浙江大学

本科实验报告

课程名称：	计算机逻辑设计基础
姓　　名：	龙永奇
学　　院：	计算机科学与技术学院
系：	本系
专　　业：	计算机科学与技术
学　　号：	3220105907
指导教师：	董亚波

2023 年 11 月 17 日

浙江大学实验报告

课程名称：计算机逻辑设计基础 实验类型：综合

实验项目名称：全加器、加减法器和 ALU 基本原理与设计

学生姓名：龙永奇 专业：计算机科学与技术 学号：3220105907

同组学生姓名：赵一帆 指导老师：董亚波

实验地点：东 4-509 实验日期：2023 年 11 月 9 日

一、实验目的和要求

1. 掌握一位全加器的工作原理和逻辑功能
2. 掌握串行进位加法器的工作原理和进位延迟
3. 掌握减法器的实现原理
4. 掌握加减法器的设计方法
5. 掌握 ALU 基本原理及在 CPU 中的作用
6. 掌握 ALU 的设计方法

二、实验内容和原理

内容：

1. 按原理图方式设计 4 位加减法器
2. 实现 4 位 ALU 及应用设计

原理：

1. **1 位全加器**

(a) 三个输入位：数据位 Ai 和 Bi，低位进位输入 Ci

二个输出位：全加和 Si，进位输出 Ci+

逻辑表达式为：

$$S_i = A_i \oplus B_i \oplus C_i$$
$$C_{i+1} = A_iB_i + A_iC_i + B_iC_i$$

(b) 根据事件简化真值表：

A_i	B_i	C_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

图 2.1 真值表

(c) 电路图如下：

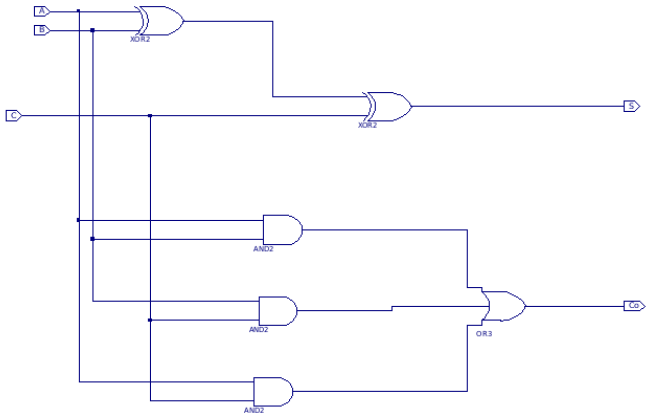

图 2.2 1 位全加器

(d) 逻辑符号图如下：

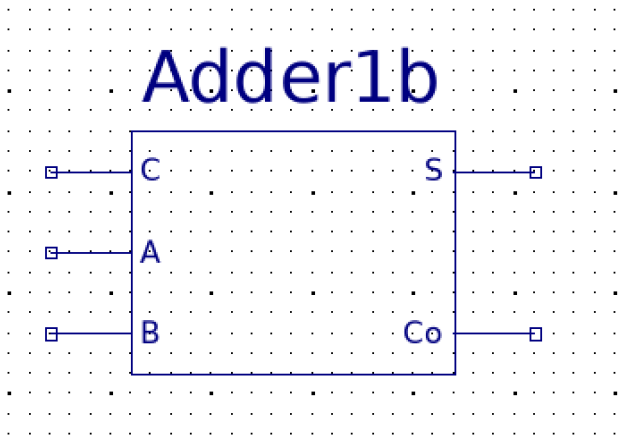

图 2.3 1 位全加器逻辑符号图

(e) Verilog 代码如下：

```
module adder_1bit(
    input wire a, b, ci,
    output wire s, co);
```

```
        and m0(c1,a,b);
        and m1(c2,b,ci);
        and m2(c3,a,ci);
        xor m3(s1,a,b);
        xor m4(s,s1,ci);
        or m5(co,c1,c2,c3);
endmodule
```

2. 多位串行进位加法器

(a) 由一位全加器将进位串接构成，低位进位 C_0 为 0，Ci 为高进位输出

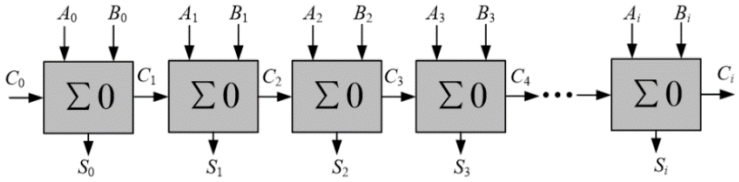

图 2.4 多位串行进位加法器

(b) 电路图如下：

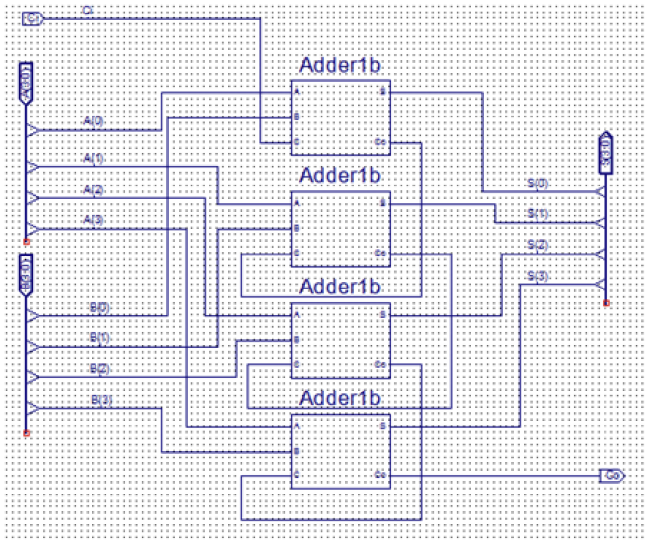

图 2.5 多为串行进位加法器

(c) 逻辑符号表达式如下：

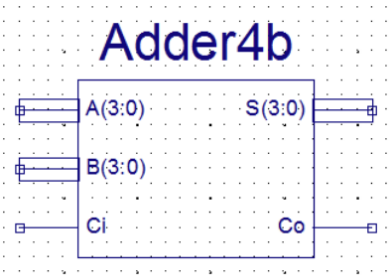

图 2.6 逻辑符号图

(d) 对应 Verilog 代码如下：

```
module adder_4bit(FA, FB, Cin, Sum, Cout);
    parameter SIZE = 4;
    input [SIZE-1:0] FA, FB;
    output [SIZE-1:0] Sum;
    input Cin;
    output Cout;
    wire [1:SIZE-1] Temp;
    adder_1bit
    adder_1(FA[0], FB[0], Cin, Sum[0], Temp[1]),
    adder_2(FA[1], FB[1], Temp[1], Sum[1], Temp[2]),
    adder_3(FA[2], FB[2], Temp[2], Sum[2], Temp[3]),
    adder_4(FA[3], FB[3], Temp[3], Sum[3], Cout);
endmodule
```

3. 1 位加减法器

(a) 用负数补码加法实现，减数当作负数求补码，共用加法器，用“异或”门控制求反，低位进位 C_0 为 1

(b) 电路图如下：

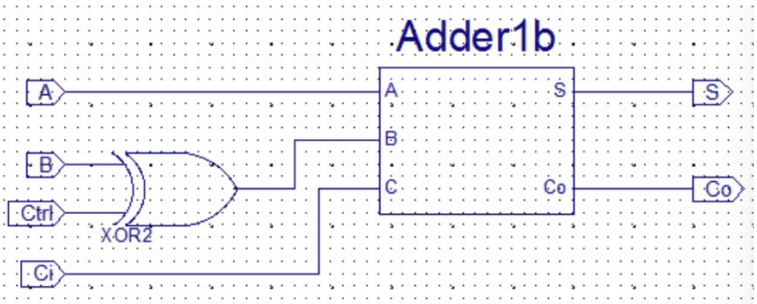

图 2.7 1 位加减法器

(c) 逻辑符号图如下：

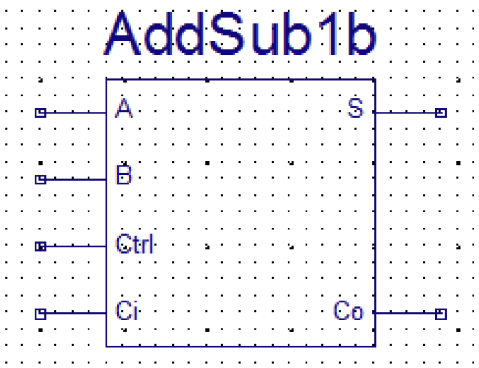

图 2.8 逻辑符号图

4. 多位串行进位全减器

(a) 用负数补码加法实现，减数当作负数求补码，共用加法器，用“异或”门控制求反，低位进位 C_0 为 1

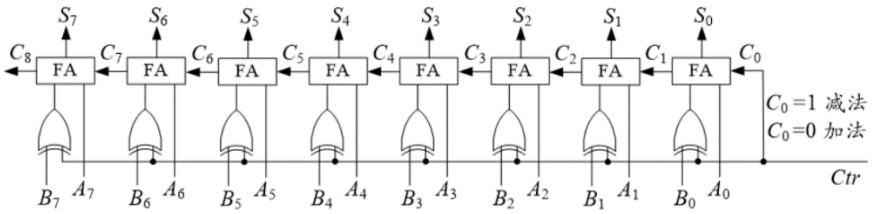

图 2.9 多位串行进位全减器

(b) 4 位加减法器电路图如下：

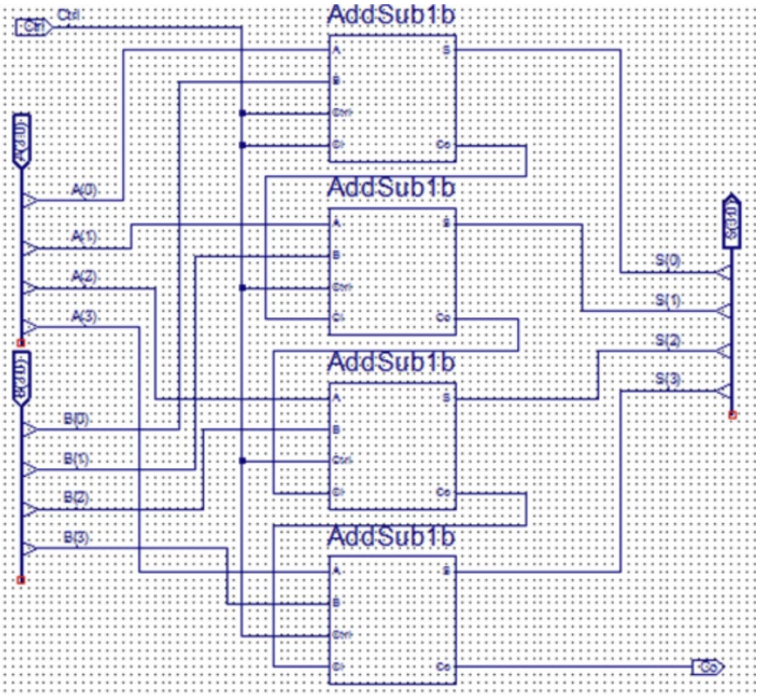

图 2.10 4 位加减法器

(c) 对应逻辑符号图如下：

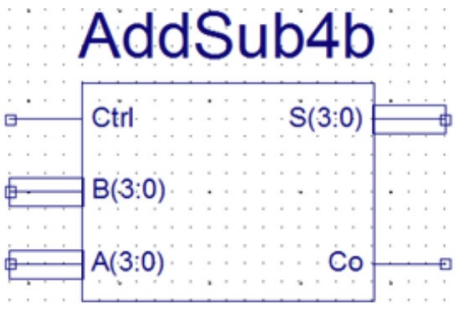

图 2.11 逻辑符号图

5. 4 位 ALU

(a)算术逻辑单元（Arithmetic logical Unit）是中央处理器 (CPU) 的执行单元，是所有中央处理器的核心组成部分，由“And Gate”（与门）和“Or Gate”（或门） 构成的算术逻辑单元，主要功能是进行二元的算术运算。其功能定义如下：

- 两个 4 位操作数 A(3:0)，B(3:0)
- S(1:0) 是 ALU 的功能选择引脚，分别选择选择加、减、与、或操作

  S(1 : 0) = 00C = A + B

  S(1 : 0) = 01C = A − B

  S(1 : 0) = 10C = A&B

  S(1 : 0) = 11C = A|B

(b)ALU 计算得到进位 Co 和结果 C(3:0)myAnd2b4、myOr2b4 分别是 4 位 2 输入与门和 4 位 2 输入或门，其电路图如下：

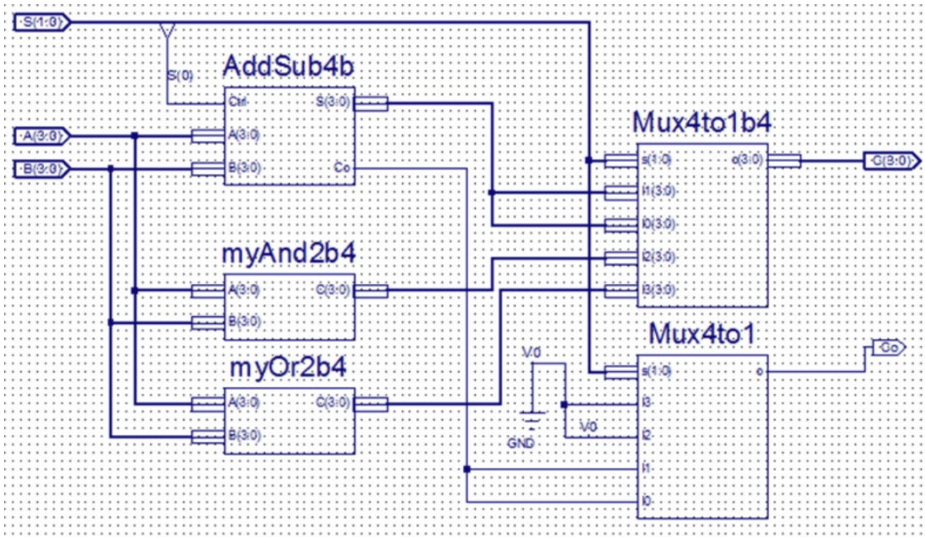

图 2.12 4 位 ALU 电路图

6. 更新 CreateNumber 按键数据输入模块

在实验 7 基础上，更新 CreateNumber 模块。4 位按键输入改变本位数值，4 位开关控制加/减操作，采用 AddSub4b 模块实现 4 位二进制数的自增/自减。

7. 按键去抖原理

(a) 抖动原因：按键按下或放开时，存在机械震动。抖动时间一般在1020ms

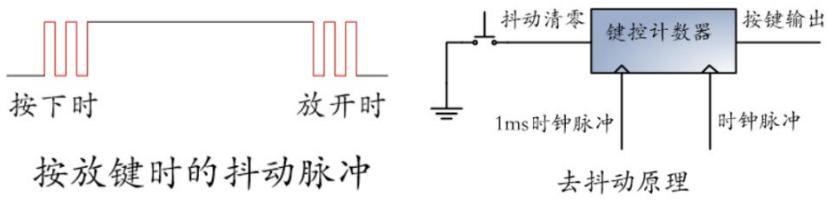

图 2.13 按键去抖

(b) 去抖方法位延迟一段时间后再检测一次，避免机械抖动

三、实验过程和数据记录

3.1 原理图方式设计 4 位加减法器

1. 1 位加法器和 1 位加减法器

(a)在 ISE 中点击 File 选项卡，点击 New Project，工程名为 MyAdder

(b)在 Sources 窗口中右键选择 New Sources 新建源文件向导中选择源文件类型为 Schematic，输入文件名 AddSub1b，勾选 Add to Project

(c)使用 Symbols 和 Schematic Editor 输入原理图如下：

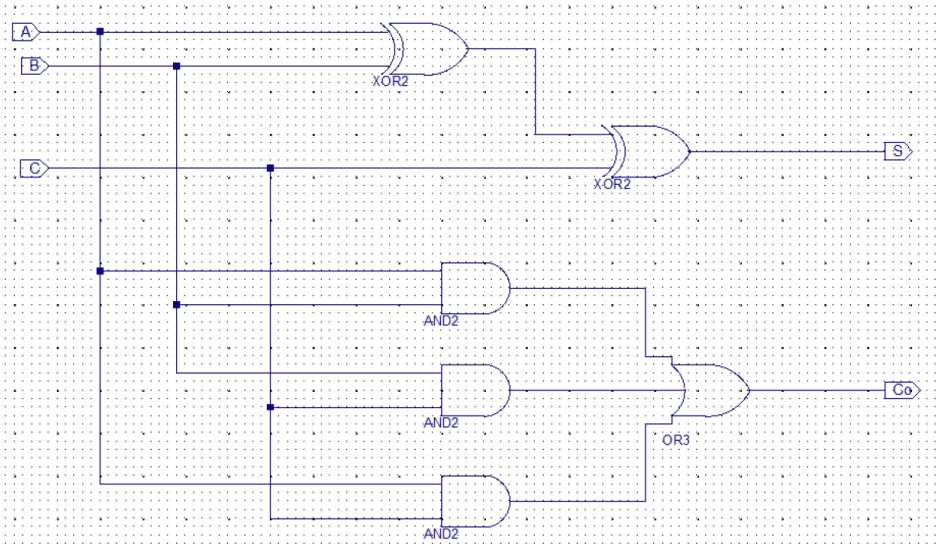

图 3.1 1 位加法器

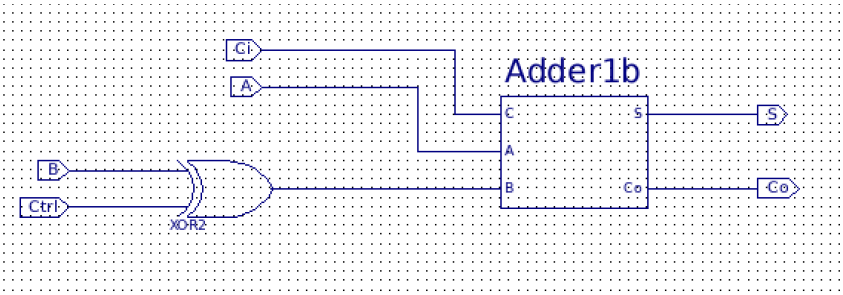

图 3.2 1 位加减法器

(d)新建仿真文件，输入以下仿真代码，对 1 位加减法器进行仿真：

```
`timescale 1ns / 1ps
module AddSub1b_AddSub1b_sch_tb();
    reg Ci;
    reg A;
    reg B;
    reg Ctrl;
    wire S;
    wire Co;
   AddSub1b UUT (
        .Ci(Ci),
        .A(A),
        .S(S),
        .Co(Co),
        .B(B),
        .Ctrl(Ctrl)
   );
    initial begin
        A=0;
        B=0;
        Ci=0;
        Ctrl=0;
        #50;
        A=0;
        B=1;
        #50
        A=1;
        B=0;
        #50
        A=1;
        B=1;
        #50
        A=0;
        B=0;
```

```
        Ctrl=1;
        Ci=1;
        #50
        A=0;
        B=1;
        #50
        A=1;
        B=0;
        #50
        A=1;
        B=1;
    end
endmodule
```

(e)得到仿真波形如下：

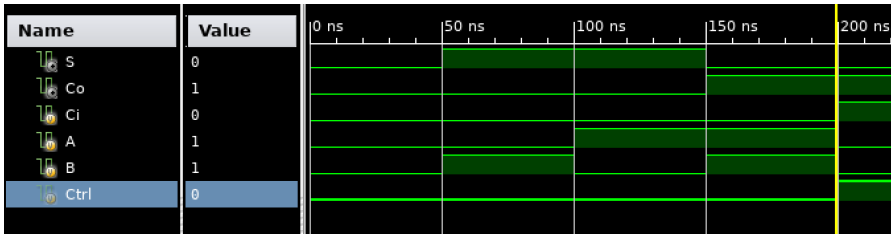

图 3.3 加法

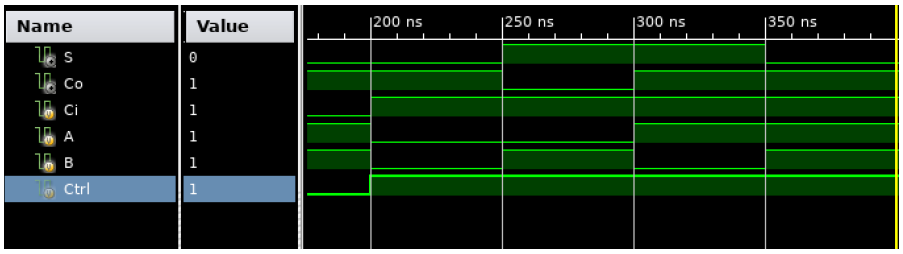

图 3.4 减法

2. 4 位加减法器设计

(a)在 Sources 窗口中右键选择 New Sources 新建源文件向导中选择源文件类型为 Schematic，输入文件名 AddSub4b，勾选 Add to Project

(b)右键设为“Set as Top Module”

(c)使用 Symbols 和 Schematic Editor 输入原理图如下：

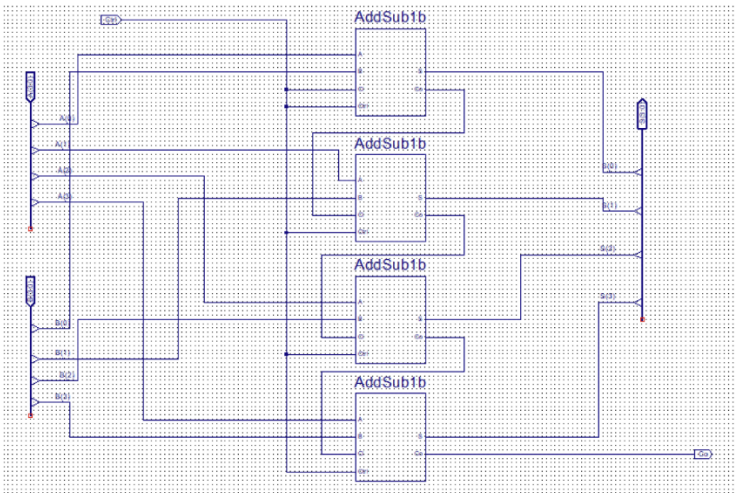

图 3.5 4 位加减法器

(d)新建仿真文件，输入以下仿真代码，对 4 位加减法器进行仿真：

```
`timescale 1ns / 1ps
module AddSub4b_AddSub4b_sch_tb();
   reg [3:0] A;
   reg Ctrl;
   reg [3:0] B;
   wire [3:0] S;
   wire Co;
   AddSub4b UUT (
        .A(A),
        .S(S),
        .Co(Co),
        .Ctrl(Ctrl),
        .B(B));
    initial begin
        Ctrl = 0;
        A=4'b1000;
        B=4'b0001;
        #50
        A=4'b1000;
        B=4'b0010;
        #50
        A=4'b1000;
        B=4'b0011;
        #50
        A=4'b1000;
        B=4'b0100;
        #50
        A=4'b1000;
        B=4'b0101;
```

```
        #50
        A=4'b1000;
        B=4'b0111;
        #50
        A=4'b1000;
        B=4'b1000;
        #50
        A=4'b1000;
        B=4'b1101;
        #50
        Ctrl = 1;
        A=4'b1000;
        B=4'b0001;
        #50
        A=4'b1000;
        B=4'b0010;
        #50
        A=4'b1000;
        B=4'b0011;
        #50
        A=4'b1000;
        B=4'b0100;
        #50
        A=4'b1000;
        B=4'b0101;
        #50
        A=4'b1000;
        B=4'b0111;
        #50
        A=4'b1000;
        B=4'b1000;
        #50
        A=4'b1000;
        B=4'b1101;
        end
endmodule
```

(e)得到仿真波形如下：

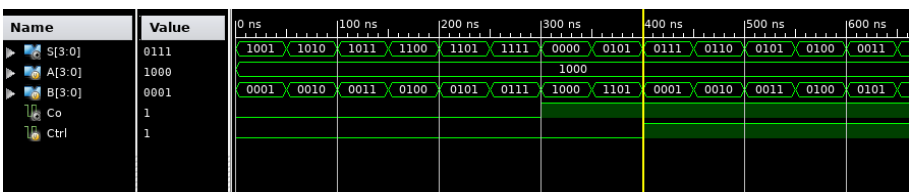

图 3.3 4 位加减器仿真波形

3.2 4 位 ALU 设计

1. 4 位 ALU 电路设计

(a)新建工程 MyALU，Top Level Source Type 用 HDL

(b)新建 Verilog 源代码文件 Top，右键设为“Set as Top Module”

(c)新建源文件 myALU，绘制电路图如下：

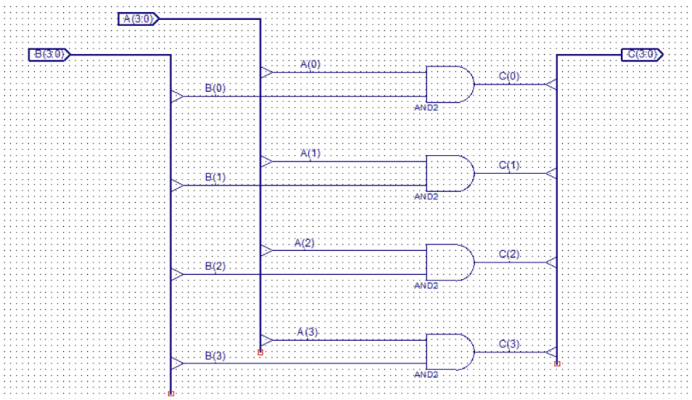

图 3.4 4 位与门 myAnd2b4

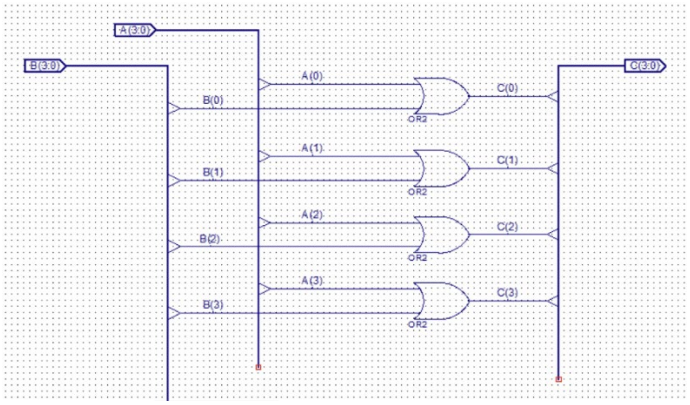

图 3.5 4 位或门 myOr2b4

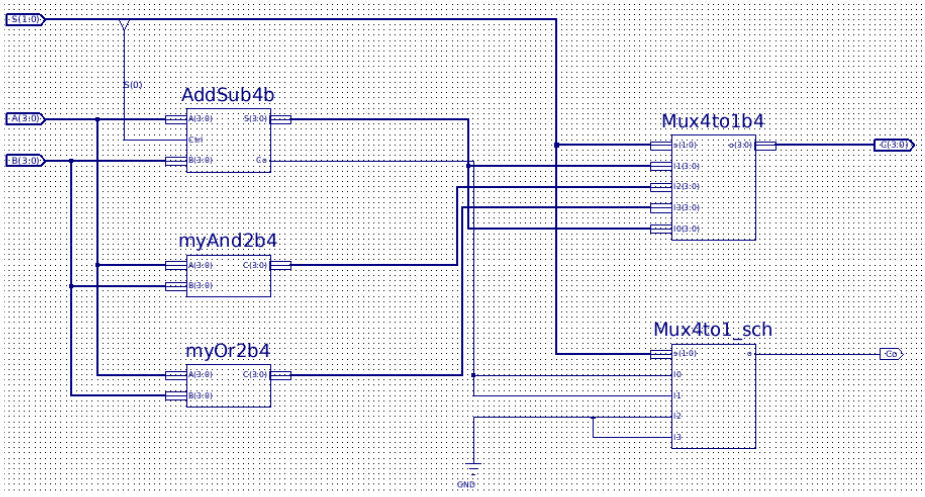

图 3.6 myALU

(d)建立仿真约束文件，输入以下仿真代码：

```
`timescale 1ns / 1ps
module myALU_myALU_sch_tb();
   reg [1:0] S;
   reg [3:0] A;
   reg [3:0] B;
   wire [3:0] C;
   wire Co;
   myALU UUT (
        .S(S),
        .A(A),
        .B(B),
        .C(C),
        .Co(Co)
   );
    initial begin
        S=0;
        A=4'b1010;
        B=0111;
        #50
        B=4'b0011;
        #50
        S=1;
        A=4'b0101;
        B=4'b1010;
        #50
        A=4'b1101;
        B=4'b0010;
        #50
        S=2;
        #50
        A=4'b1001;
        B=4'b0110;
        #50
        S=3;
        A=4'b1111;
        B=4'b1110;
        #50
        A=4'b0011;
        B=4'b0010;
        #50
        S=0;
        A=0;
        B=0;
```

```
    end
endmodule
```

(e)得到如下仿真波形：

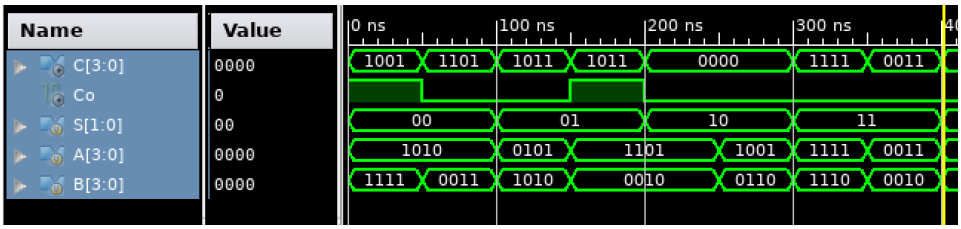

图 3.7 ALU 仿真波形

2. 顶层模块 Top 设计

(a) 实例化 pbdebounce 模块对 2 个按键进行去抖，代码如下：

```
module pbdebounce(
    input wire clk_1ms,
    input wire button,
    output reg pbreg
    );
    reg [7:0] pbshift;
    always@(posedge clk_1ms) begin
        pbshift=pbshift<<1;
        pbshift[0]=button;
        if (pbshift==8'b0)
            pbreg=0;
        if (pbshift==8'hFF)
            pbreg=1;
    end
endmodule
```

(b) 实例化 AddSub4b 模块实现 4 位加减法

(c) 实例化 clkdiv 模块，提供 1ms 时钟

(d) 用 num[3:0] 表示 A，用 num[7:4] 表示 B

(e) 实例化 CreateNumber 模块，用 2 个按键对 num[7:4]、num[3:0] 自增或自减，代码如下：

```
`timescale 1ns / 1ps
module CreateNumber(
    input wire [3:0] btn,
    input wire [3:0] sw,
    output reg [15:0] num
    );
    wire [3:0] A1,B1,C1,D1;
```

```
    initial num <= 16'b1010_1011_1100_1101; //display"AbCd"

    AddSub4b a1(.A(num[3:0]),.B(4'b001),.Ctrl(sw[0]),.S(A1));
    AddSub4b a2(.A(num[7:4]),.B(4'b001),.Ctrl(sw[1]),.S(B1));
    AddSub4b a3(.A(num[11:8]),.B(4'b001),.Ctrl(sw[2]),.S(C1));
     AddSub4b  a4(.A(num[15:12]),.B(4'b001),.Ctrl(sw[3]),.S(D1));

    always@ (posedge btn[0]) num[ 3: 0]<= A1;
    always@ (posedge btn[1]) num[ 7: 4]<= B1;
    always@ (posedge btn[2]) num[11: 8]<= C1;
    always@ (posedge btn[3]) num[15:12]<= D1;

endmodule
```

(f) 实例化 DispNum 模块，显示 A、B、CO、C

(g) 在 Lab7 中，已经完成了 disp_num、clkdiv，直接添加进入 MyALU 项目中即可使用，最后在 top.v 中输入以下代码：

```
`timescale 1ns / 1ps
module top(
    input wire clk,
    input wire [1:0]BTN,
    input wire [1:0]SW1,
    input wire [1:0]SW2,
    output wire [3:0]AN,
    output wire [7:0]SEGMENT,
    output wire BTNX4
    );

    wire [15:0] num;
    wire [1:0] btn_out;
    wire [3:0] C;
    wire Co;
    wire [31:0] clk_div;
    wire [15:0] disp_hexs;
    assign disp_hexs[15:12] = num[3:0]; //A
    assign disp_hexs[11:8] = num[7:4];  //B
    assign disp_hexs[7:4] = {3'b000, Co};
    assign disp_hexs[3:0] = C[3:0];

    pbdebounce m0(clk_div[17],BTN[0],btn_out[0]);
    pbdebounce m1(clk_div[17],BTN[1],btn_out[1]);
    clkdiv m2(.clk(clk),.rst(1'b0),.clkdiv(clk_div));
    CreateNumber m3(btn_out,SW1,num);
```

```
    myALU
m5(.A(num[3:0]),.B(num[7:4]),.S(SW2),.C(C[3:0]),.Co(Co));

    DispNum_sch
m6(.clk(clk),.HEXS(disp_hexs),.LES(4'b0),.points(4'b0),.RST(1'
b0),.AN(AN),.Segment(SEGMENT));
    assign BTNX4 = 1'b0;    //Enable button inputs
endmodule
```

(h) 模块整体结构如下：

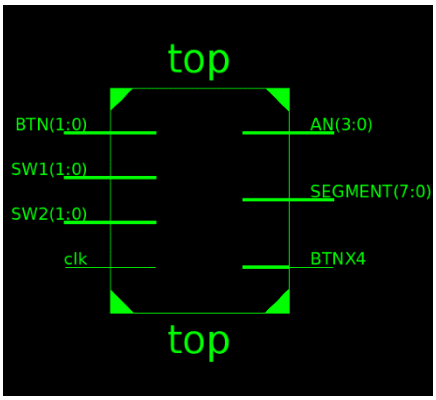

图 3.8 Top 模块

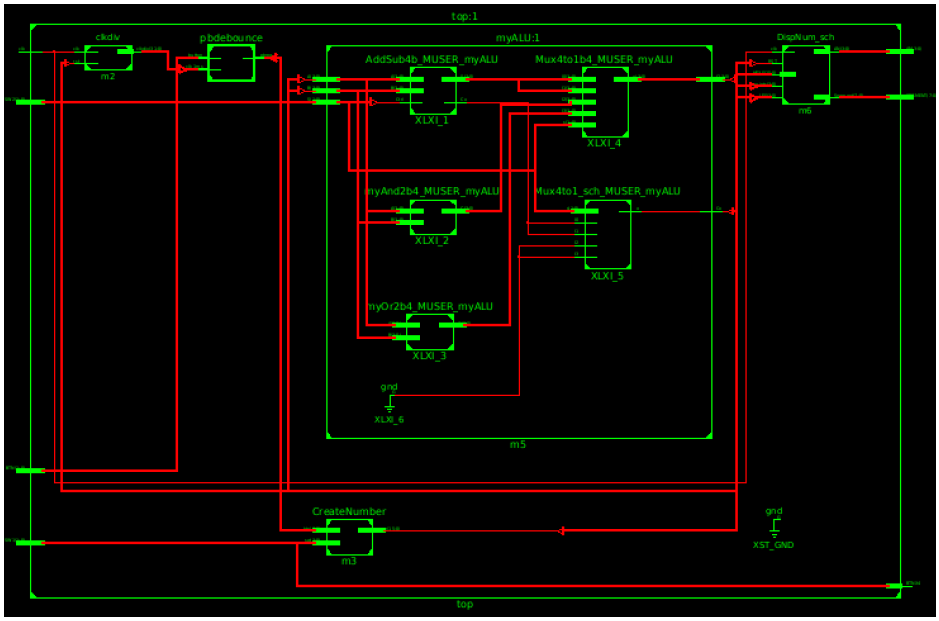

图 3.9 内部结构

3. 建立引脚约束文件

(a)输入

- 时钟：clk
- 按键控制输入：BTN[1]控制 A，BTN[0] 控制 B，关联到

BTNX4Y[0:1]

- 按键加/减方向控制：SW1[1]控制 A，SW1[0]控制 B，关联到 DSW[1:0]

- ALU 运算控制：SW2[1:0],00-加，01-减，10-与，11-或，关联到 DSW[15:14]

(b)输出

- 数码管 [0]：A - num[3:0]
- 数码管 [1]：B - num[7:4]
- 数码管 [2]: Co - Co
- 数码管 [3]: C -C
- BTNX4：按键使能输出

(c)K7 文件如下：

```
NET "clk" LOC = AC18 | IOSTANDARD = LVCMOS18;
NET "SW1[1]" LOC = AA10 | IOSTANDARD = LVCMOS15;
NET "SW1[0]" LOC = AB10 | IOSTANDARD = LVCMOS15;
NET "SW2[1]" LOC = AA12 | IOSTANDARD = LVCMOS15;
NET "SW2[0]" LOC = AA13 | IOSTANDARD = LVCMOS15;

NET "SEGMENT[0]"LOC = AB22 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[1]" LOC = AD24 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[2]" LOC = AD23 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[3]" LOC = Y21 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[4]" LOC = W20 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[5]" LOC = AC24 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[6]" LOC = AC23 | IOSTANDARD = LVCMOS33;
NET "SEGMENT[7]" LOC = AA22 | IOSTANDARD = LVCMOS33;

NET "AN[0]" LOC = AD21 | IOSTANDARD = LVCMOS33;
NET "AN[1]" LOC = AC21 | IOSTANDARD = LVCMOS33;
NET "AN[2]" LOC = AB21 | IOSTANDARD = LVCMOS33;
NET "AN[3]" LOC = AC22 | IOSTANDARD = LVCMOS33;

NET "btn[0]" LOC = V14 | IOSTANDARD = LVCMOS18;
NET "btn[1]" LOC = W14 | IOSTANDARD = LVCMOS18;
NET "BTNX4" LOC = W16 | IOSTANDARD = LVCMOS18;
```

4. **下载到 SWORD 板上进行验证**

(a)点击 Generate Programming Files，通过后点击 Configure Target

Device -> Manage Configuration Project。

(b)根据电路原理图和引脚约束文件

开关“AA10”和“AB10”控制自增自减，开关“AA12”和“AA13”控制运算方式，00-加，01-减，10-与，11-或

实验结果如下：

图 3.10 初始状态

- **验证 A 自增自减功能**

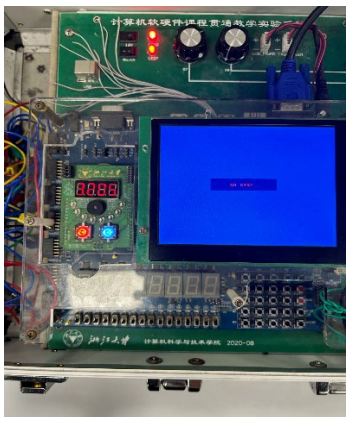

图 3.11 A 自增

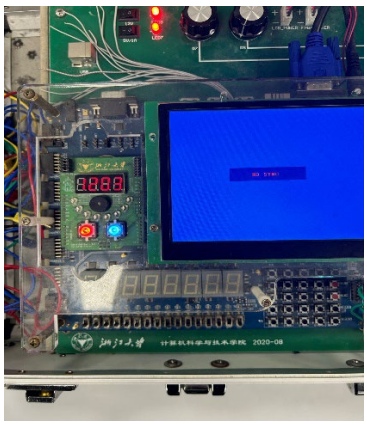

图 3.12 A 自减

- **验证 B 的自增自减功能**

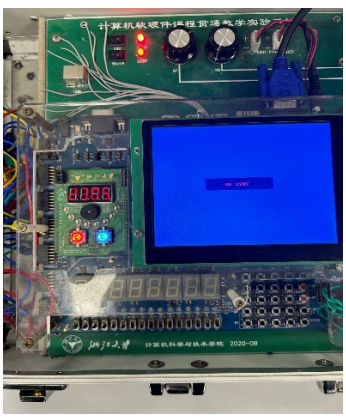

图 3.13 B 自增

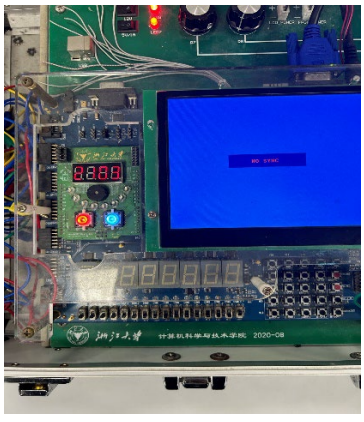

图 3.14 B 自减

- 验证加法功能

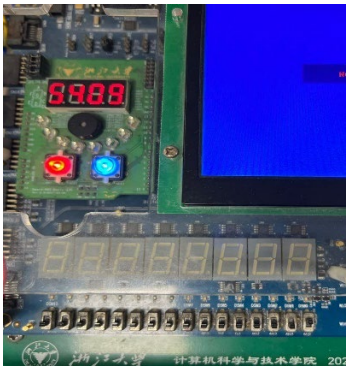

图 3.15

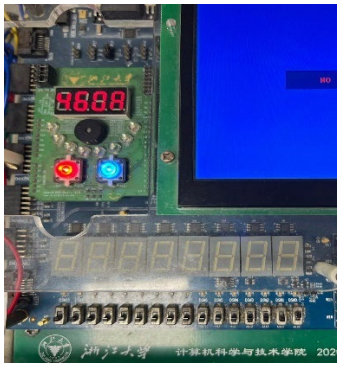

图 3.16

- 验证减法功能

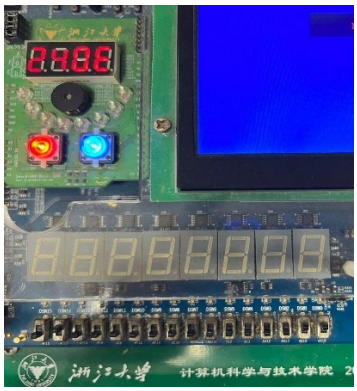

图 3.17

图 3.18

- 验证与功能

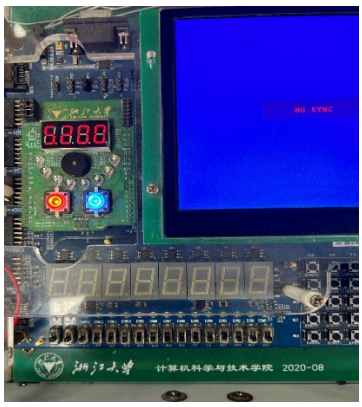

图 3.19

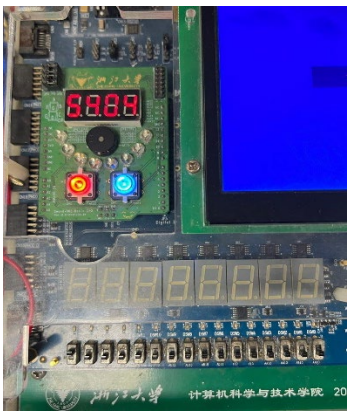

图 3.20

- 验证或功能

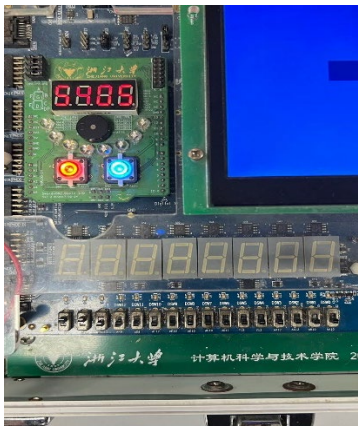

图 3.20

图 3.21

四、实验结果分析

本次实验从 ALU 设计出发，通过变量编码器的应用，依次完成硬件电路图设计、仿真模拟、开发板测试三个步骤，实验结果符合期望。但是遇到了如下问题：

Verilog 代码中的 initial 语句通常用于仿真环境，而在实际的硬件上，其行为可能会有所不同。在实际硬件上，initial 语句只在电路上电时执行一次，用于初始化模块内的寄存器或变量。然而，在实际的硬件逻辑中，并不能保证 initial 语句的执行顺序和仿真环境完全一致。

在实验板上使用 initial 语句将数字初始化为 ABCD，但实际上观察到的初始数字为 5d3b。

在询问助教老师后，这可能是由于硬件逻辑中对 initial 语句的执行时间和顺序造成的。在硬件中，由于各个部分的电路可能以不同的速度启动，导致 initial 语句的执行结果与仿真环境中的预期结果不同。

此外，在进行 1 位加减法器的仿真时，观察到在 Ctrl 为 0 时，加法操作表现正常，即 0+0=0，0+1=1，1+0=1，1+1=0（同时产生进位）。将 Ctrl 置为 1 后，通过将 Ci 同时置为 1，使得 B 在取反后加 1，结果也符合要求：0-0=0，0-1=1，1-0=1，1-1=0。对于减法操作，Co 的含义与加法相反，Co 为 0 表示有借位，Co 为 1 表示没有借位。

在对 4 位加减法器的仿真中，观察到当 Ctrl 为 0 时，Co 始终为 0；而当 Ctrl 置为 1 时，Co 为 1，表示一直没有进位。加法和减法的结果均无误，与预

期相符。通过类似的分析方法，可以得出仿真结果符合预期。

五、讨论与心得

Lab89 的实验原理图以及流程相对复杂，本次实验的难点主要在于绘制复杂的电路图，以及将不同模块整合，如果其中一个出错则整体都会有问题，如果在 AddSub4b 模块中嵌套了 AddSub1b 模块，而 AddSub1b 模块中又包含了 Adder1b 模块，那么在将 AddSub4b.sch 和 AddSub4b.sym 移动到项目目录下并添加源文件时，需要同时将 AddSub1b 和 Adder1b 模块也一并添加到项目中。

因此在进行实验时，要确保将所有相关的器件都添加到项目中，以避免在设计阶段出现错误。其他原理和概念都已经学过，在电路图的绘制和仿真过程中，细心和耐心是非常重要的，确保一切都按照预期进行，以获得准确的实验结果。同时感谢助教老师以及同组同学的帮助！