作业 1：

源码：

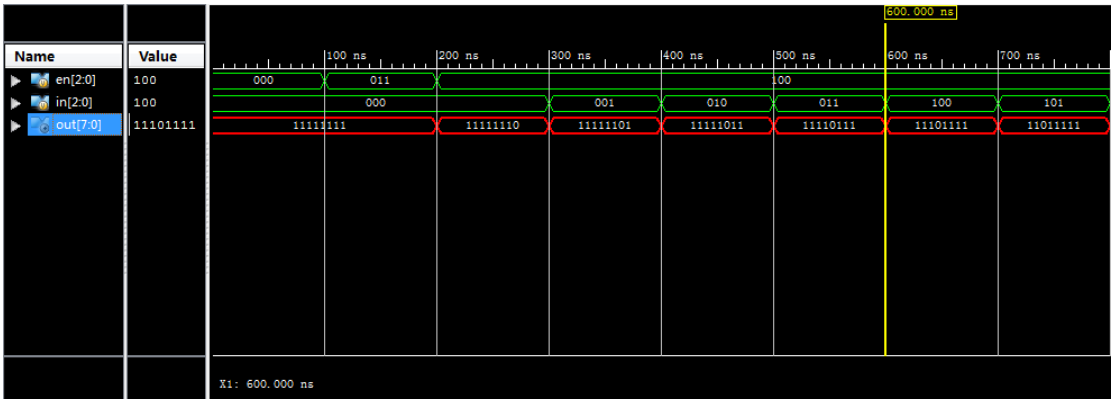
```
module decoder_74LS138( out, en, in
    );
     input [2:0] en;
     input [2:0] in;
     output reg [7:0] out;
     always @(*) begin
         if (en[2] & (~en[1]) & (~en[0])) begin //detect enable signal
              case (in)
                   3'b000 : out = 8'b11111110;
                    3'b001 : out = 8'b11111101;
                    3'b010 : out = 8'b11111011;
                    3'b011 : out = 8'b11110111;
                    3'b100 : out = 8'b11101111;
                    3'b101 : out = 8'b11011111;
                    3'b110 : out = 8'b10111111;
                    3'b111 : out = 8'b01111111;
                    default : out = 8'b11111111;
               endcase
          end
          else
              out = 8'b11111111;
     end
endmodule
```

Testbench：

```
module decoder_tb;

    // Inputs
    reg [2:0] en;
    reg [2:0] in;

    // Outputs
    wire [7:0] out;

    // Instantiate the Unit Under Test (UUT)
    decoder_74LS138 uut (
        .out(out),
        .en(en),
        .in(in)
    );
```

```
        initial begin
            // Initialize Inputs
            en = 0;
            in = 0;

            // Wait 100 ns for global reset to finish
            #100;

            // Add stimulus here
            en = 3'b011;
            #100 en = 3'b100;
            #100 in = 3'b001;
            #100 in = 3'b010;
            #100 in = 3'b011;
            #100 in = 3'b100;
            #100 in = 3'b101;
            #100 in = 3'b110;
            #100 in = 3'b111;
            #100 $finish;

        end
endmodule
```

波形图：



作业 2：

源码：

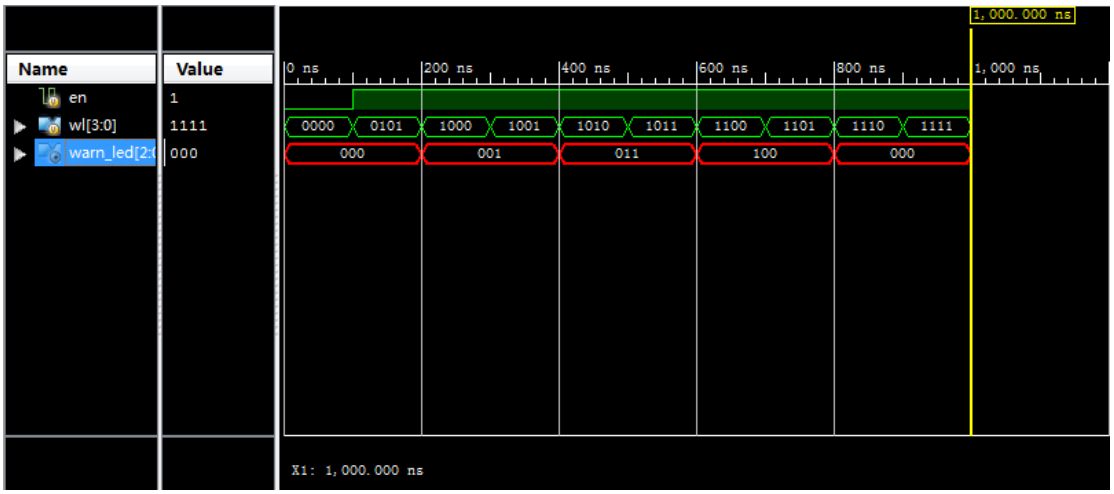
```
module water_level( warn_led, en, wl
    );

      input en;
      input [3:0] wl;
        // warn_led[0]: white led; warn_led[1]: yellow led; warn_led[2]: red led;
      output reg [2:0] warn_led;

    always @(*) begin
        if (en) begin
            if (wl < 4'b1000) warn_led = 3'b000; // water level lower than 8m
            // water level betweeen 8m and 10m
            else if((wl >= 4'b1000) && (wl < 4'b1010)) warn_led = 3'b001;
            // water level betweeen 8m and 10m
            else if((wl >= 4'b1010) && (wl < 4'b1100)) warn_led = 3'b011;
            // water level betweeen 8m and 10m
            else if((wl >= 4'b1100) && (wl < 4'b1110)) warn_led = 3'b100;
            else warn_led = 3'b000;
          end
          else
            warn_led = 3'b000;
      end
endmodule
```

Testbench：

```
module wl_tb;

    // Inputs
    reg en;
    reg [3:0] wl;

    // Outputs
    wire [2:0] warn_led;

    // Instantiate the Unit Under Test (UUT)
    water_level uut (
        .warn_led(warn_led),
        .en(en),
        .wl(wl)
    );
```

```
        initial begin
            // Initialize Inputs
            en = 0;
            wl = 0;

            // Wait 100 ns for global reset to finish
            #100;

            // Add stimulus here
            en = 1'b1;
            wl = 4'b0101;
            #100 wl = 4'b1000;
            #100 wl = 4'b1001;
            #100 wl = 4'b1010;
            #100 wl = 4'b1011;
            #100 wl = 4'b1100;
            #100 wl = 4'b1101;
            #100 wl = 4'b1110;
            #100 wl = 4'b1111;
            #100 $finish;
        end
endmodule
```

波形图：



作业 3：

源码：

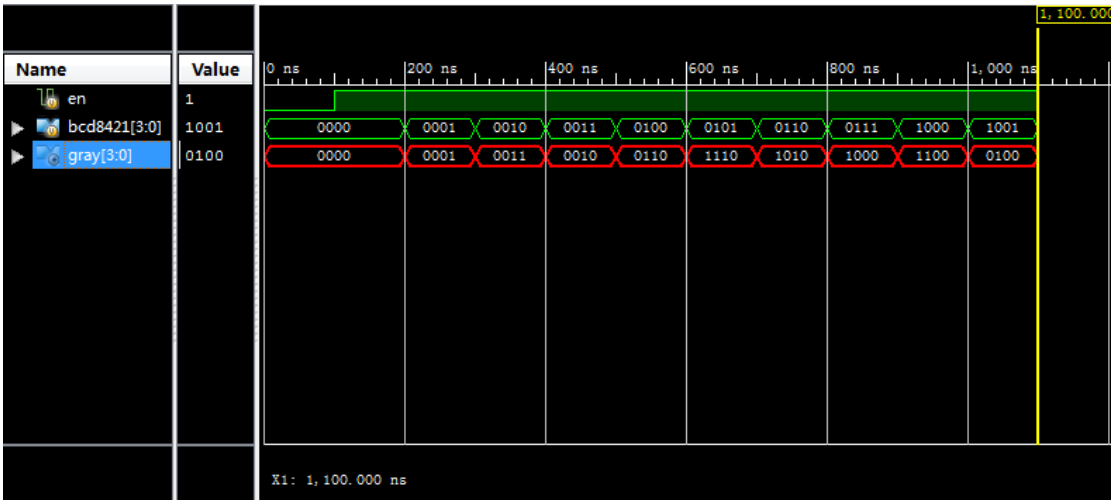
```
module bcd84212gray( gray, en, bcd8421
    );
     input en;
     input [3:0] bcd8421;
     output reg [3:0] gray;

    always @(*) begin
        if(en) begin
            case(bcd8421)
                4'b0000 : gray = 4'b0000;
                  4'b0001 : gray = 4'b0001;
                  4'b0010 : gray = 4'b0011;
                  4'b0011 : gray = 4'b0010;
                  4'b0100 : gray = 4'b0110;
                  4'b0101 : gray = 4'b1110;
                  4'b0110 : gray = 4'b1010;
                  4'b0111 : gray = 4'b1000;
                  4'b1000 : gray = 4'b1100;
                  4'b1001 : gray = 4'b0100;
              endcase
          end
          else gray = 4'b0000;
      end
endmodule
```

Testbench：

```
module bcd84212gray_tb;

    // Inputs
    reg en;
    reg [3:0] bcd8421;

    // Outputs
    wire [3:0] gray;

    // Instantiate the Unit Under Test (UUT)
    bcd84212gray uut (
        .gray(gray),
        .en(en),
        .bcd8421(bcd8421)
```

```
    );

    initial begin
        // Initialize Inputs
        en = 0;
        bcd8421 = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        en = 1;
        #100 bcd8421 = 4'b0001;
        #100 bcd8421 = 4'b0010;
        #100 bcd8421 = 4'b0011;
        #100 bcd8421 = 4'b0100;
        #100 bcd8421 = 4'b0101;
        #100 bcd8421 = 4'b0110;
        #100 bcd8421 = 4'b0111;
        #100 bcd8421 = 4'b1000;
        #100 bcd8421 = 4'b1001;
        #100 $finish;

    end
endmodule
```

波形图：



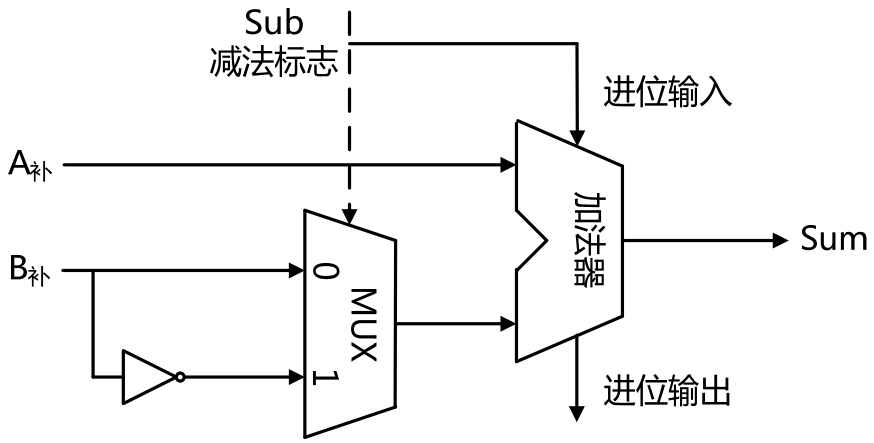
作业 4：

两个有符号数 A 和 B 的加减法：

$[A + B]_{补} = [A]_{补} + [B]_{补}$

$[A - B]_{补} = [A + (-B)]_{补} = [A]_{补} + [-B]_{补} = [A]_{补} + \sim([B]_{补}) + 1$

电路原理图如下：



溢出条件：

(1). 对于加法，两个操作数异号，则结果不溢出；两个操作数同号，若结果与操作数异号，则结果溢出。

(2). 对于减法，两个操作数同号，则结果不溢出；两个操作数异号，若结果与被减数异号，则结果溢出。

源码：

```
module add_4( carry, sum, in1, in2, sub
    );
      input [3:0] in1, in2;
      input sub;
      output reg carry;
      output reg [3:0] sum;
      reg [3:0] in2_temp;

      // one not gate, one multiplexer
      always @(*) begin
          if (sub)
```

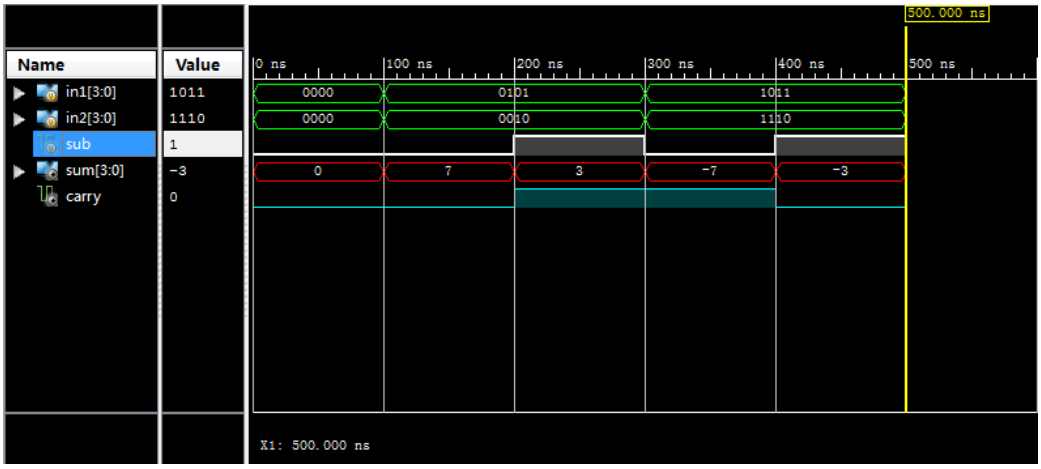
```
                in2_temp = ~in2;
            else in2_temp = in2;
        end

        // one adder
        always @(*) begin
            {carry, sum} = in1 + in2_temp + sub;
        end
endmodule
```

Testbench :

```
module add_4_tb;

    // Inputs
    reg [3:0] in1;
    reg [3:0] in2;
    reg sub;

    // Outputs
    wire carry;
    wire [3:0] sum;

    // Instantiate the Unit Under Test (UUT)
    add_4 uut (
        .carry(carry),
        .sum(sum),
        .in1(in1),
        .in2(in2),
        .sub(sub)
    );

    initial begin
        // Initialize Inputs
        in1 = 0;
        in2 = 0;
        sub = 0;

        // Wait 100 ns for global reset to finish
        #100;

        // Add stimulus here
        in1 = 4'b0101; //in1 = 5;
        in2 = 4'b0010; //in2 = 2;
        sub = 0; //add
```

```
        #100;
        in1 = 4'b0101; //in1 = 5;
        in2 = 4'b0010; //in2 = 2;
        sub = 1; //sub
        #100;
        in1 = 4'b1011; //in1 = -5
        in2 = 4'b1110; //in2 = -2;
        sub = 0; //add
        #100;
        in1 = 4'b1011; //in1 = -5
        in2 = 4'b1110; //in2 = -2;
        sub = 1; //sub
        #100 $finish;
    end
endmodule
```

波形图：



作业 5：

源码：

```
module priority( Y, I
    );

    input [7:0] I;
    output reg [2:0] Y;

    always @(*) begin
        if (I[7] == 0) Y = 4'b111; //The priority of I[7] is highest
        else if (I[6] == 0) Y = 4'b110;
        else if (I[5] == 0) Y = 4'b101;
        else if (I[4] == 0) Y = 4'b100;
        else if (I[3] == 0) Y = 4'b011;
        else if (I[2] == 0) Y = 4'b010;
        else if (I[1] == 0) Y = 4'b001;
        else if (I[0] == 0) Y = 4'b000; //The priority of I[0] is lowest
        else Y = 4'b000;
    end
endmodule
```

实验效果见附件工程，直接将.bit 文件下载到开发板上即可。