实验四：算术逻辑单元的设计

实验时间： 2017年11月20日 第十一周 星期一
实验者：16307130194 陈中钰 16级 计算机科学技术学院
座位号：30
指导老师：唐志强

1 实验目的

- 了解并设计 CPU 的核心部件算术逻辑单元 ALU
- 加深对减法运算以及补码的理解
- 熟练并准确设计控制信号
- 熟练加减法、与或非、异或运算

2 实验原理

- 功能控制表

S1	S0	M=0 逻辑运算	M=1 算术运算	
			Cn=0	Cn=1
0	0	F=not A	A+B	A+B+1
0	1	F=A and B	A-B	A-B-1
1	0	F=A or B		
1	1	F=A xor B		

- 逻辑运算：not，and，or，xor 分别对应~，&，|，^运算
- 加法运算：利用实验三实现的4位超前进位加法器，可以直接求 A+B+Cn
- 减法运算：

  A-B=A+(-B)=A+(~B+1)=A+~B+~Cn

  A-B-1=A+(-B)-1=A+(~B+1)-1=A+~B+~Cn

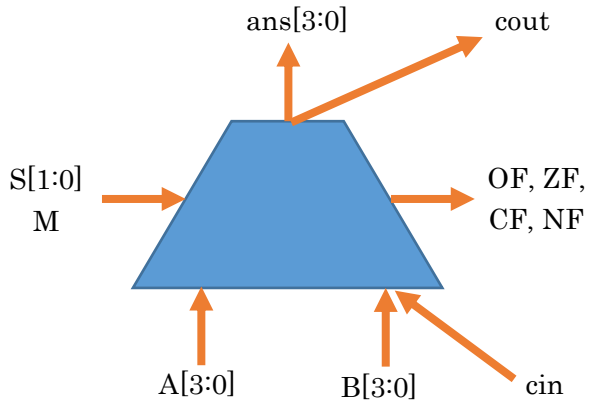
  那么同样可以利用上述超前加法器，只需把输入的 B，Cn 取反即可

  （注意：当 B 为0000时不可以取反，因为0没有补码！因此当 B 为0时需要特判）
- 条件码

条件码	意义	表达式
OF	溢出	cout^c[3]
ZF	ans 为0	~(ans[3]|ans[2]|ans[1]|ans[0])
CF	产生进位	cout
NF	ans 为负数	ans[3]

其中 ans[3:0]为计算结果，cout，c[3]均为进位

- 器件图



3 实验分析

- 由于 always@()语句中不能使用模块，因此可以在 always@()外面使用超前加法器模块，并把加法运算、减法运算的值、进位事先算出来，再在 always@()语句中判决最终的值、进位应该取哪一组结果
- 减法运算，在上述判断的时候，还要判断 B 是否为0，若 B 为0，则不能用超前进位加法器模块的计算结果，而需要重新计算

4 实验内容

4.1 Verilog 代码（加减法运算可以直接使用实验三的4位超前进位加法器的模块，故在此省略它的实现代码）

```
module ALU(input [3:0] a, input [3:0] b, input cin, input m, input
[1:0] s, output reg [3:0] ans, output reg cout);
    wire [3:0] ans1,ans2;
    wire cout1,cout2;
    carry_look_ahead_adder_4 plus1(a[3:0],b[3:0],cin,ans1[3:0],cout1);
    carry_look_ahead_adder_4 plus2(a[3:0],~b[3:0],~cin,ans2[3:0],cout2);
    always@(a,b,cin,m,s,ans1,ans2,cout1,cout2)
    begin
    if(m)
        begin
        case(s)
            2'b00: begin ans[3:0]=ans1[3:0]; cout=cout1; end
            2'b01:
            begin
            if(b==4’b0000) begin ans[3:0]=a[3:0]-cin; cout=0; end
            else begin ans[3:0]=ans2[3:0]; cout=cout2; end
            end
            default: begin ans[3:0]=4'b0000; cout=0; end
        endcase
        end
    else
```

```
        begin
        case(s)
            2'b00:ans[3:0]=~a[3:0];
            2'b01:ans[3:0]=a[3:0]&b[3:0];
            2'b10:ans[3:0]=a[3:0]|b[3:0];
            2'b11:ans[3:0]=a[3:0]^b[3:0];
        endcase
        cout=0;
        end
    end
endmodule
```

4.2 操作

- 当 M 为0时，可以实现4种逻辑运算，当 M 为1时，可以实现加减法运算

4.3 改进

- 添加 OF、ZF、CF、NF 共4个条件码
- 其中除了 OF 以外均可以直接 assign 赋值

assign ZF=~(ans[0]|ans[1]|ans[2]|ans[3]);
assign CF=cout;
assign NF=ans[3];

- 而 OF 要在超前进位加法器模块中计算，则需要修改实验三中的4位超前进位加法器，添加 OF 作为输出，且在应用该模块时也要加上 OF 作为输出，其中 OF 计算式为 assign OF=c[3]^cout;

```
module carry_look_ahead_adder_4(input [3:0] a, input [3:0] b, input
cin, output [3:0] s, output cout, output OF);
    wire [3:0] g; wire [3:0] p; wire [4:0] c;
    assign g[3:0]=a[3:0]&b[3:0]; assign p[3:0]=a[3:0]^b[3:0];
    assign c[0]=cin;
    assign c[1]=g[0]|p[0]&c[0];
    assign c[2]=g[1]|p[1]&g[0]|p[1]&p[0]&c[0];
    assign c[3]=g[2]|p[2]&g[1]|p[2]&p[1]&g[0]|p[2]&p[1]&p[0]&c[0];
    assign c[4]=g[3]|p[3]&g[2]|p[3]&p[2]&g[1]|p[3]&p[2]&p[1]&g[0]|
    p[3]&p[2]&p[1]&p[0]&c[0];
    assign cout=c[4];
    assign OF=c[3]^cout;
    xor_2 x(p[3:0], c[3:0], s[3:0]);
endmodule
```

5 实验结论

- 能够实现4位 ALU 的四种逻辑运算、加法减法运算，并能输出 OF、ZF、NF、CF 共四个条件码

6 实验感想

- 实验的时候，定义变量 cout 时不小心误写为 out，而且没有报错，结果导致算术运算都是错误的，而且完全不知道哪里有 bug，最后仔细查看一次又一次后才最

终找出来并改正

- 大大加深了对补码运算的理解
- 加强了对电路控制的设计能力，当判断分支多了之后，一定要淡定，清楚每个分支的走向和执行条件，才能很好地把握整个电路