《数字系统基础》模拟考试题

一. 逻辑运算与逻辑函数化简题

1. 利用逻辑代数基本公式和常用公式将下逻辑表达式化为最简与或式

$$Y = A + (B + C')' (A + B' + C)(A + B + C)$$

参考答案:

Y=A+(B+C’)’ ((A+C)+B’)((A+C)+B))

$\because A + BC = (A + C)(A + B)$

=A+(B+C)’(A+C)

$\because \overline{A + B} = \overline{A} \cdot \overline{B}$

=A+B’C(A+C)=A+AB’C+B’C

=A+B'C

2. 利用卡诺图法将下面逻辑表达式化为最简与或式

$$Y(A,B,C,D) = \sum m(0,1,2,5,8,9,10,12,14)$$

参考答案:

Y = A’C’D+B’C’+B’D’+AD’

卡诺图如下:

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	1	0	0	1
10	1	1	0	1

二． 电路如图 1 所示，请画出在 10 个 CP 脉冲和异步清零信号 Rd 作用下，各输出端 Q3，Q2，Q1，Q0 的电压波形。设触发器初态为 Q3Q2Q1Q0=1010

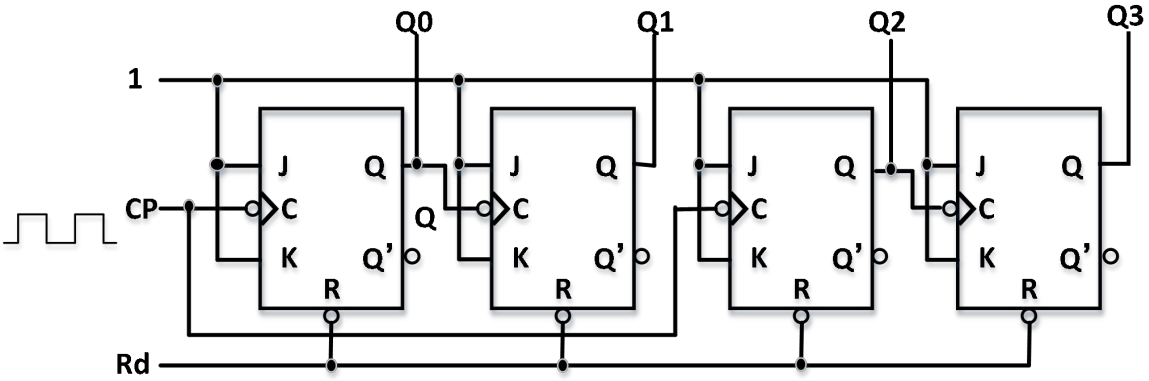
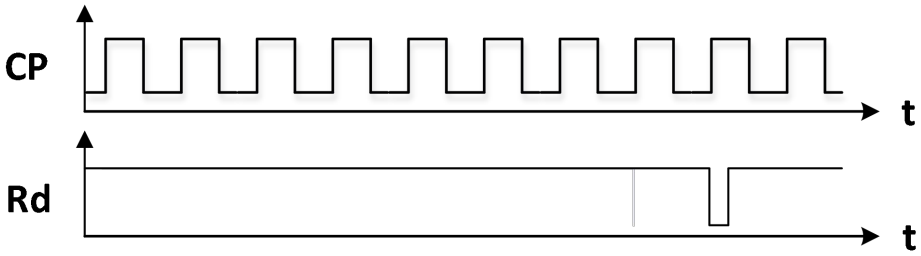


图 1



参考答案:

触发器初态为 Q3Q2Q1Q0=1010

JK 连接在一起 ，形成 T 触发器 输入为 1 功能是 翻转。

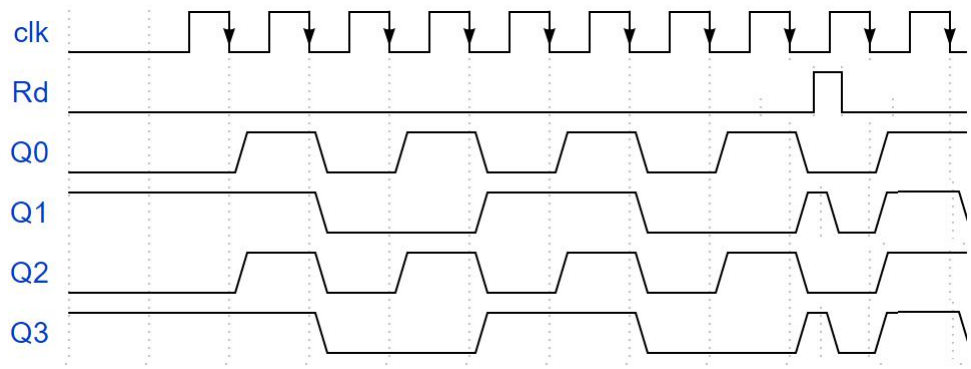
Q0 每个时钟下降沿翻转一次，其值依次为 10101010

然后 被 R 的信号清零 然后 10

Q1 当每个 Q0 的下降沿 翻转一次。然后 被 R 的信号清零 ，之后 再 翻转一次

Q2 与 Q0 相同

Q3 与 Q1 相同



三． 分析图 2 时序电路的逻辑功能。要求：

（1） 写出电路的驱动方程

（2） 写出状态方程和输出方程

（3） 画出电路的状态转换图

（4） 说明电路能否自启动。

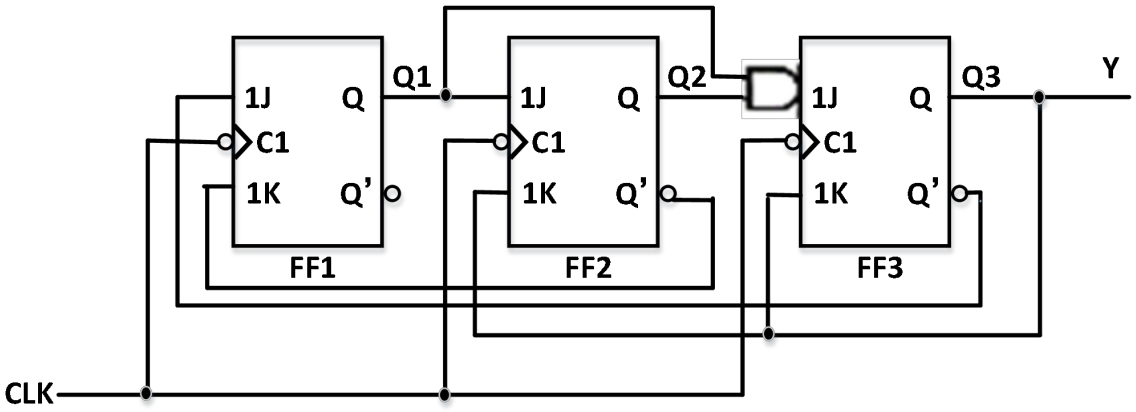


图 2

参考答案:

驱动方程：

(1)

FF3： J3 = Q2Q1
K3 = Q3

FF2： J2 = Q1
K2 = Q3

FF1: J1 = Q3’
K1 = Q2’

（2）

JK 触发器特性方程： $Q^* = JQ' + K'Q$

状态方程:

$Q1^* = Q3'Q1' + Q2Q1$

$Q2^* = Q1Q2' + Q3'Q2$

$Q3^* = Q2Q1Q3' + Q3'Q3 = Q2Q1Q3'$

输出方程：

Y = Q3

（3）

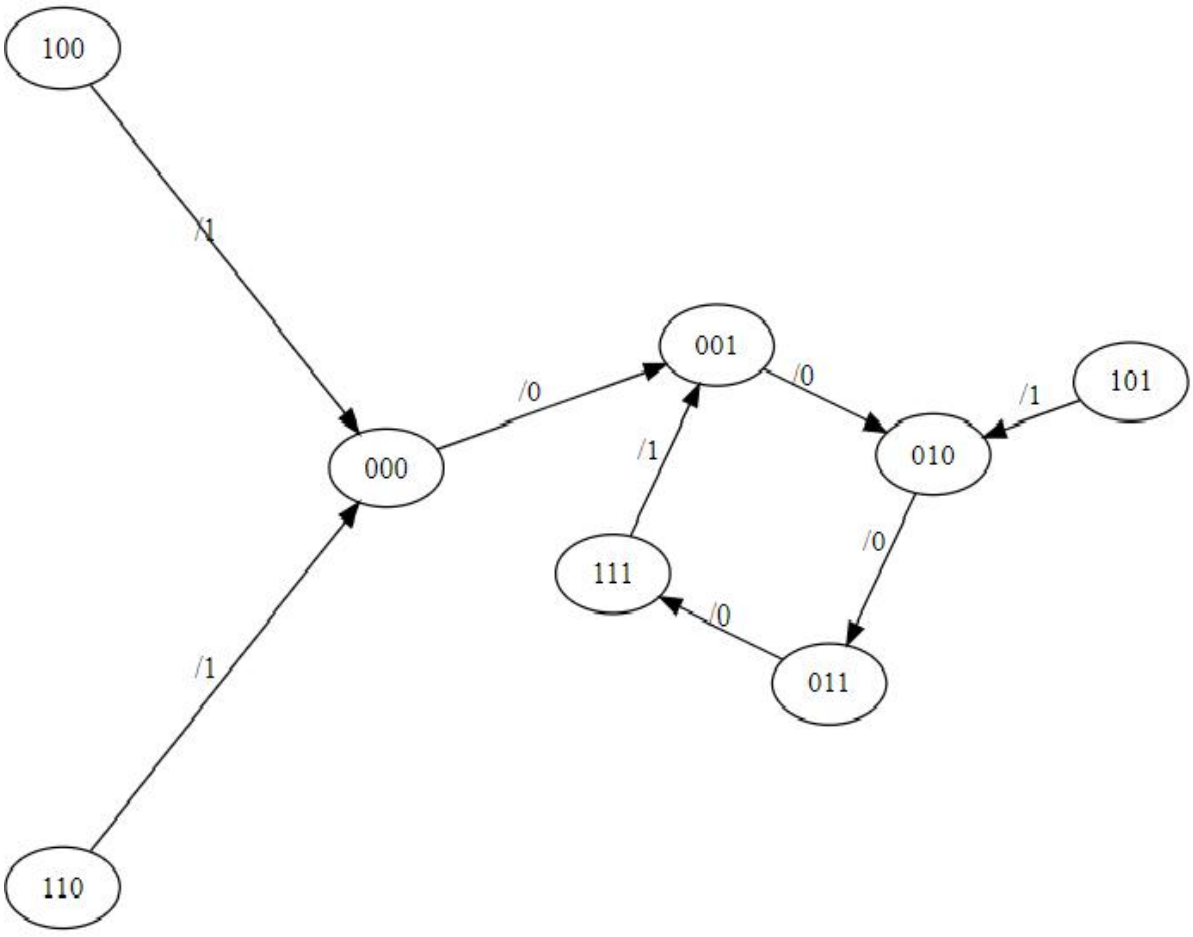
状态方程：

Q1* = Q3'Q1' + Q2Q1　　0x0+x11

Q2* = Q1Q2' + Q3'Q2　　x01+01x

Q3* = Q2Q1Q3' + Q3'Q3 = Q2Q1Q3'　　011

Q3Q2Q1	Q3*Q2*Q1*	Y
000	001	0
001	010	0
010	011	0
011	111	0
100	000	1
101	010	1
110	000	1
111	001	1



（5） 能

四. 分析 Verilog 语言描述的电路模块 A 完成何种逻辑功能,要求根据模块描述语句画出电路的状态转换图,说明模块 A 中各输入信号功能。

```
module A(clk,rst,s,out)
input clk,rst,s;
output[3:0] out;
wire[3:0] a,b;
wire flag;
reg[3:0] c;

assign a = out + 1;
assign b = out - 1;
assign flag = (out == 9)? 1:0;
assign c = (rst|flg) ? 0 : (s ? a : b);

always@(posedge clk)
  out = c;
endmodule
```

参考答案:

模块 A 中

clk 时钟

rst 复位,计数清零

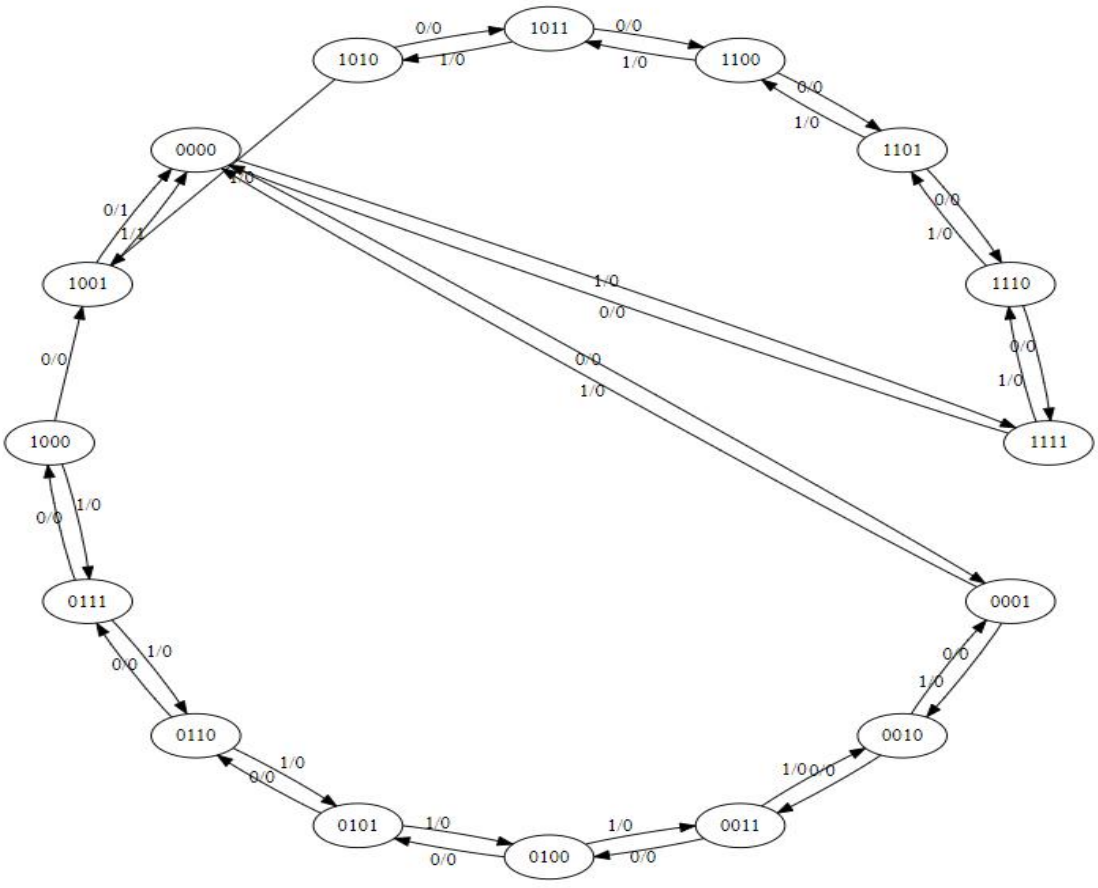
s 加、减计数选择信号，1 加法计数，0 减法计数

out 计数结果

状态表

```
# https://hifpga.com/fsm/
#状态机示例
digraph fsm {
"0000" -> "0001" [label= "0/0"]
"0001" -> "0010" [label= "0/0"]
"0010" -> "0011" [label= "0/0"]
"0011" -> "0100" [label= "0/0"]
"0100" -> "0101" [label= "0/0"]
"0101" -> "0110" [label= "0/0"]
"0110" -> "0111" [label= "0/0"]
"0111" -> "1000" [label= "0/0"]
"1000" -> "1001" [label= "0/0"]
"1001" -> "0000" [label= "0/1"]
"1010" -> "1011" [label= "0/0"]
"1011" -> "1100" [label= "0/0"]
"1100" -> "1101" [label= "0/0"]
"1101" -> "1110" [label= "0/0"]
"1110" -> "1111" [label= "0/0"]
"1111" -> "0000" [label= "0/0"]
"1010" -> "1001" [label= "1/0"]
"1011" -> "1010" [label= "1/0"]
"1100" -> "1011" [label= "1/0"]
"1101" -> "1100" [label= "1/0"]
"1110" -> "1101" [label= "1/0"]
"1111" -> "1110" [label= "1/0"]
"0000" -> "1111" [label= "1/0"]
"0001" -> "0000" [label= "1/0"]
"0010" -> "0001" [label= "1/0"]
"0011" -> "0010" [label= "1/0"]
"0100" -> "0011" [label= "1/0"]
"0101" -> "0100" [label= "1/0"]
"0110" -> "0101" [label= "1/0"]
"0111" -> "0110" [label= "1/0"]
"1000" -> "0111" [label= "1/0"]
"1001" -> "0000" [label= "1/1"]
}
```

状态图



五． 用 Verilog 语言描述一个带片选信号 CS 的 8 选 1 的数据选择器.要求片选信号 CS 为低电平有效.

参考答案:

写法一

```
module Mux8(a7, a6, a5, a4, a3, a2, a1, a0, cs, s, b) ;
    parameter       k = 1 ;
    input [k-1:0]    a0, a1, a2, a3 ;
    input [7:0]       s ; // one-hot select
    output[k-1:0]   b ;

    assign b =  cs ? {k{1'bz} : ({k{s[0]}} & a0) | ({k{s[1]}} & a1)
                |({k{s[2]}} & a2) | ({k{s[3]}} & a3| ({k{s[4]}} & a4| ({k{s[5]}} & a5
                | ({k{s[6]}} & a6| ({k{s[7]}} & a7);
endmodule
```

写法二

```
module Mux8(A,CS,D,Y)
input[2:0] A;
input CS;
input[7:0] D;
output Y;
reg Y1;
always@(A) begin
  case(A)
    3’b000: Y1 = D[0];
    3’b001: Y1 = D[1];
    3’b010: Y1 = D[2];
    3’b011: Y1 = D[3];
    3’b100: Y1 = D[4];
    3’b101: Y1 = D[5];
    3’b110: Y1 = D[6];
    3’b111: Y1 = D[7];
  endcase end
assign Y = CS ? 1’bZ : Y1;
endmodule
```

六． 用 Verilog 语言实现对图 3 所示逻辑电路的逻辑功能描述。

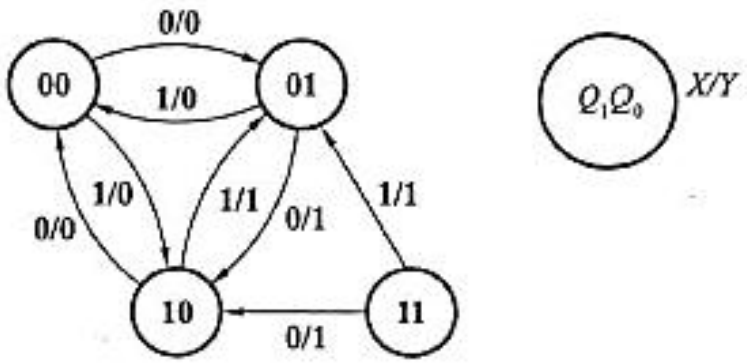


图 3

参考答案:

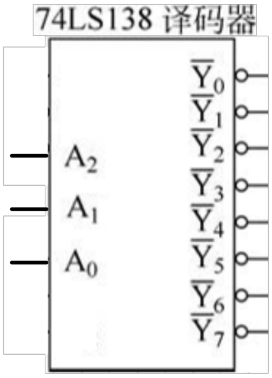
```
module FSM_3
//---------------------<端口声明>-------------------------------------
(clk,rst_n,X,Y,state_c);
input                   clk             ;
input                   rst_n           ;
input                   X               ;
output                  Y               ;
output   [1:0]          state_c
//---------------------<信号定义>-------------------------------------
reg  [1:0]              state_c         ;
reg  [1:0]              state_n         ;
//---------------------<状态机参数>-----------------------------------
localparam S0           = 2'b00         ;
localparam S1           = 2'b01         ;
localparam S2           = 2'b10         ;
localparam S3           = 2'b11         ;
```

```
//------------------------------------------------------------------------
//--   状态机第1段
//------------------------------------------------------------------------
always @(posedge clk or negedge rst_n)begin
    if(!rst_n)
        state_c <= S0;
    else
        state_c <= state_n;
end
//------------------------------------------------------------------------
//--   状态机第2段
//------------------------------------------------------------------------
always @(*)begin
    case(state_c)
        S0: begin
            if(X==1'b0)
                state_n = S1;
            else
                state_n = S2;
        end
        S1: begin
            if(X==1'b0)
                state_n = S2;
            else
                state_n = S0;
        end
        S2: begin
            if(X==1'b0)
                state_n = S0;
            else
                state_n = S1;
        end
        S3: begin
            if(X==1'b0)
                state_n = S2;
            else
                state_n = S1;
        end
        default:state_n = S0;
    endcase
end
//------------------------------------------------------------------------
//--   状态机第3段
//------------------------------------------------------------------------

always @(*)
begin
    if(state_c==S3 || (state_c==S1 && X==1'b0) || (state_c==S2 && X==1'b1))
        Y <= 1'b1;
    else
        Y <= 1'b0;
end

endmodule
```

七. 某个车间有红、黄两个故障指示灯，用来表示 A，B，C 三台设备的工作情况。如果一台设备出现故障，则黄灯亮；如果两台设备出现故障，则红灯亮；如果三台设备同时出现故障，则红灯和黄灯都亮。试用一片 74HC138 译码器和少量门电路设计一个能实现此要求的逻辑电路。



要求：

（1） 进行逻辑抽象，列出真值表

（2） 写出逻辑表达式并化简

（3） 画出电路连接图

参考答案:

（1） 进行逻辑抽象，列出真值表

输入信号 A，B，C 分别表示三个设备状态，1 表示设备故障，0 表示设备正常。

输出信号 R，Y 分别表示红、黄两个故障指示灯状态，1 表示灯亮，0 表示灯灭。

输入信号和输出信号真值表如下：

A	B	C	R	Y
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

（2） 写出最小项之和逻辑表达式

R = A’BC+AB’C+ABC’+ABC；

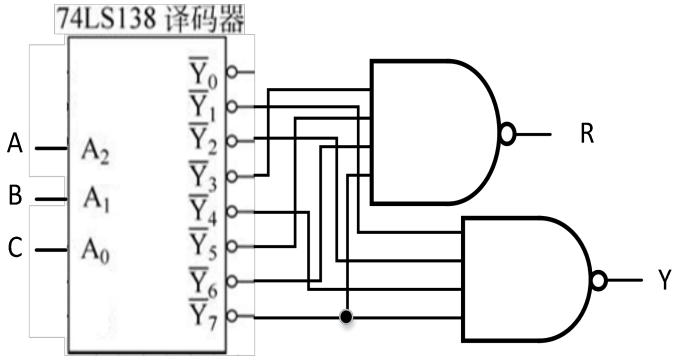
Y = A’B’C+A’BC’+AB’C’+ABC

（3） 画出电路连接图:

将 A，B，C 分别与 138 译码器的地址选择信号 A2，A1，A0 相连，则

R = Y3+Y5+Y6+Y7 = (Y3’Y5’Y6’Y7’)’;

Y = Y1+Y2+Y4+Y7 = (Y1’Y2’Y4’Y7’)’;



八． 试用 D 触发器和门电路设计一个同步七进制计数器

要求：

（1） 画出状态转换图和状态转换表

（2） 利用卡洛图进行化简得到状态方程

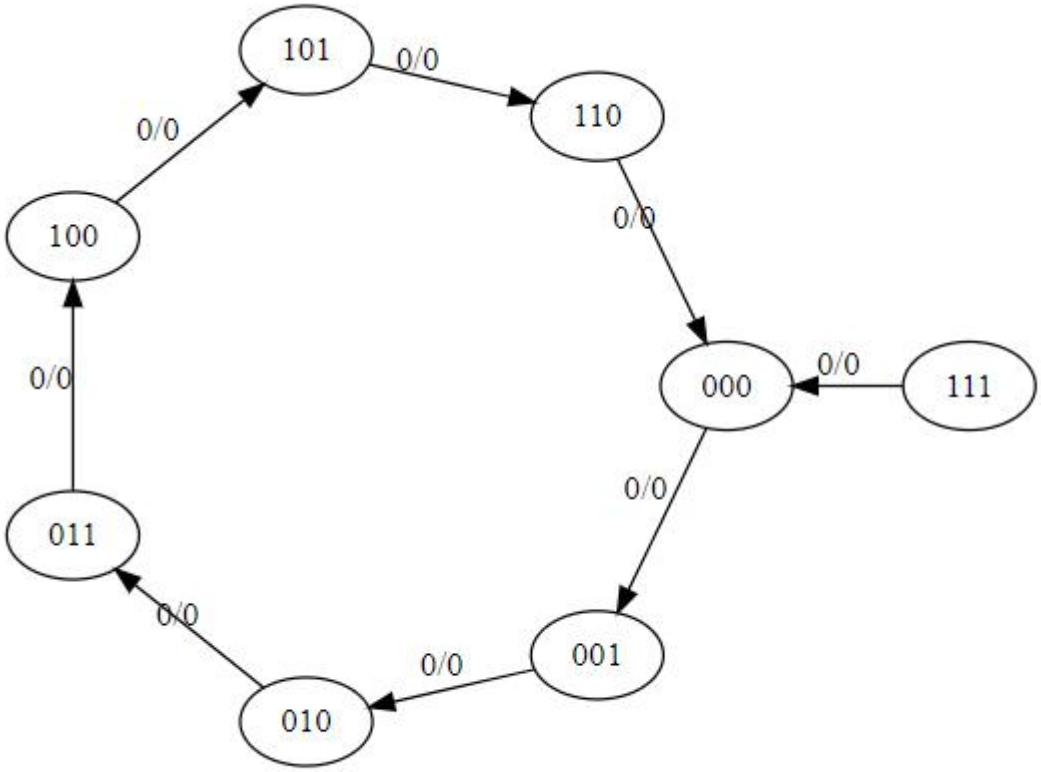
（3） 根据 D 触发器特性方程得到驱动方程

（4） 画出逻辑电路图

参考答案:

（1） 画出状态转换表

Q2*Q1*Q0* \ Q1Q0 / Q2	00	01	11	10
0	001	010	100	011
1	101	110	000	000



（2） 利用卡诺图化简得到状态方程（5 分）

$Q2^*$ \ Q1Q0 / Q2	00	01	11	10
0	0	0	1	0
1	1	1	X	0

$Q2^*=Q2Q1'+Q1Q0$

$Q1^*$ \ Q1Q0 / Q2	00	01	11	10
0	0	1	0	1
1	0	1	X	0

$Q1^*=Q1'Q0+Q2'Q1Q0'$

$Q0^*$ \ Q1Q0 / Q2	00	01	11	10
0	1	0	0	1
1	1	0	X	0

$Q0^*=Q1'Q0'+Q2'Q1Q0'$

（3） 根据 D 触发器特性方程得到驱动方程（5 分）

D2 = Q2Q1’+Q1Q0；D1 = Q1’Q0+Q2’Q1Q0’； D0 = Q1’Q0’+Q2’Q1Q0’；

（4）电路图 略

九. 一个口罩生产厂共有三条生产线L1,L2,L3可用于生产口罩,其中正常生产时只启动两条生产线L1,L2用于生产,L3为备用生产线.当市场中口罩紧缺时,才开启L3生产线进行生产.当市场对口罩需求量急剧下降时,工厂必须关闭其余生产线,只保留L1生产线用于生产.此外该口罩生产厂为了应对临时停电的情况,配备有一台发电机F,但该发电机发出的电能只能够维持生产线L1的运行,因此一旦停电(E=0),首先必须关闭全部在运行的生产线,然后开启发电机F,最后开启生产线L1.当恢复供电(E=1)时,首先必须关闭生产线L1,然后关闭发电机F,随后再开启生产线恢复生产.试设计一个逻辑控制电路用于控制生产线L1,L2,L3及发电机F的开启与关闭,假设市场供求关系用G表示,G=0表示市场供求关系平衡,G=1表示市场中口罩紧缺,G=2表示市场中口罩需求量急剧下降.

要求:

（1） 进行逻辑抽象,列出该控制电路输入和输出信号进行说明,

（2） 列出逻辑控制电路的状态转换表

（3） 用Verilog有限状态机描述该逻辑控制电路

参考答案:

(1) 设逻辑变量L1,L2,L3,F分别表示生产线L1,L2,L3和发电机的开启和关闭,开启为1,关闭为0;设逻辑变量E表示停电和来电状态,E=1表示来电,E=0表示停电;变量G表示市场供求关系,G=0表示市场供求关系平衡,G=1表示市场中口罩紧缺,G=2表示市场中口罩需求量急剧下降. 逻辑控制电路的输入信号包括E,G以及clk和复位信号rst四个量;输出信号包括L1,L2,L3,F四个量。

(2) 对题意进行分析可得出该电路共分为S0，S1，S2，S3，S4五个状态。分别表示为

解法1:

S0：正常生产；根据 G 的不同，开启和关闭不同的生产线，发电机关闭 F=0；

G = 0 时，L1=L2=1，L3=0；

G = 1 时，L1=L2=L3=1;

G = 2 时，L1=1，L2=L3=0；

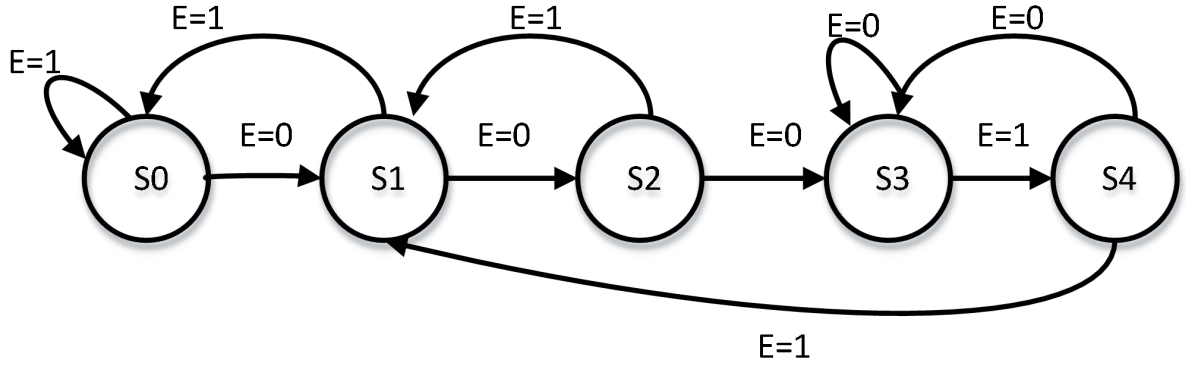
S1：停止生产；L1=L2=L3=0，发电机关闭 F=0；

S2：开启发电机；开启发电机 F=1，生产线关闭 L1=L2=L3=0；

S3：发电生产；发电机开启 F=1，L1=1，L2=L3=0；

S4：停止发电生产：L1=0，L2=L3=0；发电机开启 F=1；

系统复位 rst=1 时进入 S1 关闭所有生产线状态；用三位二进制数对 S0，S1，S2，S3，S4 进行编码分别为 000，001，010，011，100。状态转换图如下：



解法 2：S2 和 S4 状态等价可以合并，因此电路状态减少为四个。

S0：正常生产；根据 G 的不同，开启和关闭不同的生产线，发电机关闭 F=0；

G = 0 时，L1=L2=1，L3=0；

G = 1 时，L1=L2=L3=1;

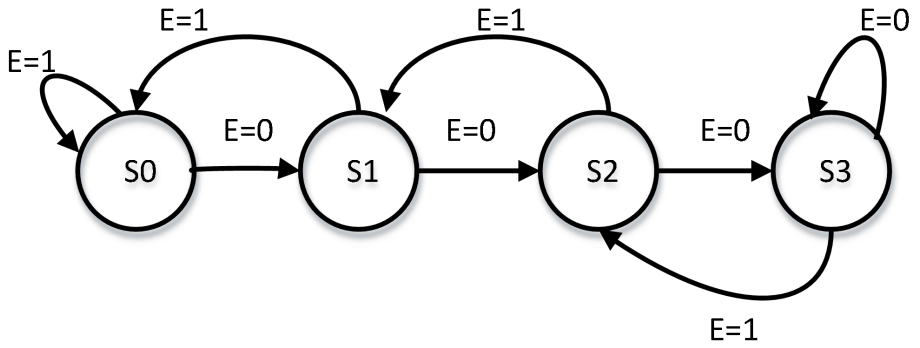
G = 2 时，L1=1，L2=L3=0；

S1：停止生产；L1=L2=L3=0，发电机关闭 F=0；

S2：准备发电生产；开启发电机 F=1，生产线关闭 L1=L2=L3=0；

S3：发电生产；发电机开启 F=1，L1=1，L2=L3=0；

系统复位 rst=1 时进入 S1 关闭所有生产线状态；用三位二进制数对 S0，S1，S2，S3 进行编码分别为 00，01，10，11。状态转换图如下：



(3)解法 1：用三位二进制数对 S0，S1，S2，S3，S4 进行编码分别为 000，001，010，011，100，根据状态转换图和输出信号分析可写出逻辑控制模块如下：

```
module control(rst,clk,G,E,L1,L2,L3,F)
input rst,clk,E;
input[1:0] G;
output L1,L2,L3,F;
wire[2:0] state;
reg[2:0] next;
reg L1,L2,L3,F;
parameter S0 = 3’b000, S1= 3’b001, S2= 3’b010, S3= 3’b011, S4= 3’b100;
// next state logic
always@(*) begin
   case(state)
   S0: next = E ?   S0 : S1;
   S1: next = E ?   S0 : S2;
   S2: next = E ?   S1 : S3;
   S3: next = E ?   S4 : S3;
   S4: next = E ?   S1 : S3;
   default：next = S1；
   endcase
```

```
end
// output logic
always@(*) begin
case(state)
   S0: begin case(G)
         0: {L1,L2,L3,F} = 4'b1100;
         1: {L1,L2,L3,F} = 4'b1110;
         2: {L1,L2,L3,F} = 4'b1000;
         default: {L1,L2,L3,F} = 4'b1100;
       end
   S1: {L1,L2,L3,F} = 4'b0000;
   S2: {L1,L2,L3,F} = 4'b0001;
   S3: {L1,L2,L3,F} = 4'b1001;
   S4: {L1,L2,L3,F} = 4'b0001;
   default：{L1,L2,L3,F}= 4'b0000;
endcase
end
// state register
always@(posedge clk or negedge rst)
if (!rst)    state <= S1;
else    state <= next;
endmodule
```

解法 2：用两位二进制数对 S0，S1，S2，S3 进行编码分别为 00，01，10，11，根据状态转换图和输出信号分析可写出逻辑控制模块如下：

```
module control(rst,clk,G,E,L1,L2,L3,F)
input rst,clk,E;
input[1:0] G;
output L1,L2,L3,F;
wire[1:0] state;
reg[1:0] next;
reg L1,L2,L3,F;
parameter S0 = 2’b00, S1= 2’b01, S2= 2’b10, S3= 2’b11;
// next state logic
always@(*) begin
   case(state)
   S0: next = E ?   S0 : S1;
   S1: next = E ?   S0 : S2;
   S2: next = E ?   S1 : S3;
   S3: next = E ?   S2 : S3;
   default：next = S1；
   endcase
end
// output logic
always@(*) begin
case(state)
   S0: begin case(G)
         0: {L1,L2,L3,F} = 4’b1100;
         1: {L1,L2,L3,F} = 4’b1110;
```

```
            2: {L1,L2,L3,F} = 4'b1000;
            default: {L1,L2,L3,F} = 4'b1100;
          end
    S1: {L1,L2,L3,F} = 4'b0000;
    S2: {L1,L2,L3,F} = 4'b0001;
    S3: {L1,L2,L3,F} = 4'b1001;
    default：{L1,L2,L3,F}= 4'b0000;
endcase
end
// state register
always@(posedge clk or negedge rst)
if (!rst) state <= S1;
else    state <= next;
endmodule
```