数字系统设计作业

学号：516021910***　　姓名：***　　日期：****

第 1 题：

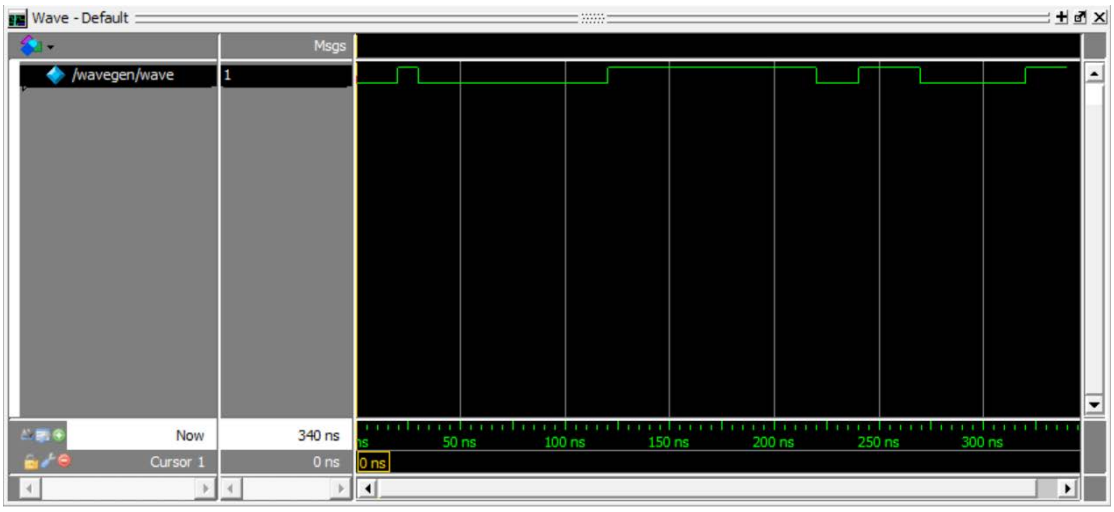
（1）设计模块

```
`timescale 10ns/1ns
module wavegen;
    reg wave;
    initial
    begin
        wave = 1'b0;
        #2 wave = ~wave;
        #1 wave = ~wave;
        #9 wave = ~wave;
        #10 wave = ~wave;
        #2 wave = ~wave;
        #3 wave = ~wave;
        #5 wave = ~wave;
        #2 $stop;
    end
endmodule
```

（2）测试模块

本题即为测试模块。

（3）测试波形图：



第 2 题：

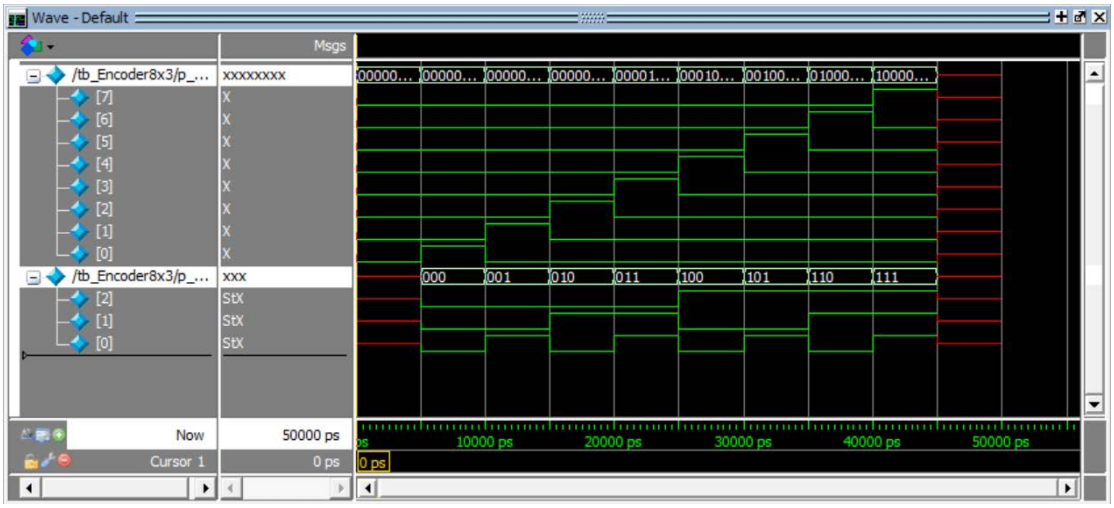
（1）设计模块

```
module Encoder8x3(output reg[2:0] code, input [7:0] data );
    always@(*)
        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
endmodule
```

（2）测试模块

```
`timescale 1ns/100ps
`include "Encoder8x3.v"
module tb_Encoder8x3();

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

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

    integer i;
    initial
    begin
        p_data = 8'b0;
        #5 p_data = 8'b1;
        for(i = 1 ; i < 8; i = i+1)
            #5 p_data = p_data << 1;
        #5 p_data = 8'bx;
        #5 $stop;
    end

    initial
    $monitor("At %4t ns, data = %8b, code = %3d", $time, p_data,
p_code);
endmodule
```

（3）测试波形图：



（4） 显示输出（可选）：

```
# At     0 ns, data = 00000000, code =   x
# At    50 ns, data = 00000001, code =   0
# At   100 ns, data = 00000010, code =   1
# At   150 ns, data = 00000100, code =   2
# At   200 ns, data = 00001000, code =   3
# At   250 ns, data = 00010000, code =   4
# At   300 ns, data = 00100000, code =   5
# At   350 ns, data = 01000000, code =   6
# At   400 ns, data = 10000000, code =   7
# At   450 ns, data = xxxxxxxx, code =   x
```

第 3 题（a）：

（1）设计模块

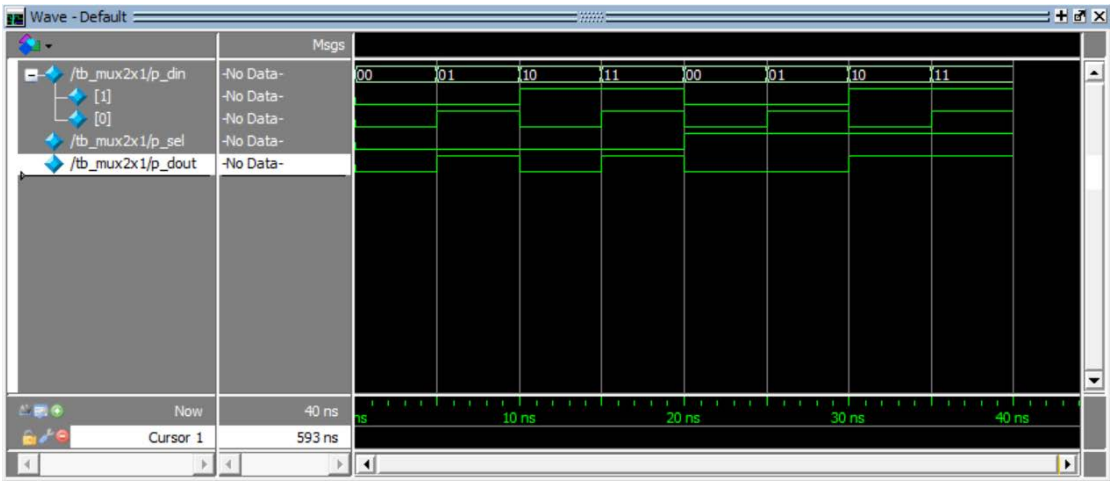
```
module mux2x1(output dout,
              input sel,
              input [1:0] din);
   bufif1 b2( dout, din[1], sel );
   bufif0 b1( dout, din[0], sel );
endmodule
```

（2）测试模块

```
`timescale 1ns/1ns
`include "mux2x1.v"
module tb_mux2x1();
   reg [1:0] p_din;
   reg p_sel;
```

```
    wire p_dout;

    mux2x1 m( .dout( p_dout ), .sel(p_sel), .din(p_din) );

    integer i;
    initial
    begin
        {p_sel, p_din} = 3'b0;
        for(i=0 ; i<7 ; i=i+1)
            #5 {p_sel, p_din} = {p_sel, p_din} + 1;
        #5 $stop;
    end

    initial
    $monitor( "At time %t ns, dout=%1b, sel=%1b, din=%2b",
            $time, p_dout, p_sel, p_din);
endmodule
```

（3）测试波形图：



（4） 显示输出（可选）：

```
# At time                0 ns, dout=0, sel=0, din=00
# At time                5 ns, dout=1, sel=0, din=01
# At time               10 ns, dout=0, sel=0, din=10
# At time               15 ns, dout=1, sel=0, din=11
# At time               20 ns, dout=0, sel=1, din=00
# At time               25 ns, dout=0, sel=1, din=01
# At time               30 ns, dout=1, sel=1, din=10
# At time               35 ns, dout=1, sel=1, din=11
```

第 3 题（b）：

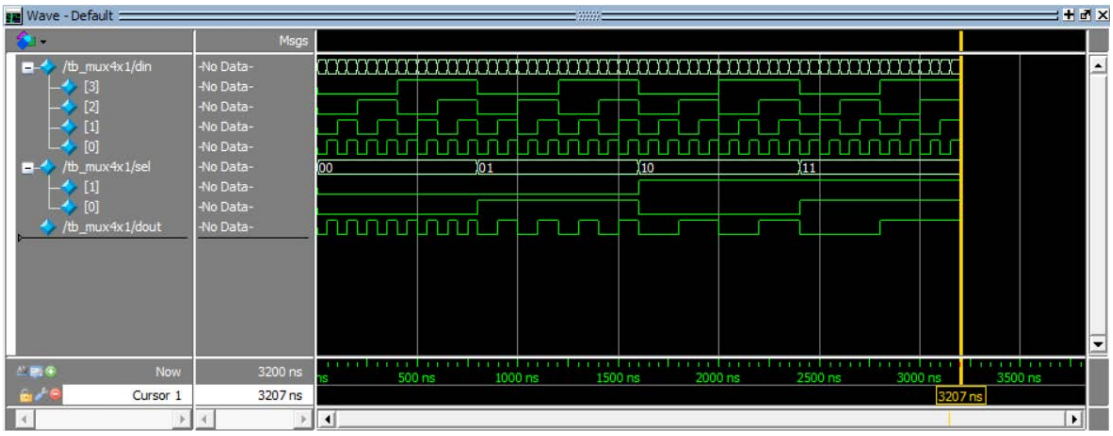
（1）设计模块

```
`include "mux2x1.v"
module mux4x1(output dout,
              input [1:0] sel, input [3:0] din);
   wire d1,d2;
   mux2x1 low1(.dout(d1),.sel(sel[0]),.din(din[3:2]));
   mux2x1 low2(.dout(d2),.sel(sel[0]),.din(din[1:0]));
   mux2x1 high(.dout(dout),.sel(sel[1]),.din({d1,d2}));
endmodule
```

（2）测试模块

```
`include "mux4x1.v"
`timescale 10ns / 1ns
module tb_mux4x1;
   reg [3:0] din;
   reg [1:0] sel;
   wire dout;
   integer i;
   mux4x1 t2(.dout(dout),.sel(sel),.din(din));
   initial
   begin
      {sel[1:0],din[3:0]} = 6'b000000;
      for (i = 1; i < 64; i = i+1)
         #5 {sel,din[3:0]} = {sel,din[3:0]} + 1;
      #5 $stop;
   end
   initial
      $monitor("At %t, sel = %b, din = %b, dout = %b",
               $time,sel,din,dout);
endmodule
```

（3）测试波形图：



（4） 显示输出（可选）：

```
# At                    0, sel = 00, din = 0000, dout = 0
# At                   50, sel = 00, din = 0001, dout = 1
# At                  100, sel = 00, din = 0010, dout = 0
# At                  150, sel = 00, din = 0011, dout = 1
# At                  200, sel = 00, din = 0100, dout = 0
# At                  250, sel = 00, din = 0101, dout = 1
# At                  300, sel = 00, din = 0110, dout = 0
# At                  350, sel = 00, din = 0111, dout = 1
# At                  400, sel = 00, din = 1000, dout = 0
# At                  450, sel = 00, din = 1001, dout = 1
# At                  500, sel = 00, din = 1010, dout = 0
# At                  550, sel = 00, din = 1011, dout = 1
# At                  600, sel = 00, din = 1100, dout = 0
# At                  650, sel = 00, din = 1101, dout = 1
# At                  700, sel = 00, din = 1110, dout = 0
# At                  750, sel = 00, din = 1111, dout = 1
# At                  800, sel = 01, din = 0000, dout = 0
# At                  850, sel = 01, din = 0001, dout = 0
# At                  900, sel = 01, din = 0010, dout = 1
# At                  950, sel = 01, din = 0011, dout = 1
# At                 1000, sel = 01, din = 0100, dout = 0
# At                 1050, sel = 01, din = 0101, dout = 0
# At                 1100, sel = 01, din = 0110, dout = 1
# At                 1150, sel = 01, din = 0111, dout = 1
# At                 1200, sel = 01, din = 1000, dout = 0
# At                 1250, sel = 01, din = 1001, dout = 0
# At                 1300, sel = 01, din = 1010, dout = 1
# At                 1350, sel = 01, din = 1011, dout = 1
# At                 1400, sel = 01, din = 1100, dout = 0
# At                 1450, sel = 01, din = 1101, dout = 0
# At                 1500, sel = 01, din = 1110, dout = 1
# At                 1550, sel = 01, din = 1111, dout = 1
# At                 1600, sel = 10, din = 0000, dout = 0
# At                 1650, sel = 10, din = 0001, dout = 0
# At                 1700, sel = 10, din = 0010, dout = 0
```

```
# At                1750, sel = 10, din = 0011, dout = 0
# At                1800, sel = 10, din = 0100, dout = 1
# At                1850, sel = 10, din = 0101, dout = 1
# At                1900, sel = 10, din = 0110, dout = 1
# At                1950, sel = 10, din = 0111, dout = 1
# At                2000, sel = 10, din = 1000, dout = 0
# At                2050, sel = 10, din = 1001, dout = 0
# At                2100, sel = 10, din = 1010, dout = 0
# At                2150, sel = 10, din = 1011, dout = 0
# At                2200, sel = 10, din = 1100, dout = 1
# At                2250, sel = 10, din = 1101, dout = 1
# At                2300, sel = 10, din = 1110, dout = 1
# At                2350, sel = 10, din = 1111, dout = 1
# At                2400, sel = 11, din = 0000, dout = 0
# At                2450, sel = 11, din = 0001, dout = 0
# At                2500, sel = 11, din = 0010, dout = 0
# At                2550, sel = 11, din = 0011, dout = 0
# At                2600, sel = 11, din = 0100, dout = 0
# At                2650, sel = 11, din = 0101, dout = 0
# At                2700, sel = 11, din = 0110, dout = 0
# At                2750, sel = 11, din = 0111, dout = 0
# At                2800, sel = 11, din = 1000, dout = 1
# At                2850, sel = 11, din = 1001, dout = 1
# At                2900, sel = 11, din = 1010, dout = 1
# At                2950, sel = 11, din = 1011, dout = 1
# At                3000, sel = 11, din = 1100, dout = 1
# At                3050, sel = 11, din = 1101, dout = 1
# At                3100, sel = 11, din = 1110, dout = 1
# At                3150, sel = 11, din = 1111, dout = 1
```

第 4 题：

（1）设计模块

```
module comb_str(output Y,
                input A, B, C, D);
   wire nD,t1,t2,nt1;
   or (t1,A,D);
   not (nD,D);
   and (t2,B,C,nD);
   not (nt1,t1);
   and (Y,nt1,t2);
endmodule

module comb_dataflow(output Y,
                     input A, B, C, D);
   wire nD,t1,t2,nt1;
   assign t1 =  A | D,
          nD = ~D,
          t2 = B & C & nD,
          nt1 = ~t1,
          Y = nt1 & t2;
endmodule
```

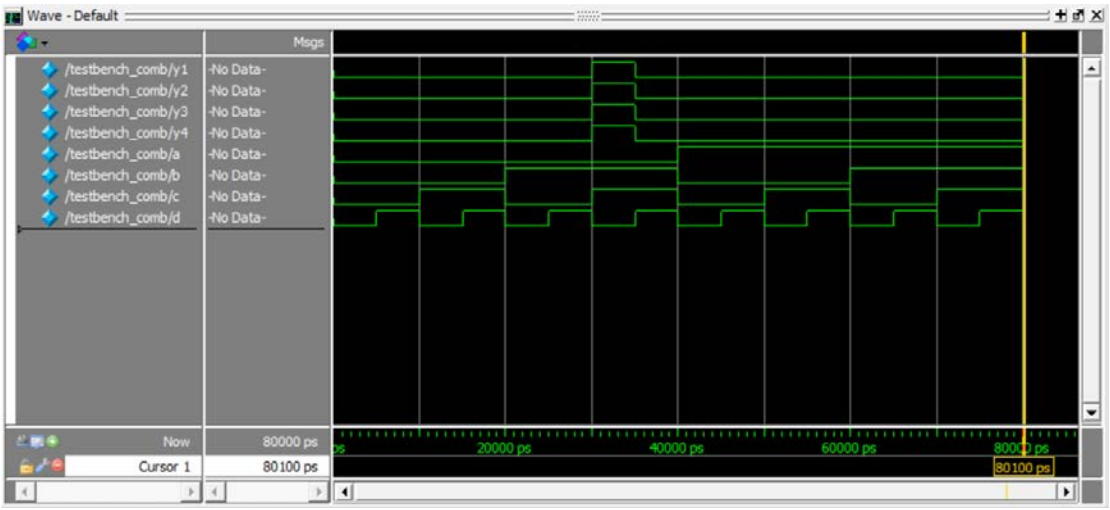
```
module comb_behavior(output reg Y,
                     input A, B, C, D);
    reg t1, t2;
    always@(*)
    begin
        t1 = (~D) & B & C;
        t2 = ~(A | D);
        Y = t1 & t2;
    end
endmodule

primitive comb_prim(output Y,
                    input A, B, C, D);
    table
    //  A B C D : Y
        ? 0 ? ? : 0;
        ? ? 0 ? : 0;
        ? ? ? 1 : 0;
        1 ? ? ? : 0;
        0 1 1 0 : 1;
    endtable
endprimitive
```

（2）测试模块

```
`include "comb_str.v"
`include "comb_dataflow.v"
`include "comb_behavior.v"
`include "comb_prim.v"
`timescale 1ns / 100ps
module testbench_comb;
    wire y1,y2,y3,y4;
    reg a,b,c,d;
    integer i;

    comb_str c1(.Y(y1),.A(a),.B(b),.C(c),.D(d));
    comb_dataflow c2(.Y(y2),.A(a),.B(b),.C(c),.D(d));
    comb_behavior c3(.Y(y3),.A(a),.B(b),.C(c),.D(d));
    comb_prim (y4,a,b,c,d);

    initial
    begin
        {a,b,c,d} = 4'b0;
```

```
        for (i = 1; i < 16; i = i+1)
            #5 {a,b,c,d} = {a,b,c,d} + 1;
        #5 $stop;
    End

    initial
        $monitor("At %t, abcd = %b, y1 = %b,y2 = %b,y3 = %b,y4 = %b",
             $time,{a,b,c,d},y1,y2,y3,y4);
endmodule
```

（3）测试波形图：



（4） 显示输出（可选）：

ModelSim 的 Transcript Window 中输出文本表示的仿真结果如下：

```
# At                    0, abcd = 0000, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                   50, abcd = 0001, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  100, abcd = 0010, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  150, abcd = 0011, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  200, abcd = 0100, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  250, abcd = 0101, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  300, abcd = 0110, y1 = 1,y2 = 1,y3 = 1,y4 = 1
# At                  350, abcd = 0111, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  400, abcd = 1000, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  450, abcd = 1001, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  500, abcd = 1010, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  550, abcd = 1011, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  600, abcd = 1100, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  650, abcd = 1101, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  700, abcd = 1110, y1 = 0,y2 = 0,y3 = 0,y4 = 0
# At                  750, abcd = 1111, y1 = 0,y2 = 0,y3 = 0,y4 = 0
```

第 5 题：

（1）设计模块

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

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

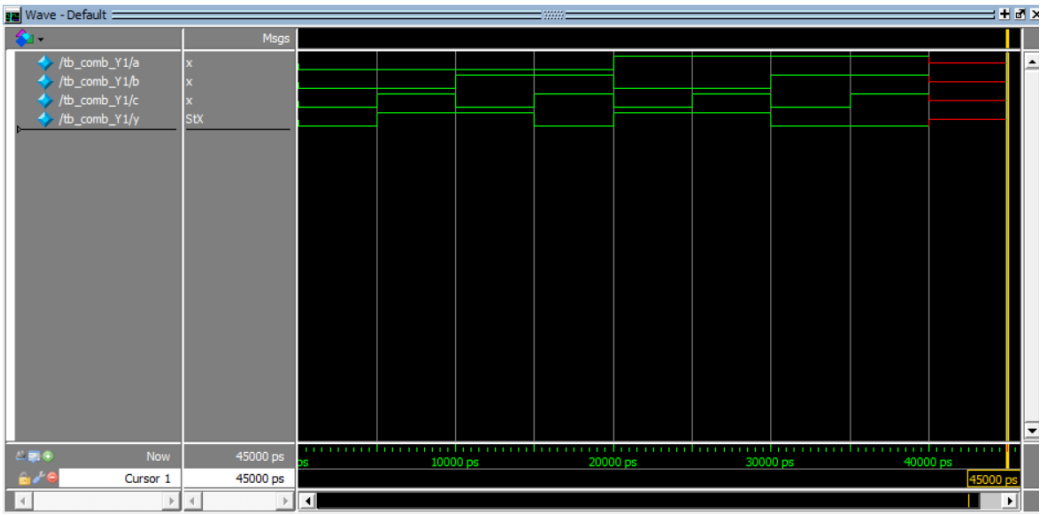
（2）测试模块

```
`include "comb_Y1.v"
`timescale 1ns / 100ps
module tb_comb_Y1;
   reg a,b,c;
   wire y;
   integer i;

   comb_Y1 y1(.Y(y),.A(a),.B(b),.C(c));

   initial
   begin
      {a,b,c} = 3'b0;
      for (i = 1; i < 8; i=i+1)
         #5 {a,b,c} = {a,b,c} + 1;
      #5 {a,b,c} = 3'bx;
      #5 $stop;
   End

   initial
      $monitor("At %t, a = %b, b = %b,c = %b, y = %b",
               $time,a,b,c,y);
endmodule

`include "comb_Y2.v"
`timescale 1ns / 100ps
module tb_comb_Y2;
   reg a,b,c,d;
   wire y;
   integer i;

   comb_Y2 y2(.Y(y),.A(a),.B(b),.C(c),.D(d));
```

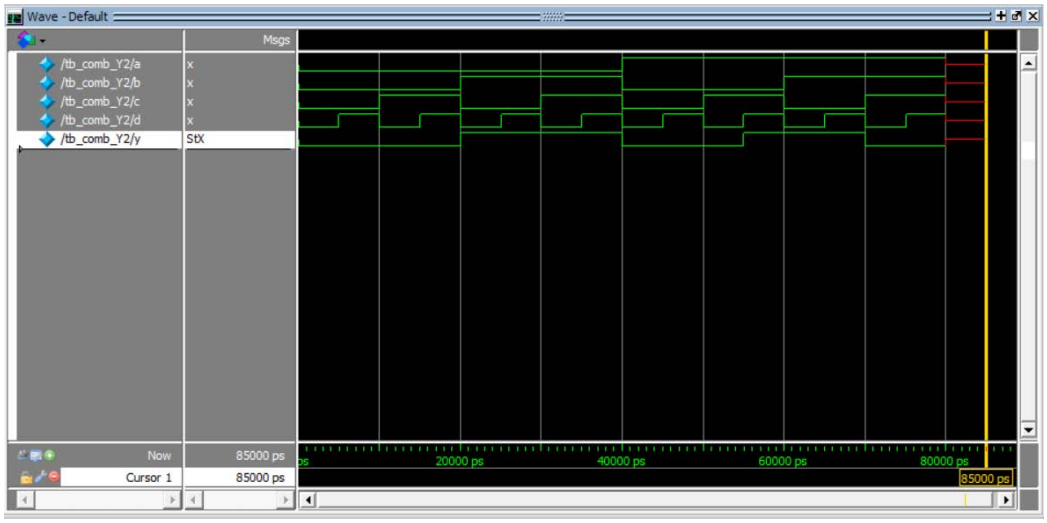
```
    initial
    begin
        {a,b,c,d} = 4'b0;
        for (i = 0; i < 15; i=i+1)
            #5 {a,b,c,d} = {a,b,c,d} + 1;
        #5 {a,b,c,d} = 4'bx;
        #5 $stop;
    End

    initial
        $monitor("At %t, a = %b, b = %b,c = %b,d = %b, y = %b",
                 $time,a,b,c,d,y);
endmodule
```

（3）测试波形图：

布尔表达式 1 共 8 种情况，波形图如下图所示：



布尔表达式 2 共 16 种情况，波形图如下图所示：



（4） 显示输出（可选）：

布尔表达式 1 的测试台模块输出如下图所示：

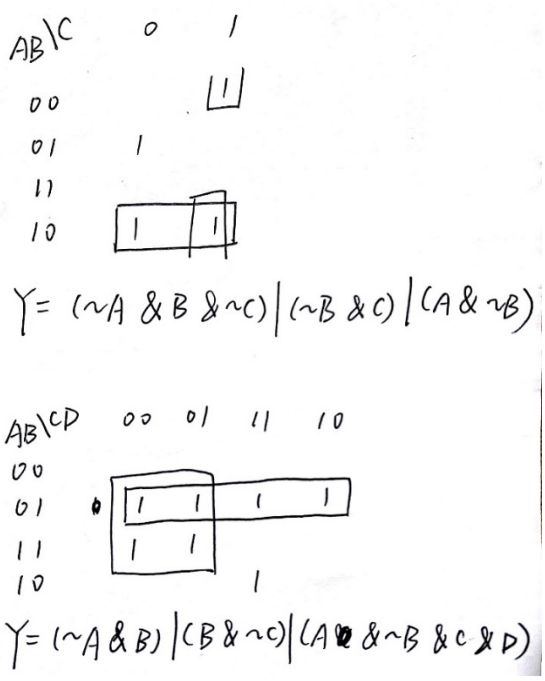
```
# At                0, a = 0, b = 0,c = 0, y = 0
# At               50, a = 0, b = 0,c = 1, y = 1
# At              100, a = 0, b = 1,c = 0, y = 1
# At              150, a = 0, b = 1,c = 1, y = 0
# At              200, a = 1, b = 0,c = 0, y = 1
# At              250, a = 1, b = 0,c = 1, y = 1
# At              300, a = 1, b = 1,c = 0, y = 0
# At              350, a = 1, b = 1,c = 1, y = 0
# At              400, a = x, b = x,c = x, y = x
```

布尔表达式 2 的测试台模块输出如下图所示：

```
# At                0, a = 0, b = 0,c = 0,d = 0, y = 0
# At               50, a = 0, b = 0,c = 0,d = 1, y = 0
# At              100, a = 0, b = 0,c = 1,d = 0, y = 0
# At              150, a = 0, b = 0,c = 1,d = 1, y = 0
# At              200, a = 0, b = 1,c = 0,d = 0, y = 1
# At              250, a = 0, b = 1,c = 0,d = 1, y = 1
# At              300, a = 0, b = 1,c = 1,d = 0, y = 1
# At              350, a = 0, b = 1,c = 1,d = 1, y = 1
# At              400, a = 1, b = 0,c = 0,d = 0, y = 0
# At              450, a = 1, b = 0,c = 0,d = 1, y = 0
# At              500, a = 1, b = 0,c = 1,d = 0, y = 0
# At              550, a = 1, b = 0,c = 1,d = 1, y = 1
# At              600, a = 1, b = 1,c = 0,d = 0, y = 1
# At              650, a = 1, b = 1,c = 0,d = 1, y = 1
# At              700, a = 1, b = 1,c = 1,d = 0, y = 0
# At              750, a = 1, b = 1,c = 1,d = 1, y = 0
# At              800, a = x, b = x,c = x,d = x, y = x
```

（5）设计说明（可选）：

本题的连续赋值表达式使用了卡诺图进行化简：



第 6 题：

（1）设计模块

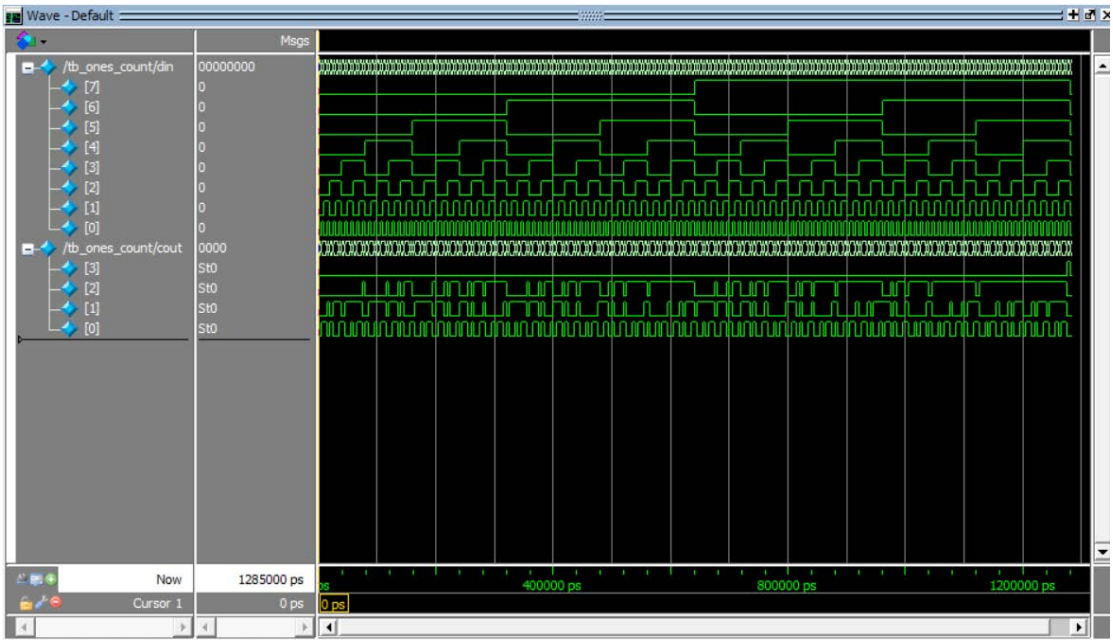
```
module ones_count(output reg [3:0] count, input [7:0] dat_in);
    integer i;
    always @(*)
    begin
        count = 4'b0;
        for (i = 0; i < 8; i=i+1)
            if (dat_in[i] == 1'b1)
                count = count + 1;
    end
endmodule
```

（2）测试模块

```
`include "ones_count.v"
`timescale 1ns / 100ps
module tb_ones_count;
    reg [7:0] din;
    wire [3:0] cout;
    integer i;

    ones_count ct(.count(cout),.dat_in(din));

    initial
    begin
        din = 8'b0;
        for (i = 0; i < 256; i=i+1)
            #5 din = din + 1;
        #5 $stop;
    End

    initial
        $monitor("At %t, dat_in = %b ,count = %b",
                  $time,din,cout);
endmodule
```

（3）测试波形图：

共有 256 种不同情况，测试结果如下：



（4） 显示输出（可选）：

测试中情况过多(共 256 种不同情况)，因篇幅所限，只截取部分显示输出以作说明。具体仿真结果可参考波形仿真图中的结果。

```
# At                    0, dat_in = 00000000 ,count = 0000
# At                   50, dat_in = 00000001 ,count = 0001
# At                  100, dat_in = 00000010 ,count = 0001
# At                  150, dat_in = 00000011 ,count = 0010
# At                  200, dat_in = 00000100 ,count = 0001
# At                  250, dat_in = 00000101 ,count = 0010
# At                  300, dat_in = 00000110 ,count = 0010
# At                  350, dat_in = 00000111 ,count = 0011
# At                  400, dat_in = 00001000 ,count = 0001
# At                  450, dat_in = 00001001 ,count = 0010
# At                  500, dat_in = 00001010 ,count = 0010
# At                  550, dat_in = 00001011 ,count = 0011
# At                  600, dat_in = 00001100 ,count = 0010
# At                  650, dat_in = 00001101 ,count = 0011
# At                  700, dat_in = 00001110 ,count = 0011
# At                  750, dat_in = 00001111 ,count = 0100
# At                  800, dat_in = 00010000 ,count = 0001
# At                  850, dat_in = 00010001 ,count = 0010
# At                  900, dat_in = 00010010 ,count = 0010
# At                  950, dat_in = 00010011 ,count = 0011
# At                 1000, dat_in = 00010100 ,count = 0010
# At                 1050, dat_in = 00010101 ,count = 0011
# At                 1100, dat_in = 00010110 ,count = 0011
# At                 1150, dat_in = 00010111 ,count = 0100
# At                 1200, dat_in = 00011000 ,count = 0010
# At                 1250, dat_in = 00011001 ,count = 0011
# At                 1300, dat_in = 00011010 ,count = 0011
# At                 1350, dat_in = 00011011 ,count = 0100
```

（5）设计说明（可选）：

本次作业中的实现对于带 x/z 的输入依然能统计出 1 的数目。四值测试情况过多，这里只进行二值测试。

第 7 题：

（1）设计模块

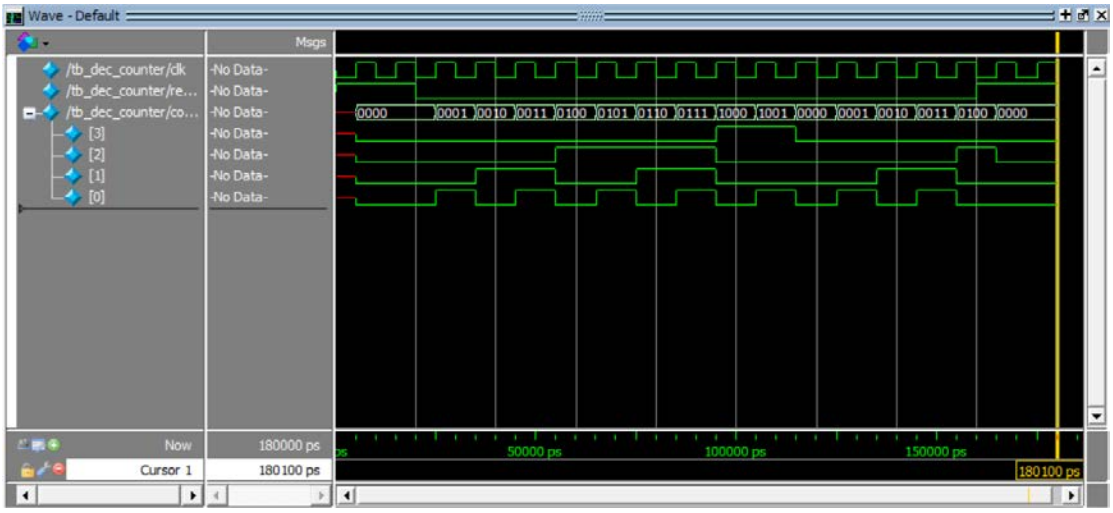
```
module dec_counter(output reg [3:0] count,
                   input clk, reset);
   always@(posedge clk)
   begin
      if (reset)
         count <= 4'b0;
      else if (count == 4'd9)
         count <= 4'b0;
      else
         count <= count + 1;
   end
endmodule
```

（2）测试模块

```
`include "dec_counter.v"
`timescale 1ns / 100ps
module tb_dec_counter;
   reg clk;
   reg reset;
   wire [3:0] count;

   dec_counter dc(.count(count),.clk(clk),.reset(reset));

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

   initial
   begin
      reset = 1;
      #20 reset = 0;
      #140 reset = 1;
```

```
        #20 $stop;
    End

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

（3）测试波形图：



（4） 显示输出（可选）：

```
# At                    0, reset = 1,count = xxxx
# At                   50, reset = 1,count = 0000
# At                  200, reset = 0,count = 0000
# At                  250, reset = 0,count = 0001
# At                  350, reset = 0,count = 0010
# At                  450, reset = 0,count = 0011
# At                  550, reset = 0,count = 0100
# At                  650, reset = 0,count = 0101
# At                  750, reset = 0,count = 0110
# At                  850, reset = 0,count = 0111
# At                  950, reset = 0,count = 1000
# At                 1050, reset = 0,count = 1001
# At                 1150, reset = 0,count = 0000
# At                 1250, reset = 0,count = 0001
# At                 1350, reset = 0,count = 0010
# At                 1450, reset = 0,count = 0011
# At                 1550, reset = 0,count = 0100
# At                 1600, reset = 1,count = 0100
# At                 1650, reset = 1,count = 0000
```

（5）设计说明（可选）：

reset 为同步复位，高电平有效。

第 8 题：

（1）设计模块

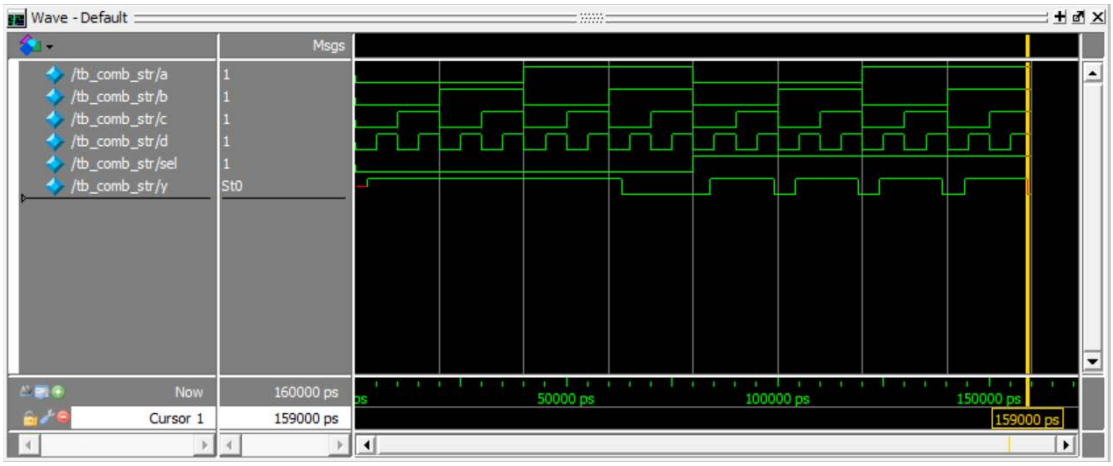
```
`timescale 1ns / 100ps
module comb_str(output y, input sel, A, B, C, D);
    wire in1,in0;
    nand #(3) u1(in0,A,B);
    nand #(4) u2(in1,C,D);
    //mux
    bufif0 b1(y,in0,sel);
    bufif1 b2(y,in1,sel);
endmodule
```

（2）测试模块

```
`timescale 1ns / 100ps
`include "comb_str.v"
module tb_comb_str;
    reg a,b,c,d,sel;
    wire y;
    integer i;

    comb_str cb(.y(y),.sel(sel),.A(a),.B(b),.C(c),.D(d));

    initial
    begin
        {sel,a,b,c,d} = 5'b0;
        for (i = 1; i < 32; i=i+1)
            #5 {sel,a,b,c,d} = {sel,a,b,c,d} + 1;
        #5 $stop;
    End

    initial
        $monitor("At %t,{a,b,c,d} = %b,sel = %b, y = %b",
                $time,{a,b,c,d},sel,y);
endmodule
```

（3）测试波形图：

共有 32 种不同的情况，仿真结果如下：



（4） 显示输出（可选）：

```
# At                    0,{a,b,c,d} = 0000,sel = 0, y = x
# At                   30,{a,b,c,d} = 0000,sel = 0, y = 1
# At                   50,{a,b,c,d} = 0001,sel = 0, y = 1
# At                  100,{a,b,c,d} = 0010,sel = 0, y = 1
# At                  150,{a,b,c,d} = 0011,sel = 0, y = 1
# At                  200,{a,b,c,d} = 0100,sel = 0, y = 1
# At                  250,{a,b,c,d} = 0101,sel = 0, y = 1
# At                  300,{a,b,c,d} = 0110,sel = 0, y = 1
# At                  350,{a,b,c,d} = 0111,sel = 0, y = 1
# At                  400,{a,b,c,d} = 1000,sel = 0, y = 1
# At                  450,{a,b,c,d} = 1001,sel = 0, y = 1
# At                  500,{a,b,c,d} = 1010,sel = 0, y = 1
# At                  550,{a,b,c,d} = 1011,sel = 0, y = 1
# At                  600,{a,b,c,d} = 1100,sel = 0, y = 1
# At                  630,{a,b,c,d} = 1100,sel = 0, y = 0
# At                  650,{a,b,c,d} = 1101,sel = 0, y = 0
# At                  700,{a,b,c,d} = 1110,sel = 0, y = 0
# At                  750,{a,b,c,d} = 1111,sel = 0, y = 0
# At                  800,{a,b,c,d} = 0000,sel = 1, y = 0
# At                  840,{a,b,c,d} = 0000,sel = 1, y = 1
# At                  850,{a,b,c,d} = 0001,sel = 1, y = 1
# At                  900,{a,b,c,d} = 0010,sel = 1, y = 1
# At                  950,{a,b,c,d} = 0011,sel = 1, y = 1
# At                  990,{a,b,c,d} = 0011,sel = 1, y = 0
# At                 1000,{a,b,c,d} = 0100,sel = 1, y = 0
```

```
# At                 1040,{a,b,c,d} = 0100,sel = 1, y = 1
# At                 1050,{a,b,c,d} = 0101,sel = 1, y = 1
# At                 1100,{a,b,c,d} = 0110,sel = 1, y = 1
# At                 1150,{a,b,c,d} = 0111,sel = 1, y = 1
# At                 1190,{a,b,c,d} = 0111,sel = 1, y = 0
# At                 1200,{a,b,c,d} = 1000,sel = 1, y = 0
# At                 1240,{a,b,c,d} = 1000,sel = 1, y = 1
# At                 1250,{a,b,c,d} = 1001,sel = 1, y = 1
# At                 1300,{a,b,c,d} = 1010,sel = 1, y = 1
# At                 1350,{a,b,c,d} = 1011,sel = 1, y = 1
# At                 1390,{a,b,c,d} = 1011,sel = 1, y = 0
# At                 1400,{a,b,c,d} = 1100,sel = 1, y = 0
# At                 1440,{a,b,c,d} = 1100,sel = 1, y = 1
# At                 1450,{a,b,c,d} = 1101,sel = 1, y = 1
# At                 1500,{a,b,c,d} = 1110,sel = 1, y = 1
# At                 1550,{a,b,c,d} = 1111,sel = 1, y = 1
# At                 1590,{a,b,c,d} = 1111,sel = 1, y = 0
```

（5）设计说明（可选）：

本题的特点是含有门延时。因而，在测试模块中，数据值改变的间隔应大于延迟，使得门延时的仿真结果可以观察。门延时在所选的几个特例的波形测试图中得到了体现。

从波形测试图上可以看出，用两个 buf 的设计很好地实现了 mux 的功能。

第 9 题：

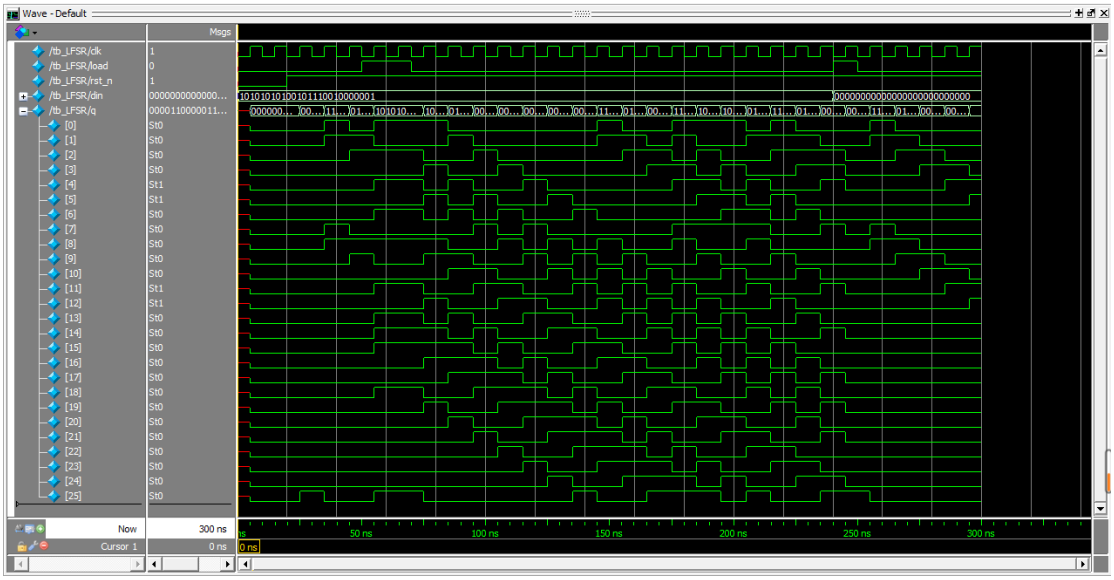
（1）设计模块

```
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.
        );
   always@(posedge clk)
   begin
   if (~rst_n)
      q <= 26'b0;
   else
      begin
      if (load)
         q <= (|din)? din:26'b1;
      else
         begin
         if (q == 26'b0)
            q <= 26'b1;
         else
            begin
            q[10:26] <= q[9:25];
            q[9] <= q[8]^q[26];//x8
            q[8] <= q[7]^q[26];//x7
            q[3:7] <= q[2:6];
            q[2] <= q[1]^q[26];//x
            q[1] <= q[26];
            end
         end
      end
   end
endmodule
```

（2）测试模块

```
`include "LFSR.v"
`timescale 1ns / 1ns
module tb_LFSR;
    reg clk,load,rst_n;
    reg [0:25] din;
    wire [0:25] q;

    LFSR lfsr(.q(q),.clk(clk),.rst_n(rst_n),.load(load),.din(din));

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

    initial
    begin
        rst_n = 1'b0;
        #20  rst_n = 1'b1;
    end

    initial
    begin
        load = 1'b0;
        din = 26'b10_1010_1010_0101_1100_1000_0001;

        #50 load = 1'b1;
        #20 load = 1'b0;

        #170 load = 1'b1;
        din = 26'b0;
        #10 load = 1'b0;
        #50 $stop;
    end

    initial
        $monitor("At time t=%4t, rst_n=%b, load=%b, din=%d, q=%d",
                  $time,rst_n,load,din,q);
endmodule
```

（3）测试波形图：



rst_n 在开始时为 0，输出 q 的确稳定为 0；

在 20 时刻 rst_n 变为 1，因为 rst_n 为同步复位信号，又因为代码中清除全 0（生成随机数时不允许全 0 情况），在 25 时刻时钟上升沿 q 变为了 1；

load 在 50 时刻变为有效，在 55 时钟上升沿的时候，将 din 中的值赋值到了 q 中

70 时刻 load 变 0，移位寄存器正常运作；

240 时刻 load 再次为 1，在时钟上升沿的 245 时刻，因为 din 全 0，赋值给 q 的是 1；

此后 load 变 0，再次照常产生随机数。

（4） 显示输出（可选）：

```
# At time t=   0, rst_n=0, load=0, din=44719233, q=       x
# At time t=   5, rst_n=0, load=0, din=44719233, q=       0
# At time t=  20, rst_n=1, load=0, din=44719233, q=       0
# At time t=  25, rst_n=1, load=0, din=44719233, q=       1
# At time t=  35, rst_n=1, load=0, din=44719233, q=50724864
# At time t=  45, rst_n=1, load=0, din=44719233, q=25362432
# At time t=  50, rst_n=1, load=1, din=44719233, q=25362432
# At time t=  55, rst_n=1, load=1, din=44719233, q=44719233
# At time t=  70, rst_n=1, load=0, din=44719233, q=44719233
# At time t=  75, rst_n=1, load=0, din=44719233, q=39005760
# At time t=  85, rst_n=1, load=0, din=44719233, q=19502880
# At time t=  95, rst_n=1, load=0, din=44719233, q= 9751440
# At time t= 105, rst_n=1, load=0, din=44719233, q= 4875720
# At time t= 115, rst_n=1, load=0, din=44719233, q= 2437860
# At time t= 125, rst_n=1, load=0, din=44719233, q= 1218930
# At time t= 135, rst_n=1, load=0, din=44719233, q=  609465
# At time t= 145, rst_n=1, load=0, din=44719233, q=50505308
# At time t= 155, rst_n=1, load=0, din=44719233, q=25252654
# At time t= 165, rst_n=1, load=0, din=44719233, q=12626327
# At time t= 175, rst_n=1, load=0, din=44719233, q=57038027
# At time t= 185, rst_n=1, load=0, din=44719233, q=45427301
# At time t= 195, rst_n=1, load=0, din=44719233, q=39621938
# At time t= 205, rst_n=1, load=0, din=44719233, q=19810969
# At time t= 215, rst_n=1, load=0, din=44719233, q=59843916
# At time t= 225, rst_n=1, load=0, din=44719233, q=29921958
# At time t= 235, rst_n=1, load=0, din=44719233, q=14960979
# At time t= 240, rst_n=1, load=1, din=       0, q=14960979
# At time t= 245, rst_n=1, load=1, din=       0, q=       1
# At time t= 250, rst_n=1, load=0, din=       0, q=       1
# At time t= 255, rst_n=1, load=0, din=       0, q=50724864
# At time t= 265, rst_n=1, load=0, din=       0, q=25362432
# At time t= 275, rst_n=1, load=0, din=       0, q=12681216
# At time t= 285, rst_n=1, load=0, din=       0, q= 6340608
# At time t= 295, rst_n=1, load=0, din=       0, q= 3170304
```

第 10 题：

（1）设计模块

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

   reg [0:3] Q;
   wire J,K;

   assign J = Q[1] & Q[2] & Q[3];
   assign K = (~Q[1]) & (~Q[2]) & (~Q[3]);

   always @(posedge clock)
   begin
      if (!reset) begin
         Q <= 4'b0;
         sig_out <= 1'b0;
```

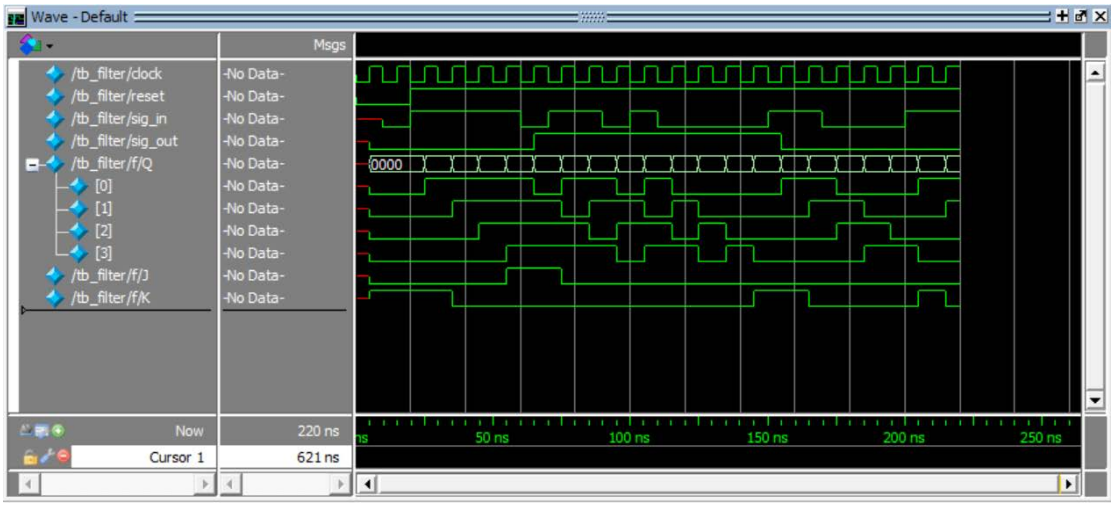
```
        end
        else begin
            Q[0:3] <= {sig_in, Q[0:2]};
            case({J,K})
                2'b10: sig_out<=1'b1;
                2'b01: sig_out<=1'b0;
                2'b11: sig_out<=~sig_out;
                default:  ;
            endcase
        end
    end
endmodule
```

（2）测试模块

```
`include "filter.v"
`timescale 1ns/1ns
module tb_filter;
    reg clock,reset,sig_in;
    wire sig_out;

    filter f
(.sig_out(sig_out), .clock(clock), .reset(reset), .sig_in(sig_in));

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

    reg [19:0] inflow;
    integer i;
    initial
    begin
        inflow = 20'b1000_1100_0010_1101_1110;
        for(i=0; i<20; i=i+1)
            #10 sig_in = inflow[i];
    end

    initial
    begin
        reset = 1'b0;
        #20 reset = 1'b1;
```

```
        #200 $stop;
    end

    initial
        $monitor("At %t,reset = %b,sig_in = %b,sig_out = %b",
                $time,reset,sig_in,sig_out);
Endmodule
```

（3）测试波形图：



对波形图进行分析：

在前 20ns 中，由于 reset 为 0，sig_out 为 0；

200 时，reset 变为 1，sig_in 开始生效：四个 D 触发器在上升沿进行四位同步移位，其中第一个触发器是接收 sig_in； J、K 跟随第 234 个触发器的 Q 端不断变化；

根据 JK 触发器的公式，在 65 时刻的时钟上升沿，由于 J=1，K=0，sig_out 变 1；

在 155 的时钟上升沿，由于 J=0，K=1，sig_out 再次变为 0。

（4） 显示输出（可选）：

```
# At                   0,reset = 0,sig_in = x,sig_out = x
# At                   5,reset = 0,sig_in = x,sig_out = 0
# At                  10,reset = 0,sig_in = 0,sig_out = 0
# At                  20,reset = 1,sig_in = 1,sig_out = 0
# At                  60,reset = 1,sig_in = 0,sig_out = 0
# At                  65,reset = 1,sig_in = 0,sig_out = 1
# At                  70,reset = 1,sig_in = 1,sig_out = 1
# At                  90,reset = 1,sig_in = 0,sig_out = 1
# At                 100,reset = 1,sig_in = 1,sig_out = 1
# At                 110,reset = 1,sig_in = 0,sig_out = 1
# At                 150,reset = 1,sig_in = 1,sig_out = 1
# At                 155,reset = 1,sig_in = 1,sig_out = 0
# At                 170,reset = 1,sig_in = 0,sig_out = 0
# At                 200,reset = 1,sig_in = 1,sig_out = 0
```

（5）设计说明（可选）：

Reset 采用了同步复位，低电平有效。

第 11 题：

（1）设计模块

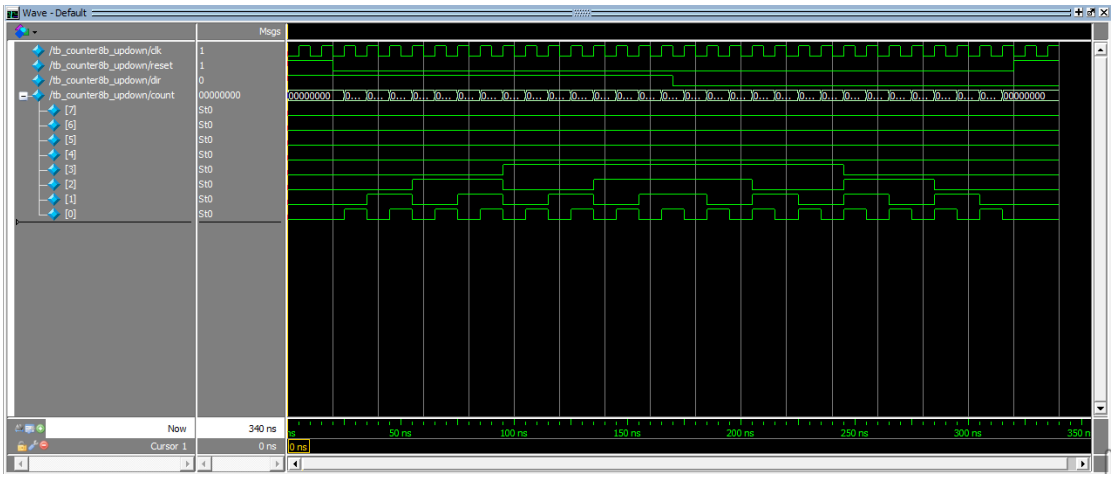
```
module counter8b_updown(output reg[7:0] count,
                input clk, reset, dir);
   always@(posedge clk or posedge reset)
   begin
      if (reset)
         count <= 8'b0;
      else if (dir)
         count <= count + 1;
      else
         count <= count - 1;
   end
endmodule
```

（2）测试模块

```
`include "counter8b_updown.v"
`timescale 1ns/ 1ns
module tb_counter8b_updown;
   reg clk,reset,dir;
   wire [7:0] count;
   counter8b_updown
m(.count(count),.clk(clk),.reset(reset),.dir(dir));

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

   initial
   begin
      reset = 1;
      dir = 1;
      #20 reset = 0;
      #150 dir = 0;
```

```
        #150 reset = 1;
        #20 $stop;
    end

    initial
        $monitor("At %t, reset=%b, dir=%b, count=%d",
                $time,reset,dir,count);
endmodule
```

（3）测试波形图：



在本题中 reset 为异步复位。

200ns 之前 reset 均为 1，count 均为 0；

20ns，reset 变为 1，则从下一个时钟上升沿，即 25ns 时，开始计数。因为 dir=1，所以进行加法；

170ns 时，dir 变为 0，此时 count=10，从下一个时钟上升沿 175ns 时开始进行减法

320ns 时，reset=0，异步复位为 0。

（4） 显示输出（可选）：

```
# At                   0, reset=1, dir=1, count=  0
# At                  20, reset=0, dir=1, count=  0
# At                  25, reset=0, dir=1, count=  1
# At                  35, reset=0, dir=1, count=  2
# At                  45, reset=0, dir=1, count=  3
# At                  55, reset=0, dir=1, count=  4
# At                  65, reset=0, dir=1, count=  5
# At                  75, reset=0, dir=1, count=  6
# At                  85, reset=0, dir=1, count=  7
# At                  95, reset=0, dir=1, count=  8
# At                 105, reset=0, dir=1, count=  9
# At                 115, reset=0, dir=1, count= 10
# At                 125, reset=0, dir=1, count= 11
# At                 135, reset=0, dir=1, count= 12
# At                 145, reset=0, dir=1, count= 13
# At                 155, reset=0, dir=1, count= 14
# At                 165, reset=0, dir=1, count= 15
# At                 170, reset=0, dir=0, count= 15
# At                 175, reset=0, dir=0, count= 14
# At                 185, reset=0, dir=0, count= 13
# At                 195, reset=0, dir=0, count= 12
# At                 205, reset=0, dir=0, count= 11
# At                 215, reset=0, dir=0, count= 10
# At                 225, reset=0, dir=0, count=  9
# At                 235, reset=0, dir=0, count=  8
# At                 245, reset=0, dir=0, count=  7
# At                 255, reset=0, dir=0, count=  6
# At                 265, reset=0, dir=0, count=  5
# At                 275, reset=0, dir=0, count=  4
# At                 285, reset=0, dir=0, count=  3
# At                 295, reset=0, dir=0, count=  2
# At                 305, reset=0, dir=0, count=  1
# At                 315, reset=0, dir=0, count=  0
# At                 320, reset=1, dir=0, count=  0
```

（5）设计说明（可选）：

本题中比较特别的地方是：reset 为异步复位，1 有效，reset 变为 1 时的时钟上升沿会被视为无效，不会引起计数。

第 12 题：

（1）设计模块

```
module ALU( output reg c_out,
          output reg[7:0] sum,
          input [2:0] oper,
          input [7:0] a, b,
          input c_in );
   parameter iand = 3'b000,
             subtract = 3'b001,
             subtract_a = 3'b010,
             or_ab = 3'b011,
             and_ab = 3'b100,
```

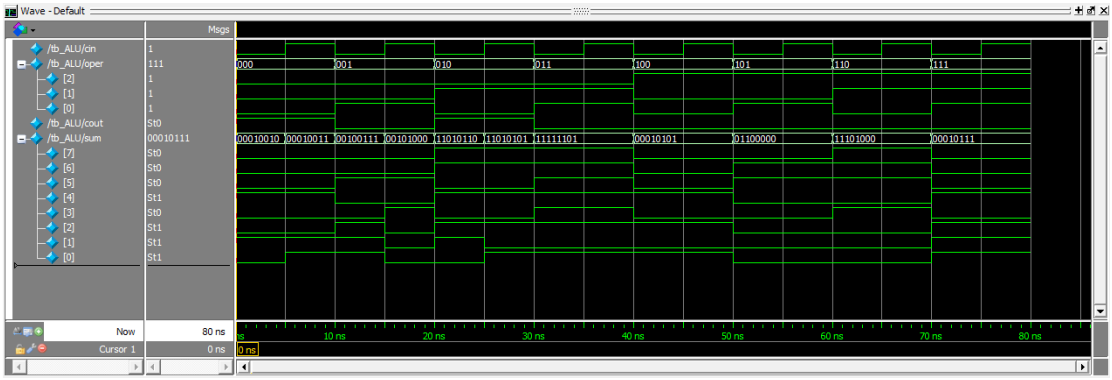
```
            not_ab = 3'b101,
            exor = 3'b110,
            exnor = 3'b111;
    always@(*)
    begin
        case(oper)
            iand:  {c_out,sum} = a + b + c_in;
            subtract:  {c_out,sum} = a + ~b + c_in;
            subtract_a: {c_out,sum} = b + ~a + ~c_in;
            or_ab:  {c_out,sum} = {1'b0, a | b };
            and_ab:  {c_out,sum} = { 1'b0, a & b };
            not_ab:  {c_out,sum} = { 1'b0, (~a) & b };
            exor:  {c_out,sum} = { 1'b0, a ^ b };
            exnor:  {c_out,sum} = { 1'b0, a ~^ b };
            default: {c_out,sum} = 9'bx;
        endcase
    end
endmodule
```

（2）测试模块

```
`timescale 1ns/1ns
`include "ALU.v"
module tb_ALU;
    reg [7:0] a,b;
    reg cin;
    reg [2:0] oper;
    wire cout;
    wire [7:0] sum;
    integer i;

    ALU
alu(.sum(sum),.c_out(cout),.c_in(cin),.a(a),.b(b),.oper(oper));

    initial
    begin
        {oper,cin} = 4'b0;
        a = 8'b1001_1101;
        b = 8'b0111_0101;
        $display("a = %b, b = %b",a,b);
        for (i = 0; i < 15; i=i+1)
            #5 {oper,cin} = {oper,cin} + 1;
    end
```

```
    initial
        $monitor("At %t ns, oper = %b, cin = %b, cout = %b, sum = %b",
                $time, oper, cin, cout, sum);
endmodule
```

（3）测试波形图：



（4） 显示输出（可选）：

```
# a = 10011101, b = 01110101
# At                    0 ns, oper = 000, cin = 0, cout = 1, sum = 00010010
# At                    5 ns, oper = 000, cin = 1, cout = 1, sum = 00010011
# At                   10 ns, oper = 001, cin = 0, cout = 0, sum = 00100111
# At                   15 ns, oper = 001, cin = 1, cout = 0, sum = 00101000
# At                   20 ns, oper = 010, cin = 0, cout = 1, sum = 11010110
# At                   25 ns, oper = 010, cin = 1, cout = 1, sum = 11010101
# At                   30 ns, oper = 011, cin = 0, cout = 0, sum = 11111101
# At                   35 ns, oper = 011, cin = 1, cout = 0, sum = 11111101
# At                   40 ns, oper = 100, cin = 0, cout = 0, sum = 00010101
# At                   45 ns, oper = 100, cin = 1, cout = 0, sum = 00010101
# At                   50 ns, oper = 101, cin = 0, cout = 0, sum = 01100000
# At                   55 ns, oper = 101, cin = 1, cout = 0, sum = 01100000
# At                   60 ns, oper = 110, cin = 0, cout = 0, sum = 11101000
# At                   65 ns, oper = 110, cin = 1, cout = 0, sum = 11101000
# At                   70 ns, oper = 111, cin = 0, cout = 0, sum = 00010111
# At                   75 ns, oper = 111, cin = 1, cout = 0, sum = 00010111
```

（5）设计说明（可选）：

本题若考虑 a、b 的所有取值，会使得情况过多。因此，我在这里选择固定了 a 与 b，仅仅考虑变化 cin 以及 oper；使用 parameter 定义操作码时，经同学提醒，and 为保留字，根据老师回答可以用其他名称代替。

此外，subtract 与减法的定义不相同，而且直接按照给出的表达式有歧义，例如 a=0，b=0，c=0 时，如果左操作数为 1 位，则结果为 1'b1，如果左操作数为 2 位，则结果为 2'b11。经过与老师的讨论，我最终仅仅是按功能的定义进行了代码编写，而没有按减法的定义去做。

第 13 题：

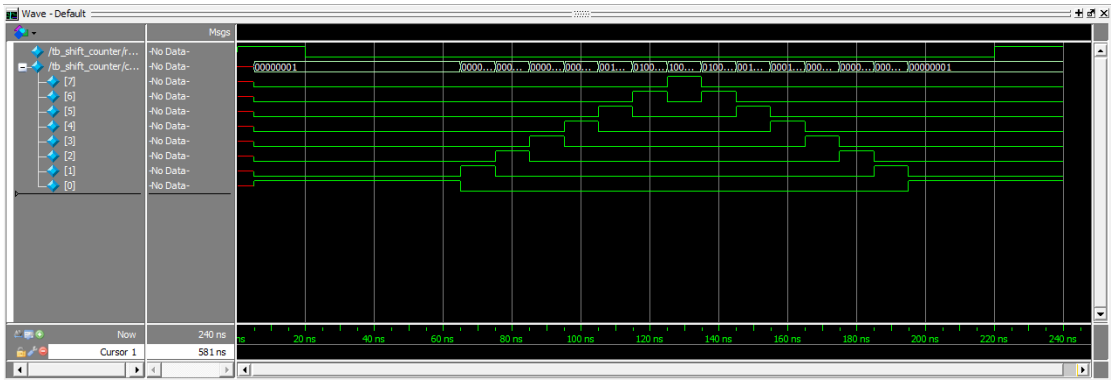
（1）设计模块

```
module shift_counter(count, clk, reset);
    output reg [7:0] count;
    input clk;
    input reset;
    reg [4:0] cnt;

    always @( posedge clk )
    begin
        if (reset)
            cnt <= 5'b0;
        else begin
            if (cnt < 17)
                cnt <= cnt + 1'b1;
            else
                cnt <= 5'b0;
        end

        case (cnt)
            0: count = 8'b0000_0001;
            1: count = 8'b0000_0001;
            2: count = 8'b0000_0001;
            3: count = 8'b0000_0001;
            4: count = 8'b0000_0010;
            5: count = 8'b0000_0100;
            6: count = 8'b0000_1000;
            7: count = 8'b0001_0000;
            8: count = 8'b0010_0000;
            9: count = 8'b0100_0000;
            10: count = 8'b1000_0000;
            11: count = 8'b0100_0000;
            12: count = 8'b0010_0000;
            13: count = 8'b0001_0000;
            14: count = 8'b0000_1000;
            15: count = 8'b0000_0100;
            16: count = 8'b0000_0010;
            17: count = 8'b0000_0001;
            default:count = 8'b0000_0001;
        endcase
    end
endmodule
```

（2）测试模块

```
`include "shift_counter.v"
`timescale 1ns /1ns
module tb_shift_counter;
    reg clk, reset;
    wire [7:0] count;
    shift_counter sc(.count(count),.clk(clk),.reset(reset));

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

    initial
    begin
        reset = 1;
        #20 reset = 0;
        #200 reset = 1;
        #20 $stop;
    end

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

（3）测试波形图：



（4） 显示输出（可选）：

```
# At                    0 ns, reset=1, count=xxxxxxxx
# At                    5 ns, reset=1, count=00000001
# At                   20 ns, reset=0, count=00000001
# At                   65 ns, reset=0, count=00000010
# At                   75 ns, reset=0, count=00000100
# At                   85 ns, reset=0, count=00001000
# At                   95 ns, reset=0, count=00010000
# At                  105 ns, reset=0, count=00100000
# At                  115 ns, reset=0, count=01000000
# At                  125 ns, reset=0, count=10000000
# At                  135 ns, reset=0, count=01000000
# At                  145 ns, reset=0, count=00100000
# At                  155 ns, reset=0, count=00010000
# At                  165 ns, reset=0, count=00001000
# At                  175 ns, reset=0, count=00000100
# At                  185 ns, reset=0, count=00000010
# At                  195 ns, reset=0, count=00000001
# At                  220 ns, reset=1, count=00000001
```

第 14 题：

（1）设计模块

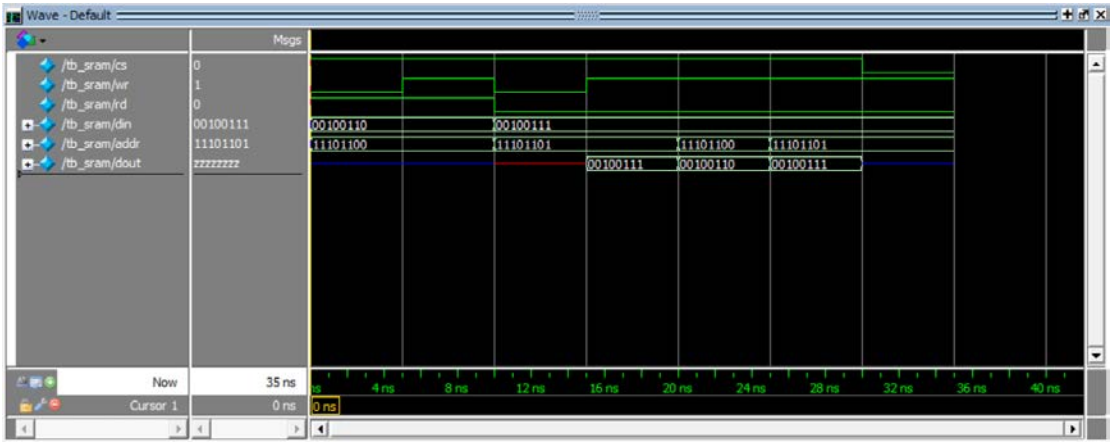
```
module sram(output [7:0] dout,
        input [7:0] din, addr,
        input wr, rd, cs );
    reg [7:0] ram [0:255];

    assign dout = (cs & !rd)? ram[addr]:8'bz;

    always@(posedge wr)
       if (cs)
          ram[addr] <= din;
endmodule
```

（2）测试模块

```
`timescale 1ns/1ns
`include "sram.v"
module tb_sram;
    reg cs,wr,rd;
    reg [7:0] din, addr;
    wire [7:0] dout;

    sram
s(.dout(dout),.din(din),.addr(addr),.wr(wr),.rd(rd),.cs(cs));
    initial
    begin
```

```
        cs = 1'b1;
        rd = 1'b1;
        wr = 1'b0;

        addr = 8'b1110_1100;
        din = 8'b0010_0110;
        #5 wr = 1'b1;
        #5 wr = 1'b0;
        rd = 1'b0;
        addr = 8'b1110_1101;
        din = 8'b0010_0111;
        #5 wr = 1'b1;

        #5 addr = 8'b1110_1100;
        #5 addr = 8'b1110_1101;

        #5 cs = 1'b0;
        #5 $stop;
    end

    initial
        $monitor("At %t ns,cs=%b,wr=%b,rd=%b,addr=%b,din=%b,dout=%b",
                 $time,cs,wr,rd,addr,din,dout);
Endmodule
```

（3）测试波形图：



这里对仿真波形进行一定的分析和说明。

在初始时刻 cs=1,wr=0,rd=1，显然不能写也不能读。

先看读信号。在 5ns 时刻，wr 出现一个上跳沿(cs=1,rd=1)，将 ram[11101100]赋值为 00100110。在 10ns 时刻 wr 回归为 0，且地址由 11101100 变为 11101101。在 15ns 时迎来第二个上跳沿(cs=1,rd=0)，将 ram[11101101]赋值为 00100111。

再看写信号。在 10ns 时刻，rd=0，此时 ram[11101101]中并没有数据，读取结果为不定

态。20ns 时，在 dout 读出 ram[11101100]为 00100110。在 25ns 时，在 dout 读出 ram[11101101]为 00100111。

最后，在 300ns 时，cs=0，不能读也不能写，dout 也变为高阻。

测试结果值得注意的一点是，从测试结果中（10ns-20ns）可以看出，我们设计的 SRAM 模块是可以同时进行读和写操作的。

（4） 显示输出（可选）：

```
# At                  0 ns,cs=1,wr=0,rd=1,addr=11101100,din=00100110,dout=zzzzzzzz
# At                  5 ns,cs=1,wr=1,rd=1,addr=11101100,din=00100110,dout=zzzzzzzz
# At                 10 ns,cs=1,wr=0,rd=0,addr=11101101,din=00100111,dout=xxxxxxxx
# At                 15 ns,cs=1,wr=1,rd=0,addr=11101101,din=00100111,dout=00100111
# At                 20 ns,cs=1,wr=1,rd=0,addr=11101100,din=00100111,dout=00100110
# At                 25 ns,cs=1,wr=1,rd=0,addr=11101101,din=00100111,dout=00100111
# At                 30 ns,cs=0,wr=1,rd=0,addr=11101101,din=00100111,dout=zzzzzzzz
```

第 15 题：

（1）设计模块

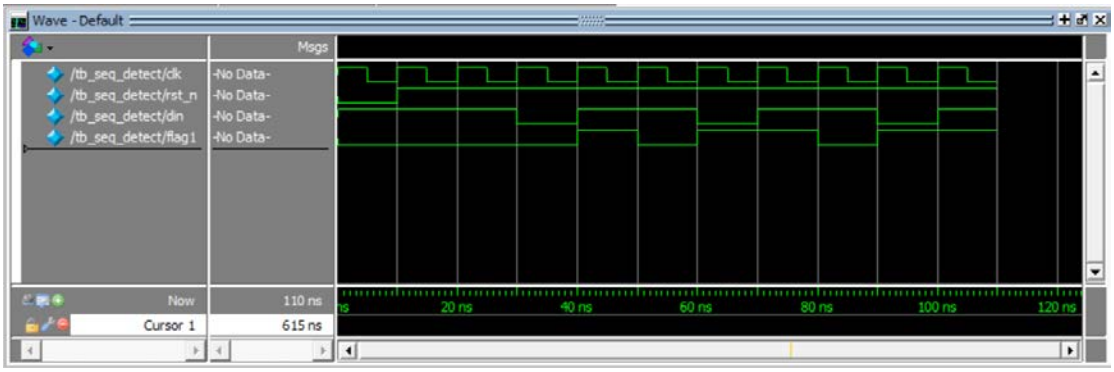
```
module seq_detect(output reg flag,
                  input din, clk, rst_n);
   parameter IDLE = 7'b1000000, A = 7'b0100000,
             B = 7'b0010000, C = 7'b0001000, D = 7'b0000100,
             E = 7'b0000010, F = 7'b0000001;
   reg [6:0] state;
   always@(posedge clk)
   begin
      if (!rst_n) begin state <= IDLE; flag<=1'b0; end
      else  begin
         case(state)
         IDLE: if (din) begin state <= B; flag <= 1'b0; end
               else  begin state <= A; flag <= 1'b0; end
         A: if (din) begin state <= C; flag <= 1'b0; end
               else  begin state <= A; flag <= 1'b0; end
         B: if (din) begin state <= D; flag <= 1'b0; end
               else  begin state <= A; flag <= 1'b0; end
         C: if (din) begin state <= E; flag <= 1'b0; end
               else  begin state <= A; flag <= 1'b0; end
         D: if (din) begin state <= D; flag <= 1'b0; end
               else  begin state <= F; flag <= 1'b0; end
         E: if (din) begin state <= D; flag <= 1'b0; end
               else  begin state <= F; flag <= 1'b1; end
         F: if (din) begin state <= C; flag <= 1'b1; end
               else  begin state <= A; flag <= 1'b0; end
```

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

（2）测试模块

```
`timescale 1ns/1ns
`include "seq_detect.v"
module tb_seq_detect;
    reg clk,rst_n,din;
    reg[9:0] inflow;
    wire flag1;
    integer i;

    seq_detect sd(.flag(flag1),.clk(clk),.rst_n(rst_n),.din(din));

    initial
    begin
        clk = 0'b1;
        forever #5 clk=~clk;
    end

    initial
    begin
        inflow = 10'b11_0110_1101;
        din = inflow[9];
        rst_n = 1'b0;
        #10 rst_n = 1'b1;
        for (i=9;i>0;i=i-1)
        begin
            #10 din = inflow[i-1];
        end
        #10 $stop;
    end

    initial
        $monitor("At%tns,rst_n=%b,din=%b,flag1=%b",
                $time,rst_n,din,flag1);
endmodule
```

（3）测试波形图：



这里对波形进行一定的分析：

本题的测试序列使用了一个非常复杂的情况：1101101101

可以看出，前四位 1101、三到六位 0110、四到七位 1101、六到九位 0110、七到十位 1101，都是需要检测的序列。从仿真结果中可以看出，flag 在 40ns、60ns、70ns、90ns、100ns 五个时刻都输出了 1，即检测到指定序列。证明设计的模块经过了仿真验证。

（4） 显示输出（可选）：

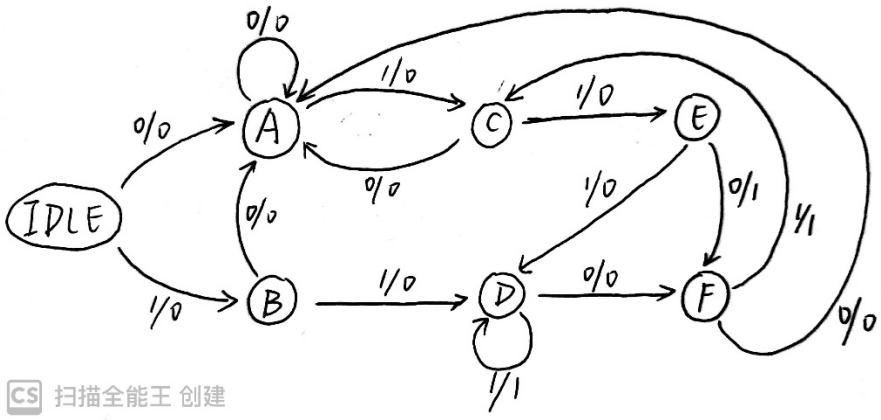
```
# At                 0ns,rst_n=0,din=1,flag1=0
# At                10ns,rst_n=1,din=1,flag1=0
# At                30ns,rst_n=1,din=0,flag1=0
# At                40ns,rst_n=1,din=1,flag1=1
# At                50ns,rst_n=1,din=1,flag1=0
# At                60ns,rst_n=1,din=0,flag1=1
# At                70ns,rst_n=1,din=1,flag1=1
# At                80ns,rst_n=1,din=1,flag1=0
# At                90ns,rst_n=1,din=0,flag1=1
# At               100ns,rst_n=1,din=1,flag1=1
```

（5）设计说明（可选）：

需要说明的是，本题需要能识别出 1101 和 0110 两种序列，情况是非常复杂的。在状态转换图的设计过程中，我遇到了很多需要考虑的特例。最终经过反复的修改，我借鉴《编译原理》中识别字符串活前缀的思想，设计出了状态转换图。设计思路如下：

1. 要识别的序列为 0110,1101
2. 0110 的活前缀有三个，为：0,01,011
3. 1101 的活前缀有三个，为：1,11,110
4. 根据各个活前缀，依次定义各个状态为：A(0),B(1),C(01),D(11),E(011),F(110)

画出的状态转移图如下所示：



第 16 题：

（1）设计模块

```
module mealy( output reg flag, input din, clk, rst );
   parameter IDLE = 8'b00000001, A = 8'b00000010,
             B = 8'b00000100, C = 8'b00001000,
             D = 8'b00010000, E = 8'b00100000,
             F = 8'b01000000, G = 8'b10000000;
   reg [7:0] state;
   always@(posedge clk or posedge rst)
   begin
      if (rst) begin state <= IDLE;flag <= 1'b0;end
      else begin
         case(state)
         IDLE: if (din) begin state <= IDLE; flag <= 1'b0; end
                 else begin state <= A; flag <= 1'b0; end
         A: if (din) begin state <= B; flag <= 1'b0; end
                 else begin state <= A; flag <= 1'b0; end
         B: if (din) begin state <= IDLE; flag <= 1'b0; end
                 else begin state <= C; flag <= 1'b0; end
         C: if (din) begin state <= D; flag <= 1'b0; end
                 else begin state <= A; flag <= 1'b0; end
         D: if (din) begin state <= IDLE; flag <= 1'b0; end
                 else begin state <= E; flag <= 1'b0; end
         E: if (din) begin state <= F; flag <= 1'b0; end
                 else begin state <= A; flag <= 1'b0; end
         F: if (din) begin state <= IDLE; flag <= 1'b0; end
                 else begin state <= G; flag <= 1'b0; end
         G: if (din) begin state <= F; flag <= 1'b1; end
                 else begin state <= A; flag <= 1'b0; end
         default: begin state <= IDLE;flag <= 1'b0; end
         endcase
      end
   end
endmodule

module moore( output reg flag, input din, clk,rst);
   parameter IDLE = 9'b000000001, A = 9'b000000010,
             B = 9'b000000100, C = 9'b000001000,
             D = 9'b000010000, E = 9'b000100000,
             F = 9'b001000000, G = 9'b010000000,
             H = 9'b100000000;
   reg [8:0] state;
```

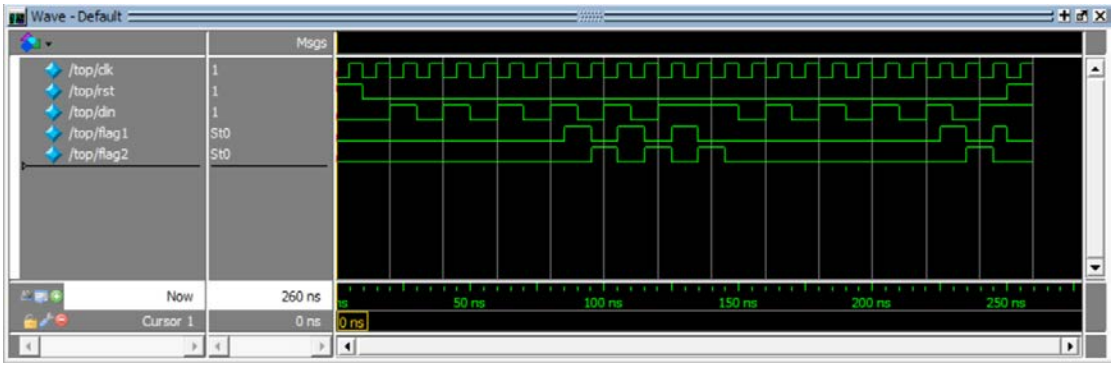
```
    always@(posedge clk or posedge rst)
    begin
        if (rst)  begin state <= IDLE; flag <= 1'b0; end
        else begin
            if (state == H) flag <= 1'b1;
            else flag <= 1'b0;
            case(state)
            IDLE: state <= (din)?IDLE:A;
            A: state <= (din)?B:A;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

（2）测试模块

```
`include "mealy.v"
`include "moore.v"
`timescale 1ns / 1ns
module top;
    reg clk,rst,din;
    reg[23:0] inflow;
    wire flag1,flag2;
    integer i;

    mealy ml(.flag(flag1),.clk(clk),.rst(rst),.din(din));
    moore mo(.flag(flag2),.clk(clk),.rst(rst),.din(din));

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

    initial
    begin
```

```
        inflow = 40'b0101_0101_0101_1101_0101_0101;
        din = inflow[23];
        rst = 1'b1;
        #10 rst = 1'b0;
        for (i=23;i>0;i=i-1)
        begin
            #10 din = inflow[i-1];
        end
        #10 rst = 1'b1;
        #10 $stop;
    end
    initial
        $monitor("At%t ns,rst=%b,din=%b,flag1=%b,flag2=%b",
                $time,rst,din,flag1,flag2);
Endmodule
```

（3）测试波形图：



这里，对测试波形进行分析：

1. 在 10ns 复位信号 rst 降为 0 后，在 85ns 时刻的时钟上升沿出现第一个 10101010 序列，使得 Mealy 机输出 1；之后的 105ns、125ns，继续检测到 10101010 序列，Mealy 输出 1；
2. 之后，由于连续输入两个 1，使得序列要重新检测。
3. 225ns 时的时钟上跳沿，再次检测到 10101010 序列，Mealy 机输出 1；此后，245ns 时刻，继续检测到 10101010 序列，Mealy 输出 1。

在这个过程中，Moore 机始终比 Mealy 机慢一个时钟周期。这在测试波形中可以明确的看出来。

（4） 显示输出（可选）：

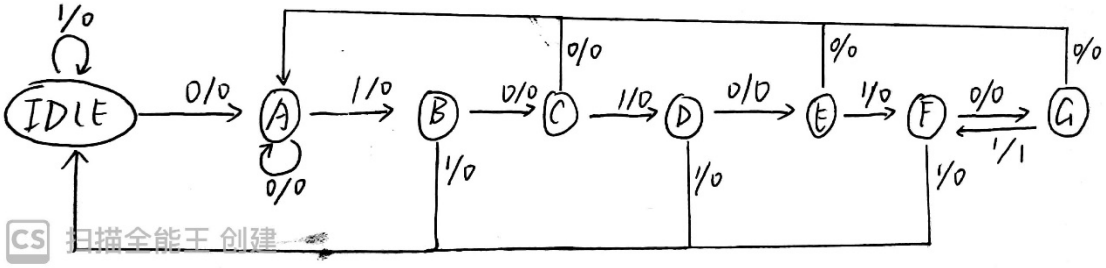
```
# At                 0 ns,rst=1,din=0,flag1=0,flag2=0
# At                10 ns,rst=0,din=0,flag1=0,flag2=0
# At                20 ns,rst=0,din=1,flag1=0,flag2=0
# At                30 ns,rst=0,din=0,flag1=0,flag2=0
# At                40 ns,rst=0,din=1,flag1=0,flag2=0
# At                50 ns,rst=0,din=0,flag1=0,flag2=0
# At                60 ns,rst=0,din=1,flag1=0,flag2=0
# At                70 ns,rst=0,din=0,flag1=0,flag2=0
# At                80 ns,rst=0,din=1,flag1=0,flag2=0
# At                85 ns,rst=0,din=1,flag1=1,flag2=0
# At                90 ns,rst=0,din=0,flag1=1,flag2=0
# At                95 ns,rst=0,din=0,flag1=0,flag2=1
# At               100 ns,rst=0,din=1,flag1=0,flag2=1
# At               105 ns,rst=0,din=1,flag1=1,flag2=0
# At               110 ns,rst=0,din=0,flag1=1,flag2=0
# At               115 ns,rst=0,din=0,flag1=0,flag2=1
# At               120 ns,rst=0,din=1,flag1=0,flag2=1
# At               125 ns,rst=0,din=1,flag1=1,flag2=0
# At               135 ns,rst=0,din=1,flag1=0,flag2=1
# At               145 ns,rst=0,din=1,flag1=0,flag2=0
# At               150 ns,rst=0,din=0,flag1=0,flag2=0
```

```
# At               160 ns,rst=0,din=1,flag1=0,flag2=0
# At               170 ns,rst=0,din=0,flag1=0,flag2=0
# At               180 ns,rst=0,din=1,flag1=0,flag2=0
# At               190 ns,rst=0,din=0,flag1=0,flag2=0
# At               200 ns,rst=0,din=1,flag1=0,flag2=0
# At               210 ns,rst=0,din=0,flag1=0,flag2=0
# At               220 ns,rst=0,din=1,flag1=0,flag2=0
# At               225 ns,rst=0,din=1,flag1=1,flag2=0
# At               230 ns,rst=0,din=0,flag1=1,flag2=0
# At               235 ns,rst=0,din=0,flag1=0,flag2=1
# At               240 ns,rst=0,din=1,flag1=0,flag2=1
# At               245 ns,rst=0,din=1,flag1=1,flag2=0
# At               250 ns,rst=1,din=1,flag1=0,flag2=0
```

（5）设计说明（可选）：

本题采用有限状态机设计。

Mealy 机的设计与上题相同，将 10101010 的七个活前缀（因为本题从 LSB 开始识别，实际上为后缀）0,10,010,1010,01010,101010,0101010 分别定义为 A,B,C,D,E,F,G 七个状态，外加初始状态 IDLE，得到 Mealy 机状态转换图如下：



修改 Mealy 机得到 Moore 机。Moore 机需要识别各个状态、且输出只与状态有关。因此，除了活前缀 0,10,010,1010,01010,101010,0101010 这七个状态和 IDLE 初始状态，还需要添加 10101010 状态。我们将其命名为状态 H。

Moore 机状态转移图如下所示：