

# 浙江大学

## 本科实验报告

课程名称: 数字逻辑电路设计

姓 名: 金祺书

学 院: 计算机科学与技术学院

专 业: 计算机科学与技术

指导教师: 洪奇军

报告日期: 2024 年 3 月 28 日

# 浙江大学实验报告

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 变量译码器设计与应用

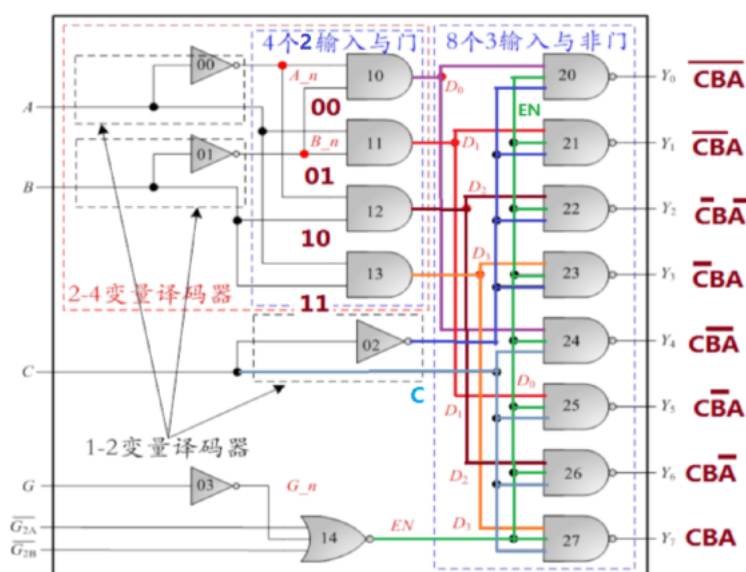
学生姓名： 金祺书 学号： 3230104248 同组学生姓名： 蒋翼泽

实验地点： 紫金港东四 509 室 实验日期： 2024 年 3 月 28 日

## 一、操作方法与实验步骤

### ①原理图设计实现 74LS138 译码器模块

- (1) 新建 Logisim 工程，修改电路名为 D\_74LS138。
- (2) 根据以下 D\_74LS138 译码器原理图绘制电路。



- (3) 绘图完成后获得 Verilog 代码，使用 Vivado 新建工程，将 D\_74LS138/verilog/circuit/ 以及 D\_74LS138/verilog/gates/ 中的代码导入到工程中。使用仿真文件对生成的 Verilog 代码进行仿真。

### ②用 74LS138 译码器实现楼道灯控制器

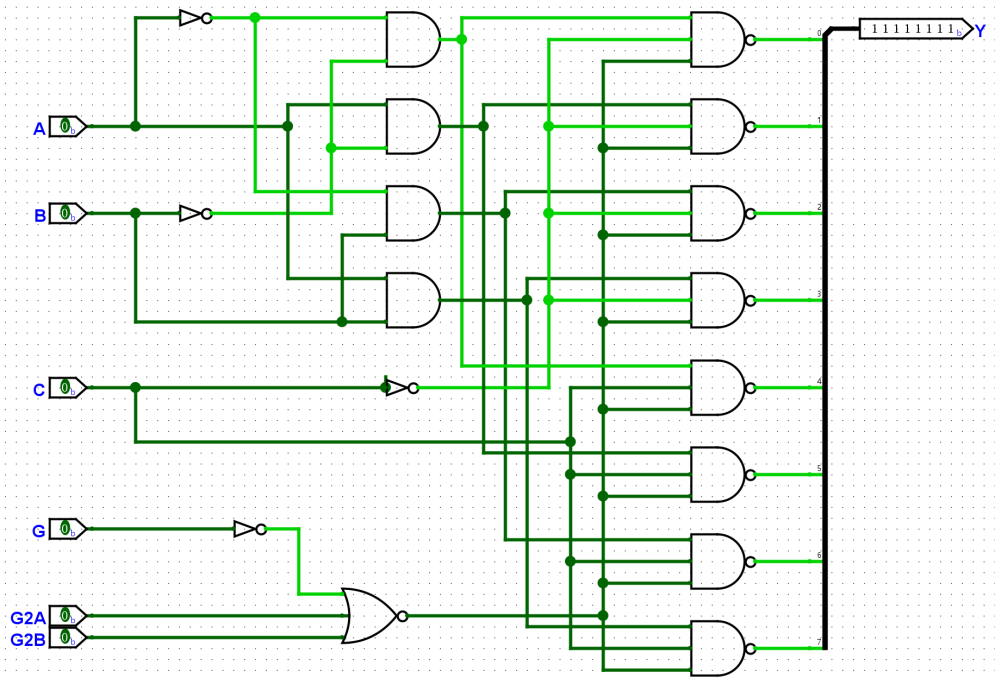
- (1) 根据上个实验得到的真值表，写出其对应的最小项或最大项形式。
- (2) 绘制电路图，绘制完成后，导出为 Verilog 文件。
- (3) 使用上一节对 D\_74LS138 仿真时新建的工程或新建一个工程，将本节得到的代码文件

导入到工程中。使用 Lab4 提供的仿真文件进行仿真。确定器件仿真行为正常后，导入 Lab4 提供的约束文件，生成比特流并烧录到板上，查看现象。

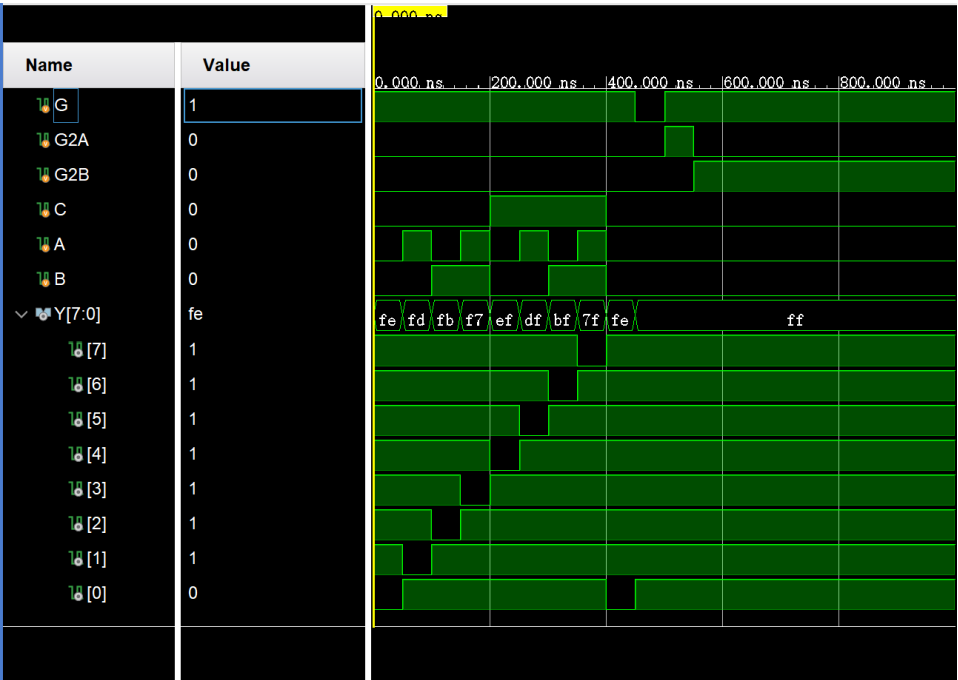
## 二、实验结果与分析

### ①原理图设计实现 74LS138 译码器模块

绘制电路图后对应的 Logisim 界面截图如下：



导入 Vivado 后的仿真界面截图如下：



74LS138 有三个使能信号 G,G2A,G2B, 其中 G 信号为高电平有效的主要使能信号, 当它处于低电位时, 所有的输出信号均处于无效电位 (高电位)。

G2A,G2B 是看起来有些冗余的使能信号, 它们为低电平有效, 当两个信号都为低电平且 G 为高电平时, 3-8 译码器能正常工作; 当 G2A 与 G2B 中有一个为高电平时, 输出均处于无效电平。

当 3-8 译码器正常工作时, 约定 A 信号为最低位的地址信号, C 为最高位的地址信号, 比如 A,B,C 信号值分别为 0,1,1 时, 我们表达的地址为 b110, 即对应选择 Y6。

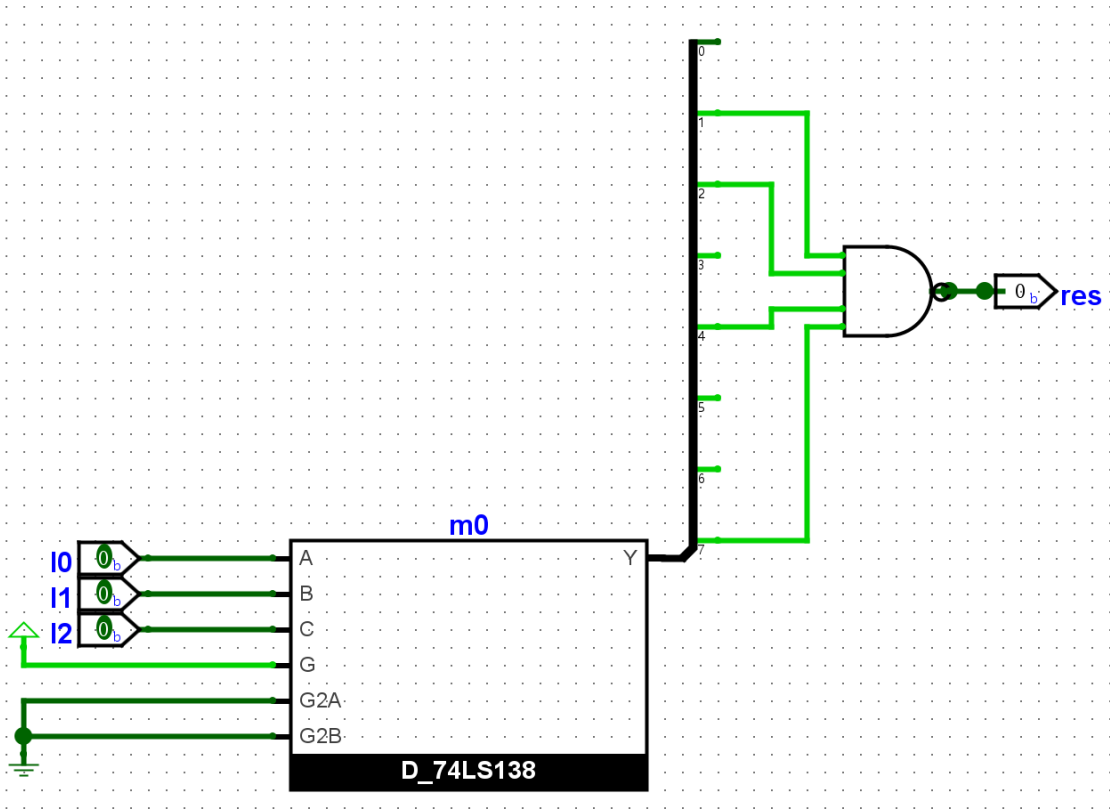
### ②用 74LS138 译码器实现楼道灯控制器

电路图真值表如下:

$I_0$	$I_1$	$I_2$	Res
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

其对应的最小项之和形式:  $Res = \bar{I_0}\bar{I_1}I_2 + \bar{I_0}I_1\bar{I_2} + I_0\bar{I_1}\bar{I_2} + I_0I_1I_2$

绘制电路图后对应的 Logisim 界面截图如下:



导入 Vivado 后的仿真界面截图如下:



当输入  $I_0=1$ ,  $I_1=0$ ,  $I_2=0$  时，在板上 LED 灯亮，符合预期结果。



### 三、讨论、心得

本次实验我们更深入地实践 Logisim 和 Vivado，过程还是比较简单轻松的，不过这过程中我并没有自己编写 Verilog 代码，对于代码编写可能还比较陌生。

# 浙江大学

## 本科实验报告

课程名称: 数字逻辑电路设计

姓 名: 金祺书

学 院: 计算机科学与技术学院

专 业: 计算机科学与技术

指导教师: 洪奇军

报告日期: 2024 年 4 月 11 日

# 浙江大学实验报告

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 七段数码管显示译码器设计与应用

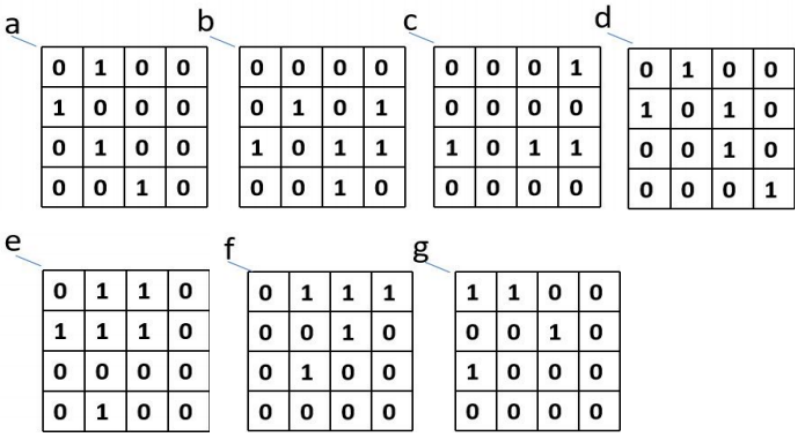
学生姓名： 金祺书 学号： 3230104248 同组学生姓名： 蒋翼泽

实验地点： 紫金港东四 511 室 实验日期： 2024 年 4 月 11 日

## 一、操作方法与实验步骤

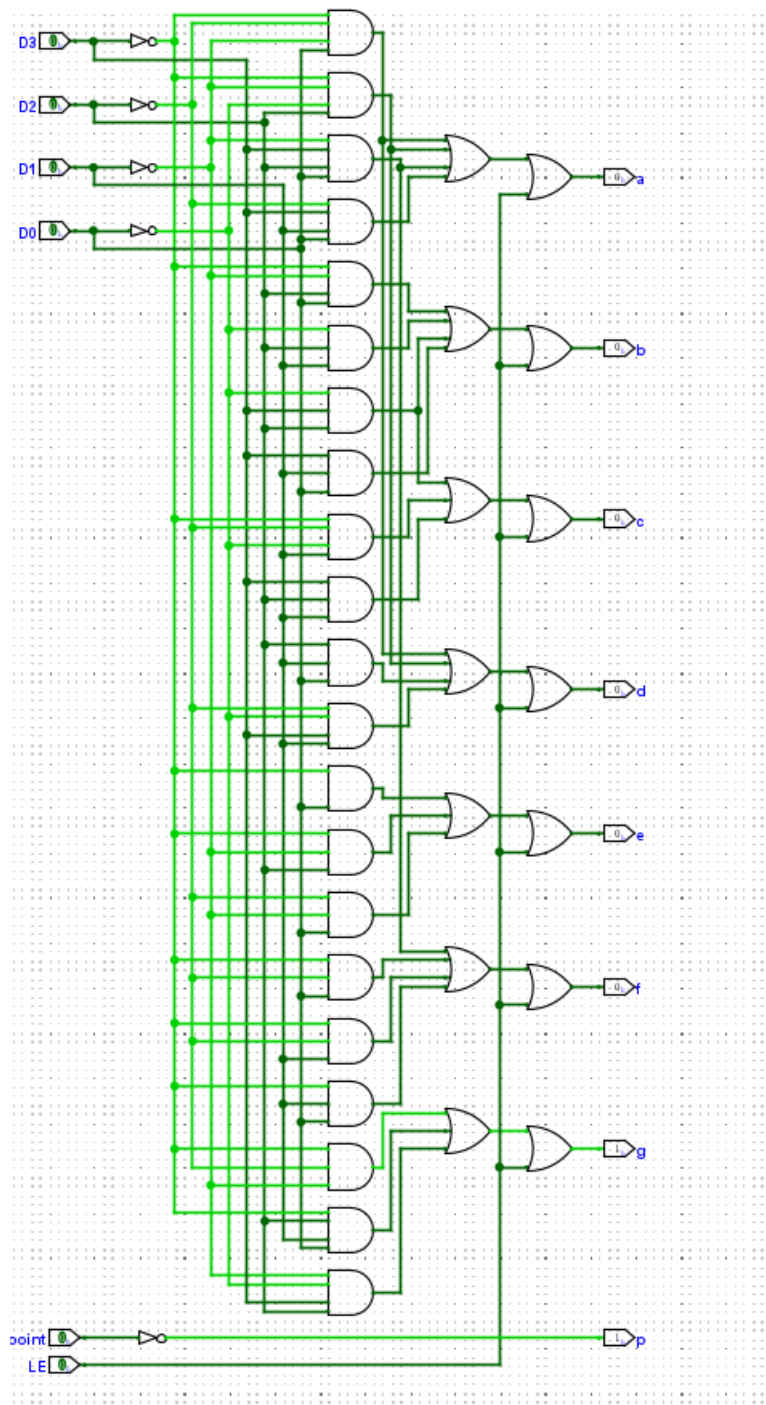
### ①原理图设计实现 MyMC14495 模块

- (1) 设计目标：利用该模块能把输入的四位二进制信号转换为对应的数码管亮暗信号输出
- (2) 利用卡诺图设计逻辑关系



- (3) 使用 Logisim 绘制 MyMC14485 的原理图，并将其转换成 Verilog Code





(4) Vivado 新建工程，对该模块进行仿真激励，检验该模块功能是否正确，仿真激励代码如下：

```
01 `timescale 1ns / 1ps
02
    //////////////////////////////////////
    //////////////////////////////////////
03 // Company:
04 // Engineer:
05 //
06 // Create Date: 2024/04/03 15:16:55
```

```

07 // Design Name:
08 // Module Name: MyMC14495_tb
09 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20
    //////////////////////////////////////
////////////////////////////////////
21
22 module MyMC14495_tb();
23
24 // Inputs
25 reg D0;
26 reg D1;
27 reg D2;
28 reg D3;
29 reg LE;
30 reg point;
31
32 // Output
33 wire p;
34 wire a;
35 wire b;
36 wire c;
37 wire d;
38 wire e;
39 wire f;
40 wire g;
41
42 // Instantiate the UUT
43 MyMC14495 MC14495_inst (
44 .D0(D0),
45 .D1(D1),
46 .D2(D2),
47 .D3(D3),
48 .LE(LE),

```

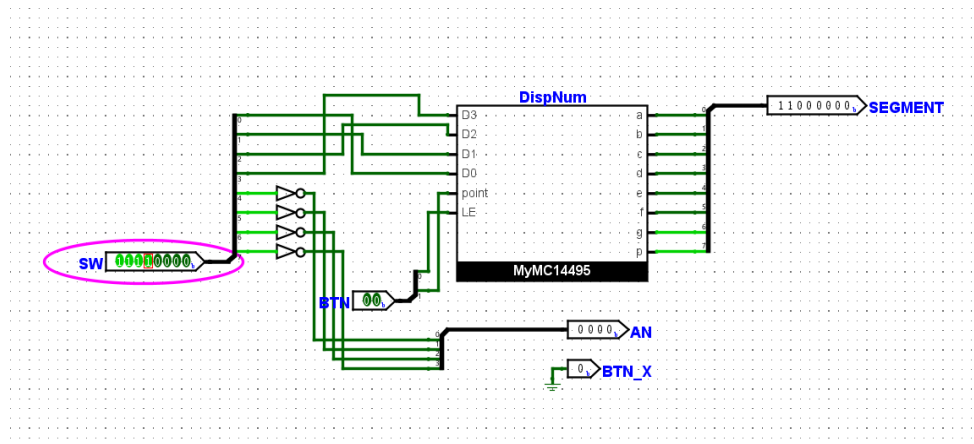
```

49 .point(point),
50 .p(p),
51 .a(a),
52 .b(b),
53 .c(c),
54 .d(d),
55 .e(e),
56 .f(f),
57 .g(g)
58 );
59
60 integer i;
61 initial begin
62     D3 = 0;
63     D2 = 0;
64     D1 = 0;
65     D0 = 0;
66     LE = 0;
67     point = 0;
68
69     for(i=0; i<=15;i=i+1) begin
70         {D3,D2,D1,D0}=i;
71         point=i;#50;
72     end
73
74     #50;
75     LE = 1;
76 end
77 endmodule

```

## ②简单使用

(1) 对我们刚刚实现的 MyMC14495 简单封装为 “DispNum” 模块

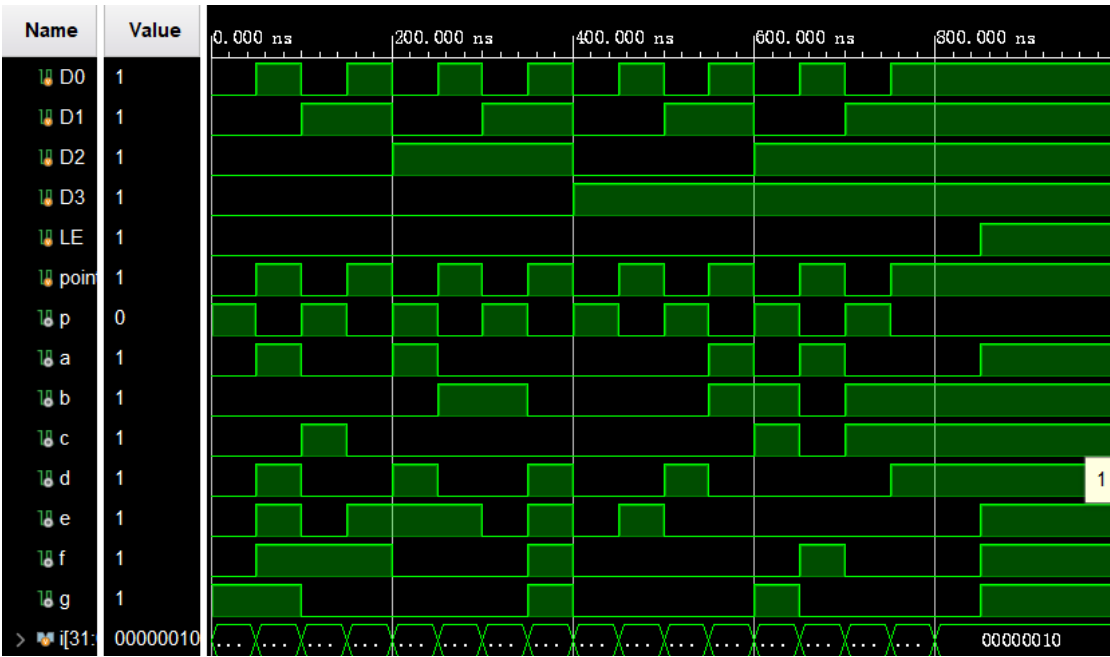


(2) 导出 Verilog 代码，通过约束文件生成比特流下板验证。

## 二、实验结果与分析

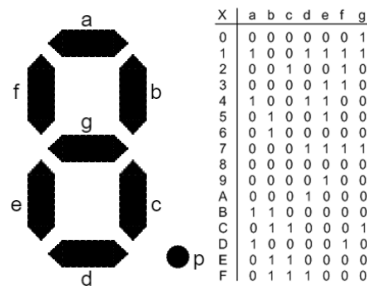
### ①原理图设计实现 MyMC14495 模块

导入 Vivado 后的仿真界面截图如下：



MC14495 模块有六个输入，八个输出，其中：


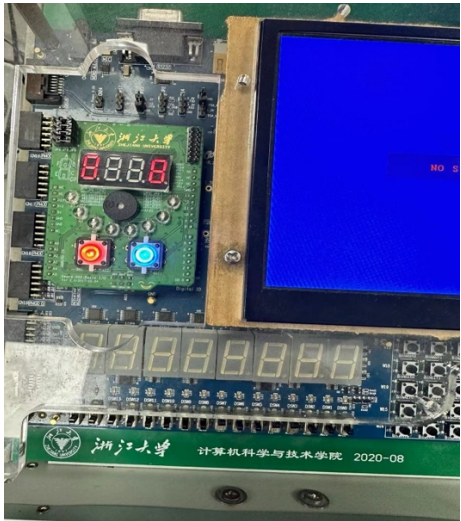

- (1) D3~D0: 输入的 4 位二进制数字
- (2) LE: 使能信号，低电平有效
- (3) point: 用来表示小数点是否点亮，高电平有效
- (4) 输出信号 a~g, p: 均为低电平有效，对应数码管以及输入对应真值表如下图：



以输入{D<sub>0</sub>,D<sub>1</sub>,D<sub>2</sub>,D<sub>3</sub>}={1,0,0,0}, LE=0, point=1 为例，即输入数字为 1（带小数点），此时 {a,b,c,d,e,f,g,p}={0,1,0,0,1,1,1,1}，根据数码管对应图可知输出正确。

### ②简单使用

SW(3:0)	SW(7:4)	效果图
---------	---------	-----

1001	0000	
1001	1000	
1001	1111	

### ③ "I Love YOU" decoder

(1) 真值表如下:

$D_2D_1D_0$	a	b	c	d	e	f	g
0	1	1	1	1	0	0	1
1	1	1	1	0	0	0	1
2	1	1	0	0	0	1	0
3	1	1	0	0	0	1	1
4	0	1	1	0	0	0	0
5	1	0	0	0	1	0	0
6	0	0	0	0	0	0	1
7	1	0	0	0	0	0	1

(2) a,b,c,g 的卡诺图分别如下:

1	1	1	1
0	1	1	0

(a)

1	1	1	1
1	0	0	0

(b)

1	1	0	0
1	0	0	0

(c)

1	1	1	0
0	0	1	1

(g)

则  $a = \overline{D_2} + D_0$ ,  $b = \overline{D_2} + \overline{D_1} \overline{D_0}$ ,  $c = \overline{D_1} \overline{D_0} + \overline{D_1} \overline{D_2}$ ,  $g = \overline{D_1} \overline{D_2} + D_1 D_0 + D_1 D_2$

### 三、讨论、心得

本次实验搭建电路，进行仿真的过程都还算顺利，但是在最后上板操作的时候却往往只能显示小数点而其他数码管无法发光，最后发现是对约束文件的理解不够透彻，照搬照抄导致无法达到预期效果，改正过后也得到了预期的效果。接下来应当自己好好编写属于自己的约束文件，对网站上的代码过于依赖；同时对 Logisim 的依赖过高，也应当学着用 Verilog 写代码而不是只画原理图。

# 浙江大学

## 本科实验报告

课程名称: 数字逻辑电路设计

姓 名: 金祺书

学 院: 计算机科学与技术学院

专 业: 计算机科学与技术

指导教师: 洪奇军

报告日期: 2024 年 4 月 18 日

# 浙江大学实验报告

课程名称： 数字逻辑设计 实验类型： 综合

实验项目名称： 多路选择器设计与应用

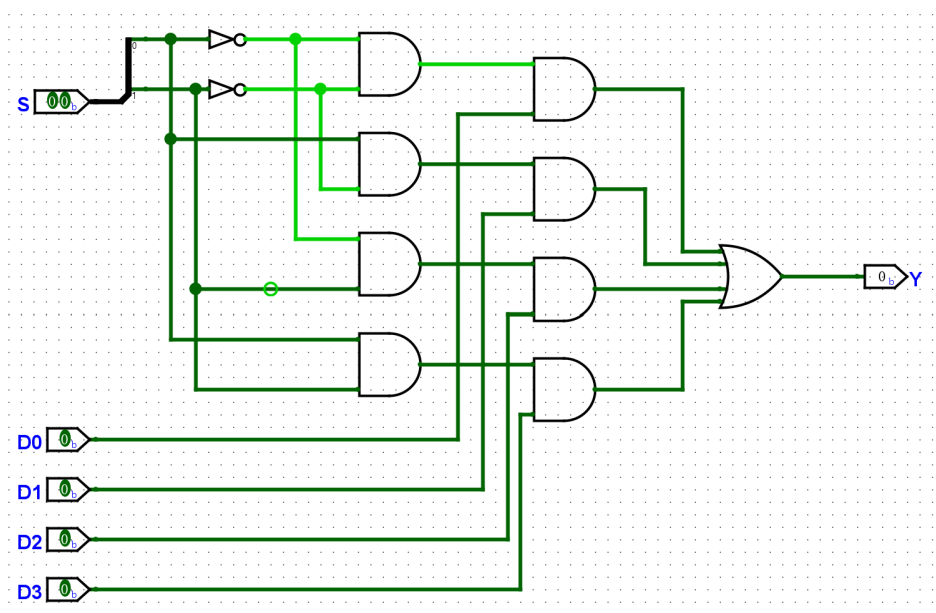
学生姓名： 金祺书 学号： 3230104248 同组学生姓名： 蒋翼泽

实验地点： 紫金港东四 511 室 实验日期： 2024 年 4 月 18 日

## 一、操作方法与实验步骤

### ①一位 4-1 多路选择器的设计

(1) 使用 Logisim 绘制原理图设计 Mux4to1 模块

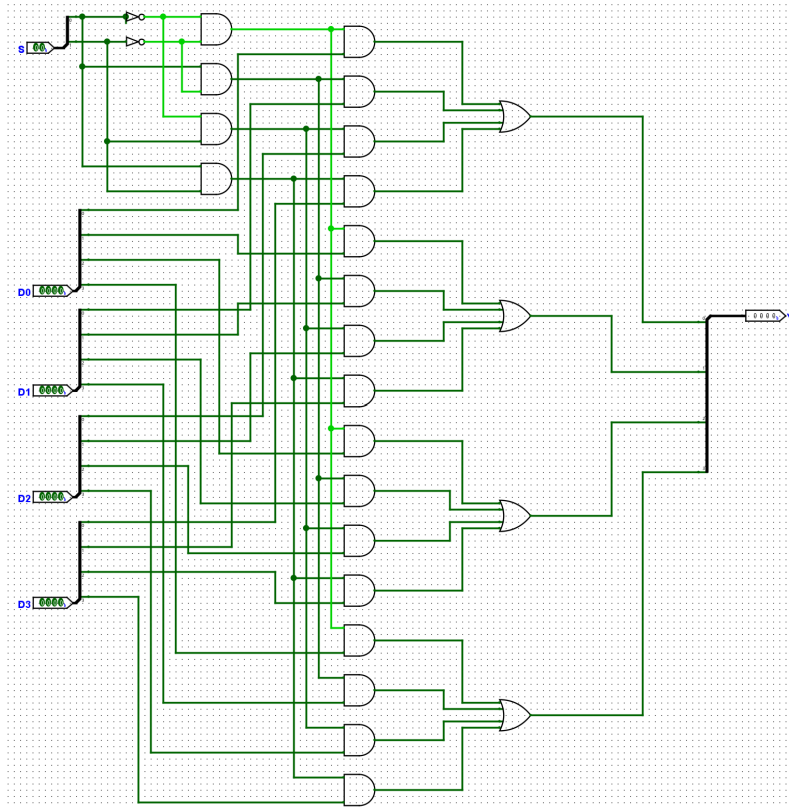


(2) 利用原理图生成 Verilog 代码

### ②四位 4-1 多路选择器的设计

(1) 使用 Logisim 绘制原理图设计 Mux4to1b4 模块





(2) 利用原理图生成 Verilog 代码

### ③对四位 4-1 多路选择器进行仿真

Vivado 新建工程，对 Mux4to1b4 模块进行仿真激励，检验该模块功能是否正确，仿真激励代码如下：

```
01 `timescale 1ns / 1ps
02
    ///////////////////////////////////////////////////
    ///////////////////////////////////
03 // Company:
04 // Engineer:
05 //
06 // Create Date: 2024/04/17 20:22:41
07 // Design Name:
08 // Module Name: Mux4to1b4_tb
09 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
```

```

18 // Additional Comments:
19 //
20
21 ///////////////////////////////////////////////////////////////////
22 ///////////////////////////////////////////////////////////////////
23 module Mux4to1b4_tb();
24 //Inputs
25     reg[1:0] S;
26     reg[3:0] D0;
27     reg[3:0] D1;
28     reg[3:0] D2;
29     reg[3:0] D3;
30 //Output
31     wire[3:0] Y;
32 //Instantiate the UUT
33     Mux4to1b4 Mux4to1b4_inst(
34         .S(S),
35         .D0(D0),
36         .D1(D1),
37         .D2(D2),
38         .D3(D3),
39         .Y(Y)
40     );
41     integer i;
42     initial begin
43         S=00;
44         D0=4'b0001;
45         D1=4'b0010;
46         D2=4'b0100;
47         D3=4'b1000;
48         for(i=0;i<=3;i=i+1) begin
49             {S}=i; #50;
50         end
51     end
52 endmodule

```

## ⑤计分板应用设计

(1) 实现一个简单的时钟分频器，其输出在每个时钟信号上升沿自增。复位信号为同步复位，当时钟信号的正边沿到来且复位信号为有效时（本实验中复位信号为高电平有效）进行复位。使用 Verilog 代码实现 clkdiv 模块如下：

```

01 `timescale 1ns / 1ps
02

```

```

////////////////////////////////////
////////////////////////////////////
03 // Company:
04 // Engineer:
05 //
06 // Create Date: 2024/04/18 08:00:19
07 // Design Name:
08 // Module Name: clkdiv
09 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20
////////////////////////////////////
////////////////////////////////////
21
22
23 module clkdiv(
24     input          clk,
25     input          rst, // Active-high
26     output reg [31:0] div_res
27 );
28
29     always @(posedge clk) begin // When postive edge of `clk`
comes
30         if(rst == 1'b1) begin
31             div_res <= 32'b0;
32         end else begin
33             div_res <= div_res + 32'b1; // Increase `div_res` by 1
34         end
35     end
36
37 endmodule
38
39

```

(1) 使用 Verilog 代码实现 CreateNumber 模块

```
01 `timescale 1ns / 1ps
```

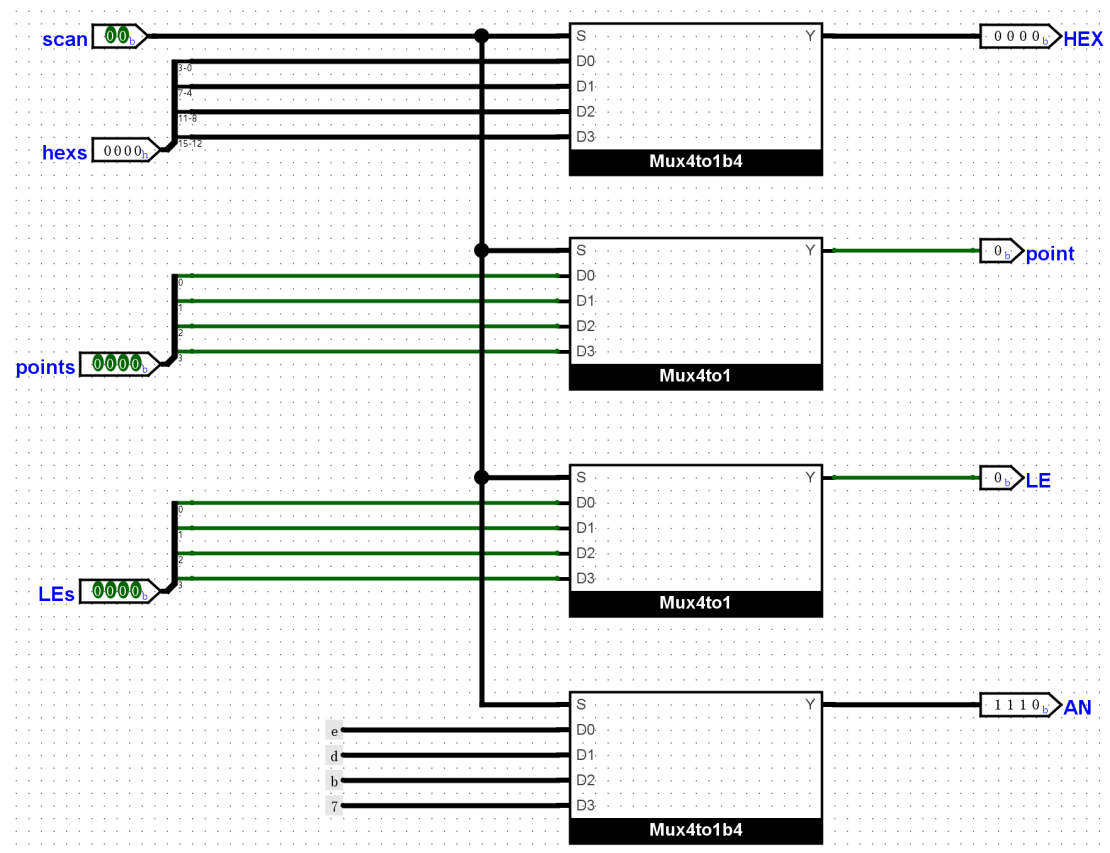
```

02
    //////////////////////////////////////
    //////////////////////////////////////
03 // Company:
04 // Engineer:
05 //
06 // Create Date: 2024/04/18 08:37:19
07 // Design Name:
08 // Module Name: CreateNumber
09 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20
    //////////////////////////////////////
    //////////////////////////////////////
21
22
23 module CreateNumber(
24     input [3:0]      btn,
25     output reg [15:0] num
26 );
27
28     wire [3:0] A, B, C, D;
29
30     initial num <= 16'b1010_1011_1100_1101;
31
32     assign A = num[15:12] + 4'b1;
33     assign B = num[11: 8] + 4'b1;
34     assign C = num[ 7: 4] + 4'b1;
35     assign D = num[ 3: 0] + 4'b1;
36
37     always @(posedge btn[0]) num[15:12] <= A;
38     always @(posedge btn[1]) num[11: 8] <= B;
39     always @(posedge btn[2]) num[ 7: 4] <= C;
40     always @(posedge btn[3]) num[ 3: 0] <= D;
41

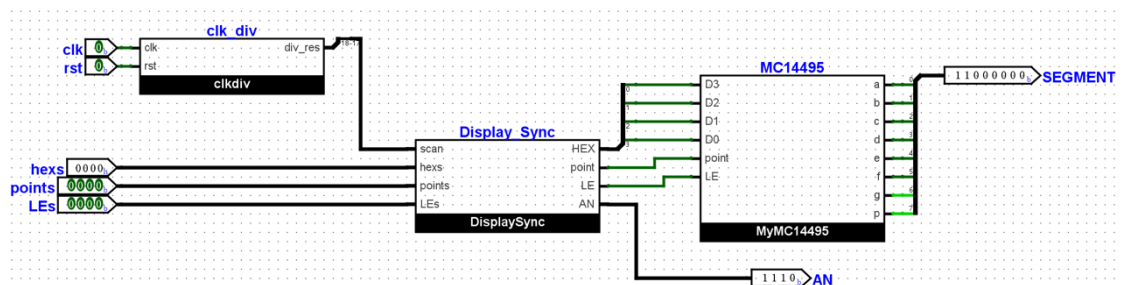
```

## 4.2 endmodule

(2) 设计动态扫描模块的子模块 DisplaySync，实现数字与使能信号的选择逻辑，使用 Logisim 绘制原理图设计 DisplaySync 模块如下：



(3) 设计动态扫描模块 DisplayNumber，使用时钟分频器获得合适的扫描信号，选用 clkdiv[18:17] 两位信号。使用刚刚编写完成的 DisplaySync 模块选择合适的数字和使能信号，并将四位数字和小数点控制信号输出到 MyMC14495 模块实例中得到七段信息。使用 Logisim 绘制原理图设计 DisplayNumber 模块如下：



(4) 使用 verilog 代码实现顶层模块，将使用之前完成的所有模块作为子模块。实现一个“计分板”应用，在 Arduino 上七段数码管查看数字结果，使用开关控制亮灭与小数点，使用四个按钮为数字实现自增。

```
01 module top(
02     input clk,
03     input [7:0] SW,
04     input [3:0] btn,
05     output [3:0] AN,
06     output [7:0] SEGMENT,
```

```

07     output      BTN_X
08 );
09
10     assign BTN_X = 1'b0;
11
12     wire [15:0] num;
13
14     CreateNumber create_inst(
15         .btn(btn),
16         .num(num)
17     );
18
19     DisplayNumber disp_inst(
20         .clk(clk),
21         .rst(1'b0),
22         .hexs(num),
23         .points(SW[7:4]),
24         .LEs(SW[3:0]),
25         .AN(AN),
26         .SEGMENT(SEGMENT)
27     );
28
29 endmodule

```

#### (5) 引脚约束

```

001 # Filename: constraints_lab7.xdc
002 ## Constraints file for Lab7
003
004 # Main clock
005 set_property PACKAGE_PIN AC18 [get_ports clk]
006 set_property IOSTANDARD LVCMOS18 [get_ports clk]
007
008 create_clock -period 10.000 -name clk [get_ports "clk"]
009
010 # Switches as inputs
011 set_property PACKAGE_PIN AA10 [get_ports {SW[0]}]
012 set_property PACKAGE_PIN AB10 [get_ports {SW[1]}]
013 set_property PACKAGE_PIN AA13 [get_ports {SW[2]}]
014 set_property PACKAGE_PIN AA12 [get_ports {SW[3]}]
015 set_property PACKAGE_PIN Y13 [get_ports {SW[4]}]
016 set_property PACKAGE_PIN Y12 [get_ports {SW[5]}]
017 set_property PACKAGE_PIN AD11 [get_ports {SW[6]}]
018 set_property PACKAGE_PIN AD10 [get_ports {SW[7]}]
019 set_property IOSTANDARD LVCMOS15 [get_ports {SW[0]}]
020 set_property IOSTANDARD LVCMOS15 [get_ports {SW[1]}]

```

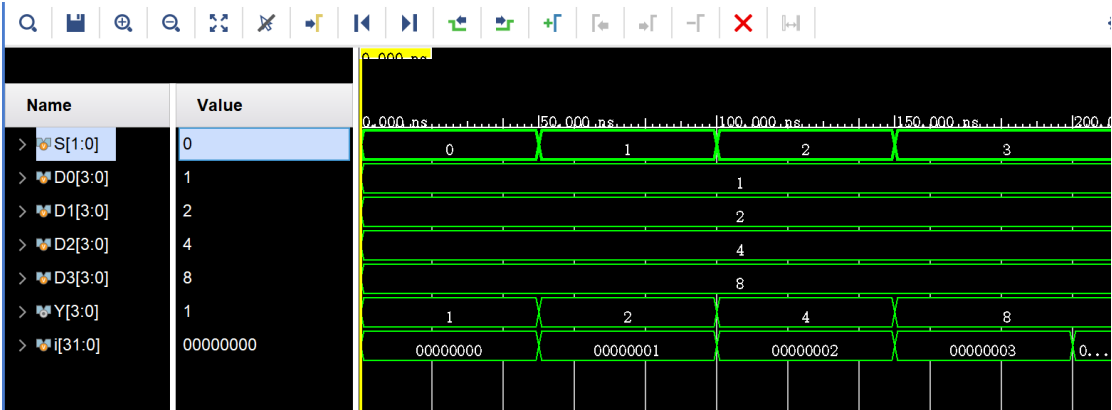
```
021 set_property IOSTANDARD LVCMOS15 [get_ports {SW[2]}]
022 set_property IOSTANDARD LVCMOS15 [get_ports {SW[3]}]
023 set_property IOSTANDARD LVCMOS15 [get_ports {SW[4]}]
024 set_property IOSTANDARD LVCMOS15 [get_ports {SW[5]}]
025 set_property IOSTANDARD LVCMOS15 [get_ports {SW[6]}]
026 set_property IOSTANDARD LVCMOS15 [get_ports {SW[7]}]
027
028 # Key as inputs
029 set_property PACKAGE_PIN W16 [get_ports BTN_X]
030 set_property IOSTANDARD LVCMOS18 [get_ports BTN_X]
031 set_property PACKAGE_PIN V18 [get_ports {btn[3]}]
032 set_property IOSTANDARD LVCMOS18 [get_ports {btn[3]}]
033 set_property PACKAGE_PIN V19 [get_ports {btn[2]}]
034 set_property IOSTANDARD LVCMOS18 [get_ports {btn[2]}]
035 set_property PACKAGE_PIN V14 [get_ports {btn[1]}]
036 set_property IOSTANDARD LVCMOS18 [get_ports {btn[1]}]
037 set_property PACKAGE_PIN W14 [get_ports {btn[0]}]
038 set_property IOSTANDARD LVCMOS18 [get_ports {btn[0]}]
039
040 set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets btn*]
041
042 # Arduino-Segment & AN
043 set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
044 set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
045 set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
046 set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
047 set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
048 set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
049 set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
050 set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
051 set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
052 set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
053 set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
054 set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
055 set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
056 set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
057 set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
058 set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]
059 set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]
060 set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]
061 set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]
062 set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]
063 set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]
064 set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]
```

```
065 set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]
066 set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]
```

二、实验结果与分析

①对四位 4-1 多路选择器进行仿真

导入 Vivado 后的仿真界面截图如下：



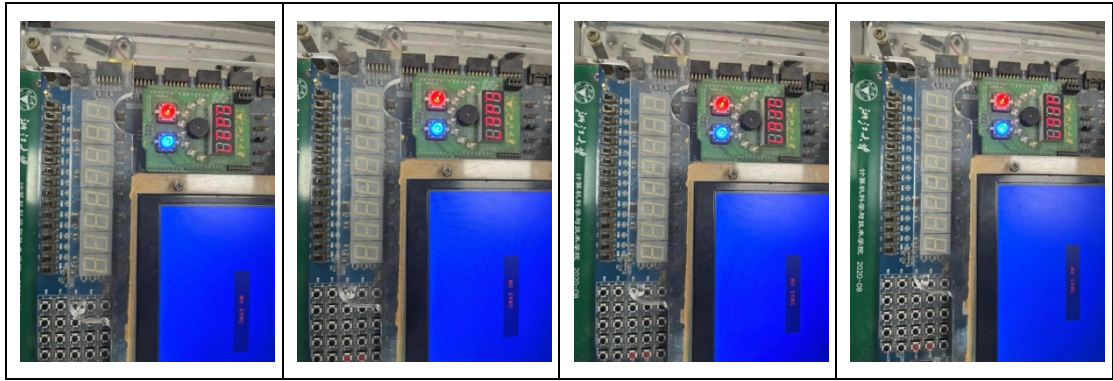
设置四个输入信号 D0,D1,D2,D3 分别为 0001, 0010, 0100, 1000（方便区分），S 为 00 时选择 D0，S 为 01 时选择 D1，S 为 10 时选择 D2，S 为 11 时选择 D3  
仿真结果能表现出该器件能正确选择输出信号。

②计分板应用设计

开关控制数码管和小数点的开关

按下按钮对应数码管示数增加





由于未设置防抖动，每次按按钮时对应数字可能会增加好几位，是正常现象

### 三、讨论、心得

本次实验过程较为顺利，但是在老师问答时钟模块与扫描信号的原理时没有答上来，说明对整个实验的原理还不够熟悉，还是应当先理解实验原理再去操作，不能照搬照抄。