**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：张郑飞扬

学生学号：PB21071416

完成日期：2022.10.27

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.ustc.edu.cn

verilog.ustc.edu.cn

**【实验过程】**

Step1：Verilog关键字

阅读实验手册，巩固学习了一些常用关键字。

Step2: Verilog代码基本结构

阅读实验手册，对Verilog代码的结构进行了进一步了解，总结如下——

*module 模块名（*

*输入端口定义 //输入端口只能是 wire 类型*

*输出端口定义 //输入信号可根据需要定义成 wire 或 reg 类型*

*）；*

*内部线信号定义 //内部信号可根据需要定义成 wire 或 reg 类型*

*模块实例化 //实例化的输出端只能接 wire 类型信号*

*assign 连续赋值语句*

*always 过程语句*

*endmodule*

Step3:Verilog数据及类型

了解了Verilog中的四种基本值:0,1,x(未知状态),z（高阻态）；三种常量：整数，实数，字符串；两种常用数据类型：wire(线网类型)和reg（寄存器类型）

Step4：Verilog操作符

Verilog 语法中的操作符，按照其功能可以分为：算数操作符、关系操作符、逻辑操作符、归约操作符、条件操作符、 移位操作符、 拼接操作符。大部分和C语言操作符相同，下面列出一些特别的：

~（按位取反） &（按位与） |（按位或）

^（按位异或） ~^（按位同或）

Step5: Verilog表达式

Verilog 中的表达式由操作数和操作符组成，可以在出现数值的任何地方使用。表达式可以是以下类型：常数、参数、线网、寄存器、位选择、部分选择、存储器单元、函数调用。这部分和C语言非常相似，但需要注意的是所有赋值语句是同时完成的，这和C语言不同。（除了begin end语法下）

Step6: 模块调用

一个模块能够在另一个模块中被引用，这样就建立了描述的层次，

使得设计大规模复杂电路的效率大大提高。这和C语言里面的函数调用有相似之处，模块实例语句的形式为：

模块名 实例化名（端口关联）；

需要注意的是有两种模块调用方式，一种是通过位置关联，另一种是通过名称关联。前者写法更简洁，后者可读性更强（推荐使用）。

下面列出两种方式的实例：

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

Step7: 代码实例

实验手册上给出了两个实例。一个是8bit位宽的4选1选择器；另一个是1~10的循环计数器。本人进行了阅读和理解。

**【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*

**错误：**if条件语句一般出现在 always 语句的过程语句部分，而不能直接在模块内部单独出现。

**修改：**

module test(

input wire a,

output reg b);

always@(\*)

begin

if(a) b = 1’b0;

else b = 1’b1;

end

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,

output [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

c=00110000 d=11110011 e=11000011 f=11001100 g=00110000

h=00000110 i=0 j=11110000 k=11000011

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语句只能赋值给线网类型信号，故应定义成wire型,可直接删去reg  
assign c = (a<b)? a : b;  
endmodule  
module test(  
input a,b,c,  
output o);  
~~reg~~ wire temp; //此处应定义为wire型，才能与上述模块中类型信号一致  
sub\_test~~(.a(a),.b(b),temp)~~ testa（.a(a）,.b(b),.c(temp));//端口信号关联时两种方式不能混用,并且应该补上它的实例化名称  
sub\_test~~(temp,c,.c(o))~~ testb（temp，c,o）;//端口信号关联时两种方式不能混用，,并且应该补上它的实例化名称  
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);//括号不应放在第二行后面，挪至第三行后。同时在正确位置上补上逗号和分号。

assign o = a + b;

endmodule

module test(

input a,b,

output c);

~~always@(\*)~~

~~begin~~

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

~~end~~

sub\_test sub\_test(a,b,c);//在always语句中，只能赋值个reg型信号，而引用的模块中信号全是wire型的，故不应该使用always

endmodule

**【总结与思考】**

本次实验使本人对Verilog语法有了进一步的学习和了解。