長安大學

数字逻辑实验报告

课程名称：　数字逻辑

实验名称：　时序逻辑电路设计

专业名称：　计算机科学与技术/卓越工程师

小组成员：　刘文越、王子卓、孙翔、康旭

指导教师：　马峻岩

一、实验目的

1. 学习并掌握运用 Verilog 语言设计时序逻辑电路的原理和方法。
2. 掌握分频电路的设计方法；
3. 基于行为级描述实现 8 位移位寄存器；
4. 掌握 7 段数码管的译码及动态显示原理；
5. 基于行为级描述实现 4 位的十进制计数器；
6. 设计一个具有输入、输出、计时等功能的小型数字系统；

主要仪器和设备：计算机，Basys3 开发板。

二、实验要求

1. Verilog 实现 8 位移位寄存器

（1）功能要求

表 2-1 移位寄存器真值表

输　　入							输出
清零	控制信号		串行输入		时钟	并行输入	Q7~Q0
nclr	S1	S0	Dsr	Dsl	clk	D7~D0	
L	×	×	×	×	×	×	L
H	L	L	×	×	×	×	Q7~Q0
H	L	H	L	×	↑	×	L Q7~Q1
H	L	H	H	×	↑	×	H Q7~Q1
H	H	L	×	L	↑	×	Q6~Q0 L
H	H	L	×	H	↑	×	Q6~Q0 H
H	H	H	×	×	↑	D7~D0	D7~D0

① 移位寄存器功能如表 2-1 所示，寄存器为上升沿触发。

② Basys3 的外部时钟信号为 100MHz，将时钟信号分频至 1 Hz 后驱动移位寄存器时钟。

③ D7~D0 使用 sw7~sw0；S1 和 S0 为输入控制信号，使用 sw9 和 sw8；Dsr 串行右移输入端，使用 sw11；Dsl 串行左移输入端，使用 sw10；nclr 低电平有效的异步清零信号，使用 sw12；输出的 Q7~Q0 使用 LED7 到 LED0 显示寄存器数据。

④ 寄存器功能由同步的控制信号 s1 和 s0 决定，包括保持、右移、左移和置数。

（2）其他要求

① 要求撰写仿真程序，对代码进行仿真测试。

② 将仿真后的 Verilog 代码进行综合与实现，并下载到 Basys3 上验证。

2. Verilog 实现 4 位的十进制计数器

（1）功能要求

① 编写 4 位的十进制计数器，并将计数值显示在 4 位 7 段数码管上。

② 计数器驱动时钟频率为 100 Hz。

③ sw0 为高电平有效的同步清零信号，sw1 为高电平有效的同步计数使能信号。

（2）其他要求

① 要求撰写仿真程序，对代码进行仿真测试。

3. Verilog 实现秒表

（1）功能要求

① 4 位 LED 数码管 M.SS.D，其中最低位 D 代表 0.1 秒，范围是 0 到 9；SS 代表秒，范围是 00 到 59；M 代表分钟，范围是 0 到 9。

② 使用 **2^n 分频**将数码管扫描频率设定在 1000 Hz 左右。

③ sw0 为高电平有效的同步清零信号，sw1 为高电平有效的同步计数使能信号。

（2）其他要求

① 使用 Logisim 自带模块，在给定的 logisim_basys3.circ 上实现上述功能。

② 源文件 div.v 实现分频，display.v 实现数码管显示功能，timer.v 为计时功能，timer_tb.v 为仿真文件。

③ 要求撰写仿真程序，对代码进行仿真测试。

④ 将仿真后的 Verilog 代码进行综合与实现，并下载到 Basys3 上验证。

三、相关外设接口及原理

1. Basys3 开发板拨码开关原理图及简介

拨码开关的电路如图 3-1 所示。当开关打到下档时，FPGA 对应引脚输入为低电平；当开关打到上档时，FPGA 对应引脚输入为低电平。

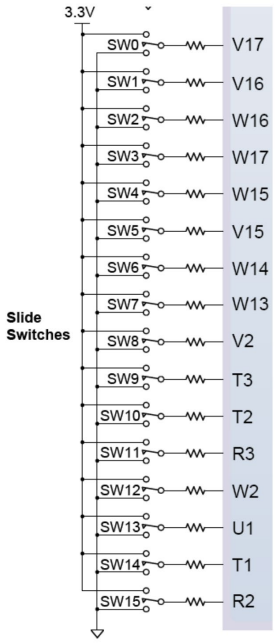

图 3-1 板拨码开关原理图

2. Basys3 开发板数码管电路原理图及简介

数码管显示部分的电路如图 3-2 所示。这里是四位带小数点的七段共阳数码管，当相应的输出脚为低电平时，该段位的 LED 点亮。位选位也是低电平选通。图 3-3 位数码管的十六进制数表示。表 3-1 为共阳极数码管十六进制显示真值表。进行显示时可以通过

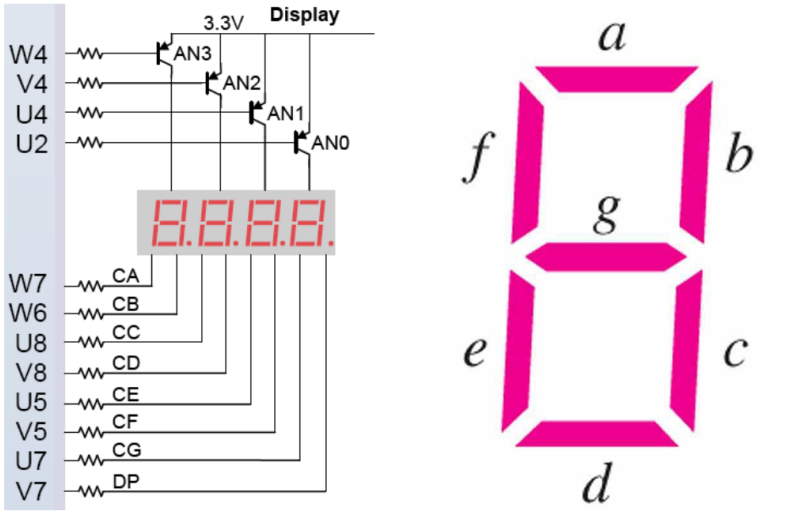

图 3-2 四位数码管电路原理图

图 3-3 数码管十六进制数表示

表 3-1 共阳极数码管十六进制显示真值表

X	a	b	c	d	e	f	g
0	0	0	0	0	0	0	1
1	1	0	0	1	1	1	1
2	0	0	1	0	0	1	0
3	0	0	0	0	1	1	0
4	1	0	0	1	1	0	0
5	0	1	0	0	1	0	0
6	0	1	0	0	0	0	0
7	0	0	0	1	1	1	1
8	0	0	0	0	0	0	0
9	0	0	0	0	1	0	0
A	0	0	0	1	0	0	0
B	1	1	0	0	0	0	0
C	0	1	1	0	0	0	1
D	1	0	0	0	0	1	0
E	0	1	1	0	0	0	0
F	0	1	1	1	0	0	0

3. 数码管扫描显示原理

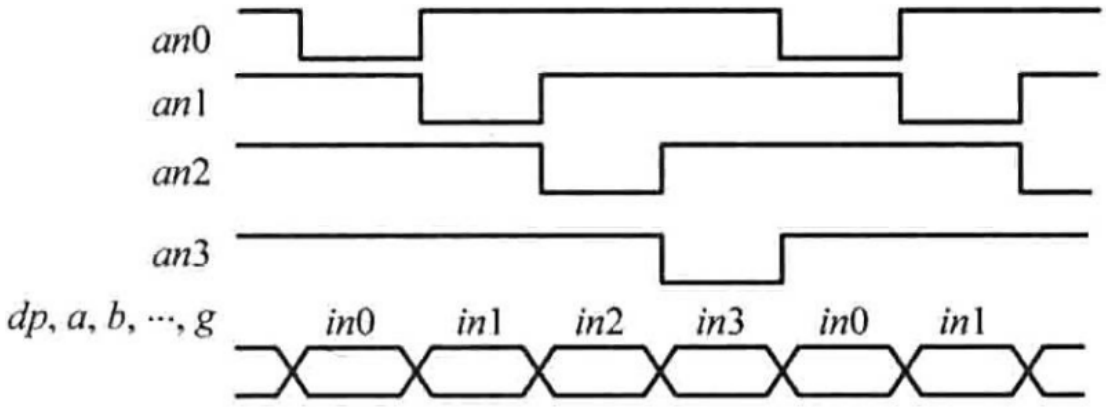

图 3-4 数码管扫描显示电路时序图

图 3-4 给出了四位数码管扫描显示电路时序图，这种控制方式采用分时复用的模

式循环轮流点亮数码管。在同一时间只会点亮一个数码管，点亮的同时公共断码信号输入端 a~g 和 dp 送入对应位需要显示的内容。分时复用的扫描显示利用了人眼的视觉暂留特性，如果扫描速度足够快，人眼就无法区分 LED 的闪烁。然而，如果数码管扫描频率过快会超过数码管开关切换的响应速度，也会产生显示的问题。扫描频率可选 1000Hz 左右。

4. 2^n 分频原理

2^n 分频电路是偶分频电路的一种。为了实现 2^n 分频，需要定义一个长度为 n-1 位的寄存器 clkdiv[n-1:0]，并用输入频率驱动对 clkdiv[n-1:0]进行自增计数，则 clkdiv[n-1]位对应输出为 2^n 分频。除了的到 2 的 n 次幂分频，还可以从 clkdiv 其他位得到 2 的 n-1 次幂、n-2 次幂、……、1 次幂分频，表 3-2 给出 2^n 分频在 100MHz 输入下的分频表。

表 3-2 2^n 分频在 100MHz 输入下的分频表

表达式	2^n 分频	频率（MHz）
clkdiv[0]	2^1 分频	100 / 2 = 50
clkdiv[1]	2^2 分频	100 / 4 = 25
clkdiv[2]	2^3 分频	100 / 8 = 12.5
clkdiv[3]	2^4 分频	100 / 16 = 6.25
clkdiv[4]	2^5 分频	100 / 32 = 3.125
clkdiv[5]	2^6 分频	100 / 64 = 1.5625
clkdiv[6]	2^7 分频	100 / 128 = 0.78125
clkdiv[7]	2^8 分频	100 / 256 = 0.39062
clkdiv[8]	2^9 分频	100 / 512 = 0.19531

5. 时序逻辑电路 Basys3 开发板相关约束设置

- **外部时钟信号约束文件**

Basys3 的外部时钟信号为 100MHz，通过 W5 引脚连接至 FPGA，使用时需要将约束文件 Clock Signal 段的时钟约束注释删除，示例代码如下。然后在顶层模块将 clk 信号作为 input 接入。

```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports clk]
set_property IOSTANDARD LVCMOS33 [get_ports clk]
```

四、实验步骤及关键代码

1. Verilog 实现 8 位移位寄存器

（1）主要模块代码

①设计文件：

shift_reg.v

```
`timescale 1ns / 1ps

module shift_reg(
    input [1: 0] sel,
    input [7: 0] D,
    input nclr, Dsr, Dsl, clk,
    output reg [7: 0] out
    );

    always @(posedge clk or negedge nclr) begin
        if(~nclr) out <= 8'b00000000;
        else
            case (sel)
                2'b01: out <= {Dsr, out[7: 1]};
                2'b10: out <= {out[6: 0], Dsl};
                2'b11: out <= D;
            endcase
    end
endmodule
```

②顶层文件：

shift_reg_top.v

```
`timescale 1ns / 1ps

module shift_reg_top(
    input [12: 0] sw,
    input clk,
    output [7: 0] led
    );

    parameter clk_number=26'd50000000;
    reg [25: 0] clk_count = 0;
    reg clk_div = 0;

    always @(posedge clk) begin
        if(clk_count == clk_number - 1) begin
```

```
            clk_count <= 0;
            clk_div <= ~clk_div;
        end
        else clk_count <= clk_count + 1;
    end

    shift_reg sr(
        .sel(sw[9: 8]),
        .D(sw[7: 0]),
        .Dsr(sw[11]),
        .Dsl(sw[10]),
        .clk(clk_div),
        .nclr(sw[12]),
        .out(led[7: 0])
    );
endmodule
```

（2）仿真测试代码

shift_reg_simulation.v

```
`timescale 1ns / 1ps

module shift_reg_simulation(
    );
    reg [1: 0] sel;
    reg [7: 0] D;
    reg nclr, Dsr, Dsl, clk;
    wire [7: 0] out;

    shift_reg sr(
        .sel(sel),
        .D(D),
        .nclr(nclr),
        .Dsr(Dsr),
        .Dsl(Dsl),
        .clk(clk),
        .out(out)
    );

    initial begin
        #0  clk = 1;
            D = 8'b11111111;
            nclr = 1;
            Dsl = 0;
```

```
            Dsr = 0;
            sel = 2'b00;
        #10 sel = 2'b11;
        #10 sel = 2'b00;
        #10 sel = 2'b01;
        #10 sel = 2'b01;
        #10 sel = 2'b10;
        #10 sel = 2'b10;
        #10 nclr = 0;
        #10 $finish;
    end
    always #5 clk = ~clk;
endmodule
```

（3）仿真波形图

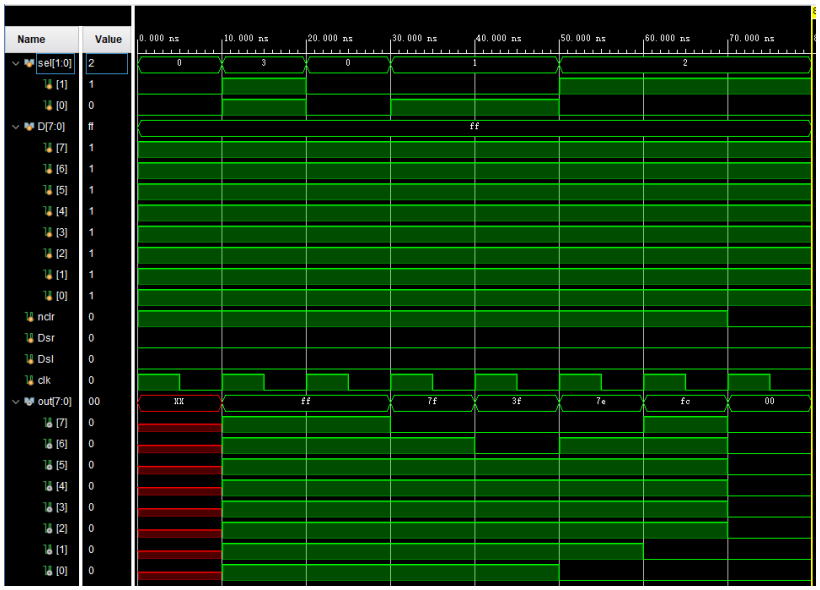

图 4-1 8 位移位寄存器仿真波形图

（4）开发板测试照片

测试照片 sw[7: 0]均为“1111111”

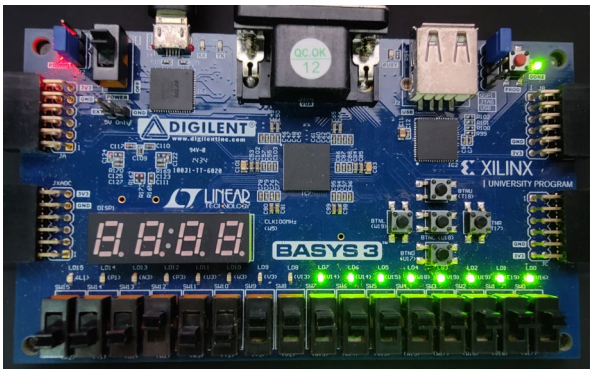

图 4-2 sw[9: 8]为 11，sw[11: 10]为 00，sw[12]为 1

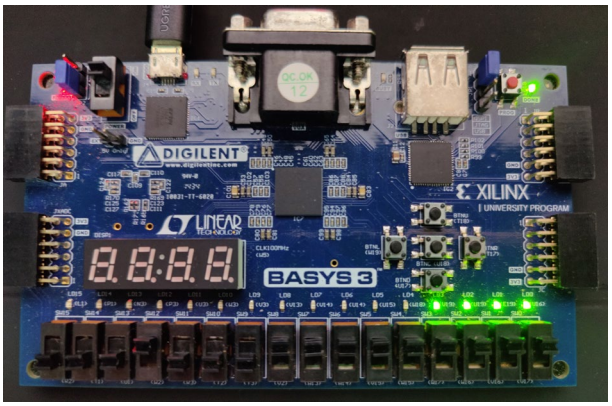

图 4-3 sw[9: 8]为 01，sw[11: 10]为 00，sw[12]为 1

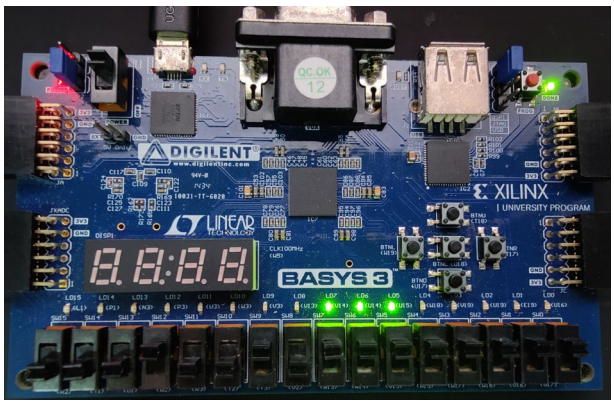

图 4-4 sw[9: 8]为 10，sw[11: 10]为 00，sw[12]为 1

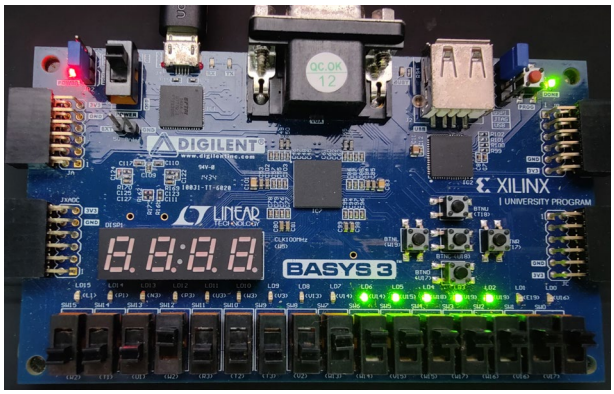

图 4-5 sw[9: 8]为 00，sw[11: 10]为 00，sw[12]为 1

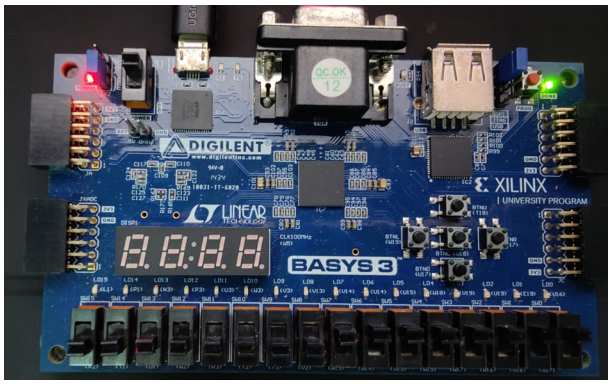

图 4-6 sw[12]为 0

2. Verilog 实现 4 位的十进制计数器

（1）主要模块代码

①设计文件：

counter_mod10.v

```
`timescale 1ns / 1ps

module counter_mod10(
    input clk, clr, en,
    output [3: 0] D,
    output out
    );
    reg [3: 0] regN;
    always @(posedge clk) begin
        if(clr || regN == 9) regN <= 4'b0000;
        else regN <= regN + en;
    end
    assign D = regN;
    assign out = (regN == 9) ? 1 : 0;
endmodule
```

hex7.v

```
`timescale 1ns / 1ps

module hex7(
    input [3: 0] D,
    output reg [6: 0] seg
    );

    always @(*) begin
        case(D)
            0:  seg = 7'b0000001;
            1:  seg = 7'b1001111;
            2:  seg = 7'b0010010;
            3:  seg = 7'b0000110;
            4:  seg = 7'b1001100;
            5:  seg = 7'b0100100;
            6:  seg = 7'b0100000;
            7:  seg = 7'b0001111;
            8:  seg = 7'b0000000;
            9:  seg = 7'b0000100;
            10: seg = 7'b0001000;
            11: seg = 7'b1100000;
            12: seg = 7'b0110001;
```

```
            13: seg = 7'b1000010;
            14: seg = 7'b0110000;
            15: seg = 7'b0111000;
        endcase
    end
endmodule
```

②顶层文件：

counter_mod10_top.v

```
`timescale 1ns / 1ps

module counter_mod10_top(
    input [1: 0] sw,
    input clk,
    output [0: 6] seg,
    output [3: 0] an,
    output dp
    );

    assign dp = 1;
    assign an = 4'b0000;
    parameter clk_number = 19'd500000;
    reg [18: 0] clk_count = 0;
    reg clk_div = 0;

    always @(posedge clk) begin
        if(clk_count == clk_number - 1) begin
            clk_count <= 0;
            clk_div <= ~clk_div;
        end
        else clk_count <= clk_count + 1;
    end

    wire [3: 0] D;

    counter_mod10(
        .clk(clk_div),
        .clr(sw[0]),
        .en(sw[1]),
        .D(D)
    );
    hex7(.D(D), .seg(seg));

endmodule
```

（2）仿真测试代码

counter_mod10_simulation.v

```
`timescale 1ns / 1ps

module counter_mod10_simulation(
    );
    reg clk, clr, en;
    wire [3: 0] D;

    counter_mod10 cm(
        .clk(clk),
        .clr(clr),
        .en(en),
        .D(D)
    );

    initial begin
        #0  clk = 1;
            clr = 1;
            en = 1;
        #10 clr = 0;
        #120 en = 0;
        #10 clr = 1;
        #10 $finish;
    end
    always #5 clk = ~clk;

endmodule
```

（3）仿真波形图

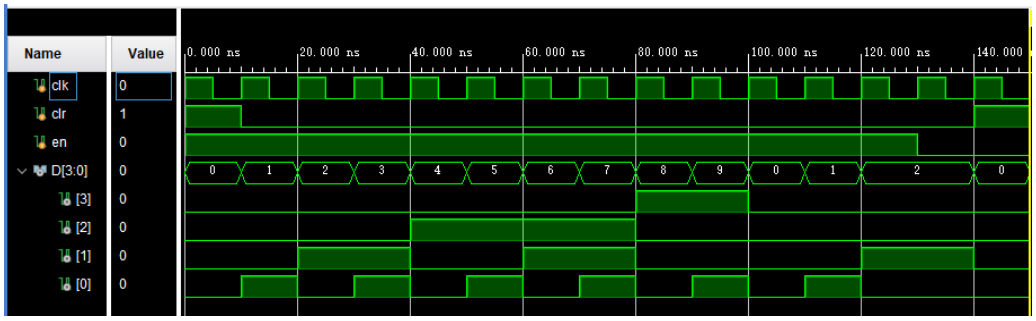

图 4-7 4 位十进制计数器仿真波形图

3. Verilog 实现秒表

（1）绘制主要块连接的逻辑框图

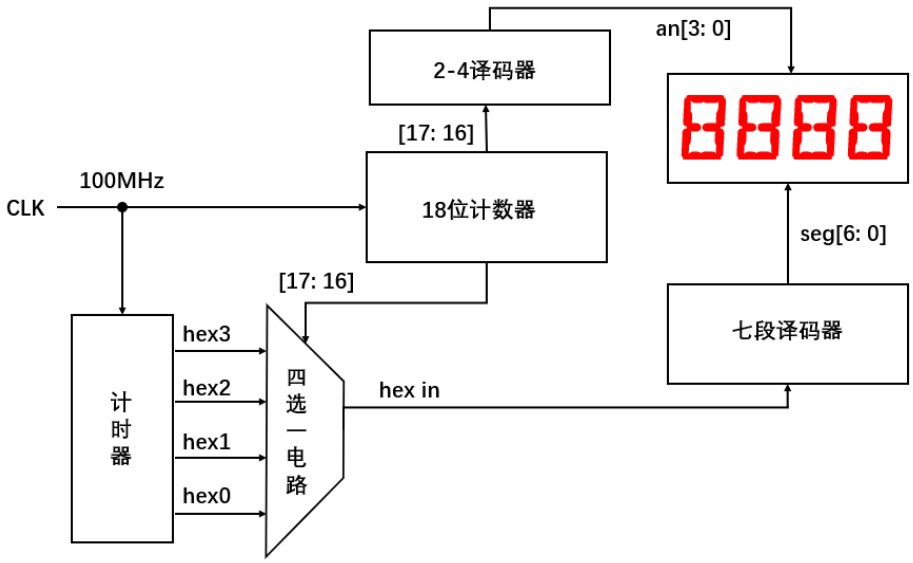

图 4-8 秒表逻辑框图

（2）主要模块代码

①设计文件：

timer.v

```
`timescale 1ns / 1ps

module timer(
    input clk, clr,
    output [3: 0] d0, d1, d2, d3
    );

    reg [3: 0] d0_reg, d1_reg, d2_reg, d3_reg;
    always @(posedge clk or posedge clr) begin
        if(clr) begin
            d0_reg = 0;
            d1_reg = 0;
            d2_reg = 0;
            d3_reg = 0;
        end
        else if(d0_reg < 9) d0_reg = d0_reg + 1;
        else begin
            d0_reg = 4'b0000;
            if(d1_reg < 9) d1_reg = d1_reg + 1;
            else begin
                d1_reg = 4'b0000;
                if(d2_reg < 5) d2_reg = d2_reg + 1;
```

```
                else begin
                    d2_reg = 4'b0000;
                    if(d3_reg < 9) d3_reg = d3_reg + 1;
                    else d3_reg = 4'b0000;
                end
            end
        end
    end

    assign d0 = d0_reg;
    assign d1 = d1_reg;
    assign d2 = d2_reg;
    assign d3 = d3_reg;

endmodule
```

div.v

```
`timescale 1ns / 1ps

module div(
    input clk,
    output [17: 0] clk_div
    );
    reg [17: 0] clk_reg = 0;
    always @(posedge clk) begin
        clk_reg <= clk_reg + 1;
    end
    assign clk_div = clk_reg;
endmodule
```

display.v

```
`timescale 1ns / 1ps

module display(
    input [3: 0] d0, d1, d2, d3,
    input [1: 0] sel,
    output reg [6: 0] seg,
    output reg [3: 0] an,
    output reg dp
    );

    reg [3: 0] D;

    always @(*) begin
        case (sel)
```

```
            2'b00: an = 4'b1110;
            2'b01: an = 4'b1101;
            2'b10: an = 4'b1011;
            2'b11: an = 4'b0111;
        endcase
    end

    always @(*) begin
        case (sel)
            2'b00: D = d0;
            2'b01: D = d1;
            2'b10: D = d2;
            2'b11: D = d3;
        endcase
    end

    always @(*) begin
        case (sel)
            2'b00: dp = 1;
            2'b01: dp = 0;
            2'b10: dp = 1;
            2'b11: dp = 0;
        endcase
    end

    always @(*) begin
        case(D)
            0:  seg = 7'b0000001;
            1:  seg = 7'b1001111;
            2:  seg = 7'b0010010;
            3:  seg = 7'b0000110;
            4:  seg = 7'b1001100;
            5:  seg = 7'b0100100;
            6:  seg = 7'b0100000;
            7:  seg = 7'b0001111;
            8:  seg = 7'b0000000;
            9:  seg = 7'b0000100;
            10: seg = 7'b0001000;
            11: seg = 7'b1100000;
            12: seg = 7'b0110001;
            13: seg = 7'b1000010;
            14: seg = 7'b0110000;
            15: seg = 7'b0111000;
        endcase
```

```
    end

endmodule
```

stopwatch.v

```
`timescale 1ns / 1ps

module stopwatch(
    input clk, clr, en,
    output [6: 0] seg,
    output [3: 0] an,
    output dp
    );

    localparam count_value = 5000000;
    wire [17: 0] clk_div;
    wire [3: 0] D0, D1, D2, D3;
    reg [22: 0] ms_reg = 0;
    reg clk_10HZ = 1;

    always @(posedge clk) begin
        if(ms_reg < count_value-1) ms_reg <= ms_reg + en;
        else begin
            ms_reg <= 0;
            clk_10HZ <= ~clk_10HZ;
        end
    end

    div(.clk(clk), .clk_div(clk_div));
    timer(
        .clk(clk_10HZ),
        .clr(clr),
        .d0(D0),
        .d1(D1),
        .d2(D2),
        .d3(D3)
    );
    display(
        .d0(D0),
        .d1(D1),
        .d2(D2),
        .d3(D3),
        .sel(clk_div[17: 16]),
        .seg(seg),
        .an(an),
```

```
        .dp(dp)
    );
endmodule
```

②顶层文件：

stopwatch _top.v

```
`timescale 1ns / 1ps

module stopwatch_top(
    input clk,
    input [1: 0] sw,
    output [0: 6] seg,
    output [3: 0] an,
    output dp
    );

    stopwatch(
        .clk(clk),
        .clr(sw[0]),
        .en(sw[1]),
        .seg(seg),
        .an(an),
        .dp(dp)
    );
endmodule
```

（3）仿真测试代码

timer_tb.v

```
`timescale 1ns / 1ps

module timer_tb(
    );
    reg clk, clr;
    wire [3: 0] d0, d1, d2, d3;

    timer t(
        .clk(clk),
        .clr(clr),
        .d0(d0),
        .d1(d1),
        .d2(d2),
        .d3(d3)
    );
```

```
    initial begin
        #0      clk = 1;
                clr = 0;
        #1300   clr = 1;
        #100    $finish;
    end
    always #1 clk = ~clk;
endmodule
```

（4）仿真波形图

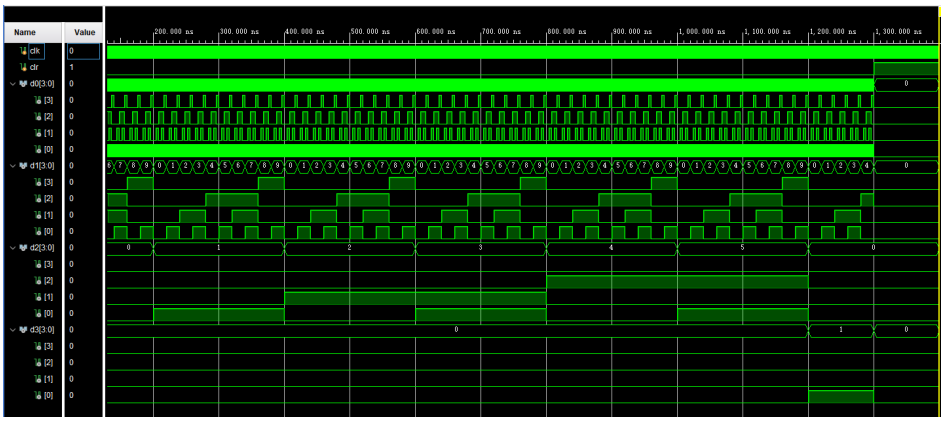

图 4-9 秒表仿真波形图

（5）开发板测试照片

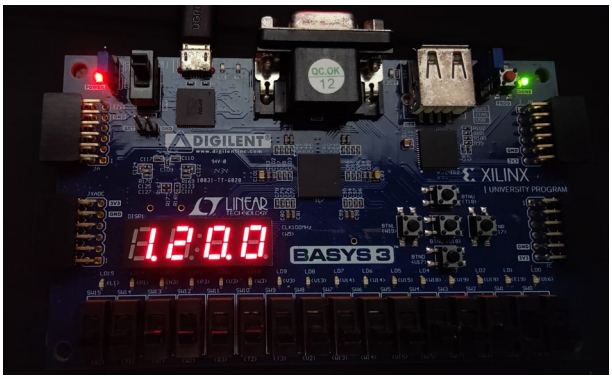

图 4-10 sw[0]为 0，sw[1]为 1

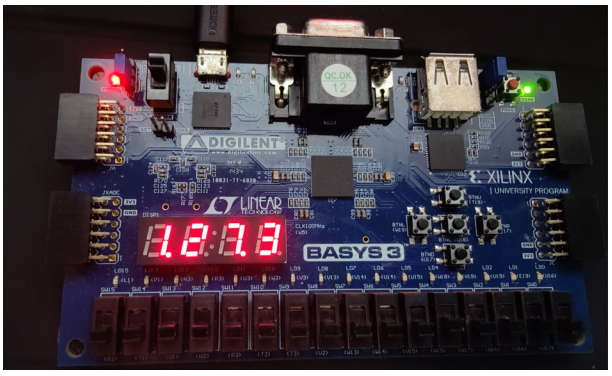

图 4-10 sw[0]为 0，sw[1]为 0

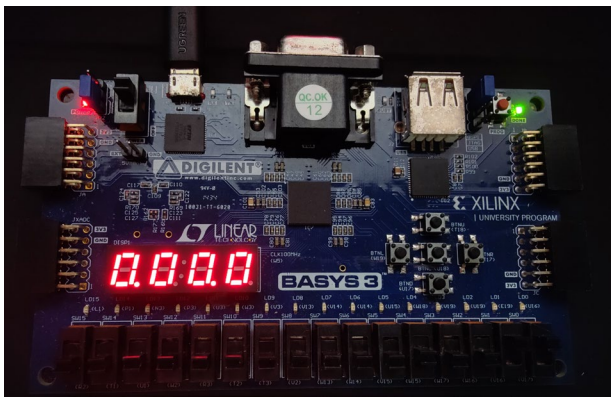

图 4-11 sw[0]为 1，sw[1]为 1

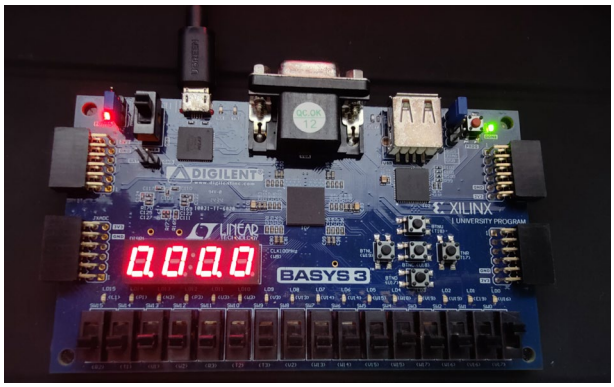

图 4-12 sw[0]为 0，sw[1]为 1