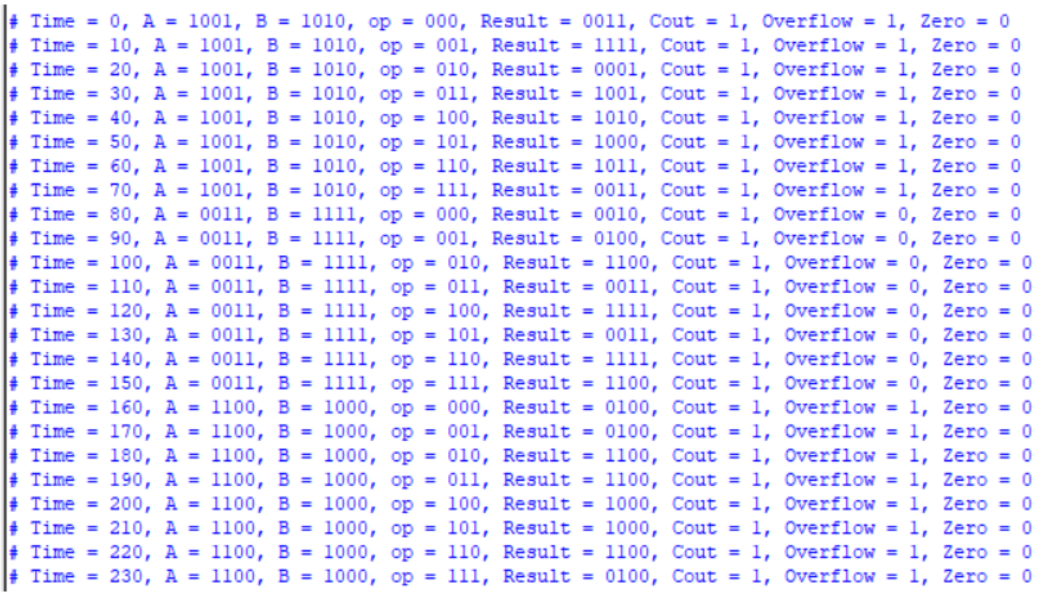
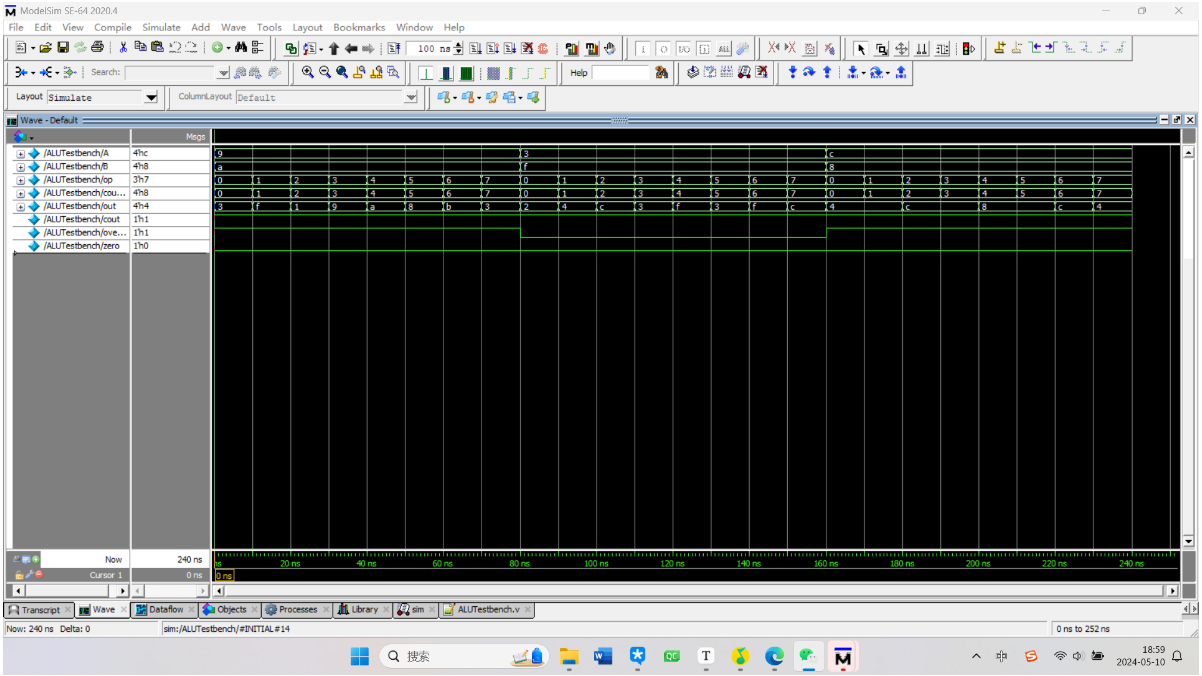
编程题1

$monitor监视：



```
# Time = 0, A = 1001, B = 1010, op = 000, Result = 0011, Cout = 1, Overflow = 1, Zero = 0
# Time = 10, A = 1001, B = 1010, op = 001, Result = 1111, Cout = 1, Overflow = 1, Zero = 0
# Time = 20, A = 1001, B = 1010, op = 010, Result = 0001, Cout = 1, Overflow = 1, Zero = 0
# Time = 30, A = 1001, B = 1010, op = 011, Result = 1001, Cout = 1, Overflow = 1, Zero = 0
# Time = 40, A = 1001, B = 1010, op = 100, Result = 1010, Cout = 1, Overflow = 1, Zero = 0
# Time = 50, A = 1001, B = 1010, op = 101, Result = 1000, Cout = 1, Overflow = 1, Zero = 0
# Time = 60, A = 1001, B = 1010, op = 110, Result = 1011, Cout = 1, Overflow = 1, Zero = 0
# Time = 70, A = 1001, B = 1010, op = 111, Result = 0011, Cout = 1, Overflow = 1, Zero = 0
# Time = 80, A = 0011, B = 1111, op = 000, Result = 0010, Cout = 1, Overflow = 0, Zero = 0
# Time = 90, A = 0011, B = 1111, op = 001, Result = 0100, Cout = 1, Overflow = 0, Zero = 0
# Time = 100, A = 0011, B = 1111, op = 010, Result = 1100, Cout = 1, Overflow = 0, Zero = 0
# Time = 110, A = 0011, B = 1111, op = 011, Result = 0011, Cout = 1, Overflow = 0, Zero = 0
# Time = 120, A = 0011, B = 1111, op = 100, Result = 1111, Cout = 1, Overflow = 0, Zero = 0
# Time = 130, A = 0011, B = 1111, op = 101, Result = 0011, Cout = 1, Overflow = 0, Zero = 0
# Time = 140, A = 0011, B = 1111, op = 110, Result = 1111, Cout = 1, Overflow = 0, Zero = 0
# Time = 150, A = 0011, B = 1111, op = 111, Result = 1100, Cout = 1, Overflow = 0, Zero = 0
# Time = 160, A = 1100, B = 1000, op = 000, Result = 0100, Cout = 1, Overflow = 1, Zero = 0
# Time = 170, A = 1100, B = 1000, op = 001, Result = 0100, Cout = 1, Overflow = 1, Zero = 0
# Time = 180, A = 1100, B = 1000, op = 010, Result = 1100, Cout = 1, Overflow = 1, Zero = 0
# Time = 190, A = 1100, B = 1000, op = 011, Result = 1100, Cout = 1, Overflow = 1, Zero = 0
# Time = 200, A = 1100, B = 1000, op = 100, Result = 1000, Cout = 1, Overflow = 1, Zero = 0
# Time = 210, A = 1100, B = 1000, op = 101, Result = 1000, Cout = 1, Overflow = 1, Zero = 0
# Time = 220, A = 1100, B = 1000, op = 110, Result = 1100, Cout = 1, Overflow = 1, Zero = 0
# Time = 230, A = 1100, B = 1000, op = 111, Result = 0100, Cout = 1, Overflow = 1, Zero = 0
```

波形图：



真值表：

A[3:0]	B[3:0]	cout	overflow	sum[3:0]
0111	0001	0	0	1000
0111	0111	1	0	1110
1000	1000	1	0	0000
0111	1001	0	0	0000

A[3:0]	B[3:0]	cout	overflow	sum[3:0]
0001	1111	0	1	0000
1000	1000	1	1	0000
1000	0001	0	0	1001
0001	0001	0	0	0010

逻辑方程：

$$cout = carry, \{carry, sum\} = A + B;$$

$$\mathrm{overflow} = (\overline{A[3]} \cdot \overline{B[3]} \cdot \mathrm{sum}[3]) + (A[3] \cdot B[3] \cdot \overline{sum[3]})$$

代码：

ALU.v

```
module ALU(output [3:0] out, output cout, output overflow, output zero,
           input [3:0] A, B,
           input [2:0] op);

   // 内部连线用于计算和、进位和相等性
   wire [3:0] sum;
   wire carry;
   wire equal;

   // 输出零标志指示结果是否为零
   // 当输出的所有位都为零时，该标志为真
   assign zero = (out == 4'b0000);

   // 输出进位标志直接赋值为进位
   assign cout = carry;

   // 根据加法或减法的结果计算溢出标志
   // 如果两个操作数都是正数且结果为负数，
   // 或者两个操作数都是负数且结果为正数，则会发生溢出
   assign overflow = ~A[3] & ~B[3] & sum[3] | A[3] & B[3] & ~sum[3];

   // 输出 out[3:0] 根据操作码（op）确定
   // 每个操作根据 op 的值分配到相应的输出
   // A + B
   // A - B
   // B - A
   // 传递 A
   // 传递 B
   // A AND B
   // A OR B
   // A XOR B
   assign out[3:0] = (op == 3'b000) ? A + B :
                     (op == 3'b001) ? A - B :
                     (op == 3'b010) ? B - A :
                     (op == 3'b011) ? A :
```

```
                     (op == 3'b100) ? B :
                     (op == 3'b101) ? (A & B) :
                     (op == 3'b110) ? (A | B) :
                                      (A ^ B);

   // 计算溢出标志所需的和和进位
   assign {carry, sum} = A + B;

endmodule
```

ALUTestbench.v

```
module ALUTestbench;

    reg [3:0] A, B; // 声明两个4位宽的输入寄存器 A 和 B
    reg [2:0] op; // 声明一个3位宽的操作码寄存器 op
    reg [3:0] counter; // 声明一个用于循环的计数器

    wire [3:0] out; // 输出结果（4位）
    wire cout, overflow, zero; // 声明进位输出、溢出和零标志信号

    // 实例化ALU模块
    ALU alu(out, cout, overflow, zero, A, B, op);

    // 测试过程
    initial begin
        // 打印头部，包括时间、输入和输出
        $monitor("Time = %0t, A = %b, B = %b, op = %b, Result = %b, Cout = %b, Overflow = %b, Zero = %b", $time, A, B, op, out, cout, overflow, zero);

        // 测试案例 1：A=1001, B=1010
        A = 4'b1001; B = 4'b1010;
        op = 3'b000; // 初始化操作码
        counter = 4'b0000; // 初始化计数器
        for (counter = 4'b0000; counter < 4'b1000; counter = counter + 1) begin
            op = counter[2:0]; // 设置操作码，只取counter的低三位
            #10; // 仿真时钟周期
        end

        // 测试案例 2：A=0011, B=1111
        A = 4'b0011; B = 4'b1111;
        op = 3'b000; // 重置操作码
        counter = 4'b0000; // 初始化计数器
        for (counter = 4'b0000; counter < 4'b1000; counter = counter + 1) begin
            op = counter[2:0]; // 设置操作码，只取counter的低三位
            #10; // 仿真时钟周期
        end

        // 测试案例 3：A=1100, B=1000
        A = 4'b1100; B = 4'b1000;
        op = 3'b000; // 重置操作码
        counter = 4'b0000; // 初始化计数器
        for (counter = 4'b0000; counter < 4'b1000; counter = counter + 1) begin
            op = counter[2:0]; // 设置操作码，只取counter的低三位
            #10; // 仿真时钟周期
        end

        // 所有测试案例结束完成后暂停仿真
        $stop;
    end

endmodule
```