# 组合电路模块

## 一、概述

1.组合逻辑电路的特点

功能上：任意时刻的输出仅仅取决于该时刻的输入，而与电路原来的状态无关，即无记忆功能。

电路结构上：只由逻辑门组成，不包含记忆元件，输出和输入之间无反馈。

2.逻辑功能的描述方式

逻辑函数表达式

逻辑电路图

真值表

卡诺图

波形图

语言描述

## 二、组合逻辑电路的分析和设计方法

1.分析

（1）分别用符号标注各级门的输出端。

（2）从输入端到输出端逐级写出输出变量对输入变量的逻辑表达式，最后得到输入变量表示的输出函数表达式。需要时用卡诺图或公式化简法化简逻辑函数成最简形式。

（3）列真值表。

（4）根据真值表或函数表达式确定电路的逻辑功能。

2.设计

（1）进行逻辑抽象

① 分析事件的因果关系，确定输入和输出变量；

② 定义逻辑状态的含意；

③ 根据因果关系列出真值表；

## 三、若干常用中规模组合逻辑电路

1.编码器（Encoder）

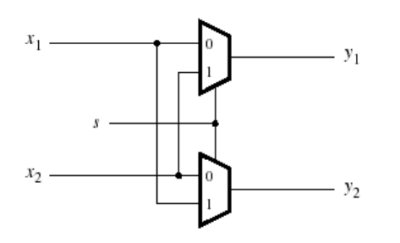
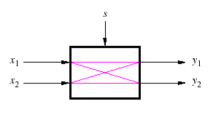
2.优先编码器

3.译码器

4.数据分配器：将一个信号，由选择信号选择输出到不同的输出中

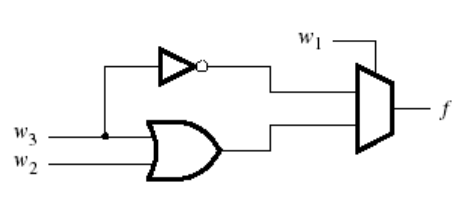
5.数据选择器 二选一、四选一、八选一和十六选一

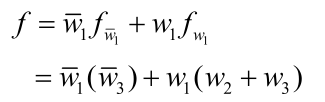
2X2横纵开关



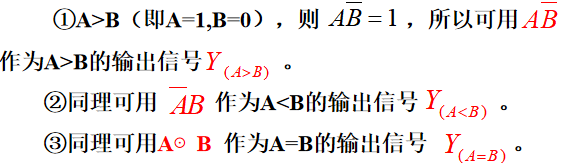
--用香农（Shannon）展开的多路器综合

任何布尔函数都可以表示成如下形式：





6.数值比较器



1. 加法器
   1. 半加器
   2. 全加器

8.多位加法器

原理：依次将低位全加器的进位输出端CO接到高位全加器的进位输入端CI即可构成多位串行加法器。

2）超前进位加法器

## 四、竞争－冒险现象

1 产生的竞争冒险的原因

竞争:当一个逻辑门的两个输入端的信号同时向**相反方向**变化，而变化的时间有差异的现象。

冒险:两个输入端的信号取值的变化方向是相反时，如门电路输出端的逻辑表达式简化成**两个互补信号相乘或者相加**，由竞争而可能产生输出干扰脉冲的现象。

2.检查竞争－冒险现象的方法

可通过逻辑函数式判断组合逻辑电路中是否有竞争－冒险存在。只要输出端的逻辑函数在一定条件下能化简成或的形式，则可判定存在竞争－冒险（此方法适用于任何瞬间只可能有一个输入变量改变状态的情况）。

3.消除方法

1)接入滤波电容

2)引入选通脉冲

3)增加冗余项的方法消除竞争－冒险现象。

# 常见组合逻辑器件代码

## 八三编码器：

**8 – 3 编码器是将2的 n 次方个分离的信息以 n 个二进制代码来表示。**

module bianma8\_3(i, y);

input[7:0] i;

output[2:0] y;

reg[2:0] y;

always @ (i)

begin

case(i[7:0])

8’b00000001: y[2:0] = 3’b000;

8’b00000010: y[2:0] = 3’b001;

8’b00000100: y[2:0] = 3’b010;

8’b00001000: y[2:0] = 3’b011;

8’b00010000: y[2:0] = 3’b100;

8’b00100000: y[2:0] = 3’b101;

8’b01000000: y[2:0] = 3’b110;

8’b10000000: y[2:0] = 3’b111;

default: y[2:0] = 3’b000;

endcase

end

endmodule

## 3 – 8译码器

**将 n 个二进制选择线，最多译码成 2 的 n 次方个分离的信息以来表示。**

①module decoder3\_8

(y, a, g1, g2, g3);

output[7:0] y;

input[2:0] a;

input g1, g2, g3;

reg[7:0] y;

always @ (a, y, g1, g2, g3)

begin

if(g1 == 0) y = 8’b1111\_1111;

else if(g2 == 1) y = 8’b1111\_1111;

else if(g3 == 1) y = 8’b1111\_1111;

else

case(a[2:0])

3’b000: y[7:0] = 8’b1111\_1110;

3’b001: y[7:0] = 8’b1111\_1101;

3’b010: y[7:0] = 8’b1111\_1011;

3’b011: y[7:0] = 8’b1111\_0111;

3’b100: y[7:0] = 8’b1110\_1111;

3’b101: y[7:0] = 8’b1101\_1111;

3’b110: y[7:0] = 8’b1011\_1111;

3’b111: y[7:0] = 8’b0111\_1111;

default: y[7:0] = 8’b1111\_1111;

endcase

end

Endmodule

②module decoder3\_8(y, a, g1, g2, g3);

output[2:0] y;

input[2:0] a;

input g1, g2, g3;

reg[2:0] y;

always @ (a, g1, g2, g3)

begin

if(g1 ==0) y = 8’b1111\_1111;

else if(g2 ==1) y = 8’b1111\_1111;

else if(g3 ==1) y = 8’b1111\_1111;

else

begin

y = 8’b0000\_0001<<a;

y = ~y;

end

end

Endmodule

## 四选一选择器

①case语句（系统级）

module mux41(y, g, d0, d1, d2, d3, a);

output y;

input[1:0] a;

input g;

input d0, d1, d2, d3;

reg y;

always @ (d0, d1, d2, d3, a, g)

begin

if(g ==0) y = 0;

else begin

case(a[1:0])

2’b00: y = d0;

2’b01: y = d1;

2’b10: y = d2;

2’b11: y = d3;

Endcade

end

end

Endmodule

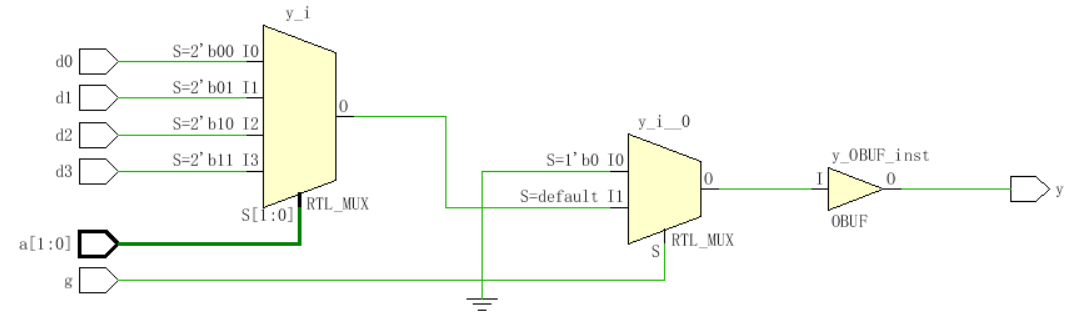


图 1系统级

②条件运算符（RTL级）

module mux4\_1a(y, g, d0, d1, d2, d3, a);

output y;

input[1:0] a;

input g;

input d0, d1, d2, d3;

reg y;

assign y = g?(a[1]?(a[0]?d3:d2) :(a[0]?d1:d0)):0;

endmodule

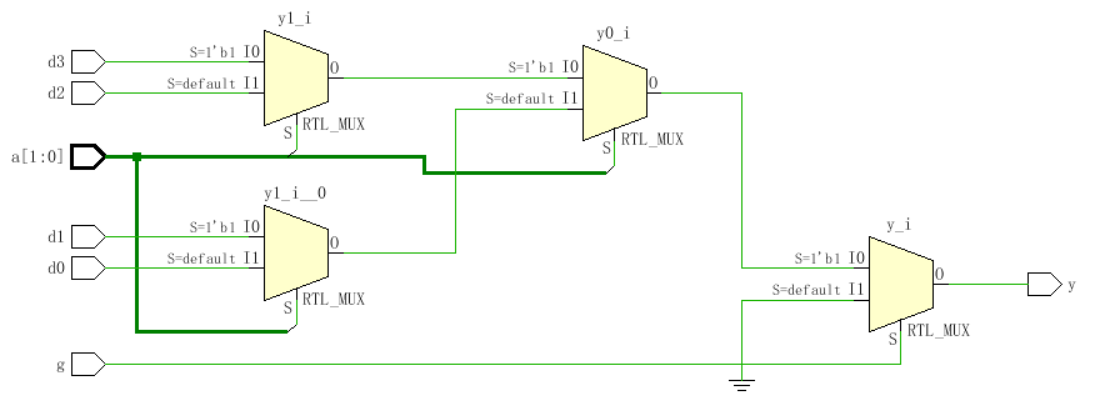


图 2 RTL级

③门元件（门级）

`timescale 1ns / 1ps

module mux4\_1\_door(y, g, d0, d1, d2, d3, a);

output y;

input[1:0] a;

input g;

input d0, d1, d2, d3;

wire y;

wire nota1, nota0, x1, x2, x3, x4;

not (nota0, a[0]),

(nota1, a[1]);

and (x1, d0, nota1, nota0),

(x2, d1,nota1, a[0]),

(x3, d2, a[1], nota0),

(x4, d3, a[1], a[0]);

or (y, x1, x2, x3, x4);

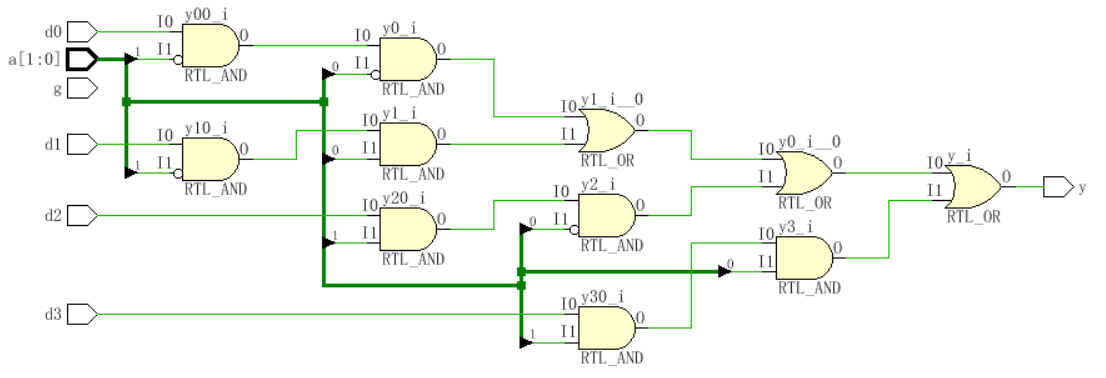
endmodule

图 3门级

④逻辑表达式（算法级）

module mux4\_1a(y, g, d0, d1, d2, d3, a);

output y;

input[1:0] a;

input g;

input d0, d1, d2, d3;

reg y;

assign y = ((d0&~a[1]&~a[0])|(d1&~a[1]&a[0])

|(d2&a[1]&~a[0])|(d3&a[1]&a[0]))&g;

endmodule

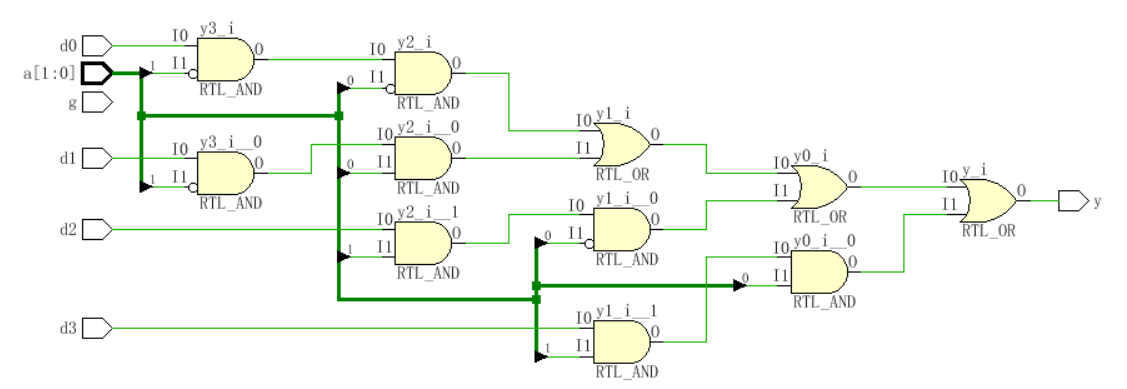


图 4算法级

## 数据分配器

module dmux (y0, y1, y2, y3, din, sel);

output y0, y1, y2, y3;

input[1:0] sel;

input din;

reg y0, y1, y2, y3;

always @ (din, sel)

begin

y0 = 0; y1 = 0; y2 = 0; y3 = 0;

case(sel[1:0])

2’b00: y0 = din;

2’b01: y1 = din;

2’b10: y2 = din;

2’b11: y3 = din;

default:;

endcase

end

Endmodule

## 数值比较器

module comparator (y1, y2, y3, a, b);

output y1, y2, y3;

input[3:0] a, b;

reg y0, y1, y2, y3;

always @ (a, b)

begin

if(a > b) begin

y1 = 1; y2 = 0; y3 = 0;

end

else if(a == b) begin

y1 = 0; y2 = 1; y3 = 0;

end

else begin

y1 = 0; y2 = 0; y3 = 1;

end

end

endmodule

## 半加法器

//调用门元件（门级）综合电路也是基于门元件

*module half\_add (sum, cout, a, b);*

*output sum, cout;*

*input a, b;*

*and (cout, a, b);*

*xor (sum, a, b);*

*Endmodule*

//数据流方式（行为描述之算法级🡪综合电路也是基于门元件）

*module half\_add (sum, cout, a, b);*

*output sum, cout;*

*input a, b;*

*assign cout = a & b;*

*assign sum = a ^ b;*

*Endmodule*

//行为描述之RTL级（系统运算符，综合电路基于系统ALU计算元件）

*module half\_add (sum, cout, a, b);*

*output sum, cout;*

*input a, b;*

*wire sum,cout;*

*assign* ***{****cout, sum} = a + b;*

*Endmodule*

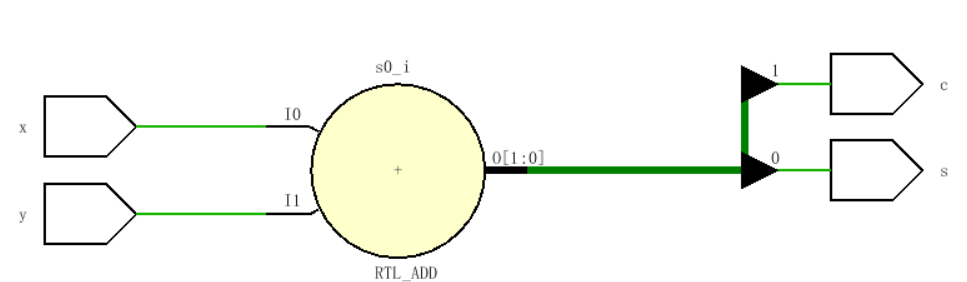


图 5RTL级（系统运算符）

//行为描述之系统级（case ,真值表）🡪综合电路基于多路选择器

*module half\_add (sum, cout, a, b);*

*output sum, cout;*

*input a, b;*

*reg[2:0] sum,cout;*

*always @ (a, b)*

*begin*

*case[{a, b}]*

*2’b00: begin cout = 0, sum = 0;end*

*2’b01: begin cout = 0, sum = 1;end*

*2’b10: begin cout = 0, sum = 1;end*

*2’b11: begin cout = 1, sum = 0;end*

*endcase*

*end*

*endmodule*

## 全加器

//一位全加器的行为描述

*module add1 (cin, sum, cout, a, b);*

*output sum, cout;*

*input a, b, cin;*

*reg sum,cout;*

*always @ (a, b, cin)*

*begin*

*sum = (a ^ b) ^ c;*

*cout = (a & b)|(a & cin)|(cin & b);*

*//行为描述（另一种方法）*

*// {cout, sum} = a + b + cin;*

*end*

*Endmodule*

## 4位全加器

//4位全加器的行为描述

module add4 (cin, sum, cout, a, b);

output[3:0] sum;

output cout;

input[3:0] a, b;

input cin;

reg cout;

reg[3:0] sum;

always @ (\*)

begin

{cout, sum} = a + b + cin;

end

endmodule

## 半减器

//行为描述

module half\_sub (dout, cout, a, b);

output dout, cout;

input a, b;

reg dout, cout;

always @ (a, b)

begin

**dout = a ^ b;**

**cout = (~a) & b;**

//行为描述（另一种方法）

// {cout, dout} = a - b;

end

endmodule

## 全减器

//行为描述，4位全减器

*module sub4 (cin, dout, cout, a, b);*

*output[3:0] dout;*

*output cout;*

*input[3:0] a, b;*

*input cin;*

*reg[3;0] dout,*

*reg cout;*

*always @ (a, b)*

*begin*

*{carryout, subout} = a – b – cin;*

*end*

*endmodule*

## 4位全减器

//行为描述，4位全减器

*module sub4 (cin, dout, cout, a, b);*

*output[3:0] dout;*

*output cout;*

*input[3:0] a, b;*

*input cin;*

*reg[3;0] dout,*

*reg cout;*

*always @ (a, b)*

*begin*

*{cout, dout} = a – b – cin;*

*end*

*endmodule*