Part Ⅳ Verilog 及其编程

Lecture 16 Verilog 组合逻辑设计

一、Verilog 逻辑设计的描述方法

1、门级结构描述

门级结构描述就是将逻辑电路图用 Verilog HDL 语言中内置的基本门元件来描述逻辑图中的元件及各元件之间的连接关系。

Verilog HDL 中内置了 12 个基本门类型， 如表 4.3.1 所示。

表4.3.1 Verilog HDL语言内置的12个基本门元件关键字

门类型	功 能 说 明	门类型	功 能 说 明
and	多输入端的与门	nand	多输入端的与非门
or	多输入端的或门	nor	多输入端的或非门
xor	多输入端的异或门	xnor	多输入端的异或非门
buf	多输入端的缓冲器	not	多输入端的反向器
bufif1	控制信号高电平的三态缓冲器	notif1	控制信号高电平的三态反向器
bufif0	控制信号低电平的三态缓冲器	notif0	控制信号低电平的三态反向器

（1）多输入门

and、nand、or、nor、xor 和 xnor 是具有多个输入端的逻辑门，它们的共同特点是：只允许有一个输出，但可以有多个输入。

调用形式为：and a1(out,in1,in2,in3, …);

调用这几个门级元件时，括号中需要列出输入输出变量，在括号左边的第一个变量必须是输

出变量。

（2）多输出门

buf、not 是具有多个输出端的逻辑门，它们的共同特点是：允许有多个输出，但只有一个输入。

调用形式为：buf b1(out1,out2,out3,…,in);

这两类输出门在调用时，要求括号右边的第

一个变量必须是输入变量。

（3）三态门

bufif1、bufif0、notif1、notif0 是三态门元件模型，这些门有一个输出、一个数据输入和一个输入控制。如果输入控制信号无效，则三态门的输出为高阻态 z 。

调用形式为：bufif1 c1(out,int,ctrl);

在调用以上三类门元件时，需要注意的是，各门级元件的输出、输入必须为 wire 型的变量。

2、行为描述

在进行复杂数字系统设计时，如果只**使用门级结构描述电路**，那么设计起来将**非常麻烦**。相比之下，**使用行为描述来说明数字电路**的行为或功能则**简单得多**。

行为描述就是对逻辑电路的功能和算法进行描述，在 Verilog HDL 模块中常用 **assign 语句**、**always 语句**生成所需的逻辑电路。

（1）用 assign 语句

assign 只能对 wire 型变量进行赋值，例如：

```
assign f = x>y? a :b;
```

（2）用 always 语句

always 既可以用于描述组合逻辑电路，也可以用于描述时序逻辑电路。

在描述组合逻辑电路时，always 在使用上有以下几个特点或规则：

①在敏感列表中使用电平敏感事件，不要使用边沿敏感事件。

②为变量赋值时使用阻塞赋值，不要使用非

阻塞赋值。

③不要在一个以上的 always 块中为同一个变量赋值。

二、Verilog HDL 组合逻辑设计的基本思路

根据真值表进行描述；

或者直接根据表达式进行描述；

或者化简后再根据简化表达式进行描述。

【例 1】在举重比赛中，有两名副裁判，一名主裁判。当两名以上裁判（必须包括主裁判在内）认为运动员上举杠铃合格，按动电钮，裁决合格信号灯亮，则该运动员成绩有效。试设计该电路。

【分析】真值表

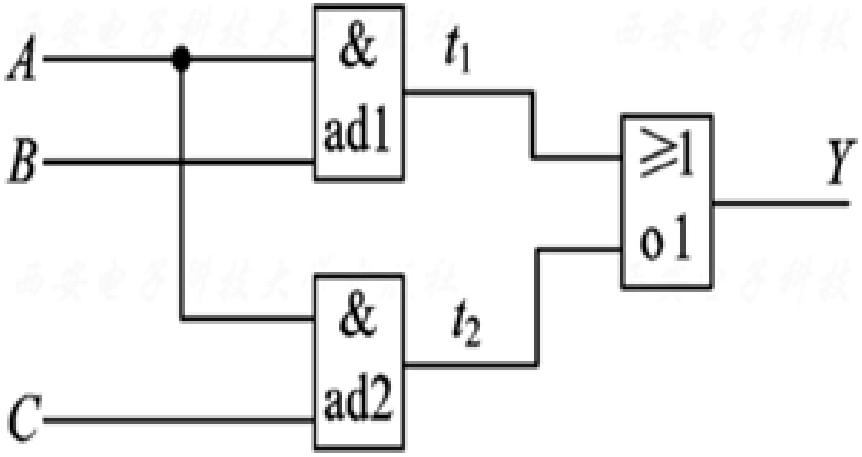
ABC	Y	ABC	Y
000	0	100	0
001	0	101	1
010	0	110	1
011	0	111	1

方法 1：结构描述

```
module ABC(A,B,C,Y);
    input A,B,C;
    output Y;
    wire t1,t2;
    and ad1(t1,A,B),ad2(t2,A,C);
    or o1(Y,t1,t2);
endmodule
```

ABC	Y	ABC	Y
000	0	100	0
001	0	101	1
010	0	110	1
011	0	111	1

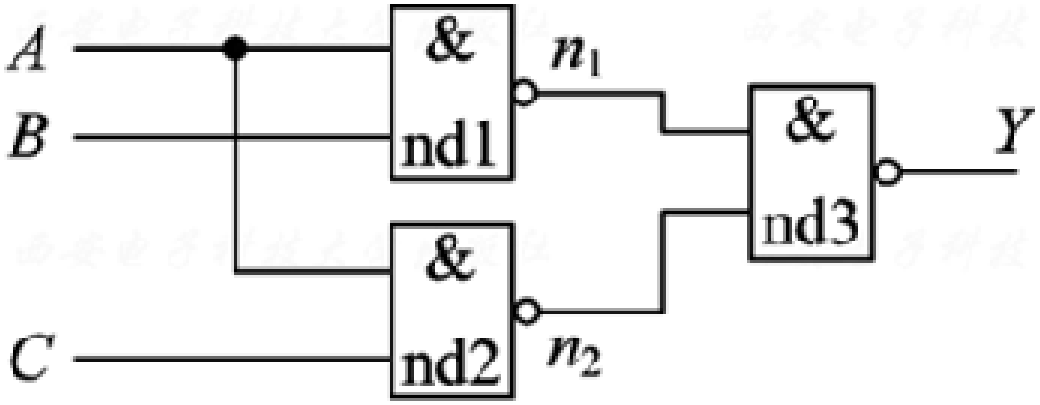
①采用与门、或门：$Y = AB + AC$



方法 1：结构描述

```
module ABC(A,B,C,Y);
    input A,B,C;
    output Y;
    wire n1,n2;
    nand nd1(n1,A,B),nd2(n2,A,C),nd3(Y,n1,n2);
endmodule
```

②采用与非门：$Y = \overline{\overline{AB} \cdot \overline{AC}}$



方法 2：行为描述

```
module ABC(A,B,C,Y);
    input A,B,C;
    output reg Y;
    always @(A or B or C)
        if (A&B|A&C|A&B&C) Y=1;
        else Y=0;
endmodule
```

ABC	Y	ABC	Y
000	0	100	0
001	0	101	1
010	0	110	1
011	0	111	1

在 always 触发事件列表中使用（A，B，C）表示只要三个输入信号电平发生变化时，就执行该模块。

需要注意的是，在组合电路中不要使用信号边沿作为触发条件。

在 always 块中也可以使用 case 语句实现该功能：

```
module ABC(A,B,C,Y);

    input A,B,C;

    output reg Y;

    always @(A or B or C)

        case ({A,B,C})

            3'B101:Y=1;

            3'B110:Y=1;

            3'B111:Y=1;

            default:Y=0;

        endcase

endmodule
```

ABC	Y	ABC	Y
000	0	100	0
001	0	101	1
010	0	110	1
011	0	111	1

方法 3：数据流描述

```
module ABC(A,B,C,Y);
    input A,B,C;
    output Y;
    assign Y=A&B|A&C;
enmodule
```

【例 2】设计一个可以实现 4 位格雷码和 4 位二进制编码的相互转换电路，有一个控制端 *S*：当 *S*=1 时，可以将输入的 4 位格雷码转换成 4 位二进制编码；当 *S*=0 时，实现将输入的 4 位二进制编码转换成 4 位格雷码。

【分析】

$$\begin{cases} G_3 = B_3 \\ G_2 = B_3 \oplus B_2 \\ G_1 = (SG_2 + \overline{S}B_2) \oplus B_1 \\ G_0 = (SG_1 + \overline{S}B_1) \oplus B_0 \end{cases}$$

表4.3.3 例4.3.2的真值表

S	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0	S	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1	1	1	0	0	1	1	0	0	1	0
0	0	0	1	1	0	0	1	0	1	0	0	1	0	0	0	1	1
0	0	1	0	0	0	1	1	0	1	0	1	1	0	0	1	0	0
0	0	1	0	1	0	1	1	1	1	0	1	1	1	0	1	0	1
0	0	1	1	0	0	1	0	1	1	0	1	0	1	0	1	1	0
0	0	1	1	1	0	1	0	0	1	0	1	0	0	0	1	1	1
0	1	0	0	0	1	1	0	0	1	1	1	0	0	1	0	0	0
0	1	0	0	1	1	1	0	1	1	1	1	0	1	1	0	0	1
0	1	0	1	0	1	1	1	1	1	1	1	1	1	1	0	1	0
0	1	0	1	1	1	1	1	0	1	1	1	1	0	1	0	1	1
0	1	1	0	0	1	0	1	0	1	1	0	1	0	1	1	0	0
0	1	1	0	1	1	0	1	1	1	1	0	1	1	1	1	0	1
0	1	1	1	0	1	0	0	1	1	1	0	0	1	1	1	1	0
0	1	1	1	1	1	0	0	0	1	1	0	0	0	1	1	1	1

方法 1：结构描述（太啰嗦，略！）

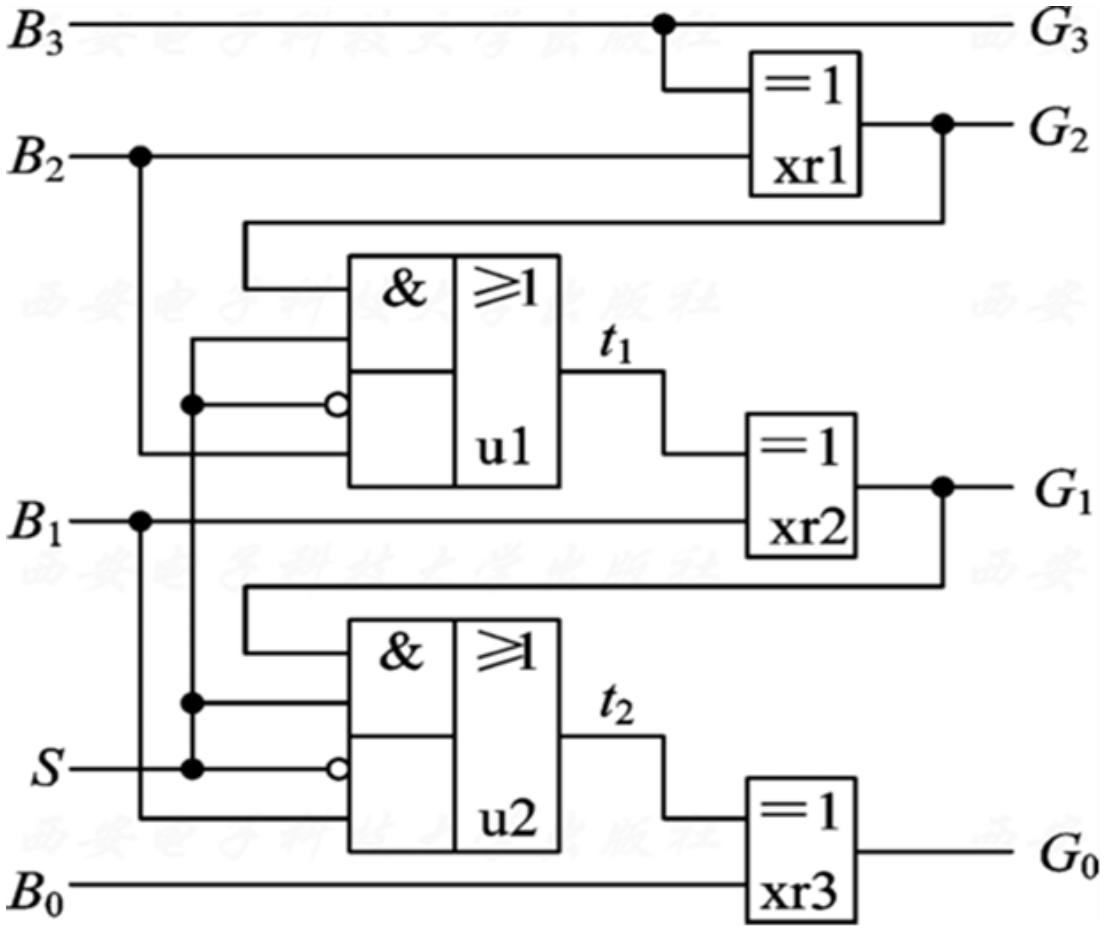


图4.3.11 例4.3.2的逻辑电路图

方法 2：行为描述

```
module ABC(B,S,G);
    input [3:0] B;
    input S;
    output reg [3:0] G;
    always @(S or B)
        if (S) //实现格雷码到二进制码的转换
            case (B)
```

4’b0000:G=4’b0000; 4’b0001:G=4’b0001;

4’b0011:G=4’b0010; 4’b0010:G=4’b0011;

4’b0110:G=4’b0100; 4’b0111:G=4’b0101;

4’b0101:G=4’b0110; 4’b0100:G=4’b0111;

4’b1100:G=4’b1000; 4’b1101:G=4’b1001;

4’b1111:G=4’b1010; 4’b1110:G=4’b1011;

4’b1010:G=4’b1100; 4’b1011:G=4’b1101;

4’b1001:G=4’b1110; 4’b1000:G=4’b1111;

```
    endcase
else //实现二进制码到格雷码的转换
    case (B)
        4'b0000:G=4'b0000; 4'b0001:G=4'b0001;
        4'b0010:G=4'b0011; 4'b0011:G=4'b0010;
        4'b0100:G=4'b0110; 4'b0101:G=4'b0111;
        4'b0110:G=4'b0101; 4'b0111:G=4'b0100;
        4'b1000:G=4'b1100; 4'b1001:G=4'b1101;
```

```
            4'b1010:G=4'b1111; 4'b1011:G=4'b1110;

            4'b1100:G=4'b1010; 4'b1101:G=4'b1011;

            4'b1110:G=4'b1001; 4'b1111:G=4'b1000;

        endcase

endmodule
```

三、常用组合逻辑电路的设计

1、加法器

（1）1 位全加器

```
module Adder (A,B,Cin,S,Cout);
    input A,B,Cin;
    output S,Cout;
    assign {Cout,S}=A+B+Cin;
endmodule
```

（2）4 位全加器

```
module Adder (A,B,Cin,S,Cout);
    input[3:0] A,B;
    input Cin;
    output[3:0] S;
    output Cout;
    assign {Cout,S}=A+B+Cin;
endmodule
```

2、编码器

（1）基本功能编码器

```
module encoder(IN,Y);
    input [7:0] IN;
    output reg [2:0] Y;
    always @(IN)
        case (IN)
            8'b01111111:Y=3'b000;
            8'b10111111:Y=3'b001;
            8'b11011111:Y=3'b010;
            8'b11101111:Y=3'b011;
            8'b11110111:Y=3'b100;
            8'b11111011:Y=3'b101;
            8'b11111101:Y=3'b110;
            8'b11111110:Y=3'b111;
            default: Y=3'bxxx;
        endcase
endmodule
```

（2）74LS148 优先编码器

```
module 74LS148(ST,IN,Y,YEX,YS);
    input ST;
    input [7:0] IN;
    output reg [2:0] Y;
    output reg YEX,YS;
    always @(ST,IN)
```

```
if (!ST) begin
    YEX=0; YS=1;
    if (IN[7]==0) Y=3'h0;
    else if (IN[6]==0) Y=3'h1;
    else if (IN[5]==0) Y=3'h2;
    else if (IN[4]==0) Y=3'h3;
    else if (IN[3]==0) Y=3'h4;
```

```
        else if (IN[2]==0)oY=3'h5;

        else if (IN[1]==0) Y=3'h6;

        else if (IN[0]==0) Y=3'h7;

        else begin Y=3'h7; YEX=1; YS=0; end

    end

    else begin Y=3'h7; YEX=1; YS=1; end

endmodule
```

3、译码器

（1）二进制译码器

```
module Decoder(STA,STB,STC,A,Y);
    input STA,STB,STC;
    input [2:0] A;
    output [7:0] Y;
    always @(STA,STB,STC,A)
```

```
if (STA&&!(STB||STC))
    case (A)
        3'b000:Y = 8'b11111110;
        3'b001:Y =8'b11111101;
        3'b010:Y =8'b11111011;
        3'b011:Y =8'b11110111;
        3'b100:Y =8'b11101111;
```

```
            3'b101:Y =8'b11011111;

            3'b110:Y =8'b10111111;

            3'b111:Y =8'b01111111;

        endcase

    else Y=8'hff;

endmodule
```

（2）二—十进制译码器

```
module DecoderD(A,Y);
    input [3:0] A;
    output reg [9:0] Y;
    always @(A)
        case (A)
            4’b0000:Y=10’h001;
```

4’b0001:Y=10’h002;

4’b0010:Y=10’h004;

4’b0011:Y=10’h008;

4’b0100:Y=10’h010;

4’b0101:Y=10’h020;

4’b0110:Y=10’h040;

4’b0111:Y=10’h080;

```
            4'b1000:Y=10'h100;
            4'b1001:Y=10'h200;
              default:Y=10'h000;
        endcase
endmodule
```

（3）七段数字译码器

```
module Decoder (flag,A,Y);
```

```
input flag;

input [3:0] A;

output reg [6:0] Y;

always @(flag,A)

    begin

        case (A)

            4'b0000:Y=7'h3f;
```

4’b0001:Y=7’h06;

4’b0010:Y=7’h5b;

4’b0011:Y=7’h4f;

4’b0100:Y=7’h66;

4’b0101:Y=7’h6d;

4’b0110:Y=7’h7d;

4’b0111:Y=7’h27;

4’b1000:Y=7’h7f;

4’b1001:Y=7’h6f;

4’b1010:Y=7’h77;

4’b1011:Y=7’h7c;

4’b1100:Y=7’h58;

4’b1101:Y=7’h5e;

4’b1110:Y=7’h79;

```
            4'b1111:Y=7'h71;

        Endcase

        If (!flag) Y=~Y;

    End

endmodule
```

4、数据选择器和数据分配器

（1）数据选择器

```
module Mux4(D,S,Q);
    input [3:0] D;
    input [1:0] S;
    output reg Q;
    always @(D,S)
        case (S)
            2'b00:Q=D[0];
```

```
        2'b01:Q=D[1];

        2'b10:Q=D[2]

        2'b11:Q=D[3]

    Endcase

endmodule
```

（2）数据分配器

```
module Demux(EN,S,D,Y);
```

```
input EN;

input D;

input [2:0] S;

output reg [7:0] Y;

always @(D,EN,S)

    begin

        Y=8'b11111111;
```

```
if (EN)
    case (S)
        3'b000:Y[0]=D;
        3'b001:Y[1]=D;
        3'b010:Y[2]=D;
        3'b011:Y[3]=D;
        3'b100:Y[4]=D;
```

```
            3'b101:Y[5]=D;

            3'b110:Y[6]=D;

            3'b111:Y[7]=D;

        Endcase

    End

endmodule
```

5、数值比较器

```
module Comparen(A,B,AGB,ALB,AEB);
    input [n-1:0] A,B;
    output reg AGB,ALB,AEB;
    parameter n=4;
    always @(A,B)
        begin
            AGB=0;
```

```
        ALB=0;

        AEB=0;

        If (A>B) AGB=1;

        else if(A==B) AEB=1;

        else ALB=1;

    end

endmodule
```

6、奇偶产生/校验电路

```
module Check(data,even,odd,Fod,Fev);

input [7:0] data;

input even,odd;

output Fod,Fev;

odd_even_check #(8) u1(data,even,odd,Fod,Fev);

endmodule
```

```
module odd_even_check(data,even,odd,Fod,Fev);

input [n-1:0] data;

input even,odd;

output reg Fod,Fev;

reg temp;

parameter n=8;

always @(data,even,odd)
```

```
begin
    temp=^data;
    case ({even,odd})
        2'b00:{Fev,Fod}=2'b11;
        2'b01:if(temp) {Fev,Fod}=2'b10;
                    Else {Fev,Fod}=2'b01;
        2'b10:if(temp) {Fev,Fod}=2'b01;
```

```
                Else {Fev,Fod}=2'b10;

            2'b11:{Fev,Fod}=2'b00;

        Endcase

    End

Endmodule
```