reg [2:0]key_flag=3'b000;
59  always @(posedge clk)
60  begin
61      if(key_8_flag==1'b0)
62          key_flag[0] <= `UD 1'b0;
63      else if(!key_out[0] && key_out_reg[0])
64          key_flag[0] <= `UD ~key_flag[0];
65      else
66          key_flag[0] <= `UD key_flag[0];
67  end
68
```

```
69  always @(posedge clk)
70  begin
71      if(key_8_flag==1'b0)
72          key_flag[1] <= `UD 1'b0;
73      else if(!key_out[1] && key_out_reg[1])
74          key_flag[1] <= `UD ~key_flag[1];
75      else
76          key_flag[1] <= `UD key_flag[1];
77  end
78
79  always @(posedge clk)
80  begin
81      if(key_8_flag==1'b0)
82          key_flag[2] <= `UD 1'b0;
83      else if(!key_out[2] && key_out_reg[2])
84          key_flag[2] <= `UD ~key_flag[2];
85      else
86          key_flag[2] <= `UD key_flag[2];
87  end
88  always @(posedge clk)
89  begin
90      key_in_led <= `UD key_flag;
91  end
92
93  //seq_data
94  always @(posedge clk)
95  begin
96      if(key_8_flag)
97          seq_data <= `UD key_in_led;
98  end
99
100 endmodule
101
```

## 5.4.4序列检测模块设计

```
1   `timescale 1ns / 1ps
2   `define UD #1
3   module seq_det
4   (
5       input            clk,
6       input            key8_led,//检测状态标记
7       input [2:0]       key_in_led,//待检测序列
8       input [7:0]       bm,//输入序列
9       output reg [3:0]  data
10  );
11
12  // 8bit data detecte 3 bit sequence, we need compare six numbers
13  reg [5:0]flag;
14
```

```
15  always @(posedge clk)
16  begin
17      if(!key8_led&&bm[7:5]==key_in_led)
18          flag[0] <= `UD 1'b1;
19      else
20          flag[0] <= `UD 1'b0;
21  end
22
23  always @(posedge clk)
24  begin
25      if(!key8_led&&bm[6:4]==key_in_led)
26          flag[1] <= `UD 1'b1;
27      else
28          flag[1] <= `UD 1'b0;
29  end
30
31  always @(posedge clk)
32  begin
33      if(!key8_led&&bm[5:3]==key_in_led)
34          flag[2] <= `UD 1'b1;
35      else
36          flag[2] <= `UD 1'b0;
37  end
38
39  always @(posedge clk)
40  begin
41      if(!key8_led&&bm[4:2]==key_in_led)
42          flag[3] <= `UD 1'b1;
43      else
44          flag[3] <= `UD 1'b0;
45  end
46
47  always @(posedge clk)
48  begin
49      if(!key8_led&&bm[3:1]==key_in_led)
50          flag[4] <= `UD 1'b1;
51      else
52          flag[4] <= `UD 1'b0;
53  end
54
55  always @(posedge clk)
56  begin
57      if(!key8_led&&bm[2:0]==key_in_led)
58          flag[5] <= `UD 1'b1;
59      else
60          flag[5] <= `UD 1'b0;
61  end
62
63  always @(posedge clk)
64  begin
65      data <= `UD flag[5] + flag[4] + flag[3] + flag[2] + flag[1] + flag[0];
66  end
67
68  endmodule
69
```

## 5.5 实验现象

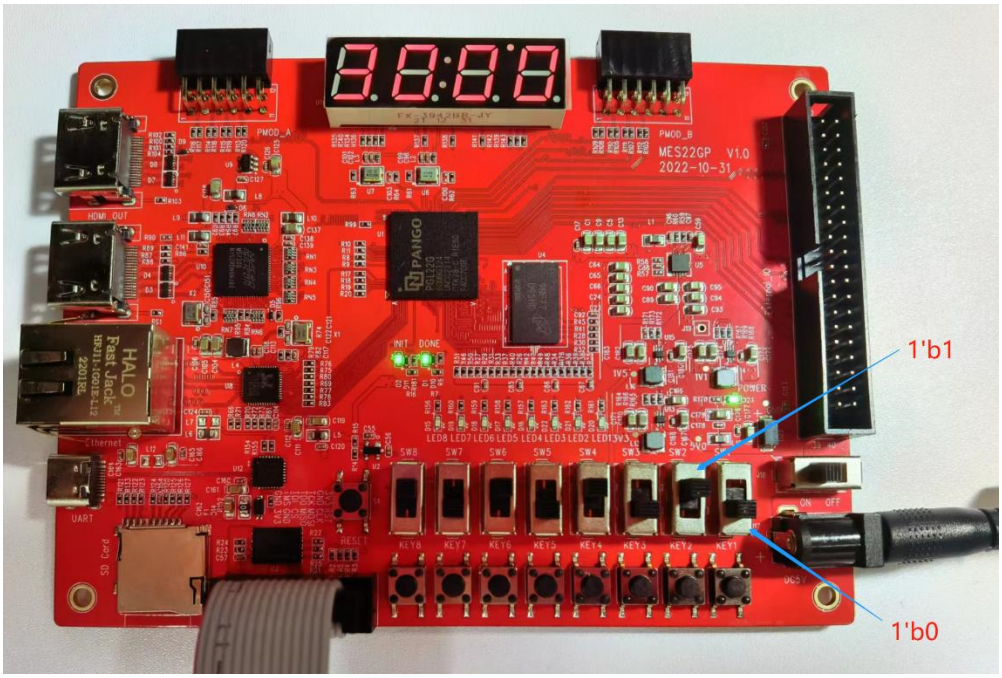
实验步骤：

1、调整输入序列，更改拨码开关 SW1-SW8 的输入值（SW[7：0]）；

2、调整固定序列，通过轻触按键 KEY1-KEY3 改变 LED 状态（LED[2:0]）；

3、按下轻触按键 KEY8，进入检测（LED8 灭），查看数码管显示的统计结果；

4、按下轻触按键 KEY8，退出检测（LED8 亮），重新执行前面三个步骤。




实验现象

当 SW[7:0]=8’ b10101010;LED[2:0]=3’ b101 时，按下 Key8 后数码管显示数字 3；

当 SW[7:0]=8’ b10101010;LED[2:0]=3’ b100 时，按下 Key8 后数码管显示数字 0；

当 SW[7:0]=8’ b11111111;LED[2:0]=3’ b111 时，按下 Key8 后数码管显示数字 6；

当 SW[7:0]=8’ b00000111;LED[2:0]=3’ b111 时，按下 Key8 后数码管显示数字 1。