

浙江大学



计算机逻辑设计基础

Lab 5

Author: 苏煜程

Student ID: 3220105481

Date: 2023 年 11 月 1 日

浙江大学实验报告

课程名称：计算机逻辑设计基础

实验名称：变量译码器设计与应用

学生姓名：苏煜程 专业：人工智能（图灵班） 学号：3220105481

同组学生姓名：张延泽 指导老师：董亚波

实验地点：东 4-509 实验日期：2023 年 10 月 19 日

1 实验目的

1. 掌握变量译码器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

2 实验任务

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

3 实验原理

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）

译码可分为：变量译码、显示译码

- 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 2^n 译码和 8421BCD 码译码
- 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

3.1 变量译码器

变量译码器是一个将 n 个输入变为 2^n 个最小项输出的多输出端的组合逻辑电路。 n 通常在 $2 \sim 64$ 之间。

3.1.1 变量译码器 74LS138

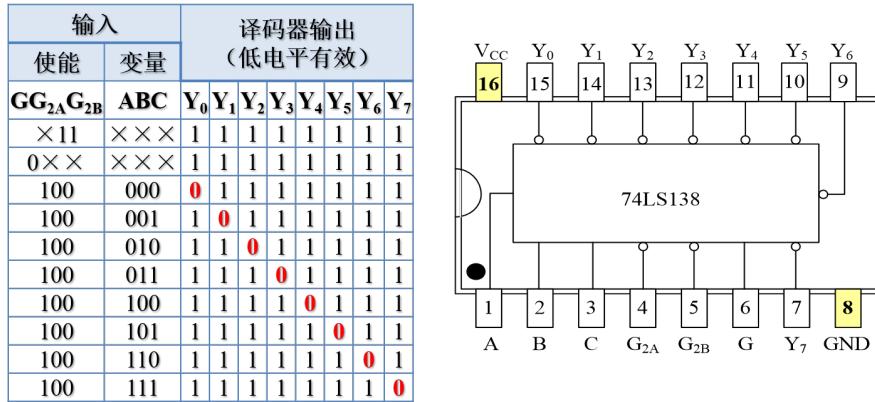


图 1: 74LS138 变量译码器功能表和引脚

带 3 个使能端的 3-8 译码器的逻辑结构由三级门电路构成，输出低电平有效。

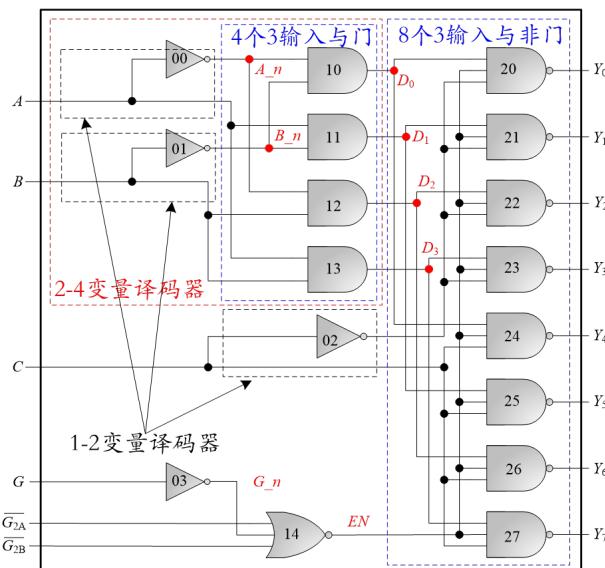


图 2: 74LS138 门电路构成

其 verilog 代码如下：

```

1 module decoder_3_8(A, B, C, G, G2A,G2B, Y);
2   input wire A, B, C, G, G2A, G2B;
3   output wire [7:0] Y;
4   not      node_0_0(A_n, A),
5     node_0_1(B_n, B),
6     node_0_2(C_n, C),
7     node_0_3(G_n, G);
8   and     node_1_0(D0, B_n, A_n),

```

```

9      node_1_1(D1, B_n, A ),
10     node_1_2(D2, B,   A_n),
11     node_1_3(D3, B,   A );
12 nor  node_1_4(EN, G_n, G2A, G2B);
13 nand node_2_0(Y[0], EN, D0, C_n),
14     node_2_1(Y[1], EN, D1, C_n),
15     node_2_2(Y[2], EN, D2, C_n),
16     node_2_3(Y[3], EN, D3, C_n),
17     node_2_4(Y[4], EN, D0, C ),
18     node_2_5(Y[5], EN, D1, C ),
19     node_2_6(Y[6], EN, D2, C ),
20     node_2_7(Y[7], EN, D3, C );
21 endmodule

```

3.1.2 变量译码器 74LS139

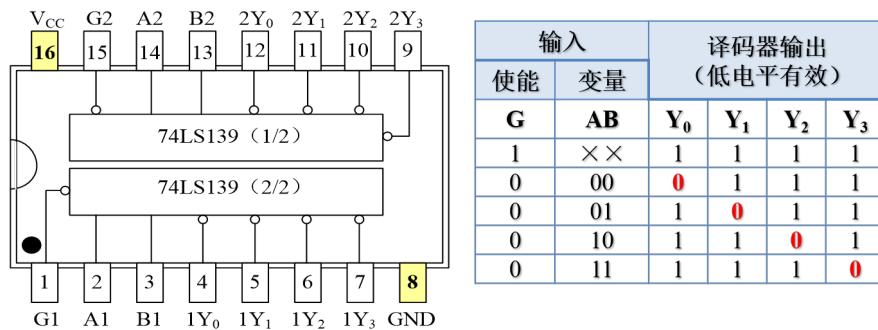


图 3: 74LS139 变量译码器功能表和引脚

3.2 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

$$F = \bar{S}_3 \bar{S}_2 S_1 + \bar{S}_3 S_2 \bar{S}_1 + S_3 \bar{S}_2 \bar{S}_1 + S_3 S_2 S_1$$

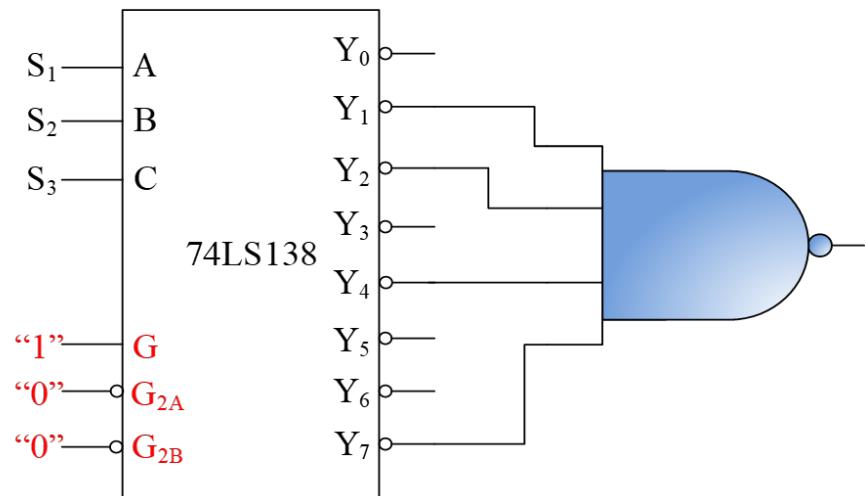


图 4: 变量译码器实现函数的组合电路

其 verilog 代码如下

```

1 module lampCtr1(s1, s2, s3, F);
2     ..... //端口及变量定义
3     decoder_3_8 Decoder3_8(s1, s2, s3, G, G2A,G2B, Y);
4     nand    node(F, Y[1], Y[2], Y[4] , Y[7]);
5 endmodule

```

4 实验内容与步骤

4.1 任务 1: 原理图设计实现 74LS138 译码器模块

4.1.1 设计实现 74LS138

1. 新建工程，工程名称用 D_74LS138_SCH
2. 新建 Schematic 源文件，文件名称用 D_74LS138
3. 原理图方式进行设计



图 5: 新建工程

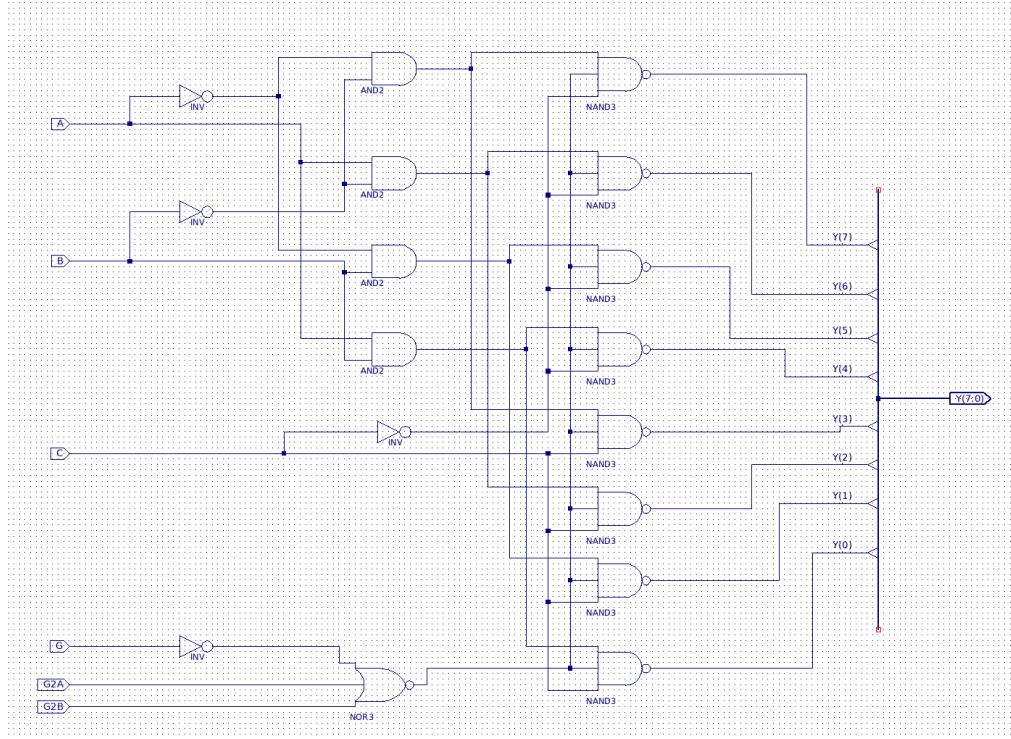


图 6: 原理图

4. Check Design Rules, 检查错误
5. View HDL Functional Model, 查看并学习 Verilog HDL 代码



图 7: Check Design Rules

4.1.2 仿真

对 D_74LS138 模块进行仿真，激励代码如下

```

1 integer i;
2 initial begin
3     A = 0;
4     B = 0;
5     C = 0;
6
7     G = 1;

```

```

8   G2A = 0;
9   G2B = 0;
10  #50;

11
12  for (i = 0; i <= 7; i = i + 1) begin
13      {C, B, A} = i;
14      #50;
15  end

16
17  assign G = 0;
18  assign G2A = 0;
19  assign G2B = 0;
20  #50;

21
22  assign G = 1;
23  assign G2A = 1;
24  assign G2B = 0;
25  #50;

26
27  assign G = 1;
28  assign G2A = 0;
29  assign G2B = 1;
30  #50;

31 end

```

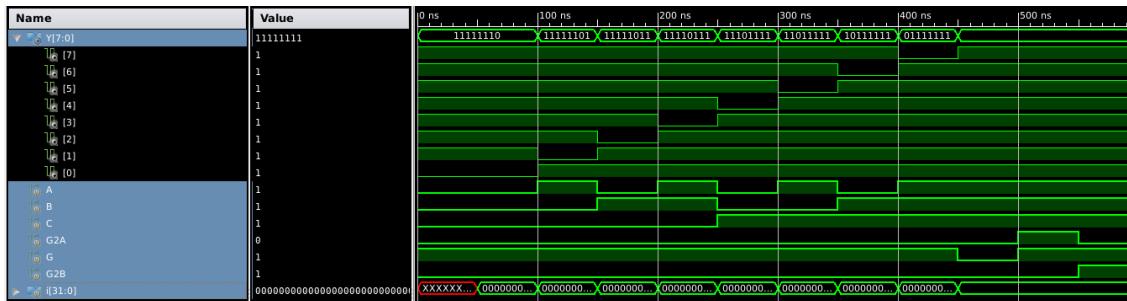
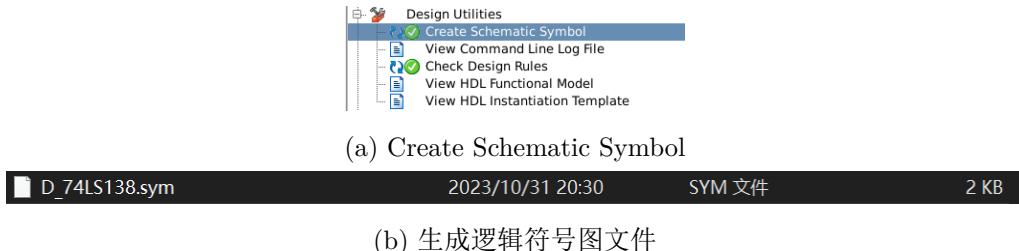


图 8: 仿真得到波形

4.1.3 生成逻辑符号图

Create Schematic Symbol, 系统生成 D_74LS138 模块的逻辑符号图文件, 文件后缀 .sym



4.1.4 验证 D_74LS138

1. 新建工程 D_74LS138_Test
2. 新建 Schematic 文件 D_74LS138_Test
3. 复制 D_74LS138.sym 和 .sch 到工程目录

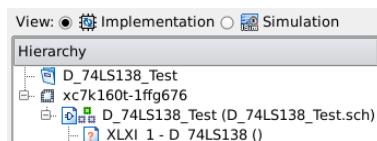


图 10: 新建工程

4. 在 symbols 框里的第一个元件，就是 D_74LS138
5. 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

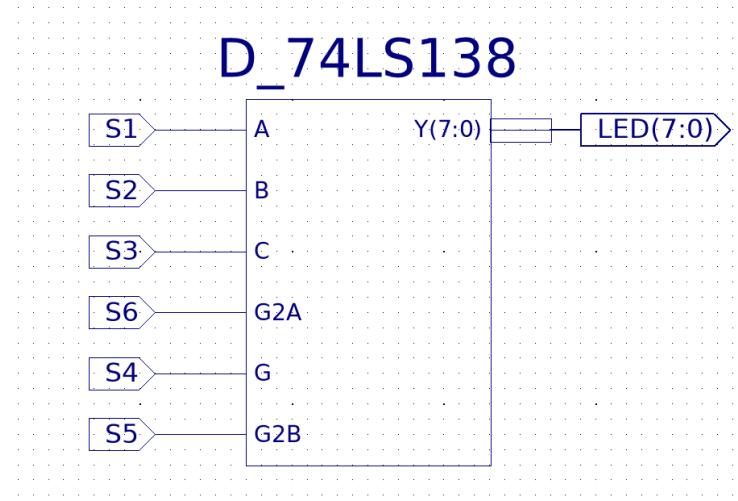


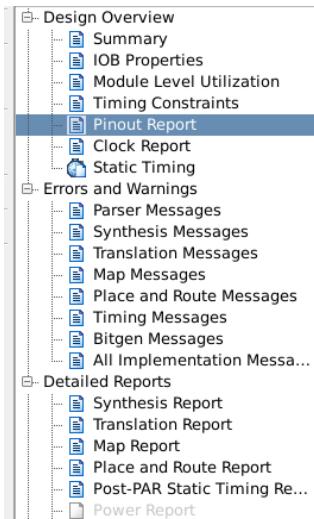
图 11: 验证模块功能

4.1.5 下载验证

建立 K7.ucf 文件，内容如下：

```
1 NET "S1" LOC = AA10 | IOSTANDARD = LVCMOS15;
2 NET "S2" LOC = AB10 | IOSTANDARD = LVCMOS15;
3 NET "S3" LOC = AA13 | IOSTANDARD = LVCMOS15;
4 NET "S4" LOC = AA12 | IOSTANDARD = LVCMOS15;
5 NET "S5" LOC = Y13 | IOSTANDARD = LVCMOS15;
6 NET "S6" LOC = Y12 | IOSTANDARD = LVCMOS15;
7 NET "LED[0]" LOC = W23 | IOSTANDARD = LVCMOS33;
8 NET "LED[1]" LOC = AB26 | IOSTANDARD = LVCMOS33;
9 NET "LED[2]" LOC = Y25 | IOSTANDARD = LVCMOS33;
10 NET "LED[3]" LOC = AA23 | IOSTANDARD = LVCMOS33;
11 NET "LED[4]" LOC = Y23 | IOSTANDARD = LVCMOS33;
12 NET "LED[5]" LOC = Y22 | IOSTANDARD = LVCMOS33;
13 NET "LED[6]" LOC = AE21 | IOSTANDARD = LVCMOS33;
14 NET "LED[7]" LOC = AF24 | IOSTANDARD = LVCMOS33;
```

生成 Pinouts Report 如下：



	Pin Number	ign z / lam	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number
1	W23	LED<0>	IOB33	IO_L8P_T1_12	OUTPUT	LVCMOS33	12
2	AB26	LED<1>	IOB33	IO_L9P_T1_DQS_12	OUTPUT	LVCMOS33	12
3	Y25	LED<2>	IOB33	IO_L10P_T1_12	OUTPUT	LVCMOS33	12
4	AA23	LED<3>	IOB33	IO_L11P_T1_SRCC_12	OUTPUT	LVCMOS33	12
5	Y23	LED<4>	IOB33	IO_L12P_T1_MRCC_12	OUTPUT	LVCMOS33	12
6	Y22	LED<5>	IOB33	IO_L13P_T2_MRCC_12	OUTPUT	LVCMOS33	12
7	AE21	LED<6>	IOB33	IO_L19N_T3_VREF_12	OUTPUT	LVCMOS33	12
8	AF24	LED<7>	IOB33	IO_L20P_T3_12	OUTPUT	LVCMOS33	12
9	AA10	S1	IOB	IO_L14P_T2_SRCC_33	INPUT	LVCMOS15	33
10	AB10	S2	IOB	IO_L14N_T2_SRCC_33	INPUT	LVCMOS15	33
11	AA13	S3	IOB	IO_L16P_T2_33	INPUT	LVCMOS15	33
12	AA12	S4	IOB	IO_L16N_T2_33	INPUT	LVCMOS15	33
13	Y13	S5	IOB	IO_L18P_T2_33	INPUT	LVCMOS15	33
14	Y12	S6	IOB	IO_L18N_T2_33	INPUT	LVCMOS15	33
15	A1			GND			

图 12: Pinouts Report

4.1.6 根据真值表验证

根据真值表，操作实验板，验证功能。

根据电路图和引脚约束文件， $S_1, S_2, S_3, G, G_{2B}, G_{2A}$ 分别对应从右往左第 1、2、3、4、5、6 个开关。

ENABLE = 0 当 $G = 0$ 时, ENABLE 为 0, 无论 S_1, S_2, S_3 如何变化, 灯始终无变化。

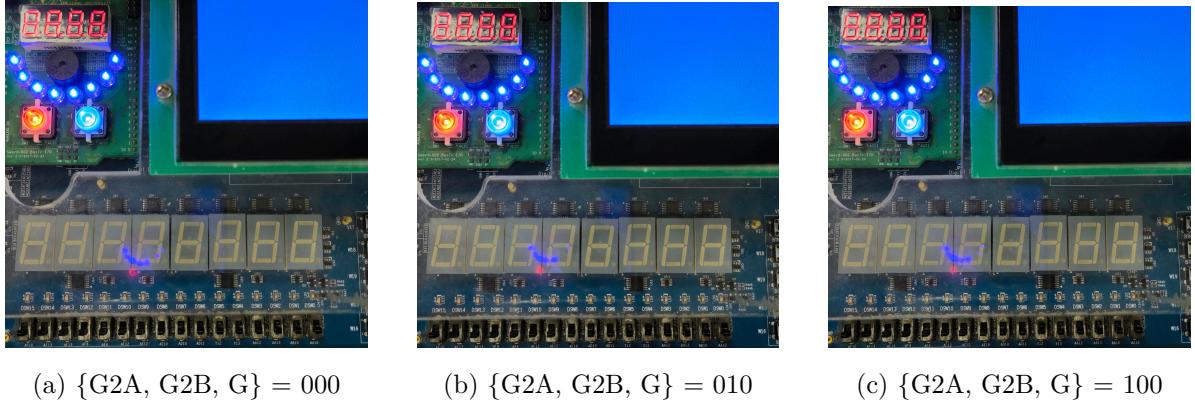


图 13: $G=0$ 时部分测试结果

当 $G = 1$ 时, 若 G_{2B}, G_{2A} 其中至少一个为 1, 则 ENABLE 也为 0, 灯始终无变化。

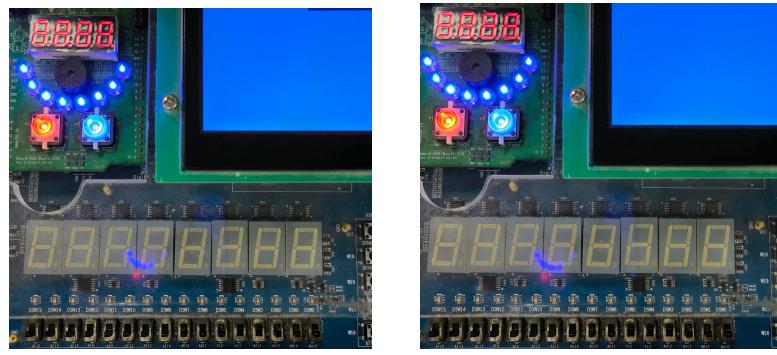


图 14: $G=1$, 且 G_{2B}, G_{2A} 其中至少一个为 1 时部分测试结果

ENABLE = 1 当且仅当 $G = 1, G_{2B} = G_{2A} = 0$ 时 $ENABLE = 1$, 可以通过 S_1, S_2, S_3 的开关控制灯的亮灭。

(Next Page)



(a) $\{S_3, S_2, S_1\} = 000$, 第 0 个灯熄灭 (b) $\{S_3, S_2, S_1\} = 001$, 第 1 个灯熄灭 (c) $\{S_3, S_2, S_1\} = 011$, 第 3 个灯熄灭



(d) $\{S_3, S_2, S_1\} = 111$, 第 7 个灯熄灭 (e) $\{S_3, S_2, S_1\} = 110$, 第 6 个灯熄灭 (f) $\{S_3, S_2, S_1\} = 100$, 第 4 个灯熄灭

图 15: $\{G_{2A}, G_{2B}, G\} = 001$ 时, 改变 $\{S_3, S_2, S_1\}$

4.2 任务 2: 用 74LS138 译码器实现楼道灯控制器

4.2.1 设计实现楼道灯电路

1. 新建工程 LampCtrl138
2. 复制 D_74LS138.sym 和 .sch 文件到工程目录, 在 symbols 框里的第一个元件就是 D_74LS138



图 16: 新建工程

3. 根据前面原理, 用原理图方式输入

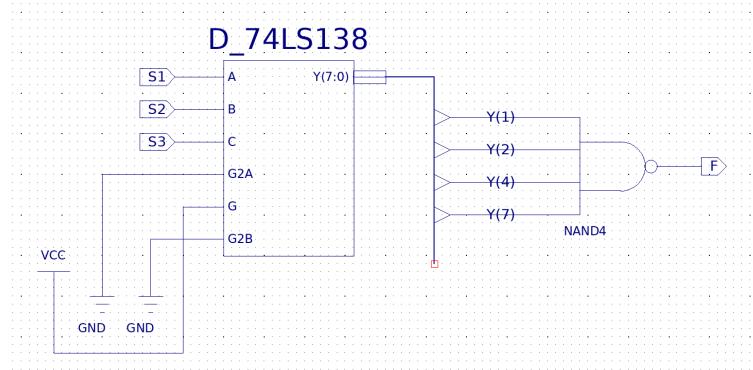


图 17: 原理图

4.2.2 仿真

1. 在 Sources 窗口空白处的右键菜单中选择 New Source , 在新建源文件向导中选择源类型为 Verilog Test Fixture, 输入文件名 LampCtrl138_sim, 并勾选 Add to Project
2. 输入仿真代码

```

1 `timescale 1ns / 1ps
2
3 module LampCtrl138_LampCtrl138_sch_tb();
4
5 // Inputs
6   reg S1;
7   reg S2;
8   reg S3;
9
10 // Output
11   wire F;
12
13 // Bidirs
14
15 // Instantiate the UUT
16   LampCtrl138 UUT (
17     .S1(S1),
18     .S2(S2),
19     .S3(S3),
20     .F(F)
21   );
22 // Initialize Inputs

```

```

23 // `ifdef auto_init
24     integer i;
25     initial begin
26         for (i = 0; i < 8; i = i + 1) begin
27             {S3, S2, S1} = i;
28             #50;
29         end
30     end
31 // `endif
32 endmodule

```

3. 在 View 窗口选择 Simulation , 选择文件并且点击 Simulate Behavioral Model , 得到波形如下 可以看出与第四次试验的仿真波形相同



图 18: 仿真波形

4.2.3 下载验证

1. 建立 K7.ucf 文件，输入引脚约束如下

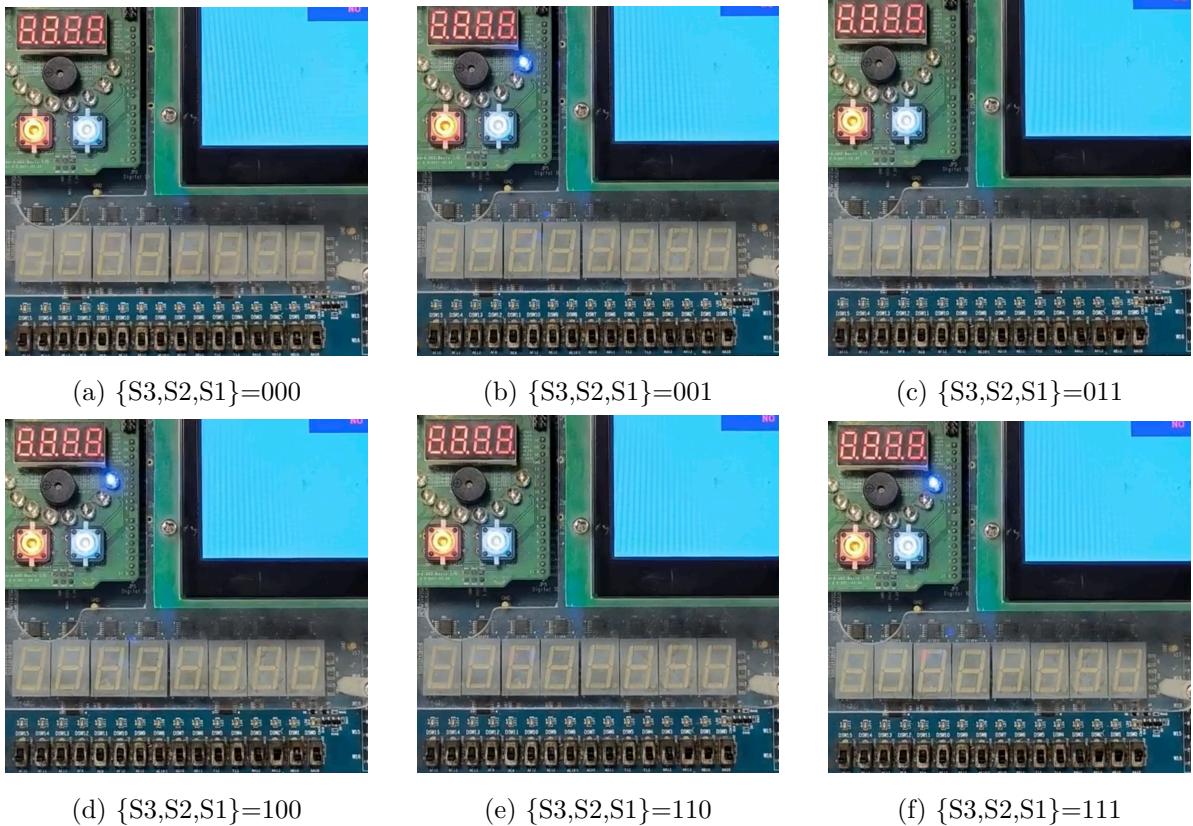
```

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;

```

2. 点击 Generate Programming File , 生成 .bit 文件

3. 将二进制文件发送到实验板上，验证结果如下



5 实验结果分析

5.1 分析硬件描述代码

查看输入电路的硬件描述代码时,可以发现里面的语句和电路图都是相对应的。比如 AND2(二输入与门), NAND(三输入与非门), INV(inverte r, 反相器), NOR3(三输入或非门) 还有 VCC(模拟信号源), GND(模拟接地). 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。

5.2 分析仿真结果

5.2.1 任务 1

仿真激励输入的代码让 ENABLE=1(即 $G = 1, G2A = G2B = 0$) 即译码器能随输入改变而改变。随后用仿真输入代码用循环的方式依次遍历真值表, 同时观察到对应编号的灯也随之关闭。由此, 输出结果与预期相同。

5.2.2 任务 2

仿真激励输入的代码让 (S_0, S_1, S_2) 从 $(0, 0, 0)$ 变换到 $(1, 1, 1)$, 即逐次遍历真值表。仿真激励输入的代码用循环来实现, 仿真波形图中可以看到对应输出结果与预期相同。

5.3 分析开发板结果

5.3.1 任务 1

让 ENABLE=1 时，三个开关以某种组合输入，其 index 对应的灯就会熄灭，即实现了译码器的功能。

若 ENABLE=0，则无论怎么拨动 S_0, S_1, S_2 灯也不会有变化

5.3.2 任务 2

结果与上一次实验的第一部分的结果完全相同：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。

6 讨论与心得

这次试验我们编写了变量译码器来实现实验 4 控制楼道灯的功能。相较于上一次实验，使用变量译码器提供了模块化的思路，使代码有更好的可移植性。