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：薛柔

学　　院：竺可桢学院

专　　业：计算机科学与技术

学　　号：3220104854

指导教师：董亚波

2023 年 12 月 14 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：实验 11、同步时序电路设计

学生姓名：薛柔 专业：计算机科学与技术 学号：3220104854

同组学生姓名：江佩遥 指导老师：董亚波

实验地点：东 4-509 实验日期：2023 年 11 月 30 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

内容：

- 任务 1：原理图方式设计 4 位同步二进制计数器
- 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

原理：

- 4 选 4 位二进制同步计数器

  根据 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：

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

图 2.1 4 位计数器的真值表

通过卡诺图化简，激励函数为：

$$D_A = \overline{Q_A}$$

$$D_B = \overline{Q_A}Q_B + Q_A\overline{Q_B} = \overline{\overline{Q_A} \oplus \overline{Q_B}}$$

$$D_C = \overline{Q_A}Q_C + \overline{Q_B}Q_C + Q_AQ_B\overline{Q_C}$$

$$= \overline{\overline{(\overline{Q_A} + \overline{Q_B})} \oplus \overline{Q_C}}$$

$$D_D = \overline{Q_A}Q_D + \overline{Q_B}Q_D + \overline{Q_C}Q_D + Q_AQ_BQ_C\overline{Q_D}$$

$$= \overline{\overline{(\overline{Q_A} + \overline{Q_B} + \overline{Q_C})} \oplus \overline{Q_D}}$$

进位 Rc 的输出函数为：

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

这样便能通过上述逻辑函数画出原理图。

- 4 位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。各触发器逻辑表达式如下：

$$D_A = \overline{Q_A}$$

$$D_B = \overline{S}(\overline{Q_A} \oplus \overline{Q_B}) + S(\overline{\overline{Q_A} \oplus \overline{Q_B}}) = \overline{S \oplus \overline{Q_A} \oplus \overline{Q_B}}$$

$$D_C = \overline{S}[\overline{(\overline{Q_A}\,\overline{Q_B}) \oplus \overline{Q_C}}] + S[\overline{(\overline{Q_A} + \overline{Q_B}) \oplus \overline{Q_C}}] = \overline{[\overline{S}\,\overline{\overline{Q_A}\,\overline{Q_B}} + S(\overline{Q_A} + \overline{Q_B})] \oplus \overline{Q_C}}$$

$$= \overline{[\overline{S}(Q_A + Q_B) + S(\overline{Q_A} + \overline{Q_B})] \oplus \overline{Q_C}}$$

$$D_D = \overline{S}[\overline{(\overline{Q_A}\,\overline{Q_B}\,\overline{Q_C}) \oplus \overline{Q_D}}] + S[\overline{(\overline{Q_A} + \overline{Q_B} + \overline{Q_C}) \oplus \overline{Q_D}}] = \overline{[\overline{S}\,\overline{\overline{Q_A}\,\overline{Q_B}\,\overline{Q_C}} + S(\overline{Q_A} + \overline{Q_B} + \overline{Q_C})] \oplus \overline{Q_D}}$$

$$= \overline{[\overline{S}(Q_A + Q_B + Q_C) + S(\overline{Q_A} + \overline{Q_B} + \overline{Q_C})] \oplus \overline{Q_D}}$$

$$R = \overline{S}\,\overline{Q_A}\,\overline{Q_B}\,\overline{Q_C}\,\overline{Q_D} + S Q_A Q_B Q_C Q_D$$ （进位、借位输出）

- 分频器

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。

三、实验过程和数据记录

1. 原理图方式设计 4 位同步二进制计数器

a. 新建工程，工程名称用 MyCounter，Top Level Source Type 用 HDL。并新建 Schematic 源文件，文件名称用 Counter4b，原理图方式进行设计。下图是绘制好的原理图：

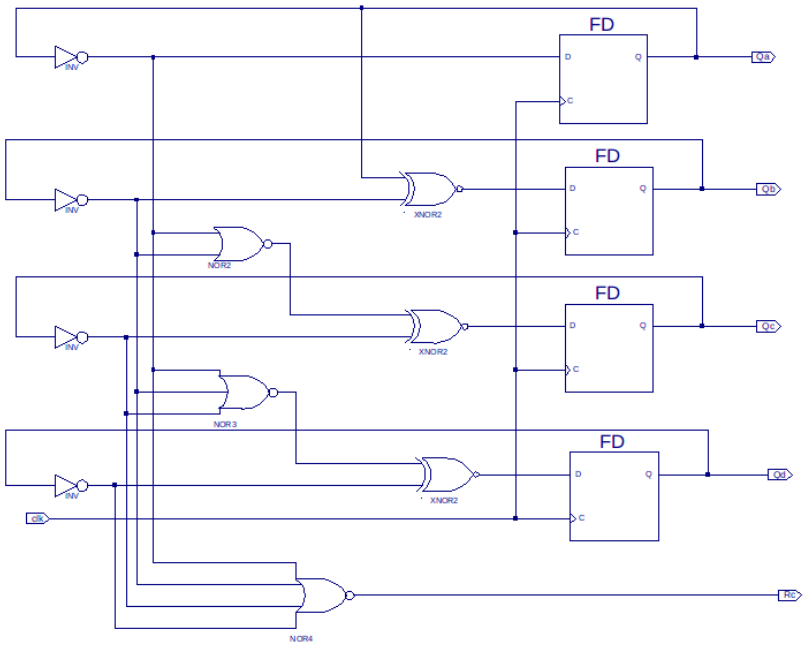


图 3.1 Counter4b 原理图

b. 建立仿真波形文件，对该模块进行仿真，激励代码如下：

```
17  // Bidirs
18
19  // Instantiate the UUT
20     Counter4b UUT (
21         .Qb(Qb),
22         .Qc(Qc),
23         .Qd(Qd),
24         .Qa(Qa),
25         .clk(clk),
26         .Rc(Rc)
27     );
28  // Initialize Inputs
29     //`ifdef auto_init
30         always begin
31         clk = 0;#10;
32         clk = 1;#10;
33         end
34     //`endif
35  endmodule
36
```

图 3.2 Counter4b 模块仿真激励代码

上述代码生成的仿真波形如下：

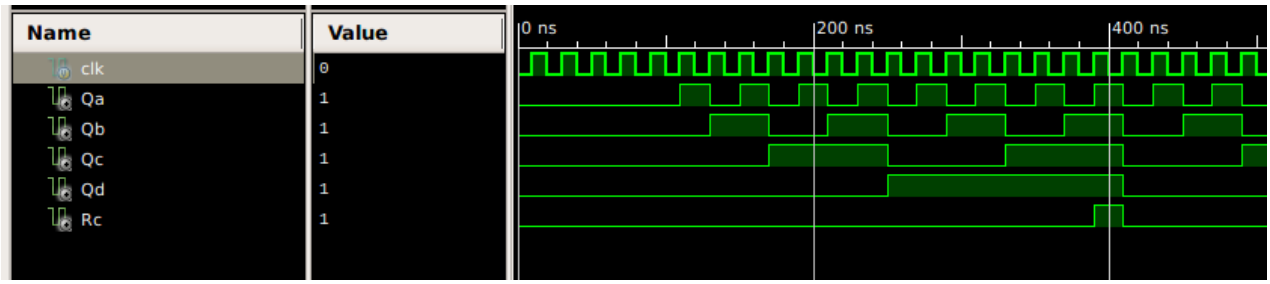


图 3.3 得到的仿真信号

c. 新建源文件，用作时钟，类型是 Verilog，文件名称用 clk_1s。用分频器原理，100MHz 信号通过 50,000,000 次分频后，得到 1Hz 的秒脉冲方波，设计 1s 的时钟。Verilog 代码如下：

```
22  module clk_1s(clk, clk_1s);
23  input wire clk;
24  output reg clk_1s;
25  reg [31:0] cnt;
26  always @ (posedge clk) begin
27     if (cnt < 50_000_000) begin
28         cnt <= cnt + 1'b1;
29     end else begin
30         cnt <= 0;
31         clk_1s <= ~clk_1s;
32     end
33  end
34  endmodule
```

图 3.4 clk_1s 模块的 Verilog 代码

d. 新建 Top.v，根据下列要求完成顶层模块设计：

- 输入为 clk（100MHZ）时钟
- 每秒自增 1
- 显示在 1 位数码管上
- Rc 显示在 LED 灯上

顶层模块的 Verilog 代码如下：

```
21  module Top(input wire clk,
22             output wire [7:0]SEGMENT,
23             output wire [3:0]AN,
24             output wire led
25      );
26      wire clk_1s;
27      wire [7:0]num;
28      assign num[7:4]=4'b0001;//让一个led官亮起
29      clk_1s m0(.clk(clk),.clk_1s(clk_1s));//时钟分频
30      Counter4b m1(.clk(clk_1s),.Qa(num[0]),
31      .Qb(num[1]),.Qc(num[2]),.Qd(num[3]),.Rc(led));
32
33      DispNumber_sch d0(.SW(num), .BTN(2'b00),
34      .SEGMENT(SEGMENT),.AN(AN));//当前数字显示在数码管上
35  endmodule
```

图 3.5 顶层模块的 Verilog 代码

实验板验证：建立用户时序约束并为模块的端口指定引脚分配。建立 K7.ucf 文件，输入如图 3.6 所示代码。

```
1  NET "SEGMENT[0]" LOC = AB22 | IOSTANDARD = LVCMOS33;#a
2  NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;#b
3  NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;#c
4  NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;#d
5  NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;#e
6  NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;#f
7  NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;#g
8  NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;#point
9  NET "led" LOC = W23 | IOSTANDARD = LVCMOS33;
10 NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;
11 NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;
12 NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;
13 NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;
14 NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18;
```

图 3.6 K7.ucf 文件

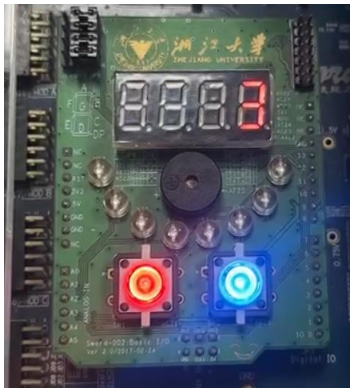
生成 bit 文件，实现设计，并在 Summury 中查看约束结果如图 3.7 所示，可以看出所有输入输出都已按 ucf 文件正确分配。

	Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number
1	AD21	AN<0>	IOB33	IO_L19P_T3_12	OUTPUT	LVCMOS33	12
2	AC21	AN<1>	IOB33	IO_L18N_T2_12	OUTPUT	LVCMOS33	12
3	AB21	AN<2>	IOB33	IO_L18P_T2_12	OUTPUT	LVCMOS33	12
4	AC22	AN<3>	IOB33	IO_L17N_T2_12	OUTPUT	LVCMOS33	12
5	AB22	SEGMENT<0>	IOB33	IO_L17P_T2_12	OUTPUT	LVCMOS33	12
6	AD24	SEGMENT<1>	IOB33	IO_L16N_T2_12	OUTPUT	LVCMOS33	12
7	AD23	SEGMENT<2>	IOB33	IO_L16P_T2_12	OUTPUT	LVCMOS33	12 12
8	Y21	SEGMENT<3>	IOB33	IO_L15N_T2_DQS_12	OUTPUT	LVCMOS33	12
9	W20	SEGMENT<4>	IOB33	IO_L15P_T2_DQS_12	OUTPUT	LVCMOS33	12
10	AC24	SEGMENT<5>	IOB33	IO_L14N_T2_SRCC_12	OUTPUT	LVCMOS33	12
11	AC23	SEGMENT<6>	IOB33	IO_L14P_T2_SRCC_12	OUTPUT	LVCMOS33	12
12	AA22	SEGMENT<7>	IOB33	IO_L13N_T2_MRCC_12	OUTPUT	LVCMOS33	12
13	AC18	clk	IOB	IO_L13P_T2_MRCC_32	INPUT	LVCMOS18*	32
14	W23	led	IOB33	IO_L8P_T1_12	OUTPUT	LVCMOS33	12
15	A1			GND			

图 3.7 约束结果图

将 bit 文件复制到连接实验箱的电脑上，对硬件设备进行下载编程，验证输出是否满足设计要求。下面一组图是数码管依次从 0 计数到 F 并进位的过程，可以看到进位的同时（显示 F 时），左数第一个 Led 灯亮起，说明满足设计要求。



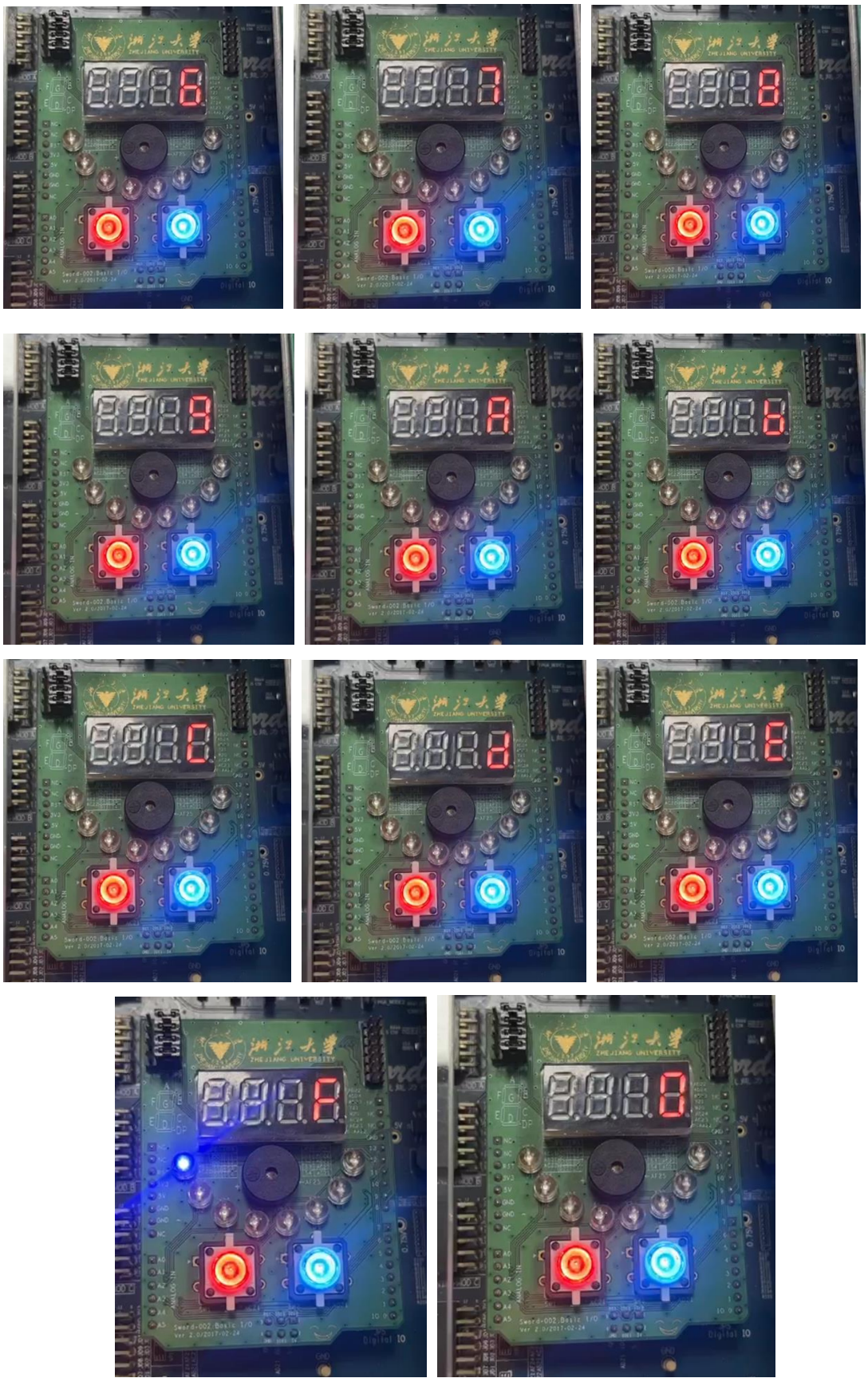


图 3.8 验证结果

2. 以 Verilog 行为描述方式设计 16 位可逆二进制同步计数器

a. 新建工程，工程名称用 myRevCounter。Top Level Source Type 用 HDL。采用行为描述方式进行设计 16 位可逆同步二进制计数器，文件名称用 RevCounter 。Verilog 代码如下：

```
21  module RevCounter(clk, s, cnt, Rc);
22     input wire clk, s;
23     output reg [15:0] cnt;
24     output wire Rc;
25     initial cnt = 0;
26     assign Rc = (~s & (~|cnt)) | (s & (&cnt));
27     always @ (posedge clk) begin
28         if (s)
29             cnt <= cnt + 1'b1;
30         else
31             cnt <= cnt - 1'b1;
32     end
33  endmodule
```

图 3.9 16 位可逆同步二进制计数器的行为描述代码

b. 建立仿真波形文件，对该模块进行仿真，激励代码如下：

```
43     initial begin
44        // Initialize Inputs
45        clk = 0;
46        s = 0;
47
48        // Wait 100 ns for global reset to finish
49        #200;
50        s=1;
51        #300;
52        s=0;
53        // Add stimulus here
54
55     end
56     always begin
57        clk=0;#20;
58        clk=1;#20;
59     end
60  endmodule
```

图 3.10 RevCounter 模块仿真激励代码

上述代码生成的仿真波形如下：

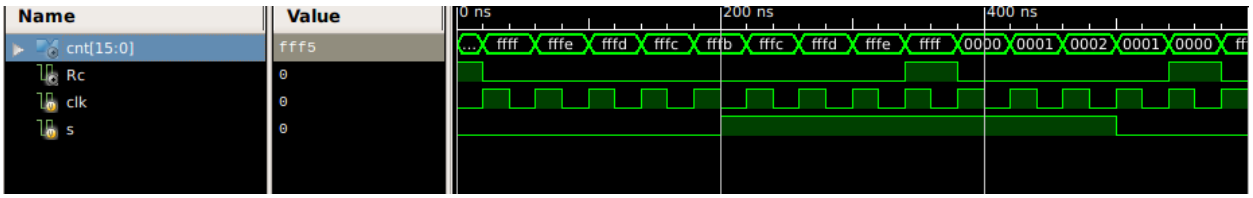


图 3.11 得到的仿真信号

c. 新建源文件，采用 Verilog 行为描述设计 100ms 时钟，类型是 Verilog，文件名称用 clk_100ms。这里即将 1s 时钟的分频次数减少 10 倍，即可得到 100ms 时钟。Verilog 代码如下：

```
22  module clk_100ms(clk, clk_100ms);
23  input wire clk;
24  output reg clk_100ms;
25  reg [31:0] cnt;
26  always @ (posedge clk) begin
27      if (cnt < 5_000_000) begin
28          cnt <= cnt + 1'b1;
29      end else begin
30          cnt <= 0;
31          clk_100ms <= ~clk_100ms;
32      end
33  end
34  endmodule
```

图 3.12 clk_100ms.v

d. 新建 Top.v，根据下列要求完成顶层模块设计：

- Top 模块的输入为 clk（100MHZ）时钟
- RevCounter 模块的时钟输入为 100ms 时钟
- 用 sw[0]控制自增/自减 1（每 0.1 秒）
- 计数结果显示在 4 位数码管上
- Rc 状态用 LED 灯来显示

顶层模块的 Verilog 代码如下：

```
21  module Top(input wire clk,
22             input wire sw,
23             output wire [7:0]SEGMENT,
24             output wire [3:0]AN,
25             output wire led//Rc显示仕led上
26      );
27      wire clk_100ms;
28      wire [15:0]num;
29      clk_100ms m0(.clk(clk),.clk_100ms(clk_100ms));
30      RevCounter r0(.clk(clk_100ms),.s(sw),.cnt(num),.Rc(led));
31      DispNum d0(.clk(clk), .HEXS(num), .LES(4'b0000),
32      .point(4'b0000), .RST(1'b0), .AN(AN), .Segment(SEGMENT));
33  endmodule
```

图 3.13 顶层模块的 Verilog 代码

实验板验证：建立用户时序约束并为模块的端口指定引脚分配。建立 K7.ucf 文件，输入如图 3.14 所示代码。

```
1  NET "SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;#a
2  NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;#b
3  NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;#c
4  NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;#d
5  NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;#e
6  NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;#f
7  NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;#g
8  NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;#point
9  NET "led" LOC = W23 | IOSTANDARD = LVCMOS33;
10 NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;
11 NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;
12 NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;
13 NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;
14 NET "sw" LOC = AA10 | IOSTANDARD = LVCMOS15;
15 NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18;
```

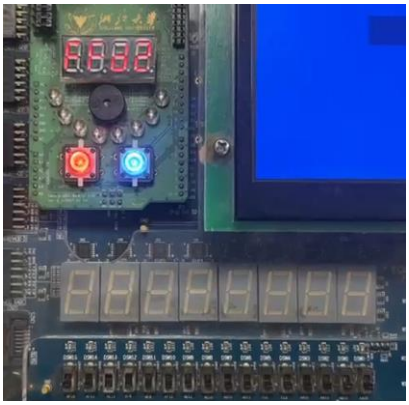
图 3.14 K7.ucf 文件

生成 bit 文件，实现设计，并在 Summury 中查看约束结果如图 3.7 所示，可以看出所有输入输出都已按 ucf 文件正确分配。

	Pin Number	Signal Name	Pin Usage	Pin Name	Direction	IO Standard	IO Bank Number
1	AD21	AN<0>	IOB33	IO_L19P_T3_12	OUTPUT	LVCMOS33	12
2	AC21	AN<1>	IOB33	IO_L18N_T2_12	OUTPUT	LVCMOS33	12
3	AB21	AN<2>	IOB33	IO_L18P_T2_12	OUTPUT	LVCMOS33	12
4	AC22	AN<3>	IOB33	IO_L17N_T2_12	OUTPUT	LVCMOS33	12
5	AB22	SEGMENT<0>	IOB33	IO_L17P_T2_12	OUTPUT	LVCMOS33	12
6	AD24	SEGMENT<1>	IOB33	IO_L16N_T2_12	OUTPUT	LVCMOS33	12
7	AD23	SEGMENT<2>	IOB33	IO_L16P_T2_12	OUTPUT	LVCMOS33	12
8	Y21	SEGMENT<3>	IOB33	IO_L15N_T2_DQS_12	OUTPUT	LVCMOS33	12
9	W20	SEGMENT<4>	IOB33	IO_L15P_T2_DQS_12	OUTPUT	LVCMOS33	12
10	AC24	SEGMENT<5>	IOB33	IO_L14N_T2_SRCC_12	OUTPUT	LVCMOS33	12
11	AC23	SEGMENT<6>	IOB33	IO_L14P_T2_SRCC_12	OUTPUT	LVCMOS33	12
12	AA22	SEGMENT<7>	IOB33	IO_L13N_T2_MRCC_12	OUTPUT	LVCMOS33	12
13	AC18	clk	IOB	IO_L13P_T2_MRCC_32	INPUT	LVCMOS18*	32
14	W23	led	IOB33	IO_L8P_T1_12	OUTPUT	LVCMOS33	12
15	AA10	sw	IOB	IO_L14P_T2_SRCC_33	INPUT	LVCMOS15	33
16	T7		IOB18	IO_25_VRP_34	UNUSED		34

图 3.15 约束结果图

对 16 位可逆二进制同步计数器，将 bit 文件复制到连接实验箱的电脑上，对硬件设备进行下载编程，验证输出是否满足设计要求。



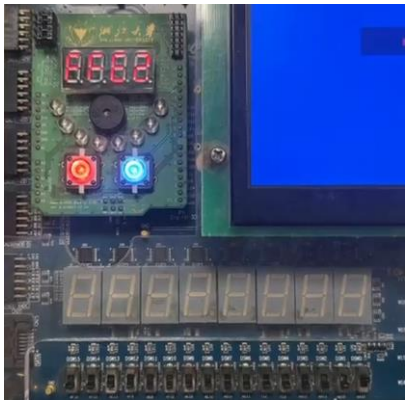


图 3.16 验证结果 1

如上面两图，右数第一个开关控制自增/自减。下拨为自减，上拨为自增。计数从 0000 开始，先下拨开关（如上左图），一段时间后显示 FFd2，说明自减正常。再上拨开关（如上右图），一段时间后数码管显示 FFE2，说明自增正常。（由于位数多、计数快，仅选取其中的几个显示展示）

接下来验证自增到进位和自减借位时的 Led 灯行为。

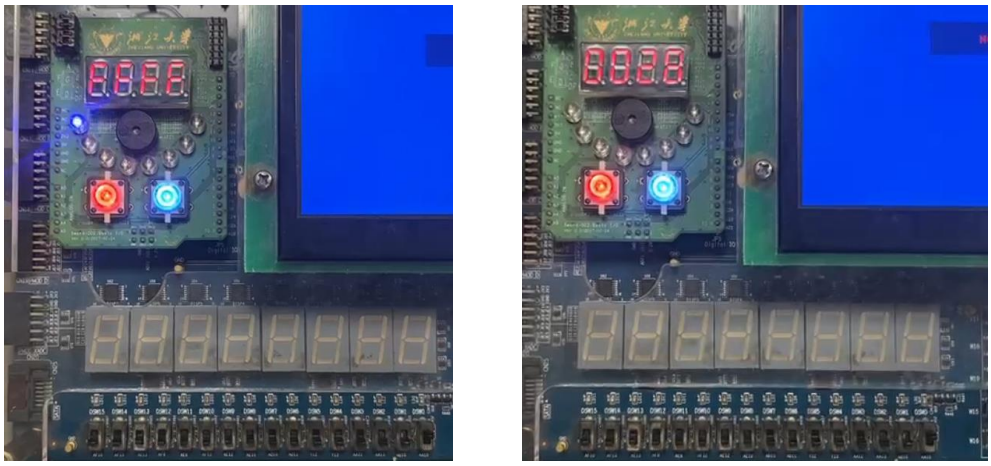


图 3.17 验证结果 2

紧接着前面的图，自增到 FFFF 时，如上左图，Led 灯亮起，代表进位成功，且该灯仅在这 100ms 亮起一瞬间。如上右图可以看到，显示为 0028，此时 Led 灯熄灭，且再次验证了前面的自增操作。

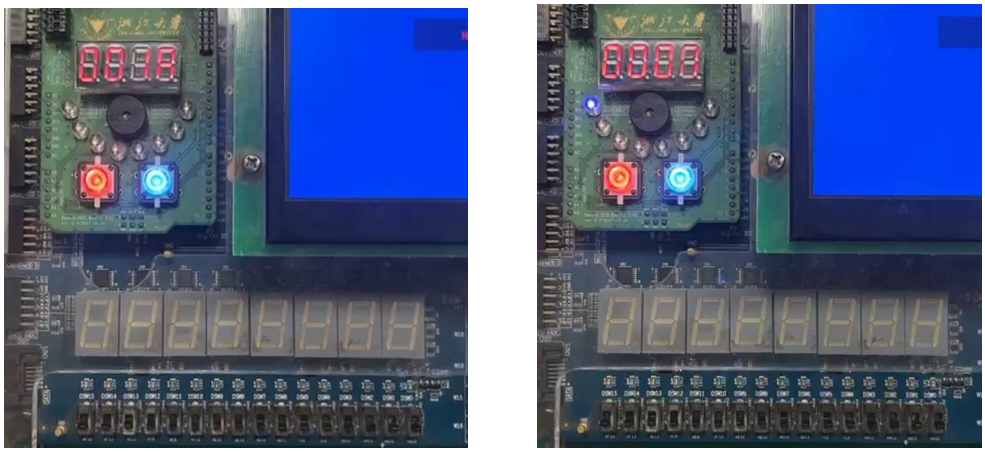


图 3.18 验证结果 3

紧接着前面的图，我们下拨开关，如上左图，开始自减，由 0028 自减到 001A。继续运行，如上右图，自减到 0000 时，Led 灯亮起，代表借位成功，符合设计要求

四、实验结果分析

1. 分析仿真结果

Counter4b 模块原理图以及对其进行仿真的激励代码和波形图结果如下，现进行分析。代码的意思和对应波形图的表现分析如下：

- always begin 产生一个周期 20ns 的时钟，可以看到时钟的波形有序地出现在波形图的第一行。
- 100ns 的初始化后计数器的各位开始有数据。先来看最低位 Qa，在原理图即为第一个 D 触发器器的输出。原理图的意思为，时钟的上升沿触发，将自身求反作为输出，可以看到波形图中的 Qa 每个周期高低电平变化一次，符合预期。
- 对第二行的触发器输出 Qb，当 Qa，即最低位为 1 的时候，需要将其求反得到新的 Qb。计数器意义上，即最低位为 1，下一周期加一，本位就需要接受进位，本位输出需要取反。波形图中，在 Qa 为高电平、时钟为上升沿时，Qb 的高低电平变化，即对自身求反。
- Qc 和 Qd 同理，不多赘述。现在看最后一行的四输入或非门的输出接的进位 Rc。显而易见，当四位全为 1 时，就需要进位了。这在波形图中也得到了证明。

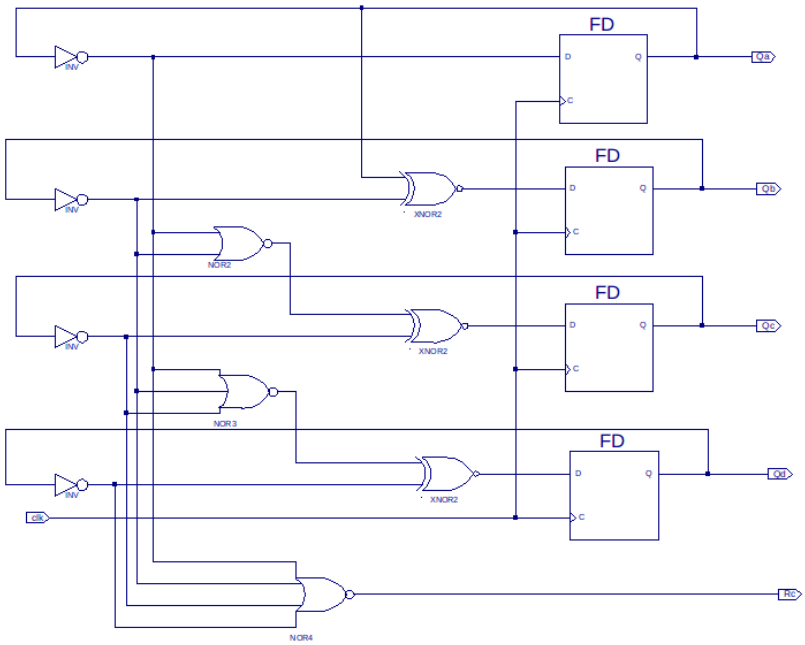


图 4.1 Counter4b 原理图

```
17  // Bidirs
18
19  // Instantiate the UUT
20     Counter4b UUT (
21         .Qb(Qb),
22         .Qc(Qc),
23         .Qd(Qd),
24         .Qa(Qa),
25         .clk(clk),
26         .Rc(Rc)
27     );
28  // Initialize Inputs
29     //`ifdef auto_init
30         always begin
31         clk = 0;#10;
32         clk = 1;#10;
33         end
34     //`endif
35  endmodule
36
```

图 4.2 Counter4b 模块仿真激励代码

上述代码生成的仿真波形如下：

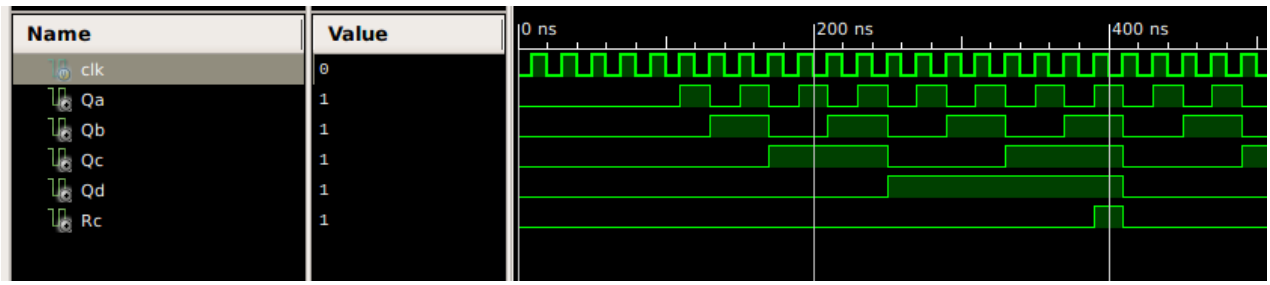


图 4.3 得到的仿真信号

对 RevCounter 模块进行仿真的激励代码和波形图如下，现进行分析。

- 仿真代码产生周期恒定的时钟，并在第 200ns 和第 500ns 改变自增自减的方向。
- 仿真波形第一行是计数的具体数值，每个时钟周期加一或者减一。自减到 fffb 时，s 变为高电平，更改增减方向。自增到 ffff 时产生进位，Rc 同时变为一周期的高电平。
- 自增到 0002 时再次更改 s 为低电平，自减到 0000，产生借位。波形图中可以看出，Rc 同时变为一时钟周期的高电平。符合预期。

```
43      initial begin
44          // Initialize Inputs
45          clk = 0;
46          s = 0;
47
48          // Wait 100 ns for global reset to finish
49          #200;
50          s=1;
51          #300;
52          s=0;
53          // Add stimulus here
54
55      end
56      always begin
57          clk=0;#20;
58          clk=1;#20;
59      end
60  endmodule
```

图 4.4 RevCounter 模块仿真激励代码

上述代码生成的仿真波形如下：

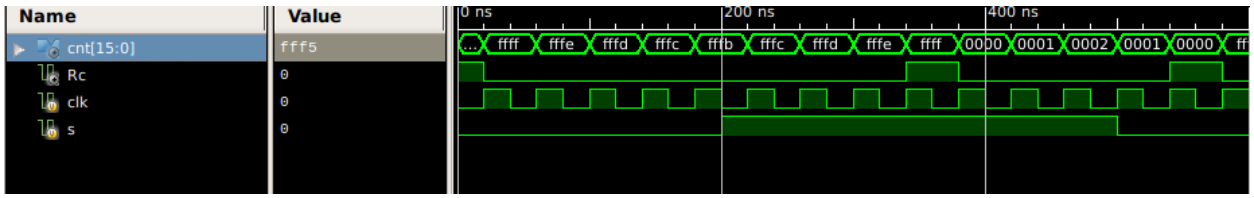
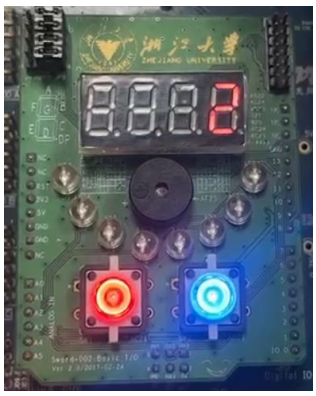
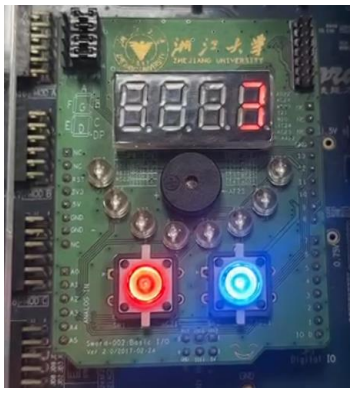


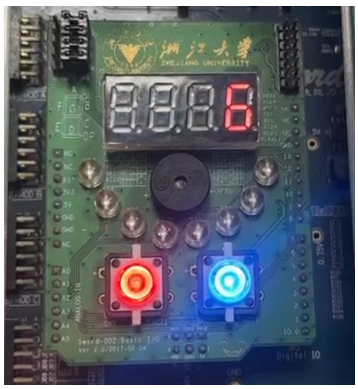
图 4.5 得到的仿真信号

2. 分析开发板结果

对正向计数器，将 bit 文件复制到连接实验箱的电脑上，对硬件设备进行下载编程，验证输出是否满足设计要求。下面一组图是数码管依次从 0 计数到 F 并进位的过程，可以看到进位的同时（显示 F 时），左数第一个 Led 灯亮起，说明满足设计要求。











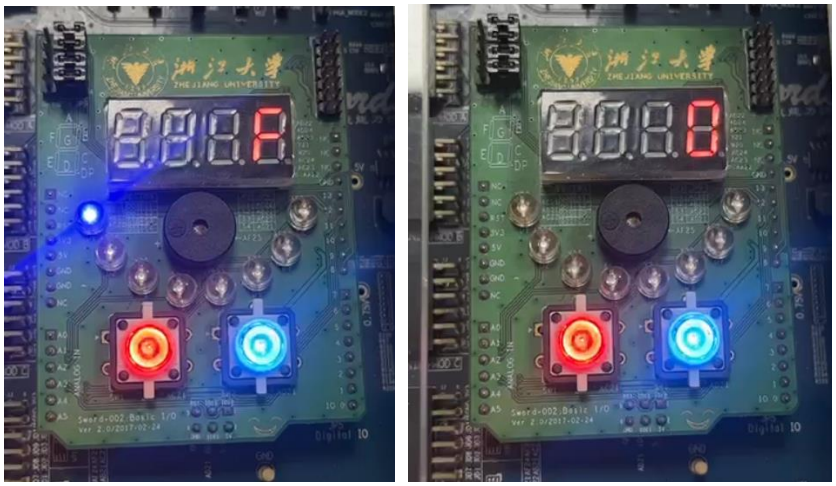


图 4.6 验证结果 1

对 16 位可逆二进制同步计数器，将 bit 文件复制到连接实验箱的电脑上，对硬件设备进行下载编程，验证输出是否满足设计要求。

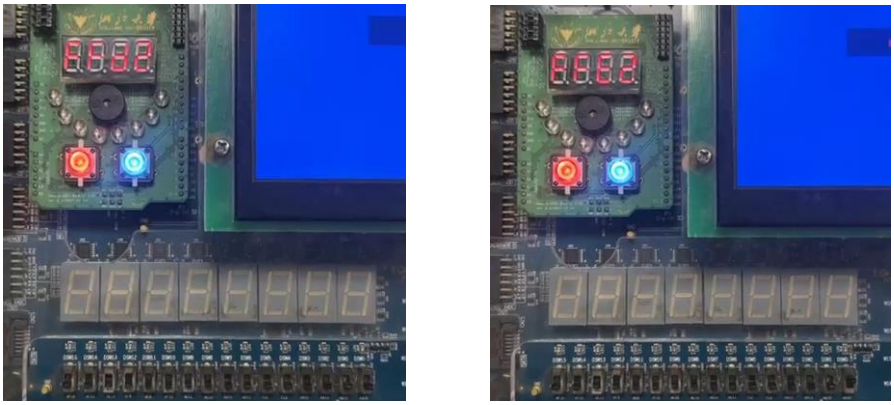


图 4.7 验证结果 2

如上面两图，右数第一个开关控制自增/自减。下拨为自减，上拨为自增。计数从 0000 开始，先下拨开关（如上左图），一段时间后显示 FFd2，说明自减正常。再上拨开关（如上右图），一段时间后数码管显示 FFE2，说明自增正常。（由于位数多、计数快，仅选取其中的几个显示展示）

接下来验证自增到进位和自减借位时的 Led 灯行为。

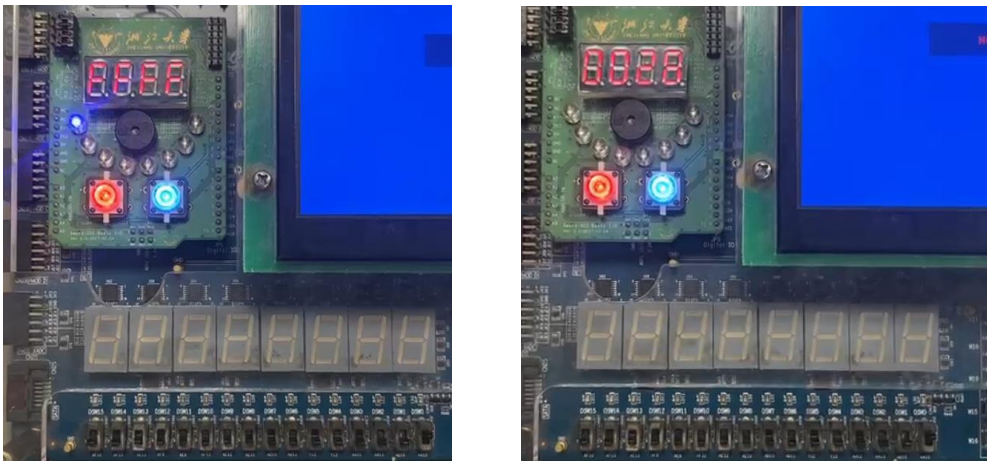


图 4.8 验证结果 3

紧接着前面的图，自增到 FFFF 时，如上左图，Led 灯亮起，代表进位成功，且该灯仅在这 100ms 亮起一瞬间。如上右图可以看到，显示为 0028，此时 Led 灯熄灭，且再次验证了前面的自增操作。

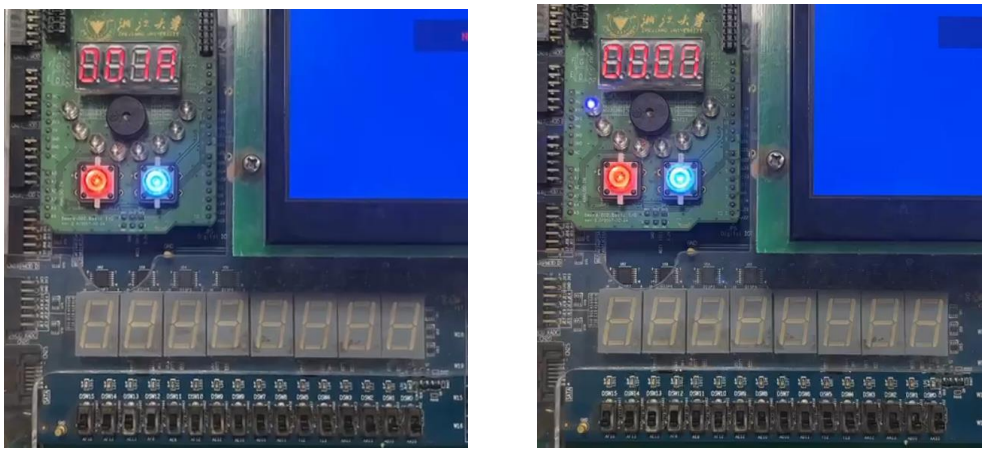


图 4.9 验证结果 4

紧接着前面的图，我们下拨开关，如上左图，开始自减，由 0028 自减到 001A。继续运行，如上右图，自减到 0000 时，Led 灯亮起，代表借位成功，符合设计要求。

3. 分析 Verilog 代码

用分频器原理设计的 1s 时钟的 Verilog 代码如下。100MHz 信号通过 50,000,000 次分频后，得到 1Hz 的秒脉冲方波，设计 1s 的时钟。分频的原理是，当 100MHz 的时钟经过 50,000,000 次的周期，相当于过了 0.5s，此时我们将输出的信号取反，即得到每 0.5s 改变一次的 1s 时钟。

```
22  module clk_1s(clk, clk_1s);
23  input wire clk;
24  output reg clk_1s;
25  reg [31:0] cnt;
26  always @ (posedge clk) begin
27     if (cnt < 50_000_000) begin
28         cnt <= cnt + 1'b1;
29     end else begin
30         cnt <= 0;
31         clk_1s <= ~clk_1s;
32     end
33  end
34  endmodule
```

图 4.10 clk_1s 模块的 Verilog 代码

下图 16 位可逆同步二进制计数器的行为描述代码，对其进行分析：

- 用一个 16 位的寄存器 cnt 计数，根据 s，每个时钟周期将寄存器原来的值+1 或者-1 存入下个周期的寄存器。
- Rc 的值取决于当前是自增还是自减，当自增时，若 cnt 为 ffff，则产生进位，反之则当 cnt 为 0000 时产生借位。

```
21  module RevCounter(clk, s, cnt, Rc);
22     input wire clk, s;
23     output reg [15:0] cnt;
24     output wire Rc;
25     initial cnt = 0;
26     assign Rc = (~s & (~|cnt)) | (s & (&cnt));
27     always @ (posedge clk) begin
28         if (s)
29             cnt <= cnt + 1'b1;
30         else
31             cnt <= cnt - 1'b1;
32     end
33  endmodule
```

图 4.11 16 位可逆同步二进制计数器的行为描述代码

4. 分析两种设计的差异

本次实验在四位计数器中用图形方式设计，在 16 位可逆同步二进制计数器采用行为描述进行设计。可以看到计数器的描述并不复杂，但要用逻辑门实现是很麻烦的。这意味着简单的行为描述代码下面可能有数不清的复杂电路，需要我们的深思。

五、讨论与心得

本次实验中，可以看到计数器的描述并不复杂，但要用逻辑门实现是很麻烦的。这意味着简单的行为描述代码下面可能有数不清的复杂电路，需要我们的深思。

遇到的一个困难仍然是之前数码管显示模块的错误，由于不当的模块复制和调用，显示出的数字有问题，并且很难改正。反复修改多次才完成了实验。