

# 实验三

## 加法器和 ALU 的设计

实验报告

日期：2020 年 9 月 26 日

姓名：朱嘉琦

学号：191220185

班级：数电实验一班

邮箱：1477194584@qq.com



# 实验三报告——加法器与ALU

191220185 朱嘉琦

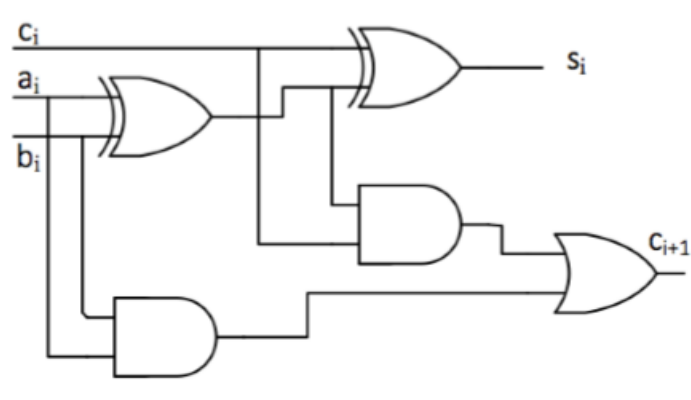
## 一、实验目的

是复习一位全加器的原理，学习设计多位加法器和简单ALU的设计方式。

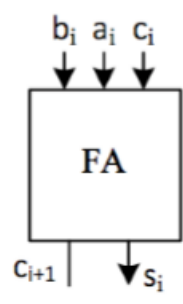
## 二、实验原理

$c_i$	$a_i$	$b_i$	$s_i$	$c_{i+1}$
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

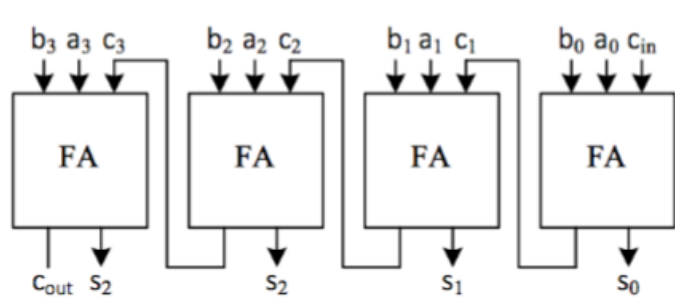
(a) 一位全加器真值表



(b) 一位全加器电路图



(c) 一位全加器框图



(d) 四位串行全加器

图 3-1: 全加器

溢出信号的判断原理：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。

减法器可由加法器实现，先求得(-B)（补码取负的方法是按位取反加1），再与A相加，得到A-B的值。

## 三、实验环境/器材

实验环境是Quarters 17.1 Lite，实验器材是DE10 开发板

## 四、程序代码或流程图

### 1.进行补码加减运算的4位加减运算器

```
1  module exp3_1(A,B,Cin,res,CF,OF,ZF);
2      input [3:0] A,B;
3      input Cin;
4      output reg[3:0] res;
5      output reg CF,OF,ZF;//carry, overflow, zero
6
7      reg [3:0]x;
8      reg sign;
9
10     always @(*) begin
11         if(Cin)
12             begin
13                 x = B;
14                 sign = 0;
15             end
16         else
17             begin
18                 x = (~B);
19                 sign = 1;
20             end
21
22         {CF, res} = A + x + sign;
23         OF = (A[3]==x[3])&&(res[3]!=A[3]);
24         if(res==0) ZF=1;
25         else ZF=0;
26     end
27
28 endmodule
29
```

### 2.4位有符号数ALU

由于部分代码比较简单而且和前面有重复，所以只截取了部分功能的实现代码

```
13  always @(*) begin
14      CF=0;
15      OF=0;
16      ZF=0;//initial
17      case(s)
18          3'd0:
19              begin
20                  {CF,res} = A+B;
21                  OF = (A[3]==B[3])&&(res[3]!=A[3]);
22                  ZF = (res==0);
23              end
24          3'd1:
25              begin
26                  x = (~B)+1;
27                  {CF, res} = A + x;
28                  OF = (A[3]==x[3])&&(res[3]!=A[3]);
29                  ZF = (res==0);
30              end
31          3'd2:
32              begin
33                  for(i=0;i<=3;i=i+1)
34                      res[i]=A[i]^1;
35                  ZF = (res==0);
36              end
37      end
```

```

56
57
58
59
60
61
62
63
64
65
3'd6:
begin
    case({A[3],B[3]})
        2'b0_0: res=(A>B);
        2'b0_1: res=1;
        2'b1_0: res=0;
        2'b1_1: res=(A[2:0]>B[2:0]);
        default: res=0;
    endcase
end

```

## 五、实验步骤

通过加法器的原理构建4位加法器，并正确设置进位和溢出符号的输出逻辑；再通过补码取反加一的原理构造减法器，并同样为它设置进位和溢出符号的输出逻辑。

按照要求，利用已经写好的加减器，添加一些功能，编写简单ALU的各项功能。

## 六、测试代码

### 1. 激励代码

通过在激励代码中枚举所有可能的情况，在仿真波形图中对结果进行验证

















#### 4位加减运算器的激励代码

```

57 A=4'b0001; B=4'b0111; Cin=1; #10;
58 A=4'b0001; B=4'b0110; Cin=1; #10;
59 A=4'b0011; B=4'b0110; Cin=1; #10;
60 A=4'b0101; B=4'b0110; Cin=1; #10;
61 A=4'b1111; B=4'b1111; Cin=1; #10;
62 A=4'b0111; B=4'b0111; Cin=0; #10;
63 A=4'b0111; B=4'b0110; Cin=0; #10;
64 A=4'b0011; B=4'b0110; Cin=0; #10;
65 A=4'b0101; B=4'b1110; Cin=0; #10;
66 A=4'b0001; B=4'b0111; Cin=0; #10;
67 A=4'b1001; B=4'b0111; Cin=0; #10;

```

#### 4位加减运算器的引脚分配

Node Name	Direction	Location	I/O Bank
 A[3]	Input	PIN_AC30	5B
 A[2]	Input	PIN_AB28	5B
 A[1]	Input	PIN_Y27	5B
 A[0]	Input	PIN_AB30	5B
 B[3]	Input	PIN_AD30	5B
 B[2]	Input	PIN_AC28	5B
 B[1]	Input	PIN_V25	5B
 B[0]	Input	PIN_W25	5B
 CF	Output	PIN_ag25	4A
 Cin	Input	PIN_AC29	5B
 OF	Output	PIN_af25	4A
 ZF	Output	PIN_ae24	4A
 res[3]	Output	PIN_ad24	4A
 res[2]	Output	PIN_ac23	4A
 res[1]	Output	PIN_ab23	5A
 res[0]	Output	PIN_aa24	5A

四位有符号数ALU的激励代码

```

57      s=3'b000;    A=4'b0001;    B=4'b0110;    #10;
58                      A=4'b0011;    B=4'b0101;    #10;
59                      A=4'b1111;    B=4'b0010;    #10;
60                      A=4'b0010;    B=4'b1110;    #10;
61      s=3'b001;    A=4'b0110;    B=4'b0001;    #10;
62                      A=4'b0101;    B=4'b0011;    #10;
63                      A=4'b1111;    B=4'b0010;    #10;
64                      A=4'b0010;    B=4'b1110;    #10;
65      s=3'b010;    A=4'b0010;    B=4'b1110;    #10;
66                      A=4'b1111;    B=4'b1110;    #10;
67      s=3'b011;    A=4'b1001;    B=4'b0110;    #10;
68                      A=4'b1111;    B=4'b0011;    #10;
69      s=3'b100;    A=4'b1001;    B=4'b0110;    #10;
70                      A=4'b0000;    B=4'b0011;    #10;
71      s=3'b101;    A=4'b1001;    B=4'b0110;    #10;
72                      A=4'b1111;    B=4'b0011;    #10;
73      s=3'b110;    A=4'b0111;    B=4'b0011;    #10;
74                      A=4'b0000;    B=4'b0111;    #10;
75                      A=4'b0001;    B=4'b1011;    #10;
76                      A=4'b1001;    B=4'b0000;    #10;
77      s=3'b111;    A=4'b1001;    B=4'b1001;    #10;
78                      A=4'b1001;    B=4'b0001;    #10;
79

```

四位有符号数ALU的引脚分配

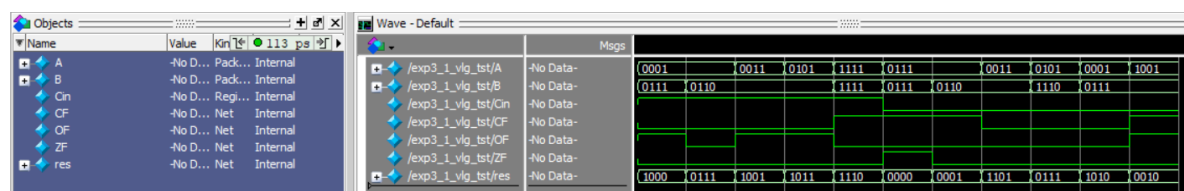
Node Name	Direction	Location
⚡ A[0]	Unknown	PIN_AB30
⚡ A[1]	Unknown	PIN_Y27
⚡ A[2]	Unknown	PIN_AB28
⚡ A[3]	Unknown	PIN_AC30
⚡ B[0]	Unknown	PIN_W25
⚡ B[1]	Unknown	PIN_V25
⚡ B[2]	Unknown	PIN_AC28
⚡ B[3]	Unknown	PIN_AD30
⚡ CF	Unknown	PIN_AG25
⚡ OF	Unknown	PIN_AF25
⚡ ZF	Unknown	PIN_AE24
⚡ res[0]	Unknown	PIN_AA24
⚡ res[1]	Unknown	PIN_AB23
⚡ res[2]	Unknown	PIN_AC23
⚡ res[3]	Unknown	PIN_AD24
⚡ s[0]	Unknown	PIN_AJ4
⚡ s[1]	Unknown	PIN_AK4
⚡ s[2]	Unknown	PIN_AA14

## 2. 硬件测试

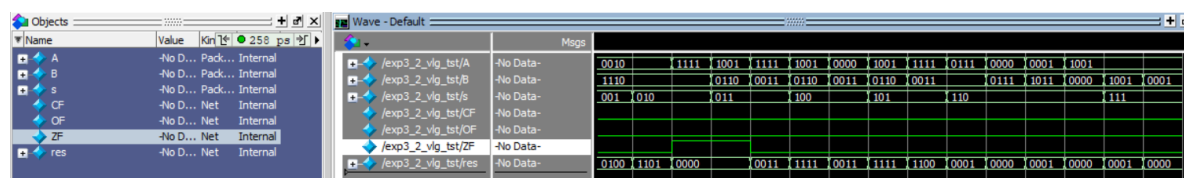
在开发板上验证所有可能的输入，观察输出结果与输出逻辑是否一致来进行验证，与真值表结果一致。

# 七、实验结果

## 1. 4位加减运算器



## 2. 4位有符号数ALU



# 八、思考

## 思考题1:

在此加减运算的运算器中，如果用判断参与运算的加数和运算结果符号位是否相同的方法来判断是否溢出，那么此时判断溢出位的时候，应该会比较操作数A、B和运算结果的符号位，还是比较A1、B1和运算结果的符号位？

应该比较A1和B1，因为最后进入加法器的两个操作数是A1和B1，所以应该比较这两个操作数和结果的运算位。

### 思考题2:

两个方法在Cin为1时（做减法操作）是不一样的，方法一是先取反，然后再进行加法的时候再加1，而方法二是直接将操作数取反加一后再进行加法，这两种方法的运算结果应该是一样的，但是进位位和溢出位不一定一样。

比如Cin=1; A=0001; B=1000; 运算结果都是1001，但是第一种运算结果是有溢出，而第二种运算没有溢出，显然 $1 - (-8) = 9$ 是溢出的，所以第一种方法是正确的。

### 思考题3:

一元约简运算。约简运算符对单个操作数进行运算，最后返回一位数，其运算过程为：首先将操作数的第一位和第二位进行与、或、非运算；然后再将运算结果和第三位进行与、或、非运算；依次类推直至最后一位。

## 九、实验中遇到的问题及解决方法

我在做实验的时候没有仔细看思考题，我在写代码的时候就用了思考题2的第二种方法，大多数结果是正确的，但在边界的几个用例上输出了错误结果，最后向老师提问之后对代码进行了修改得到了正确答案。

## 十、实验得到的启示

思考题中的问题可能就是在实验中会遇到的问题，在做实验之前可以先考虑一下思考题。

## 十一、意见和建议

希望老师讲解一下有关task的知识，看网络上的资料有些看不懂。