

浙江大学



计算机逻辑设计基础

Lab 7

Author: 苏煜程

Student ID: 3220105481

Date: 2023 年 11 月 10 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：多路选择器设计及应用

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 11 月 2 日

1 实验目的

1. 掌握数据选择器的工作原理和逻辑功能
2. 掌握数据选择器的使用方法
3. 掌握 4 位数码管扫描显示方法
4. 4 位数码管显示应用 —— 记分板设计

2 实验任务

- 任务 1：数据选择器设计
- 任务 2：记分板设计

3 实验原理

3.1 数据选择器设计

3.1.1 四选一多路选择器 MUX4to1

1. 根据事件简化真值表
2. 输出是控制信号全部最小项与或结构

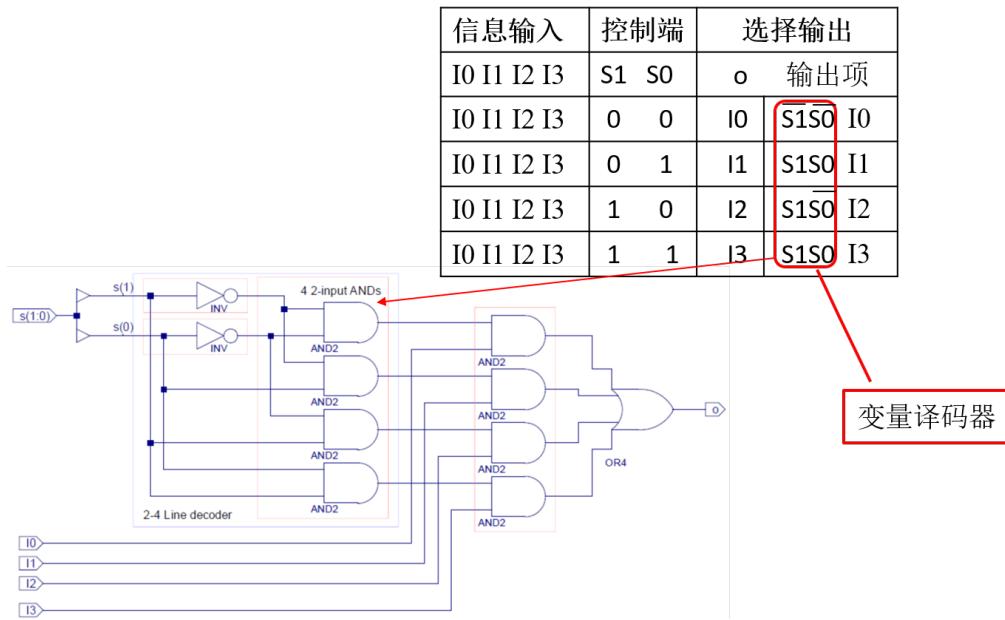


图 1: MUX4to1 线路与真值表

3.1.2 4 位四选一扩展 MUX4to1b4

控制结构不变，每路输入向量化

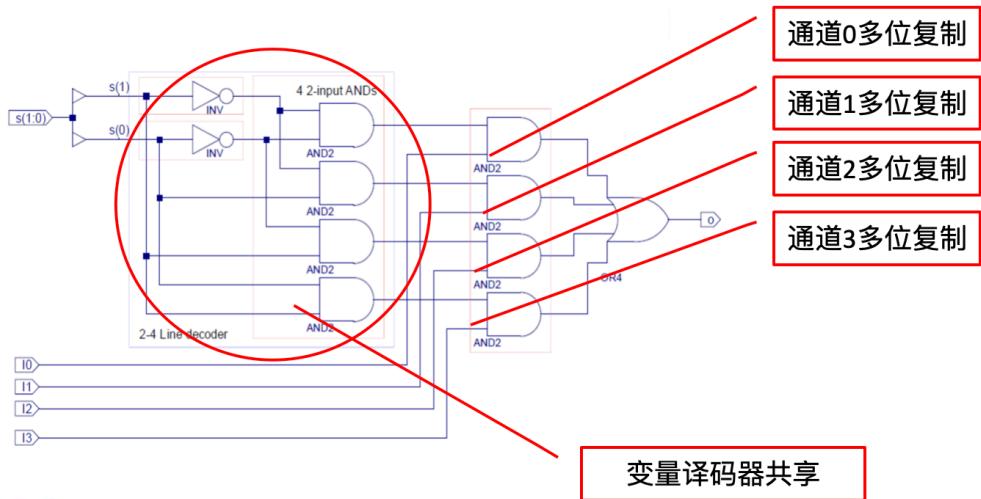


图 2: 多路选择器位扩展

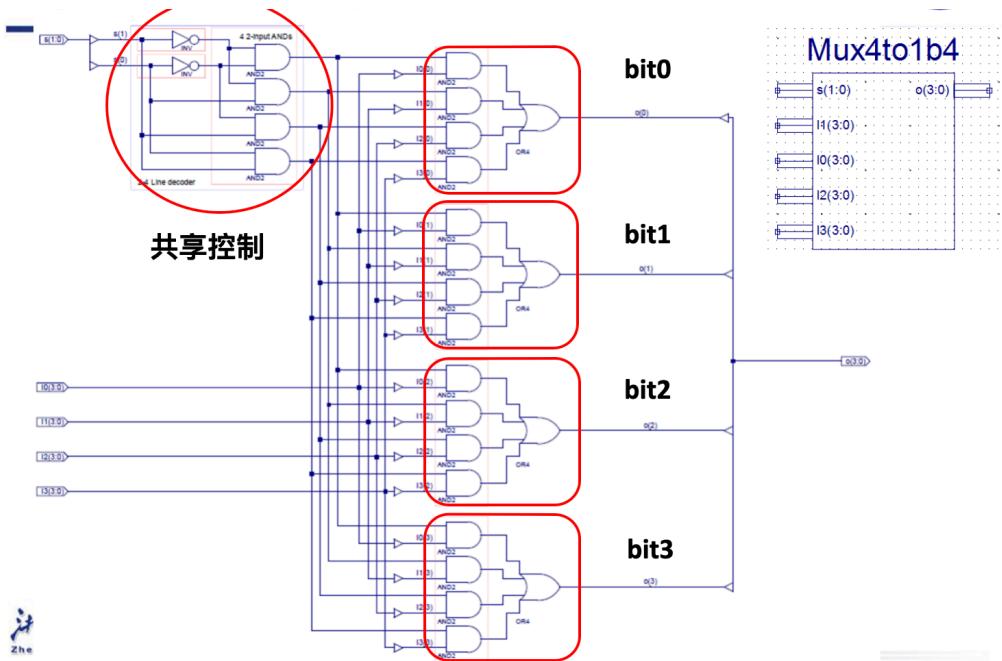


图 3: MUX4to1b4

3.2 4 位 7 段数码管动态扫描显示

1. 扫描信号来自时钟计数分频器：时序转化为组合电路
2. 由板载时钟 $\text{clk}(100\text{MHz})$ 作为计数器时钟，分频后的高两位信号 $\text{clk_div}[18:17]$ 作为扫描控制信号 $\text{Scan}[1:0]$ ，其数据为从 0、1、2、3、0、…，输入 2-4 译码器产生数码管位选信号，控制哪个数码管显示（位选择），同时输入 4 选 1 多路复用器选择需要显示哪个数据（段码选择）
3. 计数器的分频系数要适当，几 ms 切换一次，眼睛舒适即可

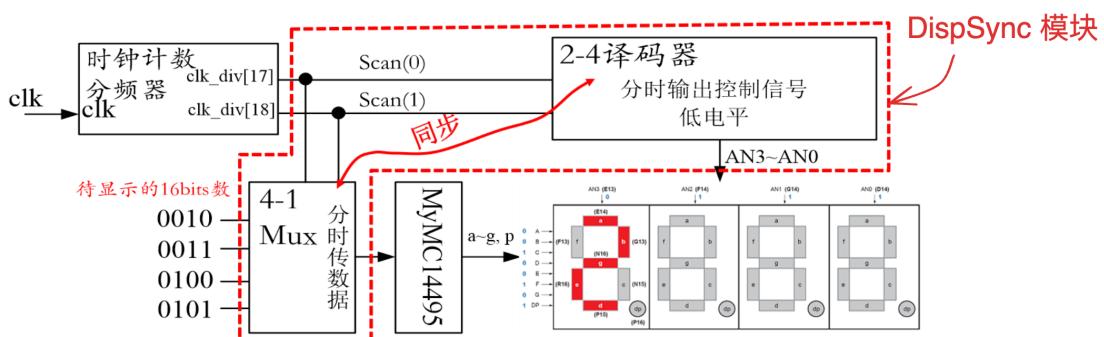


图 4: 动态扫描模块 DispNum

2-4 译码器实现位扫描控制

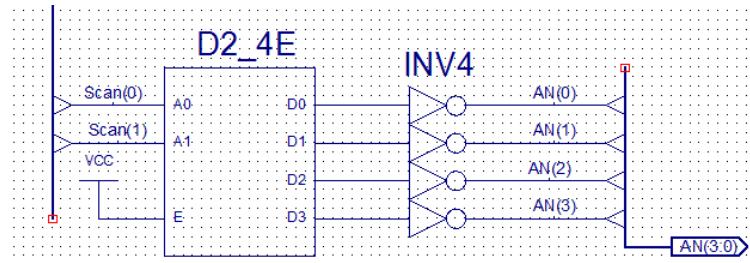


图 5: 分时输出控制信号

多路选择器实现段码选择

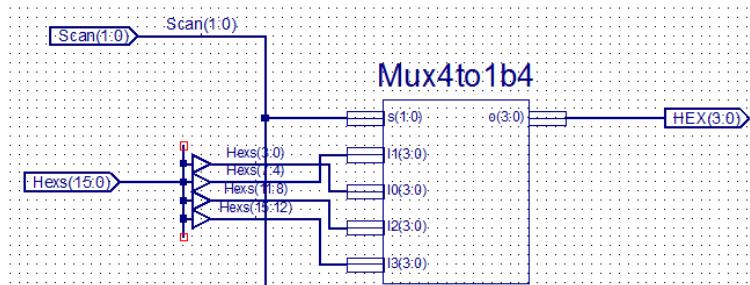


图 6: 分时传数据

小数点与消隐选择 包括在图 4 的“4-1 Mux”模块中。

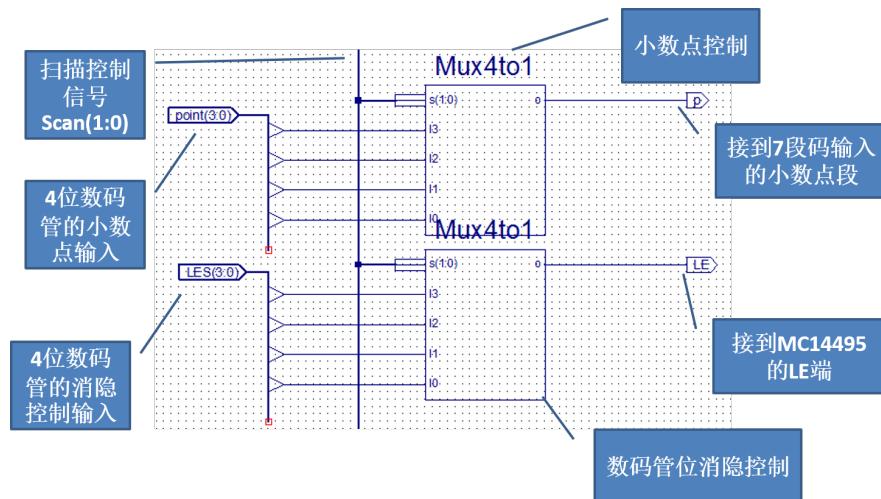


图 7: 小数点与消隐

3.2 位时钟计数分频器 clkdiv

1. 可输出 $2 \sim 2^{32}$ 分频信号，可用于一般非同步类时钟信号
2. 延时较高，要求不高的时钟也可以用
3. 本实验中用 `clkdiv(18:17)` 作为扫描控制信号，控制 4 位数码管的动态扫描，每一位显示切换时间为 $217/100M = 1.3ms$

4 实验内容与步骤

4.1 任务 1：数据选择器设计

1. 新建工程，工程名称用 `Mux4to1b4_sch`。
2. 新建源文件，类型是 Schematic，文件名称用 `Mux4to1b4`。
3. 原理图方式设计 4 位 4 选 1 数据选择器

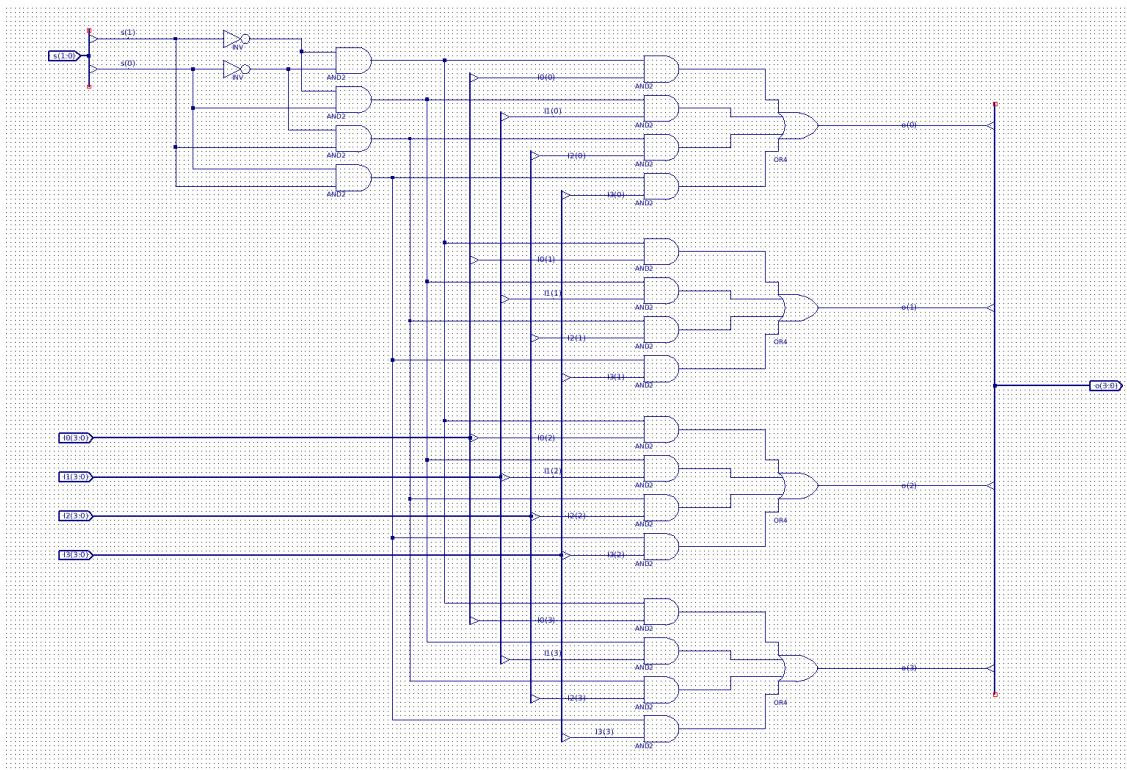


图 8: Mux4to1b4

数据选择器仿真 使用如下仿真激励代码进行仿真：

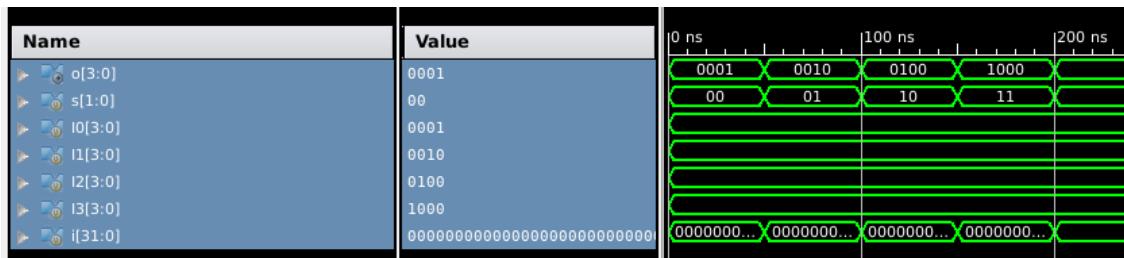
```
1 integer i;
2 initial begin
```

```

3   I0 = 'b0001;
4   I1 = 'b0010;
5   I2 = 'b0100;
6   I3 = 'b1000;
7   for (i = 0; i < 4; i = i + 1) begin
8     s = i;
9     #50;
10    end
11    s = 0;
12 end

```

得到仿真波形如下：



可见输入不同的 $s(1:0)$ ，能够选择输出对应的数据。

4.2 任务 2：记分板设计

4.2.1 子任务 1：实现 4 位 7 段数码管动态扫描显示

根据图 4、5、6、7 设计 DisplaySync 模块电路如下图

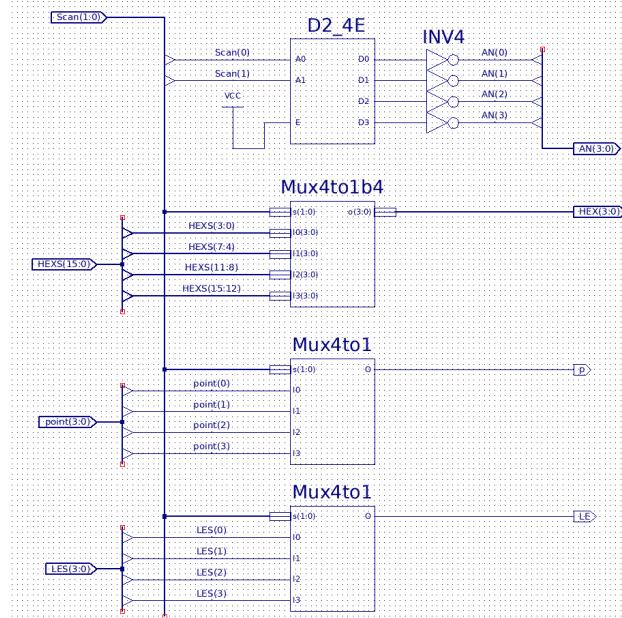


图 9: DisplaySync 模块

也可以使用 HDL 实现：

```
1 `timescale 1ns / 1ps
2 module DisplaySync(
3     input [15:0] HEXS,
4     input [1:0] Scan,
5     input [3:0] point,
6     input [3:0] LES,
7     output reg[3:0] HEX,
8     output reg p, LE,
9     output reg[3:0] AN
10 );
11
12 always @* begin
13     case(Scan)
14         2'b00: begin
15             HEX <= HEXS[3:0];
16             AN <= 4'b1110;
17             p <= point[0];
18             LE <= LES[0];
19         end
20         2'b01: begin
21             HEX <= HEXS[7:4];
22             AN <= 4'b1101;
23             p <= point[1];
24             LE <= LES[1];
25         end
26         2'b10: begin
27             HEX <= HEXS[11:8];
28             AN <= 4'b1011;
29             p <= point[2];
30             LE <= LES[2];
31         end
32         2'b11: begin
33             HEX <= HEXS[15:12];
34             AN <= 4'b0111;
35             p <= point[3];
36             LE <= LES[3];
37         end
38     endcase
39 end
40 endmodule
```

使用 Verilog HDL 设计时钟计数分频器模块如下

```
1 `timescale 1ns / 1ps
2 module clkdiv(
```

```

3   input clk,
4   input rst,
5   output reg[31:0] clkdiv
6 );
7
8   always @ (posedge clk or posedge rst) begin
9     if (rst) clkdiv <= 0;
10    else clkdiv <= clkdiv + 1'b1;
11  end
12
13 endmodule

```

设计 CreateNumber 按键数据输入模块如下

```

1 `timescale 1ns / 1ps
2 module CreateNumber(
3   input wire[3:0] btn,
4   output reg[15:0] num
5 );
6
7   wire[3:0] A, B, C, D;
8
9   initial num <= 16'b1010_1011_1100_1101; // display "AbCd"
10
11  assign A = num[ 3: 0] + 4'd1;
12  assign B = num[ 7: 4] + 4'd1;
13  assign C = num[11: 8] + 4'd1;
14  assign D = num[15:12] + 4'd1;
15
16  always@ (posedge btn[0]) num[3:0] <= A;
17  always@ (posedge btn[1]) num[7:4] <= B;
18  always@ (posedge btn[2]) num[11:8] <= C;
19  always@ (posedge btn[3]) num[15:12] <= D;
20
21 endmodule

```

设计 DispNumber 模块电路如下图

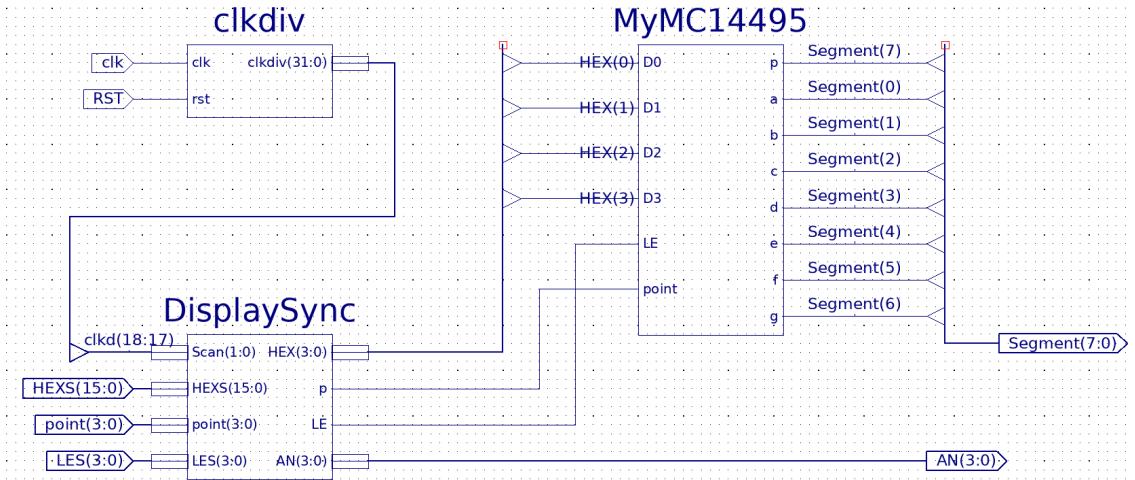


图 10: DispNumber 模块

4.2.2 子任务 2: 实现计分板功能

新建工程名为 ScoreBoard，类型为 HDL。将 CreateNumber 模块和 DispNumber 模块导入工程中。

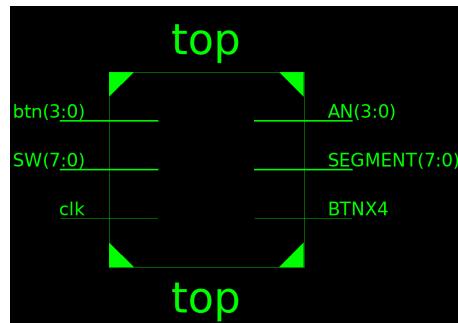
新建源文件 top，设为“Top Module”，写入如下 verilog 代码：

```

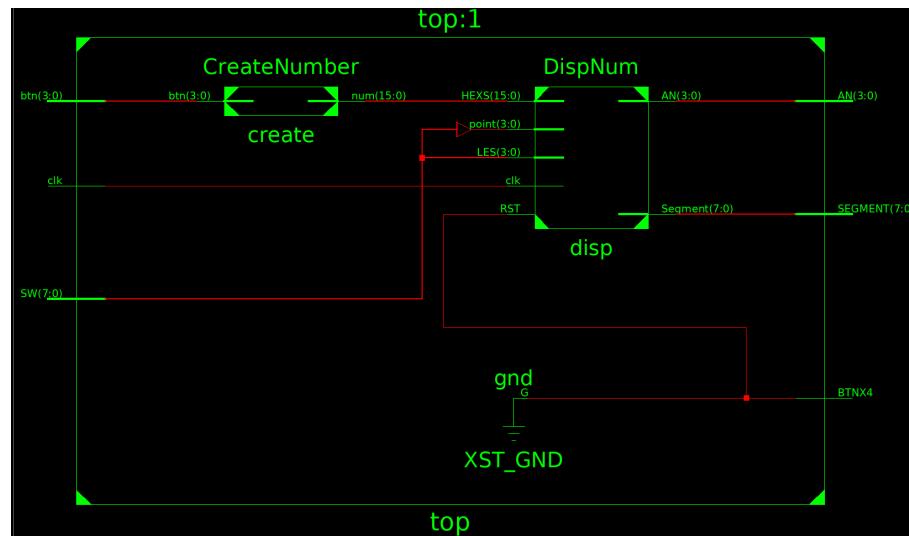
1 `timescale 1ns / 1ps
2 module top(
3     input wire clk,
4     input wire[7:0] SW,
5     input wire[3:0] btn,
6     output wire[3:0] AN,
7     output wire[7:0] SEGMENT,
8     output wire BTNX4
9 );
10
11     wire[15:0] num;
12
13     CreateNumber create(btn, num);
14     DispNum disp(clk, num, SW[7:4], SW[3:0], 1'b0, AN, SEGMENT);
15
16     assign BTNX4 = 1'b0;
17 endmodule

```

Synthesize XST → View RTL Schematic 查看顶层模块：



(a) 顶层模块



(b) 顶层模块内部结构

新建引脚约束文件 K7.ucf，写入如下内容：

```

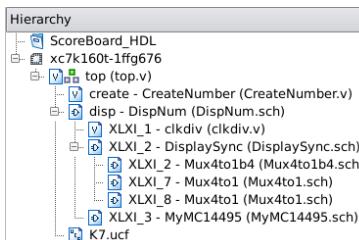
1 NET "clk"      LOC = AC18 | IOSTANDARD = LVCMOS18;
2
3 NET "SW[0]"    LOC = AA10 | IOSTANDARD = LVCMOS15;
4 NET "SW[1]"    LOC = AB10 | IOSTANDARD = LVCMOS15;
5 NET "SW[2]"    LOC = AA13 | IOSTANDARD = LVCMOS15;
6 NET "SW[3]"    LOC = AA12 | IOSTANDARD = LVCMOS15;
7 NET "SW[4]"    LOC = Y13 | IOSTANDARD = LVCMOS15;
8 NET "SW[5]"    LOC = Y12 | IOSTANDARD = LVCMOS15;
9 NET "SW[6]"    LOC = AD11 | IOSTANDARD = LVCMOS15;
10 NET "SW[7]"   LOC = AD10 | IOSTANDARD = LVCMOS15;
11
12 NET "btn[0]"   LOC = W14 | IOSTANDARD = LVCMOS18;
13 NET "btn[0]"   CLOCK_DEDICATED_ROUTE = FALSE;
14 NET "btn[1]"   LOC = V14 | IOSTANDARD = LVCMOS18;
15 NET "btn[1]"   CLOCK_DEDICATED_ROUTE = FALSE;
16 NET "btn[2]"   LOC = V19 | IOSTANDARD = LVCMOS18;
17 NET "btn[2]"   CLOCK_DEDICATED_ROUTE = FALSE;
18 NET "btn[3]"   LOC = V18 | IOSTANDARD = LVCMOS18;
19 NET "btn[3]"   CLOCK_DEDICATED_ROUTE = FALSE;
```

```

20 NET "BTNX4" LOC = W16 | IOSTANDARD = LVCMOS18;
21
22 NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;
23 NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;
24 NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;
25 NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;
26
27 NET "SEGMENT[0]" LOC = AB22 | IOSTANDARD = LVCMOS33;#a
28 NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;#b
29 NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;#c
30 NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;#d
31 NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;#e
32 NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;#f
33 NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;#g
34 NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;#point

```

随后将此前的模块导入工程，并 Generate Programming File 生成 bit 文件：



(a) 导入工程



(b) Generate Programming File

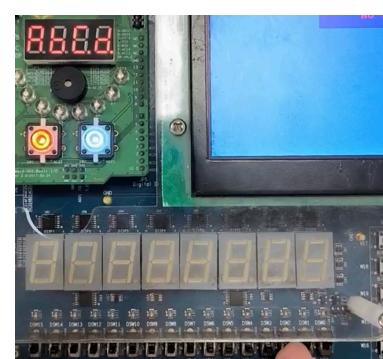
4.2.3 下载验证

根据电路原理图和引脚约束文件，从右往左八个开关分别对应 `points[0:3]`, `LE[0:3]`，最后一行的按钮从右往左分别对应 `btn[0:3]`

测试小数点显示 从右往左第一到四个开关分别控制从右往左四个小数点显示



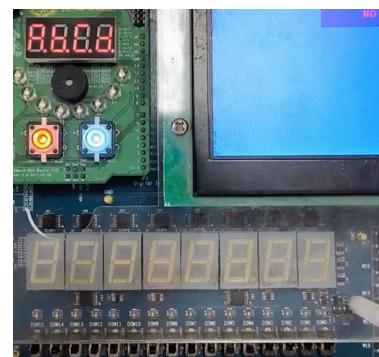
(a) point index = 0001



(b) point index = 0011



(a) point index = 0111



(b) point index = 1111

测试数码管消隐 从右往左第五到八个开关分别控制从右往左四个 LED 数码管的消隐



(a) LE index = 0001



(b) LE index = 0011



(c) LE index = 0111



(d) LE index = 1111

按钮控制 用 BTNX4Y[3:0] 这 4 个按钮，每个按钮按一次，对应的数码管的值应当加 1



(a) 按下第一个按钮之前（从左往右）



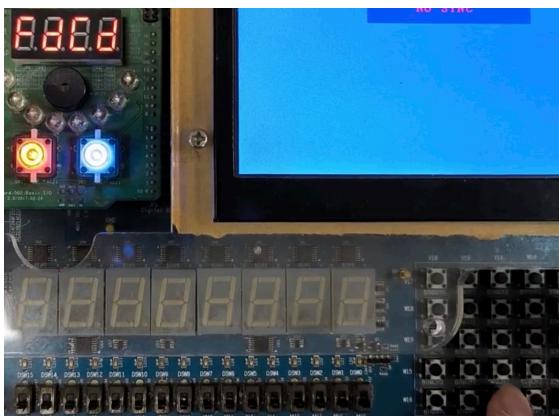
(b) 按下第一个按钮之后



(a) 按下第二个按钮之前



(b) 按下第二个按钮之后



(a) 按下第三个按钮之前



(b) 按下第三个按钮之后



(a) 按下第四个按钮之前



(b) 按下第四个按钮之后

5 实验结果分析

5.1 警告/错误分析

在 Generate Programming Files 操作时，会出现 warning 信息：

```
map      Place:1399 - A clock IOB / BUFGCTRL clock component pair have been found that are not placed at an optimal clock IOB / BUFGCTRL site pair. The clock IOB component <btn<3>> is placed at site <V18>. The corresponding BUFGCTRL component <btn_3_BUFGP/BUFG> is placed at site <BUFCTRL_XYOY>. The clock IO can use the fast path between the IOB and the Clock Buffer if the IOB is placed on a Clock Capable IOB site that has dedicated fast path to BUFGCTRL sites in its half of the device (TOP or BOTTOM). You may want to analyze why this problem exists and correct it. This is normally an ERROR but the CLOCK_DEDICATED_ROUTE constraint was applied on COMP_PIN <btn<3>-PAD> allowing your design to continue. This constraint disables all clock placer rules related to the specified COMP_PIN. The use of this override is highly discouraged as it may lead to very poor timing results. It is recommended that this error condition be corrected in the design.
```

警告指出 `btn` 的时钟 IOB 组件和 BUFGCTRL 组件没有被放置在最优的位置，可能导致时钟路径不够优化。

一般情况下 ISE 会进行报错，但对于本实验的按钮来说，按钮信号为非时钟信号，时序要求没有时钟那么严格。

我们在引脚约束文件中对四个按钮的引脚进行了 `CLOCK_DEDICATED_ROUTE = FALSE` 的约束，从而使得 ISE 只是发出警告，从而继续进行布线。

这样的设置可能会提高布线的灵活性，但可能会导致信号路径更长，延迟和不确定性可能会增加。对于按钮等输入设备，这可能是可以接受的，但对于时钟等时序关键信号可能不太适用。

5.2 下载验证结果分析

在电路设计中，我们期望的是按一下按钮，对应数字增加 1。但是实际上板过程中，按一下按钮对应数字可能会连续增加多次。

其原因是按钮存在机械抖动，导致按一次按钮产生多个上升沿：



在下一次实验中会解决这个问题。

6 讨论与心得

在本次实验中，我们进行了多个模块的设计与使用（不得不说 PPT 挺混乱的，实验原理与实验内容写在一块儿，并且也没有对各个模块进行很好的章节划分）。相较于前面几次实验，这次实验需要建立设计多个工程，更加复杂，我花了很多时间搞懂各个模块的用处和之间的关系，在自己的摸索和同学的讨论下成功完成了 bit 文件的生成，并且最终在上板中也没有出现严重的问题。