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：龙永奇

学　　院：计算机科学与技术学院

系：本系

专　　业：计算机科学与技术

学　　号：3220105907

指导教师：董亚波

2023 年 10 月 24 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：龙永奇 专业：计算机科学与技术 学号：3220105907

同组学生姓名：周楠 指导老师：董亚波

实验地点：东 4-509 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. 问题 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。

解决方法：

(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 S1,S2,S3，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，如下表：

S_3	S_2	S_1	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(d) 根据真值表分析输入输出关系 $F = S_1S_2S_3 + S_1S_2S_3 + S_1S_2S_3 + S_1S_2S_3$

可以画出电路图：

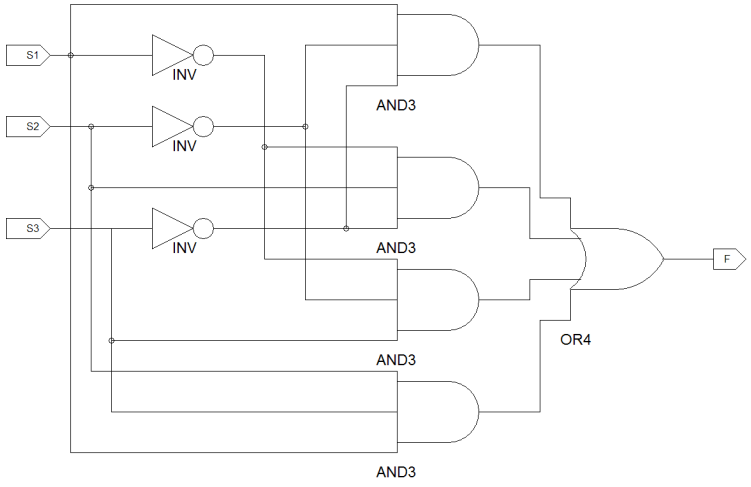


图 2.1 逻辑电路图

2. 问题 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。

解决方法：

(a) 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：

```
`timescale 1ns / 1ps module LampCtrl(
                                input wire clk,
                                input wire S1,
                                input wire S2,
                                input wire S3,
                                output wire F);
parameter C_NUM = 8;
parameter C_MAX = 8'hFF;
reg [C_NUM - 1:0] count;
wire [C_NUM - 1:0] c_next;
wire w;
initial begin // 初始化
      count = C_MAX;
end
      // button pressed
      assign w = S1 || S2 || S3;
// lamp logic
assign F = (count < C_MAX) ? 1'b1 : 1'b0;
// count logic
always @(posedge clk)
      begin
      if (w == 1'b1)
```

```
            count = 0;
else if (count < C_MAX)
      count = c_next;
end
      // next logic
      assign c_next = count + 8'b1;
endmodule
```

三、实验过程和数据记录

3.1 以图形方式输入逻辑功能描述

1. 建立楼道控制项目

(a)在 ISE 中点击 File 选项卡，点击 New Project

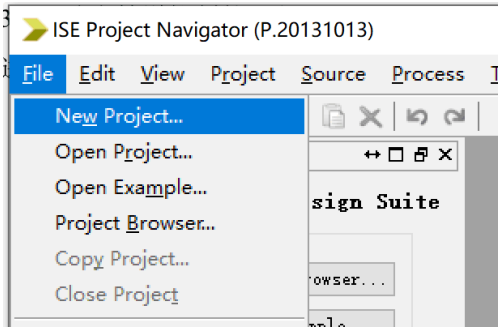


图 3.1 建立新项目

(b)设置项目属性在

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

(c)确认后一直点击 Next 直到创建完成

2. 建立原理图

(a)在 Sources 窗口中右键选择 New Sources

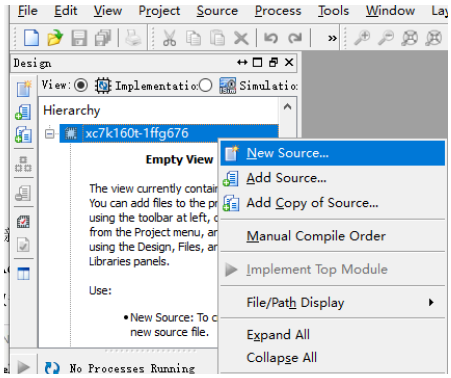


图 3.2 建立原理图

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

(b)点击 Next 直到创建完成，在 Sources 窗口双击新建文件进入编辑界面

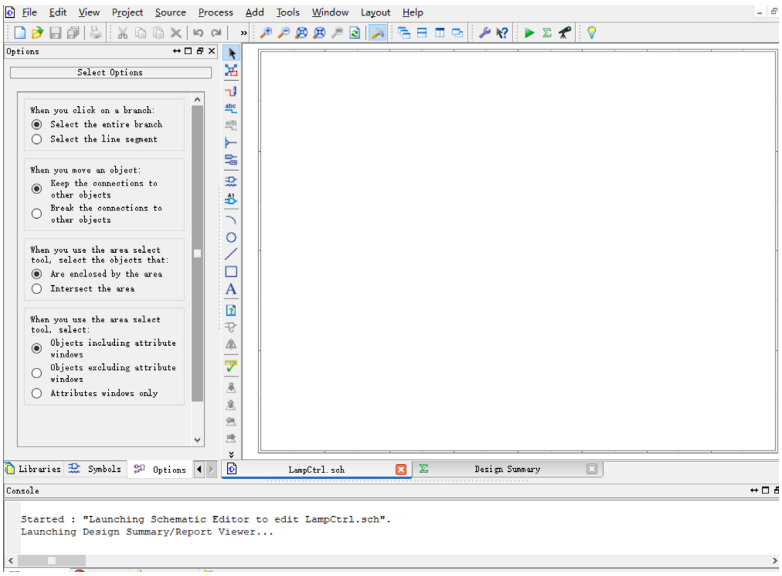


图 3.3 编辑窗口

3. 绘制楼道灯控制电路图

(a)使用 Symbols 和 Schematic Editor 输入原理图如下：

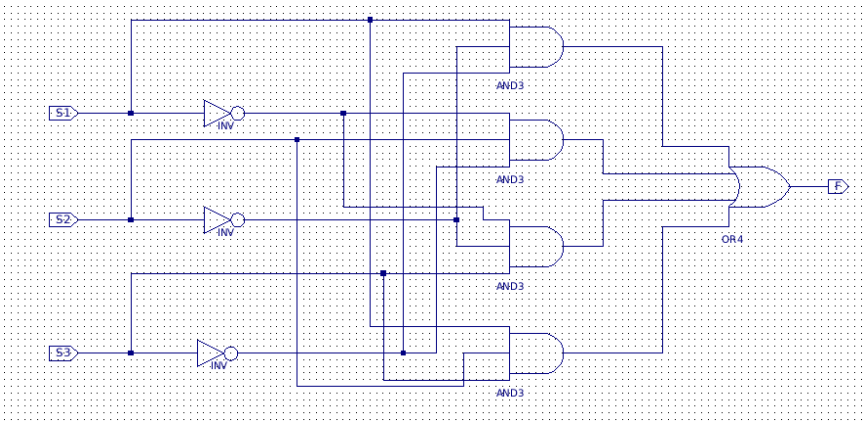


图 3.4 原理图

4. 查看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，如下图：

```
1  //////////////////////////////////////////////////////////////////////////////
2  // Copyright (c) 1995-2013 Xilinx, Inc.  All rights reserved.
3  //////////////////////////////////////////////////////////////////////////////
4  //   ____  ____
5  //  /   /\/   /
6  // /___/  \  /    Vendor: Xilinx
7  // \   \   \/     Version : 14.7
8  //  \   \         Application : sch2hdl
9  //  /   /         Filename : LampCtrl.vf
10 // /___/   /\     Timestamp : 10/12/2023 14:43:26
11 // \   \  /  \
12 //  \___\/\___\
13 //
14 //Command: sch2hdl -intstyle ise -family kintex7 -verilog /root/Xilinx_ISE_DS_Lin_14.7_1015_1/LampCtrl_sch/LampCtrl.vf -w /root/Xil
15 //Design Name: LampCtrl
16 //Device: kintex7
17 //Purpose:
18 //    This verilog netlist is translated from an ECS schematic.It can be
19 //    synthesized and simulated, but it should not be modified.
20 //
21 `timescale 1ns / 1ps
22
23 module LampCtrl(S1,
24                 S2,
25                 S3,
26                 F,
27                 LED);
28
29     input S1;
30     input S2;
31     input S3;
32    output F;
33    output [6:0] LED;
34
```

图 3.5 硬件描述代码

硬件描述代码如下：

```
`timescale 1ns / 1ps

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

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

```
wire XLXN_19;
wire XLXN_22;
wire XLXN_24;
wire XLXN_30;
wire XLXN_35;
wire XLXN_36;
wire XLXN_37;

INV XLXI_7(.I(S1),
                .O(XLXN_24));
INV XLXI_8(.I(S2),
                .O(XLXN_22));
INV XLXI_10(.I(S3),
                 .O(XLXN_19));
AND3 XLXI_14(.I0(XLXN_19),
                  .I1(XLXN_22),
                  .I2(S1),
                  .O(XLXN_30));
AND3 XLXI_17(.I0(XLXN_19),
                  .I1(S2),
                  .I2(XLXN_24),
                  .O(XLXN_35));
AND3 XLXI_18(.I0(S3),
                  .I1(XLXN_22),
                  .I2(XLXN_24),
                  .O(XLXN_36));
AND3 XLXI_19(.I0(S3),
                  .I1(S2),
                  .I2(S1),
                  .O(XLXN_37));
OR4 XLXI_20(.I0(XLXN_37),
                .I1(XLXN_36),
                .I2(XLXN_35),
                .I3(XLXN_30),
                .O(F));
GND XLXI_40(.G(LED[6]));
GND XLXI_41(.G(LED[5]));
GND XLXI_42(.G(LED[4]));
GND XLXI_43(.G(LED[3]));
GND XLXI_44(.G(LED[2]));
GND XLXI_45(.G(LED[1]));
GND XLXI_46(.G(LED[0]));
endmodule
```

5. **建立基准测试波形文件**

(a)在 New Source 中创建文件名为 LampCtrl_sim，勾选 Add to Project

(b)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

6. **仿真激励输入**

(a)在源文件中添加以下代码：

```
`timescale 1ns / 1ps

    module
    LampCtrl_LampCtrl_sch_tb();

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

// Output
wire F;

// Bidirs

// Instantiate the UUT
LampCtrl UUT(
        .S1(S1),
        .S2(S2),
        .S3(S3),
        .F(F));
// Initialize Inputs
// `ifdef auto_init
initial begin
    S1 = 0;
    S2 = 0;
    S3 = 0;
    # 50 S1 = 1;
    # 50 S1 = 0;
    S2 = 1;
    # 50 S1 = 1;
    # 50 S1 = 0;
    S2 = 0;
    S3 = 1;
    # 50 S1 = 1;
```

```
    # 50 S1 = 0;
    S2 = 1;
    # 50 S1 = 1;
    # 50 S1 = 0;
    S2 = 0;
    S3 = 0;
end
    // `endif
    endmodule
```

(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，可得到如下波形图，点击调节缩放按钮直至出现完整波形：

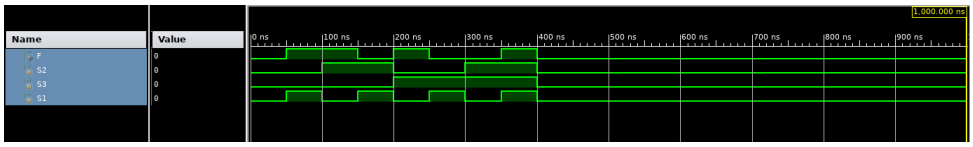


图 3.6 仿真波形图 1

(c)将源文件中代码替换为以下代码：

```
`timescale 1ns / 1ps
module LampCtrl_LampCtrl_sch_tb();
// Inputs
reg S1;
reg S2;
reg S3;
wire F;
lamp_ctrl UUT(
        .S1(S1),
        .S2(S2),
        .S3(S3),
        .F(F));
// Initialize Inputs
//   `ifdef auto_init
integer i;
initial begin
    for (i = 0; i <= 8; i = i + 1)
        begin{S3, S2, S1} <= i;
    # 50;
end
    end
        endmodule
```

仿真结果如下图：

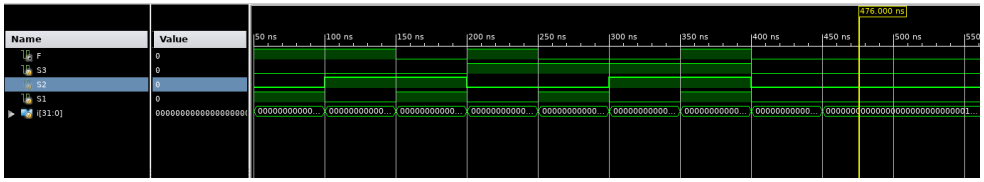


图 3.7 仿真结果波形图 2

7. 建立用户时许约束并为模块端口指定引脚分配

(a)XC7K160T 芯片引脚如下：

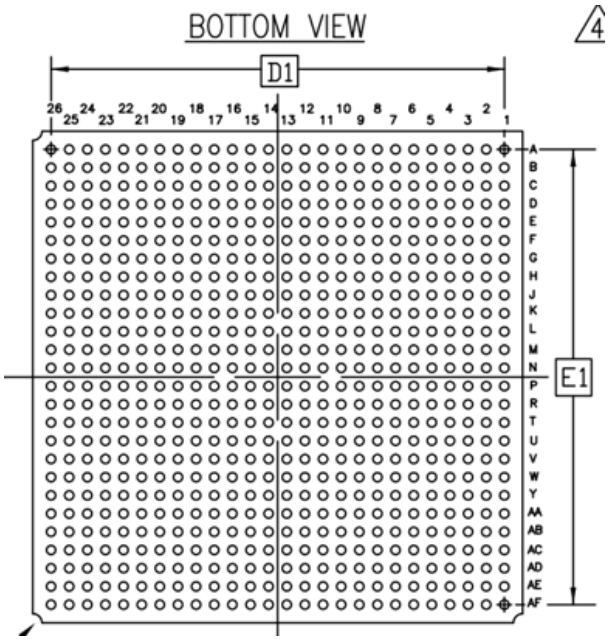


图 3.8 XC7K160T 芯片引脚

(b)创建文件名为 K7，类型为 Implementation Constraints File 的文件，勾选 Add to Project，建立时序约束文件

(c)在 K7.ucf 中输入以下代码：

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

(d)在 Lamp_Ctrl.sch 中绘制总线，如图：

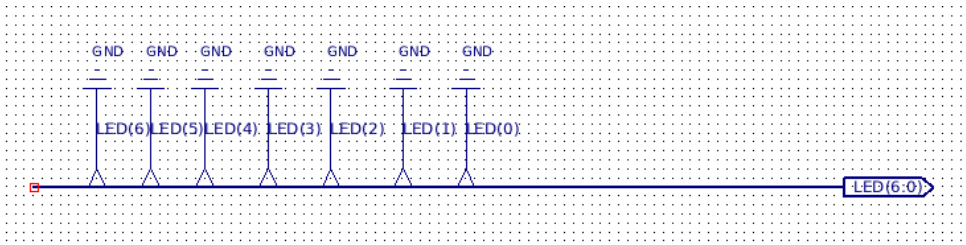


图 3.9 总线电路图

8. 设计实现

在 Degign 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

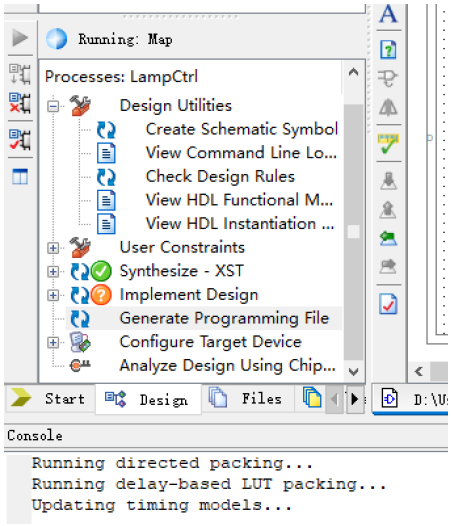
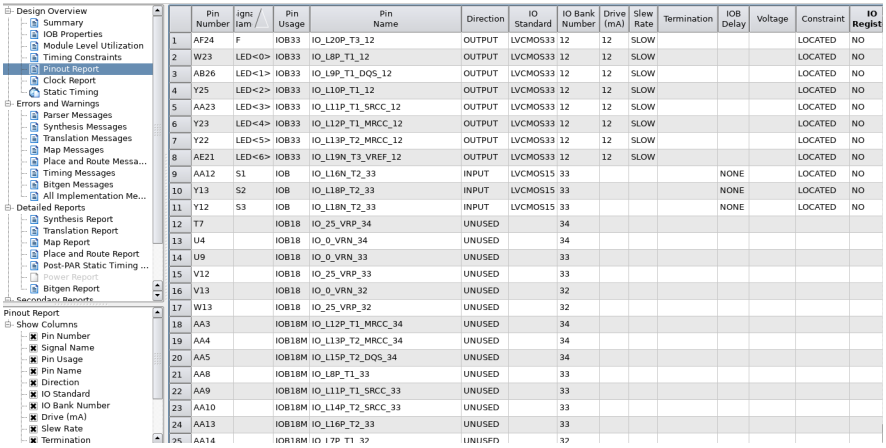


图 3.10 bit 文件生成

9. 检查约束结果

(a)在 Design Summary 文档中有如下结果：



	Pin Number	ign / lam /	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number	Drive (mA)	Slew Rate	Termination	IOB Delay	Voltage	Constraint	IO Regist
1	AF24	F	IOB33	IO_L20P_T3_12	OUTPUT	LVCMOS33	12	12	SLOW				LOCATED	NO
2	W23	LED<0>	IOB33	IO_L8P_T1_12	OUTPUT	LVCMOS33	12	12	SLOW				LOCATED	NO
3	AB26	LED<1>	IOB33	IO_L9P_T1_DQS_12	OUTPUT	LVCMOS33	12	12	SLOW				LOCATED	NO
4	Y25	LED<2>	IOB33	IO_L10P_T1_12	OUTPUT	LVCMOS33	12	12	SLOW				LOCATED	NO
5	AA23	LED<3>	IOB33	IO_L11P_T1_SRCC_12	OUTPUT	LVCMOS33	12	12	SLOW				LOCATED	NO
6	Y23	LED<4>	IOB33	IO_L12P_T1_MRCC_12	OUTPUT	LVCMOS33	12	12	SLOW				LOCATED	NO
7	Y22	LED<5>	IOB33	IO_L13P_T2_MRCC_12	OUTPUT	LVCMOS33	12	12	SLOW				LOCATED	NO
8	AE21	LED<6>	IOB33	IO_L19N_T3_VREF_12	OUTPUT	LVCMOS33	12	12	SLOW				LOCATED	NO
9	AA12	S1	IOB	IO_L16N_T2_33	INPUT	LVCMOS15	33				NONE		LOCATED	NO
10	Y13	S2	IOB	IO_L18P_T2_33	INPUT	LVCMOS15	33				NONE		LOCATED	NO
11	Y12	S3	IOB	IO_L18N_T2_33	INPUT	LVCMOS15	33				NONE		LOCATED	NO
12	T7		IOB18	IO_25_VRP_34	UNUSED		34							
13	U4		IOB18	IO_0_VRN_34	UNUSED		34							
14	U9		IOB18	IO_0_VRN_33	UNUSED		33							
15	V12		IOB18	IO_25_VRP_33	UNUSED		33							
16	V13		IOB18	IO_0_VRN_32	UNUSED		32							
17	W13		IOB18	IO_25_VRP_32	UNUSED		32							
18	AA3		IOB18M	IO_L12P_T1_MRCC_34	UNUSED		34							
19	AA4		IOB18M	IO_L13P_T2_MRCC_34	UNUSED		34							
20	AA5		IOB18M	IO_L15P_T2_DQS_34	UNUSED		34							
21	AA8		IOB18M	IO_L8P_T1_33	UNUSED		33							
22	AA9		IOB18M	IO_L11P_T1_SRCC_33	UNUSED		33							
23	AA10		IOB18M	IO_L14P_T2_SRCC_33	UNUSED		33							
24	AA13		IOB18M	IO_L16P_T2_33	UNUSED		33							

图 3.11 设计摘要

(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，出现如下 IMPACT 窗口：

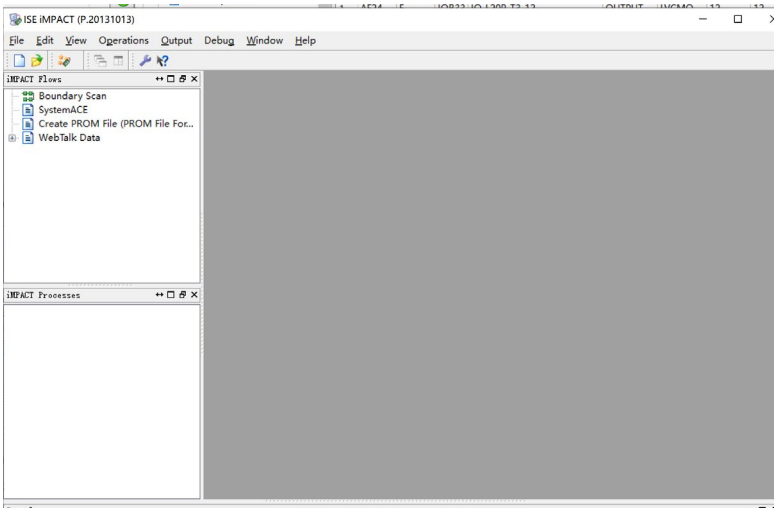


图 3.12 IMPACT 窗口

(c)双击 Boundary Scan，选择 Initialize Chain，系统自动查找已连接在电脑上的开发平台 JTAG 下载链接，出现 Assign Configuration File 对话框，选择 LampCtrl.bit 文件，将为 JTAG Chain 上的 XC7K160T 设备配置指定文件，在弹出的 Attach SPI or PRI PROM 窗口点击 NO，之后在 Device Programming Property 窗口点击 OK，右键点击 XC7K160T 设备图标，选择 Program 即可下载编程：

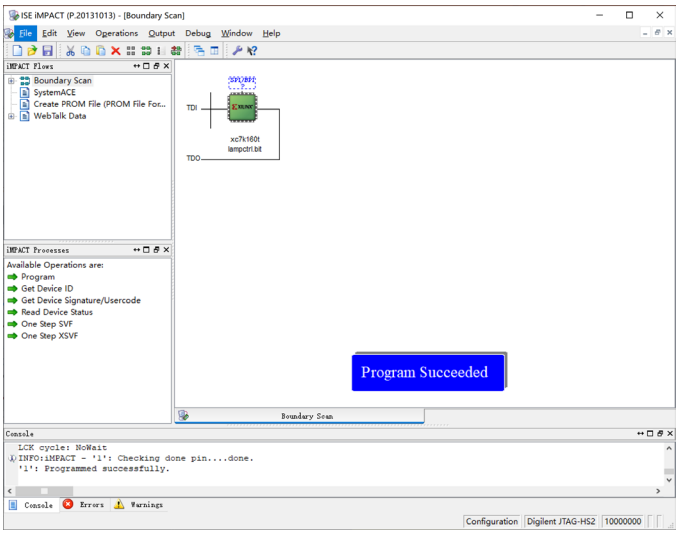


图 3.13 Program

(d)在实验板中检查是否满足要求：

图 3.14 000

图 3.15 001

图 3.16 010

图 3.17 011

图 3.18 100

图 3.19 101

图 3.20 110

图 3.21 111

3.2 Verilog 代码输入逻辑功能描述

1. **建立楼道控制项目**

Project Name: LampCtrl_HDL

Top-Level Source Type: HDL

Family: Kintex7

Device: XC7K160T

Package: FFG676

Speed: -1

确认后点击 Next 直至创建完成

2. **创建 Verilog 输入源文件**

(a)创建 New Sources，文件名为 LampCtrl，勾选 Add to Project

(b)点击 Next 直至创建完成

3. **输入楼道灯控逻辑电路 Verilog HDL 代码**

(a)输入以下代码：

```
module LampCtrl(
    input wire clk,
    input wire S1,
    input wire S2,
    input wire S3,
    output wire F);
    parameter C_NUM = 8;
    parameter C_MAX = 8'hFF;
    reg [C_NUM - 1:0] count;
    wire [C_NUM - 1:0] c_next;
    wire w;
    initial begin // ini
        count = C_MAX;
    end
    // button pressed
    assign w = S1 || S2 || S3;
    // lamp logic
    assign F = (count < C_MAX) ? 1'b1 : 1'b0;
    // count logic
always @(posedge clk)
    begin
    if (w == 1'b1)
        count = 0;
    else if (count < C_MAX)
    count = c_next;
end
    // next logic
    assign c_next = count + 8'b1;
endmodule
```

(b)在 Synthesize-XST 菜单中选择 Check Syntax 检查代码语法规则，排除输入错误：

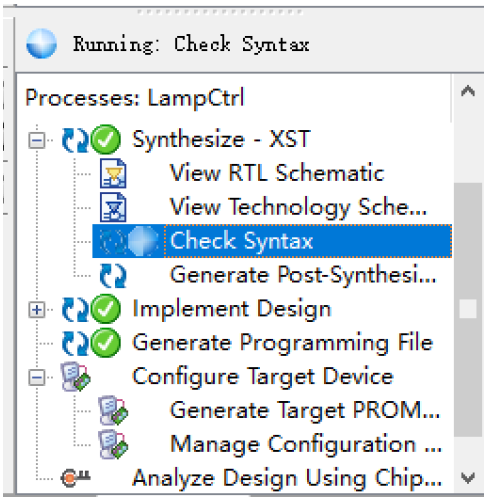


图 3.22 检查语法错误

(c)延时时间修改

仿真时设定：

parameter C_NUM = 8;

parameter C_MAX = 8’hFF;

$\Delta t = (2^8 - 1) \times 20ns = 5100ns$

下载运行时我们设定

parameter C_NUM = 28;

parameter C_MAX = 28’hFFFFFFF;

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

4. 楼道控制电路代码综合

(a)选择文件 LampCtrl.v，在 Processes 窗口运行 Synthesis XST → View RTL Schematic

(b)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 代码

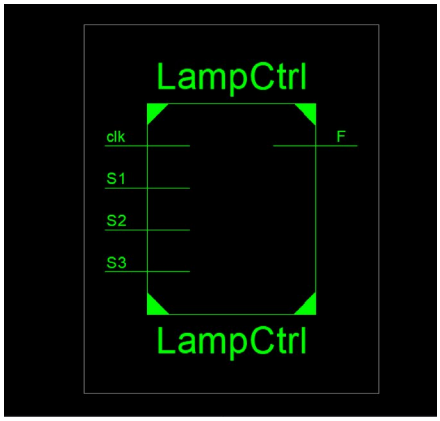


图 3.23 LampCtrl 模块

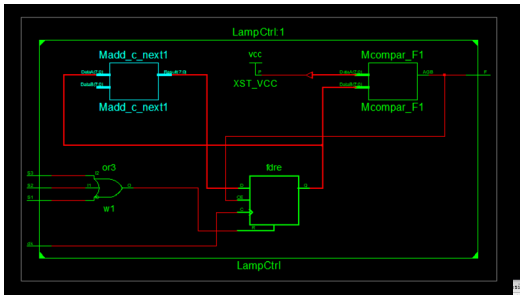


图 3.24 详细电路

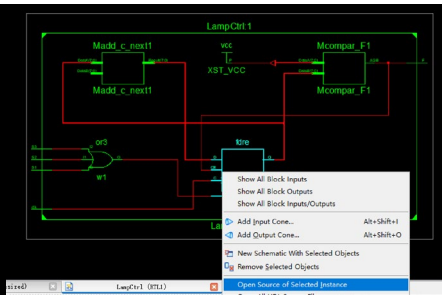


图 3.25 定位 Verilog 代码

5. 建立基准测试波形文件

(a)建立文件名为 LampCtrl_sim，源类型为 Verilog Test Fixture 文件，勾选 Add to Project

(b)点击 Finish 直至进入 LampCtrl_sim.v 窗口

6. 仿真激励输入波形

(a)为便于仿真，LampCtrl.v 代码中计数器位数采用 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;
    # 600 S1 = 1;
    # 20 S1 = 0;
    # 6000 S2 = 1;
    # 20 S2 = 0;
    # 6000 S3 = 1;
    # 20 S3 = 0;
end
always begin
    # 10 clk = 0;
    # 10 clk = 1;
end
endmodule
```

(b)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

(c)菜单栏把仿真大小改为 21us，点击“Run for the time specified on the toolbar”进行仿真，点击“Zoom to full view”显示完整波形，最后点击“zoom in”放大波形。如图所示：

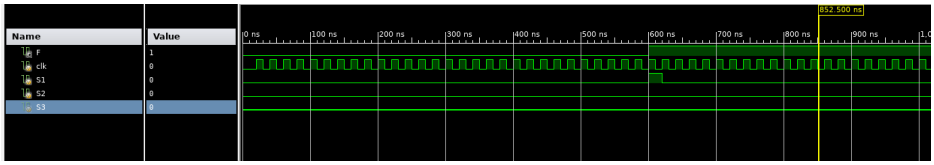


图 3.27 仿真波形

7. 建立用户时序约束并为模块端口指定引脚分配

(a)建立引脚约束文件 K7.ucf，输入代码：

```
NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18;
NET "S1" LOC = AA12 | IOSTANDARD = LVCMOS15;
NET "S2" LOC = Y13 | IOSTANDARD = LVCMOS15;
NET "S3" LOC = Y12 | IOSTANDARD = LVCMOS15;
NET "F" LOC = AF24 | IOSTANDARD = LVCMOS33;
NET "LED[0]" LOC = W23 | IOSTANDARD = LVCMOS33;
NET "LED[1]" LOC = AB26 | IOSTANDARD = LVCMOS33;
NET "LED[2]" LOC = Y25 | IOSTANDARD = LVCMOS33;
NET "LED[3]" LOC = AA23 | IOSTANDARD = LVCMOS33;
```

```
NET "LED[4]" LOC = Y23 | IOSTANDARD = LVCMOS33;
NET "LED[5]" LOC = Y22 | IOSTANDARD = LVCMOS33;
NET "LED[6]" LOC = AE21 | IOSTANDARD = LVCMOS33;
```

(b)将 LampCtrl.v 代码中计数器位数改为 28 位，修改或增加以下部分代码：

```
module LampCtrl(
    input wire clk,
    …… output wire [6 : 0] LED,
    output wire F);
parameter C_NUM = 28;
parameter C_MAX = 28'hFFFFFFF;
…… assign LED = 7'b0000000; // 需增加对应 output 端口
```

8. **下载到 SWORD 板**

(a)步骤与 3.1.9 相同

(b)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，验证结果如下：

图 3.28 开开关灯亮

图 3.29 关开关未灭

图 3.30 延迟熄灭

开关亮灭符合设计逻辑要求

四、实验结果分析

本次实验从楼道灯控制这一问题出发，依次完成硬件电路图设计、仿真模拟、开发板测试三个步骤

1. **电路图绘制与硬件描述代码**

本次实验中所使用的电路图来自课程 PPT，通过电路图生成硬件描述代码后，将代码与图中各组件依次对应，使用了多个逻辑门和反相器（INV）来实现开关信号和 LED 输出之间的逻辑关系：

(a) INV 用于对输入信号取反

(b) AND3 用于实现三个输入信号的与逻辑运算

(c) OR4 用于实现四个输入信号的或逻辑运算。

2. **仿真模拟**

本次实验共进行了两次仿真激励输入，其功能相同，均让 S1,S2,S3 遍历真值表，得到 F 的输出结果波形，两者区别在于前者代码依次输入各情况：

```
    S1 = 0;
    S2 = 0;
    S3 = 0;
# 50 S1 = 1;
# 50 S1 = 0;
    S2 = 1;
# 50 S1 = 1;
# 50 S1 = 0;
    S2 = 0;
    S3 = 1;
# 50 S1 = 1;
# 50 S1 = 0;
    S2 = 1;
# 50 S1 = 1;
# 50 S1 = 0;
    S2 = 0;
    S3 = 0;
```

后者则使用 for 循环输入：

```
for (i = 0; i <= 8; i = i + 1)
   begin{S3, S2, S1} <= i;
   # 50;
```

3. **开发板验证**

由于本组使用的开发板 AB10 开关损坏，因此将 AA12、Y13、Y12 分别对应 S1、S2、S3。实验一为立即熄灭，实验二为延时熄灭。两者均在偶数个开关闭合时熄灭，奇数个开关闭合时亮起

4. **图形方式输入和 Verilog 代码输入逻辑描述区别**

(a) 图像方式输入

手工绘制逻辑电路图后生成硬件描述代码，使用了门级逻辑（AND、OR、INV 等）来直接实现逻辑功能，涵盖了更底层的硬件细节。通过具体的线网（wire）和门（AND3、OR4 等）连接来实现逻辑，包括了输出到 LED 的逻辑，以及 LED 和中间信号的命名。

此方式包含了更多的硬件细节，代码相对较长且复杂。

(b) Verilog 代码输入

使用了较高级别的 Verilog 描述，通过 always@(posedge clk)块处理时钟信号,并使用 assign 语句进行逻辑操作。代码结构更加简洁，逻辑更加直观。通过使用参数来定义常数，代码更具通用性。没有使用门级逻辑或硬件描述，因此更容易理解和维护。

五、讨论与心得

本次实验为首次使用实验板的实验课，学长建议在实验前安装好 WSLg 以及 ISE，并按照 PPT 将实验内容大体完成，在实验课程中仅需上板实验即可，因此节省了大量的时间。由于对于子系统安装以及 Linux 系统操作的不熟练，因此第一次实验还是花费了大量时间,但同时也让我对 Linux 系统、WSL 子系统、Verilog 语言以及代码与硬件之间的关系有了初步的了解，期望在下次实验课中继续努力，同时也感谢老师、助教以及同组同学的帮助!