

浙江大学

本科实验报告

课程名称:	计算机逻辑设计基础
姓名:	李瀚轩
学院:	竺可桢学院
系:	所在系
专业:	计算机科学与技术
学号:	3220106039
指导教师:	董亚波

2024 年 1 月 3 日

浙江大学实验报告

课程名称: 计算机逻辑设计基础 实验类型: 综合

实验项目名称: EDA 实验平台与实验环境运用

学生姓名: 李瀚轩 专业: 计算机科学与技术 学号: 3220106039

同组学生姓名: 指导老师: 董亚波

实验地点: 东四 509 实验日期: 2023 年 10 月 12 日

一、实验目的和要求

1. 熟悉 Verilog HDL 语言并能用其建立基本的逻辑部件, 在 Xilinx ISE 平台进行输入、编辑、调试、行为与仿真与综合后功能仿真
2. 熟悉掌握 SWORD FPGA 开发平台, 同时在 ISE 平台上进行时序约束、引脚约束及映射布线后时序仿真
3. 运用 Xilinx ISE 具将设计验证后的代码下载到实验板上, 并在实验板上验证

二、实验内容和原理

内容:

1. 熟悉 ISE 工具软件的运行环境与安装过程
2. 设计简单组合逻辑电路，采用图形输入逻辑功能描述，建立 FPGA 实现数字系统的 Xilinx ISE 设计管理工程，并进行编辑、调试、编译、行为仿真，时序约束、引脚指定（约束）、映射布线后时序仿真及 FPGA 编程代码下载与运行验证
3. 设计简单时序逻辑电路，采用 Verilog 代码输入逻辑功能描述，建立 FPGA 实现数字系统的 ISE 设计管理工程，并进行编辑、调试、编译、行为仿真，时序约束、引脚约束、映射布线后时序仿真及 FPGA 编程代码下载与运行验证

原理

1. 某三层楼房的楼梯通道共用一盏灯，每层楼都安装了一只开关并能独立控制该灯，请设计楼道灯的控制电路
 - (a) 分析楼道灯的事件行为，用组合电路实现，用拨动开关作为电路输入 S_1, S_2, S_3 ，电路输出为 F
 - (b) 变量赋值: 开关往下为 1，往上为 0; 输出灯亮为 1，灯暗为 0
 - (c) 编写真值表并根据真值表分析输入输出关系 $F = S_1 \overline{S_2} \overline{S_3} + \overline{S_1} S_2 \overline{S_3} + \overline{S_1} \overline{S_2} S_3 + S_1 S_2 S_3$, 据此逻辑关系可画出电路图
2. 增加控制要求，灯打开后，延时若干秒自动关闭，请重新设计楼道灯的控制电路
 - (a) 分析楼道灯的事件行为，用时序电路实现，用按钮开关作为电路输入 S_1, S_2, S_3 ，电路输出为 F
 - (b) 变量赋值: 开关按下为 1，弹起为 0; 输出灯亮为 1，灯暗为 0
 - (c) 编写 Verilog 代码实现功能

三、实验步骤和结果记录

3.1 以图形方式输入逻辑功能描述

3.1.1 建立楼道控制的工程

1. 进入软件后，点击左上角 File 选项卡，然后点击下拉菜单中的第一项——New Project
2. 设置属性：
在对话框中设置：Project Name:LampCtrl_sch
Top-Level Source Type:Schematic
确认后，点击 Next 到设备属性页，设置：
Family: Kintex7
Device: XC7K160T
Package: FFG676
Speed: -1
3. 确认后，一直点击 Next 直到创建工程结束

3.1.2 创建原理图文件

1. 在 Sources 窗口 Sources 选项卡设备型号名处右键菜单选择 New Source
2. 新建源文件向导中选择源文件类型为 Schematic，输入文件名 LampCtrl，勾选 Add to Project
3. 连续点击 Next，最后点击 Finish；在 Sources 窗口中双击刚新建的文件图标，进入电路原理图编辑窗口

3.1.3 输入楼道灯控逻辑电路

1. 在 Sources 窗口中选择 Symbols 选项卡，配合 Schematic Editor 工具条输入原理图

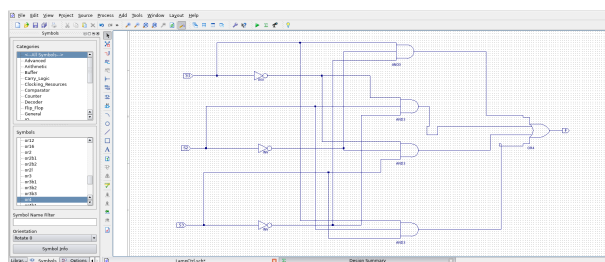


图 1: 原理图

3.1.4 查看输入电路的硬件描述代码

在 Sources 窗口中选择 Sources for: Synthesis / Implementation, 选中 LampCtrl.sch 图标, 在 Processes 窗口 Processes 选项卡中展开 Design Utilities 并双击 View HDL Functional Model

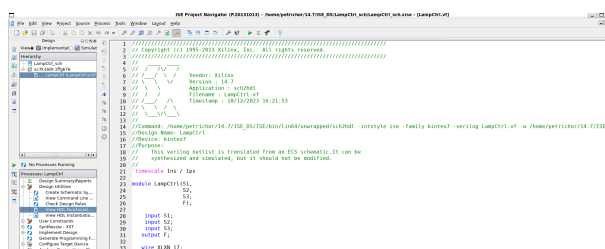


图 2: 查看输入电路的硬件描述代码

硬件描述代码如下所示:

```
'timescale 1ns / 1ps

module LampCtrl(S1,
                S2,
                S3,
                F,
                LED);

    input S1;
    input S2;
    input S3;
    output F;
    output [6:0] LED;
```

```

wire XLXN_17;
wire XLXN_27;
wire XLXN_28;
wire XLXN_30;
wire XLXN_31;
wire XLXN_32;
wire XLXN_33;

INV  XLXI_1  ( . I ( S1 ) ,
               . O ( XLXN_28 ) );
INV  XLXI_2  ( . I ( S2 ) ,
               . O ( XLXN_17 ) );
INV  XLXI_3  ( . I ( S3 ) ,
               . O ( XLXN_27 ) );
AND3  XLXI_4  ( . I0 ( XLXN_27 ) ,
                . I1 ( XLXN_17 ) ,
                . I2 ( S1 ) ,
                . O ( XLXN_30 ) );
AND3  XLXI_5  ( . I0 ( S3 ) ,
                . I1 ( XLXN_17 ) ,
                . I2 ( XLXN_28 ) ,
                . O ( XLXN_32 ) );
AND3  XLXI_6  ( . I0 ( S3 ) ,
                . I1 ( S2 ) ,
                . I2 ( S1 ) ,
                . O ( XLXN_33 ) );
AND3  XLXI_7  ( . I0 ( XLXN_27 ) ,
                . I1 ( S2 ) ,
                . I2 ( XLXN_28 ) ,
                . O ( XLXN_31 ) );
OR4   XLXI_8  ( . I0 ( XLXN_33 ) ,
                . I1 ( XLXN_32 ) ,
                . I2 ( XLXN_31 ) ,
                . I3 ( XLXN_30 ) ,

```

```

        .O(F));
GND    XLXI_11  (.G(LED[6]));
GND    XLXI_12  (.G(LED[5]));
GND    XLXI_13  (.G(LED[4]));
GND    XLXI_14  (.G(LED[3]));
GND    XLXI_15  (.G(LED[2]));
GND    XLXI_16  (.G(LED[1]));
GND    XLXI_17  (.G(LED[0]));
endmodule

```

3.1.5 建立基准测试波形文件

- (a) 在 Sources 窗口空白处的右键菜单中选择 New Source
- (b) 在新建源文件向导中选择源类型为：Verilog Test Fixture，输入文件名 LampCtrl_sim，并勾选 Add to Project
- (c) 选择 LampCtrl 模块，点击 Next，在 Summary 窗口再点击 Finish，进入 LampCtrl_sim.v 编辑窗口

3.1.6 仿真激励输入

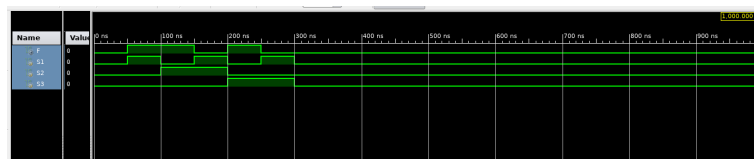
- (a) 在刚刚新建的源文件中写入如下代码

```

3  `timescale 1ns / 1ps
4
5  module LampCtrl_LampCtrl_sch_tb();
6
7  // Inputs
8      reg S1;
9      reg S2;
10     reg S3;
11
12 // Output
13     wire F;
14
15 // Bidirs
16
17 // Instantiate the UUT
18     LampCtrl UUT (
19         .S1(S1),
20         .S2(S2),
21         .S3(S3),
22         .F(F)
23     );
24 // Initialize Inputs
25 // `ifdef auto_init
26     integer i;
27     initial begin
28         for(i=0;i<=8;i=i+1)begin
29             {S3,S2,S1}<=i;
30             #50;
31         end
32     end
33 // `endif
34 endmodule
35

```

(b) View 选择 Simulation 视图，Hierarchy 窗口中选择 LampCtrl_LampCtrl_sch_tb, Process 窗口中选择 Simulate Behavioral Model，可以看到波形如下图所示：



(c) 我们可以写入如下代码，得到另一种仿真激励输入


```

3 `timescale 1ns / 1ps
4
5 module LampCtrl_LampCtrl_sch_tb();
6
7 // Inputs
8 reg S1;
9 reg S2;
10 reg S3;
11
12 // Output
13 wire F;
14
15 // Bidirs
16
17 // Instantiate the UUT
18 LampCtrl UUT (
19     .S1(S1),
20     .S2(S2),
21     .S3(S3),
22     .F(F)
23 );
24 // Initialize Inputs
25 // `ifdef auto_init
26 integer i;
27 initial begin
28     for(i=0;i<=8;i=i+1)begin
29         {S3,S2,S1}<=i;
30         #50;
31     end
32 end
33 // `endif
34 endmodule
35

```

图 3: 波形

结果如图:

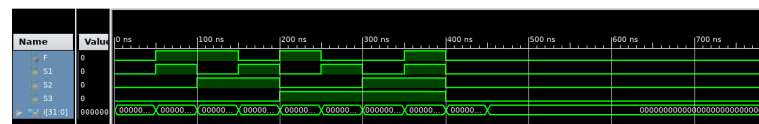


图 4: 另一种仿真激励输入波形

3.1.7 建立用户时序约束并为模块的端口指定引脚分配

(a) 在 Lamp_Ctrl.sch 里面绘制总线，如图所示

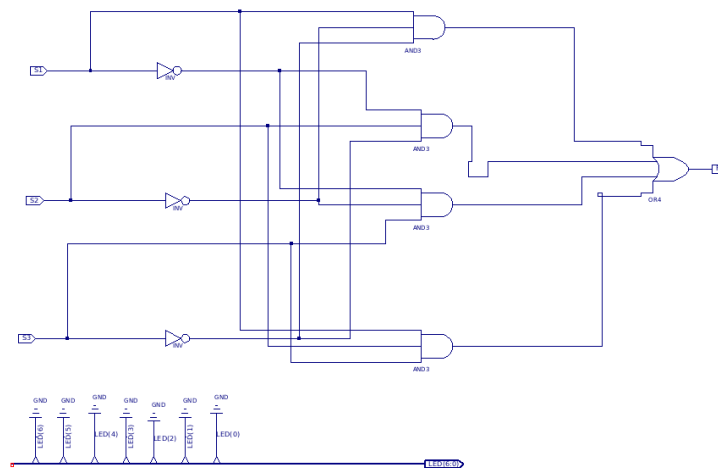


图 5: 最终电路

(b) XC7K160T 芯片的引脚如图所示

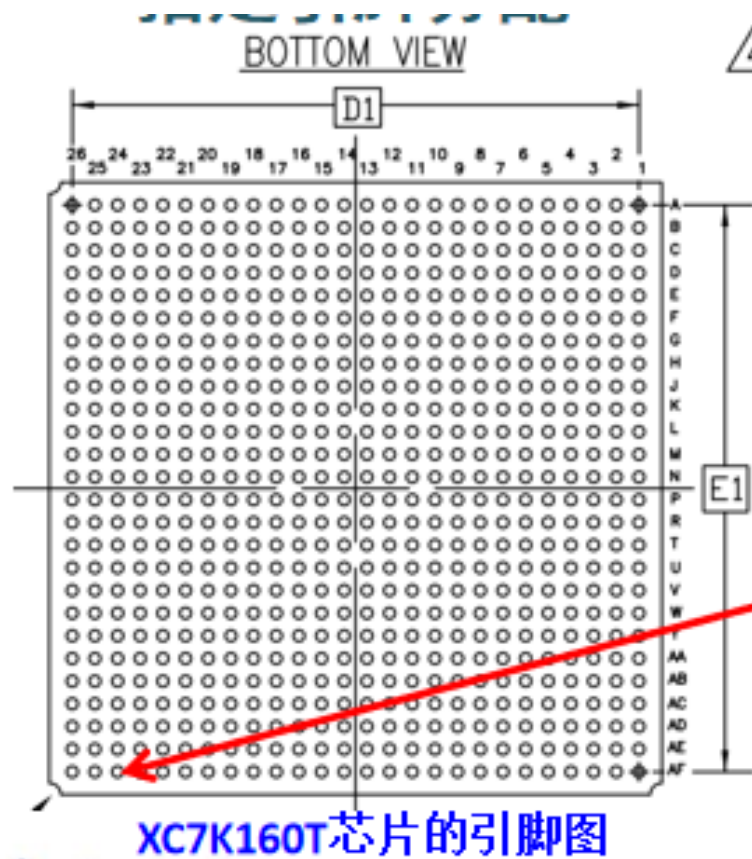


图 6: XC7K160T 芯片的引脚

(c) 在 Sources 窗口空白处的右键菜单中选择 New Source

(d) 在新建源文件向导中选择源类型为：Implementation Constraints File，输入文件名 K7，并勾选 Add to Project

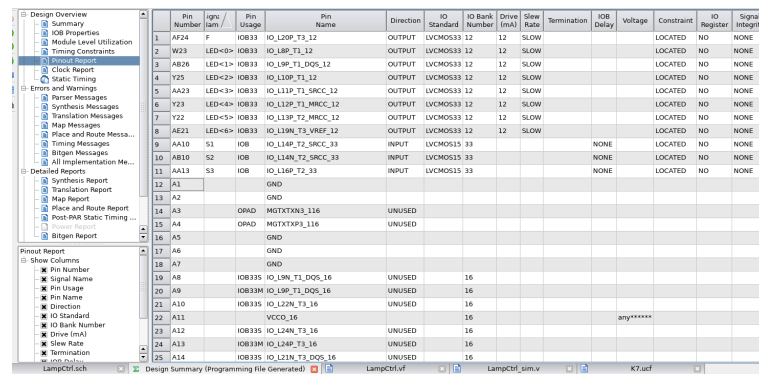
(e) 点击 Finish 进入 K7.ucf 编辑窗口，输入以下代码：

```
1 NET "S1" LOC = AA10 | IOSTANDARD = LVCMOS15;
2 NET "S2" LOC = AB10 | IOSTANDARD = LVCMOS15;
3 NET "S3" LOC = AA13 | IOSTANDARD = LVCMOS15;
4 NET "F" LOC = AF24 | IOSTANDARD = LVCMOS33;
5 NET "LED[0]" LOC = W23 | IOSTANDARD = LVCMOS33;
6 NET "LED[1]" LOC = AB26 | IOSTANDARD = LVCMOS33;
7 NET "LED[2]" LOC = Y25 | IOSTANDARD = LVCMOS33;
8 NET "LED[3]" LOC = AA23 | IOSTANDARD = LVCMOS33;
9 NET "LED[4]" LOC = Y23 | IOSTANDARD = LVCMOS33;
10 NET "LED[5]" LOC = Y22 | IOSTANDARD = LVCMOS33;
11 NET "LED[6]" LOC = AE21 | IOSTANDARD = LVCMOS33;
```

3.1.8 设计实现

在 Design 窗口中选择 Implementation，选中 LampCtrl 顶层模块；在 Processes 窗口下右键点击 Generate Programming File，选择 Run，进行物理转换、平面布图、映射、物理布线等 FPGA 综合操作，实现目标文件生成。生成的目标文件为 Lampctrl.bit。如果需要重新进行综合操作，可以选择 Rerun All

3.1.9 检查约束结果



Pin Number	I/O Bank	Pin Name	Direction	I/O Standard	Drive (mA)	Slew Rate	Termination	I/O Delay	Voltage	Constraint	I/O Register	Signal Integrity
1	AF24	F	IOB33	IO_L10P_T3_12	OUTPUT	LVCMOS33 12	12	SLOW		LOCATED	NO	NONE
2	W23	LED<0>	IOB33	IO_L10P_T1_12	OUTPUT	LVCMOS33 12	12	SLOW		LOCATED	NO	NONE
3	AB26	LED<1>	IOB33	IO_L10P_T1_D0S_12	OUTPUT	LVCMOS33 12	12	SLOW		LOCATED	NO	NONE
4	Y25	LED<2>	IOB33	IO_L10P_T1_12	OUTPUT	LVCMOS33 12	12	SLOW		LOCATED	NO	NONE
5	AA23	LED<3>	IOB33	IO_L11P_T1_SRCC_12	OUTPUT	LVCMOS33 12	12	SLOW		LOCATED	NO	NONE
6	Y23	LED<4>	IOB33	IO_L11P_T1_SRCC_12	OUTPUT	LVCMOS33 12	12	SLOW		LOCATED	NO	NONE
7	Y22	LED<5>	IOB33	IO_L11P_T2_SRCC_12	OUTPUT	LVCMOS33 12	12	SLOW		LOCATED	NO	NONE
8	AE21	LED<6>	IOB33	IO_L11P_T3_VREF_12	OUTPUT	LVCMOS33 12	12	SLOW		LOCATED	NO	NONE
9	AA10	S1	IOB	IO_L14P_T2_SRCC_33	INPUT	LVCMOS15 33			NONE	LOCATED	NO	NONE
10	AB10	S2	IOB	IO_L14P_T2_SRCC_33	INPUT	LVCMOS15 33			NONE	LOCATED	NO	NONE
11	AA13	S3	IOB	IO_L14P_T2_33	INPUT	LVCMOS15 33			NONE	LOCATED	NO	NONE
12	A1			GND								
13	A2			GND								
14	A3		ORAD	MGTXTP3_116	UNUSED							
15	A4		ORAD	MGTXTP3_116	UNUSED							
16	A5			GND								
17	A6			GND								
18	A7			GND								
19	A8		IOB335	IO_L10P_T1_D0S_16	UNUSED	16						
20	A9		IOB335	IO_L10P_T1_D0S_16	UNUSED	16						
21	A10		IOB335	IO_L22P_T3_16	UNUSED	16						
22	A11			VCCO_16		16				any*****		
23	A12		IOB335	IO_L24P_T3_16	UNUSED	16						
24	A13		IOB335	IO_L24P_T3_16	UNUSED	16						
25	A14		IOB335	IO_L21P_T3_D0S_16	UNUSED	16						

图 7: 设计摘要文档

然后将.bit 文件传送至实验板检验结果

(a) 在 Design 的 View 窗口中选择 Implementation

- (b) 在 Sources 窗口中选择 LampCtrl.sch; 在 Processes 窗口中, 用鼠标点开 Config Target Device, 双击 Manage Configuration Project(iMPACT) 选项, 出现 IMPACT 窗口
- (c) 双击 Boundary Scan 弹出下载编辑窗口
- (d) 鼠标右键选择 Initialize Chain, 系统自动查找已连接在电脑上的开发平台 JTAG 下载链
- (e) 接下来出现 Assign Configuration Files 对话框。这时从文件列表中选择 lampctrl.bit 文件, 将会为 JTAG chain 上的 xc7k160t 设备指定配置文件; 在弹出的 Attach SPI or PRI PROM 对话框弹出, 点击 NO 按钮; 在弹出的 Device Programming Property 对话框, 选择 OK 按钮即可
- (f) 右键点击 xc7k160t 设备图标, 选择菜单项 Program 后即可对硬件设备进行下载编程
- (g) 下载后, 在实验板上验证是否满足设计要求 (按 index 递增的顺序呈现)

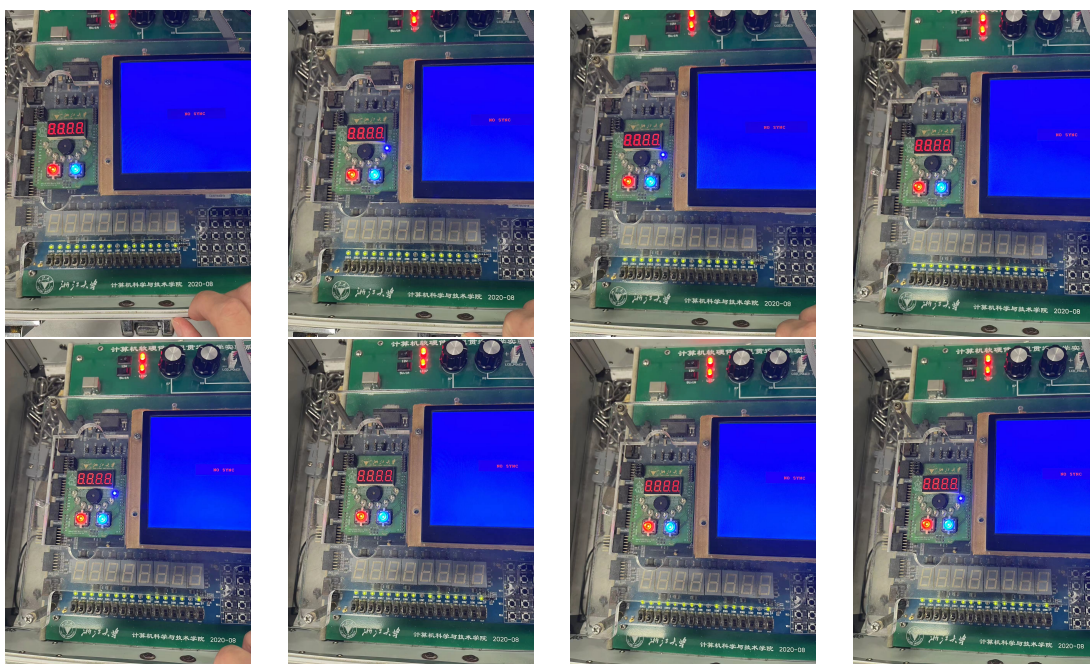


图 8: 实验板验证

3.2.Verilog 代码输入逻辑功能描述

3.2.1 建立楼道控制的工程

- (a) 依次点击菜单 File → New Project...
- (b) 在对话框中设置如下:
Project Name: LampCtrl_HDL
Top-Level Source Type: HDL
- (c) 确认后, 点击 Next 到设备属性页, 设置:
Family: Kintex7
Device: XC7K160T
Package: FFG676
Speed: -1
- (d) 确认后, 一直点击 Next 直到创建工程结束

3.2.2 创建 Verilog 输入源文件

- (a) 在 Sources 窗口空白处的右键菜单中选择 New Source
- (b) 在新建源文件向导中选择源类型为 Verilog Module, 输入文件名 LampCtrl, 勾选 Add to Project
- (c) 连续点击 Next, 直到 Finish

3.2.3 输入楼道灯控逻辑电路 Verilog HDL 代码

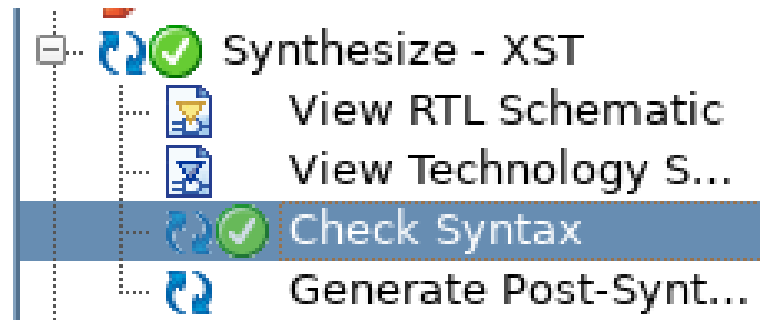
- (a) 在源代码编辑器, 输入代码

```

21 `timescale 1ns / 1ps
22 module LampCtrl(
23     input wire clk,
24     input wire S1,
25     input wire S2,
26     input wire S3,
27     output wire F
28 );
29
30 parameter C_NUM = 8;
31 parameter C_MAX = 8'hFF;
32
33 reg [C_NUM-1:0] count;
34 wire [C_NUM-1:0] c_next;
35 wire w;
36
37 initial begin
38     count = C_MAX;
39 end
40
41 assign w=S1||S2||S3;
42 assign F = (count<C_MAX)?1'b1:1'b0
43
44 always@(posedge clk)
45 begin
46     if(w==1'b1)
47         count=0;
48     else if(count< C_MAX)
49         count=c_next;
50 end
51
52 assign c_next=count+8'b1;
53
54 endmodule

```

(b) 检查输入代码的语法规则，并排除输入错误



(c) 延时时间修改

仿真时设定 `parameter C_NUM = 8; parameter C_MAX = 8' hFF;`

因此有 $\delta t = (2^8 - 1) \times 20ns = 5100ns$

下载运行时我们设定

`parameter C_NUM = 28; parameter C_MAX = 28' hFFFFFFF;`

因此有 $\delta t = (2^{28}) \times \frac{1}{100MHz} = 2.68s$

3.2.4 楼道控制电路代码的综合

(a) 在 Sources 窗口选中文件 LampCtrl.v

(b) 在 Processes 窗口运行 Synthesis XST → View RTL Schematic

(c) 观察综合的电路结构，尝试理解是否与设计目标一致。在某个模块上右键点击，在菜单里选择 Open Source of Selected Instance，可以定位到该模块对应的 Verilog

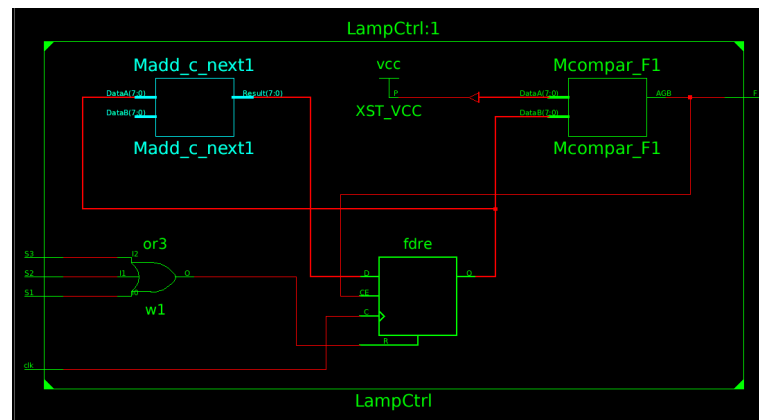
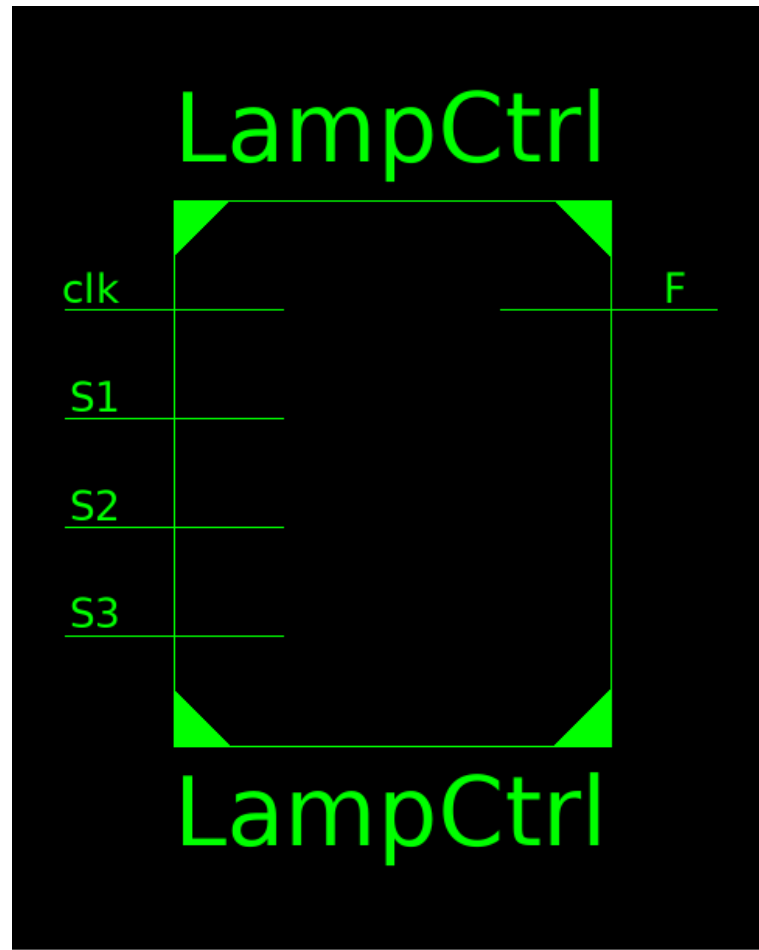


图 9: 双击之后查看到内部更详细的电路

3.2.5 建立基准测试波形文件

(a) 在 Sources 窗口空白处的右键菜单中选择 New Source

- (b) 在新建源文件向导中选择源类型为: Verilog Test Fixture, 输入文件名 LampCtrl_sim, 并勾选 Add to Project
- (c) 点击 Finish 进入 LampCtrl_sim.v 编辑窗口

3.2.6 仿真激励输入波形

为便于仿真, LampCtrl.v 代码中计数器位数采用的是 8 位长

```
module LampCtrl_sim;

    // Inputs
    reg clk;
    reg S1;
    reg S2;
    reg S3;

    // Outputs
    wire F;

    // Instantiate the Unit Under Test (UUT)
    LampCtrl uut (
        .clk(clk),
        .S1(S1),
        .S2(S2),
        .S3(S3),
        .F(F)
    );

    initial begin
        // Initialize Inputs
        clk = 0;
        S1 = 0;
        S2 = 0;
        S3 = 0;

        // Wait 100 ns for global reset to finish
```

```

        #600 S1=1;
        #20 S1=0;
        #6000 S2=1;
        #20 S2=0;
        #6000 S3=1;
        #20 S3=0;

        // Add stimulus here

    end

    always begin
        #10 clk=0;
        #10 clk=1;
    end

endmodule

```

在“Design”窗口选中“LampCtrl_sim.v”文件，“在“Processes”窗口双击“Behavioral Check Syntax”，通过语法检查后再双击“Simulate Behavioral Model”，此时会打开模拟程序软件 ISim。

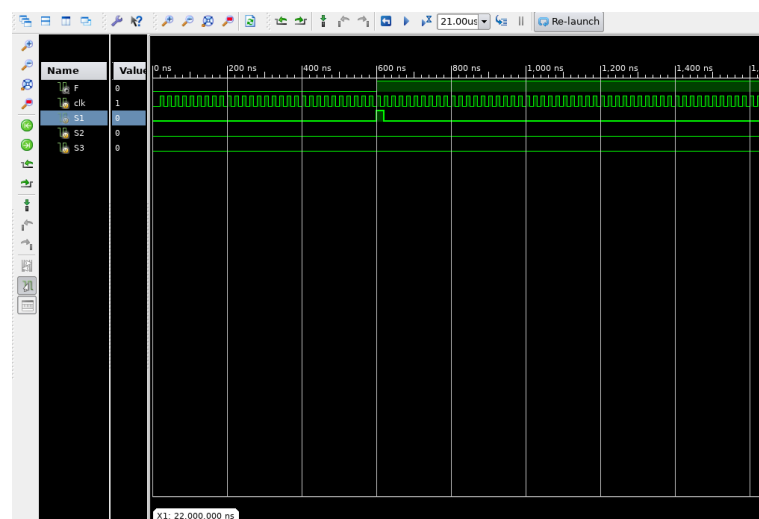


图 10: 在 ISim 中查看仿真波形

3.2.7 建立用户时序约束并为模块的端口指定引脚分配

(a) 建立引脚约束文件 k7.ucf，输入代码如下

```
1 NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18;  
2 NET "S1" LOC = AA10 | IOSTANDARD = LVCMOS15;  
3 NET "S2" LOC = AB10 | IOSTANDARD = LVCMOS15;  
4 NET "S3" LOC = AA13 | IOSTANDARD = LVCMOS15;  
5 NET "F" LOC = AF24 | IOSTANDARD = LVCMOS33;  
6 NET "LED[0]" LOC = W23 | IOSTANDARD = LVCMOS33;  
7 NET "LED[1]" LOC = AB26 | IOSTANDARD = LVCMOS33;  
8 NET "LED[2]" LOC = Y25 | IOSTANDARD = LVCMOS33;  
9 NET "LED[3]" LOC = AA23 | IOSTANDARD = LVCMOS33;  
10 NET "LED[4]" LOC = Y23 | IOSTANDARD = LVCMOS33;  
11 NET "LED[5]" LOC = Y22 | IOSTANDARD = LVCMOS33;  
12 NET "LED[6]" LOC = AE21 | IOSTANDARD = LVCMOS33;
```

(b) 将 LampCtrl.v 代码中计数器位数改成 28 位，修改代码如下：

```

22 module LampCtrl(
23     input wire clk,
24     input wire S1,
25     input wire S2,
26     input wire S3,
27     output wire [6:0] LED,
28     output wire F
29 );
30
31 parameter C_NUM = 28;
32 parameter C_MAX = 28'hFFFFFFF;
33
34
35 reg [C_NUM-1:0] count;
36 wire [C_NUM-1:0] c_next;
37 wire w;
38
39 initial begin
40     count = C_MAX;
41 end
42
43 assign w=S1||S2||S3;
44 assign F = (count<C_MAX)?1'b1:1'b0;
45
46 always@(posedge clk)
47 begin
48     if(w==1'b1)
49         count=0;
50     else if(count< C_MAX)
51         count=c_next;
52 end
53
54 ▶ assign c_next=count+8'b1;|
55
56 assign LED = 7'b0000000;
57
58 endmodule

```

3.2.8 下载到 SWORD 板

- (a) 类似 3.1.9 中的步骤：Synthesize - XST; Implement design; Generate Programming File;
- (b) 将生成 bit 文件下载到实验板

- i. 在 Design 的 View 窗口中选择 Implementation
 - ii. 在 Sources 窗口中选择 LampCtrl.sch; 在 Processes 窗口中, 用鼠标点开 Config Target Device, 双击 Manage Configuration Project(iMPACT) 选项, 出现 IMPACT 窗口
 - iii. 双击 Boundary Scan 弹出下载编辑窗口
 - iv. 鼠标右键选择 Initialize Chain, 系统自动查找已连接在电脑上的开发平台 JTAG 下载链
 - v. 接下来出现 Assign Configuration Files 对话框。这时从文件列表中选择 Lampctrl.bit 文件, 将会为 JTAG chain 上的 xc7k160t 设备指定配置文件; 在弹出的 Attach SPI or PRI PROM 对话框弹出, 点击 NO 按钮; 在弹出的 Device Programming Property 对话框, 选择 OK 按钮即可
- (c) 在实验板上运行
- (d) 根据 I/O 约束定义, 在板上用拨盘开关模拟按键操作, 查看灯的变化是否正确, 验证设计是否成功
- 如下图所示, 第一张图是打开开关灯亮起, 第二张图是关闭开关后灯仍然亮起, 第三张图是一段时间后灯熄灭

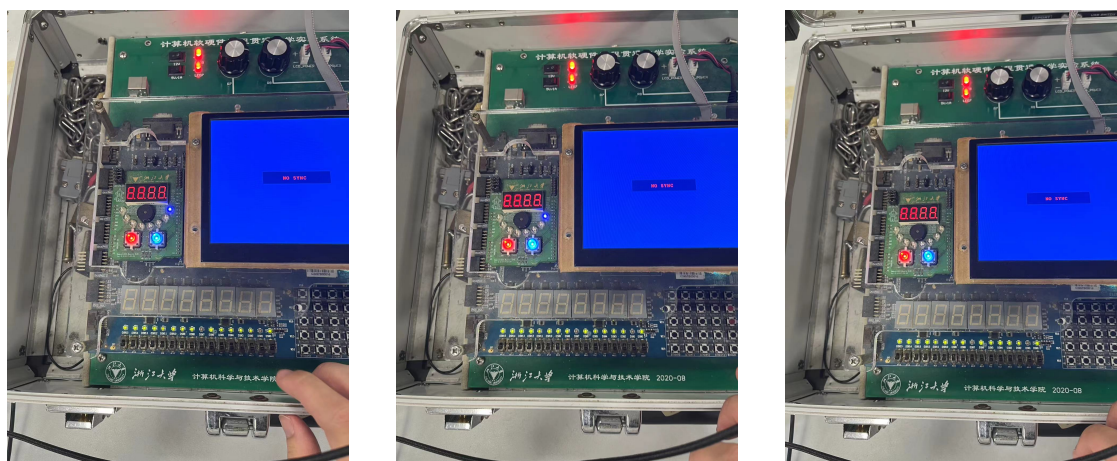


图 11: 验证实验结果

四、实验结果分析

4.1 分析硬件描述代码：

查看输入电路的硬件描述代码时，我们可以发现可以里面的语句和电路图都是相对应的。比如 OR4(四输入或门), AND3(三输入与门), INV(invert er, 反相器) 等。这些语句后面的相应代码表示各个门的输入输出。因此生成的 Verilog 代码与预期相符合。

4.2 分析仿真结果

两个仿真激励输入的代码都是让 (S1, S2, S3) 从 (0, 0, 0) 变换到 (1, 1, 1)，也就是逐次遍历真值表。第一个仿真激励输入的代码是依次表达所有的情况，而第二个仿真激励输入的代码是用循环来实现的。仿真波形图中可以看到对应输出结果与预期相同。

4.3 分析开发板结果

在第一部分的实验中，开发板可以做到无延时控制灯的开关，而且开发板上 AA10、AB10、AA13 三个开关对应 S1、S2、S3. 通过操作这三个开关得到的实验结果与预期相同，当有奇数个开关闭合时灯亮起，当有偶数个开关闭合时灯熄灭。

在第二部分的实验中，先开启一个开关（三个中的任意一个），此时灯立刻亮起，随后关闭该开关，等待一段时间（大约两秒）后灯会熄灭，即成功实现了延时关闭的功能。

因此开发板结果也符合预期

4.4 分析两种设计的差别

根据逻辑电路的不同特点，数字电路可以分为：组合逻辑和时序逻辑。而我们本次实验正是分别实现了这两种逻辑电路。

1. 组合逻辑组合逻辑的特点是任意时刻的输出仅仅取决于该时刻的输入，与电路原本的状态无关，逻辑中不牵涉跳变沿信号的处理，组合逻辑的 verilog 描述方式有两种：

- (a) `always @`（电平敏感信号列表）或者 `always @ * always` 模块的敏感列表为所有判断条件信号和输入信号，但一定要注意敏感列表的完整性。在 `always` 模块中可以使用 `if`、`case` 和 `for` 等各种 RTL 关键字结构。由于赋值语句有阻塞赋值和非阻塞赋值两类，建议读者使用阻塞赋值语句“`=`”。`always` 模块中的信号必须定义为 `reg` 型，不过最终的实现结果中并没有寄存器。这是由于在组合逻辑电路描述中，将信号定义为 `reg` 型，只是为了满足语法要求
 - (b) `assign` 描述的赋值语句信号只能被定义为 `wire` 型
2. 时序逻辑
- 时序逻辑的特点为任意时刻的输出不仅取决于该时刻的输入，而且还和电路原来的状态有关。电路里面有存储元件（各类触发器，在 FPGA 芯片结构中只有 D 触发器）用于记忆信息，从电路行为上讲，不管输入如何变化，仅当时钟的沿（上升沿或下降沿）到达时，才有可能使输出发生变化；与组合逻辑不同的是：
- (a) 在描述时序电路的 `always` 块中的 `reg` 型信号都会被综合成寄存器
 - (b) 时序逻辑中推荐使用非阻塞赋值“`<=`”
 - (c) 时序逻辑的敏感信号列表只需要加入所用的时钟触发沿即可，其余所有的输入和条件判断信号都不用加入，这是因为时序逻辑是通过时钟信号的跳变沿来控制的。

五、讨论与心得

这是第一次上板的实验课，也是我们第一次使用 ISE 平台以及 Verilog 硬件描述语言的实验。由于前三次实验的“惨不忍睹”（几乎都比较晚通过验收），这次我下定决心要提前做好准备。但是结果还是不尽如人意，我安装 ISE 的过程十分的曲折。我从实验前的一个周末就准备着手安装 ISE, 首先是博客中的命令行 `wget` 获取安装包的命令行，`get` 的是官网下载的网页内容，而非安装包。于是我前往 AMD 官网进行下载，并按照步骤依次进行。但是我遇到了一个致命问题，就是运行 `./xetup` 的时候一直出现错误：`cannot connect to X server`。我上网搜寻尝试了各种方法一直不能解决，询问助教哥哥也未能完全解决。因此我一直拖到了实验当天还没能下载 ISE，最后在实验当天下午我重装 WSL，这个问题居然神奇般的消失了，在成功安装 ISE 的

那一瞬间我松了一口气/(T o T)/。之后的实验过程都比较顺利，但是还是没能较早完成验收。如今解决了 ISE 安装的问题，下一次实验我一定会好好预习，提前完成代码以及虚拟仿真，争取下次实验课直接上板验证结果！