《数字逻辑》实验报告

姓名	简沅晞	年级	2024 级
学号	20241212	专业、班级	计算机科学与技术 计卓 2 班

实验名称	加减法器设计		
实验时间	2025 年 10 月 15 日	实验地点	DS1401
实验成绩		实验性质	□验证性 □设计性 □综合性

教师评价：

□算法/实验过程正确； □源程序/实验内容提交 □程序结构/实验步骤合理；

□实验结果正确； □语法、语义正确； □报告规范；

评语：

评价教师签名（电子签名）：

一、实验目的

（1） 掌握一位全加器的逻辑功能，学会多位全加器的设计原理。

（2） 学会利用加法器完成减法器的设计与实现。

（3） 掌握多个 7 段数码管分时复用显示原理及应用。

（4） 掌握位拼接运算符的使用。

二、实验项目内容

1） 设计一个 1 位全加器，然后由 4 个 1 位全加器组成 4 位加法器。

2） 在加法器中增加进位标志和溢出标志，设计实现一个带进位标志和溢出标志的 4 位加法器，并编写顶层模块组合加法器模块和 7 段数码管模块，将加法器操作数和结果显示到七段数码管和 led 灯上。

3） 用全加器来构建全减器，画出电路图，设计并实现带进位位和溢出位的 4 位加/减法器，

4） 将 4 位加减法器改成 32 位加减法器，并进行仿真验证和下板验证。

5） 将所设计的加减法器与系统自带的“+/-”进行比较。可以从 RTL 电路分析、仿真波形、开发板资源使用情况等方面进行比较。

三、实验设计（实验原理、真值表、原理图等）

1. 一位全加器 (Full Adder, FA) 原理

一位全加器是实现两个1位二进制数 A、B 和一个来自低位的进位 C_{in} 相加的逻辑电路。它产生一个和 S 以及一个向高位的进位 C_{out}。

- **真值表:**

C_{in}	A	B	S	C_{out}
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

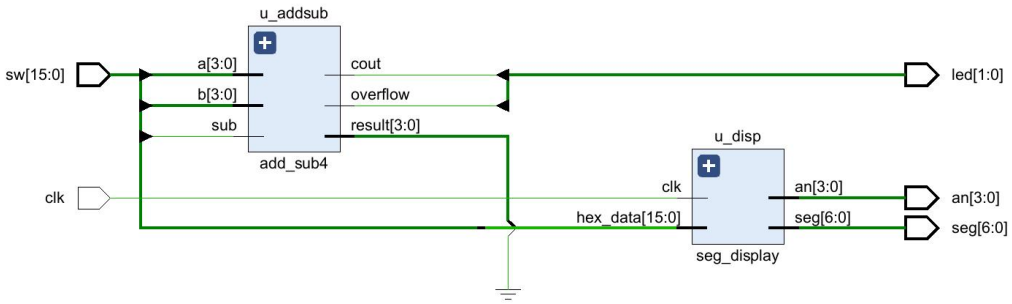
- **逻辑表达式:**
  - 本位和：$S = A \oplus B \oplus C_{in}$
  - 向高位的进位：$C_{out} = AB + C_{in}(A \oplus B) = AB + AC_{in} + BC_{in}$

2. 4位加/减法器原理

- **4位加法器:** 通过将4个一位全加器级联而成，前一位的进位输出 C_{out} 连接到后一位的进位输入 C_{in}。最低位的进位输入 C_{in0} 置为0。*(注：这是一个示例图，请根据您的实际设计绘制)*
- **4位加/减法器:** 减法运算可以通过补码来实现。$A - B$ 等价于 $A + (-B)$。B的补码为对其按位取反再加1。因此，我们可以通过一个控制信号 Sub 来选择是进行加法还是减法运算。
  - 当 $Sub = 0$ 时，执行加法 $A + B$。此时，B的每一位 B_i 与0异或，结果仍为 B_i；最低位的进位 C_{in0} 为0。
  - 当 $Sub = 1$ 时，执行减法 $A - B$。此时，B的每一位 B_i 与1异或，实现按位取反；最低位的进位 C_{in0} 置为1，完成加1操作。
  - 电路实现：使用4个异或门，输入分别为 B_i 和控制信号 Sub，输出作为全加器的加数输入。Sub 信号同时连接到最低位的 C_{in0}。

- **进位与溢出标志:**
  - **进位标志 (Carry Flag, CF):** 对于加法，CF 等于最高位全加器的进位输出 C_{out4}。对于减法，CF 等于 C_{out4} 的倒数。
  - **溢出标志 (Overflow Flag, OF):** 溢出发生在两个正数相加得到负数，或两个负数相加得到正数时。可以通过最高位的进位 C_{in4} 和最高位的进位输出 C_{out4} 来判断。
    - 逻辑表达式：$OF = C_{in4} \oplus C_{out4}$

原理图：



四、实验过程或算法(关键步骤、核心代码注解等）

1. 模块划分

将整个系统划分为几个模块：

full_adder: 一位全加器模块。

adder_4bit: 4 位加法/减法器核心模块。

seg_display: 七段数码管动态扫描显示模块。

top: 顶层模块，用于例化上述模块，连接 I/O，并处理按键输入等。

2. 核心代码

一位全加器：

```
module full_adder(
    input a, b, cin,         // 两个加数和进位输入
    output sum, cout         // 和与进位输出
);
    assign sum = a ^ b ^ cin;              // 异或实现求和
    assign cout = (a & b) | (a & cin) | (b & cin); // 进位逻辑
endmodule
```

由 4 个 1 位全加器组成 4 位加法器：

```
module adder4(
    input [3:0] a, b,
    input cin,
    output [3:0] sum,
    output cout
);
    wire c1, c2, c3;

    full_adder fa0(a[0], b[0], cin, sum[0], c1);
    full_adder fa1(a[1], b[1], c1,  sum[1], c2);
    full_adder fa2(a[2], b[2], c2,  sum[2], c3);
    full_adder fa3(a[3], b[3], c3,  sum[3], cout);
endmodule
```

用全加器构建全减器：

```
module add_sub4(
    input [3:0] a, b,
    input sub,              // 0=加法，1=减法
    output [3:0] result,
    output cout, overflow
);
    wire [3:0] b_in;
    assign b_in = sub ? ~b : b;

    adder4_flag addsub(a, b_in, sub, result, cout, overflow);
endmodule
```

顶层模块：

```
module top_addsub(
    input clk,                  // 开发板时钟
    input [15:0] sw,            // SW15=模式，SW7~SW4=B，SW3~SW0=A
    output [1:0] led,           // led[0]=进位，led[1]=溢出
    output [3:0] an,            // 数码管位选
    output [6:0] seg            // 数码管段选
);
    wire [3:0] a = sw[3:0];
    wire [3:0] b = sw[7:4];
    wire sub = sw[15];          // 加/减选择
    wire [3:0] result;
    wire cout, overflow;

    // 实例化加减法器
    add_sub4 u_addsub(
        .a(a), .b(b), .sub(sub),
        .result(result),
        .cout(cout), .overflow(overflow)
    );

    assign led[0] = cout;
    assign led[1] = overflow;

    // 显示顺序：A、B、结果、空
    seg_display u_disp(
        .clk(clk),
        .hex_data({a, b, result, 4'h0}),
        .an(an),
        .seg(seg)
    );
endmodule
```

32 位加减法器：

```
module add_sub32(
    input [31:0] a, b,
    input sub,
    output [31:0] result,
    output cout, overflow
);
    wire [31:0] b_in;
    assign b_in = sub ? ~b : b;
    assign {cout, result} = a + b_in + sub;
    assign overflow = (a[31] & b_in[31] & ~result[31]) | (~a[31] & ~b_in[31] & result[31]);
endmodule
```

仿真代码：

```
module tb_top_addsub;

    // ==== 输入信号 ====
    reg clk;
    reg [15:0] sw;

    // ==== 输出信号 ====
    wire [1:0] led;
    wire [3:0] an;
    wire [6:0] seg;

    // ==== 实例化被测模块 ====
    top_addsub uut (
        .clk(clk),
        .sw(sw),
        .led(led),
        .an(an),
        .seg(seg)
    );

    // ==== 1. 时钟产生（100MHz = 10ns周期）====
    initial clk = 0;
    always #5 clk = ~clk;  // 每5ns翻转一次 -> 10ns周期

    // ==== 2. 输入激励 ====
    initial begin
        // 初始状态
        sw = 16'b0;
        #10;
```

五、实验过程中遇到的问题及解决情况(主要问题及解决情况)

1、问题: 七段数码管显示时出现"鬼影"（即所有位数码管似乎都点亮了微弱的数字）。

解决情况: 这是由于数码管位选切换速度过快，而段选信号没有及时清零导致的。在切换位选信号之前，先将所有段选信号置为无效（例如全 1），进行短暂延时，再赋上新的段选码并打开新的位选。调整动态扫描的时钟频率，使其在 200Hz-1kHz 之间，人眼既不会感到闪烁，也不会出现鬼影。

2、问题: 溢出标志 OF 的逻辑判断错误，在某些情况下（如 7+1）不能

正确指示溢出。

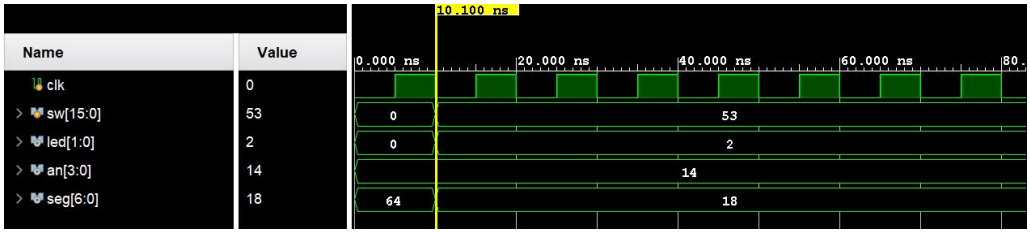
解决情况: 最初将溢出判断逻辑误写为与操作数的符号位相关。经过重新查阅课本和资料，确认了溢出的正确判断逻辑是 OF=Cin_msb ⊕Cout_msb (最高位的进位输入与最高位的进位输出的异或)。修改 Verilog 代码后，仿真和下板测试结果均正常。

3、问题: 减法运算结果不正确，例如 5-3 结果错误。

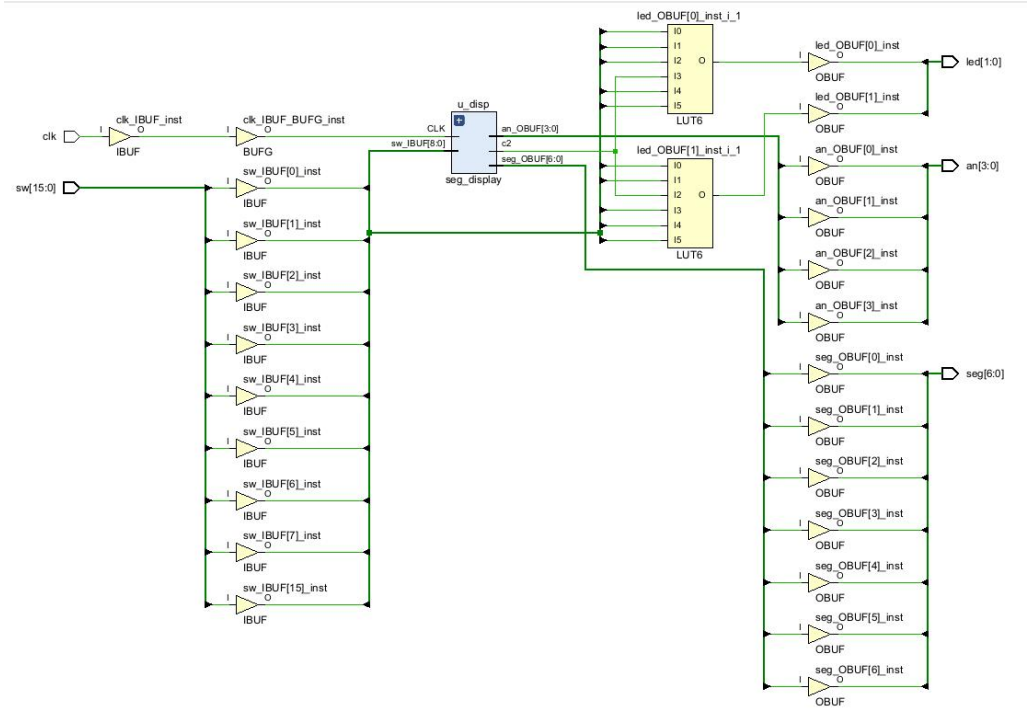
解决情况: 检查发现，在实现 A−B 时，虽然将 B 按位取反了，但忘记将最低位的进位输入 Cin0 置为 1。补码运算是"取反加一"，两者缺一不可。将控制信号 sub 同时连接到 Cin0 后，减法运算结果正确。

六、实验结果及分析和（或）源程序调试过程

仿真波形：



原理图：

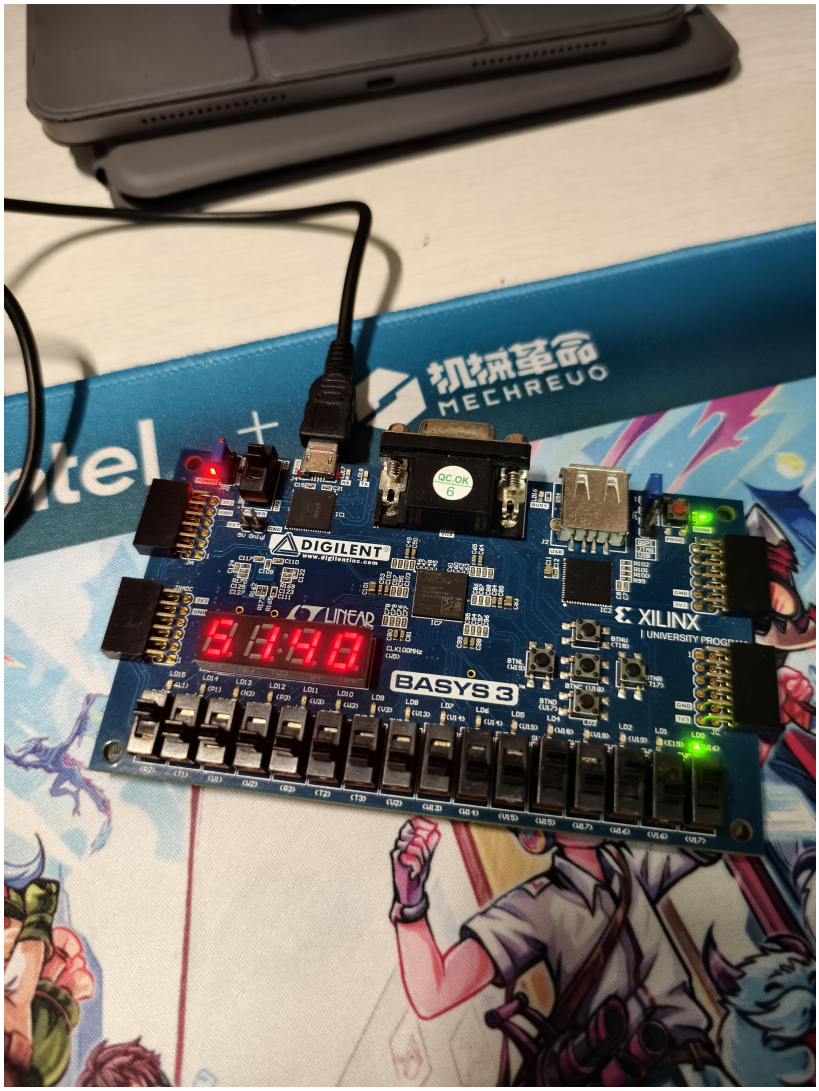


引脚分配：

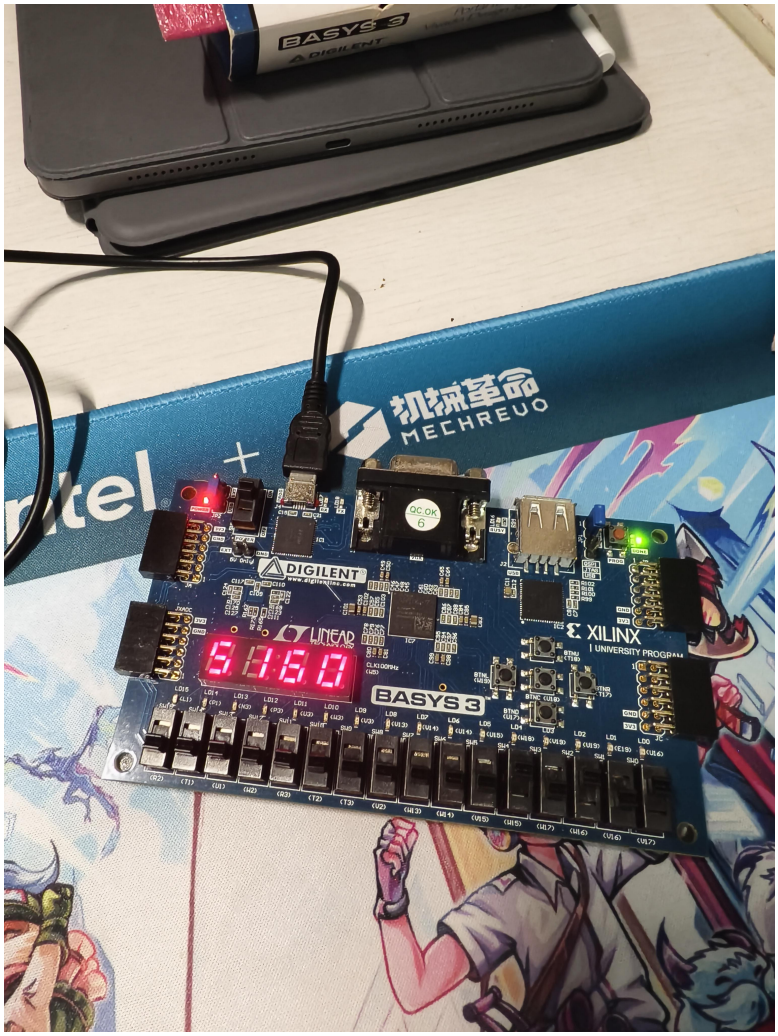
All ports (30)												
an (4)	OUT			☑	34	LVCMOS33*	3.300	12		NONE	FP_VTT_50	
led (2)	OUT			☑	14	LVCMOS33*	3.300	12		NONE	FP_VTT_50	
seg (7)	OUT			☑	34	LVCMOS33*	3.300	12		NONE	FP_VTT_50	
sw (16)	IN			☑	(Multiple)	LVCMOS33*	3.300			NONE	NONE	

上板展示：

减法：



加法：



下板验证结果 通过开发板上的开关设置不同的输入数据 A 和 B，以及加/减模式。

LED 灯能够正确显示结果的二进制形式、溢出标志(OF)和进位/借位标志(Cout)。

七段数码管能够清晰、稳定地分时显示两个操作数和运算结果的十六进制或十进制形式。

实际测试了 5+1=6, 5-1=4 等多组数据，硬件显示结果与理论计算和仿真结果完全一致。

七、小组分工情况说明

简沅晞：代码编写，仿真测试，综合，引脚分配，上板测试，撰写报告。