**北京科技大学数字逻辑实验报告**

学院：计通 专业： 班级：

姓名： 学号： 实验日期：2024 年 11 月 5 日

**1、实验名称：**实验三 时序逻辑实验（一）

**2、实验目的：**

本实验的目的是学习时序逻辑模块（状态机）在数字系统中的应用；掌握实验平台的外部功能模块在数字系统设计中的应用。

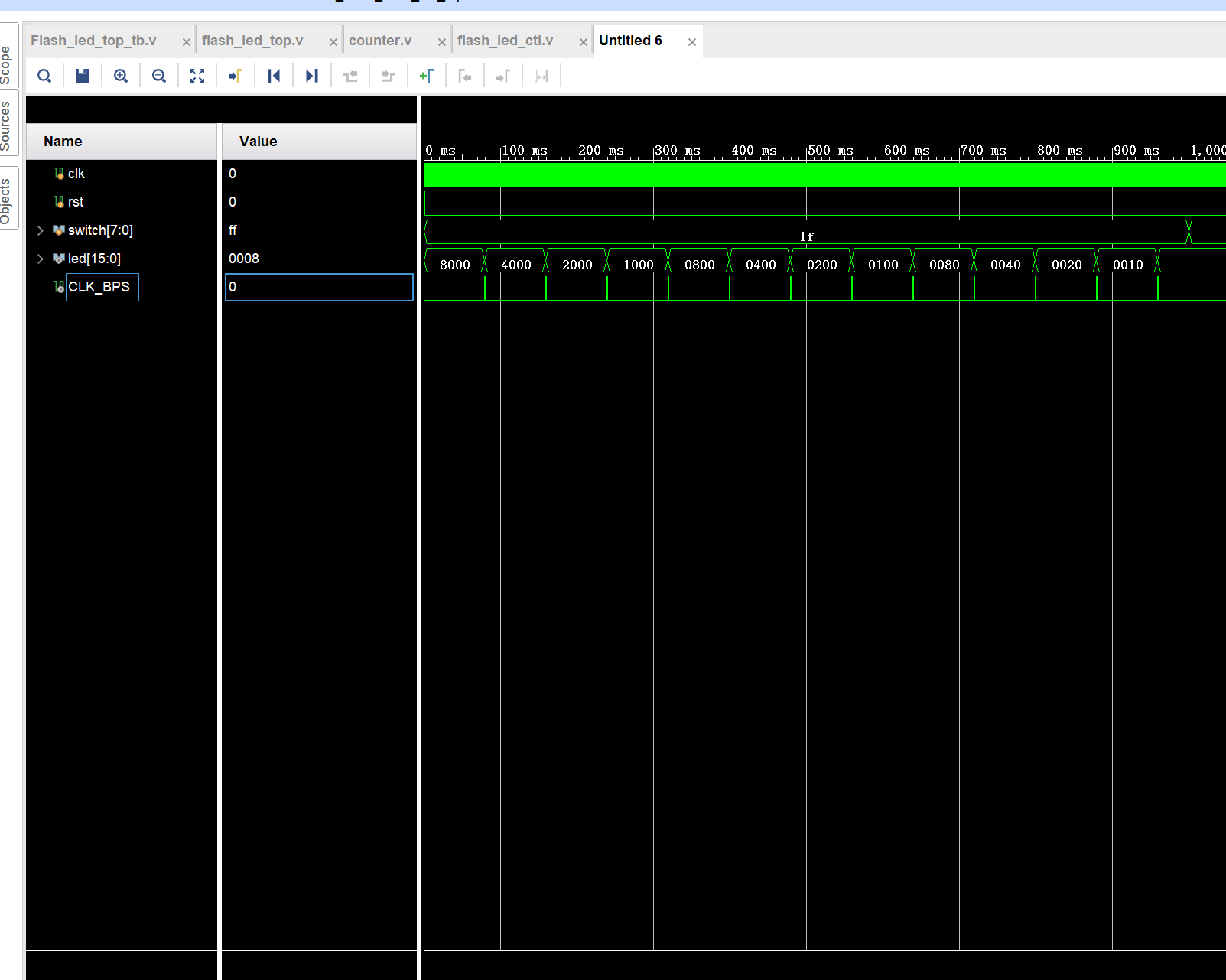
**3、实验内容：**

1. **实验3.1——流水灯**
   1. 了解分频模块的功能；
   2. 实现流水灯的流动功能；
   3. 修改代码，实现流水灯的**变速**流动功能。
2. **实验3.2——智能报警器**
   1. 了解消抖模块的功能；
   2. 完成智能报警器。
3. **实验3.3——密码锁**
   1. 按照要求实现密码锁；
   2. 本实验无需上版，只需仿真即可。

**实验步骤：**（详细的实验过程，可包括系统设计、RTL结构设计、源代码及注释等等，红色文字在最终报告中删除）

4、**实验3.1的实现及仿真验证**

（1）流水灯仿真结果截图，对波形进行简要解释



当开关开启五个的时候可以看到每次跑马灯向右移动一个灯。

（2）给出板子运行结果照片，以及操作过程

（3）counter模块的作用是什么？

对100MHz的时钟分频，降低频率，使得灯的闪烁可以被人眼识别，造成跑马灯的效果。

5、**实验3.2的实现及验证**

（1）请说明你对消抖模块功能的理解

消抖模块通过检测一段时间（大约5~20ms）内的按钮是不是连续的高电平或者低电平来判断按钮到底有没有按下去。

（2）该模块的仿真验证波形及说明（简要说明testbench设计思路，以及为什么这个结果是正确的）

`timescale 1ns / 1ps

module tb\_alarm\_top;

// 输入信号

reg rst;

reg cle;

reg clk;

reg act; // B1

reg [1:0] swc; // k1, k2

// 输出信号

wire [3:0] pos; // seg pos

wire [7:0] seg; // seg msg

wire led; // flashing leds

// 实例化待测试模块

alarm\_top dut (

.rst(rst),

.cle(cle),

.clk(clk),

.act(act),

.swc(swc),

.pos(pos),

.seg(seg),

.led(led)

);

// 时钟生成：100MHz时钟

initial begin

clk = 0;

forever #5 clk = ~clk; // 100MHz => 时钟周期 10ns

end

// 测试过程

initial begin

// 初始化信号

rst = 1;

cle = 0;

act = 0;

swc = 2'b00;

// 复位阶段

#100 rst = 0; // 在20ns后释放复位

// 随机按下act，并随机更改swc的值

act = 1; // 模拟按下按钮

swc = 2'b10; // k1 = 1, k2 = 0

#5000

act = 0; // 模拟释放按钮

#5000000; // 等待，观察波形

swc = 2'b11; // k1 = 1, k2 = 1

act = 1; // 再次模拟按下按钮

#5000

act = 0; // 模拟释放按钮

#5000000;

swc = 2'b01; // k1 = 0, k2 = 1

act = 1; // 再次模拟按下按钮

#5000

act = 0; // 模拟释放按钮

// 按下cle清除时间

#100; cle = 1; // 激活清除信号

#100 cle = 0; // 停止清除

// 等待一段时间后结束仿真

#200;

$finish;

end

endmodule

6、**实验3.3的实现及仿真验证**

（1）密码锁模块的代码，及简要解释

module lock (

input clk, // 时钟信号

input rst, // 复位信号

input in, // 输入信号

output reg unlock // 解锁信号

);

parameter IDLE = 3'b000;

parameter S0 = 3'b001;

parameter S00 = 3'b010;

parameter S001 = 3'b011;

parameter S0010 = 3'b100;

reg [2:0] state, next\_state;

always @(posedge clk or posedge rst) begin

if (rst)

state <= IDLE;

else

state <= next\_state;

end

always @(\*) begin

unlock = 1'b0;

case (state)

IDLE: begin

if (in == 1'b0)

next\_state = S0;

else

next\_state = IDLE;

end

S0: begin

if (in == 1'b0)

next\_state = S00;

else

next\_state = IDLE;

end

S00: begin

if (in == 1'b1)

next\_state = S001;

else

next\_state = S00;

end

S001: begin

if (in == 1'b0)

next\_state = S0010;

else

next\_state = IDLE;

end

S0010: begin

unlock = 1'b1;

next\_state = S0010;

end

default: next\_state = IDLE;

endcase

end

endmodule

（2）密码锁的仿真截图，并简要说明波形的正确性

图形用户界面, 应用程序

描述已自动生成

可以看到在输入0010之后unlock立刻变为高电平，并且之后保持

（3）测试激励的设计（说明：测试方案，包括数据如何产生、测试哪些情况、如何判定结果是否正确、如何对比性能等等）

`timescale 1ns / 1ps

module lock\_tb;

// 信号声明

reg clk;

reg rst;

reg in;

wire unlock;

// 实例化密码锁模块

lock uut (

.clk(clk),

.rst(rst),

.in(in),

.unlock(unlock)

);

// 时钟生成

initial begin

clk = 0;

forever #5 clk = ~clk; // 每5ns翻转一次，产生10ns的时钟周期

end

// 测试过程

initial begin

// 初始化

rst = 1;

in = 0;

#20; // 复位保持一段时间

rst = 0; // 取消复位

// 随机输入 0 和 1，持续一段时间观察解锁信号

repeat (50) begin

in = $random % 2; // 生成随机的0或1

#10; // 每10ns改变一次输入信号

end

// 手动输入 "0010" 来检查解锁状态

#10 in = 0;

#10 in = 0;

#10 in = 1;

#10 in = 0;

// 等待一段时间观察解锁信号

#50;

// 测试结束

$finish;

end

// 监视输出

initial begin

$monitor("Time = %0t | in = %b | unlock = %b", $time, in, unlock);

end

endmodule

**7、实验中遇到的问题、现象及解决方法（如没遇到可不写，帮助别人解决的也可以写）**

问题1：（截屏或拍照或文字描述）

现象：

问题原因：

解决方法：

问题2：

**9、本次实验心得体会**

（学到了什么，或理解了什么，或加深了什么的认识等，无需长篇大论）

**10、关于本次实验课程的改进建议**

（没有可以不写）