Lab 4 Report

PB22020514 郭东昊

一、 实验目的与内容

实验内容

设计定时器和分频器 (Timer & Frequency Divider, TFD)，并下载测试

实验目的

1. 熟练掌握计数器与移位器的功能及其应用：分频、定时、开关输入去抖动、数码管动态扫描显示等
2. 熟练掌握 Verilog 描述组合和时序逻辑电路的方法
3. 熟练掌握利用 EDA 工具，进行逻辑电路的设计、仿真、调试、下载测试等基本方法
4. 熟练掌握查看生成电路及其性能和资源使用情况

二、 代码实现

1. 定时器

（写代码的时候搞错了，以为 TFD 是定时器。）

```
module TFD #(
    parameter WIDTH = 32,
              RST_VLU = 0
)(
    input [WIDTH-1:0] k,//定时常数，定时时长 = (k+1)*clk
    input st,//定时开始信号
    input rst,
    input clk,
    output reg [WIDTH-1:0] q,//定时器输出信号
    output reg td//定时结束信号
    );
```

```
    reg [WIDTH-1:0] counter = 0; // 计数器

    always @(posedge clk) begin
        if (!rst) begin
            counter <= 0; // 复位计数器到 0
            td <= 1; // 设置定时结束信号
        end else if (st) begin
            counter <= k; // 装入定时常数
            td <= 0; // 清除定时结束信号
        end else if (counter > 0) begin
            counter <= counter - 1; // 开始倒计时
        end else if (counter == 0) begin
            td <= 1; // 输出定时结束信号
        end // 更新上一个时钟周期的定时开始信号
        q <= counter; // 输出计数器
    end
endmodule
```

2. 顶层文件实现

端口声明：

```
module top_TFD(
    input               clk,        // 时钟信号
    input               rstn,       // 复位信号
    input        [15:0] sw,         //sw15-0
    input               st,         //开始计时
    input               ent,
    input               del,
    input               pre,
    input               nxt,
    output     [15:0] taddr,      //led15-0
    output     [ 7:0] an,         // 数码管位选信号
    output      [ 6:0] seg,        // 数码管段选信号
    output reg           td          // 定时器到时信号，接入任何一个led灯即可
);
```

使用到的变量：

```
    wire twe;
    wire rst, tclk;
    wire  [15:0] number_h;
    wire  [15:0] number_l;
    wire   [31:0]  count; //32位当前剩余时间
    wire   [31:0]  count_reg; //32位当前剩余时间
    reg [26:0] counter = 0; // 分频器计数器
    reg clk_out2 = 0; // 分频后的时钟信号
    reg [31:0] tdin;
    wire [31:0]  tdout;
```

接入定时器的分频器模块：

```
always @(posedge clk) begin
    if(counter < 100000000) begin // 100MHz的时钟分频到1s
        counter <= counter + 1;
    end else begin
        counter <= 0;
        clk_out2 <= ~clk_out2; // 翻转输出信号
    end
end

```

例化定时器与 utu：

```
    // 实例化定时器模块
    TFD timer (
        .k({number_h,number_l}),
        .st(st),
        .rst(rstn),
        .clk(clk_out2),
        .q(count),
        .td(td)
    );

```

```
    utu UTU(
        .clk(clk),
        .rstn(rstn),
        .x(sw),
        .ent(ent),
        .del(del),
        .step(0),
        .pre(pre),
        .nxt(nxt),
        .taddr(taddr),
        .tdin(tdin),
        .tdout(tdout),
        .twe(twe),
        .rst(rst),
        .tclk(tclk),
        // .an(an),
        // .seg(seg)
    );
```

输入输出寄存器：

```
    register# ( .WIDTH(16), .RST_VAL(0))
    Number_h (
        .clk    (clk),
        .rst    (rst),
        .en     (taddr == 16'h0 && twe),
        .d      (tdout),
        .q      (number_h)
    );

    register# ( .WIDTH(16), .RST_VAL(0))
    Number_l (
        .clk    (clk),
        .rst    (rst),
        .en     (taddr == 16'h1 && twe),
        .d      (tdout),
        .q      (number_l)
    );

    register# ( .WIDTH(32), .RST_VAL(0))
    Count (
        .clk    (clk),
        .rst    (rst),
        .en     (1'b1),
        .d      (count),
        .q      (count_reg)
    );
```

接入七段数码管的分频器模块：

```
    reg [26:0] counter1 = 0; // 分频器计数器
    reg clk_out1 = 0; // 分频后的时钟信号

    always @(posedge clk) begin
        if(counter1 < 5000000) begin // 100MHz的时钟分频到20ms
            counter1 <= counter1 + 1;
        end else begin
            counter1 <= 0;
            clk_out1 <= ~clk_out1; // 翻转输出信号
        end
    end
```

七段数码管使用到的变量声明：

```
    // 数码管段选信号的生成逻辑
    reg [2:0] digit = 0; // 数码管位选计数器
    reg [3:0] display [0:7]; // 数码管显示值的寄存器数组

    always_comb begin
        display[0] = count_reg[3:0];
        display[1] = count_reg[7:4];
        display[2] = count_reg[11:8];
        display[3] = count_reg[15:12];
        display[4] = count_reg[19:16];
        display[5] = count_reg[23:20];
        display[6] = count_reg[27:24];
        display[7] = count_reg[31:28];
    end
```

数码管位选与段选信号的生成逻辑：

```
149     always @(posedge clk_out1) begin
150         digit <= digit + 1; // 更新数码管位选计数器
151         case(display[digit])
152             4'b0000: seg <= 7'b0000001;
153             4'b0001: seg <= 7'b1001111;
154             4'b0010: seg <= 7'b0010010;
155             4'b0011: seg <= 7'b0000110;
156             4'b0100: seg <= 7'b1001100;
157             4'b0101: seg <= 7'b0100100;
158             4'b0110: seg <= 7'b0100000;
159             4'b0111: seg <= 7'b0001111;
160             4'b1000: seg <= 7'b0000000;
161             4'b1001: seg <= 7'b0000100;
162             4'b1010: seg <= 7'b0001000;//A
163             4'b1011: seg <= 7'b1100000;//b
164             4'b1100: seg <= 7'b0110001;//C
165             4'b1101: seg <= 7'b1000010;//d
166             4'b1110: seg <= 7'b0110000;//E
167             4'b1111: seg <= 7'b0111000;//f
168             default: seg <= 7'b0000000;
169         endcase
170         an <= ~(1 << digit); // 更新数码管位选信号
171     end
```

输入逻辑：

```
140     always@(*) begin
141         case(taddr)
142         16'h0:   tdin = number_h;
143         16'h1:   tdin = number_l;
144         16'h2:   tdin = count_reg;
145         default: tdin = 0;
146         endcase
147     end
```

三、 仿真结果与分析

1. 定时器

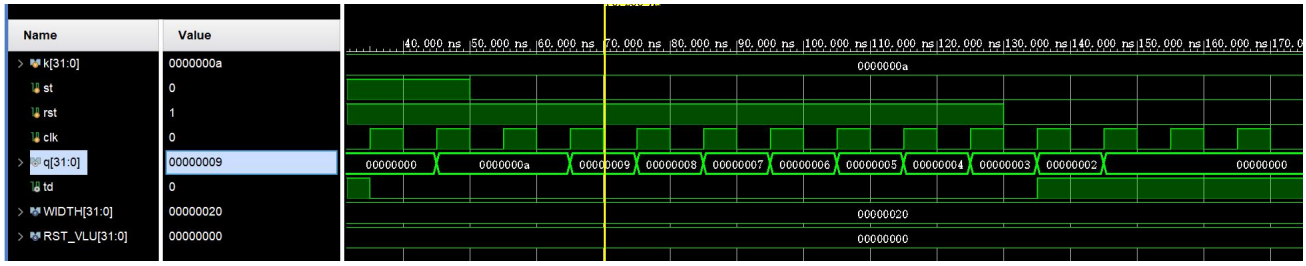
仿真文件全部代码

```
module TFD_tb();

    // Parameters
    parameter WIDTH = 32;
    parameter RST_VLU = 0;
```

```
    // Inputs
    reg [WIDTH-1:0] k;
    reg st;
    reg rst;
    reg clk;
    // Outputs
    wire [WIDTH-1:0] q;
    wire td;
    // Instantiate the TFD module
    TFD #(WIDTH,RST_VLU) dut (
        .k(k),
        .st(st),
        .rst(rst),
        .clk(clk),
        .q(q),
        .td(td)
    );
    //Clock generation
    always begin
        #5 clk = ~clk;
    end
    // Testbench logic
    initial begin
        // Initialize inputs
        k = 10;
        st = 0;
        rst = 0;
        clk = 0;
        // Apply reset
        #10 rst = 1;
        // Wait for a few clock cycles
        #20;
        // Start the timer
        st = 1;

        #20 st = 0;
        // Wait for the timer to finish
```

```
            #60;
            // Stop the timer
            st = 0;
            // Wait for a few clock cycles
            #20;
            // Apply reset
            rst = 0;
            // Wait for a few clock cycles
            #10;
        end
    endmodule
```

运行截图



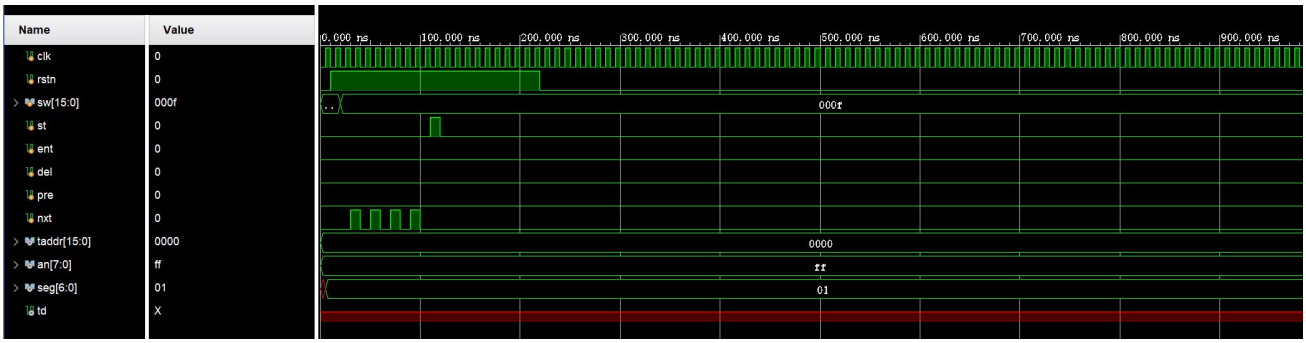
开始时，k 置为 10（0000000a），当 st = 1 即计时启动后，q[31:0]开始从 10 倒数，倒数至 2 时，由于 rst（低电平有效）置为 0，停止计时，td（停止计时标志）被置为 1，q 被直接置为 0，并没有经过 1。

2. 顶层应该要上板的 TOP 文件

仿真文件核心代码

```
58      // 初始化输入信号并进行测试
59      initial begin
60          clk = 0;
61          rstn = 0;
62          sw = 16'h0;
63          st = 0;
64          ent = 0;
65          del = 0;
66          pre = 0;
67          nxt = 0;
68
69          #10 rstn = 1; // 复位结束
70          #10 sw = 16'hF; // 改变sw的值
71          #10 nxt = 1;
72          #10 nxt = 0;//输入sw的值到taddr(16'd0)
73          #10 nxt = 1;
74          #10 nxt = 0;//将taddr切换到下一个地址(16'd1)
75          #10 nxt = 1;
76          #10 nxt = 0;//输入sw的值到taddr(16'd1)
77          #10 nxt = 1;
78          #10 nxt = 0;//将taddr切换到下一个地址(16'd2)
79          #10 st = 1;
80          #10 st = 0;//开始计时，在taddr2观察倒计时的实时显示
81
82          #100 rstn = 0; // 结束仿真
83      end
```

运行



可惜的是，仿真运行结果平静地如一潭死水，在经过我三整天的调试后依然如此，我决定放弃继续该实验的 debug，仿真与上板。

四、 测试结果与分析

实验失败。

五、 总结

本次实验我未能成功完成，是在编写含 utu 的 top 文件时出了问题，我花了大量的时间用于解决这个问题，但是最终无功而返。在 debug 的过程中，我用到了很多方法，如：编写不同的仿真文件（前后改了很多很多版）、在仿真运行时监测顶层模块中各个子模块的变量赋值是否异常、对照以前可以运行的 top 文件试图发现问题等等。同时，我也深刻认识到自己的不足，我在没做出实验的情况下，没有向助教和身边的同学请教，甚至因为其他事务繁忙而翘了一节课，更减少了我能向助教、同学请教的时间，这是非常不应该的。

通过本次实验，我熟练掌握了计数器与移位器的功能及其应用：分频、定时、开关输入去抖动、数码管动态扫描显示等。在写完上述代码后，我注意到，其实在学期初张俊霞老师给我们的 utu.sv 文件中，分频、定时、开关去抖动、数码管动态扫描显示都有相对应的实现代码，我如获至宝，阅读这些优秀的代码并与自己写的相应代码做了比较，分析优劣。