

# 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

```
1 //File: wavegen.v
2 //517021911065 刘浩文
3
4 `timescale 10 ns / 1 ns
5
6 module wavegen;
7
8 reg wave;
9
10 /*initial
11 begin
12     $dumpfile("wavegen.vcd");
13     $dumpvars(0, wavegen);
14 end*/
15
16 initial
17 fork
18     wave = 1'b0;
19 #2 wave = ~wave;
20 #3 wave = ~wave;
21 #12 wave = ~wave;
22 #22 wave = ~wave;
23 #24 wave = ~wave;
24 #27 wave = ~wave;
25 #32 wave = ~wave;
26 #35 $stop;
27 join
28
29 endmodule
```

### (2) 测试模块

无

### (3) 测试波形图

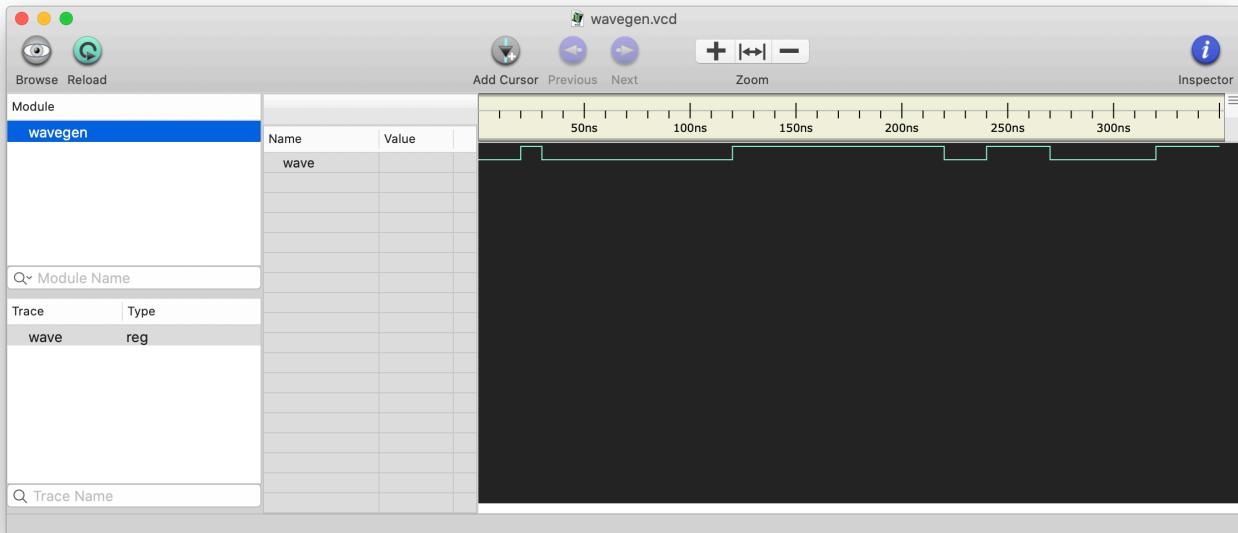


图 1: 测试波形图 2.1

## 2 第 2 题

### (1) 设计模块

```

1 //File: Encoder8x3.v
2 //517021911065 刘浩文
3
4 `timescale 1 ns / 1 ns
5
6 module Encoder8x3 (
7   output reg [2:0] code,
8   input [7:0] data
9
10 );
11 always @(*) begin
12
13   case (data)
14     8'b0000_0001: code = 3'd0;
15     8'b0000_0010: code = 3'd1;
16     8'b0000_0100: code = 3'd2;
17     8'b0000_1000: code = 3'd3;
18     8'b0001_0000: code = 3'd4;
19     8'b0010_0000: code = 3'd5;
20     8'b0100_0000: code = 3'd6;
21     8'b1000_0000: code = 3'd7;
22
23   default : code = 3'dx;
24 endcase
25
26 end
27
28 endmodule

```

## (2) 测试模块

```
1 //File: tb_Encoder8x3.v
2
3 `timescale 1 ns / 1 ns
4 `include "Encoder8x3.v"
5
6 module tb_Encoder8x3;
7
8   wire [2:0] p_code;
9   reg [7:0] p_data;
10
11 Encoder8x3 m_enc8x3(.code(p_code),
12 .data(p_data));
13
14 initial
15 begin
16   /*$dumpfile("Encoder8x3.vcd");
17   $dumpvars(0, tb_Encoder8x3);*/
18   p_data = 8'b1;
19
20   repeat(7) #10 p_data = p_data << 1;
21
22   #10 $stop;
23 end
24
25 initial
26 begin
27   $monitor("data = %b, code = %d.", p_data, p_code);
28 end
29
30 endmodule
```

## (3) 测试波形图

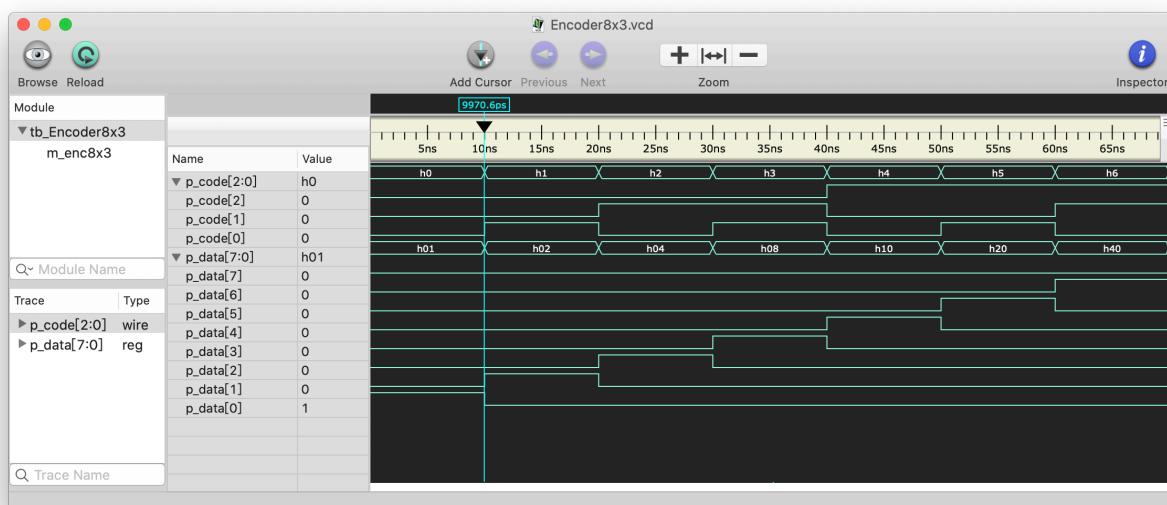
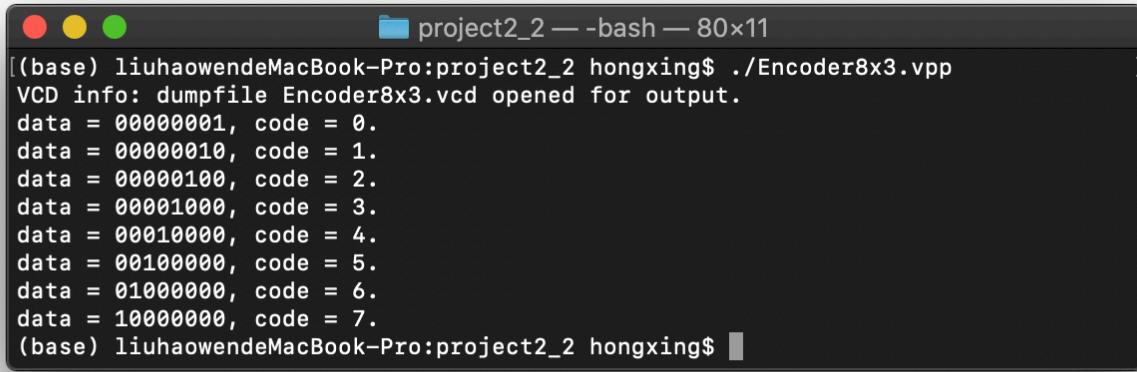


图 2: 测试波形图 2.2

#### (4) 显示输出



A screenshot of a terminal window titled "project2\_2 — bash — 80x11". The window shows the command "liuhaowendeMacBook-Pro:project2\_2 hongxing\$ ./Encoder8x3.vpp" followed by the output of a Verilog simulation. The output consists of eight lines, each containing a binary value for "data" and a corresponding "code" value from 0 to 7. The window has a dark background with light-colored text and includes standard OS X window controls at the top.

```
[base] liuhaowendeMacBook-Pro:project2_2 hongxing$ ./Encoder8x3.vpp
VCD info: dumpfile Encoder8x3.vcd opened for output.
data = 00000001, code = 0.
data = 00000010, code = 1.
data = 00000100, code = 2.
data = 00001000, code = 3.
data = 00010000, code = 4.
data = 00100000, code = 5.
data = 01000000, code = 6.
data = 10000000, code = 7.
(base) liuhaowendeMacBook-Pro:project2_2 hongxing$
```

图 3: 显示输出 2.2

## 3 第 3 题

### (1) 设计模块

```
1 //File: mux2x1.v
2 //517021911065 刘浩文
3
4 `timescale 1 ns / 1 ns
5
6 module mux2x1 (
7   output dout,
8   input sel,
9   input [1:0] din
10
11 );
12 bufif0 b0(dout, din[0], sel);
13 bufif1 b1(dout, din[1], sel);
14
15 endmodule
```

```
1 //File: mux4x1.v
2
3 `timescale 1 ns / 1 ns
4 `include "mux2x1.v"
5
6 module mux4x1 (
7   output dout,
8   input [1:0] sel,
9   input [3:0] din
10
11 );
12 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

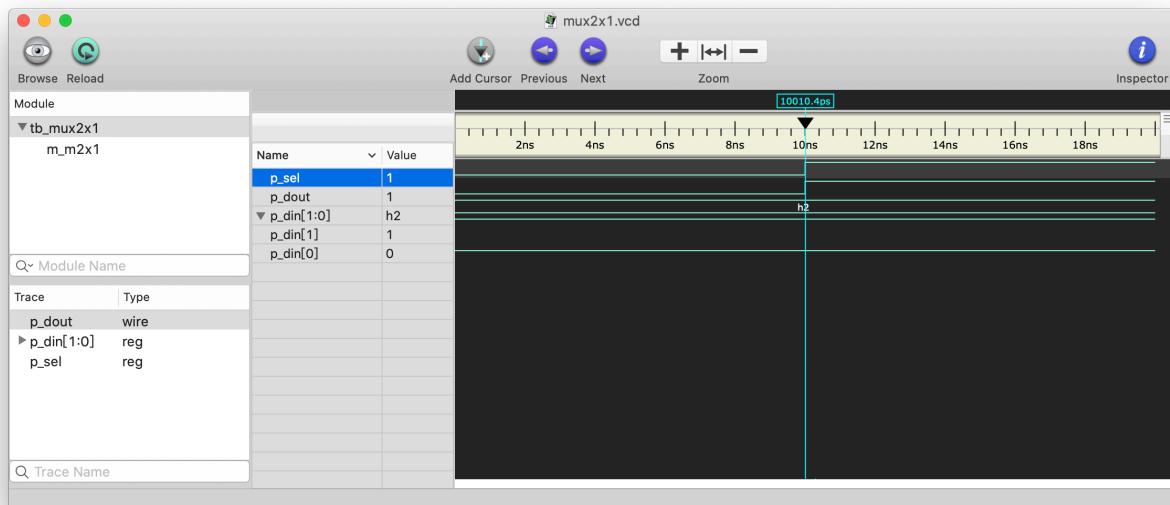


图 4: 测试波形图 *mux2x1*

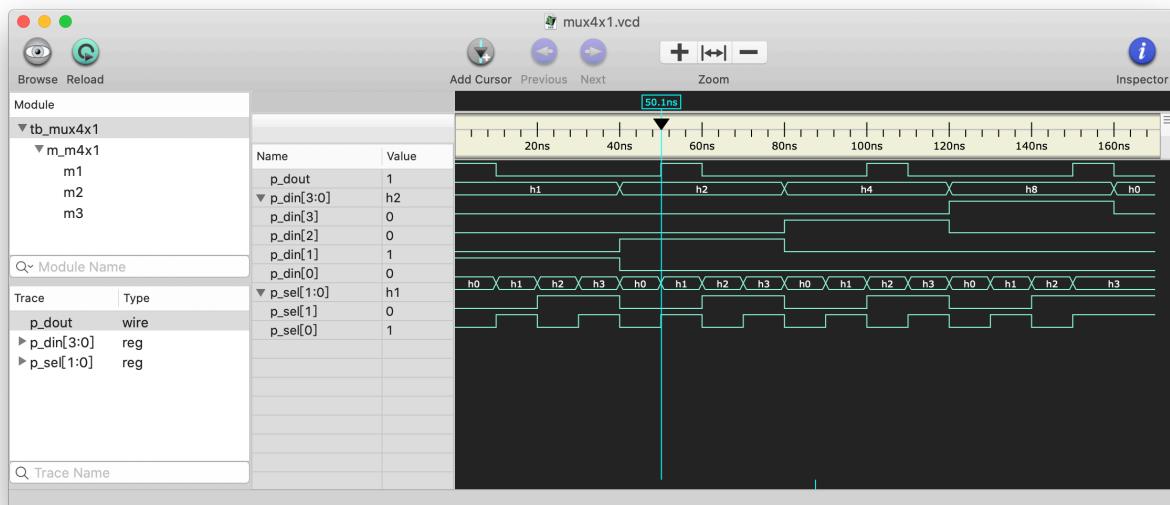


图 5: 测试波形图 *mux4x1*

#### (4) 显示输出

```
project2_3 — vvp ./mux2x1.vpp — 80x8
[(base) liuhawendeMacBook-Pro:project2_3 hongxing$ ./mux2x1.vpp
VCD info: dumpfile mux2x1.vcd opened for output.
in0 = 0, in1 = 1, s = 0, out = 0.
in0 = 0, in1 = 1, s = 1, out = 1.
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 20 ticks.
> ]
```

图 6: 显示输出 *mux2x1*

```

(base) liuhawendeMacBook-Pro:project2_3 hongxing$ ./mux4x1.vpp
VCD info: dumpfile mux4x1.vcd opened for output.
in = 0001, s = 00, out = 1.
in = 0001, s = 01, out = 0.
in = 0001, s = 10, out = 0.
in = 0001, s = 11, out = 0.
in = 0010, s = 00, out = 0.
in = 0010, s = 01, out = 1.
in = 0010, s = 10, out = 0.
in = 0010, s = 11, out = 0.
in = 0100, s = 00, out = 0.
in = 0100, s = 01, out = 0.
in = 0100, s = 10, out = 1.
in = 0100, s = 11, out = 0.
in = 1000, s = 00, out = 0.
in = 1000, s = 01, out = 0.
in = 1000, s = 10, out = 0.
in = 1000, s = 11, out = 1.
in = 0000, s = 11, out = 0.
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 170 ticks.
> 

```

图 7: 显示输出 *mux4x1*

## 4 第 4 题

### (1) 设计模块

```

1 //File: comb_str.v
2
3 `timescale 1 ns / 1 ns
4
5 module comb_str (
6   output Y,
7   input A, B, C, D
8 );
9 wire u1, u2, u3, u4;
10
11 not n1(u1, D);
12 and a1(u4, B, C, u1);
13 or o1(u3, A, D);
14 not n2(u2, u3);
15 and a2(Y, u2, u4);
16
17 endmodule

```

```

1 //File: comb_dataflow.v
2
3 `timescale 1 ns / 1 ns
4
5 module comb_dataflow (
6   output Y,
7   input A, B, C, D
8
9 );
10 wire u1, u2, u3, u4;
11
12 assign u1 = ~D, u4 = B & C & u1,
13     u3 = A | D, u2 = ~u3,
14     Y = u2 & u4;
15
16 endmodule

```

```

1 //File: comb_behavior.v
2
3 `timescale 1 ns / 1 ns
4
5 module comb_behavior (
6   output reg Y,
7   input A, B, C, D
8
9 );
10 reg u1, u2, u3, u4;
11
12 always @(*) begin
13
14   u1 = ~D;
15   u4 = B & C & u1;
16   u3 = A | D;
17   u2 = ~u3;
18   Y = u2 & u4;
19
20 end
21
22 endmodule

```

```

1 //File: comb_prim.v
2
3 `timescale 1 ns / 1 ns
4
5 primitive prim_udp(output Y,
6   input A,
7   input B,
8   input C,
9   input D
10 );
11
12 table
13   // A B C D : Y
14   0 0 0 0 : 0;
15   0 0 0 1 : 0;
16   0 0 1 0 : 0;
17   0 0 1 1 : 0;

```

```

18      0 1 0 0 : 0;
19      0 1 0 1 : 0;
20      0 1 1 0 : 1;
21      0 1 1 1 : 0;
22      1 0 0 0 : 0;
23      1 0 0 1 : 0;
24      1 0 1 0 : 0;
25      1 0 1 1 : 0;
26      1 1 0 0 : 0;
27      1 1 0 1 : 0;
28      1 1 1 0 : 0;
29      1 1 1 1 : 0;
30  endtable
31
32 endprimitive
33
34
35 module comb_prim (
36   output Y,
37   input A, B, C, D
38 );
39 prim_udp(Y, A, B, C, D);
40
41 endmodule

```

## (2) 测试模块

```

1 //File: testbench_comb.v
2
3 `timescale 1 ns / 1 ns
4 `include "comb_str.v"
5 `include "comb_dataflow.v"
6 `include "comb_behavior.v"
7 `include "comb_prim.v"
8
9 module testbench_comb;
10
11   wire [3:0] p_Y;
12
13   reg p_A, p_B, p_C, p_D;
14
15   reg [3:0] temp;
16
17   comb_str c1 (.Y(p_Y[0]) ,
18     .A(p_A), .B(p_B), .C(p_C), .D(p_D));
19
20   comb_dataflow c2 (.Y(p_Y[1]) ,
21     .A(p_A), .B(p_B), .C(p_C), .D(p_D));
22
23   comb_behavior c3 (.Y(p_Y[2]) ,
24     .A(p_A), .B(p_B), .C(p_C), .D(p_D));
25
26   comb_prim c4 (.Y(p_Y[3]) ,
27     .A(p_A), .B(p_B), .C(p_C), .D(p_D));
28
29   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.", 
49 $time, p_A, p_B, p_C, p_D, p_Y[0], p_Y[1], p_Y[2], p_Y[3]);
50 end
51 endmodule

```

### (3) 测试波形图

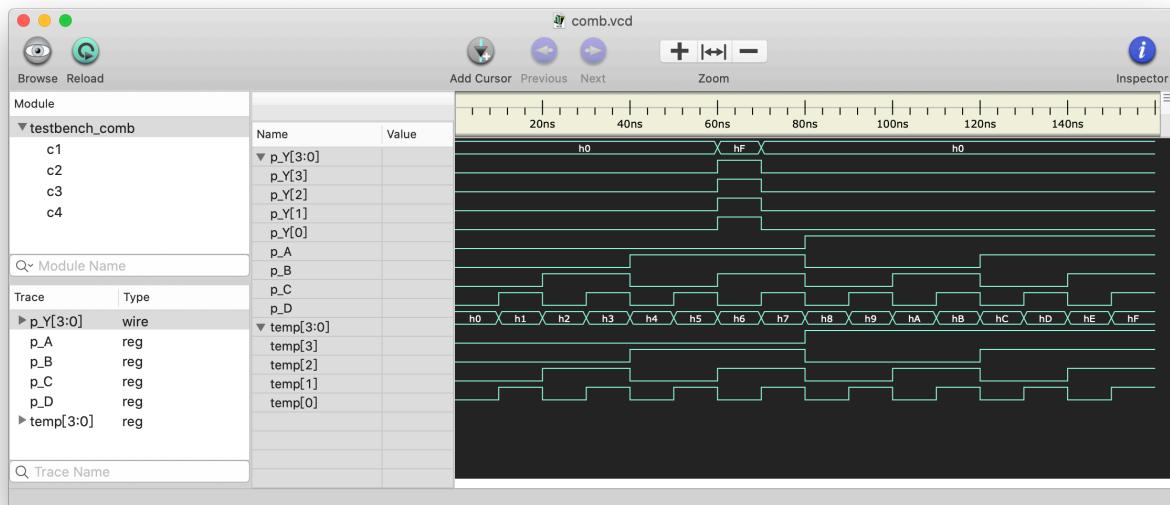


图 8: 测试波形图 2.4

### (4) 显示输出

```

project2_4 — vvp ./comb.vpp — 98x22
[(base) liuhawendeMacBook-Pro:project2_4 hongxing$ ./comb.vpp
VCD info: dumpfile comb.vcd opened for output.
At time          0, A = 0, B = 0, C = 0, D = 0, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
At time          10, A = 0, B = 0, C = 0, D = 1, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
[At time         20, A = 0, B = 0, C = 1, D = 0, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.]
[At time         30, A = 0, B = 0, C = 1, D = 1, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.]
At time          40, A = 0, B = 1, C = 0, D = 0, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
At time          50, A = 0, B = 1, C = 0, D = 1, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
At time          60, A = 0, B = 1, C = 1, D = 0, Y_str = 1, Y_df = 1, Y_bh = 1, Y_pr = 1.
At time          70, A = 0, B = 1, C = 1, D = 1, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
At time          80, A = 1, B = 0, C = 0, D = 0, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
At time          90, A = 1, B = 0, C = 0, D = 1, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
At time         100, A = 1, B = 0, C = 1, D = 0, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
At time         110, A = 1, B = 0, C = 1, D = 1, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
At time         120, A = 1, B = 1, C = 0, D = 0, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
At time         130, A = 1, B = 1, C = 0, D = 1, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
At time         140, A = 1, B = 1, C = 1, D = 0, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
At time         150, A = 1, B = 1, C = 1, D = 1, Y_str = 0, Y_df = 0, Y_bh = 0, Y_pr = 0.
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 160 ticks.
> 

```

图 9: 显示输出 2.4

## 5 第 5 题

### (1) 设计模块

```

1 //File: comb_Y.v
2
3 `timescale 1 ns / 1 ns
4
5 module comb_Y1 (
6   output Y,
7   input A, B, C
8 );
9 assign Y = ~A & ~B & C
10  | ~A & B & ~C
11  | A & ~B & ~C
12  | A & ~B & C;
13
14 endmodule
15
16 module comb_Y2 (
17   output Y,
18   input A, B, C, D
19 );
20 assign Y = ~A & B & ~C & ~D
21  | ~A & B & ~C & D
22  | ~A & B & C & ~D
23  | ~A & B & C & D
24  | A & ~B & C & D
25  | A & B & ~C & ~D

```

```

26      | A & B & ~C & D;
27
28 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

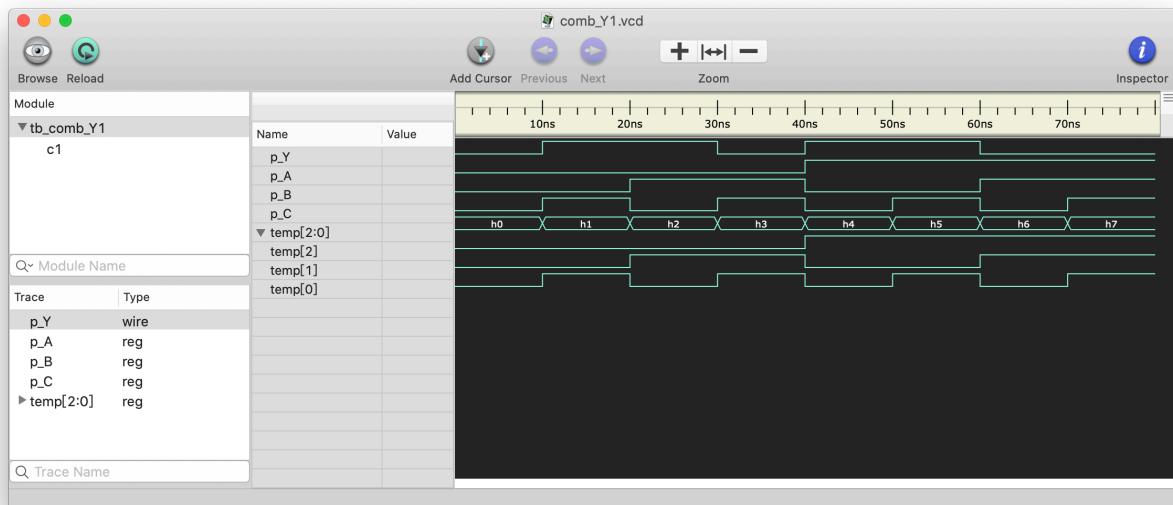


图 10: 测试波形图 Y1

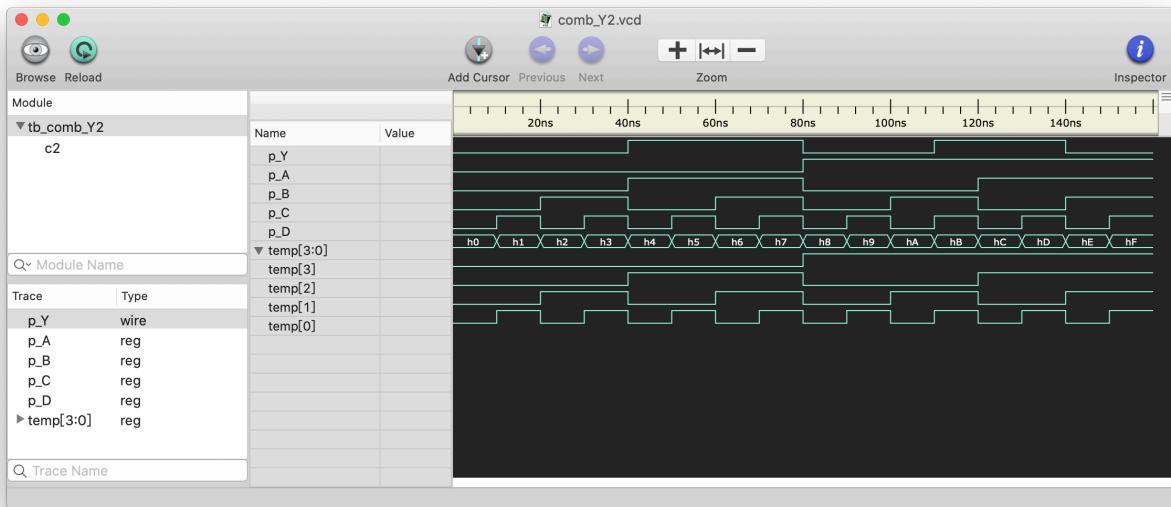


图 11: 测试波形图 Y2

#### (4) 显示输出

```
project2_5 — vvp ./comb_Y1.vpp — 86x14
(base) liuhawendeMacBook-Pro:project2_5 hongxing$ ./comb_Y1.vpp
VCD info: dumpfile comb_Y1.vcd opened for output.
At time          0, A = 0, B = 0, C = 0, Y = 0.
At time         10, A = 0, B = 0, C = 1, Y = 1.
At time         20, A = 0, B = 1, C = 0, Y = 1.
At time         30, A = 0, B = 1, C = 1, Y = 0.
At time         40, A = 1, B = 0, C = 0, Y = 1.
At time         50, A = 1, B = 0, C = 1, Y = 1.
At time         60, A = 1, B = 1, C = 0, Y = 0.
At time         70, A = 1, B = 1, C = 1, Y = 0.
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 80 ticks.
> █
```

图 12: 显示输出 Y1

```

(base) liuhawendeMacBook-Pro:project2_5 hongxing$ ./comb_Y2.vpp
VCD info: dumpfile comb_Y2.vcd opened for output.
At time          0, A = 0, B = 0, C = 0, D = 0, Y = 0.
At time         10, A = 0, B = 0, C = 0, D = 1, Y = 0.
At time         20, A = 0, B = 0, C = 1, D = 0, Y = 0.
At time         30, A = 0, B = 0, C = 1, D = 1, Y = 0.
At time         40, A = 0, B = 1, C = 0, D = 0, Y = 1.
At time         50, A = 0, B = 1, C = 0, D = 1, Y = 1.
At time         60, A = 0, B = 1, C = 1, D = 0, Y = 1.
At time         70, A = 0, B = 1, C = 1, D = 1, Y = 1.
At time         80, A = 1, B = 0, C = 0, D = 0, Y = 0.
At time         90, A = 1, B = 0, C = 0, D = 1, Y = 0.
At time        100, A = 1, B = 0, C = 1, D = 0, Y = 0.
At time        110, A = 1, B = 0, C = 1, D = 1, Y = 1.
At time        120, A = 1, B = 1, C = 0, D = 0, Y = 1.
At time        130, A = 1, B = 1, C = 0, D = 1, Y = 1.
At time        140, A = 1, B = 1, C = 1, D = 0, Y = 0.
At time        150, A = 1, B = 1, C = 1, D = 1, Y = 0.

** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 160 ticks.
> 

```

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

```

```

4  `include "ones_count.v"
5
6 module tb_ones_count;
7
8   wire [3:0] p_cnt;
9
10  reg [7:0] p_din;
11
12 ones_count one1(
13   .count(p_cnt),
14   .dat_in(p_din)
15 );
16
17 initial
18 begin
19   /*$dumpfile("ones_count.vcd");
20   $dumpvars(0, tb_ones_count);*/
21   p_din = 8'b0;
22   repeat(255)
23   begin
24     #10 p_din = p_din + 1;
25   end
26
27   #10 $stop;
28 end
29
30 initial
31 begin
32   $monitor("At time %tns, dat_in = %b, count = %d", $time, p_din, p_cnt);
33 end
34
35 endmodule

```

### (3) 测试波形图

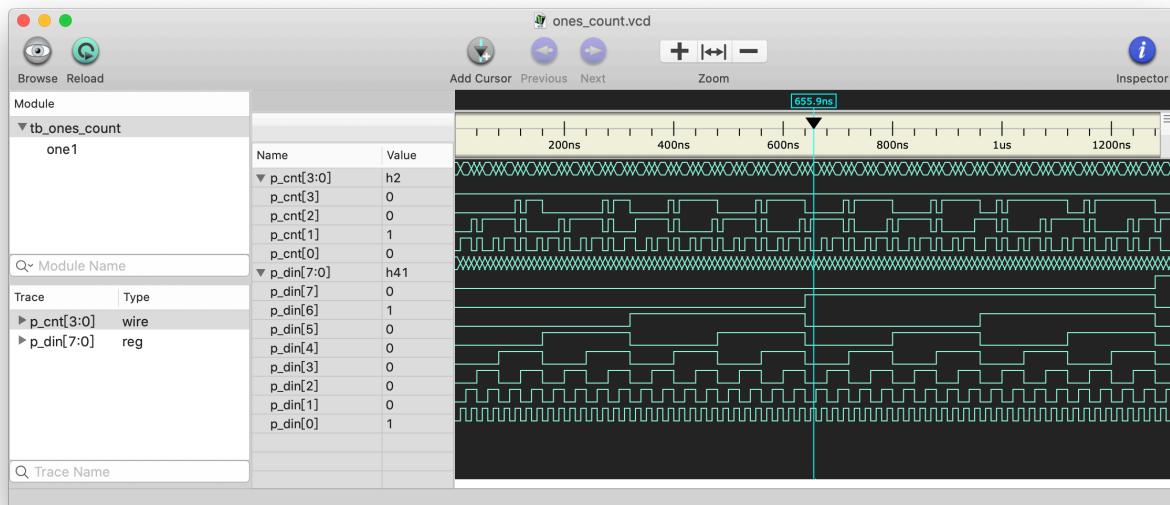


图 14: 测试波形图 2.6(1)

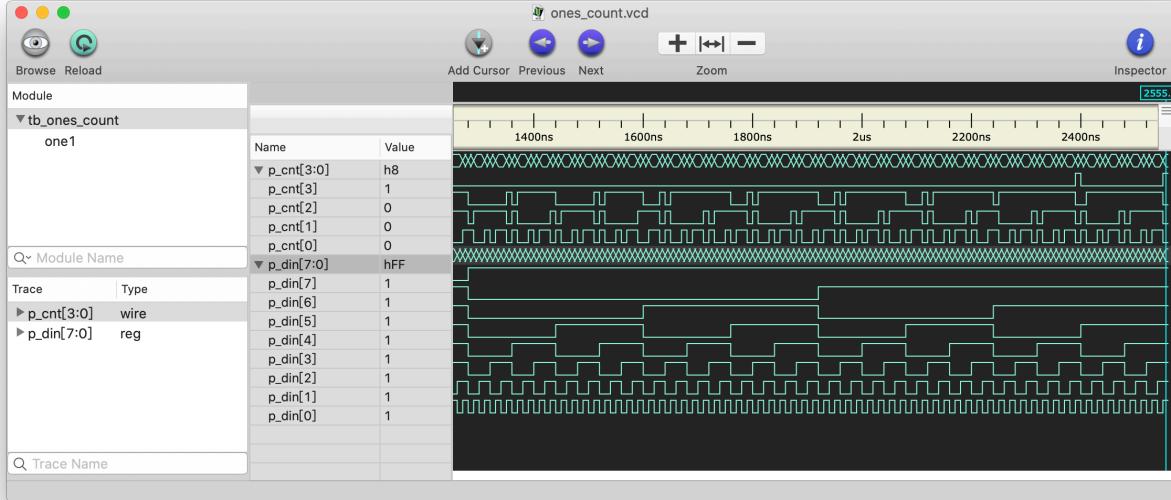


图 15: 测试波形图 2.6(2)

#### (4) 显示输出

```
project2_6 — vvp ./ones_count.vpp — 63x33
(base) liuhawendeMacBook-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 = 00000100, count =  2
At time        90ns, dat_in = 00000100, count =  3
At time       100ns, dat_in = 00000100, count =  3
At time       110ns, dat_in = 00000101, count =  4
At time       120ns, dat_in = 00000100, count =  3
At time       130ns, dat_in = 00000101, count =  4
At time       140ns, dat_in = 00000110, count =  4
At time       150ns, dat_in = 00000111, 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 — 63x33
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

```
1 //File: dec_counter.v
2
3 'timescale 10 ns / 1 ns
4
5 module dec_counter (
6   output reg [3:0] counter,
7   input clk,      // Clock
8   input reset // Asynchronous reset active low
9 );
10 );
11 always @(posedge clk)
12 begin
13   if (~reset)
14     counter <= 4'd0;
15   else if (counter == 4'd10)
16     counter <= 4'd0;
17   else counter = counter + 4'd1;
18
19 end
20
21
22 endmodule
```

### (2) 测试模块

```
1 //File: tb_dec_counter.v
2
3 'timescale 10 ns / 1 ns
4 `include "dec_counter.v"
5
6 module tb_dec_counter;
7
8   wire [3:0] p_counter;
9
10  reg p_clk, p_reset;
11
12  dec_counter m_decc(
13    .counter(p_counter),
14    .clk(p_clk),
15    .reset(p_reset)
16  );
17
18  initial
19  begin
20    /*$dumpfile("dec_counter.vcd");
21    $dumpvars(0, tb_dec_counter);*/
22
23  p_clk = 1'b0;
24  forever
25  begin
26    #10 p_clk = ~p_clk;
27  end
28
```

```

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

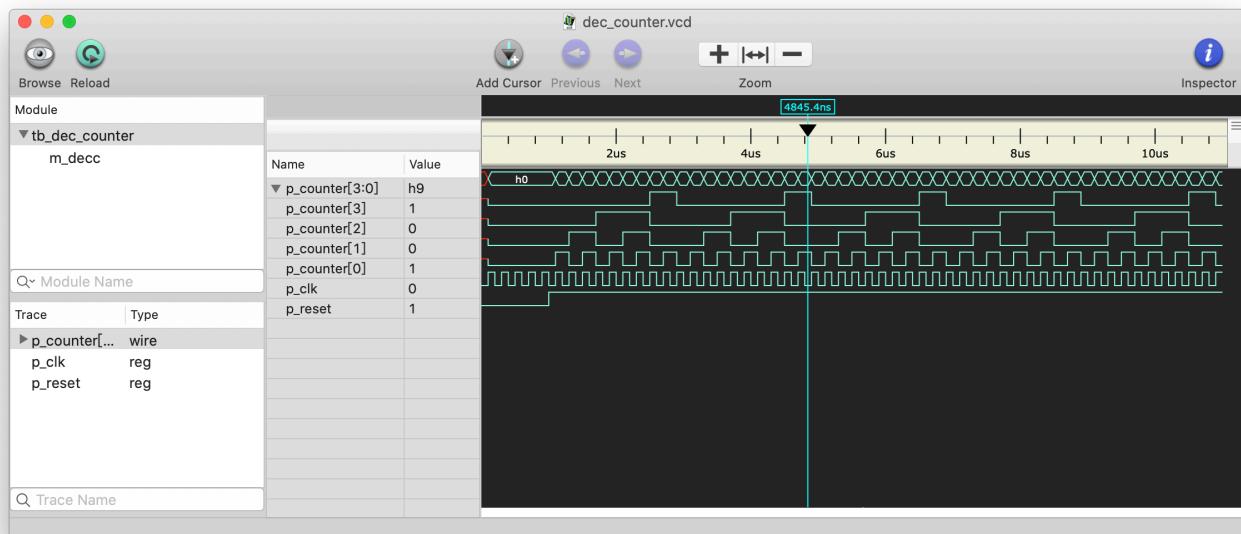


图 18: 测试波形图 2.7

### (4) 显示输出

```

(base) liuhawendeMacBook-Pro:project2_7 hongxing$ iverilog -o dec_counter dec_counter.v
(base) liuhawendeMacBook-Pro:project2_7 hongxing$ ./dec_counter.vpp
VCD info: dumpfile dec_counter.vcd opened for output.
At time          0ns, reset = 0, counter = x.
At time      100ns, reset = 0, counter = 0.
At time     1000ns, reset = 1, counter = 0.
At time     1100ns, reset = 1, counter = 1.
At time     1300ns, reset = 1, counter = 2.
At time     1500ns, reset = 1, counter = 3.
At time     1700ns, reset = 1, counter = 4.
At time     1900ns, reset = 1, counter = 5.
At time     2100ns, reset = 1, counter = 6.
At time     2300ns, reset = 1, counter = 7.
At time     2500ns, reset = 1, counter = 8.
At time     2700ns, reset = 1, counter = 9.
At time     2900ns, reset = 1, counter = 0.
At time     3100ns, reset = 1, counter = 1.
At time     3300ns, reset = 1, counter = 2.
At time     3500ns, reset = 1, counter = 3.
At time     3700ns, reset = 1, counter = 4.
At time     3900ns, reset = 1, counter = 5.
At time     4100ns, reset = 1, counter = 6.
At time     4300ns, reset = 1, counter = 7.
At time     4500ns, reset = 1, counter = 8.
At time     4700ns, reset = 1, counter = 9.
At time     4900ns, reset = 1, counter = 0.
At time     5100ns, reset = 1, counter = 1.
At time     5300ns, reset = 1, counter = 2.

```

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 (

```

```

6   output  y,
7   input   sel ,
8   input  A, B, C, D
9
10 );
11 wire u1, u2;
12
13 nand #3 na1(u1, A, B);
14 nand #4 na2(u2, C, D);
15 bufif0 b0(y, u1, sel);
16 bufif1 b1(y, u2, sel);
17
18 endmodule

```

## (2) 测试模块

```

1 //File: tb_comb_str.v
2
3 `timescale 1 ns / 1 ns
4 `include "comb_str.v"
5
6 module tb_comb_str;
7
8   wire p_y;
9
10  reg p_sel;
11  reg [3:0] temp;
12
13  comb_str m_cb1(
14    .y(p_y),
15    .sel(p_sel),
16    .A(temp[0]),
17    .B(temp[1]),
18    .C(temp[2]),
19    .D(temp[3])
20  );
21
22  initial
23 begin
24   /*$dumpfile("comb_str.vcd");
25   $dumpvars(0, tb_comb_str);*/
26   temp = 4'b0;
27   repeat(16)
28   begin
29     p_sel = 1'b0;
30     #3 p_sel = ~p_sel;
31     #3 p_sel = ~p_sel;
32     #3 p_sel = ~p_sel;
33     #3 temp = temp + 4'b1;
34   end
35
36   $stop;
37 end
38
39 initial
40 begin
41   $monitor("At time %t, 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

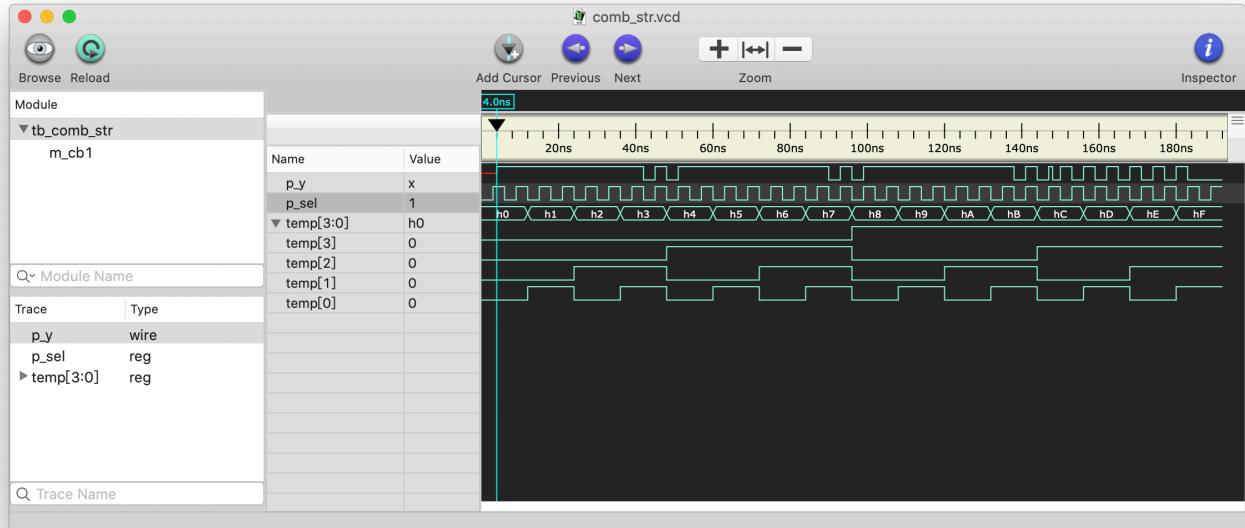


图 20: 测试波形图 2.8

### (4) 显示输出

```

project2_8 -- vvp ./comb_str.vpp -- 76x34
(base) liuhuawendeMacBook-Pro:project2_8 hongxing$ ./comb_str.vpp
VCD info: dumpfile comb_str.vcd opened for output.
At time          0ns, A = 0, B = 0, C = 0, D = 0, sel = 0, y = x.
At time          3ns, A = 0, B = 0, C = 0, D = 0, sel = 1, y = x.
At time          4ns, A = 0, B = 0, C = 0, D = 0, sel = 1, y = 1.
At time          6ns, A = 0, B = 0, C = 0, D = 0, sel = 0, y = 1.
At time          9ns, A = 0, B = 0, C = 0, D = 0, sel = 1, y = 1.
At time         12ns, A = 1, B = 0, C = 0, D = 0, sel = 0, y = 1.
At time         15ns, A = 1, B = 0, C = 0, D = 0, sel = 1, y = 1.
At time         18ns, A = 1, B = 0, C = 0, D = 0, sel = 0, y = 1.
At time         21ns, A = 1, B = 0, C = 0, D = 0, sel = 1, y = 1.
At time         24ns, A = 0, B = 1, C = 0, D = 0, sel = 0, y = 1.
At time         27ns, A = 0, B = 1, C = 0, D = 0, sel = 1, y = 1.
At time         30ns, A = 0, B = 1, C = 0, D = 0, sel = 0, y = 1.
At time         33ns, A = 0, B = 1, C = 0, D = 0, sel = 1, y = 1.
At time         36ns, A = 1, B = 1, C = 0, D = 0, sel = 0, y = 1.
At time         39ns, A = 1, B = 1, C = 0, D = 0, sel = 1, y = 1.
At time         42ns, A = 1, B = 1, C = 0, D = 0, sel = 0, y = 0.
At time         45ns, A = 1, B = 1, C = 0, D = 0, sel = 1, y = 1.
At time         48ns, A = 0, B = 0, C = 1, D = 0, sel = 0, y = 0.
At time         51ns, A = 0, B = 0, C = 1, D = 0, sel = 1, y = 1.
At time         54ns, A = 0, B = 0, C = 1, D = 0, sel = 0, y = 1.
At time         57ns, A = 0, B = 0, C = 1, D = 0, sel = 1, y = 1.
At time         60ns, A = 1, B = 0, C = 1, D = 0, sel = 0, y = 1.
At time         63ns, A = 1, B = 0, C = 1, D = 0, sel = 1, y = 1.
At time         66ns, A = 1, B = 0, C = 1, D = 0, sel = 0, y = 1.
At time         69ns, A = 1, B = 0, C = 1, D = 0, sel = 1, y = 1.
At time         72ns, A = 0, B = 1, C = 1, D = 0, sel = 0, y = 1.
At time         75ns, A = 0, B = 1, C = 1, D = 0, sel = 1, y = 1.
At time         78ns, A = 0, B = 1, C = 1, D = 0, sel = 0, y = 1.
At time         81ns, A = 0, B = 1, C = 1, D = 0, sel = 1, y = 1.
At time         84ns, A = 1, B = 1, C = 1, D = 0, sel = 0, y = 1.
At time         87ns, A = 1, B = 1, C = 1, D = 0, sel = 1, y = 1.
At time         90ns, A = 1, B = 1, C = 1, D = 0, sel = 0, y = 0.

```

图 21: 显示输出 2.8(1)

```

project2_8 -- vvp ./comb_str.vpp -- 76x34
At time          96ns, A = 0, B = 0, C = 0, D = 1, sel = 0, y = 0.
At time          99ns, A = 0, B = 0, C = 0, D = 1, sel = 1, y = 1.
At time         102ns, A = 0, B = 0, C = 0, D = 1, sel = 0, y = 1.
At time         105ns, A = 0, B = 0, C = 0, D = 1, sel = 1, y = 1.
At time         108ns, A = 1, B = 0, C = 0, D = 1, sel = 0, y = 1.
At time         111ns, A = 1, B = 0, C = 0, D = 1, sel = 1, y = 1.
At time         114ns, A = 1, B = 0, C = 0, D = 1, sel = 0, y = 1.
At time         117ns, A = 1, B = 0, C = 0, D = 1, sel = 1, y = 1.
At time         120ns, A = 0, B = 1, C = 0, D = 1, sel = 0, y = 1.
At time         123ns, A = 0, B = 1, C = 0, D = 1, sel = 1, y = 1.
At time         126ns, A = 0, B = 1, C = 0, D = 1, sel = 1, y = 1.
At time         129ns, A = 0, B = 1, C = 0, D = 1, sel = 1, y = 1.
At time         132ns, A = 1, B = 1, C = 0, D = 1, sel = 0, y = 1.
At time         135ns, A = 1, B = 1, C = 0, D = 1, sel = 1, y = 1.
At time         138ns, A = 1, B = 1, C = 0, D = 1, sel = 0, y = 0.
At time         141ns, A = 1, B = 1, C = 0, D = 1, sel = 1, y = 1.
At time         144ns, A = 0, B = 0, C = 1, D = 1, sel = 0, y = 0.
At time         147ns, A = 0, B = 0, C = 1, D = 1, sel = 1, y = 1.
At time         148ns, A = 0, B = 0, C = 1, D = 1, sel = 1, y = 0.
At time         150ns, A = 0, B = 0, C = 1, D = 1, sel = 0, y = 1.
At time         153ns, A = 0, B = 0, C = 1, D = 1, sel = 1, y = 0.
At time         156ns, A = 1, B = 0, C = 1, D = 1, sel = 0, y = 1.
At time         159ns, A = 1, B = 0, C = 1, D = 1, sel = 1, y = 0.
At time         162ns, A = 1, B = 0, C = 1, D = 1, sel = 0, y = 1.
At time         165ns, A = 1, B = 0, C = 1, D = 1, sel = 1, y = 0.
At time         168ns, A = 0, B = 1, C = 1, D = 1, sel = 0, y = 1.
At time         171ns, A = 0, B = 1, C = 1, D = 1, sel = 1, y = 0.
At time         174ns, A = 0, B = 1, C = 1, D = 1, sel = 0, y = 1.
At time         177ns, A = 0, B = 1, C = 1, D = 1, sel = 1, y = 0.
At time         180ns, A = 1, B = 1, C = 1, D = 1, sel = 0, y = 1.
At time         183ns, A = 1, B = 1, C = 1, D = 1, sel = 1, y = 0.
At time         186ns, A = 1, B = 1, C = 1, D = 1, sel = 0, y = 0.
At time         189ns, A = 1, B = 1, C = 1, D = 1, sel = 1, y = 0.
** VVP Stop(0) **
```

图 22: 显示输出 2.8(2)

## 9 第 9 题

### (1) 设计模块

```
1 //File: LFSR.v
2
3 'timescale 1 ns / 1 ns
4
5 module LFSR (
6   output reg [1:26] q, // 26 bit data output.
7   input clk,           // Clock input.
8   input rst_n,         // Synchronous reset input active low.
9   input load,          // Synchronous load input active high.
10  input [1:26] din    // 26 bit parallel data input.
11
12 );
13 always @(posedge clk) begin
14   if(~rst_n) q <= 26'b0;
15   else begin
16     if(load) q <= (|din) ? din : 26'b1;
17     else begin
18       if (!q) q <= 26'b1;
19       else begin
20         q[1] <= q[26];
21         q[2] <= q[1] ^ q[26];
22         q[3:7] <= q[2:6];
23         q[8] <= q[7] ^ q[26];
24         q[9] <= q[8] ^ q[26];
25         q[10:26] <= q[9:25];
26       end
27     end
28   end
29 end
30
31
32 endmodule
```

### (2) 测试模块

```
1 //File: tb_LFSR.v
2
3 'timescale 1 ns / 1 ns
4 `include "LFSR.v"
5
6 module tb_LFSR;
7
8   wire [1:26] p_q;
9
10  reg p_clk, p_rstn, p_load;
11  reg [1:26] p_din;
12
13  LFSR L1(
14    .q      (p_q),
15    .clk    (p_clk),
16    .rst_n  (p_rstn),
17    .load   (p_load),
18    .din    (p_din)
```

```

19 );
20
21 initial begin
22 /*$dumpfile("LFSR.vcd");
23 $dumpvars(0, tb_LFSR);*/
24
25 p_clk = 1'b0;
26 forever begin
27 #10 p_clk = ~p_clk;
28 end
29 end
30
31 initial begin
32 p_rstn = 1'b0;
33 #100 p_rstn = 1'b1;
34 end
35
36 initial begin
37 p_load = 1'b1;
38 p_din = 26'd31;
39 #155 p_load = 1'b0;
40
41 #455 p_load = 1'b1;
42 p_din = 26'd30;
43 #505 p_load = 1'b0;
44
45 #1005 p_load = 1'b1;
46 p_din = 26'b0;
47 #1055 p_load = 1'b0;
48
49 #1500 $stop;
50 end
51
52 initial begin
53 $monitor("At time %tns, rst_n = %b, load = %b, q = %b.", $time, p_rstn, p_load, p_q);
54 end
55
56 endmodule

```

### (3) 测试波形图



## 10 第 10 题

### (1) 设计模块

```
1 //File: filter.v
2 `timescale 1 ns / 1 ns
3
4 module filter (
5   output reg sig_out,
6   input  clock,      // Clock
7   input  reset,     // Asynchronous reset active low
8   input  sig_in
9 );
10 reg [3:0] rgt;
11 reg j, k;
12
13 always @(posedge clock or negedge reset) begin
14   if(~reset) begin
15     sig_out <= 1'b0;
16     rgt <= 4'b0;
17   end else begin
18     rgt[3] <= sig_in;
19     rgt[2] <= rgt[3];
20     rgt[1] <= rgt[2];
21     rgt[0] <= rgt[1];
22     k <= ~rgt[2] & ~rgt[1] & ~rgt[0];
23     j <= rgt[2] & rgt[1] & rgt[0];
24     case ({j,k})
25       2'b01: sig_out <= 1'b0;
26       2'b10: sig_out <= 1'b1;
27       2'b11: sig_out <= ~sig_out;
28       default : /* default */;
29     endcase
30   end
31 end
32
33 endmodule
```

### (2) 测试模块

```
1 //File: tb_filter.v
2 `timescale 1 ns / 1 ns
3 `include "filter.v"
4
5 module tb_filter;
6
7   wire o_sig;
8   reg i_sig;
9   reg p_clk, p_rst;
10
11   filter f1(
12     .sig_out(o_sig),
13     .clock(p_clk),
14     .reset(p_rst),
15     .sig_in(i_sig)
16   );
17
```

```

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

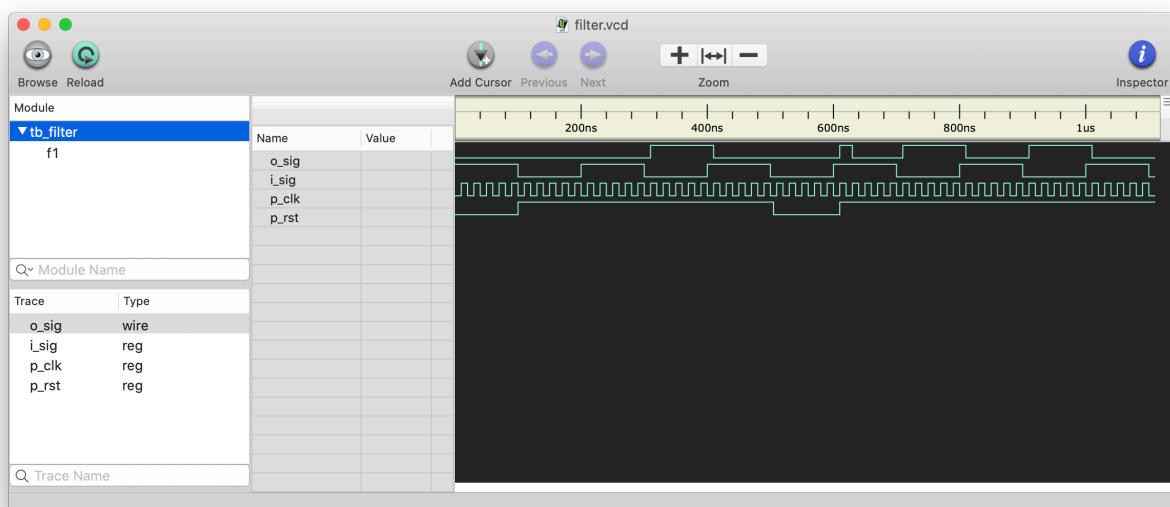
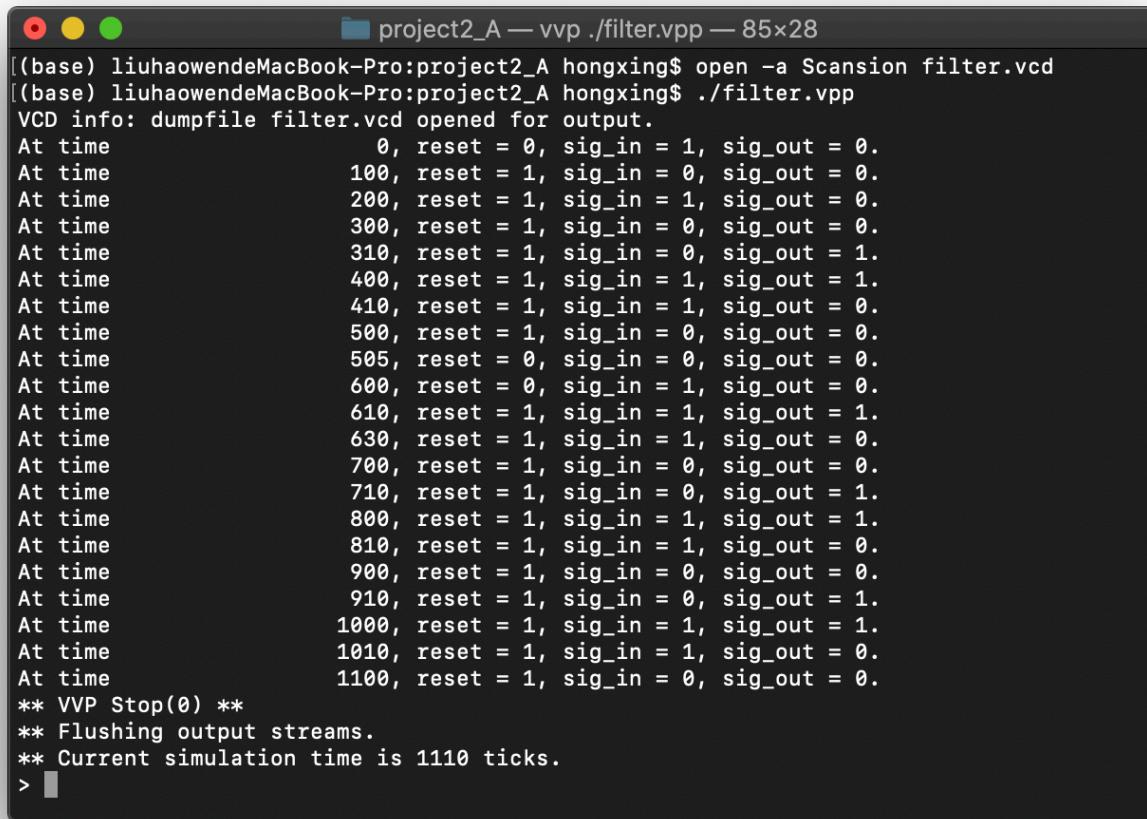


图 26: 测试波形图 2.10

#### (4) 显示输出



```
[base] liuhawendeMacBook-Pro:project2_A hongxing$ open -a Scansion filter.vcd
[base] liuhawendeMacBook-Pro:project2_A hongxing$ ./filter.vpp
VCD info: dumpfile filter.vcd opened for output.
At time          0, reset = 0, sig_in = 1, sig_out = 0.
At time        100, reset = 1, sig_in = 0, sig_out = 0.
At time        200, reset = 1, sig_in = 1, sig_out = 0.
At time        300, reset = 1, sig_in = 0, sig_out = 0.
At time        310, reset = 1, sig_in = 0, sig_out = 1.
At time        400, reset = 1, sig_in = 1, sig_out = 1.
At time        410, reset = 1, sig_in = 1, sig_out = 0.
At time        500, reset = 1, sig_in = 0, sig_out = 0.
At time        505, reset = 0, sig_in = 0, sig_out = 0.
At time        600, reset = 0, sig_in = 1, sig_out = 0.
At time        610, reset = 1, sig_in = 1, sig_out = 1.
At time        630, reset = 1, sig_in = 1, sig_out = 0.
At time        700, reset = 1, sig_in = 0, sig_out = 0.
At time        710, reset = 1, sig_in = 0, sig_out = 1.
At time        800, reset = 1, sig_in = 1, sig_out = 1.
At time        810, reset = 1, sig_in = 1, sig_out = 0.
At time        900, reset = 1, sig_in = 0, sig_out = 0.
At time        910, reset = 1, sig_in = 0, sig_out = 1.
At time       1000, reset = 1, sig_in = 1, sig_out = 1.
At time       1010, reset = 1, sig_in = 1, sig_out = 0.
At time       1100, reset = 1, sig_in = 0, sig_out = 0.
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 1110 ticks.
> |
```

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;
```

```

15    end else case (dir)
16        1'b0: count = count - 1'b1;
17        1'b1: count = count + 1'b1;
18
19    default : /* default */;
20 endcase
21end
22
23endmodule

```

## (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
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

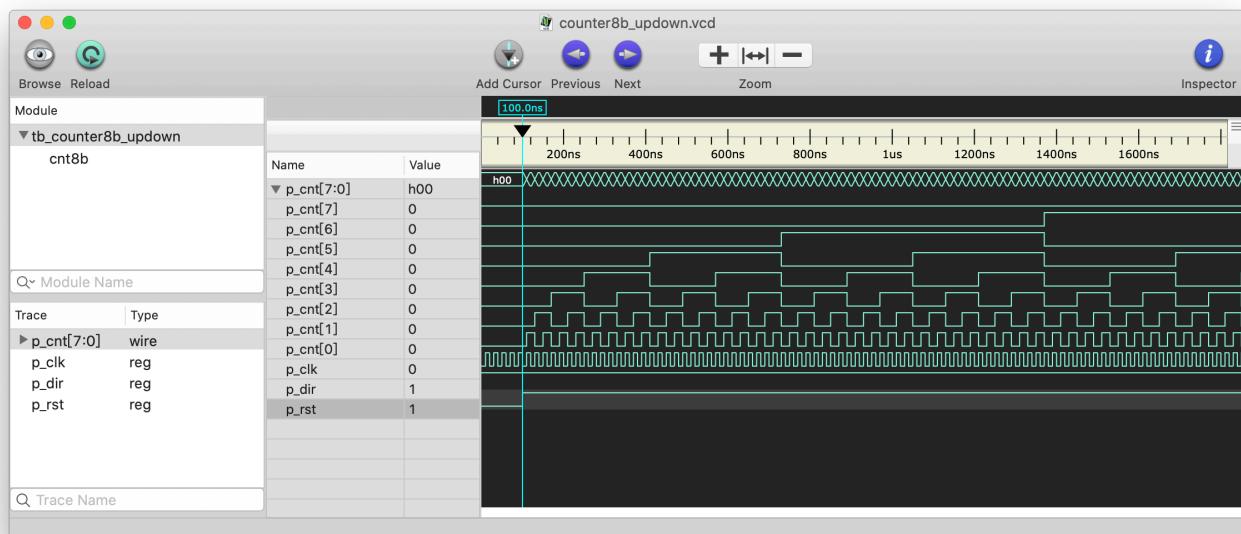


图 28: 测试波形图 2.11 (递增计数)

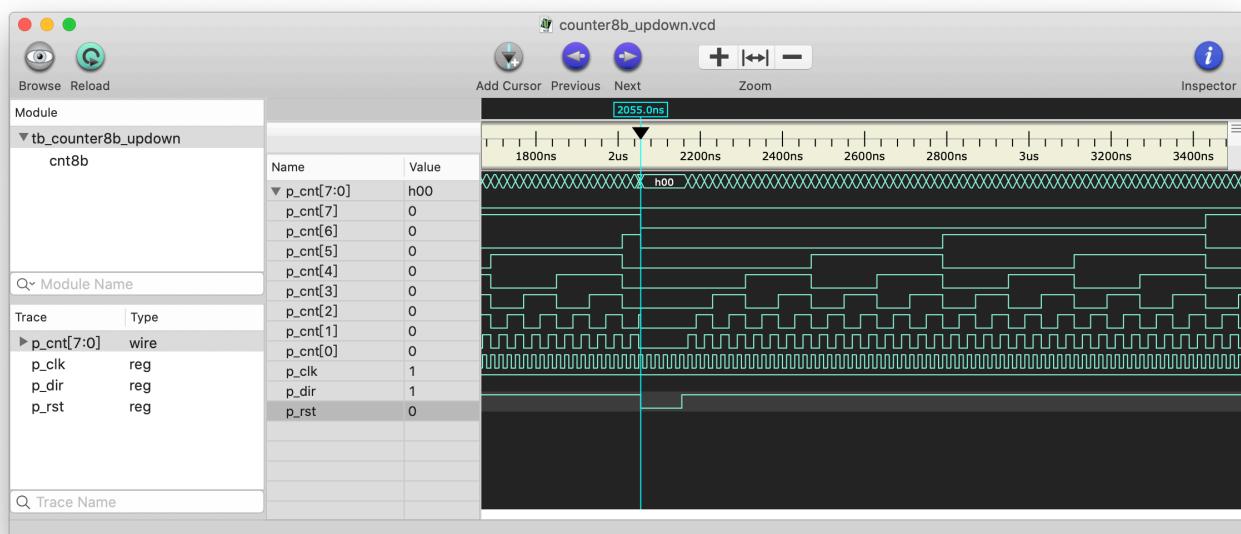


图 29: 测试波形图 2.11 (异步复位)

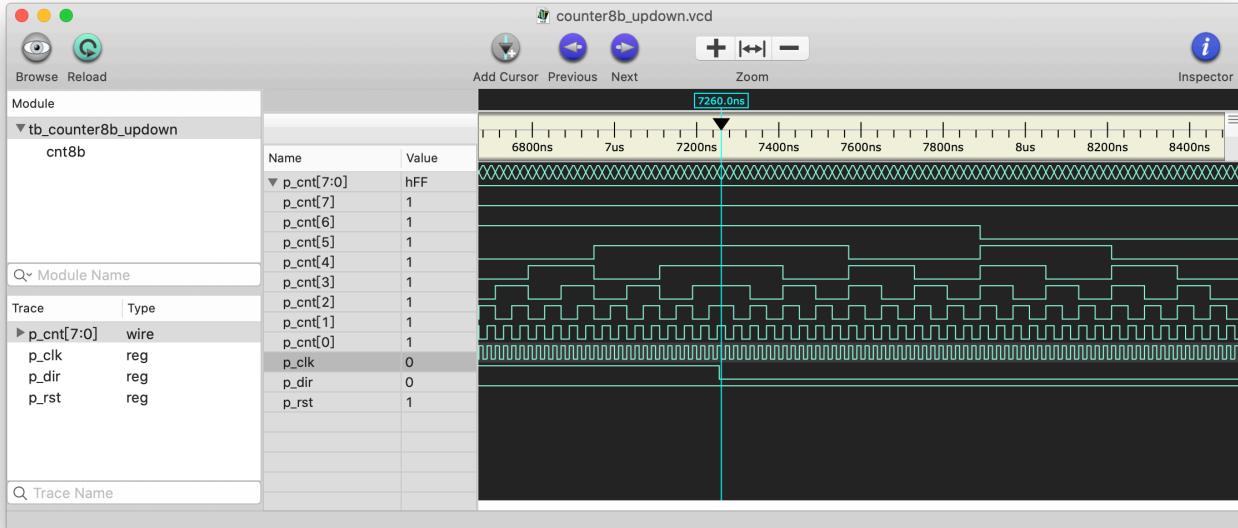


图 30: 测试波形图 2.11 (递减计数)

#### (4) 显示输出

```
project2_B — vvp ./counter8b_updown.vpp — 74x30
At time      1890ns, reset = 1, dir = 1, count = 90.
At time      1910ns, reset = 1, dir = 1, count = 91.
At time      1930ns, reset = 1, dir = 1, count = 92.
At time      1950ns, reset = 1, dir = 1, count = 93.
At time      1970ns, reset = 1, dir = 1, count = 94.
At time      1990ns, reset = 1, dir = 1, count = 95.
At time      2010ns, reset = 1, dir = 1, count = 96.
At time      2030ns, reset = 1, dir = 1, count = 97.
At time      2050ns, reset = 1, dir = 1, count = 98.
At time      2055ns, reset = 0, dir = 1, count = 0.
At time      2155ns, reset = 1, dir = 1, count = 0.
At time      2170ns, reset = 1, dir = 1, count = 1.
At time      2190ns, reset = 1, dir = 1, count = 2.
At time      2210ns, reset = 1, dir = 1, count = 3.
At time      2230ns, reset = 1, dir = 1, count = 4.
At time      2250ns, reset = 1, dir = 1, count = 5.
At time      2270ns, reset = 1, dir = 1, count = 6.
At time      2290ns, reset = 1, dir = 1, count = 7.
At time      2310ns, reset = 1, dir = 1, count = 8.
At time      2330ns, reset = 1, dir = 1, count = 9.
At time      2350ns, reset = 1, dir = 1, count = 10.
At time      2370ns, reset = 1, dir = 1, count = 11.
At time      2390ns, reset = 1, dir = 1, count = 12.
At time      2410ns, reset = 1, dir = 1, count = 13.
At time      2430ns, reset = 1, dir = 1, count = 14.
At time      2450ns, reset = 1, dir = 1, count = 15.
At time      2470ns, reset = 1, dir = 1, count = 16.
At time      2490ns, reset = 1, dir = 1, count = 17.
At time      2510ns, reset = 1, dir = 1, count = 18.
At time      2530ns, reset = 1, dir = 1, count = 19.
```

```
project2_B — vvp ./counter8b_updown.vpp — 74x30
At time      6970ns, reset = 1, dir = 1, count = 241.
At time      6990ns, reset = 1, dir = 1, count = 242.
At time      7010ns, reset = 1, dir = 1, count = 243.
At time      7030ns, reset = 1, dir = 1, count = 244.
At time      7050ns, reset = 1, dir = 1, count = 245.
At time      7070ns, reset = 1, dir = 1, count = 246.
At time      7090ns, reset = 1, dir = 1, count = 247.
At time      7110ns, reset = 1, dir = 1, count = 248.
At time      7130ns, reset = 1, dir = 1, count = 249.
At time      7150ns, reset = 1, dir = 1, count = 250.
At time      7170ns, reset = 1, dir = 1, count = 251.
At time      7190ns, reset = 1, dir = 1, count = 252.
At time      7210ns, reset = 1, dir = 1, count = 253.
At time      7230ns, reset = 1, dir = 1, count = 254.
At time      7250ns, reset = 1, dir = 1, count = 255.
At time      7255ns, reset = 1, dir = 0, count = 255.
At time      7270ns, reset = 1, dir = 0, count = 254.
At time      7290ns, reset = 1, dir = 0, count = 253.
At time      7310ns, reset = 1, dir = 0, count = 252.
At time      7330ns, reset = 1, dir = 0, count = 251.
At time      7350ns, reset = 1, dir = 0, count = 250.
At time      7370ns, reset = 1, dir = 0, count = 249.
At time      7390ns, reset = 1, dir = 0, count = 248.
At time      7410ns, reset = 1, dir = 0, count = 247.
At time      7430ns, reset = 1, dir = 0, count = 246.
At time      7450ns, reset = 1, dir = 0, count = 245.
At time      7470ns, reset = 1, dir = 0, count = 244.
At time      7490ns, reset = 1, dir = 0, count = 243.
At time      7510ns, reset = 1, dir = 0, count = 242.
At time      7530ns, reset = 1, dir = 0, count = 241.
```

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),

```

```

21     .c_in (p_cin)
22 );
23
24 initial begin
25 /*$dumpfile("ALU.vcd");
26 $dumpvars(0, tb_ALU);*/
27
28 p_a = ($random) % 256;
29 {p_cin, p_b} = ($random) % 512;
30
31 p_op = 4'd0;
32
33 repeat(79) begin
34 #10;
35
36 p_a = ($random) % 256;
37 {p_cin, p_b} = ($random) % 512;
38
39 p_op = p_op + 1'b1;
40 end
41
42 #10 $stop;
43 end
44
45 initial begin
46 $monitor("oper = %d, a = %b, b = %b, c_out = %b, sum = %b.", p_op, p_a, p_b, p_cout, p_sum);
47 end
48
49 endmodule

```

### (3) 显示输出

```

(base) liuhawendeMacBook-Pro:project2_C hongxing$ ./ALU.vpp
VCD info: dumpfile ALU.vcd opened for output.
oper = 0, a = 00100100, b = 10000001, c_out = 0, sum = 10100101.
oper = 1, a = 00001001, b = 01100011, c_out = 1, sum = 10100101.
oper = 2, a = 00001101, b = 10001101, c_out = 0, sum = 01111101.
oper = 3, a = 01100101, b = 00010010, c_out = 0, sum = 01110111.
oper = 4, a = 00000001, b = 00001101, c_out = 0, sum = 00000001.
oper = 5, a = 01110101, b = 00111101, c_out = 0, sum = 00010001.
oper = 6, a = 11101101, b = 10001100, c_out = 0, sum = 01100001.
oper = 7, a = 11111001, b = 11000110, c_out = 0, sum = 11000000.
oper = 8, a = 11000101, b = 10101010, c_out = x, sum = xxxxxxxx.
oper = 9, a = 11000101, b = 01110111, c_out = x, sum = xxxxxxxx.
oper = 10, a = 00010010, b = 10001111, c_out = x, sum = xxxxxxxx.
oper = 11, a = 11100110, b = 01011010, c_out = x, sum = xxxxxxxx.
oper = 12, a = 11110100, b = 11001101, c_out = x, sum = xxxxxxxx.
oper = 13, a = 01011100, b = 10111101, c_out = x, sum = xxxxxxxx.
oper = 14, a = 00101101, b = 01100101, c_out = x, sum = xxxxxxxx.
oper = 15, a = 01100011, b = 00001010, c_out = x, sum = xxxxxxxx.
oper = 0, a = 00000000, b = 00100000, c_out = 0, sum = 10100001.
oper = 1, a = 10101010, b = 10011101, c_out = 0, sum = 00001100.
oper = 2, a = 10010110, b = 00010011, c_out = 1, sum = 01111011.
oper = 3, a = 00001101, b = 01010011, c_out = 0, sum = 01011111.
oper = 4, a = 01101011, b = 11010101, c_out = 0, sum = 01000001.
oper = 5, a = 00000010, b = 10101110, c_out = 0, sum = 10101000.
oper = 6, a = 00011101, b = 11001111, c_out = 0, sum = 11010010.
oper = 7, a = 00100011, b = 00001010, c_out = 0, sum = 11010110.
oper = 8, a = 11001010, b = 00111100, c_out = x, sum = xxxxxxxx.
oper = 9, a = 11110010, b = 10001010, c_out = x, sum = xxxxxxxx.
oper = 10, a = 01000001, b = 11011000, c_out = x, sum = xxxxxxxx.
oper = 11, a = 01111000, b = 01010101, c_out = x, sum = xxxxxxxx.
oper = 12, a = 01101100, b = 11001010, c_out = x, sum = xxxxxxxx.
oper = 13, a = 10110101, b = 10010101, c_out = x, sum = xxxxxxxx.
oper = 14, a = 01000110, b = 00000100, c_out = x, sum = xxxxxxxx.
oper = 15, a = 11101111, b = 01101001, c_out = x, sum = xxxxxxxx.
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 800 ticks.

```

图 33: 显示输出 2.12 (1)

```

(base) liuhawendeMacBook-Pro:project2_C hongxing$ ./ALU.vpp
VCD info: dumpfile ALU.vcd opened for output.
oper = 5, a = 10001110, b = 10011100, c_out = 0, sum = 00010000.
oper = 6, a = 11111010, b = 00100010, c_out = 0, sum = 11011100.
oper = 7, a = 01110011, b = 10100011, c_out = 0, sum = 00101111.
oper = 8, a = 00101111, b = 10110011, c_out = x, sum = xxxxxxxx.
oper = 9, a = 01011111, b = 01000100, c_out = x, sum = xxxxxxxx.
oper = 10, a = 11101011, b = 11000111, c_out = x, sum = xxxxxxxx.
oper = 11, a = 11000110, b = 01011010, c_out = x, sum = xxxxxxxx.
oper = 12, a = 11011011, b = 01100101, c_out = x, sum = xxxxxxxx.
oper = 13, a = 10110101, b = 11011111, c_out = x, sum = xxxxxxxx.
oper = 14, a = 01010101, b = 01000100, c_out = x, sum = xxxxxxxx.
oper = 15, a = 01111001, b = 00101011, c_out = x, sum = xxxxxxxx.
oper = 0, a = 11010000, b = 00101010, c_out = 0, sum = 11111011.
oper = 1, a = 10101011, b = 00001100, c_out = 0, sum = 10011101.
oper = 2, a = 11011100, b = 10011010, c_out = 1, sum = 10111100.
oper = 3, a = 11111101, b = 11000011, c_out = 0, sum = 11111111.
oper = 4, a = 01010110, b = 01000110, c_out = 0, sum = 01000110.
oper = 5, a = 01100111, b = 00000101, c_out = 0, sum = 00001000.
oper = 6, a = 10110110, b = 00111000, c_out = 0, sum = 10001110.
oper = 7, a = 01111001, b = 10111000, c_out = 0, sum = 00111110.
oper = 8, a = 10010100, b = 10011000, c_out = x, sum = xxxxxxxx.
oper = 9, a = 11011011, b = 01010101, c_out = x, sum = xxxxxxxx.
oper = 10, a = 11010101, b = 01010101, c_out = x, sum = xxxxxxxx.
oper = 11, a = 11010101, b = 01010101, c_out = x, sum = xxxxxxxx.
oper = 12, a = 11010101, b = 11001010, c_out = x, sum = xxxxxxxx.
oper = 13, a = 10110101, b = 10010101, c_out = x, sum = xxxxxxxx.
oper = 14, a = 01000110, b = 00000100, c_out = x, sum = xxxxxxxx.
oper = 15, a = 11101111, b = 01101001, c_out = x, sum = xxxxxxxx.
** VVP Stop(0) **
** Flushing output streams.
** Current simulation time is 800 ticks.

```

图 34: 显示输出 2.12 (2)

#### (4) 测试波形图

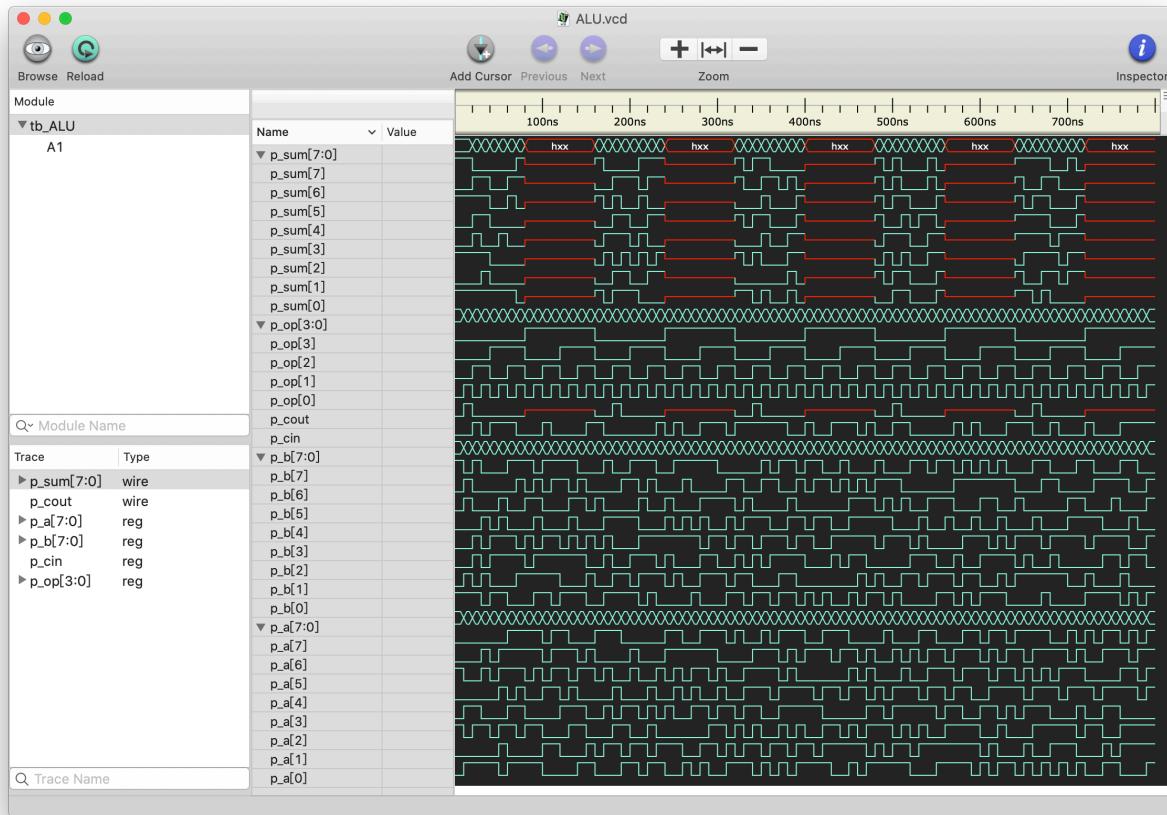


图 35: 测试波形图 2.12

(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

```

1 //File: shift_counter.v
2
3 `timescale 1 ns / 1 ns
4
5 module shift_counter (
6   output reg [7:0] count,
7   input clk,      // Clock
8   input reset     // Synchronous reset active low
9
10 );
11 reg [13:0] temp;
12
13 always @(posedge clk) begin

```

```

14    if(~reset) begin
15        count <= 8'b1;
16        temp <= 14'b1;
17    end else begin
18        temp <= {temp[12:0], temp[13]};
19        if (|temp[13:8])
20            count <= {temp[7], temp[8], temp[9], temp[10], temp[11], temp[12], temp[13], 1'b0};
21        else count <= temp[7:0];
22    end
23 end
24
25 endmodule

```

## (2) 测试模块

```

1 //tb_shift_counter.v
2
3 `timescale 1 ns / 1 ns
4 `include "shift_counter.v"
5
6 module tb_shift_counter;
7
8     wire [7:0] p_cnt;
9
10    reg p_clk, p_rst;
11
12    shift_counter shc1(
13        .count(p_cnt),
14        .clk(p_clk),
15        .reset(p_rst)
16    );
17
18    initial begin
19        /*$dumpfile("shift_counter.vcd");
20        $dumpvars(0, tb_shift_counter);*/
21
22        p_clk = 1'b0;
23        forever begin
24            #10 p_clk = ~p_clk;
25        end
26    end
27
28    initial begin
29        p_rst = 1'b0;
30        #55 p_rst = 1'b1;
31
32        #2000 $stop;
33    end
34
35    initial begin
36        $monitor("At time %tns, reset =%b, count =%b", $time, p_rst, p_cnt);
37    end
38
39 endmodule

```

## (3) 测试波形图

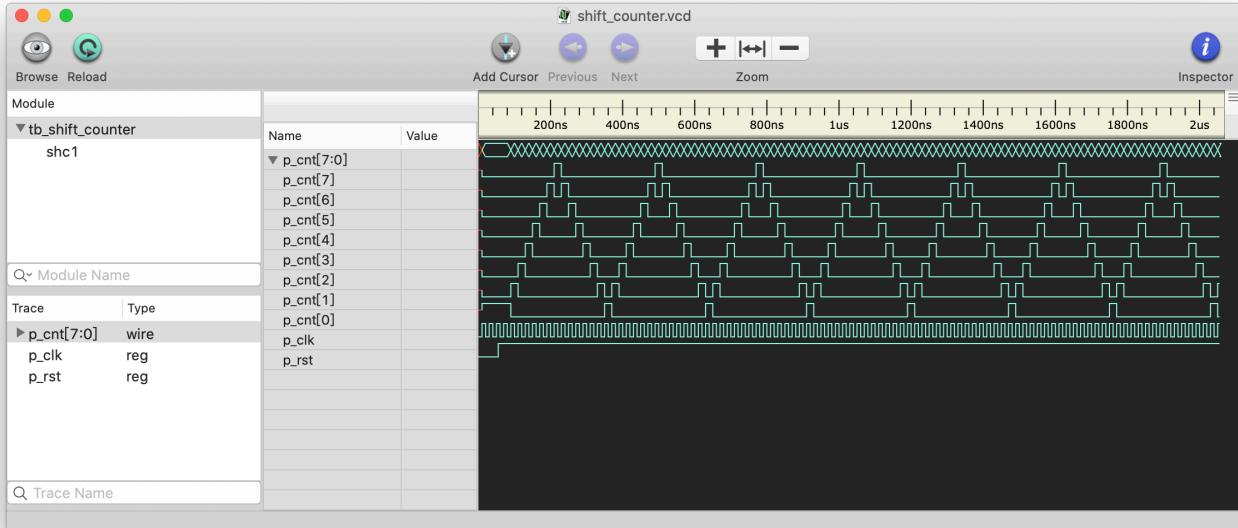


图 36: 测试波形图 2.13

#### (4) 显示输出

```
(base) liuhuawendeMacBook-Pro:project2_D hongxing$ ./shift_counter
.vpp
VCD info: dumpfile shift_counter.vcd opened for output.
At time          0ns, reset = 0, count = xxxxxxxx
At time       10ns, reset = 0, count = 00000001
At time       55ns, reset = 1, count = 00000001
At time      90ns, reset = 1, count = 00000010
At time     110ns, reset = 1, count = 00000100
At time     130ns, reset = 1, count = 00001000
At time     150ns, reset = 1, count = 00010000
At time     170ns, reset = 1, count = 00100000
At time     190ns, reset = 1, count = 01000000
At time     210ns, reset = 1, count = 10000000
At time     230ns, reset = 1, count = 01000000
At time     250ns, reset = 1, count = 00100000
At time     270ns, reset = 1, count = 00010000
At time     290ns, reset = 1, count = 00001000
At time     310ns, reset = 1, count = 00000100
At time     330ns, reset = 1, count = 00000010
At time     350ns, reset = 1, count = 00000001
At time     370ns, reset = 1, count = 00000010
At time     390ns, reset = 1, count = 00000100
At time     410ns, reset = 1, count = 00001000
At time     430ns, reset = 1, count = 00010000
At time     450ns, reset = 1, count = 00100000
At time     470ns, reset = 1, count = 01000000
At time     490ns, reset = 1, count = 10000000
At time     510ns, reset = 1, count = 01000000
At time     530ns, reset = 1, count = 00100000
At time     550ns, reset = 1, count = 00010000
```

图 37: 显示输出 2.13 (1)

```
At time          1450ns, reset = 1, count = 00000010
At time        1470ns, reset = 1, count = 00000001
At time        1490ns, reset = 1, count = 00000010
At time        1510ns, reset = 1, count = 00000100
At time        1530ns, reset = 1, count = 00001000
At time        1550ns, reset = 1, count = 00010000
At time        1570ns, reset = 1, count = 00100000
At time        1590ns, reset = 1, count = 01000000
At time        1610ns, reset = 1, count = 10000000
At time        1630ns, reset = 1, count = 01000000
At time        1650ns, reset = 1, count = 00100000
At time        1670ns, reset = 1, count = 00010000
At time        1690ns, reset = 1, count = 00001000
At time        1710ns, reset = 1, count = 00000100
At time        1730ns, reset = 1, count = 00000010
At time        1750ns, reset = 1, count = 00000001
At time        1770ns, reset = 1, count = 00000010
At time        1790ns, reset = 1, count = 00000100
At time        1810ns, reset = 1, count = 00000000
At time        1830ns, reset = 1, count = 00010000
At time        1850ns, reset = 1, count = 00100000
At time        1870ns, reset = 1, count = 01000000
At time        1890ns, reset = 1, count = 10000000
At time        1910ns, reset = 1, count = 01000000
At time        1930ns, reset = 1, count = 00100000
At time        1950ns, reset = 1, count = 00010000
At time        1970ns, reset = 1, count = 00001000
At time        1990ns, reset = 1, count = 00000100
At time        2010ns, reset = 1, count = 00000010
At time        2030ns, reset = 1, count = 00000001
```

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

```
1 //File: sram.v
2
3 'timescale 1 ns / 1 ns
4
5 module sram #(parameter DATA_WIDTH = 8, ADDR_WIDTH = 8)
6 (
7   output [DATA_WIDTH - 1:0] dout,
8   input [DATA_WIDTH - 1:0] din,
9   input [ADDR_WIDTH - 1:0] addr,
10  input wr,
11  input rd,
12  input cs
13
14 );
15 parameter RAM_DEPTH = 1 << ADDR_WIDTH;
16 reg [DATA_WIDTH - 1:0] mem [0:RAM_DEPTH - 1];
17
18 reg [DATA_WIDTH - 1:0] data;
19 reg flag;
20
21 always @(posedge rd, addr, posedge flag) begin
22   if(~rd) begin
23     flag <= 1;
24     data <= mem[addr];
25   end
26   else data <= 8'bz;
27 end
28
29 always @(posedge wr) begin
30   if(cs) begin
31     if(~rd) flag <= 0;
32     mem[addr] <= din;
33     #1;
34     if(~rd) flag <= 1;
35
36   end
37 end
38
39 assign dout = (cs&&flag) ? data : 8'bz;
40
41 endmodule
```

### (2) 测试模块

```
1 //File: tb_sram.v
2
3 'timescale 1 ns / 1 ns
4 `include "sram.v"
5
6 module tb_sram;
7
8   wire [7:0] p_dout;
```

```

10  reg [7:0] p_din;
11  reg [7:0] p_addr;
12  reg p_wr, p_rd, p_cs;
13
14 sram sram1(
15   .dout(p_dout),
16   .din (p_din),
17   .addr(p_addr),
18   .wr  (p_wr),
19   .rd  (p_rd),
20   .cs  (p_cs)
21 );
22
23 initial begin
24 /*$dumpfile("sram.vcd");
25 $dumpvars(0, tb_sram);*/
26
27 p_addr = 8'b0;
28 repeat(200) begin
29   p_din = p_addr;
30   #10 p_addr = p_addr + 1'b1;
31 end
32
33 p_addr = 8'b0;
34 p_din = 8'b1111_1111;
35 repeat(256) begin
36   #10 p_addr = p_addr + 1'b1;
37 end
38 end
39
40 initial begin
41   p_cs = 1'b0;
42   #100 p_cs = 1'b1;
43 end
44
45 initial begin
46   p_wr = 1'b0;
47   forever begin
48     #5 p_wr = ~p_wr;
49   end
50 end
51
52 initial begin
53   p_rd = 1'b1;
54   #2000 p_rd = 1'b0;
55
56   #2600 $stop;
57 end
58
59 initial begin
60   $monitor("At time %ts, cs =%b, rd =%b, wr =%b, addr =%b, din =%b, dout =%b", $time, p_cs, p_rd,
61   p_wr, p_addr, p_din, p_dout);
62 end
63 endmodule

```

### (3) 测试波形图

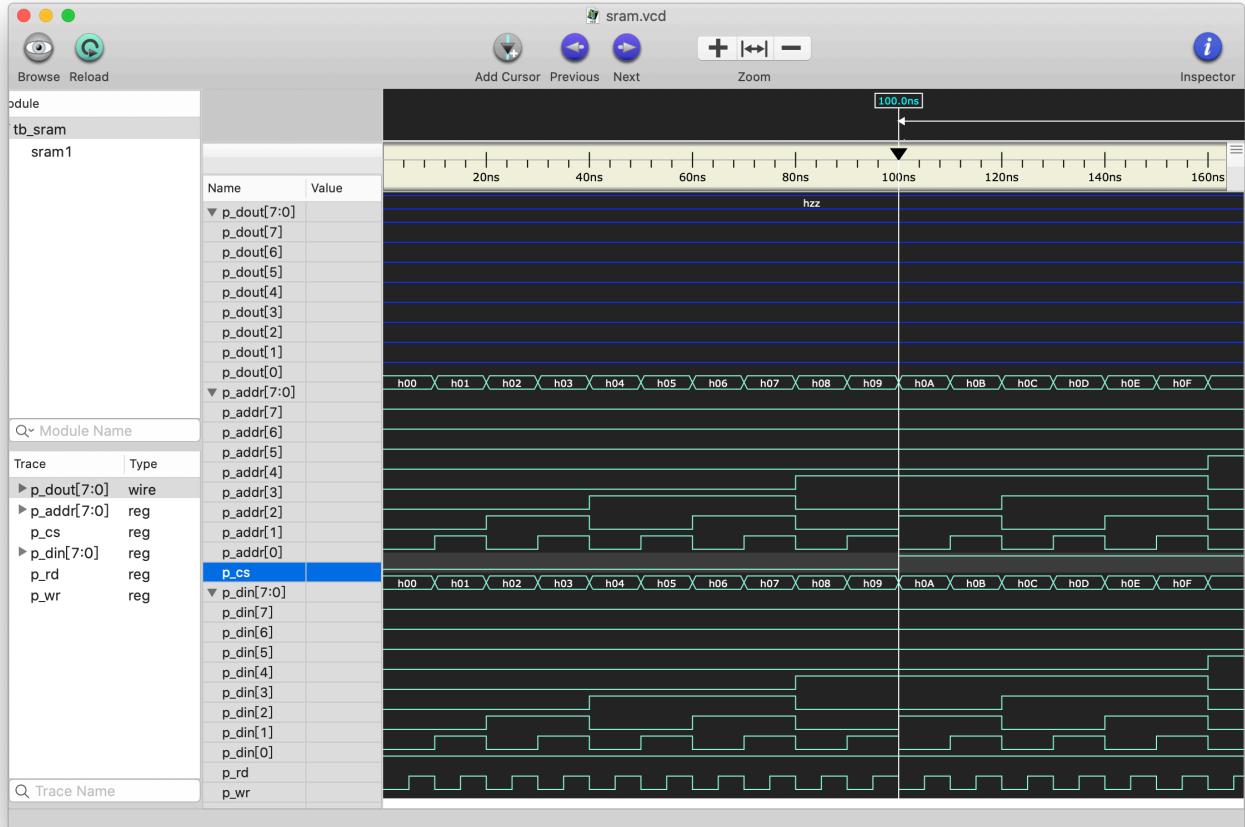


图 39: 片选从无效到有效, 读无效, 写周期有效

### 波形解读:

- 仿真时写信号全程周期有效, 即作为脉冲输入。
- 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  $8'b1$ 。
- $cs$  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。
- 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)。
- 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。

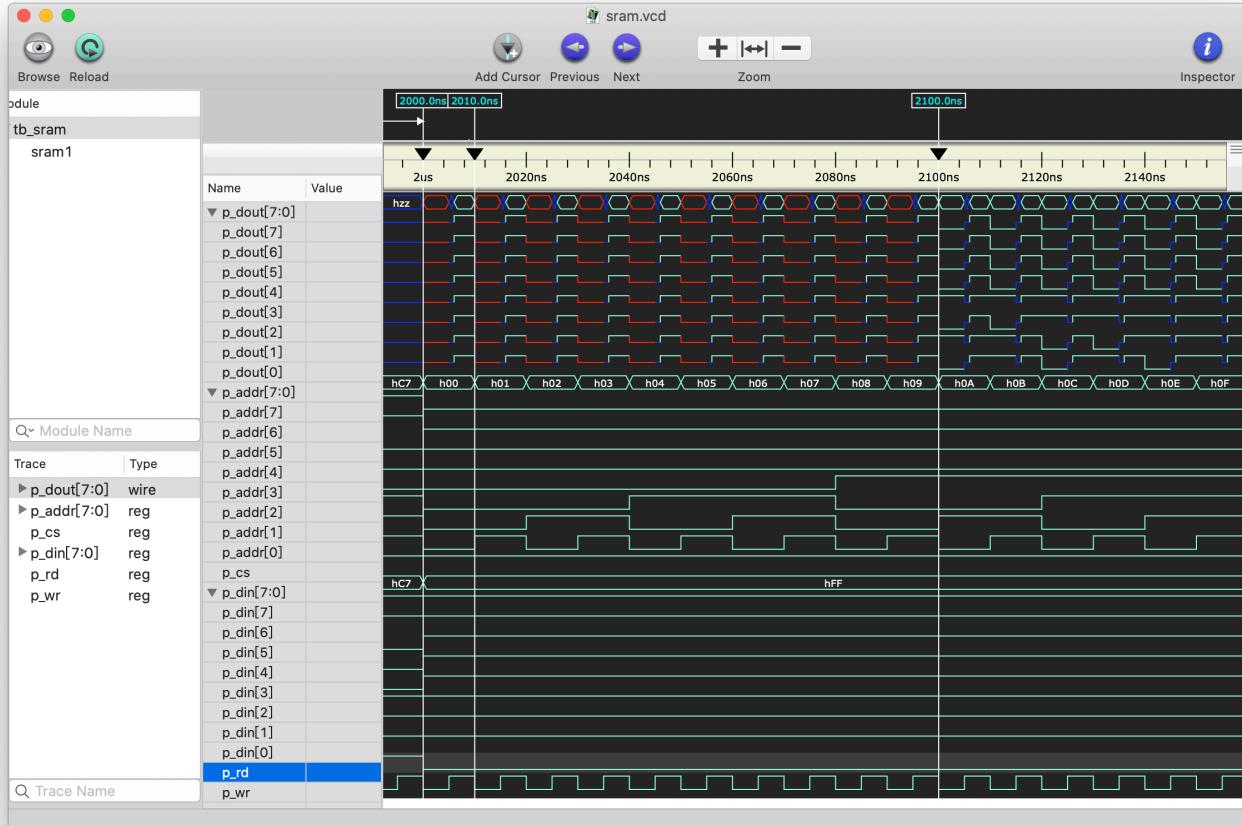


图 40: 片选有效, 读有效, 写周期有效

#### (4) 显示输出

```
project2_E -- bash -- 109x18
At time      35ns, cs = 0, rd = 1, wr = 1, addr = 00000011, din = 00000011, dout = zzzzzzzz
At time      40ns, cs = 0, rd = 1, wr = 0, addr = 00000100, din = 00000100, dout = zzzzzzzz
At time      45ns, cs = 0, rd = 1, wr = 1, addr = 00000100, din = 00000100, dout = zzzzzzzz
At time      50ns, cs = 0, rd = 1, wr = 0, addr = 00000101, din = 00000101, dout = zzzzzzzz
At time      55ns, cs = 0, rd = 1, wr = 1, addr = 00000101, din = 00000101, dout = zzzzzzzz
At time      60ns, cs = 0, rd = 1, wr = 0, addr = 00000110, din = 00000110, dout = zzzzzzzz
At time      65ns, cs = 0, rd = 1, wr = 1, addr = 00000110, din = 00000110, dout = zzzzzzzz
At time      70ns, cs = 0, rd = 1, wr = 0, addr = 00000111, din = 00000111, dout = zzzzzzzz
At time      75ns, cs = 0, rd = 1, wr = 1, addr = 00000111, din = 00000111, dout = zzzzzzzz
At time      80ns, cs = 0, rd = 1, wr = 0, addr = 00001000, din = 00001000, dout = zzzzzzzz
At time      85ns, cs = 0, rd = 1, wr = 1, addr = 00001000, din = 00001000, dout = zzzzzzzz
At time      90ns, cs = 0, rd = 1, wr = 0, addr = 00001001, din = 00001001, dout = zzzzzzzz
At time      95ns, cs = 0, rd = 1, wr = 1, addr = 00001001, din = 00001001, dout = zzzzzzzz
At time     100ns, cs = 1, rd = 1, wr = 0, addr = 00001010, din = 00001010, dout = zzzzzzzz
At time     105ns, cs = 1, rd = 1, wr = 1, addr = 00001010, din = 00001010, dout = zzzzzzzz
At time     110ns, cs = 1, rd = 1, wr = 0, addr = 00001011, din = 00001011, dout = zzzzzzzz
At time     115ns, cs = 1, rd = 1, wr = 1, addr = 00001011, din = 00001011, dout = zzzzzzzz
At time     120ns, cs = 1, rd = 1, wr = 0, addr = 00001100, din = 00001100, dout = zzzzzzzz
```

图 41: 片选无效到片选有效

```

project2_E -- vvp ./sram.vpp -- 109x30
At time      1975ns, cs = 1, rd = 1, wr = 1, addr = 11000101, din = 11000101, dout = zzzzzzzz
At time      1980ns, cs = 1, rd = 1, wr = 0, addr = 11000110, din = 11000110, dout = zzzzzzzz
At time      1985ns, cs = 1, rd = 1, wr = 1, addr = 11000110, din = 11000110, dout = zzzzzzzz
At time      1990ns, cs = 1, rd = 1, wr = 0, addr = 11000111, din = 11000111, dout = zzzzzzzz
At time      1995ns, cs = 1, rd = 1, wr = 1, addr = 11000111, din = 11000111, dout = zzzzzzzz
At time      2000ns, cs = 1, rd = 0, wr = 0, addr = 00000000, din = 11111111, dout = xxxxxxxx
At time      2005ns, cs = 1, rd = 0, wr = 1, addr = 00000000, din = 11111111, dout = zzzzzzzz
At time      2006ns, cs = 1, rd = 0, wr = 1, addr = 00000000, din = 11111111, dout = 11111111
At time      2010ns, cs = 1, rd = 0, wr = 0, addr = 00000001, din = 11111111, dout = xxxxxxxx
At time      2015ns, cs = 1, rd = 1, wr = 1, addr = 00000001, din = 11111111, dout = zzzzzzzz
At time      2016ns, cs = 1, rd = 0, wr = 1, addr = 00000001, din = 11111111, dout = 11111111
At time      2020ns, cs = 1, rd = 0, wr = 0, addr = 00000010, din = 11111111, dout = xxxxxxxx
At time      2025ns, cs = 1, rd = 0, wr = 1, addr = 00000010, din = 11111111, dout = zzzzzzzz
At time      2026ns, cs = 1, rd = 0, wr = 1, addr = 00000010, din = 11111111, dout = 11111111
At time      2030ns, cs = 1, rd = 0, wr = 0, addr = 00000011, din = 11111111, dout = xxxxxxxx
At time      2035ns, cs = 1, rd = 0, wr = 1, addr = 00000011, din = 11111111, dout = zzzzzzzz
At time      2036ns, cs = 1, rd = 0, wr = 1, addr = 00000011, din = 11111111, dout = 11111111
At time      2040ns, cs = 1, rd = 0, wr = 0, addr = 00000100, din = 11111111, dout = xxxxxxxx
At time      2045ns, cs = 1, rd = 0, wr = 1, addr = 00000100, din = 11111111, dout = zzzzzzzz
At time      2046ns, cs = 1, rd = 0, wr = 1, addr = 00000100, din = 11111111, dout = 11111111
At time      2050ns, cs = 1, rd = 0, wr = 0, addr = 00000101, din = 11111111, dout = xxxxxxxx
At time      2055ns, cs = 1, rd = 0, wr = 1, addr = 00000101, din = 11111111, dout = zzzzzzzz
At time      2056ns, cs = 1, rd = 0, wr = 1, addr = 00000101, din = 11111111, dout = 11111111
At time      2060ns, cs = 1, rd = 0, wr = 0, addr = 00000110, din = 11111111, dout = xxxxxxxx
At time      2065ns, cs = 1, rd = 0, wr = 1, addr = 00000110, din = 11111111, dout = zzzzzzzz
At time      2066ns, cs = 1, rd = 0, wr = 1, addr = 00000110, din = 11111111, dout = 11111111
At time      2070ns, cs = 1, rd = 0, wr = 0, addr = 00000111, din = 11111111, dout = xxxxxxxx
At time      2075ns, cs = 1, rd = 0, wr = 1, addr = 00000111, din = 11111111, dout = zzzzzzzz
At time      2076ns, cs = 1, rd = 0, wr = 1, addr = 00000111, din = 11111111, dout = 11111111
At time      2080ns, cs = 1, rd = 0, wr = 0, addr = 00001000, din = 11111111, dout = xxxxxxxx

```

图 42: 读无效到读有效

```

project2_E -- vvp ./sram.vpp -- 109x30
At time      2080ns, cs = 1, rd = 0, wr = 0, addr = 00001000, din = 11111111, dout = xxxxxxxx
At time      2085ns, cs = 1, rd = 0, wr = 1, addr = 00001000, din = 11111111, dout = zzzzzzzz
At time      2086ns, cs = 1, rd = 0, wr = 1, addr = 00001000, din = 11111111, dout = 11111111
At time      2090ns, cs = 1, rd = 0, wr = 0, addr = 00001001, din = 11111111, dout = xxxxxxxx
At time      2095ns, cs = 1, rd = 0, wr = 1, addr = 00001001, din = 11111111, dout = zzzzzzzz
At time      2096ns, cs = 1, rd = 1, wr = 1, addr = 00001001, din = 11111111, dout = 11111111
At time      2100ns, cs = 1, rd = 0, wr = 0, addr = 00001010, din = 11111111, dout = 00001010
At time      2105ns, cs = 1, rd = 0, wr = 1, addr = 00001010, din = 11111111, dout = zzzzzzzz
At time      2106ns, cs = 1, rd = 0, wr = 1, addr = 00001010, din = 11111111, dout = 11111111
At time      2110ns, cs = 1, rd = 0, wr = 0, addr = 00001011, din = 11111111, dout = 00001011
At time      2115ns, cs = 1, rd = 0, wr = 1, addr = 00001011, din = 11111111, dout = zzzzzzzz
At time      2116ns, cs = 1, rd = 0, wr = 1, addr = 00001011, din = 11111111, dout = 11111111
At time      2120ns, cs = 1, rd = 0, wr = 0, addr = 00001100, din = 11111111, dout = 00001100
At time      2125ns, cs = 1, rd = 0, wr = 1, addr = 00001100, din = 11111111, dout = zzzzzzzz
At time      2126ns, cs = 1, rd = 0, wr = 1, addr = 00001100, din = 11111111, dout = 11111111
At time      2130ns, cs = 1, rd = 0, wr = 0, addr = 00001101, din = 11111111, dout = 00001101
At time      2135ns, cs = 1, rd = 0, wr = 1, addr = 00001101, din = 11111111, dout = zzzzzzzz
At time      2136ns, cs = 1, rd = 0, wr = 1, addr = 00001101, din = 11111111, dout = 11111111
At time      2140ns, cs = 1, rd = 0, wr = 0, addr = 00001110, din = 11111111, dout = 00001110
At time      2145ns, cs = 1, rd = 0, wr = 1, addr = 00001110, din = 11111111, dout = zzzzzzzz
At time      2146ns, cs = 1, rd = 0, wr = 1, addr = 00001110, din = 11111111, dout = 11111111
At time      2150ns, cs = 1, rd = 0, wr = 0, addr = 00001111, din = 11111111, dout = 00001111
At time      2155ns, cs = 1, rd = 0, wr = 1, addr = 00001111, din = 11111111, dout = zzzzzzzz
At time      2156ns, cs = 1, rd = 0, wr = 1, addr = 00001111, din = 11111111, dout = 11111111
At time      2160ns, cs = 1, rd = 0, wr = 0, addr = 00010000, din = 11111111, dout = 00010000
At time      2165ns, cs = 1, rd = 0, wr = 1, addr = 00010000, din = 11111111, dout = zzzzzzzz
At time      2166ns, cs = 1, rd = 0, wr = 1, addr = 00010000, din = 11111111, dout = 11111111
At time      2170ns, cs = 1, rd = 0, wr = 0, addr = 00010001, din = 11111111, dout = 00010001
At time      2175ns, cs = 1, rd = 0, wr = 1, addr = 00010001, din = 11111111, dout = zzzzzzzz
At time      2176ns, cs = 1, rd = 0, wr = 1, addr = 00010001, din = 11111111, dout = 11111111

```

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

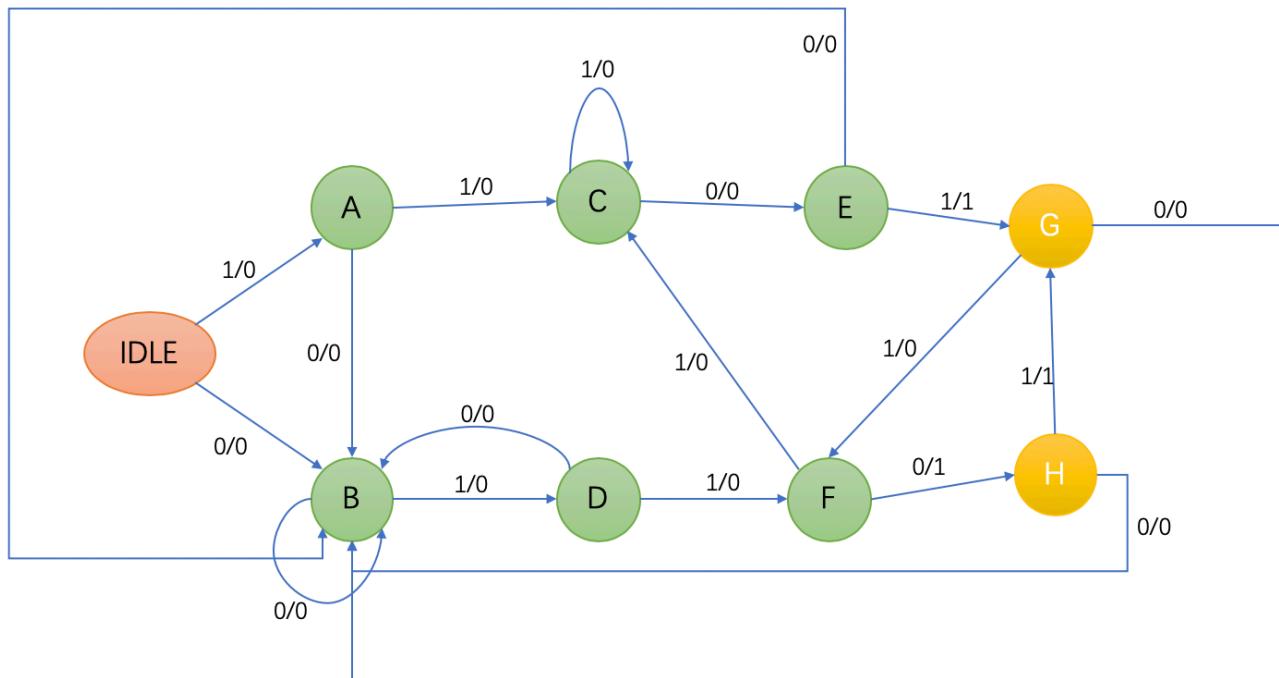


图 44: 序列检测器的状态转移图 (Mealy 机)

```

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 ,
  
```

```

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 @(negedge clk) begin
25   if(~rst_n) begin
26     flag <= 1'b0;
27     state <= IDLE;
28   end else begin
29     case (state)
30       IDLE : if(din) begin flag <= 1'b0; state <= A; end
31             else begin flag <= 1'b0; state <= B; end
32
33       A    : if(din) begin flag <= 1'b0; state <= C; end
34             else begin flag <= 1'b0; state <= B; end
35
36       B    : if(din) begin flag <= 1'b0; state <= D; end
37             else begin flag <= 1'b0; state <= B; end
38
39       C    : if(din) begin flag <= 1'b0; state <= C; end
40             else begin flag <= 1'b0; state <= E; end
41
42       D    : if(din) begin flag <= 1'b0; state <= F; end
43             else begin flag <= 1'b0; state <= B; end
44
45       E    : if(din) begin flag <= 1'b1; state <= G; end
46             else begin flag <= 1'b0; state <= B; end
47
48       F    : if(din) begin flag <= 1'b0; state <= C; end
49             else begin flag <= 1'b1; state <= H; end
50
51       G    : if(din) begin flag <= 1'b0; state <= F; end
52             else begin flag <= 1'b0; state <= B; end
53
54       H    : if(din) begin flag <= 1'b1; state <= G; end
55             else begin flag <= 1'b0; state <= B; end
56
57     default : begin flag <= 1'b0; state <= IDLE; end
58   endcase
59 end
60
61
62 endmodule

```

## (2) 测试模块

```

1 //File: tb_seq_detect.v
2
3 `timescale 1 ns / 100 ps
4 `include "seq_detect.v"
5

```

```

6 module tb_seq_detect;
7
8   wire p_flag;
9
10  reg p_clk, p_rst, p_din;
11
12  seq_detect seqd1(
13    .flag (p_flag),
14    .din  (p_din),
15    .clk   (p_clk),
16    .rst_n(p_rst)
17  );
18
19  initial begin
20   /*$dumpfile("seq_detect.vcd");
21   $dumpvars(0, tb_seq_detect);*/
22
23   p_rst = 1'b0;
24   #25 p_rst = 1'b1;
25 end
26
27 initial begin
28   p_clk = 1'b0;
29   forever begin
30     #10 p_clk = ~p_clk;
31   end
32 end
33
34 parameter SIZE = 32;
35 reg [SIZE - 1 : 0] data = 32'b0001_0110_1110_0001_0111_1010_0000_1101;
36
37 initial begin : SERIES
38   integer i;
39   p_din = 1'b0;
40   #30;
41   for (i = 0; i < SIZE; i++) begin
42     p_din = data[SIZE - 1];
43     data = data << 1;
44     #20;
45   end
46
47   $stop;
48 end
49
50 initial begin
51   #10 $display("At time %tns, reset = %b, datain = %b, flag = %b.", $time, p_rst, p_din, p_flag);
52   forever begin
53     #20 $display("At time %tns, reset = %b, datain = %b, flag = %b.", $time, p_rst, p_din, p_flag);
54   end
55 end
56
57 endmodule

```

### (3) 测试波形图

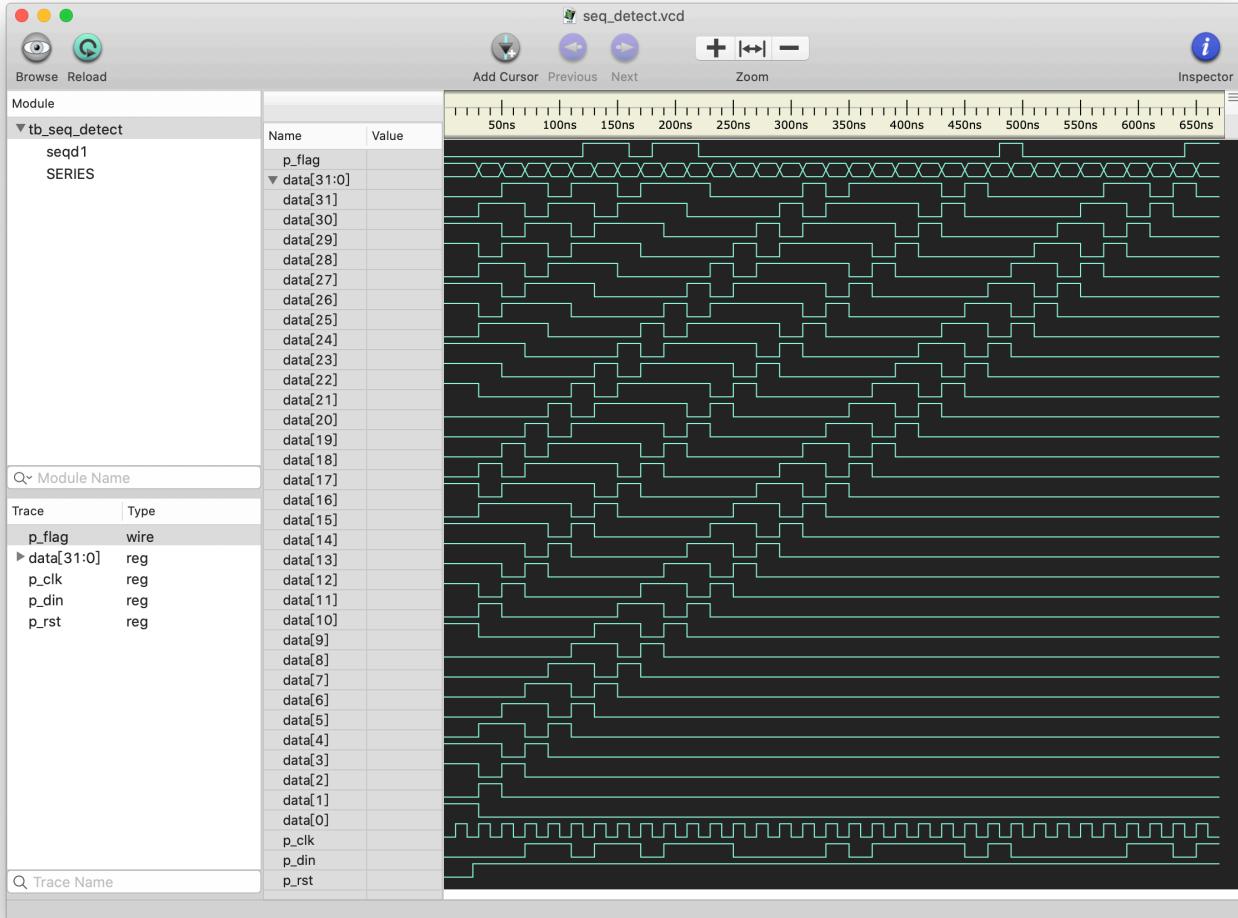


图 45: 测试波形图 2.15

#### (4) 显示输出

```
(base) liuhuawendeMacBook-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)

```
(base) liuhuawendeMacBook-Pro:project2_F hongxing$ ./seq_detect.vpp
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

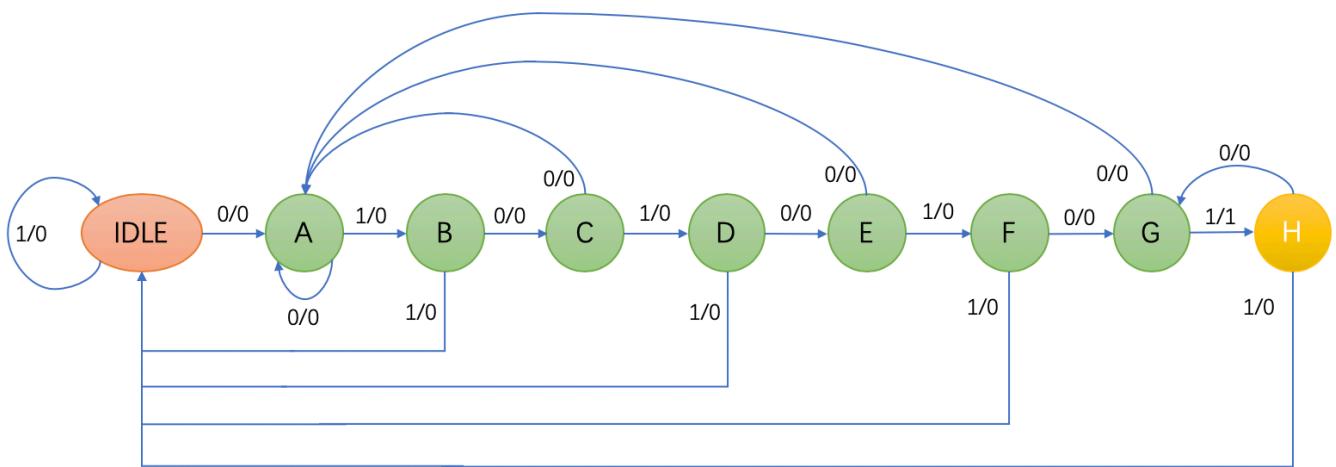


图 48: 序列检测器的状态转移图 (Mealy 机)

```
//File: mealy.v
`timescale 1 ns / 100 ps

module mealy (
    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
        case (state)
            IDLE : if(din) begin flag <= 1'b0; state <= IDLE; end
                    else begin flag <= 1'b0; state <= A; end
            A : if(din) begin flag <= 1'b0; state <= B; end
            B : if(din) begin flag <= 1'b0; state <= C; end
            C : if(din) begin flag <= 1'b0; state <= D; end
            D : if(din) begin flag <= 1'b0; state <= E; end
            E : if(din) begin flag <= 1'b0; state <= F; end
            F : if(din) begin flag <= 1'b0; state <= G; end
            G : if(din) begin flag <= 1'b0; state <= H; end
            H : if(din) begin flag <= 1'b0; state <= IDLE; end
        endcase
    end
end
```

```

34         else begin flag <= 1'b0; state <= A; end
35
36     B : if(din) begin flag <= 1'b0; state <= IDLE; end
37     else begin flag <= 1'b0; state <= C; end
38
39     C : if(din) begin flag <= 1'b0; state <= D; end
40     else begin flag <= 1'b0; state <= A; end
41
42     D : if(din) begin flag <= 1'b0; state <= IDLE; end
43     else begin flag <= 1'b0; state <= E; end
44
45     E : if(din) begin flag <= 1'b0; state <= F; end
46     else begin flag <= 1'b0; state <= A; end
47
48     F : if(din) begin flag <= 1'b0; state <= IDLE; end
49     else begin flag <= 1'b0; state <= G; end
50
51     G : if(din) begin flag <= 1'b1; state <= H; end
52     else begin flag <= 1'b0; state <= A; end
53
54     H : if(din) begin flag <= 1'b0; state <= IDLE; end
55     else begin flag <= 1'b0; state <= G; end
56
57     default : begin flag <= 1'b0; state <= IDLE; end
58   endcase
59 end
60
61
62 endmodule

```

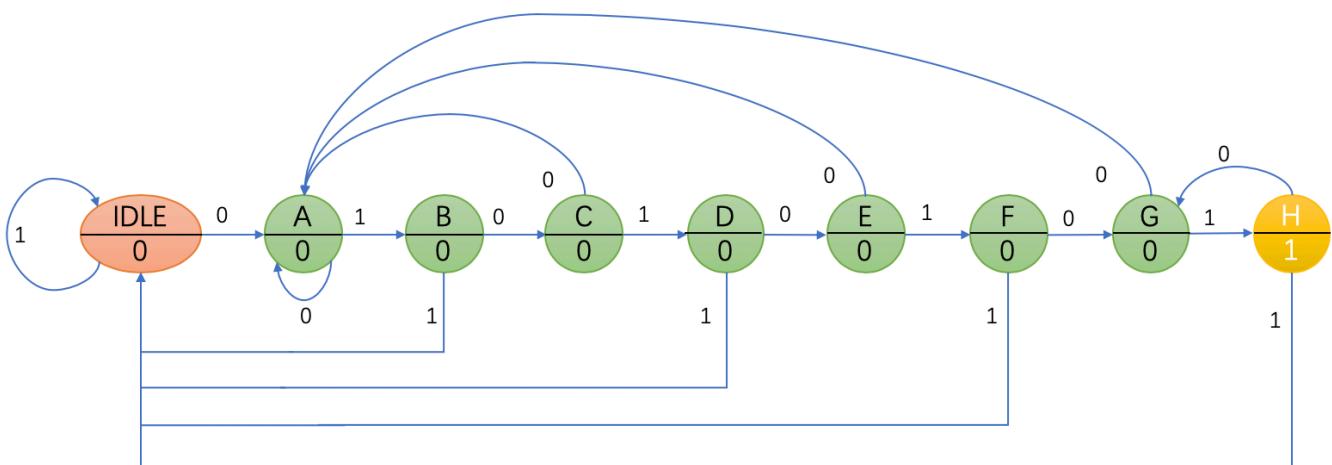


图 49: 序列检测器的状态转移图 (Moore 机)

```

1 //File: moore.v
2
3 'timescale 1 ns / 100 ps
4
5 module moore (
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     flag <= (state == H) ? 1'b1 : 1'b0;
30     case (state)
31       IDLE : state <= (din) ? IDLE : A;
32       A : state <= (din) ? B : A;
33       B : state <= (din) ? IDLE : C;
34       C : state <= (din) ? D : A;
35       D : state <= (din) ? IDLE : E;
36       E : state <= (din) ? F : A;
37       F : state <= (din) ? IDLE : G;
38       G : state <= (din) ? H : A;
39       H : state <= (din) ? IDLE : G;
40
41     default : state <= IDLE;
42   endcase
43 end
44 end
45
46 endmodule

```

## (2) 测试模块

```

1 //File: top.v
2
3 `timescale 1 ns / 100 ps
4 `include "mealy.v"
5 `include "moore.v"
6
7 module top;
8
9   wire p_mlf;
10  wire p_mrf;
11
12  reg p_clk, p_rst, p_din;
13
14  mealy meal(
15    .flag (p_mlf),

```

```

16      .din  (p_din) ,
17      .clk   (p_clk) ,
18      .rst(p_rst)
19  );
20
21 moore moor(
22     .flag (p_mrfflag) ,
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'
48     b0011_1011_0101_0101_0110_1010_1000_1101_1001_0101_0101_1010_1010_1010_1010;
49
50 initial begin : SERIES
51     integer i;
52     p_din = 1'b0;
53     #30;
54     for (i = 0; i < SIZE; i++) begin
55         p_din = data[0];
56         data = data >> 1;
57         #20;
58     end
59
60     $stop;
61 end
62
63 initial begin
64     $display("At time %tns, reset =%b, datain =%b, mealy_flag =%b, moore_flag =%b.", $time, p_rst,
65     p_din, p_mlflag, p_mrfflag);
66     forever begin
67         #20 $display("At time %tns, reset =%b, datain =%b, mealy_flag =%b, moore_flag =%b.", $time,
68         p_rst, p_din, p_mlflag, p_mrfflag);
69     end
70 end
71
72 endmodule

```

### (3) 测试波形图

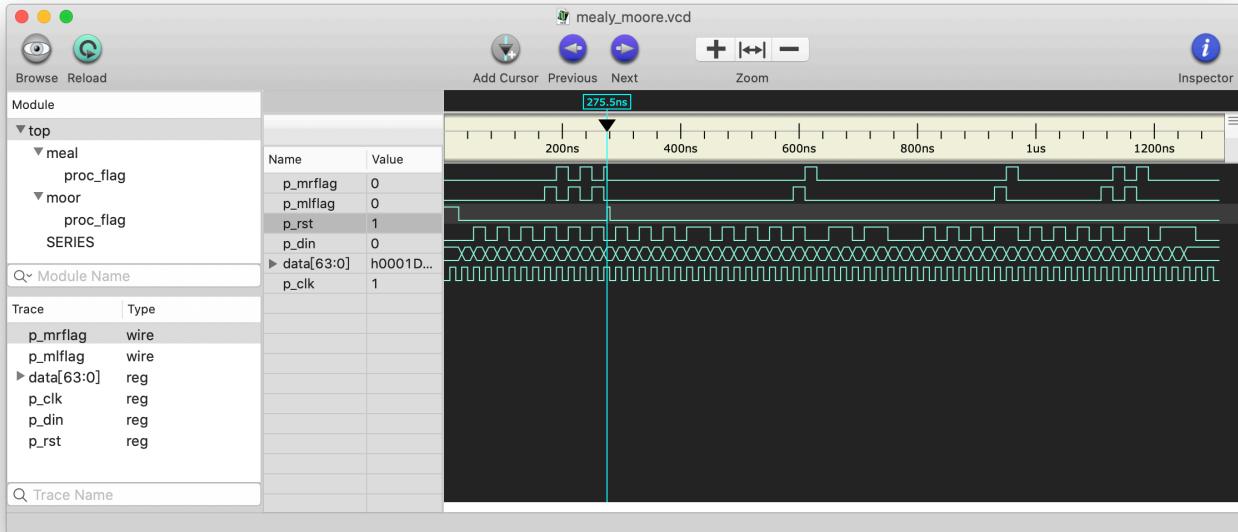


图 50: 测试波形图 2.16

### (4) 显示输出

```
project2_G -- vvp ./top.vpp -- 89x27
(base): liuhawendeMacBook-Pro:project2_G hongxing$ ./top.vpp
VCD info: dumpfile mealy_moore.vcd opened for output.
At time      0ns, reset = 1, datain = 0, mealy_flag = x, moore_flag = x.
At time     200ns, reset = 1, datain = 0, mealy_flag = 0, moore_flag = 0.
At time     400ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time     600ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time     800ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time    1000ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time    1200ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time    1400ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time    1600ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time    1800ns, reset = 0, datain = 1, mealy_flag = 1, moore_flag = 0.
At time    2000ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 1.
At time    2200ns, reset = 0, datain = 1, mealy_flag = 1, moore_flag = 0.
At time    2400ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 1.
At time    2600ns, reset = 0, datain = 1, mealy_flag = 1, moore_flag = 0.
At time    2800ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time    3000ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time    3200ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time    3400ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time    3600ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time    3800ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time    4000ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time    4200ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time    4400ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time    4600ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time    4800ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
```

图 51: 显示输出 2.16 (1)

```
project2_G -- vvp ./top.vpp -- 89x27
At time      4800ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time      5000ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      5200ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time      5400ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      5600ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time      5800ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      6000ns, reset = 0, datain = 1, mealy_flag = 1, moore_flag = 0.
At time      6200ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 1.
At time      6400ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      6600ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time      6800ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time      7000ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      7200ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time      7400ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time      7600ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      7800ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      8000ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      8200ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      8400ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time      8600ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      8800ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      9000ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
At time      9200ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time      9400ns, reset = 0, datain = 1, mealy_flag = 1, moore_flag = 0.
At time      9600ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 1.
At time      9800ns, reset = 0, datain = 0, mealy_flag = 0, moore_flag = 0.
At time     10000ns, reset = 0, datain = 1, mealy_flag = 0, moore_flag = 0.
```

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) (v10\_3)*

```

hongxing — bash — 89x20
(base) liuhawendeMacBook-Pro:~ hongxing$ iverilog -v
[Icarus Verilog version 10.3 (stable) (v10_3)
]

Copyright 1998-2015 Stephen Williams

This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License along
with this program; if not, write to the Free Software Foundation, Inc.,
51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.

iverilog: no source files.

```

图 54: 开源 RTL 仿真器 *Icarus Verilog*

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

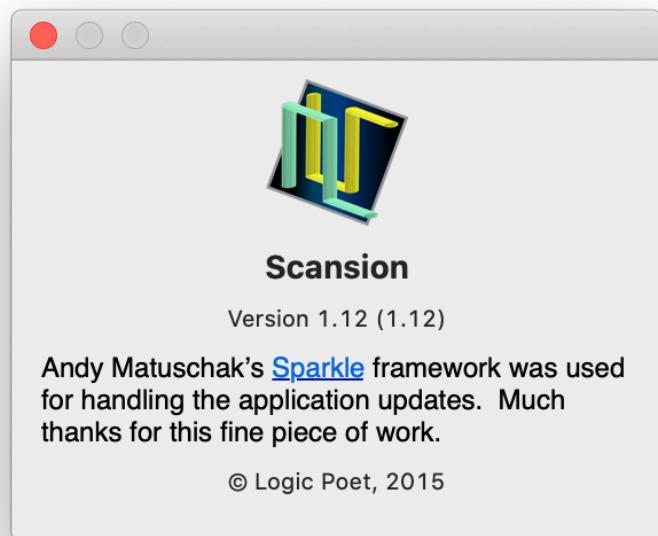


图 55: *vcd* 查看器 *Scansion*