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 24 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：同步时序电路设计

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 23 日

一、 实验目的

- 掌握典型同步时序电路的工作原理和设计方法
- 掌握时序电路的激励函数、状态图、状态方程的运用
- 掌握用 Verilog 进行有限状态机的设计、调试、仿真
- 掌握用 FPGA 实现时序电路功能

二、操作方法与实验步骤

1. 原理图方式设计 4 位同步二进制计数器

实验原理

4 位二进制同步计数器的真值表如图所示（从 0 开始每次加 1，15 后面发生进位并变为 0）：

	Q_A	Q_B	Q_C	Q_D	D_A	D_B	D_C	D_D
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	1	0	0
4	0	1	0	0	0	1	0	1
5	0	1	0	1	0	1	1	0
6	0	1	1	0	0	1	1	1
7	0	1	1	1	1	0	0	0
8	1	0	0	0	1	0	0	1
9	1	0	0	1	1	0	1	0
10	1	0	1	0	1	0	1	1
11	1	0	1	1	1	1	0	0
12	1	1	0	0	1	1	0	1
13	1	1	0	1	1	1	1	0
14	1	1	1	0	1	1	1	1
15	1	1	1	1	0	0	0	0

激励函数如下：

$$D_A = Q_B Q_C Q_D \oplus Q_A$$
$$D_B = Q_C Q_D \oplus Q_B$$
$$D_C = Q_D \oplus Q_C$$
$$D_D = \overline{Q_D}$$

进位 RC 的输出函数为：

$$R_C = Q_A Q_B Q_C Q_D$$

实验步骤

- 在 Digital 里新建 Counter4b 原理图，用原理图方式进行设计，并设计测试用例进行仿真
- 导出 Counter4b.v
- 在 Vivado 中新建工程 MyCounter
- 加入 Counter4b.v
- 设计仿真波形文件进行行为仿真
- 新建 counter_1s.v 源文件，输出秒时钟
- 新建 top.v 源文件，用行为描述进行设计
- 设计 top 文件的仿真波形代码，进行行为仿真
- 在开发板上进一步验证

2. 以 Verilog 行为描述方式设计 16 位可逆二进制同步计数器

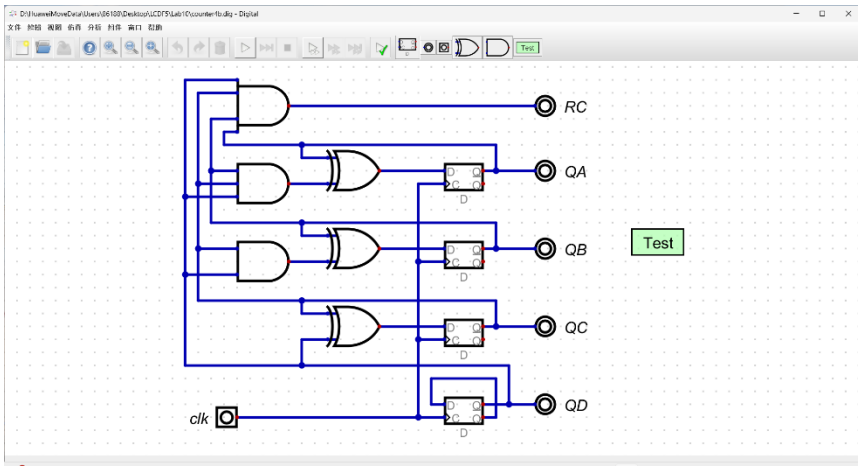
实验步骤

- 新建工程 myRevCounter
- 新建源文件，设计 16 位可逆同步二进制计数器
- 波形仿真（要求仿真出 Rc 在正向计数进位和反向计数借位的输出波形）
- 新建源文件，设计 100ms 时钟
- 新建顶层模块 Top，用行为描述进行设计
- 在开发板上进行验证

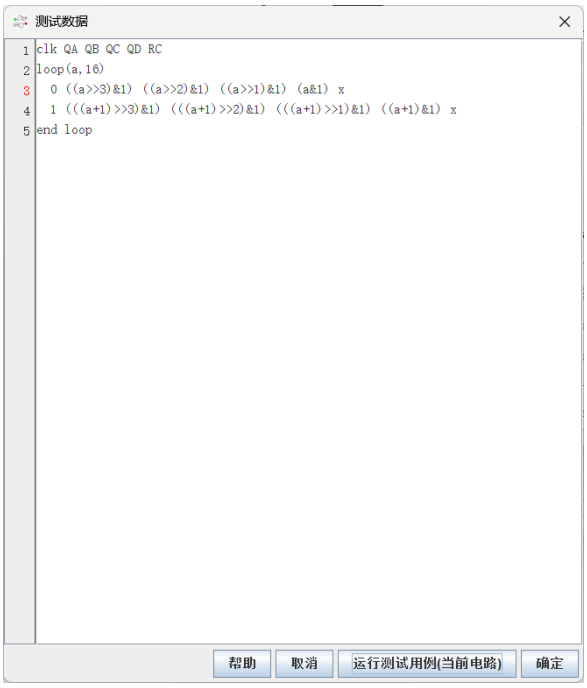
三、实验结果与分析

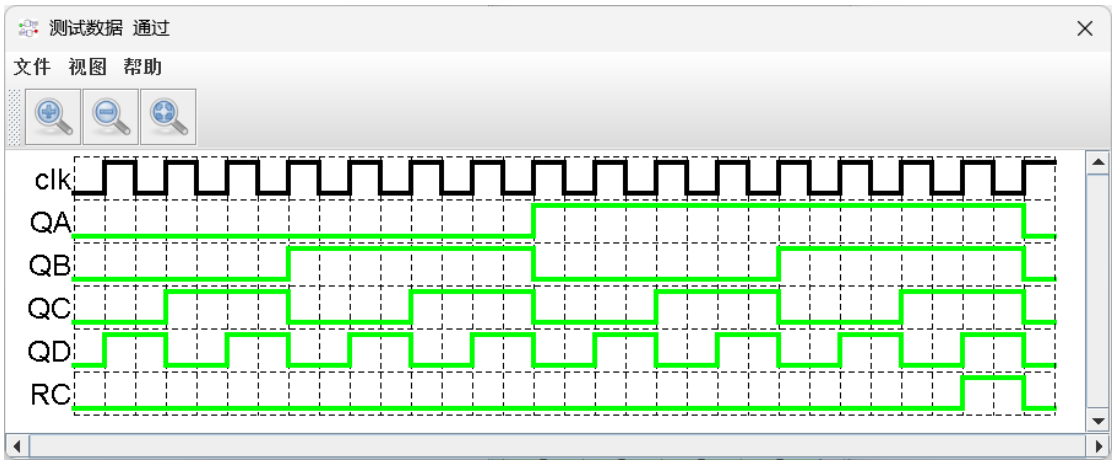
1. 原理图方式设计 4 位同步二进制计数器

按照原理图搭建的电路 Counter4b 如图所示：



输入时钟信号，结果如下（已加入预期结果）：





其中，每当 clk 信号从 0 变为 1（上升沿）时，计数器的值均会加一。当计数器的值达到 15 时，发生进位且计数器的值归零，符合预期。

将该电路导出为 Counter4b.v：

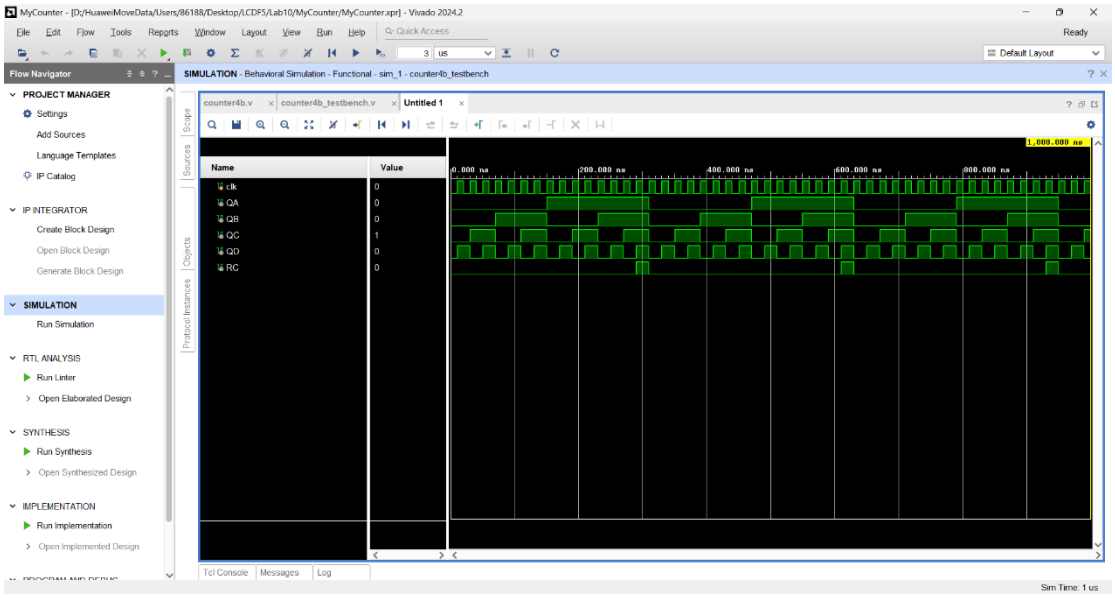
```
/*
 * Generated by Digital. Don't modify this file!
 * Any changes will be lost if this file is regenerated.
 */
module DIG_D_FF_1bit
#(
    parameter Default = 0
)
(
   input D,
   input C,
   output Q,
   output \~Q
);
    reg state;

    assign Q = state;
    assign \~Q = ~state;

    always @ (posedge C) begin
        state <= D;
    end

    initial begin
        state = Default;
    end
endmodule


module counter4b (
  input clk,
  output QA,
  output QB,
  output QC,
  output QD,
  output RC
);
  wire s0;
  wire QA_temp;
  wire s1;
  wire QB_temp;
  wire s2;
  wire QC_temp;
  wire s3;
```

```
  wire QD_temp;
  DIG_D_FF_1bit #(
    .Default(0)
  )
  DIG_D_FF_1bit_i0 (
    .D( s0 ),
    .C( clk ),
    .Q( QA_temp )
  );
  DIG_D_FF_1bit #(
    .Default(0)
  )
  DIG_D_FF_1bit_i1 (
    .D( s1 ),
    .C( clk ),
    .Q( QB_temp )
  );
  DIG_D_FF_1bit #(
    .Default(0)
  )
  DIG_D_FF_1bit_i2 (
    .D( s2 ),
    .C( clk ),
    .Q( QC_temp )
  );
  DIG_D_FF_1bit #(
    .Default(0)
  )
  DIG_D_FF_1bit_i3 (
    .D( s3 ),
    .C( clk ),
    .Q( QD_temp ),
    .\~Q ( s3 )
  );
  assign s0 = (QA_temp ^ (QB_temp & QC_temp & QD_temp));
  assign s1 = (QB_temp ^ (QC_temp & QD_temp));
  assign s2 = (QC_temp ^ QD_temp);
  assign RC = (QD_temp & QC_temp & QB_temp & QA_temp);
  assign QA = QA_temp;
  assign QB = QB_temp;
  assign QC = QC_temp;
  assign QD = QD_temp;
endmodule
```

并在 vivado 平台上进一步验证其正确性，激励代码如下：

```
`timescale 1ns / 1ps
module counter4b_testbench();
    reg clk;
    wire QA;
    wire QB;
    wire QC;
    wire QD;
    wire RC;
    counter4b uut(.clk(clk),.QA(QA),.QB(QB),.QC(QC),.QD(QD),.RC(RC));
    always begin
        clk=0;#10;
        clk=1;#10;
    end
endmodule
```

仿真结果为：



可以看到 QA,QB,QC,QD 的变化规律与 Digital 上的相同，且当计数器的值为 15 时，RC 变为 1，下一个值为 0，符合预期。

接下来将计数器的数字进行显示，需要添加之前的文件 clkdiv.v,DispNum.v（不再重复展示），并新建 counter_1s.v 及 top.v。

counter_1s.v 的代码如下：

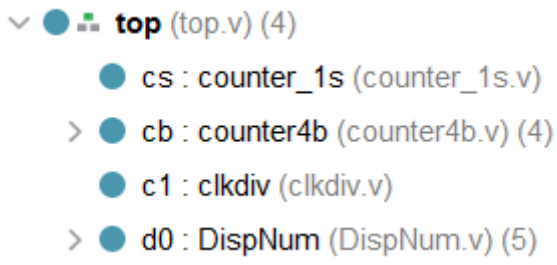
```
module counter_1s(clk, clk_1s);
input wire clk;
output reg clk_1s;
reg [31:0] cnt;
initial clk_1s = 0;
always @ (posedge clk) begin
   if (cnt < 50_000_000) begin
       cnt <= cnt + 1'b1;
```

```
   end else begin
       cnt <= 0;
       clk_1s <= ~clk_1s;
   end
end
endmodule
```

top.v 的代码如下：

```
`timescale 1ns / 1ps
module top(
    input wire clk,
    output wire [7:0] LED,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT
);
    wire QA;
    wire QB;
    wire QC;
    wire QD;
    wire clk_1s;
    wire [31:0] clk_div;
    wire [15:0] disp_hexs;
    counter_1s cs(.clk(clk),.clk_1s(clk_1s));
    counter4b
cb(.clk(clk_1s),.QA(QA),.QB(QB),.QC(QC),.QD(QD),.RC(LED[0]));
    assign disp_hexs[3:0]={QA,QB,QC,QD};
    assign disp_hexs[15:4]=12'b0;
    clkdiv c1(.clk(clk), .rst(1'b0), .clk_div(clk_div));
    DispNum d0(.scan(clk_div[18:17]), .HEXS(disp_hexs),
     .LES(4'b1110), .point(4'b0), .AN(AN), .SEGMENT(SEGMENT));
endmodule
```

文件结构如图所示：

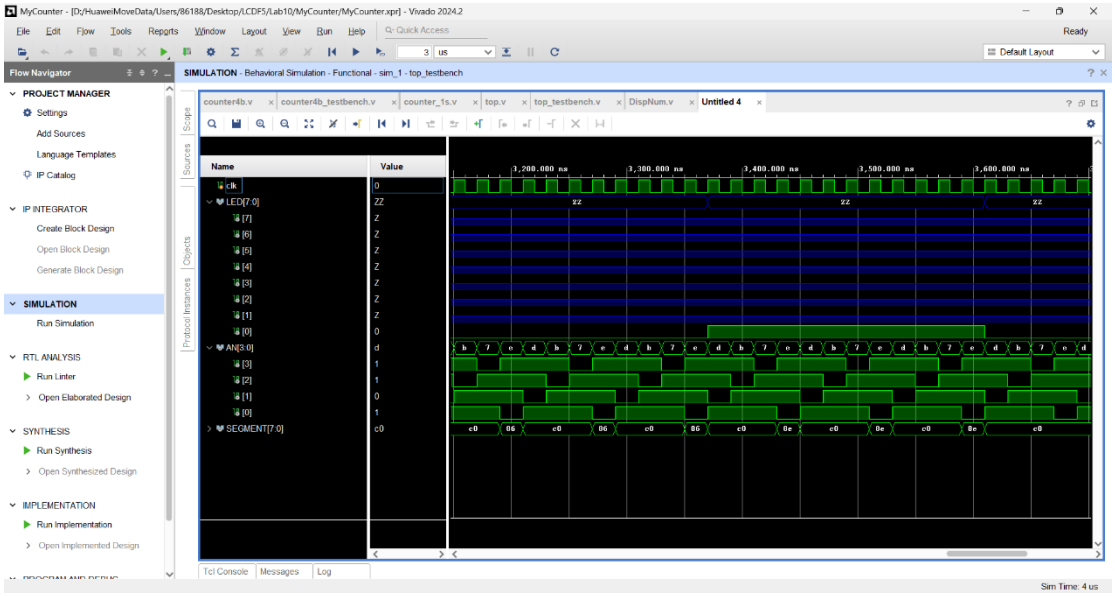


对 top.v 进行仿真，激励代码如下：

```
`timescale 1ns / 1ps
module top_testbench();
    reg clk;
    wire [7:0] LED;
    wire [3:0] AN;
```

```
    wire [7:0] SEGMENT;
    top uut(.clk(clk),.LED(LED),.AN(AN),.SEGMENT(SEGMENT));
    always begin
        clk=0;#10;
        clk=1;#10;
    end
endmodule
```

仿真时，需要将 clk[18:17]改为 clk[1:0]，并把 cnt<50_000_000 改为 cnt<5，仿真结果如下：



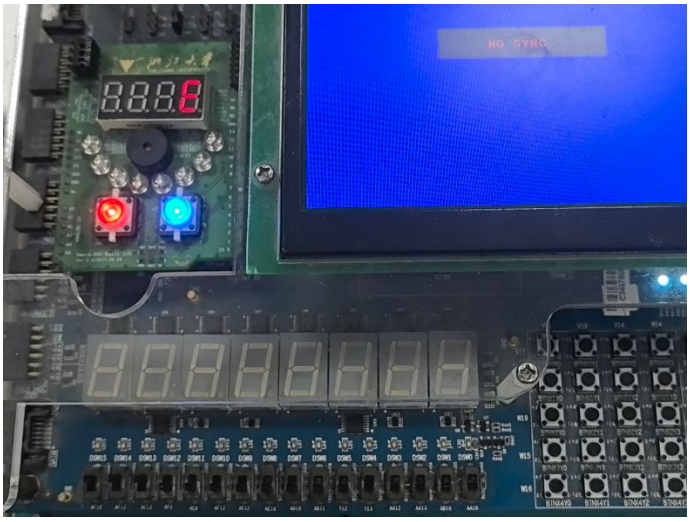
可以看到，当计数器的值为 15 时，LED[0]持续亮了一段时间，这是因为计时器的时钟周期为 100ns，其输出结果符合预期。

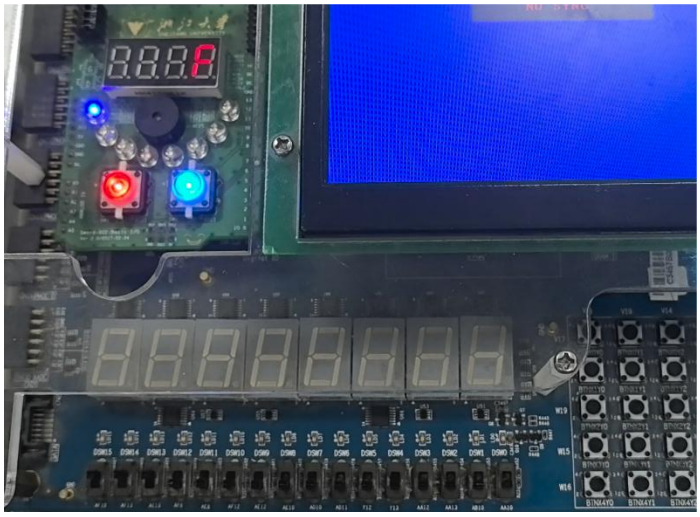
最后进行上板验证，其引脚约束文件如下：

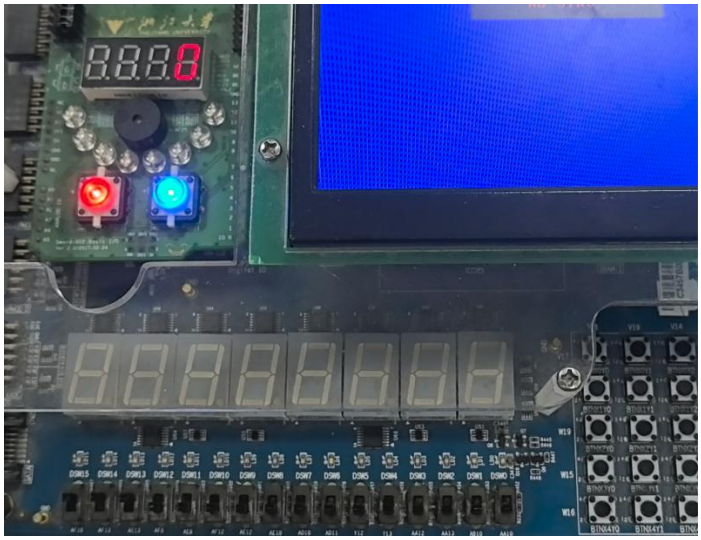
```
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]
set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]
set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]
set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]
set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]
set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]
set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]
set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]
set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]
create_clock -name clk100MHZ -period 10.0 [get_ports clk]
set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]
    set_property PACKAGE_PIN W23 [get_ports {LED[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[0]}]
    set_property PACKAGE_PIN AB26 [get_ports {LED[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[1]}]
    set_property PACKAGE_PIN Y25 [get_ports {LED[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[2]}]
    set_property PACKAGE_PIN AA23 [get_ports {LED[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[3]}]
    set_property PACKAGE_PIN Y23 [get_ports {LED[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[4]}]
    set_property PACKAGE_PIN Y22 [get_ports {LED[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[5]}]
    set_property PACKAGE_PIN AE21 [get_ports {LED[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[6]}]
    set_property PACKAGE_PIN AF24 [get_ports {LED[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {LED[7]}]
```

下载后可以看到计数器的数字每一秒钟都会增加 1 或从 F 变为 0，以下是视频中

的典型照片：







这一组照片即为数字从 E 增加到 F 后亮灯，再从 F 回到 0 后灭灯的过程。

2. 以 Verilog 行为描述方式设计 16 位可逆二进制同步计数器

按要求创建 myRevCounter 工程，需要添加之前的文件 clkdiv.v,DispNum.v（不再重复展示），并新建 RevCounter.v,counter_100ms.v（只需要对 counter_1s.v 去一个 0 即可）及 top.v：

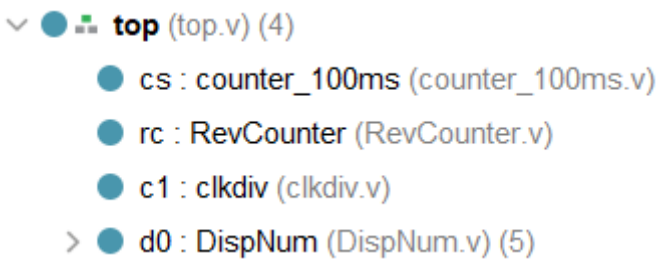
RevCounter.v 的代码如下：

```
module RevCounter(clk, s, cnt, Rc);
input wire clk, s;
output reg [15:0] cnt;
output wire Rc;
initial cnt = 0;
assign Rc = (~s & (~|cnt)) | (s & (&cnt));
always @ (posedge clk) begin
    if (s)
        cnt <= cnt + 1'b1;
    else
        cnt <= cnt - 1'b1;
end
endmodule
```

top.v 的代码如下：

```
`timescale 1ns / 1ps
module top(
    input wire clk,
    input wire s,
    output wire [7:0] LED,
    output wire [3:0] AN,
    output wire [7:0] SEGMENT
);
    wire clk_1s;
    wire [31:0] clk_div;
    wire [15:0] disp_hexs;
    counter_100ms cs(.clk(clk),.clk_1s(clk_1s));
    RevCounter rc(.clk(clk_1s),.s(s),.cnt(disp_hexs),.Rc(LED[0]));
    assign LED[7:1]=7'b0;
    clkdiv c1(.clk(clk), .rst(1'b0), .clk_div(clk_div));
    DispNum d0(.scan(clk_div[18:17]), .HEXS(disp_hexs),
     .LES(4'b0), .point(4'b0), .AN(AN), .SEGMENT(SEGMENT));
endmodule
```

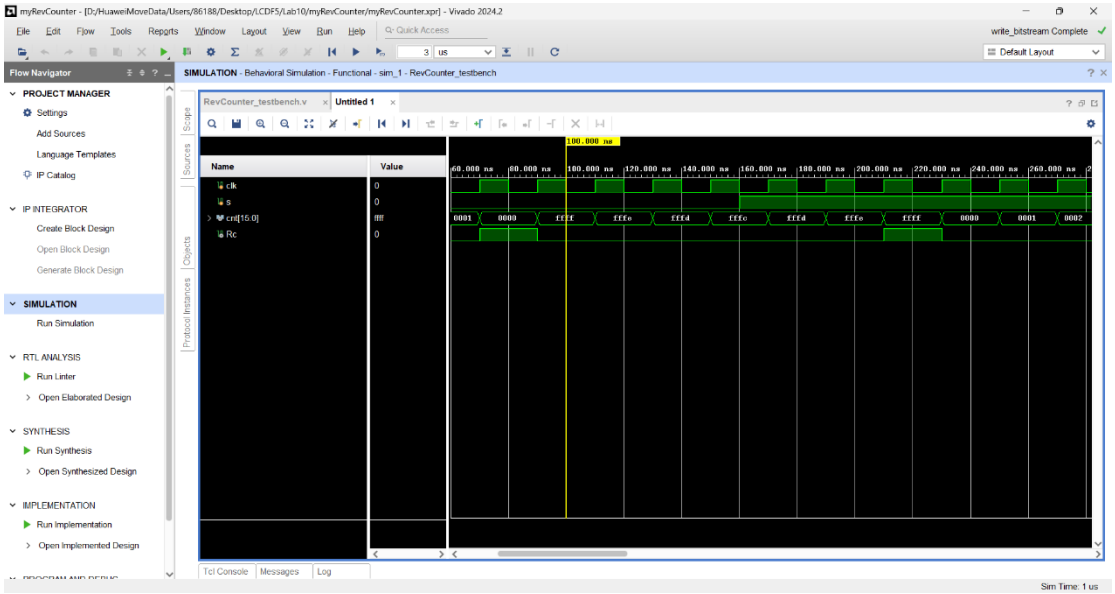
整个工程的文件结构如下：



对 **RevCounter.v** 进行仿真，激励代码如下：

```
`timescale 1ns / 1ps
module RevCounter_testbench();
    reg clk;
    reg s;
    wire [15:0] cnt;
    wire Rc;
    RevCounter uut(.clk(clk),.s(s),.cnt(cnt),.Rc(Rc));
    initial begin
        s=1; #40;
        s=0; #120;
        s=1;
    end
    always begin
        clk=0;#10;
        clk=1;#10;
    end
endmodule
```

其仿真结果如下：



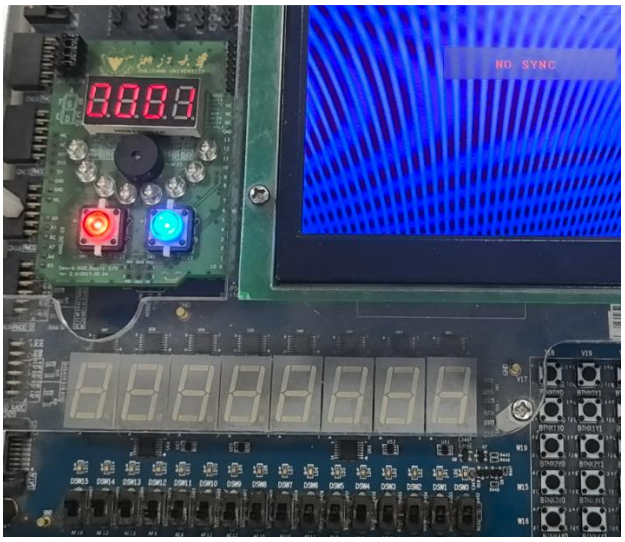
可以看到，在减法模式（s=0）下，当 cnt=0000 时，Rc 变为 1，表示借位；在加法模式（s=1）下，当 cnt=FFFF 时，Rc 变为 1，表示进位。

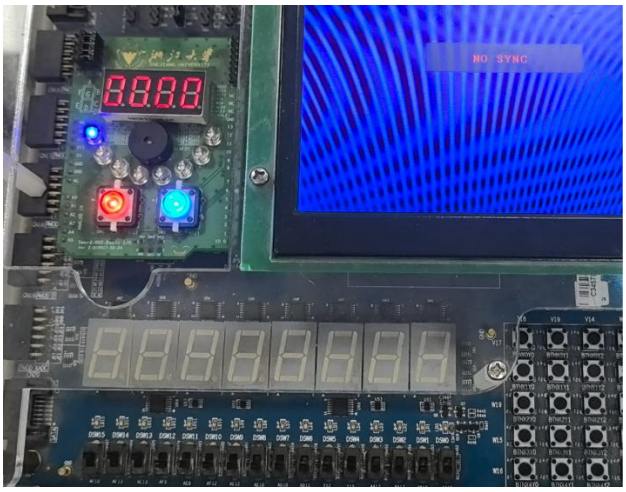
再将整个工程上板验证，其约束文件如下：

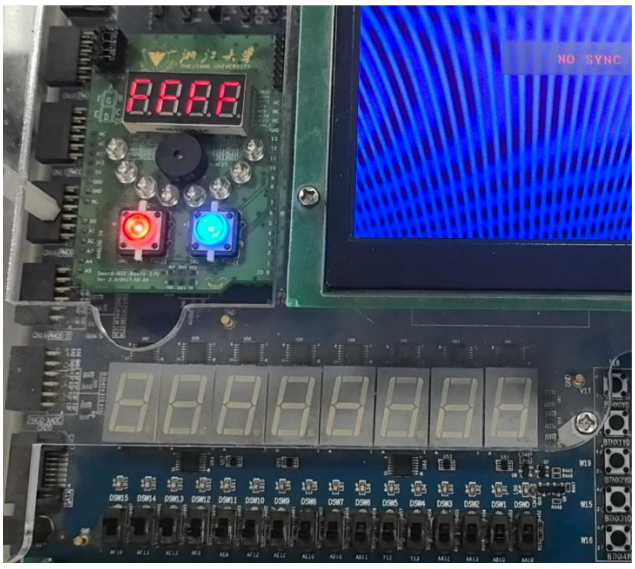
```
set_property PACKAGE_PIN AA10 [get_ports {s}]
set_property IOSTANDARD LVCMOS15 [get_ports {s}]
```

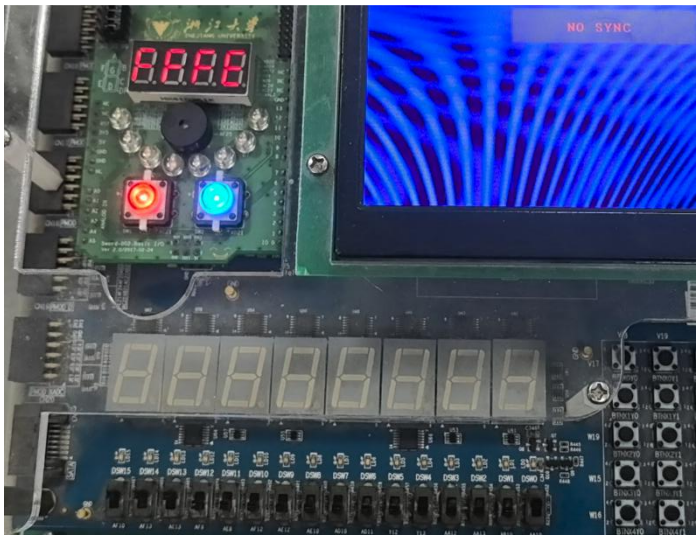
```
set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]
set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]
set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]
set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]
set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]
set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]
set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]
set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]
set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]
create_clock -name clk100MHZ -period 10.0 [get_ports clk]
set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]

set_property PACKAGE_PIN W23 [get_ports {LED[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[0]}]
set_property PACKAGE_PIN AB26 [get_ports {LED[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[1]}]
set_property PACKAGE_PIN Y25 [get_ports {LED[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[2]}]
set_property PACKAGE_PIN AA23 [get_ports {LED[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[3]}]
set_property PACKAGE_PIN Y23 [get_ports {LED[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[4]}]
set_property PACKAGE_PIN Y22 [get_ports {LED[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[5]}]
set_property PACKAGE_PIN AE21 [get_ports {LED[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[6]}]
set_property PACKAGE_PIN AF24 [get_ports {LED[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED[7]}]
```

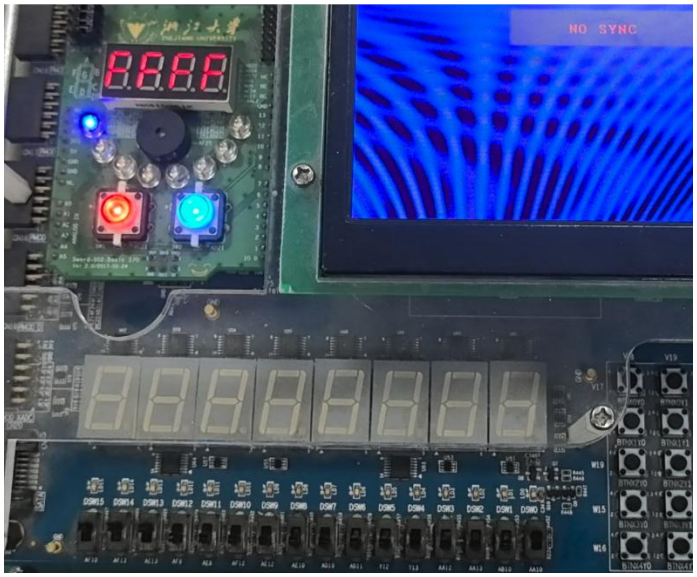
下载后可以发现调节 SW[0]=1，计数器的值在增加；调节 SW[0]=0，计数器的值在减小，以下是典型照片（0001->0000->FFFF 及 FFFE->FFFF->0000）：

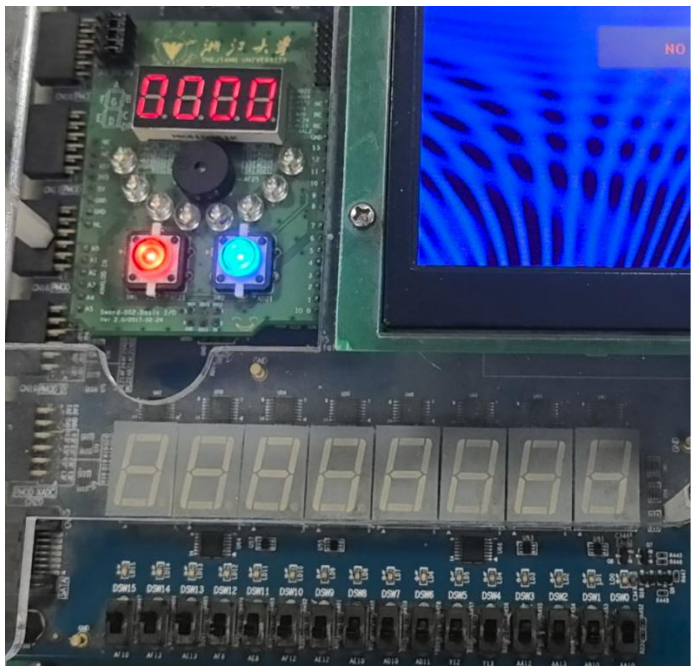












可以看到，当减法模式下出现 0000 时，LED 灯会亮起；当加法模式下出现 FFFF

时，LED 灯也会亮起，能够很好地实现 16 位可逆计数器的目的。

四、讨论、心得

这一次实验中，我进一步了解了时序逻辑电路，并成功使用时序逻辑电路实现 4 位计数器及 16 位可逆计数器，也学会了时序逻辑电路的一种运用。