数字系统设计作业

学号：518021910690　姓名：刘子涵　日期：2020/12/17

第 1 题：

2.1、 设计一个 Verilog 模块，产生图 1 所示的波形。

图 1、习题 1 波形图

要求：

（1）时间单位为：10ns；时间精度为：1ns；

（2）使用 initial 语句；模块名为：

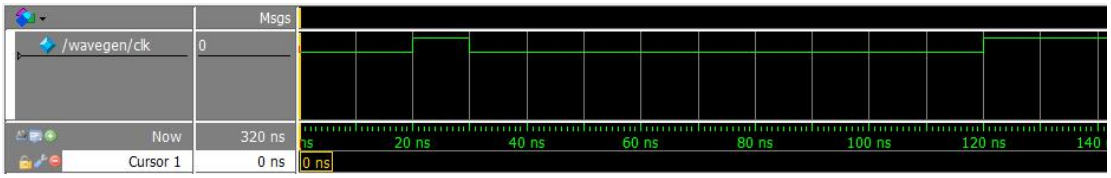
wavegen()

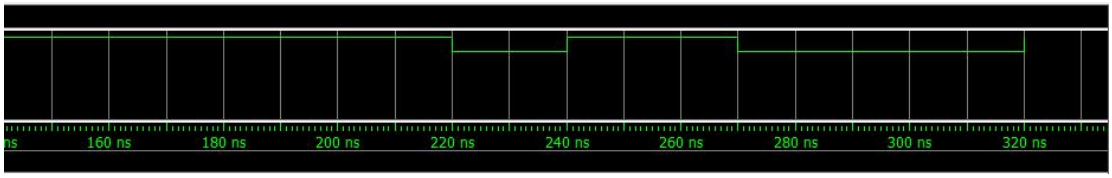
（1）设计模块

```
// File: wavegen.v

`timescale 10ns/1ns

module wavegen;

    reg wave;

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

endmodule
```

（2）测试模块（包含在设计模块中）

（3）测试波形图





第 2 题：

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

要求：

（1）在模块设计中要求明确标明基数格式。;

（2）设计模块名为：

Encoder8x3(code, data)

测试平台的模块名为：

tb_Encoder8x3 ()

表 1、8:3 编码器真值表

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

（1）设计模块

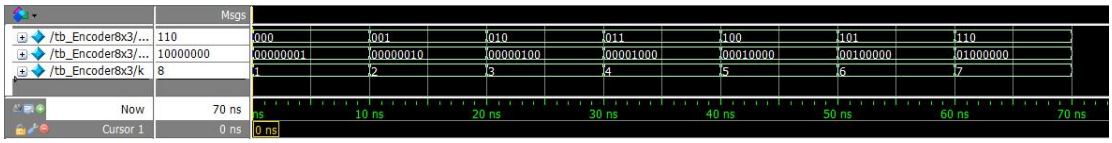
```
// File: Encoder8x3.v

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

    always @(*) begin
        case(data)
            8'b0000_0001: code = 3'd0;
            8'b0000_0010: code = 3'd1;
            8'b0000_0100: code = 3'd2;
            8'b0000_1000: code = 3'd3;
            8'b0001_0000: code = 3'd4;
            8'b0010_0000: code = 3'd5;
            8'b0100_0000: code = 3'd6;
            8'b1000_0000: code = 3'd7;
            default: code = 3'bx;
        endcase
    end
```

```
endmodule
```

（2）测试模块

```
// File: tb_Encoder8x3.v

`timescale 10 ns / 1 ns
`include "Encoder8x3.v"

module tb_Encoder8x3;

    parameter STEP = 7;

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

    Encoder8x3 a(.code(code), .data(data));

    initial begin
        data = 8'b0000_0001;
        for ( k=1; k<=STEP; k=k+1 )
            #1 data = data << 1;
    end

    initial begin
        $monitor("At time %4t, data=%b, code=%d", $time, data, code);
    end

endmodule
```

（3）测试波形图



（4）显示输出

```
# At time    0, data=00000001, code=0
# At time   10, data=00000010, code=1
# At time   20, data=00000100, code=2
# At time   30, data=00001000, code=3
# At time   40, data=00010000, code=4
# At time   50, data=00100000, code=5
# At time   60, data=01000000, code=6
# At time   70, data=10000000, code=7
```

第 3 题：

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

要求：

设计模块名为：

mux2x1(dout, sel, din)

测试平台的模块名为：

tb_ mux2x1 ()

b）以（a）设计的 2 选 1 多路选择器为底层模块，通过调用 2 选 1 多路选择器模块，设计一个 4 选 1 多路选择器，并给出仿真测试结果。

要求：

设计模块名为：

mux4x1(dout, sel, din)

测试平台的模块名为：

tb_ mux4x1 ()

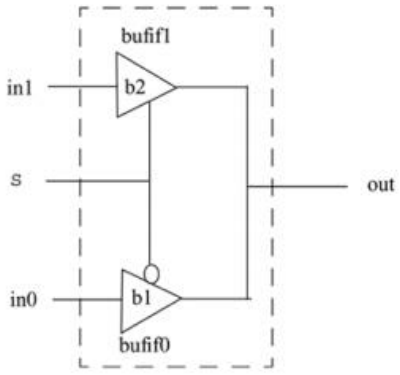


图 2、二选一多路选择器

（1）设计模块

```
// File: mux2x1.v

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

    bufif0 u0(dout, din[0], sel);
    bufif1 u1(dout, din[1], sel);

endmodule
```

```
// File: mux4x1.v

`include "mux2x1.v"

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

    wire w0, w1;
    mux2x1 m0(.dout(w0), .sel(sel[0]), .din(din[1:0])),
           m1(.dout(w1), .sel(sel[0]), .din(din[3:2])),
           m2(.dout(dout), .sel(sel[1]), .din({w1, w0}));

endmodule
```

（2）测试模块

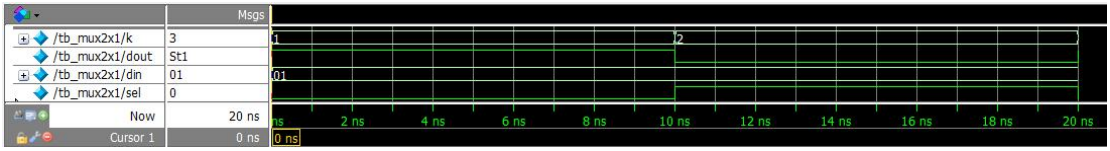
```
// File: tb_mux2x1.v

`timescale 10 ns / 1 ns
`include "mux2x1.v"

module tb_mux2x1;

    parameter STEP = 2;
    integer k;

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

    mux2x1 a(.dout(dout), .sel(sel), .din(din));

    initial begin
        sel = 1'b0;
        for ( k=1; k<=STEP; k=k+1 )
            #1 sel = ~sel;
    end

    initial begin
        din = 2'b01;
    end

    initial begin
        $monitor("At time %4t, din=%b, sel=%b, dout=%b", $time, din, sel,
                 dout);
    end

endmodule
```

```
// File: tb_mux4x1.v

`timescale 10 ns / 1 ns
`include "mux4x1.v"

module tb_mux4x1;

    parameter STEP = 3;
```

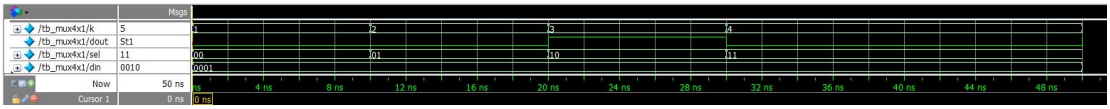
```
    integer k;

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

    mux4x1 a(.dout(dout), .sel(sel), .din(din));

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

    initial begin
        din = 4'b0001;
        for ( k=1; k<=STEP; k=k+1 )
            #5 din = din << 1;
    end

    initial begin
        $monitor("At time %4t, din=%b, sel=%b, dout=%b", $time, din, sel,
                 dout);
    end

endmodule
```

（3）测试波形图

① mux2x1 模块测试波形



② mux4x1 模块测试波形



（4）显示输出

① mux2x1 模块显示输出

```
# At time    0, din=01, sel=0, dout=1
# At time   10, din=01, sel=1, dout=0
# At time   20, din=01, sel=0, dout=1
```

② mux4x1 模块显示输出

```
# At time     0, din=0001, sel=00, dout=0
# At time    10, din=0001, sel=01, dout=0
# At time    20, din=0001, sel=10, dout=1
# At time    30, din=0001, sel=11, dout=0
# At time    50, din=0010, sel=11, dout=1
```

第 4 题：

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

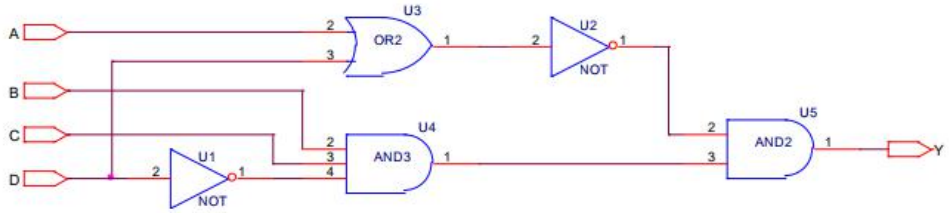


图 3、习题 4 电路原理图

要求：

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

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

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

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

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

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

（1）设计模块

```
// File: comb_str.v

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

    wire w11;
    wire w21, w22;
    wire w31;
    wire w41;

    not u1(w11, D),
        u2(w31, w21);

    or u3(w21, A, D);
```

```
    and u4(w22, B, C, w11),
        u5(w41, w31, w22);

    buf u6(Y, w41);

endmodule
```

```
// File: comb_dataflow.v

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

    wire w11;
    wire w21, w22;
    wire w31;

    assign w11 = ~D,
           w21 = A | D,
           w22 = B & C &w11,
           w31 = ~w21,
           Y = w31 & w22;

endmodule
```

```
// File: comb_behaviour.v

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

    reg r11;
    reg r21, r22;
    reg r31;

    always @(*) begin
        r11 = ~D;
        r21 = A | D;
        r22 = B & C & r11;
        r31 = ~r21;
        Y = w31 & w22;
    end

endmodule
```

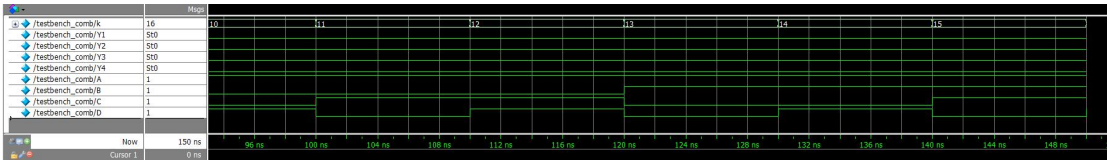
```
// File: comb_prim.v

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

    table
        // A B C D : Y ;
           0 0 ? ? : 0 ;
           0 1 0 ? : 0 ;
           0 1 1 0 : 1 ;
           0 1 1 1 : 0 ;
           1 ? ? ? : 0 ;
    endtable

endprimitive
```

（2）测试模块

```
// File: testbench_comb.v

`timescale 10 ns / 1 ns
`include "comb_str.v"
`include "comb_dataflow.v"
`include "comb_behavior.v"
`include "comb_prim.v"

module testbench_comb;

    parameter STEP = 15;
    integer k;

    wire Y1, Y2, Y3, Y4;
    reg A, B, C, D;

    comb_str a(Y1, A, B, C, D);
    comb_dataflow b(Y2, A, B, C, D);
    comb_behaviour c(Y3, A, B, C, D);
    comb_prim d(Y4, A, B, C, D);

    initial begin
        {A, B, C, D} = 4'b0;
        for ( k=1; k<=STEP; k=k+1 )
            #1 {A, B, C, D} = {A, B, C, D} + 1'b1;
    end

    initial begin
```

```
        $monitor("At time %4t, A=%b, B=%b, C=%b, D=%b, Y1=%b, Y2=%b, Y3=%b,
                  Y4=%b", $time, A, B, C, D, Y1, Y2, Y3, Y4);
    end

endmodule
```

（3）测试波形图



（4）显示输出

```
# At time     0, A=0, B=0, C=0, D=0, Y1=0, Y2=0, Y3=0, Y4=0
# At time    10, A=0, B=0, C=0, D=1, Y1=0, Y2=0, Y3=0, Y4=0
# At time    20, A=0, B=0, C=1, D=0, Y1=0, Y2=0, Y3=0, Y4=0
# At time    30, A=0, B=0, C=1, D=1, Y1=0, Y2=0, Y3=0, Y4=0
# At time    40, A=0, B=1, C=0, D=0, Y1=0, Y2=0, Y3=0, Y4=0
# At time    50, A=0, B=1, C=0, D=1, Y1=0, Y2=0, Y3=0, Y4=0
# At time    60, A=0, B=1, C=1, D=0, Y1=1, Y2=1, Y3=1, Y4=1
# At time    70, A=0, B=1, C=1, D=1, Y1=0, Y2=0, Y3=0, Y4=0
# At time    80, A=1, B=0, C=0, D=0, Y1=0, Y2=0, Y3=0, Y4=0
# At time    90, A=1, B=0, C=0, D=1, Y1=0, Y2=0, Y3=0, Y4=0
# At time   100, A=1, B=0, C=1, D=0, Y1=0, Y2=0, Y3=0, Y4=0
# At time   110, A=1, B=0, C=1, D=1, Y1=0, Y2=0, Y3=0, Y4=0
# At time   120, A=1, B=1, C=0, D=0, Y1=0, Y2=0, Y3=0, Y4=0
# At time   130, A=1, B=1, C=0, D=1, Y1=0, Y2=0, Y3=0, Y4=0
# At time   140, A=1, B=1, C=1, D=0, Y1=0, Y2=0, Y3=0, Y4=0
# At time   150, A=1, B=1, C=1, D=1, Y1=0, Y2=0, Y3=0, Y4=0
```

第 5 题：

2.5、根据下面的的布尔方程，设计两个组合逻辑电路模块：

i） $Y1(A,B,C)=\Sigma m(1,2,4,5)$

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

要求：

采用数据流（Dataflow）方式，利用连续幅值语句设计，并设计测试平台进行仿真验证。

（1）两个设计模块名为：

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

测试平台的模块名分别为：

`tb_comb_Y1()`，`tb_comb_Y2()`

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

（1）设计模块

```
// File: comb_Y1.v

module comb_Y1(output Y, input A, B, C);

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

endmodule
```

```
// File: comb_Y2.v

module comb_Y2(output Y, input A, B, C, D);

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

endmodule
```

（2）测试模块

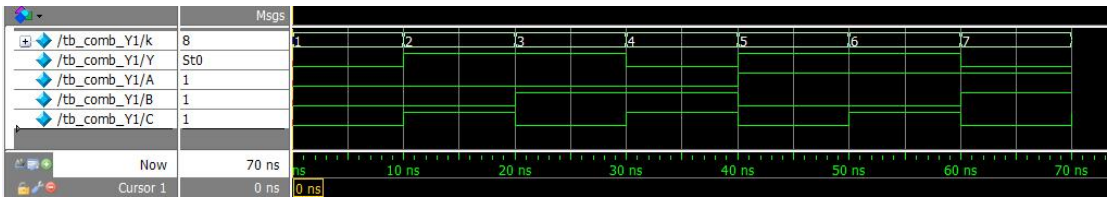
```
// File: tb_comb_Y1.v

`timescale 10 ns / 1 ns
`include "comb_Y1.v"

module tb_comb_Y1;

    parameter STEP = 7;
    integer k;

    wire Y;
    reg A, B, C;

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

    initial begin
        {A, B, C} = 3'b0;
        for ( k=1; k<=STEP; k=k+1 )
            #1 {A, B, C} = {A, B, C} + 1'b1;
    end
```

```
    initial begin
        $monitor("At time %4t, A=%b, B=%b, C=%b, Y=%b", $time, A, B, C, Y);
    end

endmodule
```

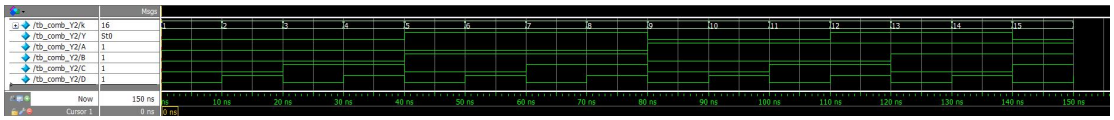
```
// File: tb_comb_Y2.v

`timescale 10 ns / 1 ns
`include "comb_Y2.v"

module tb_comb_Y2;

    parameter STEP = 15;
    integer k;

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

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

    initial begin
        {A, B, C, D} = 4'b0;
        for ( k=1; k<=STEP; k=k+1 )
            #1 {A, B, C, D} = {A, B, C, D} + 1'b1;
    end

    initial begin
        $monitor("At time %4t, A=%b, B=%b, C=%b, D=%b, Y=%b", $time, A, B,
                 C, D, Y);
    end

endmodule
```

（3）测试波形图

① **comb_Y1 模块测试波形**



② **comb_Y2 模块测试波形**



（4）显示输出

① comb_Y1 模块显示输出

```
# At time     0, A=0, B=0, C=0, Y=0
# At time    10, A=0, B=0, C=1, Y=1
# At time    20, A=0, B=1, C=0, Y=1
# At time    30, A=0, B=1, C=1, Y=0
# At time    40, A=1, B=0, C=0, Y=1
# At time    50, A=1, B=0, C=1, Y=1
# At time    60, A=1, B=1, C=0, Y=0
# At time    70, A=1, B=1, C=1, Y=0
```

② comb_Y2 模块显示输出

```
# At time     0, A=0, B=0, C=0, D=0, Y=0
# At time    10, A=0, B=0, C=0, D=1, Y=0
# At time    20, A=0, B=0, C=1, D=0, Y=0
# At time    30, A=0, B=0, C=1, D=1, Y=0
# At time    40, A=0, B=1, C=0, D=0, Y=1
# At time    50, A=0, B=1, C=0, D=1, Y=1
# At time    60, A=0, B=1, C=1, D=0, Y=1
# At time    70, A=0, B=1, C=1, D=1, Y=1
# At time    80, A=1, B=0, C=0, D=0, Y=0
# At time    90, A=1, B=0, C=0, D=1, Y=0
# At time   100, A=1, B=0, C=1, D=0, Y=0
# At time   110, A=1, B=0, C=1, D=1, Y=1
# At time   120, A=1, B=1, C=0, D=0, Y=1
# At time   130, A=1, B=1, C=0, D=1, Y=1
# At time   140, A=1, B=1, C=1, D=0, Y=0
# At time   150, A=1, B=1, C=1, D=1, Y=0
```

第 6 题：

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

要求：

（1）设计模块名为：

ones_count(count, dat_in)

这里，count 是 4 位输出端口，dat_in 为 8 位输入端口；

测试平台的模块名为：

tb_ones_count()

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

（1）设计模块

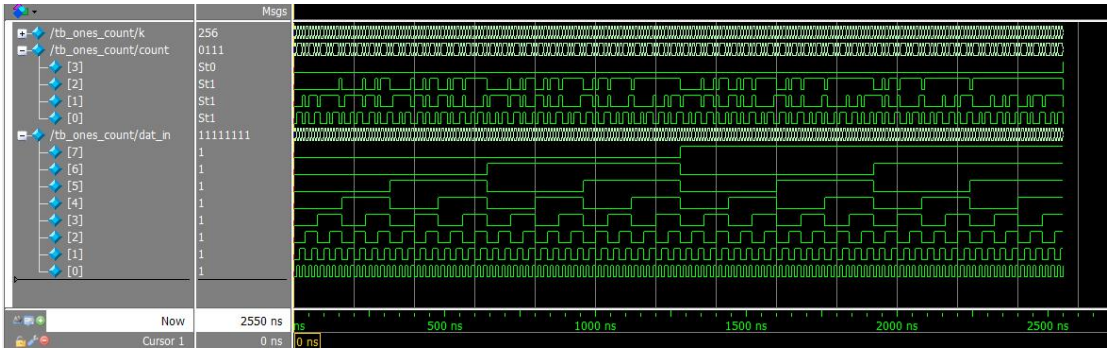
```
// File: ones_count.v

module ones_count(output reg [3:0] count, input [7:0] dat_in);

    integer k;
```

```
    always @(*) begin
        count = 4'd0;
        for(k=0; k<=7; k=k+1)
            count = count + dat_in[k];
    end

endmodule
```

（2）测试模块

```
// File: tb_ones_count.v

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

module tb_ones_count;

    parameter STEP = 255;
    integer k;

    wire [3:0] count;
    reg [7:0] dat_in;

    ones_count a(count, dat_in);

    initial begin
        dat_in = 8'b0;
        for ( k=1; k<=STEP; k=k+1 )
            #1 dat_in = dat_in + 1'b1;
    end

    initial begin
        $monitor("At time %4t, dat_in=%b, count=%b", $time, dat_in, count);
    end

endmodule
```

（3）测试波形图



（4）显示输出

```
# At time    0, dat_in=00000000, count=0000
# At time   10, dat_in=00000001, count=0001
# At time   20, dat_in=00000010, count=0001
# At time   30, dat_in=00000011, count=0010
# At time   40, dat_in=00000100, count=0001
# At time   50, dat_in=00000101, count=0010
# At time   60, dat_in=00000110, count=0010
# At time   70, dat_in=00000111, count=0011
# At time   80, dat_in=00001000, count=0001
# At time   90, dat_in=00001001, count=0010
# At time  100, dat_in=00001010, count=0010
# At time  110, dat_in=00001011, count=0011
# At time  120, dat_in=00001100, count=0010
# At time  130, dat_in=00001101, count=0011
# At time  140, dat_in=00001110, count=0011
# At time  150, dat_in=00001111, count=0100
# At time  160, dat_in=00010000, count=0001
# At time  170, dat_in=00010001, count=0010
# At time  180, dat_in=00010010, count=0010
# At time  190, dat_in=00010011, count=0011
# At time  200, dat_in=00010100, count=0010
# At time  210, dat_in=00010101, count=0011
# At time  220, dat_in=00010110, count=0011
# At time  230, dat_in=00010111, count=0100
# At time  240, dat_in=00011000, count=0010
# At time  250, dat_in=00011001, count=0011
# At time  260, dat_in=00011010, count=0011
# At time  270, dat_in=00011011, count=0100
# At time  280, dat_in=00011100, count=0011
# At time  290, dat_in=00011101, count=0100
# At time  300, dat_in=00011110, count=0100
# At time  310, dat_in=00011111, count=0101
```

第 7 题：

2.7、设计一个 10 进制计数器的 Verilog 模块。

要求：

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

（2）设计测试模块对其进行仿真；

（3）设计模块名为：

dec_counter(count, clk, reset)

测试平台的模块名为：

tb_dec_counter()

（1）设计模块

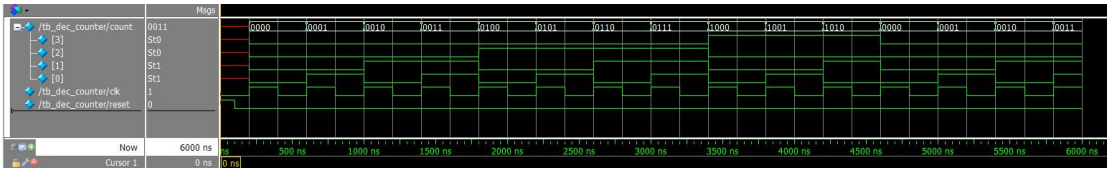
```
// File: dec_counter.v

module dec_counter(output reg [3:0] count, input clk, reset);

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

endmodule
```

（2）测试模块

```
// File: tb_dec_counter.v

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

module tb_dec_counter;

    wire [3:0] count;
    reg clk, reset;

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

    initial begin : clk_loop
        clk = 1'b0;
        forever #20 clk = ~clk;
    end

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

    initial begin
        $monitor("At time %4t, reset=%b, count=%d", $time, reset, count);
    end
```

```
    initial begin
        #600 disable clk_loop;
    end

endmodule
```

（3）测试波形图

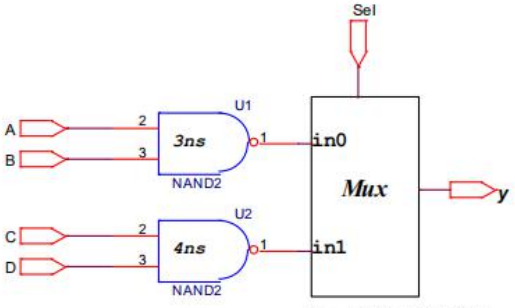


（4）显示输出

```
# At time     0, reset=1, count= x
# At time   100, reset=0, count= x
# At time   200, reset=0, count= 0
# At time   600, reset=0, count= 1
# At time  1000, reset=0, count= 2
# At time  1400, reset=0, count= 3
# At time  1800, reset=0, count= 4
# At time  2200, reset=0, count= 5
# At time  2600, reset=0, count= 6
# At time  3000, reset=0, count= 7
# At time  3400, reset=0, count= 8
# At time  3800, reset=0, count= 9
# At time  4200, reset=0, count=10
# At time  4600, reset=0, count= 0
# At time  5000, reset=0, count= 1
# At time  5400, reset=0, count= 2
# At time  5800, reset=0, count= 3
```

第 8 题：

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



Mux 多路复用器真值表

sel	y
0	in0
1	in1

图 4、习题 8 电路原理图

要求：

（1）设计模块名为：

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

测试平台的模块名为：

`tb_comb_str()`

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

（1）设计模块

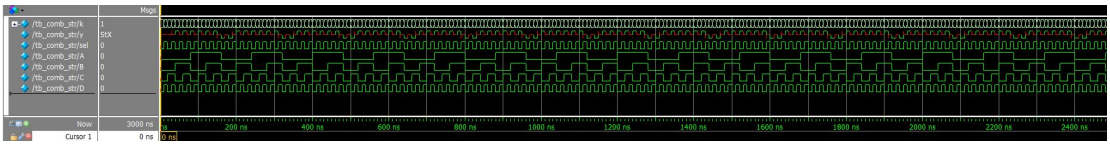
```
// File: comb_str.v

`timescale 1ns/1ns

module comb_str(output y, input sel, A, B, C, D);

    wire w0, w1;

    nand #3 u1(w0, A, B),
         #4 u2(w1, C, D);

    bufif0 u3(y, w0, sel);
    bufif1 u4(y, w1, sel);

endmodule
```

（2）测试模块

```
// File: tb_comb_str.v

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

module tb_comb_str;

    parameter STEP = 255;
    integer k;

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

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

    initial begin : clk_loop
        sel = 1'b0;
        forever #1 sel = ~sel;
    end

    initial begin
        {A, B, C, D} = 4'b0;
        for ( k=1; k<=STEP; k=k+1 )
            #1 {A, B, C, D} = {A, B, C, D} + 1'b1;
    end
```

```
    initial begin
        $monitor("At time %4t, sel=%b, A=%b, B=%b, C=%b, D=%b, y=%b",
                 $time, sel, A, B, C, D, y);
    end

    initial begin
        #300 disable clk_loop;
    end

endmodule
```

（3）测试波形图



（4）显示输出

```
# At time    0, sel=0, A=0, B=0, C=0, D=0, y=x
# At time   10, sel=1, A=0, B=0, C=0, D=1, y=x
# At time   20, sel=0, A=0, B=0, C=1, D=0, y=x
# At time   30, sel=1, A=0, B=0, C=1, D=1, y=x
# At time   40, sel=0, A=0, B=1, C=0, D=0, y=1
# At time   50, sel=1, A=0, B=1, C=0, D=1, y=x
# At time   60, sel=0, A=0, B=1, C=1, D=0, y=1
# At time   70, sel=1, A=0, B=1, C=1, D=1, y=x
# At time   80, sel=0, A=1, B=0, C=0, D=0, y=1
# At time   90, sel=1, A=1, B=0, C=0, D=1, y=x
# At time  100, sel=0, A=1, B=0, C=1, D=0, y=1
# At time  110, sel=1, A=1, B=0, C=1, D=1, y=x
# At time  120, sel=0, A=1, B=1, C=0, D=0, y=1
# At time  130, sel=1, A=1, B=1, C=0, D=1, y=x
# At time  140, sel=0, A=1, B=1, C=1, D=0, y=1
# At time  150, sel=1, A=1, B=1, C=1, D=1, y=x
# At time  160, sel=0, A=0, B=0, C=0, D=0, y=0
# At time  170, sel=1, A=0, B=0, C=0, D=1, y=x
# At time  180, sel=0, A=0, B=0, C=1, D=0, y=0
# At time  190, sel=1, A=0, B=0, C=1, D=1, y=x
# At time  200, sel=0, A=0, B=1, C=0, D=0, y=1
# At time  210, sel=1, A=0, B=1, C=0, D=1, y=x
# At time  220, sel=0, A=0, B=1, C=1, D=0, y=1
# At time  230, sel=1, A=0, B=1, C=1, D=1, y=x
# At time  240, sel=0, A=1, B=0, C=0, D=0, y=1
# At time  250, sel=1, A=1, B=0, C=0, D=1, y=x
# At time  260, sel=0, A=1, B=0, C=1, D=0, y=1
# At time  270, sel=1, A=1, B=0, C=1, D=1, y=x
# At time  280, sel=0, A=1, B=1, C=0, D=0, y=1
# At time  290, sel=1, A=1, B=1, C=0, D=1, y=x
# At time  300, sel=0, A=1, B=1, C=1, D=0, y=1
# At time  310, sel=1, A=1, B=1, C=1, D=1, y=x
# At time  320, sel=0, A=0, B=0, C=0, D=0, y=0
```

第 9 题：

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

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

这里：

输出：26 位伪随机数 q

输入：clk —— 时钟信号

rst_n —— 同步复位信号，低电平有效

load —— 加载控制，当 load=1’b1，

din —— 26 位输入，不全为 0 时，din ➔ q

要求：

（1）设计模块名为：

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

测试平台的模块名为：

tb_LFSR()

（2）并给出测试模块和测试分析结果。

（1）设计模块

```
// File: LFSR.v

module LFSR(output reg [1:26] q,
            input clk,
            input rst_n,
            input load,
            input [1:26] din
            );

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

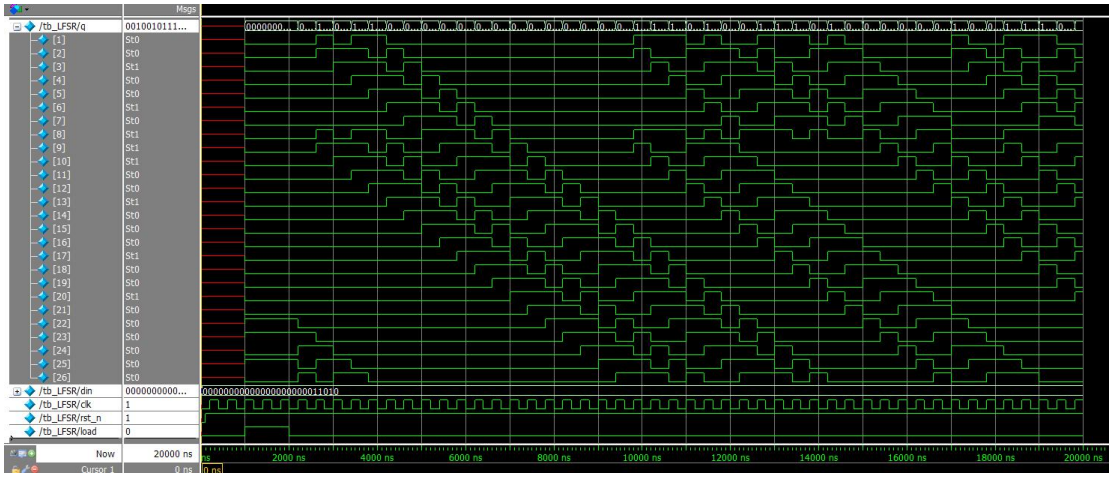
```
                q[3:7] <= q[2:6];
                q[2] <= q[1] ^ q[26];
                q[1] <= q[26];
            end
        end
    end

endmodule
```

（2）测试模块

```
// File: tb_LFSR.v

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

module tb_LFSR;

    wire [1:26] q;
    reg [1:26] din;
    reg clk, rst_n, load;

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

    initial begin : clk_loop
        clk = 1'b0;
        forever #20 clk = ~clk;
    end

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

    initial begin
        load = 1'b0;
        din = 26'b1_1010;
        #100 load = 1'b1;
        #100 load = 1'b0;
    end

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

```
    initial begin
        #2000 disable clk_loop;
    end

endmodule
```

（3）测试波形图



（4）显示输出

```
# At time    0, rst_n=0, load=0, q=       x
# At time  100, rst_n=1, load=0, q=       x
# At time 1000, rst_n=1, load=1, q=      26
# At time 2000, rst_n=1, load=0, q=      26
# At time 2200, rst_n=1, load=0, q=      13
# At time 2600, rst_n=1, load=0, q=50724870
# At time 3000, rst_n=1, load=0, q=25362435
# At time 3400, rst_n=1, load=0, q=63406081
# At time 3800, rst_n=1, load=0, q=48611328
# At time 4200, rst_n=1, load=0, q=24305664
# At time 4600, rst_n=1, load=0, q=12152832
# At time 5000, rst_n=1, load=0, q= 6076416
# At time 5400, rst_n=1, load=0, q= 3038208
# At time 5800, rst_n=1, load=0, q= 1519104
# At time 6200, rst_n=1, load=0, q=  759552
# At time 6600, rst_n=1, load=0, q=  379776
# At time 7000, rst_n=1, load=0, q=  189888
# At time 7400, rst_n=1, load=0, q=   94944
# At time 7800, rst_n=1, load=0, q=   47472
# At time 8200, rst_n=1, load=0, q=   23736
# At time 8600, rst_n=1, load=0, q=   11868
# At time 9000, rst_n=1, load=0, q=    5934
```

第 10 题：

2.10、设计一个 Verilog 模块，描述如图 5 所示的电路原理图表示的电路。并设计一个测试平台，对电路进行仿真。**要求：使用单一模块设计。**

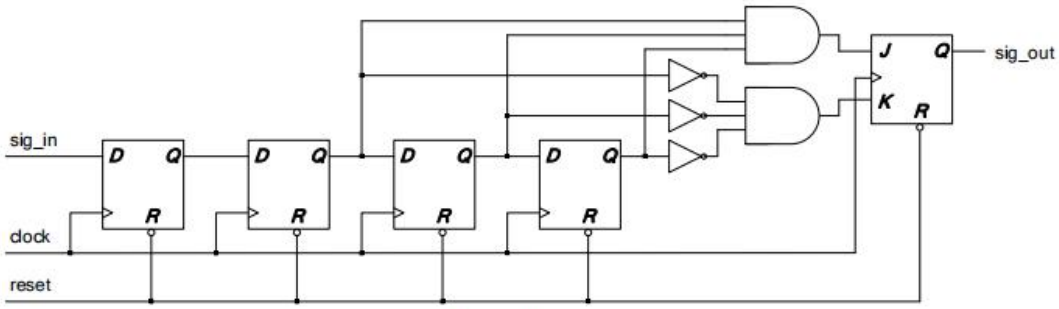


图 5、习题 10 电路原理图

要求：

（1）设计模块名为：

filter(sig_out, clock, reset, sig_in)

测试平台的模块名为：

tb_filter()

（2）并给出测试模块和测试分析结果。

（1）设计模块

```
// File: filter.v

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

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

    assign J = &Q[3:1],
           K = ~(|Q[3:1]);

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

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

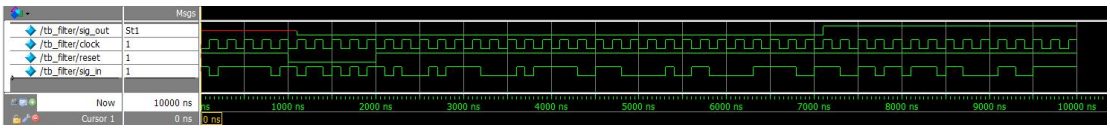
```
        else begin
            case({J, K})
                2'b01: sig_out <= 1'b0;
                2'b10: sig_out <= 1'b1;
                2'b11: sig_out <= ~sig_out;
                default: ;
            endcase
        end
    end

endmodule
```

（2）测试模块

```
// File: tb_filter.v

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

module tb_filter;

    wire sig_out;
    reg clock, reset, sig_in;

    filter a(sig_out, clock, reset, sig_in);

    initial begin : clk_loop
        clock = 1'b0;
        forever #10 clock = ~clock;
    end

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

    initial begin : sig_in_loop
        sig_in = 1'b1;
        forever #10 sig_in = $random % 2;
    end

    initial begin
        #1000 disable clk_loop;
        disable sig_in_loop;
```

```
    end

endmodule
```

（3）测试波形图



第 11 题：

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

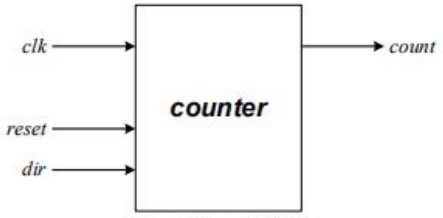


图 6、双向循环计数器

要求：

（1）采用异步复位，复位后从第一个有效时钟的上跳沿开始计数；如果此时 **dir=1** ，则递增计数，否则，递减计数。

（2）输出 count 为 8 位；

（3）对电路进行全面仿真。

（4）设计模块名为：

counter8b_updown(count, clk, reset, dir)

测试平台的模块名为：

tb_counter8b_updown()

（1）设计模块

```
// File: counter8b_updown.v

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

    always @(posedge clk, negedge reset) begin
        if(~reset)
            count <= 8'b0;
        else begin
            if(dir) begin
                if(count == 8'b1111_1111)
                    count <= 8'b0;
                else
                    count <= count + 1'b1;
```

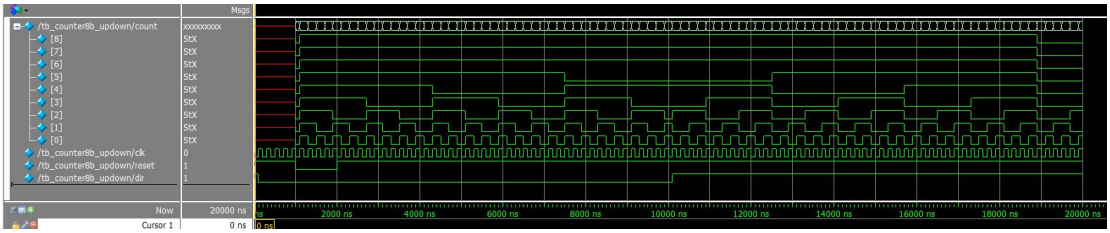
```
            end
            else begin
                if(count == 8'b0)
                    count <= 8'b1111_1111;
                else
                    count <= count - 1'b1;
            end
        end
    end

endmodule
```

（2）测试模块

```
// File: tb_counter8b_updown.v

`timescale 10 ns / 1 ns
`include "counter8b_updown.v"

module tb_counter8b_updown;

    wire [8:0] count;
    reg clk, reset, dir;

    counter8b_updown a(count, clk, reset, dir);

    initial begin : clk_loop
        clk = 1'b0;
        forever #10 clk = ~clk;
    end

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

    initial begin
        dir = 1'b1;
        #10 dir = 1'b0;
        #1000 dir = 1'b1;
    end

    initial begin
        #2000 disable clk_loop;
```

```
    end

endmodule
```

（3）测试波形图



第 12 题：

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

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

要求：

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

（2）设计模块名为：

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

测试平台的模块名为：

`tb_ALU()`

（1）设计模块

```
// File: ALU.v

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
```

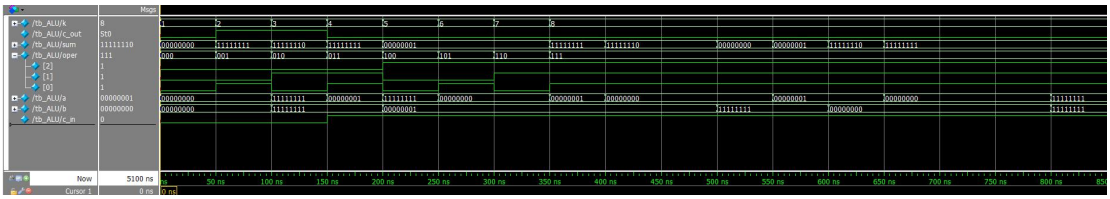
```
        );

    always @(*) begin
        case(oper)
            3'b000: {c_out, sum} = a + b + c_in; // and
            3'b001: {c_out, sum} = a + ~b + c_in; // subtract
            3'b010: {c_out, sum} = b + ~a + ~c_in; // subtract_a
            3'b011: {c_out, sum} = {1'b0, a | b}; // or_ab
            3'b100: {c_out, sum} = {1'b0, a & b}; // and_ab
            3'b101: {c_out, sum} = {1'b0, (~a) & b}; // not_ab
            3'b110: {c_out, sum} = {1'b0, a ^ b}; // exor
            3'b111: {c_out, sum} = {1'b0, a ~^ b}; // exnor
            default: {c_out, sum} = 9'bx;
        endcase
    end

endmodule
```

（2）测试模块

```
// File: tb_ALU.v

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

module tb_ALU;

    parameter STEP = 7;
    integer k;

    wire c_out;
    wire [7:0] sum;
    reg [2:0] oper;
    reg [7:0] a;
    reg [7:0] b;
    reg c_in;

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

    initial begin
        oper = 3'b0;
        for ( k=1; k<=STEP; k=k+1 )
            #5 oper = oper + 3'b1;
    end
```

```
    initial begin : a_loop
        a = 8'b0;
        forever #5 a = {a, $random % 2};
    end

    initial begin : b_loop
        b = 8'b0;
        forever #10 b = {b, $random % 2};
    end

    initial begin : c_in_loop
        c_in = 1'b0;
        forever #15 c_in = $random % 2;
    end

    initial begin
        $monitor("At time %4t, a=%b, b=%b, c_in=%b, oper=%b, sum=%b,
                  c_out=%b", $time, a, b, c_in, oper, sum, c_out);
    end

    initial begin
        #500 disable a_loop;
        disable b_loop;
        disable c_in_loop;
    end

endmodule
```

（3）测试波形图



（4）显示输出

```
# At time    0, a=00000000, b=00000000, c_in=0, oper=000, sum=00000000, c_out=0
# At time   50, a=00000000, b=00000000, c_in=0, oper=001, sum=11111111, c_out=1
# At time  100, a=11111111, b=11111111, c_in=0, oper=010, sum=11111110, c_out=1
# At time  150, a=00000001, b=11111111, c_in=1, oper=011, sum=11111111, c_out=0
# At time  200, a=11111111, b=00000001, c_in=1, oper=100, sum=00000001, c_out=0
# At time  250, a=00000000, b=00000001, c_in=1, oper=101, sum=00000001, c_out=0
# At time  300, a=00000000, b=00000001, c_in=1, oper=110, sum=00000001, c_out=0
# At time  350, a=00000001, b=00000001, c_in=1, oper=111, sum=11111111, c_out=0
# At time  400, a=00000000, b=00000001, c_in=1, oper=111, sum=11111110, c_out=0
# At time  500, a=00000000, b=11111111, c_in=1, oper=111, sum=00000000, c_out=0
# At time  550, a=00000001, b=11111111, c_in=1, oper=111, sum=00000001, c_out=0
# At time  600, a=00000001, b=00000000, c_in=1, oper=111, sum=11111110, c_out=0
# At time  650, a=00000000, b=00000000, c_in=1, oper=111, sum=11111111, c_out=0
# At time  800, a=11111111, b=11111111, c_in=1, oper=111, sum=11111111, c_out=0
# At time  900, a=00000000, b=00000000, c_in=1, oper=111, sum=11111111, c_out=0
# At time 1000, a=11111111, b=00000000, c_in=1, oper=111, sum=00000000, c_out=0
# At time 1050, a=11111111, b=00000000, c_in=0, oper=111, sum=00000000, c_out=0
# At time 1100, a=11111111, b=11111111, c_in=0, oper=111, sum=11111111, c_out=0
# At time 1150, a=00000001, b=11111111, c_in=0, oper=111, sum=00000001, c_out=0
# At time 1200, a=00000000, b=00000000, c_in=1, oper=111, sum=11111111, c_out=0
# At time 1250, a=00000001, b=00000000, c_in=1, oper=111, sum=11111110, c_out=0
# At time 1300, a=00000001, b=11111111, c_in=1, oper=111, sum=00000001, c_out=0
# At time 1350, a=00000000, b=11111111, c_in=0, oper=111, sum=00000000, c_out=0
# At time 1400, a=00000000, b=00000000, c_in=0, oper=111, sum=11111111, c_out=0
# At time 1500, a=00000000, b=00000000, c_in=1, oper=111, sum=11111111, c_out=0
# At time 1550, a=11111111, b=00000000, c_in=1, oper=111, sum=00000000, c_out=0
# At time 1600, a=00000000, b=00000001, c_in=1, oper=111, sum=11111110, c_out=0
# At time 1650, a=00000000, b=00000001, c_in=0, oper=111, sum=11111110, c_out=0
# At time 1700, a=00000000, b=00000000, c_in=0, oper=111, sum=11111111, c_out=0
# At time 1750, a=11111111, b=00000000, c_in=0, oper=111, sum=00000000, c_out=0
# At time 1800, a=00000001, b=00000001, c_in=1, oper=111, sum=11111111, c_out=0
# At time 1850, a=11111111, b=00000001, c_in=1, oper=111, sum=00000001, c_out=0
# At time 1900, a=00000000, b=00000000, c_in=1, oper=111, sum=11111111, c_out=0
# At time 1950, a=00000001, b=00000000, c_in=1, oper=111, sum=11111110, c_out=0
# At time 2000, a=00000000, b=00000001, c_in=1, oper=111, sum=11111110, c_out=0
# At time 2150, a=11111111, b=00000001, c_in=1, oper=111, sum=00000001, c_out=0
# At time 2200, a=00000000, b=11111111, c_in=1, oper=111, sum=00000000, c_out=0
# At time 2250, a=00000001, b=11111111, c_in=1, oper=111, sum=00000001, c_out=0
# At time 2300, a=00000000, b=11111111, c_in=1, oper=111, sum=00000000, c_out=0
# At time 2350, a=11111111, b=11111111, c_in=1, oper=111, sum=11111111, c_out=0
# At time 2400, a=00000000, b=00000000, c_in=1, oper=111, sum=11111111, c_out=0
# At time 2500, a=11111111, b=00000000, c_in=1, oper=111, sum=00000000, c_out=0
# At time 2550, a=00000000, b=00000000, c_in=1, oper=111, sum=11111111, c_out=0
# At time 2600, a=00000001, b=00000000, c_in=1, oper=111, sum=11111110, c_out=0
# At time 2700, a=00000001, b=11111111, c_in=1, oper=111, sum=00000001, c_out=0
# At time 2750, a=00000000, b=11111111, c_in=1, oper=111, sum=00000000, c_out=0
# At time 2800, a=00000001, b=00000001, c_in=1, oper=111, sum=11111111, c_out=0
# At time 2850, a=00000000, b=00000001, c_in=1, oper=111, sum=11111110, c_out=0
# At time 2900, a=00000000, b=00000000, c_in=1, oper=111, sum=11111111, c_out=0
# At time 3000, a=00000000, b=00000000, c_in=0, oper=111, sum=11111111, c_out=0
```

第 13 题：

2.13、设计一个具有图 7 所示功能的计数器模块。

要求：

（1）采用同步复位，复位后回到初始状态，然后从一个有效时钟的上跳沿开始计数；

（2）设计测试模块对其进行仿真验证；

（2）设计模块名为：

shift_counter(count, clk, reset)

测试平台的模块名为：

tb_shift_counter()

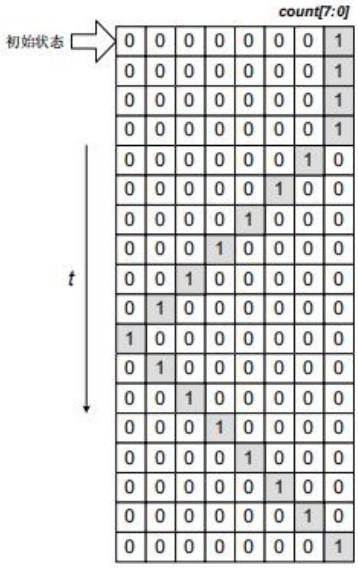


图 7、习题 13 计数器计数模式

第

（1）设计模块

```
// File: shift_counter.v

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

    reg [1:0] status;
    reg dir;

    always @(posedge clk) begin
        if(~reset) begin
            count <= 8'b1;
            status <= 2'b00;
            dir <= 1'b1;
        end
        else begin
            case(status)
                2'b00: status <= 2'b01;
                2'b01: status <= 2'b10;
                2'b10: status <= 2'b11;
                2'b11: begin
                    if(dir) begin
                        if(count == 8'b1000_0000) begin
                            count <= count >> 1;
                            dir <= 1'b0;
                        end
```

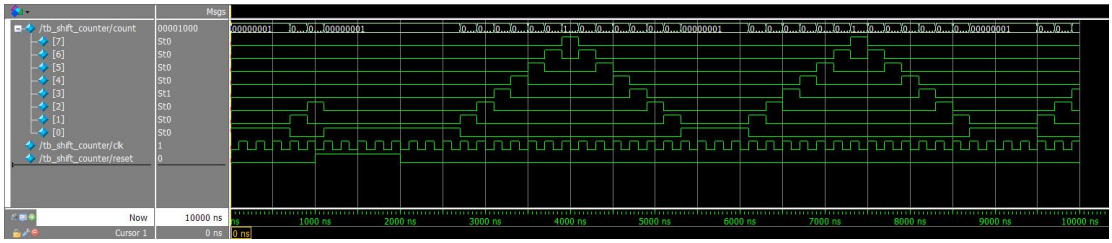
```
                    else
                        count <= count << 1;
                end
                else begin
                    if(count == 8'b1) begin
                        status <= 2'b00;
                        dir <= 1'b1;
                    end
                    else
                        count <= count >> 1;
                end
            end
            default: begin
                count <= 8'b1;
                status <= 2'b00;
                dir <= 1'b1;
            end
        endcase
    end
end

endmodule
```

（2）测试模块

```
// File: tb_shift_counter.v

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

module tb_shift_counter;

    wire [7:0] count;
    reg clk, reset;

    shift_counter a(count, clk, reset);

    initial begin : clk_loop
        clk = 1'b0;
        forever #10 clk = ~clk;
    end

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

```
        #100 reset = 1'b0;
    end

    initial begin
        $monitor("At time %4t, reset=%b, count=%b",
                 $time, reset, count);
    end

    initial begin
        #1000 disable clk_loop;
    end

endmodule
```

（3）测试波形图



（4）显示输出

```
# At time    0, reset=0, count=00000001
# At time  700, reset=0, count=00000010
# At time  900, reset=0, count=00000100
# At time 1000, reset=1, count=00000100
# At time 1100, reset=1, count=00000001
# At time 2000, reset=0, count=00000001
# At time 2700, reset=0, count=00000010
# At time 2900, reset=0, count=00000100
# At time 3100, reset=0, count=00001000
# At time 3300, reset=0, count=00010000
# At time 3500, reset=0, count=00100000
# At time 3700, reset=0, count=01000000
# At time 3900, reset=0, count=10000000
# At time 4100, reset=0, count=01000000
# At time 4300, reset=0, count=00100000
# At time 4500, reset=0, count=00010000
# At time 4700, reset=0, count=00001000
# At time 4900, reset=0, count=00000100
# At time 5100, reset=0, count=00000010
# At time 5300, reset=0, count=00000001
# At time 6100, reset=0, count=00000010
# At time 6300, reset=0, count=00000100
# At time 6500, reset=0, count=00001000
# At time 6700, reset=0, count=00010000
```

第 14 题：

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

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

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

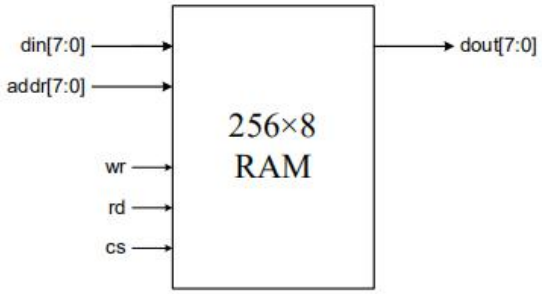


图 8、256×8bits 的 SRAM 框图

要求：

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

（2）设计测试模块对其进行仿真验证；

（3）设计模块名为：

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

测试平台的模块名为：

tb_sram()

（1）设计模块

```
// File: sram.v

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

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

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

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

    always @(*) begin
```

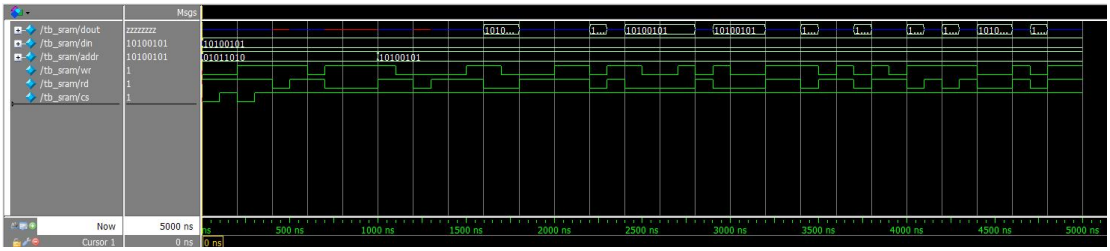
```
        if(cs && !rd)
            data <= memory[addr];
    end

endmodule
```

（2）测试模块

```
// File: tb_sram.v

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

module tb_sram;

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

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

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

    initial begin
        din = 8'b1010_0101;
        addr = 8'b0101_1010;
        #100 addr = 8'b1010_0101;
    end

    initial begin : loop
        wr = 1'b0;
        rd = 1'b1;
        forever begin
            #10;
            wr = $random % 2;
            rd = $random % 2;
        end
    end
```

```
    initial begin
        $monitor("At time %4t, din=%b, addr=%b, cs=%b, wr=%b, rd=%b,
                  dout=%b", $time, din, addr, cs, wr, rd, dout);
    end

    initial begin
        #500 disable loop;
    end

endmodule
```

（3）测试波形图



（4）显示输出

```
# At time    0, din=10100101, addr=01011010, cs=0, wr=0, rd=1, dout=zzzzzzzz
# At time  100, din=10100101, addr=01011010, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time  200, din=10100101, addr=01011010, cs=0, wr=1, rd=1, dout=zzzzzzzz
# At time  300, din=10100101, addr=01011010, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time  400, din=10100101, addr=01011010, cs=1, wr=1, rd=0, dout=xxxxxxxx
# At time  500, din=10100101, addr=01011010, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time  600, din=10100101, addr=01011010, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time  700, din=10100101, addr=01011010, cs=1, wr=1, rd=0, dout=xxxxxxxx
# At time 1000, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time 1100, din=10100101, addr=10100101, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time 1200, din=10100101, addr=10100101, cs=1, wr=0, rd=0, dout=xxxxxxxx
# At time 1300, din=10100101, addr=10100101, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time 1500, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time 1600, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time 1700, din=10100101, addr=10100101, cs=1, wr=0, rd=0, dout=10100101
# At time 1800, din=10100101, addr=10100101, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time 2000, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time 2200, din=10100101, addr=10100101, cs=1, wr=0, rd=0, dout=10100101
# At time 2300, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time 2400, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time 2500, din=10100101, addr=10100101, cs=1, wr=0, rd=0, dout=10100101
# At time 2700, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time 2800, din=10100101, addr=10100101, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time 2900, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time 3000, din=10100101, addr=10100101, cs=1, wr=0, rd=0, dout=10100101
# At time 3200, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time 3400, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time 3500, din=10100101, addr=10100101, cs=1, wr=0, rd=1, dout=zzzzzzzz
```

第 15 题：

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

要求：

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

（2）下面分别是设计模块和仿真测试模块。这里：

flag：输出端口；当检测到指定序列时，输出为 1，否则，输出为 0；

din：串行序列输入；

clk：时钟信号输入；在时钟的每个下降沿检查数据

rst_n：同步复位信号，低电平有效，复位时，flag 输出为 0。

设计模块为：

```
seq_detect( ouput reg flag, input din, clk, rst_n )
```

测试平台的模块名为：

```
tb_seq_detect()
```

（1）设计模块

```
// File: seq_detect.v

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

    reg [6:0] status;
    parameter IDLE = 7'b000_0001,
              A1 = 7'b000_0010,
              A2 = 7'b000_0100,
              A3 = 7'b000_1000,
              B1 = 7'b001_0000,
              B2 = 7'b010_0000,
              B3 = 7'b100_0000;

    always @(negedge clk) begin
        if(~rst_n) begin
            flag <= 1'b0;
            status <= IDLE;
        end
        else begin
            case(status)
                IDLE: if(din) begin flag <= 1'b0; status <= B1; end
                      else begin flag <= 1'b0; status <= A1; end
                A1: if(din) begin flag <= 1'b0; status <= A2; end
                    else begin flag <= 1'b0; status <= A1; end
                A2: if(din) begin flag <= 1'b0; status <= A3; end
                    else begin flag <= 1'b0; status <= A1; end
```

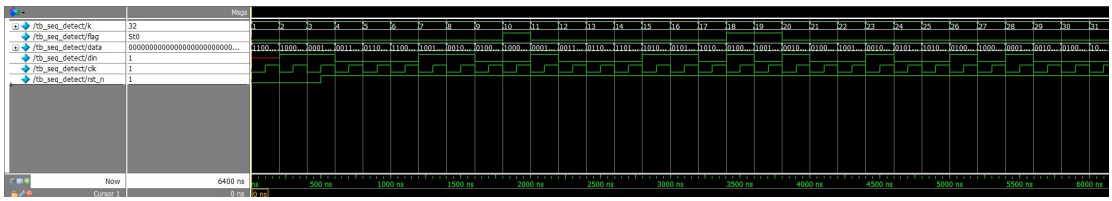
```
                A3: if(din) begin flag <= 1'b0; status <= B2; end
                    else begin flag <= 1'b1; status <= B3; end
                B1: if(din) begin flag <= 1'b0; status <= B2; end
                    else begin flag <= 1'b0; status <= A1; end
                B2: if(din) begin flag <= 1'b0; status <= B2; end
                    else begin flag <= 1'b0; status <= B3; end
                B3: if(din) begin flag <= 1'b1; status <= A2; end
                    else begin flag <= 1'b0; status <= A1; end
                default: begin flag <= 1'b0; status <= IDLE; end
            endcase
        end
    end

endmodule
```

（2）测试模块

```
// File: tb_seq_detect.v

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

module tb_seq_detect;

    parameter STEP = 31;
    integer k;

    wire flag;
    reg [31:0] data;
    reg din,clk, rst_n;

    seq_detect a(flag, din, clk, rst_n);

    initial begin : clk_loop
        clk = 1'b0;
        forever #10 clk = ~clk;
    end

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

    initial begin
        data = 32'b1100_0110_0100_0110_1010_0100_1010_0010;
```

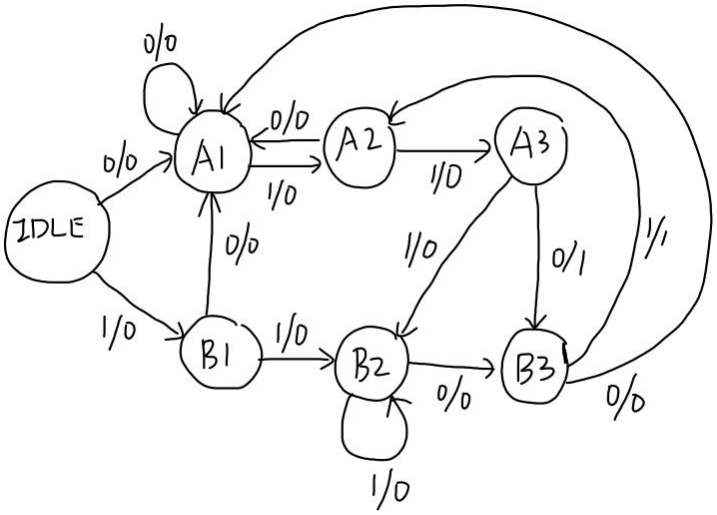
```
        for ( k=1; k<=STEP; k=k+1 ) begin
            #20;
            din = data[31];
            data = data << 1;
        end
        #20 disable clk_loop;
    end

endmodule
```

（3）测试波形图



（4）设计说明

序列检测器状态转换图：



第 16 题：

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

要求：

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

（2）下面分别是设计模块和仿真测试模块。这里：

Ready：输出端口；当检测到指定序列时，输出为 1，否则，输出为 0；

din：串行序列输入；

clk：时钟信号输入；在时钟的上升沿检查数据

rst：异步复位信号，高电平有效，复位时，Ready 输出为 0。

设计模块为：

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

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

测试平台的模块名为：

top()

（1）设计模块

```
// File: mealy.v

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

    reg [7:0] status;
    parameter IDLE = 8'b0000_0001,
                A = 8'b0000_0010,
                B = 8'b0000_0100,
                C = 8'b0000_1000,
                D = 8'b0001_0000,
                E = 8'b0010_0000,
                F = 8'b0100_0000,
                G = 8'b1000_0000;

    always @(posedge clk, posedge rst) begin
        if(rst) begin
            flag <= 1'b0;
            status <= IDLE;
        end
        else begin
            case(status)
                IDLE: if(din) begin flag <= 1'b0; status <= IDLE; end
                      else begin flag <= 1'b0; status <= A; end
```

```
                A: if(din) begin flag <= 1'b0; status <= B; end
                   else begin flag <= 1'b0; status <= A; end
                B: if(din) begin flag <= 1'b0; status <= IDLE; end
                   else begin flag <= 1'b0; status <= C; end
                C: if(din) begin flag <= 1'b0; status <= D; end
                   else begin flag <= 1'b0; status <= A; end
                D: if(din) begin flag <= 1'b0; status <= IDLE; end
                   else begin flag <= 1'b0; status <= E; end
                E: if(din) begin flag <= 1'b0; status <= F; end
                   else begin flag <= 1'b0; status <= A; end
                F: if(din) begin flag <= 1'b0; status <= IDLE; end
                   else begin flag <= 1'b0; status <= G; end
                G: if(din) begin flag <= 1'b1; status <= F; end
                   else begin flag <= 1'b0; status <= A; end
                default: begin flag <= 1'b0; status <= IDLE; end
            endcase
        end
    end

endmodule
```

```
// File: moore.v

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

    reg [8:0] status;
    parameter IDLE = 9'b0_0000_0001,
                A = 9'b0_0000_0010,
                B = 9'b0_0000_0100,
                C = 9'b0_0000_1000,
                D = 9'b0_0001_0000,
                E = 9'b0_0010_0000,
                F = 9'b0_0100_0000,
                G = 9'b0_1000_0000,
                H = 9'b1_0000_0000;

    always @(posedge clk, posedge rst) begin
        if(rst) begin
            flag <= 1'b0;
            status <= IDLE;
        end
        else begin
            flag <= (status == H) ? 1'b1 : 1'b0;
```

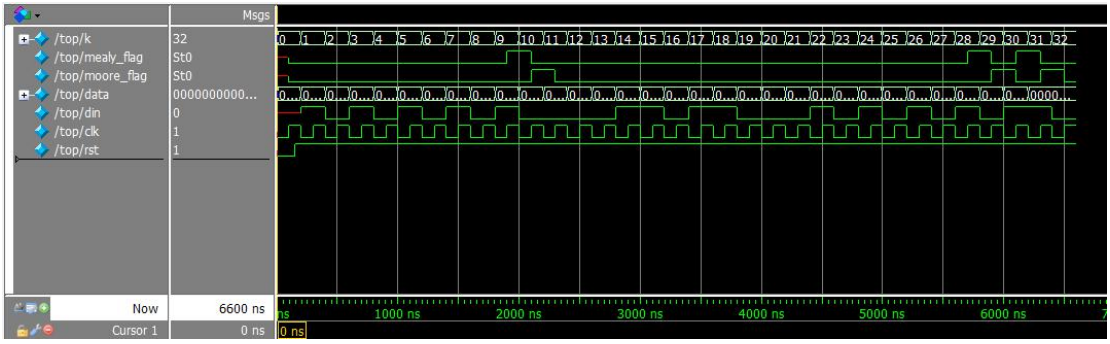
```
            case(status)
                IDLE: status <= (din) ? IDLE : A;
                A: status <= (din) ? B : A;
                B: status <= (din) ? IDLE : C;
                C: status <= (din) ? D : A;
                D: status <= (din) ? IDLE : E;
                E: status <= (din) ? F : A;
                F: status <= (din) ? IDLE : G;
                G: status <= (din) ? H : A;
                H: status <= (din) ? IDLE : G;
                default: ;
            endcase
        end
    end

endmodule
```

（2）测试模块

```
// File: top.v

`timescale 10 ns / 1 ns
`include "mealy.v"
`include "moore.v"

module top;

    parameter STEP = 31;
    integer k;

    wire mealy_flag, moore_flag;
    reg [31:0] data;
    reg din, clk, rst;

    mealy a(mealy_flag, din, clk, rst);
    moore b(moore_flag, din, clk, rst);

    initial begin : clk_loop
        clk = 1'b0;
        forever #10 clk = ~clk;
    end

    initial begin
        rst = 1'b0;
        #15 rst = 1'b1;
```

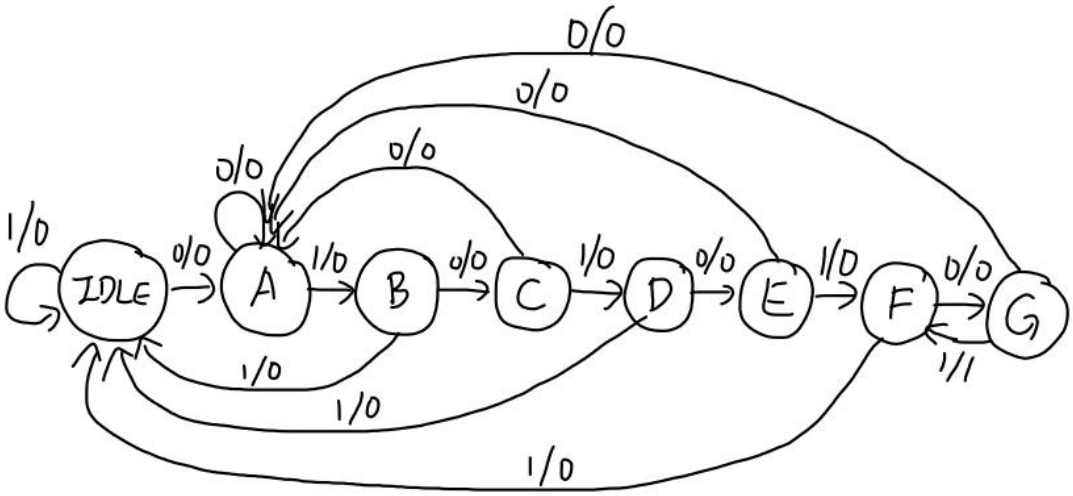
```
    end

    initial begin
        data = 32'b0110_1010_1010_0011_0110_0001_0101_0101;
        for ( k=0; k<=STEP; k=k+1) begin
            #20;
            din = data[0];
            data = data >> 1;
        end
        #20 disable clk_loop;
    end

endmodule
```

（3）测试波形图



（4）设计说明

① **Mealy 型状态机状态转换图**



② **Moore 型状态机状态转换图**