课程名称：EDA 技术综合设计

设计报告名称：设计二 数值比较器

班级：通信 214

姓名： 王峤宇

学号： 214022

一、设计内容及原理

基础任务

设计任务: 首先设计两个一位数值比较器，再设计一个四位数值比较器，输出结果用发光二极管显示即可。

一位数值比较器的真值表如表 1 所示。

表 1: 一位比较器真值表

A	B	Above	Equal	Below
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

部分四位数值比较器的真值表如表 2 所示。

表 2: 四位比较器真值表

A3	A2	A1	A0	B3	B2	B1	B0	Above	Equal	Below
1	0	0	0	0	x	x	x	1	0	0
x	1	0	0	0	0	x	x	1	0	0
x	x	1	0	0	0	0	x	1	0	0
x	x	x	1	0	0	0	0	1	0	0
0	x	x	x	1	0	0	0	0	0	1
0	0	x	x	x	1	0	0	0	0	1
0	0	0	x	x	x	1	0	0	0	1
0	0	0	0	x	x	x	1	0	0	1
0	0	0	0	0	0	0	0	0	1	0

针对四位数值比较器, 可以通过观察得知其真值表的规律, 与数值判断同理, 四位数值比较器可以有多个一位数值比较器组成, 依次从高位开始判断, 最终结果最高位决定, 若前一级的比较器得到了相等的结果则使能下一级比较器进行比较下一位。

提高内容

设计任务: 用一位数值比较器或者四位数值比较器构成一个八位数值比较器，输出结果用发光二极管显示即可（层次化设计方式）。

八位数值比较器构建与四位数值比较器的构建原理相同, 先比较高位, 高位比较器将比较结果传给下一级, 下一级判断输出结果, 与四位数值比较器用 1 为数值比较器构成略有不同的地方是, 如果采用四位数值比较器搭建, 该数值比较器需要能够接收来自上一级的四位数值比较器的输出结果。

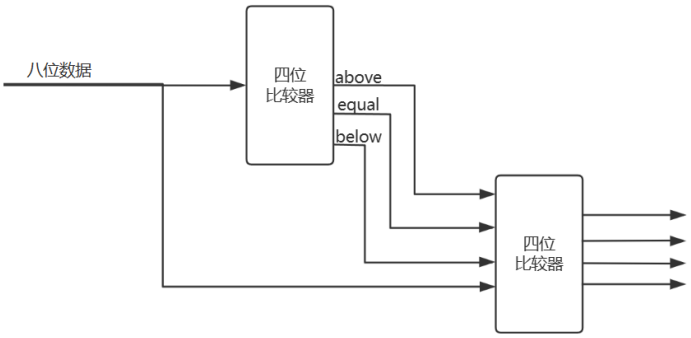

图 1: 8 位数值比较器模块框图

八位数值比较器的模块框图如图 1所示。

根据此设计的四位数值比较器需要如表 3 所示的接口。

表 3: 四位数值比较器接口

接口	接口类型	描述
A[3:0]	input	数据 A
B[3:0]	input	数据 B
I_Above	input	上级输出
I_Equal	input	上级输出
I_Below	input	上级输出
Y_Above	output	输出
Y_Equal	output	输出
Y_Below	output	输出

拓展任务

任务要求: 用一位数值比较器或者四位数值比较器构成一个八位数值比较器，输出结果用发光二极管显示即可（IP 核设计方式：自己把要用的低位数的数值比较器封装成 IP

核，然后调用设计八位数值比较器）。

该任务任然主要是通过四位数值比较器设计 8 位数值比较器, 与提高任务在原理上一致。主要是 IP 核封装的应用。IP 核是指芯片中具有独立功能的电路模块的成熟设计。该电路模块设计可以应用在包含该电路模块的其他芯片设计项目中，从而减少设计工作量，缩短设计周期，提高芯片设计的成功率, 利用 Vivado 自带的 IP 封装工具可以简单的将自己设计并测试完成的模块进行封装, 供之后调用。

二、设计过程

基础任务

一位数值比较器的源文件如下所示

Listing 1: 一位数值比较器源文件

```
module comparator_1bit(
    input a, b,
    output y_above, y_equal, y_below
    );
    assign y_above = a>b ? 1'b1 : 1'b0;      /* 置位a大于b */
    assign y_equal = a==b ? 1'b1 : 1'b0;     /* 置位相等 */
    assign y_below = a<b ? 1'b1 : 1'b0;      /* 置位a小于b */
endmodule
```

一位数值比较器的仿真文件如下所示

Listing 2: 一位数值比较器测试平台

```
module comparator_1bit_tb;
reg   a, b;
wire  y_above, y_equal, y_below;

initial begin
    #10;        /* a>b */
    a <= 1'b1;
    b <= 1'b0;
    #10;        /* a=b */
    a <= 1'b1;
    b <= 1'b1;
    #10;        /* a<b */
    a <= 1'b0;
```

```
        b <= 1'b1;
        #10;        /* 结束仿真 */
        $finish;
    end
    /* 例化模块 */
    comparator_1bit  comparator_1bit_inst (
        .a(a),
        .b(b),
        .y_above(y_above),
        .y_equal(y_equal),
        .y_below(y_below)
    );
endmodule
```

一位数值比较器的用于版上验证的约束文件如下所示, 其中约束文件中, 选择两个拨码开关分别作为输入信号 a, b, 选择三个 led 灯作为输出信号, 查看开发板原理图选择 SW7 和 SW6 分别作为 a,b, 将 LD22 LD20 作为三个输出信号, 通过原理图可以得知, LED 为高电平点亮, 与逻辑值保持一致。

Listing 3: 一位数值比较器约束文件

```
set_property PACKAGE_PIN P5 [get_ports a]
set_property IOSTANDARD LVCMOS18 [get_ports a]
set_property PACKAGE_PIN P4 [get_ports b]
set_property IOSTANDARD LVCMOS33 [get_ports b]
set_property PACKAGE_PIN J3 [get_ports y_above]
set_property IOSTANDARD LVCMOS33 [get_ports y_above]
set_property PACKAGE_PIN J2 [get_ports y_equal]
set_property IOSTANDARD LVCMOS33 [get_ports y_equal]
set_property PACKAGE_PIN K2 [get_ports y_below]
set_property IOSTANDARD LVCMOS33 [get_ports y_below]
```

四位数值比较器的源文件如下所示

Listing 4: 四位数值比较器源文件

```
module comparator_4bit(
    input [3:0] a, b,
    input i_above, i_equal, i_below,
    output y_above, y_equal, y_below
    );
    /* 相等时取上一级结果, 不相等则以当前级为输出 */
    assign y_above = a == b ? i_above : (a > b ? 1'b1 : 1'b0);
    /* 相等时, 输出与上级同步 */
```

```
    assign y_equal = a == b ? i_equal : 1'b0;
    /* 相等时取上一级结果，不相等则以当前级为输出 */
    assign y_below = a == b ? i_below : (a < b ? 1'b1 : 1'b0);
endmodule
```

四位数值比较器的仿真文件如下所示

Listing 5: 四位数值比较器仿真文件

```
module comparator_4bit_tb;
    reg [3:0] a, b;
    reg  i_above, i_equal, i_below;
    wire  y_above, y_equal, y_below;

    initial begin
        {i_above, i_equal, i_below} <= 3'b010;
        #10;        /* a>b */
        a <= 4'd12;
        b <= 4'd11;
        #10;        /* a=b */
        a <= 4'd13;
        b <= 4'd13;
        #10;        /* a<b */
        a <= 4'd8;
        b <= 4'd9;
        #10;        /* 结束仿真 */
        $finish;
    end
    /* 例化四位比较器 */
    comparator_4bit  comparator_4bit_inst (
        .a(a),
        .b(b),
        .i_above(i_above),
        .i_equal(i_equal),
        .i_below(i_below),
        .y_above(y_above),
        .y_equal(y_equal),
        .y_below(y_below)
    );
endmodule
```

四位数值比较器的硬件验证约束文件如下所示

Listing 6: 四位数值比较器的硬件验证约束文件

```
set_property IOSTANDARD LVCMOS33 [get_ports {a[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[1]}]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports {a[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[0]}]
set_property PACKAGE_PIN J3 [get_ports y_above]
set_property IOSTANDARD LVCMOS33 [get_ports y_above]
set_property PACKAGE_PIN J2 [get_ports y_equal]
set_property IOSTANDARD LVCMOS33 [get_ports y_equal]
set_property PACKAGE_PIN K2 [get_ports y_below]
set_property IOSTANDARD LVCMOS33 [get_ports y_below]

set_property PACKAGE_PIN P5 [get_ports {a[3]}]
set_property PACKAGE_PIN P4 [get_ports {a[2]}]
set_property PACKAGE_PIN P3 [get_ports {a[1]}]
set_property PACKAGE_PIN P2 [get_ports {a[0]}]

set_property PACKAGE_PIN R2 [get_ports {b[3]}]
set_property PACKAGE_PIN M4 [get_ports {b[2]}]
set_property PACKAGE_PIN N4 [get_ports {b[1]}]
set_property PACKAGE_PIN R1 [get_ports {b[0]}]

set_property PACKAGE_PIN U3 [get_ports i_above]
set_property PACKAGE_PIN U2 [get_ports i_equal]
set_property PACKAGE_PIN V2 [get_ports i_below]
set_property IOSTANDARD LVCMOS33 [get_ports i_above]
set_property IOSTANDARD LVCMOS33 [get_ports i_below]
set_property IOSTANDARD LVCMOS33 [get_ports i_equal]
```

提高任务

八位数值比较器采用两个四位数值比较器构成, 其源文件如下所示:

Listing 7: 八位数值比较器源文件

```
module comparator_8bit(
    input [7:0] a, b,
    input i_above, i_equal, i_below,
    output y_above, y_equal, y_below
    );

    /* 声明变量，建立模块间连线 */
    wire above, equal, below;

    /* 例化低位，第一级比较器 */
    comparator_4bit com0(
        .a(a[3:0]), .b(b[3:0]),
        .i_above(i_above),
        .i_equal(i_equal),
        .i_below(i_below),
```

```
        .y_above(above),
        .y_equal(equal),
        .y_below(below)
    );
    /* 例化高位，第二级比较器 */
    comparator_4bit com1(
        .a(a[7:4]), .b(b[7:4]),
        .i_above(above),
        .i_equal(equal),
        .i_below(below),
        .y_above(y_above),
        .y_equal(y_equal),
        .y_below(y_below)
    );
endmodule
```

Listing 8: 八位数值比较器 Top 文件

```
module comparator_8bit_top(
    input [7:0] a, b,
    output y_above, y_equal, y_below
    );
    /* 例化 */
    comparator_8bit  comparator_8bit_inst (
        .a(a), .b(b),
        .i_above(1'b0),
        .i_equal(1'b1),
        .i_below(1'b0),
        .y_above(y_above),
        .y_equal(y_equal),
        .y_below(y_below)
    );
endmodule
```

八位数值比较器的仿真文件如下:

Listing 9: 八位数值比较器仿真文件

```
module comparator_8bit_tb;
    reg [7:0] a, b;
    reg  i_above, i_equal, i_below;
    wire  y_above, y_equal, y_below;

    initial begin
        {i_above, i_equal, i_below} <= 3'b010;
        #10;        /* a>b */
        a <= 8'd12;
        b <= 8'd11;
        #10;        /* a<b */
        a <= 8'd8;
        b <= 8'd9;
        #10;        /* a=b */
```

```
        a <= 8'd13;
        b <= 8'd13;
        /* 测试级联输入端效果 */
        #10;        /* 上一级(低位)不等, 该级相等 */
        {i_above, i_equal, i_below} <= 3'b100;
        #10;
        $finish;
    end

    comparator_8bit  comp_8bit (
        .a(a),
        .b(b),
        .i_above(i_above),
        .i_equal(i_equal),
        .i_below(i_below),
        .y_above(y_above),
        .y_equal(y_equal),
        .y_below(y_below)
    );
endmodule
```

八位数值比较器的硬件验证约束文件如下所示

Listing 10: 八位数值比较器约束文件

```
set_property IOSTANDARD LVCMOS33 [get_ports {a[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {b[0]}]
set_property PACKAGE_PIN J3 [get_ports y_above]
set_property IOSTANDARD LVCMOS33 [get_ports y_above]
set_property PACKAGE_PIN J2 [get_ports y_equal]
set_property IOSTANDARD LVCMOS33 [get_ports y_equal]
set_property PACKAGE_PIN K2 [get_ports y_below]
set_property IOSTANDARD LVCMOS33 [get_ports y_below]

set_property PACKAGE_PIN U3 [get_ports i_above]
set_property PACKAGE_PIN U2 [get_ports i_equal]
set_property PACKAGE_PIN V2 [get_ports i_below]
set_property IOSTANDARD LVCMOS33 [get_ports i_above]
set_property IOSTANDARD LVCMOS33 [get_ports i_below]
set_property IOSTANDARD LVCMOS33 [get_ports i_equal]

set_property IOSTANDARD LVCMOS33 [get_ports {a[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {a[5]}]
```

```
26 set_property IOSTANDARD LVCMOS33 [get_ports {a[4]}]
27 set_property PACKAGE_PIN P5 [get_ports {a[7]}]
28 set_property PACKAGE_PIN P4 [get_ports {a[6]}]
29 set_property PACKAGE_PIN P3 [get_ports {a[5]}]
30 set_property PACKAGE_PIN P2 [get_ports {a[4]}]
31 set_property PACKAGE_PIN R2 [get_ports {a[3]}]
32 set_property PACKAGE_PIN M4 [get_ports {a[2]}]
33 set_property PACKAGE_PIN N4 [get_ports {a[1]}]
34 set_property PACKAGE_PIN R1 [get_ports {a[0]}]
35 set_property PACKAGE_PIN U3 [get_ports {b[7]}]
36 set_property PACKAGE_PIN U2 [get_ports {b[6]}]
37 set_property PACKAGE_PIN V2 [get_ports {b[5]}]
38 set_property PACKAGE_PIN V5 [get_ports {b[4]}]
39 set_property PACKAGE_PIN V4 [get_ports {b[3]}]
40 set_property PACKAGE_PIN R3 [get_ports {b[2]}]
41 set_property PACKAGE_PIN T3 [get_ports {b[1]}]
42 set_property PACKAGE_PIN T5 [get_ports {b[0]}]
43 set_property IOSTANDARD LVCMOS33 [get_ports {b[7]}]
44 set_property IOSTANDARD LVCMOS33 [get_ports {b[6]}]
45 set_property IOSTANDARD LVCMOS33 [get_ports {b[5]}]
46 set_property IOSTANDARD LVCMOS33 [get_ports {b[4]}]
```

拓展任务

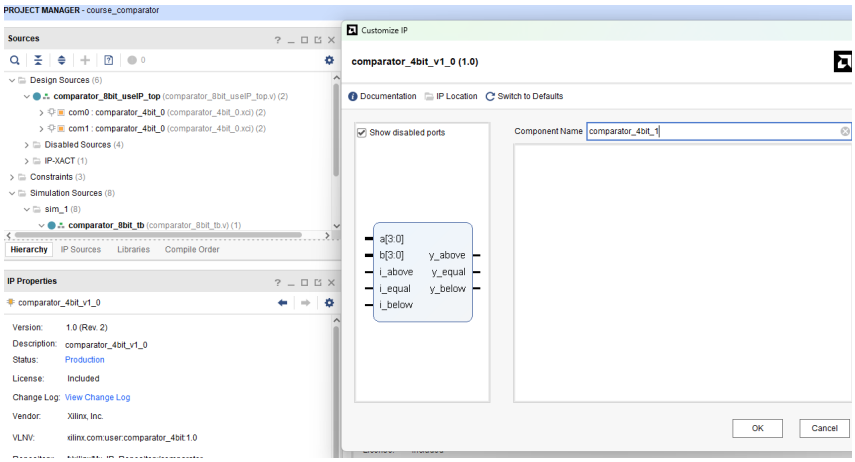

图 2: 调用四位比较器 IP 核

通过 IP 核封装的方式进行 8 位数值比较器的设计, 调用两个 IP 核结果如图 2所示, 使用两个四位比较器。

使用 IP 核的八位数值比较器的源文件 Top 如下

Listing 11: 使用 IP 核的八位数值比较器的源文件

```
1 module comparator_8bit_useIP_top(
2     input [7:0] a, b,
```

```
    output y_above, y_equal, y_below
    );

    wire above, equal, below;
    /* 为IP核配置连接方式，该四位比较器为第一级 */
    comparator_4bit_0 com0 (
        .a(a[3:0]), .b(b[3:0]),
        .i_above(1'b0),
        .i_equal(1'b1),
        .i_below(1'b0),
        .y_above(above),
        .y_equal(equal),
        .y_below(below)
    );
    /* 为IP核配置连接方式，该四位比较器为第二级 */
    comparator_4bit_0 com1 (
        .a(a[7:4]), .b(b[7:4]),
        .i_above(above),
        .i_equal(equal),
        .i_below(below),
        .y_above(y_above),
        .y_equal(y_equal),
        .y_below(y_below)
    );
endmodule
```

Listing 12: 使用 IP 核的八位数值比较器的仿真文件

```
module comparator_8bit_tb;
    reg [7:0] a, b;
    wire  y_above, y_equal, y_below;

    initial begin
        a <= 8'd44;
        b <= 8'd32;
        #10;        /* a<b */
        a <= 8'd111;
        b <= 8'd192;
        #10;        /* a=b */
        a <= 8'd127;
        b <= 8'd127;
        /* 测试级联输入端效果 */
        #10;        /* 上一级(低位)不等，该级相等 */
        $finish;
    end

    comparator_8bit_useIP_top  comp_8bit (
        .a(a),
        .b(b),
        .y_above(y_above),
        .y_equal(y_equal),
        .y_below(y_below)
```

```
25      );
26  endmodule
```

约束文件与直接例化使用两个四位数值比较器一致, 不再重复

三、仿真结果

基础任务

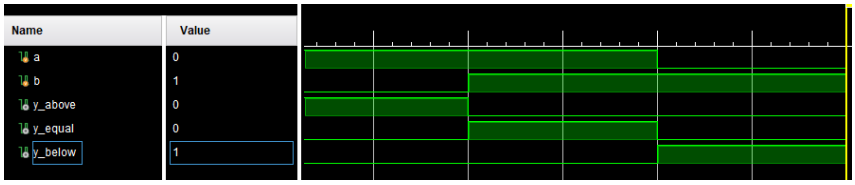

图 3: 一位数值比较器仿真波形

一位数值比较器的行为仿真结果如图 3所示。根据仿真波形图可以看到比较结果完全正确, 当 a=1'b1,b=1'b0 时输出位 a>b, 第二状态, 当 a=1'b1, b=1'b1 时输出为 y_equal, 表示二者相等, 当 a = 1'b0, b=1'b1 时输出 y_below, 表示 a<b。

图 4: 四位数值比较器仿真结果

四位数值比较器的行为仿真结果如图 4所示, 根据仿真波形图可以看到比较结果完全正确, 首先可以看到上一级输入将相等置位 1, 其余置位 0, 则该模块的输出只受到当前级别的输出, 测试了三个状态的输入, 当 a=12, b=11 时, 可以正确的检测为 a>b, a=13, b=13 时, 可以正确的检测为 a=b, 当 a=8, b=9 时可以正确的检测到 a<b。

提高任务

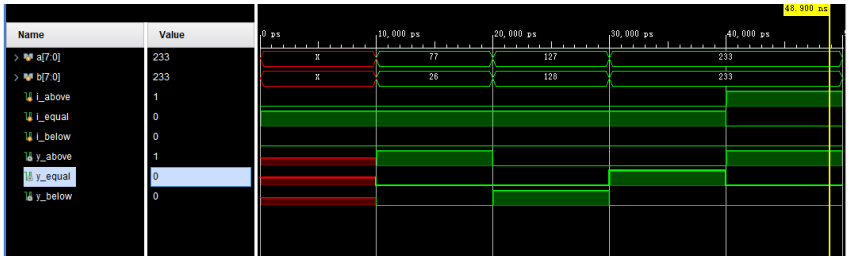

图 5: 八位数值比较器仿真结果

八位数值比较器的行为仿真结果如图 5所示, 与之前不同的时, 这次对级联输入端进行了测试, 级联输入端能够正常工作, 当本级相等时, 输出结果由更低位决定, 即上一级, 最后一个状态中, 本级比较结果相等, 上一级结果是大于, 所以输出大于, 正确工作。

拓展任务

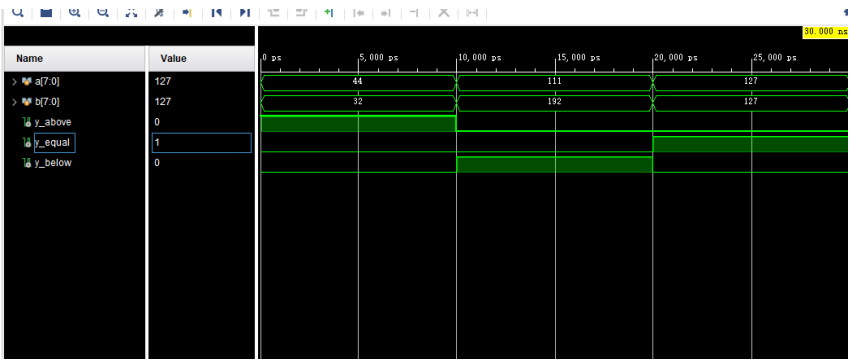

图 6: 使用四位比较器 IP 核搭建的八位数值比较器仿真结果

使用 IP 核搭建的八位数值比较器仿真结果如图 6所示, 与直接使用两个四位数值比较器例化的形式相同, 可以正确工作。

四、硬件验证结果

基础任务

一位数值比较器的硬件验证通过两个拨码开关和三个 LED 完成, 选择 SW7 和 SW6 分别作为 a,b, 将 LD2_2, LD2_1, LD2_0 作为三个输出信号, 具体连接配置可见相应的 io 约束文件。

图 7: 一位数值比较器硬件验证结果

一位数值比较器硬件验证结果如图 7所示, 输出结果对应于输入, 当 a=0, b=0 或是 a=1, b=1 时, 模块能够正确识别为相等, 当 a=0, b=1 时, 可以正确输出 a 小于 b 的信号, 相反同理。

四位数值比较器的硬件验证与一位数值比较器的输出采用相同的三个 LED 作为输出信号, 将 8 个拨码开关作为两个 4 位二进制输入, SW7-SW4 作为 a, SW3-SW0 作为 b。

图 8: 四位数值比较器硬件验证结果

四位数值比较器硬件验证结果如图 8所示, 级联输入端为 010, 不使用上一级, 默认将上一级置为相等, 对应于实际比较中低位相等, 高位的比较结果只与高位有关。四次比较结果完全正确。

提高任务

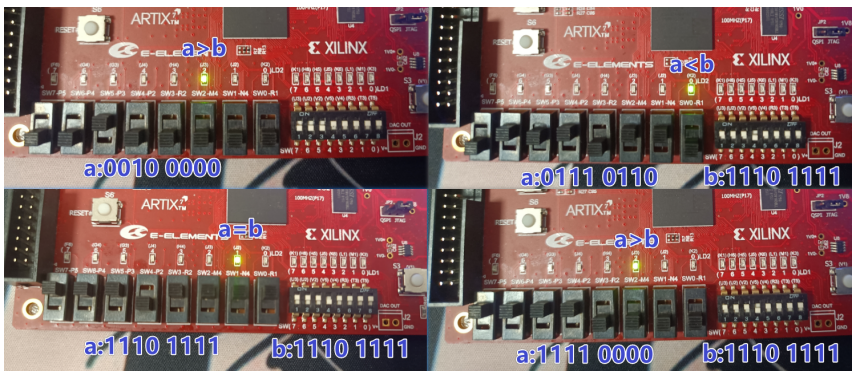

图 9: 八位数值比较器硬件验证结果

四位数值比较器硬件验证结果如图 9所示, 使用 top 文件重新例化八位比较器模块, 将级联输入端配置为相等, 实现板上 8 位数值比较器, 通过拨码开关和八位 DIP 作为数值输入, 对应于实际比较中低位相等, 高位的比较结果只与高位有关。四次比较结果完全正确, 通过逐位相等判断, 判断 IO 约束结果无误。

拓展任务

相比于提高部分的数值比较器, 使用 IP 核仅仅是改变了模块的构建方式, 对外功能和接口保持一致, 硬件映射相同。同时二者的代码 (硬件结构) 本质上是一致的, 所以不进行过多测试, 某一次测试结果如图所示。

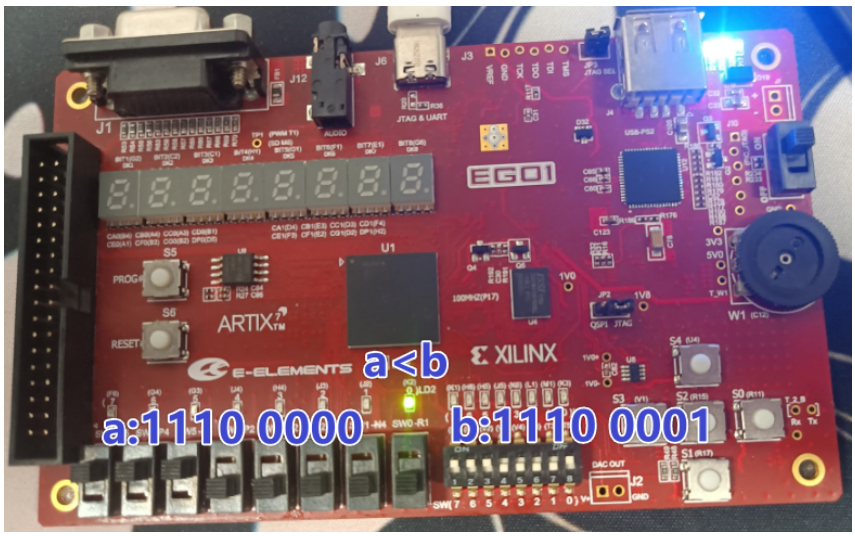

图 10: 使用 IP 核构建的八位比较器

五、问题解决

调用封装好的 IP 核后, 综合报错

解决: 查看报错信息, 报错信息指向了对调用 IP 核接口配置处, 经过查阅相关资料, 由于自己对 IP 核调用的错误使用导致。简单的 IP 核调用后是直接把对应的源文件导入工程下, 误以为是直接生成对应的模块, 而未进行例化直接使用导致报错。

使用 VS Code 作为 Vivado 的编辑器导致 Vivado 的仿真报错无法进行

解决: 对现象观察为, 通过 Vivado 配置编辑器更改后, Vivado 默认打开 VS Code 编写 testbench 时, 直接在 Vivado 中进行仿真会无法进行, 会提示文件已被占用, 针对 Vivado 的仿真问题有几种方法

1. 关闭 Vivado 自动打开的 VS Code 界面, 就可以解除占用状态, 进行仿真。
2. 手动建立 VS Code 窗口, 手动配置工作区到 Vivado 生成的 testbench 进行编辑, 此时可以在编辑 testbench 的同时进行 Vivado 界面的仿真。

3. 可以通过 VS Code 调用第三方仿真工具 modelsim 完成对 Verilog 文件的仿真。

六、心得体会

以较为完整的流程, 包括原理梳理、报告撰写即程序编写和调试完成一次 Verilog 功能实现, 熟练了对 Vivado 的使用, 并通过拓展任务部分的 IP 核设计任务, 熟悉了 Vivado 的 IP 封装工具, 并同时了解了一下 Vivado 提供的 IP 核, 如用于调试的 ILA 核、常用的时钟核以及 FIFO 核等。为之后对这些 IP 核的使用提供了一定的基础。也对 EGO1 上的外设资源有了一定的了解, 比如和信号处理相关的 DSP 模块, 板上 FPGA 芯片还集成了两个 12bit 位宽、采样率为 1MSPS 的 XADC, 可以通过 Xilinx 提供的 IP 核直接调用使用。

通过完成一个基础组合逻辑电路的实现, 以完全不同的形式实现了在数电课程中, 需要经过真值表、卡诺图化简、避免竞争和冒险等复杂过程才能设计得到的比较器, 相比之下 Verilog 实现相同的功能仅需要描述其实现思路, 剩下的底层门电路的应用全部由计算机计算得到, 也就是综合过程, 感受了 EDA 设计的强大之处。