**计算机组成与设计实验报告**

**实验4·ALU和乘法器的实现**

学 院： 网络空间安全学院（研究院）

组 员： 孟庆丹、牛中原

***实验四.ALU和乘法器的实现***

*Prime decomposition*

成员分工：

孟庆丹：编写实验代码，完成实验报告

牛中原：测试仿真结果，完成实验报告

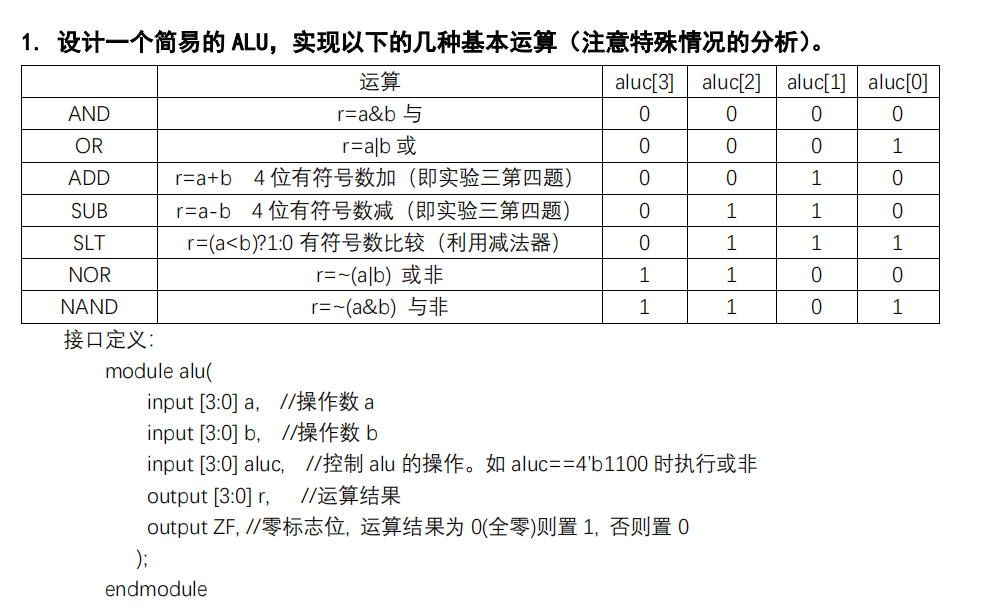
算术逻辑单元（arithmetic logic unit，缩写ALU）是进行整数运算的结构。现阶段是用电路来实现，应用在电脑芯片中。

在计算机中，算术逻辑单元（ALU）是专门执行算术和逻辑运算的数字电路。ALU是计算机中央处理器的最重要组成部分，甚至连最小的微处理器也包含ALU作计数功能。

乘法器（multiplier）是一种完成两个互不相关的模拟信号相乘作用的电子器件。它可以将两个二进制数相乘，它是由更基本的加法器组成的。乘法器可以通过使用一系列计算机算数技术来实现。

本次实验的目的是复习 ALU、乘法器的基本实现原理，使用 verilog 实现 ALU 和乘法器，熟练使用 modelsim工具进行仿真和调试，学习掌握实例化模块的方式

1. ALU设计



使用实际上的组合逻辑通过一系列case判断所有情况就可以实现ALU,加减法例化之前实现的adder模块和subber模块

代码实现：

module alu(

 input [3:0] a, //操作数 a

 input [3:0] b, //操作数 b

 input [3:0] aluc,

 output reg[3:0] r, //运算结果

 output reg ZF //零标志位, 运算结果为 0(全零)则置 1, 否则置 0

 );

wire[3:0]r1;

wire[3:0]r2;

wire OF1,OF2;

add\_sub addr(a,b,0,0,r1,OF1);

add\_sub subr(a,b,0,1,r2,OF2);

always@(\*)

begin

    case(aluc)

    4'b0010:begin

        r = r1;

    end

    4'b0110:begin

        r = r2;

    end

    4'b0000:begin

        r = a&b;

    end

    4'b0001:begin

        r = a|b;

    end

    4'b1101:begin

        r = ~(a&b);

    end

    4'b1100:begin

        r = ~(a|b);

    end

    4'b0111:begin

        if(r2[3]==1)

            r=4'b0001;

        else

            r=4'b0000;

    end

    endcase

    if(r==0)

        ZF=1;

    else

        ZF=0;

end

endmodule

约束文件：

set\_pin\_assignment { a[0] } { LOCATION = T6; }

set\_pin\_assignment { a[1] } { LOCATION = T5; }

set\_pin\_assignment { a[2] } { LOCATION = R5; }

set\_pin\_assignment { a[3] } { LOCATION = T4; }

set\_pin\_assignment { b[0] } { LOCATION = P8; }

set\_pin\_assignment { b[1] } { LOCATION = N6; }

set\_pin\_assignment { b[2] } { LOCATION = P6; }

set\_pin\_assignment { b[3] } { LOCATION = M6; }

set\_pin\_assignment { r[0] } { LOCATION = P5; }

set\_pin\_assignment { r[1] } { LOCATION = N5; }

set\_pin\_assignment { r[2] } { LOCATION = P4; }

set\_pin\_assignment { r[3] } { LOCATION = M5; }

set\_pin\_assignment { ZF } { LOCATION = N4; }

set\_pin\_assignment { aluc[0] } { LOCATION = N9; }

set\_pin\_assignment { aluc[1] } { LOCATION = P9; }

set\_pin\_assignment { aluc[2] } { LOCATION = R9; }

set\_pin\_assignment { aluc[3] } { LOCATION = N8; }

1. 乘法器设计

首先计算出结果的符号，将乘数与被乘数全部化为正数，用依次左移被乘数的方式实现乘法操作。

代码实现：

module mult4(

 input [3:0] a,

 input [3:0] b,

 output reg[7:0] z , // 乘积输出 z

 output reg ZF

 );

reg [3:0]r;

reg [7:0]a1;

reg [3:0]\_a;

reg [3:0]a2;

reg [3:0]b2;

always@(\*)

begin

    a2=a;

    b2=b;

    if(a[3])

        a2=~a+1;

    if(b[3])

        b2=~b+1;

    a1={0,0,0,0,0,a2[2],a2[1],a2[0]};

    z=0;

    if(b2[0])

        z=z+(a1);

    if(b2[1])

        z=z+(a1<<1);

    if(b2[2])

        z=z+(a1<<2);

    if(a[3]^b[3])//符号不同

        z=~z+1;

    if(z==0)

        ZF=1;

    else

        ZF=0;

end

endmodule

约束文件：

set\_pin\_assignment { a[0] } { LOCATION = T6; }

set\_pin\_assignment { a[1] } { LOCATION = T5; }

set\_pin\_assignment { a[2] } { LOCATION = R5; }

set\_pin\_assignment { a[3] } { LOCATION = T4; }

set\_pin\_assignment { b[0] } { LOCATION = P8; }

set\_pin\_assignment { b[1] } { LOCATION = N6; }

set\_pin\_assignment { b[2] } { LOCATION = P6; }

set\_pin\_assignment { b[3] } { LOCATION = M6; }

set\_pin\_assignment { z[0] } { LOCATION = P5; }

set\_pin\_assignment { z[1] } { LOCATION = N5; }

set\_pin\_assignment { z[2] } { LOCATION = P4; }

set\_pin\_assignment { z[3] } { LOCATION = M5; }

set\_pin\_assignment { z[4] } { LOCATION = N4; }

set\_pin\_assignment { z[5] } { LOCATION = N3; }

set\_pin\_assignment { z[6] } { LOCATION = M4; }

set\_pin\_assignment { z[7] } { LOCATION = M3; }

实验总结：

完成了ALU与乘法器，为以后的实验打下了基础。