实验六　ALU

实验介绍

通过本次实验，你可以了解到 ALU 的实现原理，并学习如何实现一个 ALU

实验目标

1. 了解到 ALU 的原理
2. 使用 Verilog 实现一个 ALU

实验原理

接口定义，可以直接复制到文件中

（请务必按照接口定义编写代码，在将来的实验中也是如此，模块名也请按照给出的定义命名）

```
module alu(
        input [31:0]   a,          //  32 位输入，操作数 1
        input [31:0]   b,          //  32 位输入，操作数 2
        input [3:0]    aluc,       //  4 位输入，控制 alu 的操作
        output[31:0]   r,          //  32 位输出，有 a b 经过 aluc 指定的操作生成
        output         zero,       //  0 标志位
        output         carry,      //  进位标志位
        output         negative,   //  负数标志位
        output         overflow    //  溢出标志位
    );
```

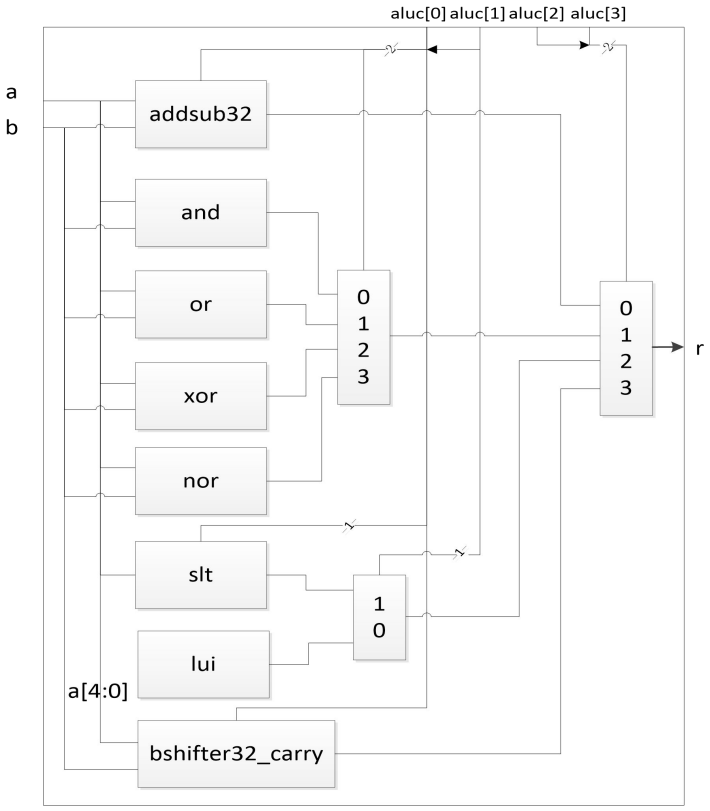
ALU 是负责运算的电路。ALU 必须实现以下几个运算：加（ADD）、减（SUB）、与（AND）、或（OR）、异或（XOR）、置高位立即数（LUI）、逻辑左移与算数左移（SLL）、逻辑右移（SRL）以及算数右移（SRA）、SLT、SLTU 等操作。输出 32 位计算结果、carry(借位进位标志位)、zero(零

标志位)、negative(负数标志位)和 overflow(溢出标志位)。

此次实现 ALU 的基本思想是：在操作数输入之后将所有可能的结果都计算出来，通过操作符 aluc 的输入来判别需要执行的操作来选择需要的结果进行输出。

	Aluc[3]	Aluc[2]	Aluc[1]	Aluc[0]
Addu r=a+b 无符号	0	0	0	0
Add r=a+b 有符号	0	0	1	0
Subu r=a-b 无符号	0	0	0	1
Sub r=a-b 有符号	0	0	1	1
And r=a & b	0	1	0	0
Or r=a | b	0	1	0	1
Xor r=a ^ b	0	1	1	0
Nor r=~（a | b）	0	1	1	1
lui r = {b[15:0],16'b0}	1	0	0	X
Slt r=(a<b)?1:0 有符号	1	0	1	1
Sltu r=(a<b)?1:0 有符号	1	0	1	0
Sra r=b>>>a	1	1	0	0
Sll/Slr r=b<<a	1	1	1	X
Srl r=b>>a	1	1	0	1

图 7.1 为 ALU 的原理图供参考



addsub32 标志位规则（根据 ARM 文档[1]）

zero 标志位	1.Z＝1 表示运算结果是 0，Z＝0 表示运算结果不是零
	2.对于 CMP 指令，Z＝1 被用来表示进行比较的两个数大小相等(此功能在 cpu 中实现，无需在 alu 中实现)
	3.所有运算均影响此标志位
carry 标志位	1.加法，无符号数运算发生上溢出
	2.减法，无符号数发生下溢出
	3.包含移位的非加减法，最后一次被移出的位的数值(在移位模块实现)
	4.有符号加减法，逻辑与、或、异或、或非不影响此标志位
negative 标志位	1.当两个补码表示的有符号整数运算时，N＝1 表示运算的结果为负数，N＝0 表示结果为正数或零
	2.所有运算均影响此标志位
overflow 标志位	1.对于加/减法运算指令，当操作数和运算结果为二进制的补码表示的带符号数时，V＝1 表示符号位溢出
	2.只有有符号加减法影响此标志位

行为级建模

本次实验允许使用行为级建模方式实现 ALU，例如可以使用“+”“-”“<<”“ >>”“>>>”等运算符号实现 ALU 中的计算模块

实验步骤

1、 新建工程
2、 将以前实验课上已经实现的选择器、加法器和移位器加入工程
3、 修改或扩展这些已有的模块编写 ALU 所需的模块
4、 测试 ALU 下的每个模块
5、 将这些模块组成一个 ALU
6、 使用 Modelsim 进行仿真，验证 ALU 的正确性