数字系统设计作业

学号： 519021910917 姓名： 费扬 日期： 2021.12.12

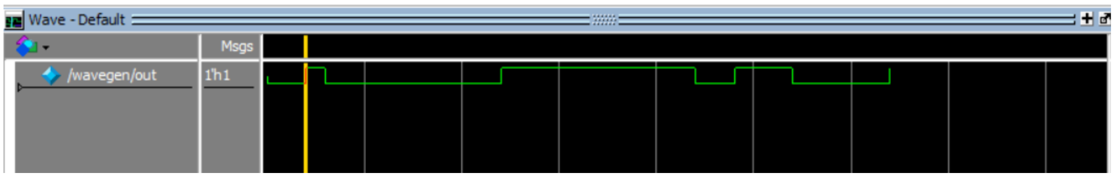
目录

第 1 题：

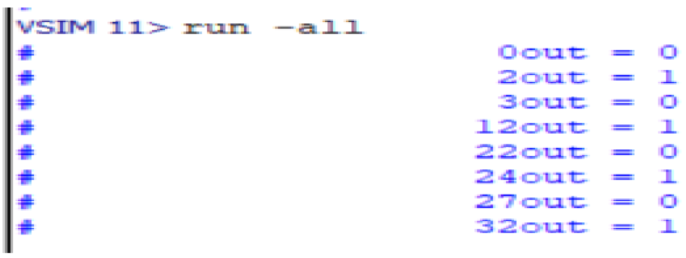
（1）设计模块

```verilog
`timescale 10 ns / 1 ns

module wavegen
  ( );
  reg out;
  initial
  begin
    out = 1'b0;
    #2 out = 1'b1;
    #1 out = 1'b0;
    #9 out = 1'b1;
    #10 out = 1'b0;
    #2 out = 1'b1;
    #3 out = 1'b0;
    #5 out = 1'b1;
end

initial
begin
    $monitor($time, "out = %b", out);
end
endmodule

```

（2）测试波形图：



（3）显示输出：



```
VSIM 11> run -all
#                             0out = 0
#                             2out = 1
#                             3out = 0
#                            12out = 1
#                            22out = 0
#                            24out = 1
#                            27out = 0
#                            32out = 1
```

第 2 题：

（1）设计模块

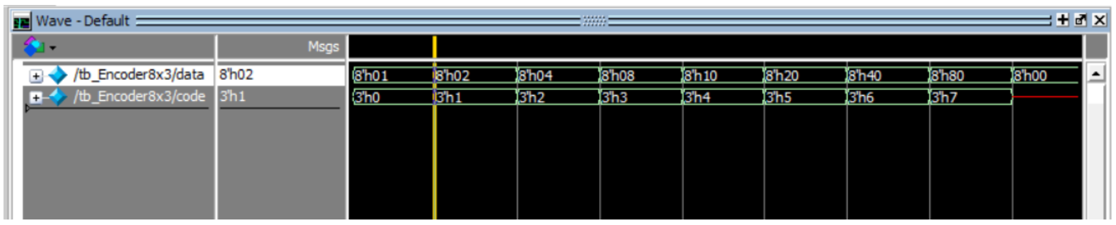
```verilog

`timescale 10 ns / 1 ns

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

always @(data)
begin
    case(data)
        8'b0000_0001: code = 0;
        8'b0000_0010: code = 1;
        8'b0000_0100: code = 2;
        8'b0000_1000: code = 3;
        8'b0001_0000: code = 4;
        8'b0010_0000: code = 5;
        8'b0100_0000: code = 6;
        8'b1000_0000: code = 7;
        default: code = 3'bxxx;
    endcase
end
endmodule

```

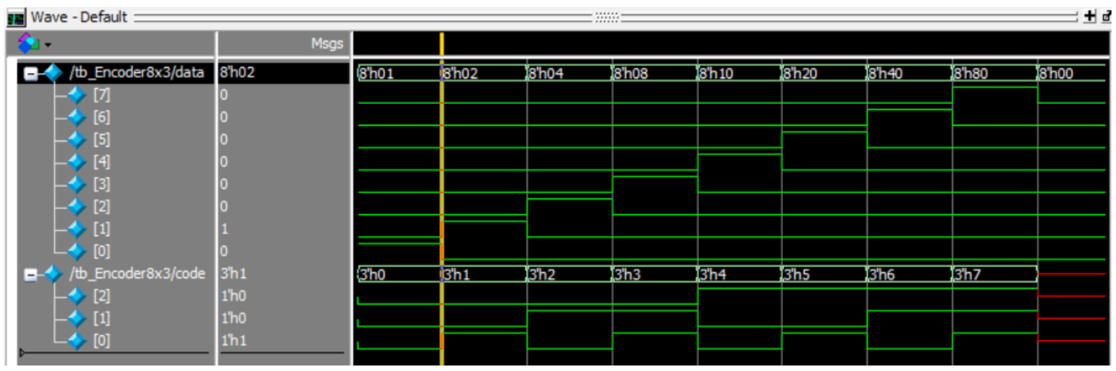
（2）测试模块

```verilog
`timescale 10 ns / 1 ns
`include "t2.v"

module tb_Encoder8x3();

reg [7:0] data;
wire [2:0] code;

initial begin
    data = 8'b0000_0001;
    forever
        #5 data = data << 1;
end

Encoder8x3 encoder8x3(code, data);

```

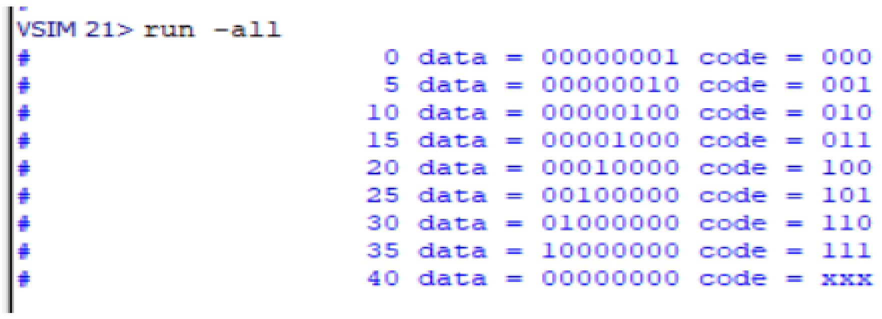
```
initial
    $monitor($time, "\tdata = %8b\tcode = %3b", data, code);

endmodule

```

（3）测试波形图：


Wave - Default
Msgs
/tb_Encoder8x3/data 8'h02
/tb_Encoder8x3/code 3'h1
8'h01 8'h02 8'h04 8'h08 8'h10 8'h20 8'h40 8'h80 8'h00
3'h0 3'h1 3'h2 3'h3 3'h4 3'h5 3'h6 3'h7


Wave - Default
Msgs
/tb_Encoder8x3/data 8'h02
[7] 0
[6] 0
[5] 0
[4] 0
[3] 0
[2] 0
[1] 1
[0] 0
/tb_Encoder8x3/code 3'h1
[2] 1'h0
[1] 1'h0
[0] 1'h1
8'h01 8'h02 8'h04 8'h08 8'h10 8'h20 8'h40 8'h80 8'h00
3'h0 3'h1 3'h2 3'h3 3'h4 3'h5 3'h6 3'h7

（4）显示输出：



```
VSIM 21> run -all
#                    0 data = 00000001 code = 000
#                    5 data = 00000010 code = 001
#                   10 data = 00000100 code = 010
#                   15 data = 00001000 code = 011
#                   20 data = 00010000 code = 100
#                   25 data = 00100000 code = 101
#                   30 data = 01000000 code = 110
#                   35 data = 10000000 code = 111
#                   40 data = 00000000 code = xxx
```

（5）设计说明：

Default 情况下输出为“code=xxx”即 3’bx。

第 3 题:

A）

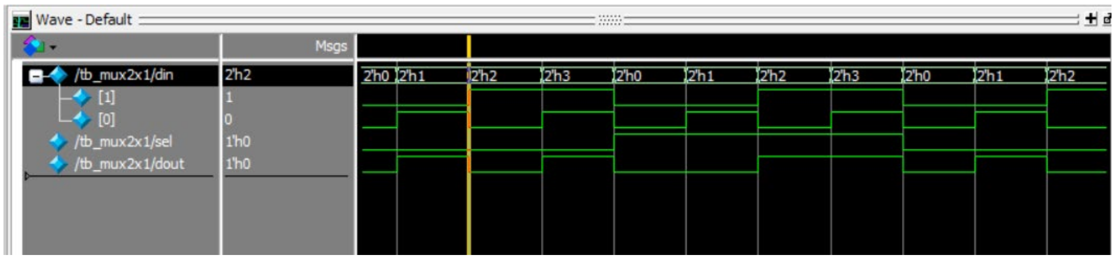
（1）设计模块

```

`timescale 10 ns / 1 ns

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

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

endmodule

```

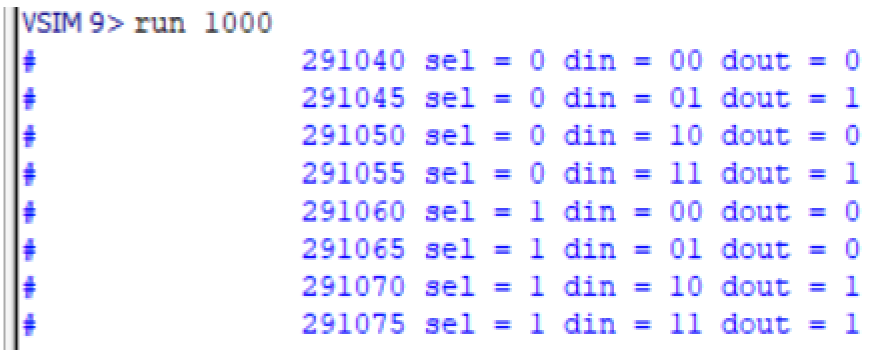
（2）测试模块

```
`timescale 10 ns / 1 ns
`include "3a.v"

module tb_mux2x1();

reg sel;
reg [1:0] din;
wire dout;

initial begin
    {sel, din} = 3'b0;
    forever
        #5 {sel, din} = {sel, din} + 1;
end

mux2x1 mux(dout, sel, din);

initial
    $monitor($time, "\tsel = %b\tdin = %2b\tdout = %b", sel, din, dout);

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：如果需要显示输出来说明模块设计的正确性；



```
VSIM 9> run 1000
#            291040 sel = 0 din = 00 dout = 0
#            291045 sel = 0 din = 01 dout = 1
#            291050 sel = 0 din = 10 dout = 0
#            291055 sel = 0 din = 11 dout = 1
#            291060 sel = 1 din = 00 dout = 0
#            291065 sel = 1 din = 01 dout = 0
#            291070 sel = 1 din = 10 dout = 1
#            291075 sel = 1 din = 11 dout = 1
```

B）

（1）设计模块

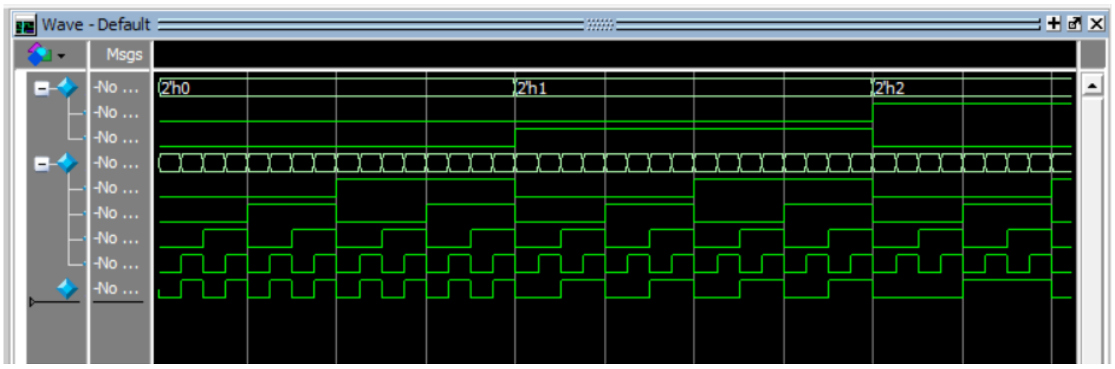
```
`timescale 10 ns / 1 ns
`include "3a.v"

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

wire doL, doH;
mux2x1 mux2x1_1(doL, sel[0], din[1:0]);
mux2x1 mux2x1_2(doH, sel[0], din[3:2]);
mux2x1 mux2x1_3(dout, sel[1], {doH, doL});

endmodule
```

（2）测试模块

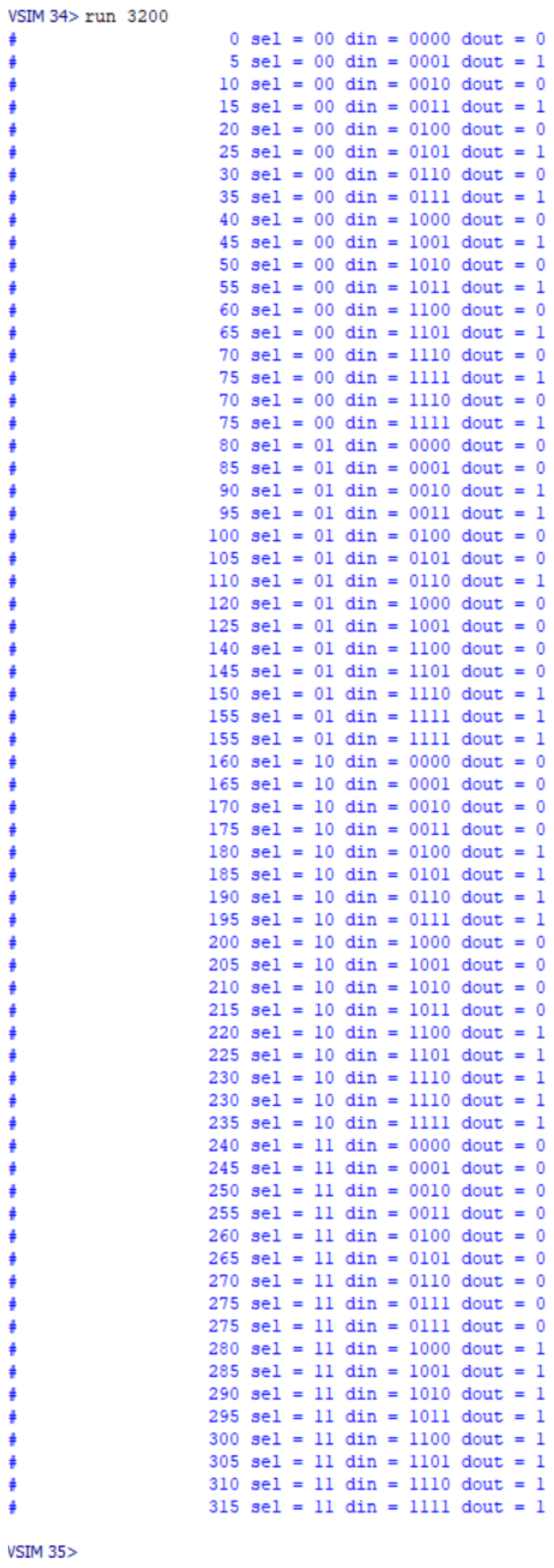
```
`timescale 10 ns / 1 ns
`include "3b.v"

module tb_mux4x1();
```

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

initial begin
    {sel, din} = 6'b0;
    forever
        #5 {sel, din} = {sel, din} + 1;
end

mux4x1 mux(dout, sel, din);

initial
    $monitor($time, "\tsel = %2b\tdin = %4b\tdout = %b", sel, din, dout);

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 34> run 3200
#                          0 sel = 00 din = 0000 dout = 0
#                          5 sel = 00 din = 0001 dout = 1
#                         10 sel = 00 din = 0010 dout = 0
#                         15 sel = 00 din = 0011 dout = 1
#                         20 sel = 00 din = 0100 dout = 0
#                         25 sel = 00 din = 0101 dout = 1
#                         30 sel = 00 din = 0110 dout = 0
#                         35 sel = 00 din = 0111 dout = 1
#                         40 sel = 00 din = 1000 dout = 0
#                         45 sel = 00 din = 1001 dout = 1
#                         50 sel = 00 din = 1010 dout = 0
#                         55 sel = 00 din = 1011 dout = 1
#                         60 sel = 00 din = 1100 dout = 0
#                         65 sel = 00 din = 1101 dout = 1
#                         70 sel = 00 din = 1110 dout = 0
#                         75 sel = 00 din = 1111 dout = 1
#                         70 sel = 00 din = 1110 dout = 0
#                         75 sel = 00 din = 1111 dout = 1
#                         80 sel = 01 din = 0000 dout = 0
#                         85 sel = 01 din = 0001 dout = 0
#                         90 sel = 01 din = 0010 dout = 1
#                         95 sel = 01 din = 0011 dout = 1
#                        100 sel = 01 din = 0100 dout = 0
#                        105 sel = 01 din = 0101 dout = 0
#                        110 sel = 01 din = 0110 dout = 1
#                        120 sel = 01 din = 1000 dout = 0
#                        125 sel = 01 din = 1001 dout = 0
#                        140 sel = 01 din = 1100 dout = 0
#                        145 sel = 01 din = 1101 dout = 0
#                        150 sel = 01 din = 1110 dout = 1
#                        155 sel = 01 din = 1111 dout = 1
#                        155 sel = 01 din = 1111 dout = 1
#                        160 sel = 10 din = 0000 dout = 0
#                        165 sel = 10 din = 0001 dout = 0
#                        170 sel = 10 din = 0010 dout = 0
#                        175 sel = 10 din = 0011 dout = 0
#                        180 sel = 10 din = 0100 dout = 1
#                        185 sel = 10 din = 0101 dout = 1
#                        190 sel = 10 din = 0110 dout = 1
#                        195 sel = 10 din = 0111 dout = 1
#                        200 sel = 10 din = 1000 dout = 0
#                        205 sel = 10 din = 1001 dout = 0
#                        210 sel = 10 din = 1010 dout = 0
#                        215 sel = 10 din = 1011 dout = 0
#                        220 sel = 10 din = 1100 dout = 1
#                        225 sel = 10 din = 1101 dout = 1
#                        230 sel = 10 din = 1110 dout = 1
#                        230 sel = 10 din = 1110 dout = 1
#                        235 sel = 10 din = 1111 dout = 1
#                        240 sel = 11 din = 0000 dout = 0
#                        245 sel = 11 din = 0001 dout = 0
#                        250 sel = 11 din = 0010 dout = 0
#                        255 sel = 11 din = 0011 dout = 0
#                        260 sel = 11 din = 0100 dout = 0
#                        265 sel = 11 din = 0101 dout = 0
#                        270 sel = 11 din = 0110 dout = 0
#                        275 sel = 11 din = 0111 dout = 0
#                        275 sel = 11 din = 0111 dout = 0
#                        280 sel = 11 din = 1000 dout = 1
#                        285 sel = 11 din = 1001 dout = 1
#                        290 sel = 11 din = 1010 dout = 1
#                        295 sel = 11 din = 1011 dout = 1
#                        300 sel = 11 din = 1100 dout = 1
#                        305 sel = 11 din = 1101 dout = 1
#                        310 sel = 11 din = 1110 dout = 1
#                        315 sel = 11 din = 1111 dout = 1

VSIM 35>
```

（5）设计说明（可选）：如果有需要说明的部分。

将 a 中 mux2_1 组合，放置于同一个 project 内较为方便。

第 4 题：

（1）设计模块

a. 结构方式：

```
`timescale 10ns / 1ns
module comb_str(Y,A,B,C,D);
  output Y;
  input A,B,C,D;

  wire ou1,ou2,ou3,ou4;
  not u1(ou1,D);
  not u2(ou2,ou3);
  or u3(ou3,A,D);
  and u4(ou4,B,C,ou1);
  and u5(Y,ou2,ou4);

endmodule
```

b. 数据流方式

```
`timescale 10 ns / 1 ns

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

assign Y = (~(A | D)) & (B & C & (~D));

endmodule
```

c. 行为方式：

```
`timescale 10 ns / 1 ns

module comb_behavior(output Y, input A, B, C, D);

reg buffer;
always @(*)
    buffer = ~(A | D) & (B & C & (~D));

assign Y = buffer;
```

```
endmodule
```

d. 用户自定义门原语：

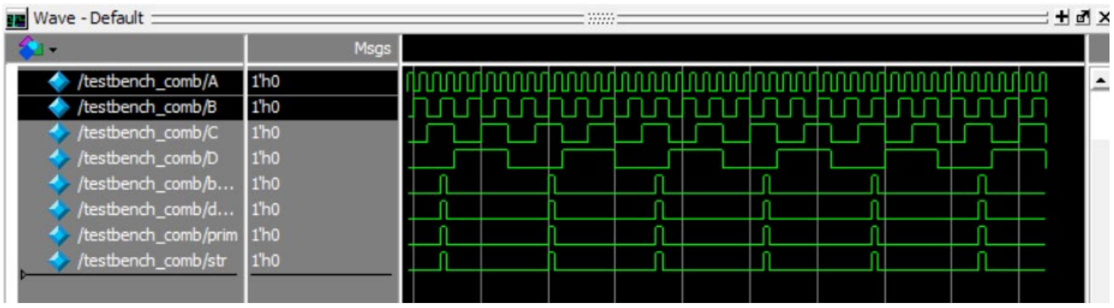
```
`timescale 10 ns / 1 ns

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

table
//  D   C   B   A   :   out ;
    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）测试模块

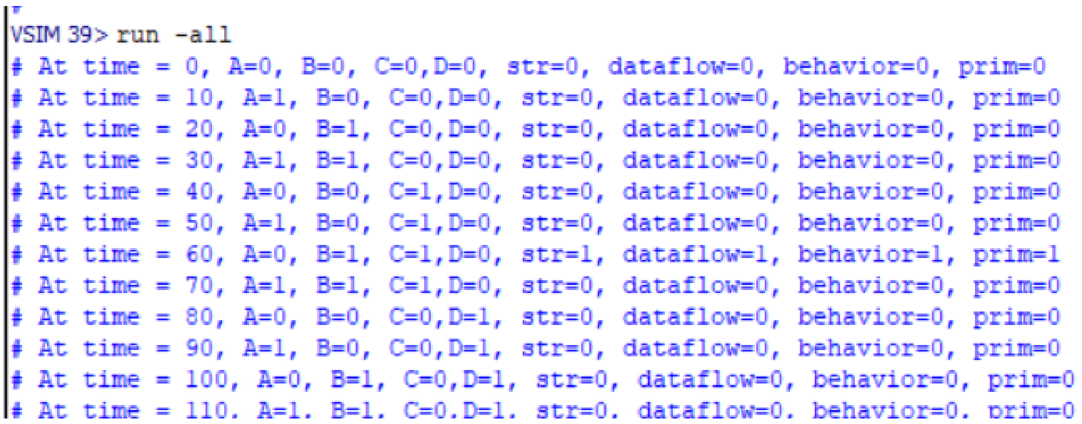
```
`timescale 10ns / 1ns
`include "t4_structure.v"
`include "t4_dataflow.v"
`include "t4_behavior.v"
`include "t4_prim.v"
module testbench_comb;

reg A, B, C, D;
wire str,dataflow,behavior,prim;
comb_str a(str,A,B,C,D);
comb_dataflow b(dataflow,A,B,C,D);
comb_behavior c(behavior,A,B,C,D);
```

```
comb_prim d(prim,A,B,C,D);

initial fork
{A,B,C,D}=4'b0;
forever #1 A=~A;
forever #2 B=~B;
forever #4 C=~C;
forever #8 D=~D;

join

initial begin
    $monitor("At time = %0t, A=%1b, B=%1b, C=%1b,D=%1b, str=%1b,
dataflow=%1b, behavior=%1b, prim=%1b",$time,
A,B,C,D,str,dataflow,behavior,prim);
end
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 39> run -all
# At time = 0, A=0, B=0, C=0,D=0, str=0, dataflow=0, behavior=0, prim=0
# At time = 10, A=1, B=0, C=0,D=0, str=0, dataflow=0, behavior=0, prim=0
# At time = 20, A=0, B=1, C=0,D=0, str=0, dataflow=0, behavior=0, prim=0
# At time = 30, A=1, B=1, C=0,D=0, str=0, dataflow=0, behavior=0, prim=0
# At time = 40, A=0, B=0, C=1,D=0, str=0, dataflow=0, behavior=0, prim=0
# At time = 50, A=1, B=0, C=1,D=0, str=0, dataflow=0, behavior=0, prim=0
# At time = 60, A=0, B=1, C=1,D=0, str=1, dataflow=1, behavior=1, prim=1
# At time = 70, A=1, B=1, C=1,D=0, str=0, dataflow=0, behavior=0, prim=0
# At time = 80, A=0, B=0, C=0,D=1, str=0, dataflow=0, behavior=0, prim=0
# At time = 90, A=1, B=0, C=0,D=1, str=0, dataflow=0, behavior=0, prim=0
# At time = 100, A=0, B=1, C=0,D=1, str=0, dataflow=0, behavior=0, prim=0
# At time = 110, A=1, B=1, C=0,D=1, str=0, dataflow=0, behavior=0, prim=0
```

（5）设计说明（可选）：如果有需要说明的部分。

Testbench 可以同时比较四种方式的结果，验证一致性。

第 5 题：

A）

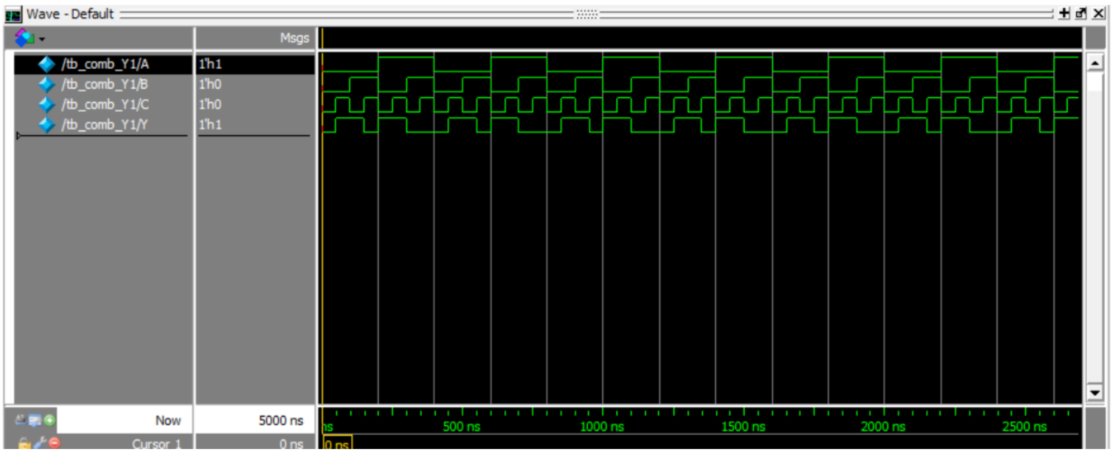
（1）设计模块

```
`timescale 10ns / 1ns
module comb_Y1(Y,A,B,C);
  output Y;
  input A,B,C;
  assign Y=(~A & ~B & C)|(~A & B & ~C)|(A & ~B & ~C)|(A & ~B & C);

endmodule
```

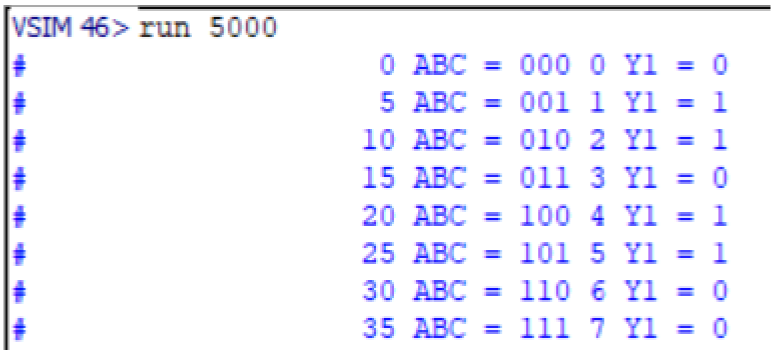
（2）测试模块

```

`timescale 10 ns / 1 ns
`include "t5.v"

module tb_comb_Y1();

reg A, B, C;
wire Y;

initial begin
    {A, B, C} = 3'b0;
    forever
        #5 {A, B, C} = {A, B, C} + 1;
end

comb_Y1 Y1(Y, A, B, C);

initial
    $monitor($time, "\tABC = %3b\t%d\tY1 = %b", {A, B, C}, {A, B, C}, Y);

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 46> run 5000
#                    0 ABC = 000 0 Y1 = 0
#                    5 ABC = 001 1 Y1 = 1
#                   10 ABC = 010 2 Y1 = 1
#                   15 ABC = 011 3 Y1 = 0
#                   20 ABC = 100 4 Y1 = 1
#                   25 ABC = 101 5 Y1 = 1
#                   30 ABC = 110 6 Y1 = 0
#                   35 ABC = 111 7 Y1 = 0
```

B）

（1）设计模块

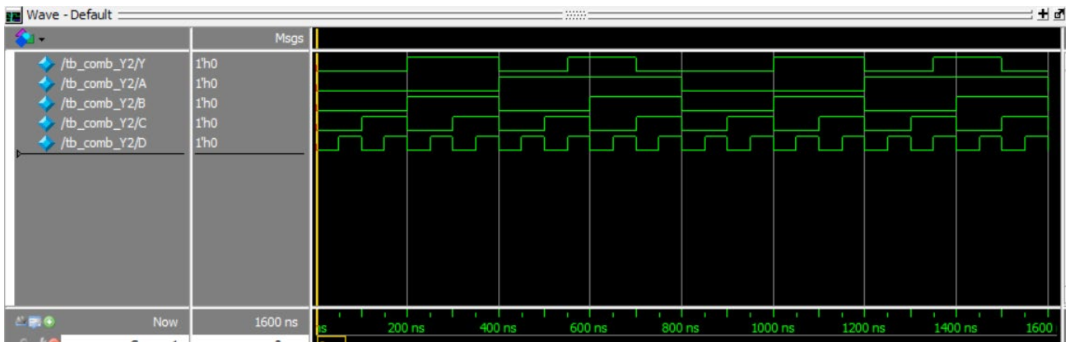
```
`timescale 10 ns / 1 ns

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

endmodule
```

（2）测试模块

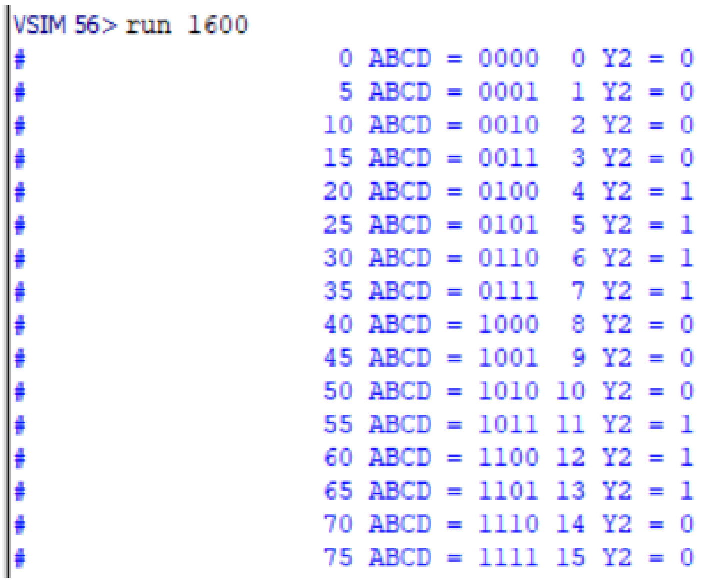
```
`timescale 10 ns / 1 ns
`include "t5b.v"

module tb_comb_Y2();
reg A, B, C, D;
wire Y;

initial begin
```

```
    {A, B, C, D} = 4'b0;
    forever
        #5 {A, B, C, D} = {A, B, C, D} + 1;
end

comb_Y2 Y2(Y, A, B, C, D);

initial
    $monitor(
        $time, "\tABCD = %4b\t%d\tY2 = %b",
        {A, B, C, D}, {A, B, C, D}, Y
    );

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 56> run 1600
#                    0 ABCD = 0000   0 Y2 = 0
#                    5 ABCD = 0001   1 Y2 = 0
#                   10 ABCD = 0010   2 Y2 = 0
#                   15 ABCD = 0011   3 Y2 = 0
#                   20 ABCD = 0100   4 Y2 = 1
#                   25 ABCD = 0101   5 Y2 = 1
#                   30 ABCD = 0110   6 Y2 = 1
#                   35 ABCD = 0111   7 Y2 = 1
#                   40 ABCD = 1000   8 Y2 = 0
#                   45 ABCD = 1001   9 Y2 = 0
#                   50 ABCD = 1010  10 Y2 = 0
#                   55 ABCD = 1011  11 Y2 = 1
#                   60 ABCD = 1100  12 Y2 = 1
#                   65 ABCD = 1101  13 Y2 = 1
#                   70 ABCD = 1110  14 Y2 = 0
#                   75 ABCD = 1111  15 Y2 = 0
```

（5）设计说明（可选）：如果有需要说明的部分。

需要用到数字电路的一些知识，布尔表达式等。

第 6 题:

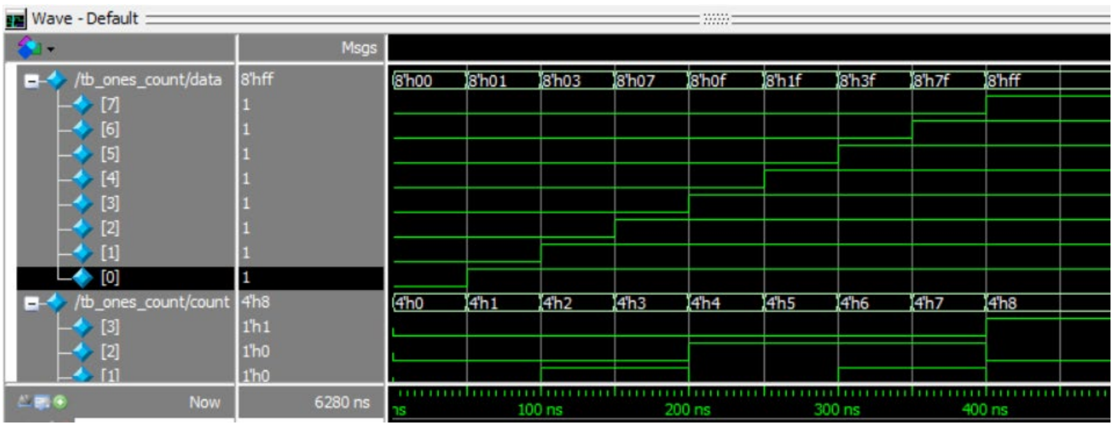
（1）设计模块

```
`timescale 10ns / 1ns
module ones_count(output [3:0] count,input [7:0] dat_in);

  assign count = dat_in[0] + dat_in[1] + dat_in[2] + dat_in[3] + dat_in[4]
+ dat_in[5] + dat_in[6] + dat_in[7];

endmodule
```

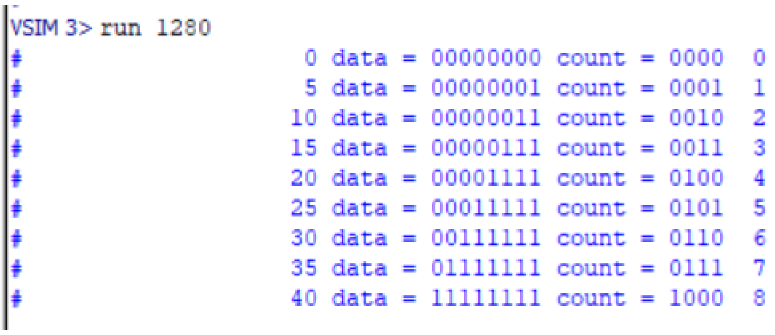
（2）测试模块

```
`timescale 10 ns / 1 ns
`include "t6.v"

module tb_ones_count();

reg [7:0] data;
wire [3:0] count;
reg [7:0] mask;

initial begin
    data = 8'b0;
    mask = 8'b1;
    forever begin
        #5 data = data | mask;
        mask = mask << 1;
    end
end

ones_count oc(count, data);

initial
    $monitor($time, "\tdata = %8b\tcount = %4b\t%d", data, count, count);

endmodule
```

（3）测试波形图：如果很多，可以提供部分波形内容；


Wave - Default
Msgs
/tb_ones_count/data 8'hff
[7] 1
[6] 1
[5] 1
[4] 1
[3] 1
[2] 1
[1] 1
[0] 1
/tb_ones_count/count 4'h8
[3] 1'h1
[2] 1'h0
[1] 1'h0
8'h00 8'h01 8'h03 8'h07 8'h0f 8'h1f 8'h3f 8'h7f 8'hff
4'h0 4'h1 4'h2 4'h3 4'h4 4'h5 4'h6 4'h7 4'h8
Now 6280 ns
ns 100 ns 200 ns 300 ns 400 ns

（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 3> run 1280
#                    0 data = 00000000 count = 0000  0
#                    5 data = 00000001 count = 0001  1
#                   10 data = 00000011 count = 0010  2
#                   15 data = 00000111 count = 0011  3
#                   20 data = 00001111 count = 0100  4
#                   25 data = 00011111 count = 0101  5
#                   30 data = 00111111 count = 0110  6
#                   35 data = 01111111 count = 0111  7
#                   40 data = 11111111 count = 1000  8
```

（5）设计说明（可选）：如果有需要说明的部分。

这里测试了数字中有 **1~8** 个 **1** 的数字，不需要测试从 **8’b0~8’b1** 的所有数字。

第 7 题：

（1）设计模块

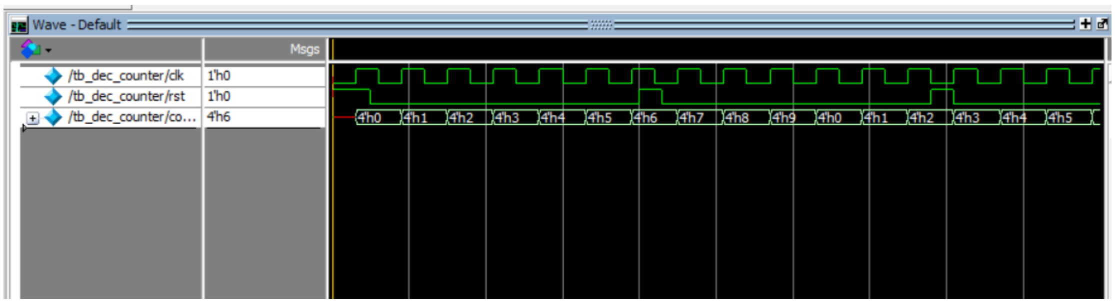
```
`timescale 10ns / 1ns
 module dec_counter(output reg [4:0] count, input clk, reset);

 always@(posedge clk) begin
     if (reset)
     count <= 4'b0;
     else begin
         if(count == 9)
         count <= 0;
         else
         count <= count+1;
     end
 end
 endmodule
```

（2）测试模块

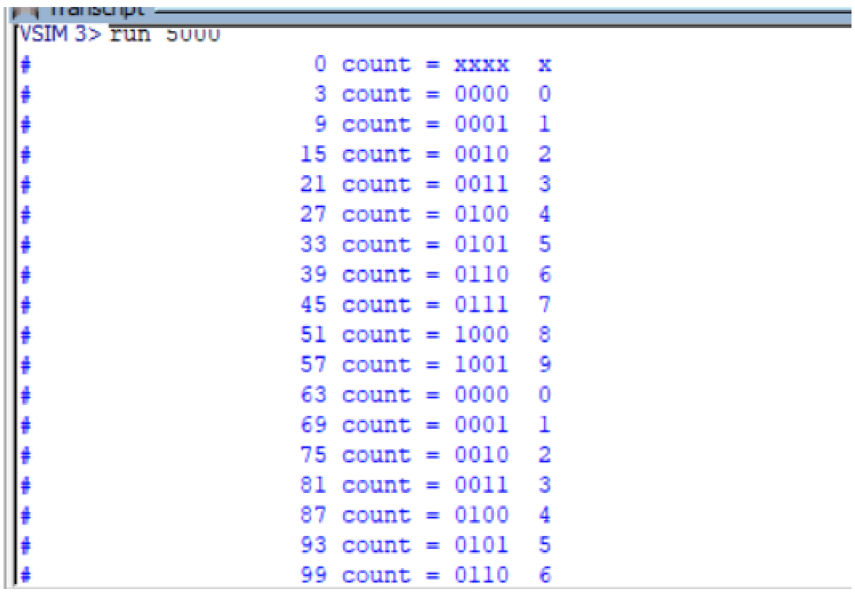
```
`timescale 10ns /1ns
`include "t7.v"

module tb_dec_counter();

reg clk;
reg rst;
wire [3:0] count;

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

initial begin
    rst = 1'b1;
    #5 rst = 1'b0;
    forever
        begin
            #35 rst = 1'b1;
            #3 rst = 1'b0;
        end
```

```
end

dec_counter counter(count, clk, rst);

initial
    $monitor($time, "\tcount = %4b\t%d", count, count);

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；



```
Transcript
VSIM 3> run 5000
#                    0 count = xxxx  x
#                    3 count = 0000  0
#                    9 count = 0001  1
#                   15 count = 0010  2
#                   21 count = 0011  3
#                   27 count = 0100  4
#                   33 count = 0101  5
#                   39 count = 0110  6
#                   45 count = 0111  7
#                   51 count = 1000  8
#                   57 count = 1001  9
#                   63 count = 0000  0
#                   69 count = 0001  1
#                   75 count = 0010  2
#                   81 count = 0011  3
#                   87 count = 0100  4
#                   93 count = 0101  5
#                   99 count = 0110  6
```

（5）设计说明（可选）：如果有需要说明的部分。

同步复位与异步复位的区别主要看是否有时钟信号参与。异步复位不需要时钟参与，一旦信号有效立即执行复位操作；同步信号需要时钟参与，只有有效的时钟信号出现，复位信号才有效。

第 8 题：

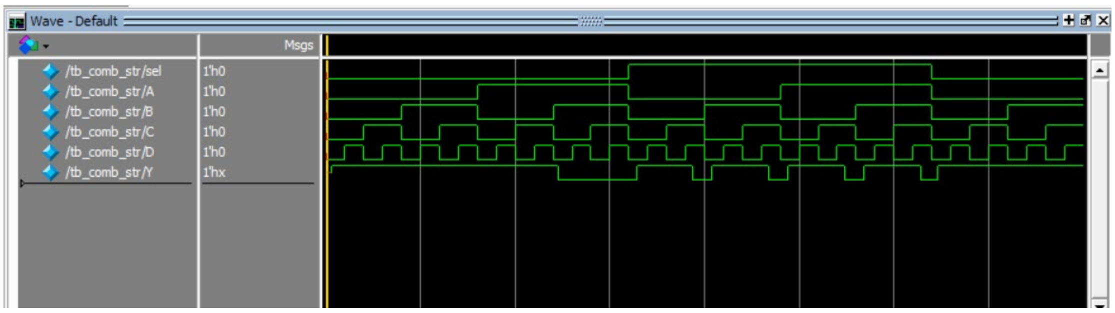
（1）设计模块

```verilog
`timescale 10ns / 1ns
module comb_str(output y,input sel, A, B, C, D);

wire in0, in1, y0, y1, sel_bar;
nand nand0(in0,A,B);
nand nand1(in1,C,D);

not notsel(sel_bar,sel);
and and0(y0,sel_bar,in0);
and and1(y1,sel,in1);
or result(y,y0,y1);

endmodule
```

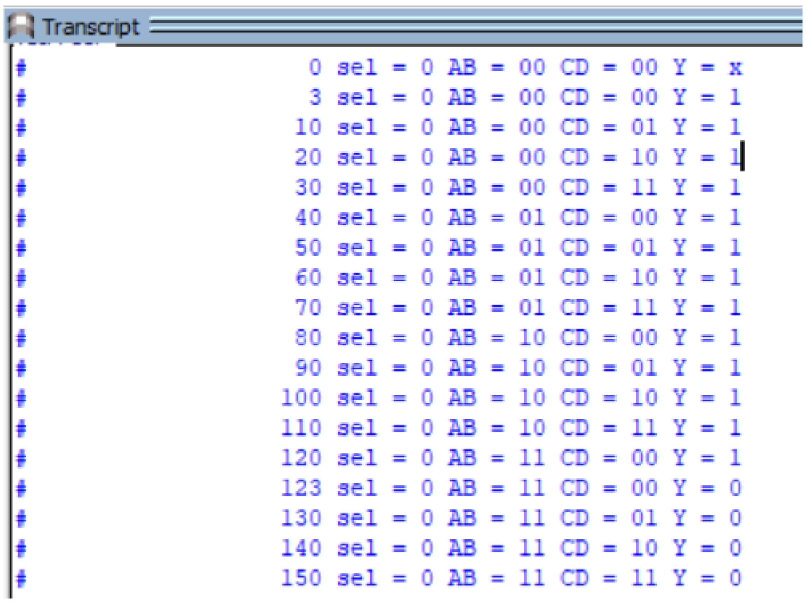
（2）测试模块

```verilog
`timescale 10 ns / 1 ns
`include "t8.v"

module tb_comb_str();

reg sel, A, B, C, D;
wire Y;

initial begin
    {sel, A, B, C, D} = 5'b0;
    forever
        #10 {sel, A, B, C, D} = {sel, A, B, C, D} + 1;
end

comb_str comb(Y, sel, A, B, C, D);

initial
    $monitor(
        $time, "\tsel = %b\tAB = %2b\tCD = %2b\tY = %b",
        sel, {A, B}, {C, D}, Y
    );

endmodule
```

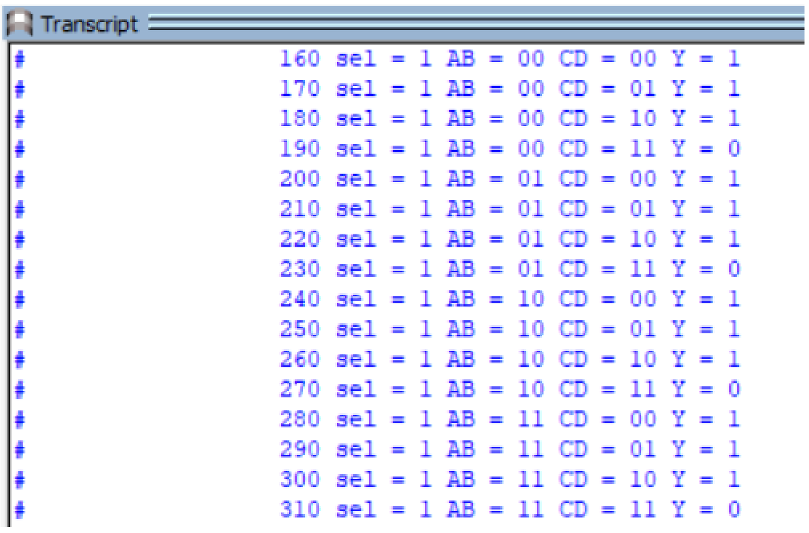
（3）测试波形图：如果很多，可以提供部分波形内容；



(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;



```
Transcript
#                    0 sel = 0 AB = 00 CD = 00 Y = x
#                    3 sel = 0 AB = 00 CD = 00 Y = 1
#                   10 sel = 0 AB = 00 CD = 01 Y = 1
#                   20 sel = 0 AB = 00 CD = 10 Y = 1
#                   30 sel = 0 AB = 00 CD = 11 Y = 1
#                   40 sel = 0 AB = 01 CD = 00 Y = 1
#                   50 sel = 0 AB = 01 CD = 01 Y = 1
#                   60 sel = 0 AB = 01 CD = 10 Y = 1
#                   70 sel = 0 AB = 01 CD = 11 Y = 1
#                   80 sel = 0 AB = 10 CD = 00 Y = 1
#                   90 sel = 0 AB = 10 CD = 01 Y = 1
#                  100 sel = 0 AB = 10 CD = 10 Y = 1
#                  110 sel = 0 AB = 10 CD = 11 Y = 1
#                  120 sel = 0 AB = 11 CD = 00 Y = 1
#                  123 sel = 0 AB = 11 CD = 00 Y = 0
#                  130 sel = 0 AB = 11 CD = 01 Y = 0
#                  140 sel = 0 AB = 11 CD = 10 Y = 0
#                  150 sel = 0 AB = 11 CD = 11 Y = 0
```



```
Transcript
#                  160 sel = 1 AB = 00 CD = 00 Y = 1
#                  170 sel = 1 AB = 00 CD = 01 Y = 1
#                  180 sel = 1 AB = 00 CD = 10 Y = 1
#                  190 sel = 1 AB = 00 CD = 11 Y = 0
#                  200 sel = 1 AB = 01 CD = 00 Y = 1
#                  210 sel = 1 AB = 01 CD = 01 Y = 1
#                  220 sel = 1 AB = 01 CD = 10 Y = 1
#                  230 sel = 1 AB = 01 CD = 11 Y = 0
#                  240 sel = 1 AB = 10 CD = 00 Y = 1
#                  250 sel = 1 AB = 10 CD = 01 Y = 1
#                  260 sel = 1 AB = 10 CD = 10 Y = 1
#                  270 sel = 1 AB = 10 CD = 11 Y = 0
#                  280 sel = 1 AB = 11 CD = 00 Y = 1
#                  290 sel = 1 AB = 11 CD = 01 Y = 1
#                  300 sel = 1 AB = 11 CD = 10 Y = 1
#                  310 sel = 1 AB = 11 CD = 11 Y = 0
```

第 9 题：

（1）设计模块

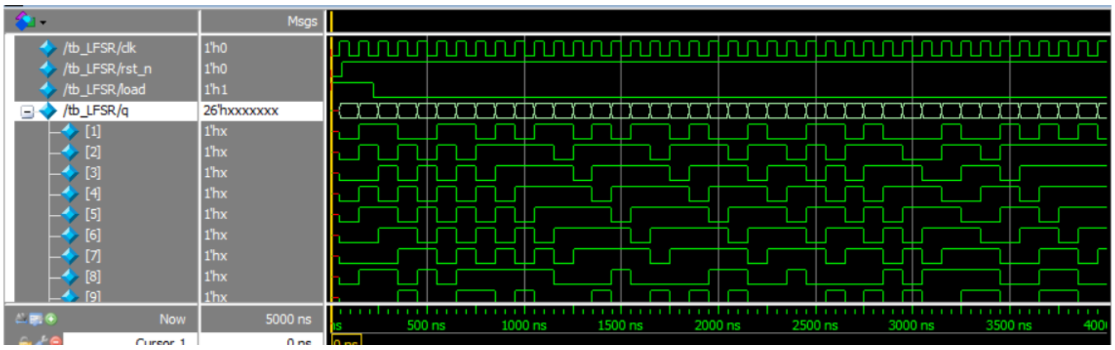
```
`timescale 10 ns / 1 ns

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];//X9 <-- X8^X26
            q[8] <= q[7]^q[26];// 8      7   26
            q[3:7] <= q[2:6];
            q[2] <= q[1]^q[26];// 2      1   26
            q[1] <= q[26]; //     1      0   26
            end
        end
    end
end

endmodule
```

（2）测试模块

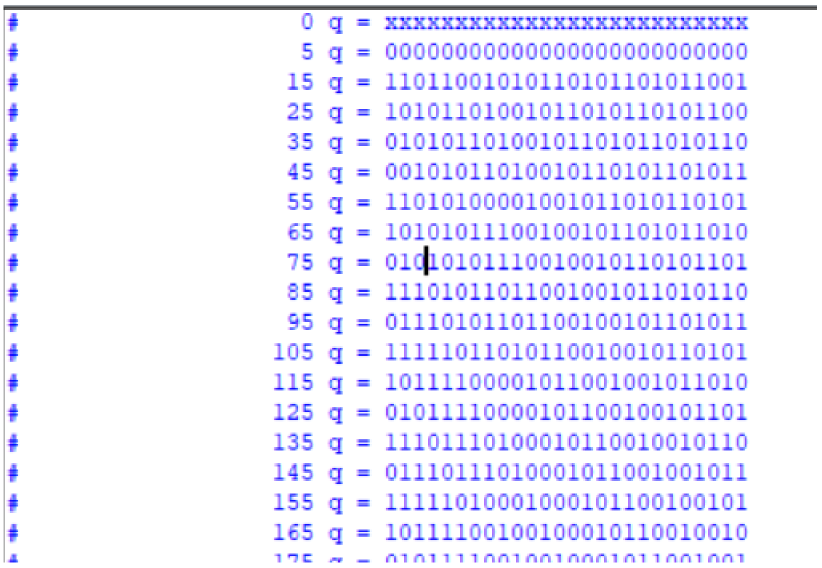
```
`timescale 10 ns / 1 ns
`include "t9.v"

module tb_LFSR();
```

```verilog
wire [1:26] q;  // 26 bit data output.
reg clk;        // Clock input.
reg rst_n;      // Synchronous reset input.
reg load;       // Synchronous load input.
reg [1:26] din; // 26 bit parallel data input.

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

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

initial begin
    din = 26'b1101_1001_0101_1010_1101_0110_01;
    load = 1'b1;
    #22 load = 1'b0;
end

LFSR lsfr(q, clk, rst_n, load, din);

initial
    $monitor($time, "\tq = %26b", q);

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：如果需要显示输出来说明模块设计的正确性；

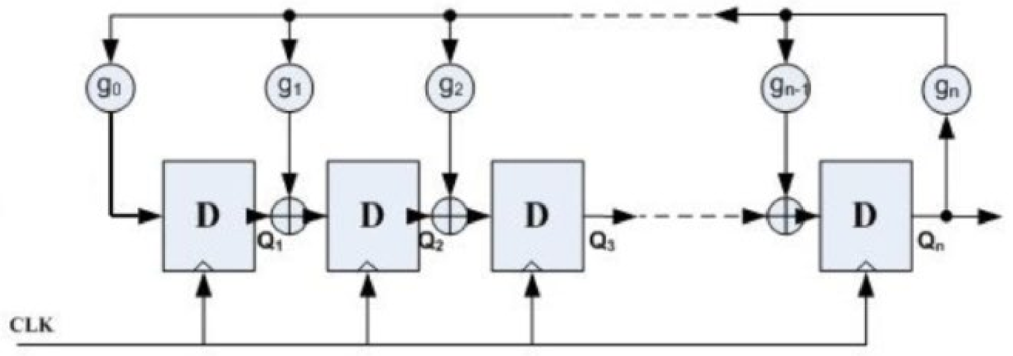


```
#                        0 q = xxxxxxxxxxxxxxxxxxxxxxxxxx
#                        5 q = 00000000000000000000000000
#                       15 q = 11011001010110101101011001
#                       25 q = 10101101001011010110101100
#                       35 q = 01010110100101101011010110
#                       45 q = 00101011010010110101101011
#                       55 q = 11010100001001011010110101
#                       65 q = 10101011100100101101011010
#                       75 q = 01010101110010010110101101
#                       85 q = 11101011011001001011010110
#                       95 q = 01110101101100100101101011
#                      105 q = 11111011010110010010110101
#                      115 q = 10111110000101100100101101
#                      125 q = 01011111000010110010010110
#                      135 q = 11101110100010110010010110
#                      145 q = 01110111010001011001001011
#                      155 q = 11111010001000101100100101
#                      165 q = 10111100100100010110010010
#                      175 q = 01011110010010001011001001
```

（5）设计说明（可选）：如果有需要说明的部分。

LFSR 用于产生可重复的伪随机序列 **PRBS**，该电路有 **n** 级触发器和一些异或门组成。

如下图所示：



第 10 题：

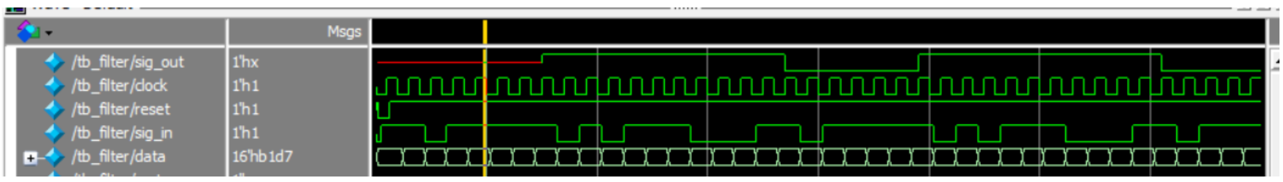
（1）设计模块

```verilog
`timescale 10 ns / 1 ns

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

reg [0:3] q;
wire j, k;

assign j = &q[1:3];
assign k = &(~q[1:3]);

always @(posedge clock)
 begin
    if (!reset)
        q <= 4'b0;
    else begin
        q[0] <= sig_in;
        q[1] <= q[0];
        q[2] <= q[1];
        q[3] <= q[2];
    end
end

always @(posedge clock)
begin
    if (!reset) begin
        sig_out <= 1'b0;
    end else begin
        case ({j, k})
            2'b00: sig_out <= sig_out;
            2'b01: sig_out <= 1'b0;
            2'b10: sig_out <= 1'b1;
            default: sig_out <= 2'bxx;
        endcase
    end
end

endmodule
```

（2）测试模块

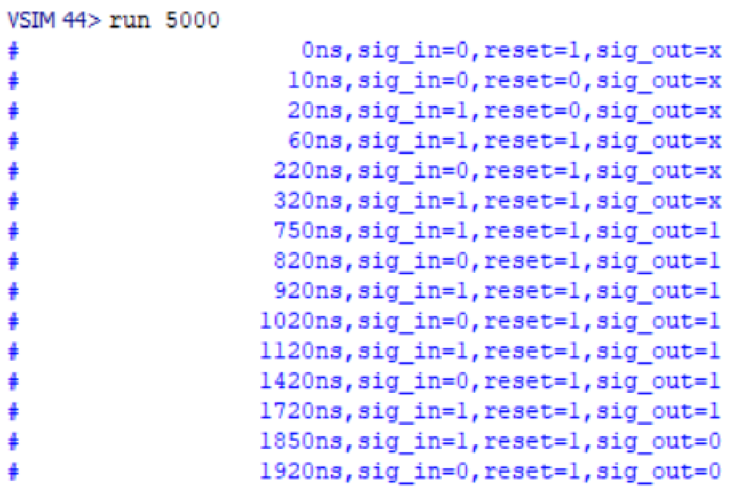
```verilog
`timescale 10 ns / 1 ns
`include "t10.v"

module tb_filter();

wire sig_out;
reg clock, reset, sig_in;
reg [15:0] data;

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

initial begin
    reset = 1'b1;
    #1 reset = 1'b0;
    #5 reset = 1'b1;
end

initial begin
    data = 16'b0001_1101_0111_1101;
    sig_in = 1'b0;
    #2 sig_in = 1'b1;
    forever
        #10 {data, sig_in} = {sig_in, data};
end

filter
ftr(.sig_out(sig_out), .clock(clock), .reset(rset), .sig_in(sig_in));

initial
begin
  $monitor("%tns,",$time,"sig_in=%b,reset=%b,sig_out=%b",sig_in,reset,si
g_out);
end
endmodule
```

（3）测试波形图：如果很多，可以提供部分波形内容；



	Msgs
/tb_filter/sig_out	1'hx
/tb_filter/clock	1'h1
/tb_filter/reset	1'h1
/tb_filter/sig_in	1'h1
/tb_filter/data	16'hb1d7

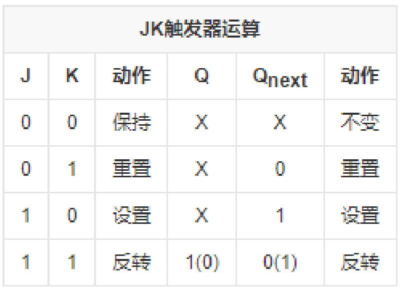
（4)显示输出(可选)：如果需要显示输出来说明模块设计的正确性；



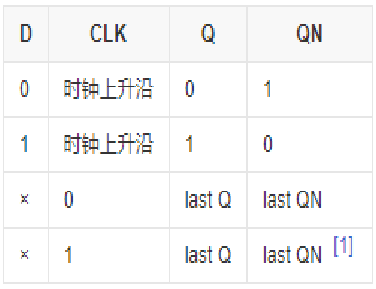
```
VSIM 44> run 5000
#                    0ns,sig_in=0,reset=1,sig_out=x
#                   10ns,sig_in=0,reset=0,sig_out=x
#                   20ns,sig_in=1,reset=0,sig_out=x
#                   60ns,sig_in=1,reset=1,sig_out=x
#                  220ns,sig_in=0,reset=1,sig_out=x
#                  320ns,sig_in=1,reset=1,sig_out=x
#                  750ns,sig_in=1,reset=1,sig_out=1
#                  820ns,sig_in=0,reset=1,sig_out=1
#                  920ns,sig_in=1,reset=1,sig_out=1
#                 1020ns,sig_in=0,reset=1,sig_out=1
#                 1120ns,sig_in=1,reset=1,sig_out=1
#                 1420ns,sig_in=0,reset=1,sig_out=1
#                 1720ns,sig_in=1,reset=1,sig_out=1
#                 1850ns,sig_in=1,reset=1,sig_out=0
#                 1920ns,sig_in=0,reset=1,sig_out=0
```

（5）设计说明（可选）：如果有需要说明的部分。

本题用到了 D 触发器和 JK 触发器，一些内容如下：



JK触发器运算					
J	K	动作	Q	Q_{next}	动作
0	0	保持	X	X	不变
0	1	重置	X	0	重置
1	0	设置	X	1	设置
1	1	反转	1(0)	0(1)	反转



D	CLK	Q	QN
0	时钟上升沿	0	1
1	时钟上升沿	1	0
×	0	last Q	last QN
×	1	last Q	last QN [1]

第 11 题：

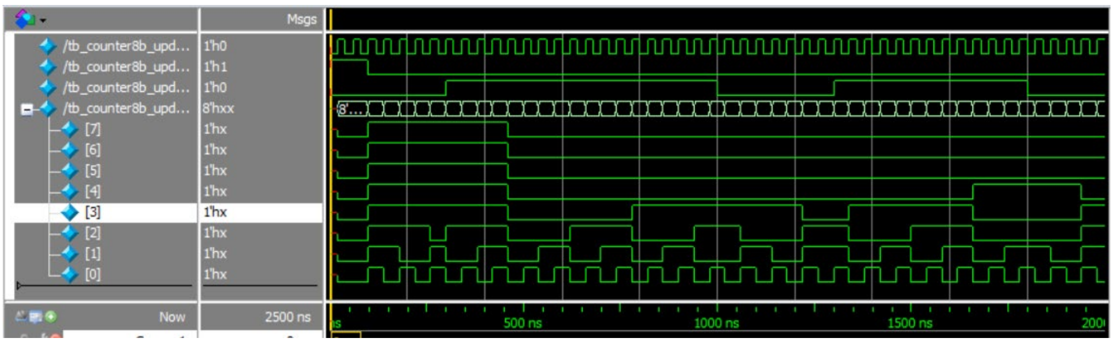
（1）设计模块

```
`timescale 10 ns / 1 ns

module counter8b_updown(output reg [7:0] count, input clk, reset, dir);

always @(posedge clk or negedge reset)
begin
  if(reset)
        count <= 4'b0;
    else if (dir)
            count <= count + 1;
    else
            count <= count - 1;
    end
endmodule
```

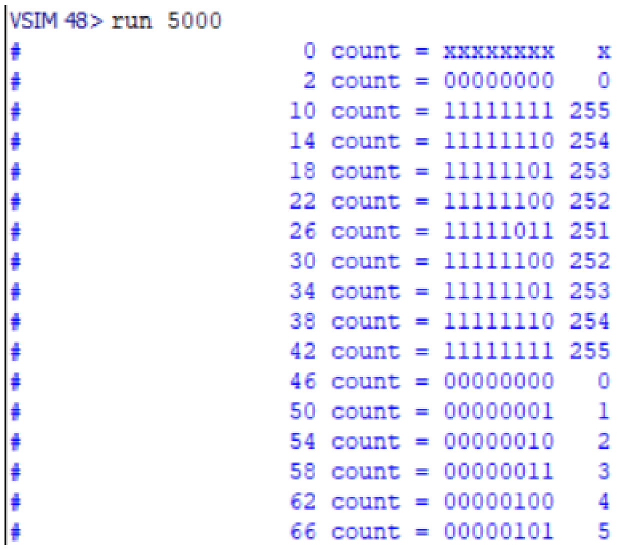
（2）测试模块

```
`timescale 10 ns / 1 ns
`include "t11.v"

module tb_counter8b_updown();

reg clk;
reg rst;
reg dir;
wire [7:0] count;

counter8b_updown
counter(.count(count), .clk(clk), .reset(rst), .dir(dir));

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

initial
begin
  rst = 1;
  #10 rst = 0;
end

initial
begin
    dir = 1'b0;
    #30 dir = 1;
    #70 dir = 0;
    #30 dir = 1;
    #50 dir = 0;
    #20 dir = 1;
    #50 $stop;
end


initial
    $monitor($time, "\tcount = %8b\t%d", count, count);

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 48> run 5000
#                    0  count = xxxxxxxx    x
#                    2  count = 00000000    0
#                   10  count = 11111111  255
#                   14  count = 11111110  254
#                   18  count = 11111101  253
#                   22  count = 11111100  252
#                   26  count = 11111011  251
#                   30  count = 11111100  252
#                   34  count = 11111101  253
#                   38  count = 11111110  254
#                   42  count = 11111111  255
#                   46  count = 00000000    0
#                   50  count = 00000001    1
#                   54  count = 00000010    2
#                   58  count = 00000011    3
#                   62  count = 00000100    4
#                   66  count = 00000101    5
```

第 12 题：

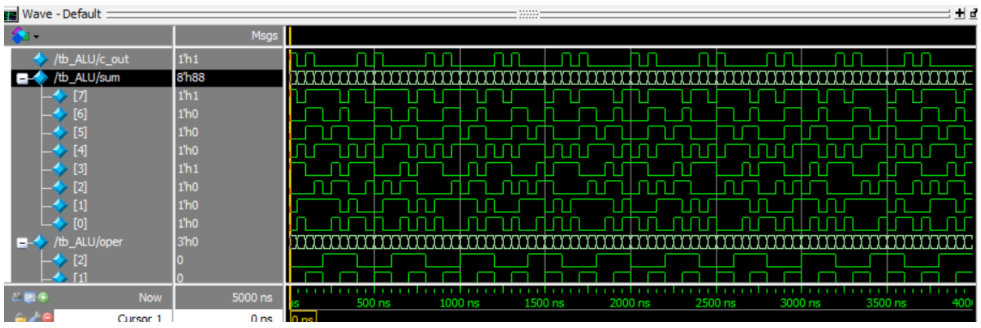
（1）设计模块

```
    `timescale 10ns / 1ns

`define OP_and          3'b000
`define OP_substract     3'b001
`define OP_substract_a  3'b010
`define OP_or_ab        3'b011
`define OP_and_ab       3'b100
`define OP_not_ab       3'b101
`define OP_exor         3'b110
`define OP_exnor        3'b111

module ALU(output reg c_out, output reg [7:0] sum, input [2:0] oper, input
[7:0] a, input [7:0] b, input c_in);
  always @(*)
  begin
    case(oper)
    `OP_and         : {c_out, sum} = a + b + c_in;
        `OP_substract   : {c_out, sum} = a + ~b + c_in;
        `OP_substract_a : {c_out, sum} = b + ~a + ~c_in;
        `OP_or_ab       : {c_out, sum} = {1'b0, a | b};
        `OP_and_ab      : {c_out, sum} = {1'b0, a & b};
        `OP_not_ab      : {c_out, sum} = {1'b0, (~a) | b};
        `OP_exor        : {c_out, sum} = {1'b0, a ^ b};
        `OP_exnor       : {c_out, sum} = {1'b0, a ~^ b};
        default: {c_out, sum} = 9'bx;
    endcase
end
endmodule
```

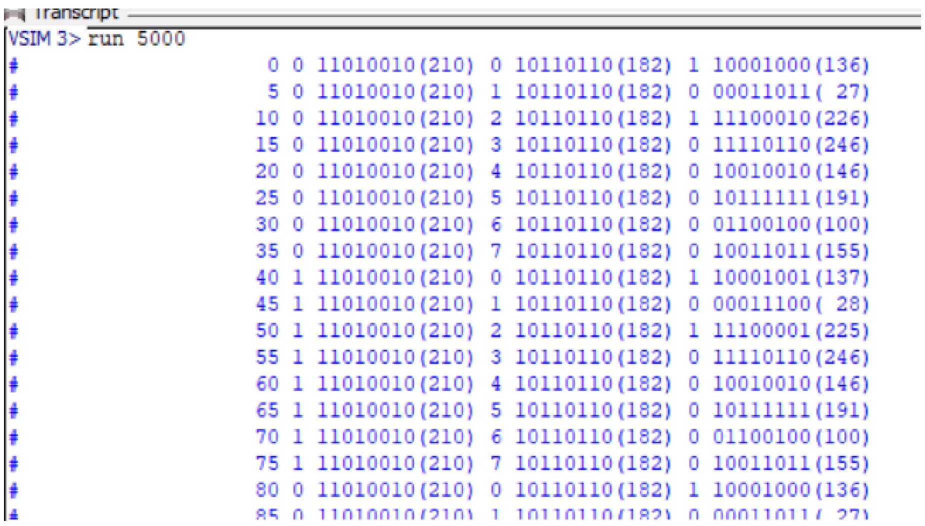
（2）测试模块

```
`timescale 10 ns / 1 ns
`include "t12.v"

module tb_ALU();
wire c_out;
wire [7:0] sum;
reg [2:0] oper;
reg [7:0] a;
reg [7:0] b;
reg c_in;
initial begin
    a = 8'b1101_0010;
    b = 8'b1011_0110;
    {c_in, oper} = 4'b0;
    forever
        #5 {c_in, oper} = {c_in, oper} + 1;
end

ALU alu(.c_out(c_out), .sum(sum), .oper(oper), .a(a), .b(b), .c_in(c_in));
initial
    $monitor(
        $time, "\t%b\t%8b(%d)\t%d\t%8b(%d)\t%b\t%8b(%d)",
        c_in, a, a, oper, b, b, c_out, sum, sum
    );
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；


```
Transcript
VSIM 3> run 5000
#                    0  0  11010010(210)  0  10110110(182)  1  10001000(136)
#                    5  0  11010010(210)  1  10110110(182)  0  00011011( 27)
#                   10  0  11010010(210)  2  10110110(182)  1  11100010(226)
#                   15  0  11010010(210)  3  10110110(182)  0  11110110(246)
#                   20  0  11010010(210)  4  10110110(182)  0  10010010(146)
#                   25  0  11010010(210)  5  10110110(182)  0  10111111(191)
#                   30  0  11010010(210)  6  10110110(182)  0  01100100(100)
#                   35  0  11010010(210)  7  10110110(182)  0  10011011(155)
#                   40  1  11010010(210)  0  10110110(182)  1  10001001(137)
#                   45  1  11010010(210)  1  10110110(182)  0  00011100( 28)
#                   50  1  11010010(210)  2  10110110(182)  1  11100001(225)
#                   55  1  11010010(210)  3  10110110(182)  0  11110110(246)
#                   60  1  11010010(210)  4  10110110(182)  0  10010010(146)
#                   65  1  11010010(210)  5  10110110(182)  0  10111111(191)
#                   70  1  11010010(210)  6  10110110(182)  0  01100100(100)
#                   75  1  11010010(210)  7  10110110(182)  0  10011011(155)
#                   80  0  11010010(210)  0  10110110(182)  1  10001000(136)
#                   85  0  11010010(210)  1  10110110(182)  0  00011011( 27)
```

（5）设计说明（可选）：如果有需要说明的部分。

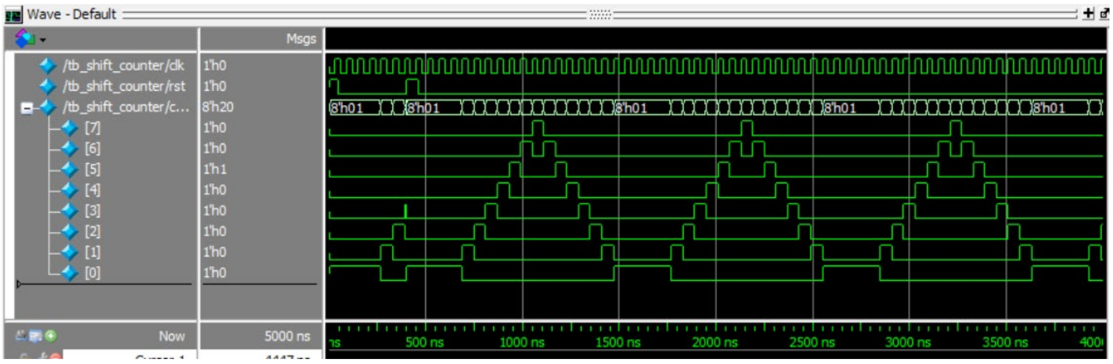
Oper 值按照表格顺序设计。

第 13 题：

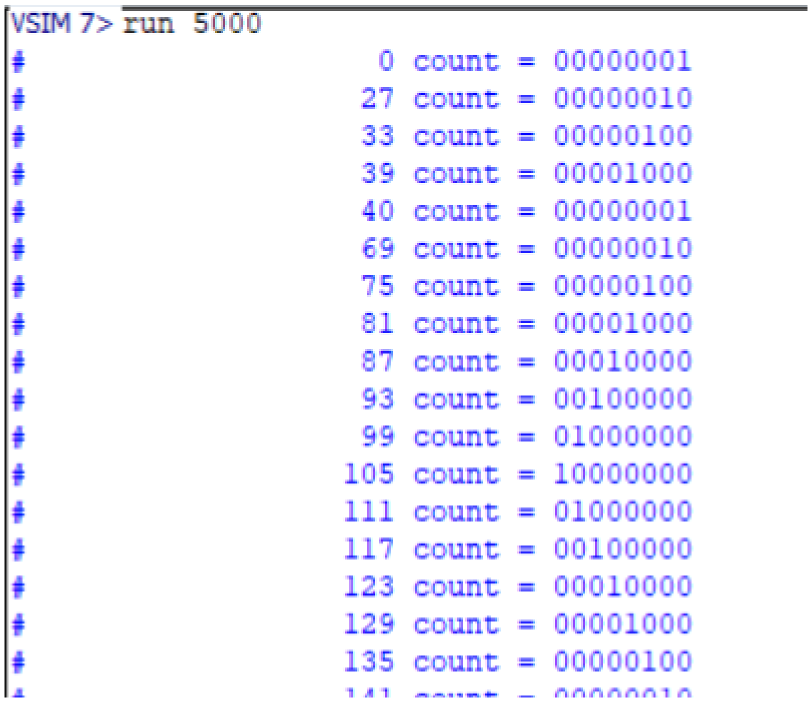
（1）设计模块

```
`timescale 10 ns / 1 ns

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

reg [4:0] state_cnter;

always @(posedge clk or posedge reset) begin
    if (reset)
        state_cnter = 5'b0;
    else if (state_cnter == 5'b10001)
        state_cnter <= 5'b00000;
    else
        state_cnter <= state_cnter + 1;
end

function [7:0] state_decoder(input [4:0] state);
    case (state)
        5'b00000: state_decoder = 8'b00000001;
        5'b00001: state_decoder = 8'b00000001;
        5'b00010: state_decoder = 8'b00000001;
        5'b00011: state_decoder = 8'b00000001;
        5'b00100: state_decoder = 8'b00000010;
        5'b00101: state_decoder = 8'b00000100;
        5'b00110: state_decoder = 8'b00001000;
        5'b00111: state_decoder = 8'b00010000;
        5'b01000: state_decoder = 8'b00100000;
        5'b01001: state_decoder = 8'b01000000;
        5'b01010: state_decoder = 8'b10000000;
        5'b01011: state_decoder = 8'b01000000;
        5'b01100: state_decoder = 8'b00100000;
        5'b01101: state_decoder = 8'b00010000;
        5'b01110: state_decoder = 8'b00001000;
        5'b01111: state_decoder = 8'b00000100;
        5'b10000: state_decoder = 8'b00000010;
        5'b10001: state_decoder = 8'b00000001;
        default: state_decoder = 8'bxxxxxxxx;
    endcase
endfunction

assign count = state_decoder(state_cnter);

endmodule
```
（2）测试模块

```
`timescale 10 ns / 1 ns
`include "t13.v"

module tb_shift_counter();

reg clk;
reg rst;
wire [7:0] count;

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

initial begin
    rst = 1'b1;
    #5 rst = 1'b0;
    #35 rst = 1'b1;
    #6 rst = 1'b0;
end

shift_counter counter(.count(count), .clk(clk), .reset(rst));

initial
    $monitor($time, "\tcount = %8b", count);

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 7> run 5000
#                    0 count = 00000001
#                   27 count = 00000010
#                   33 count = 00000100
#                   39 count = 00001000
#                   40 count = 00000001
#                   69 count = 00000010
#                   75 count = 00000100
#                   81 count = 00001000
#                   87 count = 00010000
#                   93 count = 00100000
#                   99 count = 01000000
#                  105 count = 10000000
#                  111 count = 01000000
#                  117 count = 00100000
#                  123 count = 00010000
#                  129 count = 00001000
#                  135 count = 00000100
#                  141 count = 00000010
```

第 14 题：

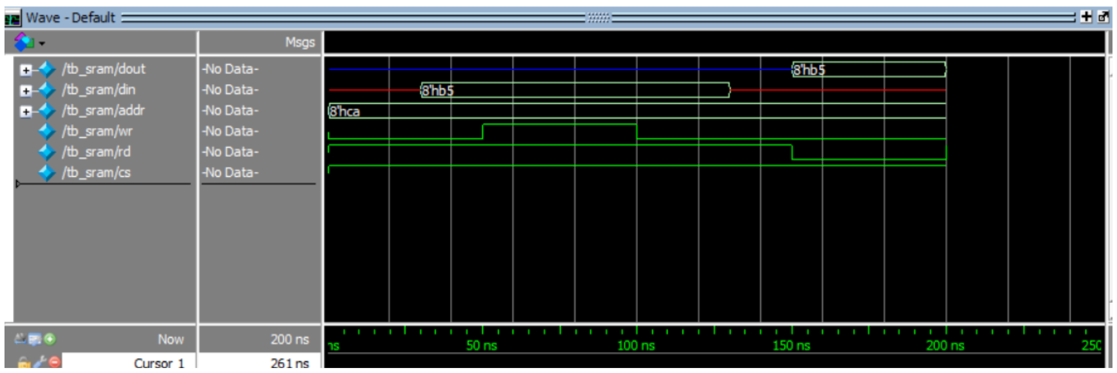
（1） 设计模块

```
    `timescale 10 ns / 1 ns

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

reg [7:0] ram[0:255];
reg [7:0] data;

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

always @(posedge wr) begin
    if (cs && wr && rd) begin
        ram[addr] <= din;
    end
end

always @(negedge rd) begin
    if (cs && !rd) begin
        data <= ram[addr];
    end
end

endmodule
```

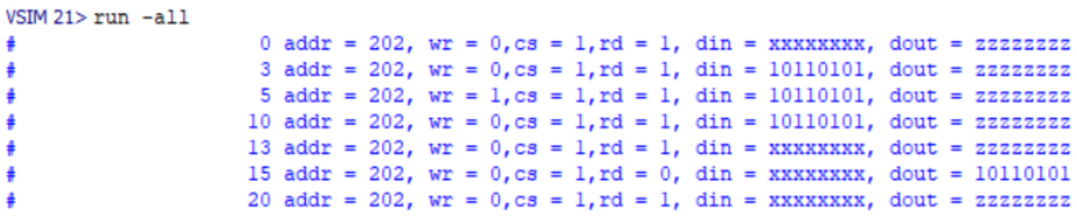
（2）测试模块

```
`timescale 10 ns / 1 ns
`include "t14.v"

module tb_sram();

wire [7:0] dout;
reg [7:0] din;
reg [7:0] addr;
reg wr;
reg rd;
reg cs;

initial begin
    addr = 8'b1100_1010;
end

initial begin
    cs = 1'b1;
    #3 din = 8'b1011_0101;
    #10 din = 8'bxxxx_xxxx;
end

initial begin
    wr = 1'b0;
    #5 wr = 1'b1;
    #5 wr = 1'b0;
end

initial begin
    rd = 1'b1;
    #15 rd = 1'b0;
    #5 rd = 1'b1;
end

sram sr(.dout(dout), .din(din), .addr(addr), .wr(wr), .rd(rd), .cs(cs));

initial
    $monitor($time, "\taddr = %d, wr = %b,cs = %b,rd = %b, din = %8b, dout
= %8b", addr, wr,cs, rd, din, dout);

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 21> run -all
#                    0 addr = 202, wr = 0,cs = 1,rd = 1, din = xxxxxxxx, dout = zzzzzzzz
#                    3 addr = 202, wr = 0,cs = 1,rd = 1, din = 10110101, dout = zzzzzzzz
#                    5 addr = 202, wr = 1,cs = 1,rd = 1, din = 10110101, dout = zzzzzzzz
#                   10 addr = 202, wr = 0,cs = 1,rd = 1, din = 10110101, dout = zzzzzzzz
#                   13 addr = 202, wr = 0,cs = 1,rd = 1, din = xxxxxxxx, dout = zzzzzzzz
#                   15 addr = 202, wr = 0,cs = 1,rd = 0, din = xxxxxxxx, dout = 10110101
#                   20 addr = 202, wr = 0,cs = 1,rd = 1, din = xxxxxxxx, dout = zzzzzzzz
```

（5）设计说明（可选）：如果有需要说明的部分。

这里 cs 都设置为 1。

第 15 题：

（1）设计模块

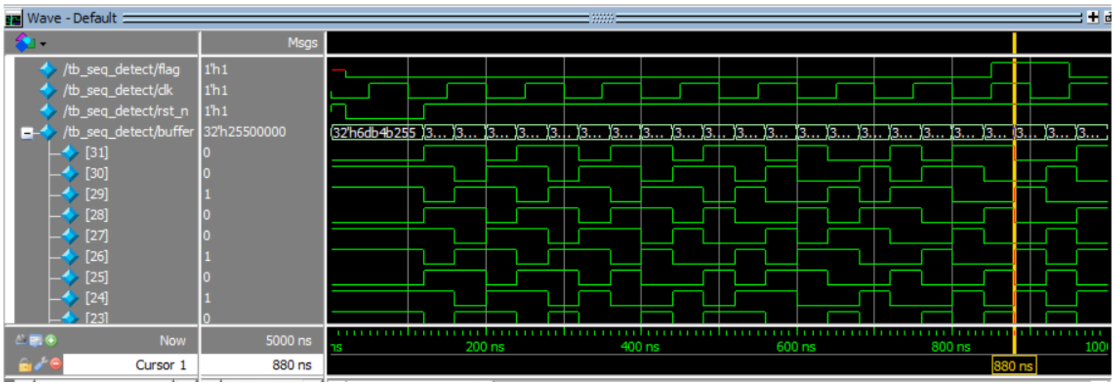
```
`timescale 10 ns / 1 ns

module seq_detect(
    output flag,
    input din,
    input clk,
    input rst_n
);

reg [8:0] current_state;
reg [8:0] next_state;

parameter idle = 9'b000000001;
parameter S1 = 9'b000000010;
parameter S3 = 9'b000000100;
parameter S5 = 9'b000001000;
parameter S7 = 9'b000010000;
parameter S0 = 9'b000100000;
parameter S2 = 9'b001000000;
parameter S4 = 9'b010000000;
parameter S6 = 9'b100000000;

always@(posedge clk or negedge rst_n)
begin
    if(~rst_n)
        current_state <= idle;
    else
        current_state <= next_state;
end

always @(*)
begin
    case (current_state)
        idle: next_state = din ? S1 : S0;
        S1: next_state = din ? S3 : S0;
        S3: next_state = din ? S3 : S5;
        S5: next_state = din ? S7 : S0;
        S7: next_state = din ? S4 : S0;
        S0: next_state = din ? S2 : S0;
        S2: next_state = din ? S4 : S0;
        S4: next_state = din ? S3 : S6;
        S6: next_state = din ? S7 : S0;
        default: next_state =  idle;
    endcase
end

assign flag = ((current_state == S7)|(current_state == S6))? 1'b1 : 1'b0;

endmodule
```

（2）测试模块

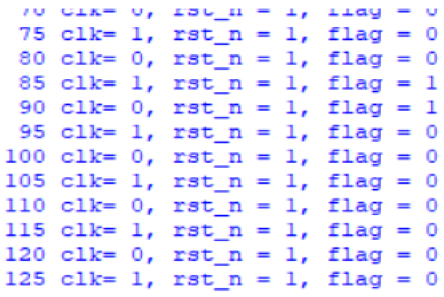
```
`timescale 10 ns / 1 ns
`include "t15.v"

module tb_seq_detect();

wire flag;
reg clk;
reg rst_n;
integer i;
reg [31:0] buffer = 32'b0110_1101_1011_0100_1011_0010_0101_0101;
reg din;

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

initial begin
    rst_n = 1'b1;
    #2 rst_n = 1'b0;
    #10 rst_n = 1'b1;
end

initial
begin
  din=0;
  #8 for(i=0;i<31;i=i+1)
  begin
    #4 din = buffer[31];
    buffer = buffer << 1;
end
```

```
end

seq_detect dectect(.flag(flag), .din(din), .clk(clk), .rst_n(rst_n));

initial
$monitor($time, "\tclk= %b, rst_n = %b, din = %b, flag = %b",clk, rst_n,
buffer, flag);
//$monitor($time, "\taddr = %d, wr = %b, rd = %b, din = %8b, dout = %8b",
addr, wr, rd, din, dout);

endmodule
```

（3）测试波形图：如果很多，可以提供部分波形内容；

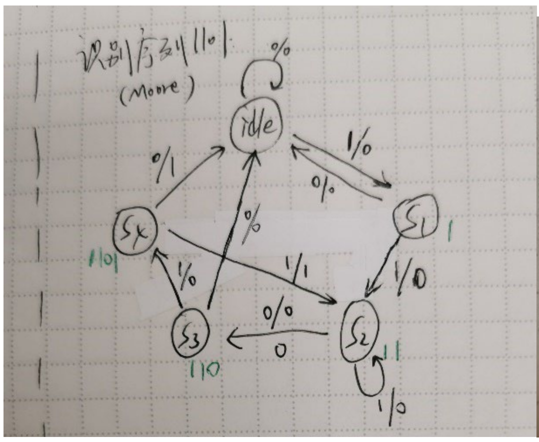


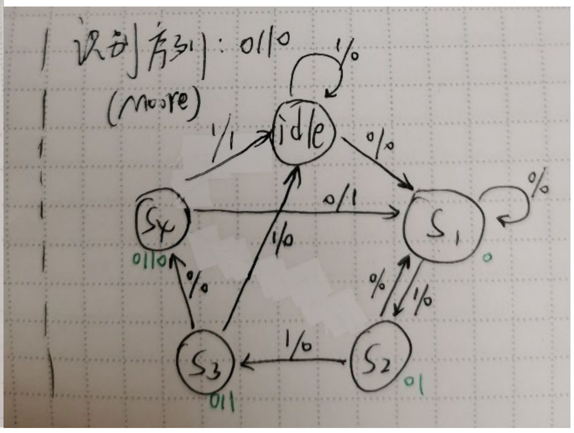
（4）显示输出（可选）：如果需要显示输出来说明模块设计的正确性；

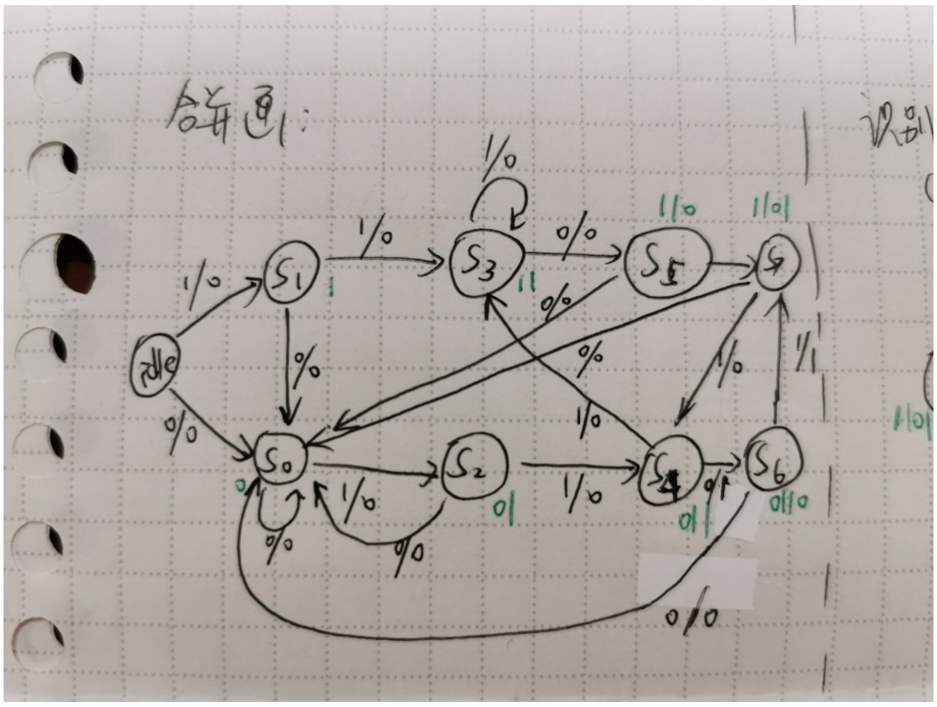


```
 70 clk= 0, rst_n = 1, flag = 0
 75 clk= 1, rst_n = 1, flag = 0
 80 clk= 0, rst_n = 1, flag = 0
 85 clk= 1, rst_n = 1, flag = 1
 90 clk= 0, rst_n = 1, flag = 1
 95 clk= 1, rst_n = 1, flag = 0
100 clk= 0, rst_n = 1, flag = 0
105 clk= 1, rst_n = 1, flag = 0
110 clk= 0, rst_n = 1, flag = 0
115 clk= 1, rst_n = 1, flag = 0
120 clk= 0, rst_n = 1, flag = 0
125 clk= 1, rst_n = 1, flag = 0
```

（5）设计说明（可选）：如果有需要说明的部分。







本题需要画出状态转移图如上。

参考：**https://blog.csdn.net/weixin_43727437/article/details/104654195**

第 16 题：

A）

Mealy:

（1）设计模块

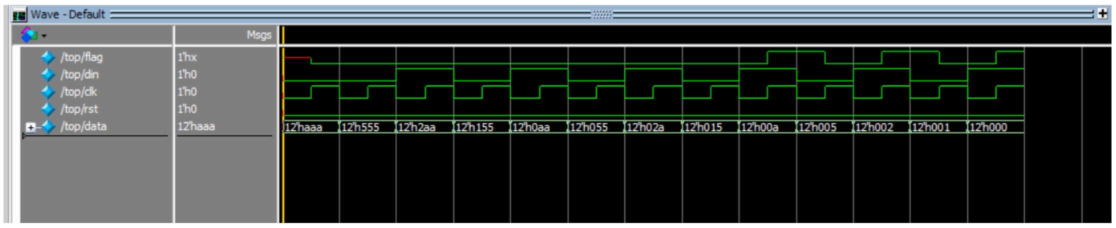
```
`timescale 10ns/1ns

module mealy(output reg flag, input din, clk, rst);
  reg[2:0] current;
  reg[2:0] next;

  parameter A=0;
  parameter B=1;
  parameter C=2;
  parameter D=3;
  parameter E=4;
  parameter F=5;
  parameter G=6;
  parameter H=7;

  always @(posedge clk or posedge rst)
  begin
    if(rst)
      current<=A;
    else
      current<=next;
  end

  always @(*)
  begin
  case (current)
  A: next = din ? A : B;
  B: next = din ? C : B;
  C: next = din ? A : D;
  D: next = din ? E : B;
  E: next = din ? A : F;
  F: next = din ? G : B;
  G: next = din ? A : H;
  H: next = din ? G : B;
  default : next = A;
```

```
endcase
end

always @(posedge clk or posedge rst)
begin
  if(rst)
    flag <=0;
  else if(current == H && din==1)
    flag<=1;
  else
    flag<=0;
end

endmodule
```

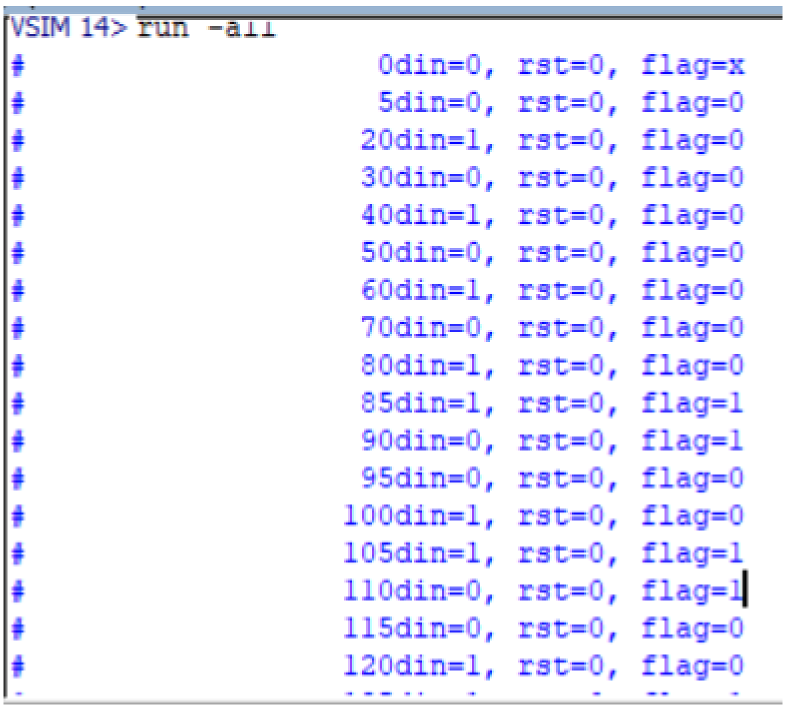
（2）测试模块

```
`timescale 10ns / 1ns
`include "t16_mealy.v"
//`include "t16_moore.v"

module top();
  wire flag;
  reg din,clk,rst;
  reg [11:0] data;

  //moore mymoore(.flag(flag), .din(din), .clk(clk), .rst(rst));
  mealy mymealy(.flag(flag), .din(din), .clk(clk), .rst(rst));

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

initial
begin
  data = 12'b101010101010;
  din = 0;
  repeat(12)
  begin
    #10 din = data & 1;
    data = data >> 1;
  end
  #10 $stop;
end

initial
begin
$monitor($time, "din=%b, rst=%b, flag=%b", din, rst, flag);
end
endmodule
```

（3）测试波形图：如果很多，可以提供部分波形内容；

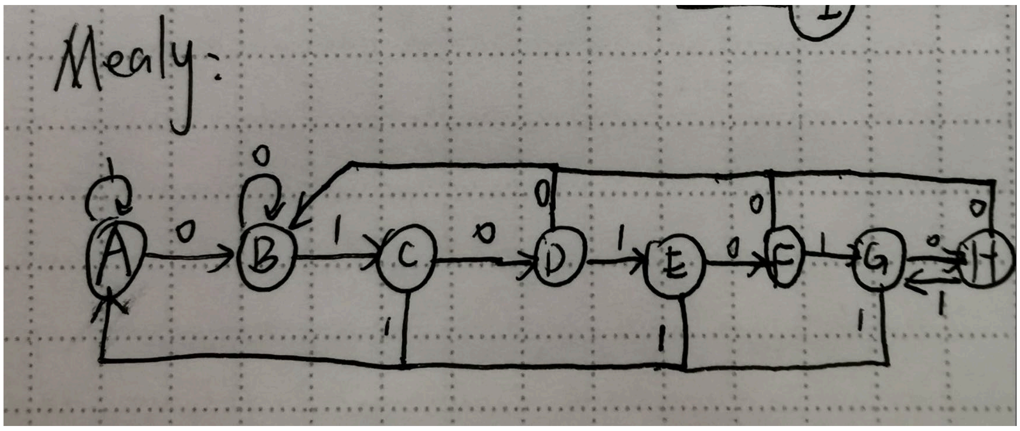


（4）显示输出（可选）：如果需要显示输出来说明模块设计的正确性；



```
VSIM 14> run -all
#                    0din=0, rst=0, flag=x
#                    5din=0, rst=0, flag=0
#                   20din=1, rst=0, flag=0
#                   30din=0, rst=0, flag=0
#                   40din=1, rst=0, flag=0
#                   50din=0, rst=0, flag=0
#                   60din=1, rst=0, flag=0
#                   70din=0, rst=0, flag=0
#                   80din=1, rst=0, flag=0
#                   85din=1, rst=0, flag=1
#                   90din=0, rst=0, flag=1
#                   95din=0, rst=0, flag=0
#                  100din=1, rst=0, flag=0
#                  105din=1, rst=0, flag=1
#                  110din=0, rst=0, flag=1
#                  115din=0, rst=0, flag=0
#                  120din=1, rst=0, flag=0
```

（5）设计说明（可选）：如果有需要说明的部分。



B）

Moore:

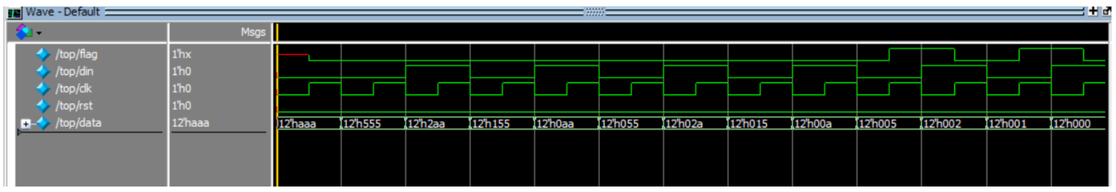
（1）设计模块

```
`timescale 10 ns / 1 ns

module moore(
    output reg flag,
    input din, clk, rst
);

reg [3:0] state;
reg [3:0] next;

parameter A = 0;
parameter B = 1;
parameter C = 2;
parameter D = 3;
parameter E = 4;
parameter F = 5;
parameter G = 6;
parameter H = 7;
parameter I = 8;


 always @(posedge clk or posedge rst)
  begin
    if(rst)
      state <= A;
    else
      state<=next;
  end

always @(*)
begin
        case (state)
            A: next <= din ? A : B;
            B: next <= din ? C : B;
            C: next <= din ? A : D;
          D: next <= din ? E : B;
            E: next <= din ? A : F;
            F: next <= din ? G : B;
            G: next <= din ? A : H;
            H: next <= din ? I : B;
            I: next <= din ? A : H;
            default: next = A;
        endcase
end


always @(posedge clk or posedge rst)
begin
  if(rst)
    flag <= 0;
  else if(state == I)
    flag<=1;
  else
    flag<=0;
  end

endmodule
```

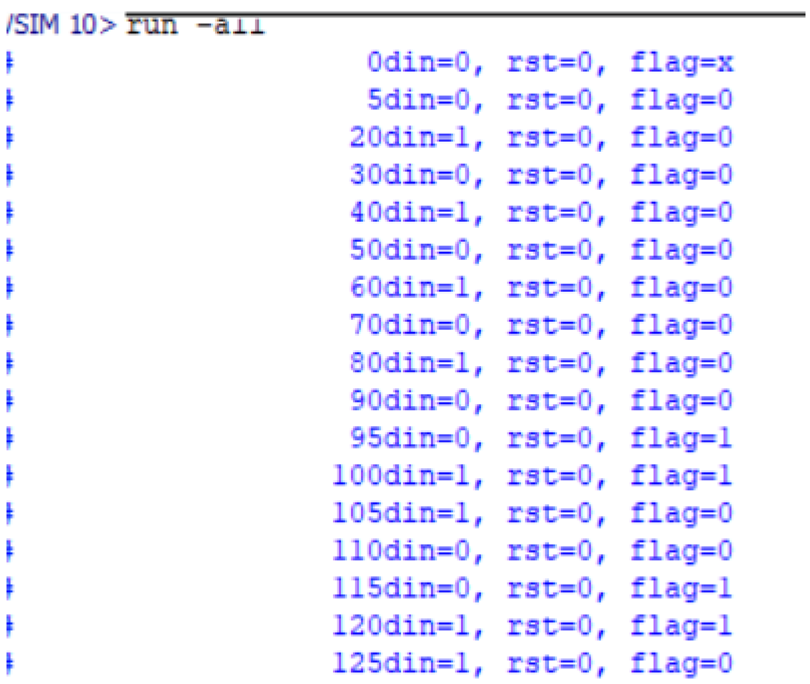
（2）测试模块

```
`timescale 10ns / 1ns
`include "t16_moore.v"
//`include "t16_mealy.v"

module top();
  wire flag;
  reg din,clk,rst;
  reg [11:0] data;

 // mealy mymealy(.flag(flag), .din(din), .clk(clk), .rst(rst));
  moore mymoore(.flag(flag), .din(din), .clk(clk), .rst(rst));

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

initial
begin
  data = 12'b101010101010;
  din = 0;
  repeat(12)
  begin
    #10 din = data & 1;
    data = data >> 1;
  end
  #10 $stop;
end

initial
begin
$monitor($time, "din=%b, rst=%b, flag=%b", din, rst, flag);
end
endmodule
```

（3）测试波形图：如果很多，可以提供部分波形内容；

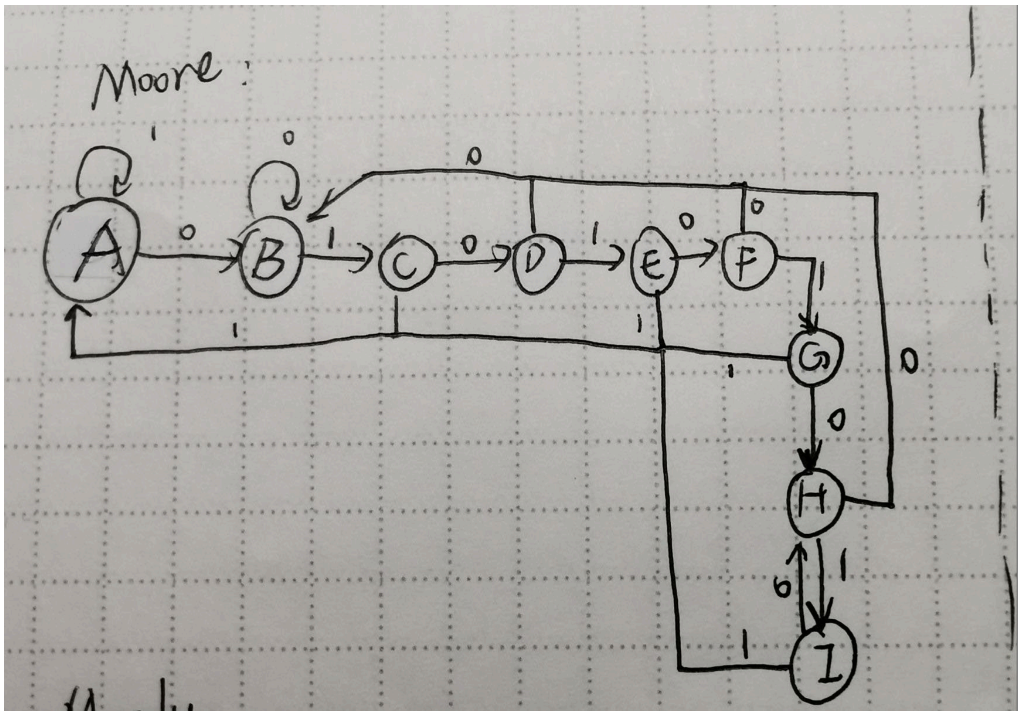


（4）显示输出（可选）：如果需要显示输出来说明模块设计的正确性；



```
VSIM 10> run -all
#                         0din=0, rst=0, flag=x
#                         5din=0, rst=0, flag=0
#                        20din=1, rst=0, flag=0
#                        30din=0, rst=0, flag=0
#                        40din=1, rst=0, flag=0
#                        50din=0, rst=0, flag=0
#                        60din=1, rst=0, flag=0
#                        70din=0, rst=0, flag=0
#                        80din=1, rst=0, flag=0
#                        90din=0, rst=0, flag=0
#                        95din=0, rst=0, flag=1
#                       100din=1, rst=0, flag=1
#                       105din=1, rst=0, flag=0
#                       110din=0, rst=0, flag=0
#                       115din=0, rst=0, flag=1
#                       120din=1, rst=0, flag=1
#                       125din=1, rst=0, flag=0
```

（5）设计说明（可选）：如果有需要说明的部分。


Moore:
A
B
C
D
E
F
G
H
I
0
1