一. 实验目的

- 学习使用 Verilog HDL 描述数字逻辑电路与系统的方法；
- 了解并掌握采用可编程逻辑器件实现数字电路与系统的方法；
- 学习并掌握采用 Vivado 软件开发可编程器件的过程；

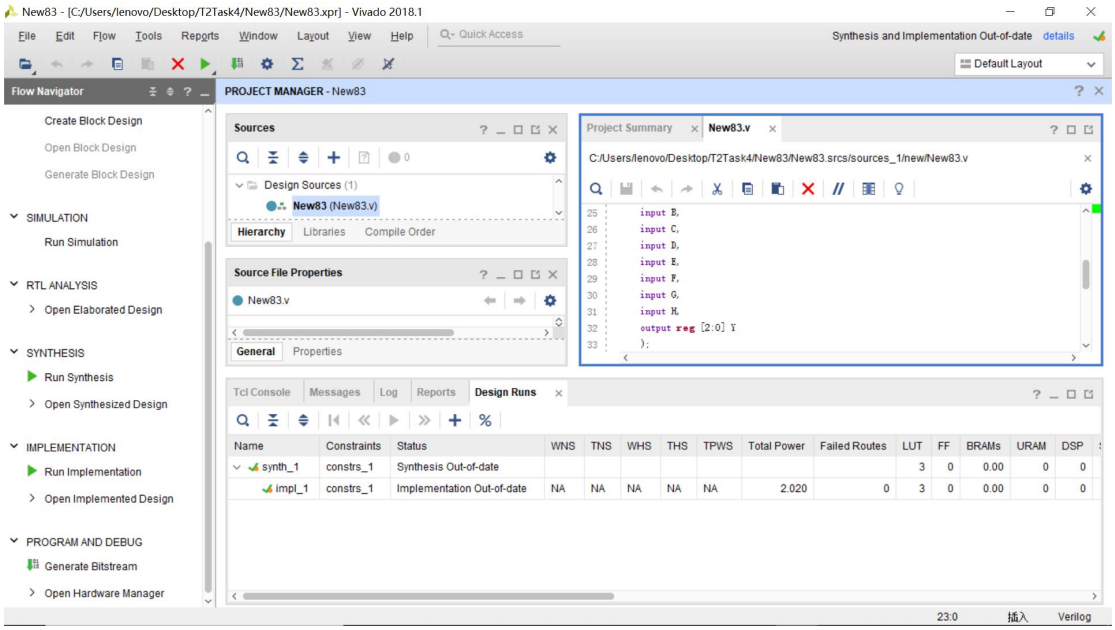
二. 实验任务与验收要求

- 使用 Vivado 软件设计、仿真、下载实现 8 线-3 线优先编码器
- 将要求 1 中 Verilog 语言采取三种方式分别实现；
- 实现文字显示器，并能够显示 HELLO，HI，HAHA，HEHE。

三. 设计平台

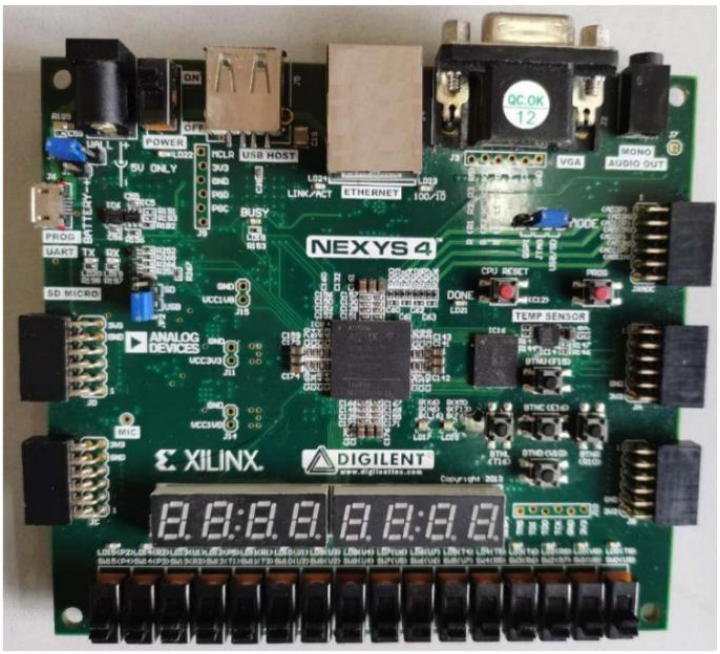
嵌入式软件：Vivado 2018.1

软件界面如下图

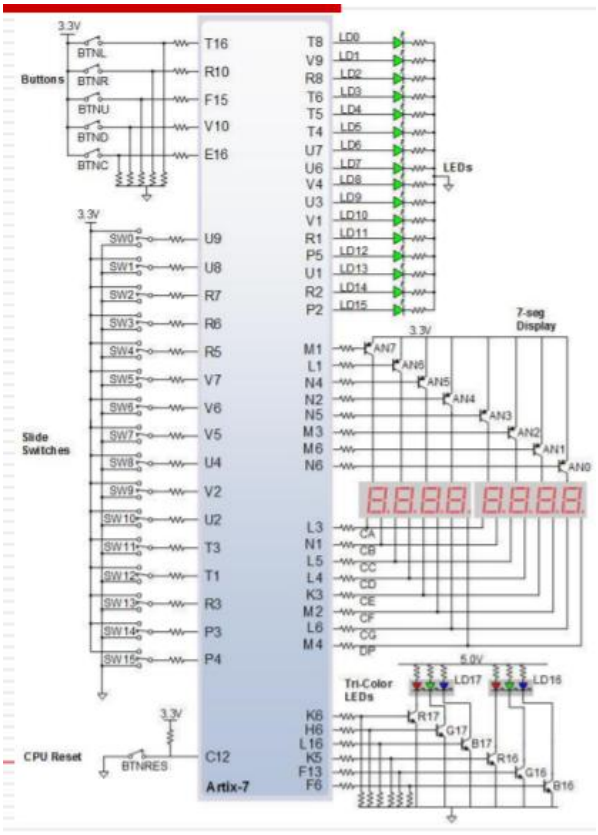


硬件平台：Nexys4

开发板外观如下图



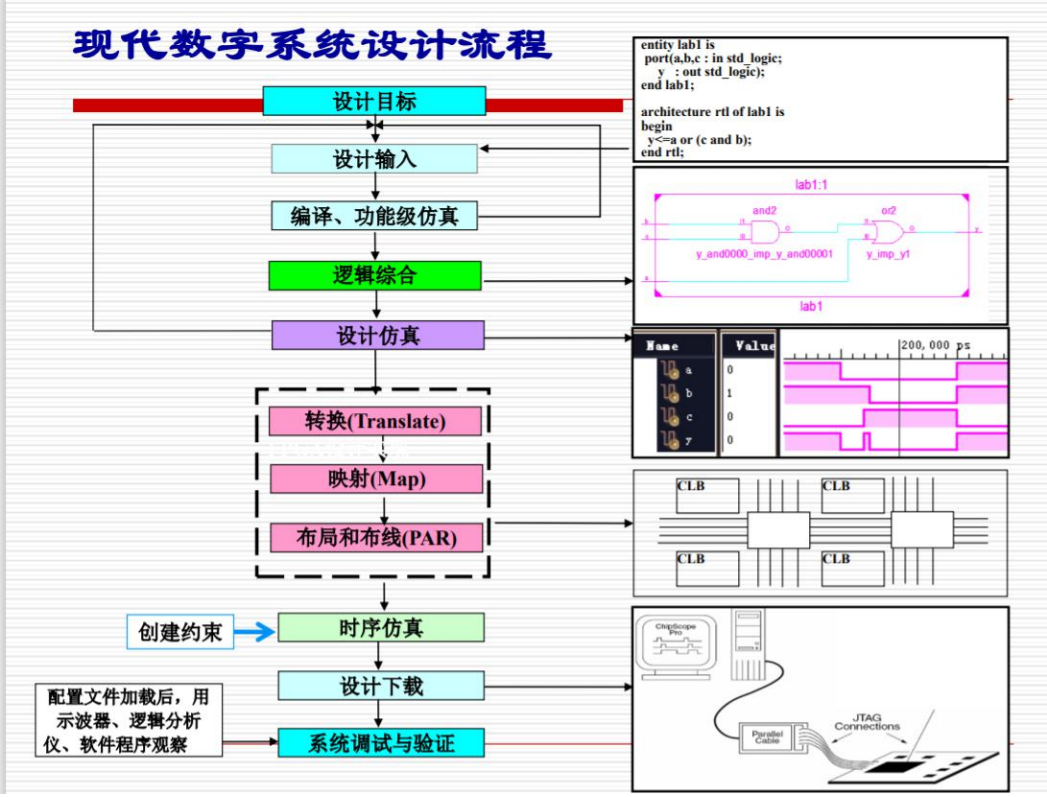
Nexys4 可用资源如下图：



四. 电路设计与实验仿真

（1） 开发流程：

如下图所示：



（2） 电路设计：

① 8-3 线优先译码器的真值表

输　　入								输　出		
din0	din1	din2	din3	din4	din5	din6	din7	out0	out1	out2
1	x	x	x	x	x	x	x	0	0	0
0	1	x	x	x	x	x	x	1	0	0
0	0	1	x	x	x	x	x	0	1	0
0	0	0	1	x	x	x	x	1	1	0
0	0	0	0	1	x	x	x	0	0	1
0	0	0	0	0	1	x	x	1	0	1
0	0	0	0	0	0	1	x	0	1	1
0	0	0	0	0	0	0	1	0	1	1

② 逻辑表达式

Y2=X4&X5&X6&X7

Y1=~(~(X2)&X4&X5|~(X3)&X4&X5|~(X6)|~(X7));

Y0=~(~(X1)&x2&X4&X6|~(X3)&X4&X6|~(X5)&X6|~(X7));

③ 实现代码

a)采用行为级描述：

```
module New83(
    input A,
    input B,
    input C,
    input D,
    input E,
    input F,
    input G,
    input H,
    output reg [2:0] Y
    );

       always @(A or B or C or D or E or F or G or H)
         begin
                if(H) Y<=3'b111;
                else if(G) Y <= 3'b110;
                else if(F) Y <= 3'b101;
                else if(E) Y <= 3'b100;
                else if(D) Y <= 3'b011;
                else if(C) Y <= 3'b010;
                else if(B) Y <= 3'b001;
                else Y <= 3'b000;

          end

endmodule
```

b)采用数据流描述：

```
module encoder83(
    input [7:0] X,
    output  [2:0] Y
    );
    assign Y[2]=X[4]&X[5]&X[6]&X[7]
    assign Y[1]=~(~(X[2])&X[4]&X[5]|~(X[3])&X[4]&X[5]|~(X[6])~(X[7]));
    assign Y[0]=~(~(X[1])&x[2]&X[4]&X[6]~(X[3])&X[4]&X[6]~(X[5])&X[6]|~(X[7]));

endmodule
```

c)采用门级描述

```
module encoder83(
    input [7:0] X,
    output  [2:0] Y
    );
    wire X[2]n,X[3]n,X[6]n,X[7]n,Y[1]n,Y[0]n;
    not N0(X[2]n,X[2]);
    not N1(X[3]n,X[3]);
    not N2(X[6]n,X[6]);
    not N3(X[7]n,X[7]);
    and U0(Y[2],X[4],X[5],X[6],X[7]);
    and U1(Y[1]n,X[2]n,X[4],X[5],X[3]n,X[4],X[5],X[6],X[7]n);
    and U2(Y[0]n,X[1],X[2],X[4],X[6],X[3],X[4],X[6],X[5]n,X[6],X[7]);
    not N4(Y[1],Y[1]n);
    not N5(Y[0],Y[0]n);

endmodule
```

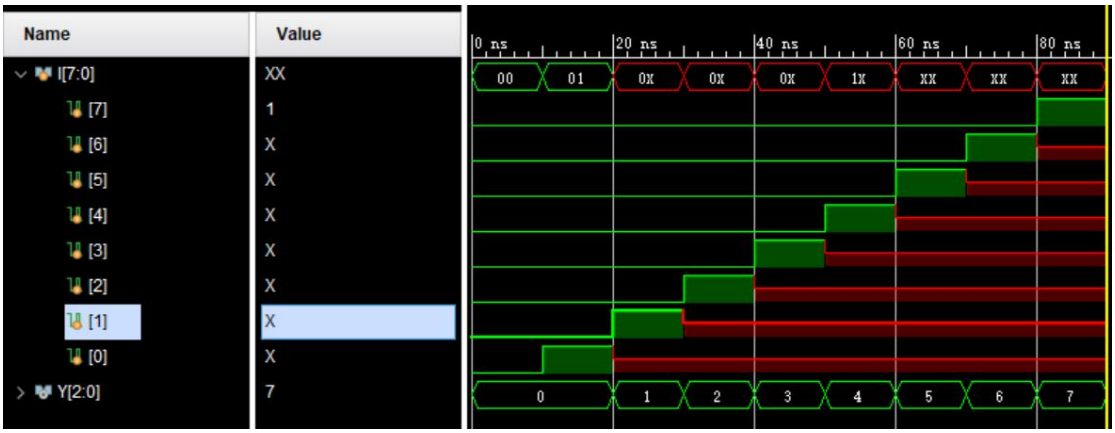
（3） 引脚约束：

```
# Switches
#Bank = 34, Pin name = IO_L21P_T3_DQS_34,                Sch name = SW0
set_property PACKAGE_PIN U9 [get_ports {A}]
    set_property IOSTANDARD LVCMOS33 [get_ports {A}]
#Bank = 34, Pin name = IO_25_34,                         Sch name = SW1
set_property PACKAGE_PIN U8 [get_ports {B}]
    set_property IOSTANDARD LVCMOS33 [get_ports {B}]
#Bank = 34, Pin name = IO_L23P_T3_34,                    Sch name = SW2
set_property PACKAGE_PIN R7 [get_ports {C}]
    set_property IOSTANDARD LVCMOS33 [get_ports {C}]
#Bank = 34, Pin name = IO_L19P_T3_34,                    Sch name = SW3
set_property PACKAGE_PIN R6 [get_ports {D}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D}]
#Bank = 34, Pin name = IO_L19N_T3_VREF_34,               Sch name = SW4
set_property PACKAGE_PIN R5 [get_ports {E}]
    set_property IOSTANDARD LVCMOS33 [get_ports {E}]
#Bank = 34, Pin name = IO_L20P_T3_34,                    Sch name = SW5
set_property PACKAGE_PIN V7 [get_ports {F}]
    set_property IOSTANDARD LVCMOS33 [get_ports {F}]
#Bank = 34, Pin name = IO_L20N_T3_34,                    Sch name = SW6
set_property PACKAGE_PIN V6 [get_ports {G}]
    set_property IOSTANDARD LVCMOS33 [get_ports {G}]
#Bank = 34, Pin name = IO_L10P_T1_34,                    Sch name = SW7
set_property PACKAGE_PIN V5 [get_ports {H}]
    set_property IOSTANDARD LVCMOS33 [get_ports {H}]
#Bank = 34, Pin name = IO_L8P_T1-34,

```

```
28  # LEDs
29  #Bank = 34, Pin name = IO_L24N_T3_34,                    Sch name = LED0
30  set_property PACKAGE_PIN T8 [get_ports {Y[0]}]
31      set_property IOSTANDARD LVCMOS33 [get_ports {Y[0]}]
32  #Bank = 34, Pin name = IO_L21N_T3_DQS_34,                Sch name = LED1
33  set_property PACKAGE_PIN V9 [get_ports {Y[1]}]
34      set_property IOSTANDARD LVCMOS33 [get_ports {Y[1]}]
35  #Bank = 34, Pin name = IO_L24P_T3_34,                    Sch name = LED2
36  set_property PACKAGE_PIN R8 [get_ports {Y[2]}]
37      set_property IOSTANDARD LVCMOS33 [get_ports {Y[2]}]
```

（4） 实验仿真

我们将输入从 00000000~1xxxxxxx 依次改变，改变的周期为 10ns,可以看到输出 Y 依次将其译码为{0,0,1,2,3,4,5,6,7}，验证得到该仿真是正确的。



（5） 附加实验：

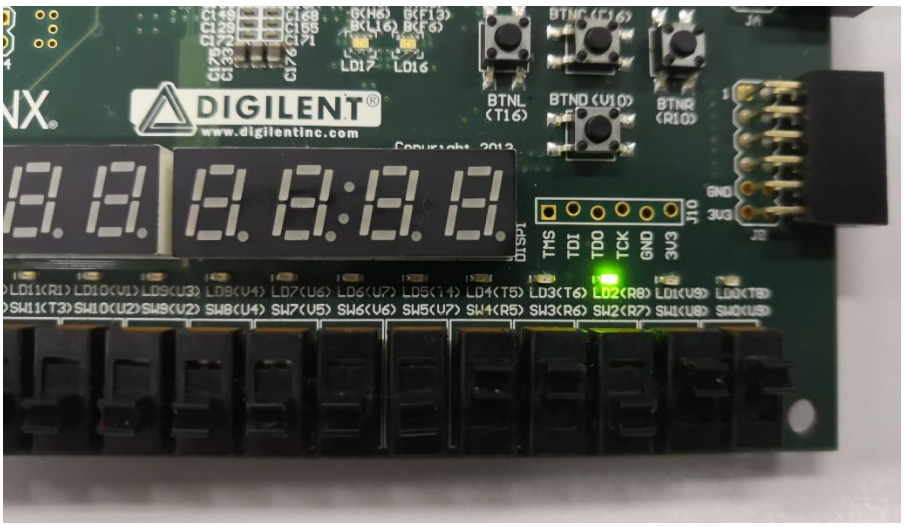
实现代码（部分核心代码）：

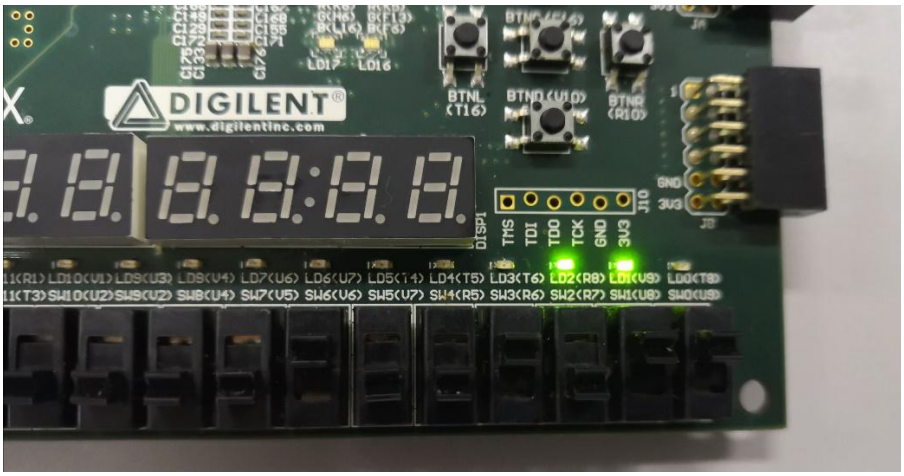
```
module haha(
    input clk,
    input choice,
    output reg [7:0]seg,
    output reg [7:0]an
    );

reg [2:0] state;
parameter interval = 100000000/1000 - 1;

integer cnt;
always @(posedge clk)
begin
  cnt <= cnt+1;
  if (cnt == interval)
  begin
    cnt <= 0;
    state <= state+1;
  end
end
```

```
always @(choice)
begin
//显示Hello Hi
    if(choice==1'b0)
    begin
          an = 8'b00000000;
                case(state)
                3'b000 :
                    begin
                    an = 8'b0111_1111;
                    seg = 8'b1000_1001;

                    end
                3'b001 : begin
                    an = 8'b1011_1111;
                    seg = 8'b10000110;
                    end
                3'b010 : begin
                    an = 8'b1101_1111;
                    seg = 8'b11000111;

                    end
```

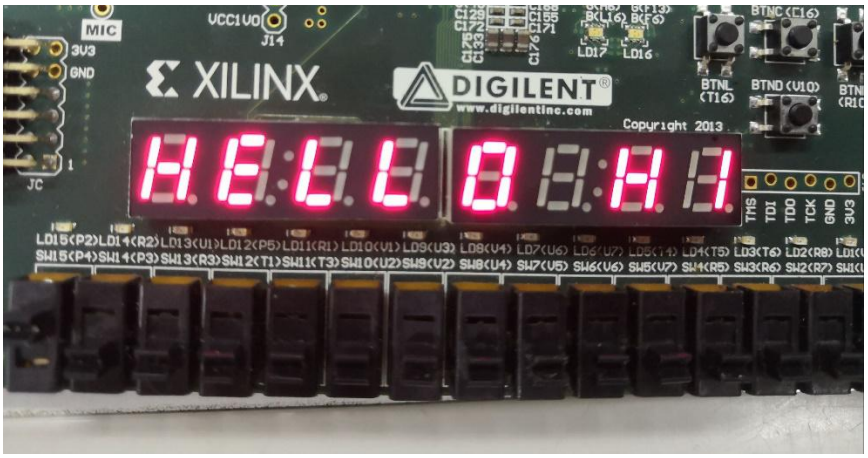
（6） 实验效果：

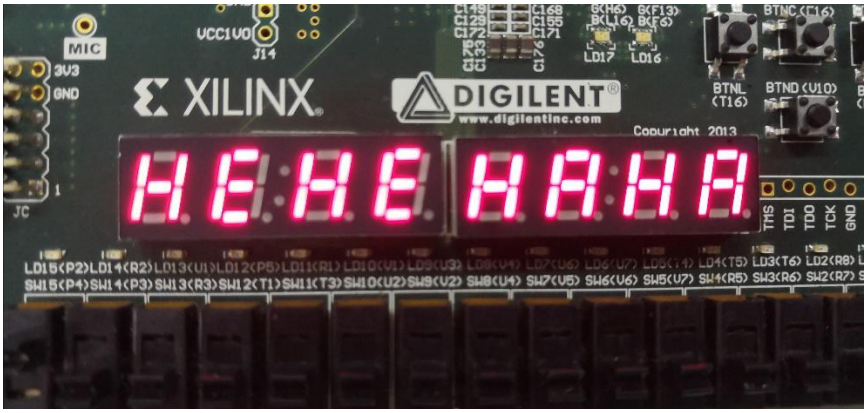
① 8-3 线优先编码器：





② 文字显示：





五. 实验问题与分析

（1） 开始仅能实现 8-3 线简单译码，而未能实现优先编码：

解：开始使用 case 语句如下：

```
//              case ({H,G,F,E,D,C,B,A})
//                 8'b00000001 : Y <= 3'b000;
//                 8'b00000010 : Y <= 3'b001;
//                 8'b00000100 : Y <= 3'b010;
//                 8'b00001000 : Y <= 3'b011;
//                 8'b00010000 : Y <= 3'b100;
//                 8'b00100000 : Y <= 3'b101;
//                 8'b01000000 : Y <= 3'b110;
//                 8'b10000000 : Y <= 3'b111;
//                 default     : Y <= 3'b000;
//              endcase
```

可见，该代码为简单编码，但如果将低位的 0 改为 x 的话，则会显示生成比特流失败，最后改为使用 if……else if 语句，成功实现。

（2） 附加实验中，七段数码管全黑，未实现实验效果。

解：开始时是引脚约束文件中，未将 Clk 与 Nexys4 上面的时钟 clk 所对应，添加

引脚约束后仍不显示；
后来在网络上查询类似情况，问题为：文字显示器部分，分频的时钟周期一定要选择合适。如果时钟周期过快（接近 FPGA 板的时钟周期），有可能存在未显示的情况，猜测的原因是信号变化太快，数码管还未点亮信号就已经消失了。如果时钟周期过慢，则显示的数码管存在闪烁的情况，本次实验选择的分频时钟周期为 1ms。修改代码后实现效果。

（3） 文字显示时，显示内容出错。
解：可知，我们使用的 Nexys4，如果想让对应的数码管点亮，段码该位应为低电平，使用网络上对应的段码为高电平，取反修改后实验效果正确。

六. 实验感想与总结

本次实验没有插板子，而是使用了一种新的硬件开发方式，与之前在面包板上完成的实验不同，而是采用了 FPGA 开发板 Nexys4 配合 Vivado 平台编程实现，这对我们来说是一种全新的体验，同时，也是一次十分有趣的实验，一个能让我们的 Verilog 知识变为现实的实验。。

在 FPGA 开发板上做实验，大大减少了搭建电路的时间，同时由于可编程，我们能实现相对来说更加复杂的电路。通过本次实验，也简单的了解了使用 FPGA 板开发的基本流程：首先在计算机上设计算法、编写代码，然后进行仿真分析，通过代码仿真保证设计方案符合实际要求，最后进行板机调试，利用配置电路将相关文件下载至 FPGA 芯片中，验证实际运行效果。

整个实验过程下来有很多收获，为我们后续更加复杂的 EDA 软件使用与组合逻辑设计熟悉了开发过程也打下了坚实的基础，也希望今后能够尝试设计更加复杂更加有趣的实验。