作业 1：

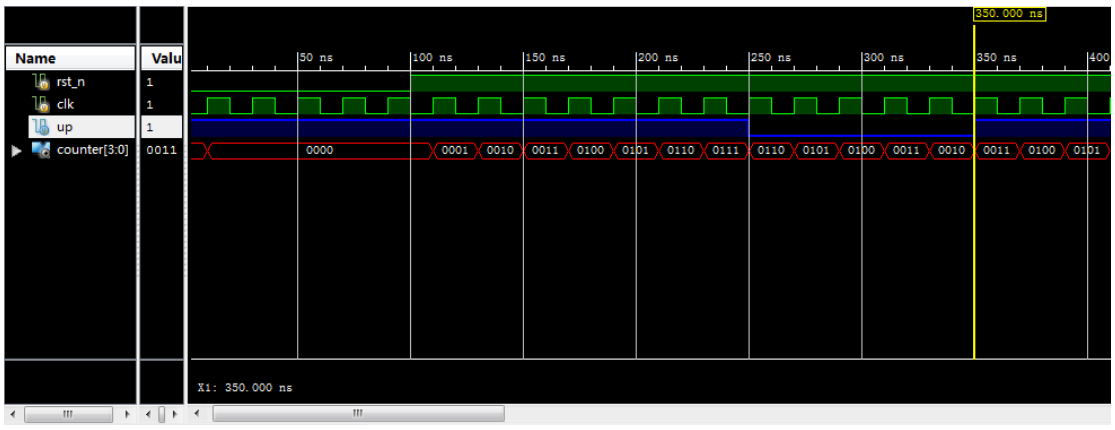
源码：

```
module rev_counter( counter, clk, rst_n, up
        );

          input clk, rst_n, up;
          output reg [3:0] counter;

          always @(posedge clk) begin
                if(~rst_n) counter <= 0;
                  else begin
                        if (up == 1) counter <= counter + 1;    // when up is "1", the counter is increased
                          else counter <= counter - 1;      // when up is "0", the counter is decreased
                  end
          end
endmodule
```

Testbench：

```
module rev_counter_tb;

        // Inputs
        reg clk;
        reg rst_n;
        reg up;

        // Outputs
        wire [3:0] counter;

        // Instantiate the Unit Under Test (UUT)
        rev_counter uut (
                .counter(counter),
                .clk(clk),
                .rst_n(rst_n),
                .up(up)
        );

        initial begin
                // Initialize Inputs
                clk = 0;
                rst_n = 0;
                up = 1;
                // Wait 100 ns for global reset to finish
                #100;

                // Add stimulus here
                rst_n = 1;

                #150 up = 0;
                #100 up = 1;
                #100 $finish;

        end

        always #10 clk = ~clk;

endmodule
```

波形图：



作业 2：

源码：由三个模块组成

分频模块（产生 6Hz 时钟）

```
module clkdiv(clk6Hz, clk, rst_n
        );
         input clk, rst_n;
         output clk6Hz;

         reg [23:0] counter;

         always @(posedge clk or negedge rst_n) begin
               if (~rst_n) counter <= 0;
                else counter <= counter + 1;
         end
         assign clk6Hz = counter[23]; // 6Hz clock frequency
endmodule
```

跑马灯模块：

```
module ledbanner( led, clk6Hz, rst_n, set
        );
         input clk6Hz, rst_n, set;
         output reg [7:0] led;
```

```
    always @(posedge clk6Hz or negedge rst_n) begin
        if(~rst_n) led <= 8'b00000000;
         else if(set) led <= 8'b00000001;
         //circularly lighten 8 leds using shifting register
         else begin
             led[7:1] <= led[6:0];
             led[0] <= led[7];
         end
    end
endmodule
```

顶层模块：

```
module ledbanner_top(led, clk, rst_n, set
    );

    input clk, rst_n, set;
    output [7:0] led;

    wire clk6Hz;

    clkdiv U1(.clk6Hz(clk6Hz), .clk(clk), .rst_n(rst_n));
    ledbanner U2(.led(led), .clk6Hz(clk6Hz), .rst_n(rst_n), .set(set));
endmodule
```

问题答案：

为了点亮 LED 灯，设计将 100MHz 时钟分频为 6Hz，所以会造成仿真时间过长。因此，在仿真时，我们可以使用一个较快的分频时钟（如 50MHz）来实现跑马灯，从而有效缩短仿真时间，上板调试时，再将分频时钟改回 6Hz 即可。

实验效果见附件工程，直接将.bit 文件下载到开发板上即可。

作业 3：

源码：由五个模块组成

分频模块（产生 190Hz 时钟）

```
module clkdiv( clk190, clk, rst_n
	);

	input clk, rst_n;
	output clk190;

	reg [18:0] counter;

	always @(posedge clk or negedge rst_n) begin
	   if (~rst_n) counter <= 0;
	    else counter <= counter + 1;
	end

	assign clk190 = counter[18]; //190Hz clock frequency
endmodule
```

计数（0 到 99）模块：

```
`define WIDTH 8
module counter99( counter, clk, rst_n, btn
	);
```

```
input clk, rst_n, btn;

output reg [`WIDTH - 1:0] counter;


reg delay1, delay2, delay3;

wire btn_pluse;


// single pluse generation

always @(posedge clk or negedge rst_n) begin

    if (~rst_n) {delay1, delay2, delay3} <= 3'b000;

     else begin

         delay1 <= btn;

          delay2 <= delay1;

          delay3 <= delay2;

     end

end


assign btn_pluse = delay1 & delay2 & (~delay3);


// counter from 0 to 99

always @(posedge clk or negedge rst_n) begin

    if (~rst_n) counter <= 0;
```

```
            else if (btn_pluse == 1'b1) begin
                if (counter < 'd99) counter <= counter + 1;
                 else counter <= 0;
            end
            else counter <= counter;
        end
endmodule
```

二进制转十进制模块：

```
`define WIDTH 8
module bin2bcd( bcd, bin
        );

        input [`WIDTH - 1:0] bin;
        output reg [15:0] bcd;

        reg [`WIDTH + 15:0] temp;
        integer i;

        always @(*)
        begin

           for (i=0; i <= 'd23; i = i + 1)
```

```
            temp[i] = 0;
        temp[7:0] = bin;
        repeat(`WIDTH)
        begin
            // the unit >= 5?
           if (temp[11:8] > 4) temp[11:8] = temp[11:8] + 3;
           // the decade >= 5?
            if (temp[15:12] > 4) temp[15:12] = temp[15:12] + 3;
           // left shift one bit
            temp[`WIDTH + 15:1] = temp[`WIDTH + 14:0];
        end
        bcd = temp[`WIDTH + 15:8];
     end
endmodule
```

7 段数码管动态显示模块：

```
module seg7_d( a_to_g, an, clk190, rst_n, x
     );

     input clk190, rst_n;
     input [15:0] x;
     output [6:0] a_to_g;
     output [3:0] an;
```

```
reg [6:0] a_to_g;

reg [3:0] an;

reg [1:0] an_sel;

reg [3:0] digit;


always @(*)

   case (digit)

       0: a_to_g = 7'b0000001;

       1: a_to_g = 7'b1001111;

       2: a_to_g = 7'b0010010;

       3: a_to_g = 7'b0000110;

       4: a_to_g = 7'b1001100;

       5: a_to_g = 7'b0100100;

       6: a_to_g = 7'b0100000;

       7: a_to_g = 7'b0001111;

       8: a_to_g = 7'b0000000;

       9: a_to_g = 7'b0000100;

       'hA: a_to_g = 7'b0001000;

       'hB: a_to_g = 7'b1100000;

       'hC: a_to_g = 7'b0110001;

       'hD: a_to_g = 7'b1000010;
       'hE: a_to_g = 7'b0110000;
       'hF: a_to_g = 7'b0111000;
       default: a_to_g = 7'b0000001;   // 0
endcase

//dynamic scanning of four 7-segment digital tubes
always @(posedge clk190 or negedge rst_n) begin

       if(~rst_n) begin
            an_sel <= 0;
              an <= 4'b1111;
       end
       else begin
            case(an_sel)
                   0: begin
                         an <= 4'b1110;
                         digit <= x[3:0];
                    end
                    1:begin
                         an <= 4'b1101;
                         digit <= x[7:4];
                    end
                    2:begin
                        an <= 4'b1011;
                        digit <= x[11:8];
                    end
                    3:begin
                        an <= 4'b0111;
                        digit <= x[15:12];
                    end
                endcase
                an_sel <= an_sel + 1;
            end
        end
endmodule
```

顶层模块：

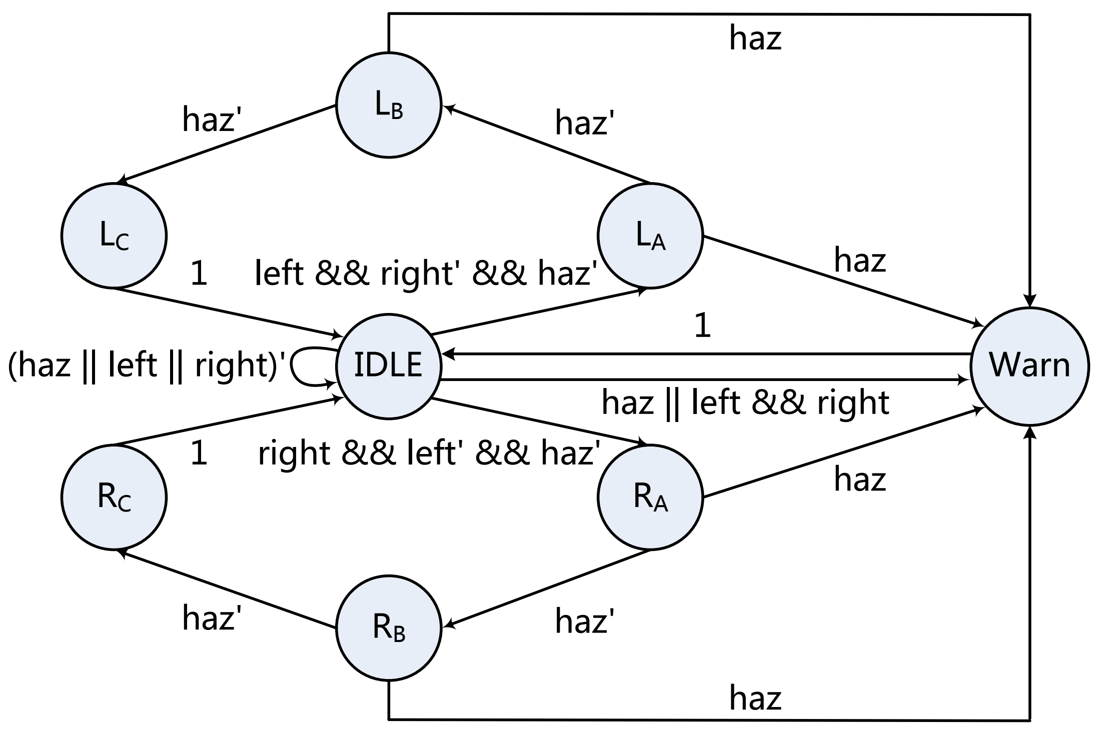
```
`define WIDTH 8
module counter99_top( a_to_g, an, clk, rst_n, btn
     );
    input clk, rst_n, btn;
     output [6:0] a_to_g;
     output [3:0] an;

     wire clk190;
```

```
	wire [`WIDTH - 1:0] counter;

	wire [15:0] seg7_data;


	clkdiv U1( .clk190(clk190), .clk(clk), .rst_n(rst_n));

	counter99 U2(.counter(counter), .clk(clk190), .rst_n(rst_n), .btn(btn));

	bin2bcd U3(.bcd(seg7_data), .bin(counter));

	seg7_d
U4(.a_to_g(a_to_g), .an(an), .clk190(clk190), .rst_n(rst_n), .x(seg7_data));

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。

作业 4：

状态机：



IDLE 表示所有灯都不亮；LA，LB 和 LC 对应左边 1 个，2 个或 3 个灯亮；RA，RB 和 RC 对应右边 1 个，2 个或 3 个灯亮；Warn 对应左边和右边 6 个灯亮。

分频模块（产生 6Hz 时钟）

```
module clkdiv(clk6Hz, clk, rst_n

    );



     input clk, rst_n;

     output clk6Hz;



     reg [23:0] counter;



     always @(posedge clk or negedge rst_n) begin
```

```
            if (~rst_n) counter <= 0;
             else counter <= counter + 1;
        end

        assign clk6Hz = counter[23]; // 6Hz clock frequency
endmodule
```

尾灯控制状态机

```
module taillight_fsm( tl_led, clk, rst_n, haz, left, right
        );

        input clk, rst_n, haz, left, right;
        output [5:0] tl_led; // vehicle taillight

        reg [5:0] tl_led;
        reg [7:0] current_state, next_state;
        parameter [7:0] IDLE = 8'b00000001, LA = 8'b00000010, LB =
8'b00000100, LC = 8'b00001000;
        parameter [7:0] RA = 8'b00010000, RB = 8'b00100000, RC =
8'b01000000, WARN = 8'b10000000;

        always @(posedge clk or negedge rst_n) begin
            if(~rst_n) current_state <= IDLE;
```

```
       else current_state <= next_state;
   end


   always @(*) begin
      if(~rst_n) next_state = IDLE;
       else begin
          case(current_state)
             IDLE : begin
                 if (haz || (left & right)) next_state = WARN;
                  else if (left && (~haz) && (~right)) next_state = LA;
                  else if (right && (~haz) && (~left)) next_state = RA;
                  else next_state = IDLE;
              end
              LA : begin
                 if (haz) next_state = WARN;
                  else next_state = LB;
              end
              LB : begin
                 if (haz) next_state = WARN;
                  else next_state = LC;
              end
              LC : next_state = IDLE;
```

```
                RA : begin
                    if (haz) next_state = WARN;
                     else next_state = RB;
                end
                RB : begin
                    if (haz) next_state = WARN;
                     else next_state = RC;
                end
                RC : next_state = IDLE;
                WARN : next_state = IDLE;
                default : next_state = IDLE;
            endcase
        end
    end

 always @(posedge clk or negedge rst_n) begin
    if(~rst_n) tl_led <= 0;
     else begin
         case(current_state)
             IDLE : tl_led <= 6'b000000;
              LA : tl_led <= 6'b001000;
              LB : tl_led <= 6'b011000;
```

```
                LC : tl_led <= 6'b111000;
                RA : tl_led <= 6'b000100;
                RB : tl_led <= 6'b000110;
                RC : tl_led <= 6'b000111;
                WARN : tl_led <= 6'b111111;
            endcase
        end
    end
endmodule
```

顶层模块

```
module taillight_top( tl_led, clk, rst_n, haz, left, right
    );

    input clk, rst_n, haz, left, right;
    output [5:0] tl_led;

    wire clk6Hz;

    clkdiv U1(.clk6Hz(clk6Hz), .clk(clk), .rst_n(rst_n));
    taillight_fsm U2(.tl_led(tl_led), .clk(clk6Hz), .rst_n(rst_n), .haz(haz), .left(left), .right(right));
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。