浙江大学

本科实验报告

课程名称：	数字逻辑设计
姓　　名：	
学　　院：	计算机科学与技术学院
专　　业：	计算机科学与技术
邮　　箱：	
QQ　　号：	
电　　话：	
指导教师：	蔡铭
报告日期：	2023 年 12 月 17 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：

实验地点：紫金港东四 509 室 实验日期：2023 年 12 月 6 日

一、操作方法和实验步骤

同步计数器

- 计数器用来存储特定事件或过程发生次数的设备，每个时钟输入的脉冲都会使计数器增加或减少。计数器电路通常由多个触发器级联连接而成。
- 本实验设计的**同步计数器**，所有触发器的时钟输入端连接在一起，由输入时钟脉冲触发，所有触发器的状态是同时改变的。

四位同步二进制计数器

- 四位同步二进制计数器的功能描述如下：该计数器将输入时钟信号，输出四位二进制值，其中，二进制从高位到低位分别为 Qd，Qc，Db，Da。在每一个时钟的上升沿，二进制的值就会自增。由此，我们可以得到该计数器的真值表。

	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

- 接下来，我们可以化简得到其布尔表达式。

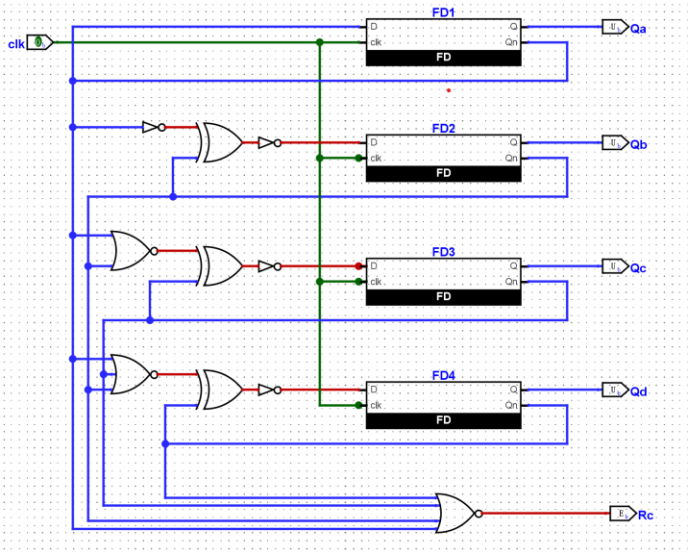
$$D_A = \overline{Q_A}$$

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

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

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

- 根据上述内容，我们可以绘制出同步计数器的原理图。



- 需要格外注意的是，由于我们绘制的 `D Latch` 无法实现自动初始化，因此在转化为 `verilog` 代码时我们需要重构触发器，让它能够在初始化条件下赋值为 `0`，防止其进入未定义状态。

- 下面是触发器的 `verilog` 代码。

```
module FD(
    input clk,
    input D,
    output Q,
    output Qn
);

    reg Q_reg = 1'b0;
    always @(posedge clk) begin
        Q_reg <= D;
    end
```

```
    assign Q = Q_reg;
    assign Qn = ~Q_reg;

endmodule
```

可逆二进制同步计数器

- 可逆二进制同步计数器可以实现以下功能：

    1. 在**时钟上升沿**对输出 `cnt` 进行修改；当 `s = 0` 时进行自增，`s = 1` 时进行自减。

    2. 非饱和计数，当前计数若为 `16'hFFFF` 则**自增**后为 `16'h0`；当前计数若为 `16'h0` 则**自减**后为 `16'hFFFF`。

- 由于该计数器的逻辑实现比较繁琐，绘制电路图的成本比较高，因此在这里我们选择用 `verilog` 代码实现。

```
`timescale 1ns / 1ps
/** module RevCounter
  * input
    *   clk: A clock signal driven by module clk_1s.
    *   s: 0 for increment, 1 for decrement
    * output
    *   cnt: a 16-bits register
    *   Rc: rise when the counter reset(i.e. carry will be set), that is,
 Rc becomes 1 when
    *         increment(s=0 & cnt=F) or decrement(s=1, cnt=0)
 */

module RevCounter(
        input wire clk,
        input wire s,
        input wire rst,
        output reg [15:0] cnt,
        output wire Rc
    );

     assign Rc = (~s&(~|cnt)) | (s & (&cnt));
     initial begin
        cnt = 0;
     end

     always @(posedge clk) begin
         if(s==0) begin
             if(cnt==16'hFFFF) begin
                 cnt <= 0;
             end
             else begin
                 cnt <= cnt + 1'b1;
             end
         end
         else begin
             if(cnt==16'h0000) begin
                 cnt <= 16'hFFFF;
             end
             else begin
```

```
                cnt <= cnt - 1'b1;
            end
        end
    end
endmodule
```

分频器的设计

- 由于 SWORD 板的时钟信号频率很高，直接用它作为输入可能导致显示的数字无法被我们观察到的情况，因此我们需要对时钟进行分频。分频之后，我们可以得到 1Hz 的脉冲方波，作为计数器的脉冲输入。

```
`timescale 1ns / 1ps

module clk_1s(
    input clk,
    output reg clk_1s
);

    reg [31:0] cnt;

    initial begin
        cnt = 32'b0;
    end

    wire[31:0] cnt_next;
    assign cnt_next = cnt + 1'b1;

    always @(posedge clk) begin
        if(cnt<50_000_000)begin
            cnt <= cnt_next;
        end
        else begin
            cnt <= 0;
            clk_1s <= ~clk_1s;
        end
    end

endmodule
```

下板验证

- 在完成上述的功能模块后，我们加入在之前设计好的 DispNumber 模块，最后完成 top 模块，就可以进行上板验证了。

```
module Top(
    input wire clk,
    input wire [1:0] SW,
    output wire LED,
    output wire [7:0] SEGMENT,
    output wire [3:0] AN
);

    wire[15:0] cnt;
    wire [3:0] Hex;
```

```
    wire clk_1s;

    clk_1s clk_div_1s (.clk(clk), .clk_1s(clk_1s));

    RevCounter counter(.clk(clk_1s), .rst(SW[1]), .s(SW[0]), .cnt(cnt),
 .Rc(LED));

    DisplayNumber display(.clk(clk), .rst(1'b0), .hexs(cnt), .LEs(4'b0
000), .points(4'b0000), .AN(AN), .SEGMENT(SEGMENT));

endmodule
```

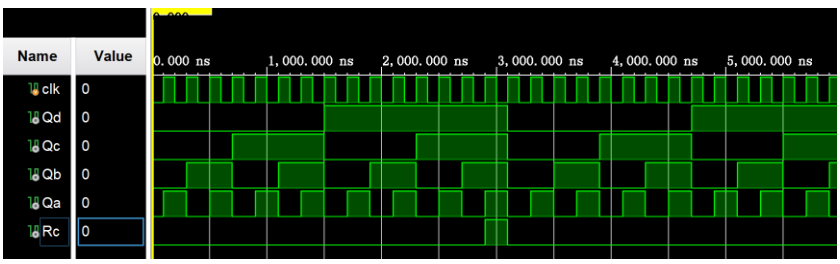
二、实验结果和分析

4 位二进制计数器仿真激励

- 以下是我们书写的仿真激励代码

```
`timescale 1ns / 1ps

module counter4b_sim();

reg clk;

Counter4b c0(
    .clk(clk),
    .Qa(Qa),
    .Qb(Qb),
    .Qc(Qc),
    .Qd(Qd),
    .Rc(Rc)
);

initial forever begin
    clk = 1'b0; #100;
    clk = 1'b1; #100;
end

endmodule
```

- 以下是仿真激励的结果



- 我们在仿真过程中模拟了 `clk` 信号的振荡，可以观察得到 `Q` 的输出在 `clk` 的上升沿发生变化。`{Qd, Qc, Qb, Qa}` 随振荡自增并呈周期性变化。`Rc` 给出进位信号。

4 位可逆二进制计数器上板验证

- 以下是本次实验中的引脚约束文件

```
# Main clock
set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]

create_clock -period 10.000 -name clk [get_ports "clk"]

set_property PACKAGE_PIN AA10 [get_ports {SW[0]}]
set_property PACKAGE_PIN AB10 [get_ports {SW[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW[1]}]


# LED
set_property PACKAGE_PIN AF24 [get_ports {LED}]
set_property IOSTANDARD LVCMOS33 [get_ports {LED}]

set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]
```

- 由于本次实验的上板验证很难用图片说明，因此我附上了一个视频。可以看到，该计数器可以实现自增自减自动计数的基本功能。

三、讨论、心得

- 本次实验主要完成了计数器的时序电路。通过原理图和 verilog 代码的书写，我对时序电路的具体实现有了更加的了解。
- 本次实验没有特别的难点，就是在搭建 top 模块的时候要注意以下各模块的关系并进行正确的连线。