

实验2

实验2

流水灯

代码

波形分析

节日彩灯

代码

波形分析

Chisel

节日彩灯（流水灯）

逻辑部分

测试部分 & 生成部分

生成的Verilog代码

波形数据

本实验有 verilog 和 chisel 两个版本的代码，两边都可以正确运行。

流水灯

代码

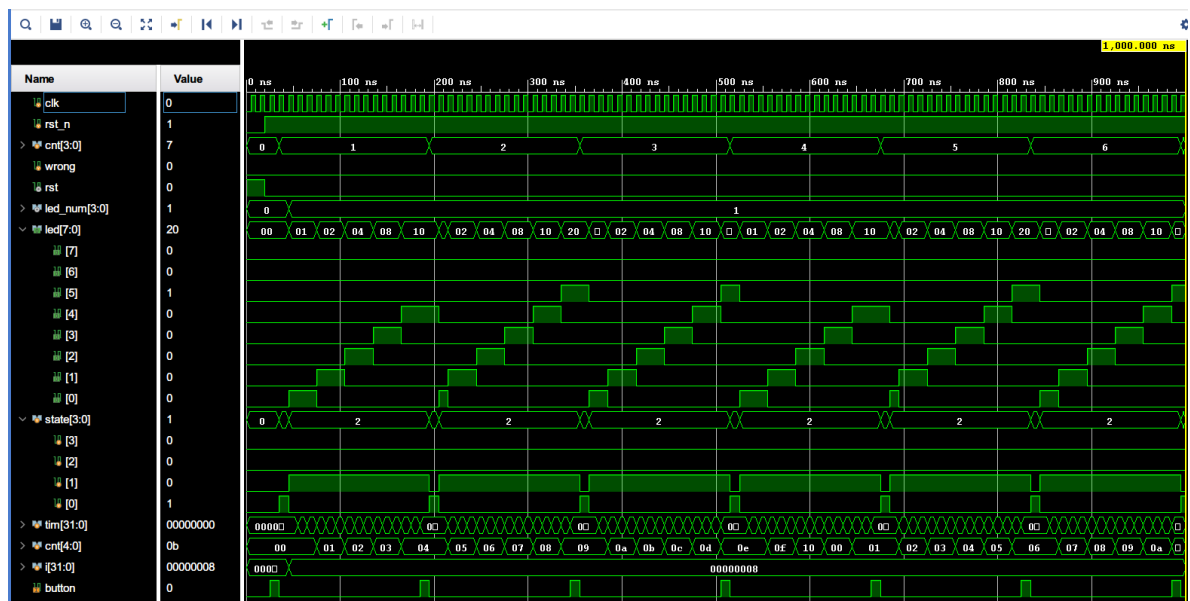
```
1 // Author: Chiro, Date: 2021/11/10
2 module flowing_water_lights (input wire clk,
3                               input wire rst,
4                               input wire button,
5                               output reg [7:0] led);
6     // parameter delay      = 32'd2;
7     // 分频数量
8     parameter delay        = 32'd100000000;
9     parameter STATE_PRE    = 0;
10    parameter STATE_INIT   = 1;
11    parameter STATE_RUN    = 2;
12    reg [3:0] state;
13    reg [31:0] tim;
14
15    always @ (posedge clk or posedge rst) begin
16        if (rst) begin
17            led    <= 8'b0;
18            tim    <= 32'b0;
19            // 初始化当前状态机为开始之前
20            state <= STATE_PRE;
21        end
22        else begin
23            if (state == STATE_PRE) begin
24                // 开始之前状态，等待按键开始，就进入初始化状态
25                if (button)
26                    state <= STATE_INIT;
27            end
28            else if (state == STATE_INIT) begin
29                // 初始化状态，设置相关信息之后进入运行状态
```

```

30         led <= 8'b0000_0001;
31         state <= STATE_RUN;
32     end
33     else if (state == STATE_RUN) begin
34         // 运行状态，如果有按键就重新初始化
35         if (button)
36             state <= STATE_INIT;
37         else begin
38             // 计数器溢出，更新
39             if (tim == delay) begin
40                 tim <= 32'b0;
41                 // 进行一个位的移
42                 led <= {led[6 -: 7], led[7]};
43             end else
44                 // 计数
45                 tim <= tim + 32'b1;
46         end
47     end
48 end
49 endmodule
50
51

```

波形分析



`flow_waterLights` 模块完成的是流水灯的功能，在根据时间进行灯的流水变化的同时还要考虑按键复位的功能。

1. 当 `rst` 拉高，系统复位
2. 当 `rst` 拉低，系统开始运行
3. 每隔2个时钟周期，`tim` 自增1；当 `tim` 达到 `delay` 表示 `tim` 计时完成，需要更新 `cnt` 再复位 `tim`
4. `cnt` 更新同时，`led` 位移一位
5. 同时，当 `button` 按下，位移复位，重新从第一位开始流水，`state` 变化为： `STATE_RUN -> STATE_INIT -> STATE_RUN`

节日彩灯

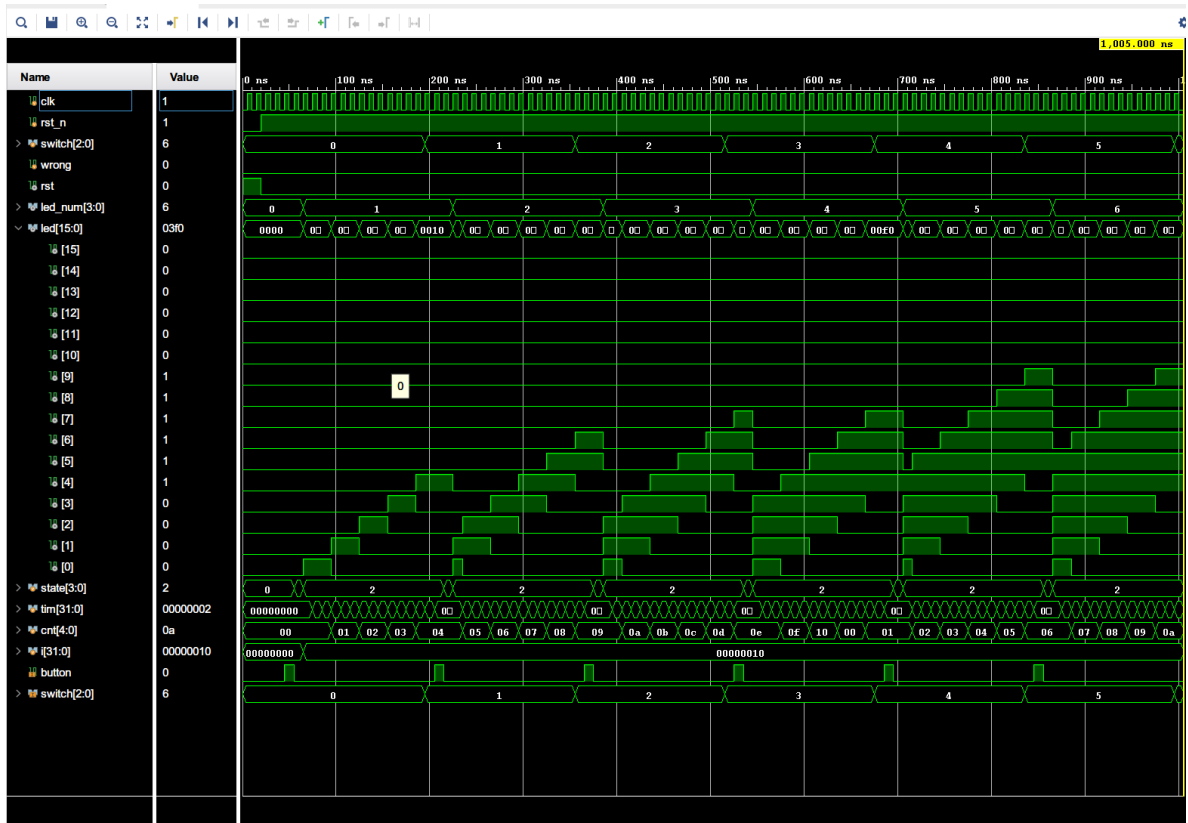
代码

```

1 // Author: Chiro, Date: 2021/11/10
2 module holidayLights (input wire clk,
3                       input wire rst,
4                       input wire button,
5                       input wire [2:0] switch,
6                       output reg [15:0] led);
7
8 // 分频数量
9 // parameter delay      = 32'd2;
10 parameter delay        = 32'd100000000;
11 parameter STATE_PRE    = 0;
12 parameter STATE_INIT   = 1;
13 parameter STATE_RUN    = 2;
14 reg [3:0] state;
15 reg [31:0] tim;
16 integer i;
17
18 always @ (posedge clk or posedge rst) begin
19     if (rst) begin
20         led    <= 16'b0;
21         tim    <= 32'b0;
22         i      <= 0;
23         // 初始化当前状态机为开始之前
24         state <= STATE_PRE;
25     end
26     else begin
27         if (state == STATE_PRE) begin
28             // 开始之前状态，等待按键开始，就进入初始化状态
29             if (button)
30                 state <= STATE_INIT;
31         end
32         else if (state == STATE_INIT) begin
33             // 初始化状态，设置相关信息之后进入运行状态
34             for (i = 0; i < 16; i = i + 1)
35                 led[i] = (i < switch + 1) ? 1'b1 : 1'b0;
36             state <= STATE_RUN;
37         end
38         else if (state == STATE_RUN) begin
39             // 运行状态，如果有按键就重新初始化
40             if (button)
41                 state <= STATE_INIT;
42             else begin
43                 // 计数器溢出，更新
44                 if (tim == delay) begin
45                     tim <= 32'b0;
46                     // 进行一个位的移
47                     led <= {led[14 -: 15], led[15]};
48                 end else
49                     // 计数
50                     tim <= tim + 32'b1;
51             end
52         end
53     end
54 endmodule

```

波形分析



holiday_lights 模块完成的是流水灯的功能，在根据时间进行灯的流水变化的同时还要考虑按键复位的功能。

1. 当 rst 拉高，系统复位
2. 当 rst 拉低，系统开始运行，state = STATE_PRE，等待 button 按下
3. 每隔2个时钟周期，tim 自增1；当 tim 达到 delay 表示 tim 计时完成，需要更新 cnt 再复位 tim
4. cnt 更新同时，led 位移一位
5. 同时，当 button 按下，位移复位，重新从第一位开始流水，state 变化为：STATE_RUN -> STATE_INIT -> STATE_RUN

Chisel

节日彩灯（流水灯）

当 ledCount=0 的时候其实就是流水灯啦。

逻辑部分

```

1 package test
2 import chisel3._
3 import chisel3.util._
4 import chisel3.experimental._
5
6 @chiselName
7 class FlowLights(ledWidth: Int = 8, delay: Int = 100000000, useSwitch:
8 Boolean = false) extends Module {
9     val io = IO(new Bundle {
10         val button = Input(UInt(1.W))
11         val ledCount = if (useSwitch) Some(Input(UInt(3.W))) else None
12         val led = Output(UInt(ledWidth.W))
13     })
14 }

```

```

14     val ledReg = RegInit(1.U(ledwidth.W))
15     val clkReg = RegInit(1.U(32.W))
16
17     def flow(old: UInt) = {
18         val bools = old.asBools
19         val newUInt = Cat((bools(ledwidth - 1) +: bools.slice(0, ledwidth -
20         1)).reverse).asUInt
21         newUInt
22     }
23
24     def generate(count: UInt) = {
25         val res = (for { a <- 0 until ledwidth } yield Mux(a.U < count, 1.U,
26         0.U) << a).reduce(_ | _)
27         // printf(p"res = ${Binary(res)}\n")
28         res
29     }
30
31     when(io.button === 1.U) {
32         ledReg := generate(if (useSwitch) (io.ledCount.get + 1.U) else 1.U)
33         // printf(p"generated: ${Binary(ledReg)}\n")
34         clkReg := delay.U(32.W)
35     } .otherwise {
36         ledReg := Mux(clkReg === 0.U, flow(ledReg), ledReg)
37         clkReg := Mux(clkReg === delay.U, 0.U, clkReg + 1.U)
38     }
39     io.led := ledReg
40 }
41
42 @chiselName
43 class FlowLightswrapper(ledwidth: Int = 16, delay: Int = 2, useSwitch:
44 Boolean = true) extends Module {
45     val io = IO(new Bundle {
46         val button = Input(UInt(1.W))
47         val ledCount = if (useSwitch) Some(Input(UInt(3.W))) else None
48         val led = Output(UInt(ledwidth.W))
49     })
50
51     val flowLights = Module(new FlowLights(delay=delay, ledwidth=ledwidth,
52     useSwitch=useSwitch))
53
54     flowLights.io.button := io.button
55     io.led := flowLights.io.led
56     if (useSwitch)
57         flowLights.io.ledCount.get := io.ledCount.get
58 }

```

测试部分 & 生成部分

```

1 package test
2 import org.scalatest._
3 import chiseltest._
4 import chisel3._
5
6 class FlowLightsTest extends FlatSpec with ChiselScalatestTester with
7 Matchers {
8     behavior of "FlowLightswrapper"
9     it should "pass the test" in {

```

```

9      test(new FlowLightsWrapper(ledwidth=16, delay=4, useSwitch=true)) { c =>
10          println("Starting test FlowLights...")
11          val ledCount = 0
12          c.io.button.poke(1.U)
13          c.io.ledCount.get.poke(ledCount.U)
14          c.clock.step(2)
15          c.io.led.expect((ledCount + 1).U)
16          // printf(s"c.io.led: ${c.io.led.peek()}\n")
17
18          c.io.button.poke(0.U)
19          c.clock.step()
20          c.io.led.expect((ledCount + 1).U)
21          c.clock.step()
22          c.clock.step()
23          c.clock.step()
24          c.clock.step()
25          c.io.led.expect(((ledCount + 1) << 1).U)
26          // c.io.led.expect(ledCount.U << 1.U)
27          // c.io.led.expect(((1 << ledCount) - 1).U)
28          println("test done.")
29      }
30  }
31  }

```

```

1  package test
2  import chisel3.stage.{ChiselStage, ChiselGeneratorAnnotation}
3
4  object testFlowLights extends App {
5      (new chisel3.stage.ChiselStage).execute(args,
6      Seq(ChiselGeneratorAnnotation(() => new FlowLightsWrapper(
7          delay=2, ledwidth=16, useSwitch=true
8      ))))
9  }

```

生成的Verilog代码

经过外部再添加一个wrapper才能用本实验的testbench。

```

1  // Author: Chiro, Date: 2021/11/10
2
3  module FlowLights(
4      input          clock,
5      input          reset,
6      input          io_button,
7      input  [2:0]   io_ledCount,
8      output [15:0]  io_led
9  );
10  `ifdef RANDOMIZE_REG_INIT
11      reg [31:0] _RAND_0;
12      reg [31:0] _RAND_1;
13  `endif // RANDOMIZE_REG_INIT
14      reg [15:0] ledReg; // @[flowLights.scala 15:23]
15      reg [31:0] clkReg; // @[flowLights.scala 16:23]
16      wire [2:0] _ledReg_T_1 = io_ledCount + 3'h1; // @[flowLights.scala
17      31:56]

```

```

17   wire [1:0] _ledReg_res_T_5 = {3'h1 < _ledReg_T_1, 1'h0}; //
    @[flowLights.scala 25:79]
18   wire [2:0] _ledReg_res_T_8 = {3'h2 < _ledReg_T_1, 2'h0}; //
    @[flowLights.scala 25:79]
19   wire [3:0] _ledReg_res_T_11 = {3'h3 < _ledReg_T_1, 3'h0}; //
    @[flowLights.scala 25:79]
20   wire [4:0] _ledReg_res_T_14 = {3'h4 < _ledReg_T_1, 4'h0}; //
    @[flowLights.scala 25:79]
21   wire [5:0] _ledReg_res_T_17 = {3'h5 < _ledReg_T_1, 5'h0}; //
    @[flowLights.scala 25:79]
22   wire [6:0] _ledReg_res_T_20 = {3'h6 < _ledReg_T_1, 6'h0}; //
    @[flowLights.scala 25:79]
23   wire [1:0] _GEN_2 = {{1'd0}, 3'h0 < _ledReg_T_1}; // @[flowLights.scala
    25:94]
24   wire [1:0] _ledReg_res_T_48 = _GEN_2 | _ledReg_res_T_5; //
    @[flowLights.scala 25:94]
25   wire [2:0] _GEN_3 = {{1'd0}, _ledReg_res_T_48}; // @[flowLights.scala
    25:94]
26   wire [2:0] _ledReg_res_T_49 = _GEN_3 | _ledReg_res_T_8; //
    @[flowLights.scala 25:94]
27   wire [3:0] _GEN_4 = {{1'd0}, _ledReg_res_T_49}; // @[flowLights.scala
    25:94]
28   wire [3:0] _ledReg_res_T_50 = _GEN_4 | _ledReg_res_T_11; //
    @[flowLights.scala 25:94]
29   wire [4:0] _GEN_5 = {{1'd0}, _ledReg_res_T_50}; // @[flowLights.scala
    25:94]
30   wire [4:0] _ledReg_res_T_51 = _GEN_5 | _ledReg_res_T_14; //
    @[flowLights.scala 25:94]
31   wire [5:0] _GEN_6 = {{1'd0}, _ledReg_res_T_51}; // @[flowLights.scala
    25:94]
32   wire [5:0] _ledReg_res_T_52 = _GEN_6 | _ledReg_res_T_17; //
    @[flowLights.scala 25:94]
33   wire [6:0] _GEN_7 = {{1'd0}, _ledReg_res_T_52}; // @[flowLights.scala
    25:94]
34   wire [6:0] _ledReg_res_T_53 = _GEN_7 | _ledReg_res_T_20; //
    @[flowLights.scala 25:94]
35   wire [7:0] _ledReg_res_T_54 = {{1'd0}, _ledReg_res_T_53}; //
    @[flowLights.scala 25:94]
36   wire [8:0] _ledReg_res_T_55 = {{1'd0}, _ledReg_res_T_54}; //
    @[flowLights.scala 25:94]
37   wire [9:0] _ledReg_res_T_56 = {{1'd0}, _ledReg_res_T_55}; //
    @[flowLights.scala 25:94]
38   wire [10:0] _ledReg_res_T_57 = {{1'd0}, _ledReg_res_T_56}; //
    @[flowLights.scala 25:94]
39   wire [11:0] _ledReg_res_T_58 = {{1'd0}, _ledReg_res_T_57}; //
    @[flowLights.scala 25:94]
40   wire [12:0] _ledReg_res_T_59 = {{1'd0}, _ledReg_res_T_58}; //
    @[flowLights.scala 25:94]
41   wire [13:0] _ledReg_res_T_60 = {{1'd0}, _ledReg_res_T_59}; //
    @[flowLights.scala 25:94]
42   wire [14:0] _ledReg_res_T_61 = {{1'd0}, _ledReg_res_T_60}; //
    @[flowLights.scala 25:94]
43   wire [15:0] ledReg_res = {{1'd0}, _ledReg_res_T_61}; //
    @[flowLights.scala 25:94]
44   wire _ledReg_T_3 = ~reset; // @[flowLights.scala 26:11]
45   wire ledReg_newUInt_lo_lo_lo_hi = ledReg[0]; // @[flowLights.scala
    19:21]

```

```

46   wire ledReg_newUInt_lo_lo_hi_lo = ledReg[1]; // @[flowLights.scala
19:21]
47   wire ledReg_newUInt_lo_lo_hi_hi = ledReg[2]; // @[flowLights.scala
19:21]
48   wire ledReg_newUInt_lo_hi_lo_lo = ledReg[3]; // @[flowLights.scala
19:21]
49   wire ledReg_newUInt_lo_hi_lo_hi = ledReg[4]; // @[flowLights.scala
19:21]
50   wire ledReg_newUInt_lo_hi_hi_lo = ledReg[5]; // @[flowLights.scala
19:21]
51   wire ledReg_newUInt_lo_hi_hi_hi = ledReg[6]; // @[flowLights.scala
19:21]
52   wire ledReg_newUInt_hi_lo_lo_lo = ledReg[7]; // @[flowLights.scala
19:21]
53   wire ledReg_newUInt_hi_lo_lo_hi = ledReg[8]; // @[flowLights.scala
19:21]
54   wire ledReg_newUInt_hi_lo_hi_lo = ledReg[9]; // @[flowLights.scala
19:21]
55   wire ledReg_newUInt_hi_lo_hi_hi = ledReg[10]; // @[flowLights.scala
19:21]
56   wire ledReg_newUInt_hi_hi_lo_lo = ledReg[11]; // @[flowLights.scala
19:21]
57   wire ledReg_newUInt_hi_hi_lo_hi = ledReg[12]; // @[flowLights.scala
19:21]
58   wire ledReg_newUInt_hi_hi_hi_lo = ledReg[13]; // @[flowLights.scala
19:21]
59   wire ledReg_newUInt_hi_hi_hi_hi = ledReg[14]; // @[flowLights.scala
19:21]
60   wire ledReg_newUInt_lo_lo_lo_lo = ledReg[15]; // @[flowLights.scala
19:21]
61   wire [7:0] ledReg_newUInt_lo =
{ledReg_newUInt_lo_hi_hi_hi,ledReg_newUInt_lo_hi_hi_lo,ledReg_newUInt_lo_h
i_lo_hi,
62
    ledReg_newUInt_lo_hi_lo_lo,ledReg_newUInt_lo_lo_hi_hi,ledReg_newUInt_lo_l
o_hi_lo,ledReg_newUInt_lo_lo_lo_hi,
63     ledReg_newUInt_lo_lo_lo_lo}; // @[Cat.scala 30:58]
64   wire [15:0] ledReg_newUInt =
{ledReg_newUInt_hi_hi_hi_hi,ledReg_newUInt_hi_hi_hi_lo,ledReg_newUInt_hi_h
i_lo_hi,
65
    ledReg_newUInt_hi_hi_lo_lo,ledReg_newUInt_hi_lo_hi_hi,ledReg_newUInt_hi_l
o_hi_lo,ledReg_newUInt_hi_lo_lo_lo,
66     ledReg_newUInt_hi_lo_lo_lo,ledReg_newUInt_lo}; // @[Cat.scala 30:58]
67   wire [31:0] _clkReg_T_2 = clkReg + 32'h1; // @[flowLights.scala 36:51]
68   assign io_led = ledReg; // @[flowLights.scala 38:10]
69   always @(posedge clock) begin
70     if (reset) begin // @[flowLights.scala 15:23]
71       ledReg <= 16'h1; // @[flowLights.scala 15:23]
72     end else if (io_button) begin // @[flowLights.scala 30:27]
73       ledReg <= ledReg_res; // @[flowLights.scala 31:12]
74     end else if (clkReg == 32'h0) begin // @[flowLights.scala 35:18]
75       ledReg <= ledReg_newUInt;
76     end
77     if (reset) begin // @[flowLights.scala 16:23]
78       clkReg <= 32'h1; // @[flowLights.scala 16:23]
79     end else if (io_button) begin // @[flowLights.scala 30:27]
80       clkReg <= 32'h2; // @[flowLights.scala 33:12]

```



```

81     end else if (clkReg == 32'h2) begin // @[flowLights.scala 36:18]
82         clkReg <= 32'h0;
83     end else begin
84         clkReg <= _clkReg_T_2;
85     end
86     `ifndef SYNTHESIS
87     `ifdef PRINTF_COND
88         if (`PRINTF_COND) begin
89     `endif
90         if (io_button & ~reset) begin
91             $fwrite(32'h80000002,"res = %b\n",ledReg_res); //
@[flowLights.scala 26:11]
92         end
93     `ifdef PRINTF_COND
94         end
95     `endif
96     `endif // SYNTHESIS
97     `ifndef SYNTHESIS
98     `ifdef PRINTF_COND
99         if (`PRINTF_COND) begin
100     `endif
101         if (io_button & _ledReg_T_3) begin
102             $fwrite(32'h80000002,"generated: %b\n",ledReg); //
@[flowLights.scala 32:11]
103         end
104     `ifdef PRINTF_COND
105         end
106     `endif
107     `endif // SYNTHESIS
108     end
109 // Register and memory initialization
110 `ifndef RANDOMIZE_GARBAGE_ASSIGN
111 `define RANDOMIZE
112 `endif
113 `ifndef RANDOMIZE_INVALID_ASSIGN
114 `define RANDOMIZE
115 `endif
116 `ifndef RANDOMIZE_REG_INIT
117 `define RANDOMIZE
118 `endif
119 `ifndef RANDOMIZE_MEM_INIT
120 `define RANDOMIZE
121 `endif
122 `ifndef RANDOM
123 `define RANDOM $random
124 `endif
125 `ifndef RANDOMIZE_MEM_INIT
126     integer initvar;
127 `endif
128 `ifndef SYNTHESIS
129 `ifndef FIRRTL_BEFORE_INITIAL
130 `FIRRTL_BEFORE_INITIAL
131 `endif
132 initial begin
133     `ifdef RANDOMIZE
134         `ifdef INIT_RANDOM
135             `INIT_RANDOM
136         `endif

```

```

137     `ifndef VERILATOR
138         `ifdef RANDOMIZE_DELAY
139             #`RANDOMIZE_DELAY begin end
140         `else
141             #0.002 begin end
142         `endif
143     `endif
144     `ifdef RANDOMIZE_REG_INIT
145         _RAND_0 = {1{`RANDOM}};
146         ledReg = _RAND_0[15:0];
147         _RAND_1 = {1{`RANDOM}};
148         clkReg = _RAND_1[31:0];
149     `endif // RANDOMIZE_REG_INIT
150     `endif // RANDOMIZE
151 end // initial
152 `ifdef FIRRTL_AFTER_INITIAL
153     `FIRRTL_AFTER_INITIAL
154 `endif
155 `endif // SYNTHESIS
156 endmodule
157 module FlowLightsWrapper(
158     input      clock,
159     input      reset,
160     input      io_button,
161     input [2:0] io_ledCount,
162     output [15:0] io_led
163 );
164     wire flowLights_clock; // @[flowLights.scala 50:26]
165     wire flowLights_reset; // @[flowLights.scala 50:26]
166     wire flowLights_io_button; // @[flowLights.scala 50:26]
167     wire [2:0] flowLights_io_ledCount; // @[flowLights.scala 50:26]
168     wire [15:0] flowLights_io_led; // @[flowLights.scala 50:26]
169     FlowLights flowLights ( // @[flowLights.scala 50:26]
170         .clock(flowLights_clock),
171         .reset(flowLights_reset),
172         .io_button(flowLights_io_button),
173         .io_ledCount(flowLights_io_ledCount),
174         .io_led(flowLights_io_led)
175     );
176     assign io_led = flowLights_io_led; // @[flowLights.scala 53:10]
177     assign flowLights_clock = clock;
178     assign flowLights_reset = reset;
179     assign flowLights_io_button = io_button; // @[flowLights.scala 52:24]
180     assign flowLights_io_ledCount = io_ledCount; // @[flowLights.scala
181     55:32]
182 endmodule
183
184 module holiday_lights (input wire clk,
185                       input wire rst,
186                       input wire button,
187                       input wire [2:0] switch,
188                       output wire [15:0] led);
189     FlowLightsWrapper u0(
190         .clock(clk),
191         .reset(rst),
192         .io_button(button),
193         .io_led(led),

```

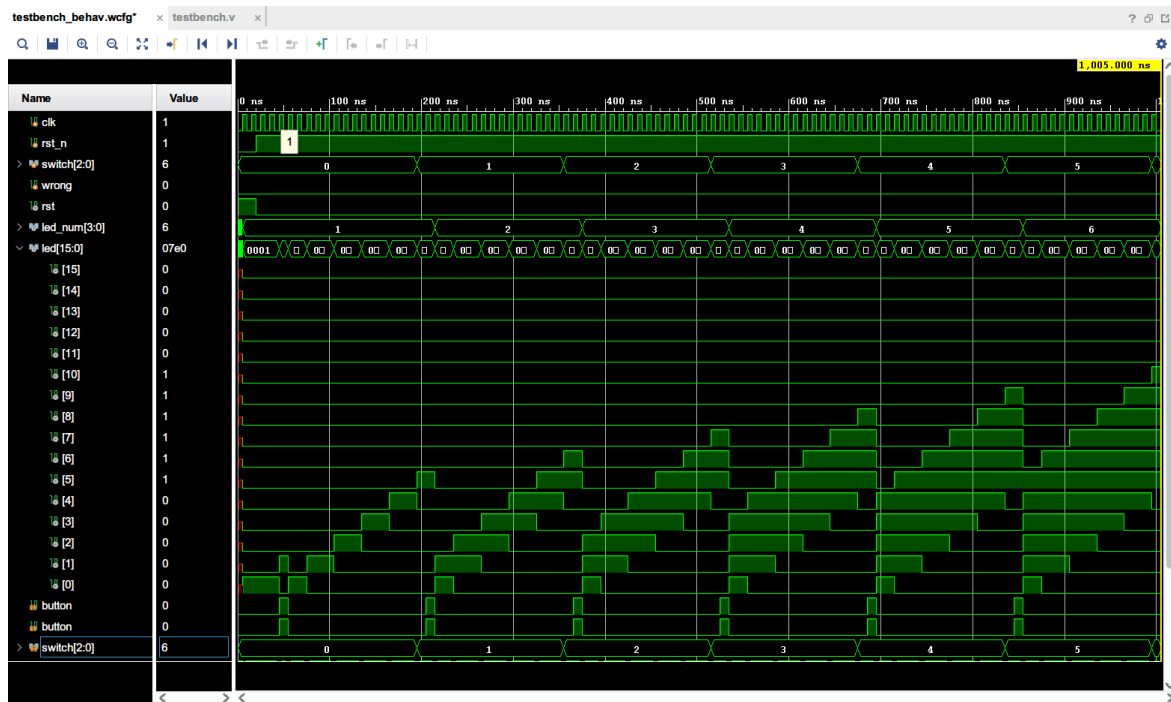
```

194         .io_ledCount(switch)
195     );
196 endmodule

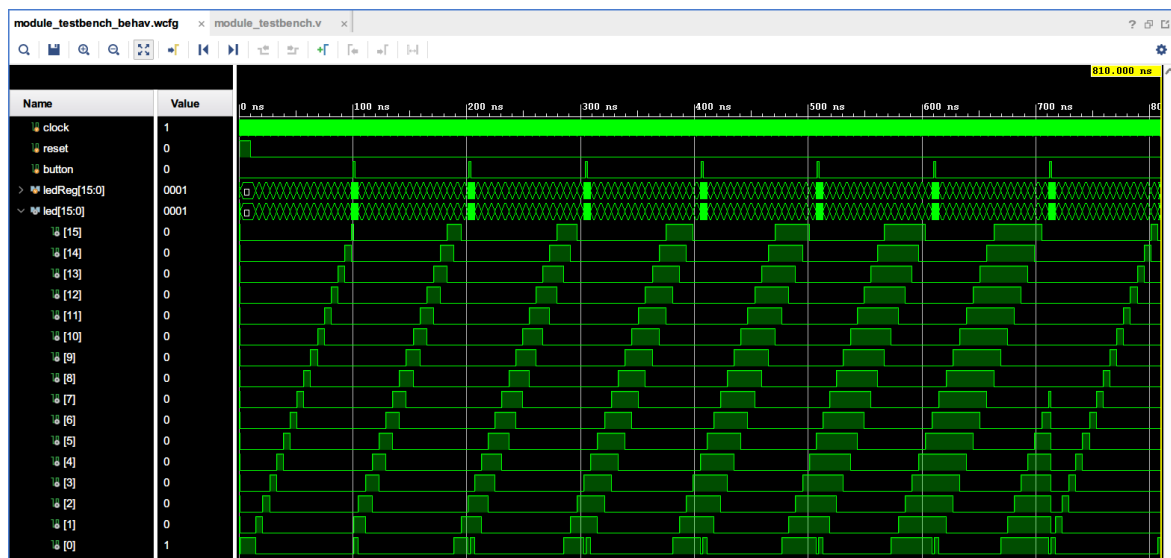
```

波形数据

本实验的 testbench。



自己的 testbench。



testbench:

```

1  `timescale 1ns / 1ps
2  ////////////////////////////////////////
3  // Company:
4  // Engineer:
5  //
6  // Create Date: 2021/11/12 17:31:54
7  // Design Name:
8  // Module Name: module_testbench

```

```

9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21 ///////////////////////////////////////////////////////////////////
22
23 module module_testbench();
24     reg clock;
25     reg reset;
26
27     reg button;
28     wire [15:0] led;
29     reg [2:0] ledCount;
30
31     always #1 clock <= ~clock;
32
33     always #100 begin
34         if (ledCount == 3'b111)
35             ledCount <= 1;
36         else
37             ledCount <= ledCount + 3'b1;
38         button <= 1'b1;
39         # 2
40         button <= 1'b0;
41     end
42
43     initial begin
44         clock <= 1'b0;
45         reset <= 1'b1;
46         button <= 1'b0;
47         ledCount <= 3'b1;
48         # 10
49         reset <= 1'b0;
50         # 800
51         $finish;
52     end
53
54     FlowLightswrapper u0(
55         .clock(clock),
56         .reset(reset),
57         .io_button(button),
58         .io_led(led),
59         .io_ledCount(ledCount)
60     );
61 endmodule
62

```