同济大学计算机系

数字逻辑课程实验报告

学　　号 **2151140**

姓　　名 王谦

专　　业 信息安全

授课老师 郭玉臣

一、实验内容

在本次实验中，我们将使用 Verilog HDL 语言实现行为级 ALU 的设计和仿真。

二、硬件逻辑图

（实验步骤中要求用 logisim 画图的实验，在该部分给出 logisim 原理图，否则该部分在实验报告中不用写）

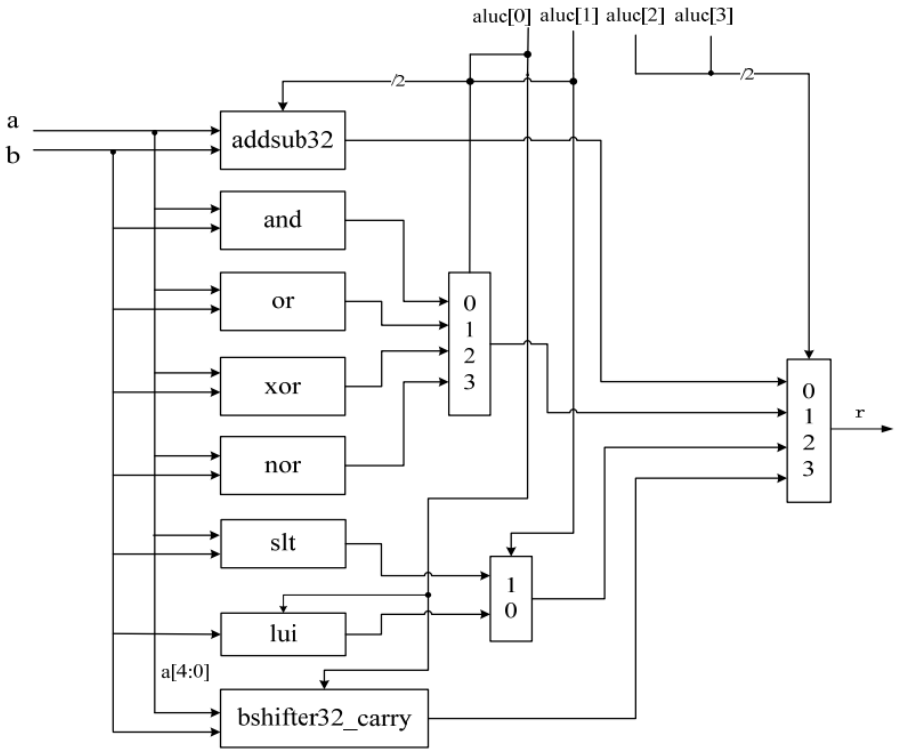


图 6.9.1 ALU 的原理图供参考

三、模块建模

（该部分要求对实验中建模的所有模块进行功能描述，并列出各模块建模的 verilog 代码）

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）。

本实验实现 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。图 6.9.1 所示为本实验的 ALU 参考原理图。表 6.9.1 所示为 aluc 的值所对应的运算。表 6.9.2 所示为 addsub32 标志位规则（仅供参考）。

表 6.9.1　aluc 的值所对应的运算

	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 无符号	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

表 6.9.2　ALU 标志位规则

zero 标志位	1.Z＝1 表示运算结果是零，Z＝0 表示运算结果不是零。 2.对于 Slt 和 Sltu 运算，如 a-b=0，则 Z＝1，表示进行比较的两个数大小相等。 3.所有运算均影响此标志位。
carry 标志位	1.无符号数加法运算（Addu）发生上溢出，则该标志位为 1。 2.无符号数减法运算（Subu）发生下溢出，则该标志位为 1。 3.无符号数比较运算（Sltu），如 a-b<0，则该标志位为 1。 4.移位运算，该标志位为最后一次被移出的位的数值（在移位模块实现）。 5.其他运算不影响此标志位。
negative 标志位	1.有符号数运算 Add 和 Sub，操作数和运算结果均采用二进制补码的形式表示，N＝1 表示运算的结果为负数，N＝0 表示结果为正数或零。 2.有符号数比较运算（Slt），如果 a-b<0，则 N=1。 3.其他运算，运算最终结果的最高位 r[31]为 1，则 N=1。
overflow 标志位	1.对于有符号加减法运算（Add 和 Sub），操作数和运算结果均采用二进制补码的形式表示，有溢出时该标志位 o=1。 2.只有有符号加减法运算影响此标志位。

```
module alu(a,b,aluc,r,zero,carry,negative,overflow);
input [31:0] a,b;
input [3:0] aluc;
output reg [31:0] r;
output reg zero,carry,negative,overflow;

always@(*)
begin
	case(aluc)
		//add
		4'b0010:
			begin
			r=a+b;
			overflow=((a[31]==b[31])&&(~r[31]==a[31]))?1:0;
			zero=(r==0)?1:0;
			carry=0;
			negative=(r<0)?1:0;
			end
		//addu
		4'b0000:
			begin
			{carry,r}=a+b;
			zero=(r==0)?1:0;
			overflow=0;
			negative=(r[31]==1)?1:0;
			end
		//sub
		4'b0011:
			begin
			r=a-b;
			overflow=((a[31]==0 && b[31]==1 && r[31]==1) || (a[31]==1 &&
b[31]==0 && r[31]==0))?1:0;
			zero=(a==b)?1:0;
			carry=0;
			negative=(r<0)?1:0;
			end
		//subu
		4'b0001:
			begin
			{carry,r}=a-b;
			zero=(r==0)?1:0;
			overflow=0;
			negative=(r[31]==1)?1:0;
			end
```

```
//and
4'b0100:
	begin
	r=a&b;
	zero=(r==0)?1:0;
	carry=0;
	overflow=0;
	negative=(r[31]==1)?1:0;
	end
//or
4'b0101:
	begin
	r=a|b;
	zero=(r==0)?1:0;
	carry=0;
	overflow=0;
	negative=(r[31]==1)?1:0;
	end
//xor
4'b0110:
	begin
	r=a^b;
	zero=(r==0)?1:0;
	carry=0;
	overflow=0;
	negative=(r[31]==1)?1:0;
	end
//nor
4'b0111:
	begin
	r=~(a|b);
	zero=(r==0)?1:0;
	carry=0;
	overflow=0;
	negative=(r[31]==1)?1:0;
	end
//slt
11'b1011:
	begin
	if(a[31]==1 && b[31]==0)
		r=1;
	else if(a[31]==0 && b[31]==1)
		r=0;
	else
```

```
            r=(a<b)?1:0;
        overflow=r;
        zero=(r==0)?1:0;
        carry=0;
        negative=(a<b)?1:0;
        end
    //sltu
    4'b1010:
        begin
        r=(a<b)?1:0;
        carry=r;
        zero=(r==0)?1:0;
        overflow=0;
        negative=(r[31]==1)?1:0;
        end
    //sll/slr
    4'b111x:
        begin
        {carry,r}=b<<a;
        overflow=0;
        zero=(r==0)?1:0;
        negative=(r[31]==1)?1:0;
        end
    //srl
    4'b1101:
        begin
        r=b>>a;
        carry=b[a-1];
        overflow=0;
        zero=(r==0)?1:0;
        negative=(r[31]==1)?1:0;
        end
    //sra
    4'b1100:
        begin
        r=($signed(b))>>>a;
        carry=b[a];
        overflow=0;
        zero=(r==0)?1:0;
        negative=(r[31]==1)?1:0;
        end
    //lui
    4'b100x:
        begin
```

```
                r={b[15:0],16'b0};
                carry=0;
                overflow=0;
                zero=(r==0)?1:0;
                negative=(r[31]==1)?1:0;
                end
        endcase
end
endmodule
```

四、测试模块建模

（要求列写各建模模块的 test bench 模块代码）

```
`timescale 1ns / 1ns
module alu_tb;

        reg [3:0] aluc;
        reg [31:0] a,b;
        wire [31:0] r;
        wire carry,overflow,zero,negative;
        alu alu_init(a,b,aluc,r,zero,carry,negative,overflow);
        initial
        begin
                //add
                aluc=4'b0010;
                a=32'h4321fedc;
                b=32'h9321fedc;
                #20
                a=32'hf321fedc;
                b=32'hf321fedc;
                #20
                //addu
                aluc=4'b0000;
                a=32'h4321fedc;
                b=32'h9321fedc;
                #20
                a=32'hf321fedc;
                b=32'hf321fedc;
                #20
                //sub
                aluc=4'b0011;
                a=32'h4321fedc;
                b=32'h9321fedc;
```

```
#20
a=32'hf321fedc;
b=32'ha321fedc;
#20
//subu
aluc=4'b0001;
a=32'h4321fedc;
b=32'h9321fedc;
#20
a=32'hf321fedc;
b=32'ha321fedc;
#20
//and
aluc=4'b0100;
a=32'h4320fed0;
b=32'h4020f0d0;
#20
a=32'h11111111;
b=32'h11111111;
#20
a=32'h11111111;
b=32'h00000000;
#20
//or
aluc=4'b0101;
a=32'h4320fed0;
b=32'h4020f0d0;
#20
a=32'h11111111;
b=32'h00000000;
#20
a=32'h00000000;
b=32'h00000000;
#20
//xor
aluc=4'b0110;
a=32'h4320fed0;
b=32'h4020f0d0;
#20
a=32'h11111111;
b=32'h00000000;
#20
a=32'h11111111;
b=32'h11111111;
```

```
#20
//nor
aluc=4'b0111;
a=32'h4320fed0;
b=32'h4020f0d0;
#20
a=32'h11111111;
b=32'h00000000;
#20
a=32'h11111111;
b=32'h11111111;
#20
//slt
aluc=4'b1011;
a=32'h4320fed0;
b=32'h4020f0d0;
#20
a=32'hf0000000;
b=32'h00000000;
#20
a=32'h00000000;
b=32'h0f000000;
#20
//sltu
aluc=4'b1010;
a=32'h4320fed0;
b=32'h4020f0d0;
#20
a=32'hf0000000;
b=32'h00000000;
#20
a=32'h00000000;
b=32'h0f000000;
#20
//sll/slr
aluc=4'b1111;
a=1;
b=32'h4020f0d0;
#20
aluc=4'b1110;
a=2;
b=32'b10110111011110111110111111011111;
#20
a=3;
```

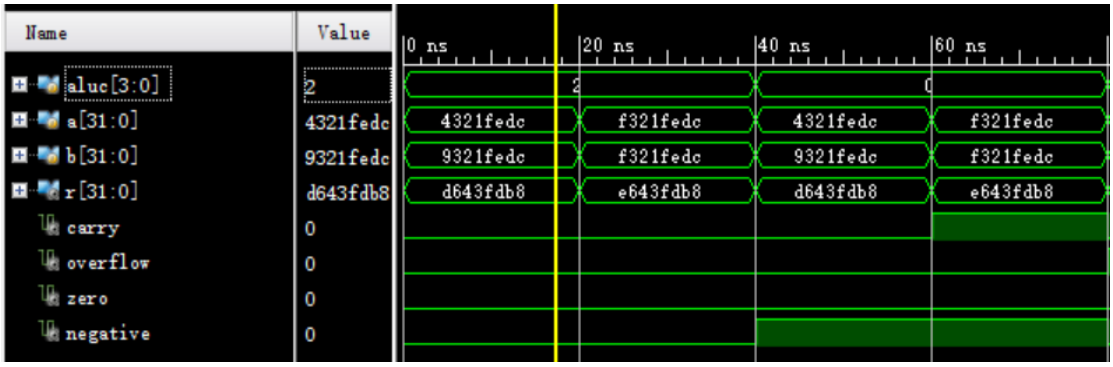
```
        b=32'b10110111011110111110111111011111;
        #20
        //srl
        aluc=4'b1101;
        a=1;
        b=32'h4020f0d0;
        #20
        a=2;
        b=32'b10110111011110111110111111011111;
        #20
        a=3;
        b=32'b10110111011110111110111111011111;
        #20
        //sra
        aluc=4'b1100;
        a=1;
        b=32'h4020f0d0;
        #20
        a=2;
        b=32'b10110111011110111110111111011111;
        #20
        a=3;
        b=32'b10110111011110111110111111011111;
        #20
        //lui
        aluc=4'b1001;
        b=32'b10110111011110111110111111011111;
        #20
        aluc=4'b1000;
        b=32'b00110111011110111110111111011111;
        #20
        b=32'b10110111011110111110111111011110;
        #20
        $stop;
    end
endmodule
```

五、实验结果

（该部分可截图说明，要求 logisim 逻辑验证图、modelsim 仿真波形图、以及下板后的实验结果贴图（实验步骤中没有下板要求的实验，不需要下板贴图））

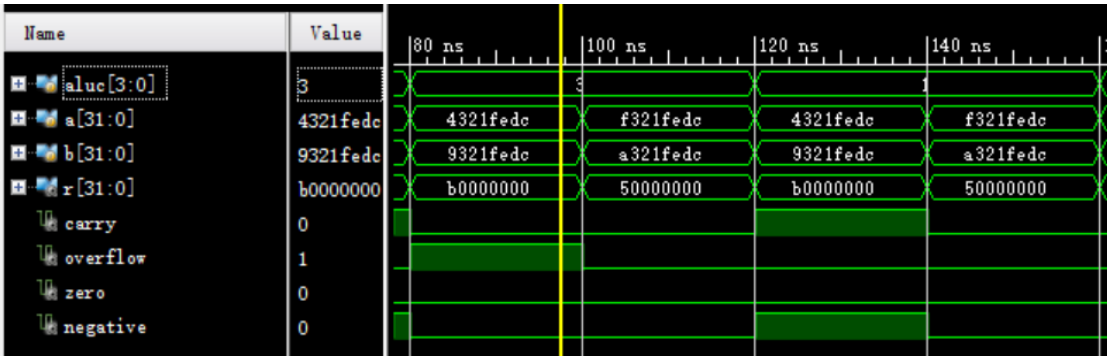
有无符号的加：

Addu r=a+b 无符号	0	0	0	0
Add r=a+b 有符号	0	0	1	0

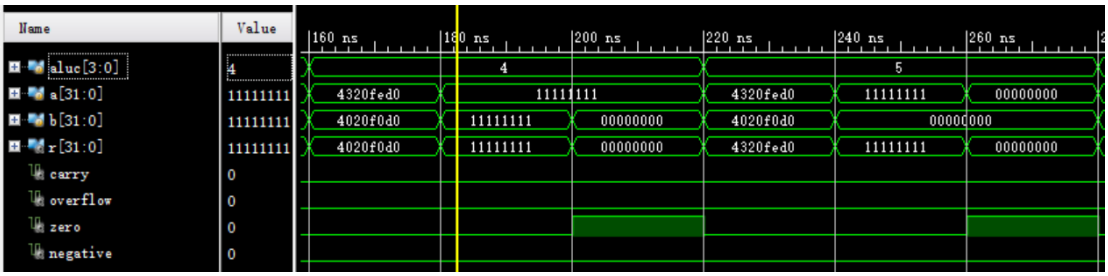


有无符号的减：

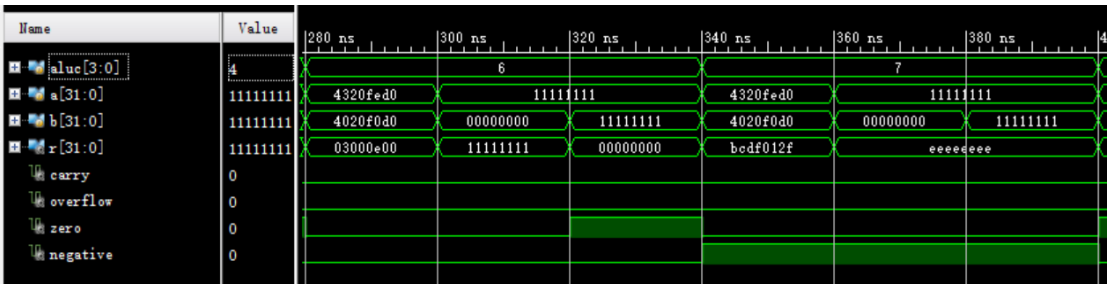
Subu r=a-b 无符号	0	0	0	1
Sub r=a-b 有符号	0	0	1	1



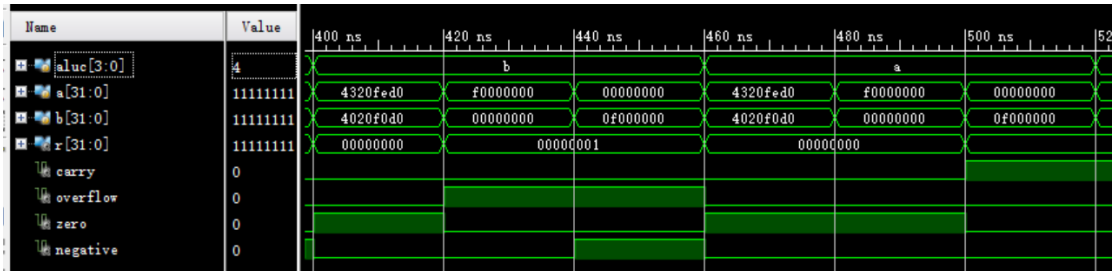
And r=a & b	0	1	0	0
Or r=a | b	0	1	0	1



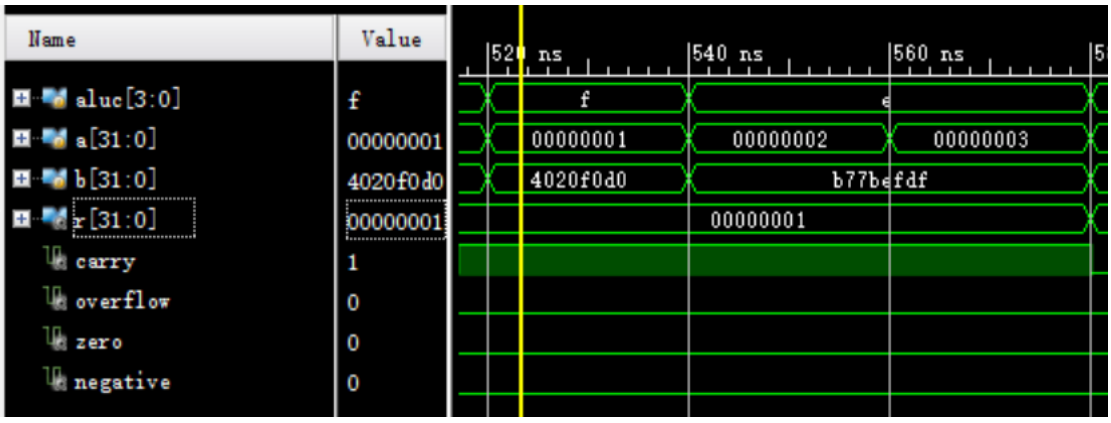
Xor	r=a ^ b	0	1	1	0
Nor	r=~ (a | b)	0	1	1	1



Slt	r=(a<b)?1:0 有符号	1	0	1	1
Sltu	r=(a<b)?1:0 无符号	1	0	1	0

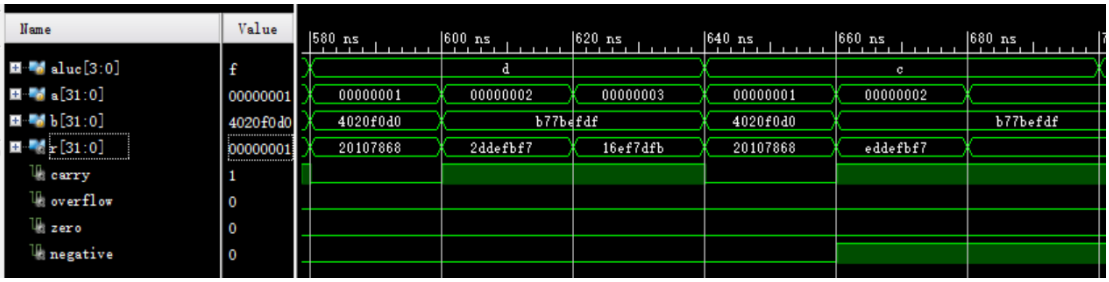


Sll/Slr	r=b<<a	1	1	1	X



Srl r=b>>a	1	1	0	1

Sra r=b>>>a	1	1	0	0



Lui r={b[15:0],16'b0}	1	0	0	X