数字电路与数字系统

实验报告

实　　验: 加法器与 ALU

姓　　名: 周心同

学　　号: 201220069

1.实验目的

复习加法器的原理

学习用简单 ALU 的设计方式

2.实验原理

2.1 4 位补码加减法

减法：对减数进行求补操作，将减法变成加法。一个数的补码就是将其连同符号位在内全部取反再加一的过程。

溢出标志位：如果两个参加加法运算的变量符号相同，而运算结果的符号与 其不相同，则运算结果不准确，产生溢出。即两个正数相加结果为负数，或者 两个负数相加结果为正数，则发生了溢出。一正一负两个数相加是不会产生溢出的。

Zero 标志位：用 if 语句，将 Result 和 “0” 相比较。或者用一元约简运算。

```
assign zero = ~(| Result);
```

|Result 即：将 Result 各位进行或运算。

2.2 4 位带符号位的补码 ALU

表 3-1: ALU 功能列表

功能选择	功能	操作
000	加法	A+B
001	减法	A-B
010	取反	Not A
011	与	A and B
100	或	A or B
101	异或	A xor B
110	比较大小	If A<B then out=1; else out=0;
111	判断相等	If A==B then out=1; else out=0;

用 case 语句分别实现

3.实验环境

- 软件环境

Quartus 17.1 Lite

- 硬件环境（预计）

开发板：DE10 Standard

FPGA：Intel Cyclone V SE 5CSXFC6D6F31C6N

4.实验步骤和结果

4.1 代码设计

4.1.1 4 位补码加减法

```
module adder(
    input  [3:0] A,
    input  [3:0] B,
    input  addsub,
    output [3:0] F,
    output cf,
    output zero,
    output of
    );

// add your code here
wire [3:0]t;
wire m;
assign t=(   { 4 {addsub} }  ^B);
assign { m, F } = A + t+addsub;
assign cf=m^addsub;
assign of= (A[3]==t[3]) && (F[3]!=A[3]) ;
assign zero=~( | F);


endmodule
```

4.1.2 4 位带符号位的补码 ALU

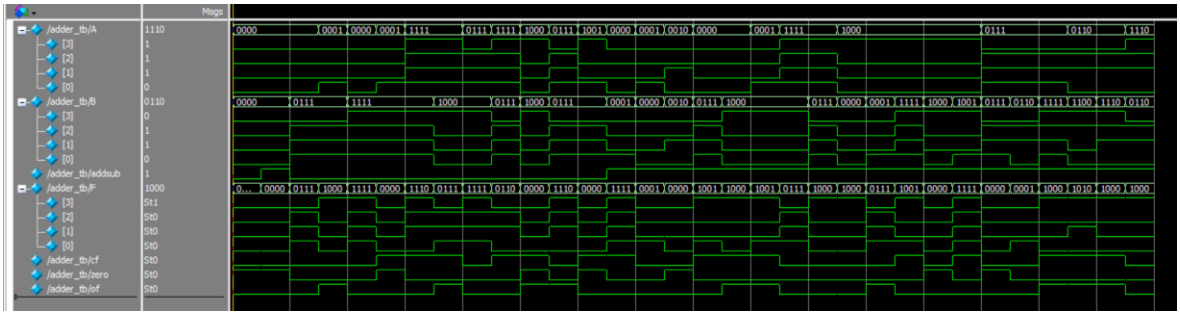
```
module ALU( input [3:0] A,
            input [3:0] B,
            input [2:0] ALUctr,
            output reg [3:0] F,
            output reg cf,
            output reg zero,
            output reg of
);

 //add your code here
 reg[4:0] t;
 always @(*)
begin
  case (ALUctr)
  0: begin
   F=A+B;
   zero=~( | F);
   cf=(A+B>F);
   of=(F!=A+B);
  end
  1:begin
   F=A-B;
   zero=~( | F);
   of=(F!=A-B);
   t=A-B;
   cf=t[4];
  end
  2:begin
    F=~A;
   cf=0;
   of=0;
   zero=~( | F);
  end
  3:begin
    F=A&B;
   cf=0;
   of=0;
    zero=~( | F);
  end
  4:begin
    F=A|B;
   cf=0;
   of=0;
    zero=~( | F);
  end
```

```
46      5:begin
47        F=A^B;
48       cf=0;
49       of=0;
50        zero=~( | F);
51      end
52      6:begin
53        if(A[3]==0)
54       begin
55            if(B[3]==0)
56          begin
57              if(A>B) F=4'b0001;
58           else F=4'b0000;
59          end
60          else
61            F=4'b0001;
62       end
63       else
64       begin
65            if(B[3]==0)
66          F=4'b0000;
67          else
68          begin
69              if(A>B) F=4'b0001;
70           else F=4'b0000;
71          end
72       end
73       cf=0;
74       of=0;
75        zero=~( | F);
76      end
77      7:begin
78      if(A==B) F=4'b0001;
79      else F=4'b0000;
80      cf=0;
81      of=0;
82      zero=~( | F);
83      end
84      endcase
85
86    end
```

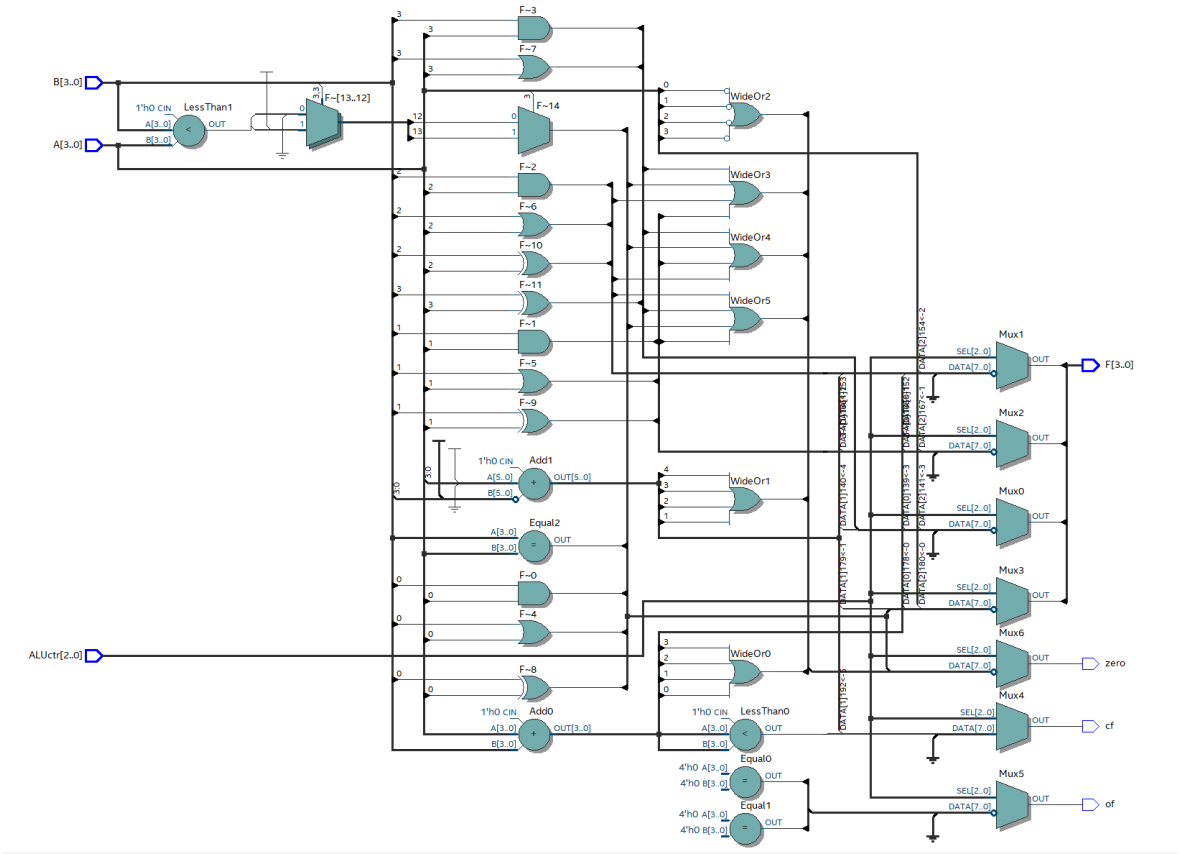
4.2 仿真测试

4.2.1 4 位补码加减法

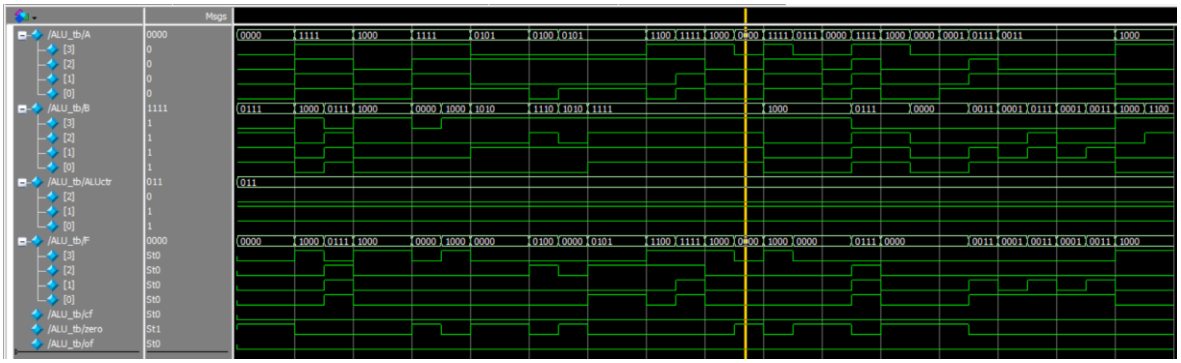
```
1  //testbench
2  `timescale 1ns/10ps
3  module adder_tb;
4
5  reg  [3:0] A;
6  reg  [3:0] B;
7  reg  addsub;
8  wire [3:0] F;
9  wire cf;
10 wire zero;
11 wire of;
12
13 adder adder(
14     .A(A),
15     .B(B),
16     .addsub(addsub),
17     .F(F),
18     .cf(cf),
19     .zero(zero),
20     .of(of)
21
22 );
23
24 initial begin
25
26      A=4'b0000; B=4'b0000; addsub=0;
27 #10 A=4'b0000; B=4'b0000; addsub=1;
28 #10 A=4'b0000; B=4'b0111; addsub=0;
29 #10 A=4'b0001; B=4'b0111; addsub=0;
30 #10 A=4'b0000; B=4'b1111; addsub=0;
31 #10 A=4'b0001; B=4'b1111; addsub=0;
32 #10 A=4'b1111; B=4'b1111; addsub=0;
33 #10 A=4'b1111; B=4'b1000; addsub=0;
34 #10 A=4'b0111; B=4'b1000; addsub=0;
35 #10 A=4'b1111; B=4'b0111; addsub=0;
36 #10 A=4'b1000; B=4'b1000; addsub=0;
37 #10 A=4'b0111; B=4'b0111; addsub=0;
38 #10 A=4'b1001; B=4'b0111; addsub=0;
39 #10 A=4'b0000; B=4'b0001; addsub=1;
40 #10 A=4'b0001; B=4'b0000; addsub=1;
41 #10 A=4'b0010; B=4'b0010; addsub=1;
42 #10 A=4'b0000; B=4'b0111; addsub=1;
43 #10 A=4'b0000; B=4'b1000; addsub=1;
44 #10 A=4'b0001; B=4'b1000; addsub=1;
45 #10 A=4'b1111; B=4'b1000; addsub=1;
46 #10 A=4'b1111; B=4'b0111; addsub=1;
47 #10 A=4'b1000; B=4'b0000; addsub=1;
48 #10 A=4'b1000; B=4'b0001; addsub=1;
49 #10 A=4'b1000; B=4'b1111; addsub=1;
50 #10 A=4'b1000; B=4'b1000; addsub=1;
51 #10 A=4'b1000; B=4'b1001; addsub=1;
52 #10 A=4'b0111; B=4'b0111; addsub=1;
53 #10 A=4'b0111; B=4'b0110; addsub=1;
54 #10 A=4'b0111; B=4'b1111; addsub=1;
55 #10 A=4'b0110; B=4'b1100; addsub=1;
56 #10 A=4'b0110; B=4'b1110; addsub=1;
57 #10 A=4'b1110; B=4'b0110; addsub=1;
58 #10;
59 end
60
61 endmodule
```



4.2.2 4 位带符号位的 ALU



```
1  //testbench
2  `timescale 1ns/10ps
3  module ALU_tb;
4
5  reg [3:0] A;
6  reg [3:0] B;
7  reg [2:0] ALUctr;
8  wire [3:0] F;
9  wire cf;
10 wire zero;
11 wire of;
12
13 ALU ALU(
14     .A(A),
15     .B(B),
16     .ALUctr(ALUctr),
17     .F(F),
18     .cf(cf),
19     .zero(zero),
20     .of(of)
21 );
22
23 initial begin
24      A=4'b0000; B=4'b0111; ALUctr=3;
25 #10 A=4'b0000; B=4'b0111; ALUctr=3;
26 #10 A=4'b1111; B=4'b1000; ALUctr=3;
27 #10 A=4'b1111; B=4'b0111; ALUctr=3;
28 #10 A=4'b1000; B=4'b1000; ALUctr=3;
29 #10 A=4'b1000; B=4'b1000; ALUctr=3;
30 #10 A=4'b1111; B=4'b0000; ALUctr=3;
31 #10 A=4'b1111; B=4'b1000; ALUctr=3;
32 #10 A=4'b0101; B=4'b1010; ALUctr=3;
33 #10 A=4'b0101; B=4'b1010; ALUctr=3;
34 #10 A=4'b0100; B=4'b1110; ALUctr=3;
35 #10 A=4'b0101; B=4'b1010; ALUctr=3;
36 #10 A=4'b0101; B=4'b1111; ALUctr=3;
37 #10 A=4'b0101; B=4'b1111; ALUctr=3;
38 #10 A=4'b1100; B=4'b1111; ALUctr=3;
39 #10 A=4'b1111; B=4'b1111; ALUctr=3;
40 #10 A=4'b1000; B=4'b1111; ALUctr=3;
41 #10 A=4'b0000; B=4'b1111; ALUctr=3;
42 #10 A=4'b1111; B=4'b1000; ALUctr=3;
43 #10 A=4'b0111; B=4'b1000; ALUctr=3;
44 #10 A=4'b0000; B=4'b1000; ALUctr=3;
45 #10 A=4'b1111; B=4'b0111; ALUctr=3;
46 #10 A=4'b1000; B=4'b0111; ALUctr=3;
47 #10 A=4'b0000; B=4'b0000; ALUctr=3;
48 #10 A=4'b0001; B=4'b0000; ALUctr=3;
49 #10 A=4'b0111; B=4'b0011; ALUctr=3;
50 #10 A=4'b0011; B=4'b0001; ALUctr=3;
51 #10 A=4'b0011; B=4'b0111; ALUctr=3;
52 #10 A=4'b0011; B=4'b0001; ALUctr=3;
53 #10 A=4'b0011; B=4'b0011; ALUctr=3;
54 #10 A=4'b1000; B=4'b1000; ALUctr=3;
55 #10 A=4'b1000; B=4'b1100; ALUctr=3;
56 #10;
57
58 end
59
60 endmodule
61
```



5. 实验中遇到的问题及解决办法

头歌题目中，需要对带符号数进行大小比较。

解决办法：先比较符号位再整体比较。

6.实验启示和建议

在普通模块中，默认变量是 wire 型，wire 型用于 assign，reg 型用于 always。

在 testbench 中，输入接 reg 型，输出接 wire 型