

**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 2](#_Toc20412)

[1.1 实验目的 2](#_Toc6924)

[1.2 实验内容 2](#_Toc21330)

[1.3 程序设计 2](#_Toc30026)

[1.4 软件仿真 2](#_Toc17760)

[1.5 实验结果 3](#_Toc20487)

[实验二 跑马灯 4](#_Toc12717)

[2.1 实验目的 4](#_Toc25401)

[2.2 实验内容 4](#_Toc15178)

[2.3 程序设计 4](#_Toc23656)

[2.4 软件仿真 5](#_Toc16726)

[2.5 实验结果 6](#_Toc2237)

[实验三 序列检测器 7](#_Toc5556)

[3.1 实验目的 7](#_Toc22826)

[3.2 实验内容 7](#_Toc3561)

[3.3 程序设计 7](#_Toc30646)

[3.4 软件仿真 11](#_Toc11651)

[3.5 实验结果 12](#_Toc25126)

[实验四 变速数字时钟 14](#_Toc10615)

[4.1 实验目的 14](#_Toc23145)

[4.2 实验内容 14](#_Toc32530)

[4.3 程序设计 14](#_Toc17926)

[4.3.1 按键消抖模块 14](#_Toc12)

[4.3.2 时钟分频模块 16](#_Toc17370)

[4.3.3 数码管计数显示模块 17](#_Toc17923)

[4.3.4 数码管驱动模块 18](#_Toc26928)

[4.3.5模式选择模块 19](#_Toc10222)

[4.3.6 顶层文件设计 19](#_Toc2137)

[4.4 软件仿真 22](#_Toc21062)

[4.5 实验结果 23](#_Toc17327)

[总结 24](#_Toc8633)

# 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

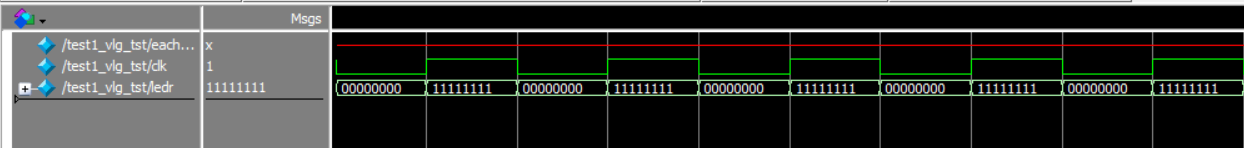
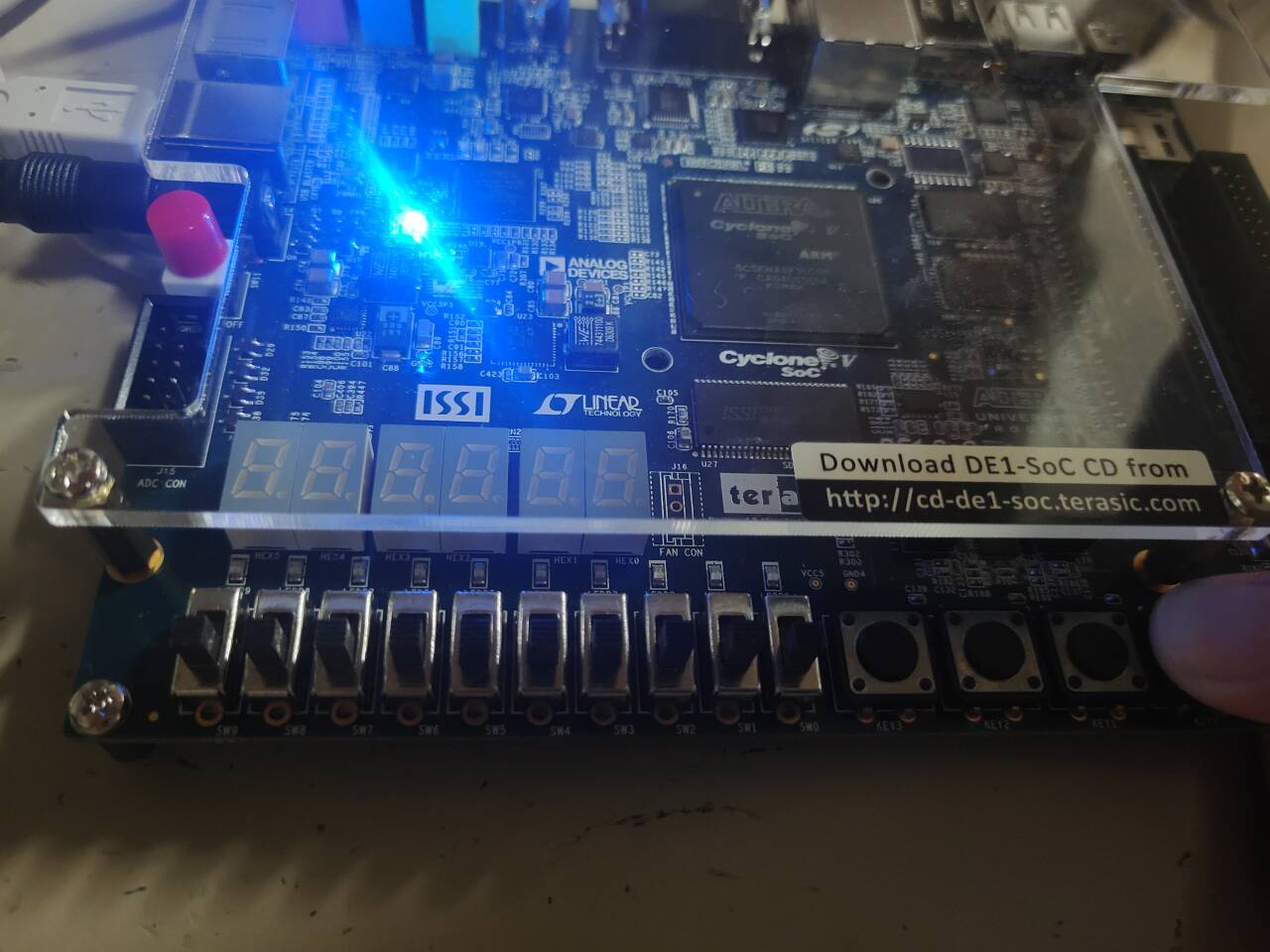
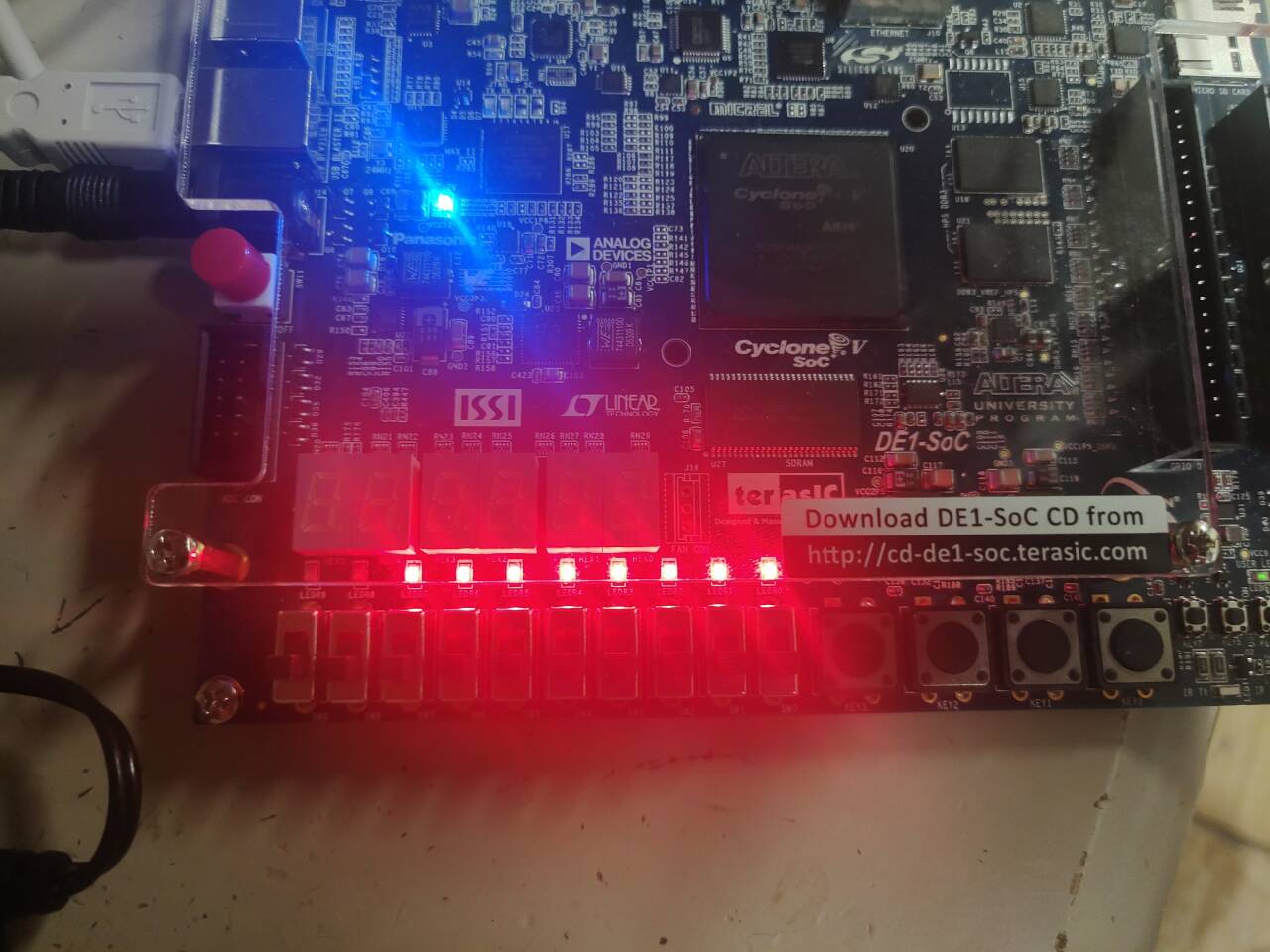
运行ModelSim，仿真结果如下图所示：

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：

|  |  |
| --- | --- |
| D`D{DQY]WS2@$2){A7LCWPI_tmb | [5$YWWMPFV%NFIF~DG$~%AI_tmb |

|  |  |
| --- | --- |
| Q19EJK6D7P[_G3AZ%[%DT~I_tmb | 3ON`G865I0PKDP`16A3H)EI_tmb |

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。

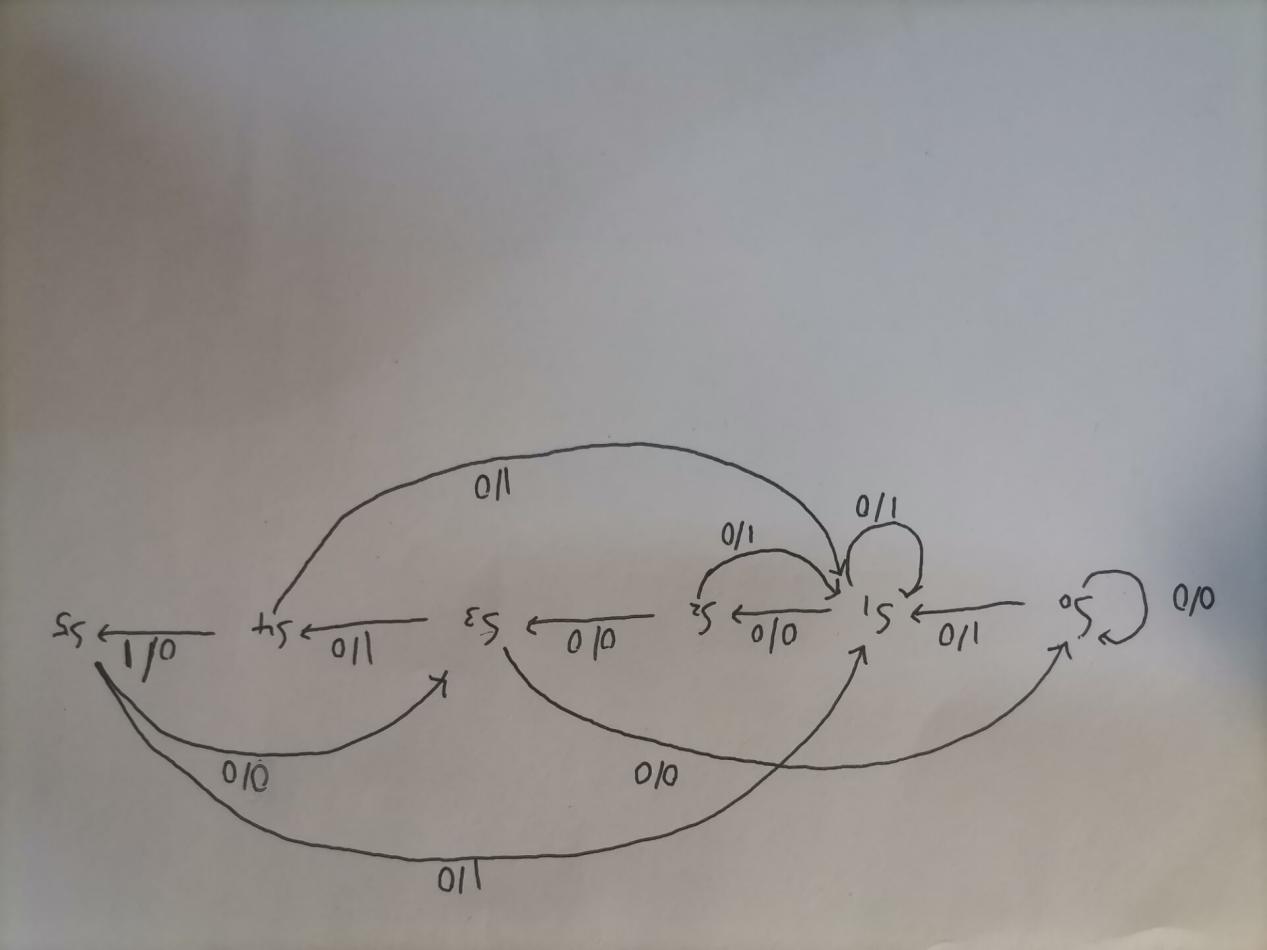
利用两个按键，分别输入 0 或者 1，实现序列输入。实验中利用 LED 来替代 Z，一旦检测到“10010”，LED 亮，否则 LED 灭

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,
5. output clk\_5KHz,
6. output reg key\_o *//经过消抖后的按键输出*
7. );
8. reg [20:0]cnt;*//定义一个21位宽的计数器*
9. reg  key\_d1;*//定义一个位宽为1位的寄存器*
10. reg  key\_d2;*//定义一个位宽为1位的寄存器*
11. *//模块一：时钟分频模块*
12. always @(posedge clk or negedge nRST)
13. begin
14. if(!nRST)
15. cnt <= 21'd0;*//复位*
16. else
17. begin
18. if(cnt<21'd9999)
19. cnt <= cnt + 1'b1;
20. else
21. cnt <= 21'd0;
22. end
23. end
24. assign  clk\_5KHz = (cnt<21'd5000)? 1'b1:1'b0;
25. *//模块二：移位模块*
26. always @(posedge clk\_5KHz or negedge nRST)
27. begin
28. if(!nRST)
29. begin
30. key\_d1 <= 0;
31. key\_d2 <= 0;
32. end
33. else
34. begin
35. key\_d1 <= key;*//D触发器*
36. key\_d2 <= key\_d1;*//D触发器*
37. end
38. end
39. *//模块三：按键消抖模块*
40. always @(posedge clk\_5KHz or negedge nRST)
41. begin
42. if(!nRST)
43. key\_o <= 1'b0;
44. else
45. begin
46. if(key\_d1 & key\_d2 & key)
47. key\_o <= 1'b1;
48. 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'b01001,S5=5'b10010;
15. always @(posedge clk or negedge rst)
16. begin
17. if(!rst)
18. begin
19. flag<=0;//指示信号复位
20. y<=S0;//状态变量复位
21. end
22. else
24. case(y)
25. S0:
26. begin
27. flag<=1'b0;
28. if (high\_d&!high) y <= S1;
29. else y <= y;
30. end
31. S1:
32. begin
33. flag<=1'b0;
34. if (high\_d&!high) y <= S1;
35. else if(low\_d&!low) y<=S2;
36. else y <= y;
37. end
38. S2:
39. begin
40. flag<=1'b0;
41. if (high\_d&!high) y <= S1;
42. else if(low\_d&!low) y<=S3;
43. else y <= y;
44. end
45. S3:
46. begin
47. flag<=1'b0;
48. if (high\_d&!high) y <= S4;
49. else if(low\_d&!low) y<=S0;
50. else y <= y;
51. end
52. S4:
53. begin
54. flag<=1'b0;
55. if (high\_d&!high) y <= S1;
56. else if(low\_d&!low) y<=S5;
57. else y <= y;
58. end
59. S5:
60. begin
61. flag<=1'b1;
62. if (high\_d&!high) y <= S1;
63. else if(low\_d&!low) y<=S3;
64. else y <= y;
65. end
66. default: y <= S0;
67. endcase
68. end
69. 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;
10. wire key1;
11. wire key2;
12. key\_handle u1(
13. .clk(clk\_50MHz),
14. .key(key\_A),
15. .nRST(nRST),
16. .clk\_5KHz(clk\_5KHz),
17. .key\_o(key1));
18. key\_handle u2(
19. .clk(clk\_50MHz),
20. .key(key\_B),
21. .nRST(nRST),
22. .key\_o(key2));
23. check u3(
24. .clk(clk\_5KHz),
25. .rst(nRST),
26. .high(key1),
27. .low(key2),
28. .y(state),
29. .flag(LED));
30. 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。

1. `timescale 1 ps/ 1 ps
2. module test3\_vlg\_tst();
3. reg clk;
4. reg high;
5. reg low;
6. reg rst;
7. wire flag;
8. wire [4:0] y;
9. check i1 (
10. .clk(clk),
11. .flag(flag),
12. .high(high),
13. .low(low),
14. .rst(rst),
15. .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：

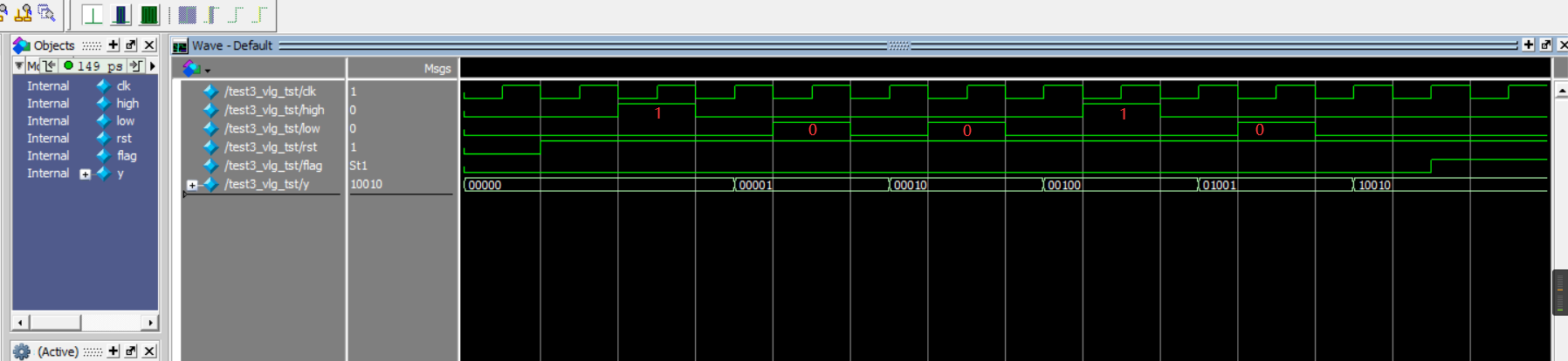
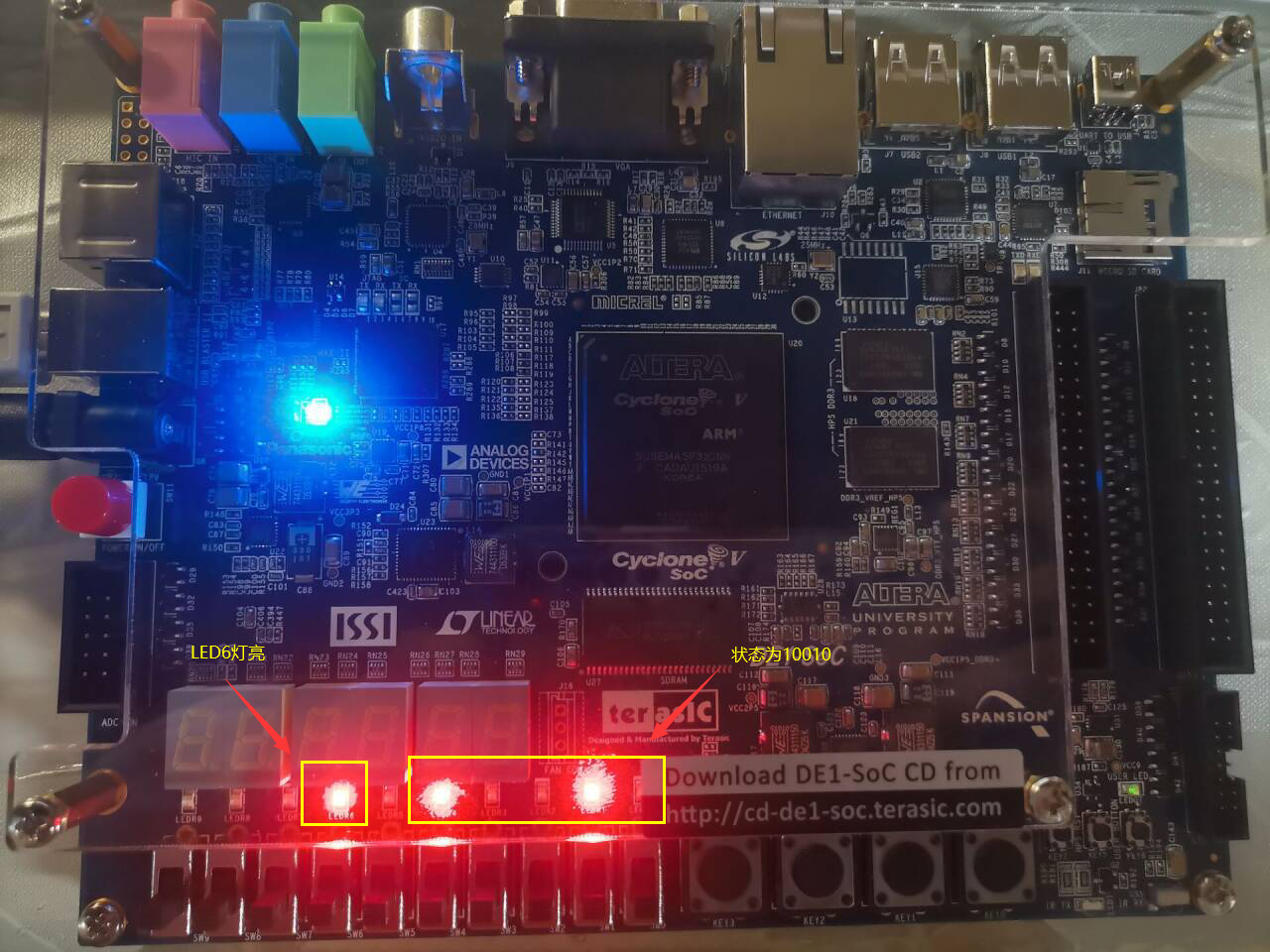


图3- 2 序列检测器ModelSim仿真波形图

## 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)，持续时问为 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. reg [20:0]cnt;         *//定义一个21位宽的计数器*
9. reg  key\_d1;         *//定义一个位宽为1位的寄存器*
10. reg  key\_d2;         *//定义一个位宽为1位的寄存器*
11. *//模块一：时钟分频模块*
12. always @(posedge clk or negedge nRST)
13. begin
14. if(!nRST)
15. cnt <= 21'd0;*//复位*
16. else
17. begin
18. if(cnt<21'd9999)
19. cnt <= cnt + 1'b1;       *//计数器在小于9999时，每来一个时钟，就加1*
20. else
21. cnt <= 21'd0;       *//当计数器等于9999时，清零*
22. end
23. end
24. assign  clk\_5KHz = (cnt<21'd5000)? 1'b1:1'b0;   *//得到一个5KHz（50MHz/10000）的分频时钟*
25. *//模块二：移位模块*
26. always @(posedge clk\_5KHz or negedge nRST)
27. begin
28. if(!nRST)
29. begin
30. key\_d1 <= 0;
31. key\_d2 <= 0;
32. end
33. else
34. begin
35. key\_d1 <= key;                   *//D触发器*
36. key\_d2 <= key\_d1;               *//D触发器*
37. end
38. end
39. *//模块三：按键消抖模块*
40. always @(posedge clk\_5KHz or negedge nRST)
41. begin
42. if(!nRST)
43. key\_o <= 1'b0;
44. else
45. begin
46. if(key\_d1 & key\_d2 & key)
47. key\_o <= 1'b1;
48. else if(!key\_d1 & !key\_d2 & !key)
49. key\_o <= 1'b0;
50. end
51. end
52. 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. reg [24:0]count\_1 = 25'd0;//计数器
8. reg [24:0]count\_10 = 25'd0;
9. reg [24:0]count\_100 = 25'd0;
10. reg clk\_1\_r = 0;//控制电平
11. reg clk\_10\_r = 0;
12. reg clk\_100\_r = 0;
13. reg clk\_timer\_r =0;
14. assign clk\_timer = clk\_timer\_r;
15. ////////////////////////////////////////
16. always @(posedge clk)//0.5s反转一次
17. begin
18. if(count\_1 == 25'd24999999)
19. begin
20. count\_1 <= 25'd0;
21. clk\_1\_r <= ~clk\_1\_r;
22. end
23. else
24. count\_1 <= count\_1 + 1;
25. end
26. always @(posedge clk)//0.05s反转一次
27. begin
28. if(count\_10 == 25'd2499999)
29. begin
30. count\_10 <= 25'd0;
31. clk\_10\_r <= ~clk\_10\_r;
32. end
33. else
34. count\_10 <= count\_10 + 1;
35. end
36. always @(posedge clk)//0.005s反转一次
37. begin
38. if(count\_100 == 25'd249999)
39. begin
40. count\_100 <= 25'd0;
41. clk\_100\_r <= ~clk\_100\_r;
42. end
43. else
44. count\_100 <= count\_100 + 1;
45. end
46. ////////////////////////////////////////
47. always @(clk\_1\_r or clk\_10\_r or clk\_100\_r or switch\_1 or switch\_2)
48. begin
49. case({switch\_1,switch\_2})
50. 2'b00 :clk\_timer\_r <= clk\_1\_r;
51. 2'b01 :clk\_timer\_r <= clk\_10\_r;
52. 2'b11 :clk\_timer\_r <= clk\_100\_r;
53. default:clk\_timer\_r <= clk\_1\_r;
54. endcase
55. end
56. 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：

1. module  digital\_show(
2. input  clk\_timer,  *// 计数时钟信号输入*
3. input  nRST,   *// 复位信号输入*
4. output [3:0]disp\_dat\_0,
5. output [3:0]disp\_dat\_1,
6. output [3:0]disp\_dat\_2,
7. output [3:0]disp\_dat\_3,
8. output [3:0]disp\_dat\_4,
9. output [3:0]disp\_dat\_5,
10. output reg LED
11. );
12. reg [4:0]hour;       *//0-23小时*
13. reg [5:0]minute;      *//0-59分钟*
14. reg [5:0]second;      *//0-59秒*
15. *// 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：

|  |
| --- |
| I6~@@$%3GOY)H8J%0UON]$C_tmb |

|  |
| --- |
| %]BCOD%SM$DLFSL(IAO4CN7 |
| 9P5~DJACBS`OK%(@XL]JJ86 |

# 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。