比较器实验报告

PB16030899 朱河勤

一、实验目的

掌握 4 位比较器的编码原理以及设计、调试方法，学会用用模块实例化。将 1bit 比较器构建成 4 比特比较器 on。

二、 实验内容

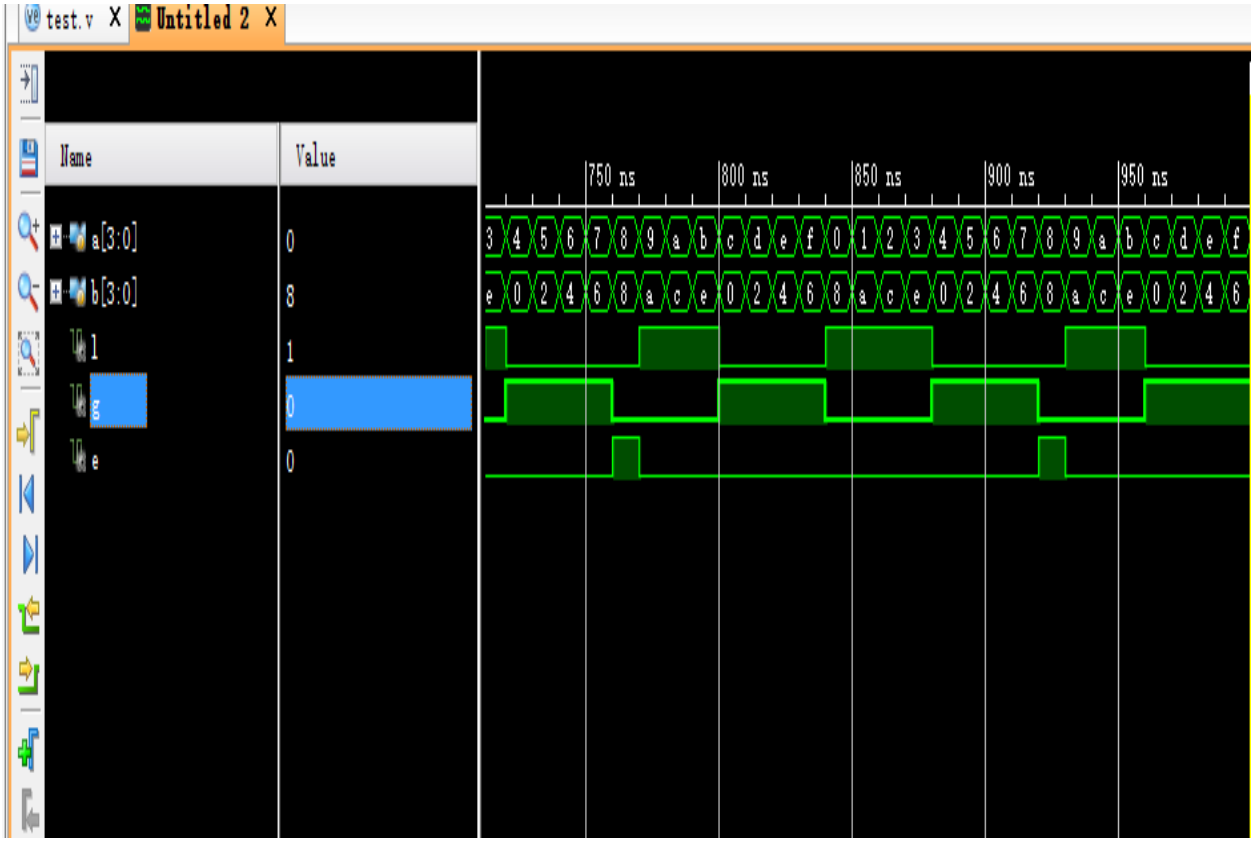
1. 设计一 4 位比较器
2. 画出门级电路图
3. 用 verilog 语言完成设计仿真
4. 在学习板上进行下载验证

a[3:0] = sw[7:4]

b[3:0] = sw[3:0]

{ob,oe,os} =Led[2:0]

三、仿真结果



四．实验分析

1. 首先要熟悉 verilog 语法，弄清楚仿真文件的写法以及实例化的方法。

2. 然后设计 1 位比较器。我的想法是：a，b 是带比较的，另有输入 l，g 表示高位比较的结果，如果 l=0,g=0,则表示高位相等，接着比较 a,b .如果 l=0，g=1 表示高位的 a>b,; l=1,g=-表示高位 b>a，这时就不需要比较 a,b，直接输出 l，g。

3. 对于仿真文件，我是让 a 增加 1，让 b 增加 2 来比较结果的

五、实验总结

通过本实验，了解了比较器逻辑功能和接线方法；加深队 Verilog HDL 硬件描述语言的基本语法的掌握。

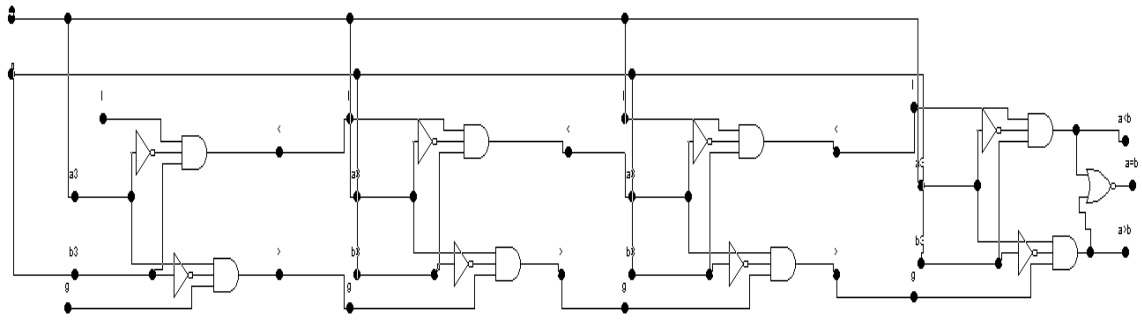
为保证电路正确，连线后可以在主要的逻辑步骤，加入一些额外输出，以检查电路各部分逻辑的正确性。一定得熟悉 verilog 语法，以避免浪费太多时间。

本来我在试验前已经写好代码在 u 盘里，但是忘了带，就重写了一遍。

重写的代码更加精炼简洁。

六．附录

原理图及源代码如下：



```
//`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2017/11/02 17:58:57
// Design Name:
// Module Name: top
```

```
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module cmp_1bit(
      input l,g,
      input a,b,
      output reg ol,og
      );
      always @(*)
        begin
            if(l|g) begin ol=l;og=g;end
            else if(a^b)begin ol=b;og=a;end
            else begin ol=0;og=0;    end
        end
endmodule

module cmp_4bit(
      input il,ig,
      input [3:0] a,[3:0] b,
      output l,g);
      wire [3:0] ol;
      wire [3:0] og;
      cmp_1bit t3(.l(il),.g(ig),.a(a[3]),.b(b[3]),.ol(ol[3]),.og(og[3]));
      cmp_1bit t2(.l(ol[3]),.g(og[3]),.a(a[2]),.b(b[2]),.ol(ol[2]),.og(og[2]));
      cmp_1bit t1(.l(ol[2]),.g(og[2]),.a(a[1]),.b(b[1]),.ol(ol[1]),.og(og[1]));
      cmp_1bit t(.l(ol[1]),.g(og[1]),.a(a[0]),.b(b[0]),.ol(ol[0]),.og(og[0]));
      assign l=ol[0];
      assign g= og[0];
endmodule


module top(
      input [7:0] SW,
```

```
    output [2:0] LED);
    wire l,g;
    cmp_4bit top(.il(0),.ig(0),.a(SW[7:4]),.b(SW[3:0]),.l(l),.g(g));
    assign LED[2] = g;
    assign LED[0] = l;
    assign LED[1] = (l&g)|(~l&~g);
endmodule
```