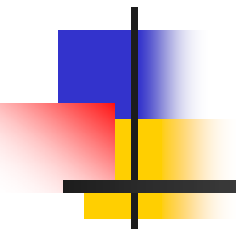

第5章

Verilog设计深入

5.1 过程中的两类赋值语句

5.1.1 阻塞式赋值

目标变量名 = 驱动表达式;

5.1.2 非阻塞式赋值

目标变量名 <= 驱动表达式;

【例 5-1】

```
assign  Q1 = A | B;
assign  Q2 = B ^ C;
assign  Q1 = C & A;
```

【例 5-2】

```
always @(A,B,C)     begin
    Q1 <= A | B;
    Q2 <= B & C;
    Q1 <= C ^ A;      end
```

5.1 过程中的两类赋值语句

5.1.2 非阻塞式赋值

【例 5-3】阻塞式赋值示例

```
always @(A,B) begin
      M1 = A ;
      M2 = B & M1;
       Q = M1 | M2;    end
```

【例 5-4】非阻塞式赋值示例

```
always @(A,B) begin
      M1 <= A ;
      M2 <= B & M1;
       Q <= M1 | M2;    end
```

5.1 过程中的两类赋值语句

5.1.3 深入认识阻塞赋值和非阻塞式赋值的特点

【例 5-5】使用非阻塞赋值符的时序模块设计

```
module DDF3(CLK,D,Q);
    output Q ; input CLK,D;
    reg a,b,Q;
     always @(posedge CLK) begin
       a <= D;
       b <= a;
       Q <= b;
     end
endmodule
```

【例 5-6】使用阻塞赋值符的时序模块设计

```
module DFF3(CLK,D,Q);
    output Q ; input CLK,D;
    reg a,b,Q;
    always @(posedge CLK) begin
        a = D;
        b = a;
        Q = b;
     end
endmodule
```

5.1 过程中的两类赋值语句

5.1.3 深入认识阻塞赋值和非阻塞式赋值的特点

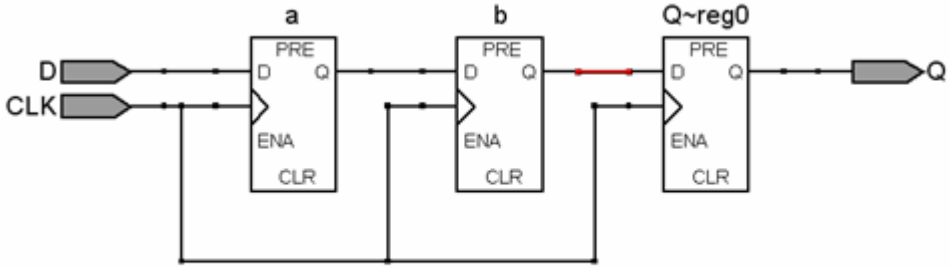

图 5-1　例 5-5 综合后的 RTL 电路

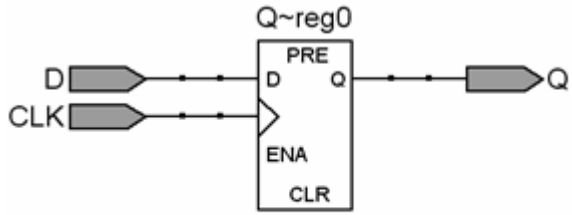

图 5-2　例 5-6 综合后的 RTL 电路

5.1 过程中的两类赋值语句

5.1.3 深入认识阻塞赋值和非阻塞式赋值的特点

【例 5-7】

```
always @( * )
  begin
    if (in1==1)  ...                    // 第 1 行
       a1 <= 4'B1010 ;                  // 第 2 行
            ...
    if (in2==0)  ...                    // 第 15+n 行
            ...
       b1 = 4'B0011 ;                   // 第 30+m 行
            ...
  end
```

【例 5-8】

```
Q = b;
b = a;
a = D;
```

5.1 过程中的两类赋值语句

【例 5-9】非阻塞赋初值导致错误

```
module mux4_1(
 D0,D1,D2,D3,S1,S0,OUT);
 output OUT ;
 input D0,D1,D2,D3,S1,S0;
  reg [2:0] T ; reg OUT;
  always @(D0,D1,D2,D3,S1,S0)
     begin   T <= 0;
       if (S0==1)   T <= T+1 ;
       if (S1==1)   T <= T+2 ;
         case (T)
           0 : OUT = D0 ;
           1 : OUT = D1 ;
           2 : OUT = D2 ;
           3 : OUT = D3 ;
     default : OUT = D0 ;
        endcase    end
endmodule
```

【例 5-10】阻塞赋初值正确

```
module mux4_1(
    D0,D1,D2,D3,S1,S0,OUT);
     output OUT ;
     input D0,D1,D2,D3,S1,S0;
       reg [2:0] T ; reg OUT;
       always @(D0,D1,D2,D3,S1,S0)
         begin   T = 0;
          if (S0==1)   T = T+1 ;
          if (S1==1)   T = T+2 ;
            case (T)
             0 : OUT = D0 ;
             1 : OUT = D1 ;
             2 : OUT = D2 ;
             3 : OUT = D3 ;
          default : OUT = D0 ;
           endcase    end
endmodule
```

5.1 过程中的两类赋值语句

5.1.3 深入认识阻塞赋值和非阻塞式赋值的特点

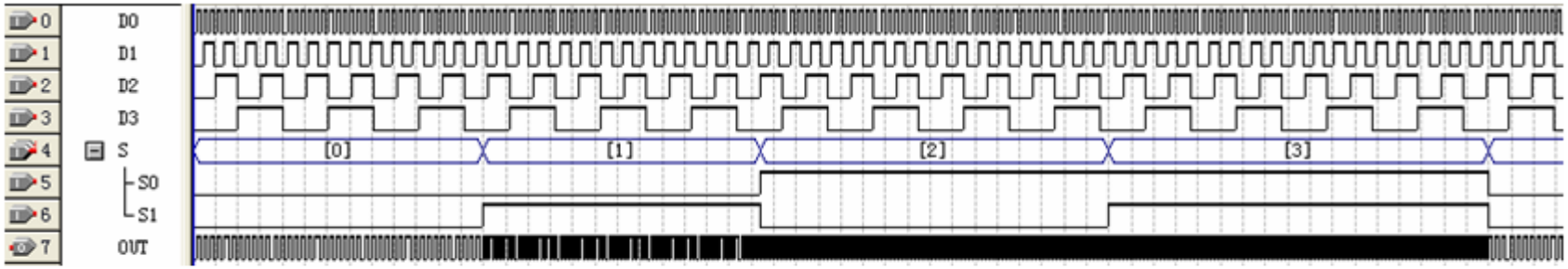

图 5-3 例 5-9 的错误工作时序

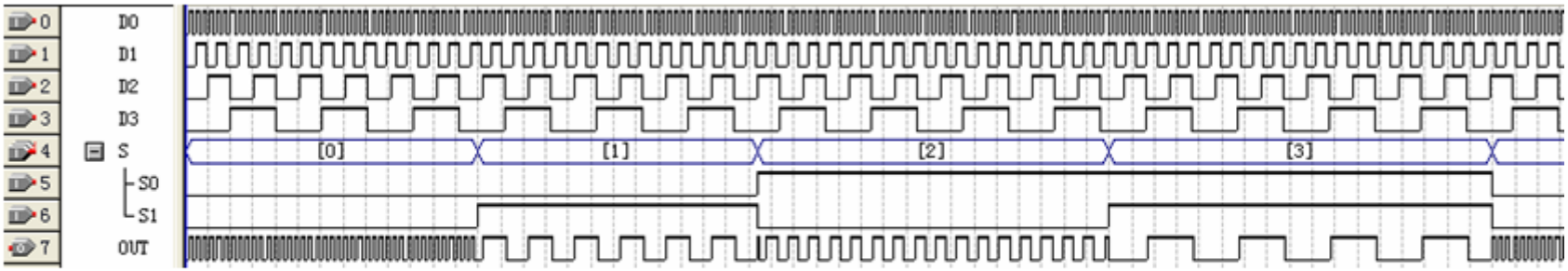

图 5-4 例 5-10 的正确工作时序

5.2 过程结构总结

1. 过程语句为一无限循环语句

2. 过程中的语句具有顺序和并行双重性

3. 过程语句本身是并行语句

4. 过程中只允许描述对应单一时钟的同步时序逻辑

5. 不完整条件语句与时序电路的关系

5.2 过程结构总结

【例 5-11】

```
module mux2_1(CLK,D,Q,RST); //2 选 1 多路选择器
    output Q ; input CLK,D,RST;  reg Q;
    always @(D or CLK or RST)
     if(CLK)  Q <= D;
        else  Q <= RST;  //注意 if 后含 else 语句
endmodule
```

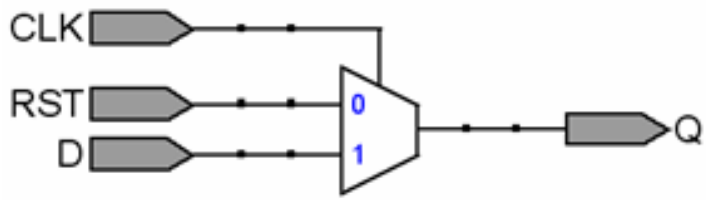

图 5-5　例 5-11 的多路选择器

5.2 过程结构总结

【例 5-12】含锁存器

```
module COMP(A,B,Q);
 output Q ; input [3:0] A,B;
    reg Q;
    always @(A,B) begin
       if(A>B)  Q = 1'b1;
  else if(A<B)  Q = 1'b0;
end      endmodule
```

【例 5-13】不含锁存器

```
 always @(A,B)
    begin
       if(A>B)  Q = 1'b1;
       else if(A<B)  Q = 1'b0;
        else         Q = 1'bz;
    end
endmodule
```

5.2 过程结构总结

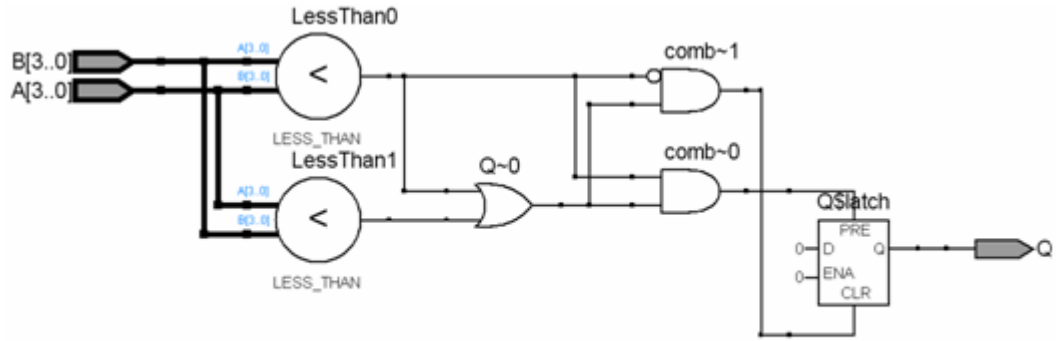

图 5-6　例 5-12 的 RTL 图，输出口被加上了锁存器

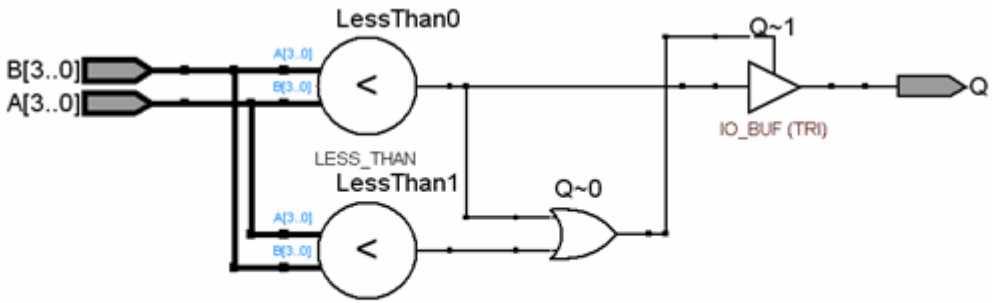

图 5-7　例 5-13 的 RLT 电路图，输出口没有锁存器，是纯组合电路

5.2 过程结构总结

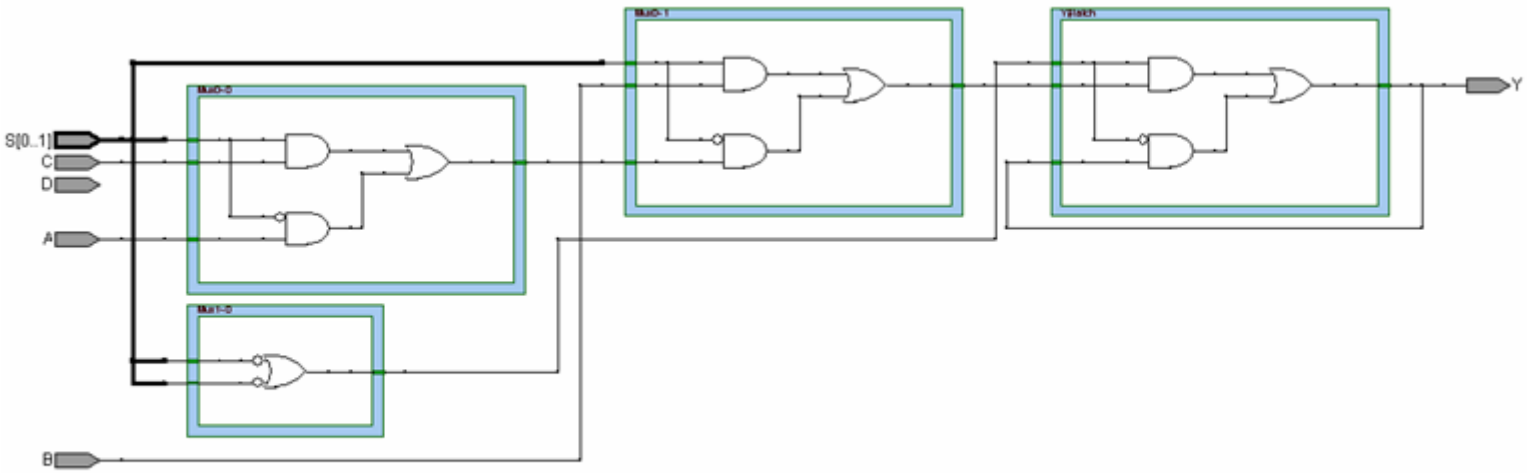

图 5-8　含时序模块的 3 选 1 多路选择器 RLT 图

5.3 移位寄存器设计

5.3.1 含同步预置功能的移位寄存器设计

```
REG8[6:0] <= REG8[7:1] ;
```

【例 5-14】

```
module SHFT1(CLK,LOAD,DIN,QB);                    // 右移移位寄存器
   output QB;  input CLK,LOAD;  input[7:0] DIN;
   reg [7:0]  REG8 ;                              //规定此变量为测试端口
   always @(posedge CLK )   begin
      if (LOAD)     REG8<=DIN ;  //注意 LOAD 是与 CLK 上升沿同步的控制信号
      else REG8[6:0] <= REG8[7:1] ;  end    //过程在此句结束
    assign QB = REG8[0] ;
endmodule
```

5.3 移位寄存器设计

5.3.1 含同步预置功能的移位寄存器设计

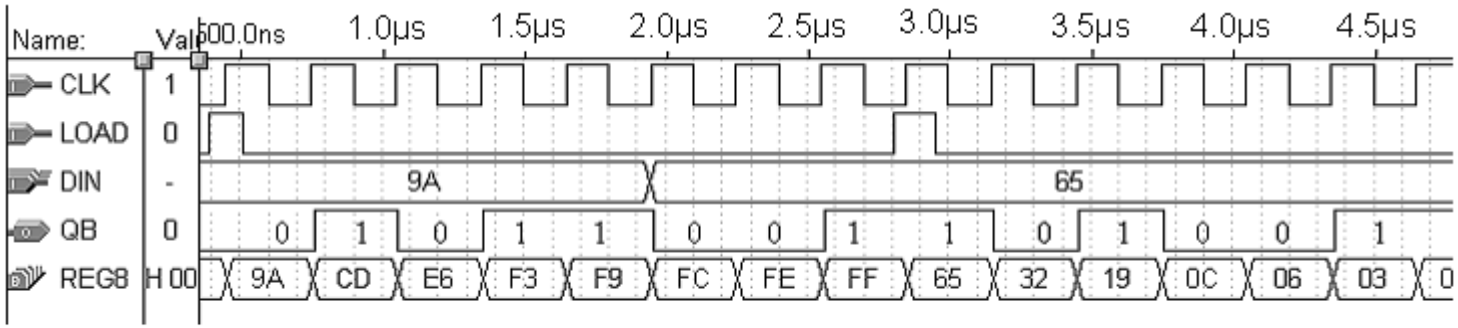

图 5-9　例 5-14 的工作时序图

5.3.2 模式可控的移位寄存器设计

【例 5-15】

```
module SHFT2 (CLK,C0,MD,D,QB,CN);
output CN ; output [7:0] QB;      //进位输出和移位数据输出
input CLK,C0;                     //时钟和进位输入
input [7:0] D;  input [2:0] MD;   //待加载移位的数据输入和移位模式控制字
reg[7:0]  REG ;     reg CY ;
always @(posedge CLK ) begin
case (MD)
1 : begin REG[0]<=C0; REG[7:1]<=REG[6:0]; CY<=REG[7]; end//带进位循环左移
2 : begin REG[0]<=REG[7] ; REG[7:1]<=REG[6:0] ;end //自循环左移
3 : begin REG[7]<=REG[0] ; REG[6:0]<=REG[7:1] ;end //自循环右移
4 : begin REG[7]<=C0; REG[6:0]<=REG[7:1];CY<=REG[0]; end //带进位循环右移
5 : begin REG<=D ; end  //加载待移数
default : begin REG <= REG ;  CY <= CY ; end  // 过程结束
endcase     end
  assign QB = REG ; assign CN = CY  ; //移位后输出，及移位后输出
endmodule
```

5.3 移位寄存器设计

5.3.2 模式可控的移位寄存器设计

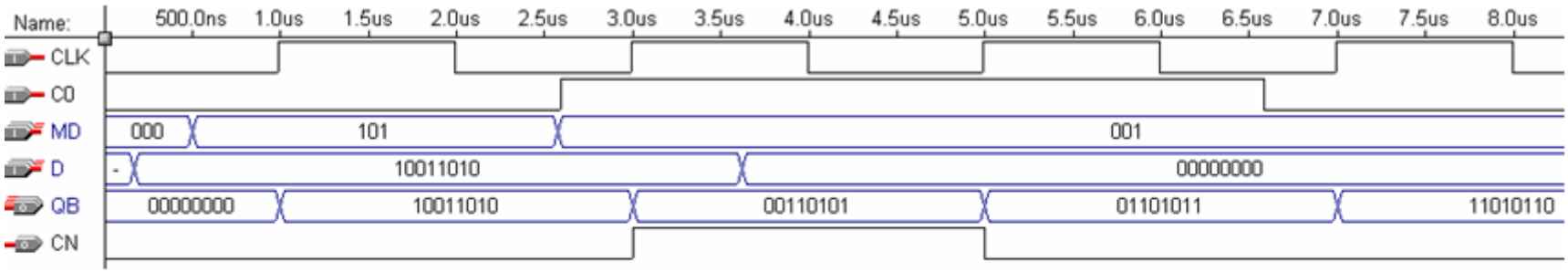

图 5-10 例 5-15 中带进位循环左移仿真波形(MD＝“001”)

5.3 移位寄存器设计

5.3.3 使用移位操作符设计移位寄存器

V>>n 或 V<<n

【例 5-16】

```
module SHIF4(DIN,CLK,RST,DOUT);
 input CLK,DIN,RST;
 output DOUT;
 reg [3:0] SHFT;
 always @(posedge CLK or posedge RST)
    if(RST) SHFT<=4'B0;
    else begin   SHFT[3]<=DIN;
      SHFT[2:0] <= SHFT[3:1];
    end
     assign DOUT=SHFT[0];
  endmodule
```

【例 5-17】

```
module SHIF4 (DIN,CLK,RST,DOUT);
 input CLK,DIN,RST;   output DOUT;
 reg [3:0] SHFT;
 always @(posedge CLK or posedge RST)
  if(RST) SHFT<=4'B0;
    else begin
       SHFT <= SHFT >> 1 ;
       SHFT[3] <= DIN;
   end
  assign DOUT = SHFT[0];
endmodule
```

5.3 移位寄存器设计

5.3.3 使用移位操作符设计移位寄存器

V >>> n 或 V <<< n

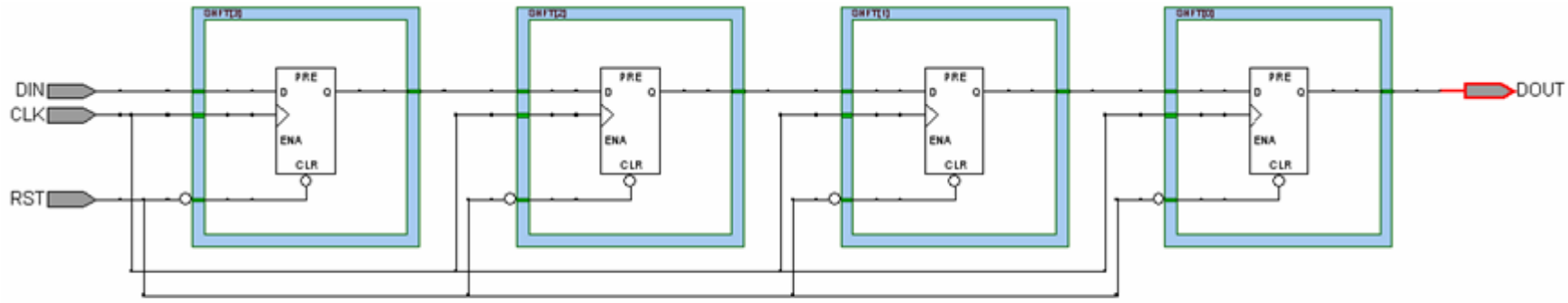

图 5-11　4 位右移移位寄存器 RLT 图

5.3 移位寄存器设计

5.3.3 使用移位操作符设计移位寄存器

```
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

```
output signed[7:0] y;
 input signed[7:0] a;
assign y = (a>>>2);
```

若 a=10101011,则输出 y=11101010

若 a=10001111,则输出 y=00011010

```
output signed [7:0] y;
  parameter c=8'sb10101011;
  assign y = (c >>> 2);
```

```
output [7:0] y;
 parameter c=8'sb10101011;
 assign y = (c >>> 2);
```

5.4 乘法器设计及相关语句应用

5.4.1 参数定义关键词parameter

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 ;
```

5.4 乘法器设计及相关语句应用

5.4.2 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=471;       // 整数完整赋值，因为 A 有 32 位
  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
```

5.4 乘法器设计及相关语句应用

5.4.3 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

5.4 乘法器设计及相关语句应用

【例 5-18】

```
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 5-19】

```
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
```

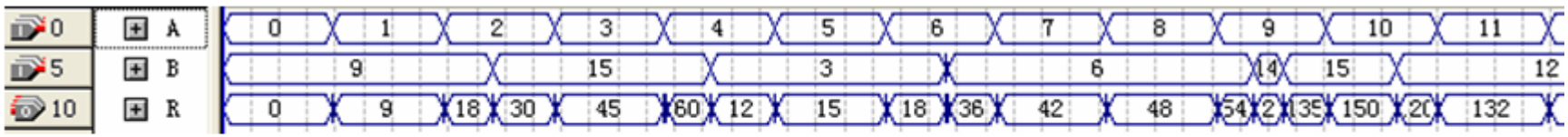

图 5-12　四位乘法器时序仿真图

5.4 乘法器设计及相关语句应用

5.4.4 repeat语句用法

repeat（循环次数表达式）

begin　循环体语句结构　end

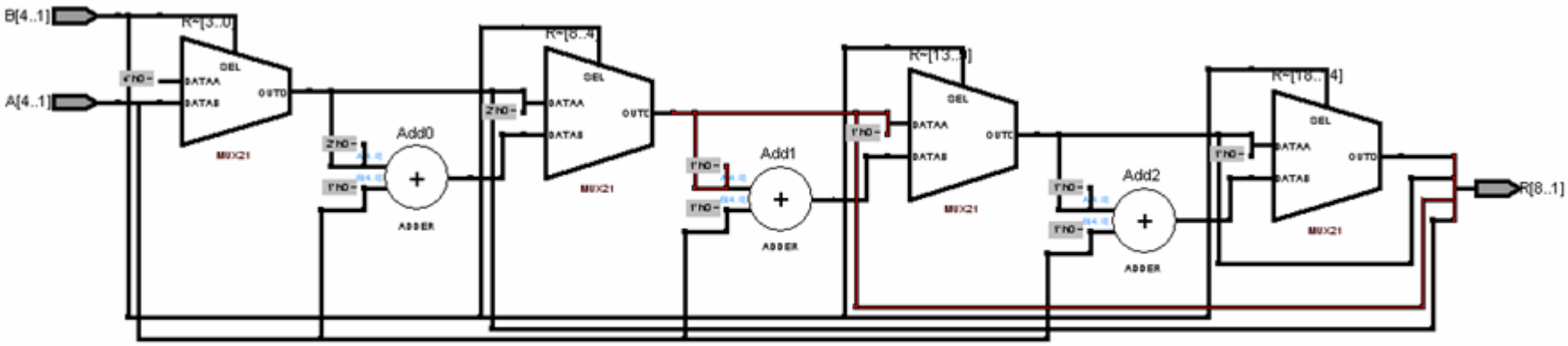

图 5-13　四位乘法器 RLT 图

5.4 乘法器设计及相关语句应用

5.4.5 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 5-20】

```
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 5-21】

```
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
```

5.4 乘法器设计及相关语句应用

5.4.5 while语句用法

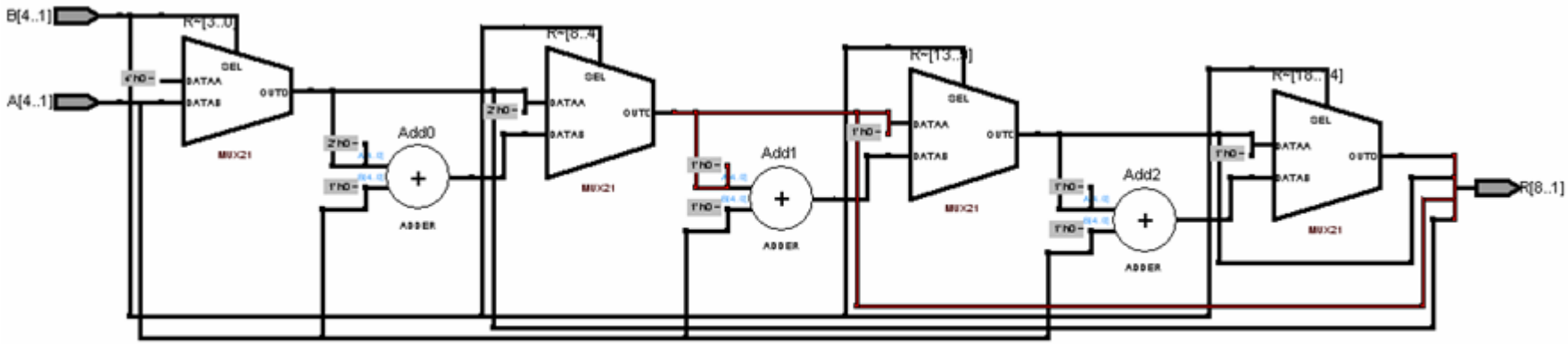

图 5-13　四位乘法器 RLT 图

5.5 if语句一般用法

```
（1）if（条件表达式）  begin  语句块；  end
（2）if（条件表达式）  begin  语句块1；  end
                else   begin  语句块2；  end
（3）if（条件表达式1）                  begin  语句块1；  end
          else if  （条件表达式2）  begin  语句块2；  end
              .
              .
              .
          else if  （条件表达式n）  begin  语句块n；  end
                              else   begin  语句块n+1；end
```

if (en==1) 可简为if (en)；if ((a&b) == 1'B1) 可简为 if (a&b) 。

```
(DIN[7]==1) & (DIN[6]==1) & (DIN[5]==1) & (DIN[4]==1) & (DIN[3]==1) &
(DIN[2]==1) & (DIN[1]==1) & (DIN[0]==0)          //这恰好与表6-1最后一行吻合。
```

【例 5-22】

```
module mux4_1(DIN,DOUT);
  output [0:2] DOUT;
  input [0:7] DIN;
  reg [0:2] DOUT;
  always @(DIN)  begin
    casez (DIN)
    8'b???????0 : DOUT<=3'b000;
    8'b??????01 : DOUT<=3'b100;
    8'b?????011 : DOUT<=3'b010;
    8'b????0111 : DOUT<=3'b110;
    8'b???01111 : DOUT<=3'b001;
    8'b??011111 : DOUT<=3'b101;
    8'b?0111111 : DOUT<=3'b011;
    8'b01111111 : DOUT<=3'b111;
    default : DOUT<=3'b000;
     endcase      end    endmodule
```

【例 5-23】

```
module mux4_1(DIN,DOUT);
  output [0:2] DOUT;
  input [0:7] DIN;
  reg [0:2] DOUT;
  always @(DIN)
   begin
      if (DIN[7]==0)  DOUT=3'b000;
else if (DIN[6]==0)  DOUT=3'b100;
else if (DIN[5]==0)  DOUT=3'b010;
else if (DIN[4]==0)  DOUT=3'b110;
else if (DIN[3]==0)  DOUT=3'b001;
else if (DIN[2]==0)  DOUT=3'b101;
else if (DIN[1]==0)  DOUT=3'b011;
else                 DOUT=3'b111;
    end
endmodule
```

5.5 if语句一般用法

表 5-1　8 线-3 线优先编码器真值表

输　入								输　出		
din0	din1	din2	din3	din4	din5	din6	din7	output0	output1	output2
x	x	x	x	x	x	x	0	0	0	0
x	x	x	x	x	x	0	1	1	0	0
x	x	x	x	x	0	1	1	0	1	0
x	x	x	x	0	1	1	1	1	1	0
x	x	x	0	1	1	1	1	0	0	1
x	x	0	1	1	1	1	1	1	0	1
x	0	1	1	1	1	1	1	0	1	1
0	1	1	1	1	1	1	1	1	1	1

注：表中的“x”为任意

5.5 if语句一般用法

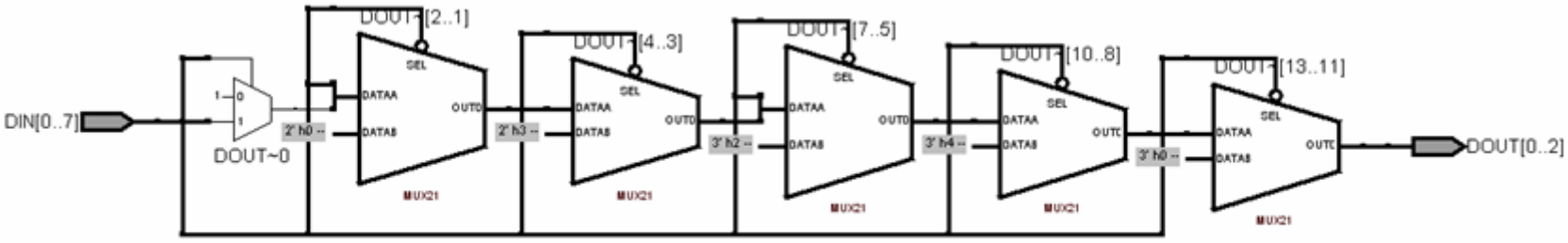

图 5-14 例 5-23 的 RTL 图

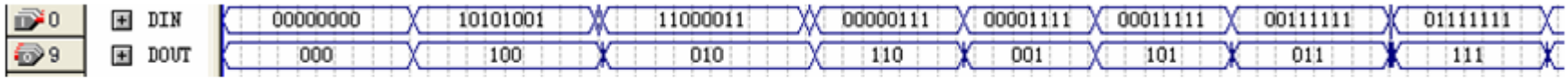

图 5-15 例 5-22 和例 5-23 的时序仿真波形图

5.5 if语句一般用法

【例 5-24】

```
module andd(A,B,Q);
  output Q ;    input A,B;
  reg Q;
  always @(A,B )
   if (A==0)
   begin   if (B==0)  Q=0;
  else  Q=1; end
  endmodule
```

【例 5-25】

```
module andd(A,B,Q);
 output Q ;     input A,B;
  reg Q;
  always @(A,B )
   if (A==0)
   begin if (B==0)  Q=0;  end
      else     Q=1;
 endmodule
```

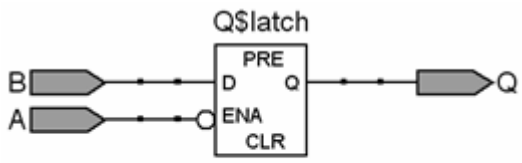

图 5-16 例 5-24 的 RTL 图

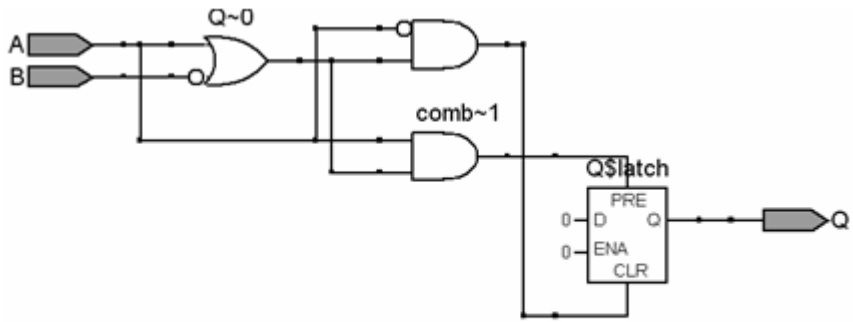

图 5-17 例 5-25 的 RTL 图

5.6 三态与双向端口设计

5.6.1 三态控制电路设计

【例 5-26】

```
module tri4B (ENA,DIN,DOUT);
   input ENA;
   input [3:0] DIN ;
   output [3:0] DOUT ;
   reg [3:0] DOUT;
   always @(DIN,ENA)  begin
       if (ENA)  DOUT <= DIN ;
           else  DOUT <= 4'HZ;
   end
endmodule
```

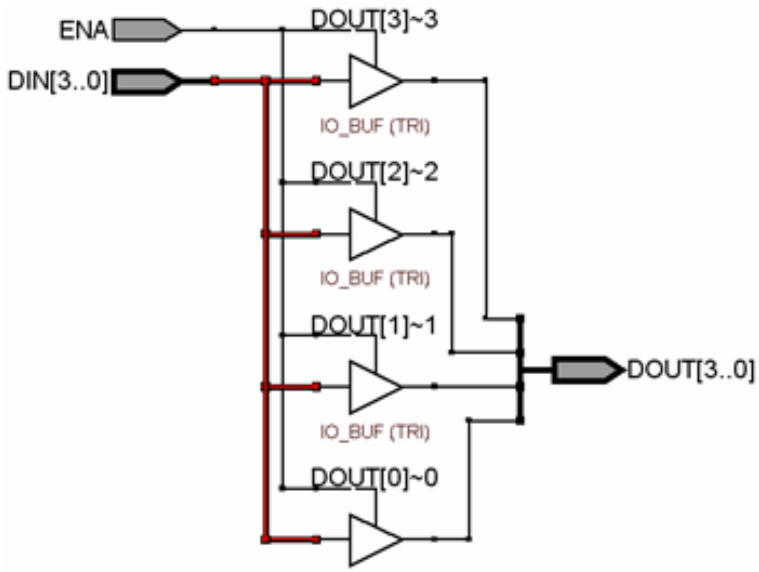

图 5-18　4 位三态控制门电路

5.6 三态与双向端口设计

5.6.2 双向端口设计

【例 5-27】

```
module bi4b(TRI_PORT,DOUT,DIN,ENA,CTRL);
   inout TRI_PORT;
   input DIN,ENA,CTRL;
   output DOUT ;
     assign TRI_PORT = ENA ? DIN : 1'bz;
     assign DOUT = TRI_PORT | CTRL;
endmodule
```

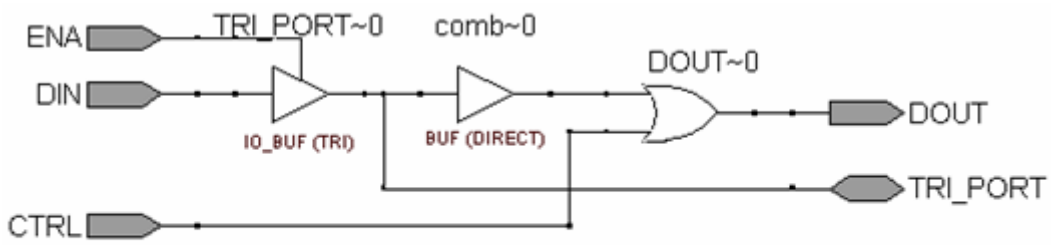

图 5-19　例 5-27 的 1 位双向端口电路设计之 RTL 图

5.6 三态与双向端口设计

【例 5-28】

```
module BI4B(CTRL,DIN,Q,DOUT);
  input CTRL;     input [3:0] DIN;
  inout [3:0] Q; output [3:0] DOUT;
  reg [3:0] DOUT,Q ;
  always @(Q,DIN,CTRL)  begin
     if (!CTRL)  DOUT<=Q ;
       else   begin
       Q<=DIN ; DOUT<=4'HZ;
    end   end
endmodule
```

【例 5-29】

```
module BI4B(CTRL,DIN,Q,DOUT);
 input CTRL;     input[3:0] DIN;
 inout[3:0] Q; output[3:0] DOUT;
 reg [3:0] DOUT,Q ;
always @(Q,DIN,CTRL)   begin
if (!CTRL)  begin DOUT <= Q  ;
      Q<=4'HZ; end
 else begin Q<=DIN;  DOUT<=4'HZ;
end  end
endmodule
```

图 5-20 例 5-28 的仿真波形图

5.6 三态与双向端口设计

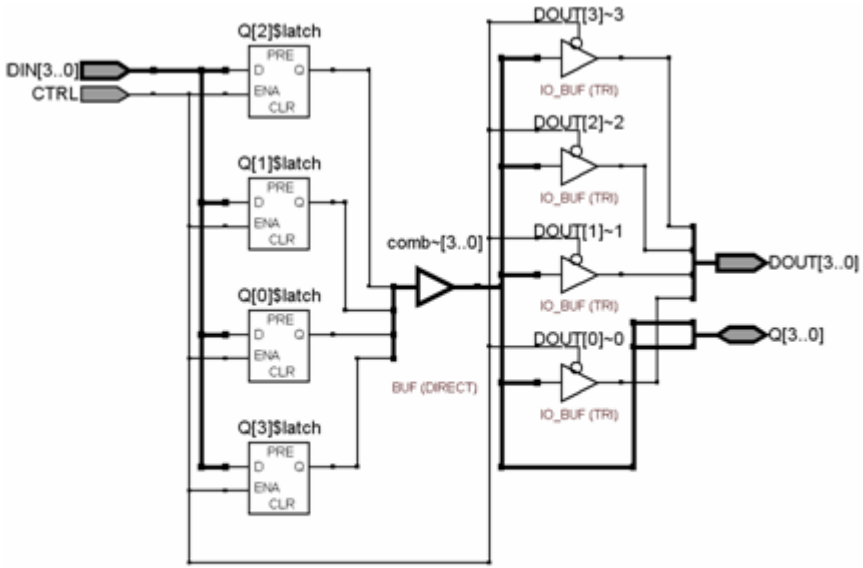

图 5-21 例 5-28 的 RTL 图

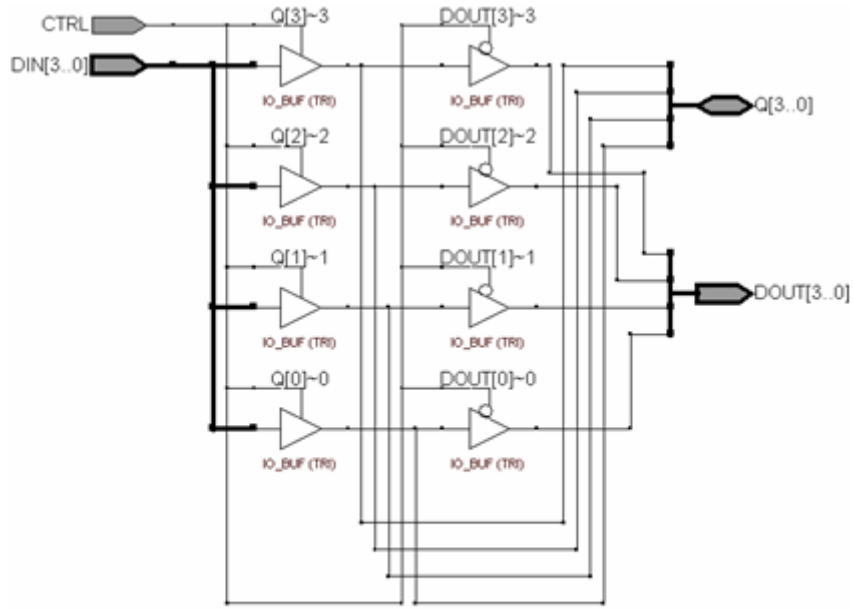

图 5-22 例 5-29 的 RLT 图

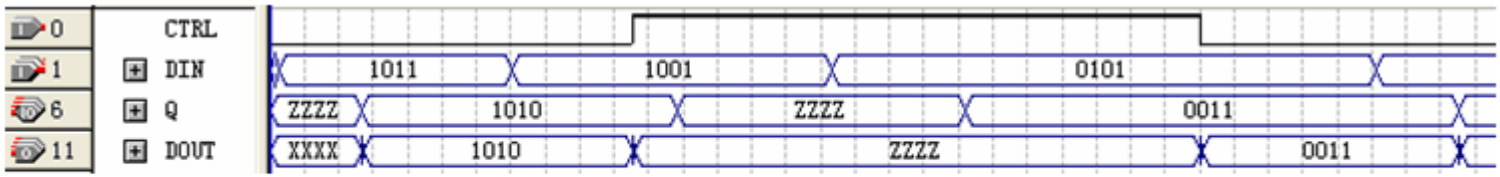

图 5-23 例 5-29 的仿真波形图

5.6.3 三态总线控制电路设计

【例 5-30】

```
module triBUS4(
  IN3,IN2,IN1,IN0,ENA,DOUT);
  input[3:0] IN3,IN2,IN1,IN0 ;
  input[1:0]  ENA;
  output[3:0] DOUT;
  reg[3:0] DOUT;
  always @(ENA,IN3,IN2,IN1,IN0)
     begin
    if (ENA==0)  DOUT=IN3 ;
        else  DOUT=4'HZ;
    if (ENA==1)  DOUT=IN2 ;
        else  DOUT=4'HZ;
   if (ENA==2)  DOUT=IN1 ;
        else  DOUT=4'HZ;
   if (ENA==3)  DOUT=IN0 ;
        else  DOUT=4'HZ;
end    endmodule
```

【例 5-31】

```
module triBUS4(
    IN3,IN2,IN1,IN0,ENA,DOUT);
  input[3:0] IN3,IN2,IN1,IN0 ;
  input[1:0]  ENA;
  output[3:0] DOUT;reg[3:0] DOUT;
  always @ (ENA or IN0)
  begin if (ENA==2'b00)  DOUT=IN0;
     else  DOUT=4'hz; end
  always @ (ENA or IN1)
  begin if (ENA==2'b01)  DOUT=IN1;
     else  DOUT=4'hz; end
  always @ (ENA or IN2)
  begin if (ENA==2'b10)  DOUT=IN2;
     else  DOUT=4'hz; end
  always @ (ENA or IN3)
  begin if (ENA==2'b11)  DOUT=IN3;
else  DOUT=4'hz;  end  endmodule
```

5.6 三态与双向端口设计

5.6.3 三态总线控制电路设计

【例 5-32】

```
module mux4_1(IN3,IN2,IN1,IN0,ENA,DOUT);
  input [3:0] IN3,IN2,IN1,IN0; input[1:0]  ENA; output[3:0] DOUT;
    assign DOUT = (ENA==2'B00) ? IN0 : 4'HZ;
    assign DOUT = (ENA==2'B01) ? IN1 : 4'HZ;
    assign DOUT = (ENA==2'B10) ? IN2 : 4'HZ;
    assign DOUT = (ENA==2'B11) ? IN3 : 4'HZ;
endmodule
```

5.6 三态与双向端口设计

5.6.3 三态总线控制电路设计

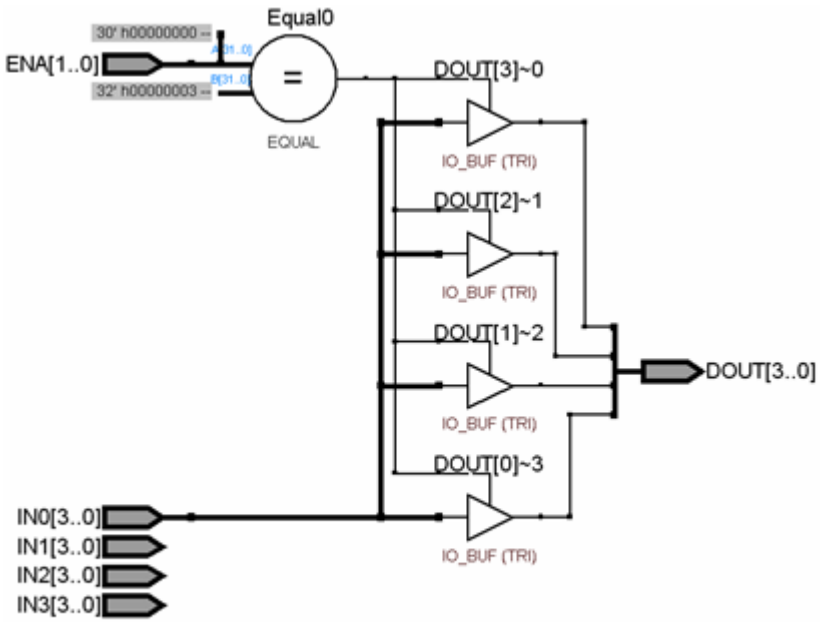

图5-24 例5-30的RTL图

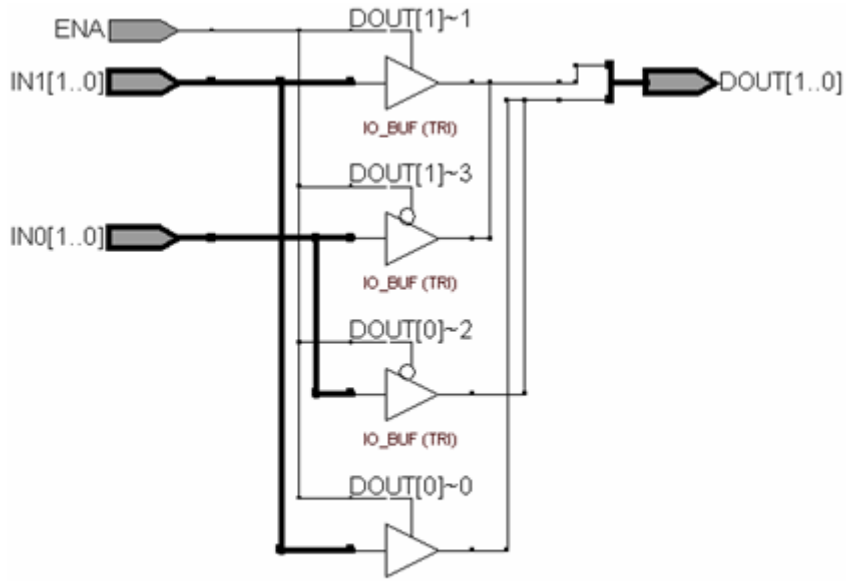

图5-25 例5-31的2位简化RTL图

5.7 模可控计数器设计

5.7.1 同步加载模型设计

【例 5-33】

```
module FDIV0 (CLK,PM,D,DOUT,RST );
   input CLK;  input RST;  input[3:0] D; output PM;  output[3:0] DOUT;
   reg [3:0] Q1;     reg FULL;
  (* synthesis,keep *) wire LD ;//设定 LD 为仿真可测试属性
      always @(posedge CLK or negedge RST  ) begin
        if (!RST)  begin Q1<=0 ; FULL<=0; end
          else if ( LD ) begin Q1<=D ; FULL<=1; end
             else  begin Q1 <= Q1+1; FULL<=0 ; end   end
     assign LD=( Q1==4'B1111 ); assign PM=FULL ; assign DOUT=Q1 ;
endmodule
```

5.7 模可控计数器设计

5.7.1 同步加载模型设计

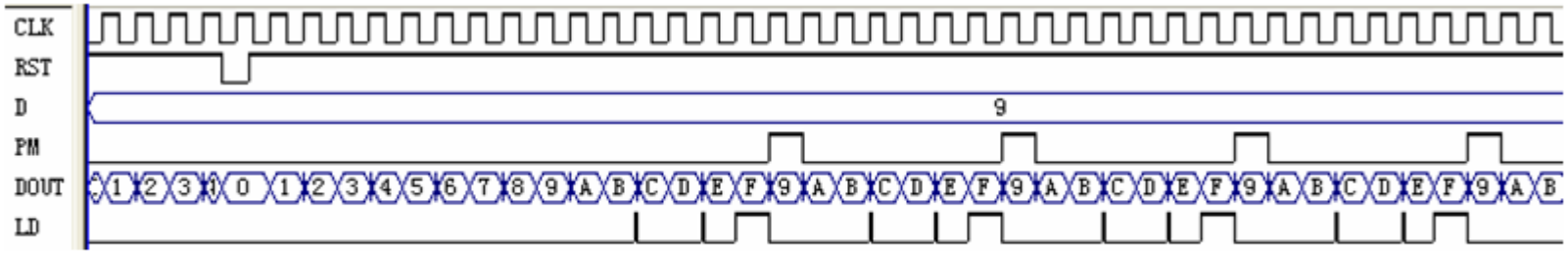

图 5-26 例 5-33 的仿真波形，同步加载模式（波形中有毛刺）

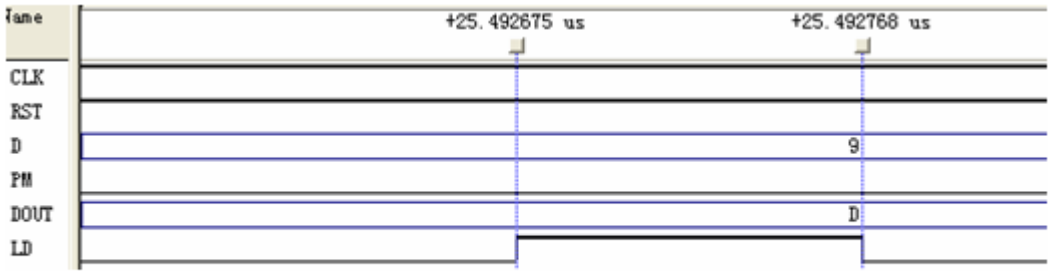

图 5-27 图 5-26 中毛刺脉冲展开后的时序，毛刺脉宽约 0.1ns

5.7 模可控计数器设计

5.7.2 异步加载模型设计

【例 5-34】

```
module fdiv1 (CLK,PM,D,DOUT,RST );//引脚锁定同例 5-33
  input CLK;  input RST; input[3:0] D;  output PM; output[3:0] DOUT;
  reg [3:0] Q1;   reg FULL;
(* synthesis,probe_port,keep *)  wire LD ;
    always @(posedge CLK or posedge LD or negedge RST  ) begin
       if (!RST)  begin Q1<=0 ; FULL<=0; end  else
       if ( LD ) begin Q1<=D ; FULL<=1;  end
         else  begin Q1 <= Q1+1; FULL<=0 ; end  end
     assign LD=( Q1==4'B0000 ) ; assign PM=FULL;  assign DOUT=Q1 ;
endmodule
```

5.7 模可控计数器设计

5.7.2 异步加载模型设计

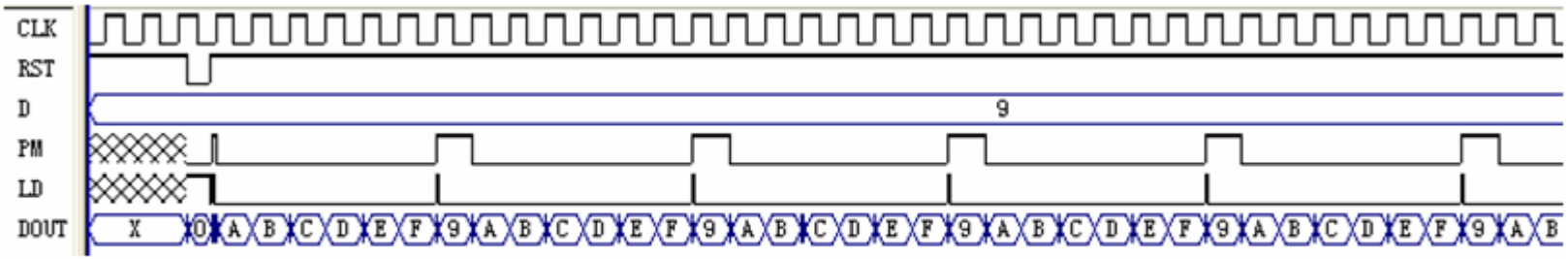

图 5-28 例 5-34 的仿真波形，异步加载模式

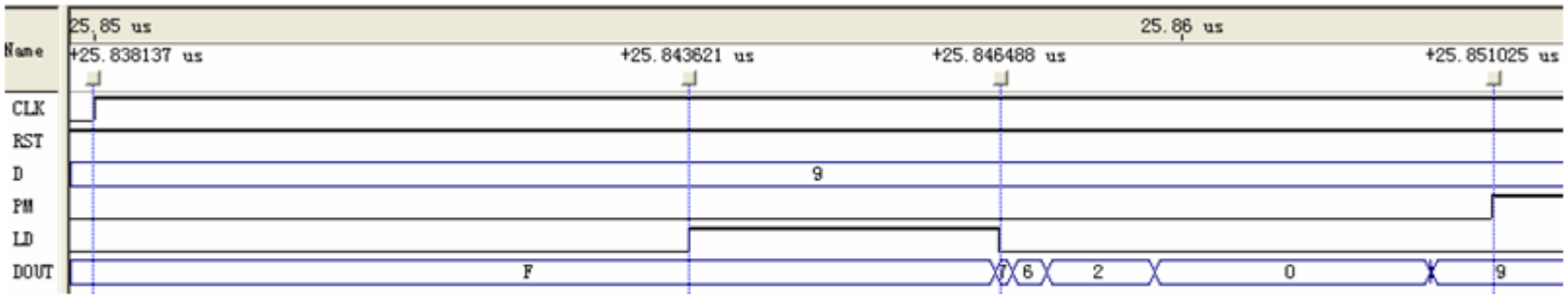

图 5-29 图 5-28 中的 LD 信号展开后的时序，LD 脉宽约 2.8ns

5.7 模可控计数器设计

5.7.3 异步清0加载模型设计

【例 5-35】

```
module fdiv1 (CLK,PM,D ); //引脚锁定同例 5-33
  input CLK; input[3:0] D;    output PM ;
  (* synthesis,probe_port,keep *)  reg [3:0] Q1;   reg FULL;
  (* synthesis,probe_port,keep *)  wire RST ;
    always @(posedge CLK or posedge RST) begin
       if (RST) begin Q1<=0; FULL<=1;  end
         else  begin Q1 <= Q1+1; FULL<=0 ;  end   end
     assign RST = ( Q1==D ) ; assign PM = FULL ;
  endmodule
```

5.7 模可控计数器设计

5.7.3 异步清0加载模型设计

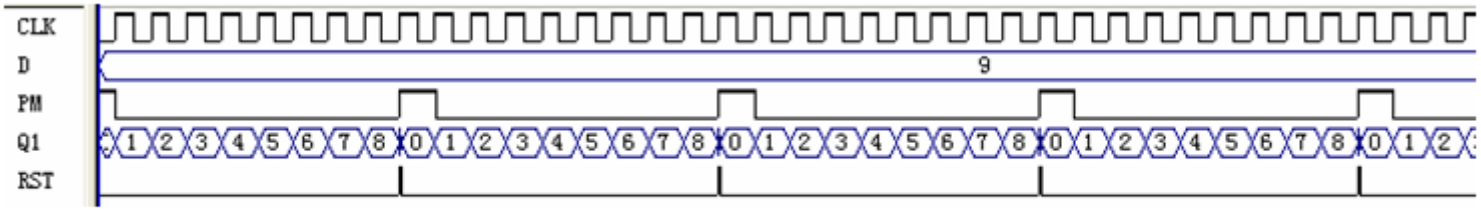

图 5-30　例 5-35 的仿真波形，异步清 0 模式

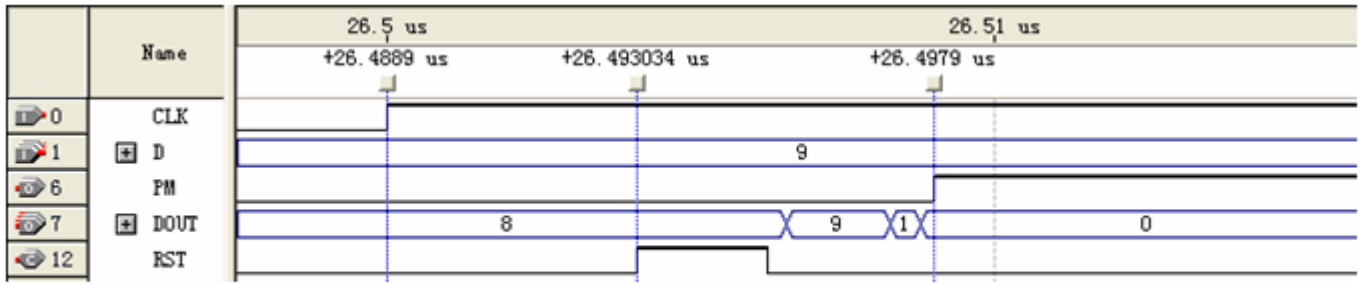

图 5-31　图 5-30 中的 RST 信号展开后的时序

5.7 模可控计数器设计

5.7.4 同步清0加载模型设计

【例 5-36】程序其余部分同例 5-35

```
always @(posedge CLK) begin
```

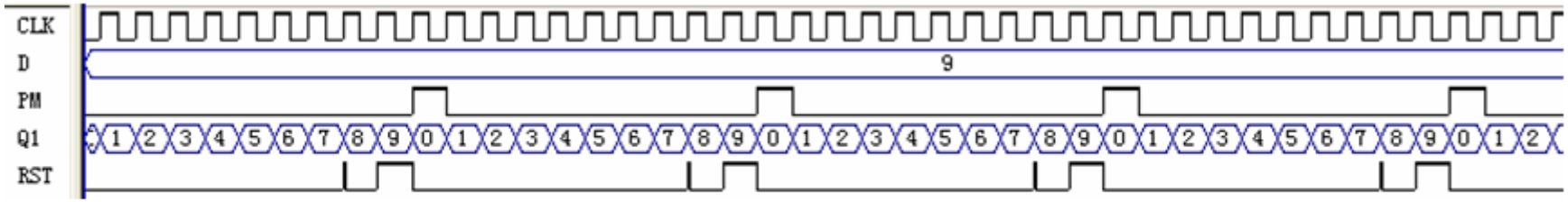

图 5-32 例 5-36 的波形，同步清 0 模式

5.8 半整数与奇数分频电路设计

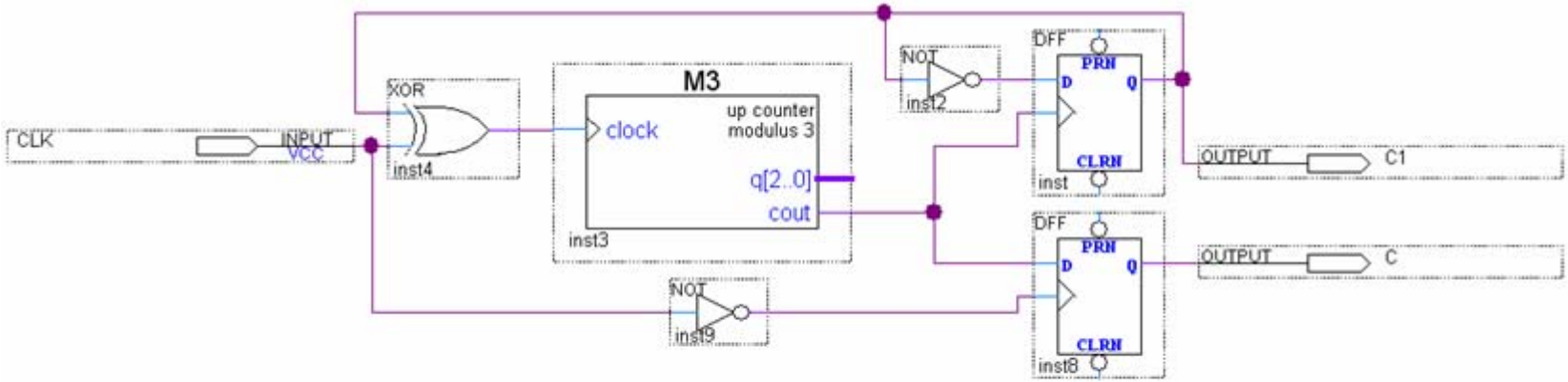

图 5-33 占空比为 50%的任意奇数次分频电路

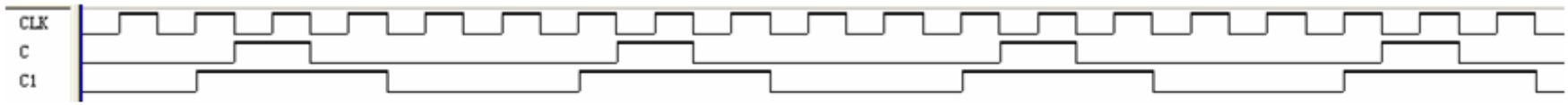

图 5-34 图 5-33 电路的仿真波形

5.8 半整数与奇数分频电路设计

【例 5-37】 占空比为 50%的任意奇数次 5 分频电路

```
module FDIV3 (CLK,K_OR,K1,K2);
input CLK;     output K_OR,K1,K2;
 reg[2:0] C1,C2;   reg M1, M2;
always @(posedge CLK)  begin
  if(C1==4)  C1<=0;   else C1<=C1+1;
  if(C1==1) M1<=~M1;  else if(C1==3) M1=~M1;  end
always @(negedge CLK)  begin
  if(C2==4)  C2<=0;    else C2<=C2+1;
  if(C2==1)  M2<=~M2;  else if(C2==3) M2=~M2;  end
assign  K1 = M1;  assign K2 = M2;
assign  K_OR = M1 | M2;
endmodule
```

5.8 半整数与奇数分频电路设计

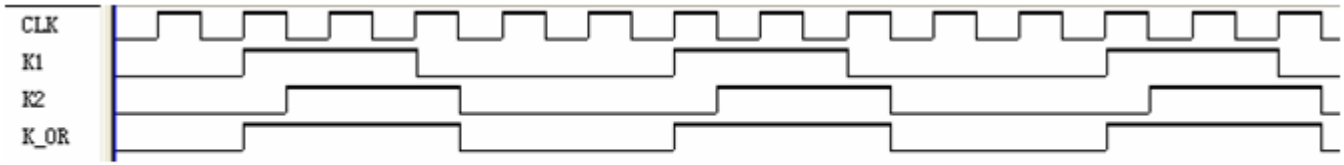

图 5-35 占空比为 50%的任意奇数次分频电路

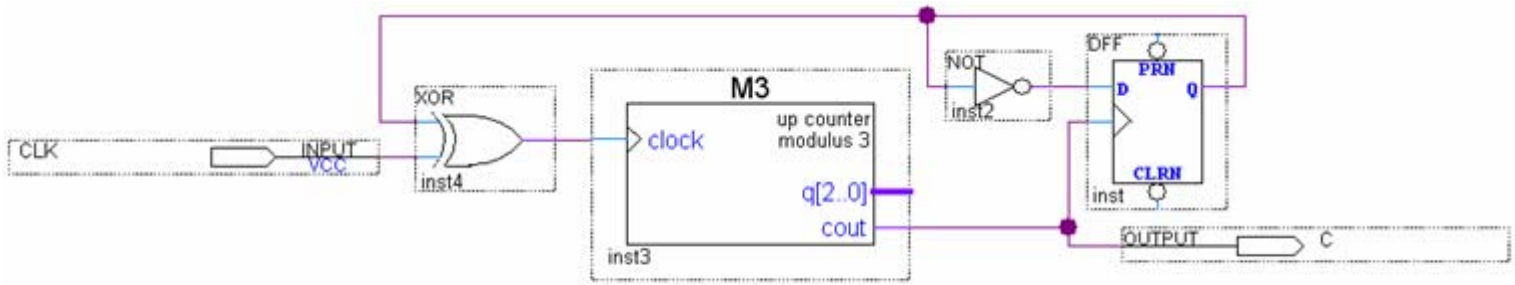

图 5-36 任意半整数分频电路

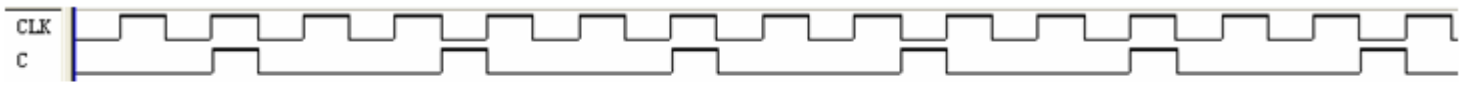

图 5-37 图 5-36 电路仿真波形图

5.9 Verilog的描述风格

5.9.1 RTL描述

5.9.2 行为描述

【例 5-38】

```
MODULE counter_up
    Clock ,reset,                PIN ;
    Counter7..counter0           PIN  ISTYPE  'COM' ;
    Cnt_ff7..cnt_ff0             NODE ISTYPE  'REG' ;
    Counter = [counter7..counter0];    Cnt = [cnt_ff7..cnt_ff0];
   EQUATIONS
    Cnt.CLK = clock ;        Cnt.AR  = reset    ;
    Cnt :=  cnt.FB + 1 ;     Counter = cnt ;      END counter_up
```

5.9 Verilog的描述风格

5.9.3 数据流描述

5.9.4 结构描述

实验与设计

5-1 半整数与奇数分频器设计

5-2 模可控计数器设计

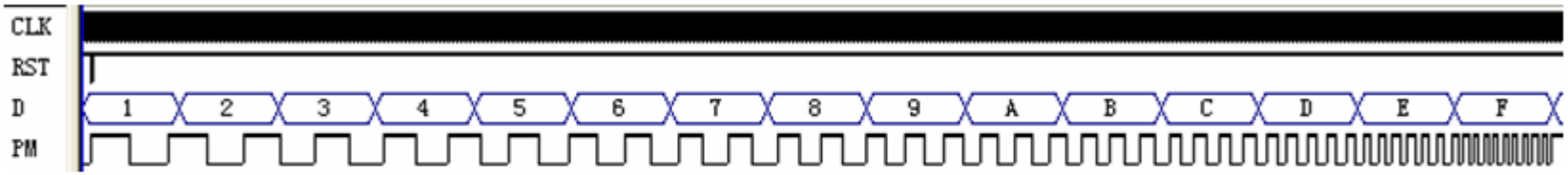

图 5-38 针对不同预置数，占空比均衡后的分频器输出

实验与设计

5-3 VGA彩条信号显示控制电路设计

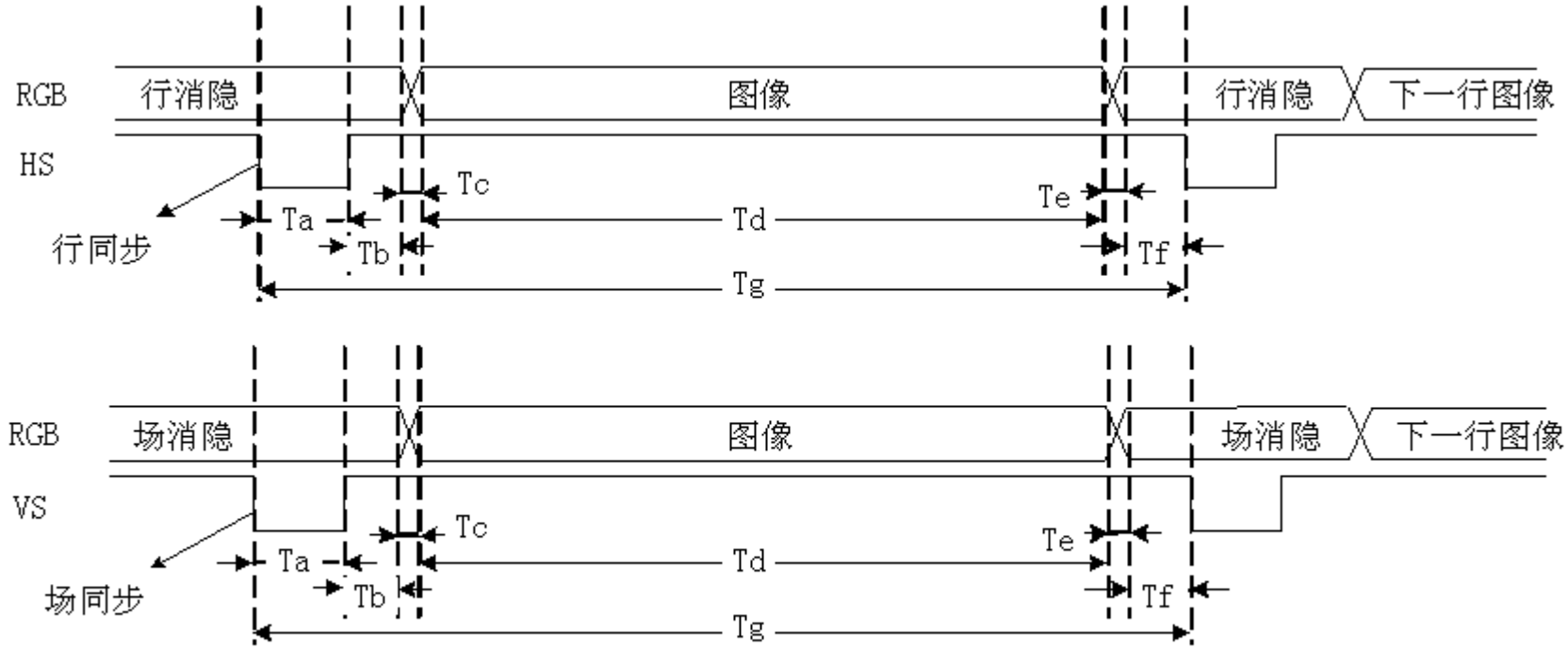

图 5-39　VGA 行扫描、场扫描时序示意图

实验与设计

5-3 VGA彩条信号显示控制电路设计

表 5-2 行扫描时序要求：(单位：像素，即输出一个像素 Pixel 的时间间隔)

		行同步头			行图像		行周期
对应位置	Tf	Ta	Tb	Tc	Td	Te	Tg
时间(Pixels)	8	96	40	8	640	8	800

表 5-3 场扫描时序要求：(单元：行，即输出一行 Line 的时间间隔)

		行同步头			行图像		行周期
对应位置	Tf	Ta	Tb	Tc	Td	Te	Tg
时间(Lines)	2	2	25	8	480	8	525

实验与设计

5-3 VGA彩条信号显示控制电路设计

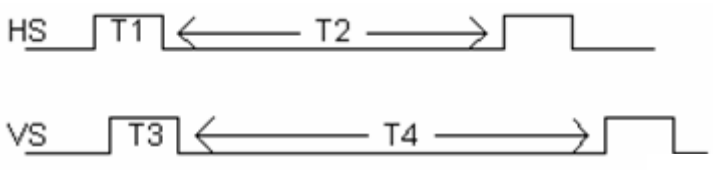

图 5-40 HS 和 VS 的时序图

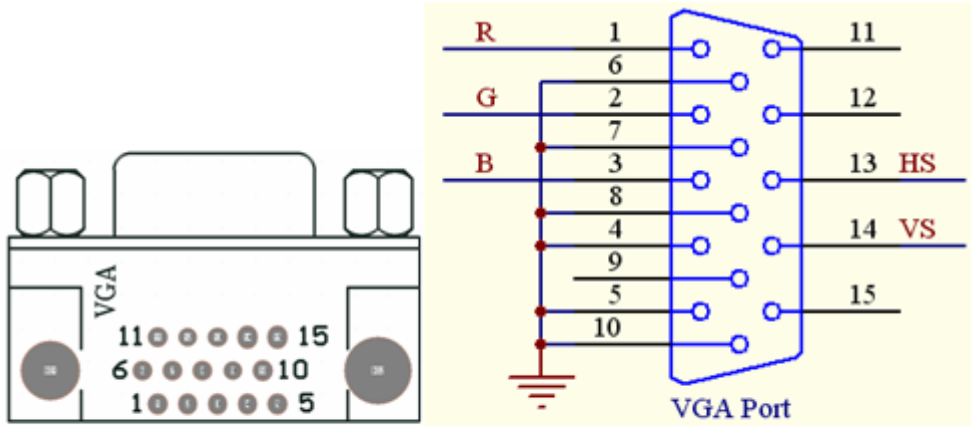

图 5-41 VGA 接口电路图，左接口从上往下看

【例 5-39】

```
module VGA_COLOR_LINE (CLK, MD, HS, VS, R, G, B);//VGA显示器 彩条 发生器
  input CLK, input MD;          output HS, VS, R,  G, B;
    wire R,G,B,VS,HS;                //红，绿，蓝信号，和场同步，行同步信号
    wire FCLK, CCLK;       reg  HS1, VS1;
   reg[1:0] MMD;   reg[4:0] FS;
   reg[4:0] CC;                      //行同步，横彩条生成
   reg[8:0] LL;                      //场同步，竖彩条生成
   reg[3:1] GRBX,GRBY,GRBP;          // X横彩条，Y竖彩条
   wire[3:1] GRB;
   assign GRB[2] = (GRBP[2] ^ MD) & HS1 & VS1 ;
   assign GRB[3] = (GRBP[3] ^ MD) & HS1 & VS1 ;
   assign GRB[1] = (GRBP[1] ^ MD) & HS1 & VS1 ;
   always @(posedge MD)  begin
         if (MMD==2'b10) MMD<=2'b00; else MMD<=MMD+1 ;  end //3种模式
   always @(MMD)   begin
         if (MMD == 2'b00)  GRBP <= GRBX ; // 选择横彩条
     else if (MMD == 2'b01)  GRBP <= GRBY ; // 选择竖彩条
     else if (MMD == 2'b10)  GRBP <= GRBX ^ GRBY ; //产生棋盘格
     else GRBP <= 3'b000 ;    end
```

接下页

接上页

```
always @(posedge CLK ) begin                    // 20MHz 21分频
       if (FS==20)  FS<=0;  else  FS<=(FS+1) ;  end
always @(posedge FCLK)   begin
       if (CC==29)  CC<=0;  else  CC<=CC+1 ;    end
always @(posedge CCLK)   begin
      if (LL==481)   LL<=0; else  LL<=LL+1 ;    end
always @(CC or LL)   begin
      if (CC > 23)  HS1<=1'b0; else HS1<=1'b1 ; //行同步
      if (LL > 479) VS1<=1'b0; else VS1<=1'b1 ;     end //场同步
always @(CC or LL)   begin
   if (CC < 3)  GRBX <= 3'b111 ; // 横彩条
   else if (CC < 6)  GRBX <= 3'b110 ;
   else if (CC < 9)  GRBX <= 3'b101 ;
   else if (CC < 12) GRBX <= 3'b100 ;
   else if (CC < 15) GRBX <= 3'b011 ;
   else if (CC < 18) GRBX <= 3'b010 ;
   else if (CC < 21) GRBX <= 3'b001 ;
   else  GRBX <= 3'b000 ;
   if (LL < 60)  GRBY <= 3'b111 ; // 竖彩条
   else if (LL < 120)  GRBY <= 3'b110 ;
   else if (LL < 180)  GRBY <= 3'b101 ;
```

接下页

接上页

```
    else if (LL < 240)  GRBY <= 3'b100 ;
    else if (LL < 300)  GRBY <= 3'b011 ;
    else if (LL < 360)  GRBY <= 3'b010 ;
    else if (LL < 420)  GRBY <= 3'b001 ;
    else    GRBY <= 0 ;    end
  assign HS = HS1 ;    assign FCLK = FS[3] ;
  assign HS = HS1 ;    assign VS = VS1   ;  assign R = GRB[2] ;
  assign G = GRB[3] ;  assign B = GRB[1] ;  assign CCLK = CC[4] ;
endmodule
```

表 5-4 颜色编码：

颜色	黑	蓝	红	品	绿	青	黄	白
R	0	0	0	0	1	1	1	1
G	0	0	1	1	0	0	1	1
B	0	1	0	1	0	1	0	1

表 5-5 彩条信号发生器 3 种显示模式，

1	横彩条	1：白黄青绿品红蓝黑	2：黑蓝红品绿青黄白
2	竖彩条	1：白黄青绿品红蓝黑	2：黑蓝红品绿青黄白
3	棋盘格	1：棋盘格显示模式 1	2：棋盘格显示模式 2

实验与设计

5-4 移位相加型8位硬件乘法器设计

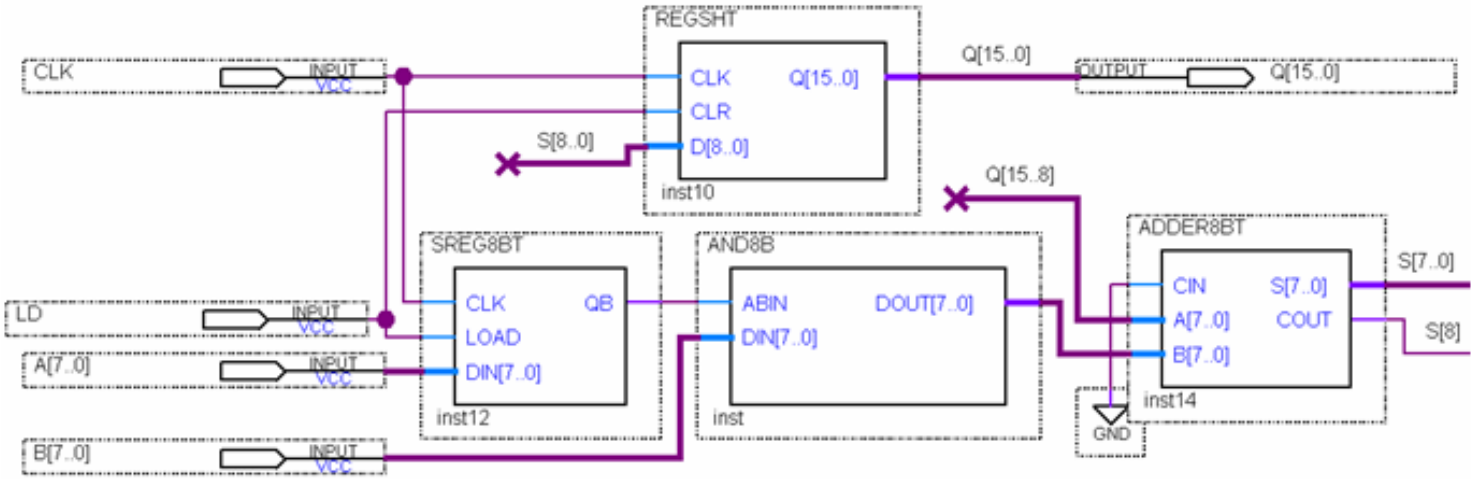

图 5-42　8 位乘法器逻辑原理图

图 5-43　8 位移位相加乘法器运算逻辑波形图

实验与设计

5-4 移位相加型8位硬件乘法器设计

【例 5-40】

```
module REGSHT (CLK, CLR, D, Q);
   input CLK, CLR;  input[8:0] D;  output[15:0] Q;
   wire[15:0] Q;   reg[15:0] R16S;
   always @(posedge CLK or posedge CLR)  begin
      if (CLR==1'b1) R16S<=16'H0000; //时钟到来时，锁存输入值，并右移低 8 位
      else   begin R16S[6:0]<=R16S[7:1]; R16S[15:7]<=D;   end
      end     assign Q = R16S ;
 endmodule
```

实验与设计

5-5 移位寄存器设计

5-6 串行静态显示控制电路设计

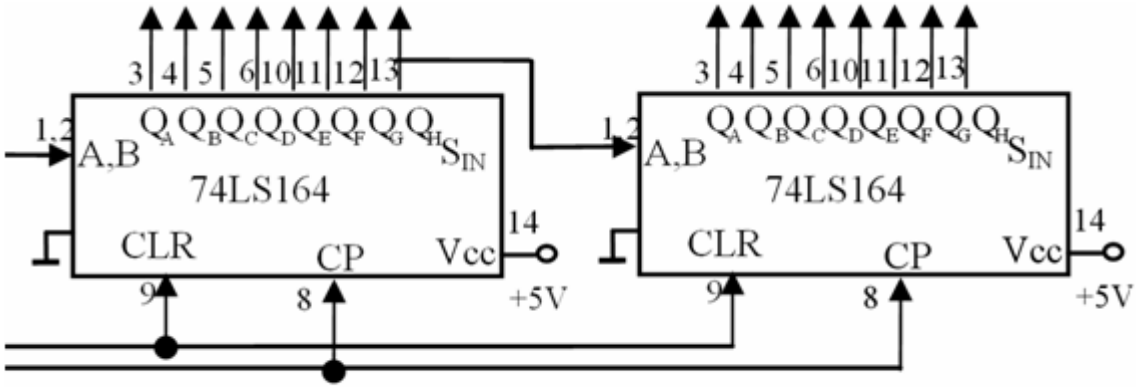

图 5-44 串/并转换数码管静态显示电路