Lab 1 report

PB22111711 陈昕琪

实验目的与内容

> 1. 在Lab1 中，学习Verilog语言的基本知识，并进行简单的Verilog编程工作。
> 2. 将Verilog语言与硬件电路对应起来，理解硬件描述语言的内涵。

逻辑设计

> ### 1. 必做内容：向量翻转
>
> 要求： 将 8bit 的输入信号按位翻转，并输出到输出端口。 逻辑实现： verilog中默认的数据类型是wire类型，因此需要定义一个reg类型的变量，将输入的信号用always语句依次赋值到临时的temp中，再用assign语句将temp赋值给out。即可实现按位反向输出。
>
> **代码如下：**

```
module top_module(
    input  [7:0] in,
    output [7:0] out
);
// Your codes should start from here.
    reg [7:0] temp;

    always @ (*) begin
        temp[7] = in[0];
        temp[6] = in[1];
        temp[5] = in[2];
        temp[4] = in[3];
        temp[3] = in[4];
        temp[2] = in[5];
        temp[1] = in[6];
        temp[0] = in[7];
    end

    assign out = temp;
// End of your codes.
endmodule
```

> ### 2. 必做内容：最大值问题
>
> **(1)使用assign语句重新完成模块功能**
>
> 要求： 用assign语句重新完成区两个数最大值的操作。 逻辑实现： assign语句用于对wire型变量赋值，这里的max是wire型变量，因此需要用条件选择语句直接为max赋值。

代码如下：

```
module MAX2 (
    input  [7:0]          num1, num2,
    output [7:0]          max
);
// Your codes should start from here.
    assign max = ((num1 >= num2)?num1:num2);
// End of your codes.
endmodule
```

(2)获取三个数的最大值

要求：　获得三个数的最大值。需要参考 MAX2 的代码，使用 always 和 if-else 语句完成该功能。 逻辑实现：　对于三个数的比较，有三种情况，用if-else语句进行判断，并相应的用always语句对max赋值。

代码如下：

```
module MAX3 (
    input       [7:0]          num1, num2, num3,
    output reg  [7:0]          max
);
// Your codes should start from here.
always @(*) begin
    if (num1> = num2 && num1 >= num3)
        max = num1;
    else if (num2 >= num3 && num2 >= num1)
        max = num2;
    else
        max = num3;
end
// End of your codes.
endmodule
```

(3)例化MAX2

要求：　获得三个数的最大值，通过例化 MAX2 模块实现该功能 逻辑实现：　模块例化是指在一个模块中引用另一个模块，对端口进行相关连接。因此只需要先比较num1和num2，取出其中较大的值，赋值到reg类型的temp中，再将temp和num3作为输入端，将比较出的最大值赋值到max输出。 思考反思：　在初次写代码时，将MAX2写在了MAX3模块里面，并进行例化，这是没有理解模块例化的原因，更改后实现了正确的模块例化。　在实际编写过程中，用VScode作为编译器，发现如果不提前例化MAX2,会有错误提醒。

代码如下：

```
//模块声明
module MAX2 (
    input      [7:0]            num1, num2,
    output reg [7:0]            max
);
always @(*) begin
    if (num1 > num2)
        max = num1;
    else
        max = num2;
end
endmodule


module MAX3 (
    input   [7:0]         num1, num2, num3,
    output  [7:0]         max
);
    reg [7:0] temp;
// Your codes should start from here.
    MAX2 max2_1 (.num1(num1), .num2(num2), .max(temp));
    MAX2 max2_2 (.num1(num3), .num2(temp), .max(max));
// End of your codes.
endmodule
```

3.必做内容：1的个数

1. 要求：　给定一个位宽为 3 的信号 in，编写 Verilog 代码以输出其中为 1 的位的数目。
2. 逻辑实现：　对于位宽为3的信号，分别定义3个reg类型的变量来表示每一位是否为3，最后再相加赋值给out.
3. 思考反思：　初次写代码时，为了优化代码的可改性，仿照C语言编程中的i++思想，以及for循环思想。但是在助教检查提醒后得知这种自加在Verilog中是不正确的。　首先，自加会出现组合环，在实际电路中是错误的。其次Verilog中for循环容易出错，因此尽量不使用。

```
module Count4Ones(
    input       [2:0]         in,
    output reg  [1:0]         out
);
// Your codes should start from here.
    reg [1:0] temp1,temp2,temp3;

    always @(*) begin
        temp1 = (in[0])?1'b1:1'b0;
        temp2 = (in[1])?1'b1:1'b0;
        temp3 = (in[2])?1'b1:1'b0;
    end
    assign out = temp1+temp2+temp3;
// End of your codes.
endmodule
```

> 4.选择性必做内容：
>
> **题目2：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, l
);
assign c = a & b;
assign d = a || b;
assign e = a ^ b;
assign f = ~a;
assign g = {a[2:0], b[3:0], {1'b1}};
assign h = b >>> 3;
assign i = &b;
assign j = (a > b) ? a : b;
assign k = a - b;
assign l = !a;
endmodule
```

> **经运算，输出的信号值如下**

```
module test(
    input  [7:0]        a, b,
    output [7:0]        c, d, e, f, g, h, i, j, k, l
);
assign c = a & b; //按位与 c = 8'b0011_0000
assign d = a || b; //逻辑或 d = 8'bxxxx_xxx1
assign e = a ^ b; //按位异或 e = 8'b1100_0011
assign f = ~a; //按位非 f = 8'b1100_1100
assign g = {a[2:0], b[3:0], {1'b1}};//拼接运算符 g = 8'0110_0001
assign h = b >>> 3;//算数右移 h = 8'b1111_1110
assign i = &b; //归约与 i = 8'bxxxx_xxx0
assign j = (a > b) ? a : b;//判断最大值 j = 8'b1111_0000
assign k = a - b; //减 k = 8'b0100_0011
assign l = !a;//逻辑非 l = 8'bxxxx_xxx0
endmodule
```

> 反思思考：　在第一次检查时，有几个结果写错了。经助教检查提醒，得知对于assign d = a || b;这种用assign语句将1赋值给8位数，只有最低位的线被赋值，其他都接空，因此，应当用x表示没有赋值的位。

总结

> 1. 本次实验学习了 Verilog 语言的基本知识，并简单进行Vivado的环境配置，为实验二的仿真测试做准备。
> 2. 本次实验我初步了解认识了Verilog，但是在实验检查时，助教指出我的代码仍留有C语言的习惯和思想。应当认识到verilog是一门硬件编程语言，设计逻辑时应当从逻辑门判断和电路连接等等

> 多方面考虑。
> 3. 在实际编写Verilog代码时，仍存在语法不熟练，含义没有搞清楚等等问题。在今后的实验中，要学会适应Verilog的代码思想并熟练编写代码。