浙江大学

本科实验报告

课程名称：数字逻辑设计

姓　　名：颜晗。

学　　院：计算机科学与技术学院

系：

专　　业：计算机科学与技术

学　　号：3200105515

指导教师：蔡铭

2021 年　12　月　11　日

浙江大学实验报告

课程名称：数字逻辑设计　　实验类型：综合

实验项目名称：加减法器和 ALU 基本原理与设计

学生姓名：颜晗　专业：计算机科学与技术　学号：3200105515

同组学生姓名：　　指导老师：蔡铭

实验地点：东四 509　　实验日期：2021 年 11 月 22 日

一、实验目的和要求

1.掌握减法器的实现原理

2.掌握加减法器的设计方法

3.掌握 ALU 基本原理及在 CPU 中的作用

4.掌握 ALU 的设计方法

5.掌握 UCF 文件

二、实验内容和原理

实验内容：任务 1：原理图方式设计 4 位加减法器；

任务 2：实现 4 位 ALU 及应用设计。

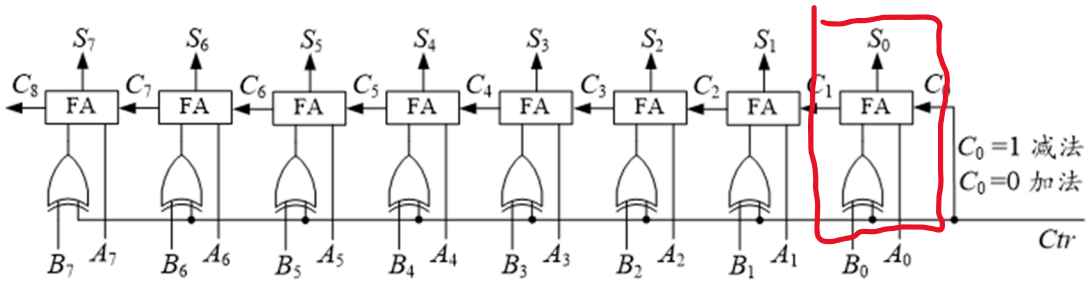
实验原理：ALU 组成与设计原理

1. 组成：ALU 需要支持两个四 bit 数的四种运算：加（+）、减（-）、与(&)、或(|)，通过修改实验八中的全加器为加减法器可以实现加减功能，与和或功能可以借助 Verilog 自带的一位 and 和 or 函数来设计四位的门模块。
2. 加减法器的修改

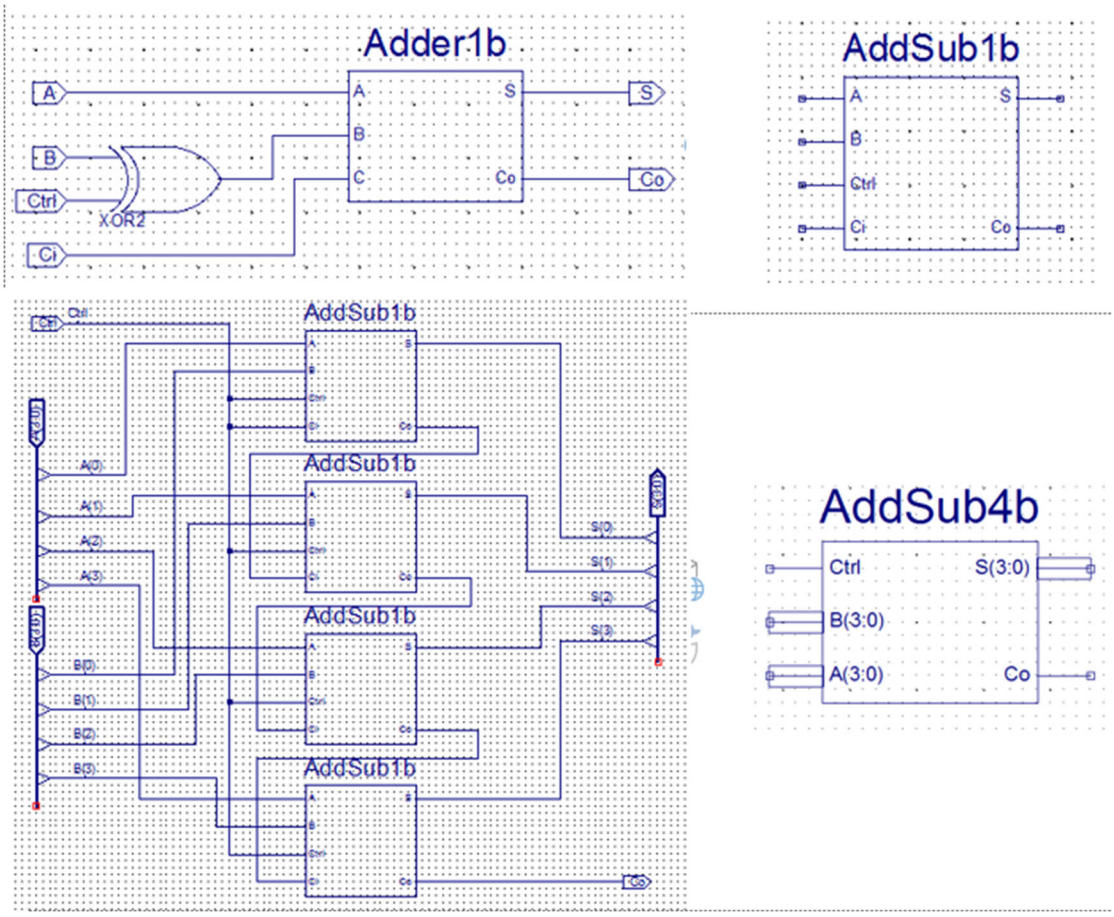
全加器的原理在实验八中已有叙述，通过逐位计算并将进位输出到下一位可以依次计算出所有位的结果。而减法实际上就是加上一个负数，而在二进制补码表示方法中，一个数的相反数即每位取反再加 1。

求反原理：“异或”门。容易得到，当其中一个输入固定为 1 时，输出与另一个输入相反，再将固定 1 的输入分支到加法器第一位的进位输入中，即可计算减法。反之，固定一个输入为 0，即可计算减法。这个“固定”的信号可作为控

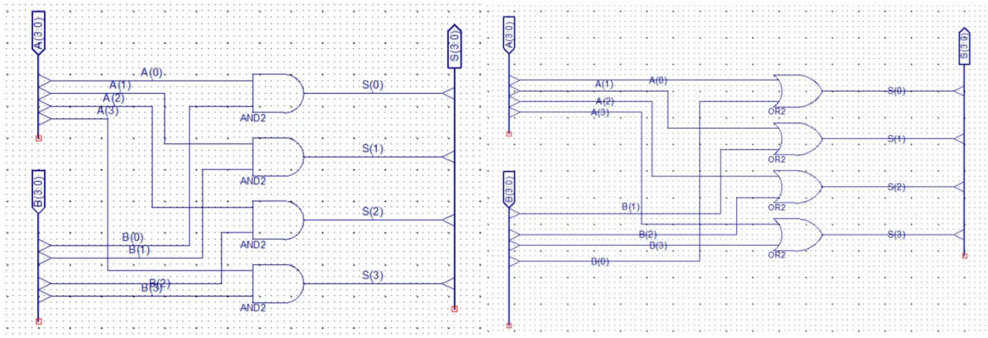
制信号由我们输入。具体如图：



具体实践中和全加器类似，我们可以先设计一位的加减法器，再将四个一位加减法器结合设计出四位的加减法器。原理图如下：

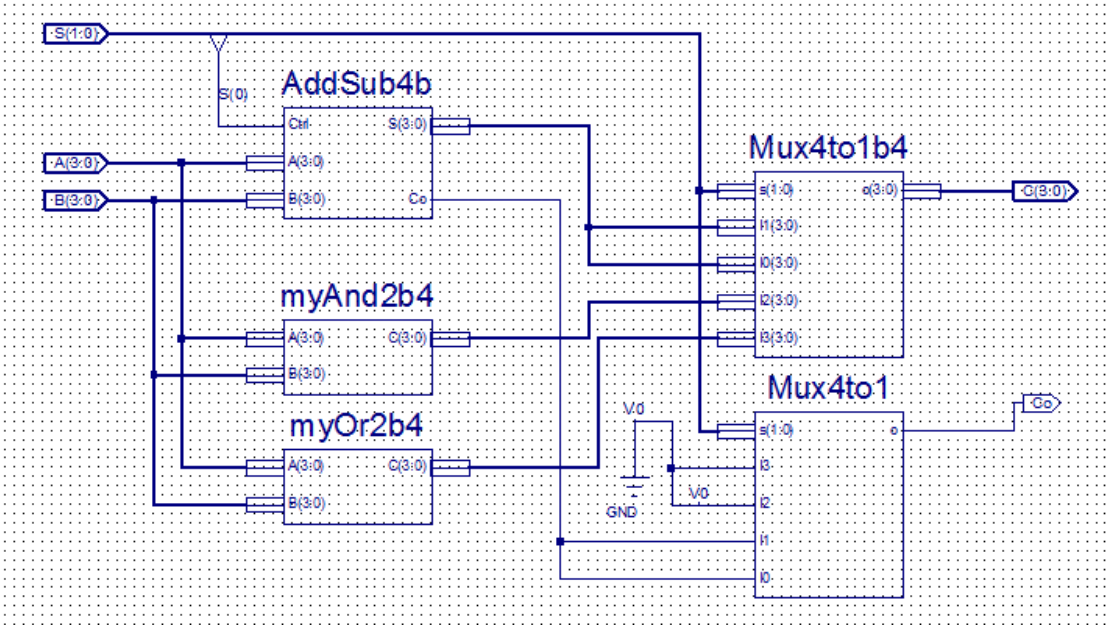


3. 或门、与门的扩展：利用自带的一位输出的 or2 和 and2 扩展至多位计算后再统一输出。原理图如下：



4．组装 ALU

输入两个四位数和控制信号，利用多路选择器和控制信号可以将四种计算结果选择性输出。原理图：



三、实验过程和数据记录

任务 1：原理图方式设计 4 位加减法器

1. 建立工程
2. 新建源文件，编写一位加减法器的 Verilog 代码（记得将实验八中设计的一位加法器添加进工程或重新设计整合加法模块）

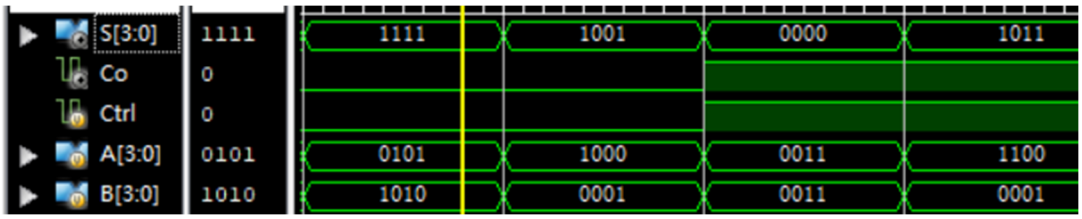
```
module Addsub1b(
    input wire A,B,Ctrl,Ci,
    output wire S,Co
    );
    wire o;
    xor(o,B,Ctrl);
    Adder1b_ver f1(A,o,Ci,S,Co);

endmodule
```

3. 新建源文件，利用一位加减法器设计四位加减法器。

```
module addsub4b(
    input wire Ctrl,
    input wire [3:0] A,
    input wire [3:0] B,
    output wire [3:0] S,
    output wire Co
    );
    wire C1,C2,C3;
    Addsub1b f1(A[0],B[0],Ctrl,Ctrl,S[0],C1);
    Addsub1b f2(A[1],B[1],Ctrl,C1,S[1],C2);
    Addsub1b f3(A[2],B[2],Ctrl,C2,S[2],C3);
    Addsub1b f4(A[3],B[3],Ctrl,C3,S[3],Co);
endmodule
```

4. 对四位加减法器进行波形仿真，代码和波形如下：

```
55          Ctrl = 0;
56         A = 4'b0101;
57         B = 4'b1010;
58         #100
59         Ctrl = 0;
60         A = 4'b1000;
61         B = 4'b0001;
62         #100
63         Ctrl = 1;
64         A = 4'b0011;
65         B = 4'b0011;
66         #100
67         Ctrl = 1;
68         A = 4'b1100;
69         B = 4'b0001;
```



Ctrl=0 加法　0101+1010=1111,无进位；1000+0001=1001，无进位

Ctrl=1 减法　0011-0011=0000，由于补码表示，有 1 进位；

1100-0001=1011，1 进位。

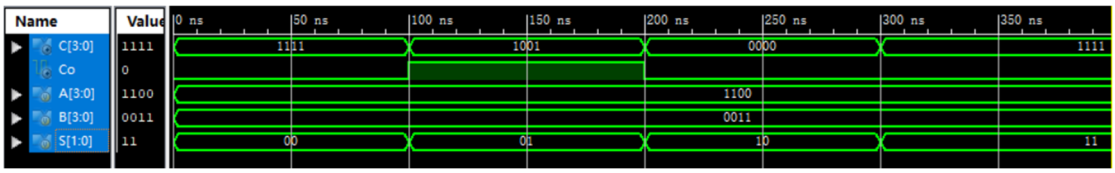
任务 2：实现 4 位 ALU 及应用设计

1. 新建工程和源文件
2. 设计 ALU 的代码（将所需模块添加进工程）

```
module ALUb4(
    input wire [3:0] A,
    input wire [3:0] B,
    input wire [1:0] S,
    output wire [3:0] C,
    output wire Co
    );
    wire [3:0] S2;
    wire Tc;
    wire [3:0] C1,C2;
    wire V0;
    addsub4b f1(.A(A[3:0]),
                .B(B[3:0]),
                .Ctrl(S[0]),
                .S(S2),
                .Co(Tc)
                );
    myAnd2b4 f2(.A(A[3:0]),
                .B(B[3:0]),
                .C(C1[3:0])
                );
    myOr2b4  f3(.A(A[3:0]),
                .B(B[3:0]),
                .C(C2[3:0])
                );
    Mux4to1  f4(.s(S[1:0]),
                .I0(Tc),
                .I1(Tc),
                .I2(V0),
                .I3(V0),
                .o(Co)
                );
    Mux4to1b4 f5(.s(S[1:0]),
                .I0(S2),
                .I1(S2),
                .I2(C1),
                .I3(C2),
                .o(C)
                );
    GND f6(.G(V0));
endmodule
```

3. 对 ALU 模块进行波形仿真（覆盖四种操作）

```
S = 2'b00;
A = 4'b1100;
B = 4'b0011;
#100
S = 2'b01;
A = 4'b1100;
B = 4'b0011;
#100
S = 2'b10;
A = 4'b1100;
B = 4'b0011;
#100
S = 2'b11;
A = 4'b1100;
B = 4'b0011;
```



以下操作因疫情无法上板与原操作有所改变

4. Disp_num 模块的设计。

```
module Disp_num(
      input wire clk,
      input wire RSTN,
      input wire [15:0] num,
      output wire [7:0] SEGMENT,
      output wire [3:0] AN
    );
      wire [31:0] clkdiv;
      wire [3:0] numo;
      wire V0,V5;
      clkdiv f1(.clk(clk),.rst(RSTN),.clkdiv(clkdiv));
      Mux4tolb4 f2(.s(clkdiv[6:5]),.I0(num[3:0]),.I1(num[7:4]),.I2(num[11:8]),.I3(num[15:12]),.o(numo[3:0]));
      Mux4tolb4 f3(.s(clkdiv[6:5]),
                   .I0({V5,V5,V5,V0}),
                   .I1({V5,V5,V0,V5}),
                   .I2({V5,V0,V5,V5}),
                   .I3({V0,V5,V5,V5}),
                   .o(AN[3:0]));
      MyMC14495 f4(.D(numo[3:0]),
                .LE(V5),
                .Point(V0),
                .LED(SEGMENT[6:0]),
                .p(SEGMENT[7]));
      GND f5(V0);
      VCC f6(V5);
endmodule
```

5. ALU 应用的顶层设计（实现功能：两个 4 位操作数 A，B，可用两个按键进行自增/减，根据控制信号计算不同算式得到结果 C 和进位 Co，把 A、B、C 和 Co 动态显示）

   因疫情改为仿真测试，仿真中不会存在抖动，因此将顶层中的时钟计数器和防抖动模块注释删除掉了；另外新增 numout 作为输出展示需要显示的四个数。具体代码如下：

```
module Top(
      input wire clk,
      input wire [1:0]btn,
      input wire [1:0]SW,
      input wire [1:0]SW2,
      input wire BN,
      input wire rst,
      output wire [3:0]AN,
      output wire [7:0]SEGMENT,
      output wire K_ROW,
      output wire [15:0] numout
      //output wire [7:0] num,
      //output wire [3:0] C,
      //output wire Co
    );
      wire [7:0] num;////
      wire [1:0] btn_out;
      wire [3:0] C;////
      wire Co;////
      wire [31:0] clk_div;

         assign numout[7:0] = num[7:0];
         assign numout[11:8] = C[3:0];
         assign numout[15:12] = {3'b000,Co};
```

```
    /*clkdiv m2(.clk(clk),
              .rst(rst),
              .clkdiv(clk_div));*/

    //pbdebounce m0(clk_div[5],btn[0],btn_out[0]);
    //pbdebounce m1(clk_div[5],btn[1],btn_out[1]);
    createNumber m3(.btn(btn[1:0]),
                    .SW(SW[1:0]),
                    .num(num[7:0]));
    /*createNumber m3(.btn(btn_out[1:0]),
                    .SW(SW[1:0]),
                    .num(num[7:0]));*/
    /*ALUb4 m4(.A(num[3:0]),
             .B(num[7:4]),
             .S(SW2[1:0]),
             .C(num[11:8]),
             .Co(num[15]));*/
    ALUb4 m4(.A(num[3:0]),
             .B(num[7:4]),
             .S(SW2[1:0]),
             .C(C[3:0]),
             .Co(Co));
    Disp_num m5( clk, rst, {num[7:0], 3'b0, Co, C[3:0]}, SEGMENT, AN);

    //Disp_num m5( clk, rst,num, SEGMENT, AN);
    BUF m6(.I(BN), .O(K_ROW));
endmodule
```

6. 仿真代码编写

```
54      integer i;
55      initial begin
56         // Initialize Inputs
57         clk = 0;
58         btn = 0;
59         SW = 2'b00;
60         SW2 = 0;
61         BN = 0;
62
63         rst = 1;
64         #10 rst = 0;
65
66         for(i=0;i<12;i=i+1)begin
67         #1 btn =2'b01;
68         #1 btn=2'b00;
69         end
70         for(i=0;i<11;i=i+1)begin
71         #1 btn =2'b10;
72         #1 btn=2'b00;
73         end
74         #50;
75         SW=2'b11;
76         btn =2'b11;
77         #1 btn=2'b00;
78         #1 btn =2'b11;
79         #1 btn=2'b00;
80
81         #50;SW2=2'b01;
82
83         #50;SW2=2'b10;
84
85         #50;SW2=2'b11;#50;
86
87         #1 btn =2'b11;
88         #1 btn=2'b00;
89         #50;SW2=2'b00;
90         #50;SW2=2'b01;
91         #50;SW2=2'b10;
92      end
93  always  #10 clk=~clk;
94  endmodule
```

7. 检查波形结果是否如预期。

四、实验结果分析

本次实验主要检查 numout 表示的四个数字是否相符，至于 AN 和 SEGMENT，不再一一检查，仅看是否可以正常输出。

Numout 即要求输出的四个数，
[3:0]为数 A　[7:4]为数 B　[11:8]为计算结果 C　　[15:12]为进位 Co

功能介绍

1. SW 控制 btn 按下时 A,B 是加还是减

A，B 分别初始化为 2，1.经过几次加后（SW=00）变为 e，c；然后在约 100ns 时 SW 变为 11，于是 btn 变化时逻辑为减，可以看见 A，B 变为 d，b。

2. SW2 控制 ALU 的计算公式。00-加，01-减，10-与，11-或

SW2 为 00 时，A(e)+B(c)=C(a)+Co(1)　⇔ 14+12=10+1*16 成立
A(d)+B(b)=C(8)+Co(1)　⇔ 13+11=8+1*16　成立

SW2 为 01 时，A(d)-B(b)=C(2)　⇔ 13-11=2　成立（由于数的表示方法为补码，实际上计算式为 13+5=2+1*16，即进位也有 1）

SW2 为 10 时，A(d) & B(b)=C(9)　⇔1011 & 1101=1001 成立

SW2 为 11 时，A(d) or B(b)=C(f)　⇔ 1011 or 1101 = 1111　成立

五、实验总结与反思

由于疫情影响，实验过程有了很大的改动，需要自己修改模块，设计仿真代码，导致自身的不足很快暴露出来；一是对 Verilog 仍不熟悉，设计规则不了解，如模拟时钟时，always 需要放在 initial 外，还有变量的赋值。另外就是不熟悉各个模块的具体作用和原理，只是一知半解，导致排查错误时毫无头绪，在重新梳理一遍还结合群中的聊天记录暴露出来的问题后，将问题一一修改，终于可以正常仿真出结果了。再次梳理才发现实际修改的部分其实并不多，只是自己的不熟练导致了很多无效功。