中国科学技术大学计算机学院

《数字电路实验》报告

实验题目：Verilog 硬件描述语言

学生姓名：Ouedraogo Ezekiel B.

学生学号：PL19215001

完成日期：11/12/2020

计算机实验教学中心制

2020 年 10 月

【实验题目】

Verilog 硬件描述语言

【实验目的】

掌握 Verilog HDL 常用语法
能够熟练阅读并理解 Verilog 代码
能够设计较复杂的数字功能电路
能够将 Verilog 代码与实际硬件相对应

【实验环境】

PC 一台：Windows 操作系统
Notepad++ 文本编辑器。

【实验过程】

1. Verilog 关键字

Verilog 语法中常用的关键字包括：
module/endmodule、

```
module test();
    //code
endmodule
```

input、output、

```
module test(
    input a,b,clk,
    output o);
    //code
endmodule
```

wire、reg、 assign、always、

```
module test(
    input wire a,b,
    output reg o
    );

    wire s;
    assign s = a & b;
    always@( * ) o = s;
```

```
endmodule
```

begin/end、posedge、negedge、if、else、

```
module test(
    input wire a,b,clk,
    output reg o);

    always@( posedge clk )
    begin
        if(a) o <= a;
        else o <= b;
    end

endmodule
```

case、endcase

```
module test(
    input wire a,b,clk,
    output reg o);

    always@( posedge clk )
    case({a,b})
        2'b00: o <= 1'b0;
        2'b01: o <= 1'b0;
        2'b10: o <= 1'b0;
        2'b11: o <= 1'b1;
    endcase

endmodule
```

2. 代码实例

8bit 位宽 4 选 1 选择器，纯组合逻辑

```
module mux_4to1( //8bit 位宽的 4 选 1 选择器
    input [7:0] a,b,c,d,
    input [1:0] sel,

    output reg [7:0] o); //always 语句内赋值的信号都应定义成 reg 类型


    always@(*) //always 语句内实现组合逻辑
```

```
    begin
        case(sel)
            2'b00: o = a; //组合逻辑使用"="进行赋值
            2'b01: o = b;
            2'b10: o = c;
            2'b11: o = d;
            default: o = 8'h0;//用 case 语句实现组合逻辑时一定要有 default
        endcase
    end

endmodule
```

1~10 循环计数的计数器

```
module cnt_1to10(
    input clk,rst_n,
    output reg [3:0] cnt);

    always@(posedge clk or negedge rst_n)
    //时序控制条件为时钟上升沿和复位的下降沿
    begin
        if(!rst_n) //复位信号优先级最高，应是第一个判断的条件
            cnt <= 4'h1;
        else if(cnt>=10)
            cnt <= 4'h1;
        else
            cnt <= cnt + 4'h1;
    end

endmodule
```

【实验练习】

1. 修改代码

```
//修改后代码
module test(
    input a,
    output b);
```

```
    //if, else 不能直接在模块内部单独出现。一
    always @ (*)
    begin
        if(a) b = 1'b0;
        else b = 1'b1
    end

endmodule
```

2. 将空白部分补充完整

```
//补充后代码
module test(
input [4:0] a,
output reg [4:0] b);

    always@(*)
        b = a;

endmodule
```

3. 输出信号的值

```
//代码
module test(
    input [7:0] a,b,
    output [7:0] c,d,e,f,g,h,i,j,k );

    assign c = a & b;
    assign d = a | b;
    assign e = a ^ b;
    assign f = ~a;
    assign g = {a[3:0],b[3:0]};
    assign h = a >> 3;
    assign i = &b;
    assign j = (a > b) ? a : b;
    assign k = a - b;

endmodule
```

当 a = 8'b0011_0011, b = 8'b1111_0000 时
c = 8'b11_0000

d = 8'b1111_0011
e = 8'b1100_0011
f = 8'b1100_1100
g = 8'b0011_0000
h = 8'b110
i = 8'b0
j = 8'b1111_0000
k = 8'b0100_0011

4. 修改代码

```
//修改后代码
module sub_test(
    input a,b,
    output c); //c 没有在 always 语句内部被赋值
        assign c = (a<b)? a : b;
endmodule

module test(
    input a,b,c,
    output o);


        wire temp;  //现在 sub_test 模块的输入输出变量都是 wire 类型
        //位置或名称两种关联方式不能混用
        //模块调用最好得有实例化名
        sub_test sub_test1(.a(a),.b(b),.c(temp));
        sub_test sub_test2(.a(temp),.b(c) ,.c(o));

endmodule
```

5. 修改代码

```
//修改过代码
module sub_test(
    input a,b
    output o);//output 一般用在模块的端口定义部分。
```

```
    assign o = a + b;

endmodule
module test(
    input a,b,
    output c);


    //always@(*) 这没意义
    sub_test sub_test(a,b,c);

endmodule
```

【总结与思考】

通过本次实验咱们掌握 Verilog 的常用语法，并用其来设计电路。