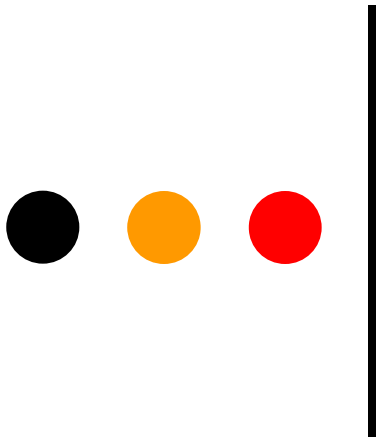

第8章

Verilog HDL深入

8.1 过程中的两类赋值语句

8.1.1 未指定延时的阻塞式赋值语句

目标变量名 = 驱动表达式;

8.1.2 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 8-1】

```
    Y1 = A^B;
#6  Y2 = A&B|C;
```

【例 8-2】

```
Y1 = A^D;
Y2 = #6 A&E|C;
```

8.1 过程中的两类赋值语句

8.1.3 未指定延时的非阻塞式赋值

目标变量名 <= 驱动表达式;

【例 8-3】

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

【例 8-4】

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

【例 8-5】

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

设过程启动后，A=2'b10，B=2'b01，C=2'b11

8.1 过程中的两类赋值语句

8.1.3 未指定延时的非阻塞式赋值

【例 8-6】

```
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：Q=1|1=1
```

【例 8-7】

```
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
```

8.1 过程中的两类赋值语句

8.1.4 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 8-8】

```
begin
   Y1 = #6 A^B;
   Y2 = #4 A|B;
   Y3 = #7 A&B;
end
```

【例 8-9】

```
begin
   Y1 <= #6 A^B;
   Y2 <= #4 A|B;
   Y3 <= #7 A&B;
end
```

【例 8-10】

```
begin
   Y1  = #5 A^B;
   Y2 <= #3 A|B;
   Y3 <= #2 A&B;
   Y4  = #4 (~B);  end
```

8.1 过程中的两类赋值语句

8.1.5 深入认识阻塞与非阻塞式赋值的特点

【例 8-11】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
```

【例 8-12】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
```

8.1 过程中的两类赋值语句

8.1.5 深入认识阻塞与非阻塞式赋值的特点

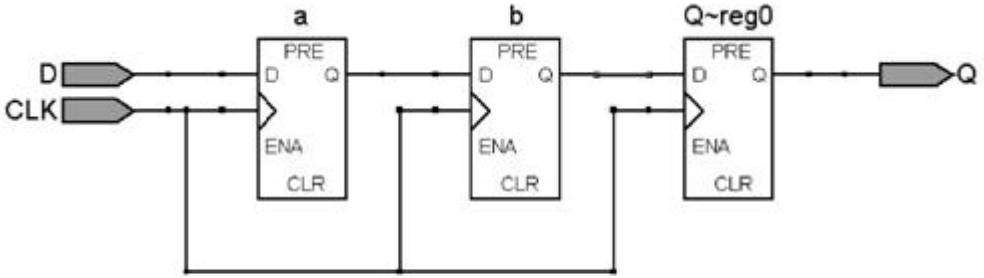

图 8-1　例 8-11 综合后的 RTL 电路

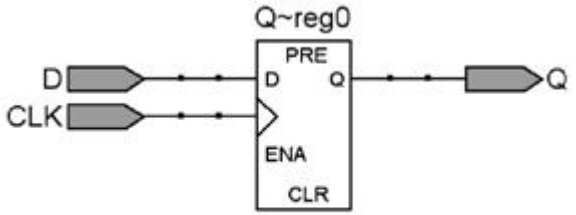

图 8-2　例 8-12 综合后的 RTL 电路

```
begin  Q = b;      b = a;     a = D;  end
```

8.1 过程中的两类赋值语句

8.1.6 不同的赋初值方式导致不同综合结果的示例

【例 8-13】

```
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 8-14】

```
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
```

8.1 过程中的两类赋值语句

8.1.6 不同的赋初值方式导致不同综合结果的示例

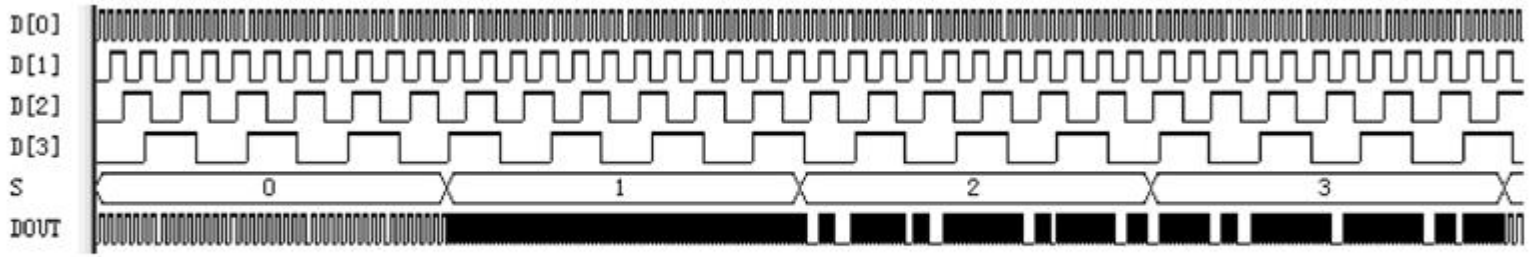

图 8-3　例 8-13 的错误工作时序

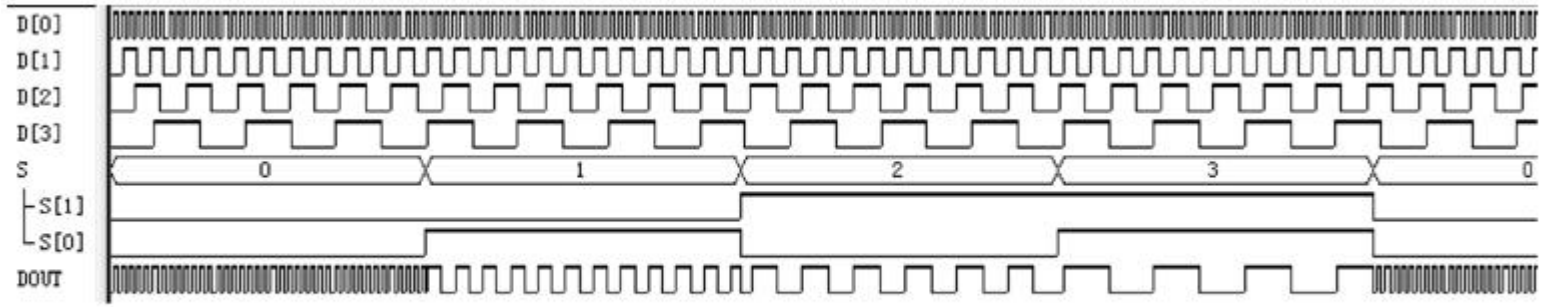

图 8-4　例 8-14 的正确工作时序

8.2 过程语句归纳

8.2.1 过程语句应用总结

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

8.2 过程语句归纳

8.2.2 深入认识不完整条件语句与时序电路的关系

【例 8-15】

```
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 8-16】

```
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 8-17】

```
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
```

8.2 过程语句归纳

8.2.2 深入认识不完整条件语句与时序电路的关系

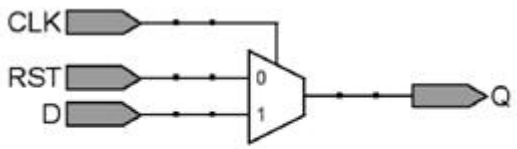

图 8-5 例 8-15 的 RTL 图

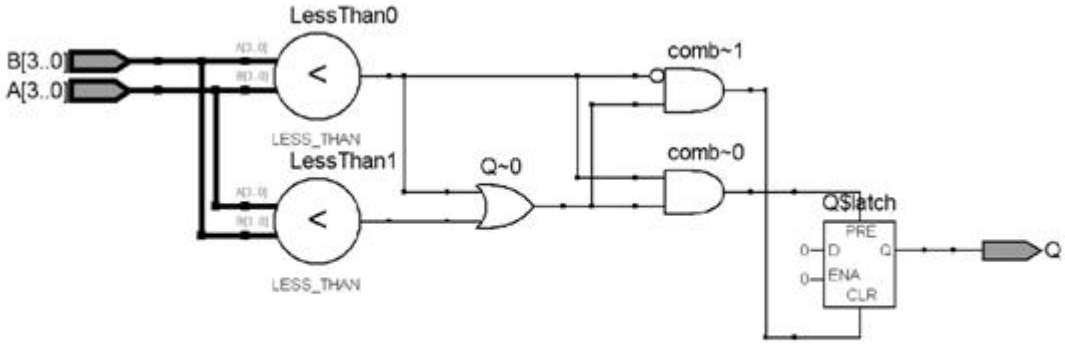

图 8-6 例 8-16 的 RTL 图，输出口被加上了锁存器

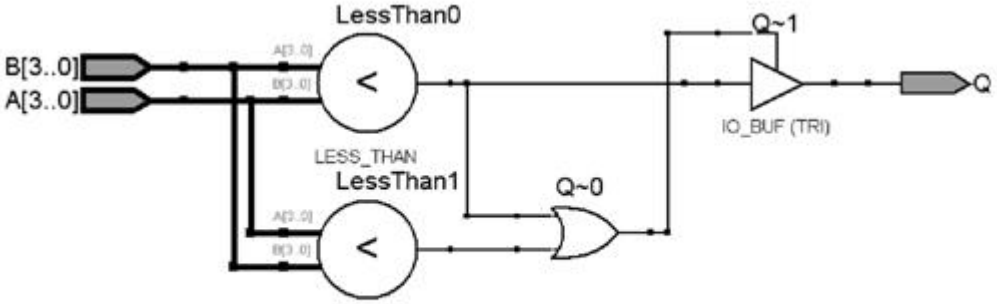

图 8-7 例 8-17 的 RLT 电路图，输出口没有锁存器，是纯组合电路

8.3 if语句归纳

8.3.1 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
```

8.3 if语句归纳

8.3.1 if语句的一般表述形式

【例 8-18】

```
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
```

【例 8-19】

```
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
```

8.3 if语句归纳

8.3.1 if语句的一般表述形式

表 8-1　8-3 线优先编码器真值表（表中的“x”为任意）

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

8.3 if语句归纳

8.3.1 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 8-1 最后一行吻合
```

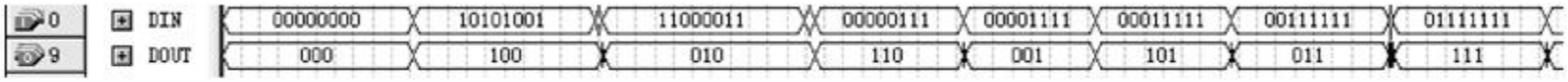

图 8-8　例 8-18/例 8-19 的时序仿真波形图

8.3 if语句归纳

8.3.2 关注if语句中的条件指示

【例 8-20】

```
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 8-21】

```
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 8-22】

```
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
```

8.3 if语句归纳

8.3.2 关注if语句中的条件指示

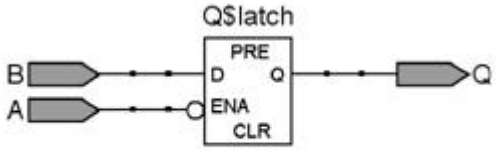

图 8-9　例 8-20/例 8-21 的 RTL 图

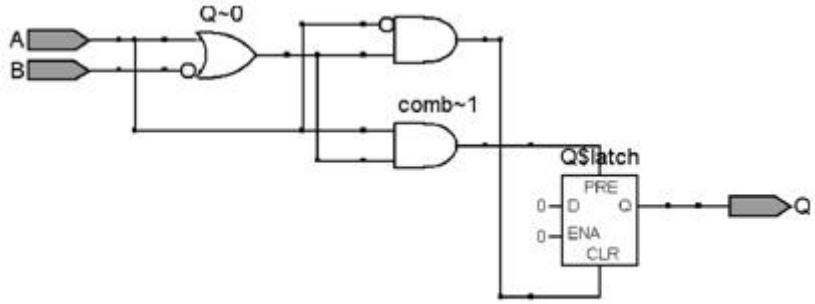

图 8-10　例 8-22 的 RTL 图

8.4 三态与双向端口设计

8.4.1 三态控制电路设计

【例 8-23】

```
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
```

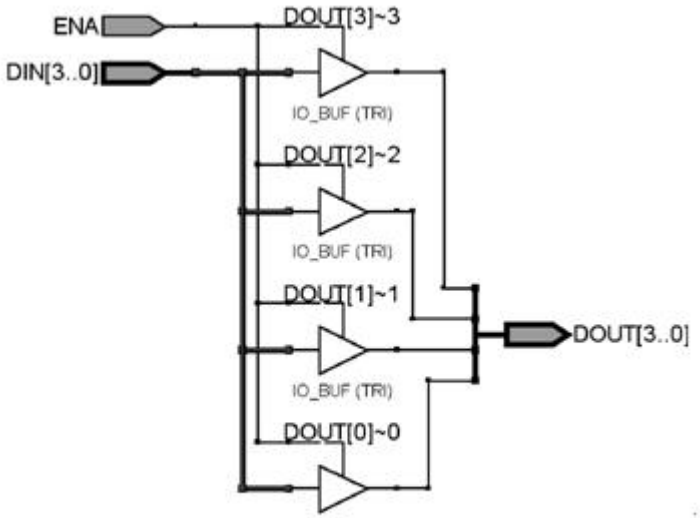

图 8-11　4 位三态控制门电路

8.4 三态与双向端口设计

8.4.2 双向端口设计

【例 8-24】

```
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
```

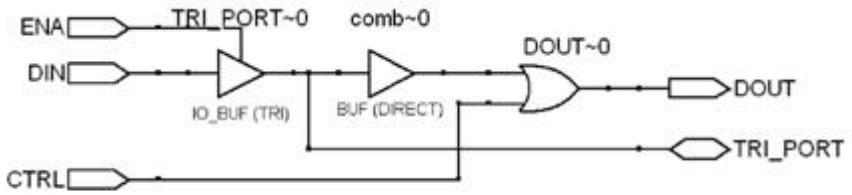

图 8-12　例 8-24 的 RTL 图

8.4 三态与双向端口设计

8.4.2 双向端口设计

【例 8-25】

```
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
```

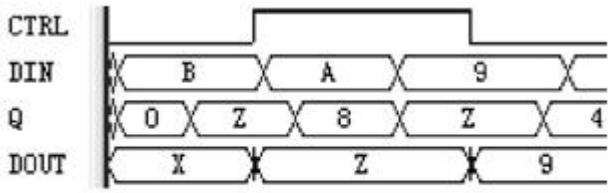

图 8-13　例 8-25 的仿真波形图

【例 8-26】

```
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
```

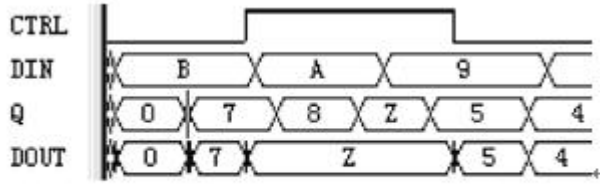

图 8-14　例 8-26 的仿真波形图

8.4 三态与双向端口设计

8.4.2 双向端口设计

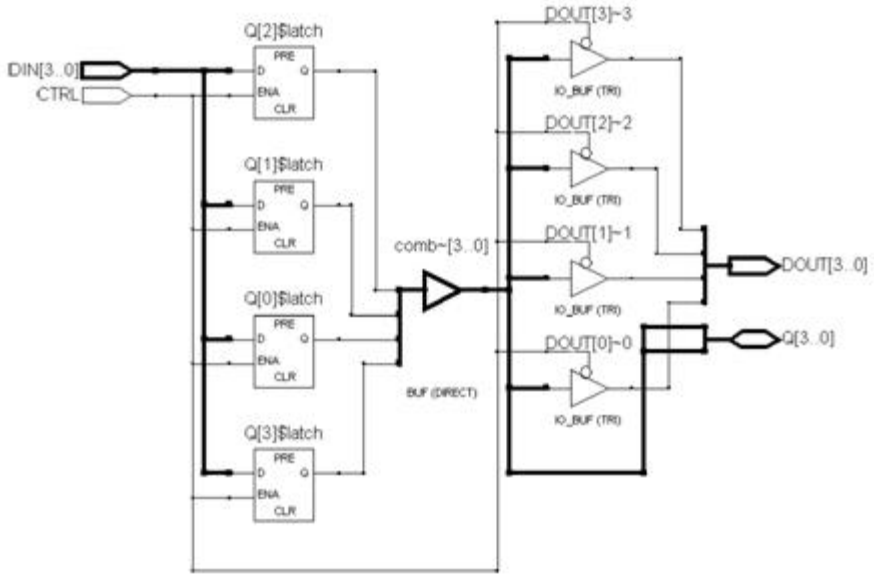

图 8-15　例 8-25 的 RTL 图

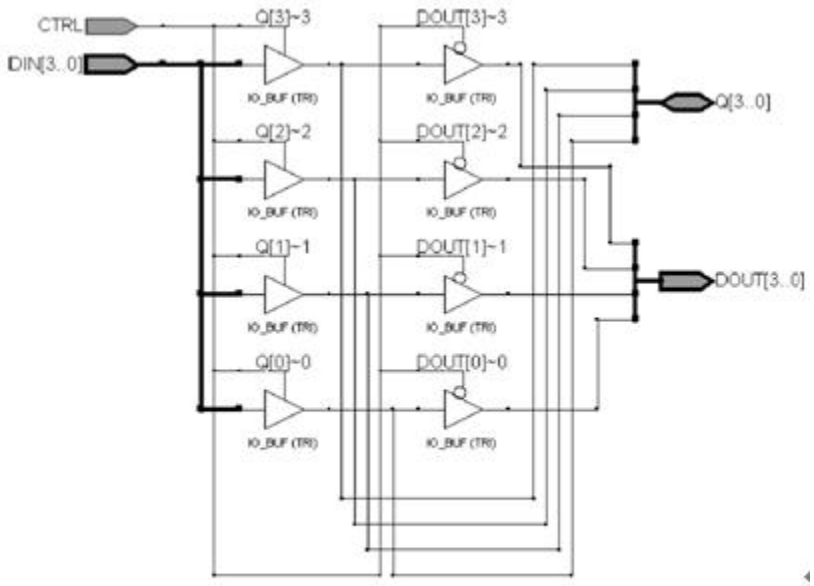

图 8-16　例 8-26 的 RLT 图

8.4 三态与双向端口设计

8.4.3 三态总线控制电路设计

【例 8-27】

```
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 8-28】

```
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
```

8.4 三态与双向端口设计

8.4.3 三态总线控制电路设计

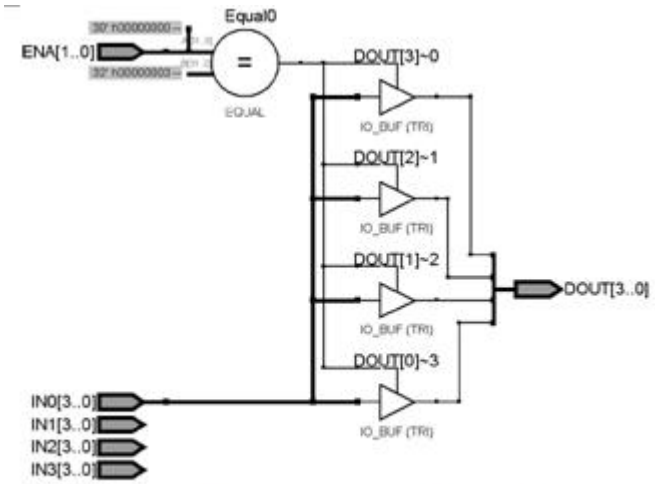

图 8-17　例 8-27 的 RTL 图

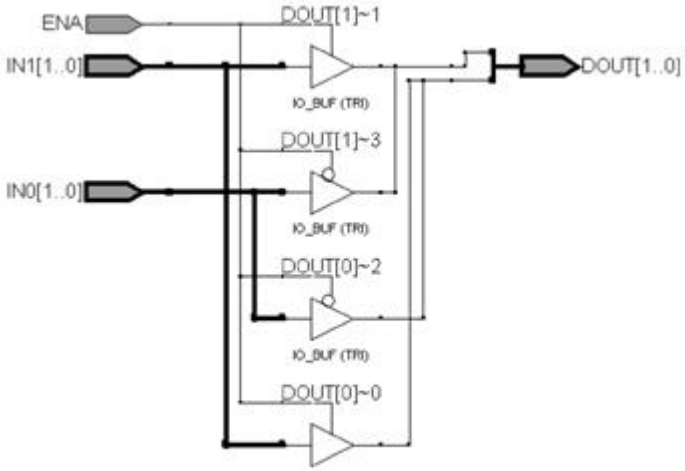

图 8-18　例 8-28 的 2 位简化 RTL 图

8.4 三态与双向端口设计

8.4.3 三态总线控制电路设计

【例 8-29】

```
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
```

8.5 Verilog系统设计优化

8.5.1 资源共享

【例 8-30】

```
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)
    begin
    if (S==1'b0)  R<=A0 * B;
      else  R<=A1 * B;
 end
endmodule
```

【例 8-31】

```
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)
  begin  if (S==1'b0)  TEMP<=A0;
   else  TEMP <= A1;   end
  assign R=TEMP * B;
endmodule
```

8.5 Verilog系统设计优化

8.5.1 资源共享

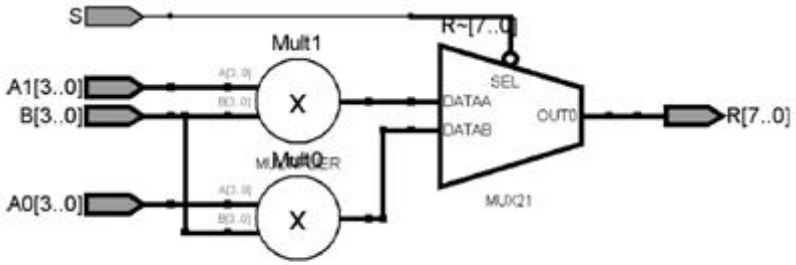

图 8-19　先乘后选择的设计方法 RTL 结构

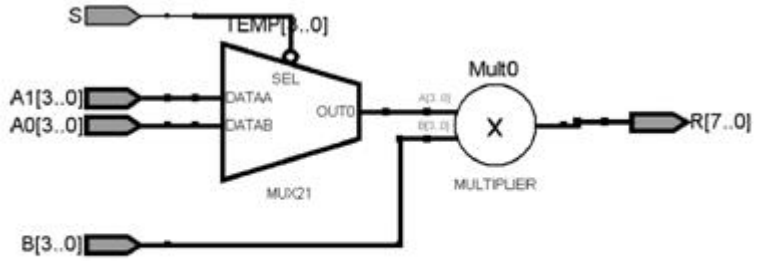

图 8-20　先选择后乘设计方法 RTL 结构

8.5 Verilog系统设计优化

8.5.1 资源共享

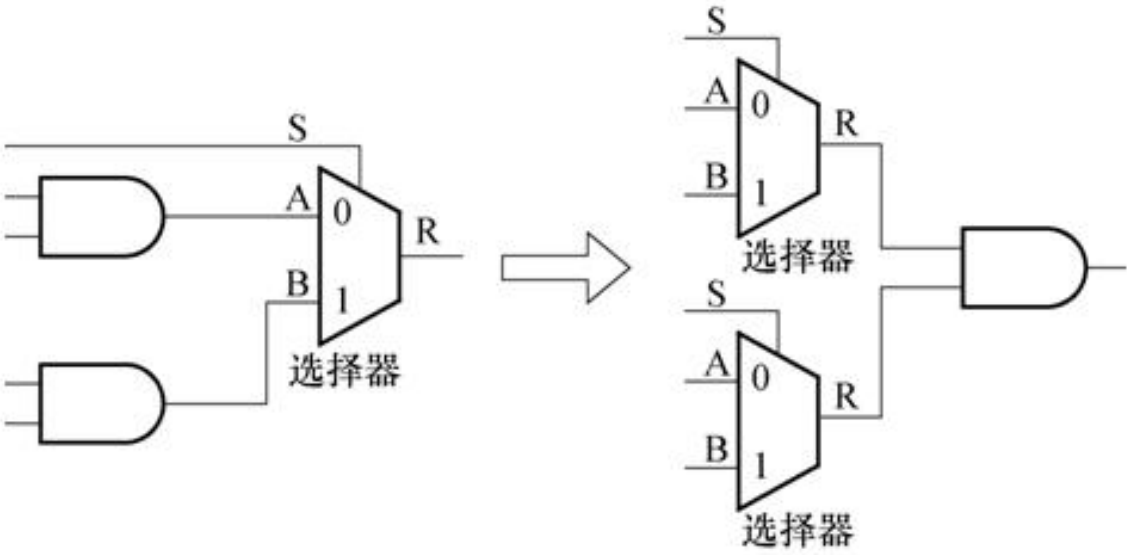

图 8-21　资源共享反例

8.5 Verilog系统设计优化

8.5.2 逻辑优化

【例 8-32】

```
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 8-33】

```
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
```

8.5 Verilog系统设计优化

8.5.3 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 8-34】

```
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)   begin
      yout <= ((a0 * b0)+(a1 * b1))+((a2 * b2)+(a3 * b3));   end
endmodule
```

8.5 Verilog系统设计优化

8.5.3 串行化

【例 8-35】

```
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
```

8.5 Verilog系统设计优化

8.5.4 流水线设计

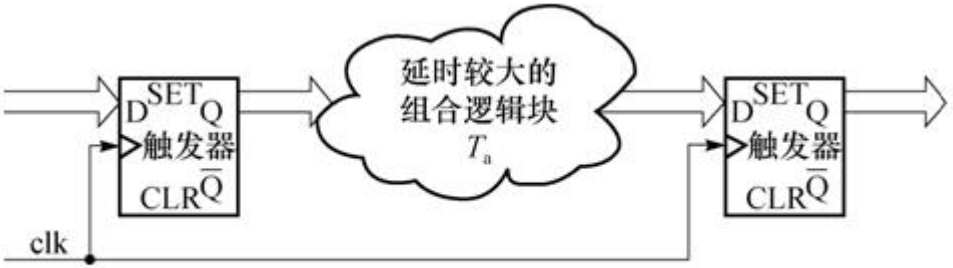

图 8-22　未使用流水线

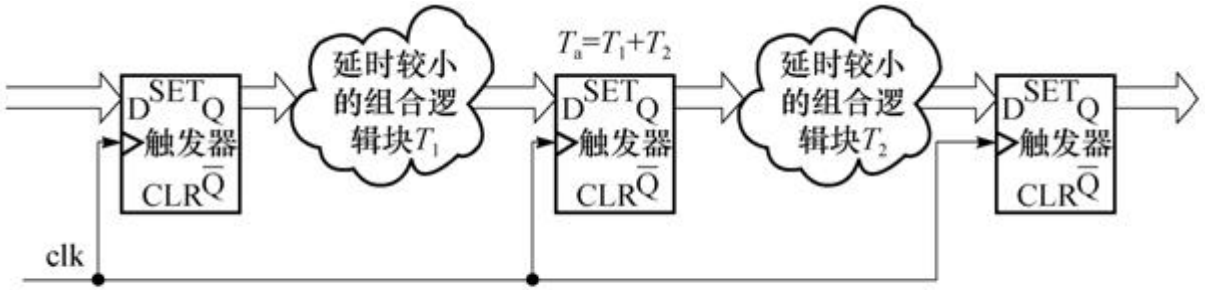

图 8-23　使用流水线结构

8.5 Verilog系统设计优化

8.5.4 流水线设计

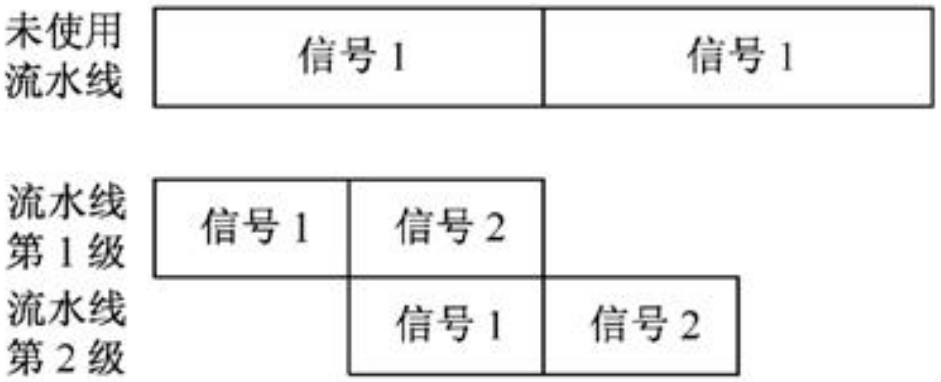

图 8-24　流水线工作图示

8.5 Verilog系统设计优化

8.5.4 流水线设计

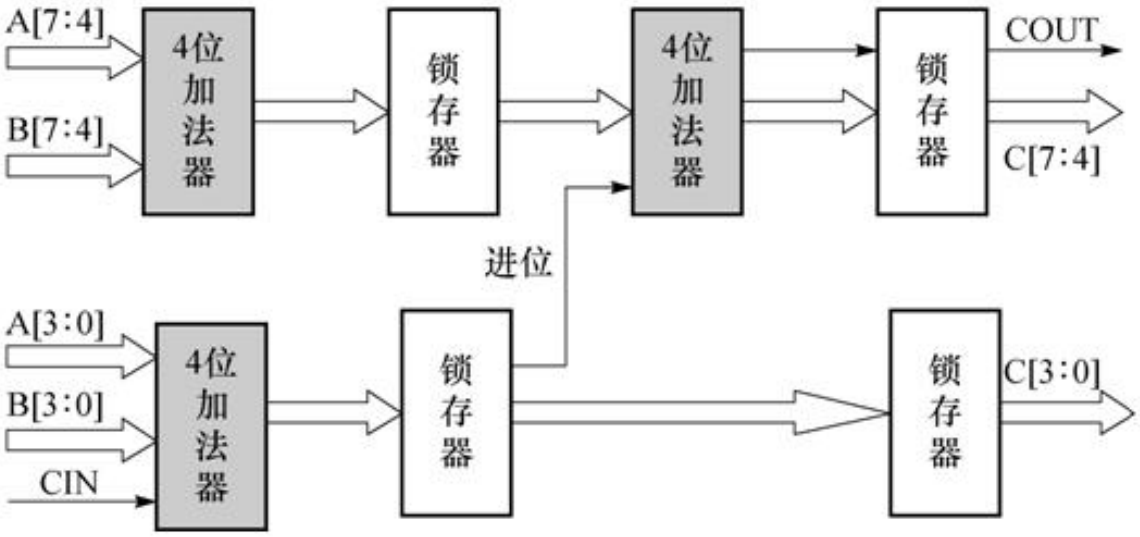

图 8-25　8 位加法器流水线工作图示

8.5 Verilog系统设计优化

8.5.4 流水线设计

【例 8-36】EP3C5 综合结果：LCs=10, REG=0（纯组合逻辑），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
```

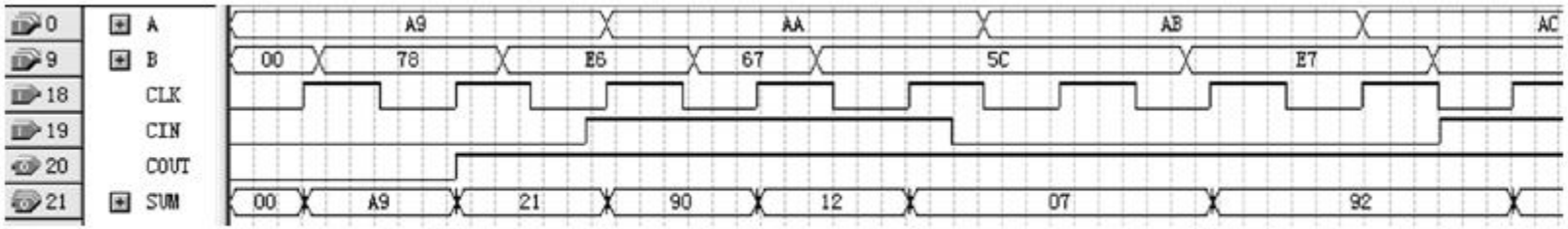

图 8-26　例 8-36 的时序仿真波形

8.5 Verilog系统设计优化

8.5.4 流水线设计

【例 8-37】EP3C5 综合结果：T=3.63ns，LCs=24，REG=22（时序逻辑）

```
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
```

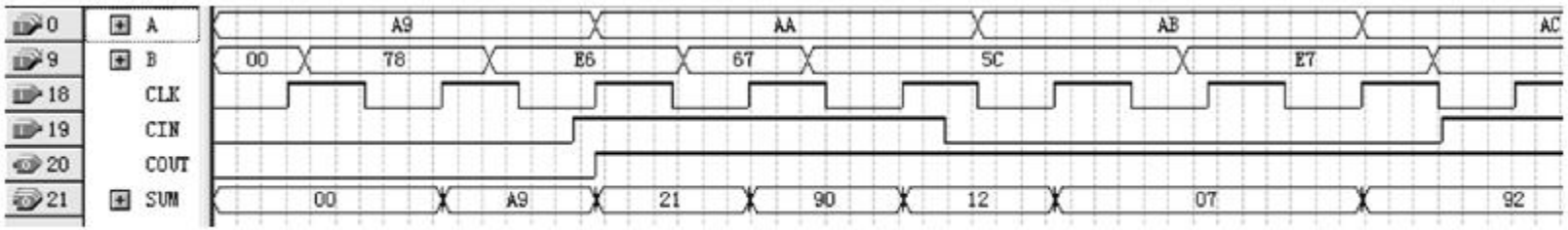

图 8-27　例 8-37 的时序仿真波形

8.5 Verilog系统设计优化

8.5.5 乒乓操作法

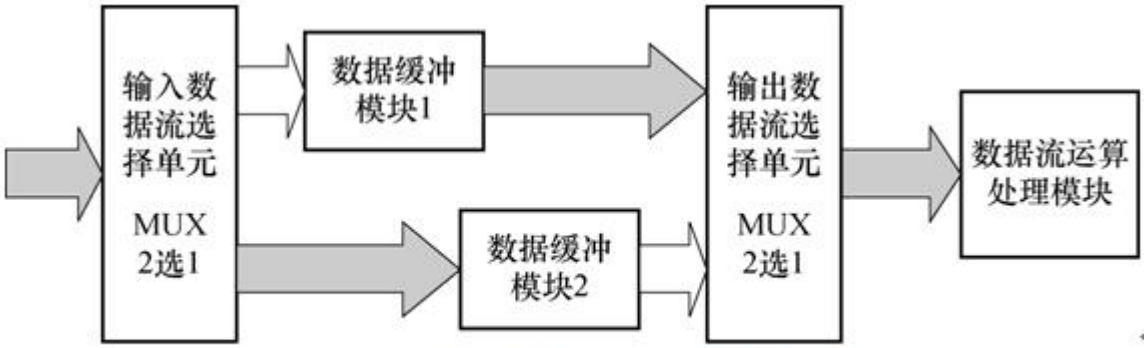

图 8-28　乒乓操作数据缓存结构示意图

8.5 Verilog系统设计优化

8.5.6 寄存器配平法

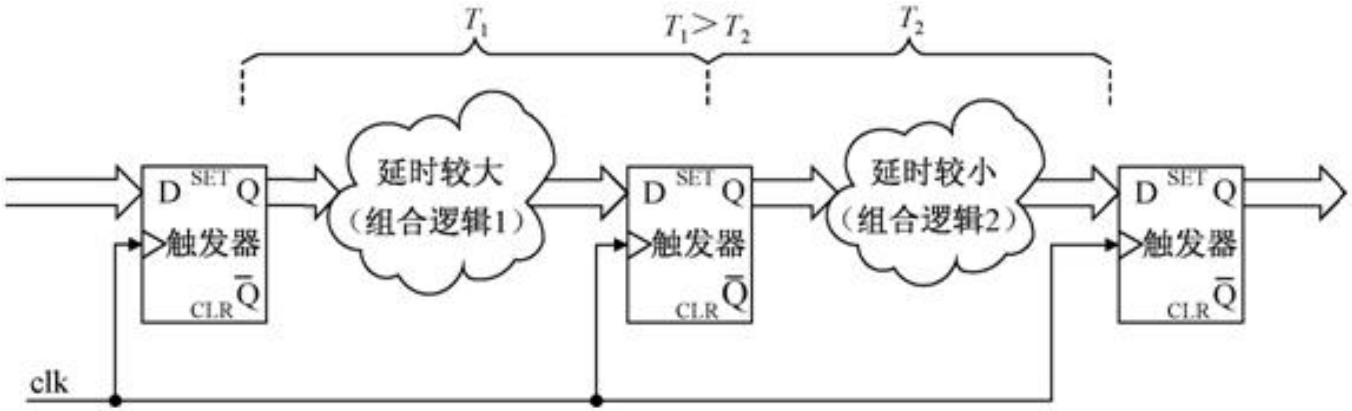

图 8-29　不合理的电路结构

8.5.7 关键路径法

习　题

8-5 详细讨论以下两个示例的代码表述方式，并给出它们各自的**RTL**电路，再根据电路情况进一步讨论含混合赋值类型语句过程的执行特点，从而深入了解阻塞和非阻塞赋值语句的用法特点。

示例 1

```
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
```

示例 2

```
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
```

习　题

8-6　以下程序和波形图（图**8-30**）分别是一个**3-8**译码器的设计程序及对应的时序波形。试根据**8.1.3**节讨论此项设计所依据的原理。

```
module DCD3_8  ( output reg[7:0]Q, input[2:0]D );
  always @(D)    begin   Q<= 8'b00000000;  Q[D]<=1;  end
endmodule
```

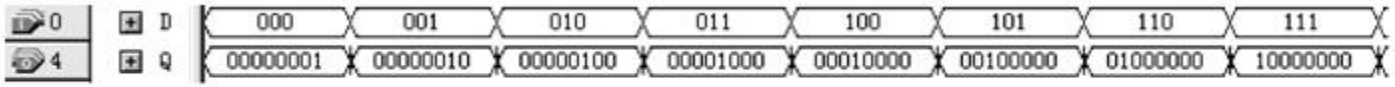

图 8-30　3-8 译码器的时序波形

习　题

8-7 利用资源共享的面积优化方法对下面程序进行优化（仅要求在面积上优化）。

```
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)   begin
    if (sel==1'b0)  Result <= A+B;  else  Result <= C+D;  end
endmodule
```

习　题

8-8　试通过优化逻辑的方式对图**8-31**所示的结构进行改进，给出**Verilog**代码和结构图。

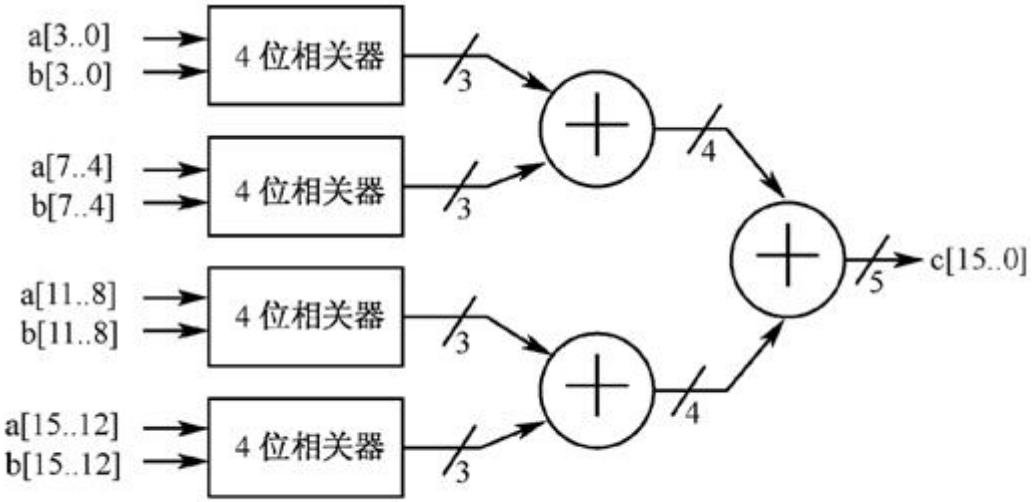

图 8-31　习题 8-8 图

实验与设计

8-1 4×4阵列键盘键信号检测电路设计

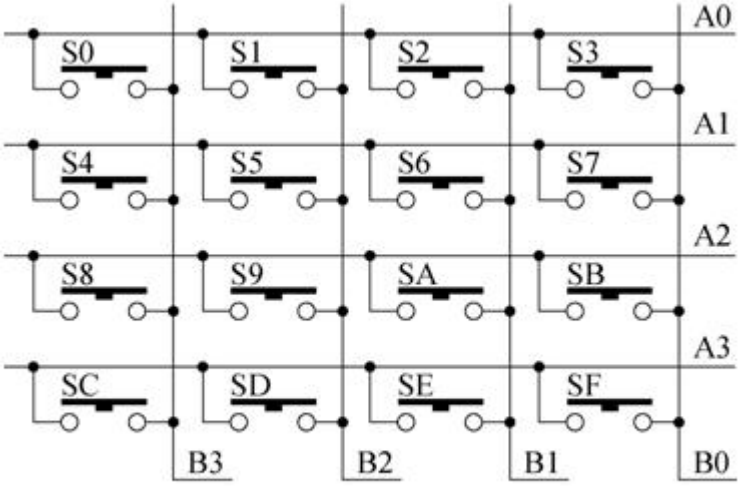

图 8-32 4×4 键盘电路

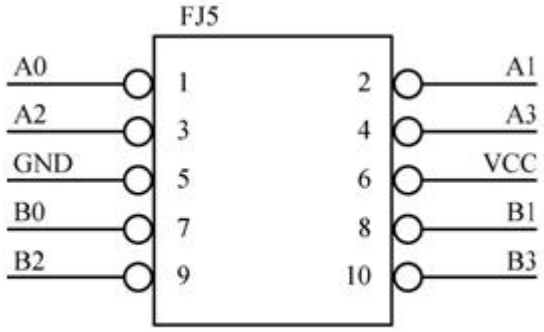

图 8-33 4×4 键盘的 10 芯接口

实验与设计

8-1 4×4阵列键盘键信号检测电路设计

【例 8-38】

```
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
```

实验与设计

8-2 直流电机综合测控系统设计

【例 8-39】

```
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
```

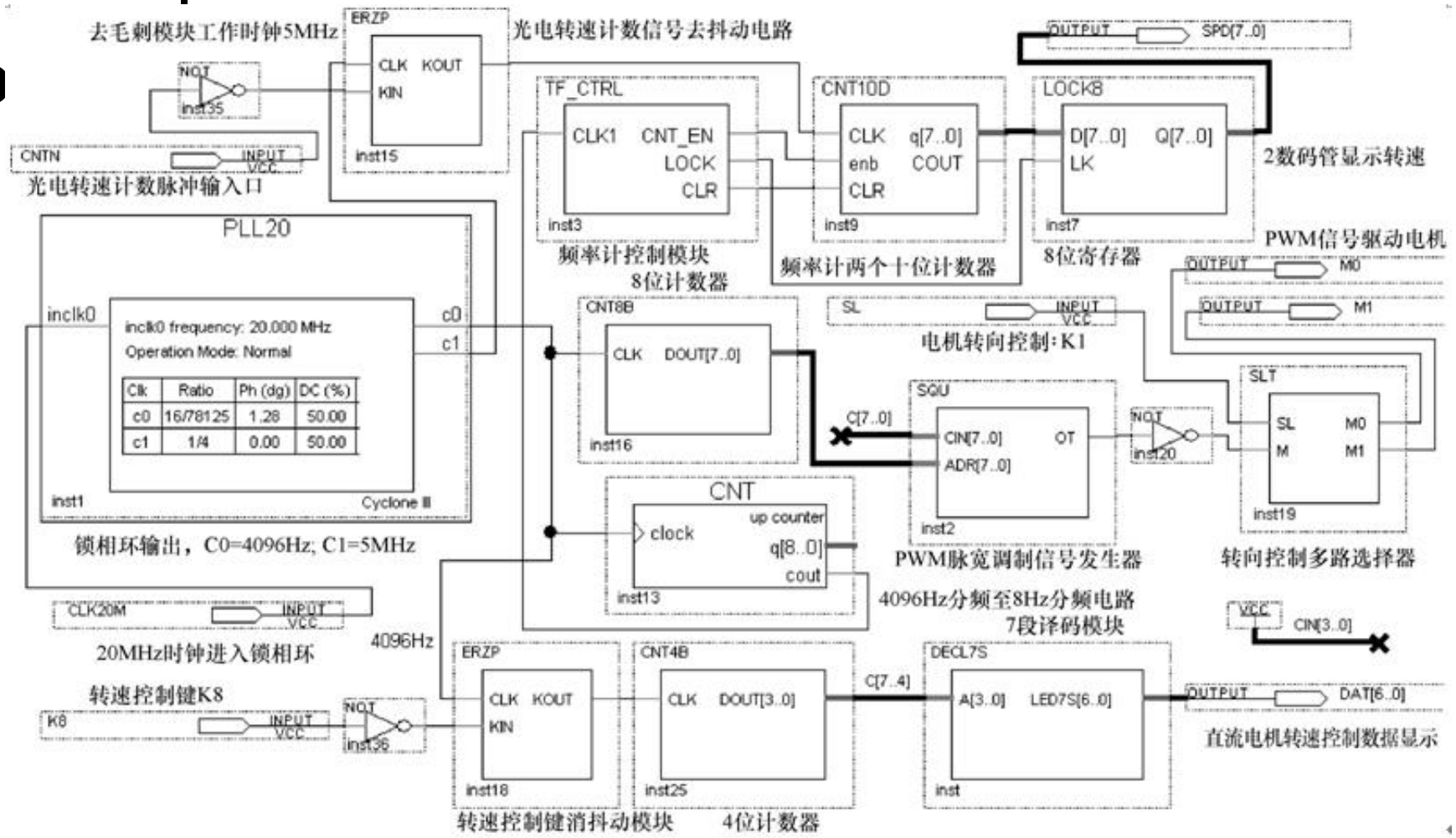

图 8-34　直流电机驱动控制电路顶层设计

实验与设计

8-4 线性反馈移位寄存器设计

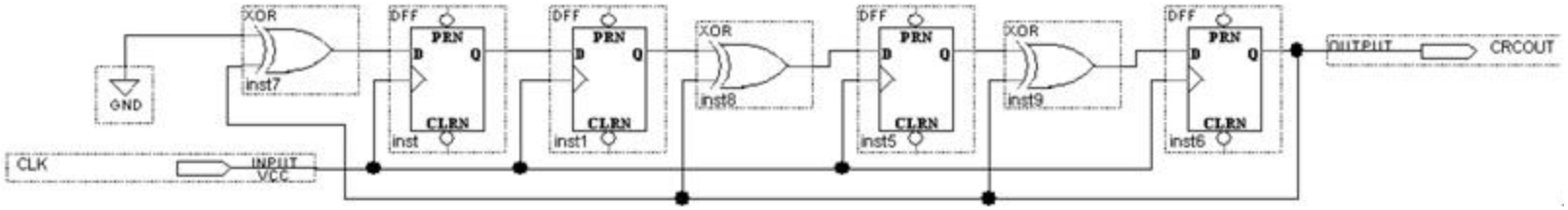

图 8-35　LFSR 举例

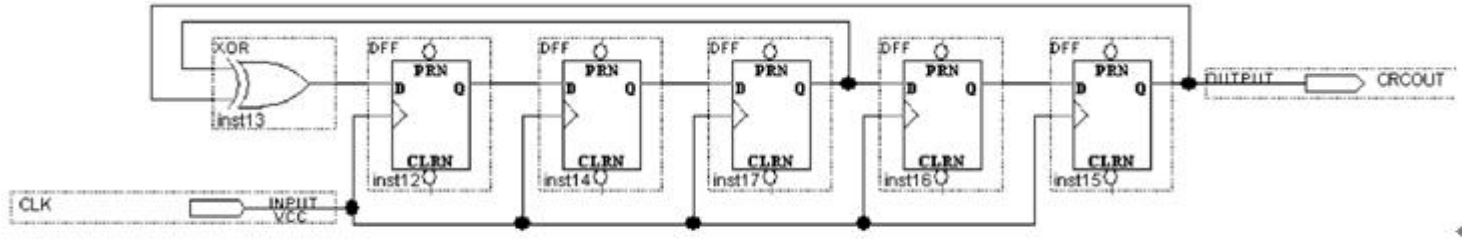

图 8-36　另一种 LFSR 结构

实验与设计

8-5 基于UART串口控制的模型电子琴设计

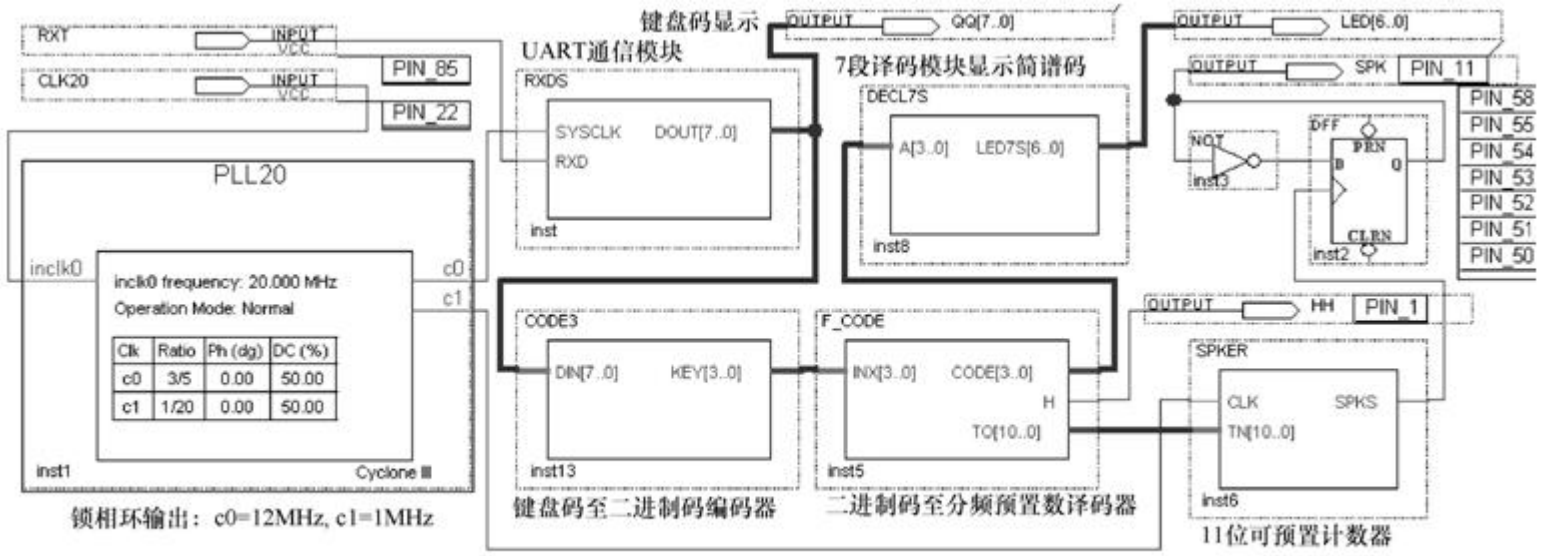

图 8-37 UART 串口控制模型电子琴电路顶层设计

实验与设计

8-5 基于UART串口控制的模型电子琴设计

【例 8-40】

```
module RXDS (SYSCLK, RXD, DOUT);
  input SYSCLK, RXD;   output[7:0] DOUT;
  wire[7:0] DOUT;  wire FRXD,  GATE;
  reg[9:0] B;  reg[3:0] R;  reg[15:0] J;
  reg GT, GTCLR, CCLK;
  always @(posedge SYSCLK or negedge GT)  begin : S1
    if (GT==1'b0)     J<=16'H0000;
     else  begin  if (J==16'H0068)  J<=16'H0000;
                  else        J<=J + 1;   end   end
  always @(J)   begin : S2
    if (J==16'H0039)   CCLK<=1'b1; else  CCLK<=1'b0;  end
  always @(posedge GATE or posedge GTCLR)  begin : S3
    if (GTCLR==1'b1)   R<=4'b0000;  else  R<=R+1;   end
  always @(GATE or R)   begin : S4
    if (R==4'b1010)  GTCLR<=~GATE;   else  GTCLR<=1'b0;  end
  always @(posedge GATE)        begin : S5
    B[9:0] <= {B[8:0], RXD};   end
  always @(posedge FRXD or posedge GTCLR)  begin : S6
    if (GTCLR==1'b1)   GT<=1'b0;  else  GT<=1'b1;  end
  assign DOUT = {1'b0, B[8:2]};
  assign GATE = GT & CCLK;        assign FRXD = ~RXD;
endmodule
```

实验与设计

8-5 基于UART串口控制的模型电子琴设计

【例 8-41】

```
module CODE3 (DIN, KEY);
  input[7:0] DIN;   output[3:0] KEY;  reg[3:0] KEY;
  always @(DIN)  begin
    case (DIN)
      8'b01000110 : KEY<=4'b0001; 8'b00100110 : KEY<=4'b0010;
      8'b01100110 : KEY<=4'b0011; 8'b00010110 : KEY<=4'b0100;
      8'b01010110 : KEY<=4'b0101; 8'b00110110 : KEY<=4'b0110;
      8'b01110110 : KEY<=4'b0111; 8'b00001110 : KEY<=4'b1000;
      8'b01001110 : KEY<=4'b1001; 8'b00000110 : KEY<=4'b0000;
          default : KEY<=4'b0000;
    endcase end
endmodule
```

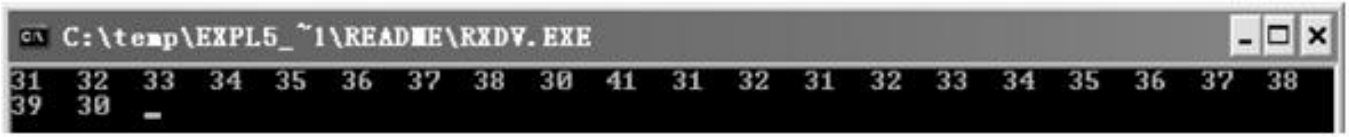

图 8-38　串口通信及键盘 ASCII 码显示程序窗口

实验与设计

8-6 PS2键盘控制模型电子琴电路设计

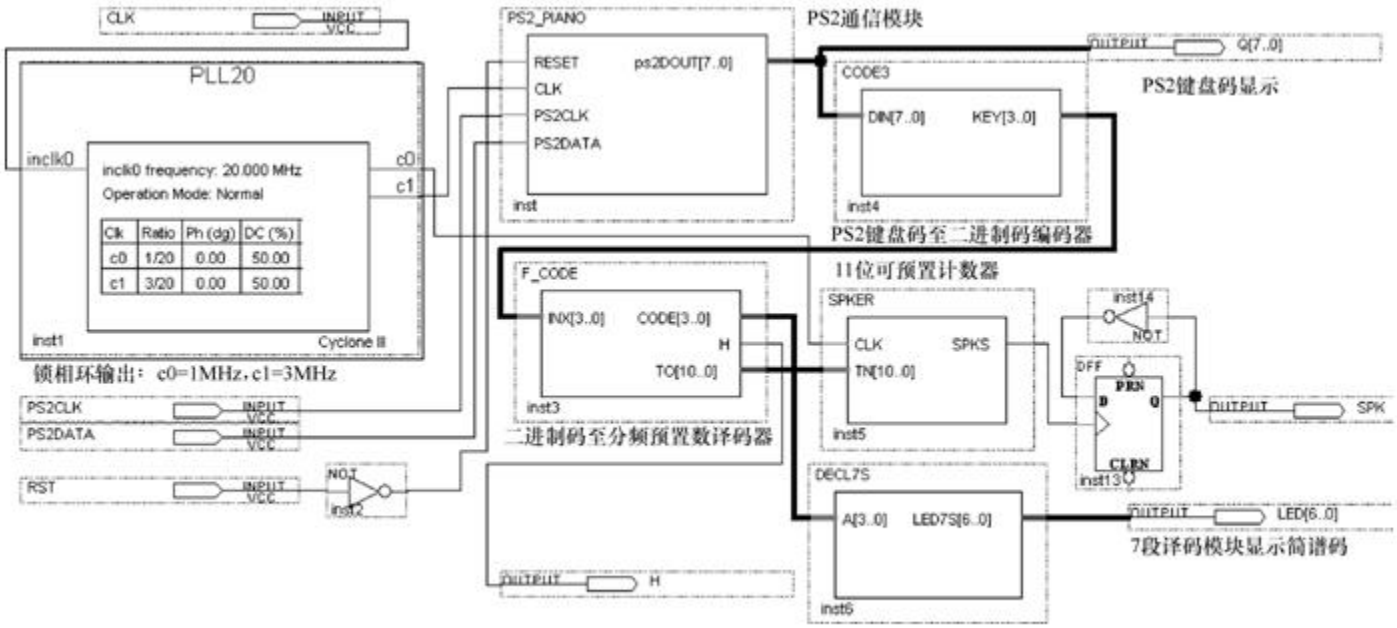

图 8-39　PS2 键盘控制模型电子琴电路顶层设计

实验与设计

8-6 PS2键盘控制模型电子琴电路设计

【例 8-42】

```
module CODE3 (input[7:0] DIN,  output reg[3:0] KEY);
     always @(DIN)  begin
      case (DIN)
       8'b00010110 : KEY<=4'b0001;      8'b00011110 : KEY<=4'b0010;
       8'b00100110 : KEY<=4'b0011;      8'b00100101 : KEY<=4'b0100;
       8'b00101110 : KEY<=4'b0101;      8'b00110110 : KEY<=4'b0110;
       8'b00111101 : KEY<=4'b0111;      8'b00111110 : KEY<=4'b1000;
       8'b01000101 : KEY<=4'b1001;      default : KEY<=4'b0000;
      endcase end
endmodule
```

实验与设计

8-6 PS2键盘控制模型电子琴电路设计

【例 8-43】

```
module PS2_PIANO(clk,kb_clk,kb_data,keycode,keydown,keyup,dataerror);
  input clk, kb_clk, kb_data;  output keydown, keyup, dataerror;
   output[7:0] keycode;
   reg[7:0] keycode, shiftdata;  reg keydown, keyup, dataerror;
   wire[7:0] kbcodereg;  reg[3:0] cnt;
   reg datacoming, kbclkfall, kbclkreg, parity, isfo;
   always @(posedge clk)   begin
      kbclkreg <= kb_clk;
      kbclkfall <= kbclkreg & (~kb_clk);    end
   always @(posedge clk)  begin
      if (kbclkfall == 1'b1 & datacoming == 1'b0 & kb_data == 1'b0)
      begin
        datacoming<=1'b1; cnt<=4'b0000; parity<=1'b0;   end
      else if (kbclkfall == 1'b1 & datacoming == 1'b1)
      begin  if (cnt == 9)
        begin
          if (kb_data == 1'b1)
           begin   datacoming<=1'b0; dataerror<=1'b0; end
          else  begin dataerror<=1'b1; end
          cnt <= cnt + 1;   end
        else if (cnt == 8)   begin if (kb_data == parity)
```

实验与设计

8-6 PS2键盘控制模型电子琴电路设计

```
            begin dataerror <= 1'b0; end
            else begin dataerror<=1'b1; end
            cnt <= cnt + 1;    end
          else   begin  shiftdata <= {kb_data, shiftdata[7:1]};
            parity <= parity ^ kb_data; cnt <= cnt + 1;   end
      end    end
    always @(posedge clk)  begin
      if (cnt == 10)   begin  if (shiftdata==8'b11110000)
        begin  isfo<=1'b1;   end
        else if (shiftdata!=8'b11100000)   begin  if (isfo==1'b1)
          begin  keyup<=1'b1;  keycode<=shiftdata;   end
          else begin  keydown<=1'b1;  keycode<=shiftdata;  end
        end      end
      else  begin  keyup<=1'b0;  keydown<=1'b0;  end    end
endmodule
```

实验与设计

8-6 PS2键盘控制模型电子琴电路设计

表 8-2 PS2 键盘键控与输出码对照表

Key	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Data	IC	32	21	23	24	2B	34	33	43	3B	42	4B	3A	31	44
Key	P	Q	R	S	T	U	V	W	X	Y	Z	0	1	2	3
Data	4D	15	2D	1B	2C	3C	2A	1D	22	35	1A	45	16	1E	26
Key	4	5	6	7	8	9	、	-	=	\	]	;	'	,	.
Data	25	2E	36	3D	3E	46	0E	4E	55	5D	5B	4C	52	41	49
Key	/	[	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	KP0
Data	4A	54	05	06	04	0C	03	0B	83	0A	01	09	78	07	70
Key	KP1	KP2	KP3	KP4	KP5	KP6	KP7	KP8	KP9	KP.	KP-	KP+	KP/	KP*	END
Data	69	72	7A	6B	73	74	6C	75	7D	71	7B	79	4A	7C	69

Key	BKSP	SPACE	TAB	CAPS	LSHFT	LCTRL	LCUI	LALT	R SHFT	R CTRL	RCUI
Data	66	29	0D	58	12	14	1F	11	59	14	27

Key	R ALT	APPS	ENTER	ESC	INSERT	HOME	PG UP	DELETE	PG DN	NUM
Data	11	2F	5A	76	70	6C	7D	71	7A	77

Key	UARROW	LARROW	D ARROW	R ARROW	KP EN	SCROLL	PRNT SCRN	PAUSE
Data	75	6B	72	74	5A	7E	12 7C	14