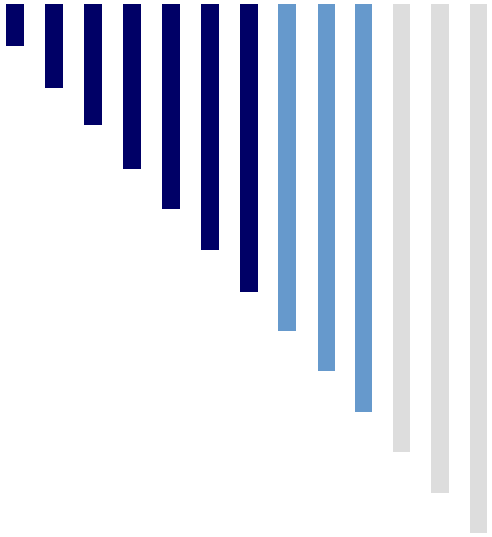


第6章

EDA技术深入

6.1 过程中的两类赋值语句

6.1.1 未指定延时的阻塞式赋值语句

```
目标变量名 = 驱动表达式;
```

1、阻塞本过程中其他语句的执行，计算出“驱动表达式”的值；
2、向“目标变量”进行赋值操作（假设没有指定延时）；
3、完成赋值，即实现目标变量的更新，允许对本过程中其他语句的执行。

6.1 过程中的两类赋值语句

6.1.2 指定了延时的阻塞式赋值

[延时] 目标变量名 = 驱动表达式;

目标变量名 = [延时] 驱动表达式;

【例 6-1】	【例 6-2】

.....//过程语句
Y1 = A^B;
#6 Y2 = A&B|C;
	..... //过程语句
	Y1 = A^D;
	Y2 = #6 A&E|C;

6.1 过程中的两类赋值语句

6.1.3 未指定延时的非阻塞式赋值

目标变量名 <= 驱动表达式;

【例6-3】

```
assign  Q1 = A|B;
assign  Q1 = B&C;
assign  Q1 = ~C;
```

【例6-4】

```
begin
Q1 = A|B;
Q1 = B&C;
Q1 = ~C ;    end
```

【例6-5】

```
begin
Q1 <= A|B;
Q1 <= B&C;
Q1 <= ~C ;    end
```

设进程启动后，A=2'b10，B=2'b01，C=2'b11

6.1 过程中的两类赋值语句

6.1.3 未指定延时的非阻塞式赋值

【例6-6】	【例6-7】

always @(A,B) begin
M1 = A ; //更新结果：M1=1
M2 = B&M1; //更新结果：M2=1&1=1
Q=M1|M2;end //更新结果：M2=1|1=1
	always @(A,B) begin
	M1 <= A ; //更新结果：M1=1
	M2 <= B&M1; //更新结果：M2=1&0=0
	Q<=M1|M2;end //更新结果：Q=0|0=0

6.1 过程中的两类赋值语句

6.1.4 指定了延时的非阻塞式赋值

[延时] 目标变量名 <= 驱动表达式;

目标变量名 <= [延时] 驱动表达式;

【例 6-8】	【例 6-9】	【例 6-10】
begin Y1 = #6 A^B; Y2 = #4 A|B; Y3 = #7 A&B; end	begin Y1 <= #6 A^B; Y2 <= #4 A|B; Y3 <= #7 A&B; end	begin Y1 = #5 A^B; Y2 <= #3 A|B; Y3 <= #2 A&B; Y4 = #4 (~B); end

6.1 过程中的两类赋值语句

6.1.5 阻塞与非阻塞式赋值特点的深入讨论

【例6-11】使用非阻塞赋值符的时序模块

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

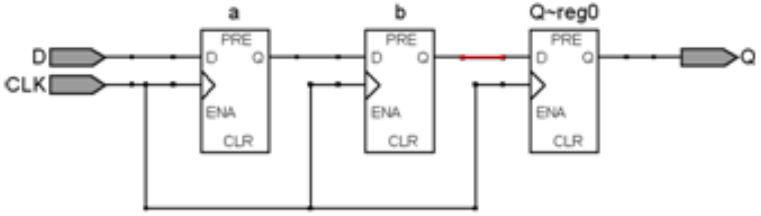


图6-1 例6-11综合后的RTL电路

【例6-12】使用阻塞赋值符的时序模块

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

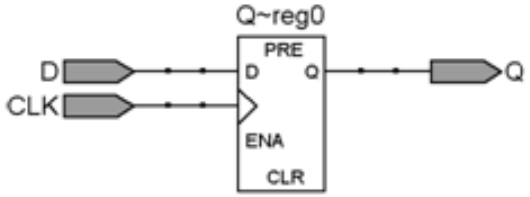


图6-2 例6-12综合后的RTL电路

6.1 过程中的两类赋值语句

6.1.6 不同赋值方式的信号赋初值导致不同综合结果

【例6-13】

```
module MUX41a(D,S,DOUT);
  output DOUT ;
  input [3:0] D; input [1:0] S;
  integer T;  reg DOUT;
  always @(D,S)
  begin   T <= 0;
    if (S[0]==1)   T<=T+1;
    if (S[1]==1)   T<=T+2;
     case (T)
        0 : DOUT = D[0] ;
        1 : DOUT = D[1] ;
        2 : DOUT = D[2] ;
        3 : DOUT = D[3] ;
     default : DOUT = D[0] ;
     endcase   end
endmodule
```

【例6-14】

```
module MUX41a(D,S,DOUT);
  output DOUT ;
  input [3:0] D; input [1:0] S;
  integer T; reg DOUT;
  always @(D,S)
  begin   T = 0;
    if (S[0]==1)   T=T+1;
    if (S[1]==1)   T=T+2;
     case (T)
        0 : DOUT = D[0] ;
        1 : DOUT = D[1] ;
        2 : DOUT = D[2] ;
        3 : DOUT = D[3] ;
     default : DOUT = D[0] ;
     endcase   end
endmodule
```

6.1 过程中的两类赋值语句

6.1.6 不同赋值方式的信号赋初值导致不同综合结果

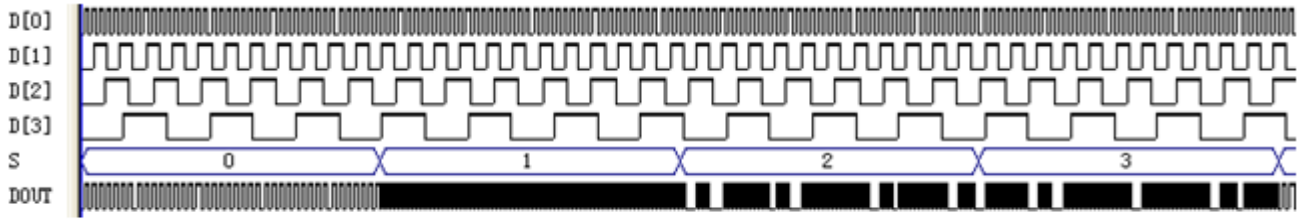


图 6-3 例 6-13 的错误工作时序

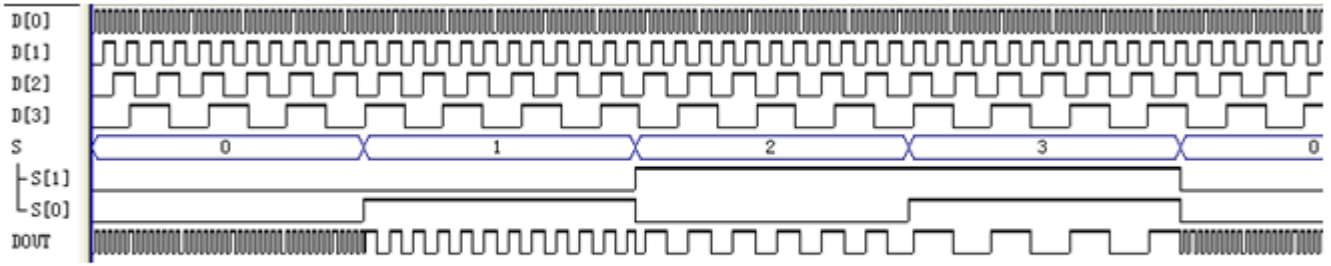


图 6-4 例 6-14 的正确工作时序

6.2 过程语句使用深入探讨

6.2.1 过程语句应用总结

```
assign DOUT = a & b;
```

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

6.2 过程语句使用深入探讨

6.2.2 深入认识不完整条件语句与时序电路的关系

【例6-15】

```
module mux2_1
  (CLK, D, Q, RST);
   output Q;
   input CLK,D,RST;
   reg Q;
 always @(D, CLK, RST)
   if(CLK)  Q <= D;
      else  Q <= RST;
endmodule
```

【例6-16】

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

【例6-17】

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

6.2 过程语句使用深入探讨

6.2.2 深入认识不完整条件语句与时序电路的关系

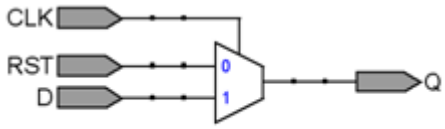


图 6-5 例 6-15 的 RTL 图

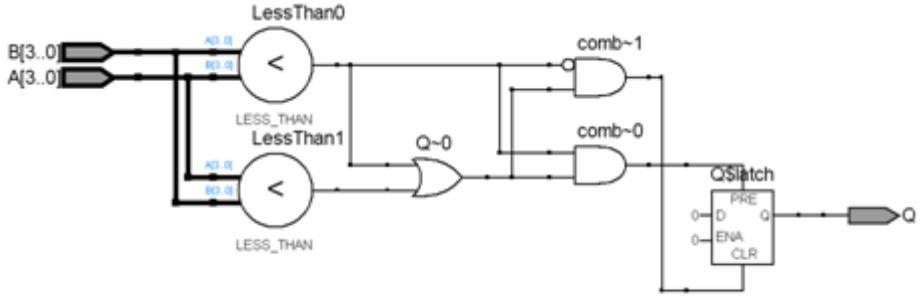


图 6-6 例 6-16 的 RTL 图，输出口被加上了锁存器

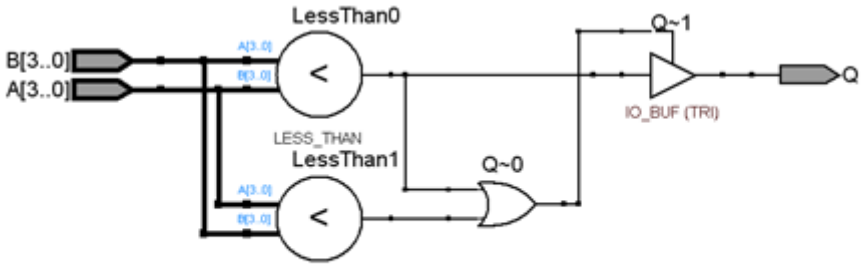


图 6-7 例 6-17 的 RLT 电路图，输出口没有锁存器，是纯组合电路

6.3 更完整地认识if语句

6.3.1 if语句的一般表述形式

```
(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
```

6.3 更完整地认识if语句

6.3.1 if语句的一般表述形式

【例6-18】

```
module CODER83 (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6-19】

```
module CODER83 (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
```

6.3 更完整地认识if语句

6.3.1 if语句的一般表述形式

表 6-1　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

6.3 更完整地认识if语句

6.3.1 if语句的一般表述形式

```
(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。
```

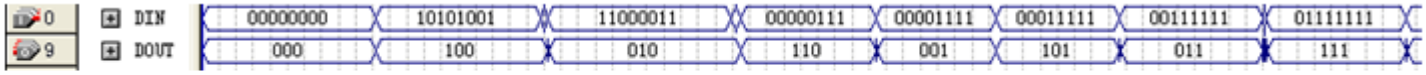


图 6-8 例 6-18 和例 6-19 的时序仿真波形图

6.3 更完整地认识if语句

6.3.2 关注if语句中的条件指示

【例6-20】

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

【例6-21】

```
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6-22】

```
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
```

6.3 更完整地认识if语句

6.3.2 关注if语句中的条件指示

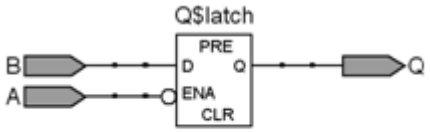


图 6-9 例 6-20/6-21 的 RTL 图

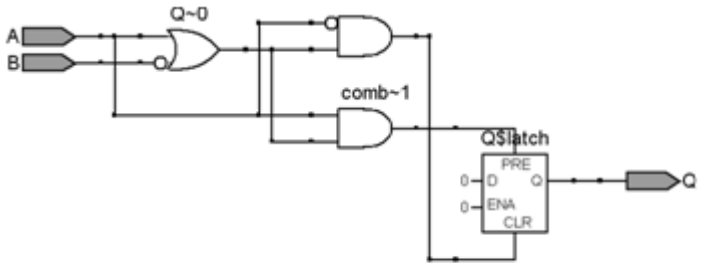


图 6-10 例 6-22 的 RTL 图

6.4 三态与双向端口设计

6.4.1 三态控制电路设计

【例6-23】

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

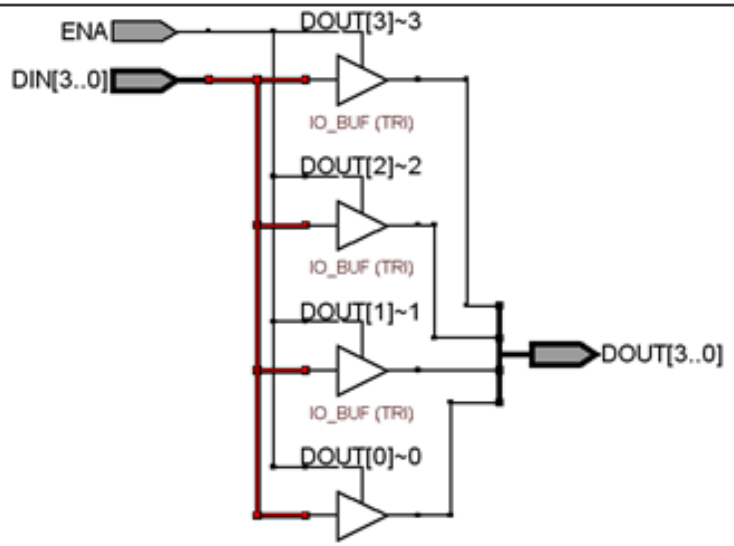


图6-11　4位三态控制门电路

6.4 三态与双向端口设计

6.4.2 双向端口设计

【例 6-24】

```
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
```

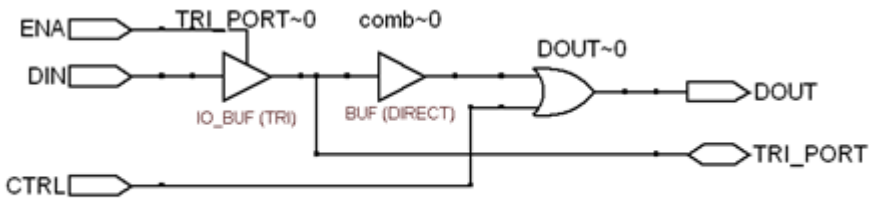


图 6-12 例 6-24 的 1 位双向端口电路设计之 RTL 图

6.4 三态与双向端口设计

6.4.2 双向端口设计

【例6-25】

```
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)
     if (!CTRL)  DOUT<=Q ;
       else
  begin Q<=DIN; DOUT<=4'HZ;  end
    endmodule
```

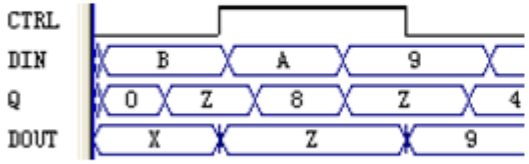


图6-13 本例的仿真波形图

【例6-26】

```
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)
 if (!CTRL)  begin DOUT<=Q;
       Q<=4'HZ; end  else
 begin Q<=DIN;  DOUT<=4'HZ;  end
 endmodule
```

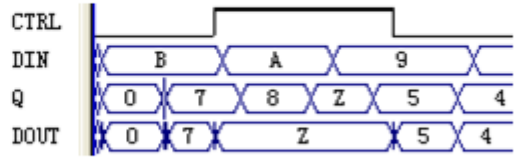


图6-14 本例的仿真波形图

6.4 三态与双向端口设计

6.4.2 双向端口设计

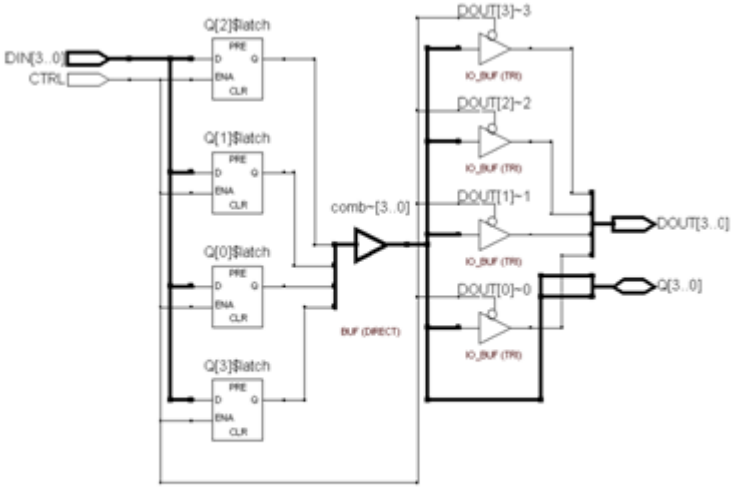


图 6-15 例 6-25 的 RTL 图

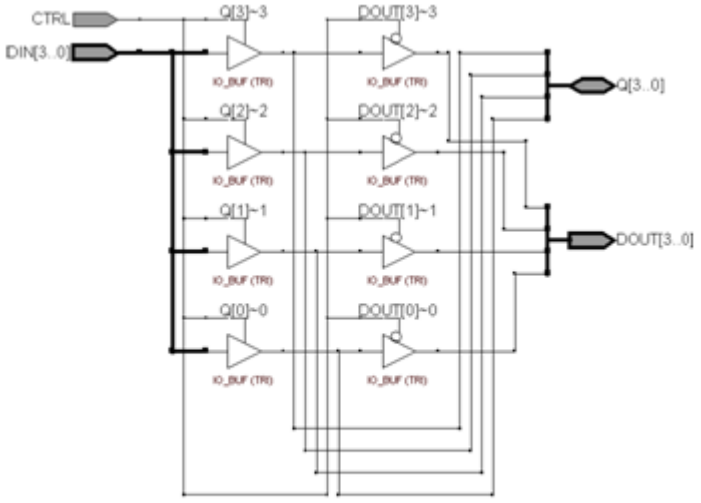


图 6-16 例 6-26 的 RLT 图

6.4 三态与双向端口设计

6.4.3 三态总线控制电路设计

【例6-27】

```
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6-28】

```
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, IN0)
  if (ENA==2'b00)  DOUT=IN0;
     else  DOUT=4'hz;
  always @(ENA, IN1)
  if (ENA==2'b01)  DOUT=IN1;
     else  DOUT=4'hz;
  always @(ENA, IN2)
  if (ENA==2'b10)  DOUT=IN2;
     else  DOUT=4'hz;
  always @(ENA, IN3)
  if (ENA==2'b11)  DOUT=IN3;
     else  DOUT=4'hz;

endmodule
```

6.4 三态与双向端口设计

6.4.3 三态总线控制电路设计

【例 6-29】

```
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
```

6.4 三态与双向端口设计

6.4.3 三态总线控制电路设计

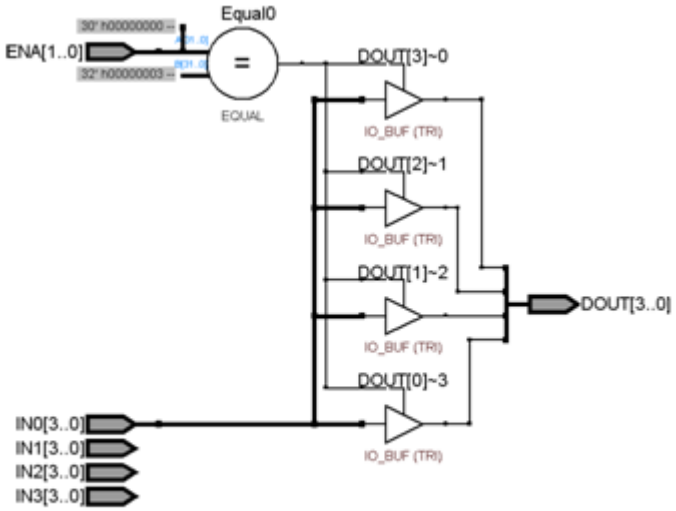


图6-17 例6-27的RTL图

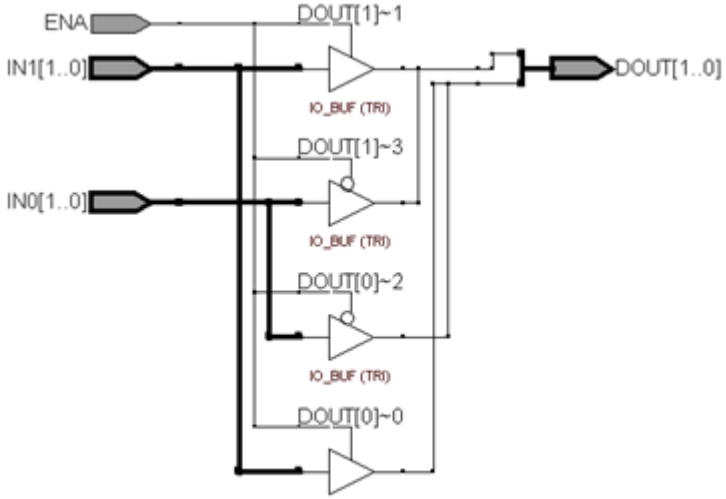


图6-18 例6-28的2位简化RTL图

6.5 系统设计优化

6.5.1 计资源优化

1、资源共享

【例6-30】

```
module multmux (A0,A1,B,S,R);
   input[3:0] A0, A1, B;
   input S;
   output[7:0] R;
   reg[7:0] R;
   always @(A0 or A1 or B or S)
     if (S==1'b0)  R<=A0 * B ;
        else  R<=A1 * B ;
endmodule
```

【例6-31】

```
module multmux (A0,A1,B,S,R);
   input[3:0] A0, A1, B;
   input S;   output[7:0] R;
   wire [7:0] R; reg [3:0] TEMP;
   always @(A0 or A1 or B or S)
    if (S==1'b0)  TEMP <= A0 ;
      else  TEMP <= A1  ;
    assign R=TEMP * B;
endmodule
```

6.5 系统设计优化

6.5.1 计资源优化

1、资源共享

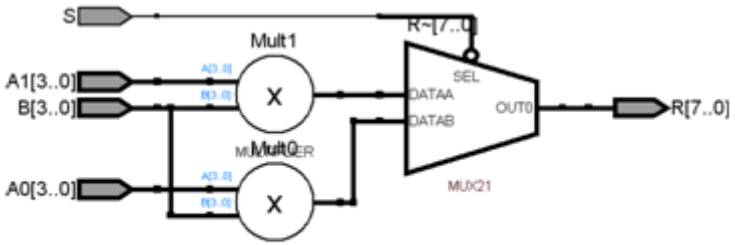


图 6-19 先乘后选择的设计方法 RTL 结构

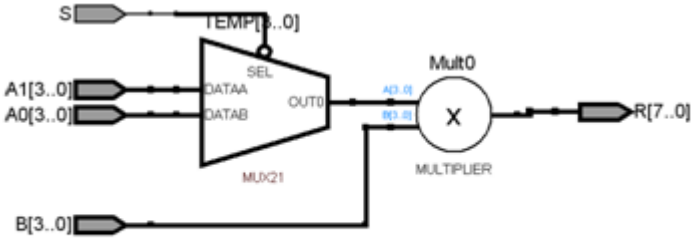


图 6-20 先选择后乘设计方法 RTL 结构

6.5 系统设计优化

6.5.1 计资源优化

1、资源共享

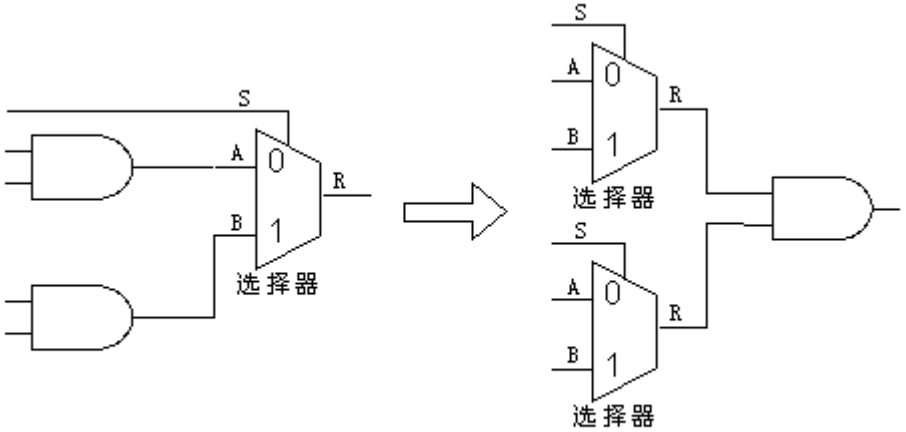


图 6-21　资源共享反例

6.5 系统设计优化

6.5.1 计资源优化

2、逻辑优化

【例6-32】

```
module mult1 (clk, ma, mc);
   input clk;  input[11:0] ma;
   output[23:0] mc;
   reg[23:0] mc; reg[11:0] ta,tb;
  always @(posedge clk)   begin
    ta <= ma; mc <= ta * tb;
    tb <= 12'b100110111001;  end
endmodule
```

【例6-33】

```
module mult2  (clk, ma, mc);
  input clk;  input[11:0] ma;
   output[23:0] mc;
  reg[23:0] mc;   reg[11:0] ta;
  parameter tb=12'b100110111001;
  always @(posedge clk)
  begin
  ta<=ma ; mc<=ta * tb ;
  end
endmodule
```

6.5 系统设计优化

6.5.1 计资源优化

3、串行化

$$yout = a_0 \times b_0 + a_1 \times b_1 + a_2 \times b_2 + a_3 \times b_3$$

【例 6-34】

```
module pmultadd  (clk, a0, a1, a2, a3, b0, b1, b2, b3, yout);
   input clk;   input[7:0] a0, a1, a2, a3, b0, b1, b2, b3;
   output[15:0] yout;   reg[15:0] yout;
   always @(posedge clk)
      yout <= ((a0 * b0)+(a1 * b1))+((a2 * b2)+(a3 * b3)) ;
endmodule
```

6.5 系统设计优化

【例 6-35】

```
module smultadd  (clk, start, a0, a1, a2, a3, b0, b1, b2, b3, yout);
 input clk, start;  input[7:0] a0, a1, a2, a3, b0, b1, b2, b3;
 output[15:0] yout;
 reg[15:0] yout, ytmp;    reg[2:0] cnt;
 wire[7:0] tmpa, tmpb; wire[15:0] tmp;
 assign tmpa=(cnt==0)? a0:(cnt==1)? a1:(cnt==2)? a2:(cnt==3)? a3:a0;
 assign tmpb=(cnt==0)? b0:(cnt==1)? b1:(cnt==2)? b2:(cnt==3)? b3:b0;
  assign tmp = tmpa * tmpb ;
 always @(posedge clk)
    begin
    if (start==1'b1)  begin  cnt<=3'b000 ; ytmp<={16{1'b0}} ;   end
       else if (cnt<4)   begin  cnt<=cnt+1  ; ytmp<=ytmp+tmp ;  end
       else if (cnt==4)  begin  yout<=ytmp ;   end
    end
    endmodule
```

6.5 系统设计优化

6.5.2 速度优化

1、流水线设计

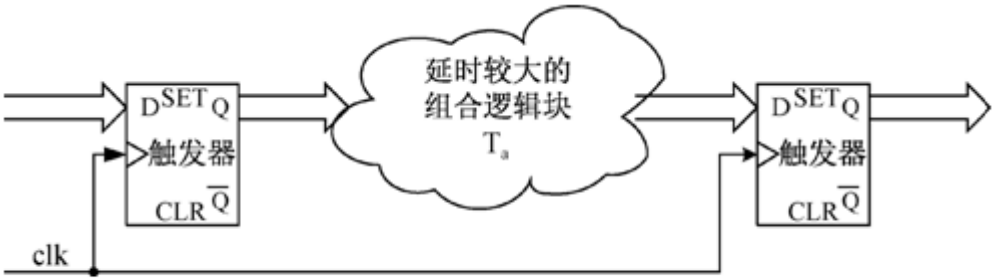


图 6-22 未使用流水线

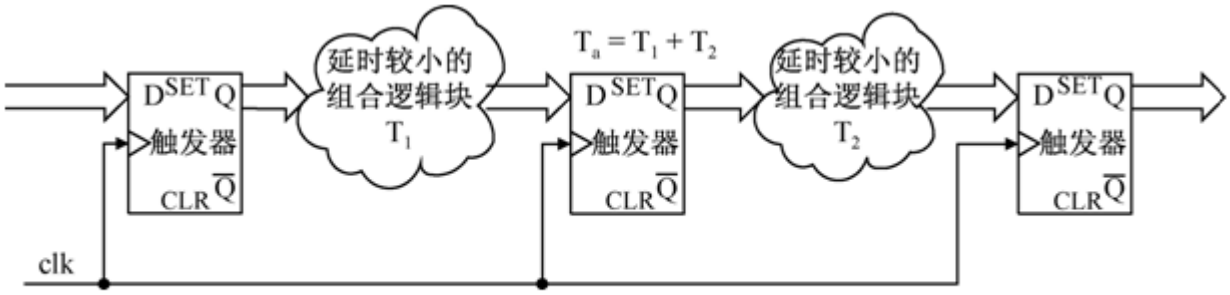


图 6-23 使用流水线结构

6.5 系统设计优化

6.5.2 速度优化

1、流水线设计

图 6-24 流水线工作图示

6.5 系统设计优化

6.5.2 速度优化

1、流水线设计

【例 6-36】普通加法器，EP3C55 综合结果：LCs=10,REG=0,**T=7.748ns.**

```
module ADDER8(CLK,SUM,A,B,COUT,CIN);
   input [7:0] A,B;  input CLK,CIN;
   output COUT;  output [7:0] SUM;
   reg COUT;   reg [7:0] SUM;
    always @(posedge CLK)   {COUT,SUM[7:0]} <= A + B + CIN;
  endmodule
```

6.5 系统设计优化

【例 6-37】流水线加法器，EP3C55 综合结果：**T=3.63ns**，LCs=24，REG=22。

```
module ADDER8(CLK,SUM,A,B,COUT,CIN);
   input [7:0] A,B;  input CLK,CIN;
   output COUT;   output[7:0] SUM;
    reg TC,COUT;   reg[3:0] TS,TA,TB;    reg[7:0] SUM;
     always @(posedge CLK)  begin
        {TC,TS} <= A[3:0]+B[3:0]+CIN ;
        SUM[3:0]<=TS;    end
     always @(posedge CLK)  begin
        TA <= A[7:4];   TB <= B[7:4];
      {COUT,SUM[7:4]} <= TA+TB+TC;
      end
endmodule
```

6.5 系统设计优化

6.5.2 速度优化

1、流水线设计

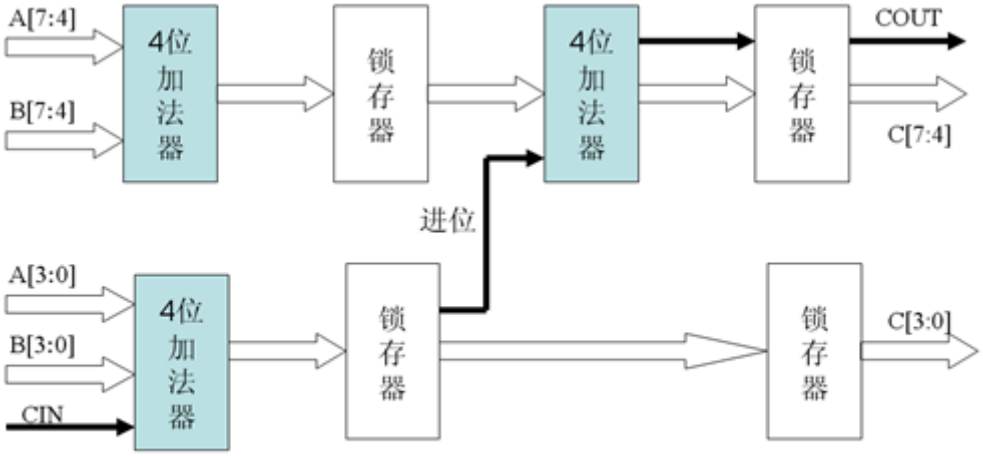


图 6-25　8 位加法器流水线工作图示

6.5 系统设计优化

6.5.2 速度优化

1、流水线设计

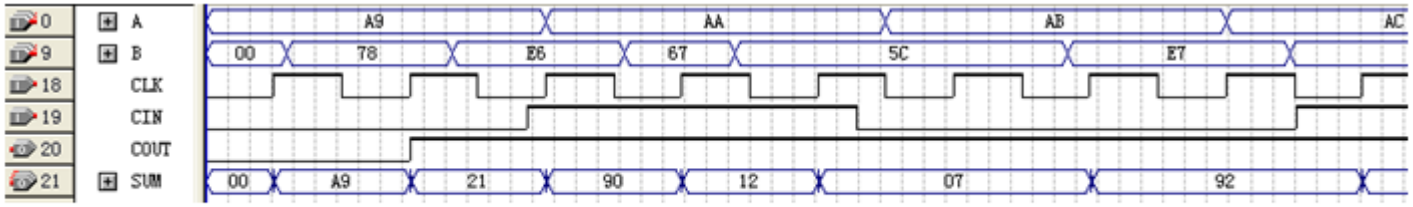


图 6-26 例 6-36 的时序仿真波形

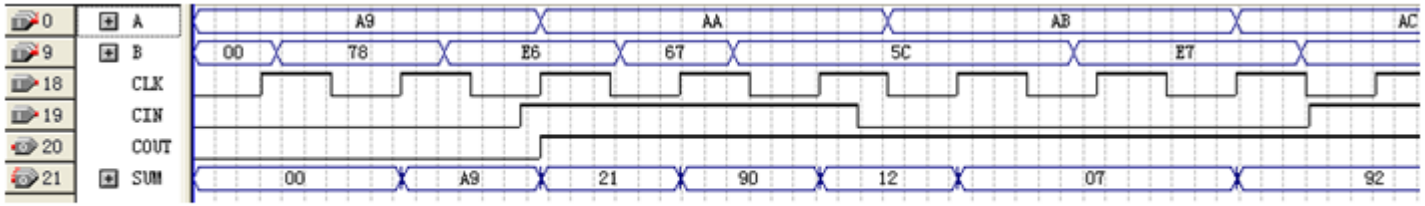


图 6-27 例 6-37 的时序仿真波形

6.5 系统设计优化

6.5.2 速度优化

2、关键路径法

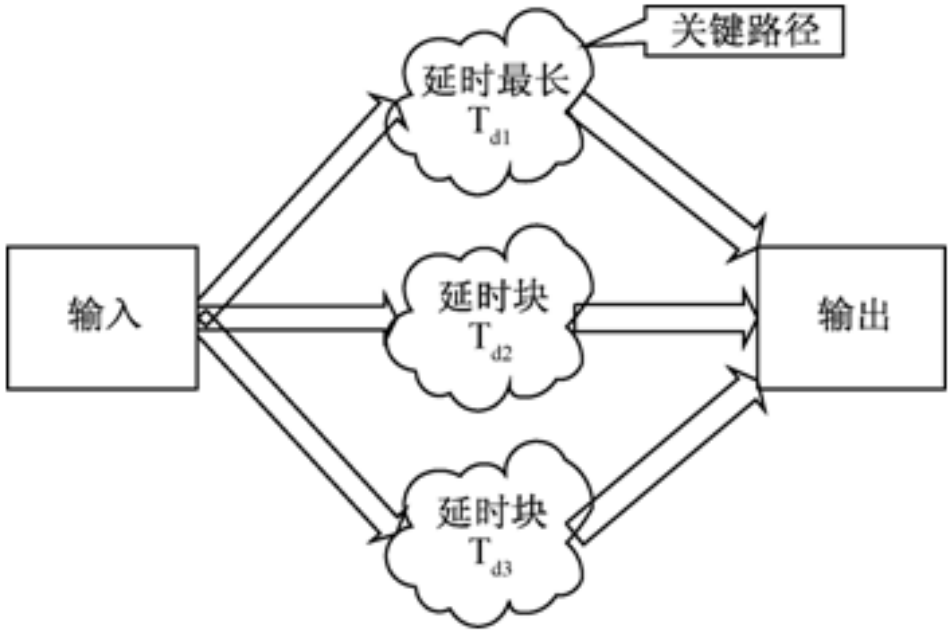


图 6-28 关键路径示意

6.5 系统设计优化

6.5.2 速度优化

3、乒乓操作法

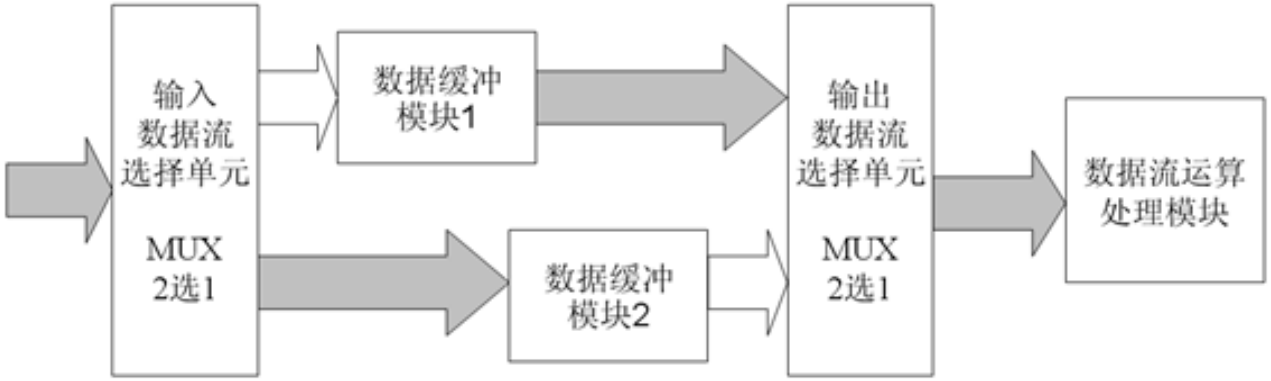
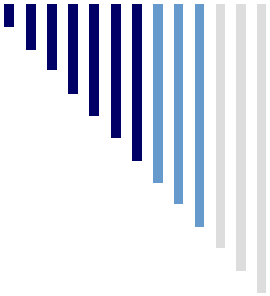


图 6-29 乒乓操作数据缓存结构示意图

4、加法树法



6.6 Verilog的描述风格

6.6.1 RTL描述

6.6.2 行为描述

6.6.3 数据流描述

6.6.4 结构描述

习　题

【例 6-38】

```
module test1 (X1,X2,A,B,C,D,CLK) ;
input CLK,X1,X2;
output A,B,C,D;
reg A,B,C,D;
 always@(posedge CLK )
  begin
    A=X1; D=X2; B<=D;  C<=A; end
endmodule
```

【例 6-39】

```
module test1 (X1,X2,A,B,C,D,CLK) ;
input CLK,X1,X2;
output A,B,C,D;
reg A,B,C,D;
 always@(posedge CLK )
   begin
  B<=D;  C<=A;  A=X1; D=X2;   end
endmodule
```

```
module addmux (A, B, C, D, sel, Result);
   input[7:0] A, B, C, D;   input sel;
   output[7:0] Result;    reg[7:0] Result;
   always @(A or B or C or D or sel)
      if (sel==1'b0)   Result<=A + B ;
      else  Result<=C + D ;
endmodule
```

习　题

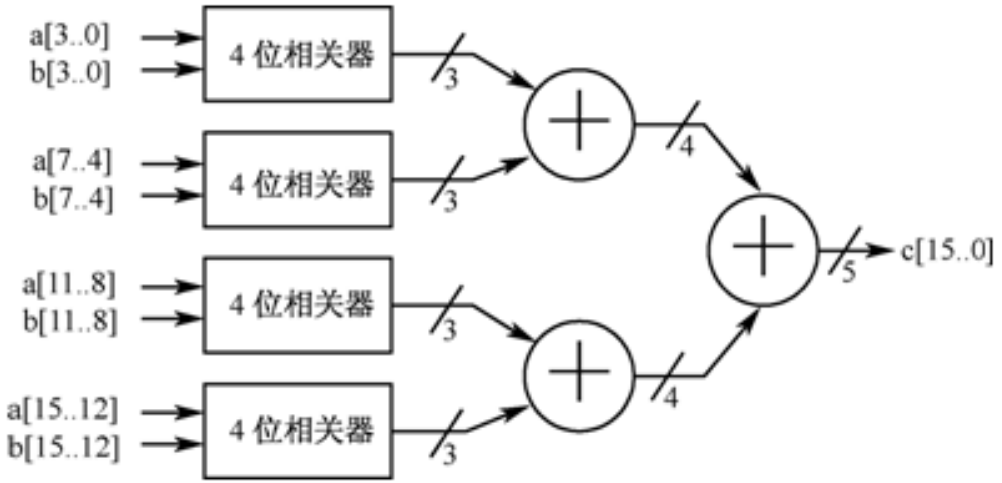


图 6-30　习题 6-17 图

EDA实验与创新实践

6-1 硬件消抖动电路设计

【例6-40】

```
module ERZP  (CLK, KIN,KOUT);
   input   CLK, KIN;                    //工作时钟和输入信号
   output  KOUT;     reg KOUT;
   reg [3:0] KH,KL;                     //定义对高电平和低电平脉宽计数之寄存器。
   always @(posedge CLK)
      if (!KIN)  KL<=KL+1 ;             //对键输入的低电平脉宽计数
        else  KL<=4'b0000;              //若出现高电平，则计数器清 0
   always @(posedge CLK)
       if (KIN)  KH<= KH+1;             //同时对键输入的高电平脉宽计数
         else KH<=4'b0000;              //若出现高电平，则计数器清 0
   always @(posedge CLK)  begin
         if (KH>4'b1100) KOUT<=1'B1;//对高电平脉宽计数一旦大于 12，则输出 1
    else if(KL>4'b0111) KOUT<=1'B0;//对低电平脉宽计数若大于 7，则输出 0
    end    endmodule
```

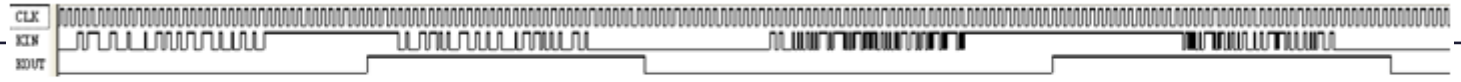


图 6-31 例 6-40 消抖动电路仿真波形

EDA实验与创新实践

6-2 4X4阵列键盘键信号检测电路设计

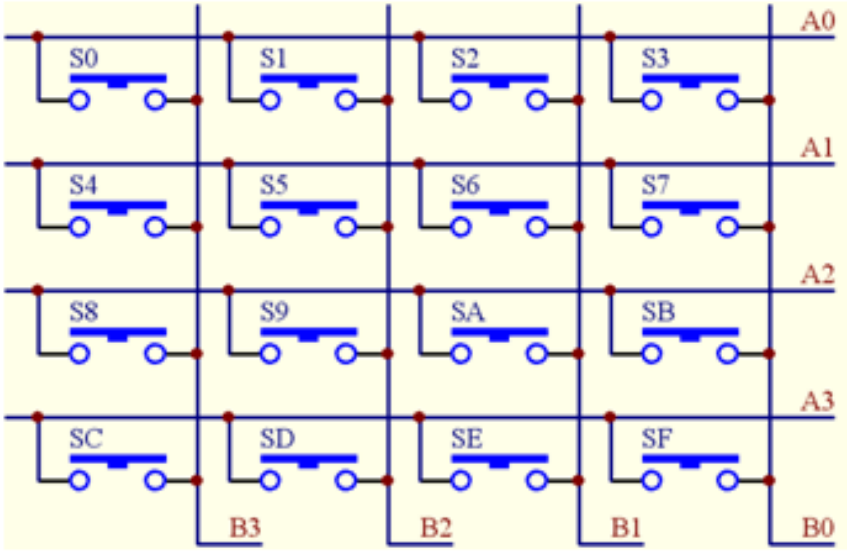


图 6-32　4X4 键盘电路

EDA实验与创新实践

【例 6-41】

```
module KEY4X4 (input CLK, input [3:0]A, output reg[3:0]B,R);
     reg [1:0] C;
    always  @ (posedge CLK)    begin     C<=C+1;
      case(C)
     0: B=4'B0111; 1: B=4'B1011; 2: B=4'B1101; 3: B=4'B1110;
      endcase
      case({B,A} )
     8'B0111_1110 : R=4'H0;   8'B0111_1101 : R=4'H1;
     8'B0111_1011 : R=4'H2;   8'B0111_0111 : R=4'H3;
     8'B1011_1110 : R=4'H4;   8'B1011_1101 : R=4'H5;
     8'B1011_1011 : R=4'H6;   8'B1011_0111 : R=4'H7;
     8'B1101_1110 : R=4'H8;   8'B1101_1101 : R=4'H9;
     8'B1101_1011 : R=4'HA;   8'B1101_0111 : R=4'HB;
     8'B1110_1110 : R=4'HC;   8'B1110_1101 : R=4'HD;
     8'B1110_1011 : R=4'HE;   8'B1110_0111 : R=4'HF;
   endcase   end
endmodule
```

EDA实验与创新实践

6-2 4X4阵列键盘键信号检测电路设计

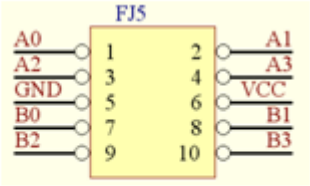


图 6-33 4X4 键盘的 10 芯接口

图 6-34 设置端口上拉

EDA实验与创新实践

6-3 直流电机综合测控系统设计

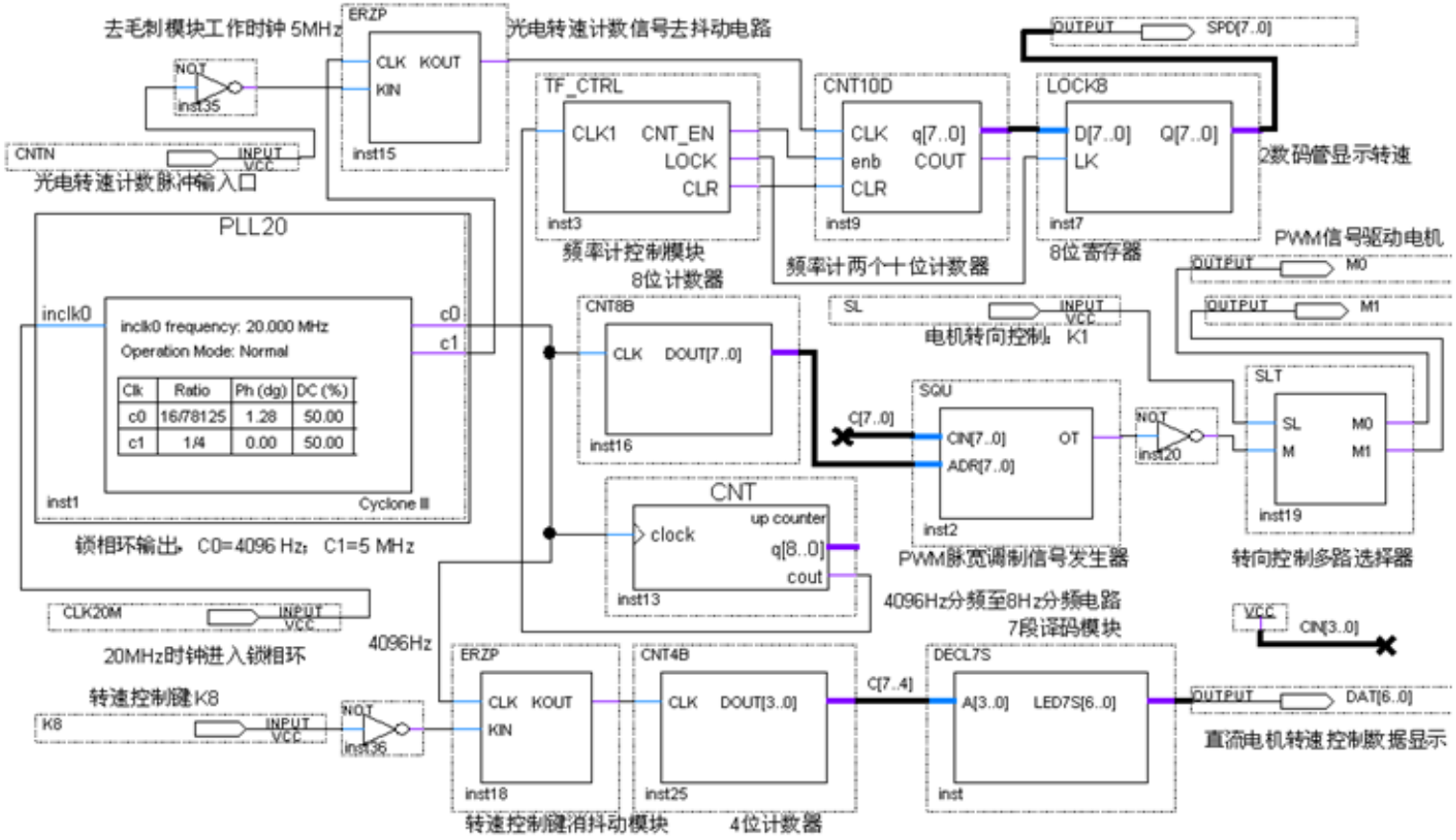


图 6-35 直流电机驱动控制电路顶层设计

EDA实验与创新实践

6-3 直流电机综合测控系统设计

【例 6-42】

```
module SQU (input[7:0] CIN, input[7:0] ADR, output reg OT) ;
   always @(CIN)  if (ADR<CIN)  OT<=1'b0;  else  OT<=1'b1 ;
endmodule
```

EDA实验与创新实践

6-4 VGA彩条信号显示控制电路设计

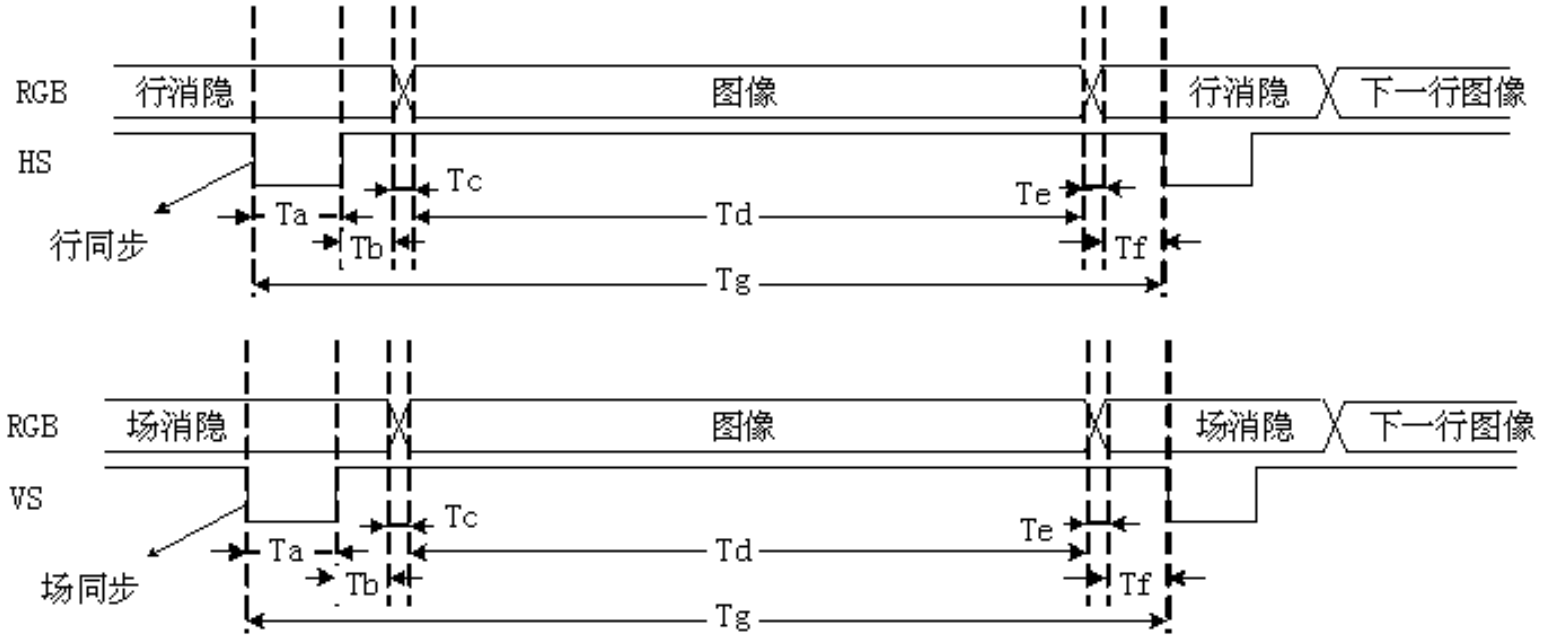


图 6-36 VGA 行扫描、场扫描时序示意图

EDA实验与创新实践

6-4 VGA彩条信号显示控制电路设计

表 6-2 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 6-3 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

【例 6-43】

```
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
  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
```

接下页

```
  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 ;
     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
```

EDA实验与创新实践

6-4 VGA彩条信号显示控制电路设计

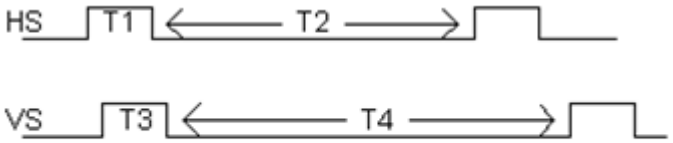


图 6-37 HS 和 VS 的时序图

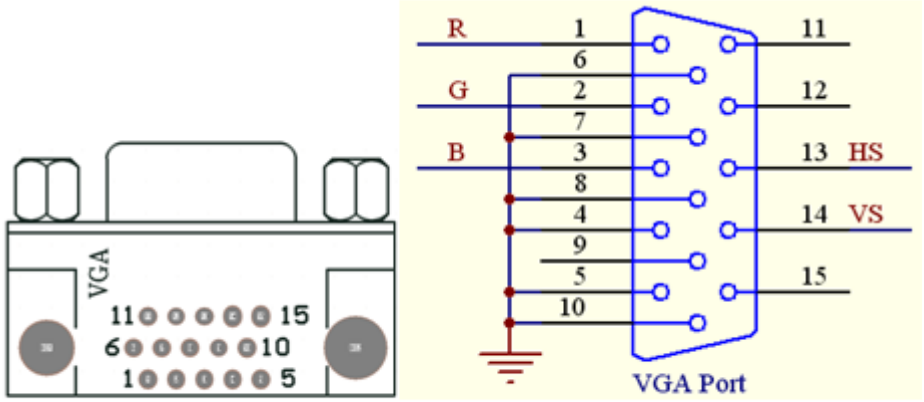


图 6-38 VGA 接口电路图，左接口从上往下看

EDA实验与创新实践

6-4 VGA彩条信号显示控制电路设计

表 6-4 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 6-5 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

EDA实验与创新实践

6-5 移位相加型8位硬件乘法器设计

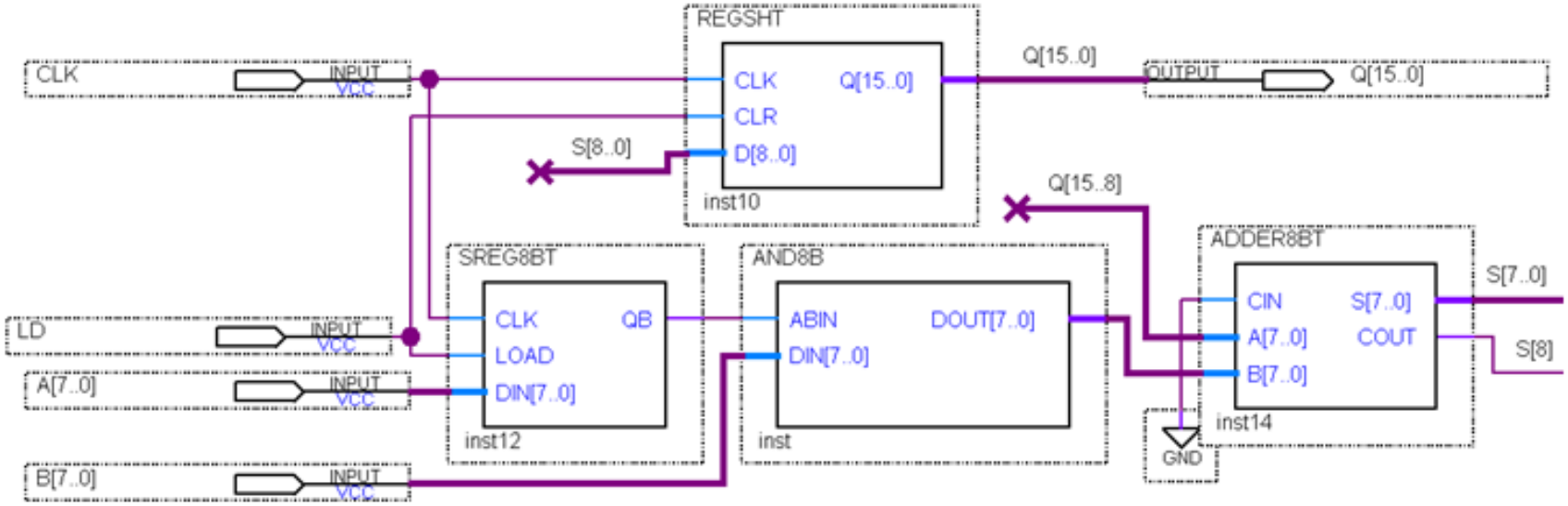


图 6-39　8 位乘法器逻辑原理图

EDA实验与创新实践

6-5 移位相加型8位硬件乘法器设计

图 6-40　8 位移位相加乘法器运算逻辑波形图

EDA实验与创新实践

6-5 移位相加型8位硬件乘法器设计

【例 6-44】

```
module REGSHT (CLK, CLR, D, Q);
   input CLK, CLR;  input[8:0] D;  output[15:0] Q;  reg[15:0] R16S;
   always @(posedge CLK or posedge CLR)
      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
      assign Q = R16S ;
 endmodule
```

EDA实验与创新实践

6-6 采用流水线技术设计高速数字相关器

6-7 线性反馈移位寄存器设计

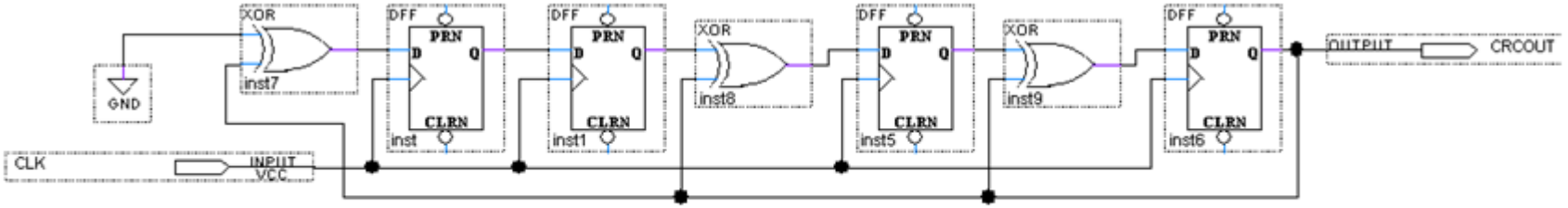


图 6-41 LFSR 举例

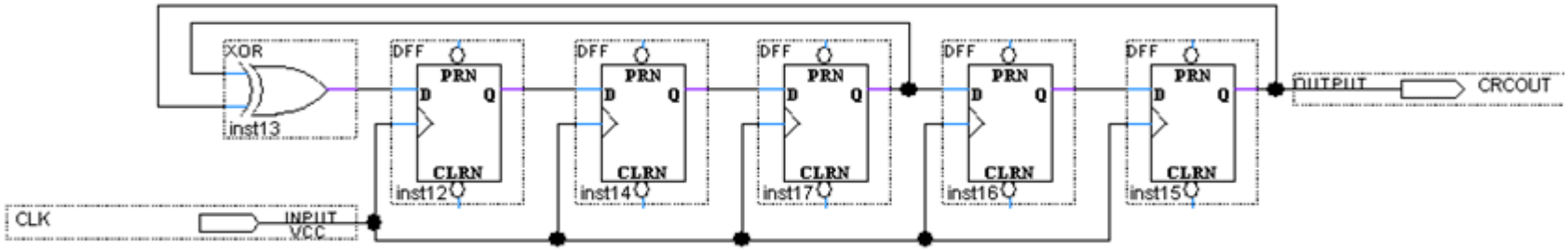


图 6-42 另一种 LFSR 结构