中国地质大学（武汉）

大规模可编程器件实验报告

学　　院：自动化学院

课　　程：大规模可编程器件

指导老师：王广君

学　　号：20201000128

班　　级：231202

姓　　名：刘瑾瑾

2023 年 12 月 10 日

目录

实验一 键控 LED

1.1 实验目的

熟悉 Modelsim 开发软件、组合逻辑电路的设计和编程、Testbench 测试文件的编写。

1.2 实验内容

键控 LED 灯仿真，设计一个键控 LED 灯，要求具有以下功能：
(1)按键 KEY 按下时，8 个 LED 灯全灭；
(2)按键 KEY 弹起时，8 个 LED 灯全亮。

1.3 程序设计

使用一个按键控制八个灯的亮灭，即把按键的状态 2 位作为输入控制 8 个灯的亮灭。

```
1. module test1(ledr,clk);
2.  output[7:0] ledr;
3.  reg[7:0] ledr;
4.  input clk;
5.  always@(clk)
6.   begin
7.    if(!clk) ledr<=8'b00000000;
8.    else ledr<=8'b11111111;
9.   end
10. endmodule
```

1.4 软件仿真

初始按键值为 0，按键值每 10ps 翻转，模拟按键按下和弹起的情况。 编写 Test Bench 文件代码如下：

```
1. `timescale 1 ps/ 1 ps
2. module test1_vlg_tst();
3. reg eachvec;
4. reg clk;
5. wire [7:0]  ledr;
6. test1 i1 (
```

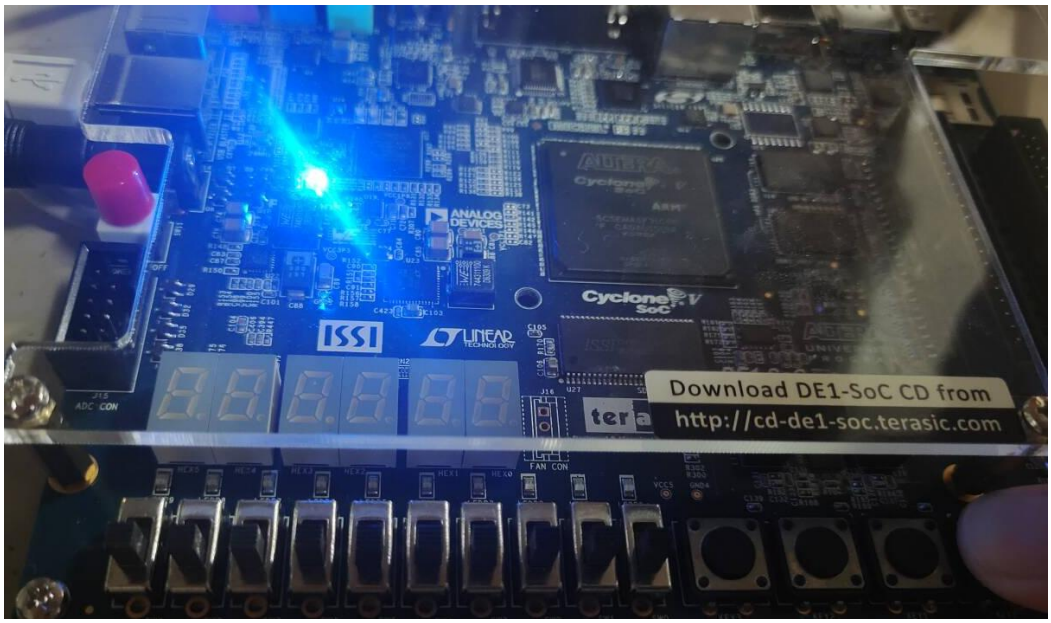
```
7.  .clk(clk),
8.  .ledr(ledr)
9. );
10. initial
11. begin
12.     clk=0;
13.   forever #10 clk=~clk;
14. $display("Running testbench");
15. end
16. endmodule
```

运行 ModelSim，仿真结果如下图所示：

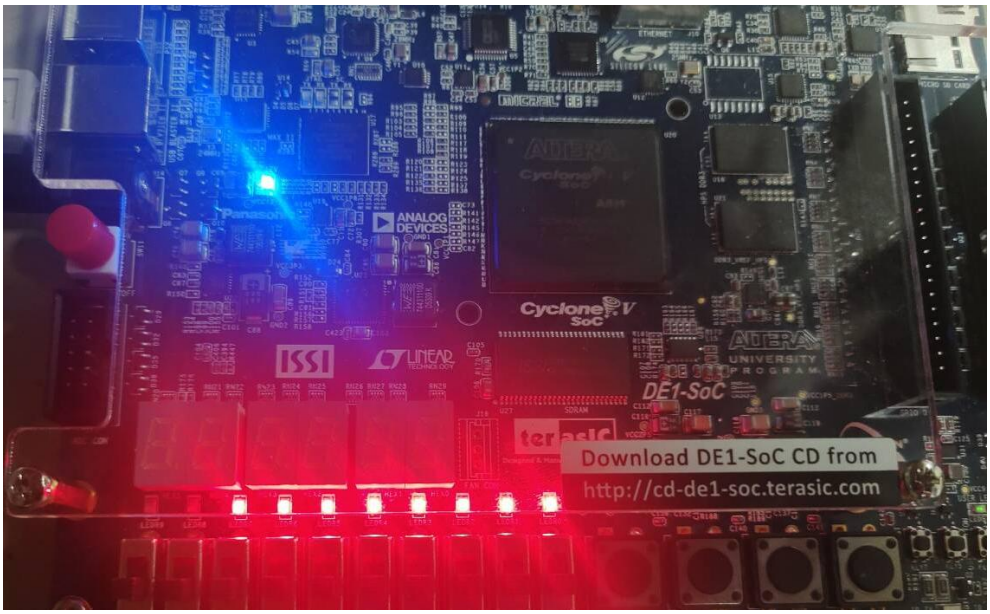
图 1- 1 键控 LED 灯 ModelSim 仿真波形图

1.5 实验结果

按下按键时，八个 LED 灯全灭：



放松按键时，LED 灯全亮：



实验二 跑马灯

2.1 实验目的

熟悉 Verilog 编程语言，掌握 Modelsim 仿真方法、时序逻辑电路设计和编程方法。

2.2 实验内容

设计一个跑马灯，要求跑马灯系统包括 8 个 LED 灯，系统开始工作后要求 8 个 LED 灯从左到右、右到左依次逐个亮灯，每次只有 1 个灯亮，其他 7 个灯灭，相邻两个 LED 灯亮灯的间隔时间为 0.5s。

2.3 程序设计

由于输入时钟频率为 50MHz，若实现亮灯间隔为 0.5s，则需计数值为 25000000。设计计数器 count，在计算从左至右或者从右至左同一个方向时，最大计数值为 2500000x8=200000000，对单个间隔计数值进行求余，余数为 0 即执行移位操作，移位值为单个间隔计数值的商的整数值，即可实现单方向跑马灯。设计状态位 state 标记跑马灯方向：0 为从左至右，1 为从右至左，控制跑马灯方

向，实现跑马灯的循环往复。

```
1. module test2(ledr,clk);
2. output[7:0] ledr;
3. reg[7:0] ledr;
4. input clk;
5. reg[29:0] count;
6. reg[0:0] state;
7. initial
8.  begin
9.  count=0;
10.  state=0;
11.  ledr=00000000;
12.  end
13. always@(posedge clk)
14.  begin
15.  count<=count+1;
16.  if(count==199999999)
17.   begin
18.    count<=0;
19.    state<=~state;
20.   end
21.  if(!(count%25000000))
22.   begin
23.    if(!state)
24.    ledr<=(8'b10000000>>(count/25000000));
25.    else
26.    ledr<=(8'b00000001<<(count/25000000））；
27.   end
28.  end
29. endmodule
```

2.4 软件仿真

由于 0.5s 对应的计数值为 25000000，所以在仿真时，设定两个 LED 灯亮灯的间隔时间为 8 个时钟周期。初始时钟值为 0，按键值每 10ps 翻转一次。

```
1. `timescale 1 ps/ 1 ps
2. module test2_vlg_tst();
3. reg eachvec;
4. reg clk;
5. wire [7:0]  ledr;
6. test2 i1 (
7.  .clk(clk),
8.  .ledr(ledr)
```

```
9. );
10. initial
11. begin
12.      clk=0;
13.   forever #10 clk=~clk;
14. $display("Running testbench");
15. end
16. endmodule
```

运行 ModelSim，仿真结果如下图所示：

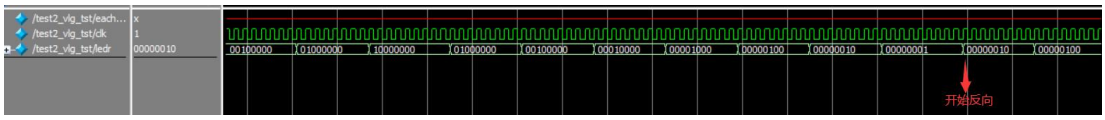
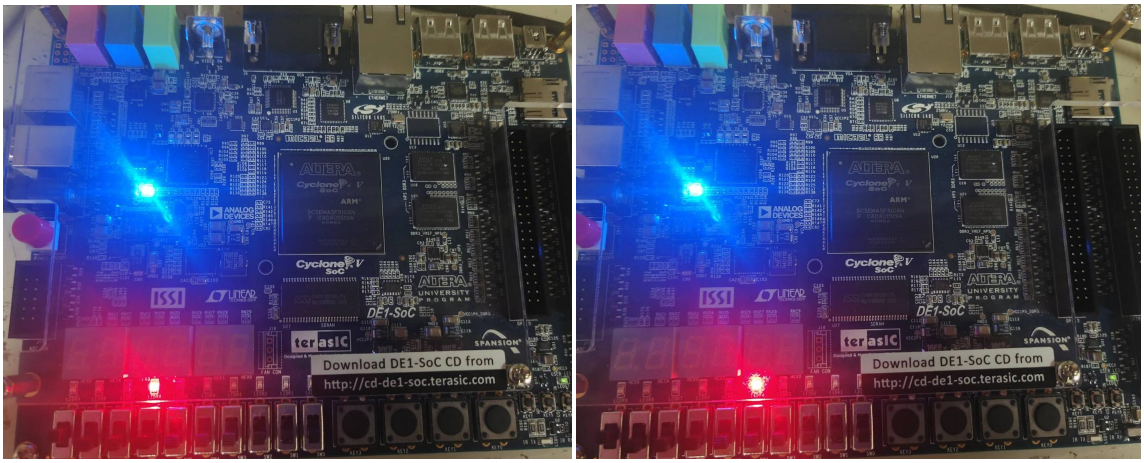


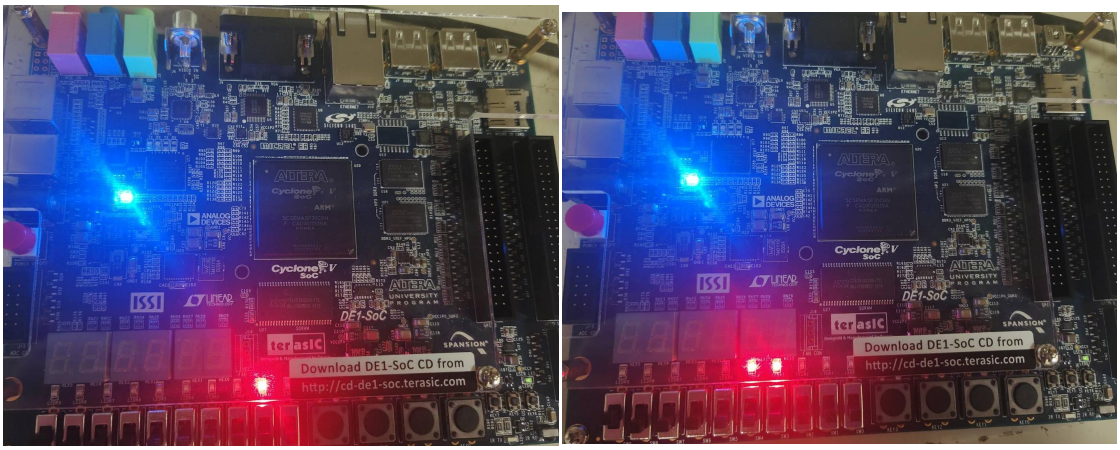
图 2- 1 跑马灯 ModelSim 仿真波形图

2.5 实验结果

当 state=0 时，跑马灯从左至右：



当 state=1 时，跑马灯从右至左：



实验三 序列检测器

3.1 实验目的

熟悉 Verilog 编程语言，掌握 Modelsim 仿真方法、状态机设计方法。

3.2 实验内容

序列检测器的设计。利用状态机设计一个序列检测器，序列检测器功能： 将一个指定序列从数字码流中识别出来。本实验要求设计一个“10010”序列的检测器。设 X 为数字码流的输入，Z 为检测出标记输出，Z 平时为高电平，一旦发现指定的序列 10010，则变为低电平。例如 X 码流 110010010000100101....，则该序列检测器将在第九个比特位后检测到“10010”，然后将 Z 置为低电平。

利用两个按键，分别输入 0 或者 1，实现序列输入。实验中利用 LED 来替代 Z，一旦检测到“10010”，LED 亮，否则 LED 灭

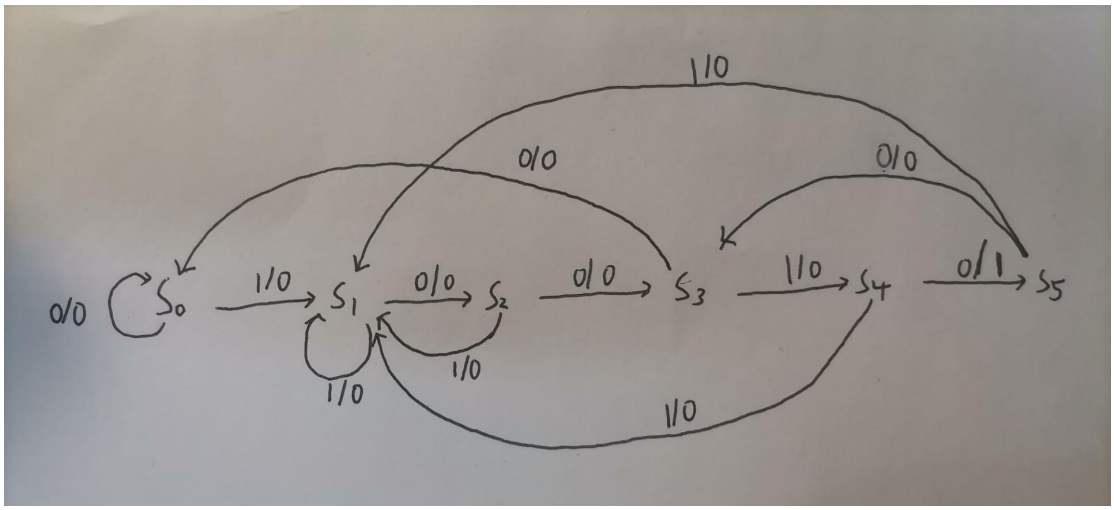


图 3- 1 序列检测器原理图

3.3 程序设计

本实验的程序分为三个文件。以下是 key_handle.v 文件代码，功能是对 FPGA 的 50MHz 时钟进行分频，然后通过连续三次判断按键状态，实现按键消抖。

```
1. module  key_handle(
2.     input clk,//50MHz 时钟
3.     input key,//原始的按键输入
4.     input nRST,
```

```
    output clk_5KHz,
    output reg key_o //经过消抖后的按键输出
    );
reg [20:0]cnt;//定义一个 21 位宽的计数器
reg  key_d1;//定义一个位宽为 1 位的寄存器
reg  key_d2;//定义一个位宽为 1 位的寄存器
//模块一：时钟分频模块
always @(posedge clk or negedge nRST)
begin
 if(!nRST)
  cnt <= 21'd0;//复位
 else
  begin
   if(cnt<21'd9999)
    cnt <= cnt + 1'b1;
   else
    cnt <= 21'd0;
  end
end
assign  clk_5KHz = (cnt<21'd5000)? 1'b1:1'b0;
//模块二：移位模块
always @(posedge clk_5KHz or negedge nRST)
begin
 if(!nRST)
  begin
   key_d1 <= 0;
   key_d2 <= 0;
  end
 else
  begin
   key_d1 <= key;//D 触发器
   key_d2 <= key_d1;//D 触发器
  end
end
//模块三：按键消抖模块
always @(posedge clk_5KHz or negedge nRST)
begin
 if(!nRST)
  key_o <= 1'b0;
 else
  begin
   if(key_d1 & key_d2 & key)
    key_o <= 1'b1;
   else if(!key_d1 & !key_d2 & !key)
```

```
49.     key_o <= 1'b0;
50.   end
51. end
52. endmodule
```

以下为 check.v 文件代码，功能是通过 Verilog 程序构建 D 触发器实现按键的下降沿检测，同时利用有限状态机实现序列检测功能。

```
1. module check(clk,rst,high,low,y,flag);
2. input clk,rst,high,low;
3. output[4:0]y;
4. output flag;
5. reg[4:0]y;
6. reg flag;
7. reg high_d;
8. reg low_d;
9. always@(posedge clk)
10.  begin
11.   high_d<=high;
12.   low_d<=low;
13.  end
14. parameter S0=5'b00000,S1=5'b00001,S2=5'b00010,S3=5'b00100,S4=5'b01
    001,S5=5'b10010;
15.
16. always @(posedge clk or negedge rst)
17. begin
18. if(!rst)
19.  begin
20.  flag<=0;//指示信号复位
21.    y<=S0;//状态变量复位
22.  end
23. else
24.
25. case(y)
26.     S0:
27.   begin
28.   flag<=1'b0;
29.   if (high_d&!high) y <= S1;
30.     else y <= y;
31.   end
32.     S1:
33.   begin
34.   flag<=1'b0;
35.   if (high_d&!high) y <= S1;
36.    else if(low_d&!low) y<=S2;
37.    else y <= y;
```

```
38.  end
39.    S2:
40.  begin
41.  flag<=1'b0;
42.  if (high_d&!high) y <= S1;
43.   else if(low_d&!low) y<=S3;
44.   else y <= y;
45.  end
46.    S3:
47.  begin
48.  flag<=1'b0;
49.  if (high_d&!high) y <= S4;
50.   else if(low_d&!low) y<=S0;
51.   else y <= y;
52.  end
53.  S4:
54.  begin
55.  flag<=1'b0;
56.  if (high_d&!high) y <= S1;
57.   else if(low_d&!low) y<=S5;
58.   else y <= y;
59.  end
60.  S5:
61.  begin
62.      flag<=1'b1;
63.  if (high_d&!high) y <= S1;
64.   else if(low_d&!low) y<=S3;
65.   else y <= y;
66.  end
67.  default: y <= S0;
68. endcase
69. end
70. endmodule
```

以下为 test3.v 文件代码，是本工程的顶层模块，功能是将上面两个文件中的模块连接起来，调用两个模块，完成实验要求功能。

```
1. module test3(
2. clk_50MHz, key_A, key_B, nRST, state, LED);
3. input wire clk_50MHz;
4. input wire key_A;
5. input wire key_B;
6. input wire nRST;
7. output wire LED;
8. output wire [4:0]state;
9. wire clk_5KHz;
```

```
wire key1;
wire key2;
key_handle u1(
.clk(clk_50MHz),
.key(key_A),
.nRST(nRST),
.clk_5KHz(clk_5KHz),
.key_o(key1));
key_handle u2(
.clk(clk_50MHz),
.key(key_B),
.nRST(nRST),
.key_o(key2));
check u3(
.clk(clk_5KHz),
.rst(nRST),
.high(key1),
.low(key2),
.y(state),
.flag(LED));
endmodule
```

3.4 软件仿真

考虑到仿真时，按键的输入信号是不含任何抖动的，不需要进行消抖。因此，这里跳过了按键消抖模块，仅对状态机模块进行仿真验证。将 check.v 设置为顶层模块，在编写 Test Bench 文件时，只对 check 模块进行测试。

```
`timescale 1 ps/ 1 ps
module test3_vlg_tst();
reg clk;
reg high;
reg low;
reg rst;
wire flag;
wire [4:0] y;
check i1 (
.clk(clk),
.flag(flag),
.high(high),
.low(low),
.rst(rst),
.y(y));
```

```
16. initial
17. begin
18. clk=1'b0;
19. rst=1'b0;
20. high=0; low=0;
21. #10 rst=1'b1;
22. #10 high=1; low=0;
23. #10 high=0; low=0;
24. #10 high=0; low=1;
25. #10 high=0; low=0;
26. #10 high=0; low=1;
27. #10 high=0; low=0;
28. #10 high=1; low=0;
29. #10 high=0; low=0;
30. #10 high=0; low=1;
31. #10 high=0; low=0;
32. end
33. always
34. #5 clk=~clk;
35. endmodule
```

运行 ModelSim，仿真结果如下图所示：

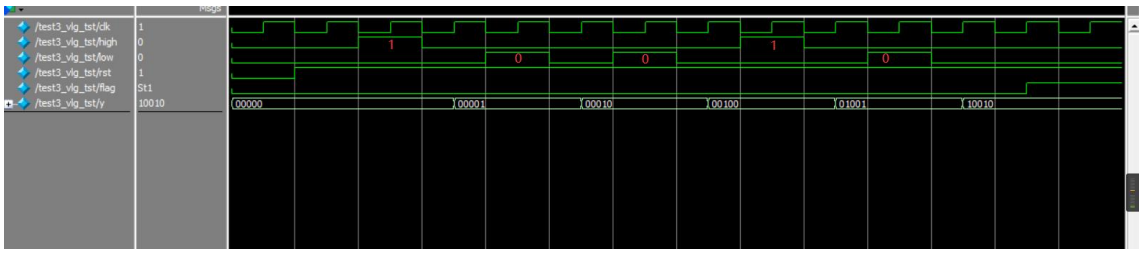
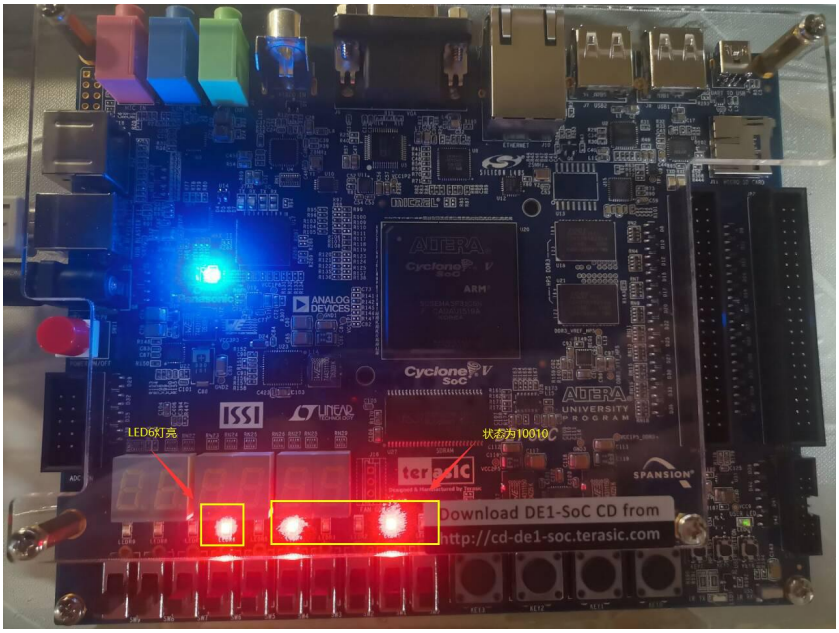


图 3- 2 序列检测器 ModelSim 仿真波形图

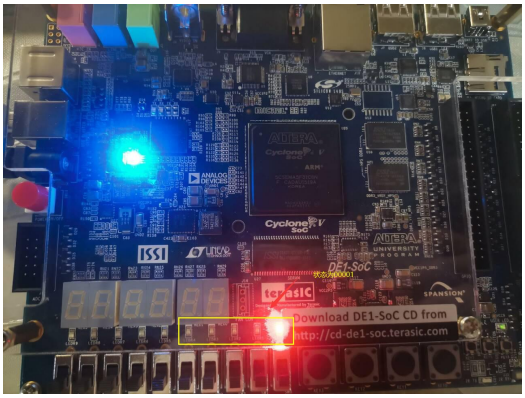
3.5 实验结果

使用 KEY0 作为复位键，KEY1 作为高电平输入，按下一次，输入一个 1；KEY2 作为低电平输入，按下一次，输入一个 0。输出信号 LED 使用 LED6，状态输出使用 LED0~LED4，方便观察状态机的状态。输入时，LED0~LED4 显示状态，若输入序列中含有“10010”，指示灯 LED6 点亮，否则，指示灯 LED6 熄灭。

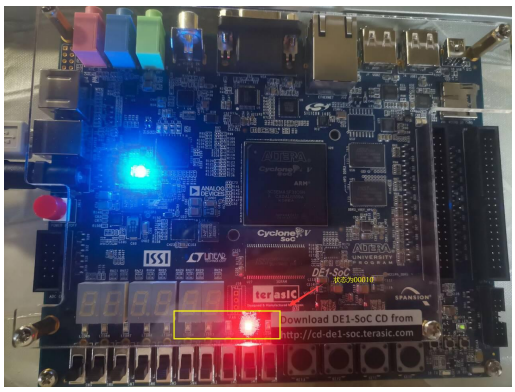
若输入序列中含有“10010”，则 LED6 灯点亮：



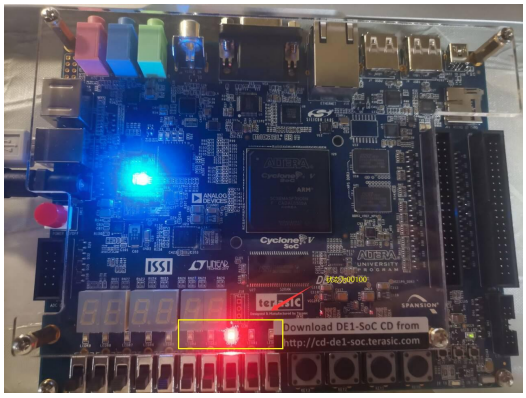
若输入序列后，状态为“00001”、“00010”、“00100”和“01001”，则 LED6 熄灭：



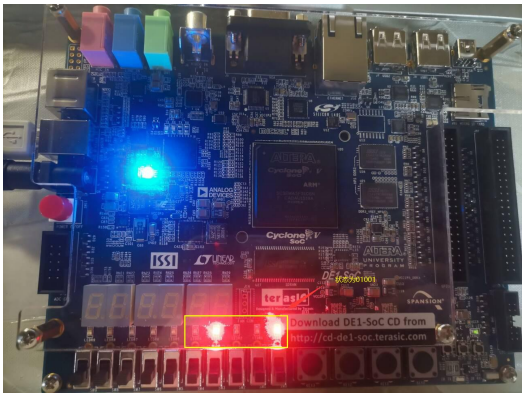
状态 00001



状态 00010



状态 00100



状态 01001

实验四 变速数字时钟

4.1 实验目的

熟悉 Verilog 编程语言，掌握 Modelsim 仿真方法、数码管显示电路控制方法、综合电路分模块设计方法和测试方法。

4.2 实验内容

设计一个变速数字时钟，要求数字时钟的速度有三个档位:

第一个档位为标准数字时钟，每隔 1S 秒计数器加 1;

第二个档位为快速数字时钟，每隔 0.1S 秒计数器加 1;

第三个档位为超快速数字时钟，每隔 0.01S 秒计数器加 1。

三个档位可用按键切换；除此之外，时钟具备按键清零功能;具有整点报时功能，即在 59 分 59 秒时给出指示信息(LED 灯亮)，持续时间为 1s/0.1s/0.01s,指示信号结束的时刻恰好为正点时刻。

4.3 程序设计

总体程序由五个模块组成：按键消抖模块、时钟分频模块、数码管驱动选择模块、数码管计数显示模块和模式选择模块。

4.3.1 按键消抖模块

程序前端的按键消抖模块使用一个时钟分频模块将系统 50MHz 时钟分频获得 5KHz 时钟，用于按键消抖模块计数时钟使用，模块输出经消抖后的按键信号至时钟分频模块与数码管驱动选择模块，按键消抖模块 eliminate_jitters 如下：

```
1. module  eliminate_jitters(
2.     input clk,          //50MHz 时钟
3.     input key,          //原始的按键输入
4.     input nRST,
5.     output clk_5KHz,
6.     output reg key_o     //经过消抖后的按键输出
7.     );
8.
9. reg [20:0]cnt;        //定义一个 21 位宽的计数器
10. reg  key_d1;         //定义一个位宽为 1 位的寄存器
```

```
reg  key_d2;        //定义一个位宽为 1 位的寄存器
//模块一：时钟分频模块
always @(posedge clk or negedge nRST)
begin
 if(!nRST)
  cnt <= 21'd0;//复位
 else
  begin
   if(cnt<21'd9999)
    cnt <= cnt + 1'b1;      //计数器在小于 9999 时，每来一个时钟，就加 1
   else
    cnt <= 21'd0;      //当计数器等于 9999 时，清零
  end
end
assign  clk_5KHz = (cnt<21'd5000)? 1'b1:1'b0;  //得到一个 5KHz
    （50MHz/10000）的分频时钟
//模块二：移位模块
always @(posedge clk_5KHz or negedge nRST)
begin
 if(!nRST)
  begin
   key_d1 <= 0;
   key_d2 <= 0;
  end
 else
  begin
   key_d1 <= key;              //D 触发器
   key_d2 <= key_d1;            //D 触发器
  end
end
//模块三：按键消抖模块
always @(posedge clk_5KHz or negedge nRST)
begin
 if(!nRST)
  key_o <= 1'b0;
 else
  begin
   if(key_d1 & key_d2 & key)
    key_o <= 1'b1;
   else if(!key_d1 & !key_d2 & !key)
    key_o <= 1'b0;
  end
end
endmodule
```

4.3.2 时钟分频模块

系统使用时钟分频模块对系统时钟分频，用于时钟计数，使得计数间隔分别为 1s、0.1s、0.01s,以下为时钟分频模块 time_devide_timer 内容：

```
1. module time_devide_timer(
2.     input clk,          //50MHz 时钟
3.     input switch_1,    // 切换频率
4.     input switch_2,
5.     output clk_timer
6.     );
7.
8. reg [24:0]count_1 = 25'd0;//计数器
9. reg [24:0]count_10 = 25'd0;
10. reg [24:0]count_100 = 25'd0;
11.
12. reg clk_1_r = 0;//控制电平
13. reg clk_10_r = 0;
14. reg clk_100_r = 0;
15. reg clk_timer_r =0;
16. assign clk_timer = clk_timer_r;
17. ////////////////////////////////////////
18. always @(posedge clk)//0.5s 反转一次
19. begin
20.  if(count_1 == 25'd24999999)
21.  begin
22.   count_1 <= 25'd0;
23.   clk_1_r <= ~clk_1_r;
24.  end
25.  else
26.   count_1 <= count_1 + 1;
27. end
28.
29. always @(posedge clk)//0.05s 反转一次
30. begin
31.  if(count_10 == 25'd2499999)
32.  begin
33.   count_10 <= 25'd0;
34.   clk_10_r <= ~clk_10_r;
35.  end
36.  else
37.   count_10 <= count_10 + 1;
38. end
39.
```

```
always @(posedge clk)//0.005s 反转一次
begin
 if(count_100 == 25'd249999)
 begin
  count_100 <= 25'd0;
  clk_100_r <= ~clk_100_r;
 end
 else
  count_100 <= count_100 + 1;
end
//////////////////////////////////////
always @(clk_1_r or clk_10_r or clk_100_r or switch_1 or switch_2)
 begin
  case({switch_1,switch_2})
   2'b00 :clk_timer_r <= clk_1_r;
   2'b01 :clk_timer_r <= clk_10_r;
   2'b11 :clk_timer_r <= clk_100_r;
   default:clk_timer_r <= clk_1_r;
  endcase
 end
endmodule
```

4.3.3 数码管计数显示模块

在数码管计数显示模块中，计数器的计数值转化为时、分、秒的数值，并将时分秒十位和个位分离，以便显示在八段码上。数码管计数显示模块 digital_show 内容如下：

```
module  digital_show(
                    input  clk_timer, // 计数时钟信号输入
                    input  nRST,  // 复位信号输入
       output [3:0]disp_dat_0,
       output [3:0]disp_dat_1,
       output [3:0]disp_dat_2,
       output [3:0]disp_dat_3,
       output [3:0]disp_dat_4,
       output [3:0]disp_dat_5,
        output reg LED
                  );
reg [4:0]hour;      //0-23 小时
reg [5:0]minute;     //0-59 分钟
reg [5:0]second;     //0-59 秒
// 时钟计数
```

```
16. always @(posedge clk_timer or negedge nRST)
17. begin
18.  if(!nRST)    // 时分秒复位
19.   begin
20.    hour <= 5'd0;
21.    minute <= 6'd0;
22.    second <= 6'd0;
23.   end
24.  else
25.   begin
26.    second <= second + 1; // 每个时钟信号到来时，秒+1
27.          // 如果时间到达 59 分 59 秒，点亮 LED，否则熄灭 LED
28.          // 如果秒到达 59，对其清零且分钟+1
29.    if(second == 6'd59) begin second <= 6'd0; minute <= minute+1'b1; end
30.          // 如果分钟到达 59，对其清零且小时+1
31.          // 如果小时到达 24，对其清零
32.    if(second == 6'd59 & minute == 6'd59)
33.    begin
34.    LED = 1'b1;
35.    minute <= 6'd0;
36.    hour <= hour + 1'b1;
37.    end
38.    else LED = 1'b0;
39.         if(hour == 5'd23 &second == 6'd59 & minute == 6'd59) begin hour <=
       5'd0; end
40.   end
41. end
42. assign disp_dat_0 = second % 10; // 秒个位
43. assign disp_dat_1 = second / 10; // 秒十位
44. assign disp_dat_2 = minute % 10; // 分个位
45. assign disp_dat_3 = minute / 10; // 分十位
46. assign disp_dat_4 = hour % 10;  // 时个位
47. assign disp_dat_5 = hour / 10;  // 时十位
48. endmodule
```

4.3.4 数码管驱动模块

将数码管计数显示模块得到的结果分别进行译码，然后显示在八段数码管中，数码管驱动模块 data_show 内容如下：

```
1. module  data_show(
2.         input [3:0]disp_dat,  // 显示数值
3.         output reg [6:0]disp_show
4.                   );
5. always @(disp_dat)
```

```
6. begin
7.     case(disp_dat) // 将显示的数值映射为对应段码
8.   4'h0:disp_show = 7'b100_0000; // 0
9.   4'h1:disp_show = 7'b111_1001; // 1
10.   4'h2:disp_show = 7'b010_0100; // 2
11.   4'h3:disp_show = 7'b011_0000; // 3
12.   4'h4:disp_show = 7'b001_1001; // 4
13.   4'h5:disp_show = 7'b001_0010; // 5
14.   4'h6:disp_show = 7'b000_0010; // 6
15.   4'h7:disp_show = 7'b111_1000; // 7
16.   4'h8:disp_show = 7'b000_0000; // 8
17.   4'h9:disp_show = 7'b001_0000; // 9
18.   default:disp_show = 7'b100_0000;
19.  endcase
20. end
21. endmodule
```

4.3.5 模式选择模块

计数器计数来源由数码管驱动选择模块选择来源于外部按键输入或者时钟分频模块输出，模式选择模块 key_time 如下：

```
1. module  key_time(
2.     input key_time_adj,          //消除抖动后按键输入,按键输入
3.     input key_time_switch,          //消除抖动后按键输入,决定数字前进是时钟还是按键
4.     input key_clk_timer,          //消除抖动后按键输入,时间输入
5.     output key_o     //经过选择后的按键输出
6.     );
7.  assign  key_o=(key_time_switch ? key_time_adj:key_clk_timer);
8. endmodule
```

4.3.6 顶层文件设计

设置五个按键，分别为复位键、模式选择键、档位设置键 1、档位设置键 2 和手动计数键，经过消抖模块后输入后续模块：两个档位设置键用于切换分频倍数，00 代表 1s 计数，10 代表 0.1s 计数，11 代表 0.01s 计数。模式选择键为 0 时，选择内部时钟计数；模式选择键为 1 时，选择外部手动输入计数，按下手动计数键计数。选择时钟工作档位后，分别对时、分和秒进行计算，拆解十位和个位，显示在 6 个八段数码管上。

```
1. module clock(
2.  input wire clk_50MHz,//50MHz 信号输入
3.  input wire key_A,//速度调节键 1
4.  input wire key_B,//速度调节键 2
5.  input wire time_switch,//时钟与按键选择
```

```
6.  input wire time_adj,//按键选择
7.  input wire nRST,//复位信号
8.  output [6:0]HEX0,//八段码
9.  output [6:0]HEX1,
10.  output [6:0]HEX2,
11.  output [6:0]HEX3,
12.  output [6:0]HEX4,
13.  output [6:0]HEX5,
14.  output wire LED//LED 输出
15. );
16. wire key_key_A;
17. wire key_key_B;
18. wire key_time_adj;
19. wire key_time_switch;
20. wire time_set;
21. wire time_devide;
22. wire [3:0]disp_dat_0;//0-9
23. wire [3:0]disp_dat_1;
24. wire [3:0]disp_dat_2;
25. wire [3:0]disp_dat_3;
26. wire [3:0]disp_dat_4;
27. wire [3:0]disp_dat_5;
28. eliminate_jitters u1(
29.  .clk(clk_50MHz),
30.  .key(key_A),
31.  .nRST(nRST),
32.  .key_o(key_key_A)
33. );
34. eliminate_jitters u2(
35.  .clk(clk_50MHz),
36.  .key(key_B),
37.  .nRST(nRST),
38.  .key_o(key_key_B)
39. );
40. eliminate_jitters u3(
41.  .clk(clk_50MHz),
42.  .key(time_adj),
43.  .nRST(nRST),
44.  .key_o(key_time_adj)//+1
45. );
46. eliminate_jitters u4(
47.  .clk(clk_50MHz),
48.  .key(time_switch),
49.  .nRST(nRST),
```

```
50.  .key_o(key_time_switch)//模式选择：选内或外按键消抖
51. );
52. time_devide_timer u5(
53.  .clk(clk_50MHz),          //50MHz 时钟
54.  .switch_1(key_key_A),    // 切换频率
55.  .switch_2(key_key_B),
56.  .clk_timer(time_devide)//分频后时钟
57. );
58. key_time u6(
59.  .key_time_adj(key_time_adj),          //消除抖动后按键输入,按键输入
60.  .key_time_switch(key_time_switch),          //消除抖动后按键输入,决定
     数字前进是时钟还是按键
61.  .key_clk_timer(time_devide),          //消除抖动后按键输入,时间输入
62.  .key_o(time_set)//选内或外
63. );
64. digital_show u7(
65.  .clk_timer(time_set), // 计数时钟信号输入
66.    .nRST(nRST),  // 复位信号输入
67.  .disp_dat_0(disp_dat_0), // 显示数值
68.  .disp_dat_1(disp_dat_1),
69.  .disp_dat_2(disp_dat_2),
70.  .disp_dat_3(disp_dat_3),
71.  .disp_dat_4(disp_dat_4),
72.  .disp_dat_5(disp_dat_5),
73.  .LED(LED)//计算数字
74. );
75. data_show u8(
76.  .disp_dat(disp_dat_0), // 显示数值
77.  .disp_show(HEX0)
78. );
79. data_show u9(
80.  .disp_dat(disp_dat_1), // 显示数值
81.  .disp_show(HEX1)
82. );
83. data_show u10(
84.  .disp_dat(disp_dat_2), // 显示数值
85.  .disp_show(HEX2)
86. );
87. data_show u11(
88.  .disp_dat(disp_dat_3), // 显示数值
89.  .disp_show(HEX3)
90. );
91. data_show u12(
92.  .disp_dat(disp_dat_4), // 显示数值
```

```
93.  .disp_show(HEX4)
94. );
95. data_show u13(
96.  .disp_dat(disp_dat_5), // 显示数值
97.  .disp_show(HEX5)
98. );
99. endmodule
```

4.4 软件仿真

初始时钟值为 0，按键值每 10ps 翻转，模拟内部时钟脉冲情况。 编写 Test Bench 文件代码如下：

```
1. `timescale 1 ns/ 1 ns
2. module clock_vlg_tst();
3. reg eachvec;
4. reg clk_50MHz;
5. reg key_A;
6. reg key_B;
7. reg nRST;
8. reg time_adj;
9. reg time_switch;
10. wire LED;
11. wire [6:0]  disp_show_0;
12. wire [6:0]  disp_show_1;
13. wire [6:0]  disp_show_2;
14. wire [6:0]  disp_show_3;
15. wire [6:0]  disp_show_4;
16. wire [6:0]  disp_show_5;
17. wire [6:0]  disp_show_6;
18. clock i1 (
19.  .LED(LED),
20.  .clk_50MHz(clk_50MHz),
21.  .disp_show_0(disp_show_0),
22.  .disp_show_1(disp_show_1),
23.  .disp_show_2(disp_show_2),
24.  .disp_show_3(disp_show_3),
25.  .disp_show_4(disp_show_4),
26.  .disp_show_5(disp_show_5),
27.  .disp_show_6(disp_show_6),
28.  .key_A(key_A),
29.  .key_B(key_B),
30.  .nRST(nRST),
31.  .time_adj(time_adj),
```

```
32.  .time_switch(time_switch)
33. );
34. initial
35. begin
36.    clk_50MHz = 0;//50MHz 信号输入
37.  key_A = 0;//速度调节键 1
38.  key_B = 0;//速度调节键 2
39.  time_switch = 0;//时钟与按键选择
40.  time_adj = 0;//按键选择
41.  nRST = 1;//复位信号
42. $display("Running testbench");
43. end
44. always
45. initial
46. begin
47. forever #10 clk_50MHz=~clk_50MHz;
48. end
49. endmodule
```

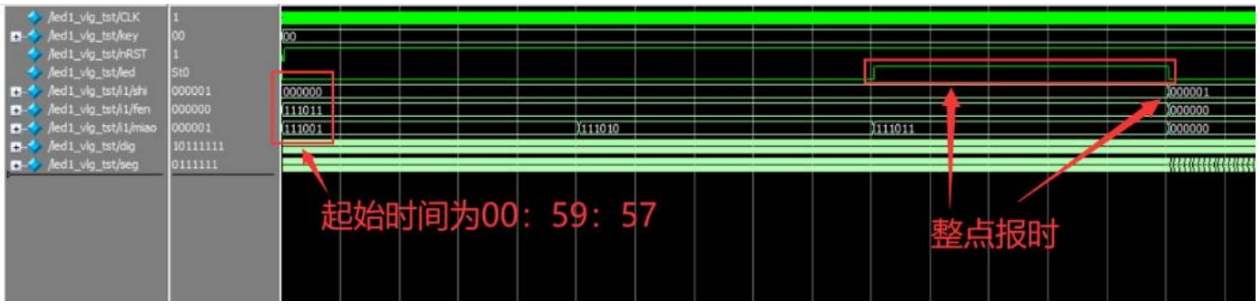
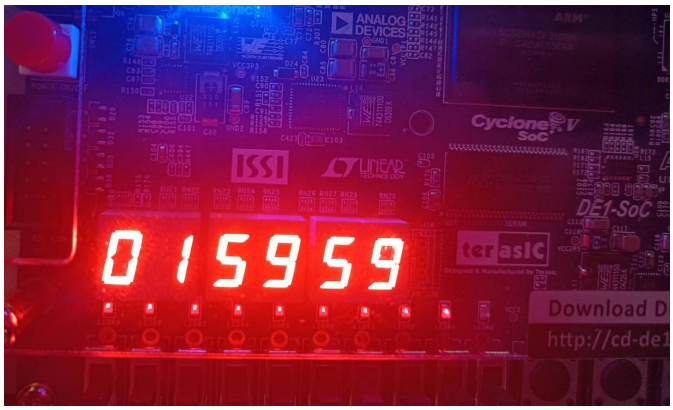
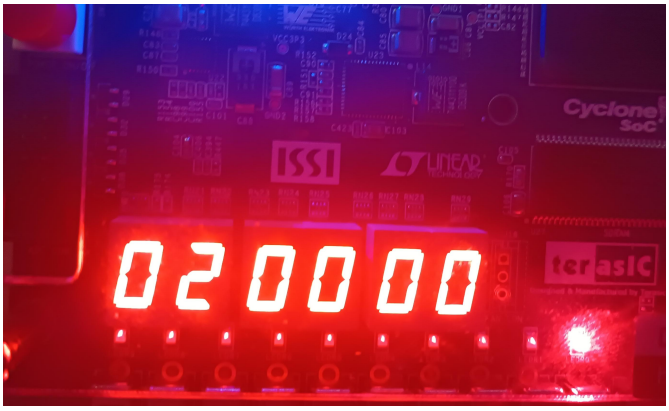


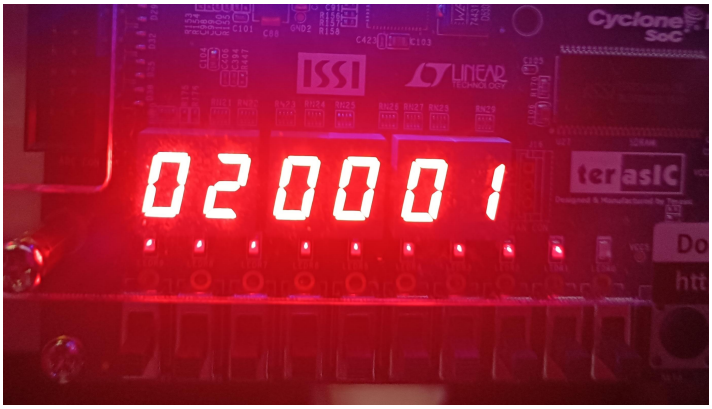
图 4- 1 变速数字时钟 ModelSim 仿真结果

4.5 实验结果

在 59 分 59 秒时，进行报时，LED 指示灯亮，持续至 00 分 00 秒，然后 LED 指示灯熄灭。以 01 时 59 分 59 秒为例，实验结果如下：







总结

经过完成大规模可编程器件的四个实验，我对于 FPGA 有了初步的认识，对于 FPGA 和嵌入式芯片如 STM32、单片机如 C51 的区别有了深刻的了解，掌握了 Quartus 和 ModelSim 的联合设计仿真，为以后学习和应用 FPGA 打下了基础。

在完成程序的过程中，我遇到了很多问题，如编译时遇到报错、ModelSim 仿真无结果或者波形出错等问题，但是都通过上网搜索和请教同学、老师完成了，在此感谢王老师的悉心指导。

与单片机和嵌入式芯片相比，FPGA 编程更加偏向底层硬件开发，对应端口的功能并不固定，需要我们自行设计，更适合设计者发挥自己的创造性。如果能用好 Verilog 编程方法，合理安排设计模块，会获得意想不到的效果，开发出更完美的作品。