浙江大学

本科实验报告

课程名称：	计算机逻辑设计基础
姓　　名：	李瀚轩
学　　院：	竺可桢学院
系：	所在系
专　　业：	计算机科学与技术
学　　号：	3220106039
指导教师：	董亚波

2023 年 11 月 6 日

浙江大学实验报告

课程名称： 计算机逻辑设计基础 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： 李瀚轩 专业： 计算机科学与技术 学号： 3220106039

同组学生姓名： 郑涵文 指导老师： 董亚波

实验地点： 东四 509 实验日期： 2023 年 11 月 2 日

一、实验目的和要求

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

二、实验内容和原理

2.1 内容

1. 数据选择器设计
2. 记分板设计

2.2 原理

1. 四选一多路选择器先根据事件简化真值表，输出是控制信号，全部是最小项与或结构

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	$\overline{S1}\,\overline{S0}$ I0
I0 I1 I2 I3	0 1	I1	$\overline{S1}$S0 I1
I0 I1 I2 I3	1 0	I2	S1$\overline{S0}$ I2
I0 I1 I2 I3	1 1	I3	S1S0 I3

2. 多路选择器位扩展控制结构不变，每路输入向量化

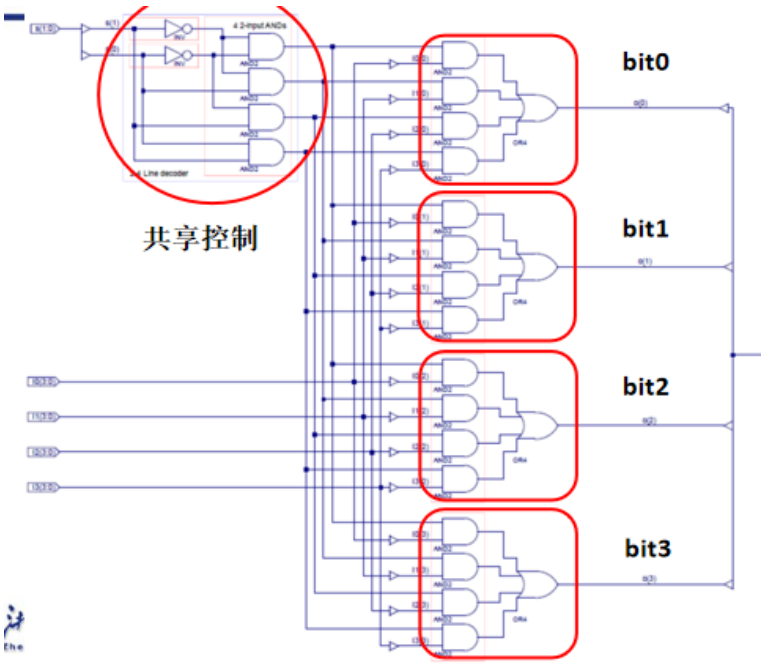


图 1: MUX4to1b4

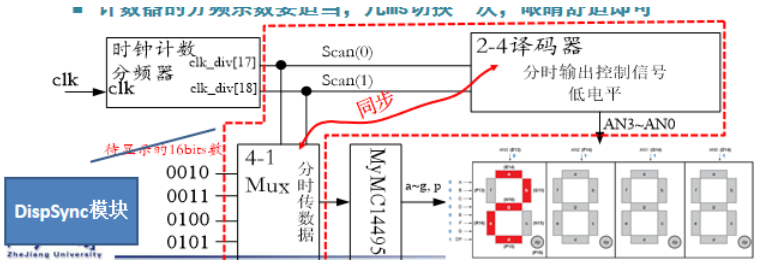
3. 记分板设计

(a) 记分板的操作方法: 用 BTNX4Y3～BTNX4Y0 这 4 个按钮，每个按钮按下

一次，对应的数码管的值加 1, 用 SW0～SW3 这 4 个开关控制每个数码管的小数点, 用 SW4～SW7 这 4 个开关控制每个数码管的消隐

(b) 实现 4 位 7 段数码管动态扫描显示：扫描信号来自时钟计数分频器：时序转化为组合电路, 由板载时钟 clk(100MHz) 作为计数器时钟，分频后的高两位信号（clk_div[18:17]）作为扫描控制信号 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）.

时钟计数分频器可输出 2-2^{32} 分频信号，可用于一般非同步类时钟信号



三、实验步骤和结果记录

3.1 数据选择器设计

3.1.1 四选一多路选择器:MUX4to1

1. 新建工程，工程名用 MUX4to1
2. 新建源文件，文件名称用 MUX4to1
3. 用原理图方式进行设计，原理图如下

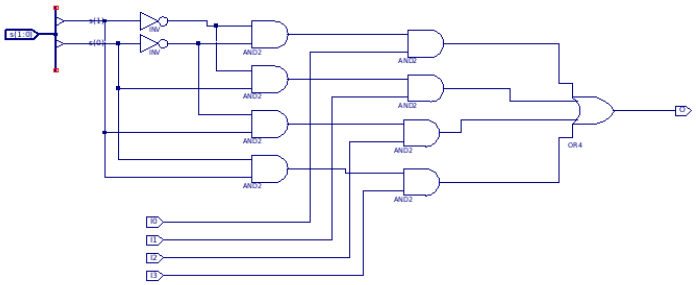


图 2: MUX4to1 原理图

4. 生成逻辑符号图

3.1.2 4 位四选一扩展：MUX4to1b4

1. 新建工程，工程名称用 MUX4to1b4_sch。
2. 新建源文件，文件名称用 MUX4to1b4。
3. 原理图方式进行设计。用 ISE 的 Edit->Change Sheet Size 菜单项来改变原理图绘图区的尺寸

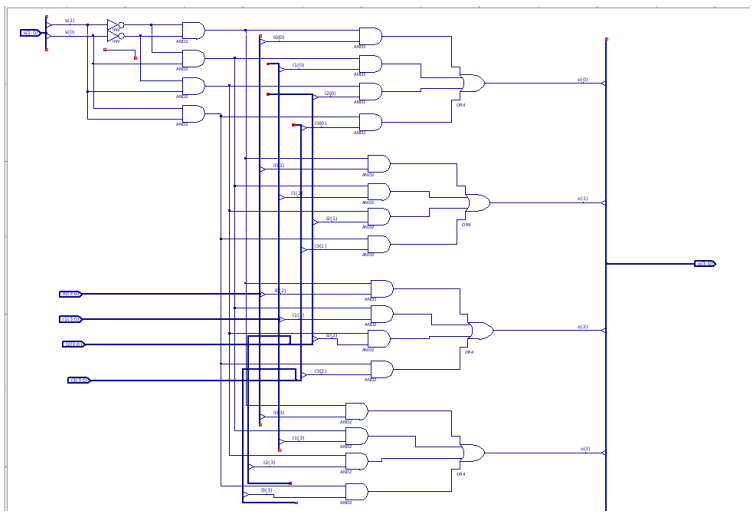


图 3: MUX4to1b4 原理图

4. Check Design Rules，检查错误
5. 对 MUX4to1b4 模块进行仿真，激励代码如下

```
‘timescale 1ns / 1ps
```

```
module Mux4to1b4_Mux4to1b4_sch_tb();

// Inputs
   reg [1:0] s;
   reg [3:0] I0;
   reg [3:0] I1;
   reg [3:0] I2;
   reg [3:0] I3;

// Output
   wire [3:0] o;

// Bidirs

// Instantiate the UUT
   Mux4to1b4 UUT (
                .s(s),
                .I0(I0),
                .I1(I1),
                .I2(I2),
                .I3(I3),
                .o(o)
   );
// Initialize Inputs
        integer i;
       initial begin
                s = 00;
                I0 = 1;
                I1 = 2;
                I2 = 4;
                I3 = 8;
                for(i=0;i<=4;i=i+1)begin
                        s<=i; #50;
                end
```

```
        end
endmodule
```

点击 Behavioral Check Synax，进行激励代码的查验。通过后，Process 窗口中选择 Simulate Behavioral Model，查看仿真激励波形。结果如下图:

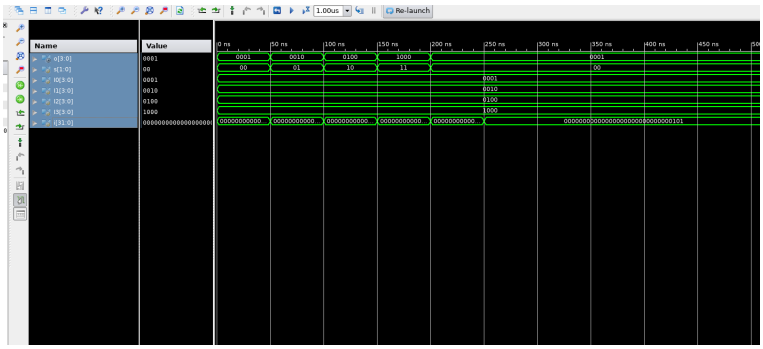


图 4: 波形图

6. 确认波形无误后，生成逻辑符号图

3.2 记分板设计

3.2.1 实现 4 位 7 段数码管动态扫描显示

1. 辅助模块: 时钟计数分频器

   (a) 模块名：clkdiv.v

   (b) 用 Verilog HDL 设计

   (c) 制作逻辑符号并修改：clkdiv.sym Verilog 代码如下：

```
module clkdiv(input clk,
                    input rst,
                    output reg[31:0]clkdiv
);

    always @(posedge clk or posedge rst) begin
            if(rst) clkdiv<=0;
            else clkdiv <= clkdiv+1'b1;
    end
```

endmodule

2. DisplaySync 模块设计

(a) 将之前设计好的模块的 sch 和 sym 文件复制到当前工程目录，并 add source。

(b) 用原理图形式设计, 并制作逻辑符号图

(c) 输入: 需要显示的 4 个 4 位二进制数 (Hex(15:0))，每位数码管的小数点 (point(3:0)), 每位数码管是否需要消隐 (LES(3:0)), 扫描控制信号 (Scan(1:0))
输出: 当前要显示的 4 位二进制数 (HEX(3:0)),4 位数码管的位选择信号 (AN(3:0))，小数点和消隐控制 (P,LE)。
设计的原理图如下

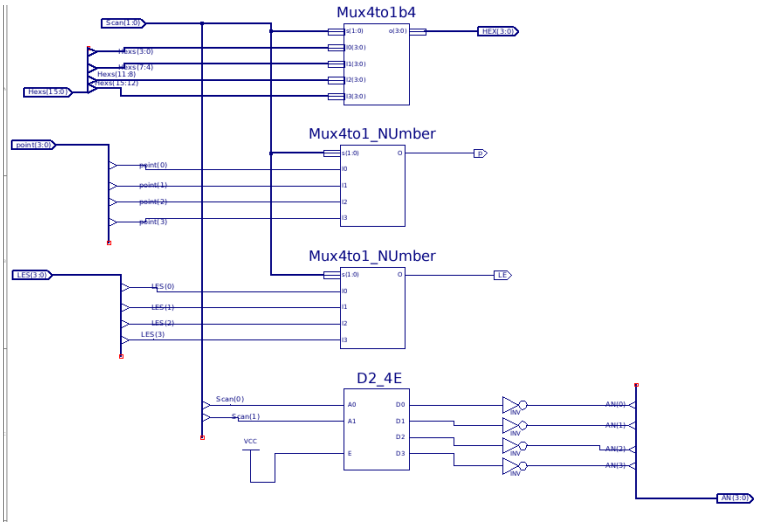


图 5: DisplaySync 原理图

3. DispNum 模块实现

(a) 将之前设计好的模块的 sch 和 sym 文件复制到当前工程目录，并 Add source

(b) 用原理图形式设计，并制作逻辑符号图

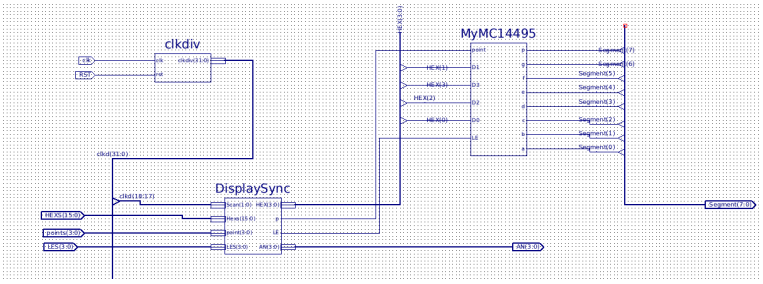


图 6: DispNum 原理图

4. CreateNumber 模块实现使用行为描述设计, 总共四个按键，各按一下，4 个 4 位 2 进制数分别加 1，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'b0101_1101_0011_1011;

 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
```

3.2.2 实现计分板功能 (各模块的汇总)

1. 新建工程，工程名称用 ScoreBoard

2. Top Level Source Type 用 HDL

3. 新建 Verilog 代码文件, 命名为 top，右键将该文件设置为 Top Module，只有最顶层 module 才能进行引脚约束操作

4. 将之前设计好的模块的 sch 和 sym 文件 (或者.v 文件) 复制到当前工程目录, 并 add source, 此时要再检查一下 top.v 是否还在 top module

5. 编写 Verilog 代码，连接 CreateNumber 和 DispNum 模块，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;
    CreateNumber c0(btn,num);

    DispNum d0(clk,num,SW[7:4],SW[3:0],1'b0,AN,SEGMENT);

    assign BTNX4=1'b0;

endmodule
```

6. 设计引脚约束文件

   (a) 七段数码管引脚约束

```
NET "SEGMENT[0]" LOC=AB22 | IOSTANDARD=LVCMOS33;#a
NET "SEGMENT[1]" LOC=AD24 | IOSTANDARD=LVCMOS33;#b
NET "SEGMENT[2]" LOC=AD23 | IOSTANDARD=LVCMOS33;#c
NET "SEGMENT[3]" LOC=Y21 | IOSTANDARD=LVCMOS33;#d
NET "SEGMENT[4]" LOC=W20 | IOSTANDARD=LVCMOS33;#e
NET "SEGMENT[5]" LOC=AC24 | IOSTANDARD=LVCMOS33;#f
NET "SEGMENT[6]" LOC=AC23 | IOSTANDARD=LVCMOS33;#g
NET "SEGMENT[7]" LOC=AA22 | IOSTANDARD=LVCMOS33;#point
NET "AN[0]" LOC=AD21 | IOSTANDARD=LVCMOS33;
NET "AN[1]" LOC=AC21 | IOSTANDARD=LVCMOS33;
```

```
NET "AN[2]" LOC=AB21 | IOSTANDARD=LVCMOS33;
NET "AN[3]" LOC=AC22 | IOSTANDARD=LVCMOS33;
```

(b) 数码管等开关的引脚约束

```
NET "SW[0]" LOC=AA10 | IOSTANDARD=LVCMOS15;
NET "SW[1]" LOC=AB10 | IOSTANDARD=LVCMOS15;
NET "SW[2]" LOC=AA13 | IOSTANDARD=LVCMOS15;
NET "SW[3]" LOC=AA12 | IOSTANDARD=LVCMOS15;
NET "SW[4]" LOC=Y13 | IOSTANDARD=LVCMOS15;
NET "SW[5]" LOC=Y12 | IOSTANDARD=LVCMOS15;
NET "SW[6]" LOC=AD11 | IOSTANDARD=LVCMOS15;
NET "SW[7]" LOC=AD10 | IOSTANDARD=LVCMOS15;
```

(c) 按钮和时钟的引脚约束

```
NET "btn[0]" LOC=W14 | IOSTANDARD=LVCMOS18;
NET "btn[0]" CLOCK_DEDICATED_ROUTE=FALSE;
NET "btn[1]" LOC=V14 | IOSTANDARD=LVCMOS18;
NET "btn[1]" CLOCK_DEDICATED_ROUTE=FALSE;
NET "btn[2]" LOC=V19 | IOSTANDARD=LVCMOS18;
NET "btn[2]" CLOCK_DEDICATED_ROUTE=FALSE;
NET "btn[3]" LOC=V18 | IOSTANDARD=LVCMOS18;
NET "btn[3]" CLOCK_DEDICATED_ROUTE=FALSE;
NET "clk" LOC=AC18 | IOSTANDARD=LVCMOS18;
```

7. 生成.bit 文件并下载到 sword 板上验证

3.3 下载验证

首先验证小数点的功能是否正常, 对应的控制的开关为最右边四个，由下图可以看到，结果符合预期

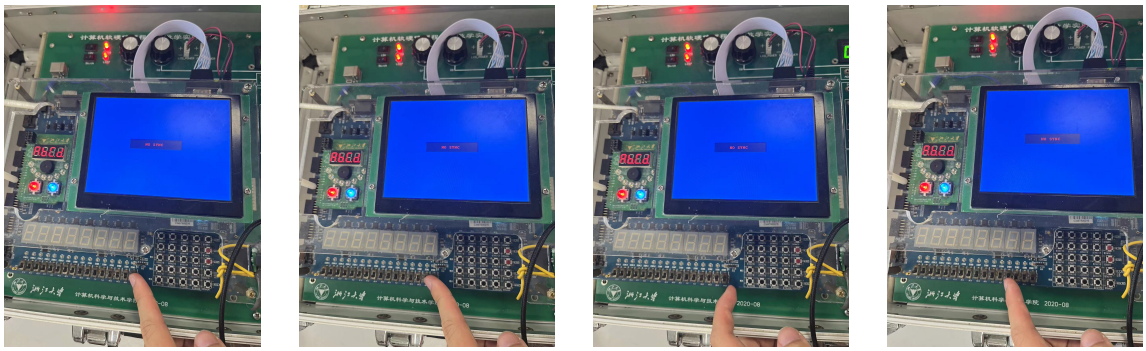


图 7: 验证小数点

接下来验证每位数码管的消隐，对应从右往左数第 5——8 个开关，由下图可以看到结果符合预期

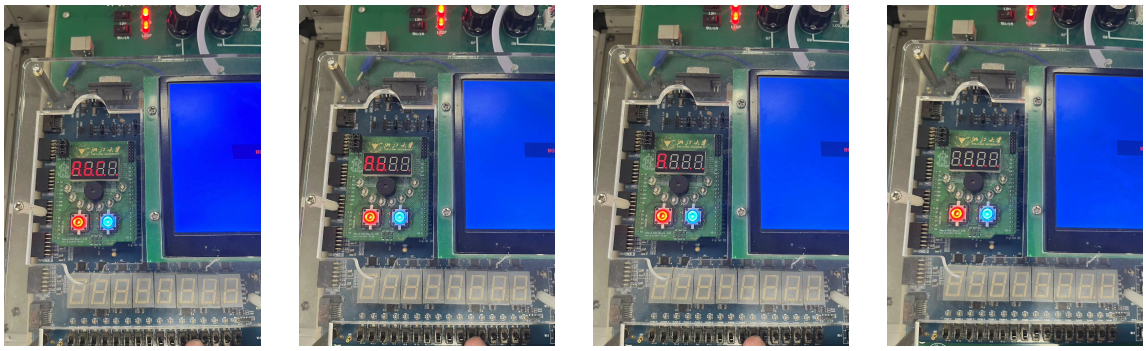


图 8: 验证数码管的消隐

最后验证记分板功能，即按下按钮对应数码管示数增加，但是数字增加可能不止一位，是正常现象，如果想要只增加一，需要设计加入放抖动模块，或者将输入接到开关上。但总体来说，结果符合预期

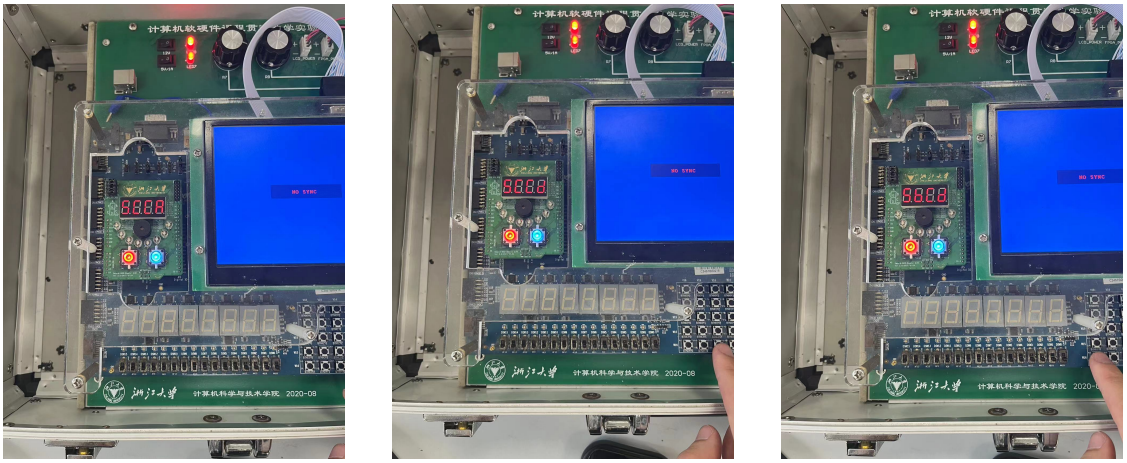


图 9: 验证记分板功能

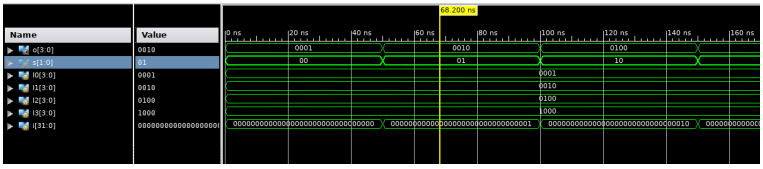
四、实验结果分析

相关结果都已经在前文写出。实验结果基本符合要求：仿真激励波形与真值表都相对应；下载到 sword 板上后，结果都与真值表相符。

1. 分析硬件描述代码对于 top.v 的代码，我们在之前设计的模块相当于是一个元器件，括号里面的内容是此元器件的输入输出接口。对于 top module 来说，接口用于在引脚约束以后对应于开发板上的按钮，LED 灯等器件，供我们在开发板上手动操作并观察效果。对于其它元器件来说，接口是供调用使用的。

   接下来的 wire[15:0]num 就相当于函数中定义的变量，它先经过 CreateNumber 被赋值，然后作为 DispNum 的参数进行输出显示。

2. 分析仿真结果：以 s=01(2) 为例



   s 作为选择器，此时信号为 01，代表选择 I1 输出，故输出 o 应该等于 I1, 而仿真波形的结果符合预期，当 s 为 00，10，11 时同理.

3. 分析开发板结果相关结果在第三部分给出，结果均符合预期。但是这里有一个问题就是开关在闭合及断开的瞬间均伴随有一连串的抖动，该开关通常为机械弹性开关，当机械触点断开，闭合时，由于机械触点的弹性作用，一个按键开关在闭合时不会马上稳定地接通，在断开时也不会一下子断开。因此数字的变化不是连续的。一个解决方法就是防抖动，可以在之前实验的基础上加上防抖动模块

```
module pbdebounce(
input wire clk,
input wire button,
output reg pbreg
);

reg [7:0] pbshift;
```

```
        always@(posedge clk) begin
                pbshift = pbshift<<1;
                pbshift[0] = button;
                if (pbshift == 8'b0)
                        pbreg=0;
                if (pbshift == 8'hFF)
                        pbreg=1;
        end
endmodule
```

五、讨论与心得

本次实验相对于前几次来说更复杂了，难度也更大了。在这次实验的设计过程中，对于这么多模块我一度无从下手，但是发现重新分析一下原理以及各模块之间的关系，其实也不算太复杂。总之在做实验的过程中，我学习到了多模块工程的思维，了解了各模块的构建规则与使用方法，对 ISE 的使用与了解更深入了，对 Verilog 代码也更熟悉了。这次实验我也提前完成了设计，并且由于模块较多担心出现问题，我也提前到实验室进行上板验证，但结果符合预期，没有出问题！所以我也提前通过了验收，增加了我的信心，让我更加期待下一次计逻实验，更深入理解计算机逻辑设计的奥妙。