HOMEWORK1

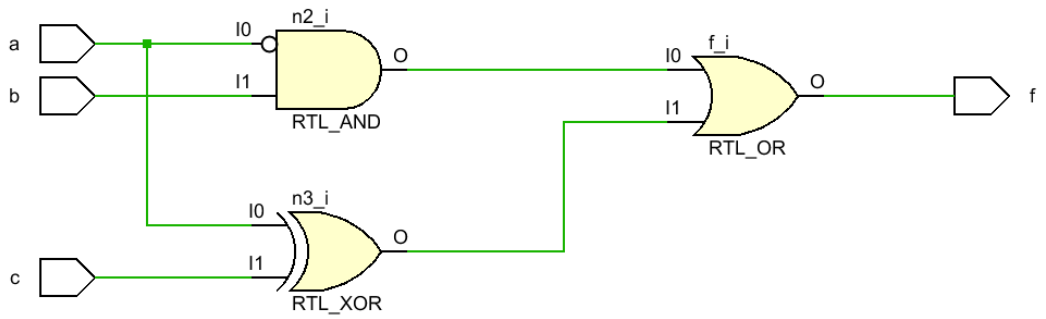
Source code

```
`timescale 1ns/1ns

module comb_logic(
    input a,b,c,
    output f
);
    wire n1,n2,n3;

    assign n1 = ~ a;
    assign n2 = n1 & b;
    assign n3 = a ^ c;
    assign f = n2 | n3;

endmodule
```

Testbench code

```
`timescale 1ns/1ns

module test;
    reg a,b,c;
    wire f;
    integer i;

    comb_logic uut(
        .a(a),
        .b(b),
        .c(c),
        .f(f)
    );

    always begin
        for(i=0;i<=7;i=i+1) begin
            {a,b,c}=i;
            #10;
        end
        $finish;
    end
endmodule
```

Simulation waveform



HOMEWORK2

Source code

```
`timescale 1ns/1ns

module odd_check #(
    parameter N=8
) (
    input [N-1:0] din,
    output dout
);

    assign dout=~(^din);

endmodule
```

Testbench code

```
`timescale 1ns/1ns

module test;

    reg [1:0] din_2;
    reg [3:0] din_4;
    reg [5:0] din_6;
    wire dout_2,dout_4,dout_6;

    odd_check #(
        .N(2)
    ) odd_check_2_inst (
        .din(din_2),
        .dout(dout_2)
    );

    odd_check #(
        .N(4)
    ) odd_check_4_inst (
        .din(din_4),
        .dout(dout_4)
    );

    odd_check #(
        .N(6)
    ) odd_check_6_inst (
        .din(din_6),
```

```
        .dout(dout_6)
    );

    integer force_i;
    initial begin
        for(force_i=0; force_i<2**6; force_i=force_i+1) begin
            din_2=force_i<2**2 ? force_i : 'bz;
            din_4=force_i<2**4 ? force_i : 'bz;
            din_6=force_i;
            #10;
        end
        $finish;
    end

    //check=1 :has odd  '1' in total
    //check=0 :has even '1' in total
    wire check_2,check_4,check_6;
    assign check_2=(^din_2)^dout_2;
    assign check_4=(^din_4)^dout_4;
    assign check_6=(^din_6)^dout_6;

endmodule
```

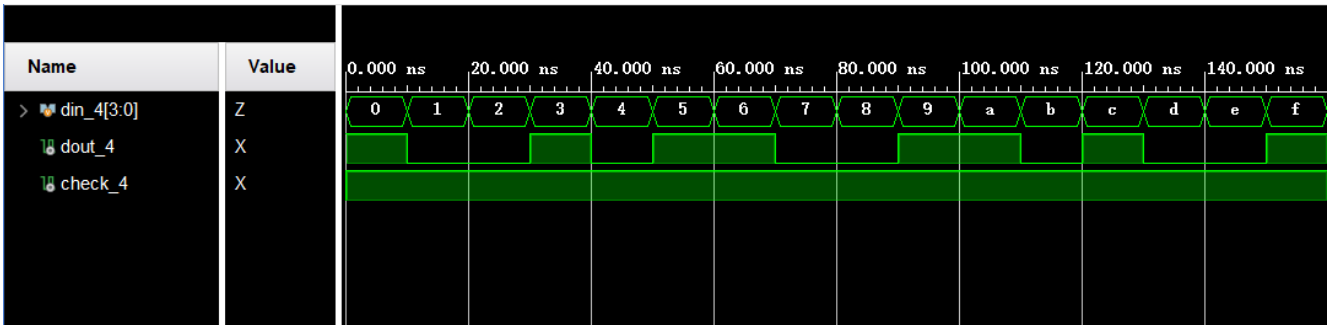
Simulation waveform

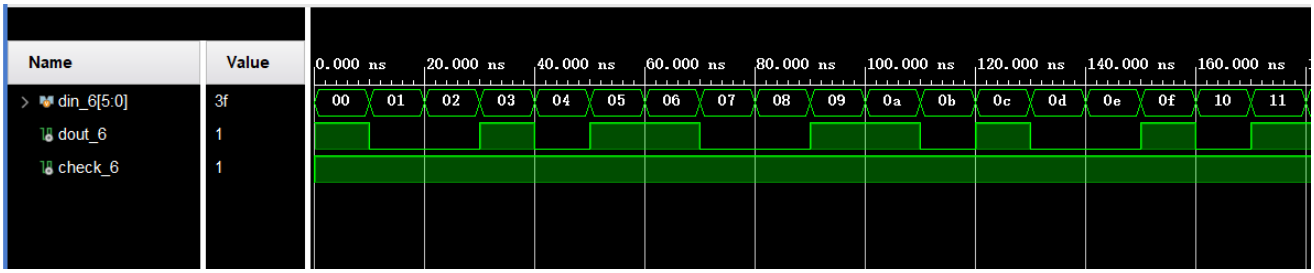
有 2 位、4 位、6 位长的模版实例

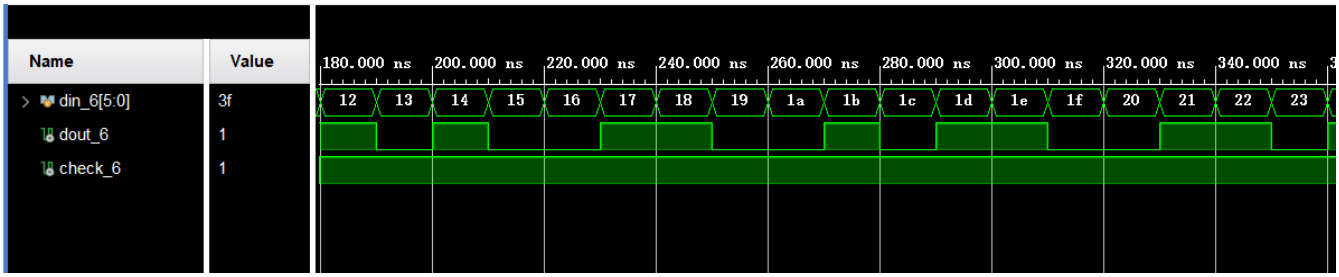
testbench 的校验（不是设计的电路的校验位，那个是 dout）：

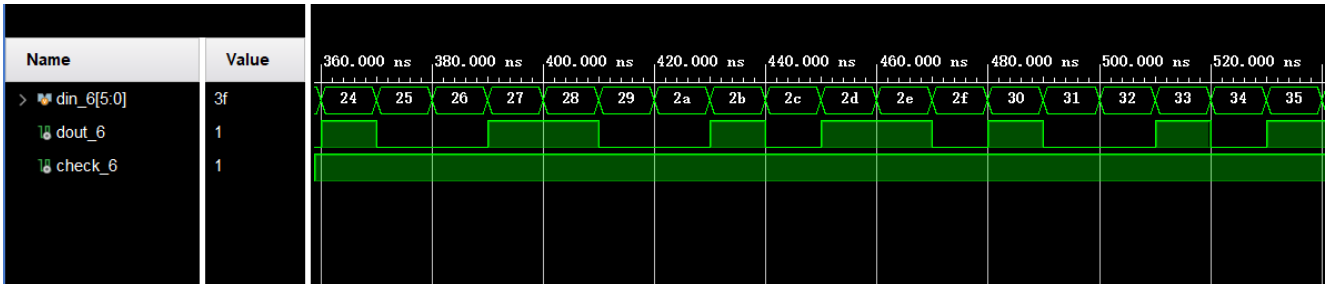
//check=1 :has odd '1' in total (din+dout)

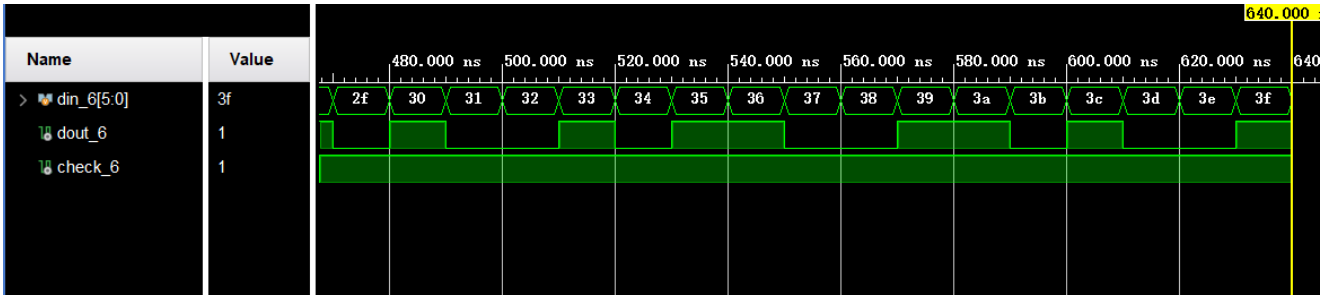
//check=0 :has even '1' in total (din+dout)


Name
Value
din_4[3:0]
Z
dout_4
X
check_4
X
0.000 ns
20.000 ns
40.000 ns
60.000 ns
80.000 ns
100.000 ns
120.000 ns
140.000 ns
0 1 2 3 4 5 6 7 8 9 a b c d e f


Name
Value
din_6[5:0]
3f
dout_6
1
check_6
1
0.000 ns
20.000 ns
40.000 ns
60.000 ns
80.000 ns
100.000 ns
120.000 ns
140.000 ns
160.000 ns
00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11


Name
Value
din_6[5:0]
3f
dout_6
1
check_6
1
180.000 ns
200.000 ns
220.000 ns
240.000 ns
260.000 ns
280.000 ns
300.000 ns
320.000 ns
340.000 ns
12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23


Name
Value
din_6[5:0]
3f
dout_6
1
check_6
1
360.000 ns
380.000 ns
400.000 ns
420.000 ns
440.000 ns
460.000 ns
480.000 ns
500.000 ns
520.000 ns
24 25 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35


Name
Value
din_6[5:0]
3f
dout_6
1
check_6
1
480.000 ns
500.000 ns
520.000 ns
540.000 ns
560.000 ns
580.000 ns
600.000 ns
620.000 ns
2f 30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f

HOMEWORK3

Source code

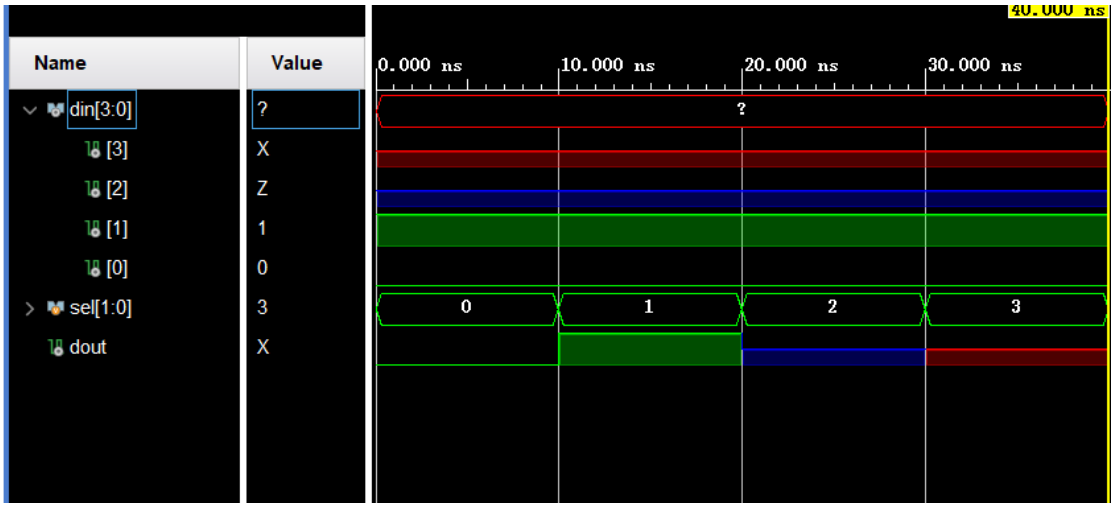
```
`timescale 1ns/1ns

module mux_4(
    input [3:0] din,
    input [1:0] sel,
    output reg dout
);

    always @(din,sel) begin
        case(sel)
            2'b00: dout=din[0];
            2'b01: dout=din[1];
            2'b10: dout=din[2];
            2'b11: dout=din[3];
            default: dout=1'bz;
        endcase
    end

    // assign dout=din[sel];

endmodule
```

Testbench code

```
`timescale 1ns/1ns

module test;
    wire [3:0] din;
    reg [1:0] sel;
    wire dout;

    mux_4 uut(
        .din(din),
        .sel(sel),
        .dout(dout)
    );

    assign din=4'bxz10;

    integer i;
    initial begin
        for(i=0;i<=3;i=i+1) begin
```

```
            sel=i;
            #10;
        end
        $finish;
    end

endmodule
```

Simulation waveform



din 的[3:2]位有意设置 z 和 x。

HOMEWORK4

Source code

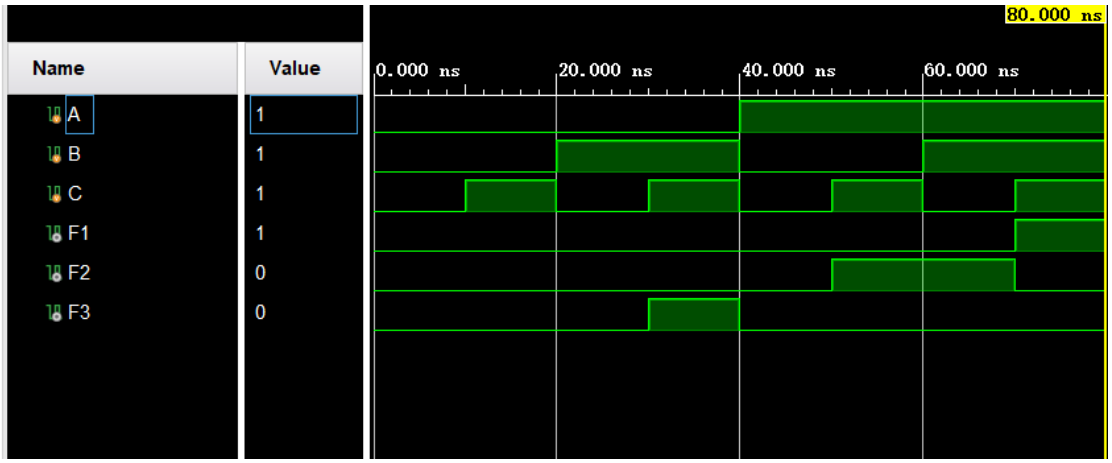
```
`timescale 1ns/1ns

module comb_quality (
    input A,B,C,
    output reg F1,F2,F3
);

    always @(A,B,C) begin
        if(A && B && C)         //{A,B,C}=3'b111
            {F1,F2,F3}=3'b100;
        else if(A && (B || C))  //{A,B,C}=3'b101 || {A,B,C}=3'b110
            {F1,F2,F3}=3'b010;
        else if(~A && (B && C)) //{A,B,C}=3'b011
            {F1,F2,F3}=3'b001;
        else
            {F1,F2,F3}=3'b000;
    end

endmodule
```

Testbench code

```
`timescale 1ns/1ns

module test;
    reg A,B,C;
    wire F1,F2,F3;

    comb_quality uut(
        .A(A),
        .B(B),
        .C(C),
        .F1(F1),
        .F2(F2),
        .F3(F3)
    );

    integer i;
    initial begin
        for(i=0;i<=7;i=i+1) begin
            {A,B,C}=i[3:0];
            #10;
```

```
        end
        $finish;
    end

endmodule
```

Simulation waveform



HOMEWORK5

Source code

```
`timescale 1ms/1ms

module counter_10 (
    input rst,
    input en,
    input clk1k,
    output reg [6:0] seg,
    output dig
);

    //clock division
    reg [7:0] cnt_clk;//0~249
    reg clk2hz;
    always @(posedge clk1k,posedge rst) begin
        if(rst) begin
            cnt_clk<=0;
            clk2hz<=0;
        end else if(en) begin
            if(cnt_clk==249) begin
                cnt_clk<=0;
                clk2hz <= ~clk2hz;
            end else
                cnt_clk<=cnt_clk+1;
        end
    end

    //count_10
    reg [3:0] cnt_num;//0~9
    always @(posedge clk2hz,posedge rst) begin
        if(rst)
            cnt_num<=0;
        else if(en) begin
            if(cnt_num==9)
                cnt_num<=0;
            else
                cnt_num<=cnt_num+1;
        end
    end

    //dig output
    always @(*) begin
        case(cnt_num)
```

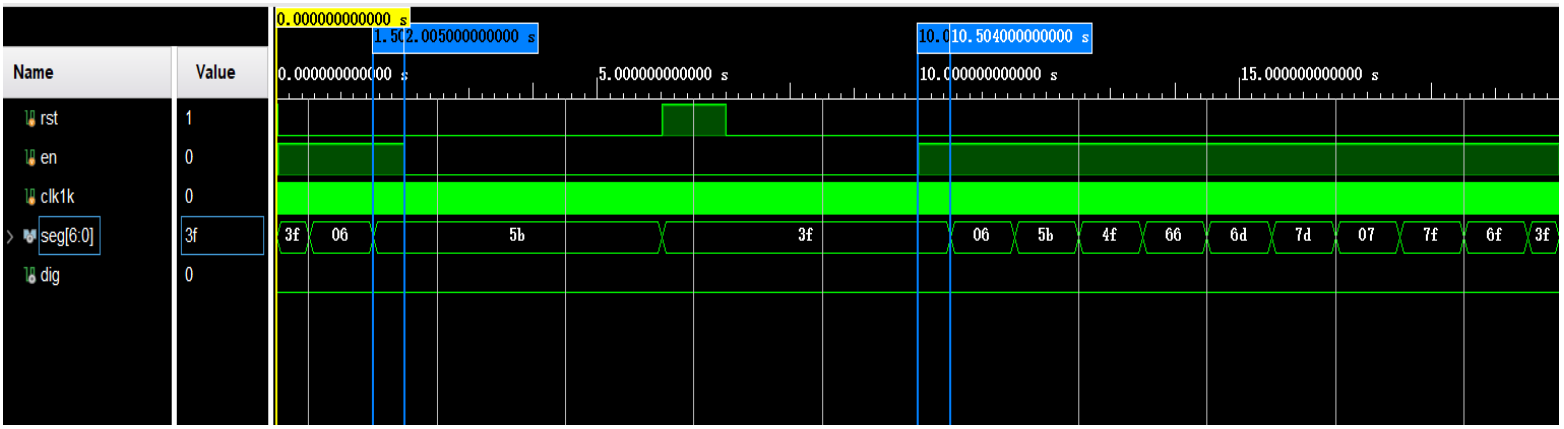
```
            4'h0 : seg = 7'h3f;
            4'h1 : seg = 7'h06;
            4'h2 : seg = 7'h5b;
            4'h3 : seg = 7'h4f;
            4'h4 : seg = 7'h66;
            4'h5 : seg = 7'h6d;
            4'h6 : seg = 7'h7d;
            4'h7 : seg = 7'h07;
            4'h8 : seg = 7'h7f;
            4'h9 : seg = 7'h6f;
        endcase
    end
    assign dig=1'b0;

endmodule
```

Testbench code

```
`timescale 1ms/1ms

module test;

    reg rst;
    reg en;
    reg clk1k;
    wire [6:0] seg;
    wire dig;

    counter_10 count_10(
        .rst(rst),
        .en(en),
        .clk1k(clk1k),
        .seg(seg),
        .dig(dig)
    );

    //clk1k generate
    initial begin
        clk1k=0;
        #10;
        forever begin
            clk1k = ~clk1k;
            #1;
        end
```

```
    end

    //rst and en
    initial begin
        rst=1;  en=0;
        #5;
        rst=0;  en=1;
        #2000;
        rst=0;  en=0;
        #4000;
        rst=1;  en=0;
        #1000;
        rst=0;  en=0;
        #3000;
        rst=0;  en=1;
    end

    //finish time
    always begin
        #500;
        if($time>=500*40)
            $finish;
    end

endmodule
```

Simulation waveform



有下板测试

从右向左是复位、使能键。

HOMEWORK6

Source code

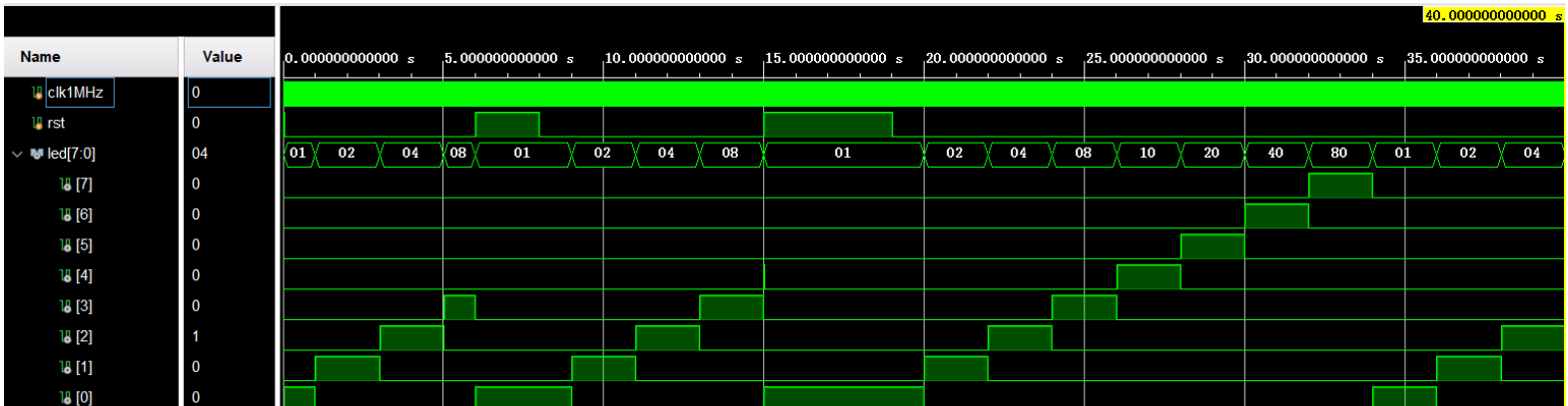
```
`timescale 1ms/1us

module water_led (
    input clk1MHz,
    input rst,
    output reg [7:0] led
);

    //clk division
    reg [18:0] cnt_clk;//0~500K
    reg clk1Hz;
    always @(posedge clk1MHz,posedge rst) begin
        if(rst) begin
            cnt_clk<=0;
            clk1Hz<=0;
        end else begin
            if(cnt_clk==500_000-1) begin
                cnt_clk<=0;
                clk1Hz <= ~clk1Hz;
            end else
                cnt_clk<=cnt_clk+1;
        end
    end

    //water led
    always @(posedge clk1Hz,posedge rst) begin
        if(rst)
            led<=8'b0000_0001;
        else
            led<={led[6:0],led[7]};
    end

endmodule
```

Testbench code

```
`timescale 1ms/1us

module test;
    reg clk1MHz;
    reg rst;
    wire [7:0] led;
```

```
    water_led water_led(
        .rst(rst),
        .clk1MHz(clk1MHz),
        .led(led)
    );

    //clk1MHz generate
    initial begin
        clk1MHz=0;
        #10;
        forever begin
            clk1MHz = ~clk1MHz;
            #0.001;
        end
    end

    //rst generate
    initial begin
        rst=1;
        #5;
        rst=0;
        #6000 rst=1;
        #2000 rst=0;
        #7000 rst=1;
        #4000 rst=0;
    end

    //finish time
    always begin
        #1000;
        if($time>=1000*40)
            $finish;
    end

endmodule
```

Simulation waveform



有下板测试

一个复位键

HOMEWORK7

Source code

Traffic_light.v

```
`timescale 1ms/100us

module traffic_light (
    input CLK1K,
    input nRST,
    output [6:0] seg,
    output [5:0] dig,
    output [7:0] led
);

    //clk division
    wire CLK1Hz;
    clock_division #(.DIVCLK_CNTMAX(499))
        light_clock(
            .clk_in(CLK1K),
            .rst_n(nRST),
            .divclk(CLK1Hz)
    );

    //led show
    wire [3:0] num_1,num_2,num_5,num_6;
    led_show led_show(
        .rst_n(nRST),
        .CLK1K(CLK1K),
        .num_1(num_1),
        .num_2(num_2),
        .num_3(4'hA),
        .num_4(4'hA),
        .num_5(num_5),
        .num_6(num_6),
        .dig(dig),
        .seg(seg)
    );

    //bcd convert
    reg [5:0] num_A,num_B;
    bin2bcd bin2bcd_A(
```

```
    .bin(num_A),
    .bcd_1(num_1),
    .bcd_2(num_2)
);
bin2bcd bin2bcd_B(
    .bin(num_B),
    .bcd_1(num_5),
    .bcd_2(num_6)
);

//traffic time counter
reg [5:0] cnt_traf;//0~59
always @(posedge CLK1Hz,negedge nRST) begin
    if(~nRST)
        cnt_traf<=59;
    else begin
        if(cnt_traf==0)
            cnt_traf<=59;
        else
            cnt_traf<=cnt_traf-1;
    end
end

//A path control(59~30:stop; 29~0:start)
reg [2:0] led_A;
always @(*) begin
    if(!nRST) begin
        led_A=3'b000;
        num_A=cnt_traf-30+1;    //as cnt_traf=59
    end else if(cnt_traf>=30) begin
        led_A=3'b100;   //red light
        num_A=cnt_traf-30+1;
    end else if(cnt_traf>=5) begin
        led_A=3'b001;   //green light
        num_A=cnt_traf-5+1;
    end else begin
        led_A=3'b010;   //yellow light
        num_A=cnt_traf+1;
    end
end

//B path control(59~30:start; 29~0:stop)
reg [2:0] led_B;
always @(*) begin
```

```
        if(!nRST) begin
            led_B=3'b000;
            num_B=cnt_traf+1;    //as cnt_traf=59
        end else if(cnt_traf<=29) begin
            led_B=3'b100;   //red light
            num_B=cnt_traf+1;
        end else if(cnt_traf<=34) begin
            led_B=3'b010;   //yellow light
            num_B=cnt_traf-30+1;
        end else begin
            led_B=3'b001;   //green light
            num_B=cnt_traf-35+1;
        end
    end

    //led mapping
    assign led[7:5]=led_B;
    assign led[2:0]=led_A;
    assign led[4:3]=2'b00;

endmodule
```

led_show.v

```
module led_show(
    input rst_n,
    input CLK1K,
    input [3:0] num_1,num_2,num_3,num_4,num_5,num_6,
    output reg [5:0] dig,
    output reg [6:0] seg
    );

    //dig generate
    always @(posedge CLK1K,negedge rst_n) begin
        if(!rst_n)
            dig<=6'b111_110;
        else
            dig<={dig[4:0],dig[5]};
    end

    //choose a num
    reg [3:0] num;
    always@(*) begin
```

```
        if(!rst_n)
            num = 4'hA;
        else begin
            case(dig)
                6'b111_110 : num = num_1;
                6'b111_101 : num = num_2;
                6'b111_011 : num = num_3;
                6'b110_111 : num = num_4;
                6'b101_111 : num = num_5;
                6'b011_111 : num = num_6;
                default : num = 4'hA;   //(A~F means seg off)
            endcase
        end
    end

    //dig output(A~F means seg off)
    always@(*)
        case(num)
            4'h0 : seg = 7'h3f;
            4'h1 : seg = 7'h06;
            4'h2 : seg = 7'h5b;
            4'h3 : seg = 7'h4f;
            4'h4 : seg = 7'h66;
            4'h5 : seg = 7'h6d;
            4'h6 : seg = 7'h7d;
            4'h7 : seg = 7'h07;
            4'h8 : seg = 7'h7f;
            4'h9 : seg = 7'h6f;
            default :seg = 7'h00;
        endcase

endmodule
```

clock_division.v

```
module clock_division
    #(parameter DIVCLK_CNTMAX = 24999)
    (clk_in,rst_n,divclk);
    input clk_in;
    input rst_n;
    output  reg divclk;

    reg [31:0] cnt;
```

```
    always@(posedge clk_in,negedge rst_n) begin
        if(~rst_n) begin
            cnt<=0;
            divclk<=0;
        end
        else if(cnt == DIVCLK_CNTMAX) begin
            cnt <= 0;
            divclk <= ~divclk;
        end
        else
            cnt <= cnt + 1;
    end

endmodule
```

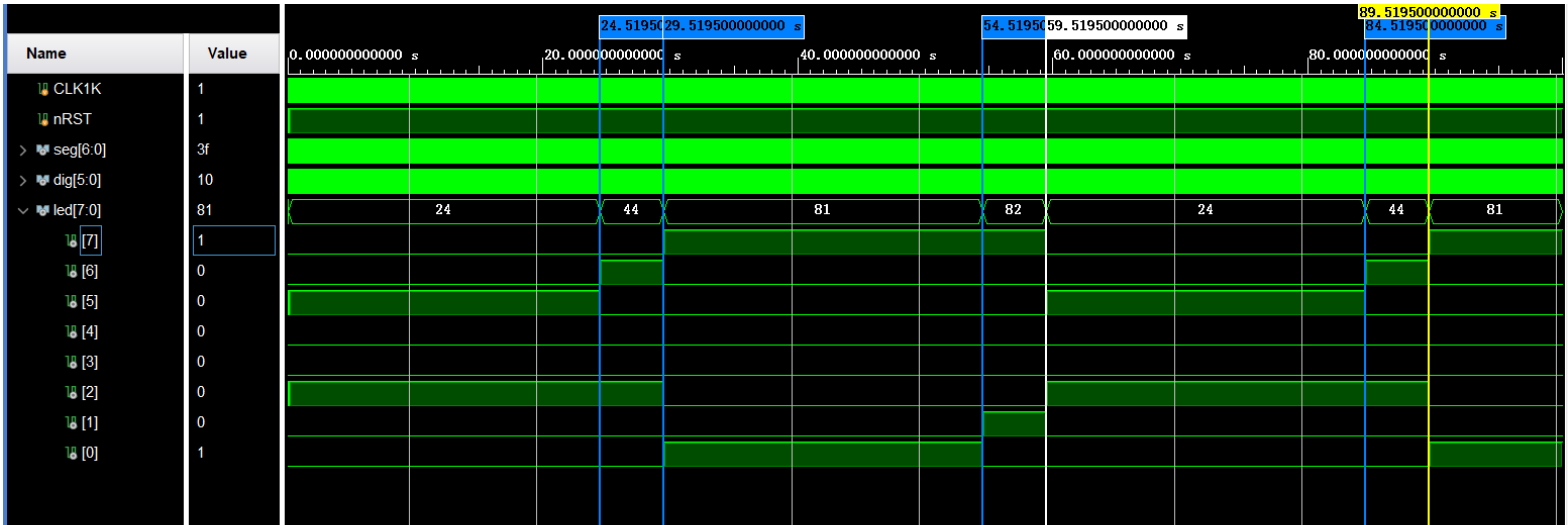
bin2bcd.v

```
module bin2bcd (
    input [5:0] bin,
    output [3:0] bcd_1,bcd_2
);
    integer i;
    reg [7:0] bcd_tmp;
    reg [5:0] bin_reg;
    always @(*) begin
        bin_reg=bin;
        bcd_tmp=0;
        repeat(6) begin
            if(bcd_tmp[3:0]>=5)
                bcd_tmp=bcd_tmp+3;
            if(bcd_tmp[7:4]>=5)
                bcd_tmp=bcd_tmp+(3<<4);
            bcd_tmp={bcd_tmp[6:0],bin_reg[5]};
            bin_reg=bin_reg<<1;
        end
    end

    assign bcd_1=bcd_tmp[3:0];
    assign bcd_2=bcd_tmp[7:4];

endmodule
```

Testbench code

```
`timescale 1ms/100us

module test;
    reg CLK1K;
    reg nRST;
    wire [6:0] seg;
    wire [5:0] dig;
    wire [7:0] led;

    traffic_light traffic_light(
        .CLK1K(CLK1K),
        .nRST(nRST),
        .seg(seg),
        .dig(dig),
        .led(led)
    );

    initial begin
        CLK1K=0;
        nRST=0;
        #20 nRST=1;
        forever
            #0.5 CLK1K=~CLK1K;
    end

    always begin
        #1000;
        if($time>=1000*100)
            $finish;
    end

endmodule
```

Simulation waveform



led[7:5]表示 B 通道的红灯、黄灯、绿灯

led[2:0]表示 A 通道的红灯、黄灯、绿灯

有下板测试

LED-T2~T4:左侧是 A 通道，LED-R7~R9:右侧是 B 通道