**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：\_\_\_\_ 黄瑞轩\_ \_\_\_\_

学生学号：\_\_ PB20111686 \_\_\_

完成日期：\_\_ 2021.11.01 \_\_

计算机实验教学中心制

2020年09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**

* VLAB实验中心平台（vlab.ustc.edu.cn）
* verilog.ustc.edu.cn

**实验过程**

1. Verilog关键字

* 实验所用示例Verilog代码如下。

// module & endmodule

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

// code

endmodule

// wire & reg

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

wire s;

// code

endmodule

// assign & always

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

wire s;

assign s = a & b;

always@(\*) o = s; // \* 表示任意时序控制

endmodule

// posedge & negedge

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

wire s;

assign s = a & b;

always@(posedge clk) o <= s;

endmodule

// if & case

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

always@(posedge clk)

begin

if(a) o <= a;

else o <= b;

end

endmodule

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

always@(posedge clk)

case({a,b})

2’b00: 0 <= 1’b0;

2’b01: 0 <= 1’b0;

2’b10: 0 <= 1’b0;

2’b11: 0 <= 1’b1;

endcase

endmodule

1. Verilog代码基本结构
2. Verilog数据及类型
3. Verilog操作符
4. Verilog表达式
5. 模块调用

* 实验所用示例Verilog代码如下。

module add(input a, b, output sum, cout);

// 模块主体

endmodule

module full\_add(input a, b, cin, output sum, cout);

wire s, carry1, carry2;

add add\_inst1(a, b, s, carry1); // 通过位置关联

add add\_inst2(.a(s), .b(cin), .sum(sum), .cout(cout)); // 通过名称关联

endmodule

1. 代码实例

* 8 bit位宽的4选1选择器Verilog代码如下。

module mux\_4to1(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;

endcase

end

endmodule

* 1~10循环计数的计数器Verilog代码如下。

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下降沿

begin

if(!rst\_n) cnt <= 4’h1;

else if (cnt >= 10) cnt <= 4’h1;

else cnt <= cnt + 4’h1;

end

endmodule

**实验练习**

**【题目1】**

* 赋值应在赋值语句中进行。

module test(input a, output b);

assign b = a ? 1’b0 : 1’b1;

endmodule

**【题目2】**

* 两个语句空白应分别填入：

output reg [4:0] b

endmodule

**【题目3】**

* 根据代码，当输入a = 8’b0011\_0011和b = 8’b1111\_0000时，各输出信号如下。

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

注：声明i是8位变量，缩位之后最低位为结果，其他位用0补充以保证逻辑值正确。

**【题目4】**

* 首先，assign只能赋值wire类型的变量，这里c是reg类型，应当改为

module sub\_test(input a, b, output reg c);

always@(\*) c = (a < b) ? a : b;

endmodule

* 其次，第二个模块中，按名字实例化和按位置实例化在一次实例化中不能混用；并且不能用reg类型去实例化wire类型；自定义模块在实例化时需要指定实例名，应当改为

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

wire temp;

sub\_test s1(.a(a), .b(b), .c(temp));

sub\_test s2(.a(temp), .b(c), .c(o));

endmodule

**【题目5】**

* 首先，第一个模块的output端口被写在了端口定义外；其次不能在always语句中进行实例化；建议实例名不要与模块名重复，以免引起错误。
* 第一个模块的端口定义应当改成

module sub\_test(input a, b, output o);

* 第二个模块实例化语句应该改成

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

sub\_test sub\_test\_inst(a, b, c);

endmodule

**总结与思考**

* 本次实验与理论课部分重叠较多，但是再次阅读Verilog HDL的基本语法逻辑相当于复习，也有不错的效果。
* 本次实验难度很小，主要是跟着指导手册进行。
* 本次实验任务量不多，有关任务量的建议将在后面叙述。
* 改进建议：

由于理论课上已经比较系统地讲过Verilog HDL的基础语法，建议讲解一些可以调试Verilog HDL代码的方法，或者增加一些利用Vivado做仿真工作的练习。