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：	张晋恺
学　　院：	竺可桢学院
系：	所在系
专　　业：	计算机科学与技术
学　　号：	3230102400
指导教师：	董亚波

2024 年 5 月 13 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： 张晋恺 专业： 计算机科学与技术 学号： 3230102400

同组学生姓名： 杨吉祥 指导老师： 董亚波

实验地点： 东 4-511 实验日期： 2024 年 5 月 15 日

一、实验目的和要求

- 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、实验内容和原理

实验内容

- 原理图方式设计 4 位同步二进制计数器
- 以 Verilog 行为描述方式设计 16 位可逆二进制同步计数器

实验原理

4 位同步二进制计数器

计数是一种最简单基本的运算。计数器就是实现这种运算的逻辑电路，计数器在数字系统中主要是对脉冲的个数进行计数，以实现测量、计数和控制的功能，同时兼有分频功能，计数器是由基本的计数单元和一些控制门所组成，计数单元则由一系列具有存储信息功能的各类触发器构成，这些触发器有 RS 触发器、T 触发器、D 触发器及 JK 触发器等。计数器在数字系统中应用广泛，如在电子计算机的控制器中对指令地址进行计数，以便顺序取出下一条指令，在运算器中作乘法、除法运算时记下加法、减法次数，又如在数字仪器中对脉冲的计数等等。计数器可以用来显示产品的工作状态，一般来说主要是用来表示产品已经完成了多少份的折页配页工作。它主要的指标在于计数器的位数，常见的有 3 位和 4 位的。很显然，3 位数的计数器最大可以显示到 999，4 位数的最大可以显示到 9999。

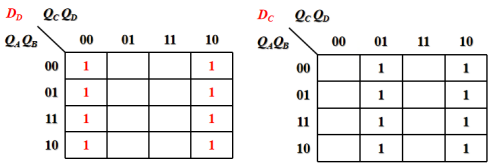
同步计数器指的是被测量累计值，其特点是大大提高了计数器工作频率，相对应的是异步计数器。对于同步计数器，由于时钟脉冲同时作用于各个触发器，克服了异步触发器所遇到的触发器逐级延迟问题，于是大大提高了计数器工作频率，各级触发器输出相差小，译码时能避免出现尖峰；但是如果同步计数器级数增加，就会使得计数脉冲的负载加重。

四位二进制同步计数器由四个触发器组成，各位触发器的时钟脉冲输入端 C 接同一个计数脉冲 Clk。设初状态从 0000 开始，每输入一个计数脉冲 Clk，最低触发端就会翻转一次，而其它触发器仅在 C = 1 的条件下，在 Clk 前沿翻转一次。一个四位二进制同步计数器的真值表如图所示：

	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

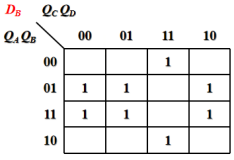
图 1: 四位二进制同步计数器的真值表

根据真值表，我们可以分别画出四位二进制同步计数器的 $D_{A\ B}$ 与 $Q_{A\ B}$ 的卡诺图以及逻辑表达式。

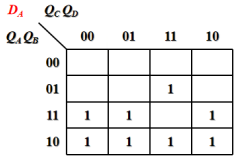


(a) D_D

(b) D_D



(c) D_B



(d) D_A

图 2: 四位二进制同步计数器的 $D_{A\ B}$ 与 $Q_{A\ B}$ 的卡诺图

由卡诺图我们可以得到 $D_{A\ B}$ 与 $Q_{A\ B}$ 的逻辑表达式：

$$
\begin{aligned}
D_D &= \overline{Q_D} \\
D_C &= \overline{Q_C}Q_D + Q_C\overline{Q_D} = Q_D \oplus Q_C \\
D_B &= Q_B\overline{Q_C} + Q_B\overline{Q_D} + \overline{Q_B}Q_CQ_D = Q_B \oplus Q_CQ_D \\
D_A &= Q_BQ_CQ_D \oplus Q_A
\end{aligned}
$$

同时，进位输出 R_C 的逻辑表达式为：$R_C = Q_AQ_BQ_CQ_D$

可逆二进制同步计数器

可逆二进制同步计数器通过控制端 S 选择正向或者反向计数

- S=1 时，正向计数
- S=0 时，反向计数。

仿照普通计数器的方法，我们也可以得到各逻辑函数的表达式

$$
\begin{aligned}
D_A &= \overline{Q_A} \\
D_B &= \overline{S}(\overline{Q_A} \oplus \overline{Q_B}) + S(\overline{\overline{Q_A} \oplus \overline{Q_B}}) = \overline{S \oplus \overline{Q_A} \oplus \overline{Q_B}} \\
D_C &= \overline{S}[\overline{(Q_AQ_B)} \oplus \overline{Q_C}] + S[\overline{(\overline{Q_A} + \overline{Q_B})} \oplus \overline{Q_C}] = \overline{[S\overline{Q_AQ_B} + S(\overline{Q_A} + \overline{Q_B})] \oplus \overline{Q_C}} \\
&= \overline{[\overline{S}(Q_A + Q_B) + S(\overline{Q_A} + \overline{Q_B})] \oplus \overline{Q_C}} \\
D_D &= \overline{S}[\overline{(Q_AQ_BQ_C)} \oplus \overline{Q_D}] + S[\overline{(\overline{Q_A} + \overline{Q_B} + \overline{Q_C})} \oplus \overline{Q_D}] = \overline{[S\overline{Q_AQ_BQ_C} + S(\overline{Q_A} + \overline{Q_B} + \overline{Q_C})] \oplus \overline{Q_D}} \\
&= \overline{[\overline{S}(Q_A + Q_B + Q_C) + S(\overline{Q_A} + \overline{Q_B} + \overline{Q_C})] \oplus \overline{Q_D}} \\
R &= \overline{SQ_AQ_BQ_CQ_D} + SQ_AQ_BQ_CQ_D \quad \text{（进位、借位输出）}
\end{aligned}
$$

可逆二进制 4 位同步计数器的行为描述

```
module RevCounter_4bit(clk, s, cnt, Rc);
input wire clk, s;
output reg [3:0] 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
```

```
13      endmodule
```

秒计数器设计

100MHz 信号通过 50,000,000 次分频后，得到 1Hz 的秒脉冲方波，作为计数器的脉冲输入

```
1       module counter_1s(clk, clk_1s);
2       input wire clk;
3       output reg clk_1s;
4       reg [31:0] cnt;
5       initial clk_1s = 0;
6       always @ (posedge clk) begin
7          if (cnt < 50_000_000) begin
8              cnt <= cnt + 1' b1;
9          end else begin
10             cnt <= 0;
11             clk_1s <= ~clk_1s;
12         end
13      end
14      endmodule
```

三、实验步骤和结果记录

原理图方式设计 4 位同步二进制计数器

在 Digital 中绘制原理图

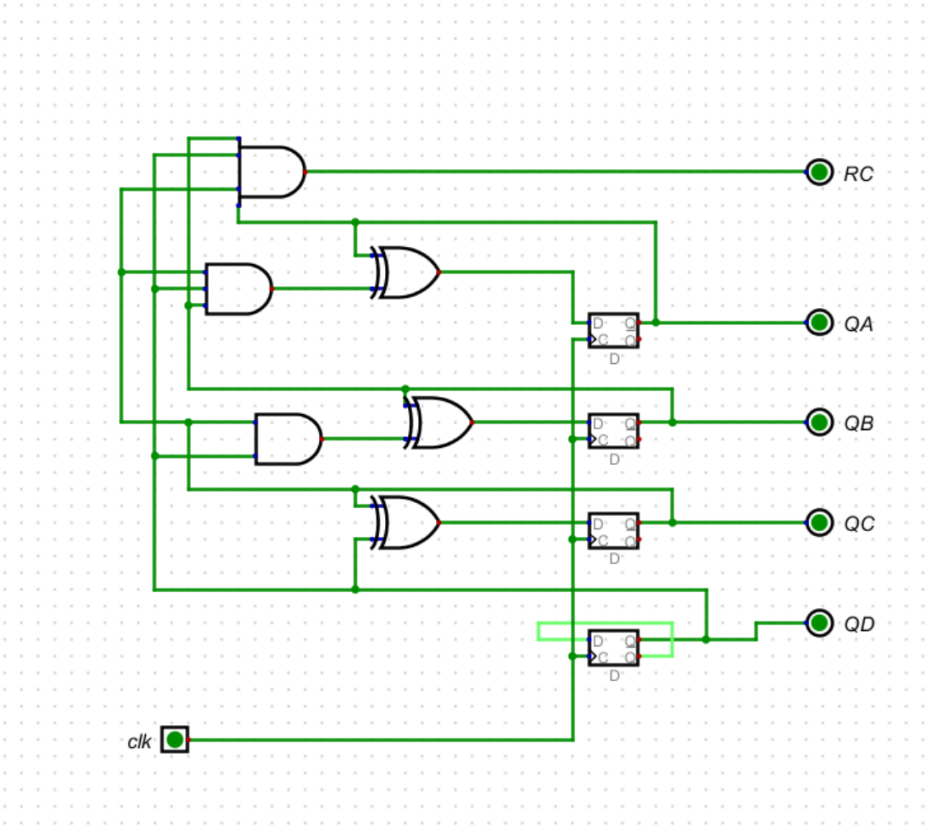


图 3: 原理图方式设计 4 位同步二进制计数器

在 Digital 中仿真测试如图所示：

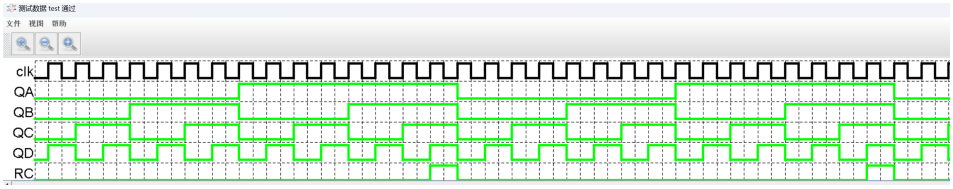


图 4: Digital 仿真测试

波形解释：从波形图中可以看出，当 clk 处于上升沿触发时，计数器的输出 Q_D 会加 1，当 Q_D 受到两次触发时，Q_C 会加 1，Q_C 受到两次触发时，Q_B 会加 1，以此类推，受到 8 次触发时 Q_A 会加 1。从波形图也可以看出，位于上方的波形图的 1 的长度为位于下方的波形图的 1 的长度的两倍。

导出 Verilog 代码, 在 Vivado 中验证

在 Vivado 中新建MyCounter工程，设计 Counter4b 模块仿真代码如下：

```
module Counter4b_tb();
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)
 );
 initial  begin
     clk=0;
     forever begin
         #20 clk=~clk;
     end
 end


endmodule
```

仿真波形如下

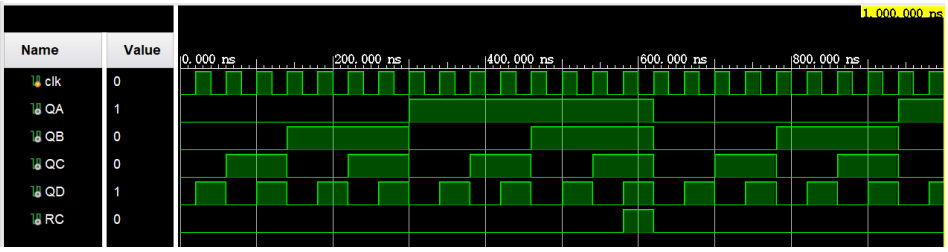


图 5: Vivado 仿真测试

如图，波形基本与 Digital 仿真测试一致，说明设计正确。同时在 Q_{A-D} 均为 1 时，RC 进位为 1.

Top 模块建立以及仿真

Top 模块需要的部分有

- 4 位计数器 Counter4b
- 时钟分频模块 counter_1s(代码见实验原理部分)
- 显示结果的模块 DispNum(在 lab7 中已经设计)，直接导入即可
- 进位显示在 LED[0] 灯上

Top 模块代码，以及需要注意的事项（注释）如下：

```
1   module Top(
2       input wire clk,
3       output wire [7:0] SEGMENT,
4       output wire [3:0] AN,
5       output wire [7:0] LED,
6       output wire BTNX4
7       );
8       wire clk_1s;
9       wire [31:0] clk_div;
10      wire [3:0] num;
11      counter_1s uut (
12          .clk(clk),
13          .clk_1s(clk_1s)
14      );//仿真时修改 counter_1s.v 中分频数为10或者更低，便于观察实验结果
```

```
        clkdiv c1(.clk(clk), .rst(1'b0), .clk_div(clk_div));
        Counter4b uut1 (
            .clk(clk_1s),
            .QA(num[3]),
            .QB(num[2]),
            .QC(num[1]),
            .QD(num[0]),
            .RC(LED[0])
        );
        DispNum uut2 (
            .LES(4'b0),
            .point(4'b0),
            .HEXS({4'b0000,4'b0000,4'b0000,num[3:0]}),//最低位显示结果，其余位显示为0
            .scan(clk_div[18:17]),//仿真时修改为clk_div[3:2]
            .AN(AN),
            .SEGMENT(SEGMENT)
        );
        assign BTNX4 = 1'b0;
    endmodule
```

行为仿真

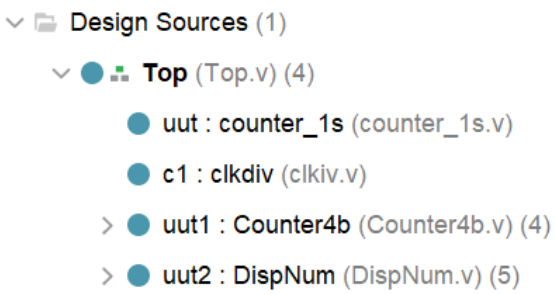


图 6: 工程层次结构

仿真代码如下

```
    module Top_tb(

```

```
    );
    reg clk;
    wire [7:0] SEGMENT;
    wire [3:0] AN;
    wire [7:0] LED;
    wire BTNX4;
    Top uut(
        .clk(clk),
        .SEGMENT(SEGMENT),
        .AN(AN),
        .LED(LED),
        .BTNX4(BTNX4)
    );
    initial begin
        clk=0;
        forever begin
            #10 clk=~clk;
        end
      end
endmodule
```

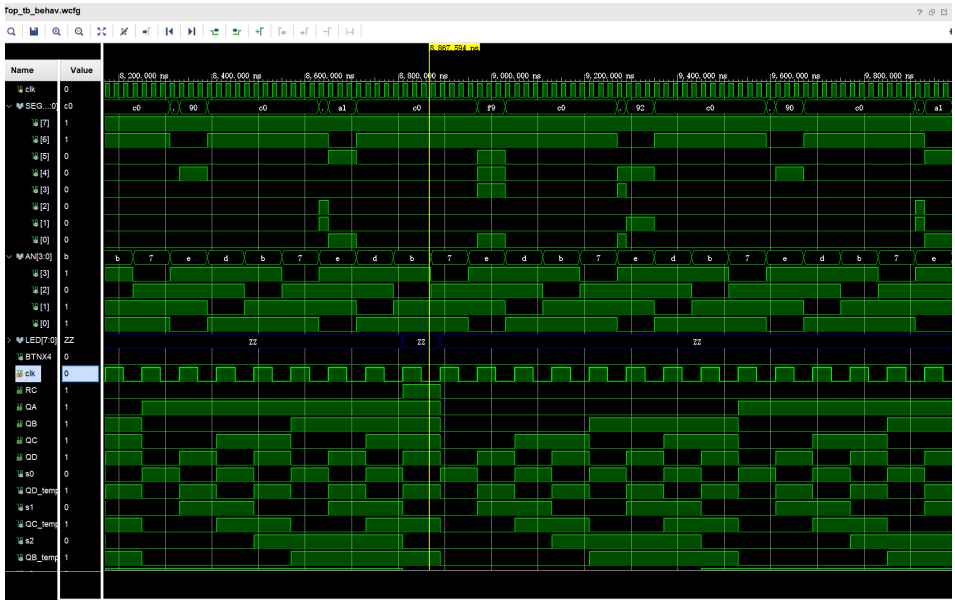
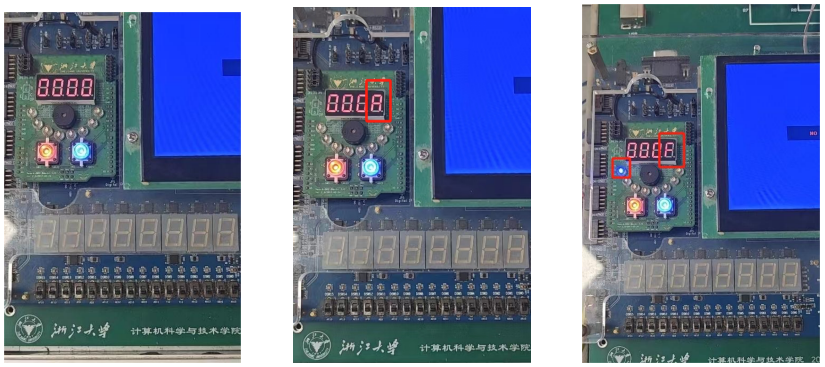


图 7: Top 模块仿真

波形解释：如图所示，当 AN[0]=0 时，与之前的实验一样，代表最低位的数码管显示数值，其数值与 QA,QB,QC,QD(所代表的 16 进制数) 的值相同。当 RC=1 时，代表进位灯 LED[0] 亮。需要注意的是，位于波形上方 clk 是仿真代码中定义的 clk，位于波形下方的 clk 是 Top 模块中实例化 Counter4b 的输入 clk，其值为 clk_1s 的输出 (仿真时 counter_1s 已经经过修改)。观察波形图可知，Top 模块设计正确，各部分功能实现无误。

在实验板上验证

编辑约束文件，将 Top 模块的输入输出与实验板上的引脚相对应，下载到实验板上，观察数码管显示结果。



(a) 初始状态　(b) 自增　(c) 进位

图 8: 实验板验证

上板结果：上板时，按下 Program 选项后，数码管初始显示为 0000，然后数码管会自增（每 1s），当数码管显示为 000F 时，进位灯 LED[0] 会亮，数码管会重新显示为 0000，如此循环。说明设计正确。

用行为描述语言设计 16 位可逆二进制计数器

- 新建工程 myRevCounter
- 在工程中新建 RevCounter 模块，代码如下

```
1        module RevCounter(clk,s,cnt,Rc);
2        input wire clk,s;
3        output reg [15:0]cnt;
4        output wire Rc;
5        initial cnt=0;
```

```
        initial begin
          cnt=16'hffff;
        end
        assign Rc=(~s&(~|cnt))|(s & (&cnt));
          always @(posedge clk) begin
              if (s)
                  cnt<=cnt+1'b1;
              else
                  cnt<=cnt-1'b1;
          end
      endmodule
```

其设计思路与四位可逆二进制计数器相同，只是位数更多，需要注意的是，将 cnt 初始化为 16'hffff，只是为了仿真和上板的方便，并没有实际作用。

对设计的 RevCounter 模块进行仿真测试

仿真代码以及解释如下

```
module Rev_counter_tb(
    );
    reg clk;
    reg s;
    wire [15:0] cnt;
    wire Rc;
    RevCounter UUT(
        .clk(clk),
        .s(s),
        .cnt(cnt),
        .Rc(Rc)
    );
    initial begin
        clk=0;
        forever begin
           #5; clk=~clk;
        end
    end
```

```
19          initial begin
20              s=1;#160;//先自增观察进位
21              s=0;#160;//再自减观察借位
22              s=1;//自增
23          end
24      endmodule
```

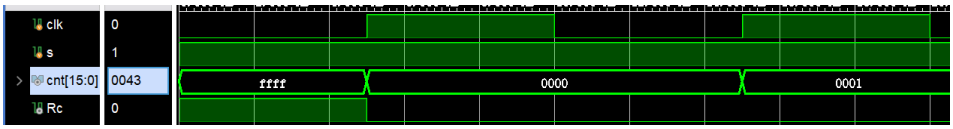


图 9: RevCounter 模块仿真进位

波形解释：如图所示，当 s=1 时，cnt 会自增，当 cnt=16'hffff 时，进位 Rc 会变为 1，继续自增，cnt 和 Rc 会重新变为 0。

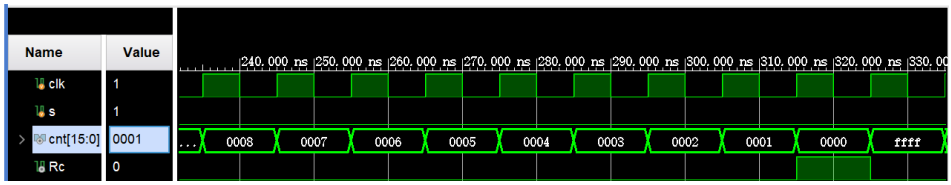


图 10: RevCounter 模块仿真借位

波形解释：如图所示，当 s=0 时，cnt 会自减，当 cnt=0 时，借位 Rc 会变为 1，继续自减，cnt 和 Rc 会重新变为 16'hffff。

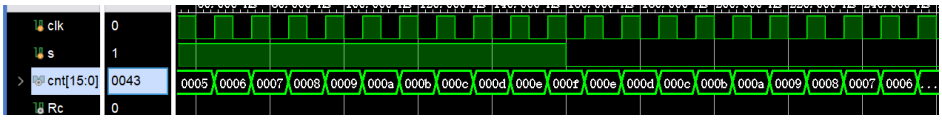


图 11: RevCounter 模块仿真

波形解释：如图所示，一般情况下 s=1 时，cnt 会自增，s=0 时，cnt 会自减。仿真正确

Top 模块设计

Top 模块需要的部分与之前的设计类似，只是需要将 RevCounter 模块实例化，并且需要将 16 位的 cnt 分成 4 位一组，分别显示在数码管上。需要注意的是，这一部分要求我们设计 100ms 的分频模块，以便于观察实验结果。同时要用 SW[0] 来控制 RevCounter 模块的计数方向。

- 新建源文件 counter_100ms.v，代码如下

```
module clk_100ms(clk, clk_100ms);
input wire clk;
output reg clk_100ms;
reg [31:0] cnt;
initial clk_100ms = 0;
always @ (posedge clk) begin
   if (cnt < 50_000_00) begin
       cnt <= cnt + 1'b1;
   end else begin
       cnt <= 0;
       clk_100ms <= ~clk_100ms;
   end
end
endmodule
```

- 在设计 1s 时钟时，我们把 $1 \times 10^8 Hz$ 分频 5×10^7 次，得到了 1Hz 的时钟，类似的，这里我们把 $1 \times 10^8 Hz$ 分频 5×10^6 次，就得到了 10Hz（100ms）的时钟。

Top 模块代码如下

```
module Top(
    input wire [3:0] SW,
    input wire clk,
    output wire [7:0] SEGMENT,
    output wire [3:0] AN,
    output wire [7:0] LED,
    );
    wire clk_100ms;
    wire [31:0] clk_div;
    wire [15:0] num;
    clk_100ms uut (
        .clk(clk),
        .clk_100ms(clk_100ms)
    );
    clkdiv c1(.clk(clk), .rst(1'b0), .clk_div(clk_div));
```

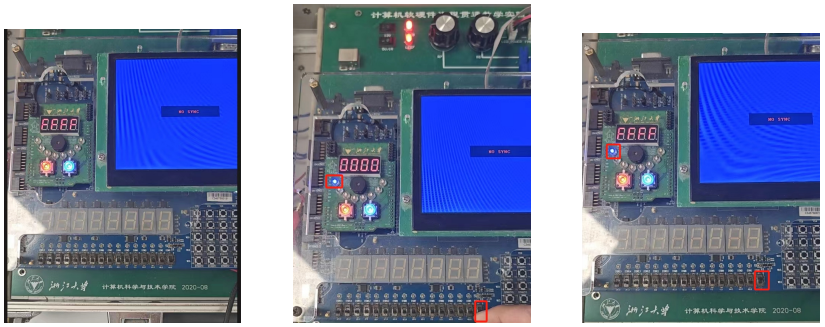
```
16          RevCounter count16
17          (
18              .clk(clk_100ms),
19              .s(SW[0]),
20              .cnt(num),
21              .Rc(LED[0])
22          );
23          DispNum uut2 (
24              .LES(4'b0),
25              .point(4'b0),
26              .HEXS(num),
27              .scan(clk_div[18:17]),
28              .AN(AN),
29              .SEGMENT(SEGMENT)
30          );
31      endmodule
```

下载运行

编辑约束文件，将 Top 模块的输入输出与实验板上的引脚相对应，下载到实验板上，观察数码管以及显示结果。

当点击 Program 后，数码管会显示为 16'hffff，然后初始状态 SW[0]=0, 数码管每隔 0.1s 会自减，当数码管显示为 0 时，进位灯 LED[0] 会亮，数码管会重新显示为 16'hffff，如此循环。当 SW[0]=1 时，数码管每隔 0.1s 会自增，当数码管显示为 16'hffff 时，进位灯 LED[0] 会亮，数码管会重新显示为 0，如此循环。

进位借位显示的实验板现象如下



(a) 初始状态　(b) 自减借位　(c) 自增进位

图 12: 实验板验证

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。仿真结果和 Verilog 代码分析在前文已经给出。

五、讨论与心得

本次实验在预习的时候花了我不少功夫，在写 Counter4b 的 Top 文件的时候，由于传递给 DispNum 动态扫描显示的 clk_div 的变量名的混乱，导致我加入了 Top 文件之后一直会报错，后面在老师的帮助下，我学会了在 Top 模块中依次实例化模块来排查错误，以及 **学会看报错的** log **文件也很有效**. 同时，本次实验需要用到以前设计的不少模块，这也让我体会到了模块化设计的好处。可以随时拿来用而不需要重新设计。

在第二部分用 Verilog 语言设计 16 位可逆二进制计数器的时候，一开始，我想的是把四个四位二进制计数器连在一起，但是如何处理进位的问题一直困扰着我，后来在研究了 PPT 上的提示之后，我发现可以简单的把寄存器改成 16 位的就可以了，这样就不需要考虑进位的问题了。这个实验让我体会到了设计的巧妙之处。

总的来说，本次实验较为顺利，让我对计数器的设计有了更深的理解，对时序电路的强大之处也有了很深的体会。同时也让我对 Verilog 语言的使用更加熟练。