实验04 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】

- < www.vlab.ustc.edu.cn >
- <www.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
```

阅读代码有如下错误：

- 在非 always 块中使用 if-else
- 如要使用 if-else 需使用 always，与此同时 output b 应改为output reg b

修改代码如下：

```
module test(
        input 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,
_____________);
always@(*)
b = a;
____________
```

1. 由赋值语句可看出 b 也是 4bit
2. 由 always 语句块得知 b 的类型应该是 reg
3. 最后还缺少 endmodule

```
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
```

根据各运算符含义得到各结果为:
c = 8'b0011_0000
d = 8'b1111_0011
e = 8'b1100_1100
f = 8'b1100_1100
g = 8'b0011_0000
h = 8'b0000_0110
i = 0
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
```

阅读代码发现如下错误：

- sub_test 中 assign c, 所以 c 不应为 reg 型
- test 中实例化 sub_test，temp 的类型也不应为 reg
- 模块实例化方法不统一，基于端口以及基于位置的实例化混用
  修正代码如下:

```
module sub_test (
input a,b,
output c
```

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

module test (
input a,b,c,
output o
);
    wire temp;
    sub_test sub_test1 (.a(a), .b(b), .c(temp));
    sub_test sub_test2 (.a(temp), .b(c), .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
```

阅读代码发现有如下错误：

- sub_test 中 output o 未在端口处声明
- 实例化不可以在 always 中进行
  修改代码如下：

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

module test (
input a,b,
output c
);
sub_test sub_test1 (a, b, c);
endmodule
```

【总结与思考】

- 本次实验的收获
  - 感谢本次实验详细的指导手册，让我系统地了解了关于 Verilog 的更多关键字、代码结构、数据类型、运算符和语法
  - 能够顺利地阅读 Verilog 代码，推断出其所描述的电路的作用，并清楚代码综合能形成什么样的电路
  - 对 Verilog 代码的纠错能力得到了提升，也从侧面促进了自己书写规范代码的能力
- 本次实验的难易程度
  终于能说适中了。
- 本次实验的任务量
  非常好，非常让人舒服，虽然课下还是要自己刷 Verilog OJ.
- 对本次实验的建议
  没有建议，只希望以后的实验都这么让人舒适！