实验报告（三）

实验名称：加法器和 ALU

姓名：常书浩

学号：201220130

1 实验目的

- 复习加法器的原理
- 学习用简单 ALU 的设计方式
- 实现一个带有逻辑运算的简单 ALU

2 实验原理

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;

如图所示为需要实现的 alu 所需要的功能

3 实验环境

- 软件环境：Quartus 17.1 Lite

4 实验步骤

4.1 工程构建

使用 SystemBuilder 构建工程，需要的器件有 Button、LED、Switch

4.2 加减法的设计

```
if(KEY[0] == 0) begin
   f = a + b;
   cf = f[4];
   if (a[3] == b[3]) begin
      if (b[3] == f[3])
         of = 0;
      else
         of = 1;
   end
   else
      of = 0;
end
else begin
   f = a - b;
   cf = a < b;
   if (a[3] != b[3]) begin
      if (b[3] != f[3])
         of = 0;
      else
         of = 1;
   end
   else
      of = 0;
end
zero = (f[3:0] == 0) ? 1 : 0;
F = f;
```

KEY[0]控制加减法，1 表示减法，0 表示加法，进行分类讨论并生成所需信号

4.3 逻辑运算的设计

```
cf = 0;
of = 0;
case(KEY[2:0])
   3'b010:begin
      a = ~a;
      F = a;
   end
   3'b011:
      F = a & b;
   3'b100:
      F = a | b;
   3'b101 :
      F = a ^ b;
   3'b110 : begin
      if(a[3] == b[3])
         F = a < b;

      if(a[3] != b[3])
         F = a > b;
   end
   3'b111 :
      F = a == b;
endcase
zero = F == 0 ? 1 : 0;
```

首先由题意得知 cf，of 置为 0，采用 case 语句对控制端进行分类讨论，最后对 zero 进行赋值

4.4 整体的组合

此处我将 KEY 的低三位作为 ALUctr，将 Switch 的 4-7 位表示 A，0-3 位表示 B，LED 的低 3 位表示运算结果，第 4 位表示 cf，第 5 位表示 zero，第 6 位表示 of

通过功能列表我们发现可以将功能分为两部分当 ALUctr 的高 2 位为 00 时，进行补码运

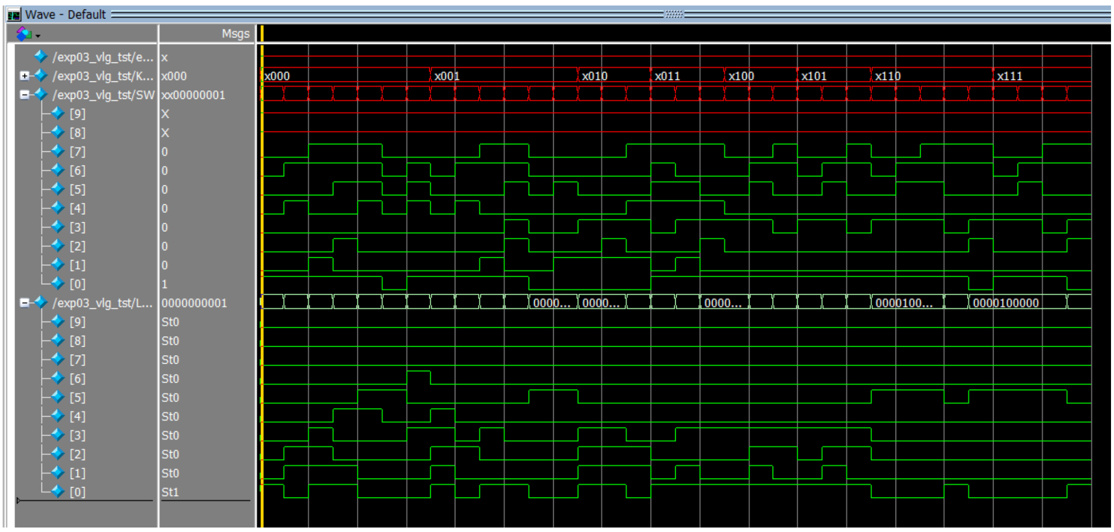
算，其余情况进行逻辑运算，因此进行分类，图略

4.5 电路仿真

编写 vt 文件

```
KEY[2:0] = 3'b000; SW[7:4] = 4'b0000; SW[3:0] = 4'b0001; #50;
KEY[2:0] = 3'b000; SW[7:4] = 4'b0101; SW[3:0] = 4'b0001; #50;
KEY[2:0] = 3'b000; SW[7:4] = 4'b1100; SW[3:0] = 4'b0011; #50;
KEY[2:0] = 3'b000; SW[7:4] = 4'b1110; SW[3:0] = 4'b0101; #50;
KEY[2:0] = 3'b000; SW[7:4] = 4'b1111; SW[3:0] = 4'b0001; #50;
KEY[2:0] = 3'b000; SW[7:4] = 4'b0000; SW[3:0] = 4'b0000; #50;
KEY[2:0] = 3'b000; SW[7:4] = 4'b0111; SW[3:0] = 4'b0001; #50;

KEY[2:0] = 3'b001; SW[7:4] = 4'b0000; SW[3:0] = 4'b0001; #50;
KEY[2:0] = 3'b001; SW[7:4] = 4'b0101; SW[3:0] = 4'b0001; #50;
KEY[2:0] = 3'b001; SW[7:4] = 4'b1100; SW[3:0] = 4'b0011; #50;
KEY[2:0] = 3'b001; SW[7:4] = 4'b1110; SW[3:0] = 4'b1101; #50;
KEY[2:0] = 3'b001; SW[7:4] = 4'b0000; SW[3:0] = 4'b0000; #50;
KEY[2:0] = 3'b001; SW[7:4] = 4'b0010; SW[3:0] = 4'b0010; #50;

KEY[2:0] = 3'b010; SW[7:4] = 4'b0000; SW[3:0] = 4'b1010; #50;
KEY[2:0] = 3'b010; SW[7:4] = 4'b0000; SW[3:0] = 4'b1110; #50;
KEY[2:0] = 3'b010; SW[7:4] = 4'b1001; SW[3:0] = 4'b1010; #50;

KEY[2:0] = 3'b011; SW[7:4] = 4'b1111; SW[3:0] = 4'b0001; #50;
KEY[2:0] = 3'b011; SW[7:4] = 4'b1011; SW[3:0] = 4'b1011; #50;
KEY[2:0] = 3'b011; SW[7:4] = 4'b1001; SW[3:0] = 4'b1101; #50;

KEY[2:0] = 3'b100; SW[7:4] = 4'b0000; SW[3:0] = 4'b1001; #50;
KEY[2:0] = 3'b100; SW[7:4] = 4'b0110; SW[3:0] = 4'b1001; #50;
KEY[2:0] = 3'b100; SW[7:4] = 4'b1100; SW[3:0] = 4'b0001; #50;

KEY[2:0] = 3'b101; SW[7:4] = 4'b0000; SW[3:0] = 4'b1001; #50;
KEY[2:0] = 3'b101; SW[7:4] = 4'b0110; SW[3:0] = 4'b1001; #50;
KEY[2:0] = 3'b101; SW[7:4] = 4'b1100; SW[3:0] = 4'b0001; #50;

KEY[2:0] = 3'b110; SW[7:4] = 4'b0000; SW[3:0] = 4'b1001; #50;
KEY[2:0] = 3'b110; SW[7:4] = 4'b0110; SW[3:0] = 4'b1001; #50;
KEY[2:0] = 3'b110; SW[7:4] = 4'b1110; SW[3:0] = 4'b1001; #50;
KEY[2:0] = 3'b110; SW[7:4] = 4'b1100; SW[3:0] = 4'b0001; #50;
KEY[2:0] = 3'b110; SW[7:4] = 4'b1100; SW[3:0] = 4'b1100; #50;

KEY[2:0] = 3'b111; SW[7:4] = 4'b0000; SW[3:0] = 4'b1001; #50;
KEY[2:0] = 3'b111; SW[7:4] = 4'b0110; SW[3:0] = 4'b1001; #50;
KEY[2:0] = 3'b111; SW[7:4] = 4'b1100; SW[3:0] = 4'b0001; #50;
KEY[2:0] = 3'b111; SW[7:4] = 4'b1100; SW[3:0] = 4'b1100; #50;
```

进行仿真



波形图符合预期