

第3章

组合电路的Verilog设计

3.1 半加器电路的Verilog描述

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

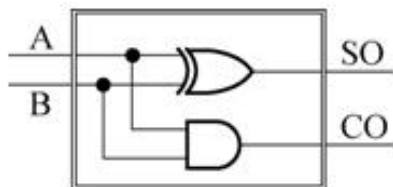


图 3-1 半加器的电路结构

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

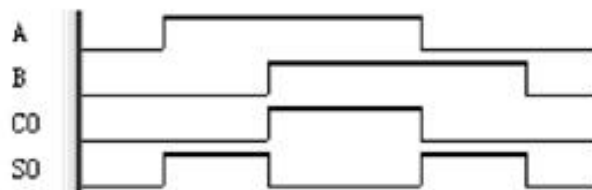


图 3-3 半加器电路的时序波形

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

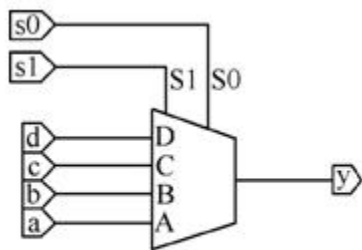


图 3-4 4 选 1 多路选择器

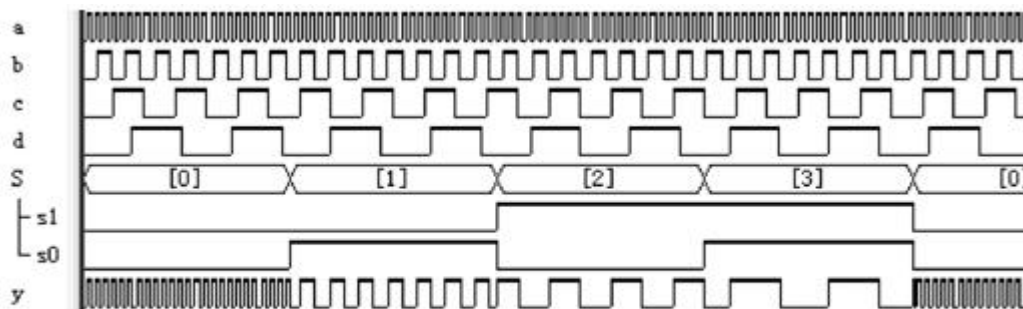


图 3-5 4 选 1 多路选择器 MUX41a 的时序波形

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
语句描述段

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的四种逻辑状态

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	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

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	$(D!=C)=0$; $(3!=4)=1$;
===	全等	$(D===C)=1$; $(E===4'b0x10)=0$;
!==	不全等	$(E!==4'b0x10)=1$;

设 : $A=4'b1011$; $B=4'b0010$; $C=4'b0z10$; $D=4'b0z10$; $E=3'bx10$

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
```

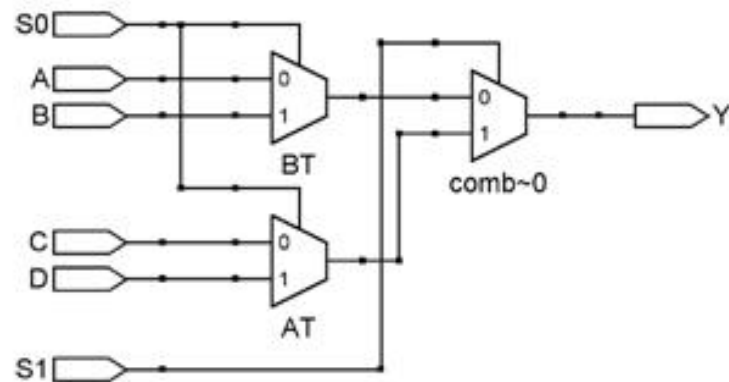


图 3-6 例 3-4 的 RTL 图

条件表达式 ? 表达式 1 : 表达式 2

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
            SEL = {S1,S0}; //块语句起始
            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

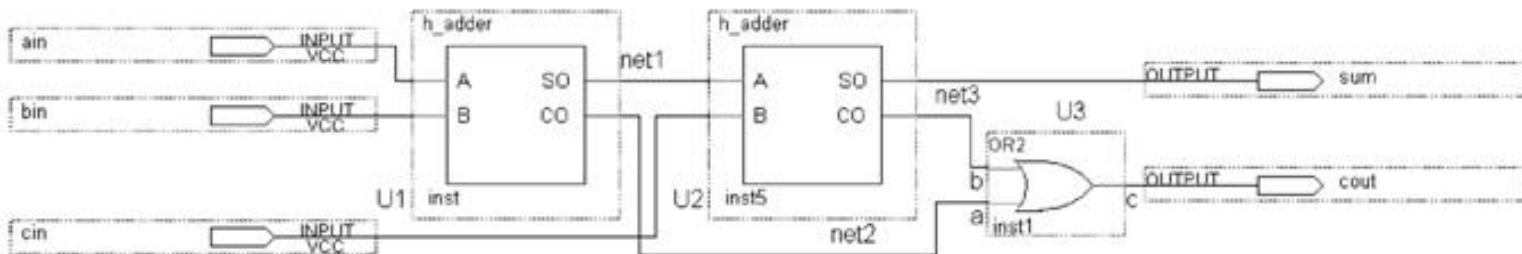


图 3-7 全加器 f_adder 电路图

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
```

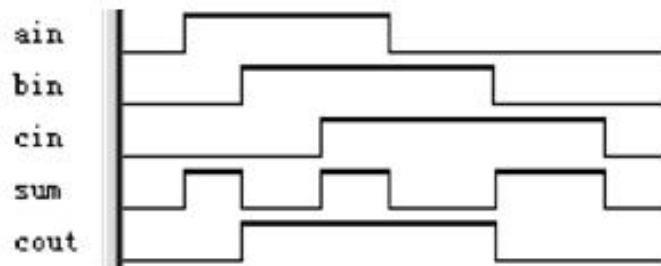


图 3-8 全加器仿真时序



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 8位加法器设计及算术操作符应用

【例 3-7】

```
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 3-8】

```
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

3.3.2 8位加法器设计及算术操作符应用

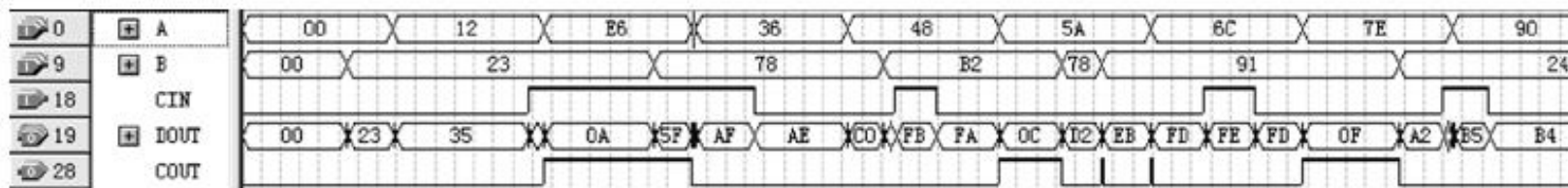


图 3-9 8 位加法器仿真波形

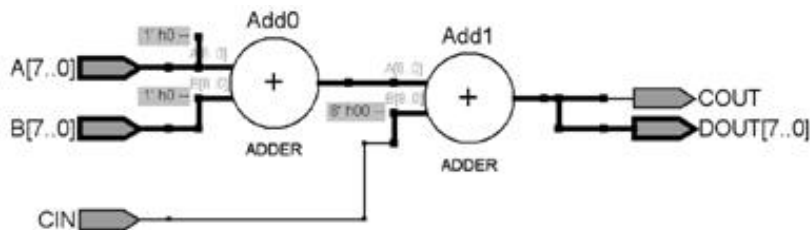


图 3-10 8 位加法器的 RTL 电路

3.3 Verilog加法器设计

3.3.3 算术运算操作符

表 3-3 算术操作符

算术操作符	功能	说明	操作示例
+	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： $A[3:0]=4'b1101$ ； $B[3:0]=4'b1011$ ；定义 S 为 $S[7:0]$

3.3 Verilog加法器设计

3.3.3 算术运算操作符

【例 3-9】

```
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
```

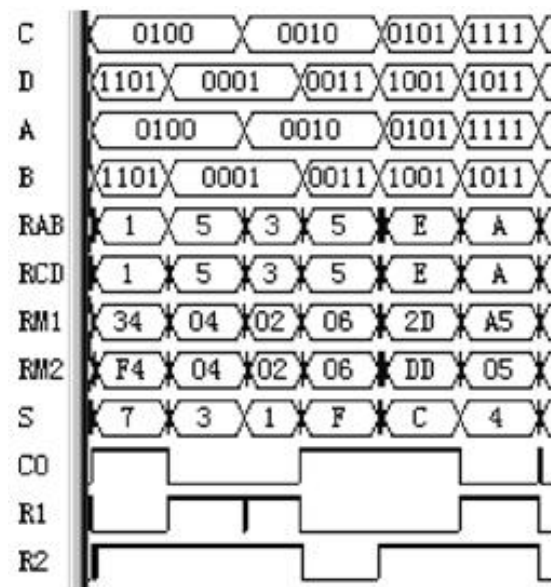


图 3-11 例 3-9 的仿真波形

3.3 Verilog加法器设计

3.3.4 BCD码加法器设计

【例 3-10】

```
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
```

3.3 Verilog加法器设计

3.3.4 BCD码加法器设计

表 3-4 不等式操作符

不等式操作符	含义	操作示例
>	大于	$(A > B) = 1$; $(A > 12) = 1$;
<	小于	$(A < B) = 0$; $(A < 20) = 1$;
\leq	小于或等于	$(A \leq 13) = 1$;
\geq	大于或等于	$(A \geq 14) = 0$;

设 : $A=4'B1101$, $B=4'B0110$

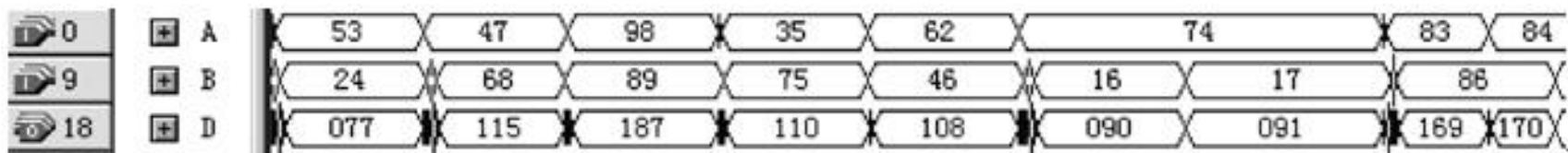


图 3-12 例 3-10 的仿真波形



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 5 个整数型 : A、B[0]、B[1]、B[2]、B[3] , 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; end // 允许二进制数直接赋给 integer 类型 B[0]  
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 >>> 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 3-11】

```
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 3-12】

```
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 end module
```


3.4 组合逻辑乘法器设计

3.4.5 两则乘法器设计示例

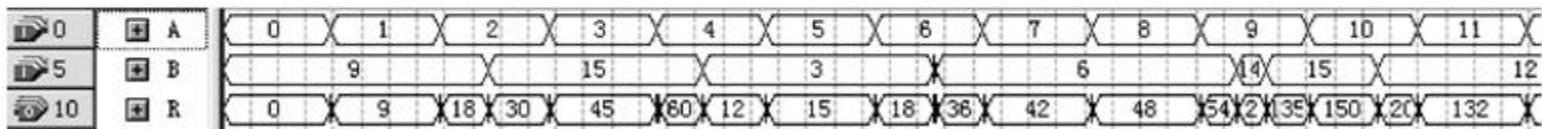
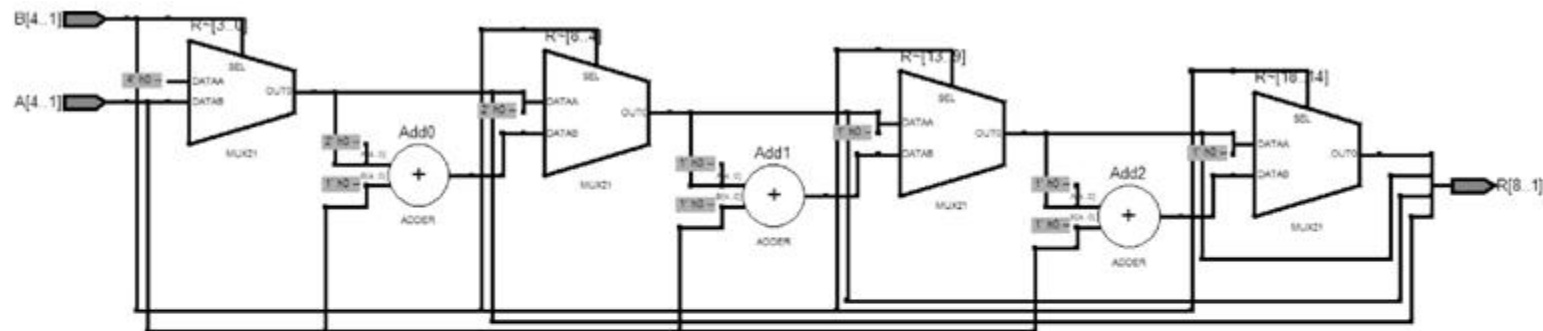


图 3-13 上图是 4 位乘法器的 RTL 逻辑电路图，下图是其时序仿真图



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

【例 3-13】

```
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 3-14】

```
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

3.4.7 while语句用法

```
while ( 循环控制条件表达式 )  
    begin 循环体语句结构 end
```

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 3-15】

```
module MULT4B  
# (parameter S) (R,A,B);  
output [2*S:1] R;  
input [S:1] A,B;  
reg [2*S:1] AT,R;  
reg [S:1] BT,CT;  
... //以下与例 3-16 相应部分相同
```

【例 3-16】

```
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
```

3.4 组合逻辑乘法器设计

3.4.8 parameter的参数传递功能

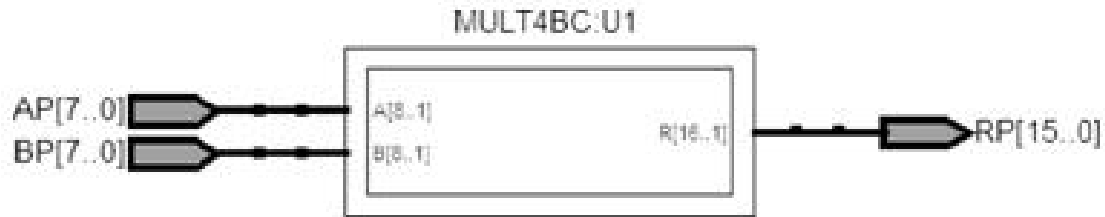


图 3-14 例 3-16 的 RTL 图，内部结构如图 3-15 所示

3.4 组合逻辑乘法器设计

3.4.8 parameter的参数传递功能

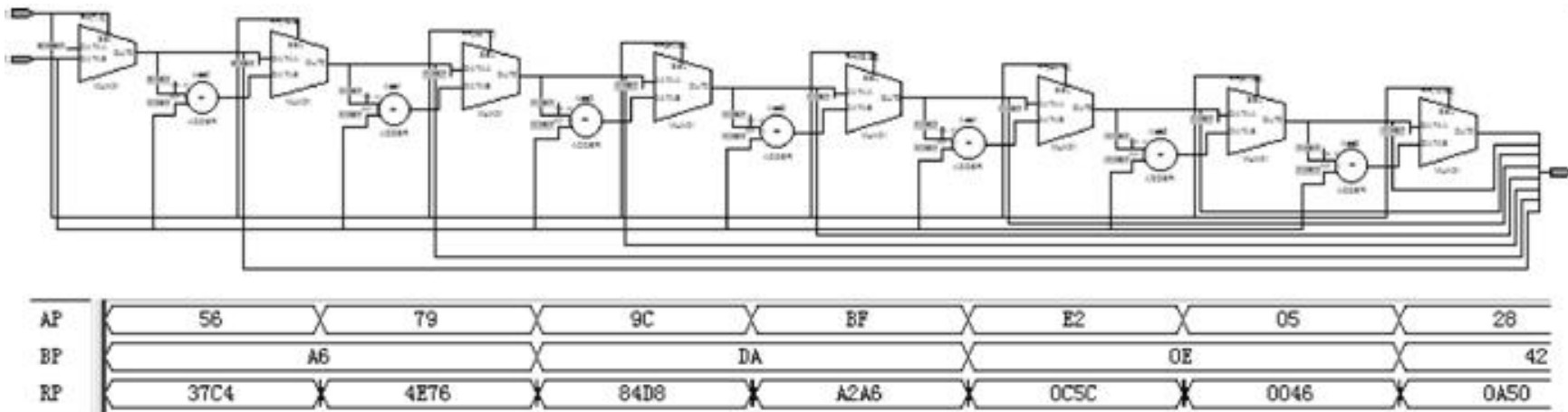


图 3-15 上图是 8 位乘法器的 RTL 逻辑电路图，下图是其仿真波形图

3.4 组合逻辑乘法器设计

3.4.8 parameter的参数传递功能

```
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。

(1) **RTL描述**

(2) **RTL级或RTL电路**

习题

3-5 图3-16所示的是双2选1多路选择器构成的电路MUXK。对于其中MUX21A，当 $s=0$ 和 $s=1$ 时，分别有 $y=a$ 和 $y=b$ 。试在一个模块结构中用两个过程来表达此电路。

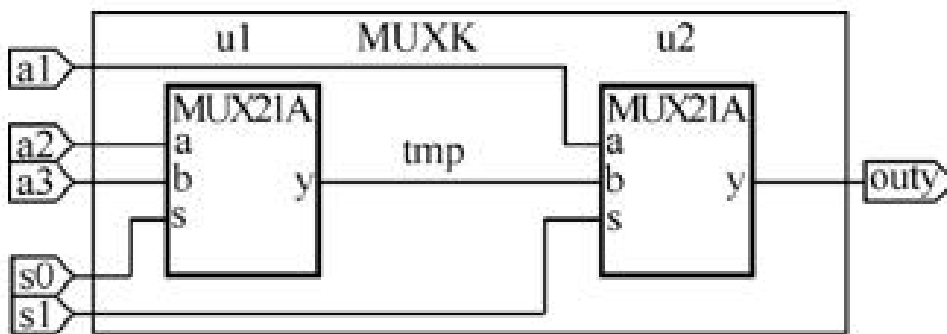


图 3-16 含 2 选 1 多路选择器的模块

习 题

3-7 给出全减器的Verilog描述。要求：

(1) 首先设计半减器，然后用例化语句将它们连接起来。图3-17中h_suber是半减器，diff是输出差，s_out是借位输出，sub_in是借位输入。根据图3-17设计全减器。

(2) 以全减器为基本硬件，构成串行借位的8位减法器，要求用例化语句来完成此项设计。

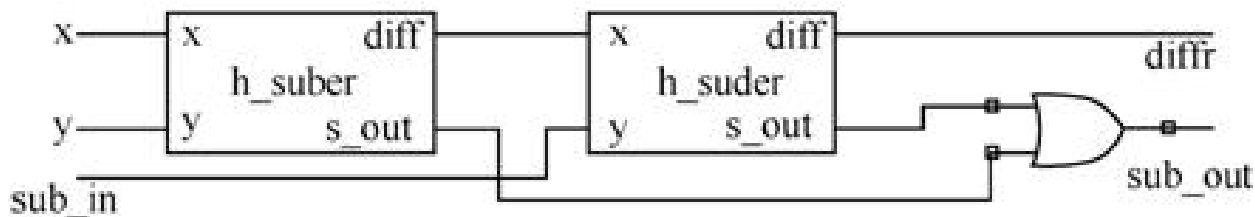


图 3-17 全减器模块图