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：刘良宇

学生学号：PB20000180

完成时间：2021. 11. 11

实验题目

Verilog 硬件描述语言：

在简单组合逻辑和时序逻辑的实验中已经对 Verilog 语法进行了简单的介绍，用户应该已经可以阅读和编写简单的 Verilog 代码，本次实验中，我们将进一步系统的学习 Verilog 代码。

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

- vlab.ustc.edu.cn
- verilog.ustc.edu.cn

实验练习

题目 1.

阅读以下 Verilog 代码，找出其语法错误，并进行修改

```
module test(
input a,
output b);
    if(a) b = 1 ' b0;
    else b = 1 ' b1;
endmodule
```

- if 和 else 一般出现在 always 语句的过程语句部分，而不能直接在模块内部单独出现。

所以根据语义，改写成：

```
module test(input a,
            output b);
    assign b = a ? 1'b0 : 1'b1;
endmodule
```

题目 2.

阅读以下 Verilog 代码，将空白部分补充完整

```
module test(
input [4:0] a,
_____________);
always@(*)
    b = a;
_____________
```

注意这里是过程赋值，b 应该是一个寄存器类型信号。

故补全代码为：

```
module test(input [4:0] a,
            output reg [4:0] b);
always@(*)
    b = a;
endmodule
```

题目 3.

阅读以下 Verilog 代码，写出当 a = 8'b0011_0011, b = 8'b1111_0000 时，各输出信号的值。

```
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
```

阅读代码写出输出：

- c = 8'b0011_0000
- d = 8'b1111_0011
- e = 8'b1100_0011
- f = 8'b1100_1100
- g = 8'b0011_0000
- h = 8'b0000_0110
- i = 8'b0000_0000
- j = 8'b1111_0000
- k = 8'b0100_0011

题目 4.

阅读以下 Verilog 代码，找出代码中的语法错误，并修改

```
module sub_test(input a,
                b,
                output reg c);
    assign c = (a<b)? a : b;
endmodule

module test(input a,
            b,
            c,
            output o);
    reg temp;
    sub_test(.a(a),.b(b),temp);
    sub_test(temp,c,.c(o));
endmodule
```

- 端口信号可以通过位置或名称进行关联，但两种关联方式不能混用。
- 实例化时需要指定模块名。
- reg 不应该通过 assign 赋值。
- 实例化的输出端只能接 wire 类型信号。

可以写出修改后的代码：

```
module sub_test(input a,
                b,
                output reg c);
    always @(*)
    begin
        if (a<b) c = a;
        else c     = b;
    end
endmodule

module test(input a,
            b,
            c,
```

```
            output o);
    wire temp;
    sub_test s1(.a(a),.b(b),.c(temp));
    sub_test s2(temp,c,o);
endmodule
```

题目 5.

阅读以下 Verilog 代码，找出其中的语法错误，说明错误原因，并进行修改。

```
module sub_test(input a,
                b);
    output o;
    assign o = a + b;
endmodule

module test(input a,
            b,
            output c);
    always@(*)
    begin
        sub_test sub_test(a,b,c);
    end
endmodule
```

- output 一般用在模块的端口定义部分。
- always 语句相当于永远在循环执行，内部不能包含模块实例化。

于是写出修改后代码：

```
module sub_test(input a,
                b,
                output o);
    assign o = a + b;
endmodule

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

总结与思考

- 本次实验让我掌握 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。
- 本次实验的难易程度适中。任务量适中。
- 改进建议：无