湖南大學
HUNAN UNIVERSITY

集成电路实验一

数字系统综合实验

Quartus II、Modelsim、FPGA 操作入门

报告人

第一组物理 2001 孙陶庵

202011010101

实验时间地点

2023/12/05

物电院 A116

集成电路实验一

2023 年 12 月 7 日

目录

1 实验原理

2 实验内容

2.1 实验步骤

使用 Quartus II 编程，写出一份可以操控电路板上 led 灯的程序，并调控参数观察现象。

实验步骤：

1. 电脑创建程序写出程序
2. 编译，debug
3. 使用 Modelsim 仿真并观察波形，确认无误后将电路板接进电脑
4. 使用 Quartus II 将程序导入电路板

2.2 实验结果

实验中使用的代码：1.

```
1
2 module led (input wire clk,input wire rst_n,output  reg[3:0] led);
3
4 reg [25:0] cnt_1s;    //1秒计数器
5
6 parameter END_CNT_1S = 499999999;  //1秒计数结束标识
7
8 always @(posedge clk or negedge rst_n)
9   if(!rst_n)
10  cnt_1s <='d0;
11  else if(cnt_1s==END_CNT_1S)
12  cnt_1s <='d0;
13  else cnt_1s <= cnt_1s+1'b1;
14 always @(posedge clk or negedge rst_n)
```

```
  if(!rst_n)
  led <= 4'b0000;
  else if(cnt_1s==END_CNT_1S)
    case(led)
    4'b0001:  led<=4'b1000;
    4'b0010:  led<=4'b0001;
    4'b0100:  led<=4'b0010;
    4'b1000:  led<=4'b0100;
    default:  led<=4'b0000;
    endcase
  else led <= led;
endmodule
```

cnt_1s 是一个 26 位的计数器，用于计算时钟信号 clk 的上升沿。在正常运行时，计数器逐次递增，表示经过的时钟周期数。

END_CNT_1S 是一个参数，它定义了计数器的结束标识。在这个例子中，计数器 *cnt_1s* 的结束标识是 *END_CNT_1S*，当计数器达到这个值时，将重新计数。

always@(posedge clk or negedge rst_n) 语句块用于处理时钟和异步复位信号。在时钟的上升沿或异步复位信号 *rst_n* 的下降沿触发时执行相应的逻辑。

在第一个 always 块中，计数器 *cnt_1s* 在每个时钟周期上升沿时递增。在异步复位信号 *rst_n* 的下降沿时，计数器被清零。

在第二个 always 块中，LED 状态机的逻辑被实现。当异步复位信号 *rst_n* 的下降沿发生时，LED 被初始化为全灭状态 (4'b0000)。在计数器 *cnt_1s* 达到结束标识 *END_CNT_1S* 时，LED 状态根据一个简单的状态机进行变化。LED 在四种状态之间循环切换，实现了一个基本的流水灯效果。

```

`timescale 1ns/1ns
module tb_led;
  reg clk;
  reg rst_n;
```

```
  wire[3:0] led;

  led led_inst(
      .clk  (clk),
      .rst_n (rst_n),
      .led  (led)
      );
always
  begin
    #10;
    clk=~clk;
  end
initial
  begin
    clk =1'b1;
    rst_n=1'b0;
    #50
    rst_n=1'b1;
    #400000000;
    $stop;
  end
endmodule
```

timescale 1ns/1ns：这是一个时间刻度（timescale）声明，指定了时序仿真的时间单位和时间精度。

module *tb_led*;：定义一个模块 *tb_led*，即测试台模块。

reg clk; 和 reg *rst_n*;：声明时钟信号 clk 和异步复位信号 *rst_n*，这些信号将用于测试。

wire [3:0] led;：声明一个 4 位宽的 led 信号，用于连接被测试的 led 模块的输出。

led *led_inst*(...);：实例化被测试的 led 模块，并连接输入输出信号。

always begin ... end：一个无限循环的 always 块，用于模拟时钟信号 clk 的

上升沿。

#10;：表示延迟 10 个时间单位，模拟时钟周期。clk = clk;：在每个时钟周期的上升沿和下降沿交替改变 clk 的值。initial begin ... end：初始块，用于在仿真开始时初始化测试环境和控制仿真流程。

clk = 1'b1;：初始时钟信号为高电平。*rst_n* = 1'b0;：初始时，异步复位信号为低电平。#50：等待一段时间，模拟异步复位信号的持续时间。*rst_n* = 1'b1;：将异步复位信号置为高电平，启动 LED 模块的正常操作。#400000000;：等待足够的时间，以确保仿真足够长以捕获 LED 模块的运行情况。$stop;：停止仿真。这个测试台通过时钟信号和异步复位信号对 led 模块进行仿真测试，可以观察模块在仿真时的行为。

2.3 拓展实验

读懂代码，试修改代码，完成以下实验步骤。

a. 如何改变流水灯的移动速度？

b. 如何改变流水灯的移动方向？

c. 如何移动两个 LED 灯？

d. 分别用 *Key_1*、*Key_2*、*Key_3* 三个按键来控制移动频率 1Hz、2Hz 和移动方向

e. 查看资源占有率及 RTL 视图

拓展实验结果：

a.LED 的移动速度是通过计时器 counter 和计时结束标识 *end_cnt* 来控制的。所以可以透过修改这一部分来改变移动速度

```
1    always @(posedge clk or negedge rst_n or negedge key_1 or negedge
     key_2)
2    if (!rst_n)
3        counter <= 0;
4        end_cnt <= END_CNT_1S;  // 初始设置为1秒的计时结束标识
5    else if (!key_1)  // 按下后周期变为1S
6        end_cnt <= END_CNT_1S;
```

```
    else if (!key_2)  // 按下后周期变为2S
        end_cnt <= END_CNT_2S;
    else if (counter == end_cnt)
        counter <= 0;
    else
        counter <= counter + 1'b1;
```

通过按下 *Key*_1，LED 的速度将被设置为 1 秒一次，而按下 *Key*_2，LED 的速度将被设置为 2 秒一次。这是因为 *end*_*cnt* 在这里用于确定计时器何时达到结束状态，进而改变 LED 的状态。

因此，如果希望改变 LED 的移动速度，可以通过按下不同的按键来选择不同的计时结束标识。

b.LED 的移动方向是通过按键输入 *key*_3 控制的。按下 *key*_3 时，direction 寄存器会取反，从而改变 LED 的移动方向。

```

always @(posedge clk or negedge rst_n or negedge key_1 or negedge
   key_2 or negedge key_3)
    if (!rst_n)
    begin
        counter <= 0;
        end_cnt <= END_CNT_1S;  // 初始设置为1秒的计时结束标识
        direction <= 1'b0;  // 初始方向为0
    end
    else if (!key_1)  // 按下后周期变为1S
        end_cnt <= END_CNT_1S;
    else if (!key_2)  // 按下后周期变为2S
        end_cnt <= END_CNT_2S;
    else if (!key_3)  // 按下后改变方向
        direction <= ~direction;  // 取反方向
    else if (counter == end_cnt)
        counter <= 0;
    else
```

```
18          counter <= counter + 1'b1;
```

当按下 *key*_3 时，direction 的值将翻转，从而改变 LED 的移动方向。这是通过 direction <= direction; 这一语句实现的。

因此，如果希望改变 LED 的移动方向，只需按下 *key*_3 即可。

c. 可以尝试使用两个计时器和两个计时结束标识来分别控制两个 LED 的移动。

```
1
2  module dual_led (
3      input wire clk,
4      input wire rst_n,
5      input wire key_1,
6      input wire key_2,
7      input wire key_3,
8      output reg[3:0] led1,
9      output reg[3:0] led2
10 );
11
12 reg [25:0] counter1;     // 1秒计数器1
13 reg [25:0] counter2;     // 1秒计数器2
14 reg [25:0] end_cnt1;     // 结束时间周期1
15 reg [25:0] end_cnt2;     // 结束时间周期2
16 reg direction1;          // 方向1，0左1右
17 reg direction2;          // 方向2，0左1右
18
19 parameter END_CNT_1S = 4999999;  // 1秒计数结束标识
20 parameter END_CNT_2S = 9999999;  // 2秒计数结束标识
21
22 always @(posedge clk or negedge rst_n or negedge key_1 or negedge
       key_2 or negedge key_3)
23     if (!rst_n)
24     begin
25         counter1 <= 0;
26         end_cnt1 <= END_CNT_1S;
```

```
27          direction1 <= 1'b0;  // 初始方向为0
28      end
29      else if (!key_1)
30          end_cnt1 <= END_CNT_1S;
31      else if (!key_2)
32          end_cnt1 <= END_CNT_2S;
33      else if (!key_3)
34          direction1 <= ~direction1;
35
36      // Counter1 logic
37      if (counter1 == end_cnt1)
38          counter1 <= 0;
39      else
40          counter1 <= counter1 + 1'b1;
41
42  always @(posedge clk or negedge rst_n or negedge key_1 or negedge
        key_2 or negedge key_3)
43      if (!rst_n)
44      begin
45          counter2 <= 0;
46          end_cnt2 <= END_CNT_1S;
47          direction2 <= 1'b0;  // 初始方向为0
48      end
49      else if (!key_1)
50          end_cnt2 <= END_CNT_1S;
51      else if (!key_2)
52          end_cnt2 <= END_CNT_2S;
53      else if (!key_3)
54          direction2 <= ~direction2;
55
56      // Counter2 logic
57      if (counter2 == end_cnt2)
58          counter2 <= 0;
59      else
60          counter2 <= counter2 + 1'b1;
```

```
61
62 always @(posedge clk or negedge rst_n)
63 begin
64     if (!rst_n)
65     begin
66         led1 <= 4'b0001;
67         led2 <= 4'b0001;
68     end
69     else
70     begin
71         // LED1 logic
72         case({counter1 == end_cnt1, direction1})
73             2'b10: led1 <= led1 << 1;
74             2'b01: led1 <= led1 >> 1;
75             default: led1 <= led1;
76         endcase
77
78         // LED2 logic
79         case({counter2 == end_cnt2, direction2})
80             2'b10: led2 <= led2 << 1;
81             2'b01: led2 <= led2 >> 1;
82             default: led2 <= led2;
83         endcase
84     end
85 end
86
87 endmodule
```

分别由两个计时器 counter1 和 counter2 控制。按键 *key*_3 仍然用于改变方向，而按键 *key*_1 和 *key*_2 分别用于选择 1 秒和 2 秒的计时结束标识。LED 的移动逻辑分别由两个独立的 case 语句控制。

d. *Key*_1 选择 1Hz 的计时结束标识和向左移动。*Key*_2 选择 2Hz 的计时结束标识和向左移动。*Key*_3 按下则切换移动方向。