

# 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:	王晓宇
学 院:	竺可桢学院
专 业:	计算机
邮 箱:	<a href="mailto:1657946908@qq.com">1657946908@qq.com</a>
QQ 号:	1657946908
电 话:	19550222634
指导教师:	洪奇军
报告日期:	2023 年 11 月 23 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： 同步时序电路设计

学生姓名： 王晓宇 学号： 3220104364 同组学生姓名：

实验地点： 紫金港东四 509 室 实验日期： 2023 年 11 月 23 日

## 一、操作方法与实验步骤

实验简介：

4 位同步二进制计数器：

### 4位同步二进制计数器状态表



#### 状态变化条件

- ☑ 计数触发
- ☑ 无外部输入

#### 输入激励

- ☑ 满足次态的输入要求
- ☑ 输入方程

#### 状态分配

- ☑ 计数值决定

#### 触发器选择

- ☑ D 触发器
- ☑ 4 个

	当前状态(现态)				下一状态(次态)				触发器激励(输入)			
	$Q_A$	$Q_B$	$Q_C$	$Q_D$	$Q_A^{n+1}$	$Q_B^{n+1}$	$Q_C^{n+1}$	$Q_D^{n+1}$	$D_A$	$D_B$	$D_C$	$D_D$
0	0	0	0	0	1	0	0	0	1	0	0	0
1	1	0	0	0	0	1	0	0	0	1	0	0
2	0	1	0	0	1	1	0	0	1	1	0	0
3	1	1	0	0	0	0	1	0	0	0	1	0
4	0	0	1	0	1	0	1	0	1	0	1	0
5	1	0	1	0	0	1	1	0	0	1	1	0
6	0	1	1	0	1	1	1	0	1	1	1	0
7	1	1	1	0	0	0	0	1	0	0	0	1
8	0	0	0	1	1	0	0	1	1	0	0	1
9	1	0	0	1	0	1	0	1	0	1	0	1
10	0	1	0	1	1	1	0	1	1	1	0	1
11	1	1	0	1	0	0	1	1	0	0	1	1
12	0	0	1	1	1	0	1	1	1	0	1	1
13	1	0	1	1	0	1	1	1	0	1	1	1
14	0	1	1	1	1	1	1	1	1	1	1	1
15	1	1	1	1	0	0	0	0	0	0	0	0

D 触发器原理，在 clk 作用下  $Q = D$ ，4 位计数器的 Q 和 D 关系如下表

	<i><b>QA</b></i>	<i><b>QB</b></i>	<i><b>QC</b></i>	<i><b>QD</b></i>	<i><b>DA</b></i>	<i><b>DB</b></i>	<i><b>DC</b></i>	<i><b>DD</b></i>
<b>0</b>	0	0	0	0	1	0	0	0
<b>1</b>	1	0	0	0	0	1	0	0
<b>2</b>	0	1	0	0	1	1	0	0
<b>3</b>	1	1	0	0	0	0	1	0
<b>4</b>	0	0	1	0	1	0	1	0
<b>5</b>	1	0	1	0	0	1	1	0
<b>6</b>	0	1	1	0	1	1	1	0
<b>7</b>	1	1	1	0	0	0	0	1
<b>8</b>	0	0	0	1	1	0	0	1
<b>9</b>	1	0	0	1	0	1	0	1
<b>10</b>	0	1	0	1	1	1	0	1
<b>11</b>	1	1	0	1	0	0	1	1
<b>12</b>	0	0	1	1	1	0	1	1
<b>13</b>	1	0	1	1	0	1	1	1
<b>14</b>	0	1	1	1	1	1	1	1
<b>15</b>	1	1	1	1	0	0	0	0

$$D_A = \overline{Q_A}$$

$$\begin{aligned} D_B &= \overline{Q_A}Q_B + Q_A\overline{Q_B} \\ &= \overline{Q_A \oplus Q_B} \end{aligned}$$

$$\begin{aligned} D_C &= \overline{Q_A}Q_C + \overline{Q_B}Q_C + Q_AQ_B\overline{Q_C} \\ &= \overline{(\overline{Q_A} + \overline{Q_B}) \oplus Q_C} \end{aligned}$$

$$\begin{aligned} D_D &= \overline{Q_A}Q_D + \overline{Q_B}Q_D + \overline{Q_C}Q_D + Q_AQ_BQ_C\overline{Q_D} \\ &= \overline{(\overline{Q_A} + \overline{Q_B} + \overline{Q_C}) \oplus Q_D} \end{aligned}$$

$$R_C = \overline{Q_A} + \overline{Q_B} + \overline{Q_C} + \overline{Q_D}$$

### 任务 1：原理图方式设计 4 位同步二进制计数器

►新建工程

工程名称用 MyCounter。Top Level Source Type 用 HDL

New Project Wizard

**Create New Project**  
Specify project location and type.

Enter a name, locations, and comment for the project

Name: MyCounter

Location: /tmp/Logic/MyCounter

Working Directory: /tmp/Logic/MyCounter

Description:

Select the type of top-level source for the project

Top-level source type: HDL

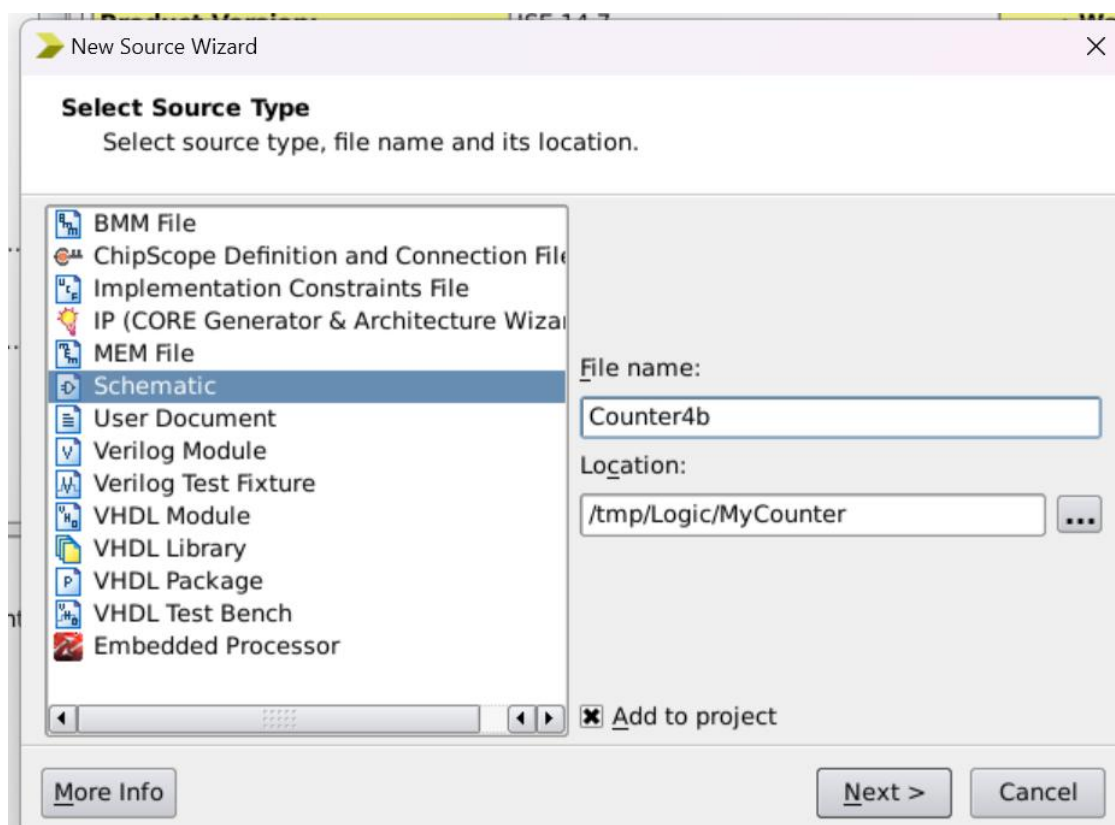
More Info

Next >

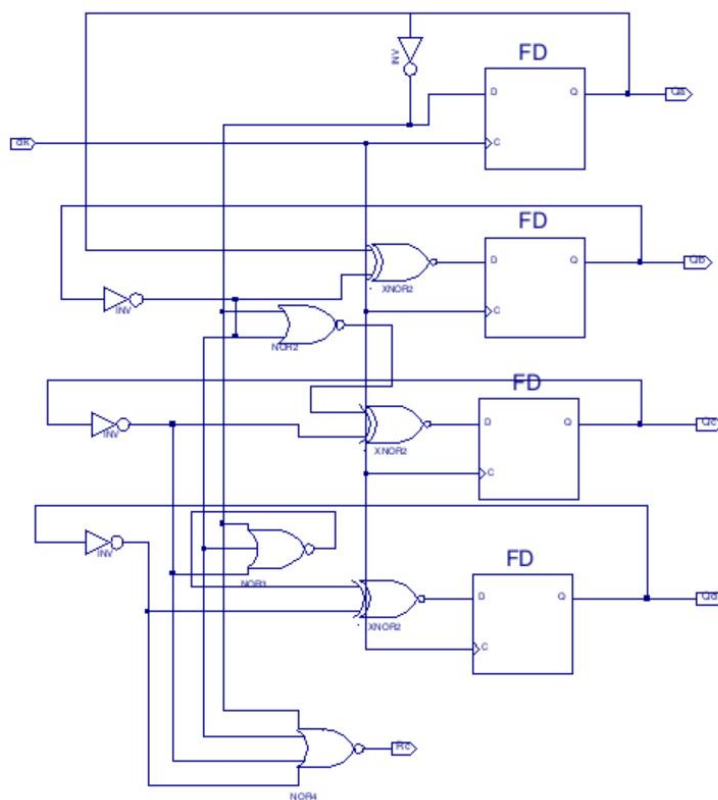
Cancel

➤新建源文件

类型是 Schematic 文件名称用 Counter4b



➤原理图方式进行设计



➤进行波形仿真

```
module Counter4b_Counter4b_sch_tb();

// Inputs
reg clk;

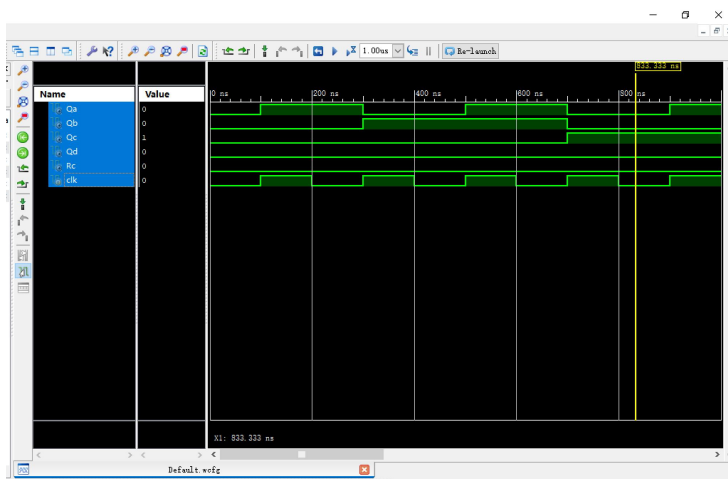
// Output
wire Qa;
wire Qb;
wire Qc;
wire Qd;
wire Rc;

// Bidirs

// Instantiate the UUT
Counter4b UUT (
    .clk(clk),
    .Qa(Qa),
    .Qb(Qb),
    .Qc(Qc),
    .Qd(Qd),
    .Rc(Rc)
);

// Initialize Inputs

always begin
    clk = 0; #20;
    clk = 1; #20;
end
endmodule
```



➤新建源文件，用作时钟

类型是 Verilog

文件名称用 clk\_1s

```
///////////////////////////////////////////////////
module clk_1s(
    input wire clk,
    output reg clk_1s
);
    reg [31:0] cnt;
    always @(posedge clk) begin
        if(cnt < 50_000_000) begin
            cnt<=cnt +1;
        end else begin
            cnt <=0;
            clk_1s <= ~clk_1s;
        end
    end
end

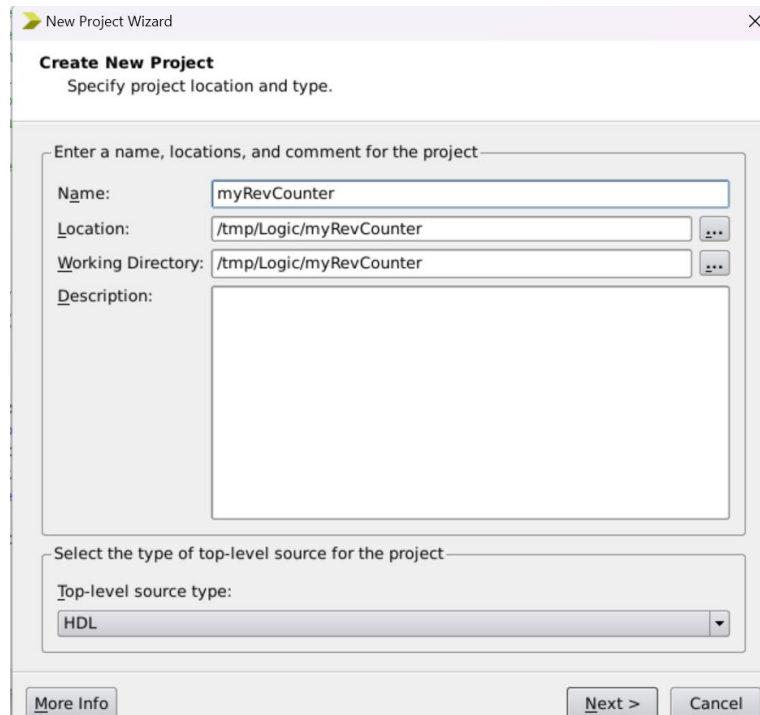
endmodule
```

## 任务 2：以 Verilog 行为描述方式设计 16 位可逆二进制同步计数器

➤新建工程

工程名称用 myRevCounter

Top Level Source Type 用 HDL



➤新建源文件

类型是 Verilog

文件名称用 RevCounter

结构化描述方式进行设计

```
0 //////////////////////////////////////////////////
1 module RevCounter(
2     input wire clk,
3     input wire s,
4     output reg [15:0] cnt,
5     output Rc
6 );
7     initial begin cnt = 0;
8     end
9     assign Rc = (~s & (~cnt)) | (s & cnt);
10    always @(posedge clk) begin
11        if(s)begin
12            cnt <= cnt + 1'b1;
13        end else begin
14            cnt <= cnt - 1'b1;
15        end
16    end
17
18 endmodule
```

➤波形仿真（包含正向计数和反向计数）

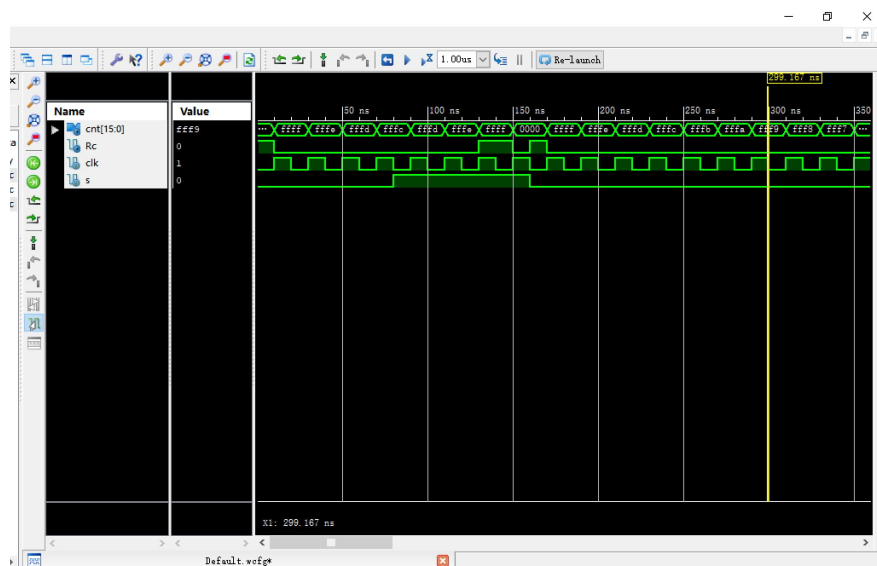
```
module test_rever;

    // Inputs
    reg clk;
    reg s;

    // Outputs
    wire [15:0] cnt;
    wire Rc;

    // Instantiate the Unit Under Test (UUT)
    RevCounter uut (
        .clk(clk),
        .s(s),
        .cnt(cnt),
        .Rc(Rc)
    );

    always begin
        s = 0;#1600;
        s = 1;#1600;
    end
    always begin
        clk = 0;#20;
        clk = 1;#20;
    end
endmodule
```



➤新建源文件，设计 100ms 时钟

类型是 Verilog

文件名称用 clk\_100ms

```
//////////////////////////////////////////
module clk_100ms(
    input wire clk,
    output reg clk_100ms
);
    reg [31:0] cnt;
    always @(posedge clk)begin
        if(cnt < 5000_000)begin
            cnt<=cnt +1;
        end else begin
            cnt <=0;
            clk_100ms <= ~clk_100ms;
        end
    end
endmodule
```



### 任务 3: Top 模块创建

```

////////////////////////////////////
module Top(
    input wire clk,
    input wire SW,
    output wire [7:0] SEGMENT,
    output wire [3:0] AN,
    output wire led,
    output ledclk,
    output ledsout,
    output ledclrn,
    output LEDEN
);
    wire [15:0] cnt;
    wire clk_100ms;
    wire [31:0] div;
    clk_100ms M1(.clk(clk),.clk_100ms(clk_100ms));
    RevCounter M2(.clk(clk_100ms),.s(SW),.cnt(cnt),.Rc(led));
    disp_num M3(.clk(clk),.HEXS(cnt),.LES(4'b0000),.points(4'b0000),.RST(1'b0),.AN(AN),.segment(SEGMENT));
    clkdiv M4(.clk(clk),.clkdiv(div),.rst(1'b0));
    LEDP2S #(DATA_BITS(16),.DATA_COUNT_BITS(4),.DIR(0))
        U7(.clk(clk),.rst(1'b0),.Start(div[20]),.PData(cnt),.sclk(ledclk),.sclrn(ledclrn),.sout(ledsout),.EN(LEDEN));
endmodule

```

```

// Netlist for Top module
NET "clk" LOC = AC18 | IOSTANDARD=LVCMS18;
// NET "BTN[0]" LOC=W14 | IOSTANDARD=LVCMS18;
// NET "BTN[0]" clock_dedicated_route = false;
// NET "BTN[1]" LOC=V14 | IOSTANDARD=LVCMS18;
// NET "BTN[1]" clock_dedicated_route = false;
// NET "BTN4" LOC=W16 | IOSTANDARD=LVCMS18;
NET "led" LOC = W23 | IOSTANDARD=LVCMS33;
NET "ledclk" LOC = N26 | IOSTANDARD = LVCMS33;
NET "ledclrn" LOC = N24 | IOSTANDARD = LVCMS33;
NET "ledsout" LOC = M26 | IOSTANDARD = LVCMS33;
NET "LEDEN" LOC = P18 | IOSTANDARD = LVCMS33;
NET "SW" LOC=AF10 | IOSTANDARD=LVCMS15;
// NET "SW1[1]" LOC=AF13 | IOSTANDARD=LVCMS15;
// NET "SW2[0]" LOC=AE13 | IOSTANDARD=LVCMS15;
// NET "SW2[1]" LOC=AF8 | IOSTANDARD=LVCMS15;



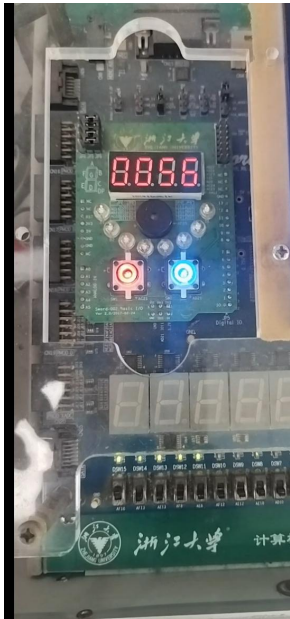
// NET "SW[0]" LOC=AA10 | IOSTANDARD=LVCMS15;
// NET "SW[1]" LOC=AB10 | IOSTANDARD=LVCMS15;
// NET "SW[2]" LOC=AA13 | IOSTANDARD=LVCMS15;
// NET "SW[3]" LOC=AA12 | IOSTANDARD=LVCMS15;
// NET "SW[4]" LOC=Y13 | IOSTANDARD=LVCMS15;
// NET "SW[5]" LOC=Y12 | IOSTANDARD=LVCMS15;
// NET "SW[6]" LOC=AD11 | IOSTANDARD=LVCMS15;
// NET "SW[7]" LOC=AD10 | IOSTANDARD=LVCMS15;
// NET "SW[8]" LOC=AE10 | IOSTANDARD=LVCMS15;
// NET "SW[9]" LOC=AE12 | IOSTANDARD=LVCMS15;
// NET "SW[10]" LOC=AF12 | IOSTANDARD=LVCMS15;
// NET "SW[11]" LOC=AE8 | IOSTANDARD=LVCMS15;

NET "SEGMENT[0]" LOC=AB22 | IOSTANDARD=LVCMS33; #a
NET "SEGMENT[1]" LOC=AD24 | IOSTANDARD=LVCMS33; #b
NET "SEGMENT[2]" LOC=AD23 | IOSTANDARD=LVCMS33; #c
NET "SEGMENT[3]" LOC=Y21 | IOSTANDARD=LVCMS33; #d
NET "SEGMENT[4]" LOC=W20 | IOSTANDARD=LVCMS33; #e
NET "SEGMENT[5]" LOC=AC24 | IOSTANDARD=LVCMS33; #f
NET "SEGMENT[6]" LOC=AC23 | IOSTANDARD=LVCMS33; #g
NET "SEGMENT[7]" LOC=AA22 | IOSTANDARD=LVCMS33; #point
NET "AN[0]" LOC=AD21 | IOSTANDARD=LVCMS33;
NET "AN[1]" LOC=AC21 | IOSTANDARD=LVCMS33;
NET "AN[2]" LOC=AB21 | IOSTANDARD=LVCMS33;
NET "AN[3]" LOC=AC22 | IOSTANDARD=LVCMS33;

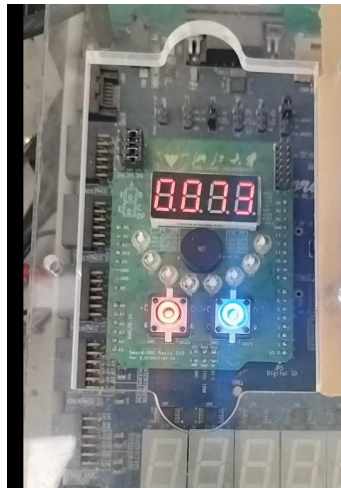
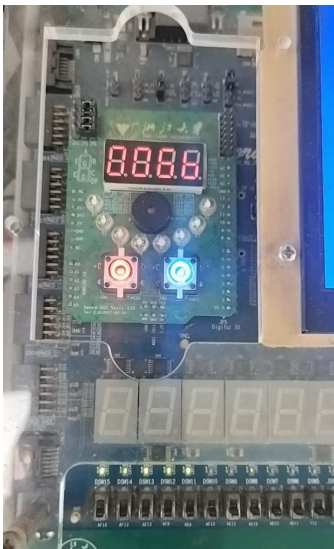
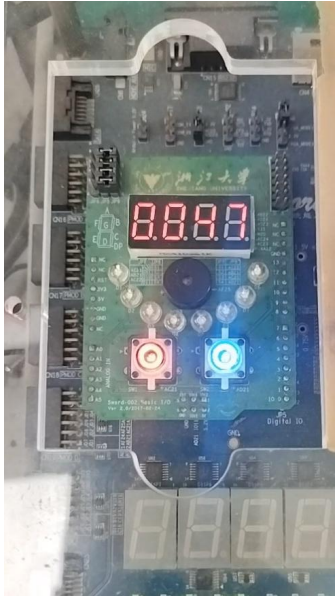
```

二、实验结果与分析

自增功能

0038	0044	0056
		

自减功能

0073	006b	0047
		

在 bit 文件运行后 后示数即开始自增/自减。  
在递减时，数值显示 0000 时进位灯亮；在递增时，数值显示 FFFF 时进位灯亮。符合预期。

### 三、讨论、心得

这次实验是做了三个函数，一个是自增的函数，另一个是可逆的计数器，另外还有一个分频器来达成计时的目的。总体看来是比较简单的，但是两个计数器的形成一个是通过画图得来，另一个是通过行为描述语言来形成的。对于线束比较多的直接连线非常复杂，而且不一定有很高的正确率，此时用行为描述语言就很方便了。但是经过实践后发现如此写完的模块如果形成 `sym` 文件的话，里面的硬件会很复杂，基本不是我们常见的与门或门之类的，其运行效率有待验证。

做实验的心得如果说总结一下的话，大概就是：

1. 重启能解决 70% 问题
2. 要理解实验目的再开始编写工程
3. 做实验记得带 U 盘
4. 虚拟机别安装到移动硬盘里面，一旦接触不稳就虚拟机自锁了哈哈哈哈哈哈哈哈呜呜呜呜呜呜呜呜（别问我怎么知道的 T——T
5. 如果空间允许的话，尽量使用 windows 下放的 WSL-Linux 子系统，运行依靠本地配置非常流畅，可能有点不足是本地 Simulation 有点困难，需要安装 GCC，但是国内接不到在美国的网站，镜像站操作又过于麻烦（嗯，所以办法也会有的，大家慢慢探索
6. 不懂就问老师，非常贴心的老师交流会给你的项目带来一线生机