浙江大学

本科实验报告

课程名称：	数字逻辑设计
姓　　名：	
学　　院：	竺可桢学院
专　　业：	混合班
指导教师：	董亚波
报告日期：	2025 年 4 月 3 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：多路选择器设计及应用

学生姓名：__ 学号：___同组学生姓名：无

实验地点：紫金港东四 509 室 实验日期：2025 年 4 月 2 日

一、 实验目的

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

二、操作方法与实验步骤

1. 数据选择器设计

实验原理为：

- 4 选 1 多路选择器：Mux4to1。根据事件简化真值表，其输出是控制信号全部最小项与或结构。

信息输入	控制端	选择输出	
I0 I1 I2 I3	S1 S0	o	输出项
I0 I1 I2 I3	0 0	I0	S1S0 I0
I0 I1 I2 I3	0 1	I1	S1S0 I1
I0 I1 I2 I3	1 0	I2	S1S0 I2
I0 I1 I2 I3	1 1	I3	S1S0 I3

- 多路选择器的位扩展：可共享变量译码器，多位复制 4 个通道，使得结构不变，每路输入向量化。

实验步骤如下：

- 在 Digital 中新建电路，名称用 Mux4to1，并按照原理图设计 4 选 1 多路选择器。
- 对 4 选 1 多路选择器进行测试验证。
- 将 4 选 1 多路选择器进行位扩展，在 Digital 中新建电路，名称用 Mux4to1b4，并按照原理图设计 4 位 4 选 1 多路选择器。

- 对 4 位 4 选 1 多路选择器进行测试验证。

2. 4 位七段数码管扫描显示控制模块设计

实验原理为：

- 静态显示：每个 7 段码对应一个显示译码电路。
- 动态扫描显示：利用人眼视觉残留，将一个 7 段码译码电路分时为每个 7 段码提供译码。
- 控制时序：用定时计数信号控制公共极，分时输出对应七段码的显示信号。
- 4 位七段码结构：正极作为公共端，七段信号并联。

实验步骤如下：

- 在 Digital 中新建电路，名称用 DispNum，并按照原理图设计 4 位 7 段数码管扫描显示控制模块，需要用到 Mux4to1，Mux4to1b4，以及上节课设计的 MyMC14495。
- 在 Digital 中新建电路，名称用 DispNum_Test，用于测试 DispNum 电路的正确性。
- 正确性检验完成后，将 DispNum 电路导出为 DispNum.v。

3. 记分板设计

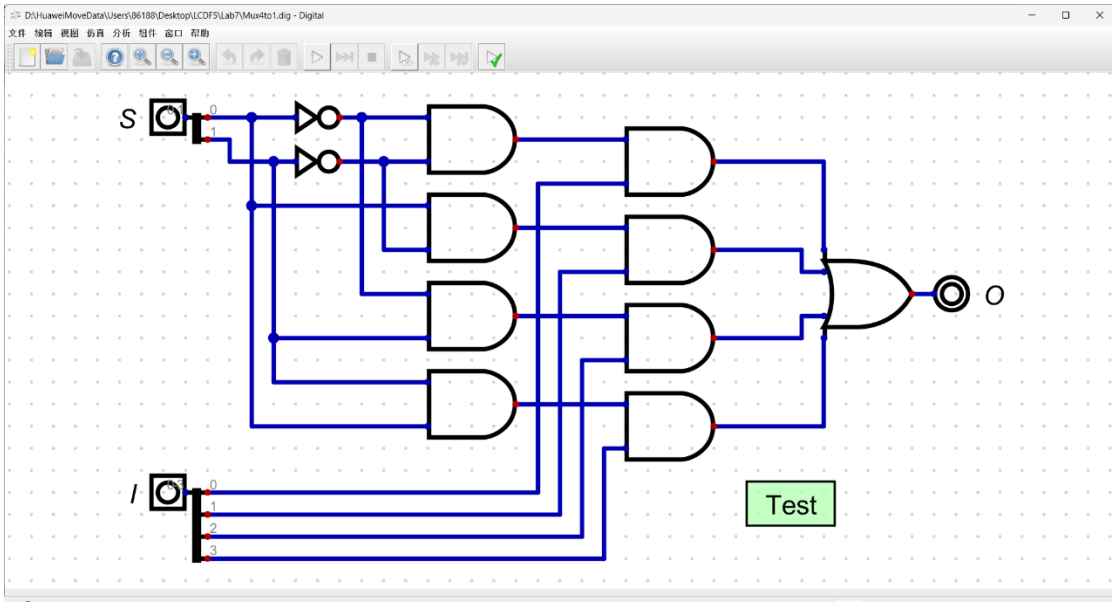
实验步骤如下：

- 在 Vivado 中新建工程 ScoreBoard，并导入 DispNum.v，创建 clkdiv.v 文件、CreateNumber.v 文件，以及 top.v 文件。
- 对 top 模块进行行为仿真。
- 综合下载运行该模块。

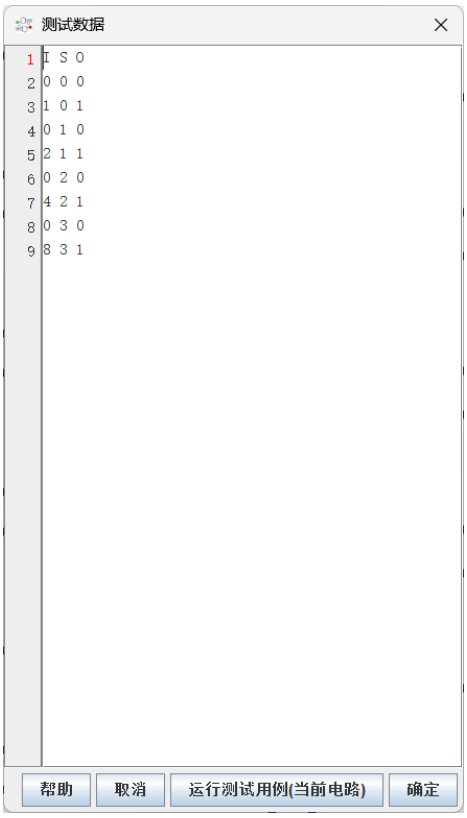
三、实验结果与分析

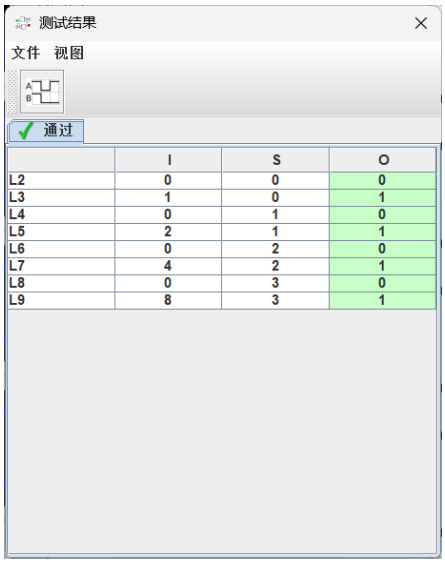
1. 数据选择器设计

按照原理图搭建的电路 Mux4to1 如图所示：

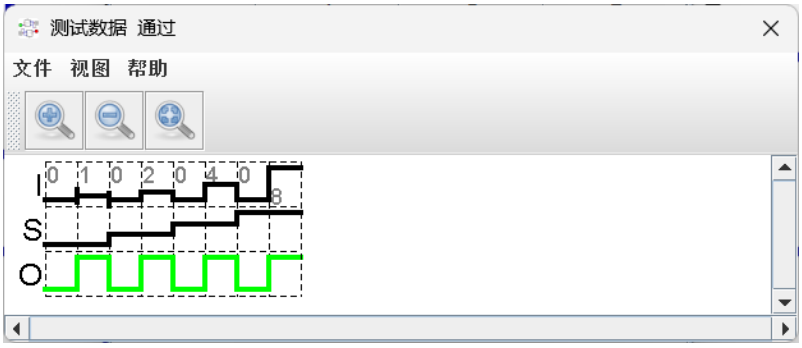


对该电路图进行正确性检验，结果如下（已加入预期结果）：



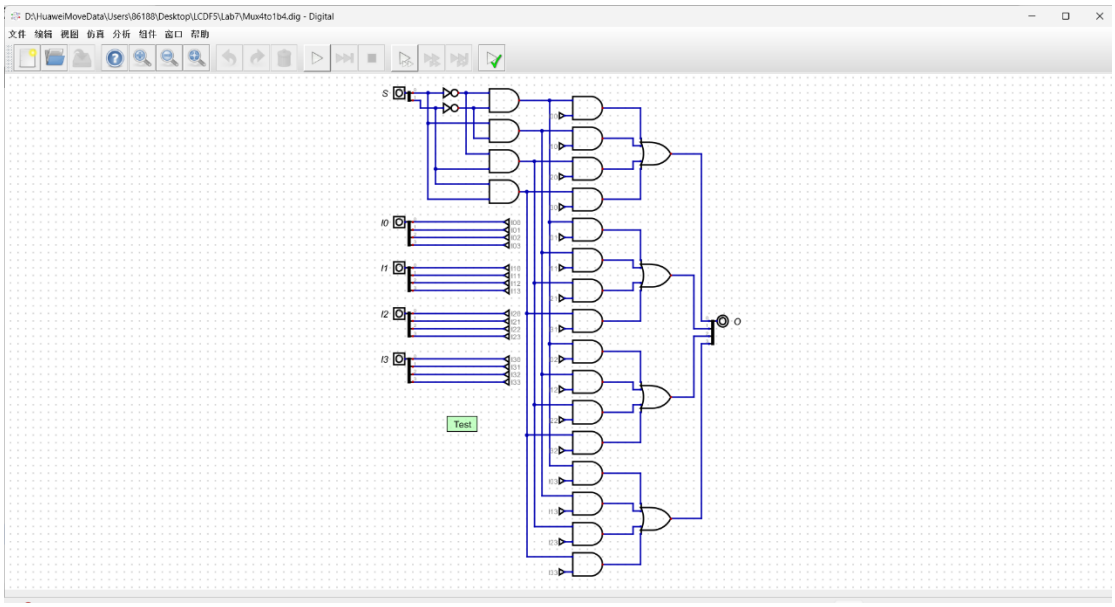


	I	S	O
L2	0	0	0
L3	1	0	1
L4	0	1	0
L5	2	1	1
L6	0	2	0
L7	4	2	1
L8	0	3	0
L9	8	3	1



其中，当输入信号显示 2^k 时，表明第 k 位的值为 1；而输出信号显示为 k 时，表明选择器选择了第 k 位，因此该输出符号预期。

接下来按照原理图搭建电路 Mux4to1b4：

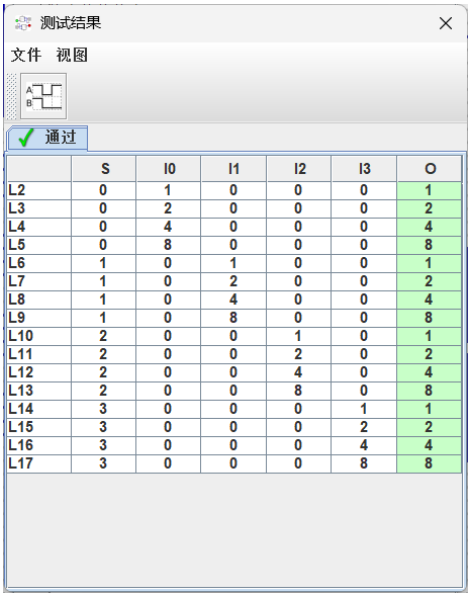


对该电路图进行正确性验证（已加入预期结果）：

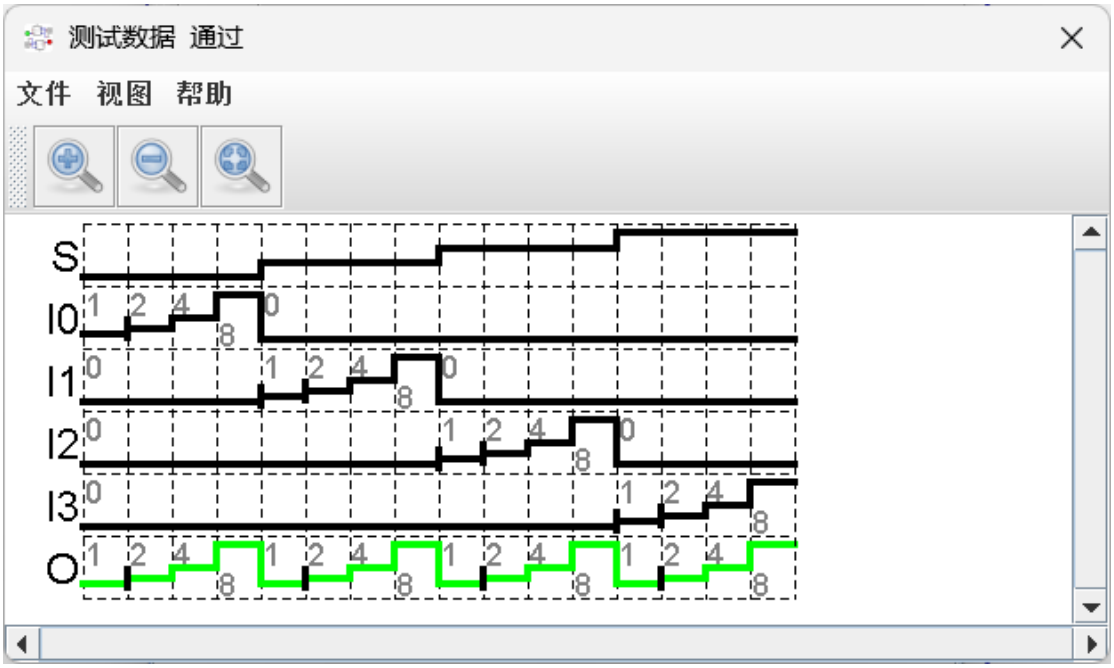
测试数据

	S	I0	I1	I2	I3	O
1	0	1	0	0	0	1
2	0	2	0	0	0	2
3	0	4	0	0	0	4
4	0	8	0	0	0	8
5	1	0	1	0	0	1
6	1	0	2	0	0	2
7	1	0	4	0	0	4
8	1	0	8	0	0	8
9	2	0	0	1	0	1
10	2	0	0	2	0	2
11	2	0	0	4	0	4
12	2	0	0	8	0	8
13	3	0	0	0	1	1
14	3	0	0	0	2	2
15	3	0	0	0	4	4
16	3	0	0	0	8	8

帮助　取消　运行测试用例(当前电路)　确定



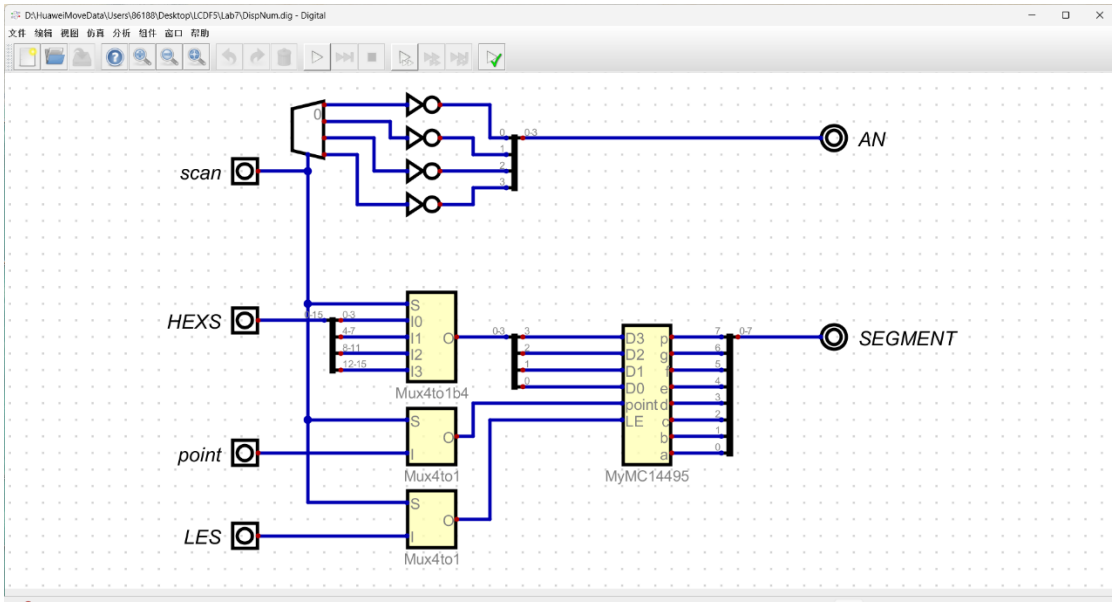
	S	I0	I1	I2	I3	O
L2	0	1	0	0	0	1
L3	0	2	0	0	0	2
L4	0	4	0	0	0	4
L5	0	8	0	0	0	8
L6	1	0	1	0	0	1
L7	1	0	2	0	0	2
L8	1	0	4	0	0	4
L9	1	0	8	0	0	8
L10	2	0	0	1	0	1
L11	2	0	0	2	0	2
L12	2	0	0	4	0	4
L13	2	0	0	8	0	8
L14	3	0	0	0	1	1
L15	3	0	0	0	2	2
L16	3	0	0	0	4	4
L17	3	0	0	0	8	8



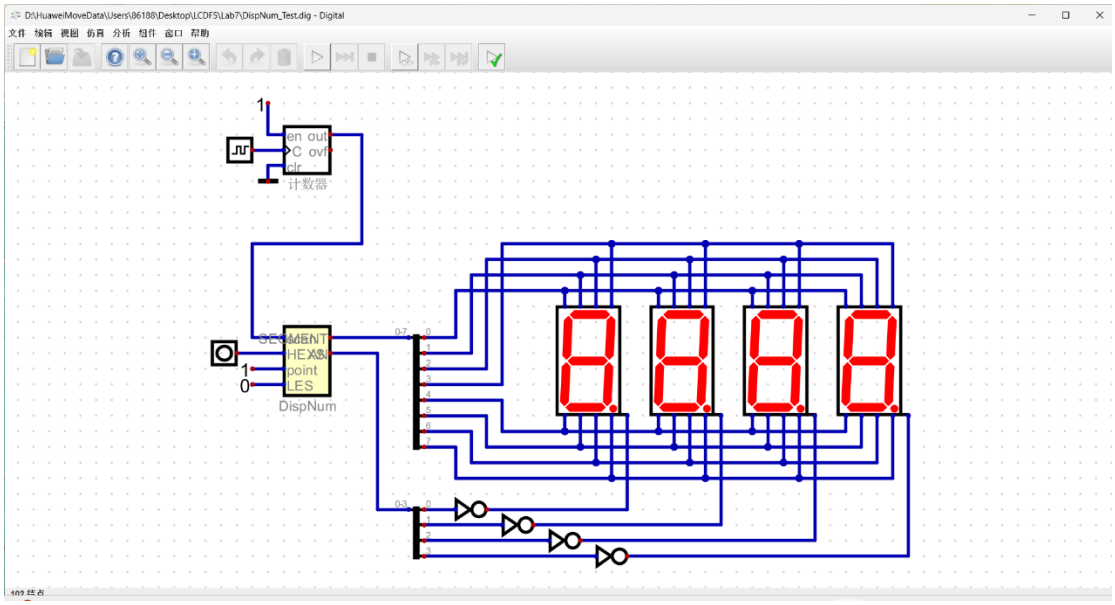
其中当 S 的值为 k 时，表明多路选择器选择了 Ik 的输入；当 Ik 的值为 2^x 时，表明 Ik 的第 x 位是 1；当 O 的输出值为 2^x 时，表明所选择的输入的第 x 位为 1。因此该输出符合预期。

2. 4 位七段数码管扫描显示控制模块设计

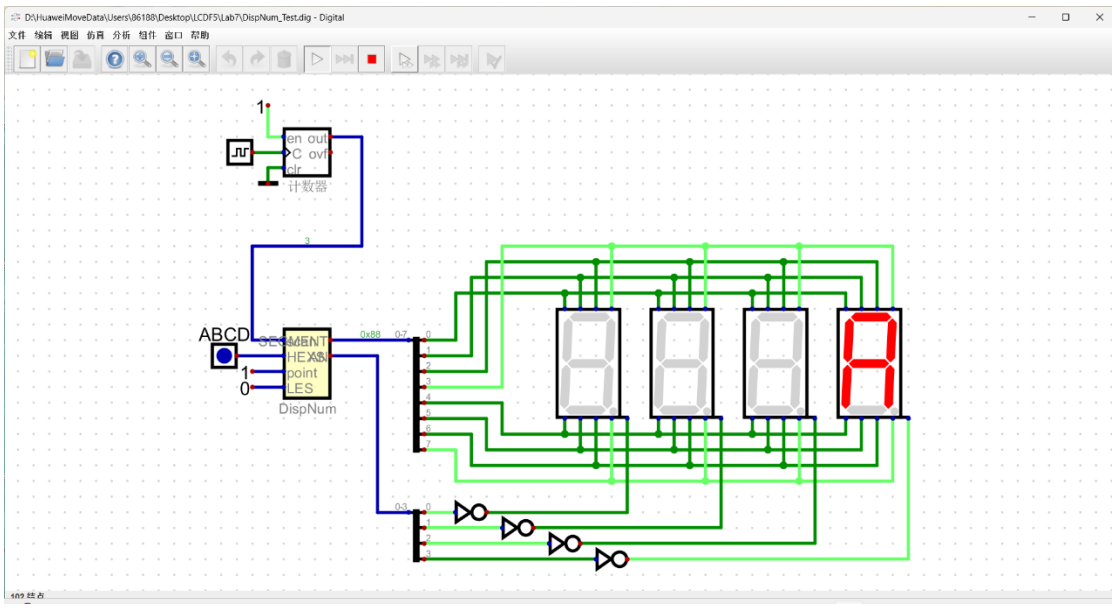
按照原理图搭建的电路 DispNum 如图所示：

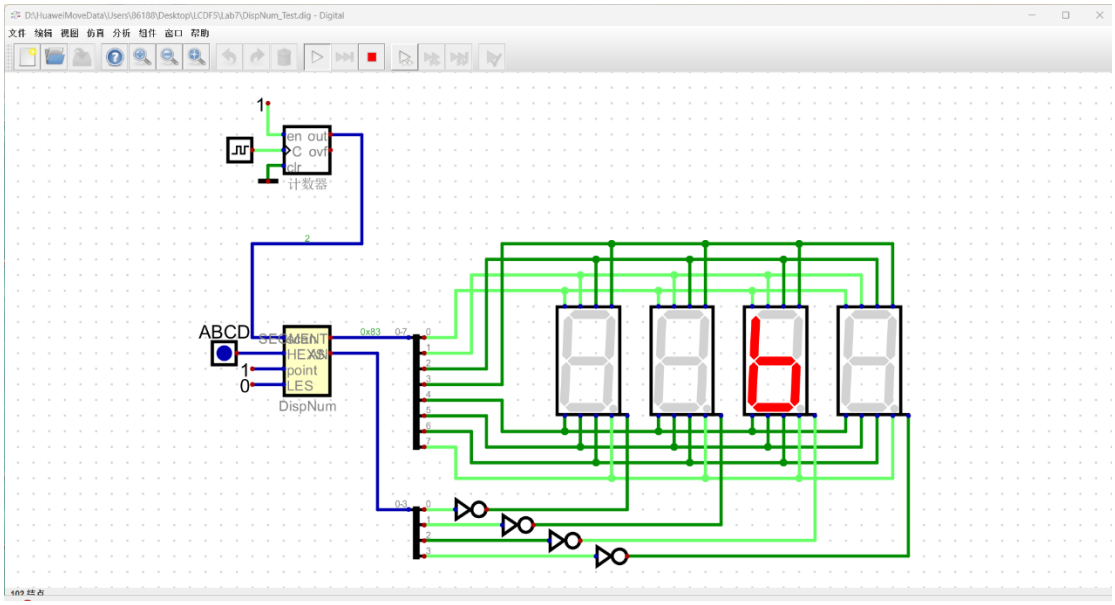


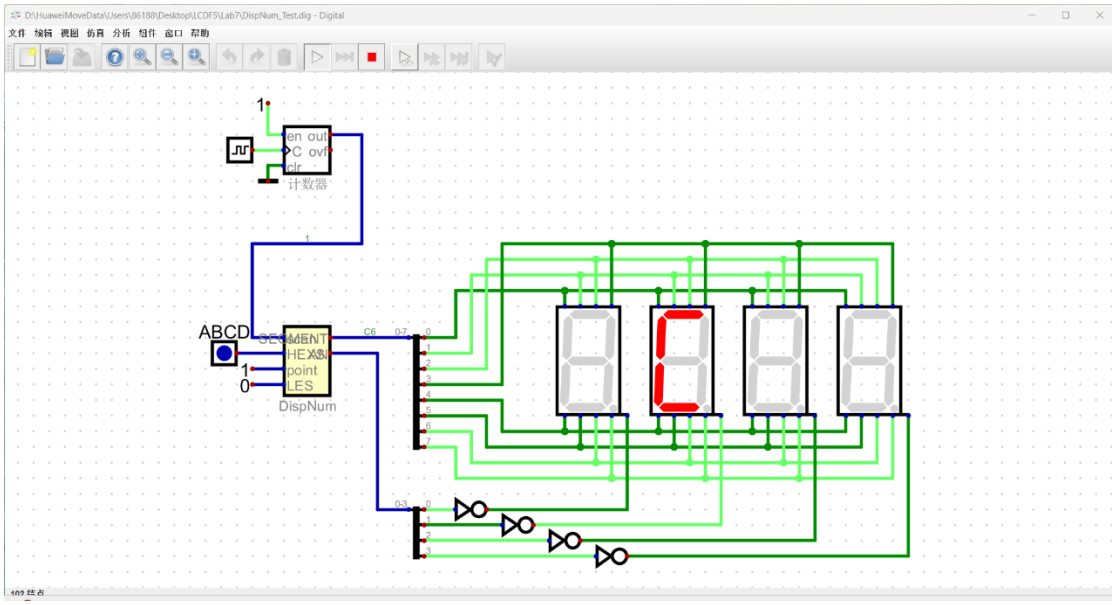
为了检验其正确性，需要按照原理图搭建电路 DispNum_Test，并按照要求调节计数器的频率为 1，位数为 2：

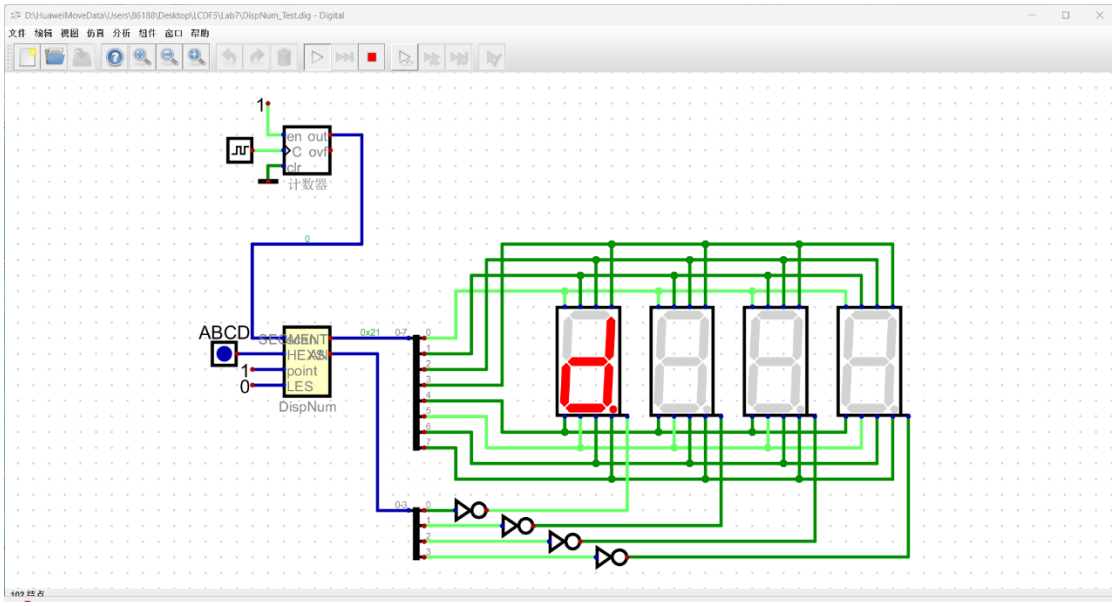


启动该电路，输入 16 进制数 ABCD，观察输出（数码管中从左往右分别对应从低位到高位）：


D:\HuaweiMoveData\Users\86188\Desktop\LCDF5\Lab7\DispNum_Test.dig - Digital
文件 编辑 视图 仿真 分析 组件 窗口 帮助
en out
C ovf
clr
计数器
3
ABCD
SEGMENT
HEX
point
LES
1
0
DispNum
0x88
0-7
0-3


D:\HuaweiMoveData\Users\86188\Desktop\LCDF5\Lab7\DispNum_Test.dig - Digital
文件 编辑 视图 仿真 分析 组件 窗口 帮助
en out
C ovf
clr
计数器
2
ABCD
SEGMENT
HEX
point
LES
1
0
DispNum
0x83
0-7
0-3


D:\HuaweiMoveData\Users\86188\Desktop\LCDF5\Lab7\DispNum_Test.dig - Digital
文件 编辑 视图 仿真 分析 组件 窗口 帮助
en out
C ovf
clr
计数器
1
ABCD
SEGMENT
HEX
point
LES
1
0
DispNum
C6
0-7
0-3



其输出均符合预期。

将 DispNum 电路导出为 verilog 文件，代码如下：

```
/*
 * Generated by Digital. Don't modify this file!
 * Any changes will be lost if this file is regenerated.
 */

module Decoder2 (
    output out_0,
    output out_1,
    output out_2,
    output out_3,
    input [1:0] sel
);
    assign out_0 = (sel == 2'h0)? 1'b1 : 1'b0;
    assign out_1 = (sel == 2'h1)? 1'b1 : 1'b0;
    assign out_2 = (sel == 2'h2)? 1'b1 : 1'b0;
    assign out_3 = (sel == 2'h3)? 1'b1 : 1'b0;
endmodule


module Mux4to1 (
  input [1:0] S,
  input [3:0] I,
  output O
);
  assign O = (((~ S[0] & ~ S[1]) & I[0]) | ((S[0] & ~ S[1]) & I[1]) |
((~ S[0] & S[1]) & I[2]) | ((S[1] & S[0]) & I[3]));
endmodule
```

```
module Mux4to1b4 (
  input [1:0] S,
  input [3:0] I0,
  input [3:0] I1,
  input [3:0] I2,
  input [3:0] I3,
  output [3:0] O
);
  wire s0;
  wire s1;
  wire s2;
  wire s3;
  wire s4;
  wire s5;
  wire s6;
  wire s7;
  assign s0 = S[0];
  assign s1 = S[1];
  assign s2 = ~ s0;
  assign s3 = ~ s1;
  assign s7 = (s1 & s0);
  assign s4 = (s2 & s3);
  assign s5 = (s0 & s3);
  assign s6 = (s2 & s1);
  assign O[0] = ((s4 & I0[0]) | (s5 & I1[0]) | (s6 & I2[0]) | (s7 &
I3[0]));
  assign O[1] = ((s4 & I0[1]) | (s5 & I1[1]) | (s6 & I2[1]) | (s7 &
I3[1]));
  assign O[2] = ((s4 & I0[2]) | (s5 & I1[2]) | (s6 & I2[2]) | (s7 &
I3[2]));
  assign O[3] = ((s4 & I0[3]) | (s5 & I1[3]) | (s6 & I2[3]) | (s7 &
I3[3]));
endmodule

module MyMC14495 (
  input D3,
  input D2,
  input D1,
  input D0,
  input point,
  input LE,
  output p,
  output g,
  output f,
```

```
  output e,
  output d,
  output c,
  output b,
  output a
);
  wire s0;
  wire s1;
  wire s2;
  wire s3;
  assign s0 = ~ D3;
  assign s1 = ~ D2;
  assign s2 = ~ D1;
  assign s3 = ~ D0;
  assign p = ~ point;
  assign g = (((s2 & s1 & s0) | (D0 & D1 & D2 & s0) | (s3 & s2 & D2 &
D3)) | LE);
  assign f = (((D0 & s1 & s0) | (D1 & s1 & s0) | (D0 & D1 & s0) | (D0 &
s2 & D2 & D3)) | LE);
  assign e = (((D0 & s0) | (s2 & D2 & s0) | (D0 & s2 & s1)) | LE);
  assign d = (((D0 & s2 & s1 & s0) | (s3 & s2 & D2 & s0) | (D0 & D1 &
D2) | (s3 & D1 & s1 & D3)) | LE);
  assign c = (((s3 & D1 & s1 & s0) | (s3 & D2 & D3) | (D1 & D2 & D3)) |
LE);
  assign b = (((D0 & s2 & D2 & s0) | (s3 & D1 & D2) | (s3 & D2 & D3) |
(D0 & D1 & D3)) | LE);
  assign a = (((D0 & s2 & s1 & s0) | (s3 & s2 & D2 & s0) | (D0 & D1 & s1
& D3) | (D0 & s2 & D2 & D3)) | LE);
endmodule

module DispNum (
  input [1:0] scan,
  input [15:0] HEXS,
  input [3:0] point,
  input [3:0] LES,
  output [7:0] SEGMENT,
  output [3:0] AN
);
  wire s0;
  wire s1;
  wire s2;
  wire s3;
  wire [3:0] s4;
  wire [3:0] s5;
```

```
wire [3:0] s6;
wire [3:0] s7;
wire [3:0] s8;
wire s9;
wire s10;
wire s11;
wire s12;
wire s13;
wire s14;
wire s15;
wire s16;
wire s17;
wire s18;
wire s19;
wire s20;
wire s21;
wire s22;
Decoder2 Decoder2_i0 (
  .sel( scan ),
  .out_0( s0 ),
  .out_1( s1 ),
  .out_2( s2 ),
  .out_3( s3 )
);
Mux4to1 Mux4to1_i1 (
  .S( scan ),
  .I( point ),
  .O( s13 )
);
Mux4to1 Mux4to1_i2 (
  .S( scan ),
  .I( LES ),
  .O( s14 )
);
assign s4 = HEXS[3:0];
assign s5 = HEXS[7:4];
assign s6 = HEXS[11:8];
assign s7 = HEXS[15:12];
assign AN[0] = ~ s0;
assign AN[1] = ~ s1;
assign AN[2] = ~ s2;
assign AN[3] = ~ s3;
Mux4to1b4 Mux4to1b4_i3 (
  .S( scan ),
```

```
    .I0( s4 ),
    .I1( s5 ),
    .I2( s6 ),
    .I3( s7 ),
    .O( s8 )
  );
  assign s9 = s8[3];
  assign s10 = s8[2];
  assign s11 = s8[1];
  assign s12 = s8[0];
  MyMC14495 MyMC14495_i4 (
    .D3( s9 ),
    .D2( s10 ),
    .D1( s11 ),
    .D0( s12 ),
    .point( s13 ),
    .LE( s14 ),
    .p( s15 ),
    .g( s16 ),
    .f( s17 ),
    .e( s18 ),
    .d( s19 ),
    .c( s20 ),
    .b( s21 ),
    .a( s22 )
  );
  assign SEGMENT[7] = s15;
  assign SEGMENT[6] = s16;
  assign SEGMENT[5] = s17;
  assign SEGMENT[4] = s18;
  assign SEGMENT[3] = s19;
  assign SEGMENT[2] = s20;
  assign SEGMENT[1] = s21;
  assign SEGMENT[0] = s22;
endmodule
```

3. 记分板设计

按要求创建 ScoreBoard 工程，其结构如图所示：

DispNum.v 为导入文件，代码已展示，故不再重复展示。

clkdiv.v 代码如下：

```
`timescale 1ns / 1ps
module clkdiv(input wire clk,
    input wire rst,
    output reg[31:0] clk_div
    );
    //used in simulation
    initial begin
        clk_div = 0;
    end
    //clock divider
   always @(posedge clk or posedge rst) begin
     if (rst) clk_div <= 0;
     else clk_div <= clk_div + 1'b1;
   end
endmodule
```

CreateNumber.v 代码如下：

```
module CreateNumber(
    input wire [3:0] btn,
    output reg [15:0] num
    );
    wire [3:0] A,B,C,D;

    initial num <= 16'b1010_1011_1100_1101;

    assign A = num[3:0]+4'd1;
    assign B = num[7:4]+4'd1;
    assign C = num[11:8]+4'd1;
    assign D = num[15:12]+4'd1;

    always@(posedge btn[0]) num[3:0]<=A;
    always@(posedge btn[1]) num[7:4]<=B;
    always@(posedge btn[2]) num[11:8]<=C;
    always@(posedge btn[3]) num[15:12]<=D;

endmodule
```

top.v 代码如下：

```
module top(input wire clk,
    input wire [7:0] SW,
    input wire [3:0] BTN,
```

```
    output wire [3:0] AN,
    output wire [7:0] SEGMENT,
    output wire BTNX4);

    wire [15:0] num;
    wire [31:0] clk_div;
    CreateNumber c0(.btn(BTN),.num(num));
    DispNum d0(.scan(clk_div[18:17]),  .HEXS(num), .LES(SW[7:4]),
           .point(SW[3:0]), .AN(AN), .SEGMENT(SEGMENT));
    clkdiv c1(.clk(clk), .rst(1'b0), .clk_div(clk_div));

    assign BTNX4 = 1'b0;
endmodule
```

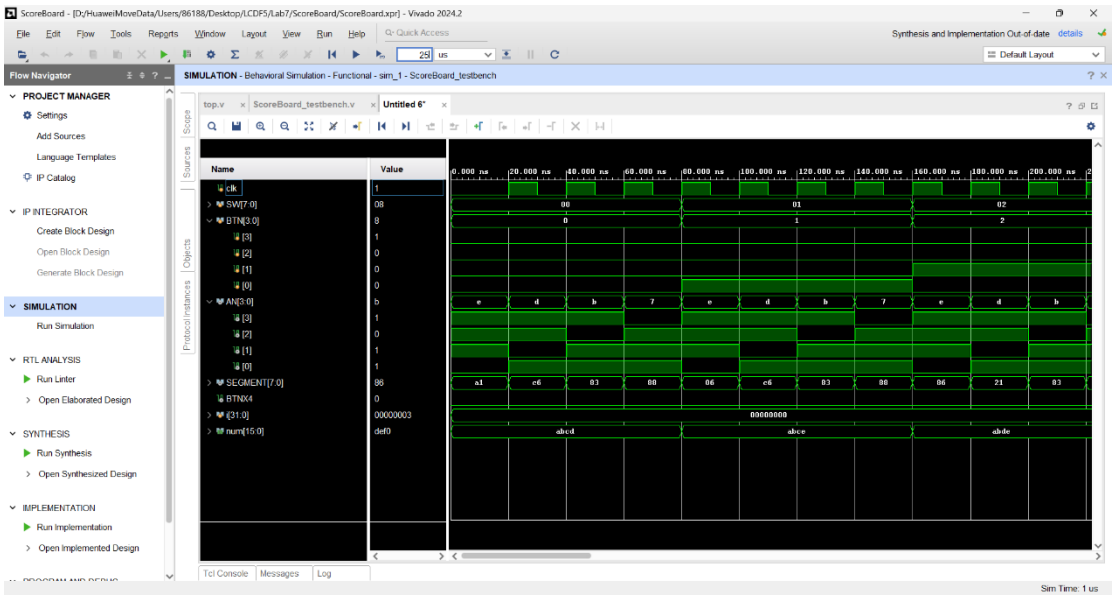
ScoreBoard_testbench.v 如下：

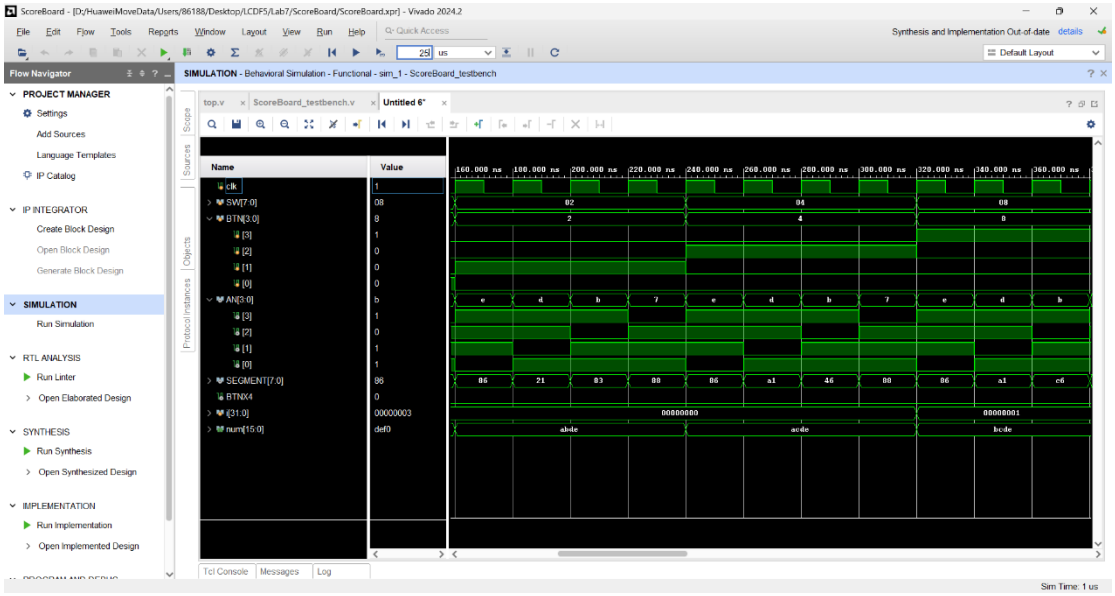
```
`timescale 1ns / 1ps
module ScoreBoard_testbench();
    reg clk;
    reg [7:0] SW;
    reg [3:0] BTN;
    wire [3:0] AN;
    wire [7:0] SEGMENT;
    wire BTNX4;
    top uut(
       .clk(clk),
       .SW(SW),
       .BTN(BTN),
       .AN(AN),
       .SEGMENT(SEGMENT),
       .BTNX4(BTNX4)
    );
    integer i;
    initial begin
       clk=0;
       SW[7:0]=8'b00000000;
       BTN[3:0]=4'b0000;
       for(i=0;i<=3;i=i+1) begin
           #80
           BTN[3]=0;
           SW[3]=0;
           BTN[0]=1;
           SW[0]=1;
           #80
           BTN[0]=0;
           SW[0]=0;
```

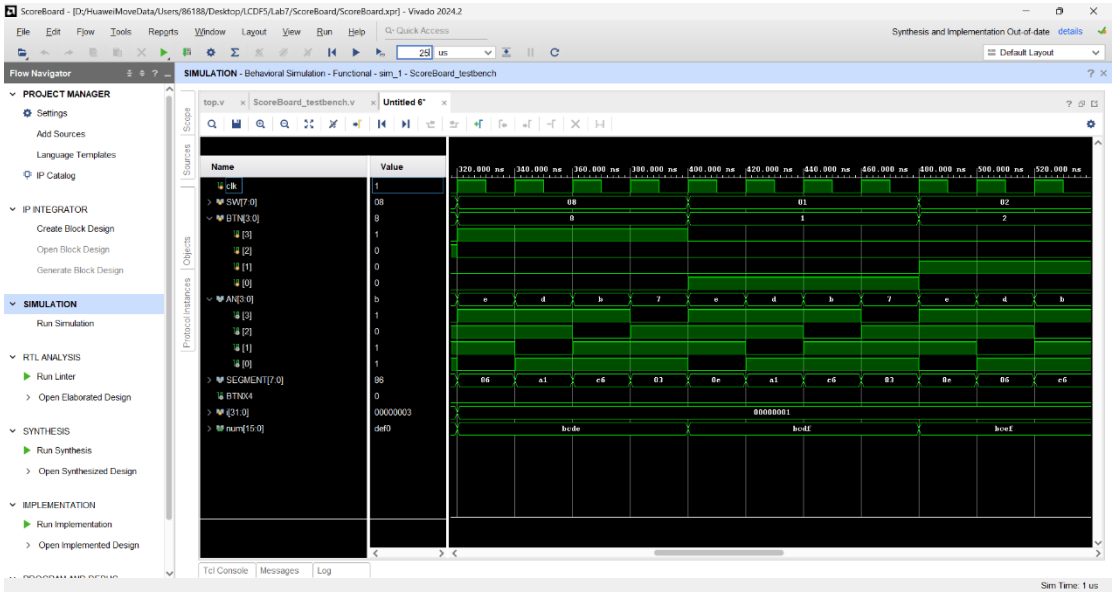
```
            BTN[1]=1;
            SW[1]=1;
            #80
            BTN[1]=0;
            SW[1]=0;
            BTN[2]=1;
            SW[2]=1;
            #80
            BTN[2]=0;
            SW[2]=0;
            BTN[3]=1;
            SW[3]=1;
        end
    end
    always begin
        #10 clk = 0;
        #10 clk = 1;
    end
endmodule
```

注：在仿真中 clk_div[18:17]应改为 clk_div[1:0]，此时 AN 每 20ns 变化一次，故 BTN 每 80ns 按下一次。

仿真结果如下图所示：







其中每当 BTN 变化一次，AN 会变化四次，也即当 BTN 状态不变时，AN 能够完整遍历（从低位到高位）每一位 16 进制数。

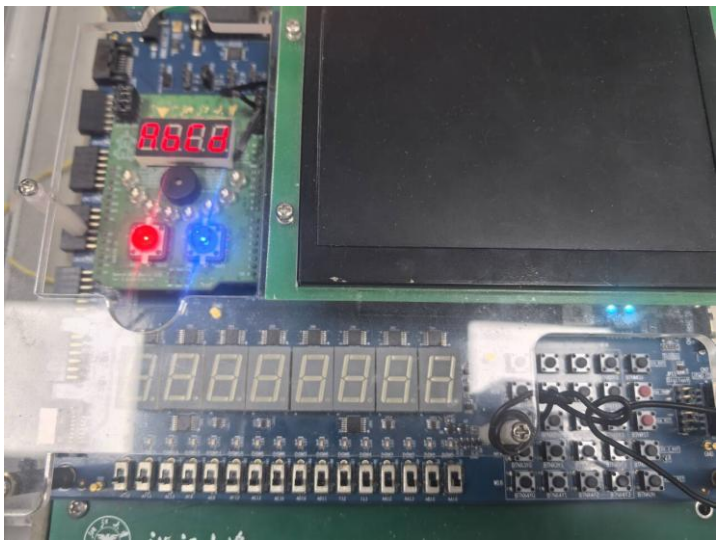
SEGMENT 中的最高二进制位表示小数点状态，将其忽略（如果 16 进制数高位大于等于 8，就减去 8）后，整理其出现的所有状态：

- 08：只有 4 号管不亮，数码管显示 A
- 03：只有 1,2 号管不亮，数码管显示 b
- 46：2,3,7 号管不亮，数码管显示 C
- 21：只有 1,6 号管不亮，数码管显示 d
- 06：只有 2,3 号管不亮，数码管显示 E
- 0e：2,3,4 号管不亮，数码管显示 F

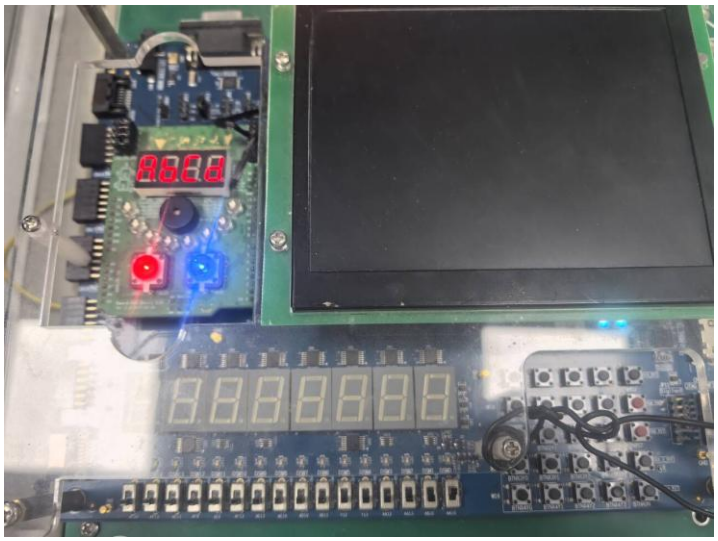
然后进行对照，发现显示无误，说明整个工程无误。

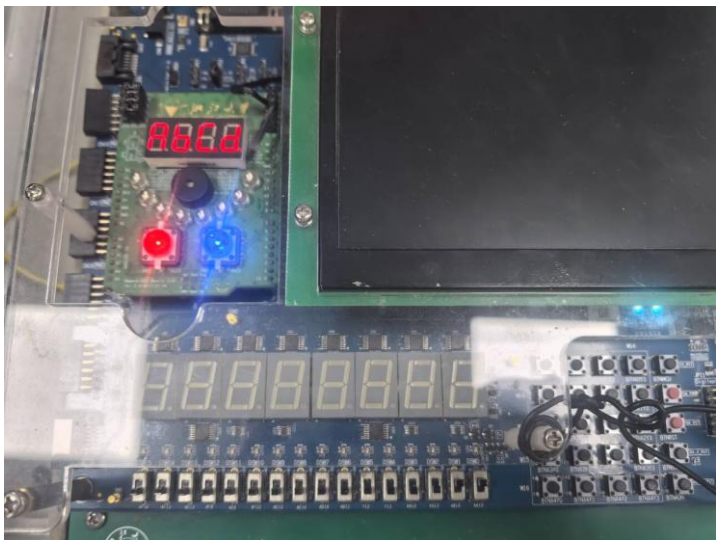
将其下载到板上进行验证，结果如下：

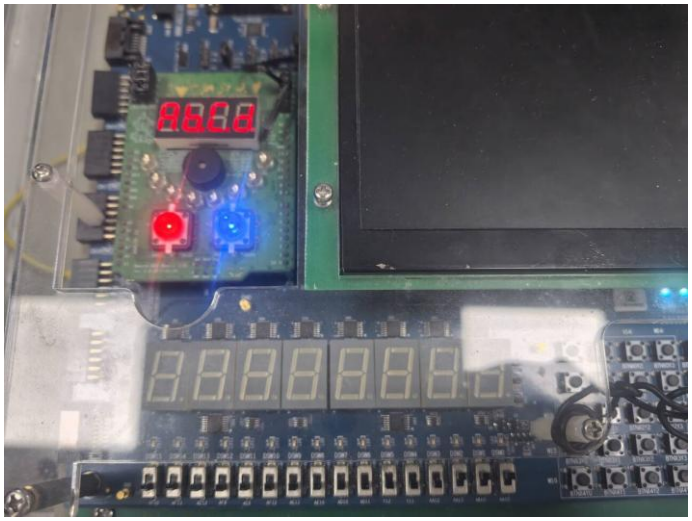
下载完成：

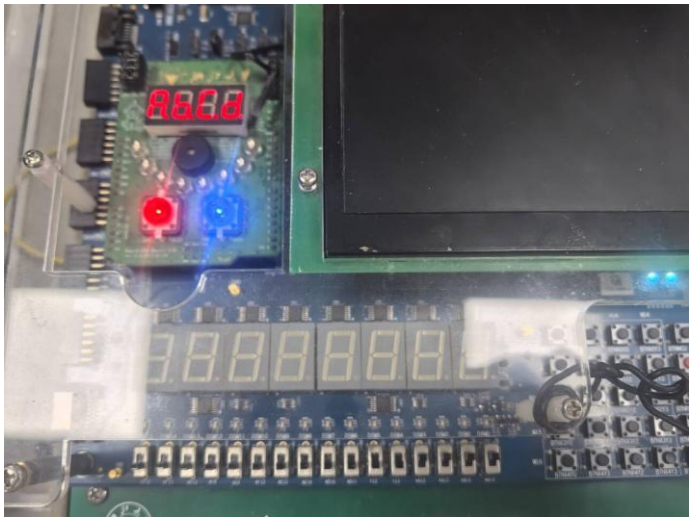


逐位显示小数点：

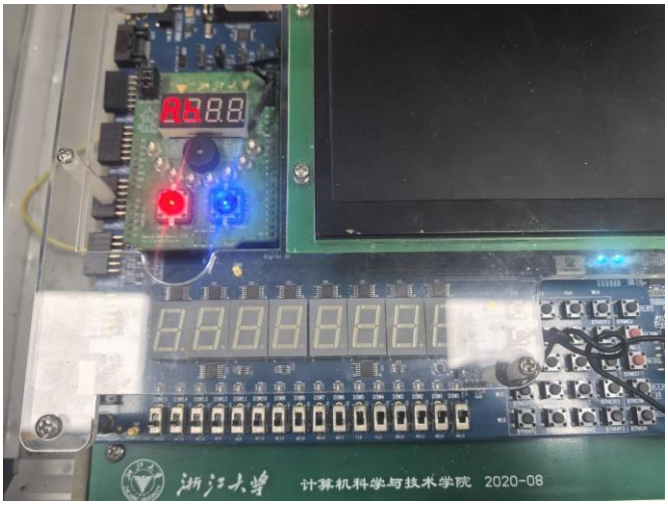




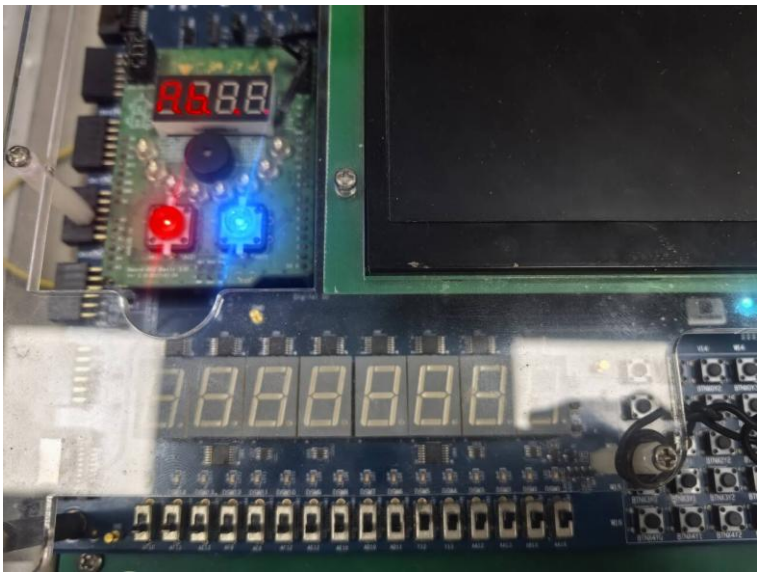


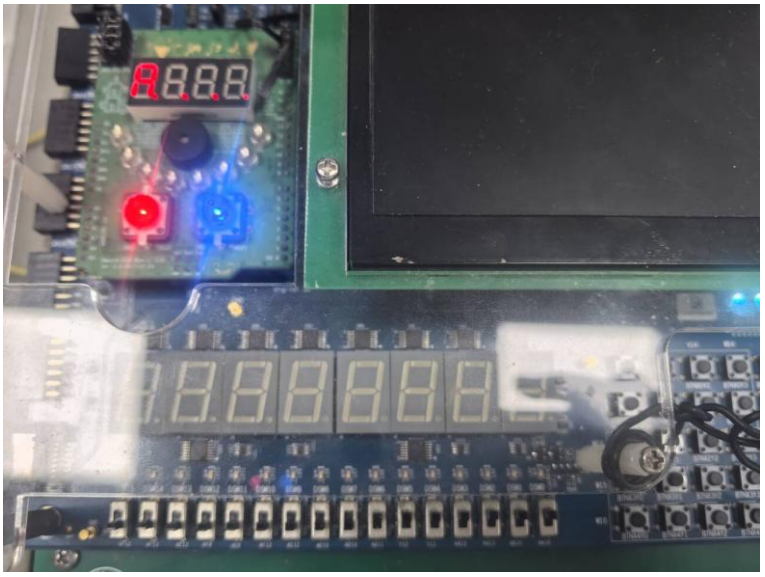


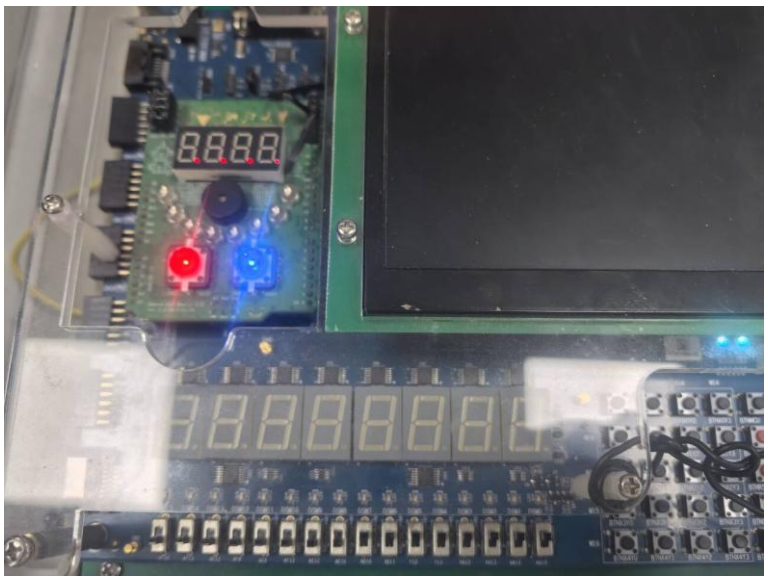
逐位禁用（由于手机曝光时间较短，有些数码不是很清晰）：



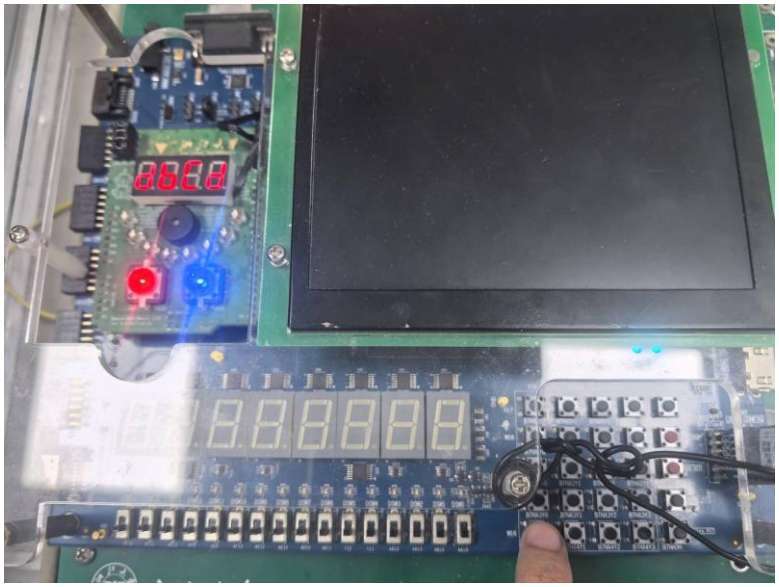
（只能隐约看到 C）

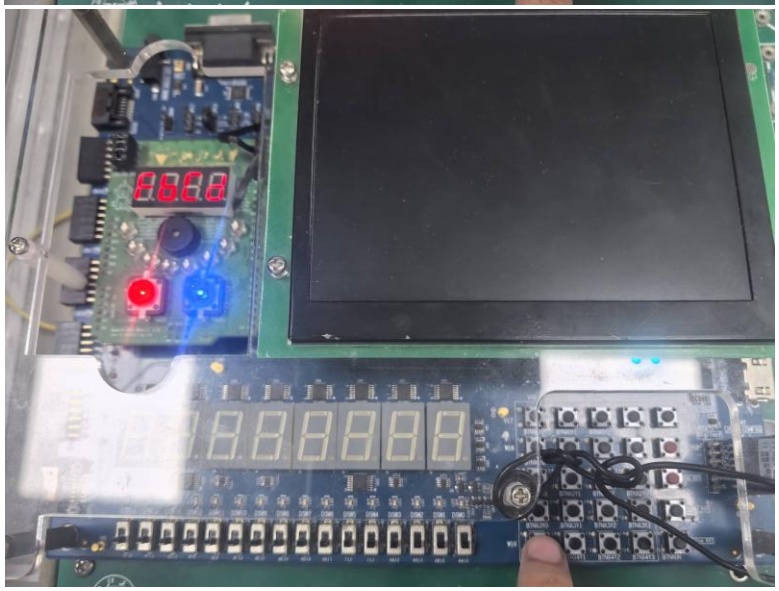


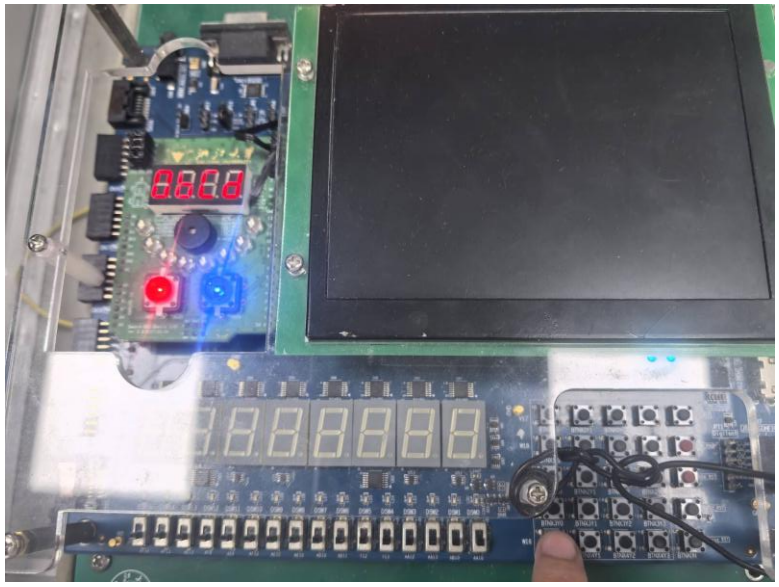


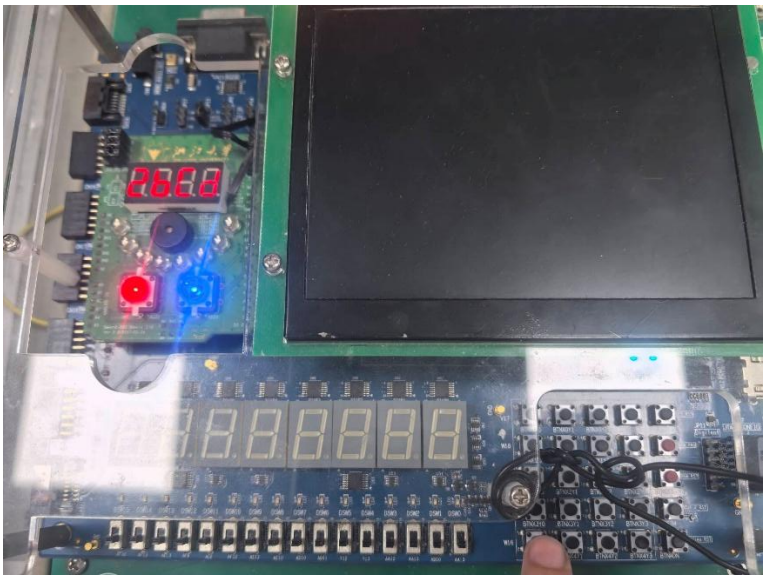


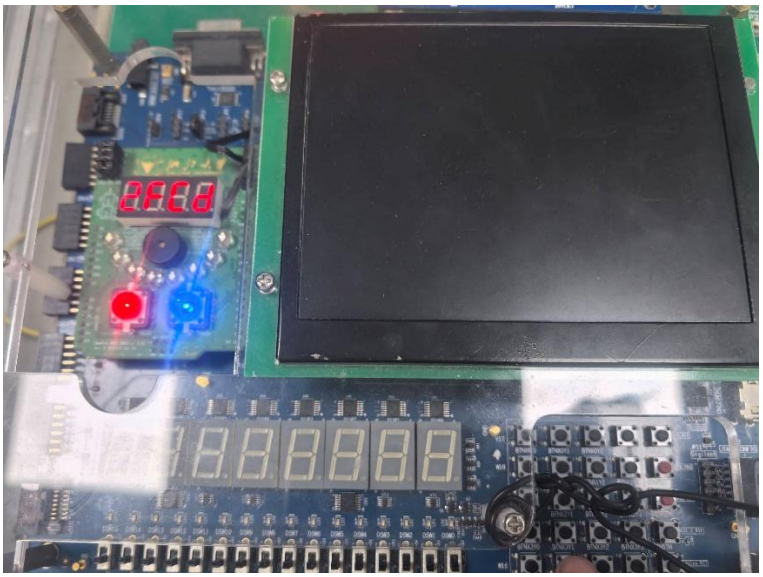
将小数点除去、数字全部使能，对按钮进行操作：

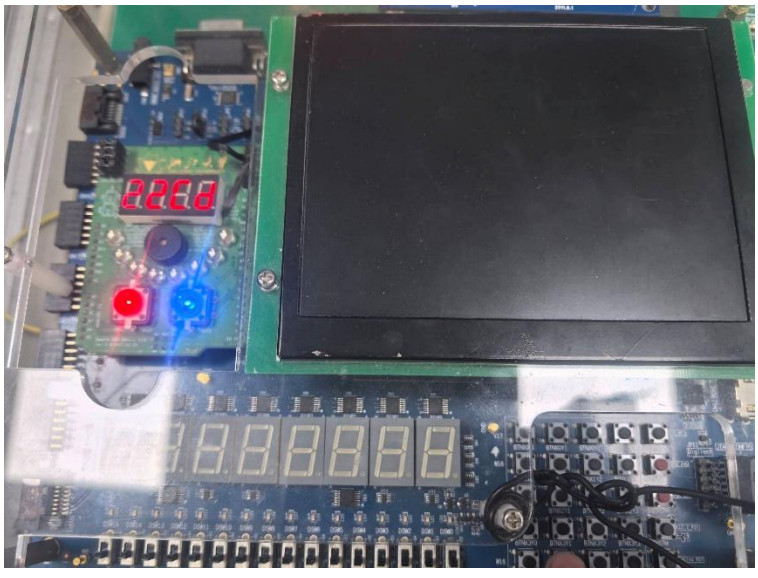


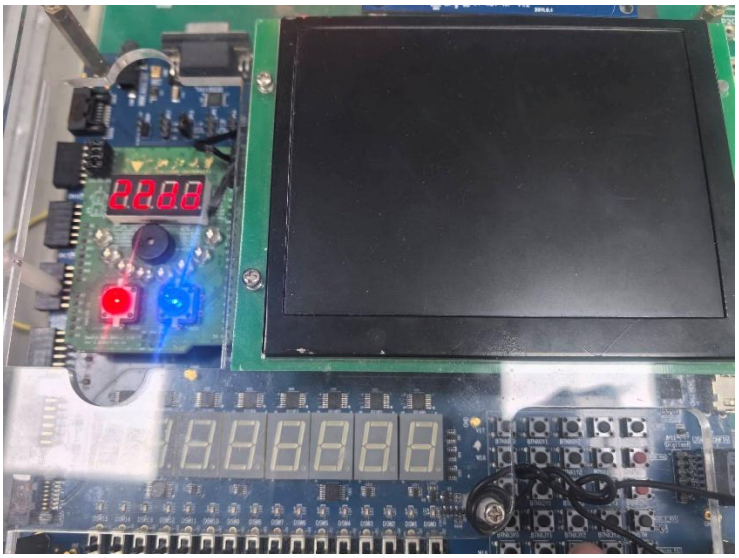


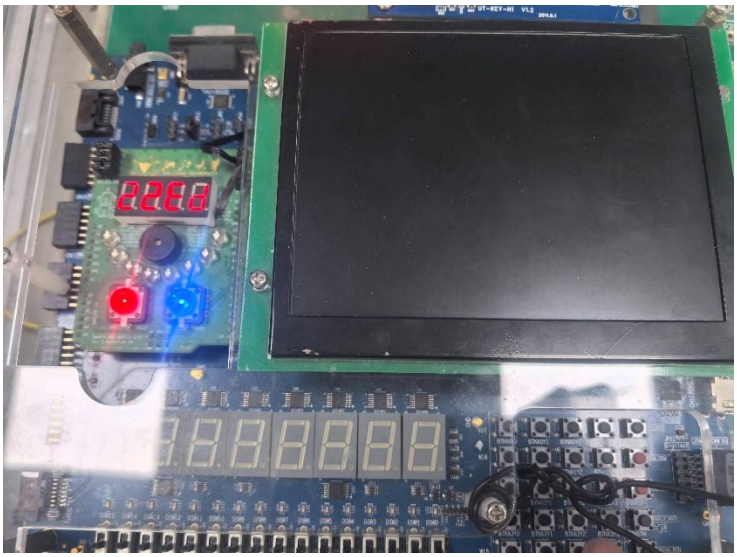


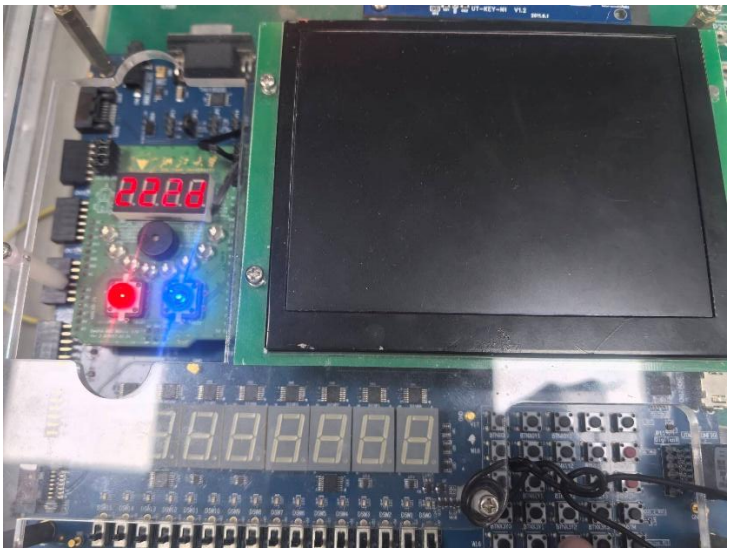


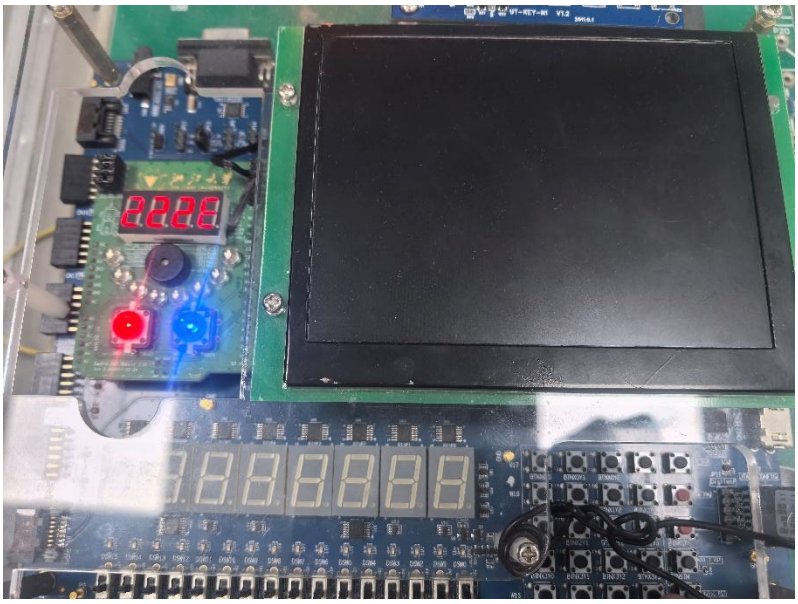












（手在角落）

注意到有时候按动一下按钮，数字会变化多次，可能原因为按键抖动。除此之外实验结果均符合预期。

四、讨论、心得

这一次实验中，由于激励代码需要全部自己写，且没有固定测试方法，挑战性大幅提升，在这次实验中花费的时间也远远比前两次板上实验的时间久，期间多次测试用例都无法通过验收。不过最后当我成功通过验收时，我对激励代码的写法的熟练度也得到了大幅度提升。