数字系统设计作业报告

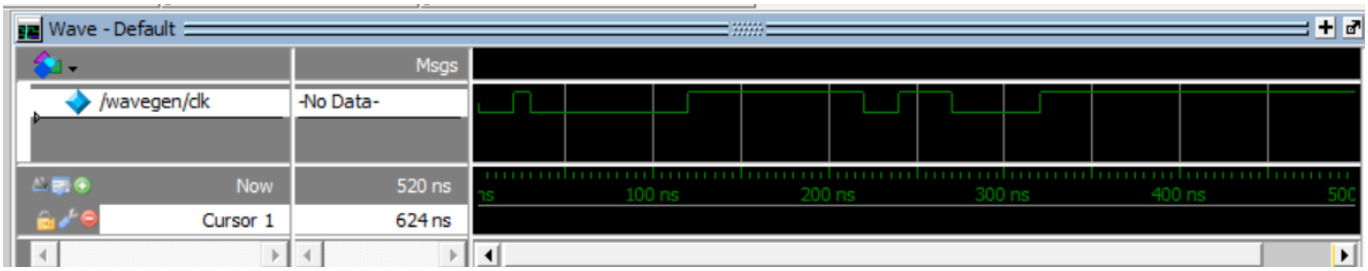
- 学号：518021910628
- 姓名：吴小茜
- 日期：2020.12.23

EXP2-1

设计/测试模块

```
// File: wavegen.v

`timescale 10 ns / 1 ns

module wavegen;

    reg clk;

    initial begin
        clk = 1'b0;
        #2 clk = 1'b1;
        #1 clk = 1'b0;
        #9 clk = 1'b1;
        #10 clk = 1'b0;
        #2 clk = 1'b1;
        #3 clk = 1'b0;
        #5 clk = 1'b1;
    #20 $stop;
    end

endmodule
```

仿真结果



EXP2-2

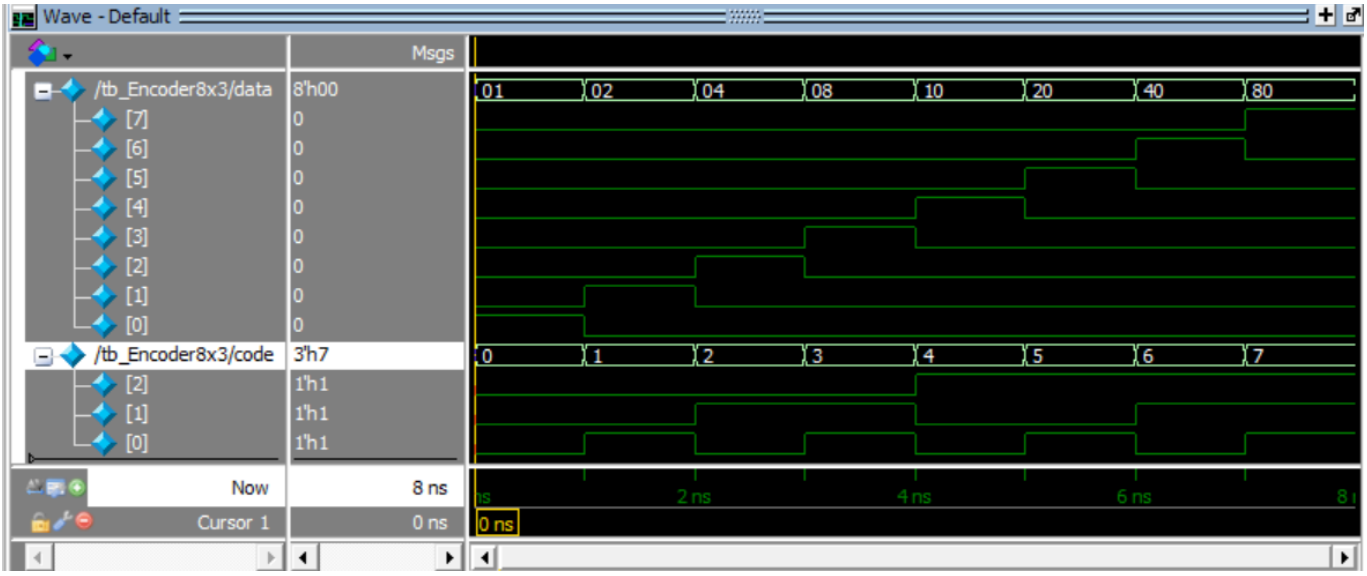
设计模块

```
// 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
```

测试模块

```
// File: tb_encoder8x3.v

`include "encoder8x3.v"

module tb_Encoder8x3;

    parameter STEP = 9;

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

    Encoder8x3 a(code, 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
```

仿真结果




```
# At time     0, data=00000001, code=0
# At time     1, data=00000010, code=1
# At time     2, data=00000100, code=2
# At time     3, data=00001000, code=3
# At time     4, data=00010000, code=4
# At time     5, data=00100000, code=5
# At time     6, data=01000000, code=6
# At time     7, data=10000000, code=7
# At time     8, data=00000000, code=x
```

EXP2-3

a) 2选1多路选择器

设计模块

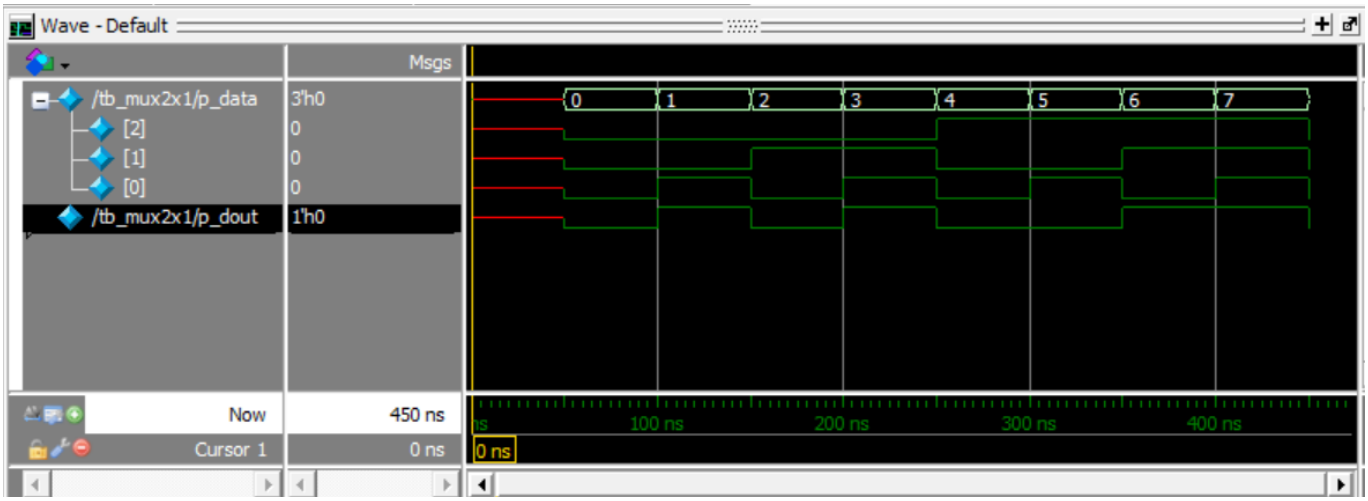
```
// File: mux2x1.v
module mux2x1( dout, sel, din );
 output dout;
 input sel;
 input [1:0] din;
 bufif1 b2( dout, din[1], sel );
 bufif0 b1( dout, din[0], sel );
endmodule
```

仿真模块

```
// File: tb_mux2x1.v
`timescale 10ns / 1ns
```

```
`include "mux2x1.v"

module tb_mux2x1();
 reg [2:0] p_data;
 wire p_dout;
 integer i;
 initial begin
 p_data = 3'bx;
 #5 p_data = 3'b0;
 for ( i = 1; i < 9; i = i + 1 )
  #5 p_data = p_data + 3'b1;
 end

 mux2x1 m0( .dout( p_dout ), .sel(p_data[2]), .din(p_data[1:0]) );
 initial
 $monitor( "At time %t, dout=%1b, sel=%1b, din=%2b", $time, p_dout, p_data[2],
p_data[1:0]);

endmodule
```

仿真结果



显示输出

```
VSIM 58> run -all
# At time                    0, dout=x, sel=x, din=xx
# At time                   50, dout=0, sel=0, din=00
# At time                  100, dout=1, sel=0, din=01
# At time                  150, dout=0, sel=0, din=10
# At time                  200, dout=1, sel=0, din=11
# At time                  250, dout=0, sel=1, din=00
# At time                  300, dout=0, sel=1, din=01
# At time                  350, dout=1, sel=1, din=10
# At time                  400, dout=1, sel=1, din=11
# At time                  450, dout=0, sel=0, din=00
```

b) 4选1多路选择器

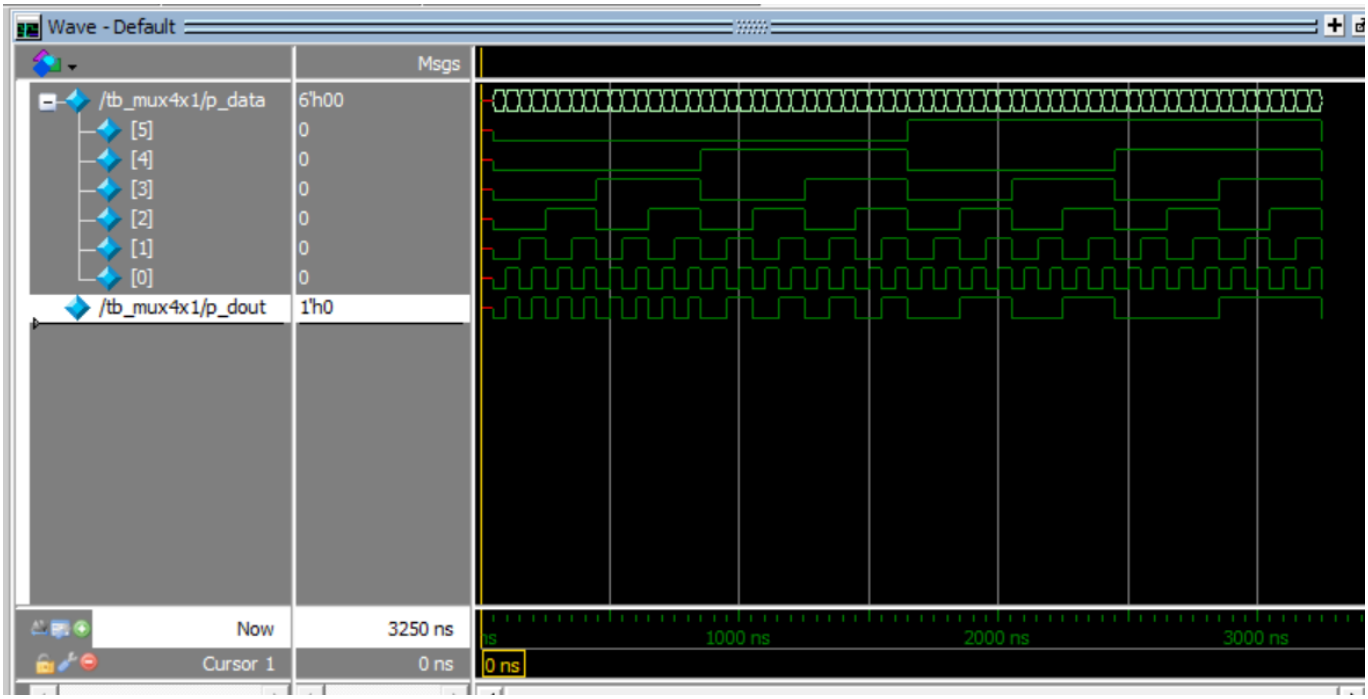
设计模块

```
// File: mux4x1.v

`include "mux2x1.v"

module mux4x1(dout, sel, din);

    output dout;
    input [1:0] sel;
    input [3:0] din;

    wire ta, tb;

    mux2x1 a(ta, sel[0], din[1:0]),
           b(tb, sel[0], din[3:2]),
           c(dout, sel[1], {tb, ta});

endmodule
```

仿真模块

```
// File: tb_mux4x1.v
`timescale 10ns / 1ns
`include "mux4x1.v"

module tb_mux4x1();
 reg [5:0] p_data;
 wire p_dout;
 integer i;
 initial begin
 p_data = 6'bx;
 #5 p_data = 6'b0;
 for ( i = 1; i < 65; i = i + 1 )
  #5 p_data = p_data + 6'b1;
 end

 mux4x1 m0( .dout( p_dout ), .sel(p_data[5:4]), .din(p_data[3:0]) );
 initial
 $monitor( "At time %t, dout=%1b, sel=%2b, din=%4b", $time, p_dout, p_data[5:4],
p_data[3:0]);

endmodule
```

仿真结果


Wave - Default
Msgs
/tb_mux4x1/p_data
6'h00
[5]
0
[4]
0
[3]
0
[2]
0
[1]
0
[0]
0
/tb_mux4x1/p_dout
1'h0
Now
3250 ns
Cursor 1
0 ns
1000 ns
2000 ns
3000 ns
0 ns

显示输出

```
VSIM 90> run -all
# At time                    0, dout=x, sel=xx, din=xxxx
# At time                   50, dout=0, sel=00, din=0000
# At time                  100, dout=1, sel=00, din=0001
# At time                  150, dout=0, sel=00, din=0010
# At time                  200, dout=1, sel=00, din=0011
# At time                  250, dout=0, sel=00, din=0100
# At time                  300, dout=1, sel=00, din=0101
# At time                  350, dout=0, sel=00, din=0110
# At time                  400, dout=1, sel=00, din=0111
# At time                  450, dout=0, sel=00, din=1000
# At time                  500, dout=1, sel=00, din=1001
# At time                  550, dout=0, sel=00, din=1010
# At time                  600, dout=1, sel=00, din=1011
# At time                  650, dout=0, sel=00, din=1100
# At time                  700, dout=1, sel=00, din=1101
# At time                  750, dout=0, sel=00, din=1110
# At time                  800, dout=1, sel=00, din=1111
# At time                  850, dout=0, sel=01, din=0000
# At time                  900, dout=0, sel=01, din=0001
# At time                  950, dout=1, sel=01, din=0010
# At time                 1000, dout=1, sel=01, din=0011
# At time                 1050, dout=0, sel=01, din=0100
# At time                 1100, dout=0, sel=01, din=0101
# At time                 1150, dout=1, sel=01, din=0110
# At time                 1200, dout=1, sel=01, din=0111
# At time                 1250, dout=0, sel=01, din=1000
# At time                 1300, dout=0, sel=01, din=1001
# At time                 1350, dout=1, sel=01, din=1010
# At time                 1400, dout=1, sel=01, din=1011
# At time                 1450, dout=0, sel=01, din=1100
# At time                 1500, dout=0, sel=01, din=1101
# At time                 1550, dout=1, sel=01, din=1110
# At time                 1600, dout=1, sel=01, din=1111
# At time                 1650, dout=0, sel=10, din=0000
# At time                 1700, dout=0, sel=10, din=0001
# At time                 1750, dout=0, sel=10, din=0010
# At time                 1800, dout=0, sel=10, din=0011
# At time                 1850, dout=1, sel=10, din=0100
# At time                 1900, dout=1, sel=10, din=0101
# At time                 1950, dout=1, sel=10, din=0110
# At time                 2000, dout=1, sel=10, din=0111
# At time                 2050, dout=0, sel=10, din=1000
# At time                 2100, dout=0, sel=10, din=1001
# At time                 2150, dout=0, sel=10, din=1010
```

```
# At time                    2200, dout=0, sel=10, din=1011
# At time                    2250, dout=1, sel=10, din=1100
# At time                    2300, dout=1, sel=10, din=1101
# At time                    2350, dout=1, sel=10, din=1110
# At time                    2400, dout=1, sel=10, din=1111
# At time                    2450, dout=0, sel=11, din=0000
# At time                    2500, dout=0, sel=11, din=0001
# At time                    2550, dout=0, sel=11, din=0010
# At time                    2600, dout=0, sel=11, din=0011
# At time                    2650, dout=0, sel=11, din=0100
# At time                    2700, dout=0, sel=11, din=0101
# At time                    2750, dout=0, sel=11, din=0110
# At time                    2800, dout=0, sel=11, din=0111
# At time                    2850, dout=1, sel=11, din=1000
# At time                    2900, dout=1, sel=11, din=1001
# At time                    2950, dout=1, sel=11, din=1010
# At time                    3000, dout=1, sel=11, din=1011
# At time                    3050, dout=1, sel=11, din=1100
# At time                    3100, dout=1, sel=11, din=1101
# At time                    3150, dout=1, sel=11, din=1110
# At time                    3200, dout=1, sel=11, din=1111
# At time                    3250, dout=0, sel=00, din=0000
```

EXP2-4

设计模块——利用门原语设计

```
// File: comb_str.v
module comb_str(output Y, input A,B,C,D);
 wire M1,M2,M3,M4;
 not u1(M1,D);
 and u4(M4,B,C,M1);
 or u3(M3,A,D);
 not u2(M2,M3);
 and u5(Y,M2,M4);
endmodule
```

设计模块——利用连续赋值语句设计

```
//File comb_dataflow.v
module comb_dataflow(output Y, input A,B,C,D);
 assign Y=(~D&B&C)&~(A|D);
endmodule
```

设计模块——利用过程语句设计

```
//File: comb_behavior.v
module comb_behavior(output reg Y, input A,B,C,D);
 always@(*) begin
  Y = ~(A | D) & (B & C & ~D);
 end
endmodule;
```

设计模块——利用UDP设计

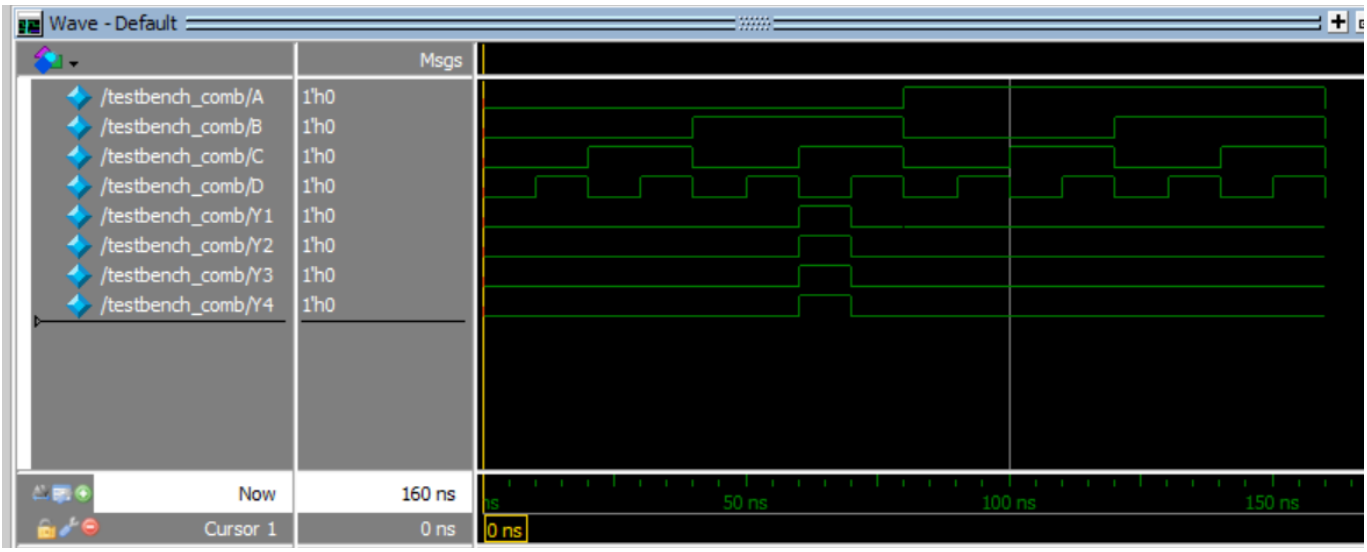
```
// File: comb_prim.v
primitive comb_prim(output Y, input A,B,C,D);
 table
 //a b c d : y
   0 0 ? ? : 0;
   0 1 0 ? : 0;
   0 1 1 0 : 1;
   0 1 1 1 : 0;
   1 ? ? ? : 0;
 endtable
endprimitive
```

测试模块

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

module testbench_comb();
 wire Y1,Y2,Y3,Y4;
 reg A,B,C,D;
 integer k;

 comb_str a(Y1, A, B, C, D);
 comb_dataflow b(Y2, A, B, C, D);
 comb_behavior 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<16;k=k+1)
```

```
    #1 {A, B, C, D} = {A, B, C, D} + 1'b1;
   #1 {A,B,C,D}=4'b0;
  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
```

仿真结果



显示输出

```
VSIM 144> run -all
# 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
# At time  160, A=0, B=0, C=0, D=0, Y1=0, Y2=0, Y3=0, Y4=0
```

EXP2-5

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

    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
```

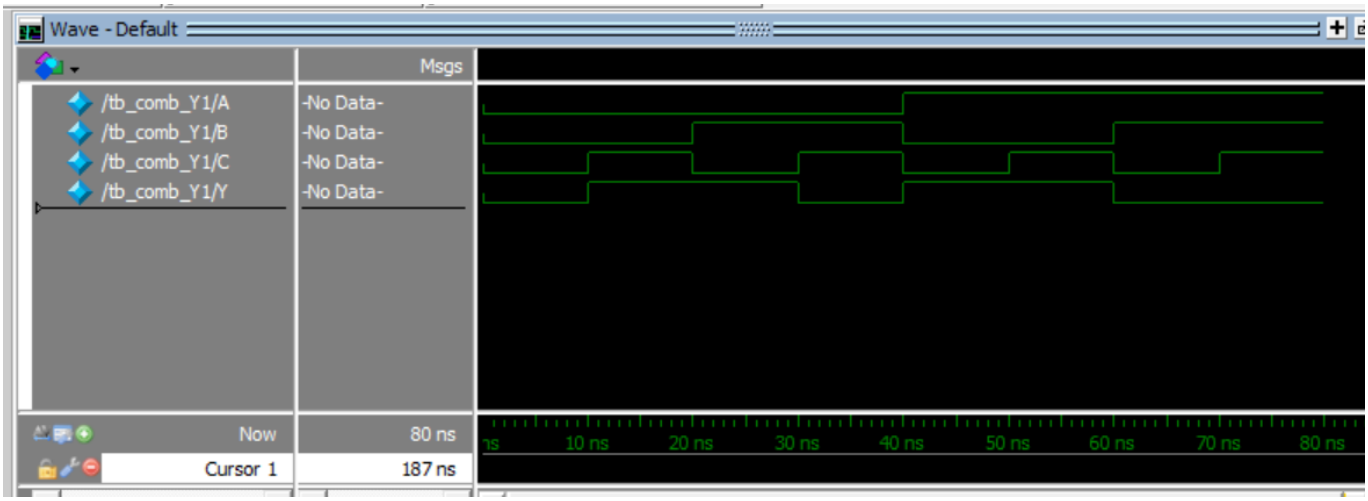
测试模块

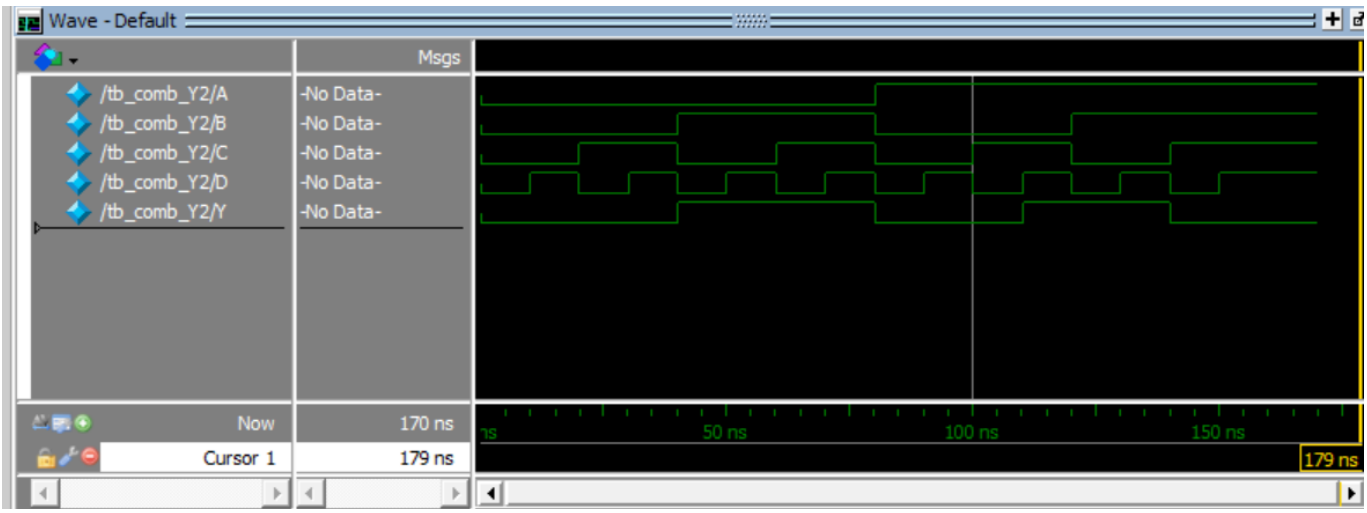
```
// File: tb_comb_Y1.v
`timescale 10ns / 1ns
`include "comb_Y1.v"

module tb_comb_Y1;

    parameter STEP = 8;
    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 10ns / 1ns
`include "comb_Y2.v"

module tb_comb_Y2;

    parameter STEP = 16;
    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
```

仿真结果


Wave - Default
Msgs
/tb_comb_Y1/A
/tb_comb_Y1/B
/tb_comb_Y1/C
/tb_comb_Y1/Y
-No Data-
-No Data-
-No Data-
-No Data-
Now
80 ns
Cursor 1
187 ns
10 ns
20 ns
30 ns
40 ns
50 ns
60 ns
70 ns
80 ns


Wave - Default
Msgs
/tb_comb_Y2/A
/tb_comb_Y2/B
/tb_comb_Y2/C
/tb_comb_Y2/D
/tb_comb_Y2/Y
-No Data-
-No Data-
-No Data-
-No Data-
-No Data-
Now
170 ns
Cursor 1
179 ns
50 ns
100 ns
150 ns
179 ns

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
```

```
# 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
```

EXP2-6

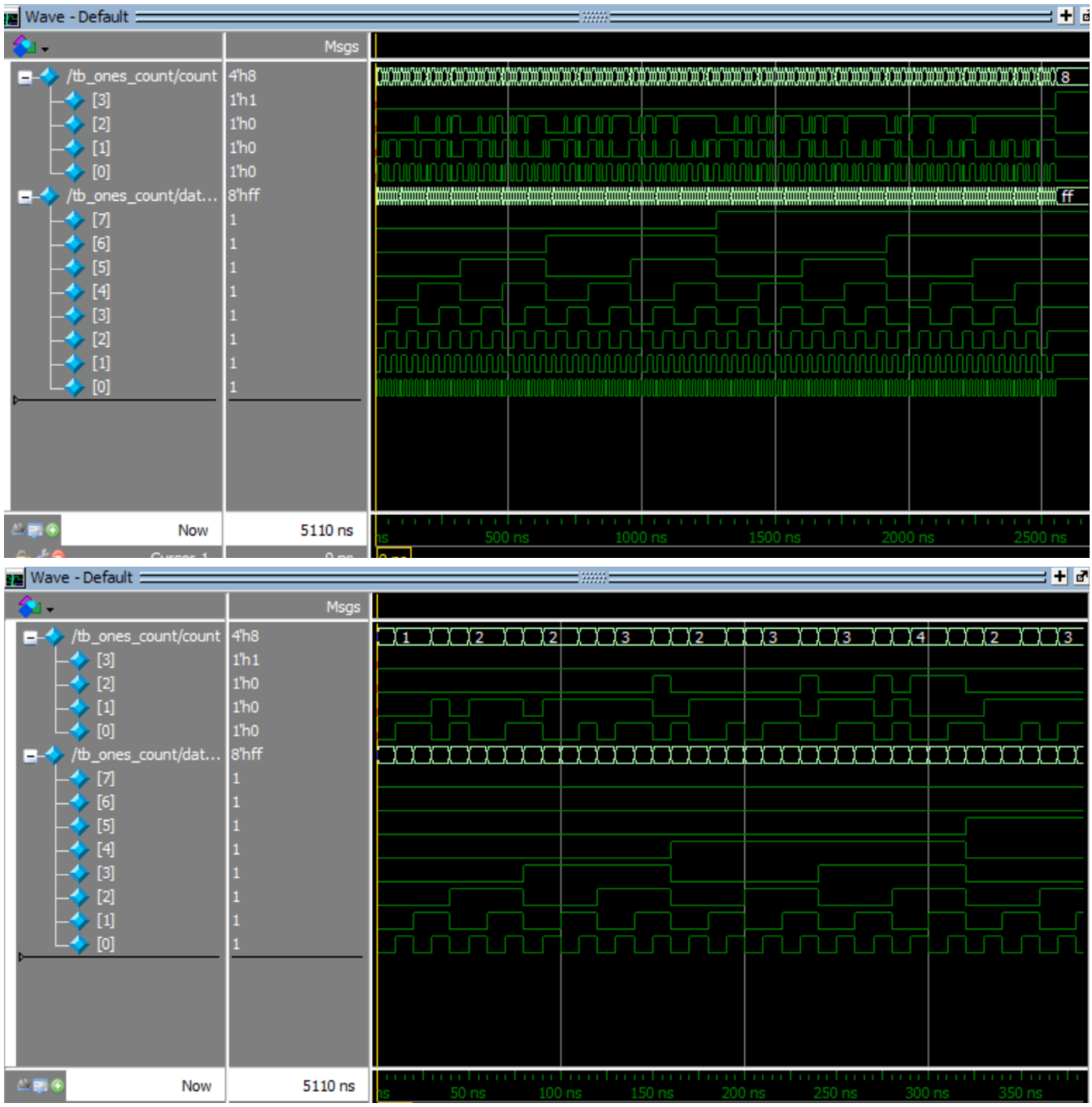
设计模块

```
//File: ones_count.v
module ones_count(output reg [3:0] count, input [7:0] dat_in);
 integer k;
 always@(dat_in)begin
  count=4'b0;
  for (k=0;k<8;k=k+1)
   count = count+dat_in[k];
 end
endmodule
```

测试模块

```
// File: tb_ones_count.v
`timescale 10ns / 1ns
`include "ones_count.v"

module tb_ones_count;

    parameter STEP = 256;
    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
```

仿真结果


Wave - Default
Msgs
/tb_ones_count/count
4'h8
[3]
1'h1
[2]
1'h0
[1]
1'h0
[0]
1'h0
/tb_ones_count/dat...
8'hff
[7]
1
[6]
1
[5]
1
[4]
1
[3]
1
[2]
1
[1]
1
[0]
1
Now
5110 ns
500 ns
1000 ns
1500 ns
2000 ns
2500 ns
8
ff
Wave - Default
Msgs
/tb_ones_count/count
4'h8
[3]
1'h1
[2]
1'h0
[1]
1'h0
[0]
1'h0
/tb_ones_count/dat...
8'hff
[7]
1
[6]
1
[5]
1
[4]
1
[3]
1
[2]
1
[1]
1
[0]
1
Now
5110 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350 ns
1
2
2
3
2
3
3
4
2
3

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
# At time  320, dat_in=00100000, count=0001
# At time  330, dat_in=00100001, count=0010
# At time  340, dat_in=00100010, count=0010
# At time  350, dat_in=00100011, count=0011
# At time  360, dat_in=00100100, count=0010
```

```
# At time  370, dat_in=00100101, count=0011
# At time  380, dat_in=00100110, count=0011
# At time  390, dat_in=00100111, count=0100
# At time  400, dat_in=00101000, count=0010
# At time  410, dat_in=00101001, count=0011
# At time  420, dat_in=00101010, count=0011
# At time  430, dat_in=00101011, count=0100
# At time  440, dat_in=00101100, count=0011
```

```
# At time  450, dat_in=00101101, count=0100
# At time  460, dat_in=00101110, count=0100
# At time  470, dat_in=00101111, count=0101
# At time  480, dat_in=00110000, count=0010
# At time  490, dat_in=00110001, count=0011
# At time  500, dat_in=00110010, count=0011
# At time  510, dat_in=00110011, count=0100
# At time  520, dat_in=00110100, count=0011
# At time  530, dat_in=00110101, count=0100
# At time  540, dat_in=00110110, count=0100
# At time  550, dat_in=00110111, count=0101
# At time  560, dat_in=00111000, count=0011
# At time  570, dat_in=00111001, count=0100
# At time  580, dat_in=00111010, count=0100
# At time  590, dat_in=00111011, count=0101
# At time  600, dat_in=00111100, count=0100
# At time  610, dat_in=00111101, count=0101
# At time  620, dat_in=00111110, count=0101
# At time  630, dat_in=00111111, count=0110
# At time  640, dat_in=01000000, count=0001
# At time  650, dat_in=01000001, count=0010
# At time  660, dat_in=01000010, count=0010
# At time  670, dat_in=01000011, count=0011
# At time  680, dat_in=01000100, count=0010
# At time  690, dat_in=01000101, count=0011
# At time  700, dat_in=01000110, count=0011
# At time  710, dat_in=01000111, count=0100
# At time  720, dat_in=01001000, count=0010
# At time  730, dat_in=01001001, count=0011
```

```
# At time  740, dat_in=01001010, count=0011
# At time  750, dat_in=01001011, count=0100
# At time  760, dat_in=01001100, count=0011
# At time  770, dat_in=01001101, count=0100
# At time  780, dat_in=01001110, count=0100
# At time  790, dat_in=01001111, count=0101
# At time  800, dat_in=01010000, count=0010
# At time  810, dat_in=01010001, count=0011
# At time  820, dat_in=01010010, count=0011
# At time  830, dat_in=01010011, count=0100
# At time  840, dat_in=01010100, count=0011
# At time  850, dat_in=01010101, count=0100
# At time  860, dat_in=01010110, count=0100
# At time  870, dat_in=01010111, count=0101
# At time  880, dat_in=01011000, count=0011
# At time  890, dat_in=01011001, count=0100
# At time  900, dat_in=01011010, count=0100
# At time  910, dat_in=01011011, count=0101
# At time  920, dat_in=01011100, count=0100
# At time  930, dat_in=01011101, count=0101
# At time  940, dat_in=01011110, count=0101
# At time  950, dat_in=01011111, count=0110
# At time  960, dat_in=01100000, count=0010
# At time  970, dat_in=01100001, count=0011
# At time  980, dat_in=01100010, count=0011
# At time  990, dat_in=01100011, count=0100
# At time 1000, dat_in=01100100, count=0011
# At time 1010, dat_in=01100101, count=0100
# At time 1020, dat_in=01100110, count=0100
# At time 1030, dat_in=01100111, count=0101
# At time 1040, dat_in=01101000, count=0011
# At time 1050, dat_in=01101001, count=0100
# At time 1060, dat_in=01101010, count=0100
# At time 1070, dat_in=01101011, count=0101
# At time 1080, dat_in=01101100, count=0100
# At time 1090, dat_in=01101101, count=0101
# At time 1100, dat_in=01101110, count=0101
```

```
# At time 1110, dat_in=01101111, count=0110
# At time 1120, dat_in=01110000, count=0011
# At time 1130, dat_in=01110001, count=0100
# At time 1140, dat_in=01110010, count=0100
# At time 1150, dat_in=01110011, count=0101
# At time 1160, dat_in=01110100, count=0100
# At time 1170, dat_in=01110101, count=0101
# At time 1180, dat_in=01110110, count=0101
# At time 1190, dat_in=01110111, count=0110
# At time 1200, dat_in=01111000, count=0100
# At time 1210, dat_in=01111001, count=0101
# At time 1220, dat_in=01111010, count=0101
# At time 1230, dat_in=01111011, count=0110
# At time 1240, dat_in=01111100, count=0101
# At time 1250, dat_in=01111101, count=0110
# At time 1260, dat_in=01111110, count=0110
# At time 1270, dat_in=01111111, count=0111
# At time 1280, dat_in=10000000, count=0001
# At time 1290, dat_in=10000001, count=0010
# At time 1300, dat_in=10000010, count=0010
# At time 1310, dat_in=10000011, count=0011
# At time 1320, dat_in=10000100, count=0010
# At time 1330, dat_in=10000101, count=0011
# At time 1340, dat_in=10000110, count=0011
# At time 1350, dat_in=10000111, count=0100
# At time 1360, dat_in=10001000, count=0010
# At time 1370, dat_in=10001001, count=0011
# At time 1380, dat_in=10001010, count=0011
# At time 1390, dat_in=10001011, count=0100
# At time 1400, dat_in=10001100, count=0011
# At time 1410, dat_in=10001101, count=0100
# At time 1420, dat_in=10001110, count=0100
# At time 1430, dat_in=10001111, count=0101
# At time 1440, dat_in=10010000, count=0010
# At time 1450, dat_in=10010001, count=0011
# At time 1460, dat_in=10010010, count=0011
# At time 1470, dat_in=10010011, count=0100
```

```
# At time 1480, dat_in=10010100, count=0011
# At time 1490, dat_in=10010101, count=0100
# At time 1500, dat_in=10010110, count=0100
# At time 1510, dat_in=10010111, count=0101
# At time 1520, dat_in=10011000, count=0011
# At time 1530, dat_in=10011001, count=0100
# At time 1540, dat_in=10011010, count=0100
# At time 1550, dat_in=10011011, count=0101
# At time 1560, dat_in=10011100, count=0100
# At time 1570, dat_in=10011101, count=0101
# At time 1580, dat_in=10011110, count=0101
# At time 1590, dat_in=10011111, count=0110
# At time 1600, dat_in=10100000, count=0010
# At time 1610, dat_in=10100001, count=0011
# At time 1620, dat_in=10100010, count=0011
# At time 1630, dat_in=10100011, count=0100
# At time 1640, dat_in=10100100, count=0011
# At time 1650, dat_in=10100101, count=0100
# At time 1660, dat_in=10100110, count=0100
# At time 1670, dat_in=10100111, count=0101
# At time 1680, dat_in=10101000, count=0011
# At time 1690, dat_in=10101001, count=0100
# At time 1700, dat_in=10101010, count=0100
# At time 1710, dat_in=10101011, count=0101
# At time 1720, dat_in=10101100, count=0100
# At time 1730, dat_in=10101101, count=0101
# At time 1740, dat_in=10101110, count=0101
# At time 1750, dat_in=10101111, count=0110
# At time 1760, dat_in=10110000, count=0011
# At time 1770, dat_in=10110001, count=0100
# At time 1780, dat_in=10110010, count=0100
# At time 1790, dat_in=10110011, count=0101
# At time 1800, dat_in=10110100, count=0100
# At time 1810, dat_in=10110101, count=0101
# At time 1820, dat_in=10110110, count=0101
# At time 1830, dat_in=10110111, count=0110
# At time 1840, dat_in=10111000, count=0100
```

```
# At time 1850, dat_in=10111001, count=0101
# At time 1860, dat_in=10111010, count=0101
# At time 1870, dat_in=10111011, count=0110
# At time 1880, dat_in=10111100, count=0101
# At time 1890, dat_in=10111101, count=0110
# At time 1900, dat_in=10111110, count=0110
# At time 1910, dat_in=10111111, count=0111
# At time 1920, dat_in=11000000, count=0010
# At time 1930, dat_in=11000001, count=0011
# At time 1940, dat_in=11000010, count=0011
# At time 1950, dat_in=11000011, count=0100
# At time 1960, dat_in=11000100, count=0011
# At time 1970, dat_in=11000101, count=0100
# At time 1980, dat_in=11000110, count=0100
# At time 1990, dat_in=11000111, count=0101
# At time 2000, dat_in=11001000, count=0011
# At time 2010, dat_in=11001001, count=0100
# At time 2020, dat_in=11001010, count=0100
# At time 2030, dat_in=11001011, count=0101
# At time 2040, dat_in=11001100, count=0100
# At time 2050, dat_in=11001101, count=0101
# At time 2060, dat_in=11001110, count=0101
# At time 2070, dat_in=11001111, count=0110
# At time 2080, dat_in=11010000, count=0011
# At time 2090, dat_in=11010001, count=0100
# At time 2100, dat_in=11010010, count=0100
# At time 2110, dat_in=11010011, count=0101
# At time 2120, dat_in=11010100, count=0100
# At time 2130, dat_in=11010101, count=0101
# At time 2140, dat_in=11010110, count=0101
# At time 2150, dat_in=11010111, count=0110
# At time 2160, dat_in=11011000, count=0100
# At time 2170, dat_in=11011001, count=0101
# At time 2180, dat_in=11011010, count=0101
# At time 2190, dat_in=11011011, count=0110
# At time 2200, dat_in=11011100, count=0101
# At time 2210, dat_in=11011101, count=0110
```

```
# At time 2220, dat_in=11011110, count=0110
# At time 2230, dat_in=11011111, count=0111
# At time 2240, dat_in=11100000, count=0011
# At time 2250, dat_in=11100001, count=0100
# At time 2260, dat_in=11100010, count=0100
# At time 2270, dat_in=11100011, count=0101
# At time 2280, dat_in=11100100, count=0100
# At time 2290, dat_in=11100101, count=0101
# At time 2300, dat_in=11100110, count=0101
# At time 2310, dat_in=11100111, count=0110
# At time 2320, dat_in=11101000, count=0100
# At time 2330, dat_in=11101001, count=0101
# At time 2340, dat_in=11101010, count=0101
# At time 2350, dat_in=11101011, count=0110
# At time 2360, dat_in=11101100, count=0101
# At time 2370, dat_in=11101101, count=0110
# At time 2380, dat_in=11101110, count=0110
# At time 2390, dat_in=11101111, count=0111
# At time 2400, dat_in=11110000, count=0100
# At time 2410, dat_in=11110001, count=0101
# At time 2420, dat_in=11110010, count=0101
# At time 2430, dat_in=11110011, count=0110
# At time 2440, dat_in=11110100, count=0101
# At time 2450, dat_in=11110101, count=0110
# At time 2460, dat_in=11110110, count=0110
# At time 2470, dat_in=11110111, count=0111
# At time 2480, dat_in=11111000, count=0101
# At time 2490, dat_in=11111001, count=0110
# At time 2500, dat_in=11111010, count=0110
# At time 2510, dat_in=11111011, count=0111
# At time 2520, dat_in=11111100, count=0110
# At time 2530, dat_in=11111101, count=0111
# At time 2540, dat_in=11111110, count=0111
# At time 2550, dat_in=11111111, count=1000
```

EXP2-7

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'b0;
 else begin
```

```
  case (count)
  4'd0: count <= 4'd1;
  4'd1: count <= 4'd2;
  4'd2: count <= 4'd3;
  4'd3: count <= 4'd4;
  4'd4: count <= 4'd5;
  4'd5: count <= 4'd6;
  4'd6: count <= 4'd7;
  4'd7: count <= 4'd8;
  4'd8: count <= 4'd9;
  4'd9: count <= 4'd10;
  4'd10: count <= 4'd0;
  default: count <= 4'b0;
  endcase
 end
 end

endmodule
```

测试模块

```
// File: tb_dec_counter.v
`include "dec_counter.v"

module tb_dec_counter;

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

    dec_counter a(count, clk, reset);

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

    initial begin
        reset = 1'b0;
        forever #150 reset = ~reset;
    end

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

endmodule
```

仿真结果

显示输出

```
# At time     0, reset=0, count= x
# At time    20, reset=0, count= 0
# At time    60, reset=0, count= 1
# At time   100, reset=0, count= 2
# At time   140, reset=0, count= 3
# At time   150, reset=1, count= 3
# At time   180, reset=1, count= 0
# At time   300, reset=0, count= 1
# At time   340, reset=0, count= 2
# At time   380, reset=0, count= 3
# At time   420, reset=0, count= 4
# At time   450, reset=1, count= 4
# At time   460, reset=1, count= 0
```

EXP2-8

设计模块

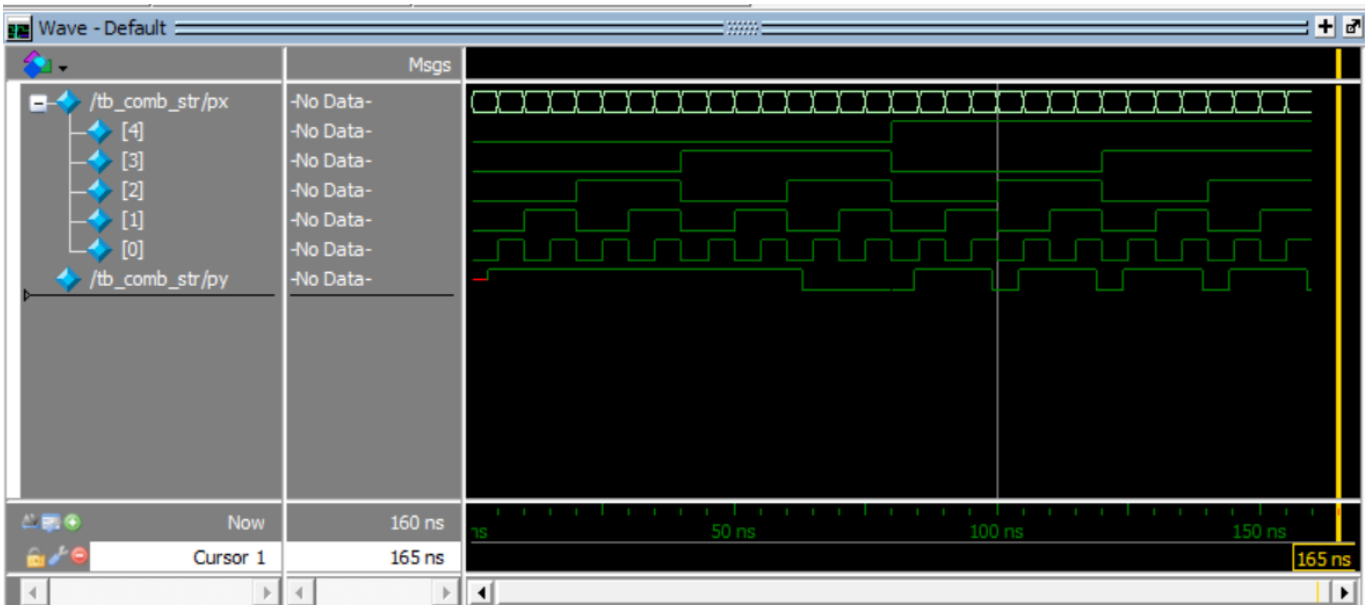
```
// File: comb_str_2.v
`include "mux2x1.v"
module comb_str( y, sel, a, b, c, d);
 output y;
 input sel;
 input a, b, c, d;
 wire s0, s1;
 nand #3 u1( s0, a, b);
 nand #4 u2( s1, c, d);

 mux2x1 m0( y, sel, {s1,s0});
```

```
endmodule
```

测试模块

```
// File: tb_comb_str_2.v

`include "comb_str_2.v"
`define STOP 32

module tb_comb_str();
 reg [4:0] px;
 wire py;
 integer i;

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

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

仿真结果



显示输出

```
# At time                0, y=x, sel=0, a=0, b=0, c=0, d=0
# At time                3, y=1, sel=0, a=0, b=0, c=0, d=0
# At time                5, y=1, sel=0, a=0, b=0, c=0, d=1
# At time               10, y=1, sel=0, a=0, b=0, c=1, d=0
# At time               15, y=1, sel=0, a=0, b=0, c=1, d=1
# At time               20, y=1, sel=0, a=0, b=1, c=0, d=0
# At time               25, y=1, sel=0, a=0, b=1, c=0, d=1
# At time               30, y=1, sel=0, a=0, b=1, c=1, d=0
# At time               35, y=1, sel=0, a=0, b=1, c=1, d=1
# At time               40, y=1, sel=0, a=1, b=0, c=0, d=0
# At time               45, y=1, sel=0, a=1, b=0, c=0, d=1
# At time               50, y=1, sel=0, a=1, b=0, c=1, d=0
# At time               55, y=1, sel=0, a=1, b=0, c=1, d=1
# At time               60, y=1, sel=0, a=1, b=1, c=0, d=0
# At time               63, y=0, sel=0, a=1, b=1, c=0, d=0
# At time               65, y=0, sel=0, a=1, b=1, c=0, d=1
# At time               70, y=0, sel=0, a=1, b=1, c=1, d=0
# At time               75, y=0, sel=0, a=1, b=1, c=1, d=1
# At time               80, y=0, sel=1, a=0, b=0, c=0, d=0
# At time               84, y=1, sel=1, a=0, b=0, c=0, d=0
# At time               85, y=1, sel=1, a=0, b=0, c=0, d=1
# At time               90, y=1, sel=1, a=0, b=0, c=1, d=0
# At time               95, y=1, sel=1, a=0, b=0, c=1, d=1
# At time               99, y=0, sel=1, a=0, b=0, c=1, d=1
# At time              100, y=0, sel=1, a=0, b=1, c=0, d=0
# At time              104, y=1, sel=1, a=0, b=1, c=0, d=0
# At time              105, y=1, sel=1, a=0, b=1, c=0, d=1
# At time              110, y=1, sel=1, a=0, b=1, c=1, d=0
# At time              115, y=1, sel=1, a=0, b=1, c=1, d=1
# At time              119, y=0, sel=1, a=0, b=1, c=1, d=1
# At time              120, y=0, sel=1, a=1, b=0, c=0, d=0
# At time              124, y=1, sel=1, a=1, b=0, c=0, d=0
# At time              125, y=1, sel=1, a=1, b=0, c=0, d=1
# At time              130, y=1, sel=1, a=1, b=0, c=1, d=0
# At time              135, y=1, sel=1, a=1, b=0, c=1, d=1
# At time              139, y=0, sel=1, a=1, b=0, c=1, d=1
# At time              140, y=0, sel=1, a=1, b=1, c=0, d=0
# At time              144, y=1, sel=1, a=1, b=1, c=0, d=0
# At time              145, y=1, sel=1, a=1, b=1, c=0, d=1
# At time              150, y=1, sel=1, a=1, b=1, c=1, d=0
# At time              155, y=1, sel=1, a=1, b=1, c=1, d=1
# At time              159, y=0, sel=1, a=1, b=1, c=1, d=1
```

EXP2-9

设计模块

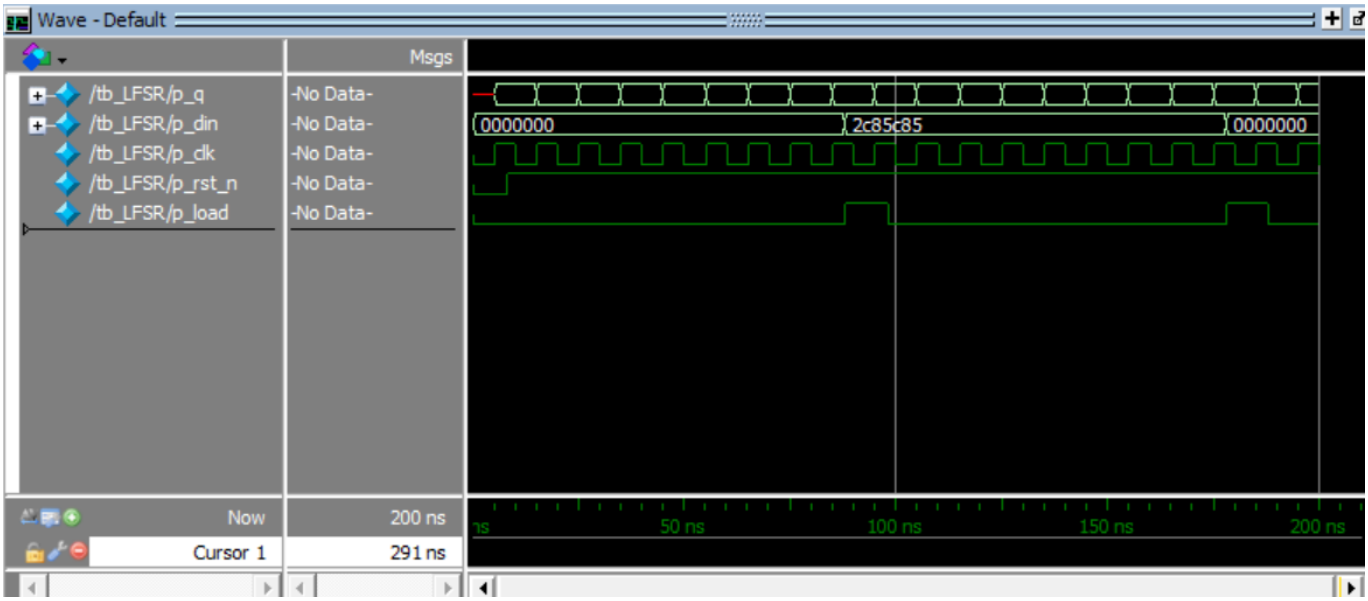
```
//File: LFSR.v
module LFSR(q, clk, rst_n, load, din);

    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 begin
                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
    end

endmodule
```

测试模块

```
//File: tb_LFSR.v

`include "LFSR.v"

module tb_LFSR();
 wire [1:26] p_q;
 reg [1:26] p_din;
 reg p_clk, p_rst_n, p_load;

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

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

```
    end

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

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

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

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

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

仿真结果



显示输出

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

EXP2-10

设计模块

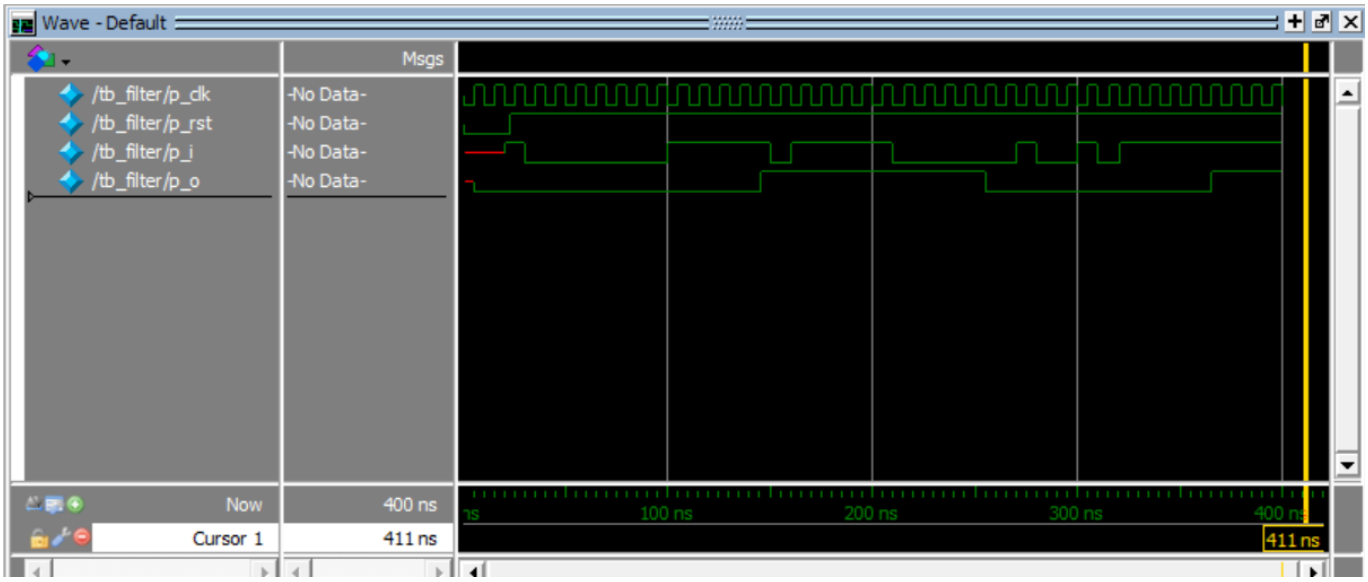
```
//File: filter.v
module filter(sig_out, clock, reset, sig_in);
 output sig_out;
 input clock, reset, sig_in;

 reg [3:0] data;
 reg dout;

 always @(posedge clock) begin
 if (!reset) begin
 data <= 4'b0;
 dout <= 1'b0;
 end
 else begin
```

```
 data <= {data[2:0], sig_in};

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

测试模块

```
//File: tb_filter.v

`include "filter.v"
module tb_filter;
 wire p_o;
 reg p_clk, p_rst, p_i;

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

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

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

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

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

仿真结果



显示输出

```
# At time                   0, reset=0, sig_in=x, sig_out=x
# At time                   5, reset=0, sig_in=x, sig_out=0
# At time                  20, reset=0, sig_in=1, sig_out=0
# At time                  23, reset=1, sig_in=1, sig_out=0
# At time                  30, reset=1, sig_in=0, sig_out=0
# At time                 100, reset=1, sig_in=1, sig_out=0
# At time                 145, reset=1, sig_in=1, sig_out=1
# At time                 150, reset=1, sig_in=0, sig_out=1
# At time                 160, reset=1, sig_in=1, sig_out=1
# At time                 210, reset=1, sig_in=0, sig_out=1
# At time                 255, reset=1, sig_in=0, sig_out=0
# At time                 270, reset=1, sig_in=1, sig_out=0
# At time                 280, reset=1, sig_in=0, sig_out=0
# At time                 300, reset=1, sig_in=1, sig_out=0
# At time                 310, reset=1, sig_in=0, sig_out=0
# At time                 320, reset=1, sig_in=1, sig_out=0
# At time                 365, reset=1, sig_in=1, sig_out=1
```

EXP2-11

设计模块

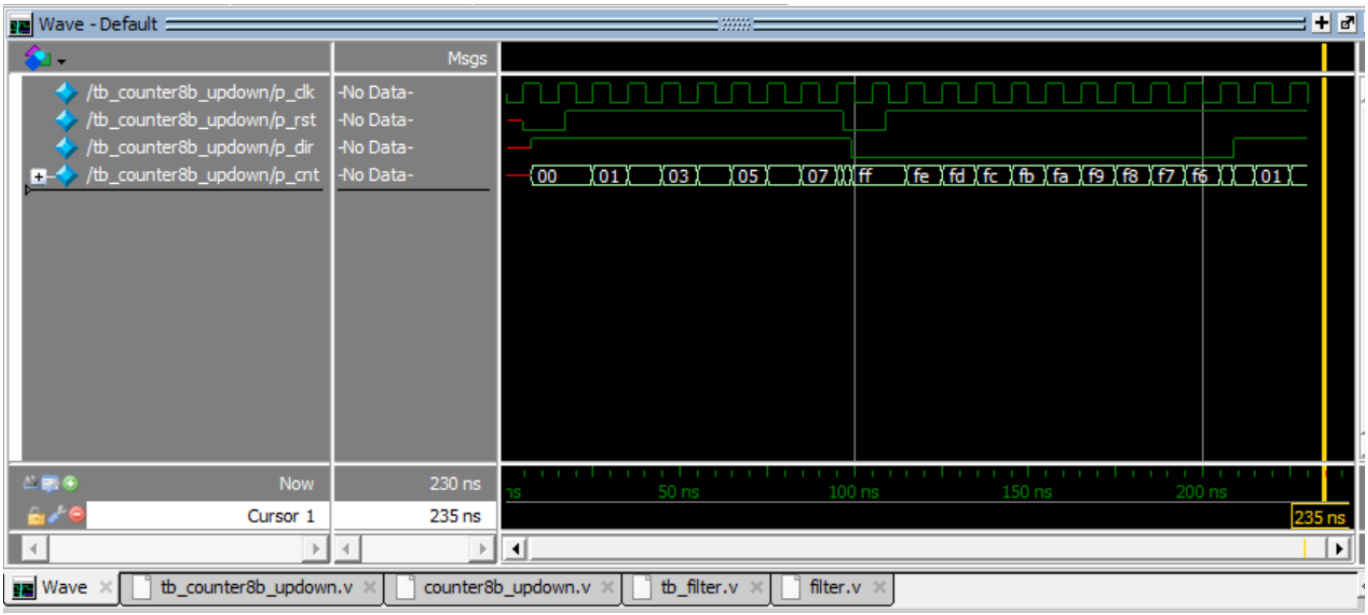
```
// File: counter8b_updown.v
module counter8b_updown(output [7:0] count, input clk, reset, dir);
 reg [7:0] count0,  count1;
 always @( posedge clk or negedge reset )
 begin
  if ( reset == 1'b0 ) begin
```

```
    count0 <= 8'b0;
    count1 <= 8'b1111_1111;
   end
  else if ( dir )
   count0 <= count0 + 1;
  else
   count1 <= count1 - 1;
  end
assign count = (dir) ?  count0 : count1;
endmodule
```

测试模块

```
// File: tb_counter8b_updown.v

`include "counter8b_updown.v"

module tb_counter8b_updown();
 wire [7:0] p_cnt;
 reg p_clk,  p_rst,  p_dir;

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

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

 counter8b_updown m0(.count(p_cnt), .clk(p_clk), .reset(p_rst), .dir(p_dir) );
 initial
 $monitor("At time %t, <----> count=%b, dir=%b" ,  $time,  p_cnt,  p_dir );

endmodule
```

仿真结果



显示输出

```
# At time                   0, <----> count=xxxxxxxx, dir=x
# At time                   7, <----> count=00000000, dir=1
# At time                  25, <----> count=00000001, dir=1
# At time                  35, <----> count=00000010, dir=1
# At time                  45, <----> count=00000011, dir=1
# At time                  55, <----> count=00000100, dir=1
# At time                  65, <----> count=00000101, dir=1
# At time                  75, <----> count=00000110, dir=1
# At time                  85, <----> count=00000111, dir=1
# At time                  95, <----> count=00001000, dir=1
# At time                  97, <----> count=00000000, dir=1
# At time                  99, <----> count=11111111, dir=0
# At time                 115, <----> count=11111110, dir=0
# At time                 125, <----> count=11111101, dir=0
# At time                 135, <----> count=11111100, dir=0
# At time                 145, <----> count=11111011, dir=0
# At time                 155, <----> count=11111010, dir=0
# At time                 165, <----> count=11111001, dir=0
# At time                 175, <----> count=11111000, dir=0
# At time                 185, <----> count=11110111, dir=0
# At time                 195, <----> count=11110110, dir=0
# At time                 205, <----> count=11110101, dir=0
# At time                 209, <----> count=00000000, dir=1
# At time                 215, <----> count=00000001, dir=1
# At time                 225, <----> count=00000010, dir=1
```

EXP2-12

设计模块

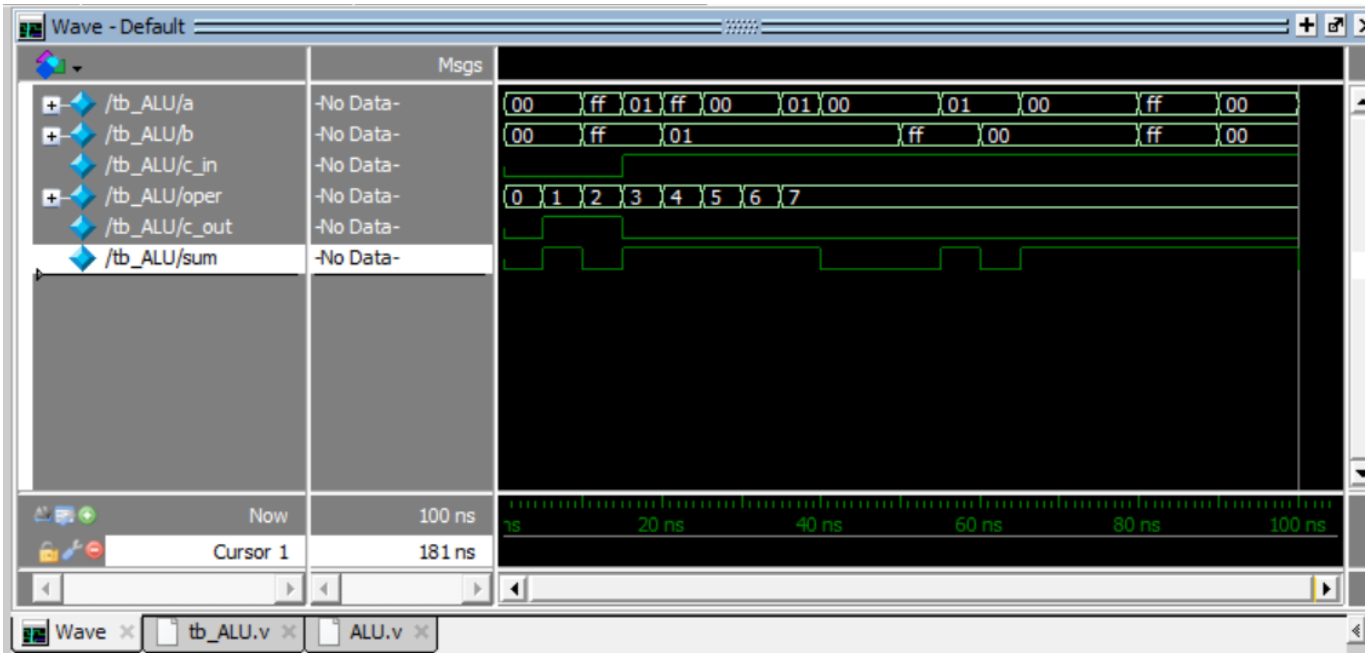
```
//File: ALU.v
module ALU(c_out, sum, oper, a, b, c_in);

    output reg [7:0] sum;
    output reg c_out;
    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} <= ~a + b + ~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
```

测试模块

```
// File: tb_ALU.v

`include "ALU.v"

module tb_ALU;

    parameter STEP = 8;
    integer k;

    wire c_out, 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 = 8'b0;
        forever #5 a = {a[7:0], $random % 2};
    end

    initial begin
        b = 8'b0;
        forever #10 b = {b[7:0], $random % 2};
    end

    initial begin
        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

endmodule
```

仿真结果



显示输出

```
# At time     0, a=00000000, b=00000000, c_in=0, oper=000, sum=0, c_out=0
# At time     5, a=00000000, b=00000000, c_in=0, oper=001, sum=1, c_out=1
# At time    10, a=11111111, b=11111111, c_in=0, oper=010, sum=0, c_out=1
# At time    15, a=00000001, b=11111111, c_in=1, oper=011, sum=1, c_out=0
# At time    20, a=11111111, b=00000001, c_in=1, oper=100, sum=1, c_out=0
# At time    25, a=00000000, b=00000001, c_in=1, oper=101, sum=1, c_out=0
# At time    30, a=00000000, b=00000001, c_in=1, oper=110, sum=1, c_out=0
# At time    35, a=00000001, b=00000001, c_in=1, oper=111, sum=1, c_out=0
# At time    40, a=00000000, b=00000001, c_in=1, oper=111, sum=0, c_out=0
# At time    50, a=00000000, b=11111111, c_in=1, oper=111, sum=0, c_out=0
# At time    55, a=00000001, b=11111111, c_in=1, oper=111, sum=1, c_out=0
# At time    60, a=00000001, b=00000000, c_in=1, oper=111, sum=0, c_out=0
# At time    65, a=00000000, b=00000000, c_in=1, oper=111, sum=1, c_out=0
# At time    80, a=11111111, b=11111111, c_in=1, oper=111, sum=1, c_out=0
# At time    90, a=00000000, b=00000000, c_in=1, oper=111, sum=1, c_out=0
```

EXP2-13

设计模块

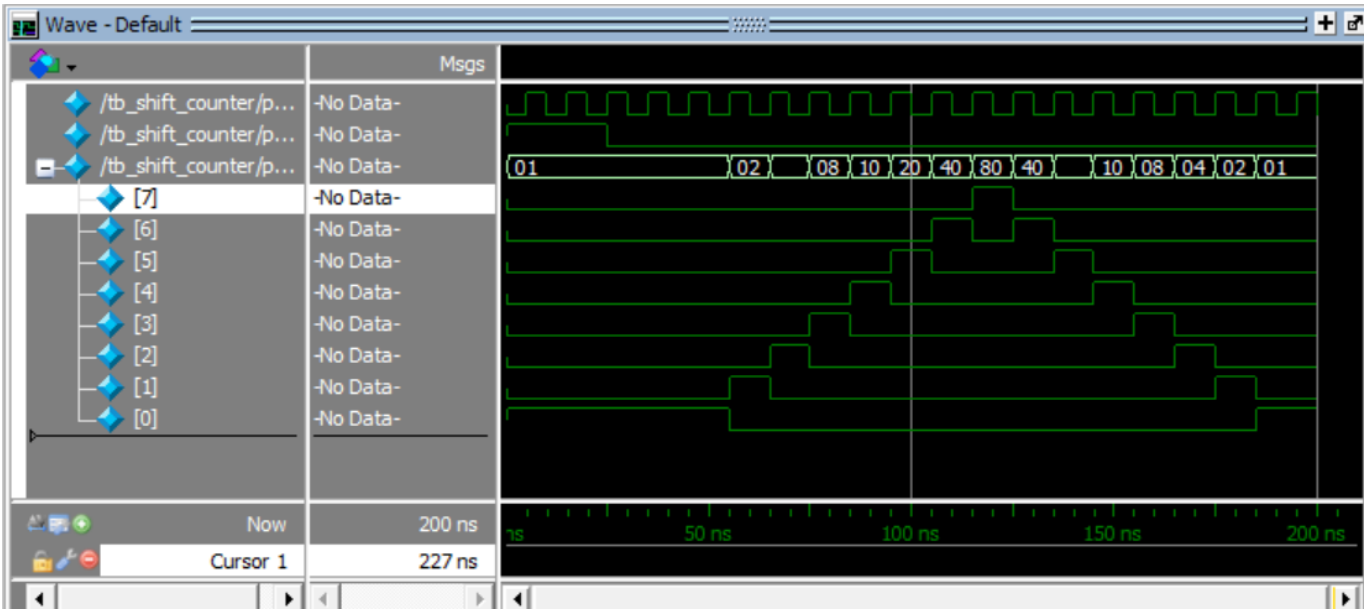
```
// File: shift_counter.v
module shift_counter(count, clk, reset);
 output [7:0] count;
 input clk;
 input reset;
 parameter CNTSUP = 17;
 reg [4:0] cnt;
 always @( posedge clk ) begin
  if ( reset )
   cnt <= 5'b0;
  else
  begin
  if (cnt < CNTSUP)
   cnt <= cnt + 1'b1;
  else
   cnt <= 5'b0;
  end
 end

 function [7:0] val( input [4:0] c );
 begin
 case ( c)
  0: val = 8'b0000_0001;
  1: val = 8'b0000_0001;
  2: val = 8'b0000_0001;
  3: val = 8'b0000_0001;
  4: val = 8'b0000_0010;
  5: val = 8'b0000_0100;
  6: val = 8'b0000_1000;
```

```
    7: val = 8'b0001_0000;
    8: val = 8'b0010_0000;
    9: val = 8'b0100_0000;
    10: val = 8'b1000_0000;
    11: val = 8'b0100_0000;
    12: val = 8'b0010_0000;
    13: val = 8'b0001_0000;
    14: val = 8'b0000_1000;
    15: val = 8'b0000_0100;
    16: val = 8'b0000_0010;
    17: val = 8'b0000_0001;
    default:val = 8'b0000_0001;
  endcase
  end
  endfunction
  assign count = val(cnt);
endmodule
```

测试模块

```
`include "shift_counter.v"
`define PULSE 5
module tb_shift_counter;
  wire [7:0] p_cnt;
  reg p_clk;
  reg p_rst;
  initial begin
    p_rst = 1'b1;
    #25 p_rst = 1'b0;
  end

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

  shift_counter u0(.count(p_cnt), .clk(p_clk), .reset(p_rst));
  initial
  $monitor( "At time %4t, reset=%b, count=%b", $time, p_rst, p_cnt );
endmodule
```

仿真结果



显示输出

```
# At time     0, reset=1, count=00000001
# At time    25, reset=0, count=00000001
# At time    55, reset=0, count=00000010
# At time    65, reset=0, count=00000100
# At time    75, reset=0, count=00001000
# At time    85, reset=0, count=00010000
# At time    95, reset=0, count=00100000
# At time   105, reset=0, count=01000000
# At time   115, reset=0, count=10000000
# At time   125, reset=0, count=01000000
# At time   135, reset=0, count=00100000
# At time   145, reset=0, count=00010000
# At time   155, reset=0, count=00001000
# At time   165, reset=0, count=00000100
# At time   175, reset=0, count=00000010
# At time   185, reset=0, count=00000001
```

EXP2-14

设计模块

```
// File: sram.v
module sram(dout, din, addr, wr, rd, cs);

    output [7:0] dout;
    input [7:0] din;
    input [7:0] addr;
```

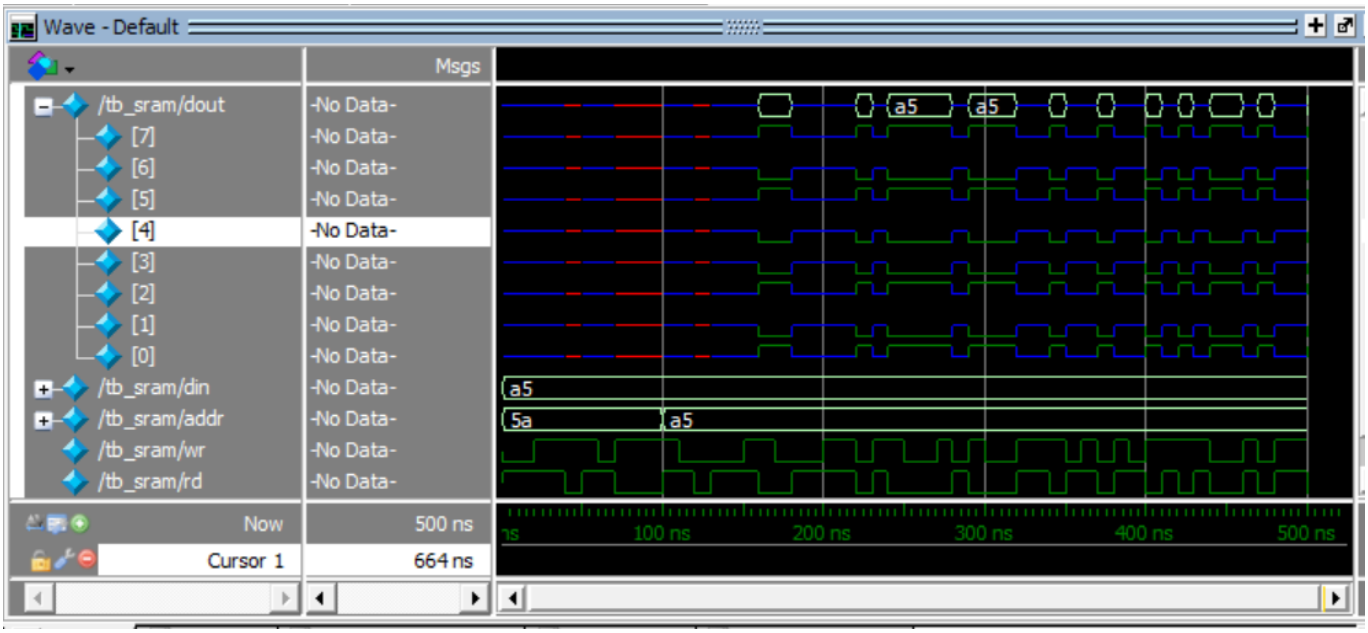
```
    input wr, rd, cs;

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

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

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

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

endmodule
```

测试模块

```
// File: tb_sram.v
`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
        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

endmodule
```

仿真结果



显示输出

```
# At time    0, din=10100101, addr=01011010, cs=0, wr=0, rd=1, dout=zzzzzzzz
# At time   10, din=10100101, addr=01011010, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time   20, din=10100101, addr=01011010, cs=0, wr=1, rd=1, dout=zzzzzzzz
# At time   30, din=10100101, addr=01011010, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time   40, din=10100101, addr=01011010, cs=1, wr=1, rd=0, dout=xxxxxxxx
# At time   50, din=10100101, addr=01011010, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time   60, din=10100101, addr=01011010, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time   70, din=10100101, addr=01011010, cs=1, wr=1, rd=0, dout=xxxxxxxx
# At time  100, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time  110, din=10100101, addr=10100101, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time  120, din=10100101, addr=10100101, cs=1, wr=0, rd=0, dout=xxxxxxxx
# At time  130, din=10100101, addr=10100101, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time  150, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time  160, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time  170, din=10100101, addr=10100101, cs=1, wr=0, rd=0, dout=10100101
# At time  180, din=10100101, addr=10100101, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time  200, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time  220, din=10100101, addr=10100101, cs=1, wr=0, rd=0, dout=10100101
# At time  230, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time  240, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time  250, din=10100101, addr=10100101, cs=1, wr=0, rd=0, dout=10100101
# At time  270, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time  280, din=10100101, addr=10100101, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time  290, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time  300, din=10100101, addr=10100101, cs=1, wr=0, rd=0, dout=10100101
# At time  320, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time  340, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time  350, din=10100101, addr=10100101, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time  360, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time  370, din=10100101, addr=10100101, cs=1, wr=0, rd=0, dout=10100101
# At time  380, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time  390, din=10100101, addr=10100101, cs=1, wr=0, rd=1, dout=zzzzzzzz
# At time  400, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time  410, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
# At time  420, din=10100101, addr=10100101, cs=1, wr=1, rd=0, dout=10100101
# At time  430, din=10100101, addr=10100101, cs=1, wr=1, rd=1, dout=zzzzzzzz
```

EXP2-15

设计模块

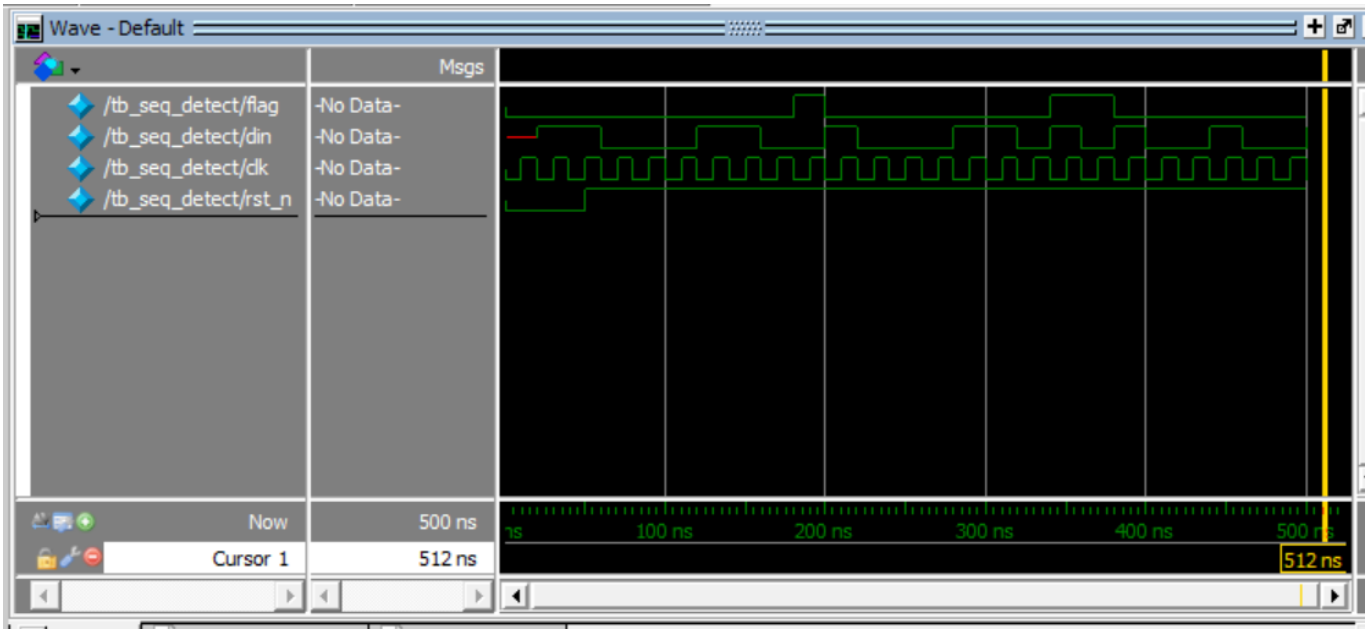
```
// File: seq_detect.v

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

    output reg flag;
    input din, clk, rst_n;

    parameter S10 = 9'b0_0000_0001; // IDLE(0)
    parameter S11 = 9'b0_0000_0010; // A(0)
```

```
    parameter S12 = 9'b0_0000_0100; // B(0)
    parameter S13 = 9'b0_0000_1000; // C(0)
    parameter S14 = 9'b0_0001_0000; // D(1)

    parameter S20 = 9'b0_0000_0001; // IDLE(0)
    parameter S21 = 9'b0_0010_0000; // E(0)
    parameter S22 = 9'b0_0100_0000; // F(0)
    parameter S23 = 9'b0_1000_0000; // G(0)
    parameter S24 = 9'b1_0000_0000; // H(1)

    reg [8:0] state1;   // 1101
    reg [8:0] state2;   // 0110
    reg flag1, flag2;

    always @ ( * ) begin
        flag <= flag1 | flag2;
    end

    always @ ( negedge clk ) begin
        if (!rst_n) begin
            flag1 <= 1'b0;
            state1 <= S10;
        end else begin
            flag1 <= (state1 == S14) ? 1'b1 : 1'b0;
            case (state1)
                S10: state1 <= (din) ? S11 : S10;
                S11: state1 <= (din) ? S12 : S10;
                S12: state1 <= (din) ? S12 : S13;
                S13: state1 <= (din) ? S14 : S10;
                S14: state1 <= (din) ? S12 : S10;
                default: begin state1 <= S10; flag1 <= 1'b0; end
            endcase
        end
    end

    always @ ( negedge clk ) begin
        if (!rst_n) begin
            flag2 <= 1'b0;
            state2 <= S20;
        end else begin
            flag2 <= (state2 == S24) ? 1'b1 : 1'b0;
            case (state2)
                S20: state2 <= (din) ? S20 : S21;
                S21: state2 <= (din) ? S22 : S21;
                S22: state2 <= (din) ? S23 : S21;
                S23: state2 <= (din) ? S20 : S24;
                S24: state2 <= (din) ? S22 : S21;
                default: begin state2 <= S20; flag2 <= 1'b0; end
            endcase
        end
    end

endmodule
```

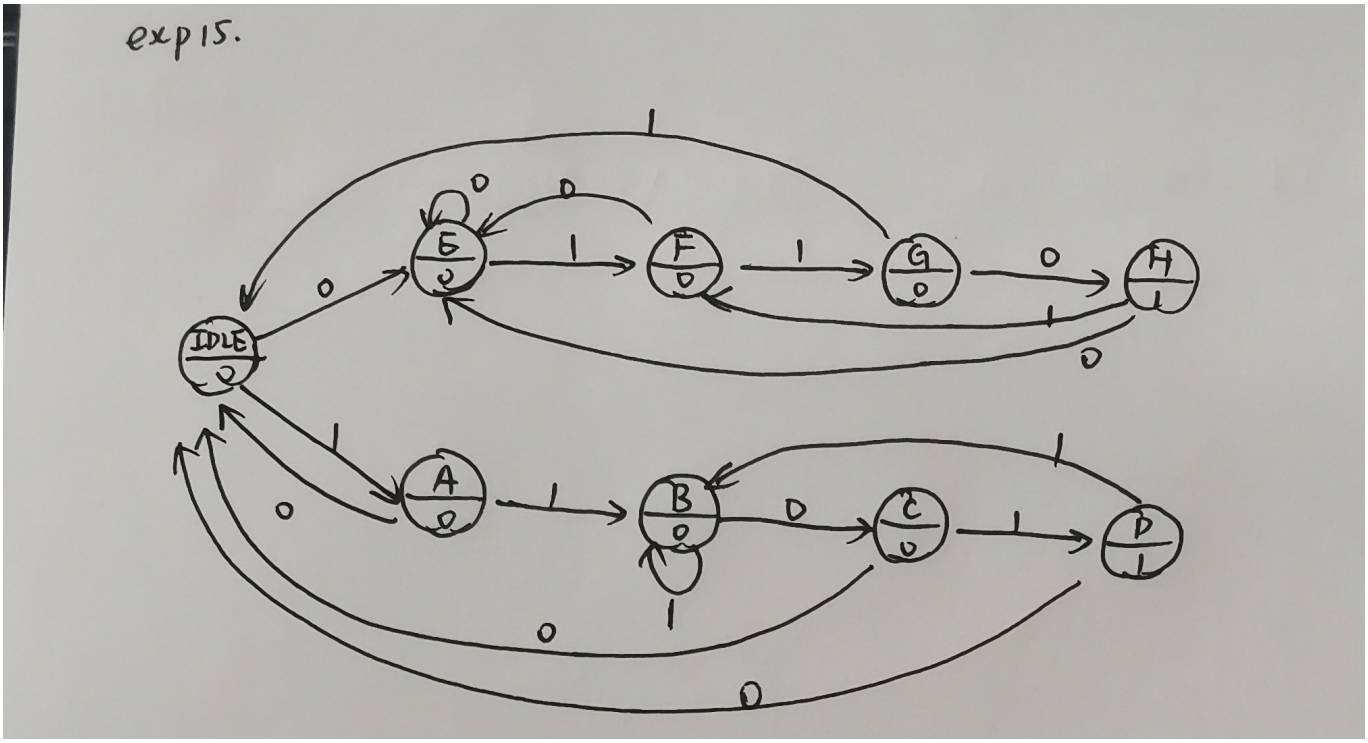
测试模块

```
// File: tb_seq_detect.v

`include "seq_detect.v"

module tb_seq_detect;

    parameter STEP = 32;
    integer k;

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

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

    initial begin
        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
    end

endmodule
```

仿真结果



设计说明--状态转移图



EXP2-16

设计模块

```
// File: mealy.v

module mealy(flag, din, clk, rst);

    output reg flag;
    input din, clk, rst;
```

```
    parameter S0 = 8'b0000_0001;  // IDLE(0)
    parameter S1 = 8'b0000_0010;  // A(0)
    parameter S2 = 8'b0000_0100;  // B(0)
    parameter S3 = 8'b0000_1000;  // C(0)
    parameter S4 = 8'b0001_0000;  // D(0)
    parameter S5 = 8'b0010_0000;  // E(0)
    parameter S6 = 8'b0100_0000;  // F(0)
    parameter S7 = 8'b1000_0000;  // G(0)

    reg [8:0] state;

    always @ ( posedge rst ) begin
        flag <= 1'b0;
        state <= S0;
    end

    always @ ( posedge clk ) begin
        case (state)
            S0: if (din) begin state <= S0; flag <= 1'b0; end
                else begin state <= S1; flag <= 1'b0; end
            S1: if (din) begin state <= S2; flag <= 1'b0; end
                else begin state <= S1; flag <= 1'b0; end
            S2: if (din) begin state <= S0; flag <= 1'b0; end
                else begin state <= S3; flag <= 1'b0; end
            S3: if (din) begin state <= S4; flag <= 1'b0; end
                else begin state <= S1; flag <= 1'b0; end
            S4: if (din) begin state <= S0; flag <= 1'b0; end
                else begin state <= S5; flag <= 1'b0; end
            S5: if (din) begin state <= S6; flag <= 1'b0; end
                else begin state <= S1; flag <= 1'b0; end
            S6: if (din) begin state <= S0; flag <= 1'b0; end
                else begin state <= S7; flag <= 1'b0; end
            S7: if (din) begin state <= S6; flag <= 1'b1; end
                else begin state <= S1; flag <= 1'b0; end
            default: begin state <= S0; flag <= 1'b0; end
        endcase
    end

endmodule
```

```
// File: moore.v

module moore(flag, din, clk, rst);

    output reg flag;
    input din, clk, rst;

    parameter S0 = 9'b0_0000_0001;  // IDLE(0)
    parameter S1 = 9'b0_0000_0010;  // A(0)
    parameter S2 = 9'b0_0000_0100;  // B(0)
```

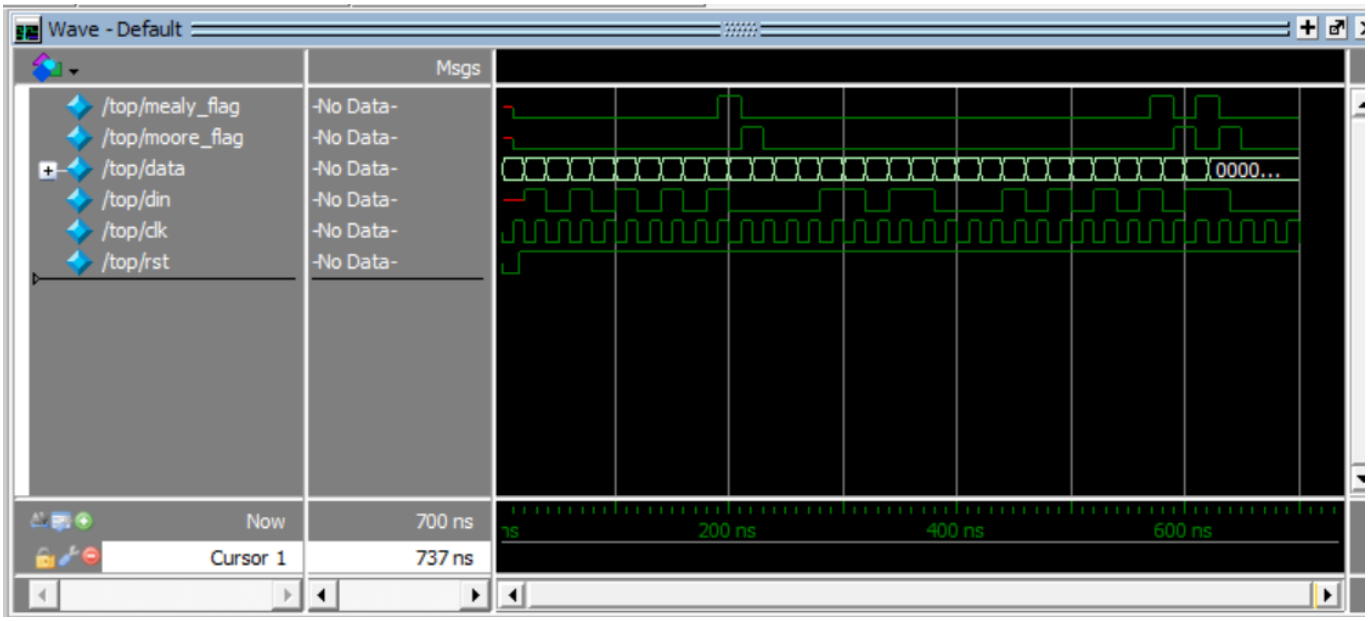
```
    parameter S3 = 9'b0_0000_1000;  // C(0)
    parameter S4 = 9'b0_0001_0000;  // D(0)
    parameter S5 = 9'b0_0010_0000;  // E(0)
    parameter S6 = 9'b0_0100_0000;  // F(0)
    parameter S7 = 9'b0_1000_0000;  // G(0)
    parameter S8 = 9'b1_0000_0000;  // H(1)

    reg [8:0] state;

    always @ ( posedge rst ) begin
        flag <= 1'b0;
        state <= S0;
    end

    always @ ( posedge clk ) begin
        flag <= (state == S8) ? 1'b1 : 1'b0;
        case (state)
            S0: state <= (din) ? S0 : S1;
            S1: state <= (din) ? S2 : S1;
            S2: state <= (din) ? S0 : S3;
            S3: state <= (din) ? S4 : S1;
            S4: state <= (din) ? S0 : S5;
            S5: state <= (din) ? S6 : S1;
            S6: state <= (din) ? S0 : S7;
            S7: state <= (din) ? S8 : S1;
            S8: state <= (din) ? S0 : S7;
            default: begin state <= S0; flag <= 1'b0; end
        endcase
    end

endmodule
```

测试模块

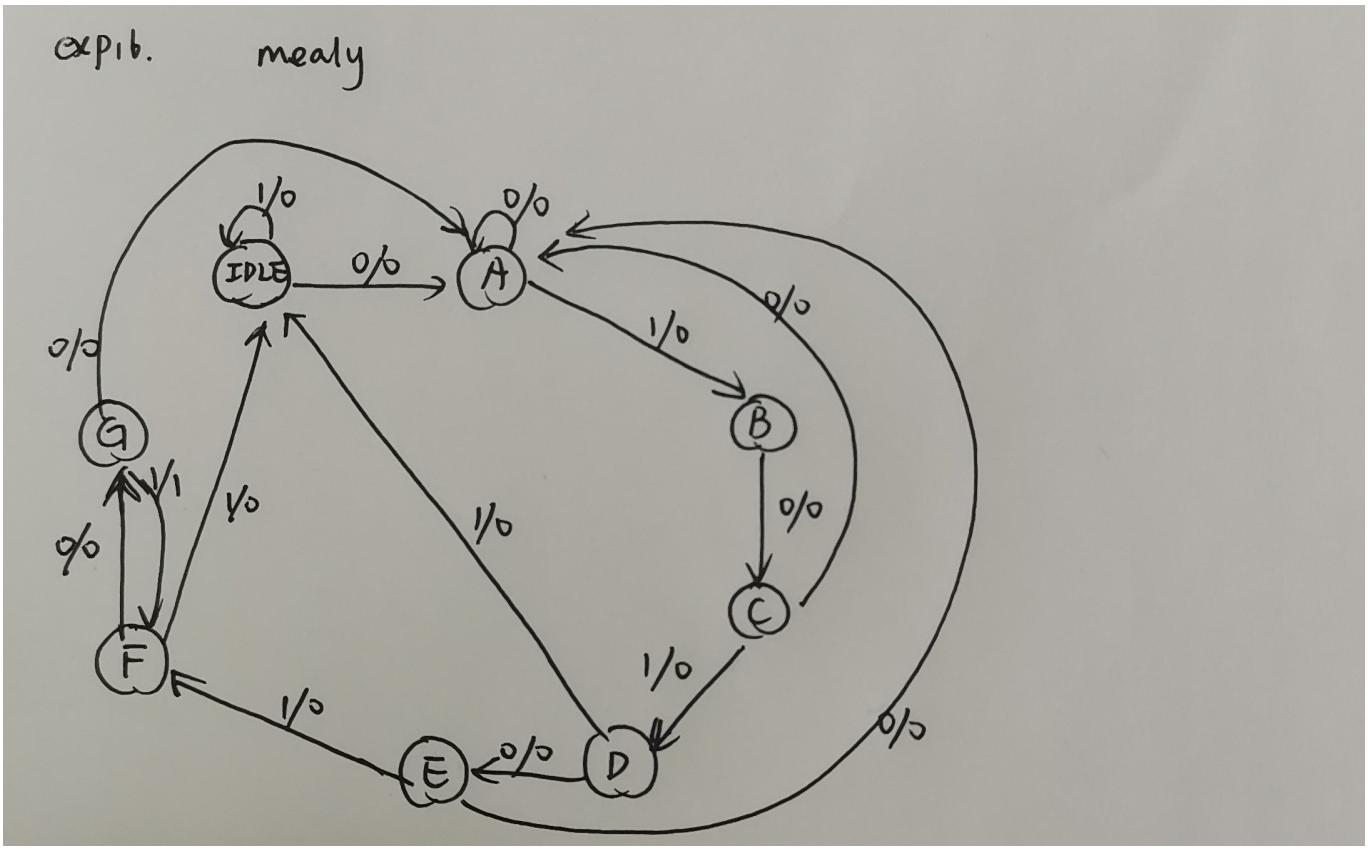
```
// File: top.v

`include "mealy.v"
`include "moore.v"

module top;

    parameter STEP = 32;
    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);
```

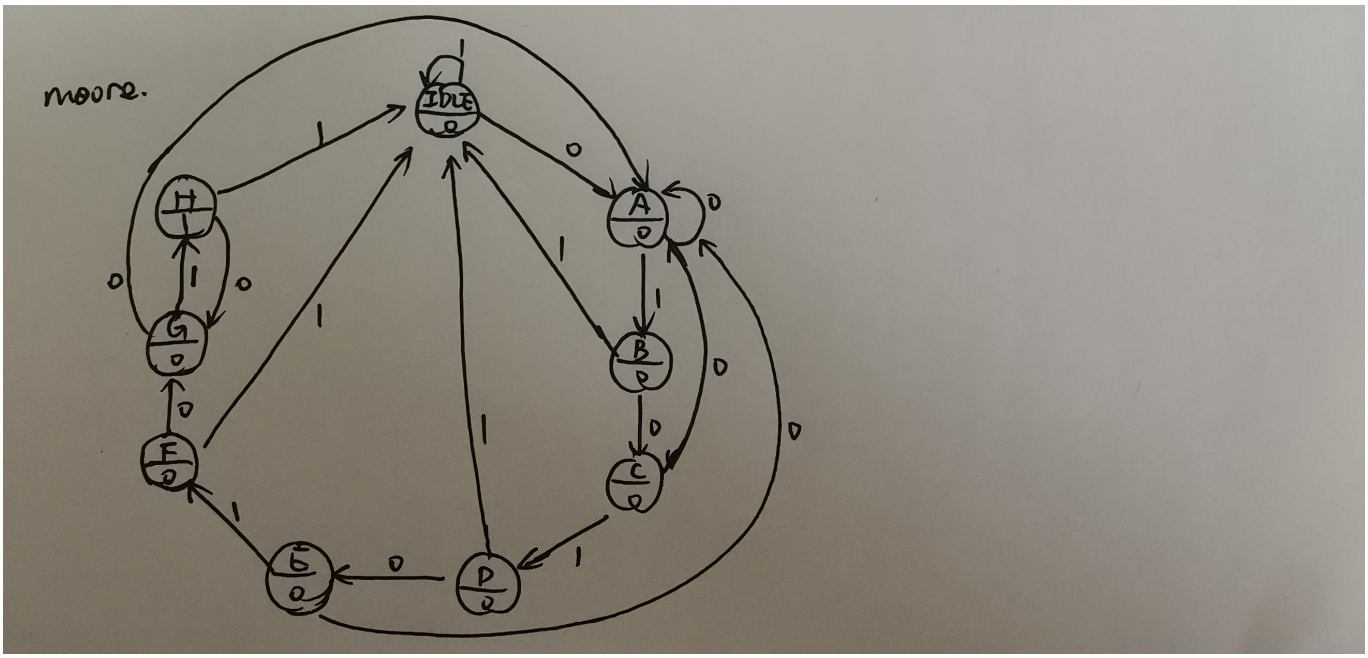
```
    initial begin
        clk = 1'b0;
        forever #10 clk = ~clk;
    end

    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
    end

endmodule
```

仿真结果



设计说明--状态转移图


expl6. mealy
IDLE
A
B
C
D
E
F
G
1/0
0/0


moore.
IDLE
A
B
C
D
E
F
G
H