杭州電子科技大学

计算机组成原理（甲）

实 验 报 告

学　　院	网络空间安全学院
专　　业	网络工程
班　　级	192702401
学　　号	19061440
学生姓名	F001
教师姓名	袁理峰
完成日期	2020.11.20
成　　绩	

实验三 多功能 ALU 设计实验 （实验名称）

一、 实验目的

（1） 学习多功能 ALU 的工作原理，掌握运算器的设计方法。

（2） 掌握运用 Verilog HDL 进行行为描述与建模的技巧和方法

二、 实验原理

本实验设计一个具有 8 种运算功能的 32 位 ALU，并能够产生运算结果的标志：结果为零标志 ZF 和溢出标志 OF。ALU 的控制线的功能表如下一一对应

000 按位与运算

001 按位或运算

010 按位异或运算

011 按位或非运算

100 算术加运算

101 算术减运算

110 若 A<B，则输出 1；否则输出 0

111 B 逻辑左移 A 所指定的数

根据 ZF 的含义可以得到 ZF 的逻辑表达式为：

ZF=(F31+F32+F30+……F1+F0)的非

根据 OF 的含义可以得到 OF 的逻辑表达式为

OF=C32 异或 C31

OF=A31 异或 B31 异或 F31 异或 C32

通过顶层测试模块，调用基本的 ALU 模块，从而进行相关的调试

三、 实验环境

所用电脑的软硬件配置：

实验所用的软件：

四、 主要操作步骤及实验结果记录（不能光截图，要有相应的文字说明）

（对实验过程中的主要操作步骤进行描述，并随时记录实验过程中观察到的结果，必要时可辅助截图）

任务一：在 Xilinx ISE 中创建工程，编辑程序源代码，然后编译、综合

```
//////////////////////////////////////////////////////////////////////////////////
module ALU_8F_EXP(
    input [2:0] ALU_OP,
    input [2:0] AB_SW,
    input [2:0] F_LED_SW,
    output reg [7:0] LED
    );

    reg [31:0] A,B;
    wire [31:0] F;
    wire ZF,OF;

    ALU_8F ALU_UUT(A,B,ALU_OP,F,ZF,OF);

    always @(*)
    begin
      case (AB_SW)      //use 2 bit switch to choose special 8 data test
         3'b000:  begin A = 32'h0000_0000;   B = 32'hFFFF_FFFF;   end
         3'b001:  begin A = 32'h0000_0003;   B = 32'h0000_0607;   end
         3'b010:  begin A = 32'h8000_0000;   B = 32'h8000_0000;   end
         3'b011:  begin A = 32'h7FFF_FFFF;   B = 32'h7FFF_FFFF;   end
         3'b100:  begin A = 32'hFFFF_FFFF;   B = 32'hFFFF_FFFF;   end
         3'b101:  begin A = 32'h8000_0000;   B = 32'hFFFF_FFFF;   end
         3'b110:  begin A = 32'hFFFF_FFFF;   B = 32'h8000_0000;   end
         3'b111:  begin A = 32'h1234_5678;   B = 32'h3333_2222;   end
         default: begin A = 32'h9ABC_DEF0;   B = 32'h1111_2222;   end
      endcase
    end

    always @(*)
    begin
    /*
    use 2 bit switch--F_LED_SW[1:0]--and a botton--F_LED_SW[2]--choose show
    32bit caculate answer's 4 bytes(if button is released F_LED_SW[2]==0) or
    show the statu of the answer ZF and OF(if button is pressed F_LED_SW[2]==1)
    */
      case (F_LED_SW)
         3'b000:  LED = F[7:0];
         3'b001:  LED = F[15:8];
         3'b010:  LED = F[23:16];
         3'b011:  LED = F[31:24];
         default: begin LED[7] = ZF; LED[0] = OF;  LED[6:1] = 6'b0;  end
      endcase
    end

endmodule

```

以上为顶层组件的代码

输入信号主要为 ALU_OP，负责运算功能的选择

AB_SW，负责指定 8 组测试数据之一

F_LED_SW 负责显示运算结果的四个字节，或者是标志 ZF 和 OF，主要由 F_LED_SW[2]决定

```
//
//////////////////////////////////////////////////////////////////////////////////
module ALU_8F(
    input [31:0] A,
    input [31:0] B,
    input [2:0] ALU_OP,
    output reg [31:0] F,
    output ZF,
    output OF
    );

   parameter Zero_32 = 32'h0000_0000, One_32 = 32'h0000_0001;
   reg C32;

   always @(*)
   begin
      C32 = 1'b0;
      case (ALU_OP)
         3'b000:  F = A & B;
         3'b001:  F = A | B;
         3'b010:  F = A ^ B;
         3'b011:  F = ~(A | B);
         3'b100:  {C32,F} = A + B;
         3'b101:  {C32,F} = A - B;
         3'b110:  F = (A < B) ? One_32 : Zero_32;
         3'b111:  F = B >> A;
         default: F = Zero_32;
      endcase
   end
   assign ZF = (F == Zero_32) ? 1'b1 : 1'b0;
   assign OF = ((ALU_OP==3'b100)||(ALU_OP==3'b101))&&(A[31] ^ B[31] ^ F[31] ^ C32);

endmodule

```

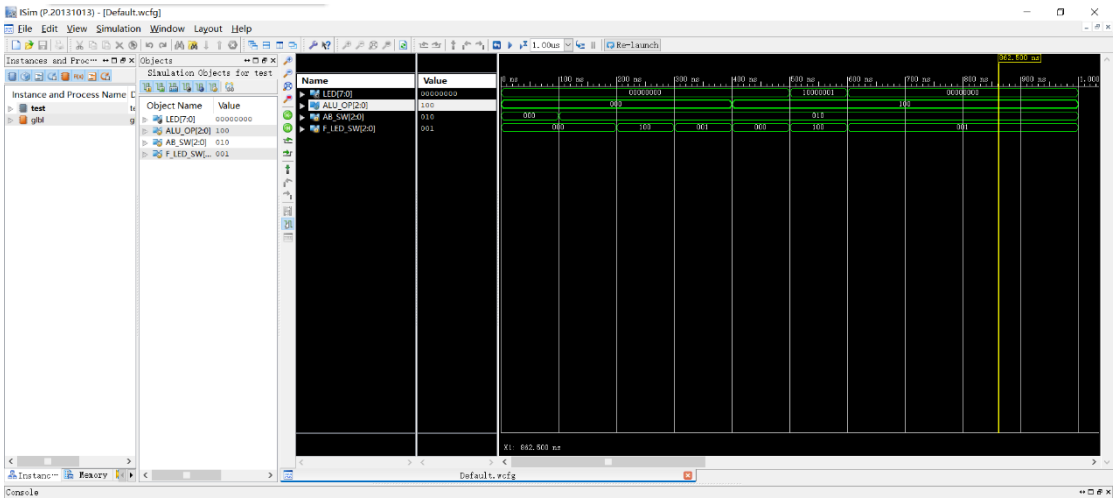
以上为底层 ALU 的代码

所有代码均通过相关语法检查

任务二：编写激励代码，观察仿真波形，验证逻辑是否有误

```
41      );
42
43      initial begin
44          // Initialize Inputs
45          ALU_OP = 0;
46          AB_SW = 0;
47          F_LED_SW = 0;
48
49          // Wait 100 ns for global reset to finish
50          #100;
51
52          // Add stimulus here
53          ALU_OP=000; AB_SW=010; F_LED_SW=000;
54          #100;
55
56          ALU_OP=000; AB_SW=010; F_LED_SW=100;
57          #100;
58
59          ALU_OP=000; AB_SW=010; F_LED_SW=001;
60          #100;
61
62          ALU_OP=100; AB_SW=010; F_LED_SW=000;
63          #100;
64
65          ALU_OP=100; AB_SW=010; F_LED_SW=100;
66          #100;
67
68          ALU_OP=100; AB_SW=010; F_LED_SW=001;
69          #100;
70      end
71
72  endmodule
73
74
```

激励代码如上所示



仿真波形如上图所示

为了验证方便，直接选用了相同的数据，并且更改了运算功能进行实现，在输出中，也选择了显示运算结果的字节或 ZF 及 OF，均有了正确的验证

五、 实验分析总结及心得

（结合所学知识对实验过程中观察到的实验结果进行分析总结，以便加深对知识的理解，并总结通过实验学到的知识或技术）

随着试验次数越来越多，遇到的问题也越来越少，倒是实验本身给我的启发着实越来越多。

通过查阅相关资料，我认为该 ALU 不能够实现 MIPS 核心指令集的所有指令。解决这个问题的方法并不能通过一一对应来解释，因为我并没有找到 MIPS 的核心指令集，因此难以通过一个个对应的方式进行解决。我才用的方式是研究 MIPS 的特点，经过比较后我认为该 ALU 与 MIPS 的特点基本一致，因此认为是可以实现大部分指令的。但是对于转移指令类的操作，该 ALU 是不支持的。因为 MIPS 分为为 3 个指令类：存储器访问指令类、算术逻辑指令类、转移指令类。而转移指令类该 ALU 无法支持。

SF 符号标志即为结果的最高位，SF=F[31]

PF 就标志可以通过对各个位数相加后的结果对 2 取余的方式实现。PF=((F[31]+F[30]+……+F[1]+F[0])%2+1)%2

CF 进位/借位标记，加法时，若 C32=1,则 CF=1;若 C32=0,则 CF=0;减法时，若 C32=1,则 CF=0;若 C32=0,则 CF=1。

算术右移将操作数右移 s 位，并且在左边空出来的位置补 s 位操作数的符号，对于算术右移操作来说，将二进制的数值左移 n 位等同于将原来的数值除 2 n 2^n2n；逻辑右移将操作数右移 s 位，并且在左边空出来的位置补 s 位 0，忽略操作数的符号。例如，要对 4 位二进制数 1100(4′b1100)执行右移一位的操作。执行逻辑右移运算得到的结果为 0110，而执行算术右移就是运算得到的结果为 1110。在 Verilog 语言中，操作符>>>执行算术右移操作，而操作符>>执行逻辑右移操作。