EDA技术

第8章 Verilog设计深入



《EDA技术实用教程——Verilog HDL版》
（第七版）
配套课件

EDA Lab

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

阻塞式（blocking）赋值“=”，目标变量名=驱动表达式

执行过程分为三步：

（1）阻塞本过程中的其他语句的执行，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。

EDA Lab

8.1 过程中的两类赋值语句



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;

赋值号左侧的[延时]是指对此整条语句执行的延时，即相隔与上一条语句执行的延时量。

赋值号右侧的[延时]是指，在赋值语句的右侧表达式得出运算结果后，延时一段指定的时间，然后再将运算结果赋值给赋值号左边的变量。

【例8-1】	【例8-2】
`Y1 = A^B;` `#6 Y2 = A&B|C;`	`Y1 = A^D;` `Y2 = #6 A&E|C;`

EDA Lab

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

非阻塞式（non-blocking）赋值 “<=”，目标变量名<=驱动表达式

非阻塞式赋值有2个阶段：

第1个阶段，计算“驱动表达式”的值，理论上是可以立即完成的；

第2个阶段，向目标变量进行赋值操作，需要一个δ延时段，在这个延迟时间段，允许其他同类赋值语句的执行或者说赋值操作。

EDA Lab

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8-3】

```
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

【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
```

EDA Lab

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
```

EDA Lab

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

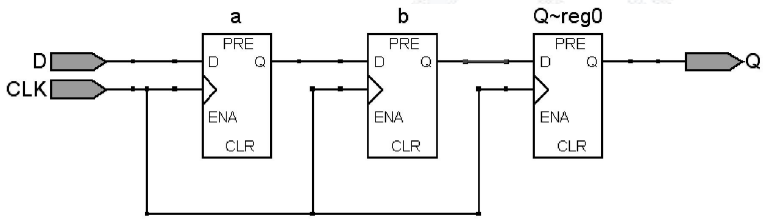


图8-1 例8-11综合后的RTL电路

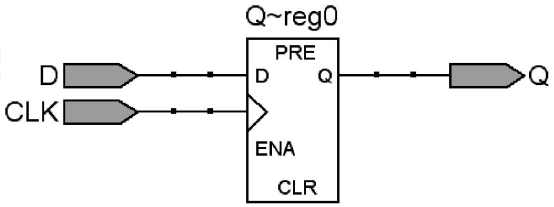


图8-2 例8-12综合后的RTL电路

可利用时序特性，倒置例8-12的赋值顺序，可得图8-1综合结果

```
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
```

EDA Lab

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

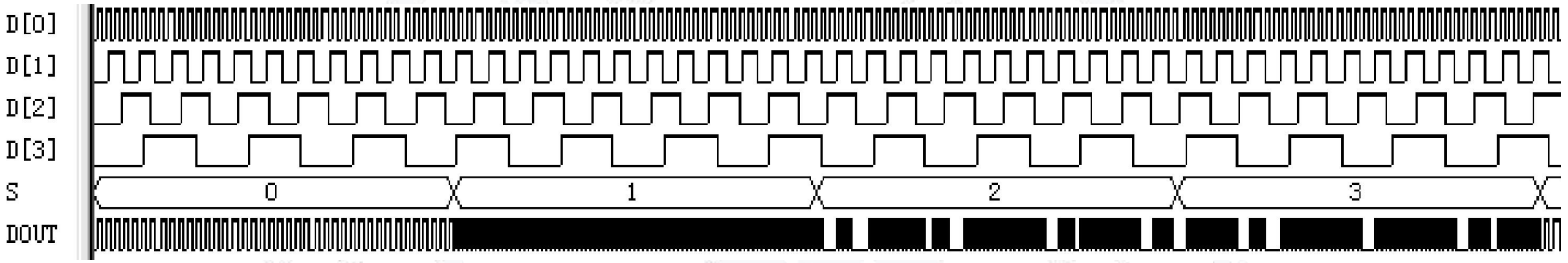


图8-3 例8-13的错误工作时序

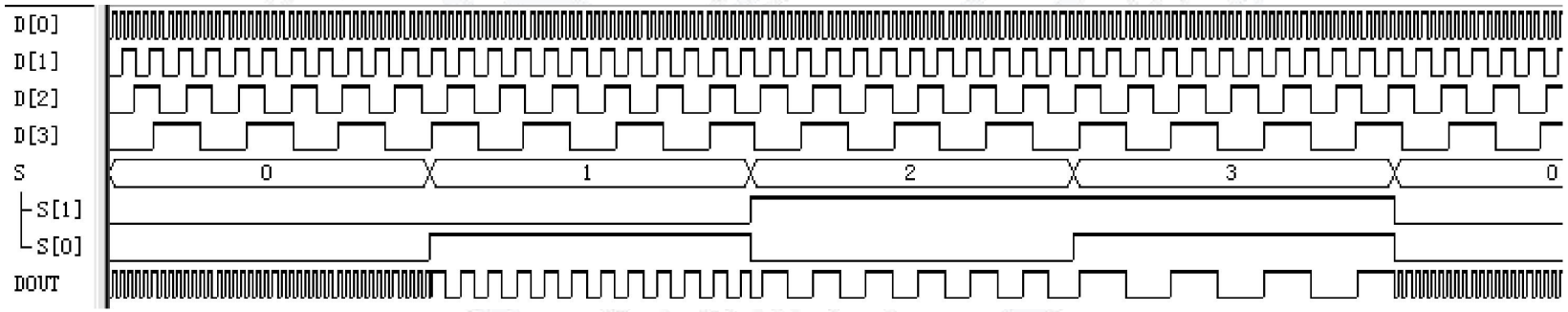


图8-4 例8-14的正确工作时序

8.2 过程语句归纳



8.2.1 过程语句应用总结

- 和软件语言不同，过程语句always的执行依赖于敏感信号的变化
- 在一个模块中可以包含任意个过程语句结构，所有的过程语句结构本身都属于并行语句，而由任一过程引导的各类语句结构都属于顺序语句
- 与引导顺序语句的always语句相对应，assign引导的语句属于并行语句

EDA Lab

8.2 过程语句归纳



8.2.1 过程语句应用总结

- 此语句被执行一次的唯一条件是等号右侧的某一或某些变量发生了改变，因此等号右边所有相关信号都是启动执行此赋值语句的敏感信号，与always引导过程语句相同
- 当一个模块中出现多个assign语句时，由于此类语句的并行性，同一目标变量名下是不允许有多个不同赋值表达式的

EDA Lab

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

不完整条件语句与时序电路的关系。利用不完整的条件语句的描述方式是Verilog描述时序电路的常用途径之一（通常用于描述电平触发类锁存器电路）

【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
```

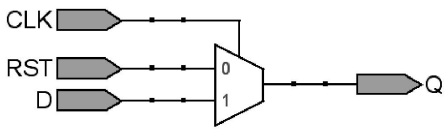


图8-5 例8-15的RTL图

【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
```

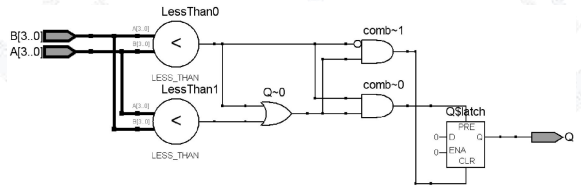


图8-6 例8-16的RTL图，输出口被加上了锁存器

【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
```

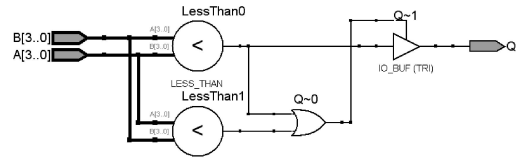


图8-7 例8-17的RLT电路图，输出口没有锁存器，是纯组合电路

EDA Lab

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

- 在利用条件语句进行纯组合电路设计时，如果没有充分考虑电路中所有可能出现的问题（条件），即没有列全所有的条件及其对应的处理方法，将导致不完整的条件语句的出现，从而综合出了设计者不希望的组合与时序电路的混合体。

EDA Lab

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

- 如果设计不当，case语句也会产生不希望的锁存器电路模块。由于case语句中的种种遗漏导致的时序模块的引入，在EDA软件的资源利用报告中是看不到的，因为此类情况下产生的时序模块都是电平敏感性锁存器，且多数情况下不是专用时序模块，有可能严重影响电路系统的速度，效率，甚至功能

EDA Lab

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

第（3）种if语句是一种多重if语句嵌套式条件句，其主要特点为，其任一分支顺序语句的执行条件是以上各分支所确定条件的相与，即相关条件同时成立。

EDA Lab

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
```

EDA Lab

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
```

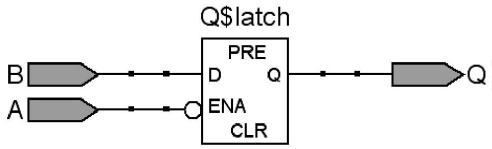


图8-9 例8-20/例8-21的RTL图

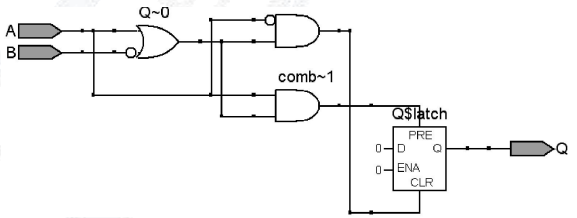


图8-10 例8-22的RTL图

Verilog默认，else与最近的没有else 的if相关联

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
```

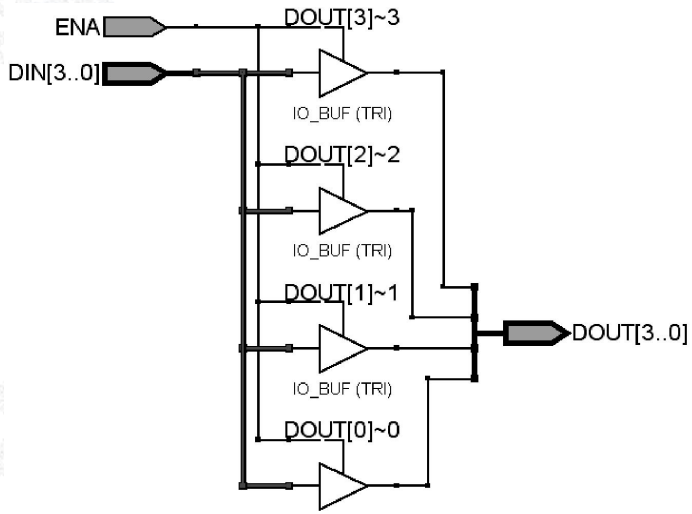


图8-11 4位三态控制门电路

三态门元件符号是tri

8.4 三态与双向端口设计

8.4.2 双向端口设计

用inout端口模式设计双向端口也必须考虑三态口的使用

【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
```

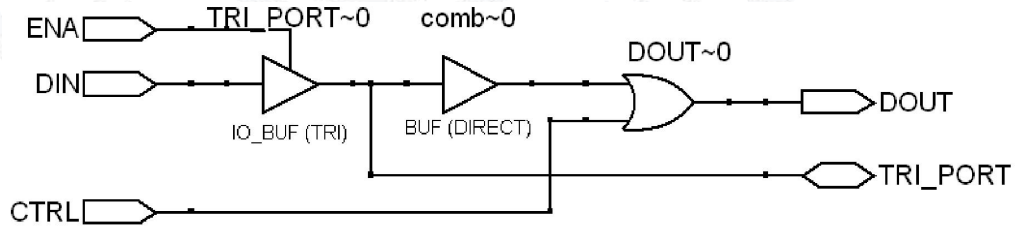


图8-12 例8-24的RTL图

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
```

【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
```

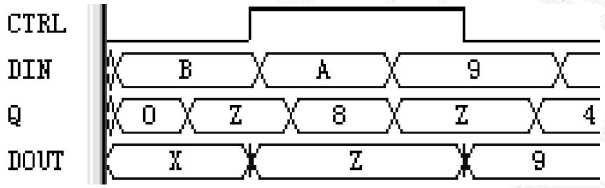


图8-13 例8-25的仿真波形图

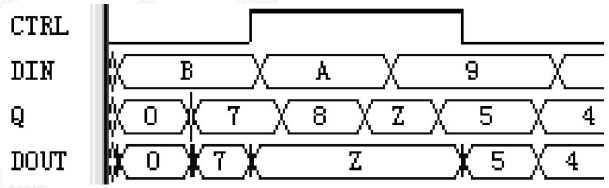


图8-14 例8-26的仿真波形图

8.4 三态与双向端口设计

8.4.2 双向端口设计

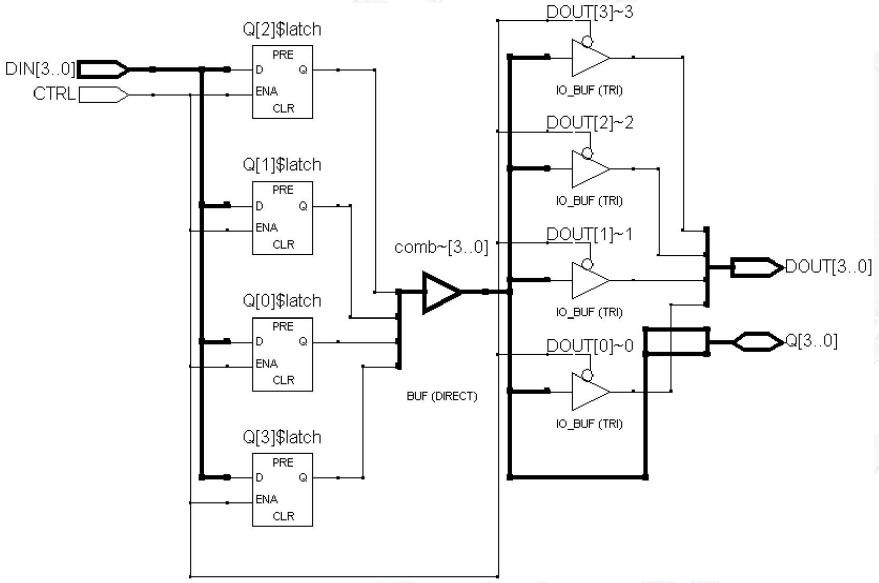


图8-15 例8-25的RTL图

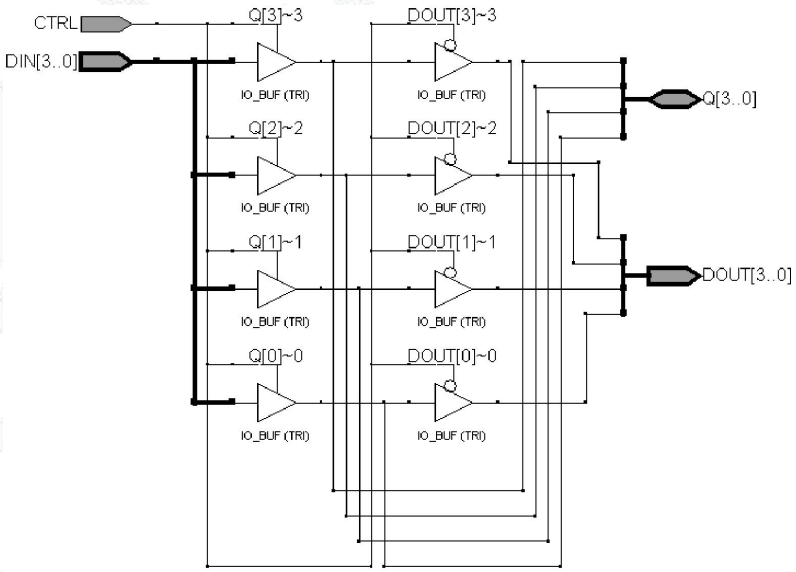


图8-16 例8-26的RLT图

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
```

EDA Lab

8.4 三态与双向端口设计

8.4.3 三态总线控制电路设计

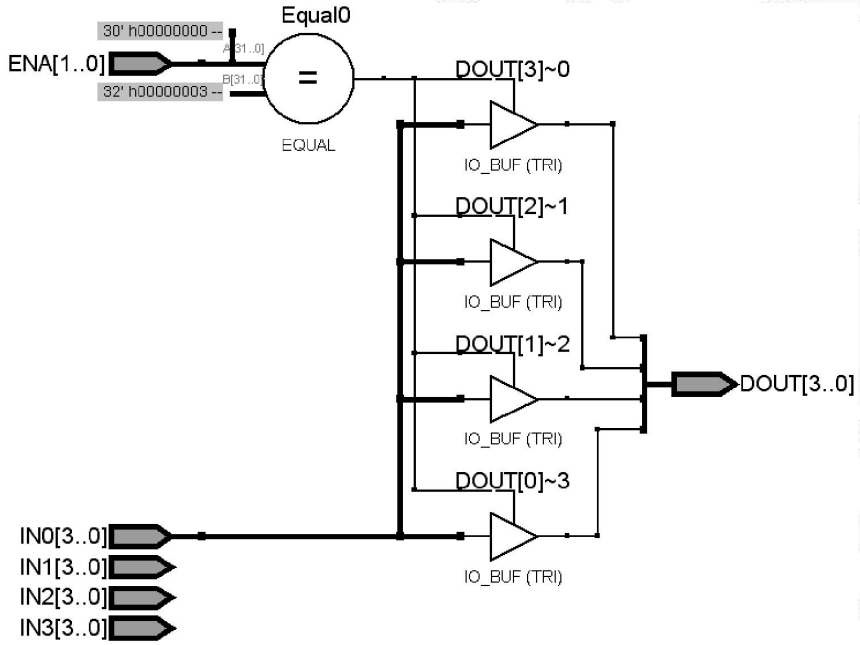


图8-17 例8-27的RTL图

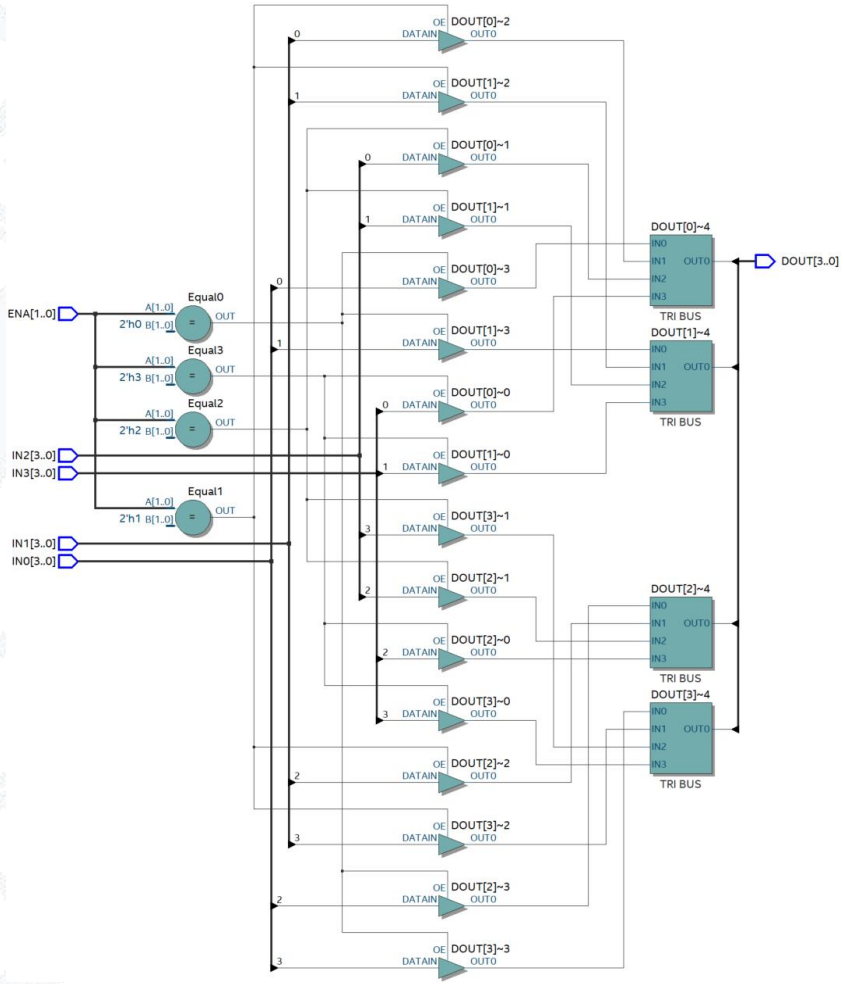


图8-18 例8-28的2位简化RTL电路图

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
```

要设计出能产生独立控制的多通道的三态总线电路结构，必须使用并行语句。

EDA Lab

8.5 Verilog系统设计优化



- 在EDA的硬件系统设计中，对于相同的功能要求，所实现的不同的电路构建往往会有迥异的**性能指标**，这主要表现在系统速度、资源利用率、可靠性等方面。

- 因此，EDA实用技术中必须包括优化设计和验证测试等方面的技术手段。此外，好的程序设计风格，即所谓编码风格（Coding Style）对于设计出拥有良好性能的系统也关系重大。

EDA Lab

8.5 Verilog系统设计优化



- FPGA/CPLD资源利用率的优化（习惯上称“面积”优化）具有以下实用意义：
  - 通过优化，可以使用规模更小的可编程器件，从而降低系统成本，提高性价比。
  - 对于某些PLD器件，当耗用资源过多时会严重影响优化的实现。
  - 为以后的技术升级，留下更多的可编程资源，方便添加产品的功能。
  - 对于多数可编程逻辑器件，资源耗用太多会使器件功耗显著上升。

EDA Lab

8.5 Verilog系统设计优化



8.5.1 资源共享

两个4位乘法器A0 * B，A1 * B，乘法器资源占用率较高

【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
```

EDA Lab

8.5 Verilog系统设计优化

8.5.1 资源共享

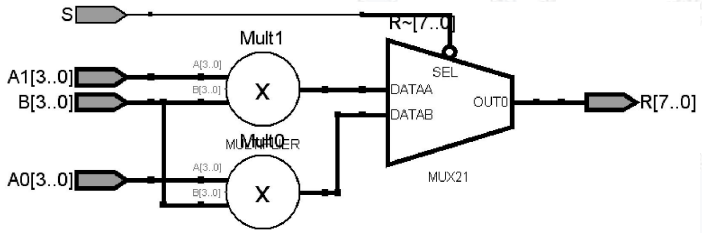


图8-19 先乘后选择的设计方法RTL结构

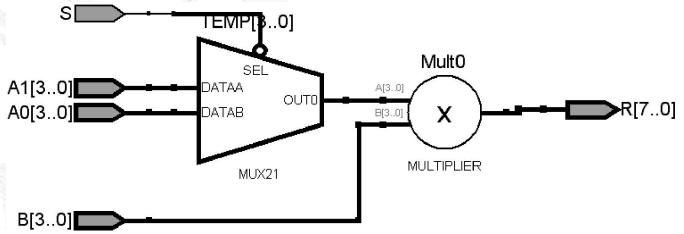


图8-20 先选择后乘设计方法RTL结构

9-2使用sel信号选择A0，A1作为乘法器的输入，B信号固定作为共享乘法器的输入，与9-1比逻辑结果上一样，却节省了一个代价高昂的乘法器，使得整个设计占用面积几乎减小一半。

此类资源优化思路具有一般性意义，主要针对数据通路中耗费逻辑资源比较多的模块，通过选择、复用的方式共享使用该模块，以减少该模块的使用个数，达到减少资源使用、优化面积的目的，也对应HDL特定目标的编码风格。

EDA Lab

8.5 Verilog系统设计优化

8.5.1 资源共享

并不是在任何情况下都能以此法实现资源优化。若对图9-3中输入与门之类的模块使用资源共享，通常是无意义的，有时甚至会增加资源的使用（多路选择器的面积显然要大于与门）。

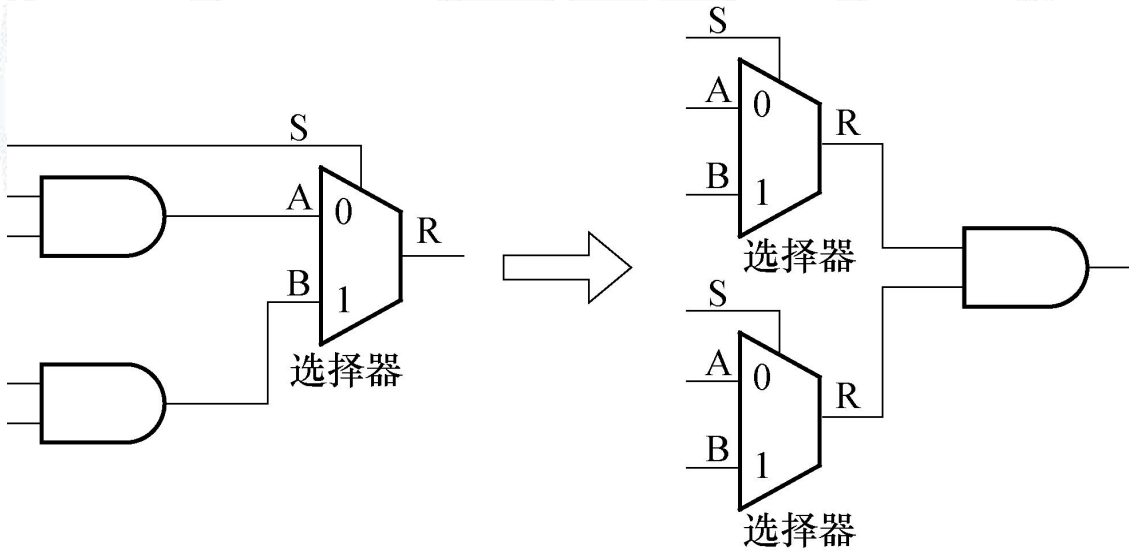


图8-21 资源共享反例

8.5 Verilog系统设计优化



8.5.2 逻辑优化

在实际的设计中常常会遇到两个数相乘，而其中一个为常数的情况。

【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
```

若适配于EPF10K20上，例8-32共耗费LC 167个，例8-33耗费LC 93个。

EDA Lab

8.5 Verilog系统设计优化



8.5.3 串行化

串行化指把原来耗用资源巨大、单时钟周期内完成的并行执行的逻辑块分割开来，提取出相同的逻辑模块（一般为组合逻辑块），在实际上复用该逻辑模块，用多个时钟周期完成相同的功能。代价是降低了工作速度。

【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
```

并行逻辑设计：共耗用4个8位乘法器和一些加法器。若适配于EP3C5，共耗用460个LC。

EDA Lab

8.5 Verilog系统设计优化



8.5.3 串行化

串行逻辑设计：比并行复杂，加入了时序电路进行控制

【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
```

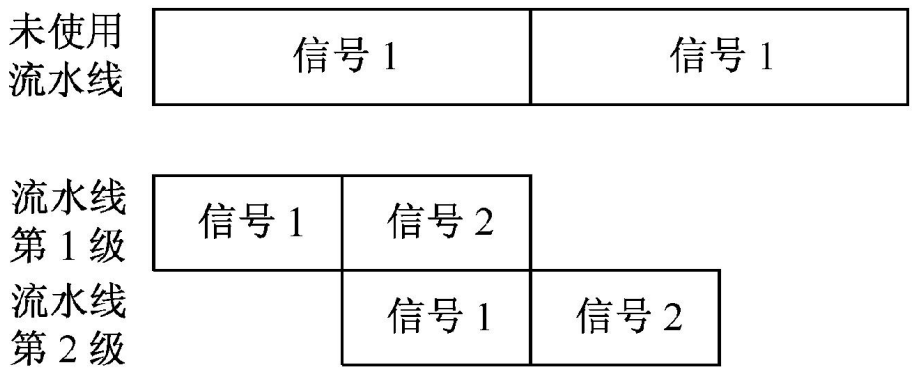
只需用1个8位乘法器和1个16位加法器。也加入了3位二进制计数器和2个大的选择器。使用相同的综合/适配设置，LC的耗用数为186个。

EDA Lab

8.5.4 流水线设计

流水线技术在速度优化中是最常用的技术之一。它能显著地提高设计电路的运行速度上限。

事实上在设计中加入流水线，并不会减少原设计中的总延时，有时甚至还会略微增加插入的寄存器的延时即信号同步的时间差，但却可以提高总体的运行速度，这并不存在矛盾。



8.5 Verilog系统设计优化

8.5.4 流水线设计

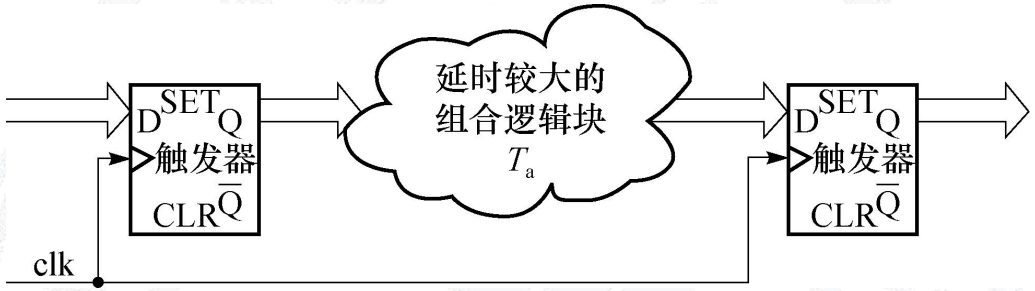


图8-22 未使用流水线

$$T_a \le T_{CLK} \Rightarrow \frac{1}{T_a} \ge \frac{1}{T_{CLK}} = f_{CLK}$$

时钟周期TCLK不能小于该设计从输入到输出的延时Ta

8.5 Verilog系统设计优化

8.5.4 流水线设计

使用二级流水线，把延时较大的组合逻辑块（Ta）切割成两块延时大致相等的组合逻辑块（T1和T2），设置T1=T2，T1+T2=Ta，在两个逻辑块中插入寄存器

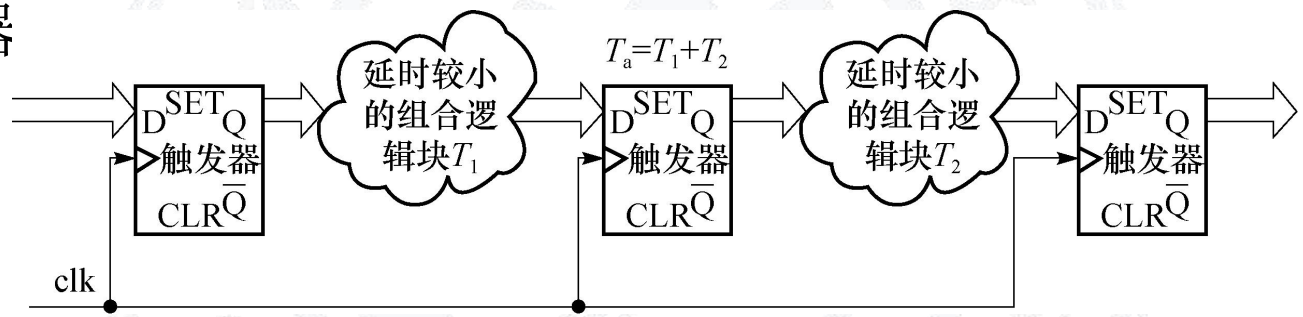


图8-23 使用流水线结构

最高工作频率 $F_{max} \approx F_{max1} \approx F_{max2} \approx 1/T_1$

$$T_1 \le T_{CLK} \Rightarrow \frac{1}{T_1} \ge \frac{1}{T_{CLK}} \Rightarrow \frac{2}{T_a} \ge \frac{1}{T_{CLK}} = f_{CLK}$$

8.5.4 流水线设计

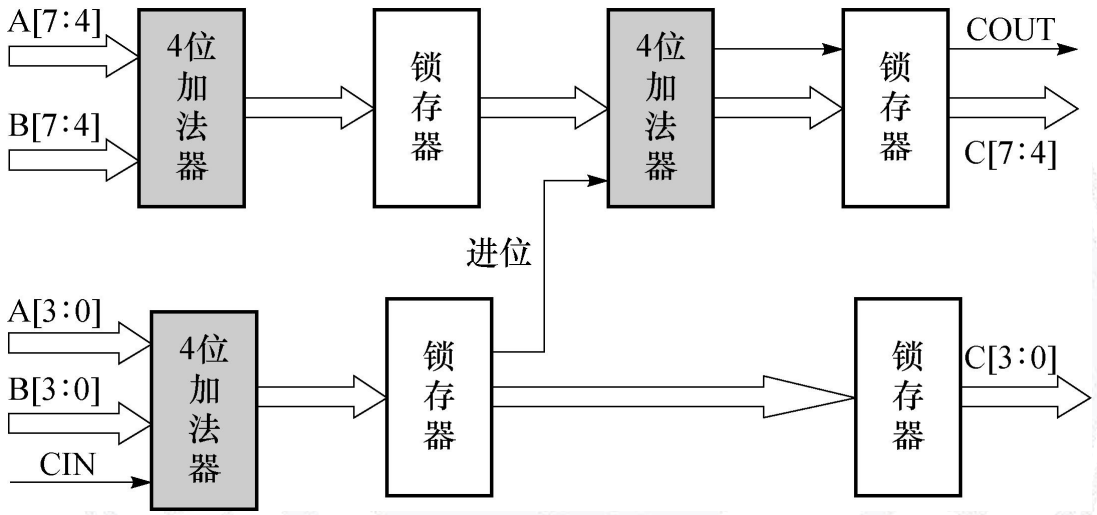


图8-25 8位加法器流水线工作图示

8位加法器流水线描述：将8位加法分成两个4位加法操作，其中用锁存器隔离。

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
```

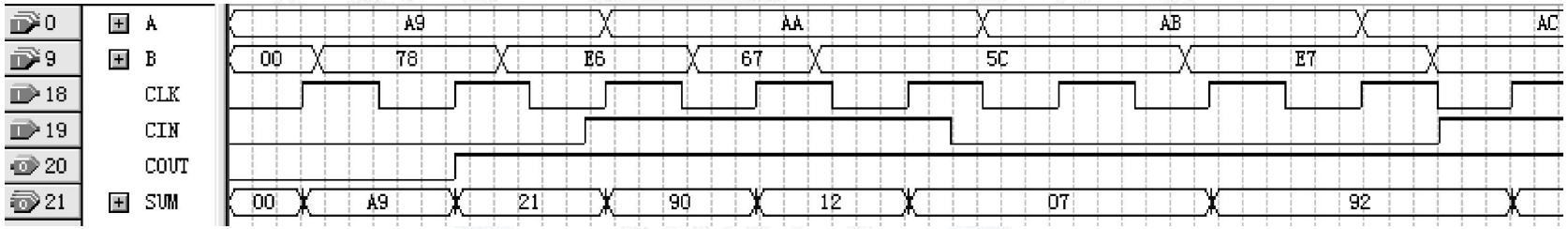


图8-26 例8-36的时序仿真波形

8.5.4 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
```

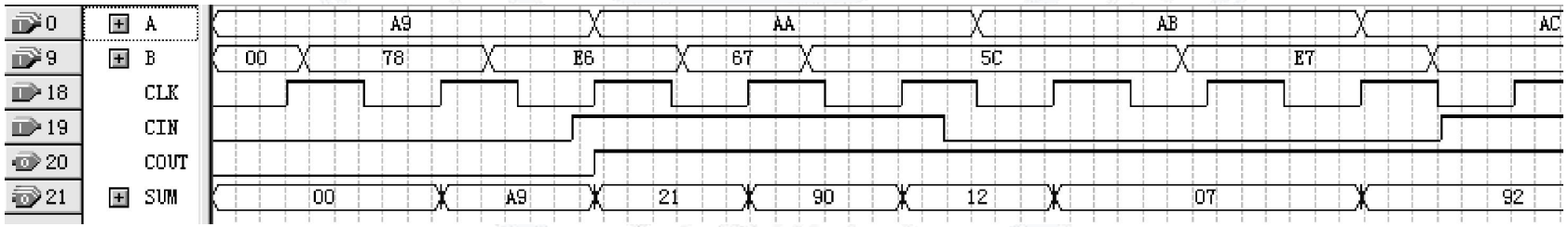


图8-27 例8-37的时序仿真波形

8.5.5 乒乓操作法

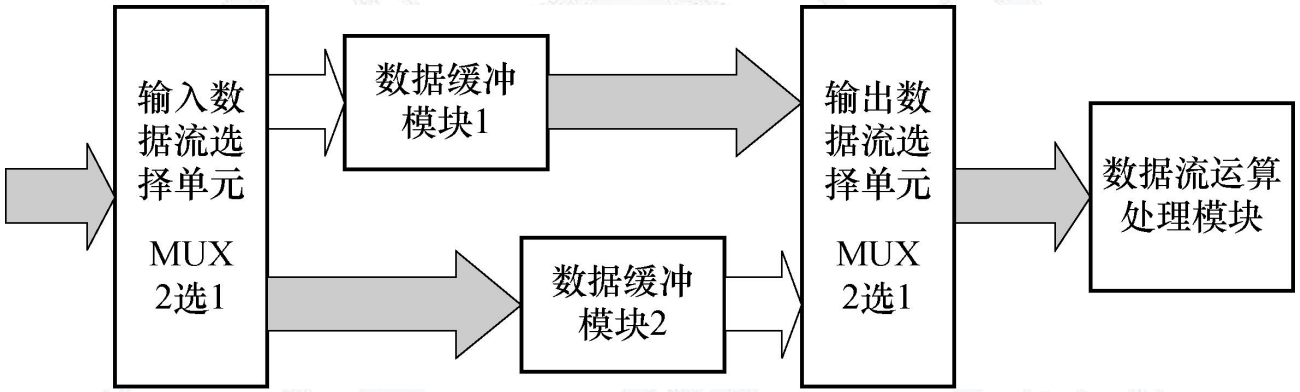


图8-28 乒乓操作数据缓存结构示意图

通过“输入数据流选择单元”和“输出数据流选择单元”按节拍、相互配合的切换，将经过缓冲的数据流“无缝”地（即没有时间停顿的）送到“数据流运算处理模块”进行处理。另一特点是可节约缓冲器空间。

8.5 Verilog系统设计优化

8.5.6 寄存器配平

总体的工作频率取决于最大的延时模块。

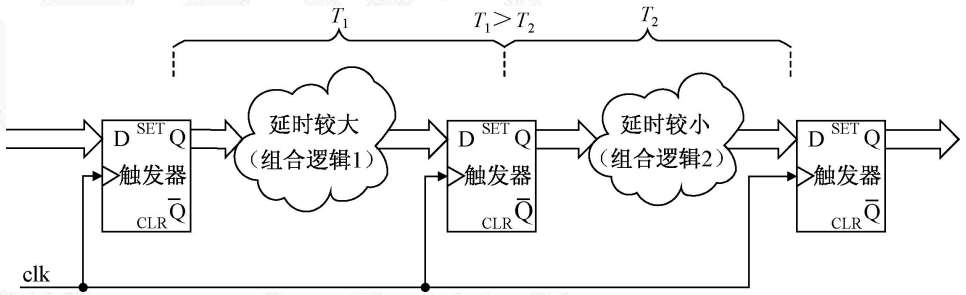


图8-29 不合理的电路结构

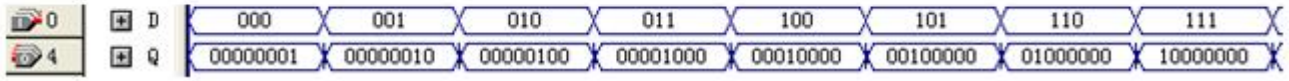
把组合逻辑1的部分逻辑转移到组合逻辑2中，小组合逻辑1的延时T1，使两部分延时相近。

寄存器配平法（Register Balancing）的关键是配平寄存器之间的组合逻辑块

习　题

示例 1	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 A=X1; D=X2; B<=D; C<=A; end Endmodule	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 DCD3_8  (output reg [7:0]Q, input[2:0]D);
  always @( D )    begin  Q<= 8'b00000000;  Q[D]<=1;  end
endmodule
```



实验与设计



- 8-1 4×4阵列键盘键信号检测电路设计
- 8-2 直流电机综合测控系统设计
- 8-3 采用流水线技术设计高速数字相关器
- 8-4 线性反馈移位寄存器设计
- 8-5 基于UART串口控制的模型电子琴设计
- 8-6 PS2键盘控制模型电子琴电路设计

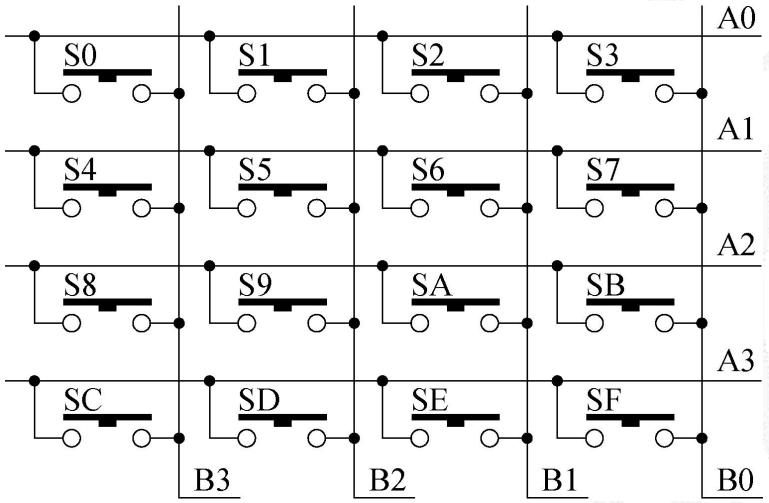
EDA Lab

8-1 4×4阵列键盘键信号检测电路设计

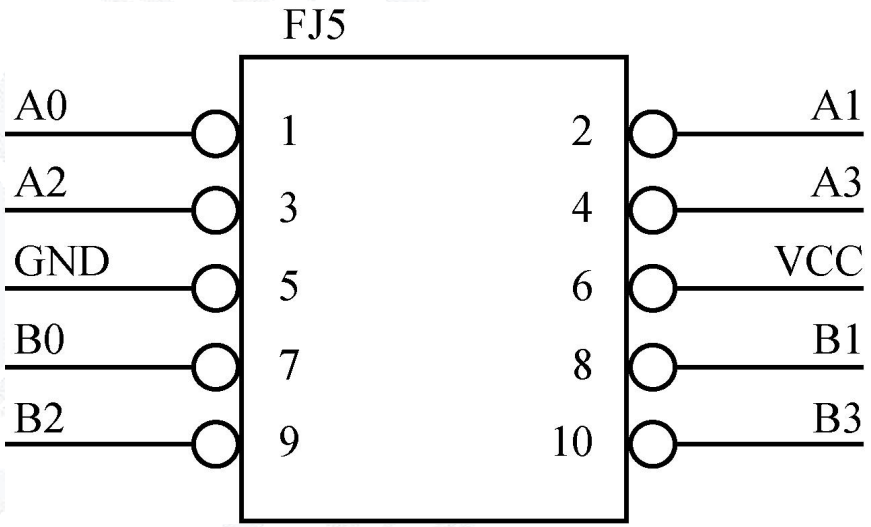
【例8-38】

```
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
```

8-1 4×4阵列键盘键信号检测电路设计



4×4键盘电路



4×4键盘的10芯接口

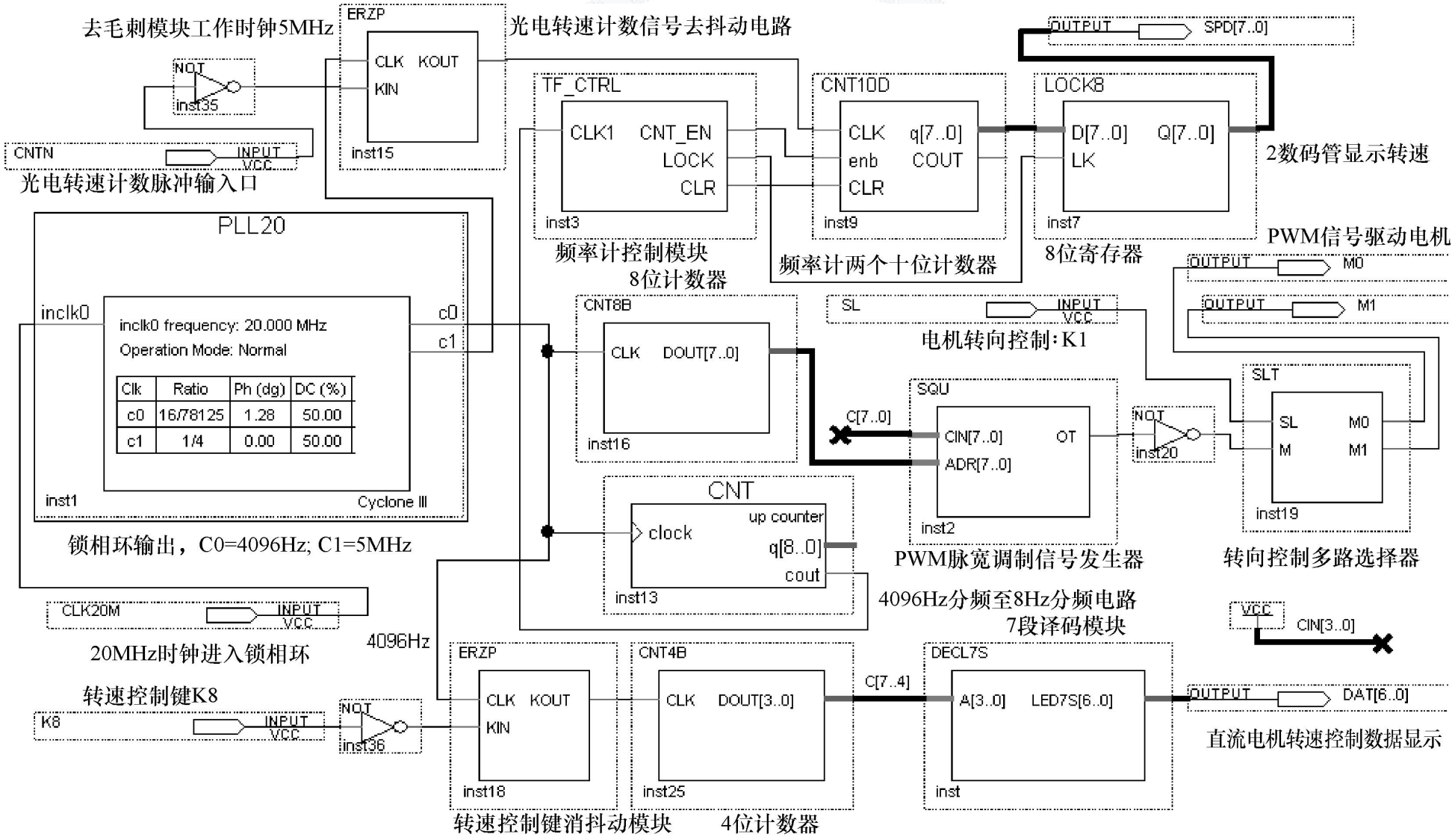
8-2直流电机综合测控系统设计

【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
```

EDA Lab

实验与设计


去毛刺模块工作时钟5MHz
ERZP
CLK KOUT
KIN
inst15
NOT
inst35
光电转速计数信号去抖动电路
CNTN
INPUT
VCC
光电转速计数脉冲输入口
TF_CTRL
CLK1 CNT_EN
LOCK
CLR
inst3
频率计控制模块
8位计数器
CNT10D
CLK q[7..0]
enb COUT
CLR
inst9
频率计两个十位计数器
OUTPUT SPD[7..0]
LOCK8
D[7..0] Q[7..0]
LK
inst7
8位寄存器
2数码管显示转速
PWM信号驱动电机
OUTPUT M0
OUTPUT M1
PLL20
inclk0
inclk0 frequency: 20.000 MHz
Operation Mode: Normal
c0
c1
Clk Ratio Ph (dg) DC (%)
c0 16/78125 1.28 50.00
c1 1/4 0.00 50.00
inst1
Cyclone III
锁相环输出，C0=4096Hz; C1=5MHz
CNT8B
CLK DOUT[7..0]
inst16
SL
INPUT
VCC
电机转向控制：K1
SQU
CIN[7..0] OT
ADR[7..0]
inst2
C[7..0]
PWM脉宽调制信号发生器
NOT
inst20
SLT
SL M0
M M1
inst19
转向控制多路选择器
CNT
clock
up counter
q[8..0]
cout
inst13
4096Hz分频至8Hz分频电路
7段译码模块
CLK20M
INPUT
VCC
20MHz时钟进入锁相环
4096Hz
转速控制键K8
K8
INPUT
VCC
NOT
inst36
ERZP
CLK KOUT
KIN
inst18
转速控制键消抖动模块
CNT4B
CLK DOUT[3..0]
inst25
4位计数器
C[7..4]
DECL7S
A[3..0] LED7S[6..0]
inst
VCC
CIN[3..0]
OUTPUT DAT[6..0]
直流电机转速控制数据显示

8-4 线性反馈移位寄存器设计

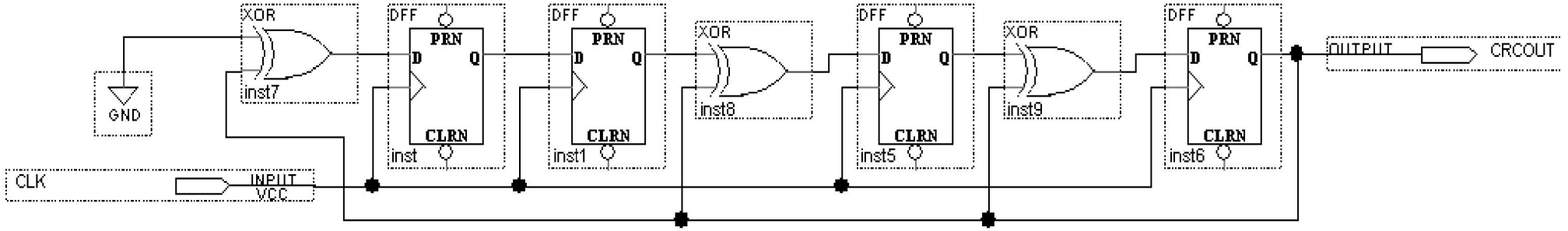


图8-35 LFSR举例

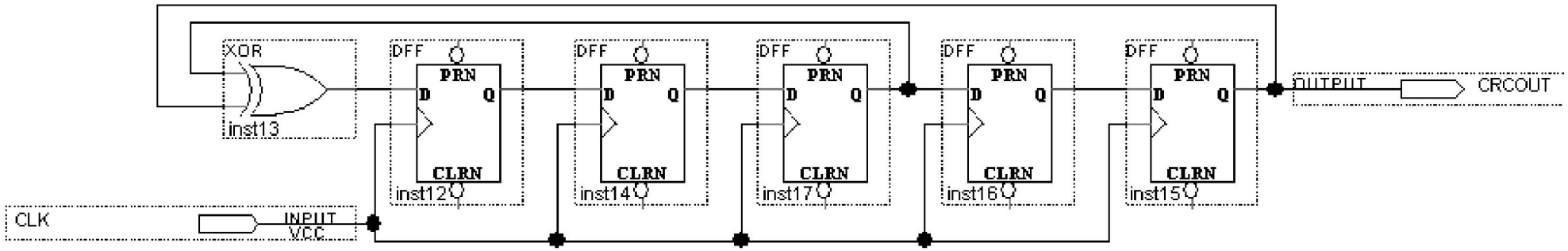


图8-36 另一种LFSR结构

8-5 基于UART串口控制的模型电子琴设计

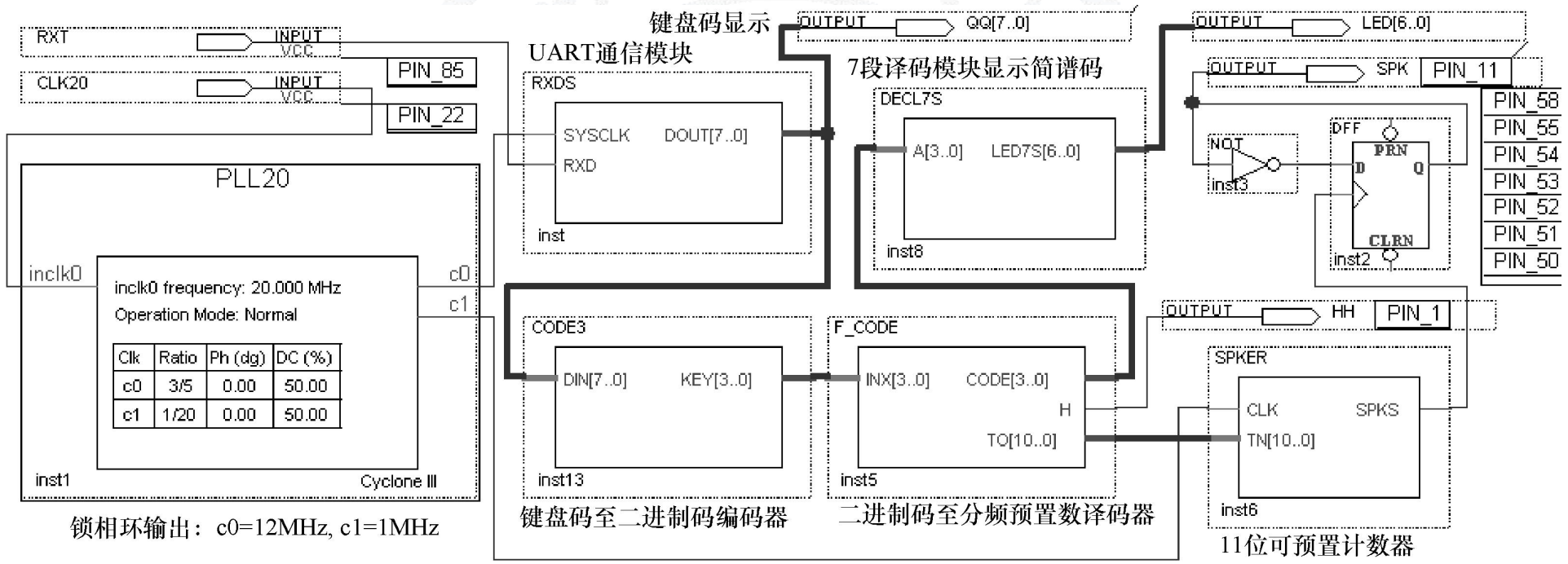


图8-37 UART串口控制模型电子琴电路顶层设计

实验与设计 8-5 基于UART串口控制的模型电子琴设计



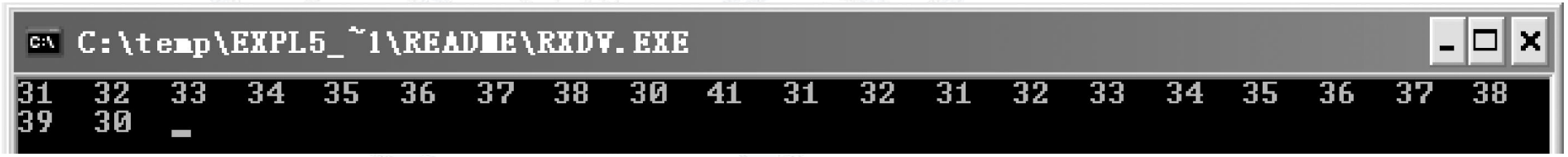
【例8-40】

```
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
```

8-5 基于UART串口控制的模型电子琴设计

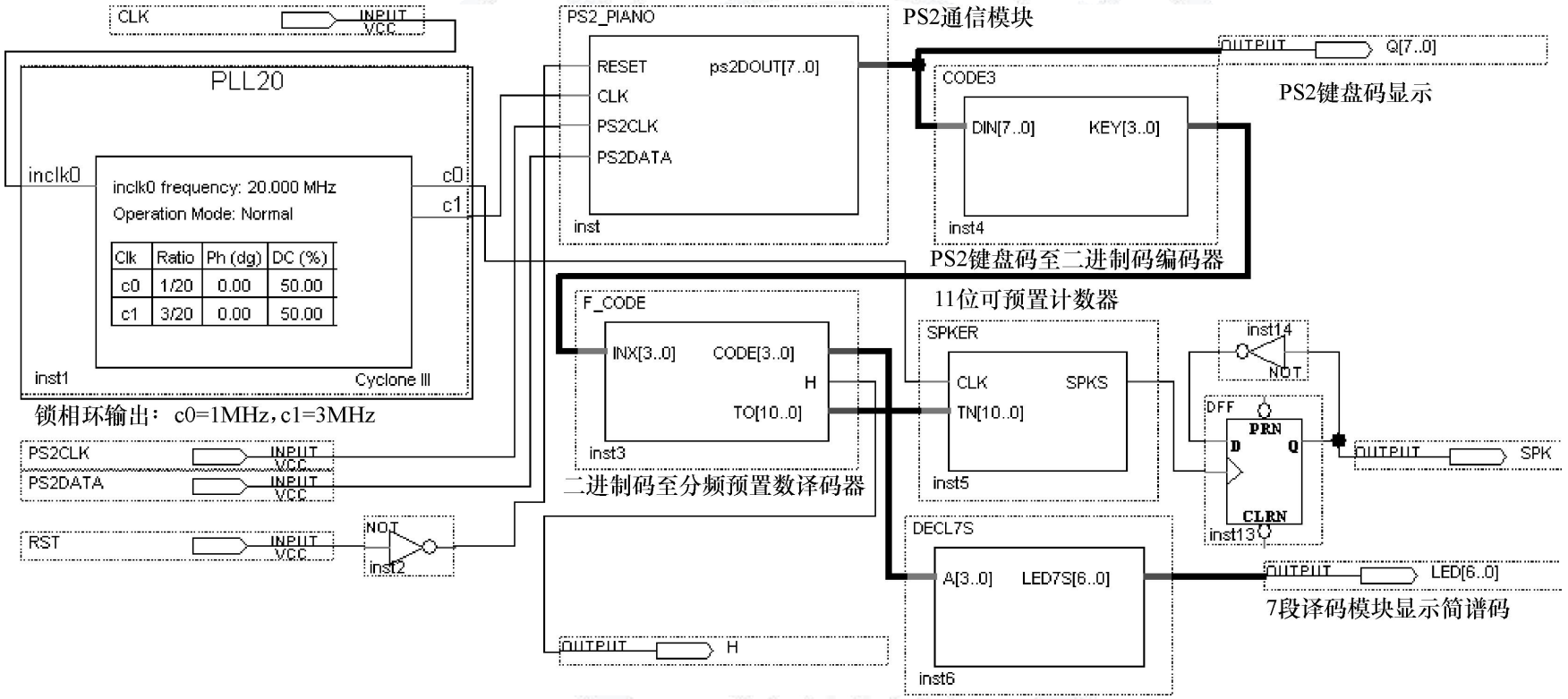
【例**8-41**】

```
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
```



串口通信及键盘ASCII码显示程序窗口

8-6 PS2键盘控制模型电子琴电路设计



PS2键盘控制模型电子琴电路顶层设计



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
```

EDA Lab

实验与设计



【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)
```

EDA Lab



【例8-43】

```
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)
        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
```

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