数字系统设计作业

2019 年秋季学期

第二部分、*Verilog 设计*

2.1、设计一个 Verilog 模块，产生图 1 所示的波形。

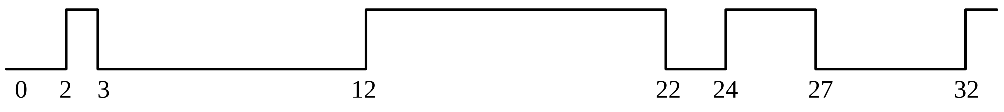


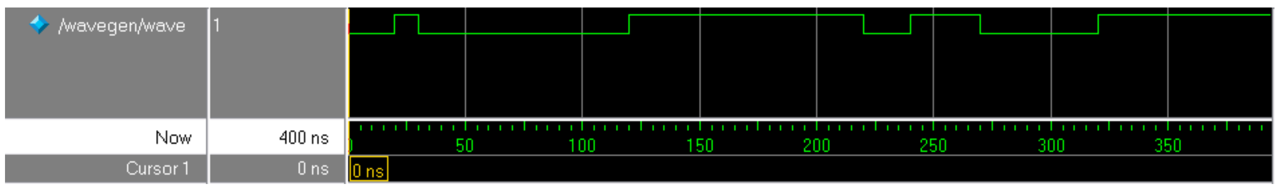
图 1、习题 1 波形图

要求：

（1）时间单位为：10ns；时间精度为：1ns；（2）使用 initial 语句；模块名为：**wavegen()**

模块设计：

```
`timescale 10ns/1ns
module wavegen;
    reg wave;
    initial begin
        wave = 1'b0;

        #2  wave = 1'b1;
        #1  wave = 1'b0;
        #9  wave = 1'b1;
        #10 wave = 1'b0;
        #2  wave = 1'b1;
        #3  wave = 1'b0;
        #5  wave = 1'b1;

        #10 $stop;
    end
endmodule
```



2.2、8:3 编码器的真值表如表 1 所示，使用 always－case 结构，设计一个 8:3 编码器。并设计一个测试平台，对电路进行仿真。

表 1、8:3 编码器真值表

输入 data[7:0]	输出 code[2:0]
0000_0001	0
0000_0010	1
0000_0100	2
0000_1000	3
0001_0000	4
0010_0000	5
0100_0000	6
1000_0000	7

要求：

（1）在模块设计中要求明确标明基数格式。;

（2）设计模块名为：

  Encoder8x3(code, data)

  测试平台的模块名为：

  tb_Encoder8x3（）

模块设计：

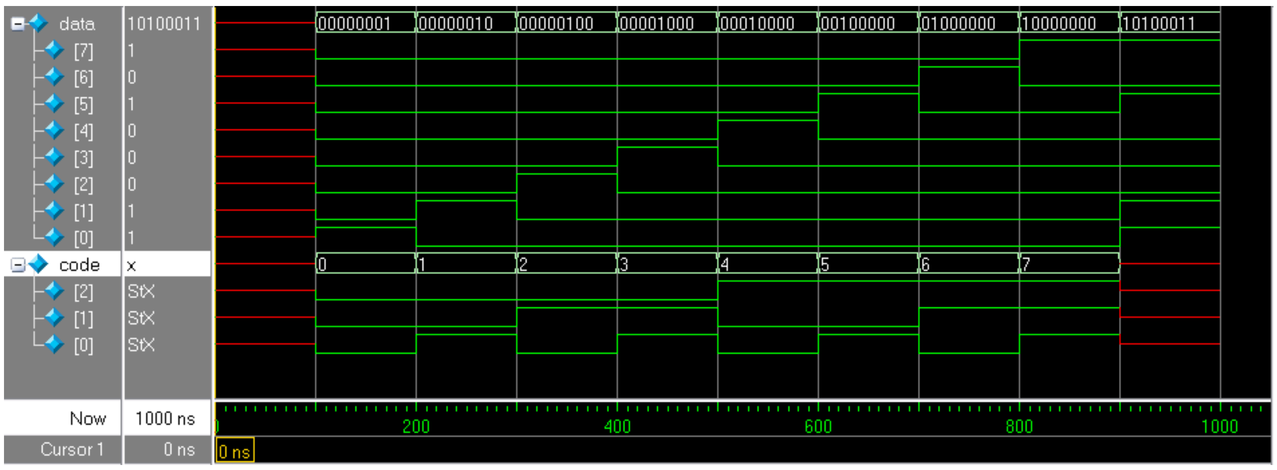
（1）

```
// File: encoder8x3.v
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'bx;
        endcase
    end
endmodule
```

（2）

```
// File: tb_encoder8x3.v
`timescale 1ns/1ns
`include "encoder8x3.v"
module tb_encoder8x3();
    // 测试激励信号
    reg [7:0] stim1, stim2;
    // 连接输入端口
    reg [7:0] p_data;
    // 连接输出端口
    wire [2:0] p_code;
    integer k;

    encoder8x3 m_u( .code(p_code), .data(p_data) );
    // 初始化激励信号
```

```
   initial  begin
       stim1 = 8'b0000_0001;
       stim2 = 8'b1010_0011;
   end

   initial // 产生激励信号
   begin
       for ( k = 0; k < 8; k = k+1 )
          #10 p_data = stim1 << k;
       #10 p_data = stim2;
       #10 $stop;
   end
   // 监测仿真输出
   initial begin
       $monitor("At time: t= %4t(ns), <--> data= %8b, code= %3b",
                 $time, p_data, p_code );
   end
endmodule
```



2.3、a）如图 2 所示，使用 bufif0 和 bufif1 设计一个二选一多路选择器，并给出测试激励模块，和仿真测试结果。

要求：

设计模块名为：

mux2x1(dout, sel, din)

测试平台的模块名为：

tb_ mux2x1 ()

b）根据多路选择器的原理，使用连续赋值语句设计一个 4 选一多路选择器，并给出仿真测试结果。

要求：

设计模块名为：

mux4x1(dout, sel, din)

测试平台的模块名为：

tb_ mux4x1 ()

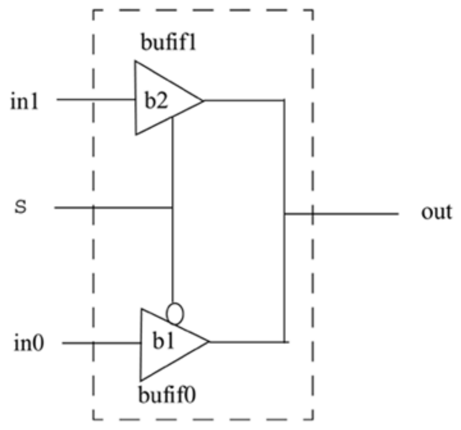


图 2、二选一多路选择器

a）

```
// File: mux2x1.v
module mux2x1( dout, sel, din );
    output dout;
    input sel;
    input [1:0] din;

    bufif1 b2( dout, din[1], sel );
    bufif0 b1( dout, din[0], sel );
endmodule

// File: tb_mux2x1.v
`timescale  10ns / 1ns
`include "mux2x1.v"
module tb_mux2x1();
    reg [2:0] p_data;
    wire p_dout;
    integer i;
    initial  begin
        p_data = 3'bx;
        #5 p_data = 3'b0;
        for ( i = 1; i < 8; i = i + 1 )
             #5 p_data = p_data + 3'b1;
    end
    mux2x1 m0( .dout( p_dout ), .sel(p_data[2]), .din(p_data[1:0]) );
    initial
        $monitor( "At time %t, dout=%1b, sel=%1b, din=%2b",
                 $time, p_dout, p_data[2], p_data[1:0]);
endmodule
```

b）
```
// File: mux4x1.v
module mux4x1( output dout, input [1:0] sel, input [3:0] din );
   assign dout = (sel[1]) ? ( sel[0] ? din[3] : din[2]) : ( sel[0] ? din[1] :
din[0]);
endmodule

// File: tb_mux4x1.v
`timescale 1ns/1ns
`include "mux4x1.v"
module tb_mux4x1;
   parameter STEP = 20;
   wire pDout;
   reg [1:0] pSel;
   reg [3:0] pDin;
   integer k, m;

   mux4x1 uut( .dout(pDout), .sel(pSel), .din(pDin) );

   initial begin
      pSel = 2'b00;
      for ( k = 0; k < STEP; k=k+1 ) begin
         #1;
         pSel = pSel + 1'b1;
      end
   end

   initial begin
      pDin = 4'b0;
      for ( m = 0; m < STEP; m=m+1 ) begin
         #1;
         pDin = pDin + 1'b1;
      end
   end

   initial
      $monitor("At time %8t, dout=%b, sel=%b, din=%4b",
                 $time, pDout, pSel, pDin );
endmodule
```

2.4、设计一个 Verilog 模块，描述如图 3 所示的电路原理图表示的电路。并设计一个测试平台，对电路进行仿真。

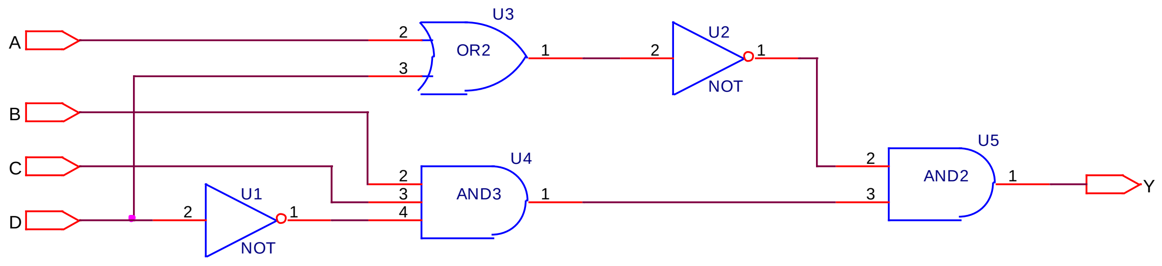


图 3、习题 4 电路原理图

要求：

（1）利用 Verilog 的基本门（门级原语），采用结构（Structural）描述方式设计；此时，模块名为：
comb_str(Y, A, B, C, D)

（2）利用连续幅值语句，采用数据流（Dataflow）方式设计；此时，模块名为：
comb_dataflow(Y, A, B, C, D)

（3）利用 always 过程语句，采用行为和算法（Behvioral or algorithmic）方式设计；此时，模块名为：
comb_behavior(Y, A, B, C, D)

（4）利用用户定义原语（UDP），使用真值表描述方式设计；此时，模块名为：
comb_prim(Y, A, B, C, D)

（5）测试平台的模块名为：**testbench_comb**（）；注意，测试平台模块没有端口。

（6）对电路进行全面仿真，提供仿真波形以证明设计的正确性；并使用系统任务 $display 和 $monitor 监控仿真结果，在 ModelSim 的 ***Transcript Window*** 中输出文本表示的仿真结果。

模块设计：

（1）利用 Verilog 的基本门（门级原语）

```
// File: comb_str.v
module comb_str( output y, input a, b, c, d);
   wire s1, s2, s3, s4, s5;

   not u1( s3, d );
   not u2( s2, s1 );
   or  u3( s1, a, d );
   and u4( s4, b, c, s3 );
   and u5( y, s2, s4 );
endmodule
```

```verilog
// File: comb_tb.v
`timescale  10ns / 1ns
`include "comb_str.v"
`define     STOP    16
module testbench_comb();
   reg [3:0] data;
   wire p_y;
   integer i;
   initial begin
      data = 4'b0;
      for ( i = 1; i < `STOP; i = i + 1 ) begin
         #5  data = data + 1'b1;
      end
      #10 $stop;
   end

   comb_str m0( .y(p_y), .a(data[3]), .b(data[2]), .c(data[1]), .d(data[0]) );

   initial begin
      $display( "a b c d    :  y\n");
      $monitor( "%1b %1b %1b %1b    :  %1b",
                 data[3], data[2], data[1], data[0], p_y);
   end
endmodule
```

（2）利用连续幅值语句

```verilog
// File: comb_dataflow.v
module comb_dataflow( output y, input a, b, c, d);
   wire s1, s2, s3, s4, s5;

   assign s3 = ~d,
          s4 = b & c & s3,
          s1 = a | d,
          s2 = ~s1,
          y  = s2 & s4;
endmodule
```

（3）利用 always 过程语句

```verilog
// File: comb_behavior.v
module comb_behavior( output reg y, input a, b, c, d);
   always @(*) y = ( ~( a | d) ) & ( ( b & c ) & ~d);
endmodule
```

（4）利用用户定义原语（UDP）

```
// File: comb_prim.v
primitive comb_prim( y, a, b, c, d);
   output y;
   input a, b, c, d;
   // 状态表定义
   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
```

2.5、采用数据流（Dataflow）方式设计，利用连续幅值语句，设计两个组合逻辑电路模块，描述下面的布尔方程，并设计测试平台进行仿真验证。

i） $Y1(A,B,C)=\Sigma m(1,2,4,5)$

ii） $Y2(A,B,C,D)=\Sigma m(4,5,6,7,11,12,13)$

要求：

（1）两个设计模块名为：

comb_Y1(Y, A, B, C)、**comb_Y2(Y, A, B, C, D)**

测试平台的模块名分别为：

tb_comb_Y1()，**tb_comb_Y2()**

（2）对电路进行全面仿真，提供仿真波形以证明设计的正确性；并使用系统任务$monitor 监控仿真结果。

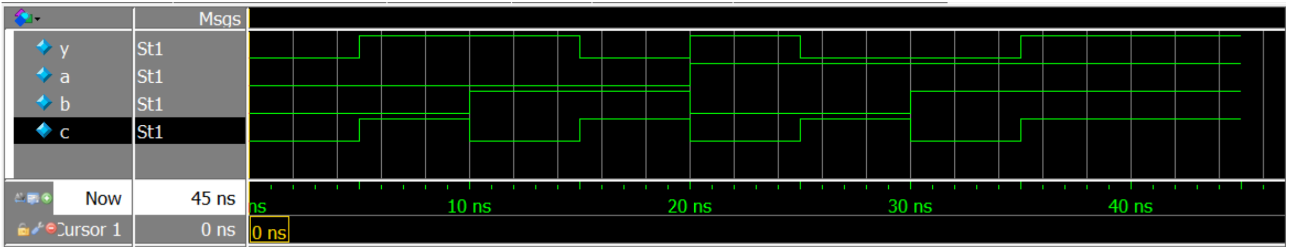
解答：（1） comb_Y1 模块：

```
module comb_y1( output y, input a, b, c);

    assign  y = (~a & ~b & c ) | ( ~a & b & ~c) | ( a & ~b & ~c) | ( a & b & c);

endmodule
```

（2） tb_comb_Y1 模块：

```
`timescale  1ns / 1ns

`include "comb_y1.v"

`define  STOP 8
`define  DELAY 5

module tb_comb_y1();

    reg [2:0] px;

    wire py;

    integer i;

    initial
    begin
        px = 3'b0;

        for ( i = 1; i < `STOP; i = i+1 )
            #`DELAY px = px + 1;
    end

    comb_y1 m1( .y(py), .a(px[2]), .b(px[1]), .c(px[0]) );

    initial
        $monitor("At time %4t, <----> y=%1b, a=%1b, b=%1b, c=%1b",
                  $time, py, px[2], px[1], px[0]);
endmodule
```

（3）tb_comb_Y1 仿真波形和输出



```
# At time    0, <----> y=0, a=0, b=0, c=0
# At time    5, <----> y=1, a=0, b=0, c=1
# At time   10, <----> y=1, a=0, b=1, c=0
# At time   15, <----> y=0, a=0, b=1, c=1
# At time   20, <----> y=1, a=1, b=0, c=0
# At time   25, <----> y=0, a=1, b=0, c=1
# At time   30, <----> y=0, a=1, b=1, c=0
# At time   35, <----> y=1, a=1, b=1, c=1
```

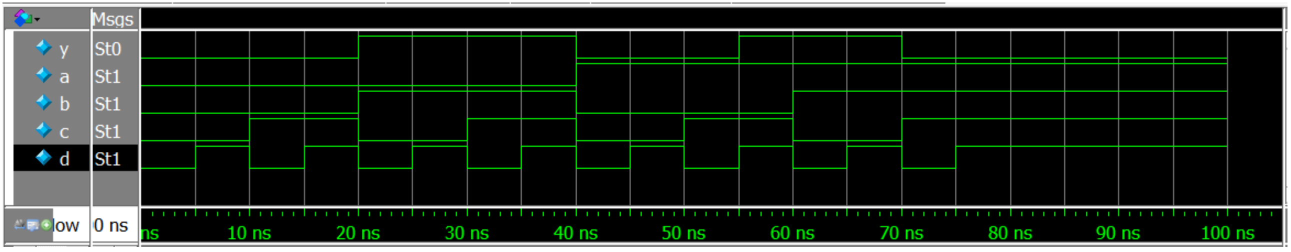
（4）comb_Y2 模块：

```
module comb_y2( output y, input a, b, c, d );

    wire  p4, p5, p6, p7, p11, p12, p13;

    assign p4  = ~a &  b & ~c & ~d,
           p5  = ~a &  b & ~c &  d,
           p6  = ~a &  b &  c & ~d,
           p7  = ~a &  b &  c &  d,
           p11 =  a & ~b &  c &  d,
           p12 =  a &  b & ~c & ~d,
           p13 =  a &  b & ~c &  d,
           y   =  p4 | p5 | p6 | p7 | p11 | p12 | p13;
endmodule
```

（5）tb_comb_Y2 模块：

```
`timescale  1ns / 1ns

`include "comb_y2.v"

`define  STOP 16
`define  DELAY 5

module tb_comb_y2();

    reg [3:0] px;

    wire py;

    integer i;

    initial
    begin
        px = 4'b0;

        for ( i = 1; i < `STOP; i = i+1 )
            #`DELAY px = px + 1;
    end

    comb_y2 m2( .y(py), .a(px[3]), .b(px[2]), .c(px[1]), .d(px[0]) );

    initial
        $monitor("At time %4t, <----> y=%1b, a=%1b, b=%1b, c=%1b, d=%1b",
                  $time, py, px[3], px[2], px[1], px[0]);
endmodule
```

（6）tb_comb_Y2 模块仿真波形：



2.6、设计一个 Verilog 模块，使用 4 位输出码表示 8 位输入字中 1 的个数。并设计测试平台进行仿真验证。

要求：

（1）设计模块名为：

ones_count(count, dat_in)

这里，count 是 4 位输出端口，dat_in 为 8 位输入端口；

测试平台的模块名为：

tb_ones_count()

（2）对电路进行全面仿真，提供仿真波形以证明设计的正确性；并使用系统任务$monitor 监控仿真结果。

解答：（1）

```
// File: one_count.v
module one_count( output [3:0] count, input [7:0] dat_in);
   assign count =  dat_in[7] + dat_in[6] + dat_in[5] + dat_in[4]
                 + dat_in[3] + dat_in[2] + dat_in[1] + dat_in[0];
endmodule
```

（2）

```
// File: tb_one_count.v
`timescale  1ns / 1ns
`include "one_count.v"
`define     STOP    256
`define     DELAY   5
module tb_one_count();
   reg [7:0] p_din;
   wire [3:0] p_cnt;
   integer i;

   initial  begin
      p_din = 8'b0;
      for ( i = 1; i < `STOP; i = i+1 )
         #`DELAY p_din = p_din + 1;
      #`DELAY $stop;
   end

   one_count mu( .count(p_cnt), .dat_in(p_din) );

   initial
      $monitor("At time %6t, <----> dat_in=%8b, count=%4b",
                  $time, p_din, p_cnt);
endmodule
```

2.7、设计一个 10 进制计数器的 Verilog 模块。

要求：

（1）计算器从 0 到 10 计数，然后返回到 0 重新开始计数；采用同步复位；

（2）设计测试模块对其进行仿真；

（3）设计模块名为：

   dec_counter(count, clk, reset)

   测试平台的模块名为：

   tb_dec_counter()

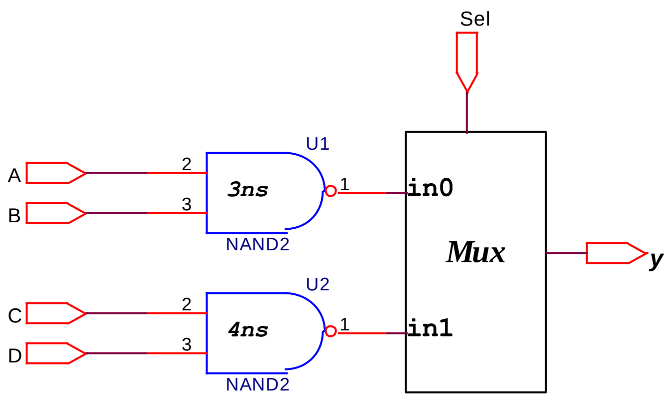
```
// File: dec_counter.v
module dec_counter(output reg [3:0] count, input clk, reset);
   always @( posedge clk )
   begin
      if ( reset == 1'b0 )
         count <= 4'b0;
      else
      begin
         if ( count < 4'b1001 )
```

```
                count <= count + 1'b1;
            else
                count <= 4'b0;
        end
    end
endmodule
```

（2）

```
// File: tb_dec_counter.v
`timescale  1ns / 1ns
`include "dec_counter.v"
`define     PULSE   5
module tb_dec_counter();
    wire [3:0] p_cnt;
    reg p_clk;
    reg p_rst;
    initial begin
        p_clk = 0;
        forever #`PULSE p_clk = ~p_clk;
    end
    initial begin
        #5 p_rst = 1'b0;
        #10 p_rst = 1'b1;
        #60 p_rst = 1'b0;
        #15 p_rst = 1'b1;
    end

    dec_counter m0( .count(p_cnt), .clk(p_clk), .reset(p_rst) );

    initial
        $monitor("At time %t, <----> count=%b", $time, p_cnt );

endmodule
```

2.8、采用结构（Structural）描述方式设计一个 Verilog 模块，描述图 4 所示电路原理图表示的电路。



Mux 多路复用器真值表

sel	y
0	in0
1	in1

图 4、习题 8 电路原理图

要求：

（1）设计模块名为：

comb_str(y, sel, A, B, C, D)

测试平台的模块名为：

tb_comb_str()

（2）对电路进行全面仿真，提供仿真波形以证明设计的正确性；并使用系统任务$monitor 监控仿真结果。

解答：（1）

```verilog
// File: mux2x1.v
module mux2x1( y, sel, in1, in0 );
    output y;
    input sel;
    input in0, in1;
    assign y = ( sel ) ? in1 : in0;
endmodule
```

（2）

```verilog
// File: comb_str.v
`include "mux2x1.v"
module comb_str( y, sel, a, b, c, d);
    output y;
    input sel;
    input a, b, c, d;
    wire s0, s1;
    nand #3 u1( s0, a, b);
    nand #4 u2( s1, c, d);

    mux2x1 m0( .y(y), .sel(sel), .in1(s1), .in0(s0) );
endmodule
```

（3）

```verilog
// File: tb_comb_str.v
```

```
`timescale  1ns / 1ns
`include "comb_str.v"
`define     STOP    32
module tb_comb_str();
   reg [4:0] px;
   wire py;
   integer i;

   initial begin
      px = 5'b0;
      for ( i = 1; i < `STOP; i = i + 1 ) begin
         #5  px = px + 1'b1;
      end
   end
   comb_str m1(.y(py),.sel(px[4]),.a(px[3]),.b(px[2]),.c(px[1]),.d(px[0]));

   initial
      $monitor( "At time %t, y=%1b, sel=%1b, a=%1b, b=%1b, c=%1b, d=%1b",
                $time, py, px[4], px[3], px[2], px[1], px[0] );
endmodule
```

2.9、根据下面本源多项式公式，设计一个线性反馈移位寄存器，要求使用内部反馈方式设计。

$$P(x) = x^{26} + x^{8} + x^{7} + x + 1$$

这里：

输出：26 位伪随机数 q

输入：clk —— 时钟信号

rst_n —— 同步复位信号，低电平有效

load —— 同步加载控制，当 load = 1’b1，

din —— 26 位输入，不全为 0 时，din ➔ q

要求：

（1）设计模块名为：

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

测试平台的模块名为：

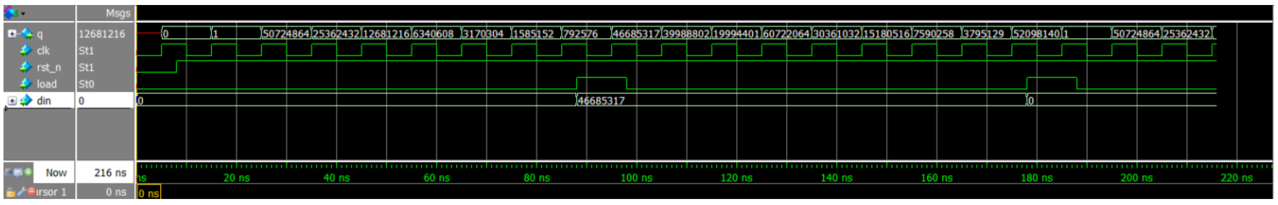
tb_LFSR()

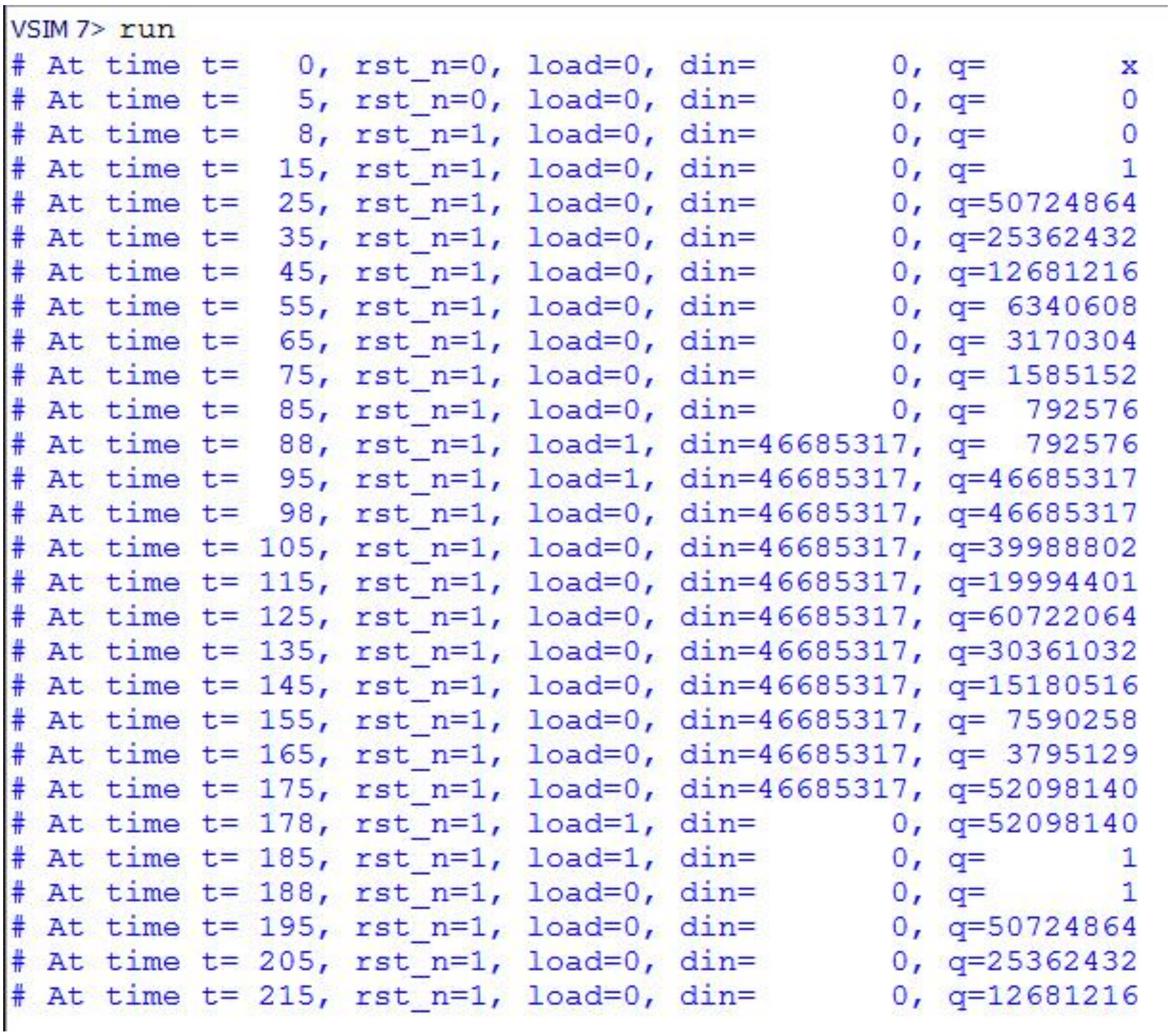
（2）并给出测试模块和测试分析结果。

(1)
```
// File: LFSR.v
module LFSR( output reg [1:26] q,   // 26 bit data output.
             input clk,             // Clock input.
             input rst_n,           // Synchronous reset input.
             input load,            // Synchronous load input.
             input [1:26] din       // 12 bit parallel data input.
            );

   always @( posedge clk ) begin
      if ( ~rst_n )
         q <= 26'b0;
      else begin
         if (load)
            q <= (|din) ? din : 26'b01;
         else begin
            if ( q == 26'b0 )
               q <= 26'b01;
            else begin
               q[10:26] <= q[ 9: 25];
               q[ 9] <= q[ 8] ^ q[26]; //  X9 <-- X8^26
               q[ 8] <= q[ 7] ^ q[26]; //  X8 <-- X7^X26
               q[ 3: 7] <= q[ 2: 6];
               q[ 2] <= q[ 1] ^ q[26]; //  X2 <-- X1^X26
               q[ 1] <= q[26];         //  X1 <-- X0(X26)
            end
         end
      end
   end
endmodule
```

(2)
```
`timescale 1ns/1ns
`include "LFSR.v"
module LFSR_tb();
   wire [1:26] p_q;
   reg [1:26] p_din;
   reg p_clk, p_rst_n, p_load;

   LFSR u0(.q(p_q), .clk(p_clk), .rst_n(p_rst_n), .load(p_load), .din(p_din));

   initial begin
      p_clk = 0;
      forever #5 p_clk = ~p_clk;
   end
```

```
    initial begin
       p_rst_n = 0;
       #8 p_rst_n = 1;
    end

    initial begin
       p_load = 0;
       p_din = 26'd0;

       #88 p_load = 1'b1;
       p_din = 26'd4668_5317;  // 26'b10_1100_1000_0101_1100_1000_0101
       #10 p_load = 1'b0;

       #80 p_load = 1'b1;
       p_din = 26'd0;
       #10 p_load = 1'b0;
    end

    initial begin
       $monitor("At time t=%4t, rst_n=%b, load=%b, din=%d, q=%d",
                $time, p_rst_n, p_load, p_din, p_q);
    end
endmodule
```





```
VSIM 7> run
# At time t=   0, rst_n=0, load=0, din=       0, q=       x
# At time t=   5, rst_n=0, load=0, din=       0, q=       0
# At time t=   8, rst_n=1, load=0, din=       0, q=       0
# At time t=  15, rst_n=1, load=0, din=       0, q=       1
# At time t=  25, rst_n=1, load=0, din=       0, q=50724864
# At time t=  35, rst_n=1, load=0, din=       0, q=25362432
# At time t=  45, rst_n=1, load=0, din=       0, q=12681216
# At time t=  55, rst_n=1, load=0, din=       0, q= 6340608
# At time t=  65, rst_n=1, load=0, din=       0, q= 3170304
# At time t=  75, rst_n=1, load=0, din=       0, q= 1585152
# At time t=  85, rst_n=1, load=0, din=       0, q=  792576
# At time t=  88, rst_n=1, load=1, din=46685317, q=  792576
# At time t=  95, rst_n=1, load=1, din=46685317, q=46685317
# At time t=  98, rst_n=1, load=0, din=46685317, q=46685317
# At time t= 105, rst_n=1, load=0, din=46685317, q=39988802
# At time t= 115, rst_n=1, load=0, din=46685317, q=19994401
# At time t= 125, rst_n=1, load=0, din=46685317, q=60722064
# At time t= 135, rst_n=1, load=0, din=46685317, q=30361032
# At time t= 145, rst_n=1, load=0, din=46685317, q=15180516
# At time t= 155, rst_n=1, load=0, din=46685317, q= 7590258
# At time t= 165, rst_n=1, load=0, din=46685317, q= 3795129
# At time t= 175, rst_n=1, load=0, din=46685317, q=52098140
# At time t= 178, rst_n=1, load=1, din=       0, q=52098140
# At time t= 185, rst_n=1, load=1, din=       0, q=       1
# At time t= 188, rst_n=1, load=0, din=       0, q=       1
# At time t= 195, rst_n=1, load=0, din=       0, q=50724864
# At time t= 205, rst_n=1, load=0, din=       0, q=25362432
# At time t= 215, rst_n=1, load=0, din=       0, q=12681216
```

2.10、设计一个 Verilog 模块，描述如图 5 所示的电路原理图表示的电路。并设计一个测试平台，对电路进行仿真。

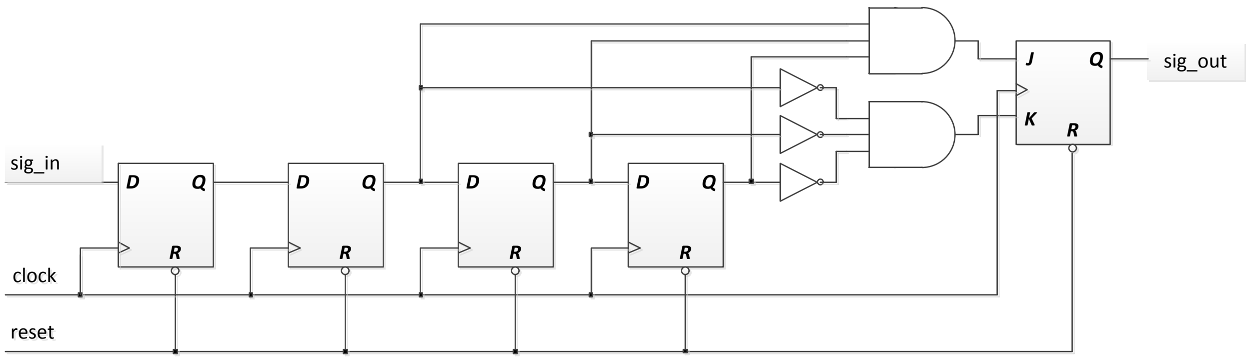


图 5、习题 2.10 电路原理图

要求：

（1）设计模块名为：

    filter(sig_out, clock, reset, sig_in)

测试平台的模块名为：

    tb_filter()

（2）并给出测试模块和测试分析结果。

```
module filter(sig_out, clock, reset, sig_in);
   output sig_out;
   input clock, reset, sig_in;

   reg [3:0] data;
   reg dout;

   always @(posedge clock) begin
      if (!reset) begin
         data <= 4'b0;
         dout <= 1'b0;
      end
      else begin
         data <= {data[2:0], sig_in};

         case ( { &data[3:1], &(~data[3:1])} )
            2'b01:   dout <= 1'b0;
            2'b10:   dout <= 1'b1;
            2'b11:   dout <= ~dout;
            default: ;
         endcase
      end
   end
   assign sig_out = dout;
endmodule
```

(2)

```
`timescale 1ns/1ns
`include "filter.v"
module filter_tb;
   wire p_o;
   reg  p_clk, p_rst, p_i;

   filter u0(.sig_out(p_o), .clock(p_clk), .reset(p_rst), .sig_in(p_i));

   initial begin
      p_clk = 0;
      forever #5 p_clk = ~p_clk;
   end

   initial begin
      p_rst = 1'b0;
      #23 p_rst = 1'b1;
   end

   integer k;
   reg [31:0] din;
   initial begin
      din = 32'b1101_0010_0000_0111_1101_1111_0000_0001;
      #10;
      for (k=0; k<32; k=k+1)
         #10 p_i = din[k];
   end

   initial
      $monitor( "At time %t, reset=%b, sig_in=%b, sig_out=%b",
                  $time, p_rst, p_i, p_o);
endmodule
```

2.11、设计一个能够递增和递减的 8 位双向循环计数器，计数器的示意图如图 6 所示。

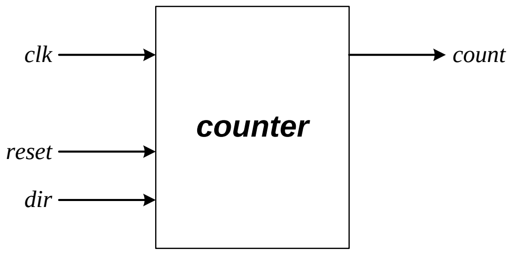


图 6、双向循环计数器

要求：
（1）采用异步复位，复位后从第一个有效时钟的上跳沿开始计数；如果此时 **dir=1** ，则递增计数，否则，递减计数。
（2）输出 count 为 8 位；
（3）对电路进行全面仿真。
（4）设计模块名为：
  counter8b_updown(count, clk, reset, dir)
  测试平台的模块名为：
  tb_counter8b_updown()

```
// File: counter8b_updown.v
module counter8b_updown( count, clk, reset, dir );
   output [7:0] count;
   input clk, reset, dir;
   reg [7:0] count0, count1;
   always @( posedge clk or negedge reset ) begin
      if ( reset == 1'b0 ) begin
         count0 <= 8'b0;
         count1 <= 8'b1111_1111;
      end
      else
         if ( dir )
            count0 <= count0 + 1;
         else
            count1 <= count1 - 1;
   end
   assign count = (dir) ? count0 : count1;
endmodule
// File: tb_counter8b_updown.v
`timescale 1ns / 1ns
`include "counter8b_updown.v"
`define    PULSE   5
module tb_counter8b_updown();
   wire [7:0] p_cnt;
   reg p_clk, p_rst, p_dir;
```

```
    initial begin
        p_clk = 0;
        forever #`PULSE p_clk = ~p_clk;
    end

    initial begin
        #5 p_rst = 1'b0;
        #2 p_dir = 1'b1;
        #10 p_rst = 1'b1;
        #80 p_rst = 1'b0;
        #2 p_dir = 1'b0;
        #10 p_rst = 1'b1;
        #100 p_dir = 1'b1;
    end

    counter8b_updown m0( .count(p_cnt), .clk(p_clk), .reset(p_rst), .dir(p_dir) );

    initial
        $monitor("At time %t, <----> count=%b, dir=%b", $time, p_cnt, p_dir );
endmodule
```

2.12、设计一个 8 位算术逻辑单元（ALU)，该单元的输入为操作数 a 和 b，以及操作码 oper， 输出为 { c_out , sum }，并且具有如下表所示的功能。

操作码	功能
add	a + b + c_in
subtract	a + ~b + c_in
subtract_a	b + ~a + ~c_in
or_ab	{ 1'b0, a | b }
and_ab	{ 1'b0, a & b }
not_ab	{ 1'b0, (~a) & b }
exor	{ 1'b0, a ^ b }
exnor	{ 1'b0, a ~^ b }

要求：

（1）使用 always-case 结构设计 ALU 模块，并设计测试模块对其进行仿真验证；

（2）设计模块名为：

ALU(c_out, sum, oper, a, b, c_in)

测试平台的模块名为：

tb_ALU()

```
// File: alu.v
module alu( c_out, sum, oper, a, b, c_in );
    output c_out;
    output [7:0] sum;
    reg [8:0] buf_data;
    input [2:0] oper;
    input [7:0] a;
    input [7:0] b;
    input c_in;

    parameter   op_add          = 3'b000,
                op_subtract     = 3'b001,
                op_subtract_a   = 3'b010,
                op_or_ab        = 3'b011,
                op_and_ab       = 3'b100,
                op_not_ab       = 3'b101,
                op_exor         = 3'b110,
                op_exnor        = 3'b111;
    always @(*)
    begin
        case ( oper )
            op_and:        buf_data =  a +  b +  c_in;
            op_subtract:    buf_data =  a + ~b +  c_in;
            op_subtract_a:  buf_data = ~a +  b + ~c_in;
            op_or_ab:       buf_data = {1'b0,   a  |   b };
            op_and_ab:      buf_data = {1'b0,   a  &   b };
            op_not_ab:      buf_data = {1'b0, (~a) |   b };
            op_exor:        buf_data = {1'b0,   a  ^   b };
            op_exnor:       buf_data = {1'b0,   a  ^ (~b)};
            default:        buf_data = 9'bx;
        endcase
    end
    assign {c_out, sum} = buf_data;
endmodule
```

2.13、设计一个具有图 7 所示功能的计数器模块。

要求：

（1）采用同步复位，复位后回到初始状态，然后从第一个有效时钟的上跳沿开始计数；

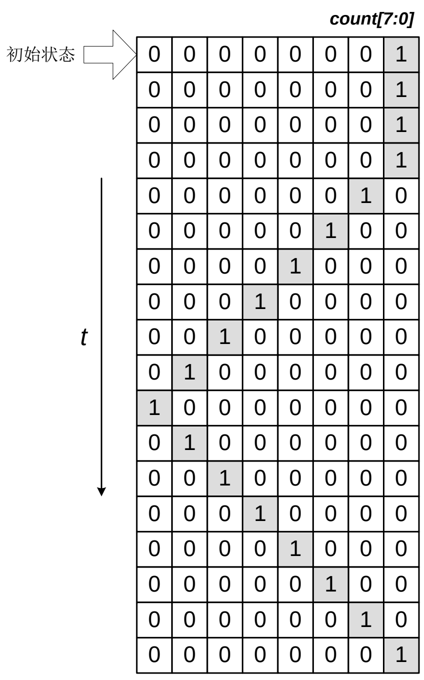
（2）设计测试模块对其进行仿真验证；

（2）设计模块名为：

```
shift_counter( count, clk, reset )
```

测试平台的模块名为：

```
tb_shift_counter()
```



0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0
0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	1	0	0	0
0	0	0	0	0	1	0	0
0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1

图 7、习题 13 计数器计数模式

```
// File: shift_counter.v
module shift_counter(count, clk, reset);
    output [7:0] count;
    input        clk;
    input        reset;

    parameter CNTSUP = 17;
    reg [4:0] cnt;

    always @( posedge clk ) begin
        if ( reset )
            cnt <= 5'b0;
        else begin
            if (cnt < CNTSUP)
                cnt <= cnt + 1'b1;
            else
                cnt <= 5'b0;
        end
    end
    function [7:0] val( input [4:0] c );
    begin
        case ( c)
            0:  val = 8'b0000_0001;
            1:  val = 8'b0000_0001;
            2:  val = 8'b0000_0001;
            3:  val = 8'b0000_0001;
            4:  val = 8'b0000_0010;
            5:  val = 8'b0000_0100;
            6:  val = 8'b0000_1000;
            7:  val = 8'b0001_0000;
            8:  val = 8'b0010_0000;
            9:  val = 8'b0100_0000;
            10: val = 8'b1000_0000;
            11: val = 8'b0100_0000;
            12: val = 8'b0010_0000;
            13: val = 8'b0001_0000;
            14: val = 8'b0000_1000;
            15: val = 8'b0000_0100;
            16: val = 8'b0000_0010;
            17: val = 8'b0000_0001;
            default:val = 8'b0000_0001;
         endcase
      end
      endfunction

      assign  count = val(cnt);
   endmodule

   `include "shift_counter.v"
   `define     PULSE   5
   module tb_shift_counter;
      wire [7:0] p_cnt;
      reg p_clk;
      reg p_rst;

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

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

      shift_counter u0(.count(p_cnt), .clk(p_clk), .reset(p_rst));

      initial
         $monitor( "At time %4t, reset=%b, count=%b", $time, p_rst, p_cnt );
   endmodule
```

2.14、图 8 是一个 256×8bits 的 SRAM 的框图，din[7:0]是 8 条数据输入线，dout[7:0] 是 8 条数据输出线。wr 为写控制线，rd 为读控制线，cs 为片选控制线。其工作方式为：

（1）当 cs = 1，wr 信号由低变高（上升沿）时，din 上的数据将写入由 addr 所指定的存储单元；

（2）当 cs = 1，rd = 0 时，由 addr 所指定的存储单元的内容将从 dout 的数据线上输出。

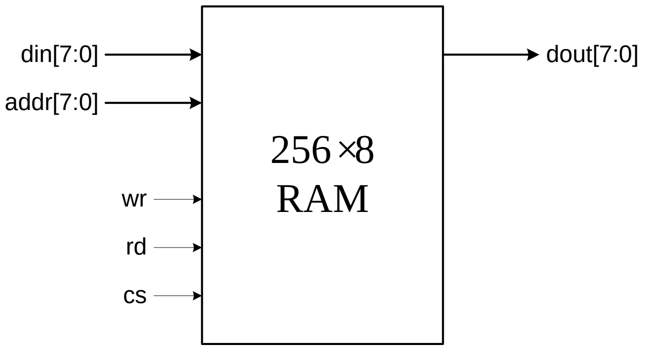


图 8、256×8bits 的 SRAM 框图

要求：

（1）设计一个 Verilog 模块描述一个 256×8bits 的 SRAM。

（2）设计测试模块对其进行仿真验证；

（3）设计模块名为：

sram(dout, din, addr, wr, rd, cs)

测试平台的模块名为：

tb_sram()

```
module sram( dout, din, addr, wr, rd, cs );
   parameter   BYTEWIDTH = 8,  ADDRWIDTH = 8;
   output  [BYTEWIDTH-1:0]  dout;
   input   [BYTEWIDTH-1:0]  din;
   input  [ADDRWIDTH-1:0]  addr;
   input  wr, rd, cs;

   reg  [BYTEWIDTH-1:0]  ram  [ADDRWIDTH-1:0];

   always @( posedge wr ) if ( cs ) ram[addr] <= din;
   assign dout = (  cs  &  !rd )  ? ram[addr]  :  8'bz;

endmodule
```

2.15、设计一个序列检测器，在时钟的每个下降沿检查数据。当检测到输入序列 din 中出现 1101 或 0110 时，输出 flag 为 1，否则输出为 0。

要求：

（1）画出序列检测器的状态转移图，根据状态转移图设计一个 Verilog 模块描述序列检测器。

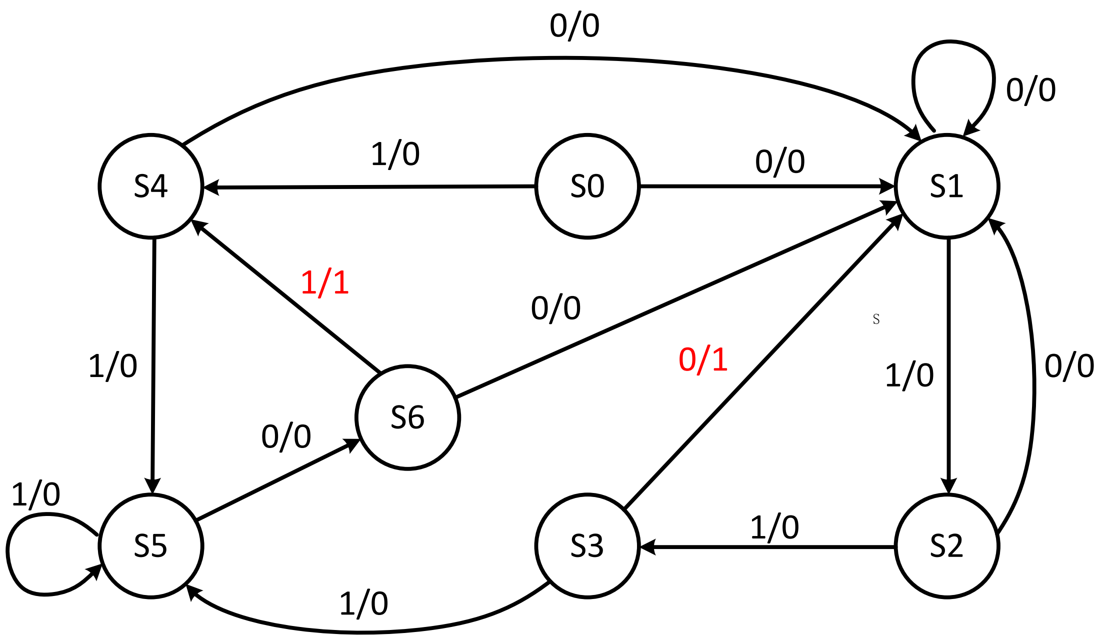
（2）设计测试模块对其进行仿真验证；

（3）设计模块名为：

```
seq_detect( flag, din, clk )
```

测试平台的模块名为：

```
tb_seq_detect()
```



```
// File: seq_detect.v
module seq_detect( flag, din, clk, reset );

   // Internal Declarations
   output reg flag;
   input  din, clk, reset;

   // State encoding
   parameter   s0 = 7'b000_0001,
              s1 = 7'b000_0010,
              s2 = 7'b000_0100,
              s3 = 7'b000_1000,
              s4 = 7'b001_0000,
              s5 = 7'b010_0000,
              s6 = 7'b100_0000;
```

```
    reg [6:0] current_state, next_state;

    always @( * )
    begin : next_state_block_proc
       case (current_state)
          s0:    next_state = (din) ? s4 : s1;
          s1:    next_state = (din) ? s2 : s1;
          s2:    next_state = (din) ? s3 : s1;
          s3:    next_state = (din) ? s5 : s1;
          s4:    next_state = (din) ? s5 : s1;
          s5:    next_state = (din) ? s5 : s6;
          s6:    next_state = (din) ? s4 : s1;
          default:next_state = s0;
       endcase
    end // Next State Block


    always @( * )
    begin : output_block_proc
       // Combined Actions

       case (current_state)
          s0:    flag =  0;
          s1:    flag =  0;
          s2:    flag =  0;
          s3:    flag = (din) ? 0 : 1;
          s4:    flag =  0;
          s5:    flag =  0;
          s6:    flag = (din) ? 1 : 0;
          default:flag =  0;
       endcase-
    end // Output Block

    //------------------------------------------------------------------
    // Clocked Block for machine fsm
    //------------------------------------------------------------------
    always @( posedge clk or negedge reset )
    begin : clocked_block_proc
       if (!reset)
          current_state <= s0;
       else
          current_state <= next_state;
    end // Clocked Block

endmodule // fsm
```

2.16、设计一个能在串行输入比特流中检测到序列 10101010 的状态机。这里，输入序列的到达方式为最低有效位（LSB）先到，即：第一个 0 先到，然后是 1，接着，又是第二个 0，…，等等。当检测到输入序列 din 中出现 10101010 时，输出 Ready 为 1，否则输出为 0。

要求：

（1）分别采用 Mealy 和 Moore 型状态机设计，首先，画出相应类型状态机的状态转移图，然后，根据状态转移图设计 Verilog 模块。

（2）下面分别是设计模块和仿真测试模块。这里：

Ready：输出端口；当检测到指定序列时，输出为 1，否则，输出为 0；

din：串行序列输入；

clk：时钟信号输入；在时钟的上升沿检查数据

rst：异步复位信号，高电平有效，复位时，Ready 输出为 0。

设计模块为：

（a）**mealy(ouput reg flag, input din, clk, rst)**

（b）**moore(ouput reg flag, input din, clk, rst)**

测试平台的模块名为：

top()

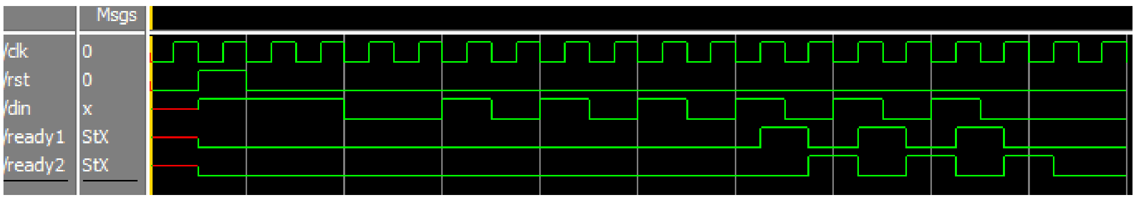
```verilog
`timescale 10 ns / 1 ns

module mealy(output reg ready, input din, clk, rst);
  parameter IDLE = 3'b0, A = 3'b1, B = 3'b10, C = 3'b11,
            D = 3'b100, E = 3'b101, F = 3'b110, G = 3'b111;

  reg [2:0]state;

  always @(posedge clk or posedge rst)
  begin
    if(rst)  begin  state <= IDLE; ready <= 1'b0; end
    else
    begin
      case(state)
        IDLE: if(din)  {ready, state} <= {1'b0, IDLE};
              else     {ready, state} <= {1'b0, A};
        A: if(din)  {ready, state} <= {1'b0, B};
           else     {ready, state} <= {1'b0, A};
        B: if(din)  {ready, state} <= {1'b0, IDLE};
           else     {ready, state} <= {1'b0, C};
        C: if(din)  {ready, state} <= {1'b0, D};
           else     {ready, state} <= {1'b0, A};
        D: if(din)  {ready, state} <= {1'b0, IDLE};
           else     {ready, state} <= {1'b0, E};
        E: if(din)  {ready, state} <= {1'b0, F};
           else     {ready, state} <= {1'b0, A};
        F: if(din)  {ready, state} <= {1'b0, IDLE};
           else     {ready, state} <= {1'b0, G};
        G: if(din)  {ready, state} <= {1'b1, F};
           else     {ready, state} <= {1'b0, A};
        default:;
      endcase
    end
  end
endmodule
```

```verilog
module moore(output reg ready, input din, clk, rst);
  parameter IDLE = 4'b0, A = 4'b1, B = 4'b10, C = 4'b11, D = 4'b100,
            E = 4'b101, F = 4'b110, G = 4'b111, H = 4'b1000;

  reg [3:0]state;

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

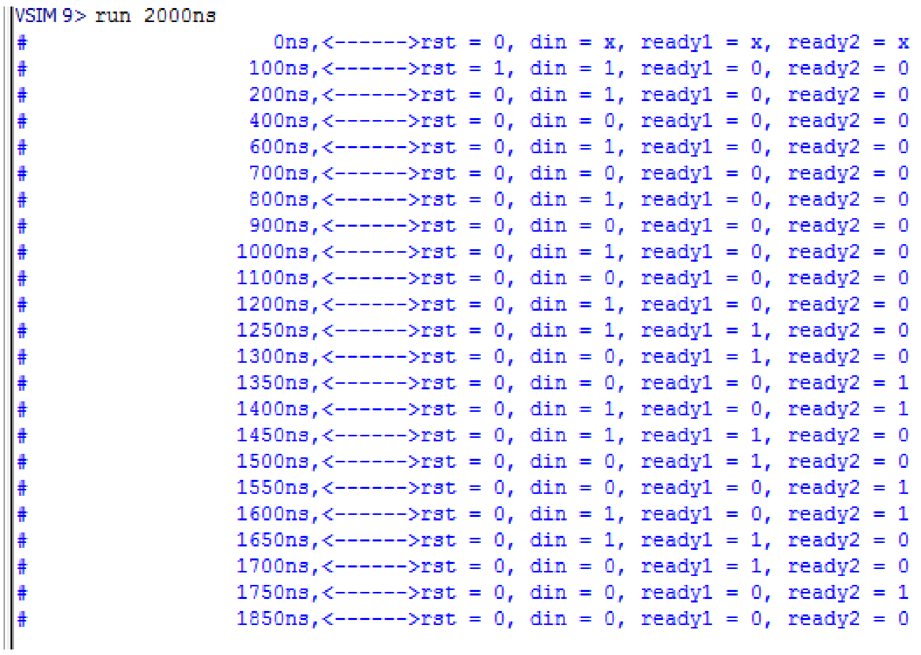
（2）测试模块

```
module top();
  wire ready1, ready2;
  reg  din, clk, rst;

  reg [19:0] data = 20'b1010_1010_1010_0111;
  integer i;

  mealy u1(ready1, din, clk, rst);
  moore u2(ready2, din, clk, rst);

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

  initial
  begin
    rst = 1'b0;
    #10 rst = 1'b1;
    #10 rst = 1'b0;
  end

  initial
  begin
    for(i=0 ; i<20 ; i=i+1)
      #10 din = data[i];
  end

  initial $monitor("%tns,", $time, "<------>rst = %b, din = %b, ready1 = %b, ready2 = %b",
                    rst, din, ready1, ready2);
endmodule
```

（3）测试波形图：



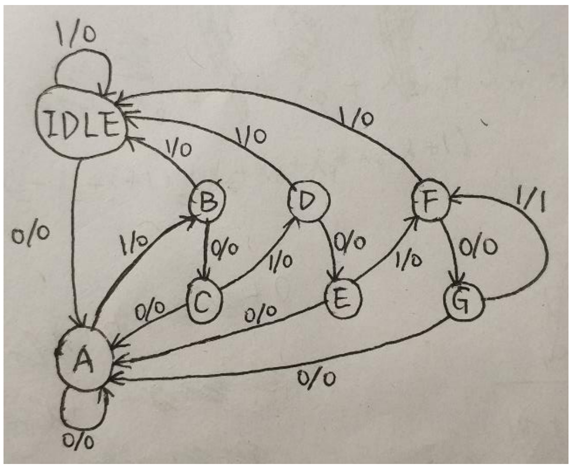
（4）显示输出：



```
VSIM 9> run 2000ns
#                    0ns,<------>rst = 0, din = x, ready1 = x, ready2 = x
#                  100ns,<------>rst = 1, din = 1, ready1 = 0, ready2 = 0
#                  200ns,<------>rst = 0, din = 1, ready1 = 0, ready2 = 0
#                  400ns,<------>rst = 0, din = 0, ready1 = 0, ready2 = 0
#                  600ns,<------>rst = 0, din = 1, ready1 = 0, ready2 = 0
#                  700ns,<------>rst = 0, din = 0, ready1 = 0, ready2 = 0
#                  800ns,<------>rst = 0, din = 1, ready1 = 0, ready2 = 0
#                  900ns,<------>rst = 0, din = 0, ready1 = 0, ready2 = 0
#                 1000ns,<------>rst = 0, din = 1, ready1 = 0, ready2 = 0
#                 1100ns,<------>rst = 0, din = 0, ready1 = 0, ready2 = 0
#                 1200ns,<------>rst = 0, din = 1, ready1 = 0, ready2 = 0
#                 1250ns,<------>rst = 0, din = 1, ready1 = 1, ready2 = 0
#                 1300ns,<------>rst = 0, din = 0, ready1 = 1, ready2 = 0
#                 1350ns,<------>rst = 0, din = 0, ready1 = 0, ready2 = 1
#                 1400ns,<------>rst = 0, din = 1, ready1 = 0, ready2 = 1
#                 1450ns,<------>rst = 0, din = 1, ready1 = 1, ready2 = 0
#                 1500ns,<------>rst = 0, din = 0, ready1 = 1, ready2 = 0
#                 1550ns,<------>rst = 0, din = 0, ready1 = 0, ready2 = 1
#                 1600ns,<------>rst = 0, din = 1, ready1 = 0, ready2 = 1
#                 1650ns,<------>rst = 0, din = 1, ready1 = 1, ready2 = 0
#                 1700ns,<------>rst = 0, din = 0, ready1 = 1, ready2 = 0
#                 1750ns,<------>rst = 0, din = 0, ready1 = 0, ready2 = 1
#                 1850ns,<------>rst = 0, din = 0, ready1 = 0, ready2 = 0
```

（5）设计说明：

Mealy 机：

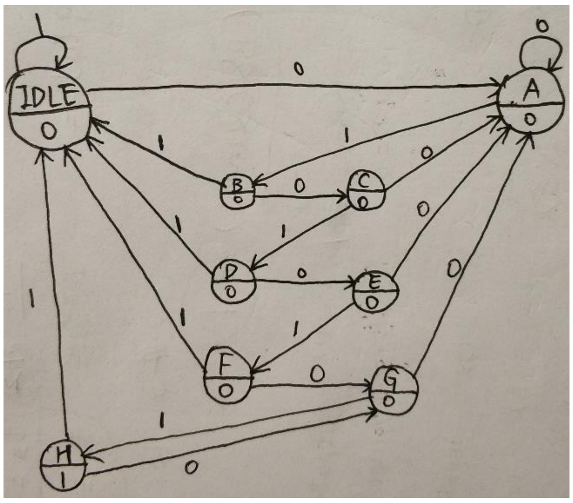


各状态说明：

IDLE：等待第一个 0　　A ：0　　B：01　　C：010　　D：0101

E：01010　　F：010101　　G：0101010

Moore 机：



各状态说明：

IDLE：等待第一个 0　　A ：0　　B：01　　C：010　　D：0101

E：01010　　F：010101　　G：0101010　　H：01010101

第 16 题：

（1）设计模块

```
module mealy(output reg flag, input din, clk, rst);
    parameter IDLE = 8'b0000_0001,
              A = 8'b0000_0010,
              B = 8'b0000_0100,
              C = 8'b0000_1000,
              D = 8'b0001_0000,
              E = 8'b0010_0000,
              F = 8'b0100_0000,
              G = 8'b1000_0000;
    reg [7:0] p_state, n_state;

    always @(posedge clk, posedge rst) begin
        if (rst) begin
            p_state <= IDLE;
        end
        else begin
            p_state <= n_state;
        end
    end

    always @(*) begin
        case(p_state)
            IDLE: n_state = din ? IDLE : A;
               A: n_state = din ?    B : A;
               B: n_state = din ? IDLE : C;
```

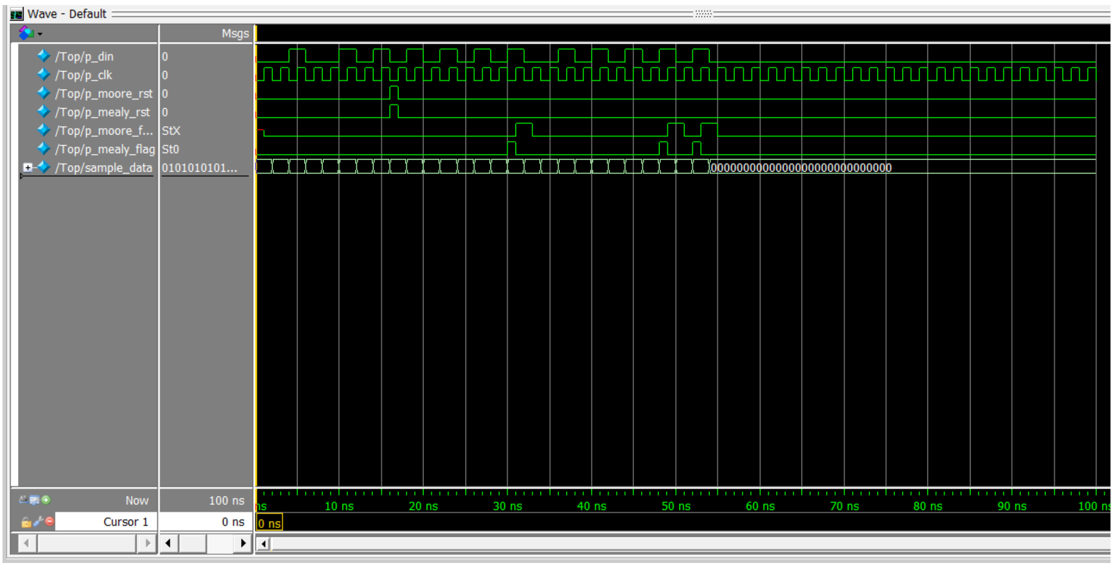
```
                C: n_state = din ?    D : A;
                D: n_state = din ? IDLE : E;
                E: n_state = din ?    F : A;
                F: n_state = din ? IDLE : G;
                G: n_state = din ?    F : A;
             default: n_state = IDLE;
         endcase
         flag = (p_state == G && din == 1'b1) ? 1'b1 : 1'b0;
     end
endmodule
```

```
module moore(output reg flag, input din, clk, 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] p_state, n_state;

    always @(posedge clk, posedge rst) begin
        if (rst) begin
            p_state <= IDLE;
        end
        else begin
            p_state <= n_state;
        end
    end

    always @(*) begin
        case(p_state)
            IDLE: n_state = din ? IDLE : A;
               A: n_state = din ?    B : A;
               B: n_state = din ? IDLE : C;
               C: n_state = din ?    D : A;
               D: n_state = din ? IDLE : E;
               E: n_state = din ?    F : A;
               F: n_state = din ? IDLE : G;
               G: n_state = din ?    H : A;
               H: n_state = din ? IDLE : G;
               default: n_state = IDLE;
        endcase
        flag = (p_state == H) ? 1'b1 : 1'b0;
    end
endmodule
```

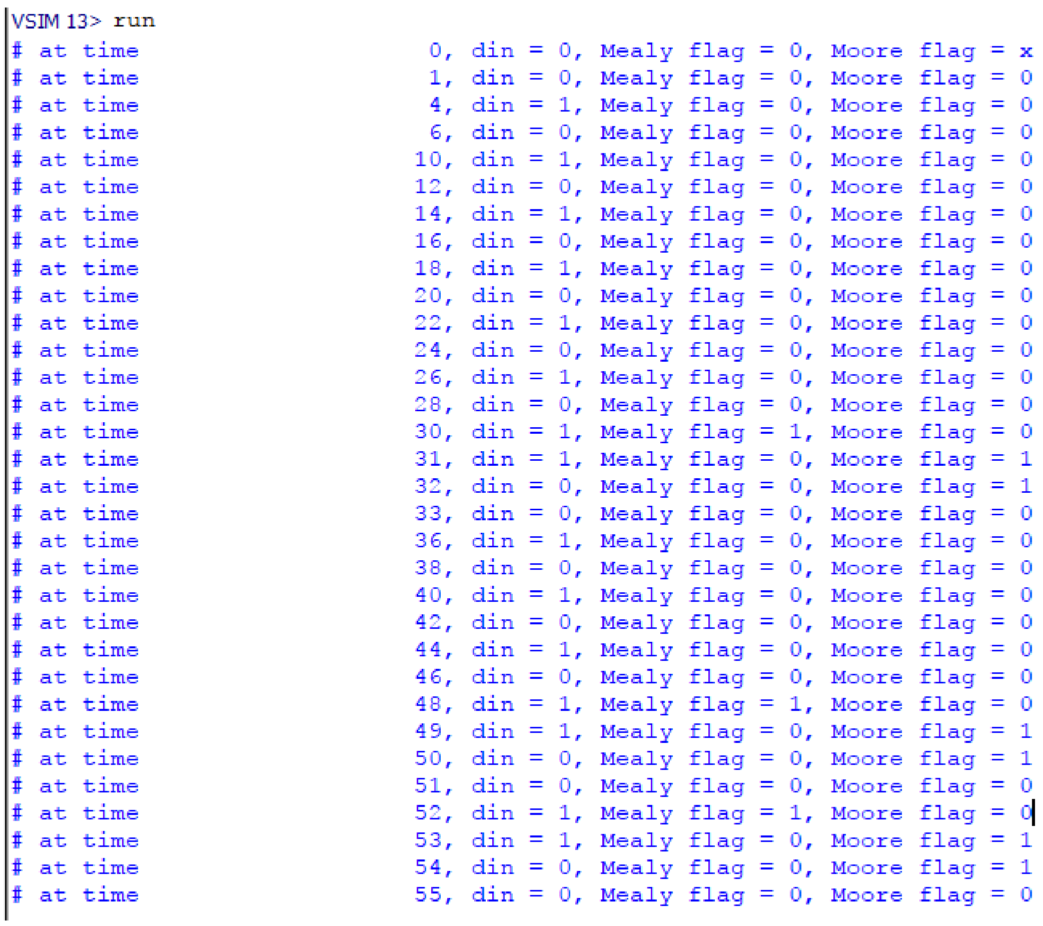
（2）测试模块

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

module Top();
    reg p_din, p_clk;
    reg p_moore_rst, p_mealy_rst;
    wire p_moore_flag, p_mealy_flag;
    localparam DELTA = 1;
    localparam T = 2 * DELTA;
    reg [27:0] sample_data =
28'b0101_0101_0100_1010_1010_1010_0100;

    mealy
mealy_inst(.clk(p_clk), .din(p_din), .rst(p_mealy_rst), .flag(p_mea
ly_flag));
    moore
moore_inst(.clk(p_clk), .din(p_din), .rst(p_moore_rst), .flag(p_moo
re_flag));

    initial begin
        p_clk = 1'b0;
        forever #(DELTA) p_clk = ~p_clk;
    end

    initial begin
        forever begin
            p_din = sample_data[0];
            #(T) sample_data = sample_data >> 1'b1;
        end
    end

    initial begin
        {p_mealy_rst, p_moore_rst} = 2'b00;
        #(T * 8) {p_mealy_rst, p_moore_rst} = 2'b11;
        #(DELTA) {p_mealy_rst, p_moore_rst} = 2'b00;
    end
    initial
        $monitor("at time %t, din = %b, Mealy flag = %b, Moore flag
= %b",
            $time, p_din, p_mealy_flag, p_moore_flag);
endmodule
```

（3）测试波形图：如果很多，可以提供部分波形内容；



（4）显示输出（可选）：如果需要显示输出来说明模块设计的正确性；



```
VSIM 13> run
# at time                    0, din = 0, Mealy flag = 0, Moore flag = x
# at time                    1, din = 0, Mealy flag = 0, Moore flag = 0
# at time                    4, din = 1, Mealy flag = 0, Moore flag = 0
# at time                    6, din = 0, Mealy flag = 0, Moore flag = 0
# at time                   10, din = 1, Mealy flag = 0, Moore flag = 0
# at time                   12, din = 0, Mealy flag = 0, Moore flag = 0
# at time                   14, din = 1, Mealy flag = 0, Moore flag = 0
# at time                   16, din = 0, Mealy flag = 0, Moore flag = 0
# at time                   18, din = 1, Mealy flag = 0, Moore flag = 0
# at time                   20, din = 0, Mealy flag = 0, Moore flag = 0
# at time                   22, din = 1, Mealy flag = 0, Moore flag = 0
# at time                   24, din = 0, Mealy flag = 0, Moore flag = 0
# at time                   26, din = 1, Mealy flag = 0, Moore flag = 0
# at time                   28, din = 0, Mealy flag = 0, Moore flag = 0
# at time                   30, din = 1, Mealy flag = 1, Moore flag = 0
# at time                   31, din = 1, Mealy flag = 0, Moore flag = 1
# at time                   32, din = 0, Mealy flag = 0, Moore flag = 1
# at time                   33, din = 0, Mealy flag = 0, Moore flag = 0
# at time                   36, din = 1, Mealy flag = 0, Moore flag = 0
# at time                   38, din = 0, Mealy flag = 0, Moore flag = 0
# at time                   40, din = 1, Mealy flag = 0, Moore flag = 0
# at time                   42, din = 0, Mealy flag = 0, Moore flag = 0
# at time                   44, din = 1, Mealy flag = 0, Moore flag = 0
# at time                   46, din = 0, Mealy flag = 0, Moore flag = 0
# at time                   48, din = 1, Mealy flag = 1, Moore flag = 0
# at time                   49, din = 1, Mealy flag = 0, Moore flag = 1
# at time                   50, din = 0, Mealy flag = 0, Moore flag = 1
# at time                   51, din = 0, Mealy flag = 0, Moore flag = 0
# at time                   52, din = 1, Mealy flag = 1, Moore flag = 0
# at time                   53, din = 1, Mealy flag = 0, Moore flag = 1
# at time                   54, din = 0, Mealy flag = 0, Moore flag = 1
# at time                   55, din = 0, Mealy flag = 0, Moore flag = 0
```

（5）设计说明（可选）：如果有需要说明的部分。

这道题中我采用组合逻辑用来得到输出，对于 Mealy 型状态机，输出逻辑是由现态和输入决定的；对于 Moore 型状态机，输出是由现态决定的。因此假设对于 Mealy 状态机，能使 flag=1 的状态为 X，输入为 a，那么在波形上表示为 Mealy 状态机的输出在状态为 X 的时候，一旦输入为 a，输出就变为 1。而 Moore 状态机则是只有在时钟上跳沿到来的时候，转换为能使 flag=1 的状态之后，输出才能为 1，而且由于状态的转换都是同步于时钟的上跳沿的，因此 flag 的转换也是和时钟上跳沿对齐的。

第 16 题：

（1）设计模块：

```
module mealy(flag,din,clk,rst);
       parameter
s0=9'b0000_0001,s1=9'b0000_0010,s2=9'b0000_0100,s3=9'b0000_1000,

s4=9'b0001_0000,s5=9'b0010_0000,s6=9'b0100_0000,s7=9'b0_1000_0000,
                      s8=9'b1_0000_0000;
   reg [8:0]state;
   output flag;
   reg flag;
   input din,clk,rst;
   always@(posedge clk or posedge rst)begin
      if(rst)begin
         state<=s0;
         flag<=0;
      end
   else begin
      case(state)
         s0:{flag,state}<=(din)?{1'b0,s0}:{1'b0,s1};
         s1:{flag,state}<=(din)?{1'b0,s2}:{1'b0,s1};
         s2:{flag,state}<=(din)?{1'b0,s0}:{1'b0,s3};
         s3:{flag,state}<=(din)?{1'b0,s4}:{1'b0,s1};
         s4:{flag,state}<=(din)?{1'b0,s0}:{1'b0,s5};
         s5:{flag,state}<=(din)?{1'b0,s6}:{1'b0,s1};
         s6:{flag,state}<=(din)?{1'b0,s0}:{1'b0,s7};
         s7:{flag,state}<=(din)?{1'b1,s8}:{1'b0,s1};
         s8:{flag,state}<=(din)?{1'b0,s0}:{1'b0,s7};
         default:begin flag<=1'b0;state<=s0;end
         endcase
   end
   end
endmodule
```

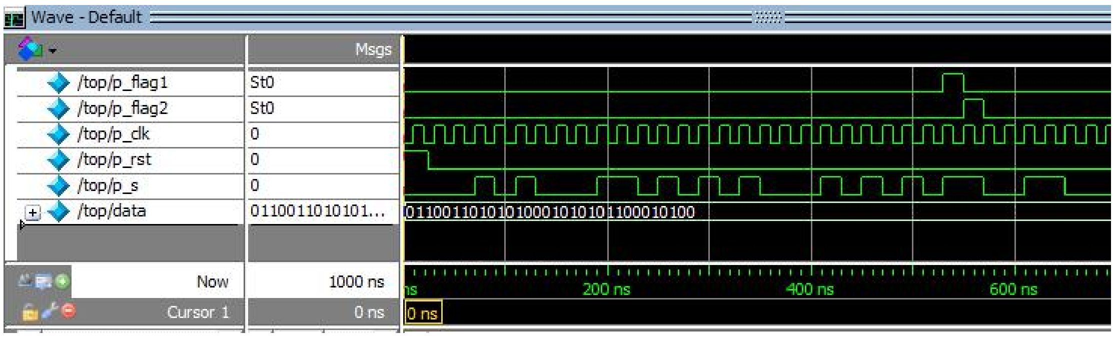
```
module moore(flag,din,clk,rst);
   parameter
s0=9'b0000_0001,s1=10'b0000_0010,s2=10'b0000_0100,s3=9'b0000_1000,

s4=9'b0001_0000,s5=9'b0010_0000,s6=9'b0100_0000,s7=9'b0_1000_0000,
                    s8=9'b1_0000_0000;
   reg [8:0]state;
   output flag;
   reg flag;
   input din,clk,rst;
   always@(posedge clk or posedge rst)begin
      if(rst)begin
         state<=s0;
         flag<=0;
      end
      else begin
         flag<= (state==s8)?1'b1:1'b0;
         case (state)
            s0: state<=(din)?s0:s1;
            s1: state<=(din)?s2:s1;
            s2: state<=(din)?s0:s3;
            s3: state<=(din)?s4:s1;
            s4: state<=(din)?s0:s5;
            s5: state<=(din)?s6:s1;
            s6: state<=(din)?s0:s7;
            s7: state<=(din)?s8:s1;
            s8: state<=(din)?s0:s7;
            default:state<=s0;
         endcase
      end
   end
endmodule
```

（2）测试模块：

```
`timescale 1ns/1ns;
`include "hw16.v";
module top();
      wire p_flag1,p_flag2;
      reg    p_clk, p_rst, p_s;

      initial begin p_rst = 1'b1; #25 p_rst = 1'b0; end
      initial begin p_clk = 0;    forever #10 p_clk = ~p_clk; end

      parameter SIZE = 32;
```

```
reg [SIZE-1 : 0] data = 32'b0110_0110_1010_1000_1010_1011_0001_0100;

initial begin: SERIES
        integer i;
        p_s = 0;
        #30;
        for ( i = 0; i < SIZE; i= i+1) begin
                p_s = data[0];
                data = data >> 1;
                #20;
        end
end
mealy u1( .flag(p_flag1), .clk(p_clk), .rst(p_rst), .din(p_s) );
moore u2( .flag(p_flag2), .clk(p_clk), .rst(p_rst), .din(p_s) );
endmodule
```

（3）测试波形图：



第 16 题(a)：

（1）设计模块:

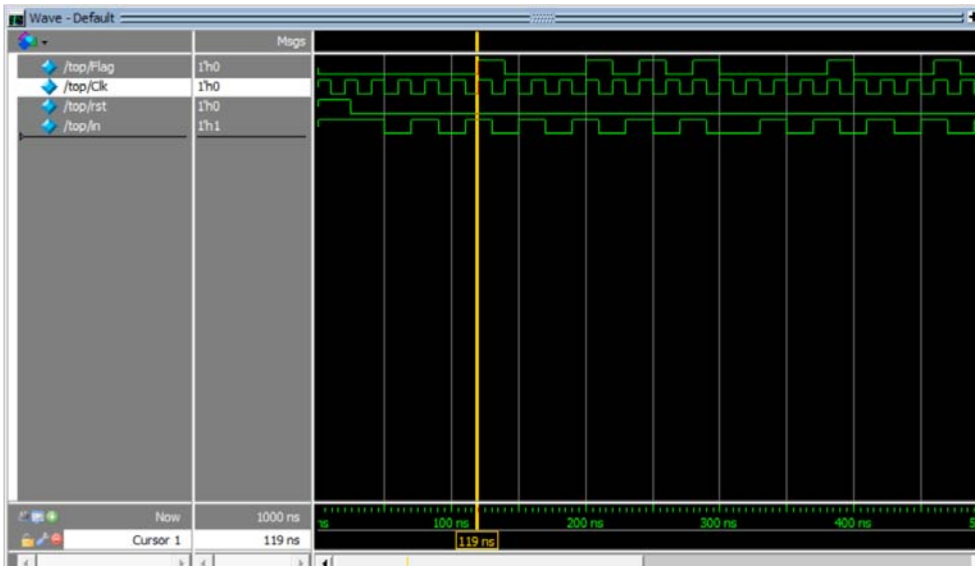
```
`timescale 1ns/1ns
module mealy(flag,din,clk,rst);
output reg flag;
input din,clk,rst;
reg[2:0]state;
always@(posedge clk or posedge rst)
begin
if(rst) begin flag<=1'b0; state<=0; end
else  begin
       case(state)
         0: if(~din) begin flag<=1'b0; state<=1; end
               else begin flag<=1'b0; state<=0; end
         1: if(din) begin flag<=1'b0; state<=2;end
                else begin flag<=1'b0; state<=1; end
         2: if(~din) begin flag<=1'b0; state<=3; end
```

```
                else begin flag<=1'b0; state<=0; end
             3: if(din) begin flag<=1'b1; state<=4; end
                else  begin flag<=1'b0; state<=1; end
             4: if(din) begin flag<=1'b0; state<=0; end
                else begin flag<=1'b0; state<=5; end
             5: if(din) begin flag<=1'b0; state<=6; end
                else begin flag<=1'b0; state<=1; end
             6: if(din) begin flag<=1'b0; state<=0; end
                else begin flag<=1'b0; state<=7; end
             7: if(din) begin flag<=1'b1; state<=6; end
                else begin flag<=1'b0; state<=1; end
             default: begin flag<=1'b0; state<=0; end
            endcase
           end
end

endmodule
```

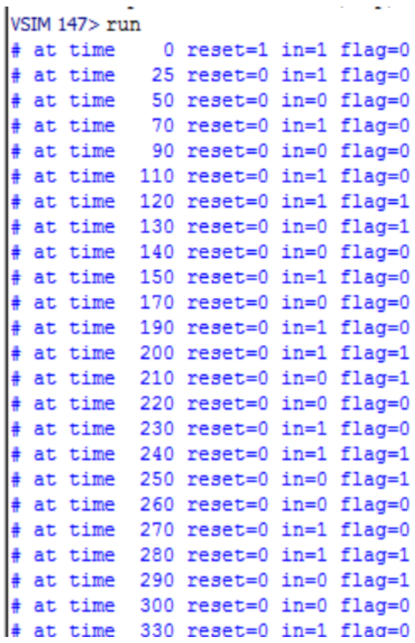
（2）测试模块:

```
module top();
wire Flag;
reg  Clk,rst,in;
reg [23:0]data=24'b1010_1010_1001_0101_0101_0101;
mealy m1(Flag,in,Clk,rst);
initial
begin
rst=1'b1;
#25 rst=1'b0;
end
initial
begin
Clk=1'b1;
forever
#10 Clk=~Clk;
end
initial
begin:seq
integer i;
in=1'b1;
#30;
for(i=0;i<24;i=i+1)
  begin
        in=data[0];
        data=data>>1;
        #20;
  end
end
initial
$monitor("at time %4d reset=%b in=%b flag=%b",$time,rst,in,Flag);
endmodule
```

（3）测试波形图:

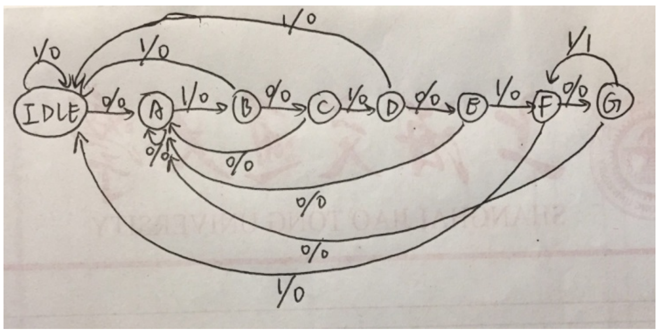


（4）显示输出：



```
VSIM 147> run
# at time    0 reset=1 in=1 flag=0
# at time   25 reset=0 in=1 flag=0
# at time   50 reset=0 in=0 flag=0
# at time   70 reset=0 in=1 flag=0
# at time   90 reset=0 in=0 flag=0
# at time  110 reset=0 in=1 flag=0
# at time  120 reset=0 in=1 flag=1
# at time  130 reset=0 in=0 flag=1
# at time  140 reset=0 in=0 flag=0
# at time  150 reset=0 in=1 flag=0
# at time  170 reset=0 in=0 flag=0
# at time  190 reset=0 in=1 flag=0
# at time  200 reset=0 in=1 flag=1
# at time  210 reset=0 in=0 flag=1
# at time  220 reset=0 in=0 flag=0
# at time  230 reset=0 in=1 flag=0
# at time  240 reset=0 in=1 flag=1
# at time  250 reset=0 in=0 flag=1
# at time  260 reset=0 in=0 flag=0
# at time  270 reset=0 in=1 flag=0
# at time  280 reset=0 in=1 flag=1
# at time  290 reset=0 in=0 flag=1
# at time  300 reset=0 in=0 flag=0
# at time  330 reset=0 in=1 flag=0
```

（5）设计说明：mealy 型状态机状态转移图如下：



第 16 题(b)：

（1）设计模块:

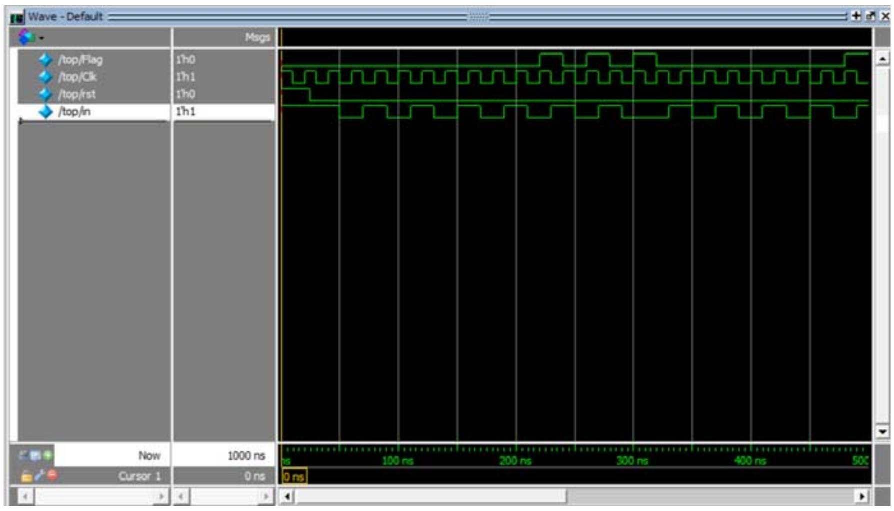
```
`timescale 1ns/1ns
module moore(flag,din,clk,rst);
output reg flag;
input din,clk,rst;
reg[3:0]state;
always@(posedge clk or posedge rst)
begin
if(rst) begin flag<=1'b0; state<=0; end
else  begin
       flag<=(state==8)? 1'b1:1'b0;
       case(state)
         0: state<=(din)? 0:1;
         1: state<=(din)? 2:1;
         2: state<=(din)? 0:3;
         3: state<=(din)? 4:1;
         4: state<=(din)? 0:5;
         5: state<=(din)? 6:1;
         6: state<=(din)? 0:7;
         7: state<=(din)? 8:1;
         8: state<=(din)? 0:7;
         default: begin flag<=1'b0; state<=0; end
        endcase
       end
end

endmodule
```

（2）测试模块:

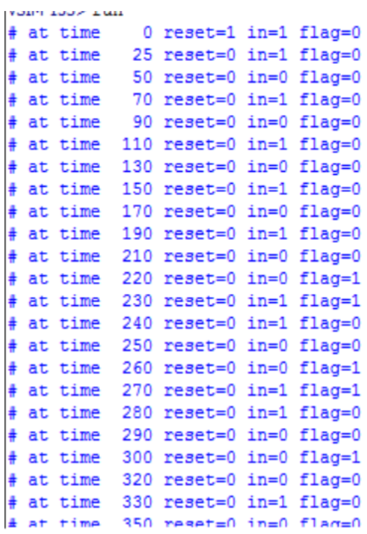
```
module top();
wire Flag;
reg  Clk,rst,in;
reg [23:0]data=24'b1010_1010_1001_0101_0101_0101;
moore m1(Flag,in,Clk,rst);
initial
begin
rst=1'b1;
#25 rst=1'b0;
end
initial
begin
Clk=1'b1;
forever
#10 Clk=~Clk;
end
initial
begin:seq
integer i;
in=1'b1;
#30;
for(i=0;i<24;i=i+1)
  begin
        in=data[0];
        data=data>>1;
        #20;
  end
end
```

```
initial
$monitor("at time %4d reset=%b in=%b flag=%b",$time,rst,in,Flag);
endmodule
```

（3）测试波形图：

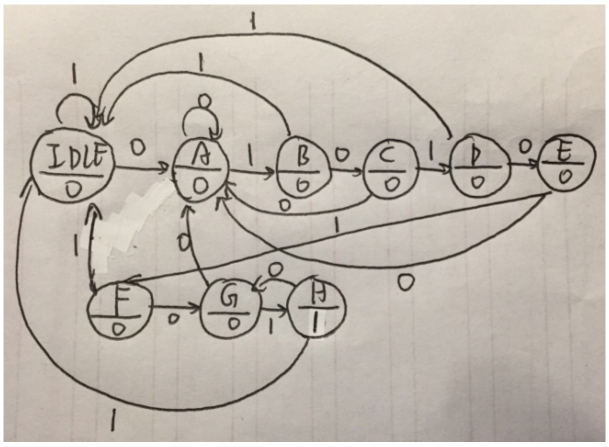


（4）显示输出：



```
VSIM 133> run
# at time    0 reset=1 in=1 flag=0
# at time   25 reset=0 in=1 flag=0
# at time   50 reset=0 in=0 flag=0
# at time   70 reset=0 in=1 flag=0
# at time   90 reset=0 in=0 flag=0
# at time  110 reset=0 in=1 flag=0
# at time  130 reset=0 in=0 flag=0
# at time  150 reset=0 in=1 flag=0
# at time  170 reset=0 in=0 flag=0
# at time  190 reset=0 in=1 flag=0
# at time  210 reset=0 in=0 flag=0
# at time  220 reset=0 in=0 flag=1
# at time  230 reset=0 in=1 flag=1
# at time  240 reset=0 in=1 flag=0
# at time  250 reset=0 in=0 flag=0
# at time  260 reset=0 in=0 flag=1
# at time  270 reset=0 in=1 flag=1
# at time  280 reset=0 in=1 flag=0
# at time  290 reset=0 in=0 flag=0
# at time  300 reset=0 in=0 flag=1
# at time  320 reset=0 in=0 flag=0
# at time  330 reset=0 in=1 flag=0
# at time  350 reset=0 in=0 flag=0
```

（5）设计说明：moore 型状态机状态转移图如下：


IDLE
0
A
0
B
0
C
0
D
0
E
0
F
0
G
0
H
1
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1
0
1