


第2章

组合电路Verilog设计

2.1 半加器电路的Verilog描述

2.1.1 半加器的数据流建模描述方式

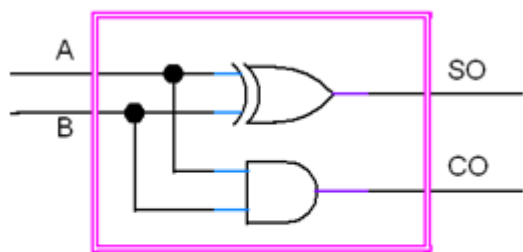


图 2-1 半加器的电路结构

A	B	SO	CO
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

图 2-2 半加器的真值表

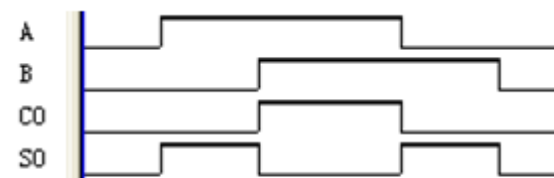


图 2-3 半加器的仿真功能波形图

$$SO = A \oplus B ; \quad CO = A \cdot B$$



2.1 半加器电路的Verilog描述

2.1.1 半加器的数据流建模描述方式

【例 2-1】

```
module h_adder (A,B,S0,C0) ;  
    input A,B;  
    output S0,C0;  
    assign S0 = A ^ B;  
    assign C0 = A & B;  
endmodule
```



2.1 半加器电路的Verilog描述

2.1.1 半加器的数据流建模描述方式

1. 模块语句及其表达方式

```
module 模块名 (模块端口名表) ;  
    模块端口和模块功能描述 .  
endmodule
```

2. 端口语句、端口信号名和端口模式

```
input  端口名 1, 端口名 2, ... ;  
output 端口名 1, 端口名 2, ... ;  
inout  端口名 1, 端口名 2, ... ;  
input [msb : lsb] 端口名 1, 端口名 2, ... ;
```

output [3:0] C,D;

- (1) input : 输入端口。
- (2) output : 输出端口。
- (3) inout : 双向端口。

```
module h_adder (input A, input B, output SO, output CO) ;
```



2.1 半加器电路的Verilog描述

2.1.1 半加器的数据流建模描述方式

1. 模块语句及其表达方式

```
module 模块名 (模块端口名表) ;  
    模块端口和模块功能描述 .  
endmodule
```

2. 端口语句、端口信号名和端口模式

```
input 端口名 1, 端口名 2, ... ;  
output 端口名 1, 端口名 2, ... ;  
inout 端口名 1, 端口名 2, ... ;  
input [msb : lsb] 端口名 1, 端口名 2, ... ;
```



2.1 半加器电路的Verilog描述

2.1.1 半加器的数据流建模描述方式

3. 逻辑操作符

逻辑与“&”和逻辑异或“^”

4. 连续赋值语句

```
assign 目标变量名 = 驱动表达式;  
assign [延时] 目标变量名 = 驱动表达式;  
assign #6 R1 = A & B;  
`timescale 10ns/100ps
```

5. 关键字

6. 标识符



2.1 半加器电路的Verilog描述

2.1.2 半加器的门级原语和UDP结构建模描述方式

【例 2-2】

```
primitive XOR2 (DOUT, X1, X2) ;
  input X1, X2;  output DOUT;
  table // X1  X2  : DOUT
        0  0  :  0;
        0  1  :  1;
        1  0  :  1;
        1  1  :  0;

  endtable
endprimitive
```

【例 2-3】

```
module H_ADDER (A, B, S0, C0) ;
  input A, B;
  output S0, C0;
  XOR2 U1 (S0, A, B) ; // 调用元件 XOR2
  and U2 (C0, A, B) ; // 调用元件 and
endmodule
```



2.1 半加器电路的Verilog描述

2.1.2 半加器的门级原语和UDP结构建模描述方式

1. 库元件及其调用
 2. 用户自定义原语
 3. 注释符号
 4. 规范的程序书写格式
 5. 文件取名和存盘
-

2.2 多路选择器不同形式的 Verilog 描述

2.2.1 4选1多路选择器及其顺序语句表述方式

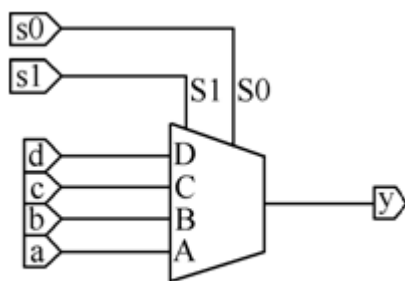


图 2-4 4 选 1 多路选择器

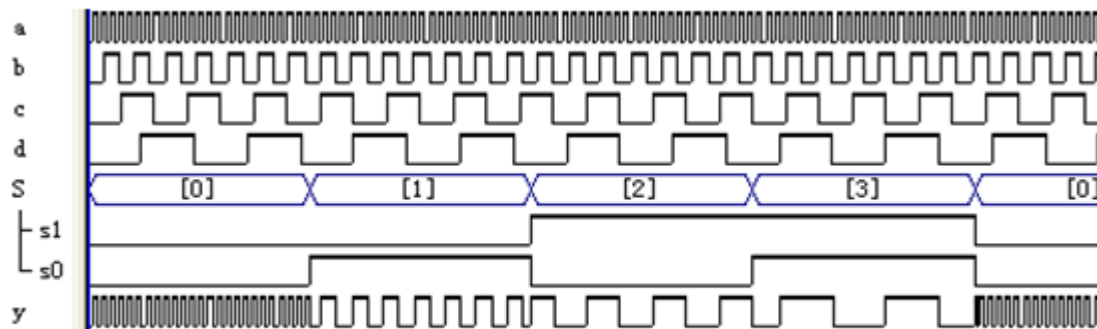


图 2-5 4 选 1 多路选择器 MUX41a 的时序波形

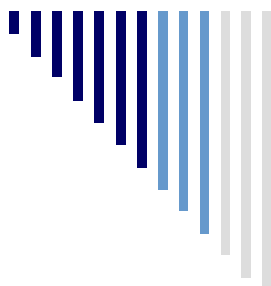


2.2 多路选择器不同形式的 Verilog描述

2.2.1 4选1多路选择器及其顺序语句表述方式

【例 2-4】

```
module MUX41a (a,b,c,d,s1,s0,y) ;
    input a,b,c,d,s1,s0 ;    output y ;
    reg y ;                  //定义输出端口信号 y 为寄存器型变量
    always @ (a or b or c or d or s1 or s0)
        begin : MUX41
            case ({s1,s0})
                2'b00 : y <= a;
                2'b01 : y <= b;
                2'b10 : y <= c;
                2'b11 : y <= d;
                default : y <= a;
            endcase
        end
endmodule
```



2.2 多路选择器不同形式的Verilog描述

2.2.1 4选1多路选择器及其顺序语句表述方式

1. reg型变量定义

```
reg 变量名 1, 变量名 2, . . . ;  
reg [msb:lsb] 变量名 1, 变量名 2, . . . ;
```

```
module seg_7 (input [3:0] num, input en, output reg [6:0] seg );
```

2. 过程语句

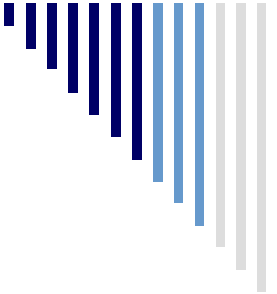
`always @` (敏感信号及敏感信号列表或表达式)

包括块语句的各类顺序语句

3. 块语句begin _end

begin_end 块语句的一般格式如下:

```
begin [: 块名]  
    语句 1; 语句 2; . . . ; 语句 n;  
end
```



2.2 多路选择器不同形式的 Verilog描述

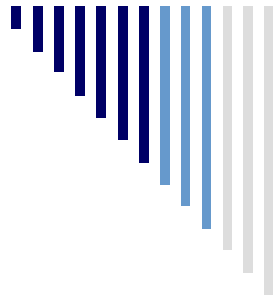
2.2.1 4选1多路选择器及其顺序语句表述方式

4. case条件语句

```
case (表达式)
    取值 1 : begin 语句 1; 语句 2; ... ; 语句 n;    end
    取值 2 : begin 语句 n+1; 语句 n+2; ... 语句 n+m; end
    ...
    default : begin 语句 n+m+1; ... ; end
endcase
```

5. Verilog的4种逻辑状态

1、0、z 和 x



2.2 多路选择器不同形式的Verilog描述

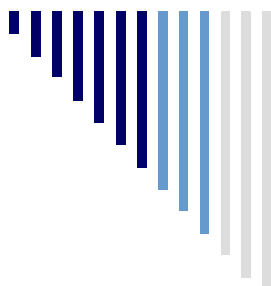
2.2.1 4选1多路选择器及其顺序语句表述方式

6. 并位操作运算符

$$\{a1, b1, 4\{a2, b2\}\} = \{a1, b1, \{a2, b2\}, \{a2, b2\}, \{a2, b2\}, \{a2, b2\}\} = \{a1, b1, a2, b2, a2, b2, a2, b2, a2, b2\}$$

7. Verilog的数字表达形式

<位宽> '<进制> <数字>



2.2 多路选择器不同形式的Verilog描述

2.2.2 4选1多路选择器及其并行语句表述方式

【例 2-5】

```
module MUX41a (a,b,c,d,s1,s0,y);  
    input a,b,c,d,s1,s0;        output y;  
    wire [1:0] SEL;             // 定义2元素位矢量SEL为网线型变量wire  
    wire AT, BT, CT, DT;       //定义中间变量, 以作连线或信号节点  
    assign SEL = {s1,s0};       //对s1,s0进行并位操作, 即SEL[1]=s1; SEL[0]=s0  
    assign AT = (SEL==2'D0);    //assign语句中的输出变量必须是网线型变量  
    assign BT = (SEL==2'D1);  
    assign CT = (SEL==2'D2);  
    assign DT = (SEL==2'D3);  
    assign y = (a & AT) | (b & BT) | (c & CT) | (d & DT); //4个逻辑信号相或  
endmodule
```

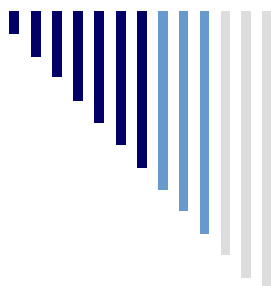
2.2 多路选择器不同形式的 Verilog描述

2.2.2 4选1多路选择器及其并行语句表述方式

1. 按位逻辑操作符

表 2-1 逻辑操作符

逻辑操作符	逻辑功能	A,B 逻辑操作结果	C,D 逻辑操作结果	C,E 逻辑操作结果
~	逻辑取反	$\sim A = 1'b1$	$\sim C = 4'b0011$	$\sim E = 6'b101001$
	逻辑或	$A B = 1'b1$	$C D = 4'b1111$	$C E = 6'b011110$
&	逻辑与	$A \& B = 1'b0$	$C \& D = 4'b1000$	$C \& E = 6'b000100$
^	逻辑异或	$A \wedge B = 1'b1$	$C \wedge D = 4'b0111$	$C \wedge E = 6'b011010$
~^ 或 ~^	逻辑同或	$A \sim \wedge B = 1'b0$	$C \sim \wedge D = 4'b1000$	$C \sim \wedge E = 6'b100101$
设: $A=1'b0$; $B=1'b1$; $C[3:0]=4'b1100$; $D[3:0]=4'b1011$; $E[5:0]=6'b010110$;				



2.2 多路选择器不同形式的 Verilog描述

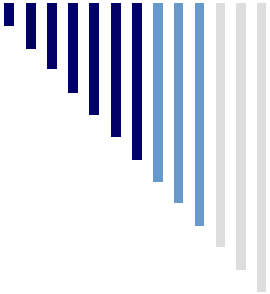
2.2.2 4选1多路选择器及其并行语句表述方式

2. 等式操作符

表 2-2 等式操作符

等式操作符	含义	等式操作示例
==	等于	(3==4) = 0; (A==4'b1011) = 1; (B==4'b1011) = 0;
!=	不等于	(D!=C) = 1; (3!=4) = 1;
===	全等	(D===C) = 1 ; (E===4'b0x10) = 0 ;
!==	不全等	(E!==4'b0x10) = 1;

设: A=5'b01011; B=4'b0010; C=4'b0z10; D=4'b0z10; E=3'bx10



2.2 多路选择器不同形式的 Verilog描述

2.2.2 4选1多路选择器及其并行语句表述方式

3. wire定义网线型变量

```
wire 变量名 1, 变量名 2, . . . ;  
wire [msb:lsb] 变量名 1, 变量名 2, . . . ;
```

```
wire a1, a2;  
assign Y = a1 ^ a2;
```

2.2 多路选择器不同形式的 Verilog描述

2.2.3 4选1多路选择器及其条件操作语句表述方式

【例 2-6】

```
module MUX41a (A,B,C,D,S1,S0,Y);  
    input A,B,C,D,S1,S0;  
    output Y;  
    //若S0=1, 则AT=D; 若S0=0, 则AT=C  
    wire AT = S0 ? D : C ;  
    //若S0=1, 则BT=B; 若S0=0, 则BT=A  
    wire BT = S0 ? B : A ;  
    //若S1=1, 则Y=AT; 若S1=0, 则Y=BT  
    wire Y = (S1 ? AT : BT);  
endmodule
```

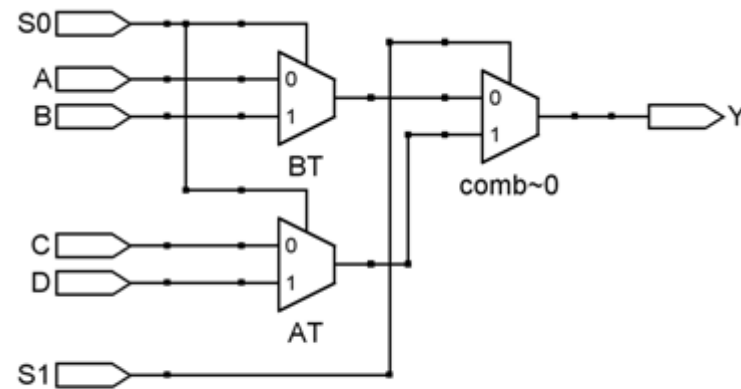
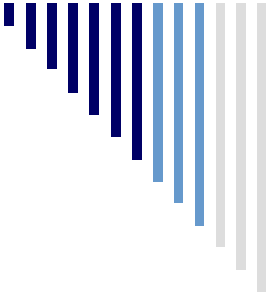


图 2-6 例 2-6 的 RTL 图

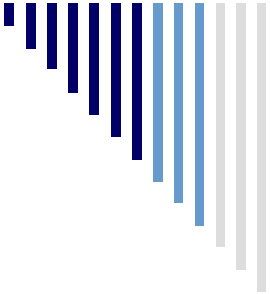


2.2 多路选择器不同形式的 Verilog描述

2.2.4 4选1多路选择器及其条件语句表述方式

【例 2-7】

```
module MUX41a (A,B,C,D,S1,S0,Y);
    input A,B,C,D,S1,S0;    output Y;
    reg [1:0] SEL;    reg Y;
    always @(A,B,C,D,SEL)
        begin
            SEL = {S1,S0};    //块语句起始
            if (SEL==0) Y = A;    //把 S1,S0 并位为 2 元素矢量变量 SEL[1:0]
            //当 SEL==0 成立, 即 (SEL==0)=1 时, Y=A;
        else if (SEL==1) Y = B;    //当 (SEL==1) 为真, 则 Y=B;
        else if (SEL==2) Y = C;    //当 (SEL==2) 为真, 则 Y=C;
        else Y = D;    //当 SEL==3, 即 SEL==2'b11 时, Y = D;
        end
    //块语句结束
endmodule
```



2.2 多路选择器不同形式的 Verilog描述

2.2.4 4选1多路选择器及其条件语句表述方式

1. if_else条件语句

```
            if (S)  Y = A;  else  Y = B;  
  
    if (S)  Y=A;  else  
    begin Y=B;  Z=C;  Q=1'b0;  end
```

2. 过程赋值语句

- (1) 阻塞式赋值。
- (2) 非阻塞式赋值。

3. 数据类型表示方式

2.2 多路选择器不同形式的Verilog描述

2.2.5 4选1多路选择器及其利用UDP元件的结构表述方式

【例 2-8】

```
primitive
MUX41_UDP (Y,D3,D2,D1,D0,S1,S0) ;
input D3,D2,D1,D0,S1,S0; output Y;
table //D3 D2 D1 D0 S1 S0 : Y
    ? ? ? 1 0 0 : 1;
    ? ? ? 0 0 0 : 0;
    ? ? 1 ? 0 1 : 1;
    ? ? 0 ? 0 1 : 0;
    ? 1 ? ? 1 0 : 1;
    ? 0 ? ? 1 0 : 0;
    1 ? ? ? 1 1 : 1;
    0 ? ? ? 1 1 : 0;
endtable
endprimitive
```

【例 2-9】

```
module MUX41UDP (D,S,DOUT) ;
input [3:0] D;
input [1:0] S;
output DOUT;
MUX41_UDP (DOUT,D[3],D[2],
           D[1],D[0],S[1],S[0]);
endmodule
```

2.3 Verilog加法器设计

2.3.1 全加器设计及例化语句应用

1. 全加器原理图结构

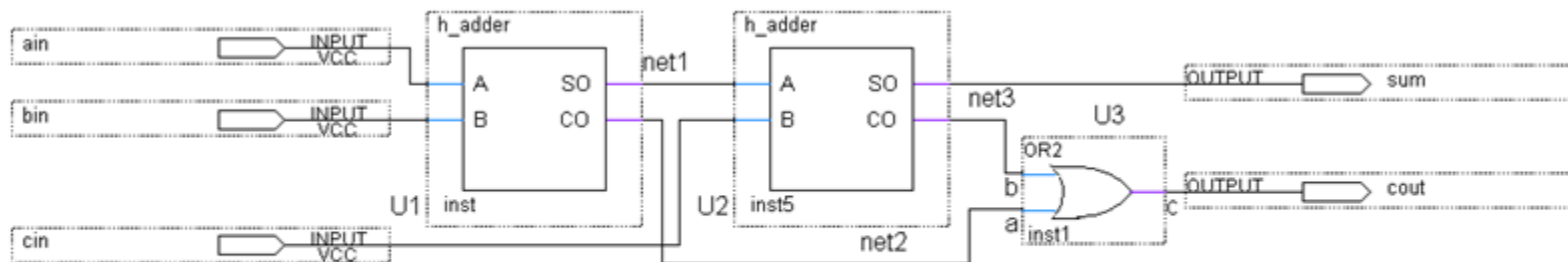


图 2-7 全加器 f_adder 电路图

2.3 Verilog加法器设计

2.3.1 全加器设计及例化语句应用

2. 全加器顶层设计文件

【例 2-10】//全加器顶层设计描述

```
module f_adder(ain,bin,cin,cout,sum);
    output cout,sum;    input ain,bin,cin;
    //定义网线型变量用作内部元件间连线
    wire net1,net2,net3;
    //使用位置关联法进行例化
    h_adder U1( ain, bin, net1, net2);
    h_adder U2(.A(net1), .SO(sum),
               .B(cin), .CO(net3) );
    or U3(cout, net2, net3); //使用位置关联法进行例化
endmodule
```

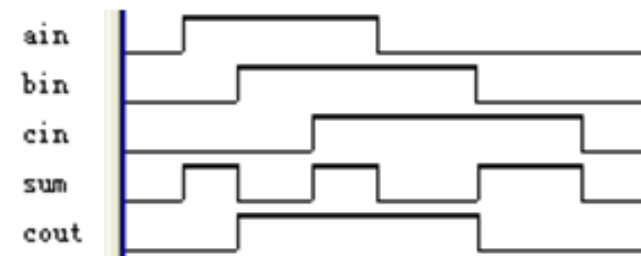


图 2-8 全加器仿真时序



2.3 Verilog加法器设计

2.3.1 全加器设计及例化语句应用

3. Verilog例化语句应用方法

<模块元件名> <例化元件名> (.例化元件端口 (例化元件外接端口名), ...);

```
h_adder U2 (.A(net1), .SO(sum), .B(cin), .CO(net3));
```

```
h_adder U2 (.B(cin), .CO(net3), .A(net1), .SO(sum));
```

```
h_adder U1( ain, bin, net1, net2);
```

2.3 Verilog加法器设计

2.3.2 8位加法器设计及算术操作符应用

【例 2-11】	【例 2-12】
<pre>module ADDER8B (A, B, CIN, COUT, DOUT); output [7:0] DOUT; output COUT; input [7:0] A, B; input CIN; wire [8:0] DATA; //加操作的进位自动进入 DATA [8] assign DATA = A + B + CIN; assign COUT = DATA[8]; assign DOUT = DATA[7:0]; endmodule</pre>	<pre>module ADDER8B (A, B, CIN, COUT, DOUT); output [7:0] DOUT; output COUT; input [7:0] A, B; input CIN; //加操作的进位进入并位 COUT assign {COUT, DOUT} = A + B + CIN; endmodule</pre>

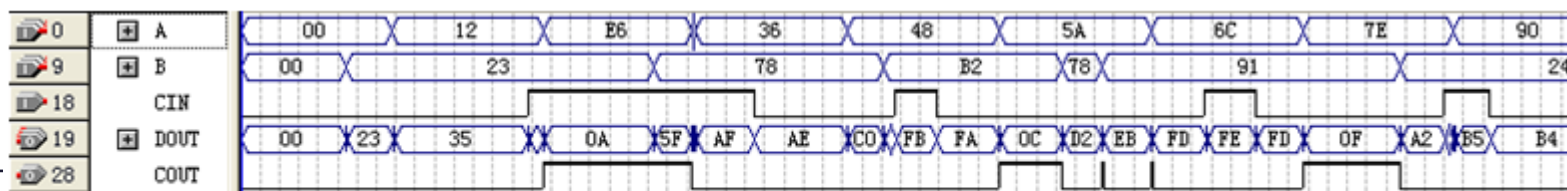


图 2-9 八位加法器仿真波形

2.3 Verilog加法器设计

2.3.2 8位加法器设计及算术操作符应用

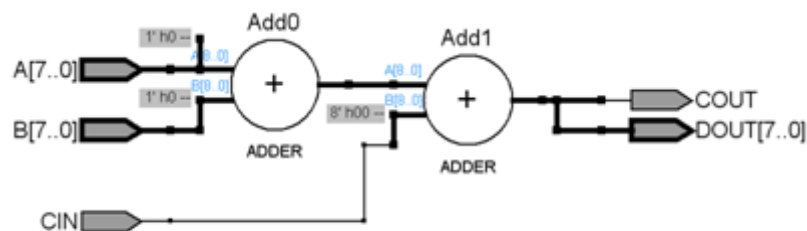


图 2-10 8 位加法器 QuartusII 综合之 RTL 电路

表 2-3 算术操作符

逻辑操作符	功能	说明	示例
+	加		$S = A + B = 8'b00011000$
-	减		$S = B - A = 8'b11111110$
*	乘		$S = A * B = 8'b10001111 = 2'H8F$
/	除	结果: 小数抛弃	$S = A / 3 = 8'b00000100$
%	求余	除法求余数	$S = A \% 3 = 8'b00000001$
设示例数据是: $A[3:0] = 4'b1101$; $B[3:0] = 4'b1011$; 定义 s 为 $s[7:0]$			

2.3 Verilog加法器设计

2.3.3 BCD码加法器设计

【例 2-13】

```
module BCD_ADDER (A,B,D) ;
    input [7:0] A,B;    output [8:0] D;
    wire [4:0] DT0, DT1 ; reg [8:0] D; reg S;
    always@ (DT0)
        begin if (DT0[4:0] >= 5'b01010 )
            //如果低位 BCD 码的和大于等于 10,则使和加上 6, 且有进位, 使进位标志 s 等于 1。
                begin D[3:0] = (DT0[3:0]+4'b0110); S=1'b1; end
                else begin D[3:0] = DT0[3:0] ; S=1'b0; end
            end //否则, 将低位值赋予低位 BCD 码 D[3:0]输出, 无进位, 使进位标志 s 等于 0。
    always@ (DT1)
        begin if (DT1[4:0] >= 5'b01010 )
                begin D[7:4] = (DT1[3:0]+4'b0110); D[8]=1'b1; end
                else begin D[7:4] = DT1[3:0] ; D[8]=1'b0; end
            end
    assign DT0 = A[3:0] + B[3:0] ; //设没有来自低位的进位。
    assign DT1 = A[7:4] + B[7:4] + S; //S 是来自低位 BCD 码相加的进位。
endmodule
```

2.3 Verilog加法器设计

2.3.3 BCD码加法器设计

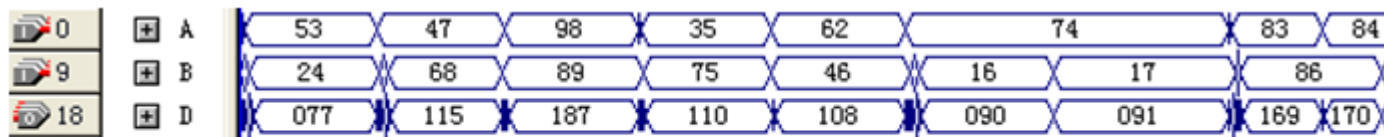


图 2-11 例 2-13 的仿真波形

表 2-4 不等式操作符

等式操作符	含义	操作示例
>	大于	(A < B) = 0; (A > B) = 1;
<	小于	(A < 20) = 1; (A > 12) = 1;
<=	小于或等于	(A >= 14) = 0; (A <= 13) = 1;
>=	大于或等于	注，以上示例中，设 A=4'B1101 ; B=4'B0110



2.4 组合逻辑乘法器设计

2.4.1 参数定义关键词parameter和localparam

parameter 标识符名 1 = 表达式或数值 1, 标识符名 2 = 表达式或数值 2, ... ;

```
parameter A=15 , B=4'b1011, C=8'hAC ;  
parameter d=8'b1001_0011 , e=8'sb10101101 ;
```

2.4 组合逻辑乘法器设计

2.4.2 整数型寄存器类型定义

`integer` 标识符 1, 标识符 2, ... , 标识符 n [msb: lsb] ;

```
module EXAPL (R,G);
parameter S=4;          //定义参数 s
output[2*S:1] R,G ;    //定义两个 8 位输出变量
integer A, B[3:0];     //定义了 5 个 integer 类型: A、B[0]、B[1]、B[3]等, 都是 32 位
reg[2*S:1] R,G;
always @( A, B )
begin
    B[2] = 367;        // 整数完整赋值, 因为 B[2]有 32 位
    R=B[2];           // 32 位 integer 整数类型 B[2]赋给 8 位 reg 类型 R, B[2]高位被截
    A=-20;            // 整数完整赋值, 因为 A 有 32 位, 将-20 赋予 A, 对应无符号数 65516
    G=A;              // 32 位 integer 整数类型 A 赋给 8 位 reg 类型 G, A 高位被截
    B[0]= 3'B101;    // 允许二进制数直接赋给 integer 类型 B[0]。
end
endmodule
```



2.4 组合逻辑乘法器设计

2.4.3 for语句用法

for (循环初始值设置表达式; 循环控制条件表达式; 循环控制变量增值表达式)
begin 循环体语句结构 end

2.4.4 移位操作符应用法

$V \gg n$ 或 $V \ll n$

$V \ggg n$ 或 $V \lll n$

```
output signed[7:0] y;  
input signed[7:0] a;  
assign y = (a<<<2);  
若 a=10101011,则输出 y=10101100  
若 a=10001111,则输出 y=00111100
```

```
parameter C=8'sb10101011;  
parameter D=8'sb01001110;  
output [7:0] Y1,Y2;  
assign Y1=(C>>>2);//结果: Y1=11101010  
assign Y2=(D>>>2);//结果: Y2=00010011
```

2.4 组合逻辑乘法器设计

2.4.5 两则乘法器设计示例

【例 2-14】	【例 2-15】
<pre> module MULT4B (R,A,B); parameter S=4; output [2*S:1] R ; input [S:1] A,B ; reg [2*S:1] R ; integer i; always @(A or B) begin R = 0 ; for(i=1; i<=S; i=i+1) if(B[i]) R=R+(A<<(i-1)); end endmodule </pre>	<pre> module MULT4B (R,A,B); parameter S=4; output [2*S:1] R ; input [S:1] A,B ; reg [2*S:1] R,AT; reg [S:1] BT,CT; always @(A,B) begin R=0; AT = {{S{1'B0}},A}; BT = B; CT = S; for(CT=S; CT>0; CT=CT-1) begin if(BT[1]) R=R+AT; AT = AT<<1; BT = BT>>1; end end endmodule </pre>

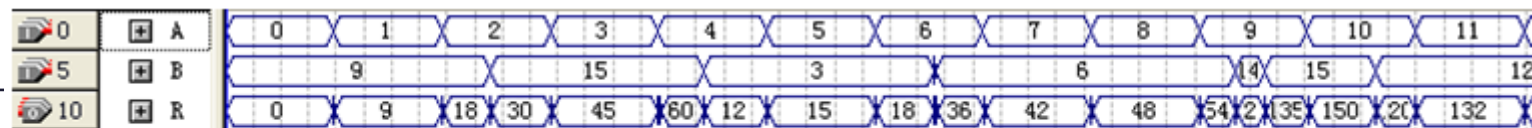


图 2-12 四位乘法器时序仿真图



2.4 组合逻辑乘法器设计

2.4.6 repeat语句用法

repeat (循环次数表达式)

begin 循环体语句结构 **end**

【例 2-16】

```
module MULT4B (R, A, B);  
    parameter S=4;  
    output [2*S:1] R; input [S:1] A, B ;  
    reg [2*S:1] TA, R;  
    reg [S:1] TB;  
    always @(A or B) begin  
        R = 0 ; TA = A ; TB = B ;  
        repeat(S) begin  
            if(TB[1]) begin R=R+TA; end  
            TA = TA<<1;  
            TB = TB>>1;  
        end  
    end  
endmodule
```



2.4 组合逻辑乘法器设计

2.4.7 while语句用法

while (循环控制条件表达式)

begin 循环体语句结构 **end**

【例 2-17】

```
module MULT4B(A,B,R);
    parameter S=4;
    input[S:1] A,B;
    output[2*S:1] R;
    reg[2*S:1] R,AT;
    reg[S:1] BT,CT;
    always@(A or B) begin
        R=0; AT={{S{1'b0}},A};
        BT=B; CT=S;
        while(CT>0) begin
            if(BT[1]) R=R+AT; else R=R;
        begin CT=CT-1; AT=AT<<1; BT=BT>>1;
        end end
    end
endmodule
```



2.4 组合逻辑乘法器设计

2.4.8 Verilog循环语句的特点

2.4.9 parameter的参数传递功能

【例 2-18】

```
module MULTB (RP,AP,BP);  
    output[15:0] RP ; input[7:0] AP,BP ;  
    MULT4B #(.S(8)) U1(.R(RP), .A(AP), .B(BP) );  
endmodule
```

```
module SUB_E  
    #(parameter S1=4, parameter S2=5, parameter S3=2) (A,B,C);
```

```
SUB_E
```

```
    #(.S1(8), .S2(9), .S3(7)) U1(.C(CP), .A(AP), .B(BP) );
```

习题

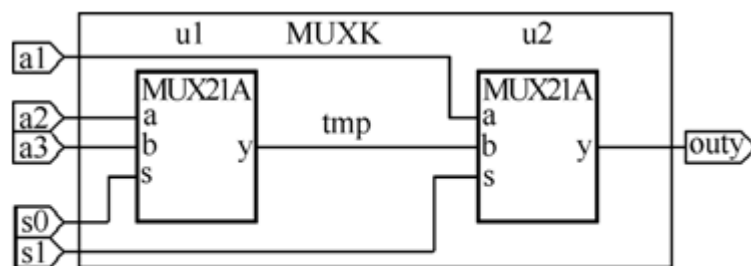


图 2-13 含 2 选 1 多路选择器的模块

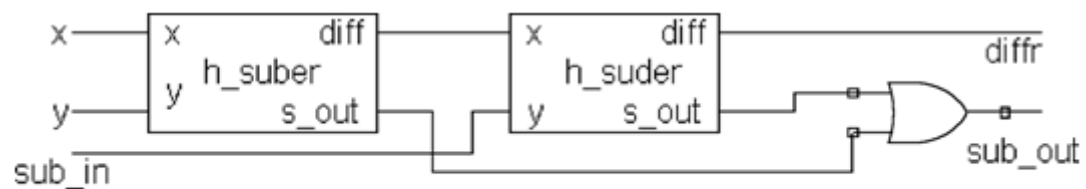


图 2-14 全减器模块图