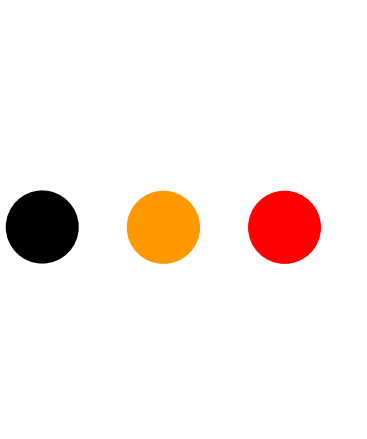

第3章

组合电路的Verilog设计

3.1 半加器电路的Verilog描述

$SO = A \oplus B$ ；　　$CO = A \cdot B$

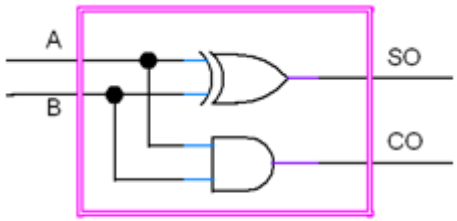

图 3-1　半加器的电路结构

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

图 3-2　半加器的真值表

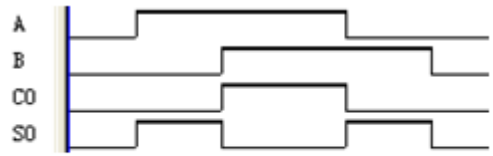

图 3-3　半加器的仿真功能波形图

3.1 半加器电路的Verilog描述

【例 3-1】

```
module h_adder (A,B,SO,CO) ;
    input A,B;
    output SO,CO;
    assign SO = A ^ B;    //将变量A和B执行异或逻辑后将结果赋给输出信号SO
    assign CO = A &B;     //将变量A和B执行与逻辑后将结果赋给输出信号CO
endmodule
```

3.1 半加器电路的Verilog描述

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;
```

3.1 半加器电路的Verilog描述

3．逻辑操作符

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;
```

```
`timescale 10ns/100ps
 assign #6 R1 = A & B;
```

3.1 半加器电路的Verilog描述

5. 关键字

6. 标识符

7. 注释符号

8. 规范的程序书写格式

9. 文件取名和存盘

3.2 多路选择器的Verilog描述

3.2.1 4选1多路选择器及其case语句表述方式

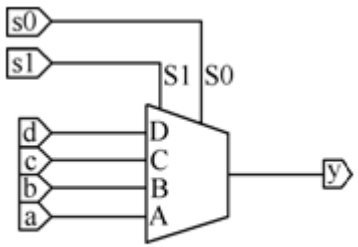

图 3-4 4 选 1 多路选择器

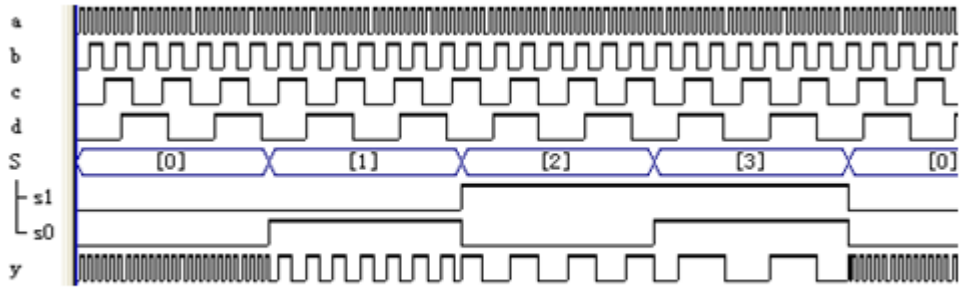

图 3-5 4 选 1 多路选择器 MUX41a 的时序波形

3.2 多路选择器的Verilog描述

3.2.1 4选1多路选择器及其case语句表述方式

【例 3-2】

```
module MUX41a (a, b, c, d, s1, s0, y);
        input   a, b, c, d ;
        input   s1, s0 ;
        output  y ;
        reg  y ;
       always @ ( a  or  b  or  c  or  d  or  s1  or  s0 )
            begin     :  MUX41   //块语句开始
                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
```

电路模块端口说明和定义段

信号类型定义段

具体描述电路功能的Verilog语句

电路模块功能描述段

Verilog表述的可综合的完整电路模块

3.2 多路选择器的Verilog描述

3.2.1 4选1多路选择器及其case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
```

3.2 多路选择器的Verilog描述

3.2.1 4选1多路选择器及其case语句表述方式

3．块语句begin _end

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

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
```

3.2 多路选择器的Verilog描述

3.2.1 4选1多路选择器及其case语句表述方式

5．Verilog的4种逻辑状态

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>

3.2 多路选择器的Verilog描述

3.2.2 4选1多路选择器及assign语句表述方式

【例 3-3】

```
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 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
```

3.2 多路选择器的Verilog描述

3.2.2 4选1多路选择器及assign语句表述方式

1．按位逻辑操作符

表 3-1 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	~A= 1'b1	~C =4'b0011	~E =6'b101001
|	逻辑或	A|B= 1'b 1	C|D =4'b1111	C|E =6'b011110
&	逻辑与	A&B =1'b 0	C &D =4'b1000	C &E =6'b000100
^	逻辑异或	A^B =1'b 1	C ^ D =4'b0111	C ^ E =6'b011010
~^ 或 ^~	逻辑同或	A~^ B =1'b 0	C ~^ D =4'b1000	C ~^ 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;				

3.2 多路选择器的Verilog描述

3.2.2 4选1多路选择器及assign语句表述方式

2．等式操作符

表 3-2 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		

3.2 多路选择器的Verilog描述

3.2.2 4选1多路选择器及assign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;
```

3.2 多路选择器的Verilog描述

3.2.3 4选1多路选择器及条件赋值语句表述方式

【例 3-4】

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

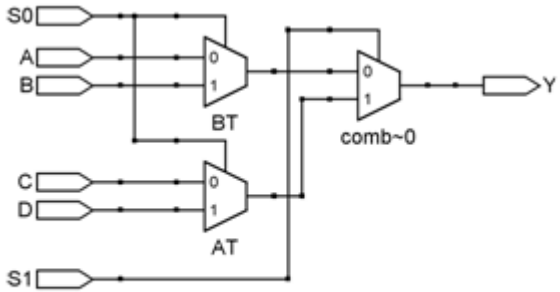

图 3-6　例 3-4 的 RTL 图

3.2 多路选择器的Verilog描述

3.2.4 4选1多路选择器及其条件语句表述方式

【例 3-5】

```
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                          //块语句起始
            SEL = {S1,S0};           //把 S1,S0 并位为 2 元素矢量变量 SEL[1:0]
        if (SEL==0)  Y = A;          //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
```

3.2 多路选择器的Verilog描述

3.2.4 4选1多路选择器及其条件语句表述方式

1．if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

3.3 Verilog加法器设计

3.3.1 全加器设计及例化语句应用

1. 全加器原理图结构

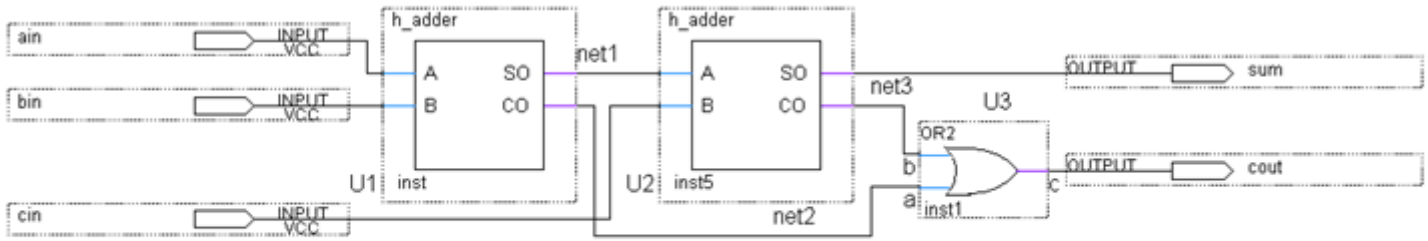

图 3-7 全加器 f_adder 电路图

3.3 Verilog加法器设计

3.3.1 全加器设计及例化语句应用

2. 全加器顶层设计文件

【例 3-6】

```
module f_adder(ain,bin,cin,cout,sum);
   output cout,sum;   input ain,bin,cin;
   wire  net1,net2,net3;
   h adder  U1(  ain, bin, net1, net2);
   h adder  U2(.A(net1), .SO(sum),
               .B(cin), .CO(net3) );
        or  U3(cout, net2, net3);
  endmodule
```

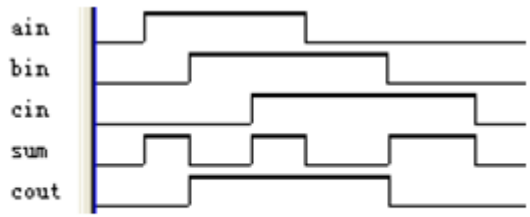

图 3-8　全加器仿真时序

3.3 Verilog加法器设计

3.3.1 全加器设计及例化语句应用

3. Verilog例化语句及其用法

```
<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);
```

3.3 Verilog加法器设计

3.3.2 半加器的UDP结构建模描述方式

【例 3-7】

```
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 3-8】

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

1. 库元件及其调用

2. 用户自定义原语

3.3 Verilog加法器设计

3.3.3 利用UDP元件设计多路选择器

【例 3-9】

```
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 3-10】

```
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
```

3.3 Verilog加法器设计

3.3.4 8位加法器设计及算术操作符应用

【例 3-11】

```
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
```

【例 3-12】

```
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
```

3.3 Verilog加法器设计

3.3.4 8位加法器设计及算术操作符应用

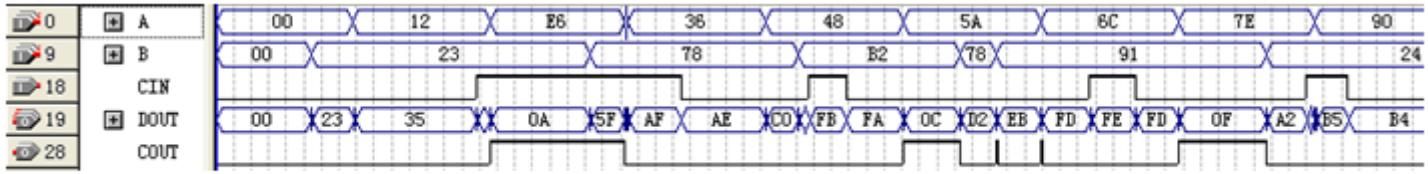

图 3-9 八位加法器仿真波形

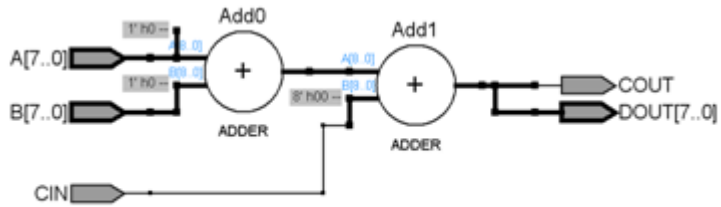

图 3-10 8 位加法器之 RTL 电路

3.3 Verilog加法器设计

3.3.5 算术运算操作符

表 3-3　算术操作符功能及其示例

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]			

3.3 Verilog加法器设计

【例 3-13】

```
module test1 (A,B,C,D,RCD,RAB,RM1,RM2,S,C0,R1,R2);
input [3:0] C,D ;  input signed [3:0] A,B;
output [3:0] RCD;  output [3:0] RAB;
output [7:0] RM1;  output [7:0] RM2;
output [3:0] S;   output C0;  output R1,R2;
reg [3:0] S ;  reg C0;
reg [3:0] RCD ;    reg [7:0] RM1 ;
reg signed [3:0] RAB;  reg signed [7:0] RM2;
reg R1,R2;
always@(A,B,C,D) begin
  RCD <= C+D;  RAB <= A+B;
 RM1 <= C*D; RM2 <= A*B;
  {C0,S} <= {1'b0,C} - {1'b0,D};//注意并位操作
   R1  <= (C>D); R2<=(A>B);
  end
endmodule
```

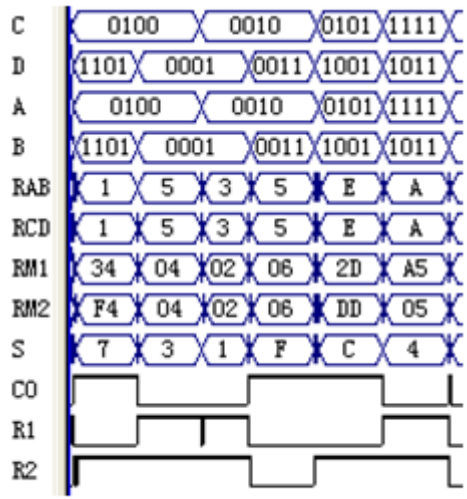

图 3-11 例 3-13 的仿真波形

3.3 Verilog加法器设计

3.3.6 BCD码加法器设计

表 3-4　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

【例 3-14】

```
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
```

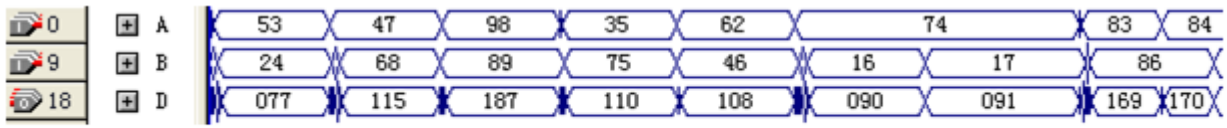

图 3-12 例 3-14 的仿真波形

3.4 组合逻辑乘法器设计

3.4.1 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 ;
```

3.4 组合逻辑乘法器设计

3.4.2 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
```

3.4 组合逻辑乘法器设计

3.4.3 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

3.4 组合逻辑乘法器设计

3.4.4 移位操作符及其用法

V >> n 或 V <<n

V >> 1 的值是 8'b01100100 ;

V << 3 的值是 8'b01001000

V >>> n 或 V <<< 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
```

3.4 组合逻辑乘法器设计

3.4.5 两则乘法器设计示例

【例 3-15】

```
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
```

【例 3-16】

```
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
```

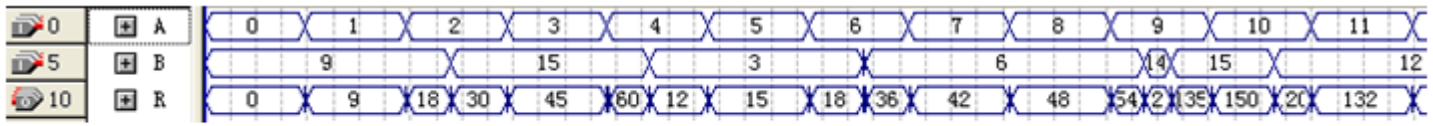

图 3-13　四位乘法器时序仿真图

3.4 组合逻辑乘法器设计

3.4.6 repeat语句用法

repeat（循环次数表达式）
  begin　循环体语句结构　end

3.4 组合逻辑乘法器设计

3.4.7 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 3-17】

```
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
```

【例 3-18】

```
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
```

3.4 组合逻辑乘法器设计

3.4.8 parameter的参数传递功能

```
module MULT4B #(parameter S=4)(R,A,B);
```

或：

```
module MULT4B #(parameter S)(R,A,B);
```

【例 3-19】

```
module MULT4B
 #(parameter S)(R,A,B);
  output [2*S:1] R; input [S:1] A,B ;
  reg [2*S:1] TA,R;
  reg [S:1] TB;
...  //以下与例 3-16 相同
```

【例 3-20】

```
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) );
```

3.5 RTL概念

RTL（Register Transport Level）的概念最初产生于对某类电路的描述。

习　题

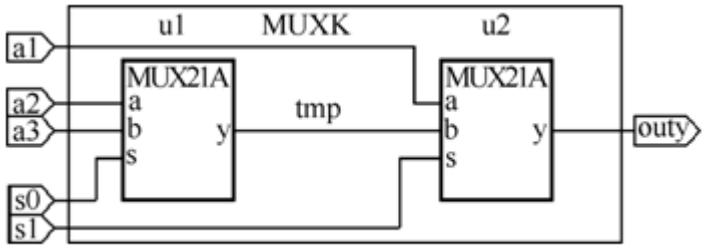

图 3-14　含 2 选 1 多路选择器的模块

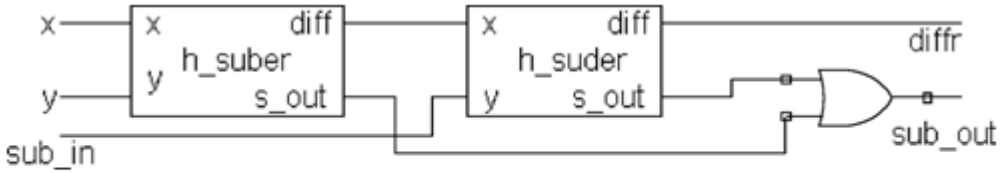

图 3-15　全减器模块图