**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：\_ \_\_\_\_ \_\_牛庆源\_\_\_\_\_ \_\_\_

学生学号：\_\_\_ \_\_\_\_PB21111733\_\_ \_\_\_\_\_\_

完成日期：\_\_\_\_ \_\_\_2022.11.3\_\_\_\_ \_\_\_\_

【实验题目】

阅读一些verilog代码并改错或者填写空白

【实验目的】

1. 了解verilog的基本语法，关键字，以及模块的调用之类的。

【实验环境】

Vscode

【实验练习】

题目一

module test(

input a,

output reg b);

always @ (\*)

begin

    if(a) b = 1'b0; // 组合逻辑赋值用等号

    else b = 1'b1;

end

endmodule

// if else语句用在always语句的过程语句部分，不能在模块内部独立出现

// always中赋值的变量用reg类型声明

题目二

module test(

    input [4:0] a,

    output reg [4:0] b);

always @ (\*)

    b = a;

endmodule

题目三

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

// 在输入a = 8’b0011\_0011,

// 且输入b = 8’b1111\_0000时

// 输出为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 = 1'b0000\_0000

// j = 8'b1111\_0000

// k = 8'b0100\_0011

题目四

module small\_test(

input a, b,

output c);

assign c = (a < b)? a : b;

endmodule

module test(

input a, b, c,

output o);

wire temp;

small\_test small\_test1(.a(a),.b(b),.c(temp));

small\_test small\_test2(.a(temp),.b(c),.c(o));

endmodule

// 模块实现的是比较两个一位二进制数的大小并输出的功能，模块名为sub\_test显然不合，改为small\_test

// small\_test两次调用应该赋予不用的模块名

// 模块调用时候通过名称关联和通过位置关联不能混用

// assign连续赋值语句中的变量用wire声明

// 模块中c是wire型变量，第一次使用模块时temp为reg型，所以应该改为wire型

题目五

module add\_test(

input a,b,

output o);

assign o = a + b;

endmodule

module test(

input a,b,

output c);

add\_test add\_test(a,b,c);

endmodule

// 首先模块是实现两个1位二进制数的加法，但模块名字为sub\_test显然不和，改为add\_test

// 模块调用时候若选择通过位置关联则需要知道模块声明有什么变量

// 这里只有input a,b，没有output，需要把output o放在模块的括号内声明

// always 中调用的模块内部使用 assign 时，output为wire则always报错，为reg则assign报错，删除always直接例化即可。

【总结与思考】

1. 了解了assign，always等verilog关键字；debug能力得到提高；对wire和reg类型更加敏感；熟悉了基本的模块调用方法。