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 31 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：变量译码器设计与应用

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：孟德立 指导老师：董亚波

实验地点：东 4-509 实验日期：2023 年 10 月 19 日

一、实验目的和要求

1. 掌握变量译码器的的逻辑构成和逻辑功能
2. 用变量译码器实现组合函数
3. 采用原理图设计电路模块
4. 进一步熟悉 ISE 平台及下载实验平台物理验证

二、实验内容和原理

内容：

1. 原理图设计实现 74LS138 译码器模块
2. 用 74LS138 译码器实现楼道灯控制，具体功能和操作同实验四任务 1

原理：

1. 译码器是将一种输入编码转换成另一种编码的电路，即将给定的代码进行“翻译”并转换成指定的状态或输出信号（脉冲或电平）
2. 译码可分为：变量译码、显示译码
   (a) 变量译码一般是将一种较少位输入变为较多位输出的器件，如 2n 译码和 8421BCD 码译码
   (b) 显示译码主要进行 2 进制数显示成 10 进制或 16 进制数的转换，可分为驱动 LED 和 LCD 两类
3. 变量译码器
   (a) 变量译码器—74LS138
   变量译码器是一个将 n 个输入变为 2n 个最小项输出的多输出端的

组合逻辑电路。n 通常在 26　4 之间。

图 2.1 变量译码器

74LS138 变量译码器功能表和引脚如下：

输入		译码器输出（低电平有效）							
使能	变量								
$GG_{2A}G_{2B}$	ABC	Y_0	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6	Y_7
×11	×××	1	1	1	1	1	1	1	1
0××	×××	1	1	1	1	1	1	1	1
100	000	**0**	1	1	1	1	1	1	1
100	001	1	**0**	1	1	1	1	1	1
100	010	1	1	**0**	1	1	1	1	1
100	011	1	1	1	**0**	1	1	1	1
100	100	1	1	1	1	**0**	1	1	1
100	101	1	1	1	1	1	**0**	1	1
100	110	1	1	1	1	1	1	**0**	1
100	111	1	1	1	1	1	1	1	**0**

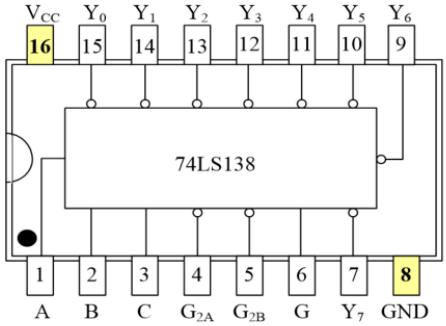

图 2.2 译码器功能表和引脚

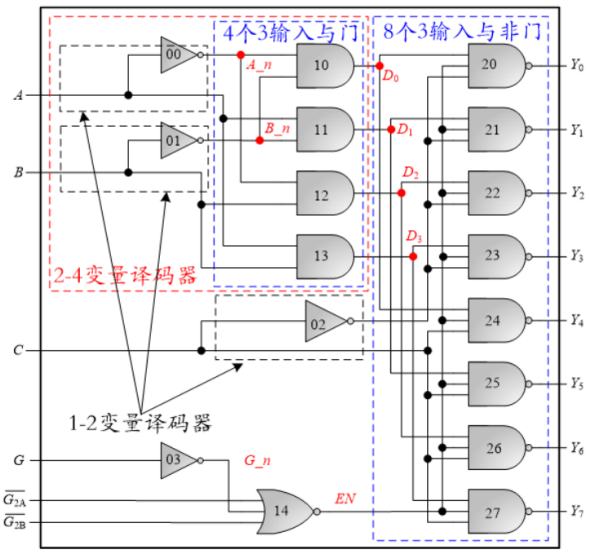

图 2.3 74LS138 门电路图

门电路图对应的 Verilog 代码如下：

```
module decoder_3_8(A, B, C, G, G2A,G2B, Y);
input wire A, B, C, G, G2A, G2B;
output wire [7:0] Y;
```

```
not     node_0_0(A_n, A),
    node_0_1(B_n, B),
    node_0_2(C_n, C),
    node_0_3(G_n, G);
and node_1_0(D0, B_n, A_n),
    node_1_1(D1, B_n, A  ),
    node_1_2(D2, B,   A_n),
    node_1_3(D3, B,   A  );
nor node_1_4(EN, G_n, G2A, G2B);
nand    node_2_0(Y[0], EN, D0, C_n),
    node_2_1(Y[1], EN, D1, C_n),
    node_2_2(Y[2], EN, D2, C_n),
    node_2_3(Y[3], EN, D3, C_n),
    node_2_4(Y[4], EN, D0, C  ),
    node_2_5(Y[5], EN, D1, C  ),
    node_2_6(Y[6], EN, D2, C  ),
    node_2_7(Y[7], EN, D3, C  );
endmodule
```

(b) 变量译码器—74LS139

74LS139 功能表和引脚如下：

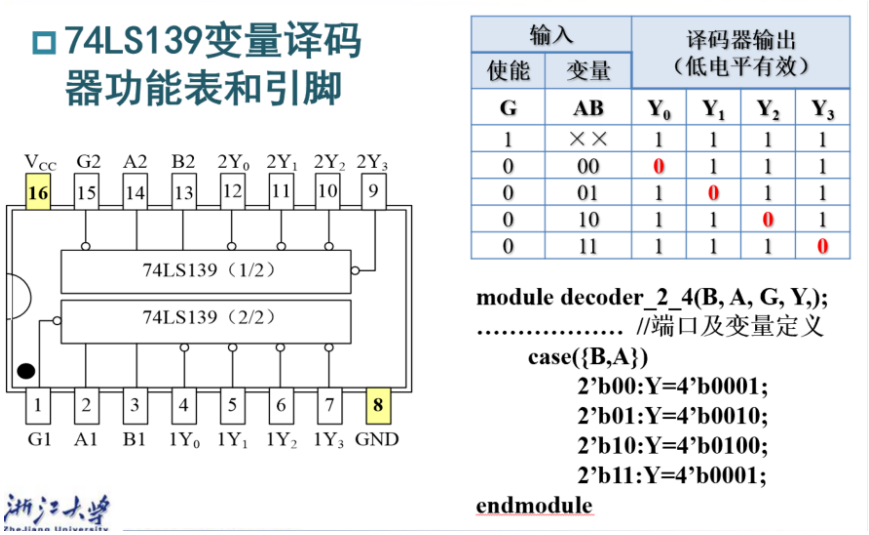

图 2.4 74LS139 译码器功能表和引脚

4. 用变量译码器实现楼道灯控制函数

变量译码器的输出对应所有输入变量的最小项组合，如果将函数转换成最小项和的形式，则可以用变量译码器实现函数的组合电路： F = S3`S2`S1 + S3`S2 S1` + S3 S2`S1` + S3 S2 S1

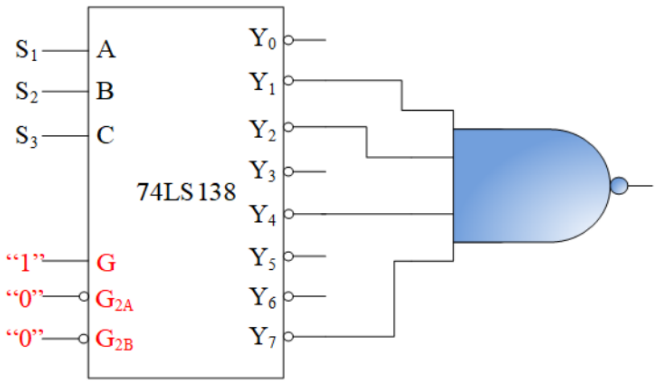

图 2.5 楼道灯控制实现电路图

三、实验过程和数据记录

3.1 原理图设计实现 74LS138 译码器模块

1. 设计实现 74LS138

(a)在 ISE 中点击 File 选项卡，点击 New Project，工程名为 D_74LS138_SCH

(b)在 Sources 窗口中右键选择 New Sources 新建源文件向导中选择源文件类型为 Schematic,输入文件名 D_74LS138,勾选 Add to Project

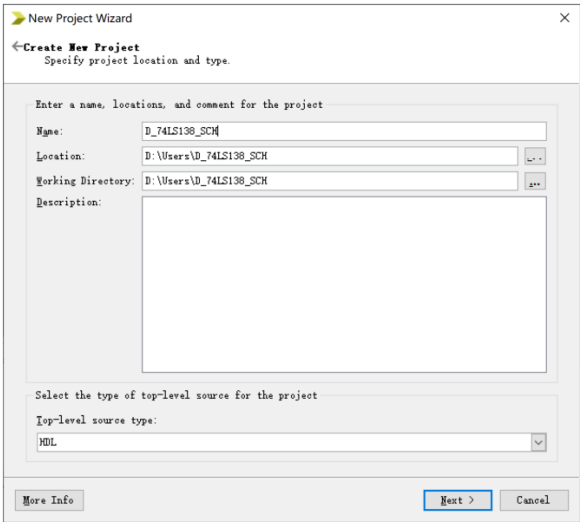

图 3.1 新建工程

(c)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

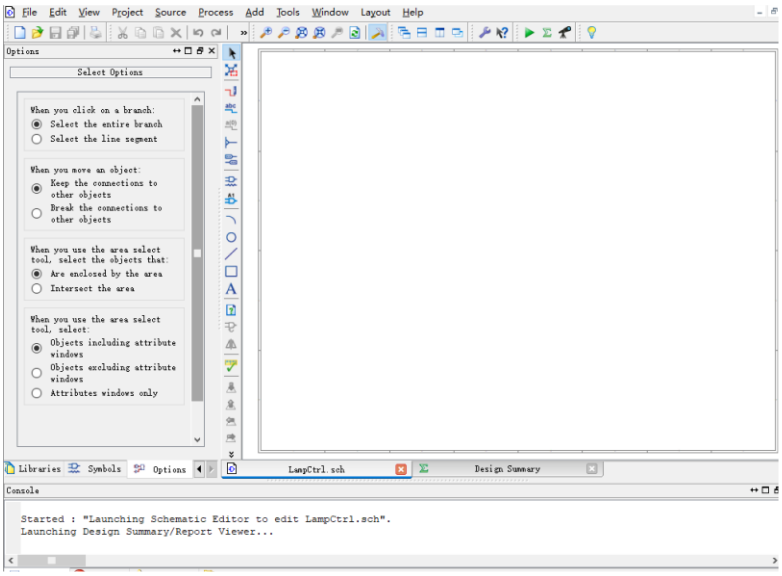

图 3.2 编辑窗口

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

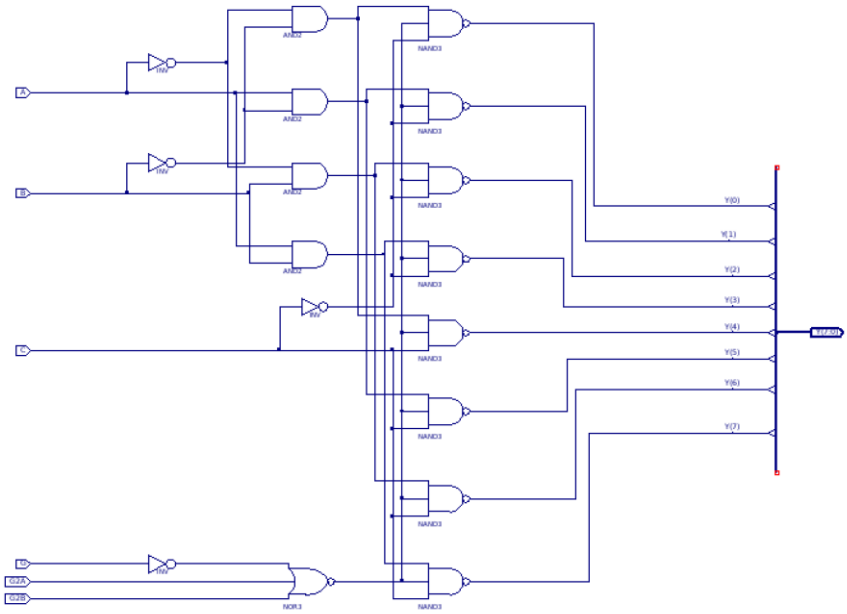

图 3.3 原理图

(e)检查错误并查看电路的硬件描述代码

在 Sources 窗口中选择 Sources for:Synthesis/Implementation，选中 D_74LS138 图标，在 Processes 窗口 Processes 选项卡中展开 Design Utilities 双击 Check Design Rules 无误后双击 View HDL Functional Model，如下图：

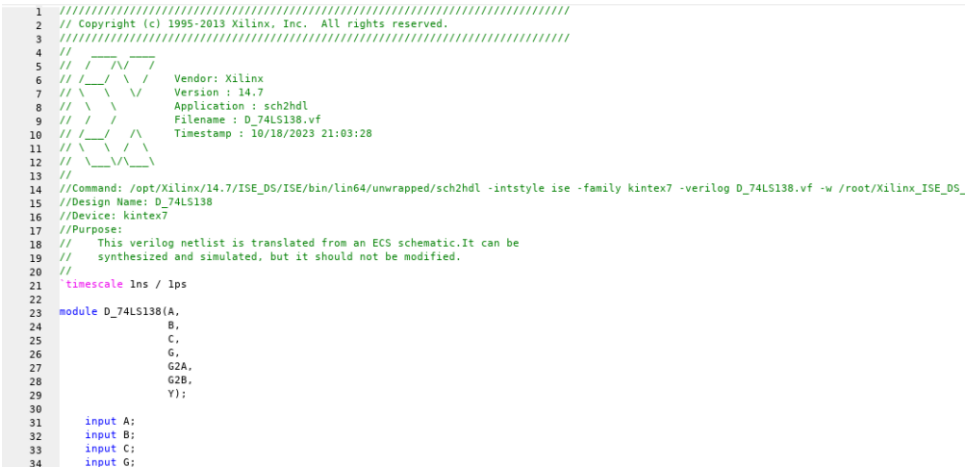

图 3.4 硬件描述代码

硬件描述代码如下：

```
`timescale 1ns / 1ps

module D_74LS138(A,
                 B,
                 C,
                 G,
                 G2A,
                 G2B,
                 Y);

    input A;
    input B;
    input C;
    input G;
    input G2A;
    input G2B;
   output [7:0] Y;

   wire XLXN_2;
   wire XLXN_4;
   wire XLXN_5;
   wire XLXN_6;
   wire XLXN_7;
   wire XLXN_8;
   wire XLXN_11;
   wire XLXN_12;
   wire XLXN_48;

   INV  XLXI_1 (.I(A),
```

```
            .O(XLXN_11));
  INV  XLXI_2 (.I(B),
              .O(XLXN_12));
  AND2  XLXI_3 (.I0(XLXN_12),
                .I1(XLXN_11),
                .O(XLXN_2));
  AND2  XLXI_4 (.I0(XLXN_12),
                .I1(A),
                .O(XLXN_4));
  AND2  XLXI_5 (.I0(B),
                .I1(XLXN_11),
                .O(XLXN_5));
  INV  XLXI_7 (.I(G),
              .O(XLXN_48));
  NOR3  XLXI_8 (.I0(G2B),
                .I1(G2A),
                .I2(XLXN_48),
                .O(XLXN_8));
  NAND3  XLXI_9 (.I0(XLXN_7),
                 .I1(XLXN_8),
                 .I2(XLXN_2),
                 .O(Y[0]));
  NAND3  XLXI_10 (.I0(XLXN_7),
                  .I1(XLXN_8),
                  .I2(XLXN_4),
                  .O(Y[1]));
  NAND3  XLXI_11 (.I0(XLXN_7),
                  .I1(XLXN_8),
                  .I2(XLXN_5),
                  .O(Y[2]));
  NAND3  XLXI_12 (.I0(XLXN_7),
                  .I1(XLXN_8),
                  .I2(XLXN_6),
                  .O(Y[3]));
  NAND3  XLXI_13 (.I0(C),
                  .I1(XLXN_8),
                  .I2(XLXN_2),
                  .O(Y[4]));
  NAND3  XLXI_14 (.I0(C),
                  .I1(XLXN_8),
                  .I2(XLXN_4),
                  .O(Y[5]));
  NAND3  XLXI_15 (.I0(C),
                  .I1(XLXN_8),
```

```
                .I2(XLXN_5),
                .O(Y[6]));
  NAND3  XLXI_16 (.I0(C),
                .I1(XLXN_8),
                .I2(XLXN_6),
                .O(Y[7]));
  AND2  XLXI_17 (.I0(B),
                .I1(A),
                .O(XLXN_6));
  INV  XLXI_19 (.I(C),
               .O(XLXN_7));
endmodule
```

2. **建立基准测试波形文件**

(a)在 New Source 中创建文件名为 D_74LS138_sim,勾选 Add to Project 选择 D_74LS138 模块，点击 Next，在 Summary 窗口再点击 Finish，进入 D_74LS138sim.v 编辑窗口

(b)仿真激励输入

在源文件中添加以下代码：

```
`timescale 1ns / 1ps
module D_74LS138_D_74LS138_sch_tb();
// Inputs
   reg C;
   reg B;
   reg A;
   reg G2A;
   reg G2B;
   reg G;
// Output
   wire [7:0] Y;
// Bidirs
// Instantiate the UUT
   D_74LS138 UUT (
        .C(C),
        .B(B),
        .A(A),
        .G2A(G2A),
        .G2B(G2B),
        .G(G),
        .Y(Y)
   );
// Initialize Inputs
```

```
// `ifdef auto_init
    integer i;
    initial begin
        C = 0;
        B = 0;
        A = 0;
        G = 1;
        G2A = 0;
        G2B = 0;
        #50;
        for (i=0; i<=7;i=i+1) begin
            {C,B,A} = i;
        #50;
    end
        assign G = 0;
        assign G2A = 0;
        assign G2B = 0;
        #50;

        assign G = 1;
        assign G2A = 1;
        assign G2B = 0;
        #50;

        assign G = 1;
        assign G2A = 0;
        assign G2B = 1;
        #50;
    end
    endmodule
```

可得到如下波形图，点击调节缩放按钮直至出现完整波形：

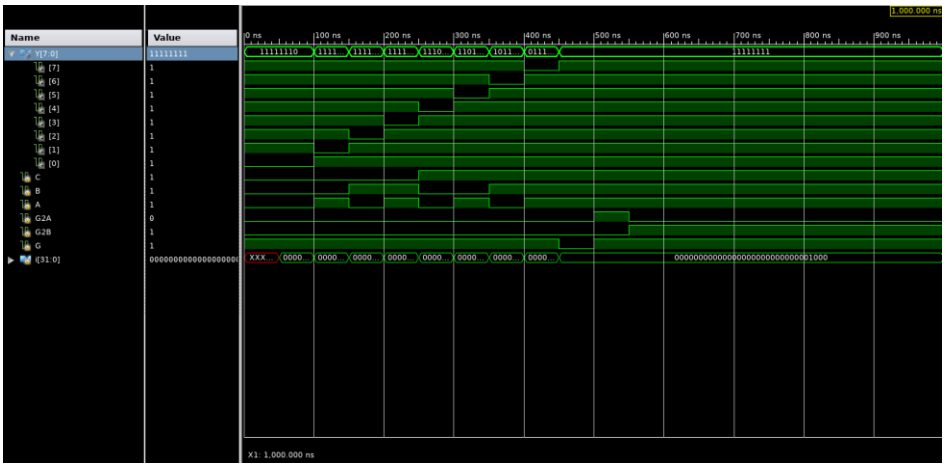

图 3.5 仿真波形图

3. 生成逻辑符号图

(a)Create Schematic Symbol

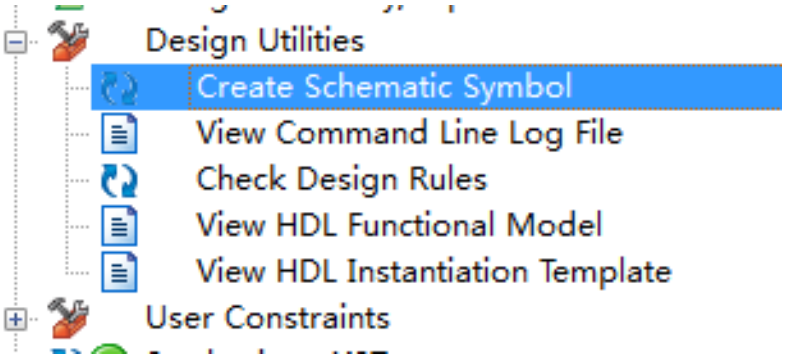

图 3.6 Create Schematic Symbol

(b)符号图位于工程根目录

自动生成的符号可修改：可以用 Tools 菜单的 Symbol Wizard，也可以打开.sym 文件直接修改

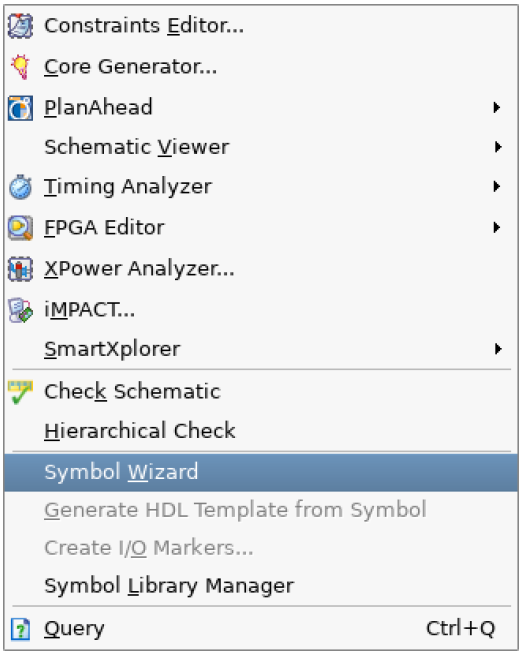

图 3.7 Symbol Wizard

(c)在新工程中使用时，把.sym 和.sch 复制到对应工程目录

4. 验证 D_74LS138

(a)新建工程“D_74LS138_Test”

(b)新建 Schematic 文件“D_74LS138_Test”

(c)复制 D_74LS138.sym 和.sch 到工程目录:

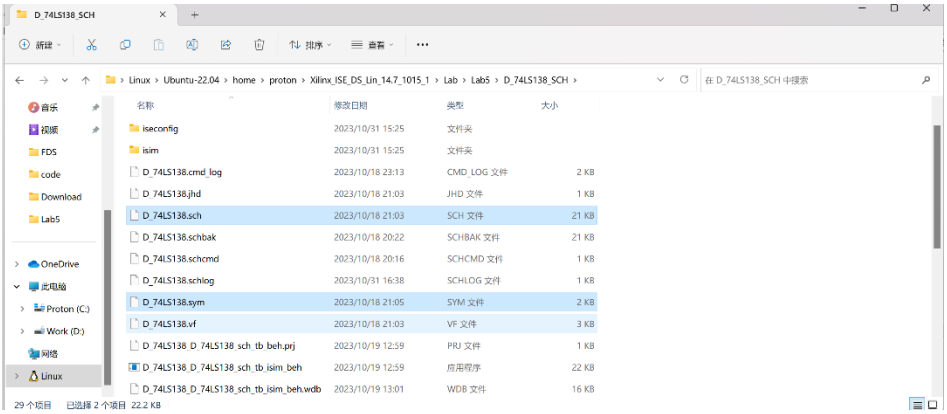

图 3.8 .sym 和.sch

右键单击 Source 面板，单击 Add New Source 将这两个文件加入 Project 需要注意的是我们在选择添加文件时要选择 All Files，这样才能看到.sym 文件。

(d)在 symbol 框里的第一个元件就是 D_74LS138：

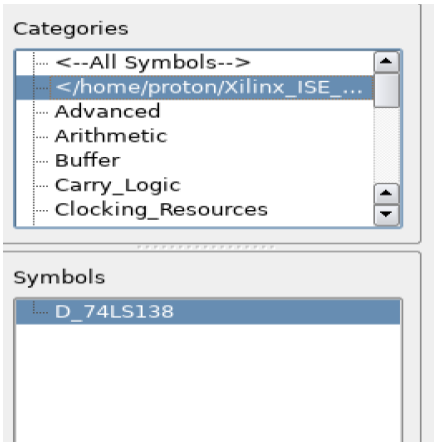

图 3.9 D_74LS138 元件

(e)用拨盘开关控制模块的输入，用 LED(7:0)作为模块的输出，验证模块的功能。在.sch 中绘制测试电路，具体的电路图如下:

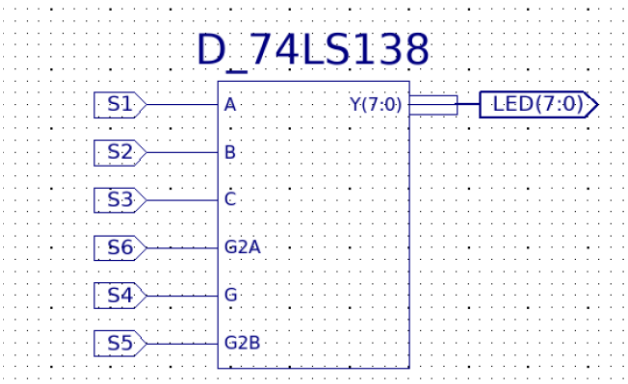

图 3.10 LED(7:0)电路图

在 D_74LS138 模块上点右键，在菜单的 Symbol->Push into Symbol 可以参看模块的原理图：

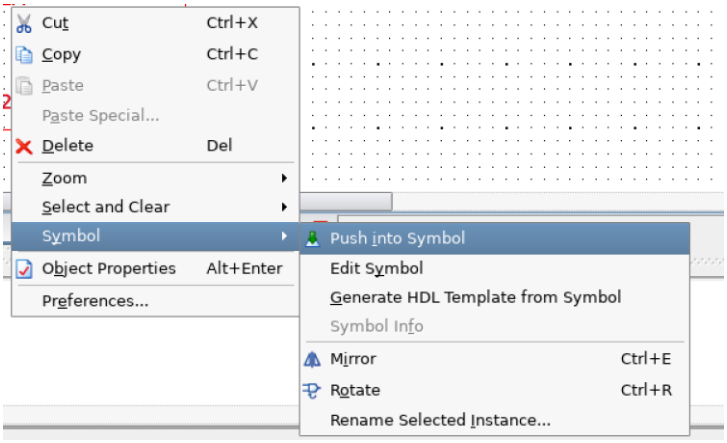

图 3.11 查看原理图

空白处右键菜单里的 Pop to calling Schematic 回到上层模块：

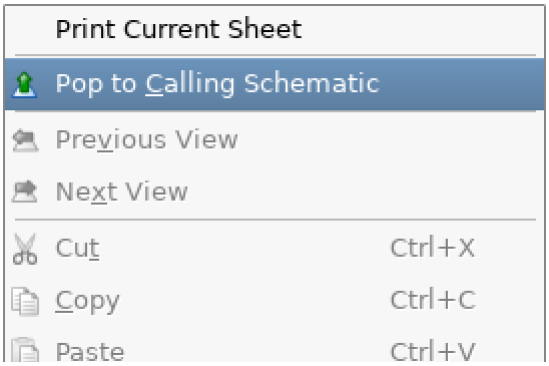

图 3.12 回到上层模块

5. 下载到 SWORD 板验证

(a)建立 K7.ucf 文件，代码如下：

```
NET "S1" LOC = AA10 | IOSTANDARD = LVCMOS15;
NET "S2" LOC = AB10 | IOSTANDARD = LVCMOS15;
NET "S3" LOC = AA13 | IOSTANDARD = LVCMOS15;
NET "S4" LOC = AA12 | IOSTANDARD = LVCMOS15;
NET "S5" LOC = Y13 | IOSTANDARD = LVCMOS15;
NET "S6" LOC = Y12 | IOSTANDARD = LVCMOS15;
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;
NET "LED[7]" LOC = AF24 | IOSTANDARD = LVCMOS33;
```

(b)点击 Generate Programming File，生成二进制 bit 文件。可以在 Design Summary 中查看 Pinouts Report：

	Pin Number	gn an	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number	Drive (mA)	Slew Rate	Termination	IOB Delay	Voltage	Constraint	IO Register	Signal Integrity
1	W23	LED<...	IOB33	IO_L8P_T1_12	OUTPUT	LVCMO...	12	12	SL...				LOCATED	NO	NONE
2	AB26	LED<...	IOB33	IO_L9P_T1_DQS_12	OUTPUT	LVCMO...	12	12	SL...				LOCATED	NO	NONE
3	Y25	LED<...	IOB33	IO_L10P_T1_12	OUTPUT	LVCMO...	12	12	SL...				LOCATED	NO	NONE
4	AA23	LED<...	IOB33	IO_L11P_T1_SRCC_12	OUTPUT	LVCMO...	12	12	SL...				LOCATED	NO	NONE
5	Y23	LED<...	IOB33	IO_L12P_T1_MRCC_12	OUTPUT	LVCMO...	12	12	SL...				LOCATED	NO	NONE
6	Y22	LED<...	IOB33	IO_L13P_T2_MRCC_12	OUTPUT	LVCMO...	12	12	SL...				LOCATED	NO	NONE
7	AE21	LED<...	IOB33	IO_L19N_T3_VREF_12	OUTPUT	LVCMO...	12	12	SL...				LOCATED	NO	NONE
8	AF24	LED<...	IOB33	IO_L20P_T3_12	OUTPUT	LVCMO...	12	12	SL...				LOCATED	NO	NONE
9	AA10	S1	IOB	IO_L14P_T2_SRCC_33	INPUT	LVCMO...	33				NO...		LOCATED	NO	NONE
10	AB10	S2	IOB	IO_L14N_T2_SRCC_33	INPUT	LVCMO...	33				NO...		LOCATED	NO	NONE
11	AA13	S3	IOB	IO_L16P_T2_33	INPUT	LVCMO...	33				NO...		LOCATED	NO	NONE
12	AA12	S4	IOB	IO_L16N_T2_33	INPUT	LVCMO...	33				NO...		LOCATED	NO	NONE
13	Y13	S5	IOB	IO_L18P_T2_33	INPUT	LVCMO...	33				NO...		LOCATED	NO	NONE
14	Y12	S6	IOB	IO_L18N_T2_33	INPUT	LVCMO...	33				NO...		LOCATED	NO	NONE
15	Y26		IOB...	IO_L10N_T1_12	UNUSED		12								
16	Y24			VCCO_12			12					3.30			
17	Y21		IOB...	IO_L15N_T2_DQS_12	UNUSED		12								
18	Y20		IOB33	IO_25_12	UNUSED		12								
19	Y19			GND											
20	Y18		IOB...	IO_L19N_T3_VREF_32	UNUSED		32								
21	Y17		IOB...	IO_L19P_T3_32	UNUSED		32								

图 3.13 Pinouts Report

(c)按照实验 4 的步骤将二进制 bit 文件发送到实验板上。验证结果如下：其中 S1,S2,S3,G,G2B,G2A 分别对应从右往左的第 1、2、3、4、5、6 个开关。

ENABLE = 0 时：

a) G = 0 时，无论 $S_1 S_2 S_3$ 状态如何，灯都不会改变，下图选取三个开关 index 分别为 001、010、111 时灯的亮灭状态：

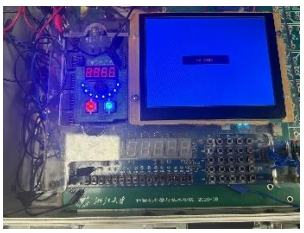

图 3.14 idx=001

图 3.15 idx=010

图 3.16 idx=111

b) G = 1 时，若至少有一个 G_{2A} 或 G_{2B} 开关为 1，则灯也不会变化，下图选取三个开关 index 分别为 100、010、111 时灯的亮灭状态

图 3.17 idx=100

图 3.18 idx=010

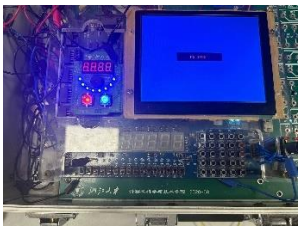

图 3.19 idx=111

ENABLE = 1 时：

即 G = 1 且 G_{2A} = G_{2B} = 0，此时可通过改变三个开关控制灯的亮灭，实际验证如下：

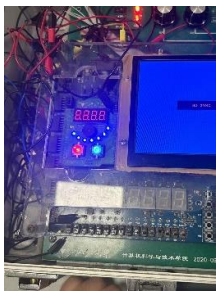
图 3.20 000


图 3.21 001


图 3.22 010


图 3.23 011


图 3.24 100


图 3.25 101


图 3.26 110


图 3.27 111

3.2 实现楼道灯控制

1. 设计实现楼道灯电路

(a)新建工程 LampCtrl138

(b)复制 D_74LS138.sym 和.sch 文件到工程目录并加入工程(Add Source)

(c)在 Symbols 框里的第一个元件，就是之前设计的译码器 D_74LS138

(d)根据实验原理，用原理图输入的方式输入如下电路原理图：

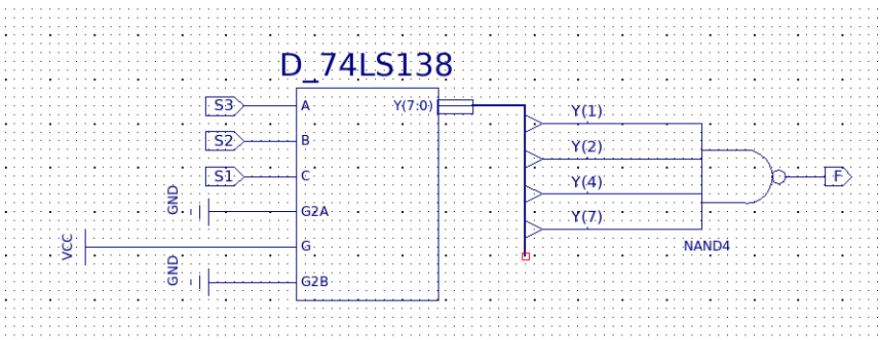

图 3.28 楼道灯控制电路原理图

2. 模拟仿真

(a)创建 New Sources，新建源文件类型为 Verilog Test Fixture，输入

文件名 LampCtrl138_sim，并勾选 Add to Project.

(b)双击进入文件添加仿真代码如下：

```
`timescale 1ns / 1ps
module LampCtrl138_LampCtrl138_sch_tb();
   reg S2;
   reg S3;
   reg S1;
   wire F;
   LampCtrl138 UUT (
        .S3(S3),
        .S2(S2),
        .S1(S1),
        .F(F)
   );
integer i;
initial begin
    for (i = 0; i <= 8; i = i + 1)
        begin{S3, S2, S1} <= i;
    # 50;
    end
end
endmodule
```

(c)在 View 窗口选择 Simulation,选择文件并点击 Simulate Behavioral Model，得到如下波形：

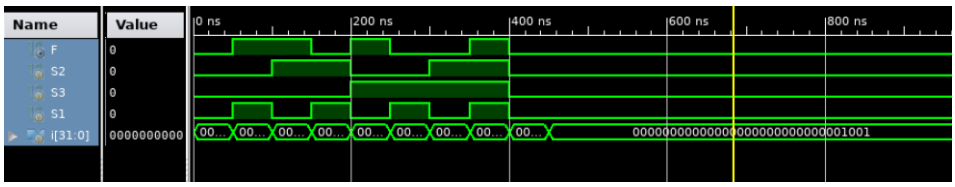

图 3.29 仿真波形

3. 下载到 SWORD 板验证

(a)建立 K7.ucf 文件，代码如下：

```
NET "S1" LOC = AA10 | IOSTANDARD = LVCMOS15;
NET "S2" LOC = AB10 | IOSTANDARD = LVCMOS15;
NET "S3" LOC = AA13 | IOSTANDARD = LVCMOS15;
NET "F" LOC = AF24 | IOSTANDARD = LVCMOS33;
```

(b)点击 Generate Programming File，生成二进制 bit 文件

(c)按照实验 4 的步骤将 bit 文件发送到实验板上。根据引脚约束文件，S1,S2,S3 分别对应从左往右的第 1、2、3 个开关，结果如下：

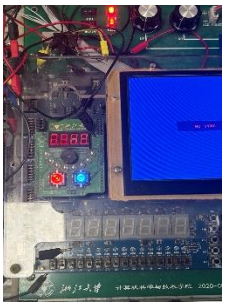

图 3.30 000

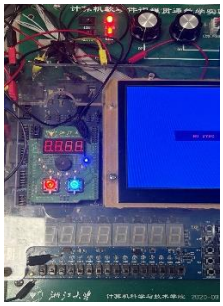

图 3.31 001

图 3.32 010

图 3.33 011

图 3.34 100

图 3.35 101

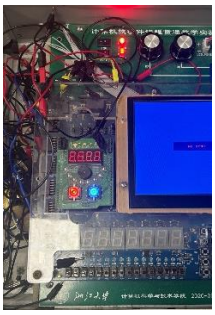

图 3.36 110

图 3.37 111

四、实验结果分析

本次实验从楼道灯控制这一问题出发，通过变量编码器的应用，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

(a) 第一个仿真激励代码令 ENABLE=1，译码器随输入改变而改变，并用用循环的方式依次遍历真值表，因此能观察到对应编号的灯也随之关闭，输出结果与预期相同。

(b) 第二个仿真激励输入的代码是令 S1， S2， S3 从 0， 0， 0 变换到 1，

1, 1，用循环逐次遍历真值表。仿真波形图与对应输出结果与预期相同。

3. **开发板验证**

当 ENABLE = 1 时，三个开关以任意组合输入，对应的灯会熄灭，实现了译码器功能。当 ENABLE = 0 时，灯不受开关影响。

实验二结果与实验一相同，灯均在偶数个开关闭合时熄灭，奇数个开关闭合时亮起。符合预期结果。

五、讨论与心得

本次实验前已经按照 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。第二次上板实验让我对 Verilog 语言以及代码与硬件之间的关系有了进一步的了解，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！