

# 浙江大学

## 本科实验报告

课程名称:	数字逻辑设计
姓 名:	王浩雄
学 院:	竺可桢学院
系:	混合班
专 业:	计算机科学与技术
学 号:	3230106032
指导教师:	马德

2025 年 4 月 8 日

# 浙江大学实验报告

课程名称: 数字逻辑设计 实验类型: \_\_\_\_\_  
实验项目名称: 加法器、加减法器和 ALU 基本原理与设计  
学生姓名: 王浩雄 专业: 混合班 学号: 3230106032  
同组学生姓名: 无 指导老师: 马德  
实验地点: 紫金港东 4-509 实验日期: 2025 年 4 月 8 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

- ① 设计 4 位加减法器
- ② 实现 4 位 ALU 及应用设计

## 三、 主要仪器设备

- ① 装有 Vivado 2024.2 Enterprise 的计算机 1 台
- ② SWORD 开发板 1 套

## 四、 操作方法与实验步骤

### 1、1 位加减法器设计与验证

① 在 Vivado 中新建文件 addsub\_1b.v，结合设计要求和真值表，编写硬件描述代码如下：

```
module addsub_1b(  
    input wire A, B, C, Ctrl,  
    output wire S, Co  
);  
  
wire c1, c2, c3, s1, bc;  
  
and m0(c1,A,bc);  
and m1(c2,bc,C);  
and m2(c3,A,C);  
xor m3(s1,A,bc);  
xor m4(S,s1,C);  
or m5(Co,c1,c2,c3);  
xor m6(bc, Ctrl, B);  
  
endmodule
```

② 新建仿真激励文件 addsub\_1b\_tb.v 如下：

```
module addsub_1b_tb;  
    reg A, B, C, Ctrl;  
    wire S, Co;  
  
    addsub_1b uut (.A(A),.B(B),.C(C),.Ctrl(Ctrl),.S(S),.Co(Co));  
  
    initial begin  
        // 测试加法  
        Ctrl = 0; A=0; B=0; C=0;  
        #10 A=0; B=0; C=1;  
        #10 A=0; B=1; C=0;  
        #10 A=0; B=1; C=1;  
        #10 A=1; B=0; C=0;  
        #10 A=1; B=0; C=1;  
        #10 A=1; B=1; C=0;  
        #10 A=1; B=1; C=1;  
    end  
endmodule
```

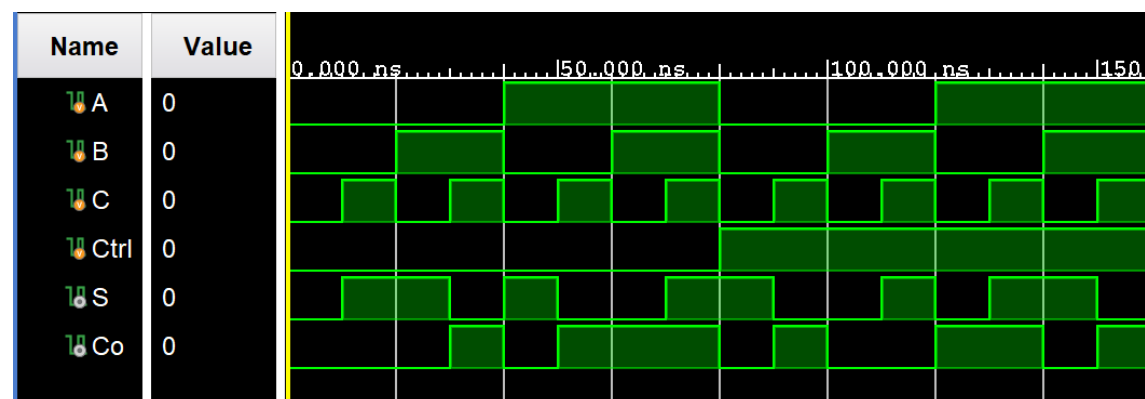
```

// 测试减法
#10 Ctrl = 1; A=0; B=0; C=0;
#10 A=0; B=0; C=1;
#10 A=0; B=1; C=0;
#10 A=0; B=1; C=1;
#10 A=1; B=0; C=0;
#10 A=1; B=0; C=1;
#10 A=1; B=1; C=0;
#10 A=1; B=1; C=1;

end
endmodule

```

③ 运行仿真，得到如下的仿真波形。其中，A、B 为两个 1 位操作数，C 为输入的进位标识，Ctrl 为加减法运算标识，S 为输出结果，Co 为输出的进位标识。该波形与预期相符，表明硬件描述代码编写的正确性。



## 2、4 位加减法器设计与验证

① 在 Vivado 中新建文件 addsub\_4b.v，结合设计要求，复用刚刚设计的 1 位加减法器，编写硬件描述代码如下：

```

module addsub_4b(
    input wire [3:0]A,
    input wire [3:0]B,
    input wire Ctrl,
    output wire [3:0]S,
    output wire Co
);

wire [3:0] carry;

```

```

addsub_1b m0
(.A(A[0]), .B(B[0]), .C(Ctrl), .Ctrl(Ctrl), .S(S[0]), .Co(carry[0]));
addsub_1b m1
(.A(A[1]), .B(B[1]), .C(carry[0]), .Ctrl(Ctrl), .S(S[1]), .Co(carry[1]));
addsub_1b m2
(.A(A[2]), .B(B[2]), .C(carry[1]), .Ctrl(Ctrl), .S(S[2]), .Co(carry[2]));
addsub_1b m3
(.A(A[3]), .B(B[3]), .C(carry[2]), .Ctrl(Ctrl), .S(S[3]), .Co(Co));

endmodule

```

② 新建仿真激励文件 addsub\_4b\_tb.v 如下：

```

module addsub_4b_tb;
    reg [3:0] A;
    reg [3:0] B;
    reg Ctrl;

    wire [3:0] S;
    wire Co;

    addsub_4b uut (.A(A), .B(B), .Ctrl(Ctrl), .S(S), .Co(Co));

    initial begin
        // 测试加法
        Ctrl = 0; A=4'b0000; B=4'b0000; // 0 + 0
        #10 A=4'b0001; B=4'b0001; // 1 + 1
        #10 A=4'b0010; B=4'b0011; // 2 + 3
        #10 A=4'b0101; B=4'b0101; // 5 + 5
        #10 A=4'b1000; B=4'b0111; // 8 + 7
        #10 A=4'b1111; B=4'b0001; // 15 + 1
        #10 A=4'b1111; B=4'b1111; // 15 + 15

        // 测试减法
        #10 Ctrl = 1; A=4'b0000; B=4'b0000; // 0 - 0
        #10 A=4'b0001; B=4'b0001; // 1 - 1
        #10 A=4'b0011; B=4'b0010; // 3 - 2
        #10 A=4'b0101; B=4'b0010; // 5 - 2
        #10 A=4'b1000; B=4'b1111; // 8 - 15
        #10 A=4'b1111; B=4'b0001; // 15 - 1
        #10 A=4'b1111; B=4'b1111; // 15 - 15
        #10 A=4'b0001; B=4'b0010; // 1 - 2

    end
endmodule

```

③ 运行仿真，得到如下的仿真波形。其中，A、B 为两个 4 位操作数，Ctrl 为加减法运算标识，S 为输出结果，Co 为输出的进位标识。该波形 1 处波形为加法溢出情形，2、3 处波形为减法结果为负情形，它们均与预期相符，表明硬件描述代码编写的正确性。



### 3、4 位 ALU 设计与验证

① 在 Vivado 中新建文件，结合设计要求，复用刚刚设计的 4 位加减法器，又额外设计了 4 位与运算器、4 位或运算器、4 输入多选器，编写硬件描述代码如下：

文件：and\_4b.v

```
module and_4b(  
    input wire [3:0] A,  
    input wire [3:0] B,  
    output wire [3:0] C  
);  
  
assign C = A & B;  
  
endmodule
```

文件：or\_4b.v

```
module or_4b(  
    input wire [3:0] A,  
    input wire [3:0] B,  
    output wire [3:0] C  
);  
  
assign C = A | B;  
  
endmodule
```

文件：ALU\_4b.v

```
module ALU_4b(
    input wire [1:0] S,
    input wire [3:0] A,
    input wire [3:0] B,
    output wire [3:0] C,
    output wire Co
);

wire [3:0] w1;
wire [3:0] w2;
wire [3:0] w3;
wire x1;

addsub_4b addsub_4b(.Ctrl(S[0]),.A(A),.B(B),.S(w1),.Co(x1));
and_4b and_4b(.A(A),.B(B),.C(w2));
or_4b or_4b(.A(A),.B(B),.C(w3));
my_mux4to1_4b mux4to1_4b(.s(S),.I0(w1),.I1(w1),.I2(w2),.I3(w3),.o(C));
my_mux4to1_1b mux4to1_1b(.s(S),.I0(x1),.I1(x1),.I2(0),.I3(0),.o(Co));

endmodule
```

② 新建仿真激励文件 alu\_4b\_tb.v 如下：

```
module ALU_4b_tb;
    reg [1:0] S;
    reg [3:0] A, B;

    wire [3:0] C;
    wire Co;

    ALU_4b uut (.S(S),.A(A),.B(B),.C(C),.Co(Co));

    initial begin

        // 初始值
        S = 2'b00;
        A = 4'b0000;
        B = 4'b0000;

        // 测试加法
        S = 2'b00; A=4'b0000; B=4'b0000; // 0 + 0
        #10 A=4'b0010; B=4'b0011; // 2 + 3
        #10 A=4'b0101; B=4'b0101; // 5 + 5
    end
endmodule
```

```

#10 A=4'b1111; B=4'b1111; // 15 + 15

// 测试减法
#10 S = 2'b01; A=4'b0000; B=4'b0000; // 0 - 0
#10 A=4'b1000; B=4'b0111; // 8 - 7
#10 A=4'b1111; B=4'b1111; // 15 - 15
#10 A=4'b0001; B=4'b0010; // 1 - 2

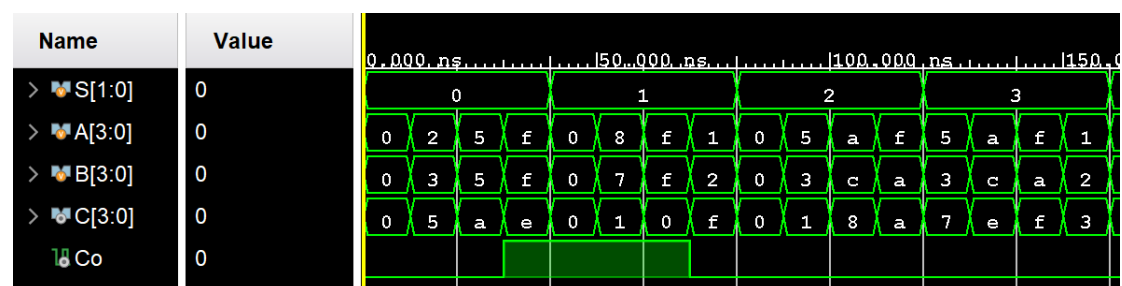
// 测试按位与
#10 S = 2'b10; A=4'b0000; B=4'b0000; // 0 & 0
#10 A=4'b0101; B=4'b0011; // 0101 & 0011 = 0001
#10 A=4'b1010; B=4'b1100; // 1010 & 1100 = 1000
#10 A=4'b1111; B=4'b1010; // 1111 & 1010 = 1010

// 测试按位或
#10 S = 2'b11; A=4'b0101; B=4'b0011; // 0101 | 0011 = 0111
#10 A=4'b1010; B=4'b1100; // 1010 | 1100 = 1110
#10 A=4'b1111; B=4'b1010; // 1111 | 1010 = 1111
#10 A=4'b0001; B=4'b0010; // 0001 | 0010 = 0011

end
endmodule

```

③ 运行仿真，得到如下的仿真波形。其中，A、B 为两个 4 位操作数，C 为运算结果，S 为运算类型选择（0 为加法，1 为减法，2 为与运算，3 为或运算），Co 为输出的进位标识。该波形与预期相符，表明硬件描述代码编写的正确性。



#### 4、顶层模块设计与下板验证

① 在上一次实验中，因为按键过程中会有机械原因导致的信号抖动，所以一次按下按钮数字会跳动多次。我们可以实现一个去抖动模块，当信号稳定一段时间后才更改信号值。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

```

② 复用上次实验中设计的 clk\_div、CreateNum、DispNum 等模块，使用 Verilog 代码实现顶层连接如下：

```

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];
assign disp_hexs[11:8] = num[7:4];
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(btn_out[1:0]),.sw(SW1[1:0]),.num(num));

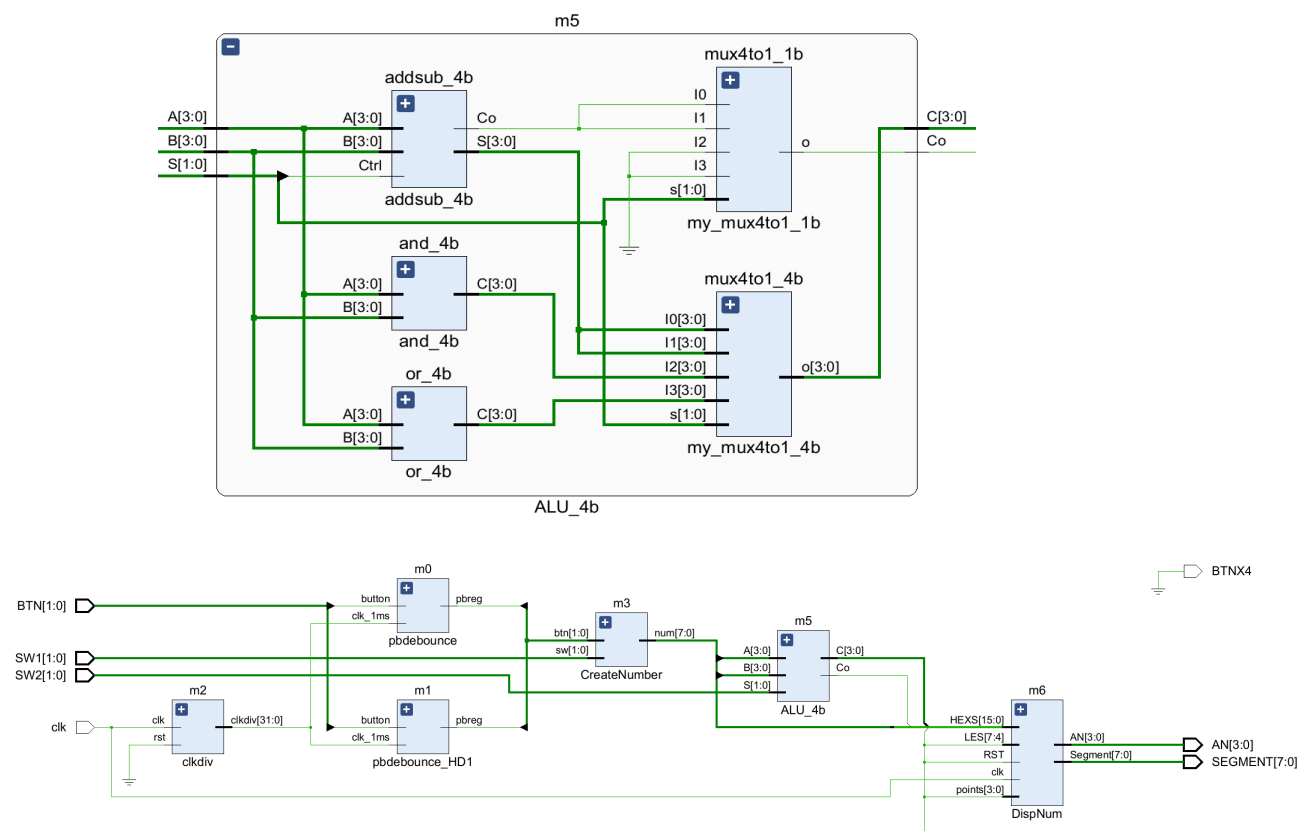
ALU_4b m5(.A(num[3:0]), .B(num[7:4]), .S(SW2[1:0]), .C(C[3:0]), .Co(Co));

DispNum
m6(.clk(clk),.HEXS(disp_hexs),.LES(4'b0),.points(4'b0),.RST(1'b0),.AN(A
N),.Segment(SEGMENT));

assign BTNX4 = 1'b0;

endmodule
```

③ 在 Vivado 中运行 Linter Design, 查看由 Verilog 代码生成的连接图。经检查, 该图与设计要求中提供的连接图逻辑相同。



④ 将 Xilinx ISE 中使用的 .ucf 引脚约束文件修改为 Vivado 的 .xdr 引脚约束文件，并补充完全。

```
set_property PACKAGE_PIN AC18 [get_ports clk]
set_property IOSTANDARD LVCMOS18 [get_ports clk]

set_property PACKAGE_PIN W14 [get_ports {BTN[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {BTN[0]}]

set_property PACKAGE_PIN V14 [get_ports {BTN[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {BTN[1]}]

set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets btn*]

set_property PACKAGE_PIN W16 [get_ports BTN4]
set_property IOSTANDARD LVCMOS18 [get_ports BTN4]

set_property PACKAGE_PIN AB22 [get_ports {SEGMENT[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[0]}]

set_property PACKAGE_PIN AD24 [get_ports {SEGMENT[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[1]}]

set_property PACKAGE_PIN AD23 [get_ports {SEGMENT[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[2]}]

set_property PACKAGE_PIN Y21 [get_ports {SEGMENT[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[3]}]

set_property PACKAGE_PIN W20 [get_ports {SEGMENT[4]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[4]}]

set_property PACKAGE_PIN AC24 [get_ports {SEGMENT[5]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[5]}]

set_property PACKAGE_PIN AC23 [get_ports {SEGMENT[6]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[6]}]

set_property PACKAGE_PIN AA22 [get_ports {SEGMENT[7]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SEGMENT[7]}]

set_property PACKAGE_PIN AD21 [get_ports {AN[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[0]}]
```

```

set_property PACKAGE_PIN AC21 [get_ports {AN[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[1]}]

set_property PACKAGE_PIN AB21 [get_ports {AN[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[2]}]

set_property PACKAGE_PIN AC22 [get_ports {AN[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {AN[3]}]

set_property PACKAGE_PIN AA10 [get_ports {SW1[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW1[0]}]



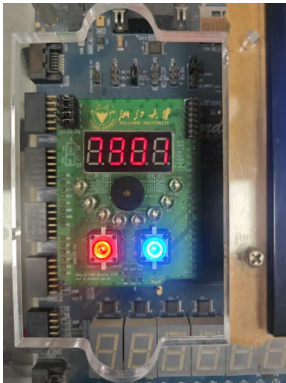
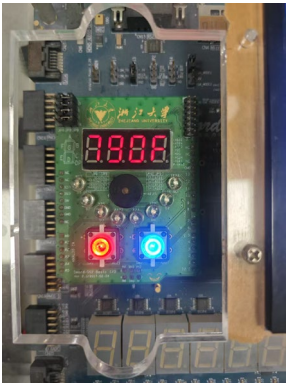
set_property PACKAGE_PIN AB10 [get_ports {SW1[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW1[1]}]

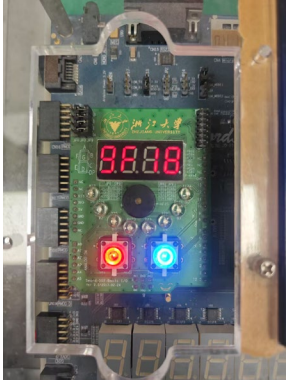
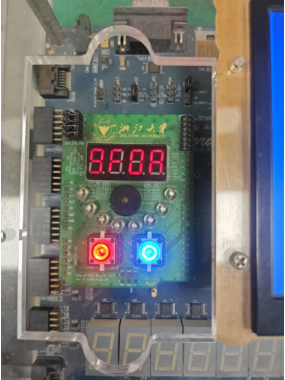
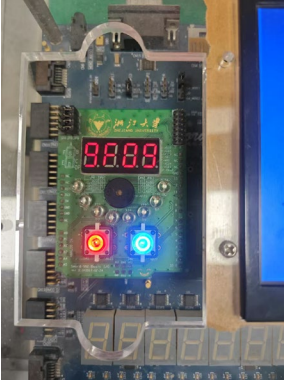


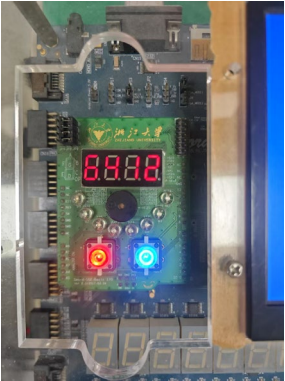
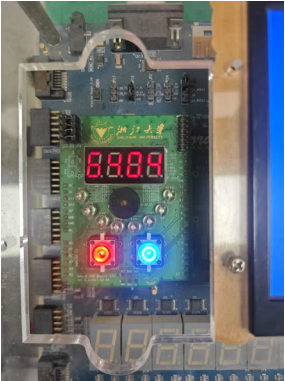
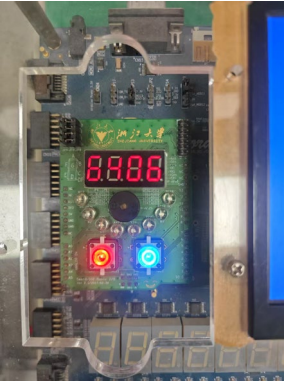
set_property PACKAGE_PIN AA13 [get_ports {SW2[0]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW2[0]}]

set_property PACKAGE_PIN AA12 [get_ports {SW2[1]}]
set_property IOSTANDARD LVCMOS15 [get_ports {SW2[1]}]

```

⑤ 下板，按其功能设计要求进行验证。得到如下的结果。该结果与设计要求和预期相符。

加法运算	减法运算	与运算	或运算
 <p>7+9=10 (溢出)</p>	 <p>7-9=-2 (补码 1110, 显示 E)</p>	 <p>7&amp;9=1</p>	 <p>7 9=F</p>

加法运算	减法运算	与运算	或运算
 $9+F=18$ (溢出)	 $9-F=-6$ (补码 1010, 显示 A)	 $9\&F=9$	 $9 F=F$
 $6+4=A$	 $6-4=2$	 $6\&4=4$	 $6 4=6$

五、 实验结果与分析

(请见上方分析)

六、 讨论、心得

在本次实验中，我使用 Vivado 实现了四位 ALU 的设计与验证。虽然我在《计算机组成》课程中已经详细了解了 ALU 的相关原理，但通过本次实验，我对时钟分频器、去抖动模块的作用和原理有了更深一步的理解。