数字系统设计作业

学号：517021911065　姓名：刘浩文

2019 年 12 月 14 日

1　第 1 题

(1) 设计模块

```
//File: wavegen.v
//517021911065 刘浩文

`timescale 10 ns / 1 ns

module wavegen;

  reg wave;

  /*initial
  begin
    $dumpfile("wavegen.vcd");
    $dumpvars(0, wavegen);
  end*/

  initial
  fork
    wave = 1'b0;
    #2  wave = ~wave;
    #3  wave = ~wave;
    #12 wave = ~wave;
    #22 wave = ~wave;
    #24 wave = ~wave;
    #27 wave = ~wave;
    #32 wave = ~wave;
    #35 $stop;
  join

endmodule
```

(2) 测试模块

无

(3) 测试波形图

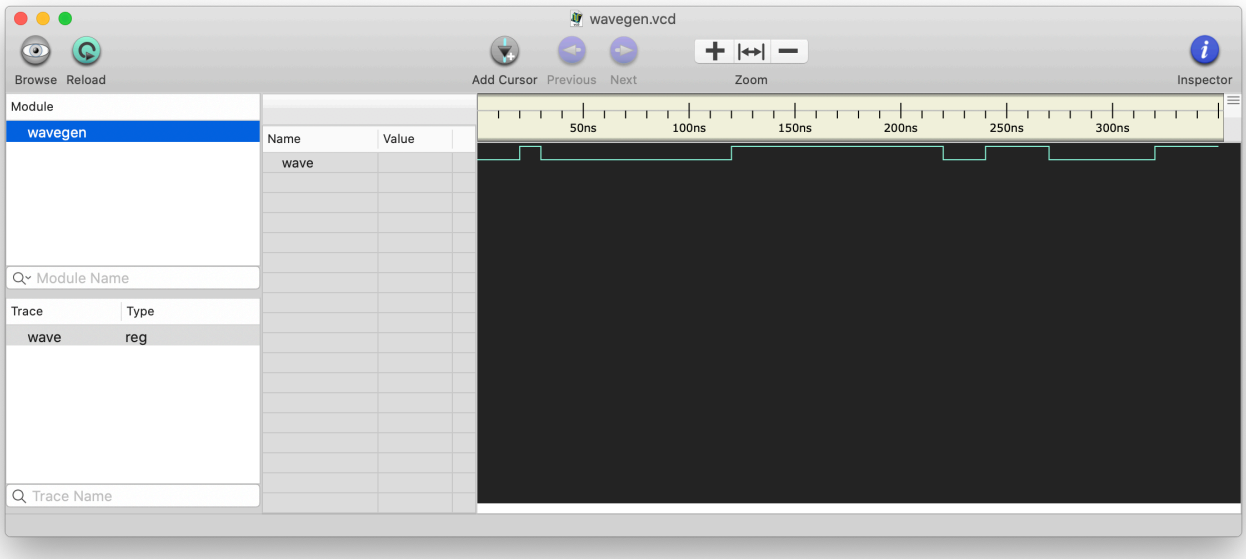

图 1: 测试波形图 *2.1*

2 第 2 题

(1) 设计模块

```
//File: Encoder8x3.v
//517021911065 刘浩文

`timescale 1 ns / 1 ns

module Encoder8x3 (
  output reg [2:0] code,
  input [7:0] data

);
always @(*) begin

  case (data)
    8'b0000_0001: code = 3'd0;
    8'b0000_0010: code = 3'd1;
    8'b0000_0100: code = 3'd2;
    8'b0000_1000: code = 3'd3;
    8'b0001_0000: code = 3'd4;
    8'b0010_0000: code = 3'd5;
    8'b0100_0000: code = 3'd6;
    8'b1000_0000: code = 3'd7;

    default :   code = 3'dx;
  endcase

end

endmodule
```

(2) 测试模块

```
//File: tb_Encoder8x3.v

`timescale 1 ns / 1 ns
`include "Encoder8x3.v"

module tb_Encoder8x3;

  wire [2:0] p_code;
  reg [7:0] p_data;

  Encoder8x3 m_enc8x3(.code(p_code),
    .data(p_data));

  initial
  begin
    /*$dumpfile("Encoder8x3.vcd");
    $dumpvars(0, tb_Encoder8x3);*/
    p_data = 8'b1;

    repeat(7) #10 p_data = p_data << 1;

    #10 $stop;
  end

  initial
  begin
    $monitor("data = %b, code = %d.", p_data, p_code);
  end

endmodule
```

(3) 测试波形图

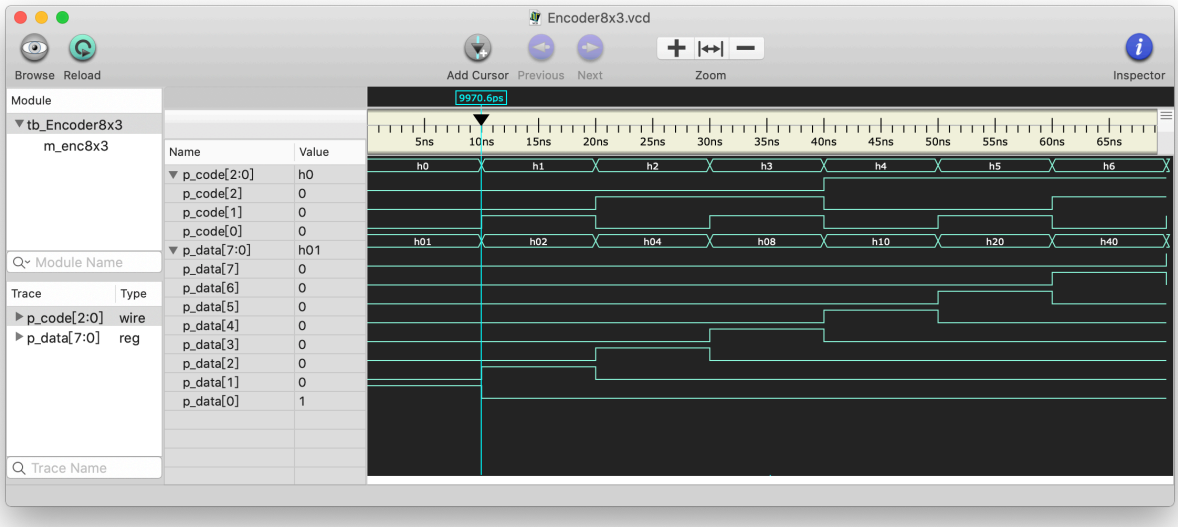

图 2: 测试波形图 *2.2*

(4) 显示输出

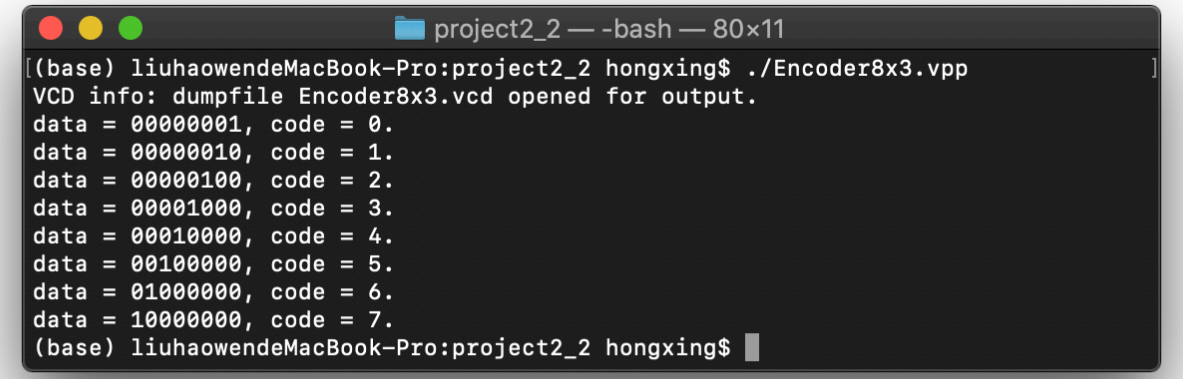

图 3: 显示输出 *2.2*

3 第 3 题

(1) 设计模块

```
//File: mux2x1.v
//517021911065 刘浩文

`timescale 1 ns / 1 ns

module mux2x1 (
  output dout,
  input sel,
  input [1:0] din

);
bufif0 b0(dout, din[0], sel);
bufif1 b1(dout, din[1], sel);

endmodule
```

```
//File: mux4x1.v

`timescale 1 ns / 1 ns
`include "mux2x1.v"

module mux4x1 (
  output dout,
  input [1:0] sel,
  input [3:0] din

);
wire [1:0] dm;
```

```
13
14 wire [3:0] din1 = {din[3], din[1], din[2], din[0]};
15
16 mux2x1 m1(.dout(dm[0]),
17   .sel(sel[1]),
18   .din(din1[1:0]));
19
20 mux2x1 m2(.dout(dm[1]),
21   .sel(sel[1]),
22   .din(din1[3:2]));
23
24 mux2x1 m3(.dout(dout),
25   .sel(sel[0]),
26   .din(dm));
27
28 endmodule
```

(2) 测试模块

```
1  //File: tb_mux2x1.v
2
3  `timescale 1 ns / 1 ns
4  `include "mux2x1.v"
5
6  module tb_mux2x1;
7
8    wire p_dout;
9
10   reg [1:0] p_din;
11   reg p_sel;
12
13   mux2x1 m_m2x1(.dout(p_dout),
14     .sel(p_sel),
15     .din(p_din));
16
17   initial
18   begin
19     /*$dumpfile("mux2x1.vcd");
20     $dumpvars(0, tb_mux2x1);*/
21
22     p_din = 2'b10;
23     p_sel = 1'b0;
24     #10 p_sel = 1'b1;
25
26     #10 $stop;
27   end
28
29   initial
30   begin
31     $monitor("in0 = %b, in1 = %b, s = %b, out = %b.", p_din[0], p_din[1], p_sel, p_dout);
32   end
33
34 endmodule
```

```
1  //File: tb_mux4x1.v
2  `timescale 1 ns / 1 ns
3  `include "mux4x1.v"
```

```
4
5  module tb_mux4x1;
6
7    wire p_dout;
8    reg [3:0] p_din;
9    reg [1:0] p_sel;
10
11   mux4x1 m_m4x1(.dout(p_dout),
12     .sel(p_sel),
13     .din(p_din));
14
15   initial
16   begin
17     /*$dumpfile("mux4x1.vcd");
18     $dumpvars(0, tb_mux4x1);*/
19     p_din = 4'b0001;
20     repeat(4)
21     begin
22       p_sel = 2'b00;
23       #10 p_sel = 2'b01;
24       #10 p_sel = 2'b10;
25       #10 p_sel = 2'b11;
26       #10 p_din = p_din << 1;
27     end
28     #10 $stop;
29   end
30
31   initial
32   begin
33     $monitor("in = %b, s = %b, out = %b.", p_din, p_sel, p_dout);
34   end
35 endmodule
```

(3) 测试波形图

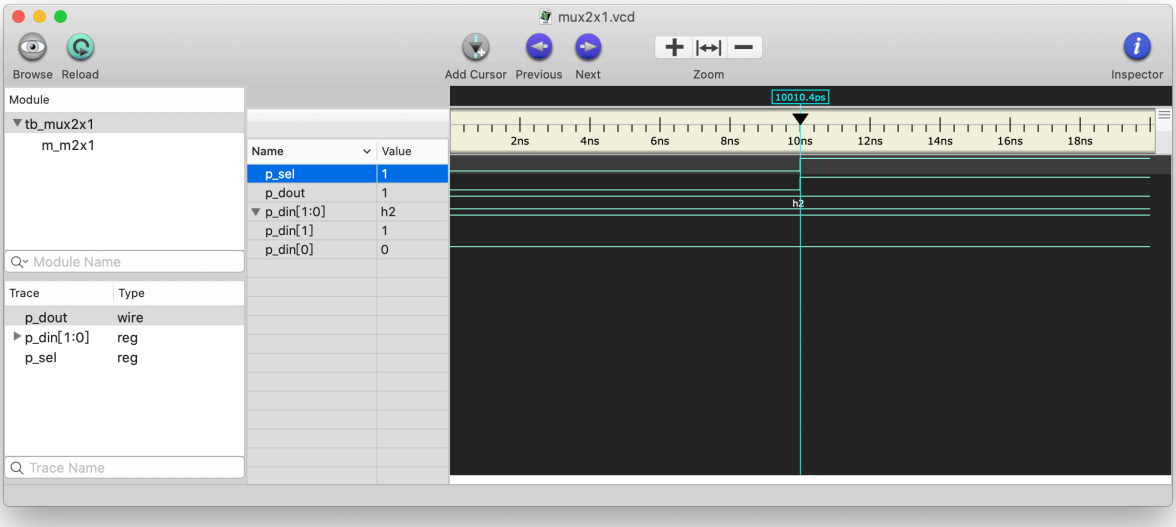

图 4: 测试波形图 $mux2x1$

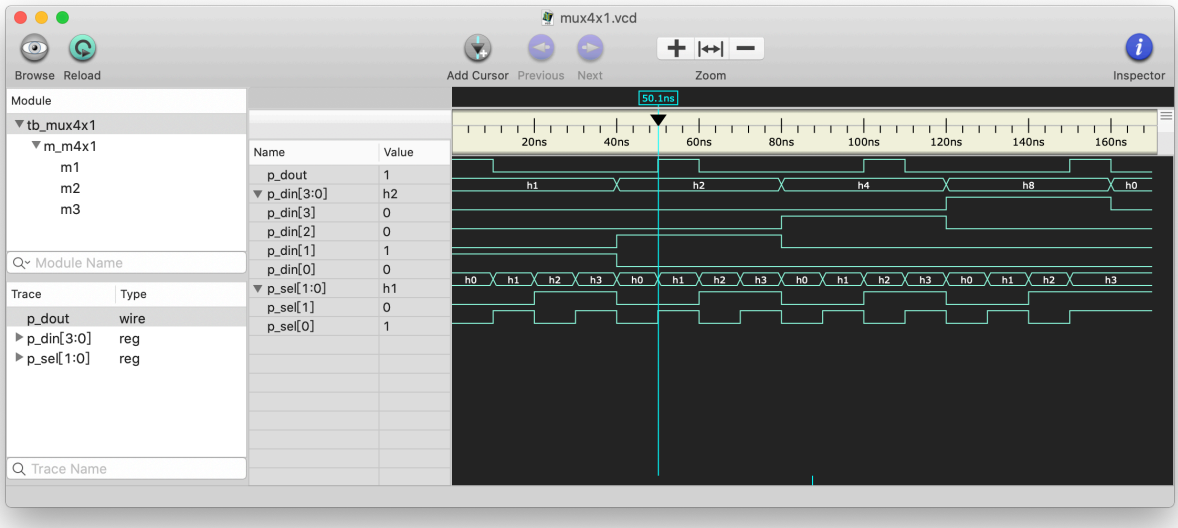

图 5: 测试波形图 $mux4x1$

(4) 显示输出

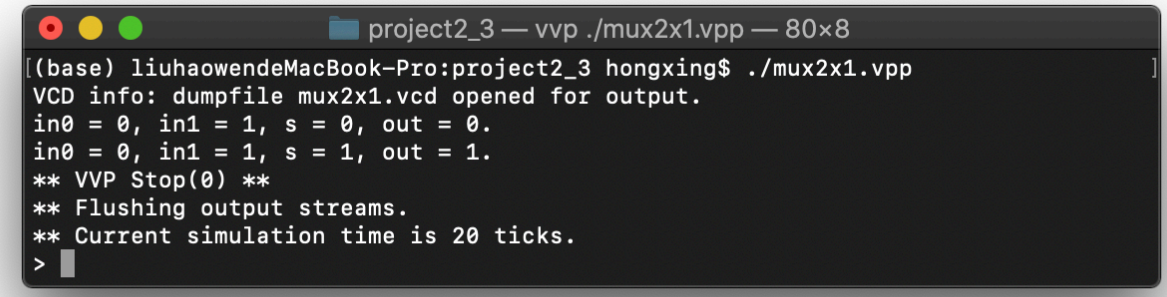

图 6: 显示输出 $mux2x1$

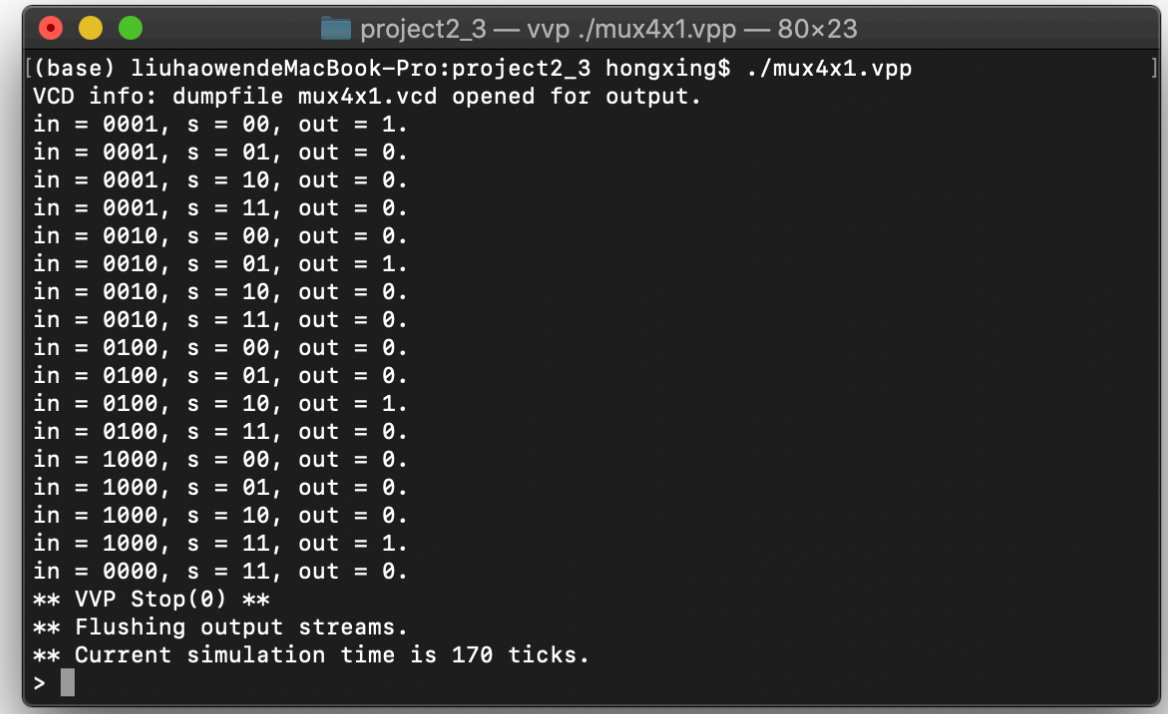

图 7: 显示输出 $mux4x1$

4 第 4 题

(1) 设计模块

```
//File: comb_str.v

`timescale 1 ns / 1 ns

module comb_str (
  output Y,
  input A, B, C, D

);
wire u1, u2, u3, u4;

not n1(u1, D);
and a1(u4, B, C, u1);
or o1(u3, A, D);
not n2(u2, u3);
and a2(Y, u2, u4);

endmodule
```

```
//File: comb_dataflow.v

`timescale 1 ns / 1 ns

module comb_dataflow (
  output Y,
  input A, B, C, D

);
wire u1, u2, u3, u4;

assign  u1 = ~D,   u4 = B & C & u1,
    u3 = A | D, u2 = ~u3,
    Y = u2 & u4;

endmodule
```

```
//File: comb_behavior.v

`timescale 1 ns / 1 ns

module comb_behavior (
  output reg Y,
  input A, B, C, D

);
reg u1, u2, u3, u4;

always @(*) begin

  u1 = ~D;
  u4 = B & C & u1;
  u3 = A | D;
  u2 = ~u3;
  Y = u2 & u4;

end

endmodule
```

```
//File: comb_prim.v

`timescale 1 ns / 1 ns

primitive prim_udp(output Y,
  input A,
  input B,
  input C,
  input D
  );

table
  //  A B C D : Y
    0 0 0 0 : 0;
    0 0 0 1 : 0;
    0 0 1 0 : 0;
    0 0 1 1 : 0;
```

```
    0 1 0 0 : 0;
    0 1 0 1 : 0;
    0 1 1 0 : 1;
    0 1 1 1 : 0;
    1 0 0 0 : 0;
    1 0 0 1 : 0;
    1 0 1 0 : 0;
    1 0 1 1 : 0;
    1 1 0 0 : 0;
    1 1 0 1 : 0;
    1 1 1 0 : 0;
    1 1 1 1 : 0;
endtable

endprimitive


module comb_prim (
  output Y,
  input A, B, C, D

);
prim_udp(Y, A, B, C, D);

endmodule
```

(2) 测试模块

```
//File: testbench_comb.v

`timescale 1 ns / 1 ns
`include "comb_str.v"
`include "comb_dataflow.v"
`include "comb_behavior.v"
`include "comb_prim.v"

module testbench_comb;

  wire [3:0] p_Y;

  reg p_A, p_B, p_C, p_D;

  reg [3:0] temp;

  comb_str c1(.Y(p_Y[0]),
    .A(p_A), .B(p_B), .C(p_C), .D(p_D));

  comb_dataflow c2(.Y(p_Y[1]),
    .A(p_A), .B(p_B), .C(p_C), .D(p_D));

  comb_behavior c3(.Y(p_Y[2]),
    .A(p_A), .B(p_B), .C(p_C), .D(p_D));

  comb_prim c4(.Y(p_Y[3]),
    .A(p_A), .B(p_B), .C(p_C), .D(p_D));

  initial
```

```
30    begin
31      /*$dumpfile("comb.vcd");
32      $dumpvars(0, testbench_comb);*/
33
34      temp = 4'b0;
35      repeat(15)
36      begin
37        {p_A, p_B, p_C, p_D} = temp;
38        #10 temp = temp + 1;
39      end
40      {p_A, p_B, p_C, p_D} = temp;
41
42      #10 $stop;
43
44    end
45
46    initial
47    begin
48      $monitor("At time %t, A = %b, B = %b, C = %b, D = %b, Y_str = %b, Y_df = %b, Y_bh = %b, Y_pr = %b.",
        $time, p_A, p_B, p_C, p_D, p_Y[0], p_Y[1], p_Y[2], p_Y[3]);
49    end
50
51  endmodule
```

(3) 测试波形图

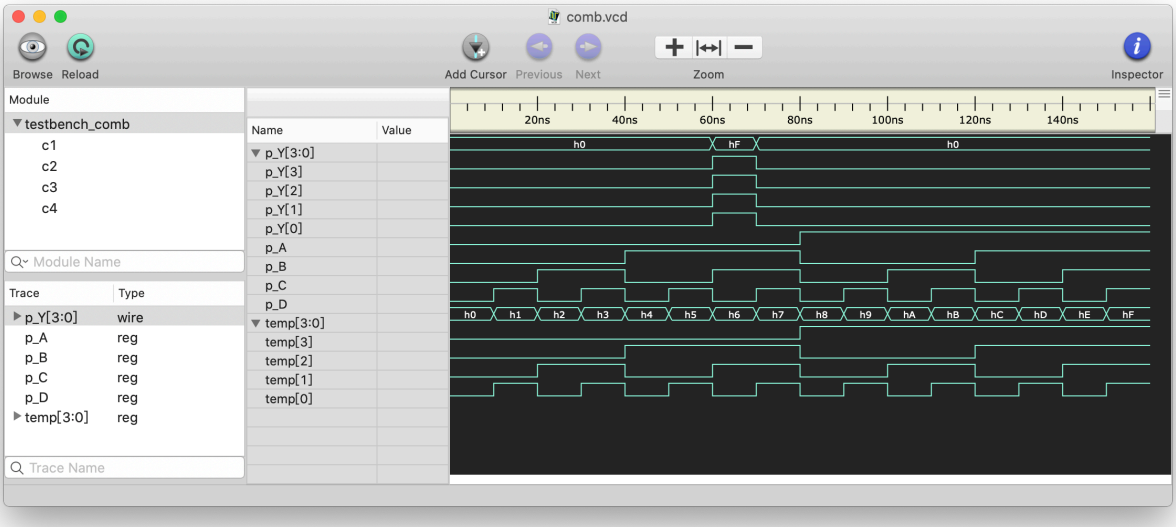

图 8: 测试波形图 *2.4*

(4) 显示输出

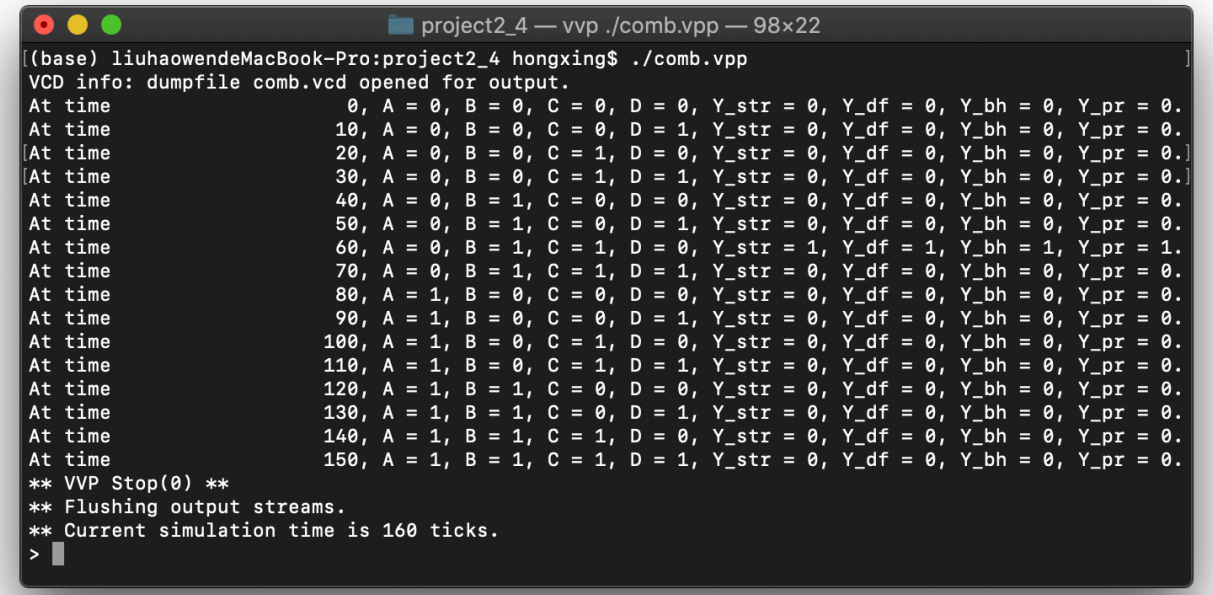

图 9: 显示输出 *2.4*

5 第 5 题

(1) 设计模块

```
//File: comb_Y.v

`timescale 1 ns / 1 ns

module comb_Y1 (
  output  Y,
  input A, B, C
);
assign  Y = ~A & ~B & C
    | ~A & B & ~C
    | A & ~B & ~C
    | A & ~B & C;

endmodule

module comb_Y2 (
  output  Y,
  input A, B, C, D
);
assign  Y = ~A & B & ~C & ~D
    | ~A & B & ~C & D
    | ~A & B & C & ~D
    | ~A & B & C & D
    | A & ~B & C & D
    | A & B & ~C & ~D
```

```
    | A & B & ~C & D;

endmodule
```

(2) 测试模块

```
//File: tb_comb_Y1.v

`timescale 1 ns / 1 ns
`include "comb_Y.v"

module tb_comb_Y1;

  wire p_Y;

  reg p_A, p_B, p_C;

  reg [2:0] temp;

  comb_Y1 c1(
    .Y(p_Y),
    .A(p_A),
    .B(p_B),
    .C(p_C)
    );

  initial
  begin
    /*$dumpfile("comb_Y1.vcd");
    $dumpvars(0, tb_comb_Y1);*/
    temp = 3'b0;
    repeat(7)
    begin
      {p_A, p_B, p_C} = temp;
      #10 temp = temp + 1;
    end
    {p_A, p_B, p_C} = temp;
    #10 $stop;
  end

  initial
  begin
    $monitor("At time %t, A = %b, B = %b, C = %b, Y = %b.", $time, p_A, p_B, p_C, p_Y);
  end
endmodule
```

```
//File: tb_comb_Y2.v

`timescale 1 ns / 1 ns
`include "comb_Y.v"

module tb_comb_Y2;

  wire p_Y;

  reg p_A, p_B, p_C, p_D;

```

```
12    reg [3:0] temp;
13
14    comb_Y2 c2(
15        .Y(p_Y),
16        .A(p_A),
17        .B(p_B),
18        .C(p_C),
19        .D(p_D)
20        );
21
22    initial
23    begin
24        /*$dumpfile("comb_Y2.vcd");
25        $dumpvars(0, tb_comb_Y2);*/
26
27        temp = 4'b0;
28        repeat(15)
29        begin
30            {p_A, p_B, p_C, p_D} = temp;
31            #10 temp = temp + 1;
32        end
33        {p_A, p_B, p_C, p_D} = temp;
34
35        #10 $stop;
36    end
37
38    initial
39    begin
40        $monitor("At time %t, A = %b, B = %b, C = %b, D = %b, Y = %b.", $time, p_A, p_B, p_C, p_D, p_Y);
41    end
42
43  endmodule
```

(3) 测试波形图

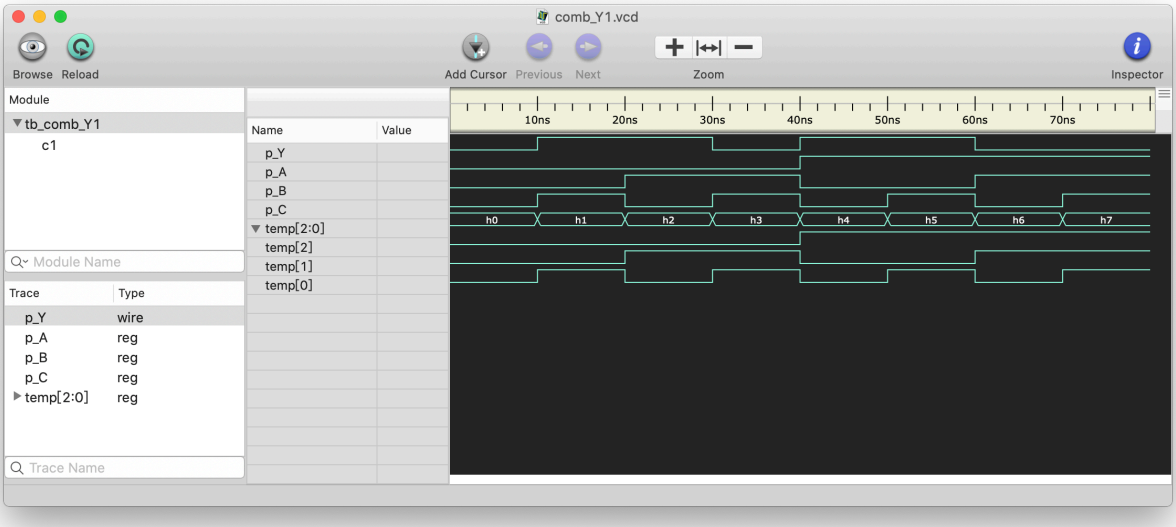

图 10: 测试波形图 $Y1$

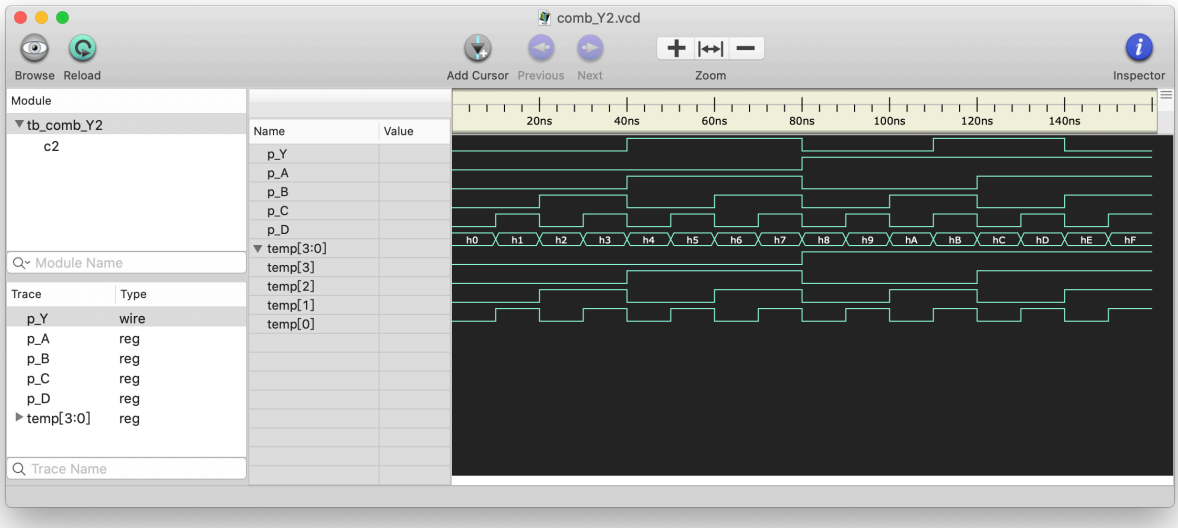

图 11: 测试波形图 $Y2$

(4) 显示输出

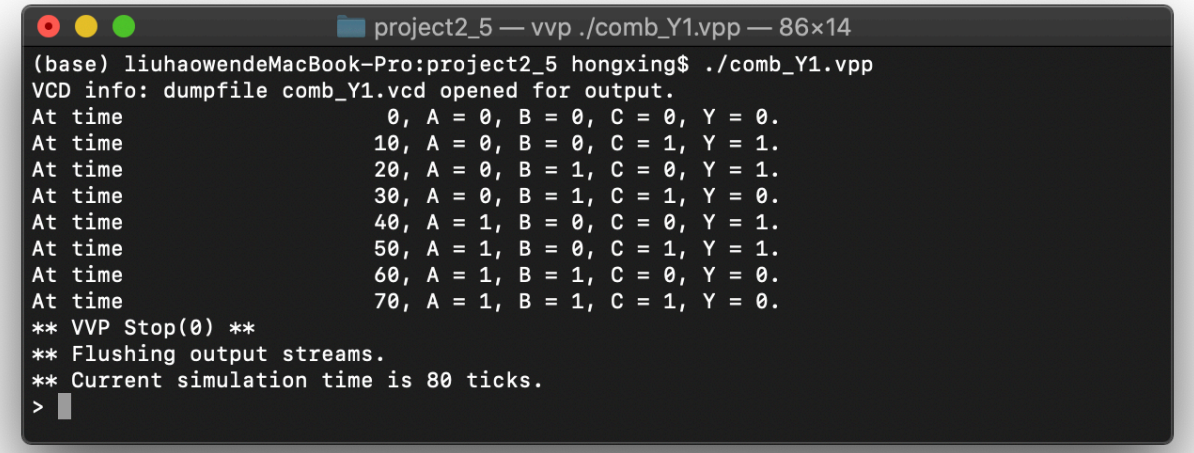

图 12: 显示输出 $Y1$

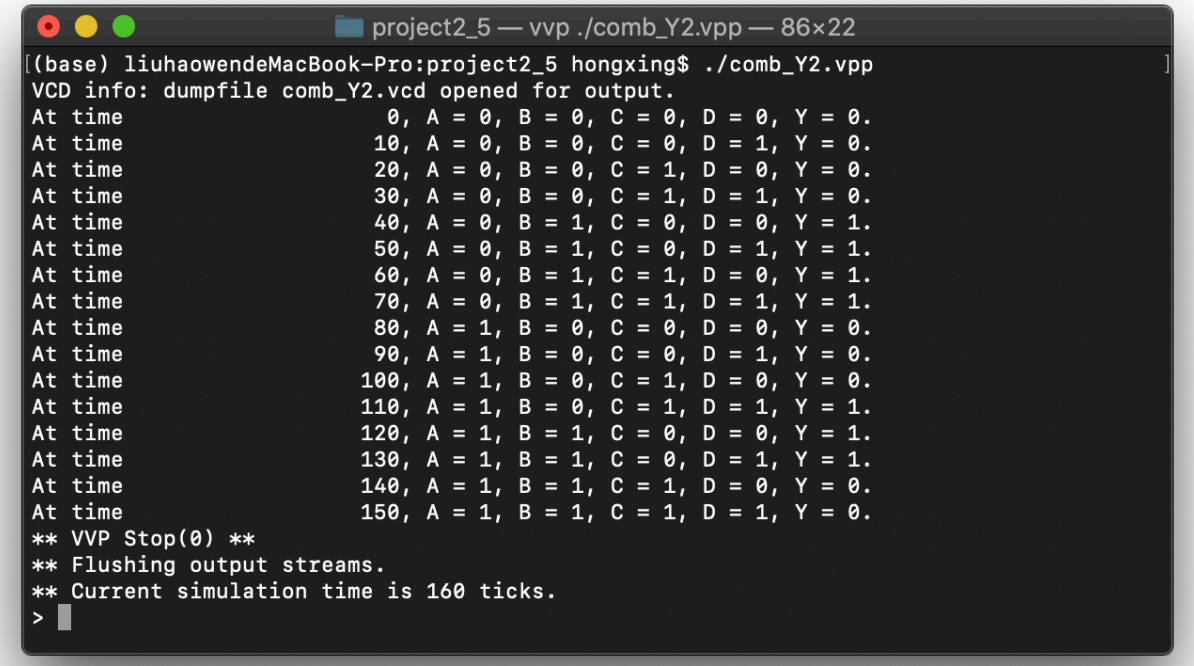

图 13: 显示输出 $Y2$

6 第 6 题

(1) 设计模块

```
1  //File: ones_count.v
2
3  `timescale 1 ns / 1 ns
4
5  module ones_count (
6    output [3:0] count,
7    input [7:0] dat_in
8
9  );
10 assign count =  dat_in[0] + dat_in[1] +
11         dat_in[2] + dat_in[3] +
12         dat_in[3] + dat_in[5] +
13         dat_in[6] + dat_in[7];
14
15 endmodule
```

(2) 测试模块

```
1  //File: tb_ones_count.v
2
3  `timescale 1 ns / 1 ns
```

```v
`include "ones_count.v"

module tb_ones_count;

  wire [3:0] p_cnt;

  reg [7:0] p_din;

  ones_count one1(
    .count(p_cnt),
    .dat_in(p_din)
    );

  initial
  begin
    /*$dumpfile("ones_count.vcd");
    $dumpvars(0, tb_ones_count);*/
    p_din = 8'b0;
    repeat(255)
    begin
      #10 p_din = p_din + 1;
    end

    #10 $stop;
  end

  initial
  begin
    $monitor("At time %tns, dat_in = %b, count = %d", $time, p_din, p_cnt);
  end

endmodule
```

(3) 测试波形图

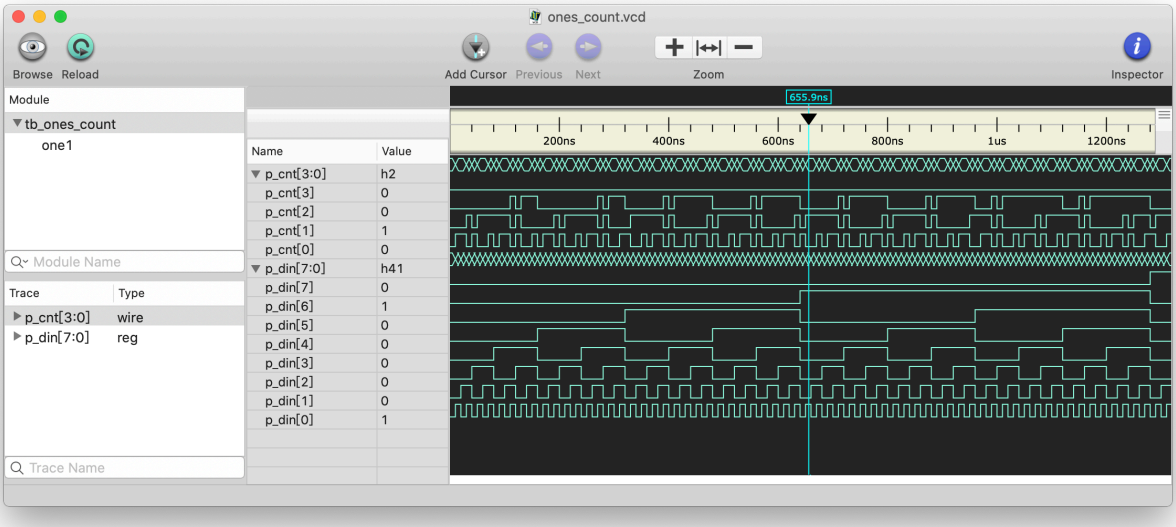

图 14: 测试波形图 *2.6(1)*

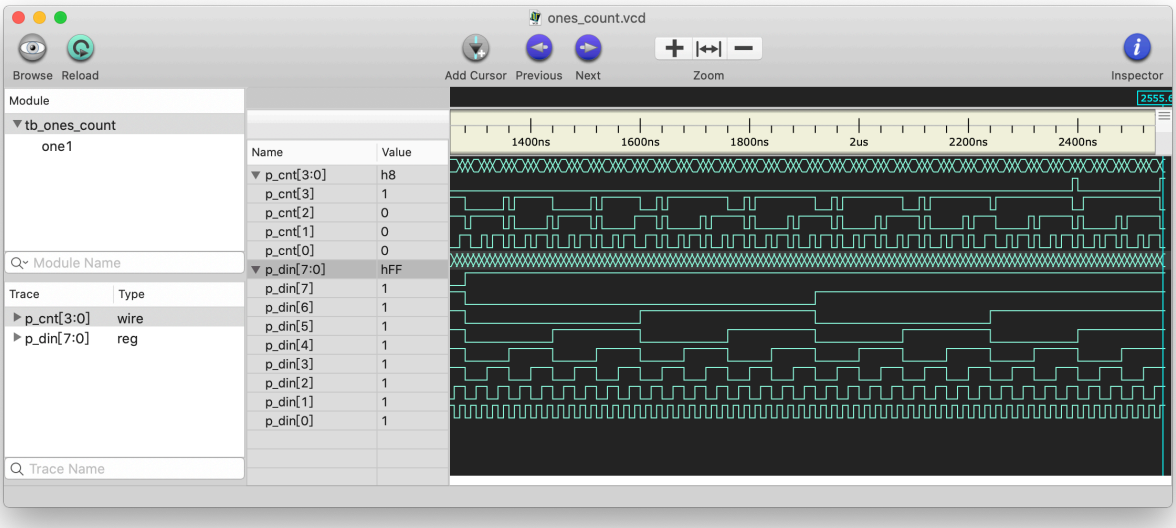

图 15: 测试波形图 *2.6(2)*

(4) 显示输出

```
project2_6 — vvp ./ones_count.vpp — 63×33
(base) liuhaowendeMacBook-Pro:project2_6 hongxing$ ./ones_count
.vpp
VCD info: dumpfile ones_count.vcd opened for output.
At time                    0ns, dat_in = 00000000, count =  0
At time                   10ns, dat_in = 00000001, count =  1
At time                   20ns, dat_in = 00000010, count =  1
At time                   30ns, dat_in = 00000011, count =  2
At time                   40ns, dat_in = 00000100, count =  1
At time                   50ns, dat_in = 00000101, count =  2
At time                   60ns, dat_in = 00000110, count =  2
At time                   70ns, dat_in = 00000111, count =  3
At time                   80ns, dat_in = 00001000, count =  2
At time                   90ns, dat_in = 00001001, count =  3
At time                  100ns, dat_in = 00001010, count =  3
At time                  110ns, dat_in = 00001011, count =  4
At time                  120ns, dat_in = 00001100, count =  3
At time                  130ns, dat_in = 00001101, count =  4
At time                  140ns, dat_in = 00001110, count =  4
At time                  150ns, dat_in = 00001111, count =  5
At time                  160ns, dat_in = 00010000, count =  0
At time                  170ns, dat_in = 00010001, count =  1
At time                  180ns, dat_in = 00010010, count =  1
At time                  190ns, dat_in = 00010011, count =  2
At time                  200ns, dat_in = 00010100, count =  1
At time                  210ns, dat_in = 00010101, count =  2
At time                  220ns, dat_in = 00010110, count =  2
At time                  230ns, dat_in = 00010111, count =  3
At time                  240ns, dat_in = 00011000, count =  2
At time                  250ns, dat_in = 00011001, count =  3
At time                  260ns, dat_in = 00011010, count =  3
At time                  270ns, dat_in = 00011011, count =  4
At time                  280ns, dat_in = 00011100, count =  3
At time                  290ns, dat_in = 00011101, count =  4
```

图 16: 显示输出 *2.6(1)*

```
project2_6 — vvp ./ones_count.vpp — 63×33
At time                  2260ns, dat_in = 11100010, count =  4
At time                  2270ns, dat_in = 11100011, count =  5
At time                  2280ns, dat_in = 11100100, count =  4
At time                  2290ns, dat_in = 11100101, count =  5
At time                  2300ns, dat_in = 11100110, count =  5
At time                  2310ns, dat_in = 11100111, count =  6
At time                  2320ns, dat_in = 11101000, count =  5
At time                  2330ns, dat_in = 11101001, count =  6
At time                  2340ns, dat_in = 11101010, count =  6
At time                  2350ns, dat_in = 11101011, count =  7
At time                  2360ns, dat_in = 11101100, count =  6
At time                  2370ns, dat_in = 11101101, count =  7
At time                  2380ns, dat_in = 11101110, count =  7
At time                  2390ns, dat_in = 11101111, count =  8
At time                  2400ns, dat_in = 11110000, count =  3
At time                  2410ns, dat_in = 11110001, count =  4
At time                  2420ns, dat_in = 11110010, count =  4
At time                  2430ns, dat_in = 11110011, count =  5
At time                  2440ns, dat_in = 11110100, count =  4
At time                  2450ns, dat_in = 11110101, count =  5
At time                  2460ns, dat_in = 11110110, count =  5
At time                  2470ns, dat_in = 11110111, count =  6
At time                  2480ns, dat_in = 11111000, count =  5
At time                  2490ns, dat_in = 11111001, count =  6
At time                  2500ns, dat_in = 11111010, count =  6
At time                  2510ns, dat_in = 11111011, count =  7
At time                  2520ns, dat_in = 11111100, count =  6
At time                  2530ns, dat_in = 11111101, count =  7
At time                  2540ns, dat_in = 11111110, count =  7
At time                  2550ns, dat_in = 11111111, count =  8
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 2560 ticks.
```

图 17: 显示输出 *2.6(2)*

7 第 7 题

(1) 设计模块

```
//File: dec_counter.v

`timescale 10 ns / 1 ns

module dec_counter (
  output reg [3:0] counter,
  input clk,     // Clock
  input reset   // Asynchronous reset active low

);
always @(posedge clk)
begin
  if(~reset)
    counter <= 4'd0;
  else if(counter == 4'd10)
    counter <= 4'd0;
  else counter = counter + 4'd1;

end


endmodule
```

(2) 测试模块

```
//File: tb_dec_counter.v

`timescale 10 ns / 1 ns
`include "dec_counter.v"

module tb_dec_counter;

  wire [3:0] p_counter;

  reg p_clk, p_reset;

  dec_counter m_decc(
    .counter(p_counter),
    .clk     (p_clk),
    .reset   (p_reset)
    );

  initial
  begin
    /*$dumpfile("dec_counter.vcd");
    $dumpvars(0, tb_dec_counter);*/

    p_clk = 1'b0;
    forever
    begin
      #10 p_clk = ~p_clk;
    end

```

```
29    end
30
31    initial
32    begin
33      p_reset = 1'b0;
34      #100 p_reset = 1'b1;
35      #1000 $stop;
36
37    end
38
39    initial
40    begin
41      $monitor("At time %tns, reset = %b, counter = %d.", $time, p_reset, p_counter);
42    end
43
44  endmodule
```

(3) 测试波形图

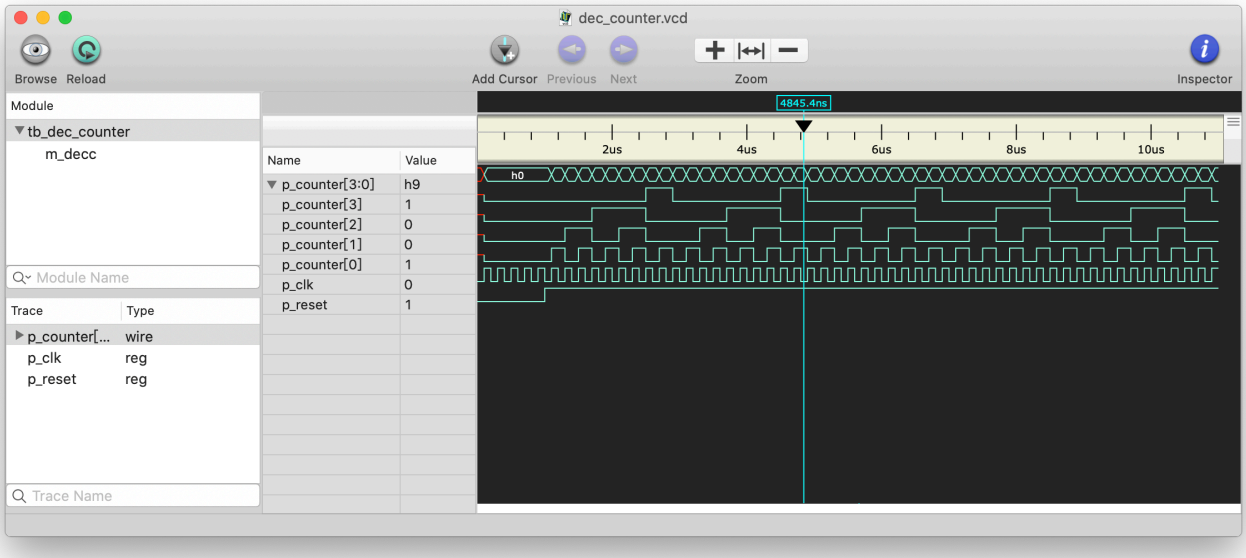

图 18: 测试波形图 *2.7*

(4) 显示输出

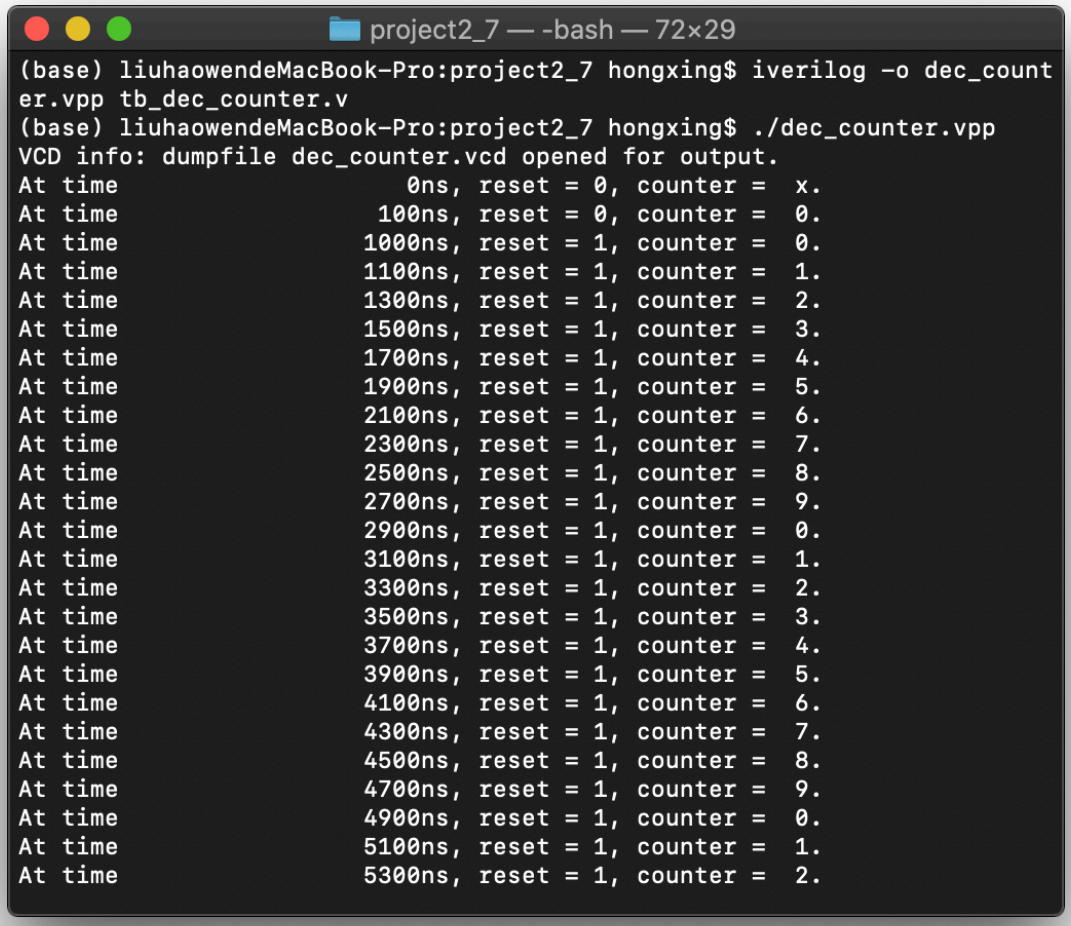

图 19: 显示输出 *2.7*

(5) **设计说明** 本题题目一开始说要求“设计一个 10 进制计数器的 $Verilog$ 模块”，但后面又说“计算器从 0 到 10 计数，然后返回到 0 重新开始计数”，如果按后者，出现一个输出为 10 的状态的话就是 11 进制了。所以我还是按题干的要求十进制计数，计数到 9 为止，不出现输出为 10 的状态。

8 第 8 题

(1) 设计模块

```
1 //File: comb_str.v
2
3 `timescale 1 ns / 1 ns
4
5 module comb_str (
```

```
  output  y,
  input sel,
  input A, B, C, D

);
wire u1, u2;

nand #3 na1(u1, A, B);
nand #4 na2(u2, C, D);
bufif0 b0(y, u1, sel);
bufif1 b1(y, u2, sel);

endmodule
```

(2) **测试模块**

```
//File: tb_comb_str.v

`timescale 1 ns / 1 ns
`include "comb_str.v"

module tb_comb_str;

  wire p_y;

  reg p_sel;
  reg [3:0] temp;

  comb_str m_cb1(
    .y(p_y),
    .sel(p_sel),
    .A(temp[0]),
    .B(temp[1]),
    .C(temp[2]),
    .D(temp[3])
    );

  initial
  begin
    /*$dumpfile("comb_str.vcd");
    $dumpvars(0, tb_comb_str);*/
    temp = 4'b0;
    repeat(16)
    begin
      p_sel = 1'b0;
      #3 p_sel = ~p_sel;
      #3 p_sel = ~p_sel;
      #3 p_sel = ~p_sel;
      #3 temp = temp + 4'b1;
    end

    $stop;
  end

  initial
  begin
    $monitor("At time %tns, A = %b, B = %b, C = %b, D = %b, sel = %b, y = %b.", $time, temp[0], temp[1],
```

```
    temp[2], temp[3], p_sel, p_y);
42  end
43
44 endmodule
```

(3) 测试波形图

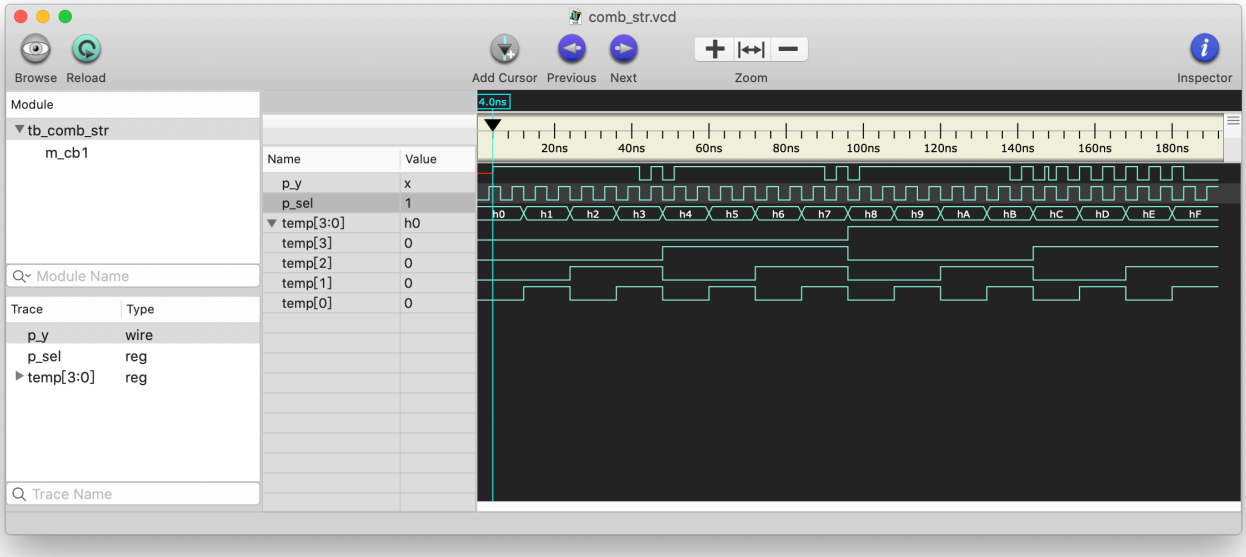

图 20: 测试波形图 *2.8*

(4) 显示输出

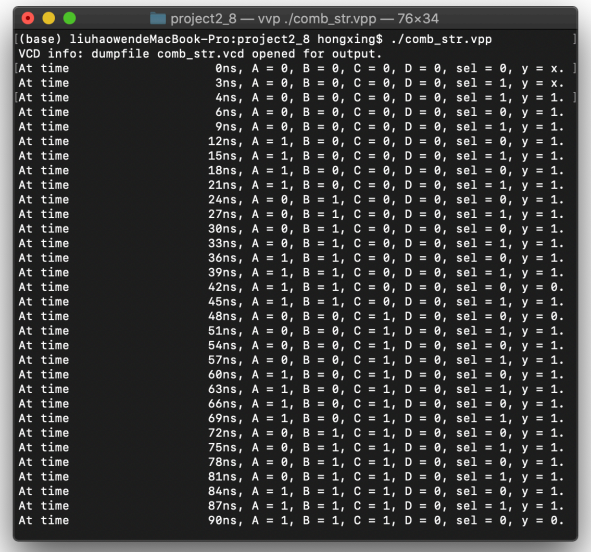

图 21: 显示输出 *2.8(1)*

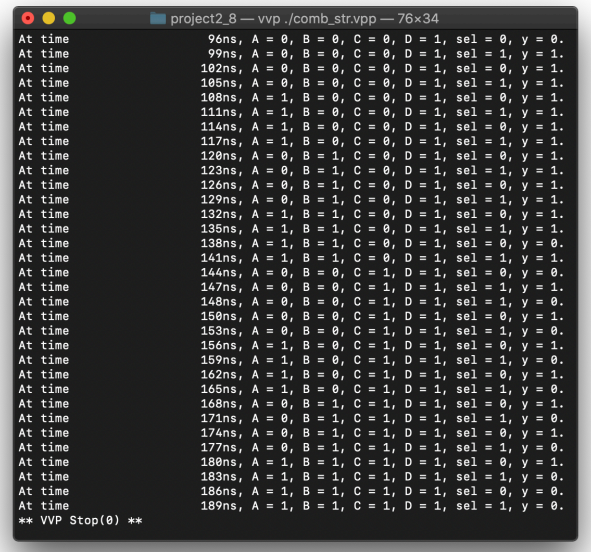

图 22: 显示输出 *2.8(2)*

9 第 9 题

(1) 设计模块

```
//File: LFSR.v

`timescale 1 ns / 1 ns

module LFSR (
  output reg [1:26] q,  // 26 bit data output.
  input clk,           // Clock input.
  input rst_n,         // Synchronous reset input active low.
  input load,        // Synchronous load input active high.
  input [1:26] din     // 26 bit parallel data input.

);
always @(posedge clk) begin
  if(~rst_n) q <= 26'b0;
  else begin
    if(load) q <= (|din) ? din : 26'b1;
    else begin
      if(!q) q <= 26'b1;
      else begin
        q[1] <= q[26];
        q[2] <= q[1] ^ q[26];
        q[3:7] <= q[2:6];
        q[8] <= q[7] ^ q[26];
        q[9] <= q[8] ^ q[26];
        q[10:26] <= q[9:25];
      end
    end
  end
end


endmodule
```

(2) 测试模块

```
//File: tb_LFSR.v

`timescale 1 ns / 1 ns
`include "LFSR.v"

module tb_LFSR;

  wire [1:26] p_q;

  reg p_clk, p_rstn, p_load;
  reg [1:26] p_din;

  LFSR L1(
    .q    (p_q),
    .clk  (p_clk),
    .rst_n(p_rstn),
    .load (p_load),
    .din  (p_din)
```

```
    );

  initial begin
    /*$dumpfile("LFSR.vcd");
    $dumpvars(0, tb_LFSR);*/

    p_clk = 1'b0;
    forever begin
      #10 p_clk = ~p_clk;
    end
  end

  initial begin
    p_rstn = 1'b0;
    #100 p_rstn = 1'b1;
  end

  initial begin
    p_load = 1'b1;
    p_din = 26'd31;
    #155 p_load = 1'b0;

    #455 p_load = 1'b1;
    p_din = 26'd30;
    #505 p_load = 1'b0;

    #1005 p_load = 1'b1;
    p_din = 26'b0;
    #1055 p_load = 1'b0;

    #1500 $stop;
  end

  initial begin
    $monitor("At time %tns, rst_n = %b, load = %b, q = %b.", $time, p_rstn, p_load, p_q);
  end

endmodule
```

(3) 测试波形图

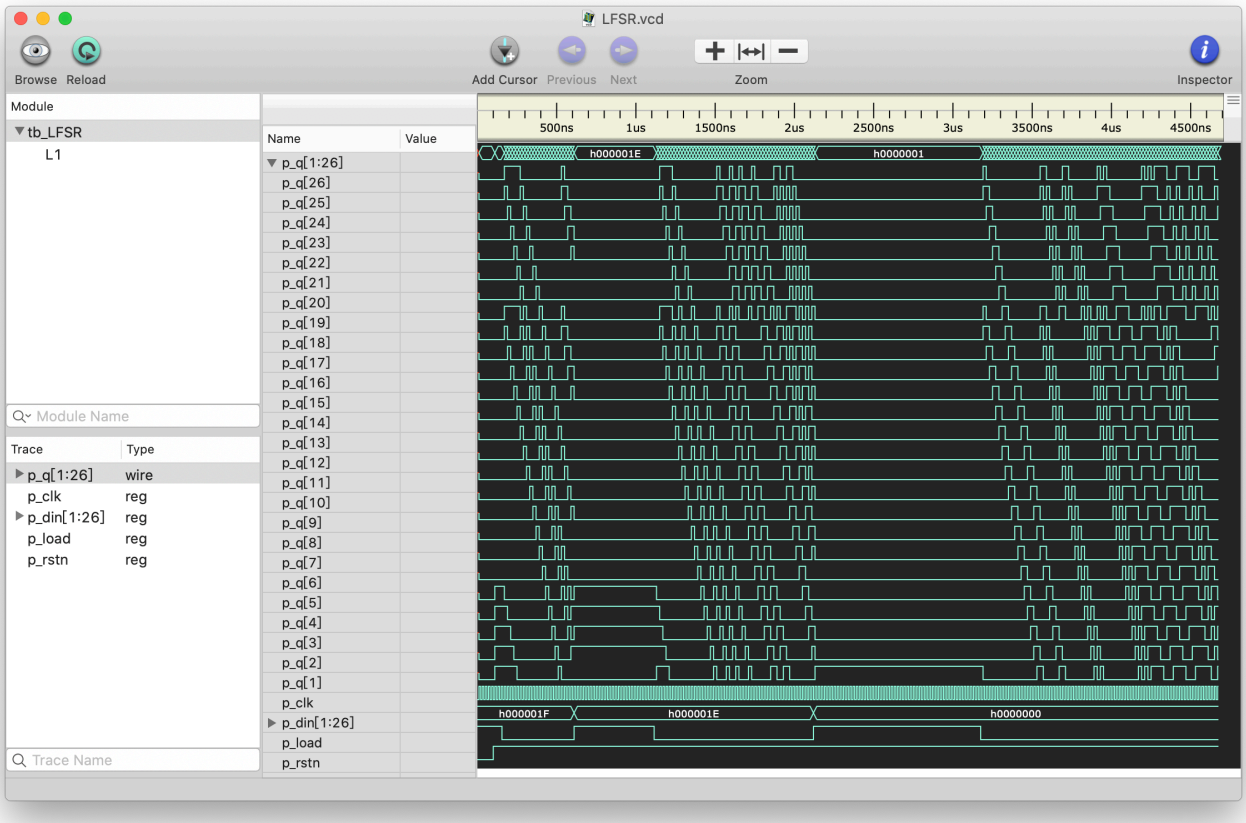

图 23: 测试波形图 *2.9*

(4) 显示输出

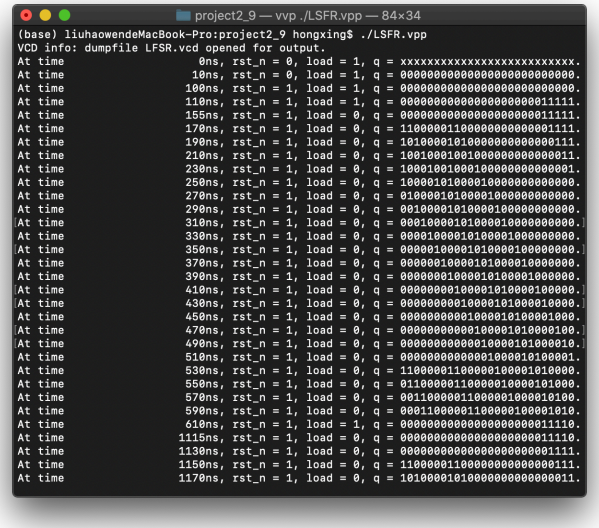

图 24: 显示输出 *2.9(1)*

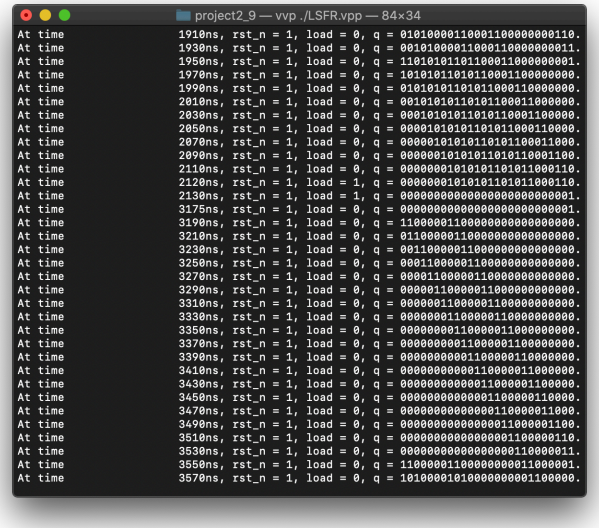

图 25: 显示输出 *2.9(2)*

10 第 10 题

(1) 设计模块

```
//File: filter.v
`timescale 1 ns / 1 ns

module filter (
  output reg sig_out,
  input clock,    // Clock
  input reset,  // Asynchronous reset active low
  input sig_in
);
reg [3:0] rgt;
reg j, k;

always @(posedge clock or negedge reset) begin
  if(~reset) begin
    sig_out <= 1'b0;
    rgt <= 4'b0;
  end else begin
    rgt[3] <= sig_in;
    rgt[2] <= rgt[3];
    rgt[1] <= rgt[2];
    rgt[0] <= rgt[1];
    k <= ~rgt[2] & ~rgt[1] & ~rgt[0];
    j <= rgt[2] & rgt[1] & rgt[0];
    case ({j,k})
      2'b01:  sig_out <= 1'b0;
      2'b10:  sig_out <= 1'b1;
      2'b11:  sig_out <= ~sig_out;
      default : /* default */;
    endcase
  end
end

endmodule
```

(2) 测试模块

```
//File: tb_filter.v
`timescale 1 ns / 1 ns
`include "filter.v"

module tb_filter;

  wire o_sig;
  reg i_sig;
  reg p_clk, p_rst;

  filter f1(
    .sig_out(o_sig),
    .clock  (p_clk),
    .reset  (p_rst),
    .sig_in (i_sig)
    );

```

```
18   initial begin
19     /*$dumpfile("filter.vcd");
20     $dumpvars(0, tb_filter);*/
21     p_clk = 1'b0;
22     forever begin
23       #10 p_clk = ~p_clk;
24     end
25   end
26
27   initial begin
28     p_rst = 1'b0;
29     #100 p_rst = 1'b1;
30
31     #405 p_rst = 1'b0;
32     #105 p_rst = 1'b1;
33     #500 $stop;
34   end
35
36   initial begin
37     forever begin
38       i_sig = 1;
39       #100 i_sig = 0;
40       #100;
41     end
42   end
43
44   initial begin
45     $monitor("At time %t, reset = %b, sig_in = %b, sig_out = %b.", $time, p_rst, i_sig, o_sig);
46   end
47
48 endmodule
```

(3) 测试波形图

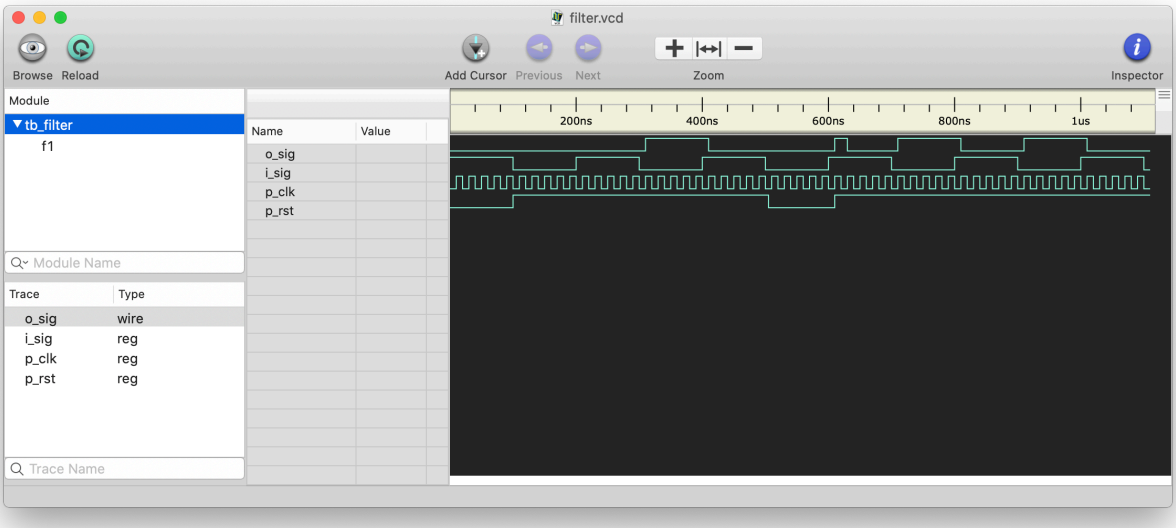

图 26: 测试波形图 *2.10*

(4) 显示输出

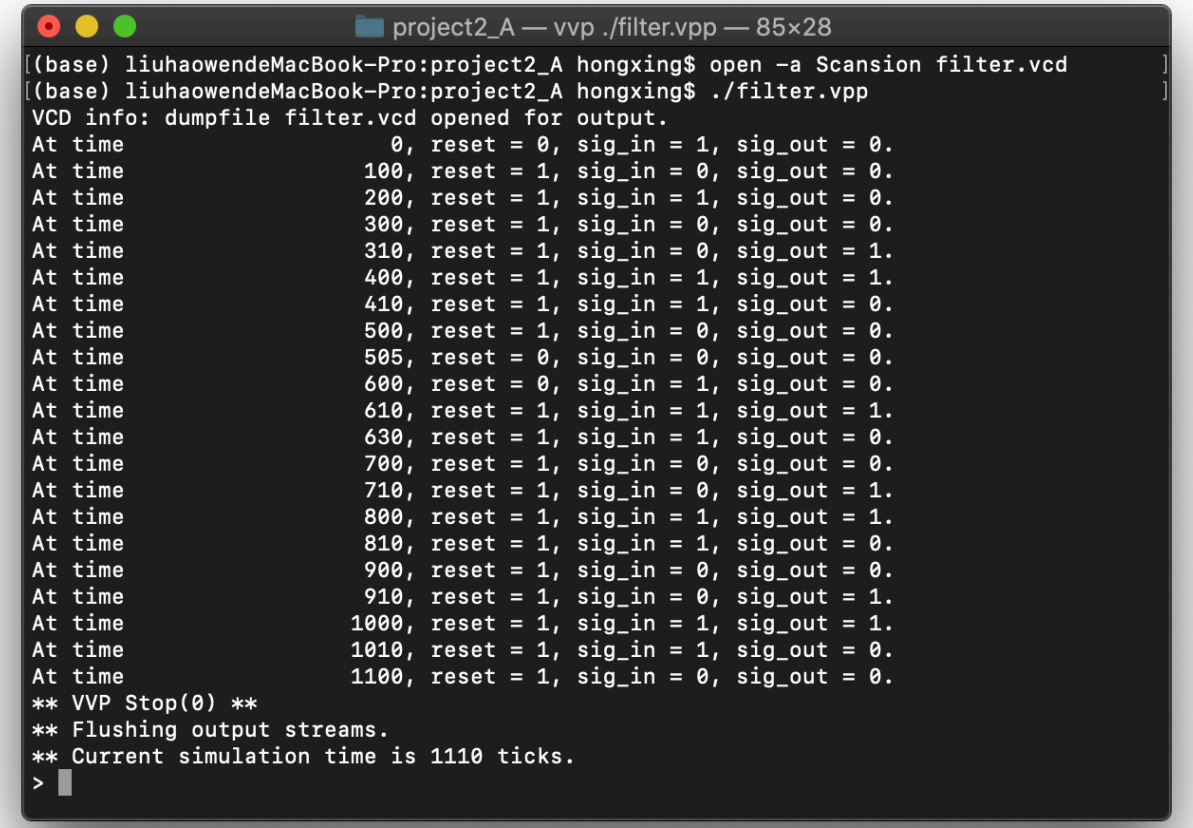

图 27: 显示输出 *2.10*

11 第 11 题

(1) 设计模块

```
1  //File: counter8b_updown.v
2
3  `timescale 1 ns / 1 ns
4
5  module counter8b_updown (
6    output reg [7:0] count,
7    input clk,     // Clock
8    input reset,   // Asynchronous reset active low
9    input dir
10
11 );
12 always @(posedge clk or negedge reset) begin
13   if(~reset) begin
14     count <= 8'b0;
```

```
  end else case (dir)
    1'b0: count = count - 1'b1;
    1'b1: count = count + 1'b1;

    default : /* default */;
  endcase
end

endmodule
```

(2) 测试模块

```
//File: tb_counter8b_updown.v

`timescale 1 ns / 1 ns
`include "counter8b_updown.v"

module tb_counter8b_updown;

  wire [7:0] p_cnt;

  reg p_clk, p_rst, p_dir;

  counter8b_updown cnt8b(
    .count(p_cnt),
    .clk   (p_clk),
    .reset(p_rst),
    .dir   (p_dir)
    );

  initial begin
    /*$dumpfile("counter8b_updown.vcd");
    $dumpvars(0, tb_counter8b_updown);*/

    p_clk = 1'b0;
    forever begin
      #10 p_clk = ~p_clk;
    end
  end

  initial begin
    p_rst = 1'b0;
    p_dir = 1'b1;
    #100 p_rst = 1'b1;

    #1955 p_rst = 1'b0;
    #100 p_rst = 1'b1;

    #5100 p_dir = 1'b0;

    #1955 p_rst = 1'b0;
    #100 p_rst = 1'b1;

    #5120 $stop;

  end

```

```
46    initial begin
47      $monitor("At time %tns, reset = %b, dir = %b, count = %d.", $time, p_rst, p_dir, p_cnt);
48    end
49
50  endmodule
```

(3) 测试波形图

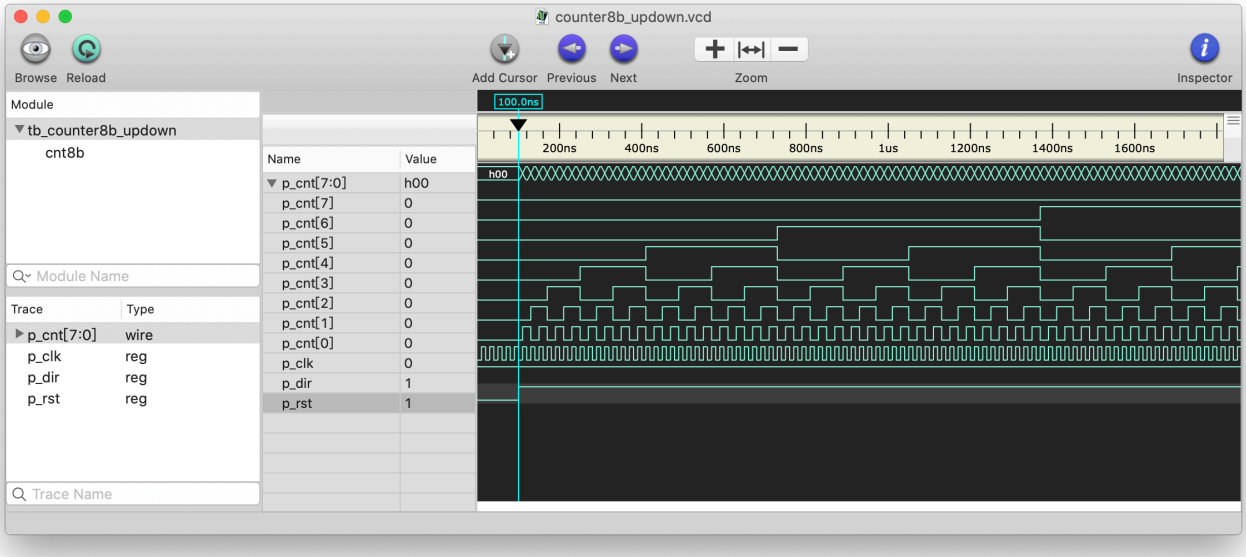

图 28: 测试波形图 *2.11*（递增计数）

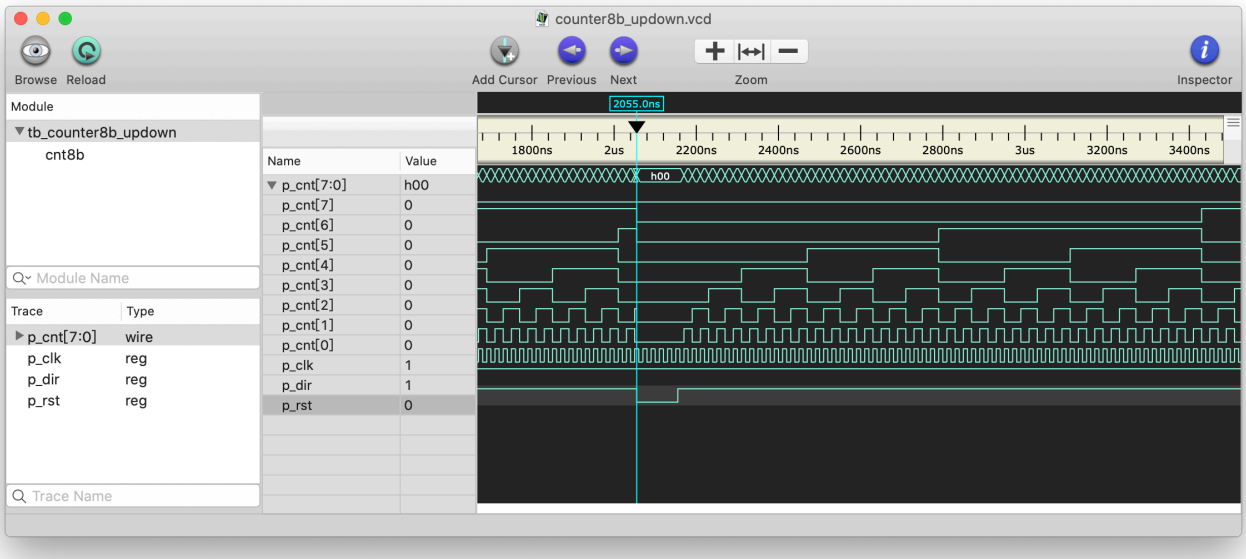

图 29: 测试波形图 *2.11*（异步复位）

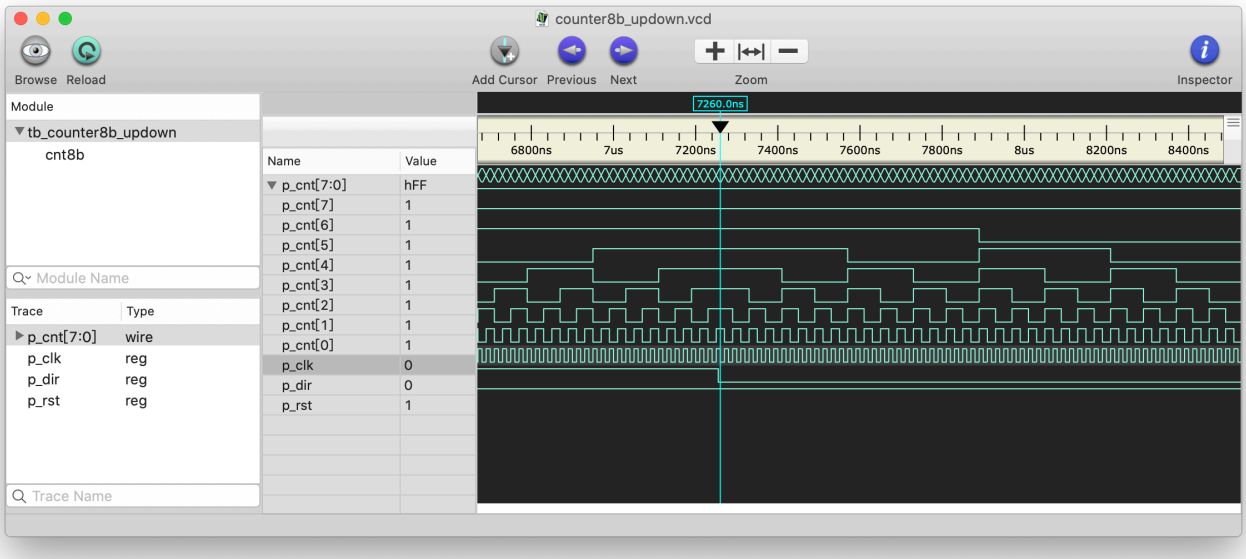

图 30: 测试波形图 *2.11*（递减计数）

(4) 显示输出

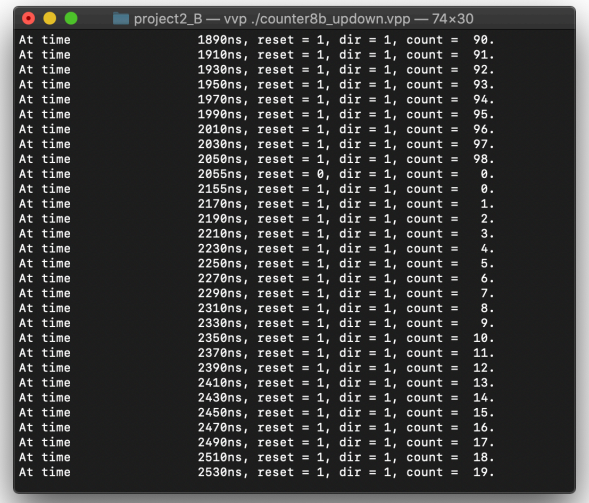

图 31: 显示输出 *2.11*（异步复位与递增计数）

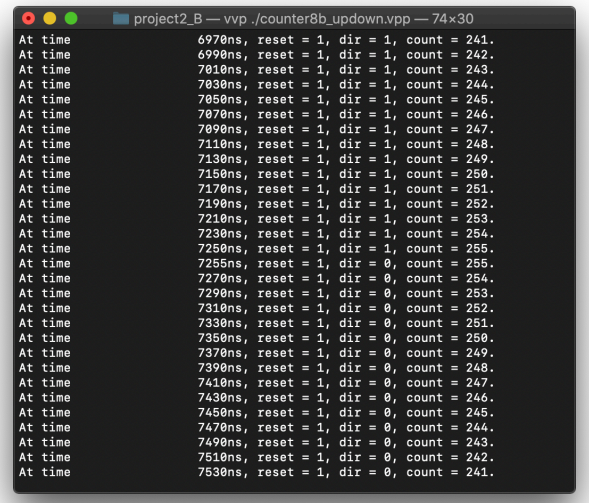

图 32: 显示输出 *2.11*（递增计数与递减计数）

12 第 12 题

(1) 设计模块

```
1 //File: ALU.v
2
3 `timescale 1 ns / 1 ns
```

```
4
5  `define OP_AND     4'd0
6  `define OP_SUBSTR  4'd1
7  `define OP_SUBSTR_A 4'd2
8  `define OP_OR_AB   4'd3
9  `define OP_AND_AB  4'd4
10 `define OP_NOT_AB  4'd5
11 `define OP_EXOR    4'd6
12 `define OP_EXNOR   4'd7
13
14 module ALU (
15   output reg c_out,
16   output reg [7:0] sum,
17   input [3:0] oper,
18   input [7:0] a,
19   input [7:0] b,
20   input c_in
21 );
22 always @(*) begin
23   case (oper)
24     `OP_AND    : {c_out, sum} = a + b + c_in;
25     `OP_SUBSTR  : {c_out, sum} = a + ~b + c_in;
26     `OP_SUBSTR_A: {c_out, sum} = b + ~a + ~c_in;
27     `OP_OR_AB : {c_out, sum} = {1'b0, a | b};
28     `OP_AND_AB  : {c_out, sum} = {1'b0, a & b};
29     `OP_NOT_AB  : {c_out, sum} = {1'b0, (~a) & b};
30     `OP_EXOR  : {c_out, sum} = {1'b0, a ^ b};
31     `OP_EXNOR : {c_out, sum} = {1'b0, a ~^ b};
32      default    : {c_out, sum} = 9'bx;
33   endcase
34
35 end
36
37 endmodule
```

(2) 测试模块

```
1  //tb_ALU.v
2
3  `timescale 1 ns / 1 ns
4  `include "ALU.v"
5
6  module tb_ALU;
7
8    wire [7:0] p_sum;
9    wire p_cout;
10
11   reg [7:0] p_a, p_b;
12   reg [3:0] p_op;
13   reg p_cin;
14
15   ALU A1(
16     .c_out(p_cout),
17     .sum  (p_sum),
18     .oper (p_op),
19     .a    (p_a),
20     .b    (p_b),
```

```
    .c_in (p_cin)
    );

  initial begin
    /*$dumpfile("ALU.vcd");
    $dumpvars(0, tb_ALU);*/

    p_a = ($random) % 256;
    {p_cin, p_b} = ($random) % 512;

    p_op = 4'd0;

    repeat(79) begin
      #10;

      p_a = ($random) % 256;
      {p_cin, p_b} = ($random) % 512;

      p_op = p_op + 1'b1;
    end

    #10 $stop;
  end

  initial begin
    $monitor("oper = %d, a = %b, b = %b, c_out = %b, sum = %b.", p_op, p_a, p_b, p_cout, p_sum);
  end

endmodule
```

(3) 显示输出

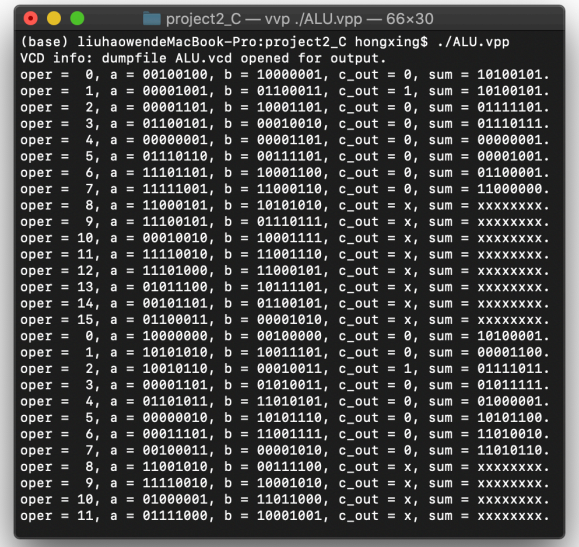

图 33: 显示输出 *2.12 (1)*

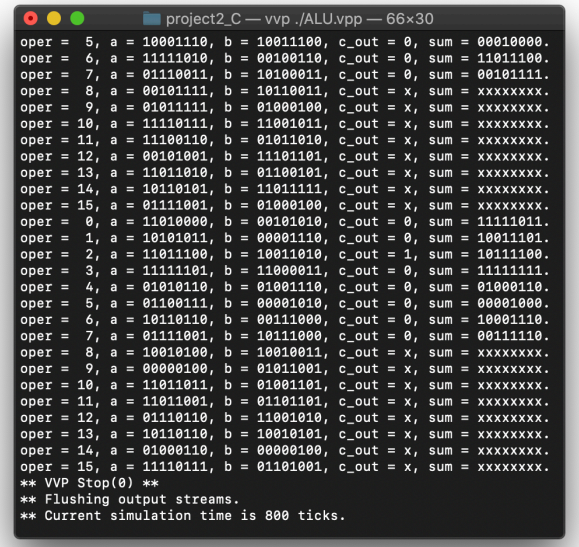

图 34: 显示输出 *2.12 (2)*

(4) **测试波形图**

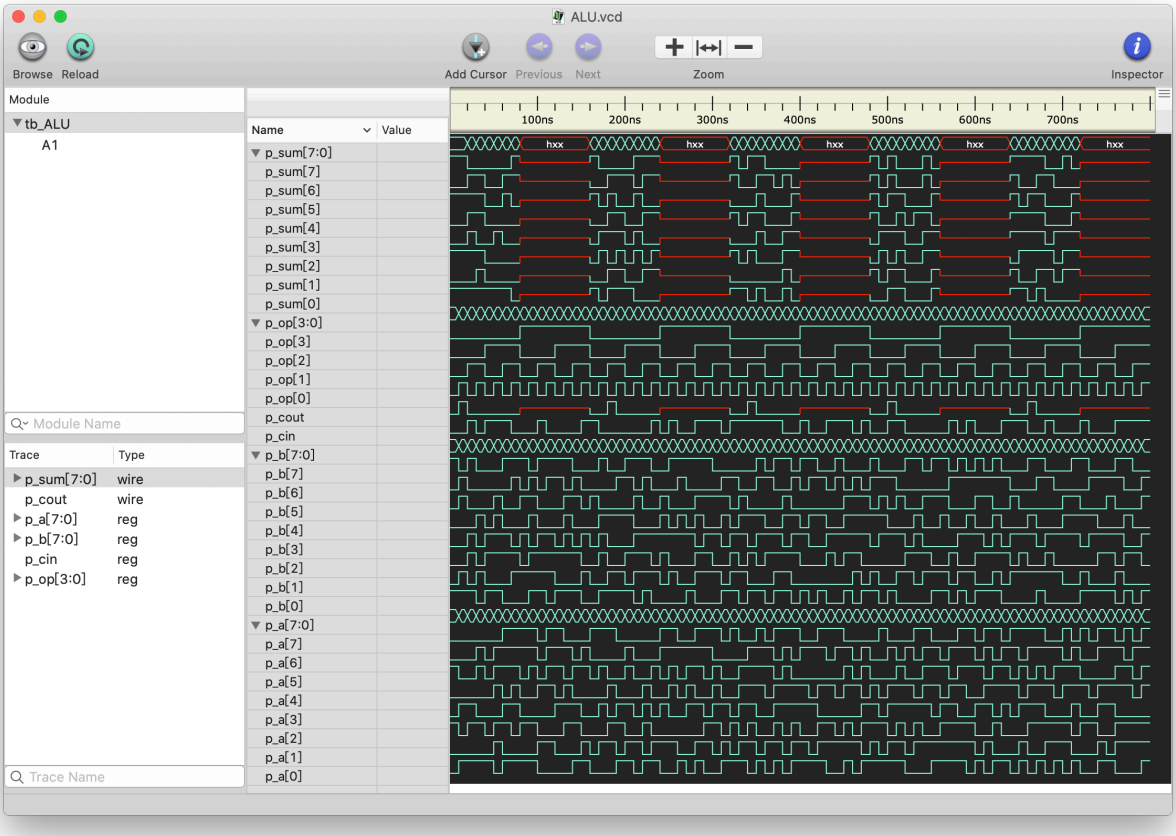

图 35: 测试波形图 *2.12*

(5) **设计说明** 本设计中 *oper* 取四位，因此仿真时会有大于 7 的 *oper* 输入，此时采用 $default$ 输出 $8'dx$。

13 第 13 题

(1) **设计模块**

```
//File: shift_counter.v

`timescale 1 ns / 1 ns

module shift_counter (
  output reg [7:0] count,
  input clk,     // Clock
  input reset   // Synchronous reset active low

);
reg [13:0] temp;

always @(posedge clk) begin
```

```
  if(~reset) begin
    count <= 8'b1;
    temp <= 14'b1;
  end else begin
    temp <= {temp[12:0], temp[13]};
    if(|temp[13:8])
      count <= {temp[7], temp[8], temp[9], temp[10], temp[11], temp[12], temp[13], 1'b0};
    else count <= temp[7:0];
  end
end

endmodule
```

(2) **测试模块**

```
//tb_shift_counter.v

`timescale 1 ns / 1 ns
`include "shift_counter.v"

module tb_shift_counter;

  wire [7:0] p_cnt;

  reg p_clk, p_rst;

  shift_counter shc1(
    .count(p_cnt),
    .clk  (p_clk),
    .reset(p_rst)
    );

  initial begin
    /*$dumpfile("shift_counter.vcd");
    $dumpvars(0, tb_shift_counter);*/

    p_clk = 1'b0;
    forever begin
      #10 p_clk = ~p_clk;
    end
  end

  initial begin
    p_rst = 1'b0;
    #55 p_rst = 1'b1;

    #2000 $stop;
  end

  initial begin
    $monitor("At time %tns, reset = %b, count = %b", $time, p_rst, p_cnt);
  end

endmodule
```

(3) **测试波形图**

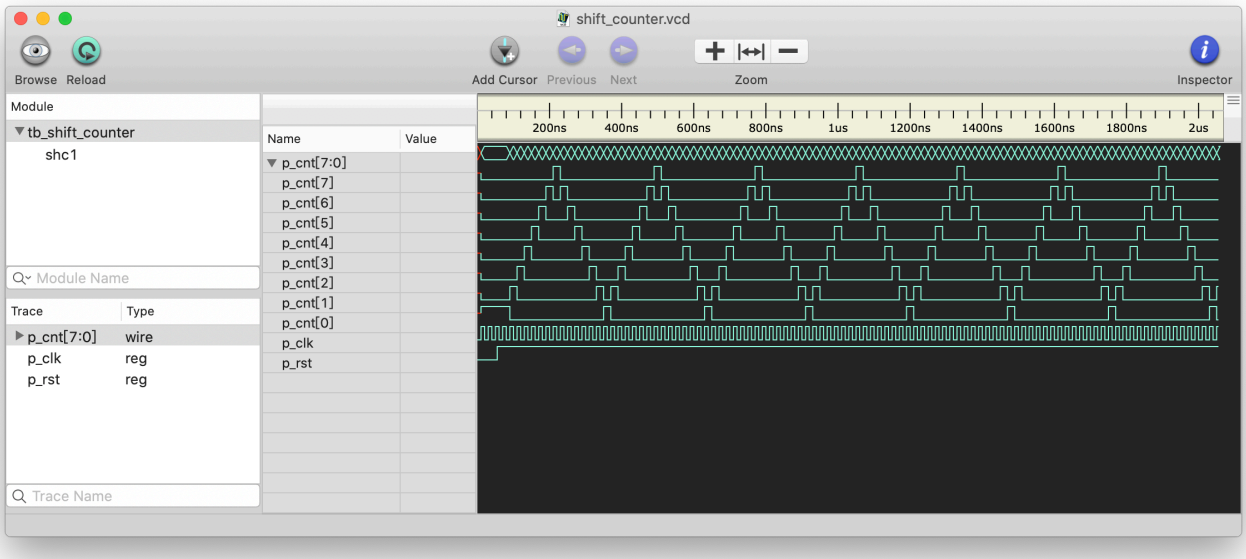

图 36: 测试波形图 *2.13*

(4) **显示输出**

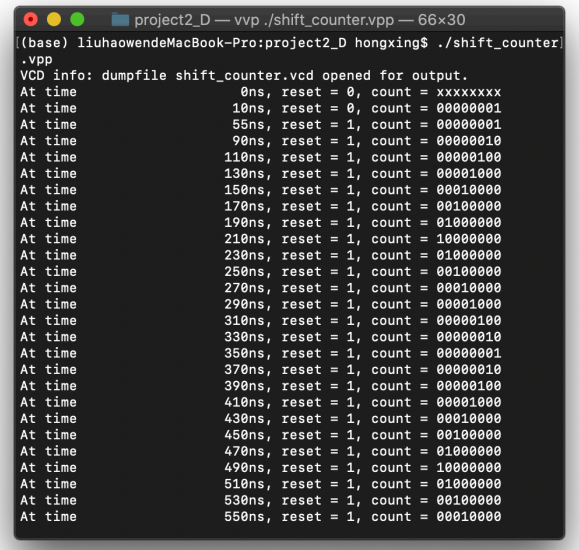

图 37: 显示输出 *2.13 (1)*

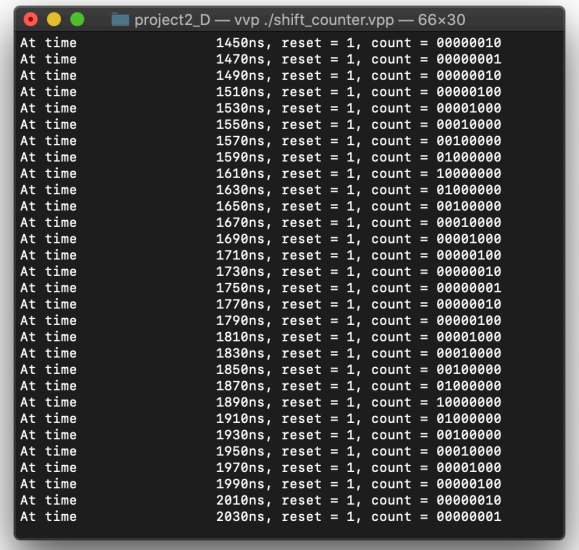

图 38: 显示输出 *2.13 (2)*

(5) **设计说明** 本计数器的设计思想类似于“反折”：新建一个 14 位循环左移寄存器组，将这个 14 位寄存器组映射到目标的 8 位计数器上，映射规则是：最低 8 位恒等映射，最高 6 位反折映射到目标 8 位计数器的除最高位和最低位的中间 6 位上，如设计模块代码第 20 行，由此完成目的功能。

14 第 14 题

(1) 设计模块

```
//File: sram.v

`timescale 1 ns / 1 ns

module sram #(parameter DATA_WIDTH = 8, ADDR_WIDTH = 8)
(
  output [DATA_WIDTH - 1:0] dout,
  input [DATA_WIDTH - 1:0] din,
  input [ADDR_WIDTH - 1:0] addr,
  input wr,
  input rd,
  input cs

);
parameter RAM_DEPTH = 1 << ADDR_WIDTH;
reg [DATA_WIDTH - 1:0] mem [0:RAM_DEPTH - 1];

reg [DATA_WIDTH - 1:0] data;
reg flag;

always @(negedge rd, addr, posedge flag) begin
  if(~rd) begin
    flag <= 1;
    data <= mem[addr];
  end
  else data <= 8'bz;
end

always @(posedge wr) begin
  if(cs) begin
    if(~rd) flag <= 0;
    mem[addr] <= din;
    #1;
    if(~rd) flag <= 1;

  end
end

assign dout = (cs&&flag) ? data : 8'bz;

endmodule
```

(2) 测试模块

```
//File: tb_sram.v

`timescale 1 ns / 1 ns
`include "sram.v"

module tb_sram;

  wire [7:0] p_dout;

```

```
  reg [7:0] p_din;
  reg [7:0] p_addr;
  reg p_wr, p_rd, p_cs;

  sram sram1(
    .dout(p_dout),
    .din (p_din),
    .addr(p_addr),
    .wr  (p_wr),
    .rd  (p_rd),
    .cs  (p_cs)
    );

  initial begin
    /*$dumpfile("sram.vcd");
    $dumpvars(0, tb_sram);*/

    p_addr = 8'b0;
    repeat(200) begin
      p_din = p_addr;
      #10 p_addr = p_addr + 1'b1;
    end

    p_addr = 8'b0;
    p_din = 8'b1111_1111;
    repeat(256) begin
      #10 p_addr = p_addr + 1'b1;
    end
  end

  initial begin
    p_cs = 1'b0;
    #100 p_cs = 1'b1;
  end

  initial begin
    p_wr = 1'b0;
    forever begin
      #5 p_wr = ~p_wr;
    end
  end

  initial begin
    p_rd = 1'b1;
    #2000 p_rd = 1'b0;

    #2600 $stop;
  end

  initial begin
    $monitor("At time %tns, cs = %b, rd = %b, wr = %b, addr = %b, din = %b, dout = %b", $time, p_cs, p_rd,
      p_wr, p_addr, p_din, p_dout);
  end

endmodule
```

(3) 测试波形图

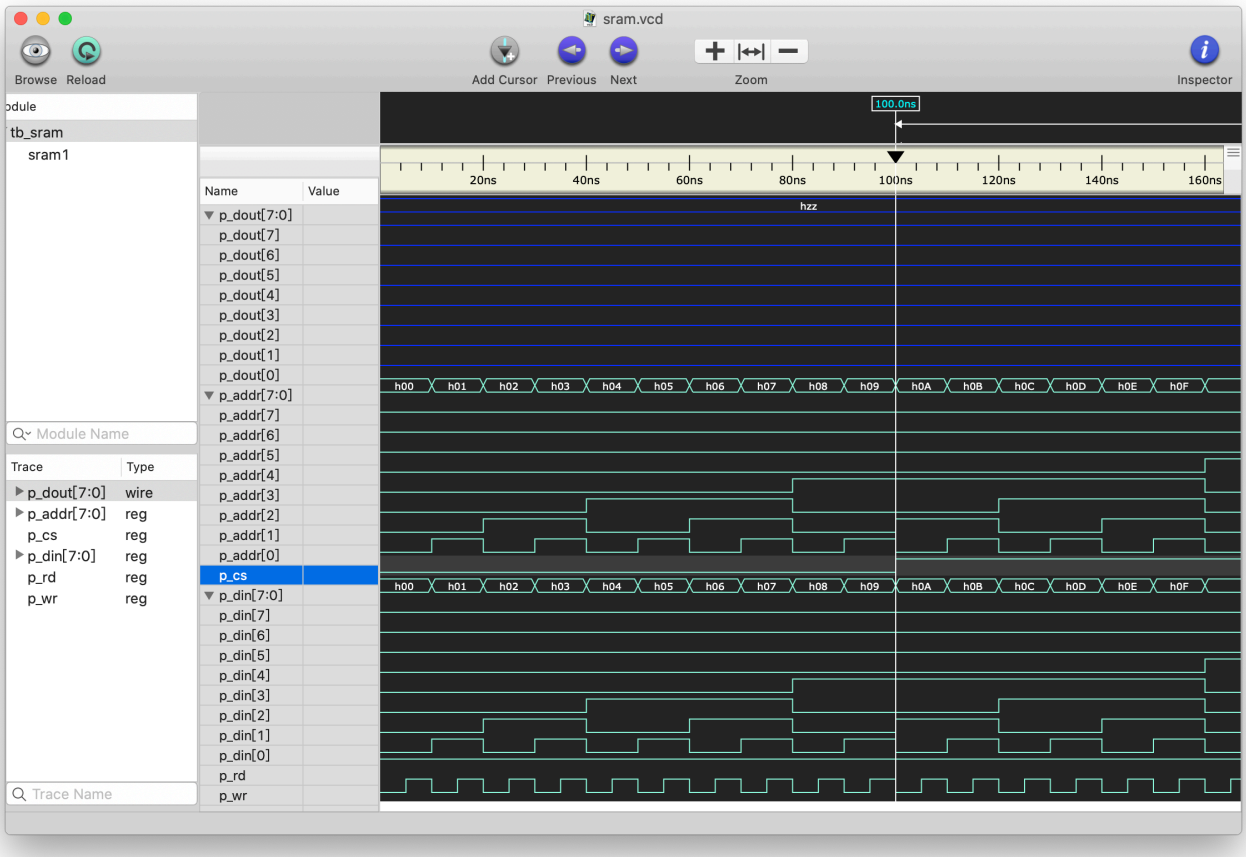

图 39: 片选从无效到有效，读无效，写周期有效

波形解读：

(i) 仿真时写信号全程周期有效，即作为脉冲输入。

(ii) 仿真时地址信号从 0 开始不断递增，在读信号有效后复位并再次递增。且输入数据在读信号无效时等于地址的数值，在读信号有效后等于 $8'b1$。

(iii) cs 无效时，即使写有效也无法写入数据，因此这段时间地址经过的 $SRAM$ 存储位置由于未被赋值，均保持初始不定态 $8'bx$，且输出高阻，如图**??**的前 $100ns$。$100ns$ 后片选有效，开始写入数据，此后的 $SRAM$ 存储位置才有数值且等于其地址数值。读无效时，输出高阻，如图**??**的 $100ns$ 之后。

(iv) 读信号有效后输出数据便一直等于地址指向的 $SRAM$ 的数据，地址变化、$SRAM$ 对应位置数值变化，输出都随之变化，除非写信号上升沿到来，此时输出 $8'bz$ 避免冲突（蓝色部分）。

(v) 如图**??**所示，读信号有效后的 $100ns$ 内，地址变化后写信号上升沿到来前输出都是 $8'bx$，这是因为最开始 $100ns$ 的片选无效时未这段地址的 $SRAM$ 位置未写入数据；在写信号上升沿到来时，为避免读写冲突，在写数据时输出 $8'bz$，写完数据后地址未改变，则输出变为 $8'b1$，即写入的数据。在读信号有效 $100ns$ 后，地址变化后写信号上升沿到来前输出等于当前地址数值，这是因为此时已经进入之前片选有效的地址范围；在写信号上升沿到来时同样输出 $8'bz$ 避免冲突，写完数据后地址未改变，输出变为 $8'b1$，即写入的数据。

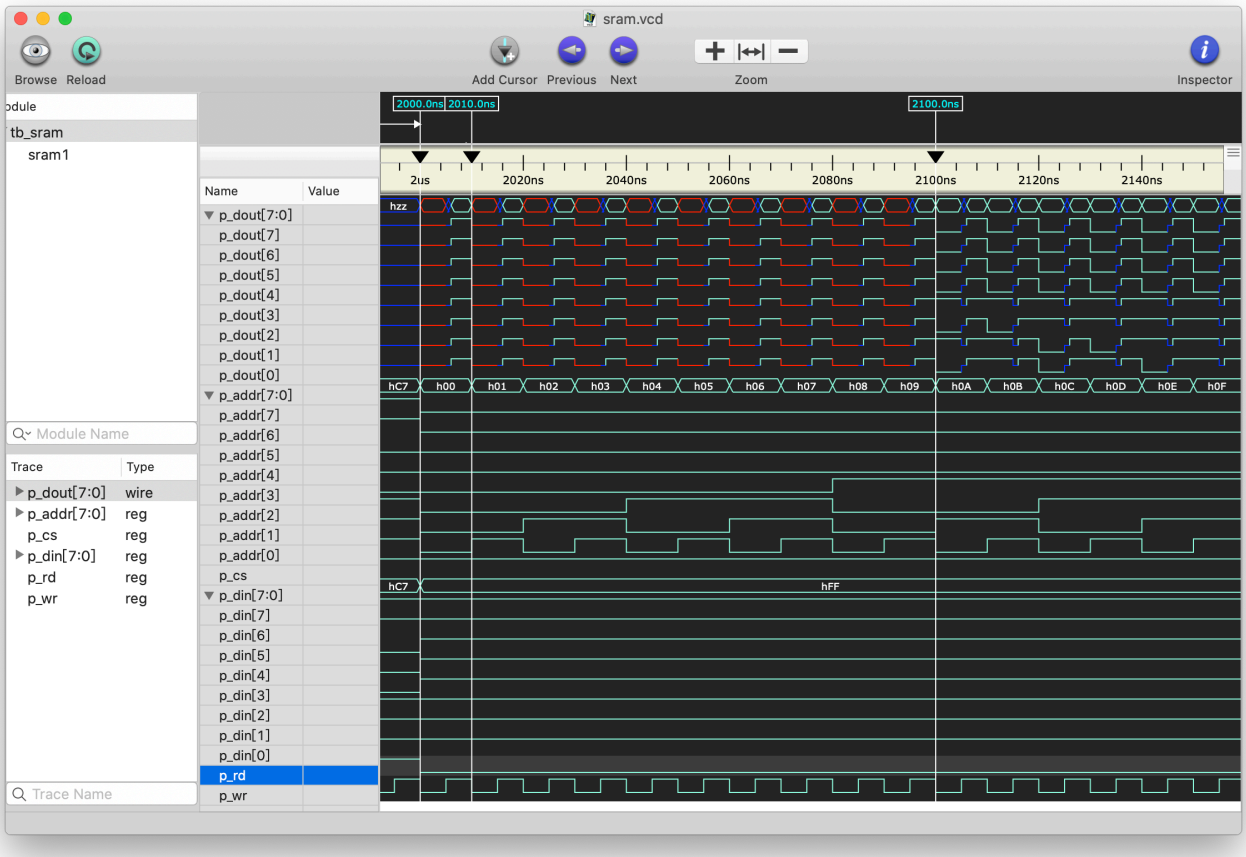

图 40: 片选有效，读有效，写周期有效

(4) 显示输出

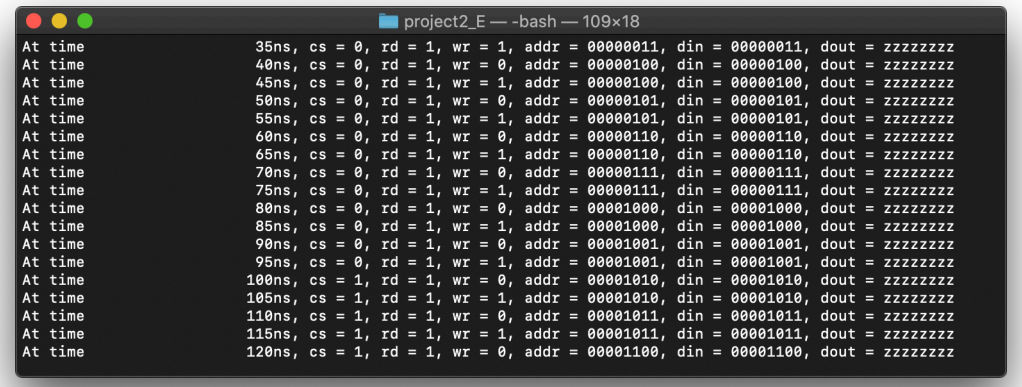

图 41: 片选无效到片选有效

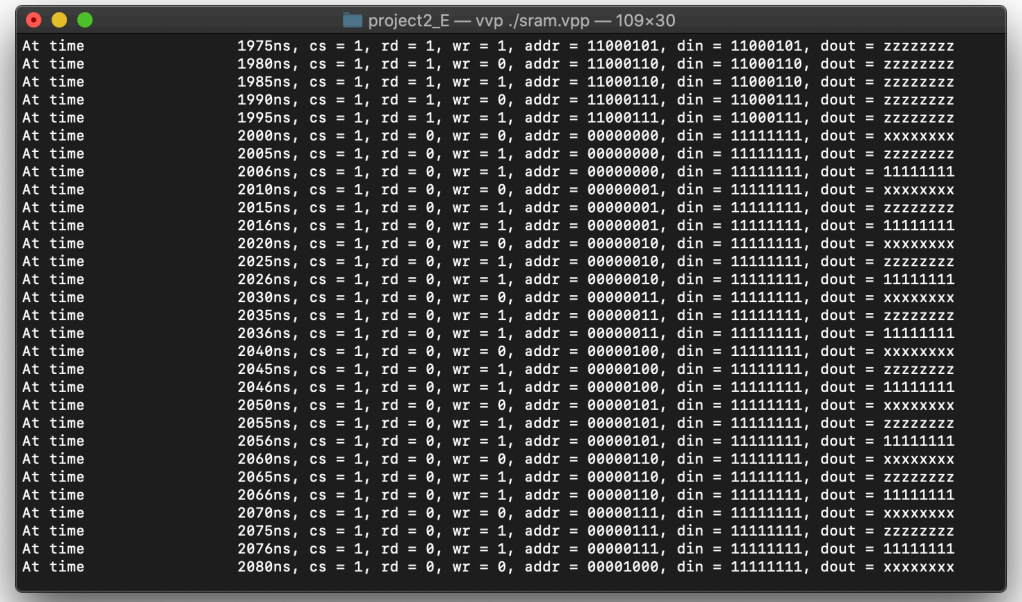

图 42: 读无效到读有效

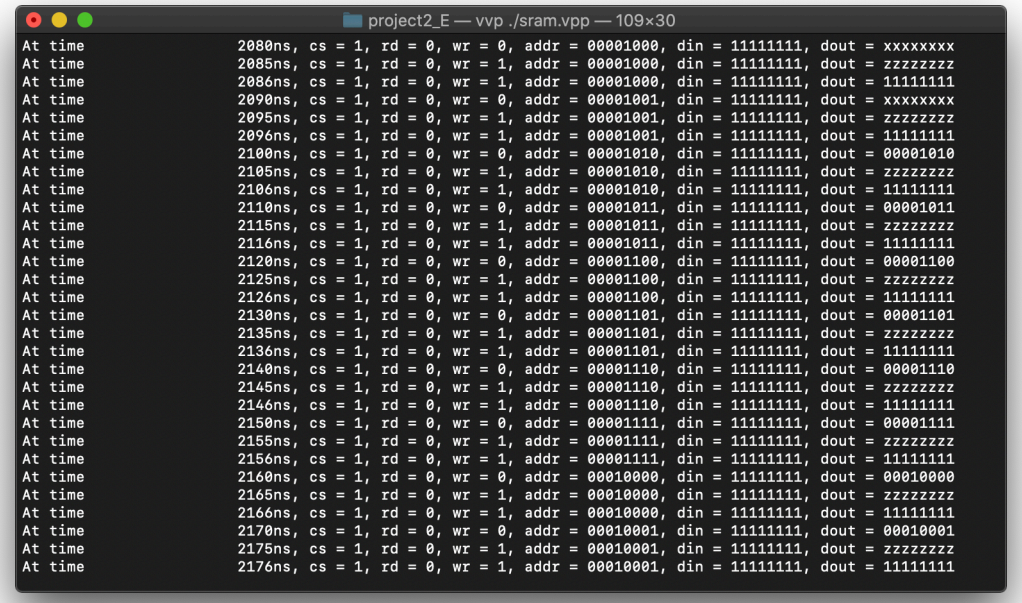

图 43: 从无数据的地址段到有数据的地址段

(5) **设计说明** 由于写信号有效是上升沿有效，而读信号有效是低电平有效，读写冲突只是在读有效的同时写信号上升沿到来的那一个瞬间发生，在这个瞬间读信号失效，写入数据并输出高

阻。而这个写入数据的瞬间难以表示，仿真时也是直接表现出数据变化而不会表现出写入数据的瞬间。所以在设计 $SRAM$ 模块时，为了表示出写入数据的过程，我自定义了该 $SRAM$ 写入数据需要 $1ns$ 时间，即写信号上升沿到来 $1ns$ 后才能完成写入数据，在这 $1ns$ 内若读信号有效则有读写冲突，读信号失效，写入数据并输出高阻。因此本设计模块理论上是无法综合的，若要综合则需要将设计模块代码第 33 行的 $\#1$ 删去，但这样就无法表示出写入数据的瞬间，也就不能表现出读写冲突。或者可以认为由于写入数据的时间忽略不计，所以没有读写冲突。

15 第 15 题

(1) 设计模块

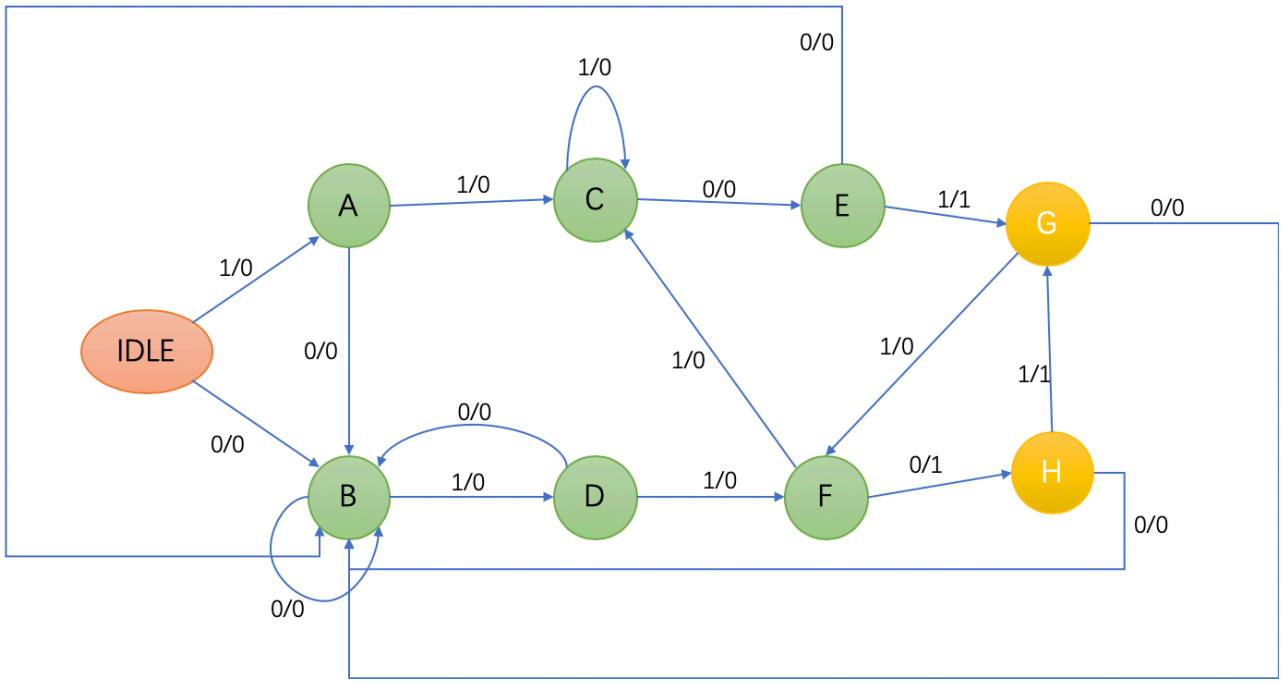

图 44: 序列检测器的状态转移图（$Mealy$ 机）

```
1  //seq_detect.v
2
3  `timescale 1 ns / 100 ps
4
5  module seq_detect (
6    output reg flag,
7    input din,
8    input clk,
9    input rst_n
10
11 );
12 parameter IDLE= 9'b0_0000_0001,
13       A = 9'b0_0000_0010,
```

```
      B   = 9'b0_0000_0100,
      C   = 9'b0_0000_1000,
      D   = 9'b0_0001_0000,
      E   = 9'b0_0010_0000,
      F   = 9'b0_0100_0000,
      G   = 9'b0_1000_0000,
      H = 9'b1_0000_0000;

reg [8:0] state;

always @(negedge clk) begin
  if(~rst_n) begin
    flag <= 1'b0;
    state <= IDLE;
  end else begin
    case (state)
      IDLE  : if(din) begin flag <= 1'b0; state <= A; end
              else begin flag <= 1'b0;  state <= B; end

      A     : if(din) begin flag <= 1'b0; state <= C; end
              else begin flag <= 1'b0;  state <= B; end

      B     : if(din) begin flag <= 1'b0; state <= D; end
              else begin flag <= 1'b0;  state <= B; end

      C     :   if(din) begin flag <= 1'b0; state <= C; end
              else begin flag <= 1'b0;  state <= E; end

      D     :   if(din) begin flag <= 1'b0; state <= F; end
              else begin flag <= 1'b0;  state <= B; end

      E     : if(din) begin flag <= 1'b1; state <= G; end
              else begin flag <= 1'b0;  state <= B; end

      F     : if(din) begin flag <= 1'b0; state <= C; end
              else begin flag <= 1'b1;  state <= H; end

      G     : if(din) begin flag <= 1'b0; state <= F; end
              else begin flag <= 1'b0;  state <= B; end

      H     :   if(din) begin flag <= 1'b1; state <= G; end
              else begin flag <= 1'b0;  state <= B; end

      default :   begin flag <= 1'b0; state <= IDLE; end
    endcase
  end
end

endmodule
```

(2) 测试模块

```
//File: tb_seq_detect.v

`timescale 1 ns / 100 ps
`include "seq_detect.v"

```

```
module tb_seq_detect;

  wire p_flag;

  reg p_clk, p_rst, p_din;

  seq_detect seqd1(
    .flag (p_flag),
    .din  (p_din),
    .clk  (p_clk),
    .rst_n(p_rst)
    );

  initial begin
    /*$dumpfile("seq_detect.vcd");
    $dumpvars(0, tb_seq_detect);*/

    p_rst = 1'b0;
    #25 p_rst = 1'b1;
  end

  initial begin
    p_clk = 1'b0;
    forever begin
      #10 p_clk = ~p_clk;
    end
  end

  parameter SIZE = 32;
  reg [SIZE - 1 : 0] data = 32'b0011_0110_1110_0001_0111_1010_0000_1101;

  initial begin : SERIES
    integer i;
    p_din = 1'b0;
    #30;
    for (i = 0; i < SIZE; i++) begin
      p_din = data[SIZE - 1];
      data = data << 1;
      #20;
    end

    $stop;
  end

  initial begin
    #10 $display("At time %tns, reset = %b, datain = %b, flag = %b.", $time, p_rst, p_din, p_flag);
    forever begin
      #20 $display("At time %tns, reset = %b, datain = %b, flag = %b.", $time, p_rst, p_din, p_flag);
    end
  end

endmodule
```

(3) 测试波形图

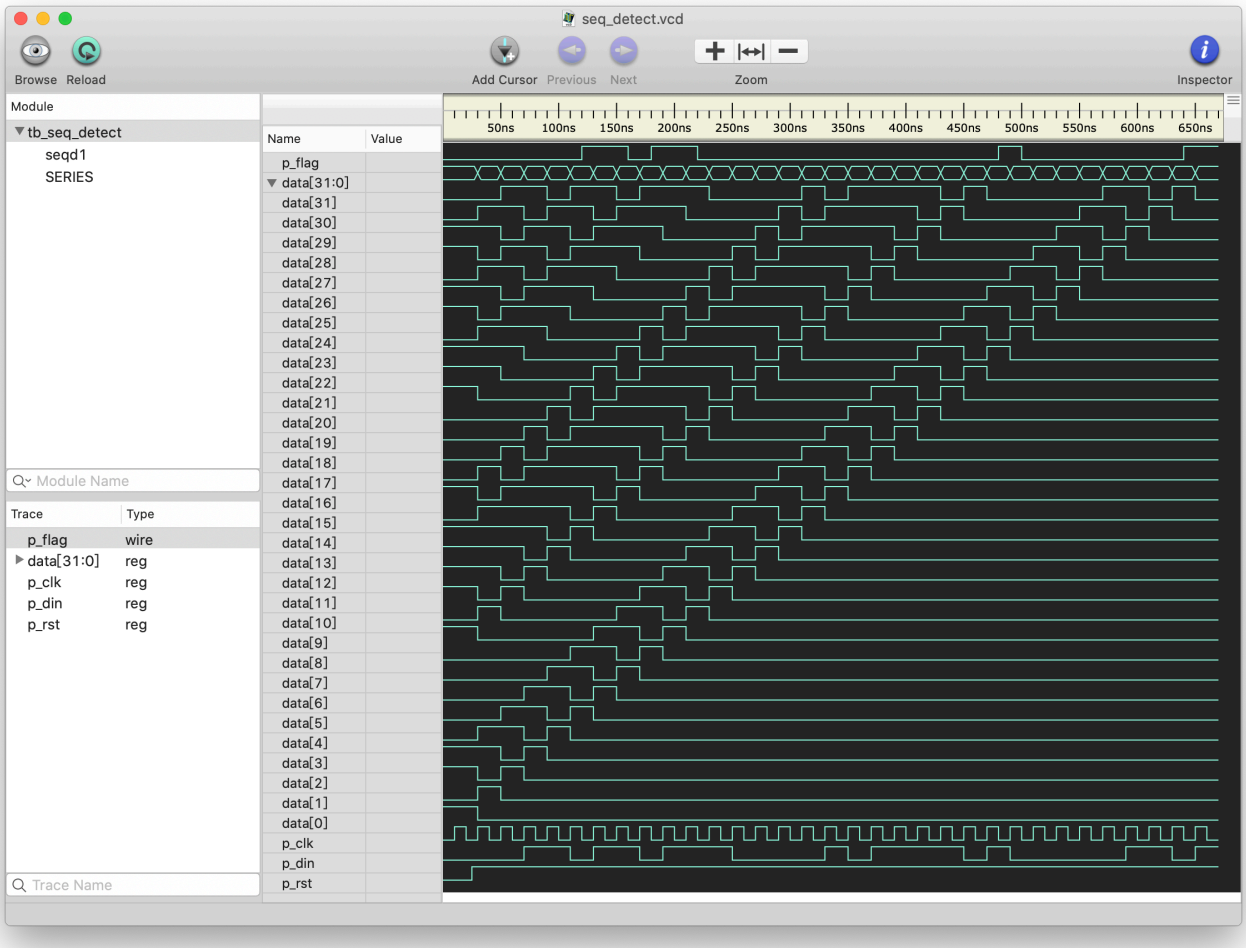

图 45: 测试波形图 *2.15*

(4) 显示输出

```
project2_F — vvp ./seq_detect.vpp — 68×18
(base) liuhaowendeMacBook-Pro:project2_F hongxing$ ./seq_detect.vpp
VCD info: dumpfile seq_detect.vcd opened for output.
At time                100ns, reset = 0, datain = 0, flag = 0.
At time                300ns, reset = 1, datain = 0, flag = 0.
At time                500ns, reset = 1, datain = 0, flag = 0.
At time                700ns, reset = 1, datain = 1, flag = 0.
At time                900ns, reset = 1, datain = 1, flag = 0.
At time               1100ns, reset = 1, datain = 0, flag = 0.
At time               1300ns, reset = 1, datain = 1, flag = 1.
At time               1500ns, reset = 1, datain = 1, flag = 1.
At time               1700ns, reset = 1, datain = 0, flag = 0.
At time               1900ns, reset = 1, datain = 1, flag = 1.
At time               2100ns, reset = 1, datain = 1, flag = 1.
At time               2300ns, reset = 1, datain = 1, flag = 0.
At time               2500ns, reset = 1, datain = 0, flag = 0.
At time               2700ns, reset = 1, datain = 0, flag = 0.
At time               2900ns, reset = 1, datain = 0, flag = 0.
At time               3100ns, reset = 1, datain = 0, flag = 0.
```

图 46: 显示输出 *2.15 (1)*

```
project2_F — vvp ./seq_detect.vpp — 68×18
At time               3100ns, reset = 1, datain = 0, flag = 0.
At time               3300ns, reset = 1, datain = 1, flag = 0.
At time               3500ns, reset = 1, datain = 0, flag = 0.
At time               3700ns, reset = 1, datain = 1, flag = 0.
At time               3900ns, reset = 1, datain = 1, flag = 0.
At time               4100ns, reset = 1, datain = 1, flag = 0.
At time               4300ns, reset = 1, datain = 1, flag = 0.
At time               4500ns, reset = 1, datain = 0, flag = 0.
At time               4700ns, reset = 1, datain = 1, flag = 0.
At time               4900ns, reset = 1, datain = 0, flag = 1.
At time               5100ns, reset = 1, datain = 0, flag = 0.
At time               5300ns, reset = 1, datain = 0, flag = 0.
At time               5500ns, reset = 1, datain = 0, flag = 0.
At time               5700ns, reset = 1, datain = 0, flag = 0.
At time               5900ns, reset = 1, datain = 1, flag = 0.
At time               6100ns, reset = 1, datain = 1, flag = 0.
At time               6300ns, reset = 1, datain = 0, flag = 0.
At time               6500ns, reset = 1, datain = 1, flag = 1.
```

图 47: 显示输出 *2.15 (2)*

16 第 16 题

(1) 设计模块

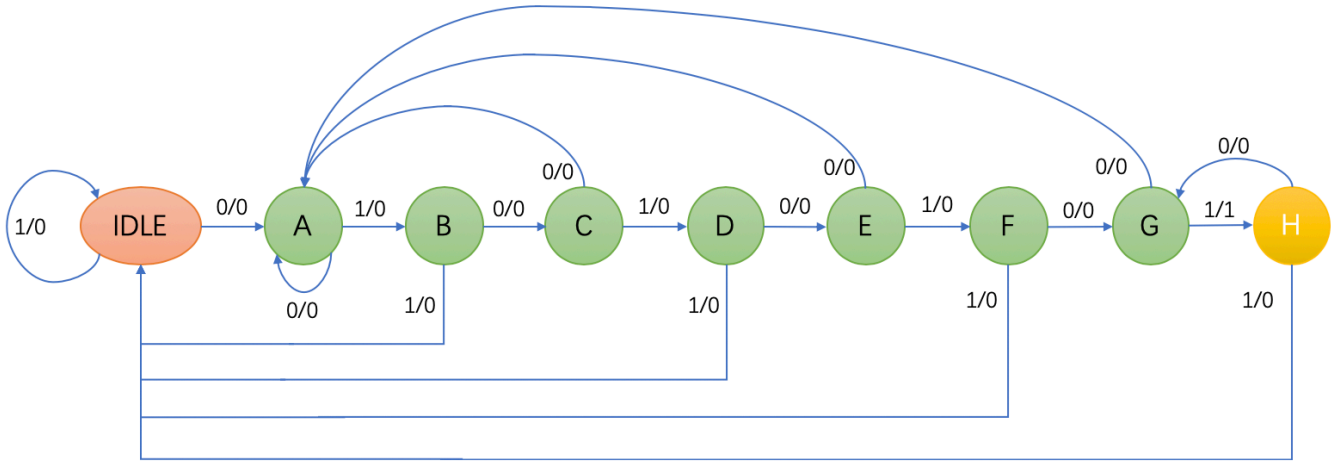

图 48: 序列检测器的状态转移图（*Mealy* 机）

```
1  //File: mealy.v
2
3  `timescale 1 ns / 100 ps
4
5  module mealy (
6    output reg flag,
7    input din,
8    input clk,
9    input rst
10
11 );
12 parameter IDLE= 9'b0_0000_0001,
13       A = 9'b0_0000_0010,
14       B   = 9'b0_0000_0100,
15       C   = 9'b0_0000_1000,
16       D   = 9'b0_0001_0000,
17       E   = 9'b0_0010_0000,
18       F   = 9'b0_0100_0000,
19       G   = 9'b0_1000_0000,
20       H = 9'b1_0000_0000;
21
22 reg [8:0] state;
23
24 always @(posedge clk or posedge rst) begin : proc_flag
25   if(rst) begin
26     flag <= 1'b0;
27     state <= IDLE;
28   end else begin
29     case (state)
30       IDLE   : if(din) begin flag <= 1'b0; state <= IDLE; end
31              else begin flag <= 1'b0;  state <= A; end
32
33       A      : if(din) begin flag <= 1'b0; state <= B; end
```

```
            else begin flag <= 1'b0;  state <= A; end

      B     : if(din) begin flag <= 1'b0; state <= IDLE; end
            else begin flag <= 1'b0;  state <= C; end

      C     :   if(din) begin flag <= 1'b0; state <= D; end
            else begin flag <= 1'b0;  state <= A; end

      D     :   if(din) begin flag <= 1'b0; state <= IDLE; end
            else begin flag <= 1'b0;  state <= E; end

      E     : if(din) begin flag <= 1'b0; state <= F; end
            else begin flag <= 1'b0;  state <= A; end

      F     : if(din) begin flag <= 1'b0; state <= IDLE; end
            else begin flag <= 1'b0;  state <= G; end

      G     : if(din) begin flag <= 1'b1; state <= H; end
            else begin flag <= 1'b0;  state <= A; end

      H     :   if(din) begin flag <= 1'b0; state <= IDLE; end
            else begin flag <= 1'b0;  state <= G; end

       default :   begin flag <= 1'b0; state <= IDLE; end
    endcase
  end
end

endmodule
```

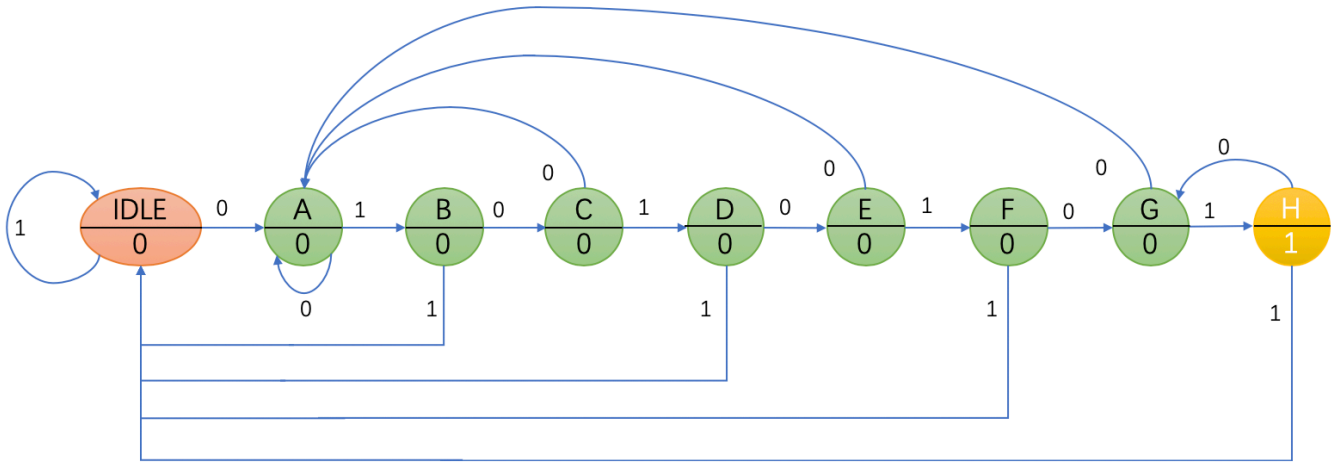

图 49: 序列检测器的状态转移图（*Moore* 机）

```
//File: moore.v

`timescale 1 ns / 100 ps

module moore (
  output reg flag,
  input din,
```

```
  input clk,
  input rst

);
parameter IDLE= 9'b0_0000_0001,
      A = 9'b0_0000_0010,
      B   = 9'b0_0000_0100,
      C   = 9'b0_0000_1000,
      D   = 9'b0_0001_0000,
      E   = 9'b0_0010_0000,
      F   = 9'b0_0100_0000,
      G   = 9'b0_1000_0000,
      H = 9'b1_0000_0000;

reg [8:0] state;

always @(posedge clk or posedge rst) begin : proc_flag
  if(rst) begin
    flag <= 1'b0;
    state <= IDLE;
  end else begin
    flag <= (state == H) ? 1'b1 : 1'b0;
    case (state)
      IDLE  : state <= (din) ? IDLE : A;
      A     : state <= (din) ? B : A;
      B     : state <= (din) ? IDLE : C;
      C     : state <= (din) ? D : A;
      D     : state <= (din) ? IDLE : E;
      E     : state <= (din) ? F : A;
      F     :   state <= (din) ? IDLE : G;
      G     : state <= (din) ? H : A;
      H   : state <= (din) ? IDLE : G;

      default :   state <= IDLE;
    endcase
  end
end

endmodule
```

(2) 测试模块

```
//File: top.v

`timescale 1 ns / 100 ps
`include "mealy.v"
`include "moore.v"

module top;

  wire p_mlflag;
  wire p_mrflag;

  reg p_clk, p_rst, p_din;

  mealy meal(
    .flag (p_mlflag),
```

```
16      .din   (p_din),
17      .clk   (p_clk),
18      .rst(p_rst)
19      );
20
21    moore moor(
22      .flag (p_mrflag),
23      .din   (p_din),
24      .clk   (p_clk),
25      .rst(p_rst)
26      );
27
28    initial begin
29      /*$dumpfile("mealy_moore.vcd");
30      $dumpvars(0, top);*/
31
32      p_rst = 1'b1;
33      #25 p_rst = 1'b0;
34
35      #250.5 p_rst = 1'b1;
36      #5 p_rst = 1'b0;
37    end
38
39    initial begin
40      p_clk = 1'b0;
41      forever begin
42        #10 p_clk = ~p_clk;
43      end
44    end
45
46    parameter SIZE = 64;
47    reg [SIZE - 1 : 0] data = 64'
        b0011_1011_0101_0101_0110_1010_1000_1101_1001_0101_0101_1010_1010_1010_1010_1010;
48
49    initial begin : SERIES
50      integer i;
51      p_din = 1'b0;
52      #30;
53      for (i = 0; i < SIZE; i++) begin
54        p_din = data[0];
55        data = data >> 1;
56        #20;
57      end
58
59      $stop;
60    end
61
62    initial begin
63      $display("At time %tns, reset = %b, datain = %b, mealy_flag = %b, moore_flag = %b.", $time, p_rst,
        p_din, p_mlflag, p_mrflag);
64      forever begin
65        #20 $display("At time %tns, reset = %b, datain = %b, mealy_flag = %b, moore_flag = %b.", $time,
        p_rst, p_din, p_mlflag, p_mrflag);
66      end
67    end
68
69  endmodule
```

(3) 测试波形图

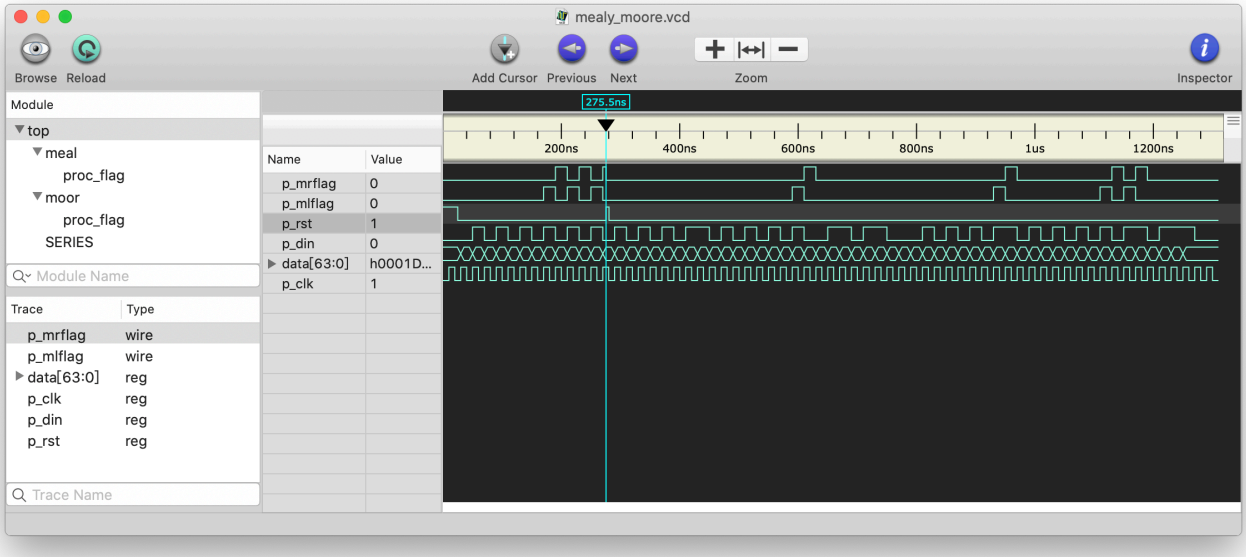

图 50: 测试波形图 *2.16*

(4) 显示输出

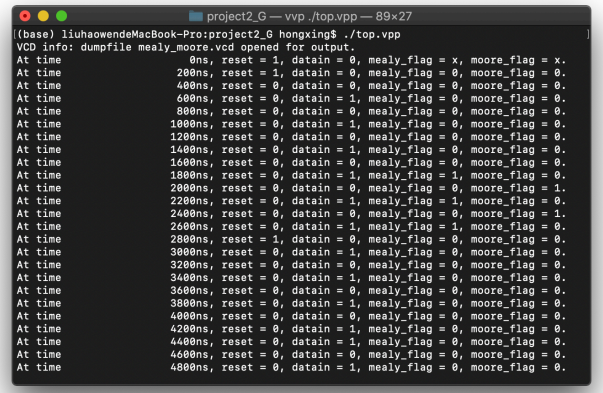

图 51: 显示输出 *2.16*（*1*）

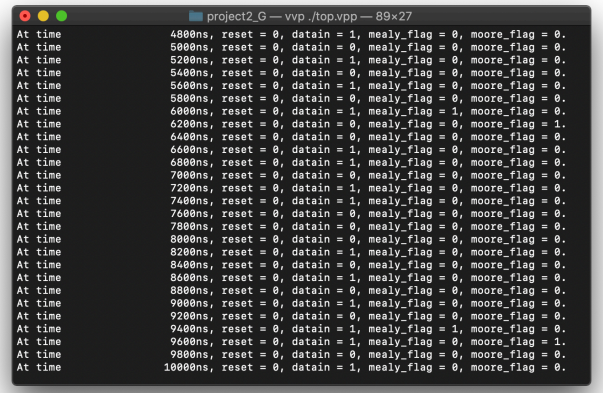

图 52: 显示输出 *2.16*（*2*）

17 附录

由于电脑是 *macOS* 系统，电脑资源不足以安装 *Windows* 虚拟机来使用 *Modelsim* 完成大作业，因此直接采取了终端（命令行）的 *iverilog* 仿真加 *Scansion* 观察波形的方式完成大作业。另外，本报告中的代码在 *Modelsim* 中应该均能跑通。

以下列出本大作业基本调试环境：

- **机器：** *MacBook Pro* (13 – *inch*, 2016, *Four Thunderbolt* 3 *Ports*)

- **系统：** *macOS Catalina Version*10.15.2

图 53: 基于的机器与系统

- **仿真软件：** *Icarus Verilog version* 10.3 (*stable*) (*v*10_3)

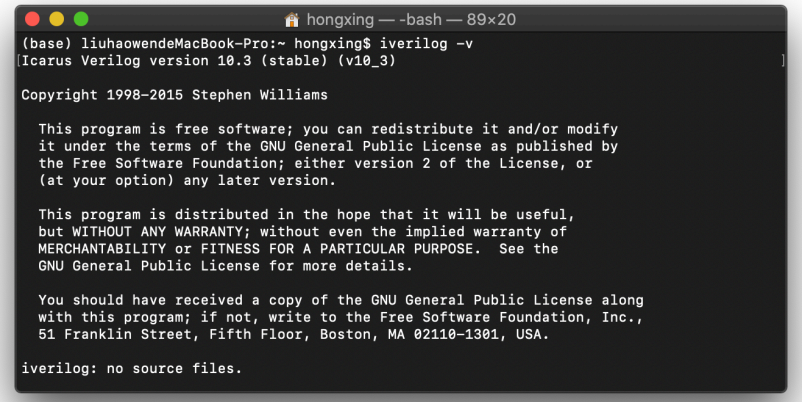

图 54: 开源 *RTL* 仿真器 *Icarus Verilog*

- **波形观察器：** *Scansion Version* 1.12(1.12)

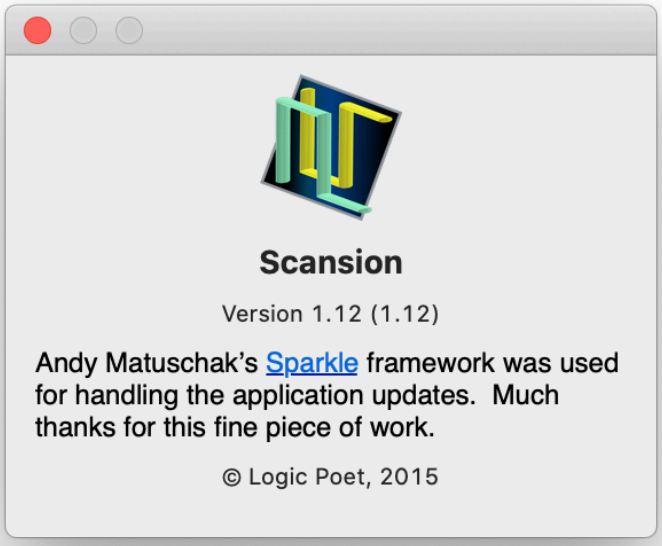

图 55: *vcd* 查看器 *Scansion*