EDA技术实用教程

第8章

系统优化和时序分析

8.1 资源优化

8.1.1 资源共享

【例 8-1】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_arith.all;
ENTITY multmux IS
    PORT (A0, A1, B  : IN  std_logic_vector(3 downto 0);
                  s  : IN  std_logic;
                  R  : OUT std_logic_vector(7 downto 0));
END multmux;
ARCHITECTURE rtl OF multmux IS
BEGIN
    process(A0,A1,B,s)    begin
       if(s='0') then   R<=A0 * B;  else  R<=A1 * B;  end if;
    end process;
END rtl;
```

8.1 资源优化

8.1.1 资源共享

【例 8-2】

```
ARCHITECTURE rtl OF muxmult IS
   signal temp : std_logic_vector(3 downto 0);
BEGIN
   process(A0,A1,B,s)    begin
      if(s='0') then    temp<=A0; else  temp<=A1;   end if;
      R <= temp * B;
   end process;
END rtl;
```

8.1 资 源 优 化

8.1.1 资源共享

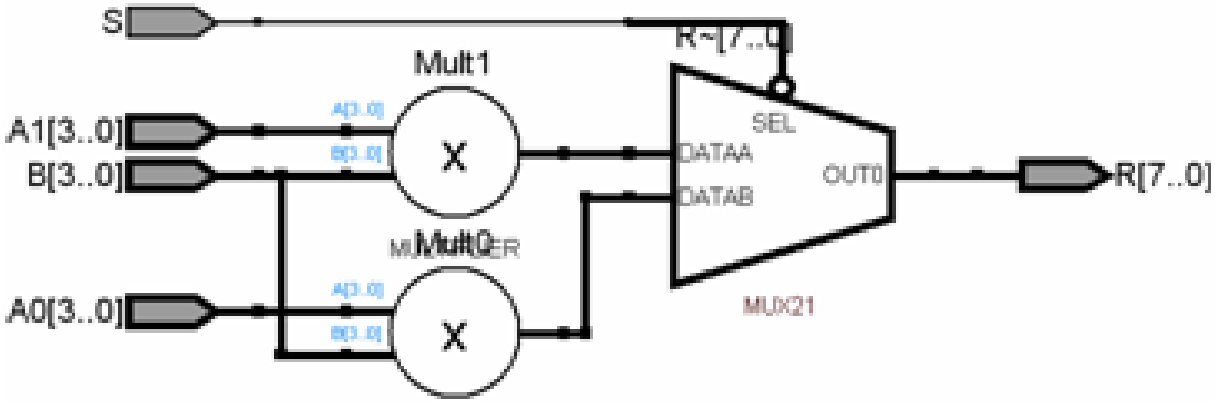

图 8-1 先乘后选择的设计方法 RTL 结构

8.1 资 源 优 化

8.1.1 资源共享

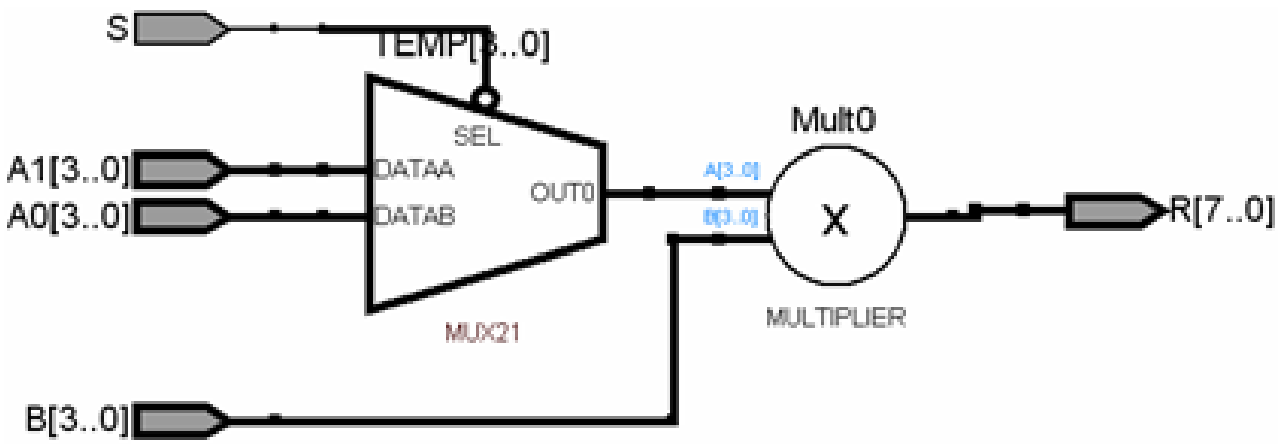

图 8-2　先选择后乘设计方法 RTL 结构

8.1 资 源 优 化

8.1.1 资源共享

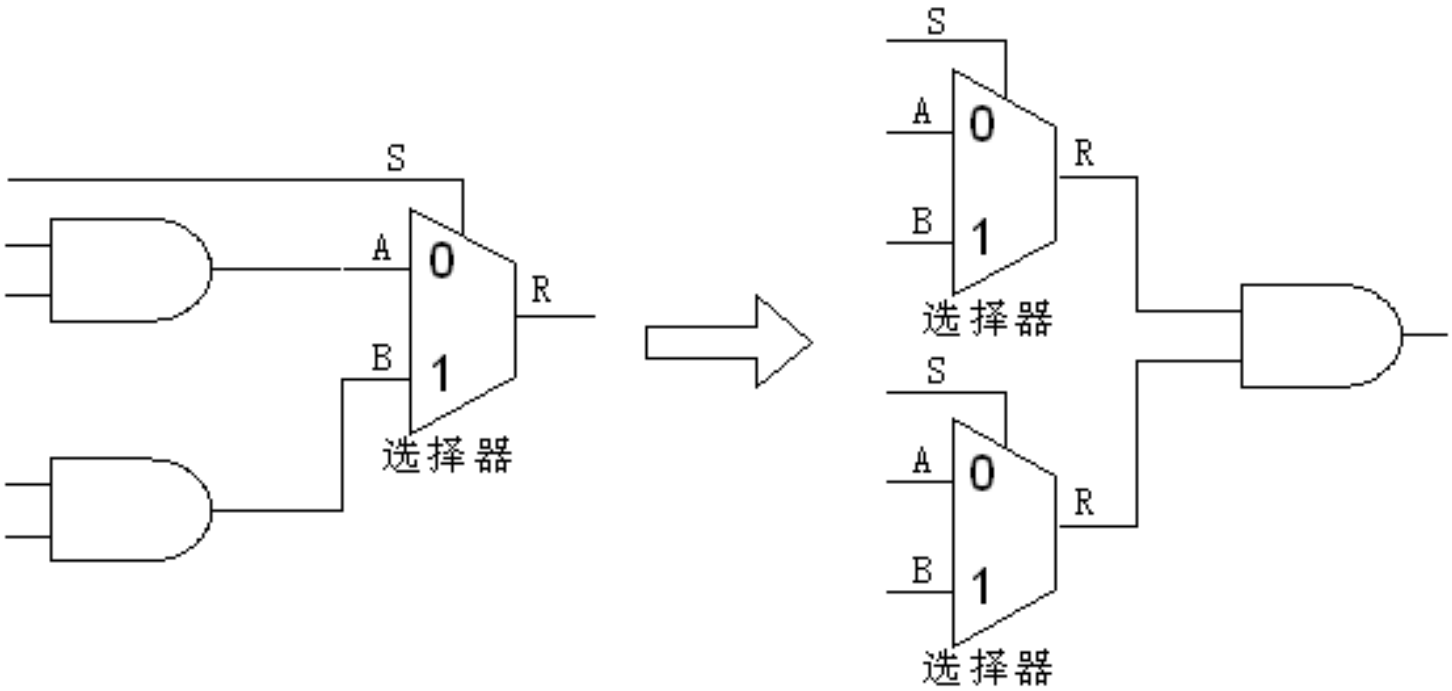

图 8-3　资源共享反例

8.1 资源优化

8.1.2 逻辑优化

【例 8-3】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
ENTITY mult1 IS
   PORT(clk : in std_logic;
        ma : In std_logic_vector(11 downto 0);
        mc : out std_logic_vector(23 downto 0));
END mult1;
ARCHITECTURE rtl OF mult1 IS
   signal ta, tb : std_logic_vector(11 downto 0);
BEGIN
process(clk)  begin
   if(clk'event and clk = '1') then
      ta <= ma;  tb <= "100110111001";   mc <= ta * tb;
   end if;
end process;
END rtl;
```

8.1 资源优化

8.1.2 逻辑优化

【例 8-4】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
ENTITY mult2 IS
   PORT(clk : in std_logic;
        ma : In std_logic_vector(11 downto 0);
        mc : out std_logic_vector(23 downto 0));
END mult2;
ARCHITECTURE rtl OF mult2 IS
   signal ta : std_logic_vector(11 downto 0);
   constant tb : std_logic_vector(11 downto 0) := "100110111001";
BEGIN
process(clk) begin
   if(clk'event and clk = '1') then  ta<=ma; mc<=ta*tb;  end if;
end process;
END rtl;
```

8.1 资源优化

8.1.3 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 8-5】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
ENTITY pmultadd IS
   PORT(clk : in std_logic;
        a0,a1,a2,a3 : in std_logic_vector(7 downto 0);
        b0,b1,b2,b3 : in std_logic_vector(7 downto 0);
        yout : out std_logic_vector(15 downto 0));
END pmultadd;
ARCHITECTURE p_arch OF pmultadd IS
BEGIN
process(clk) begin
   if(clk'event and clk = '1') then
      yout <= ((a0*b0)+(a1*b1))+((a2*b2)+(a3*b3));  end if;
end process;
END p_arch;
```

8.1 资源优化

8.1.3 串行化

【例 8-6】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;
ENTITY smultadd IS
    PORT(clk,  start : in std_logic;
         a0,a1,a2,a3 : In std_logic_vector(7 downto 0);
         b0,b1,b2,b3 : In std_logic_vector(7 downto 0);
         yout : out std_logic_vector(15 downto 0));
END smultadd;
ARCHITECTURE s_arch OF smultadd IS
    signal cnt : std_logic_vector(2 downto 0);
    signal tmpa, tmpb : std_logic_vector(7 downto 0);
    signal  tmp,  ytmp : std_logic_vector(15 downto 0);
  BEGIN
```

接下页

8.1 资源优化

接上页

```
tmpa <= a0 when cnt = 0 else
        a1 when cnt = 1 else
        a2 when cnt = 2 else
        a3 when cnt = 3 else    a0;
tmpb <= b0 when cnt = 0 else
        b1 when cnt = 1 else
        b2 when cnt = 2 else
        b3 when cnt = 3 else    b0;
tmp <= tmpa * tmpb;
process(clk) begin
   if(clk'event and clk = '1') then
      if(start = '1') then  cnt <= "000"; ytmp <= (others=>'0');
      elsif (cnt<4) then  cnt <= cnt + 1; ytmp <= ytmp + tmp;
      elsif (cnt = 4) then  yout <= ytmp;
      end if;   end if;
end process;
END s_arch;
```

8.2 速度优化

8.2.1 流水线设计

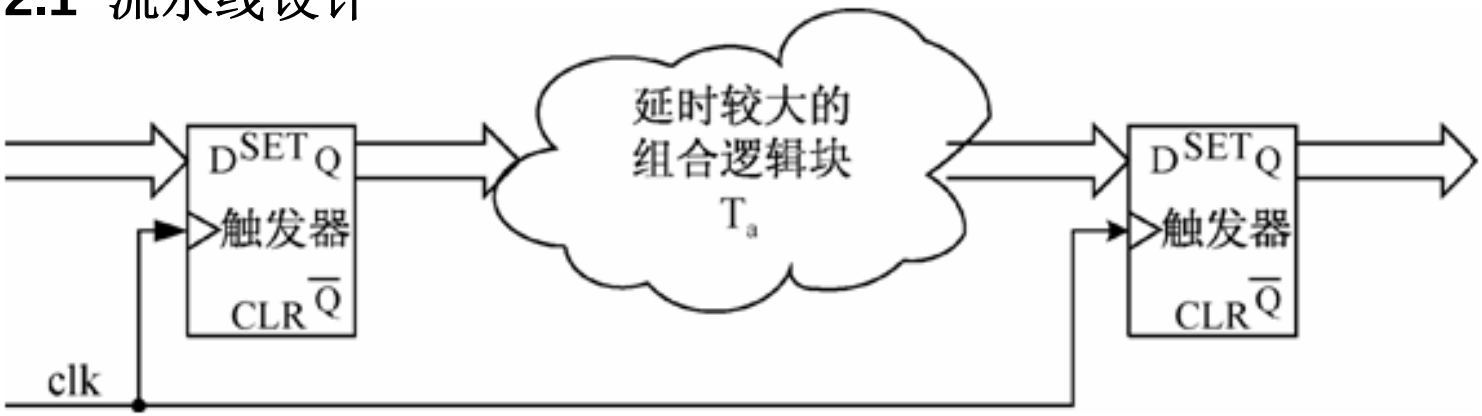

图 8-4 未使用流水线

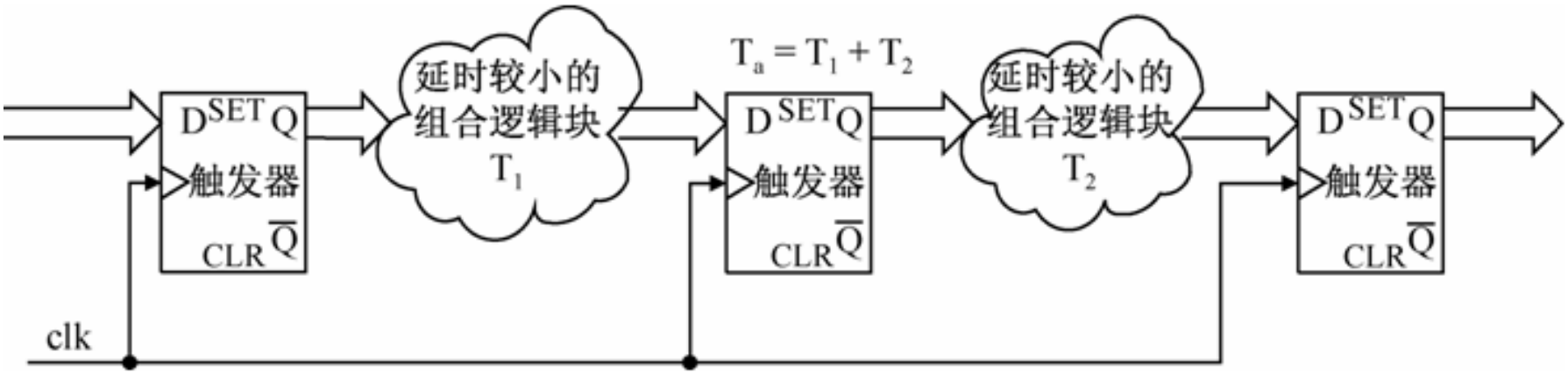

图 8-5 使用流水线结构

8.2 速 度 优 化

8.2.1 流水线设计

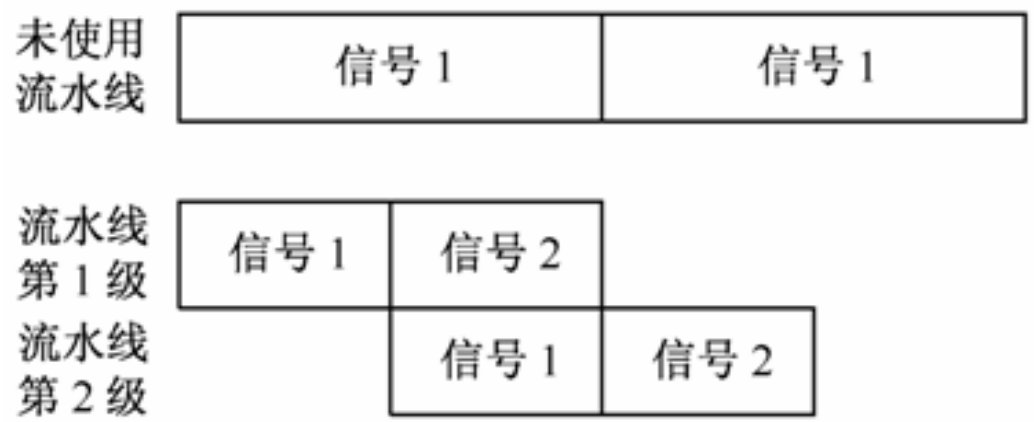

图 8-6 流水线工作图示

$$F_{max} \approx F_{max1} \approx F_{max2} \approx 1 / T_1$$

8.2 速度优化

8.2.1 流水线设计

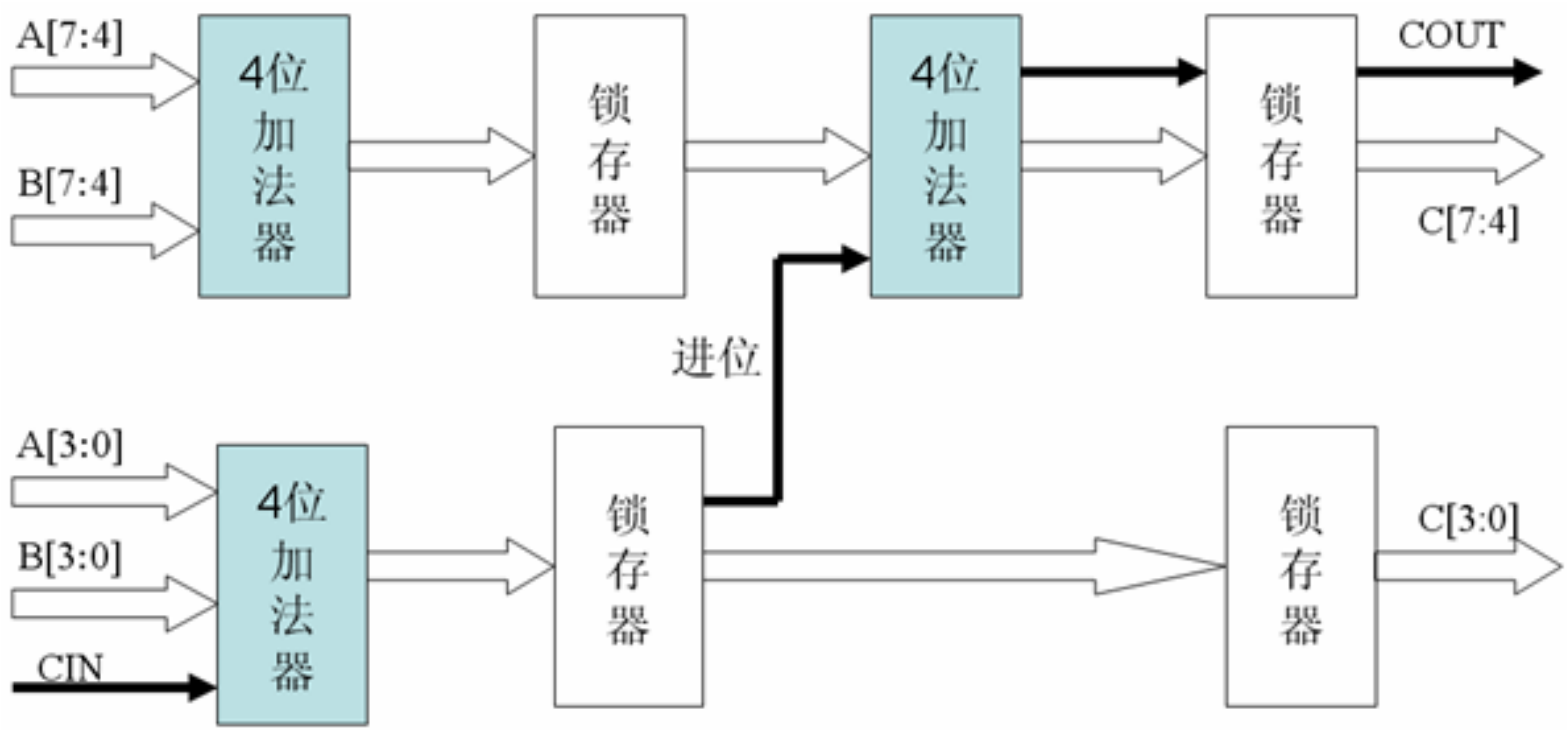

图 8-7　8 位加法器流水线工作图示

【例 8-7】普通加法器，EP3C5 FPGA 综合结果：LCs=10,REG=0,T=7.748ns.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_arith.all;
ENTITY ADDER8 IS
   PORT (A, B : IN std_logic_vector(7 downto 0);
         CLK,CIN  : IN  std_logic;
            COUT  : OUT  std_logic;
             SUM  : OUT std_logic_vector(7 downto 0));
END ADDER8 ;
ARCHITECTURE rtl OF ADDER8  IS
SIGNAL SUMC,A0,B0 : std_logic_vector(8 downto 0);
BEGIN
     A0<='0'& A ; B0<='0' & B ;
  process(CLK)   begin
     IF (RISING_EDGE(CLK)) THEN   SUMC <= A0+B0+CIN;  END IF;
  end process;
     COUT<=SUMC(8) ; SUM<=SUMC(7 downto 0);
END rtl;
```

【例 8-8】流水线加法器，EP3C5 综合结果：CLK=275MHz,T=3.63ns, LCs=24, REG=2

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_arith.all;
ENTITY CNT10 IS
    PORT (A, B : IN std_logic_vector(7 downto 0);
          CLK,CIN  : IN  std_logic;
             COUT  : OUT  std_logic;
              SUM  : OUT std_logic_vector(7 downto 0));
END CNT10 ;
ARCHITECTURE rtl OF CNT10  IS
SIGNAL SUMC,A9,B9 : std_logic_vector(8 downto 0);
SIGNAL AB5,A5,B5,TA,TB,S : std_logic_vector(4 downto 0);
BEGIN
     A5<='0'& A(3 downto 0); B5<='0'& B(3 downto 0);
 process(CLK)   begin
     IF (RISING_EDGE(CLK)) THEN
       AB5<=A5+B5+CIN;  SUM(3 downto 0)<=AB5(3 downto 0); END IF;
   end process;
 process(CLK)   begin
     IF (RISING_EDGE(CLK)) THEN
     S<=('0'& A(7 downto 4))+('0'& B(7 downto 4))+ AB5(4); END IF;
   end process;
     COUT<=S(4) ; SUM(7 downto 4)<=S(3 downto 0);
END rtl;
```

8.2 速 度 优 化

8.2.1 流水线设计

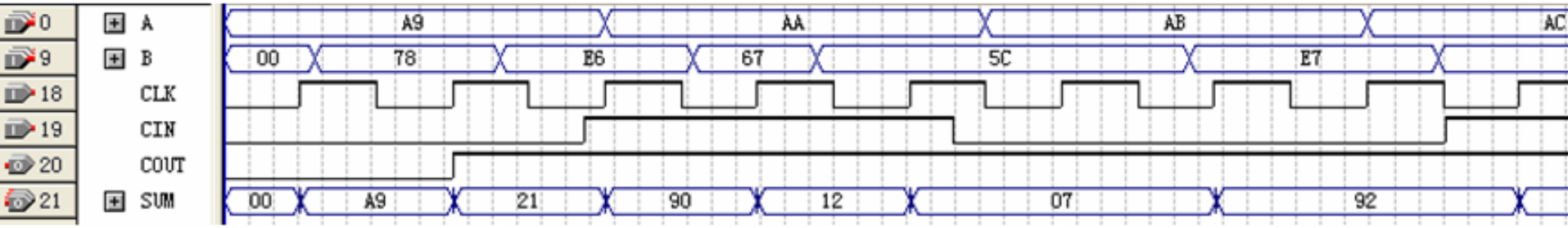

图 8-8　例 8-7 的时序仿真波形

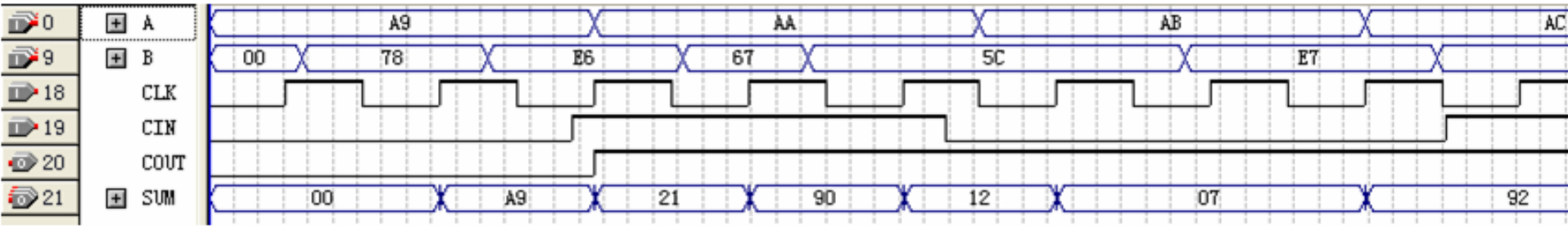

图 8-9　例 8-8 的时序仿真波形

8.2 速 度 优 化

8.2.2 寄存器配平

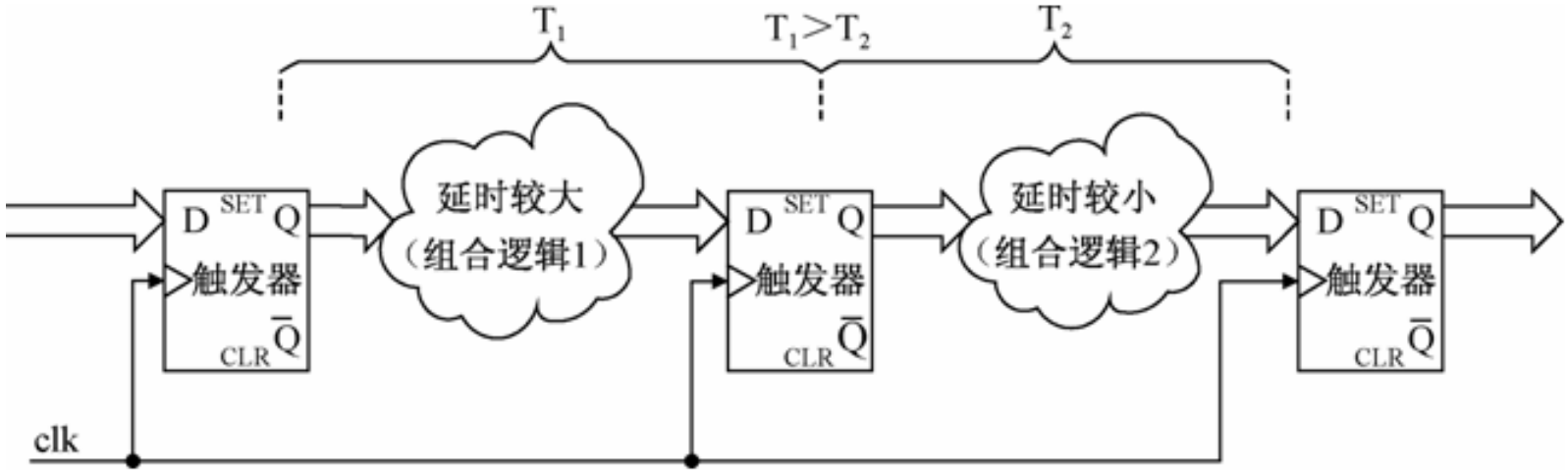

图 8-10 不合理的电路结构

8.2 速 度 优 化

8.2.2 寄存器配平

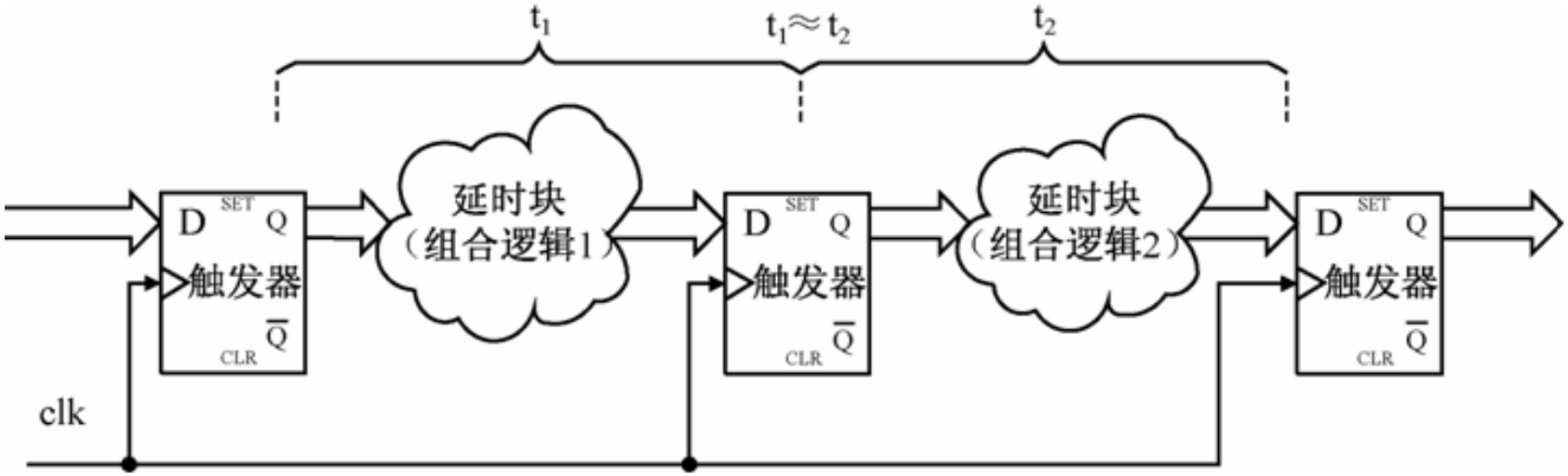

图 8-11 寄存器配平后的结构

8.2 速 度 优 化

8.2.3 关键路径法

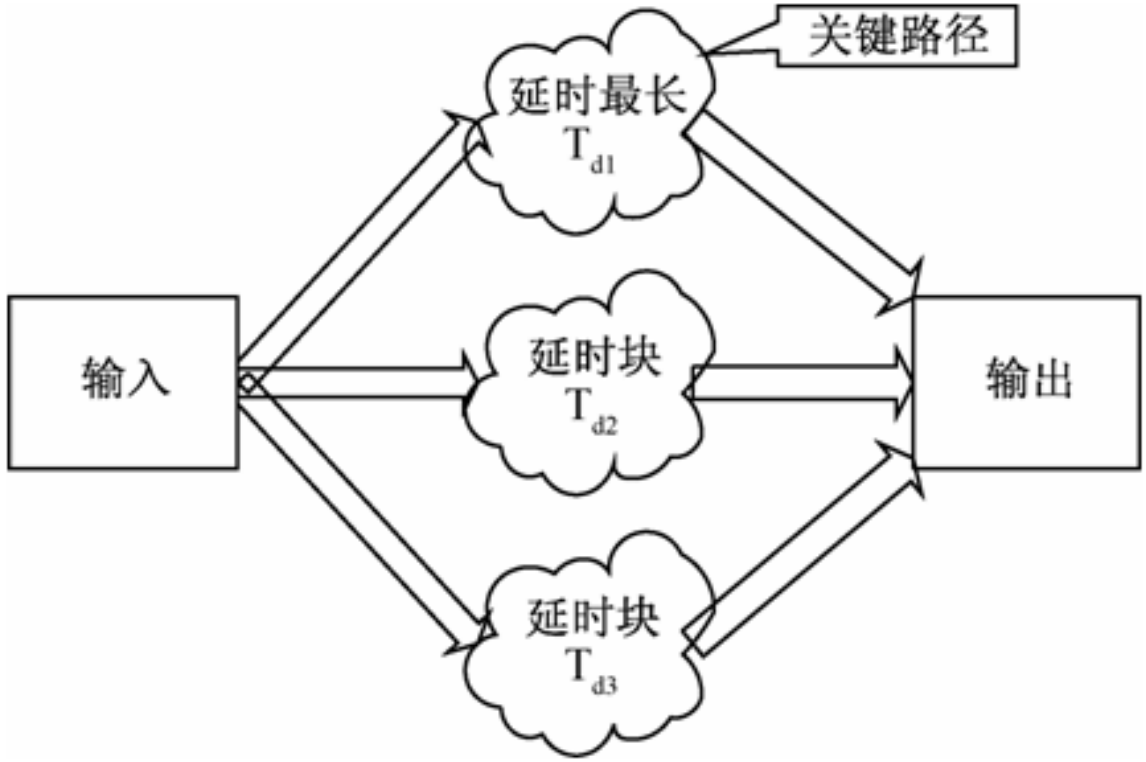

图 8-12 关键路径示意

8.2 速 度 优 化

8.2.4 乒乓操作法

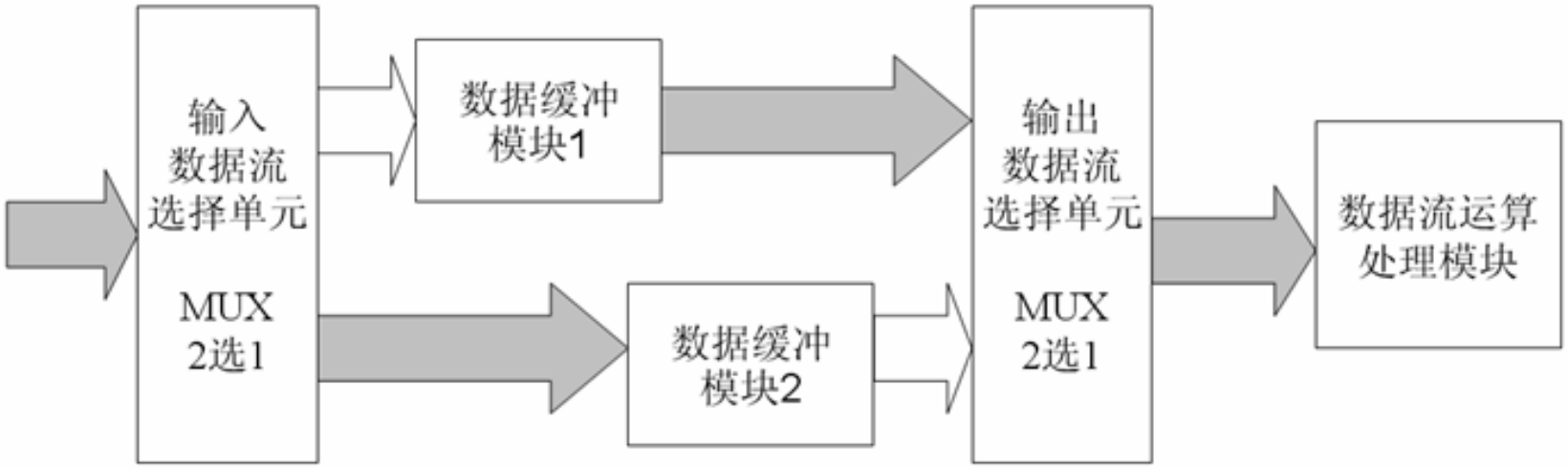

图 8-13　乒乓操作数据缓存结构示意图

8.2 速 度 优 化

8.2.5 加法树法

加法树速度优化技术部分类似于流水线法。

2输入加法树结构

若将加法树逐级拓展，可以实现更长的树结构。

8.3 优化设置与时序分析

8.3.1 使用Design Assistant检查设计可靠性

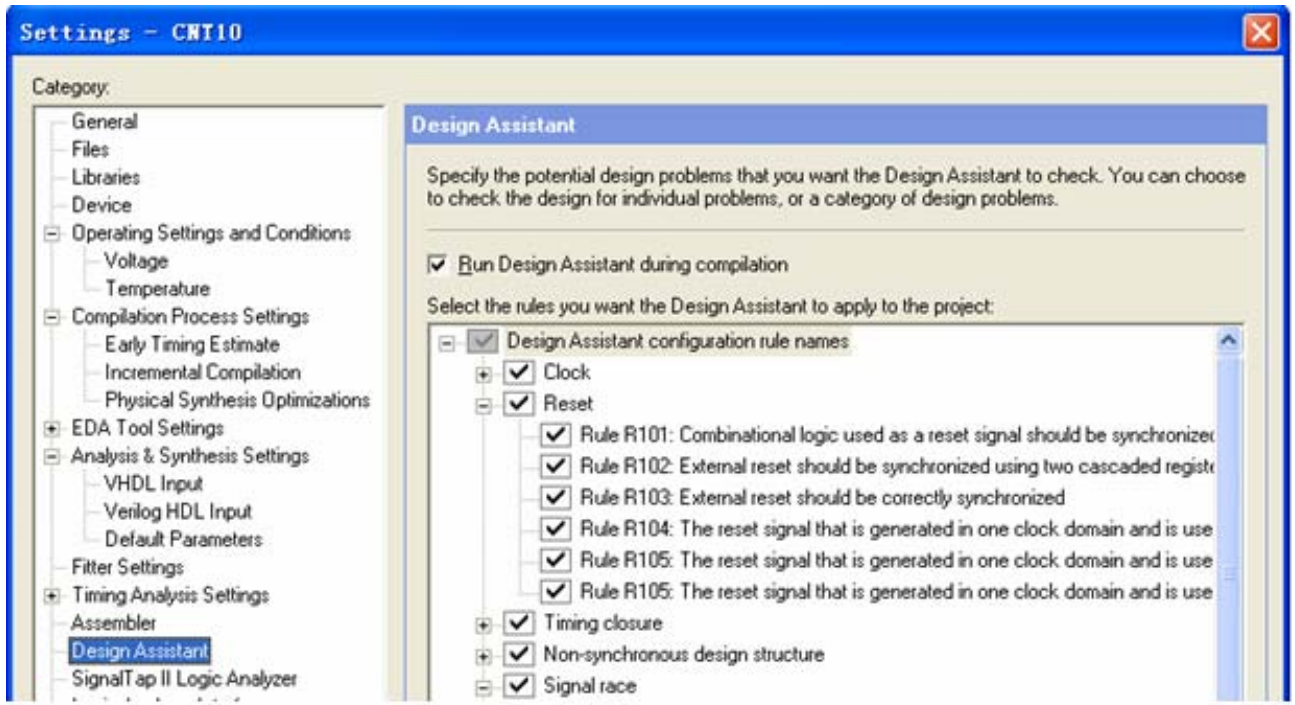

图 8-14　Design Assistant 设置

8.3 优化设置与时序分析

8.3.2 增量布局布线控制设置

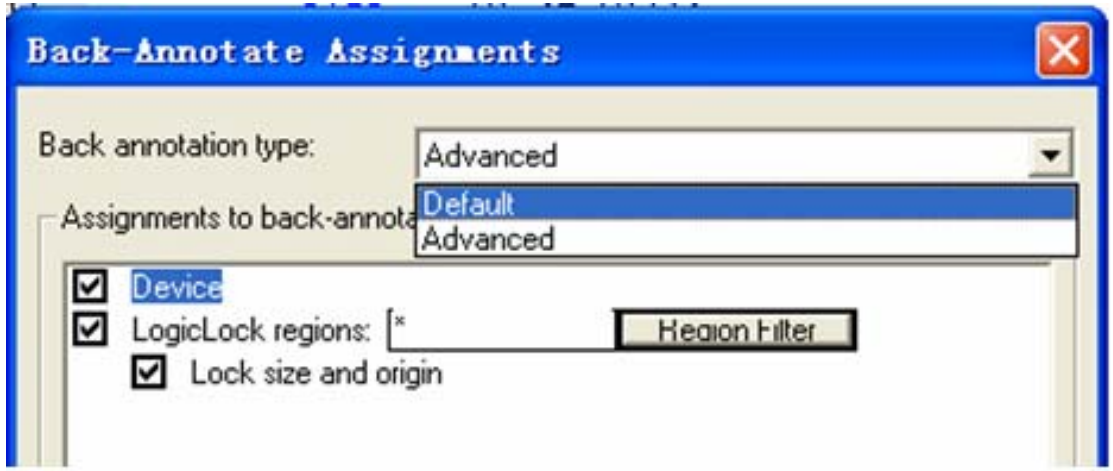

图 8-15 反标设置

8.3 优化设置与时序分析

8.3.3 时序设置与分析

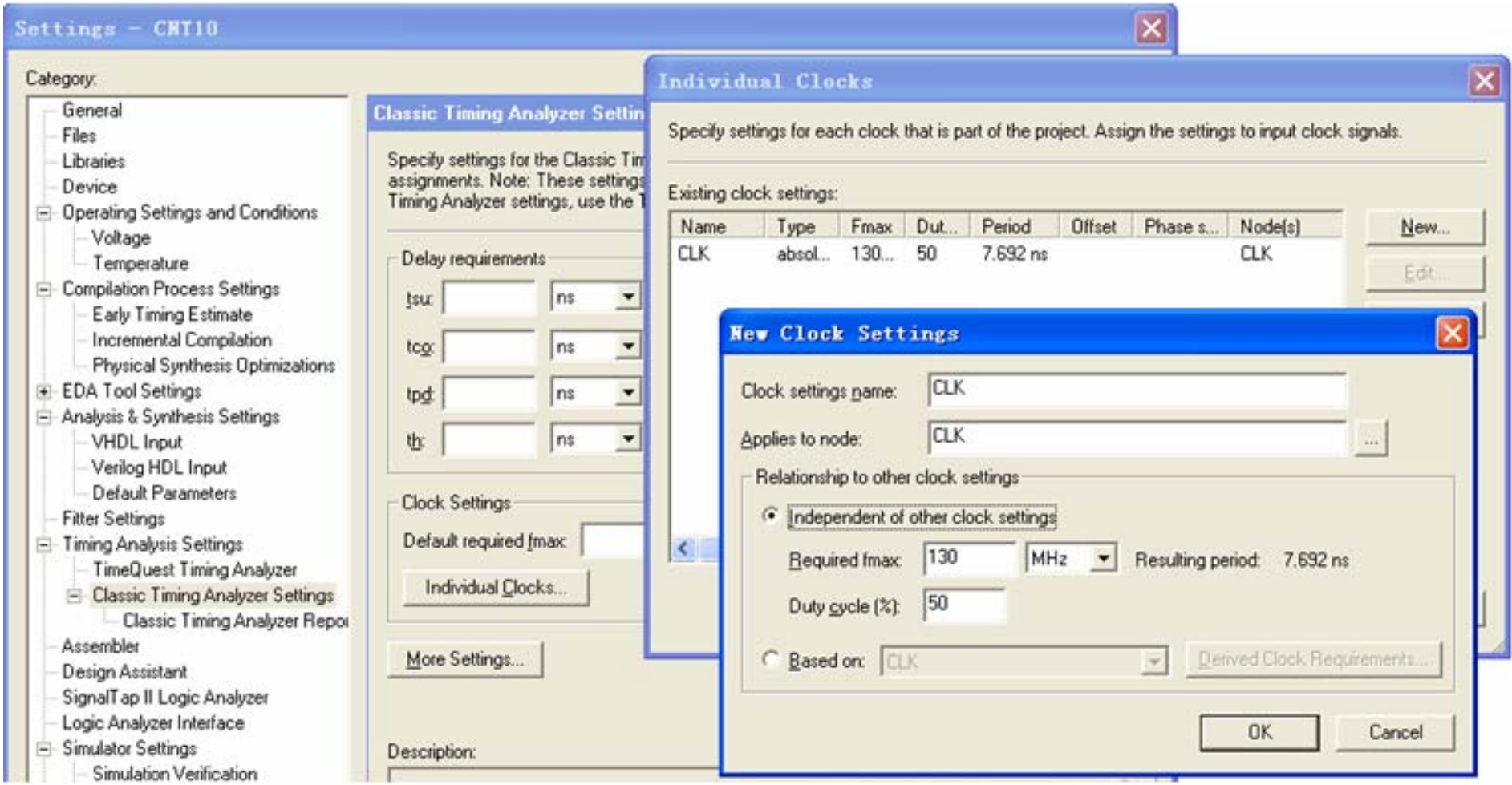

图 8-16 全编译前时序条件设置（设置时钟信号 CLK 不低于 130MHz）

8.3 优化设置与时序分析

8.3.4 查看时序分析结果

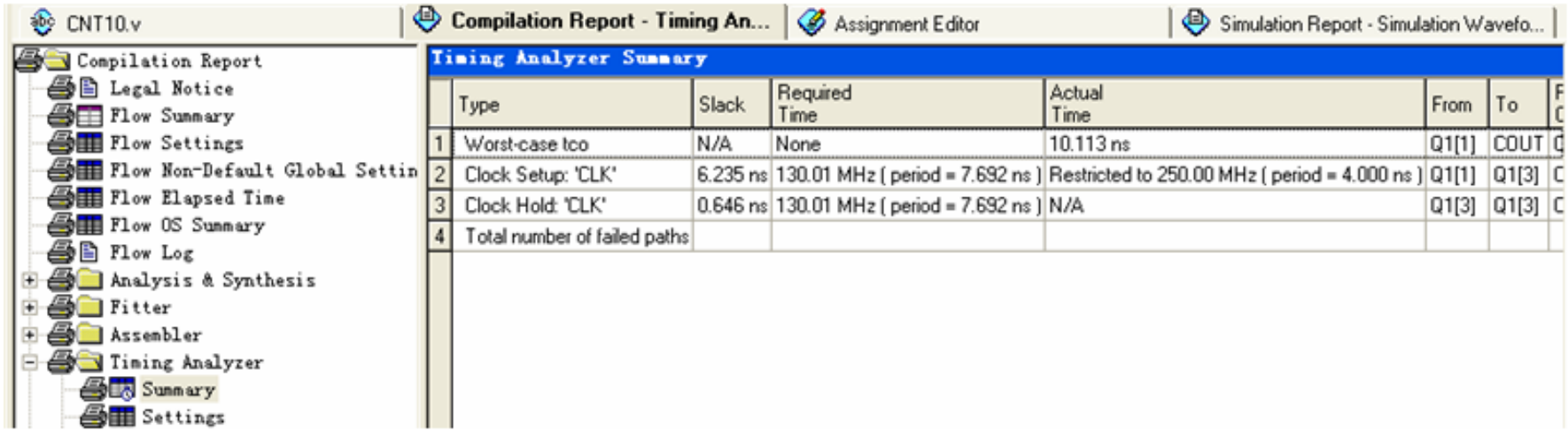

图 8-17 时序分析报告窗口

8.3 优化设置与时序分析

8.3.5 适配优化设置示例

（1）建立工程

（2）打开**Assignment Editor**对话框

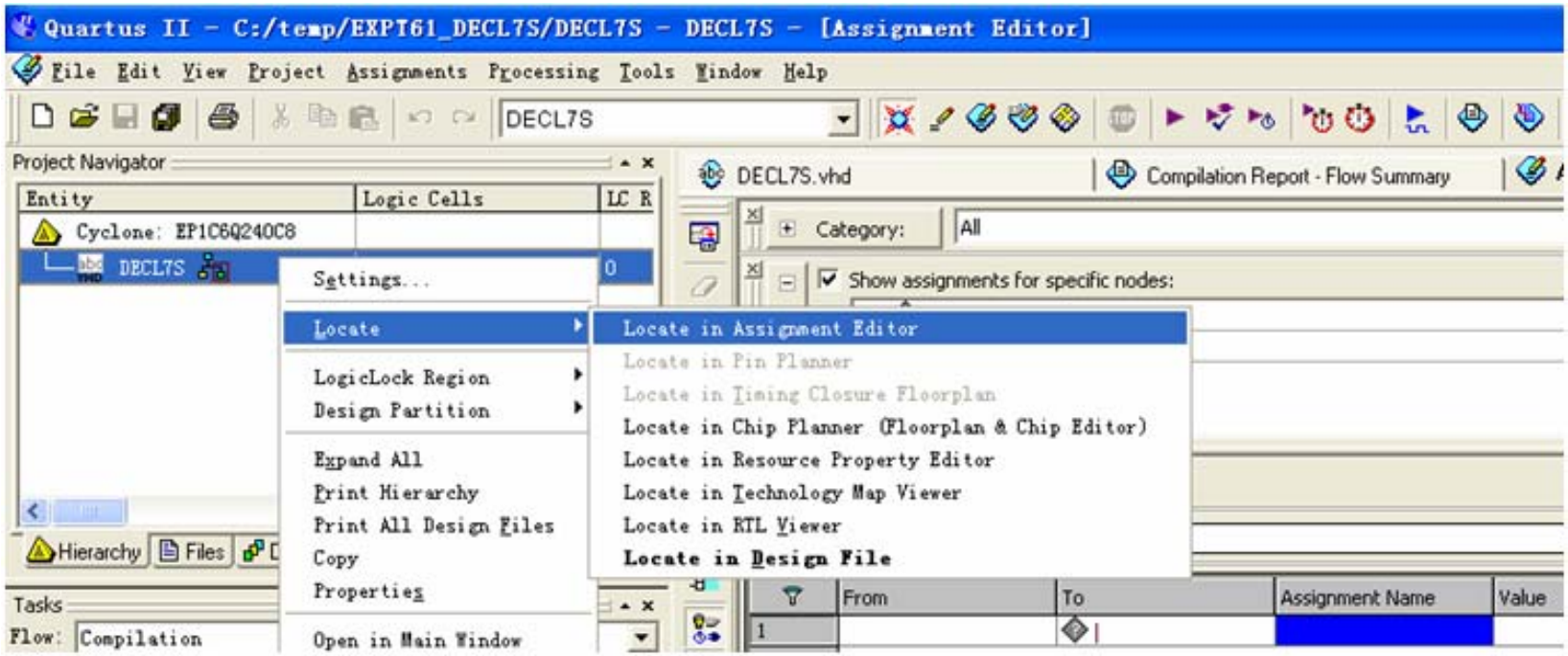

图 8-18 打开 Assignment Editor

8.3 优化设置与时序分析

8.3.5 适配优化设置示例

（3）选项设置

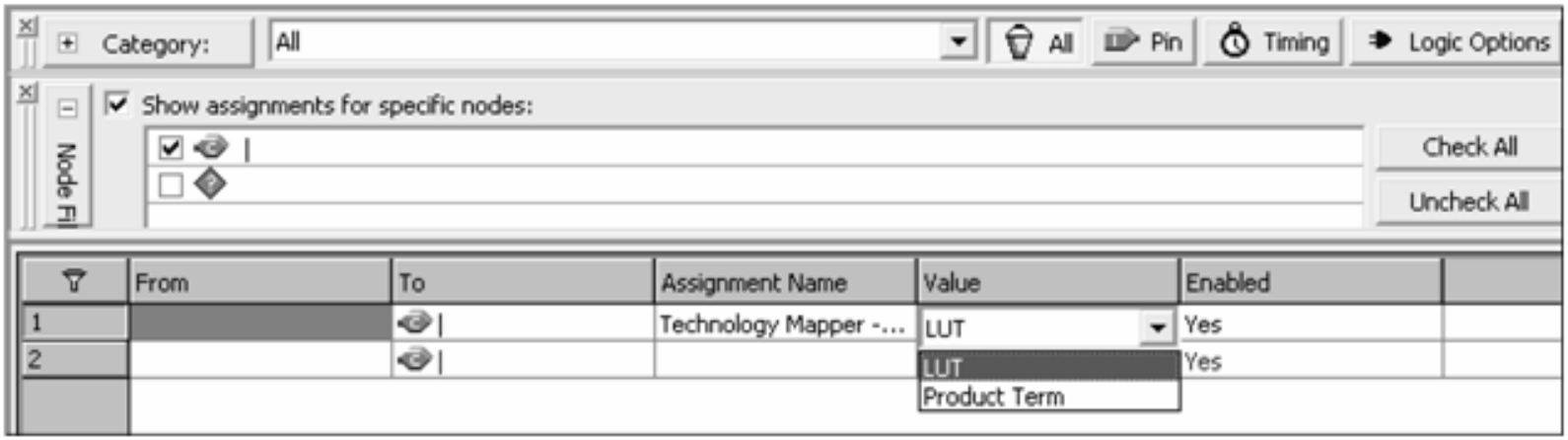

图 8-19 选用乘积项逻辑优化

8.3 优化设置与时序分析

8.3.6 LogicLock优化技术

Quartus II提供了一种非常优秀的优化技术，即逻辑锁定技术（**Logic Lock**）。

Quartus II支持逻辑锁定技术的**FPGA**器件系列有**APEX20K**、**APEXII**、**Excalibur**、**Cyclone/II/III**和**Stratix/II/III**等。

习　题

8-1　利用资源共享的面积优化方法对例**8-9**程序进行优化（仅要求在面积上优化）。

【例 8-9】

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_arith.all;
ENTITY addmux IS
   PORT (A,B,C,D   : IN  std_logic_vector(7 downto 0);
           sel     : IN  std_logic;
             R     : OUT std_logic_vector(7 downto 0));
END addmux;
ARCHITECTURE rtl OF addmux IS
BEGIN
   process(A,B,C,D,sel)    begin
      if(sel='0') then  R<=A+B;  else   R<=C+D;  end if;
   end process;
END rtl;
```

习　题

8-2　试通过优化逻辑的方式对图**8-20**所示的结构进行改进，给出**VHDL**代码和结构图。

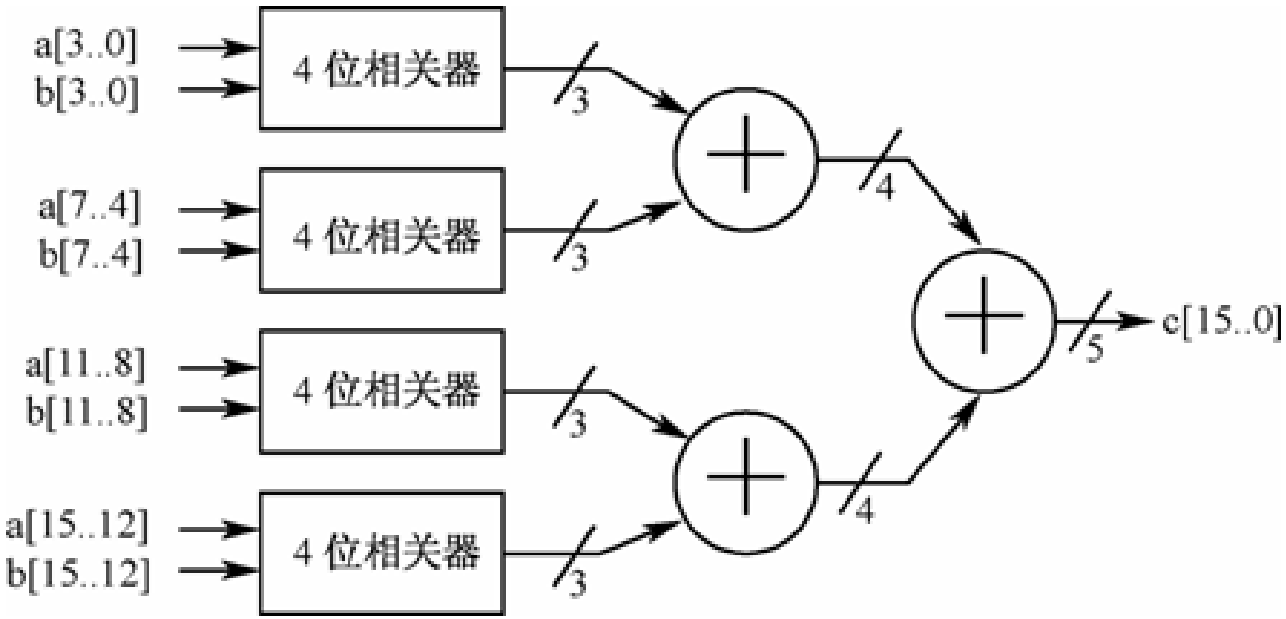

图 8-20　习题 8-2 图

习　题

8-3 已知4阶直接型FIR滤波器的数学表达式如下：
y(n)=x(n)h(0)+x(n-1)+x(n-2)h(2)+x(n-3)h(3)
x(n)与 **x(n-m)**，**m=0**，**1**，**2**，**3**是延迟关系，**m**表示延迟的**clk** 数。**x(n-m)**与**h(m)**的位宽均为**8**位，**y(n)**为**10**位，其中**h(m)**在模块例化后为常数。该模块的输入为**x(n)** 、**clk** ，输出为**y (n)**，试实现该逻辑。
8-4 对习题**8-3**中的**FIR**滤波器在速度上进行优化**(**在**h(m)**固定的情况下**)**，试采用流水线技术。
8-5 利用**FLEX**的**LUT**结构，构建资源占用较小的常数乘法器，改进习题**8-3**和习题**8-4**的设计，减少模块的资源使用。
8-6 若对速度要求不高，但目标芯片的容量较小，试把习题**8-3**中的**FIR**滤波器用串行化的方式实现。
8-7 设计一个连续乘法器，输入为**a0**、**a1**、**a2**、**a3**，位宽各为**8**位，输出**rout**为**32**位，完成**rout=a0*a1*a2*a3**。试实现之。
8-8 对习题**8-7**进行优化，判断以下实现方法中哪种方法更好？
（1）rout=((a0 * a1) * a2) * a3
（2）rout=(a0 * a1) * (a2 * a3)
8-9 为提高速度，对习题**8-8**中的前一种方法加上流水线技术进行实现。
8-10 试对以上的习题解答通过设置**Quartus II**相关选项的方式，提高速度，减小面积。

实验与设计

8-1 采用流水线技术设计高速数字相关器

（1）实验目的：

（2）实验原理：

（3）实验任务1：

【例 8-10】

```
stemp <= a XOR b;
PROCESS(stemp)  BEGIN
 CASE stemp IS
 WHEN "0000" => c <= "100";                                          --4
 WHEN "0001"|"0010"|"0100"|"1000" => c <= "011";                     --3
 WHEN "0011"|"0101"|"1001"|"0110"|"1010"|"1100" =>c<= "010"; --2
 WHEN "0111"|"1011"|"1101"|"1110" => c <= "001";    --1
 WHEN "1111" => c <= "000";  -- 0;
 WHEN OTHERS => c <= "000";
 END CASE;
END PROCESS;
```

实验与设计

8-1 采用流水线技术设计高速数字相关器

（4）实验任务2：

（5）实验任务3：

（6）实验任务4：

（7）思考题：

（8）实验报告：

实验与设计

8-2 线性反馈移位寄存器设计

（1）实验目的：

（2）实验原理：

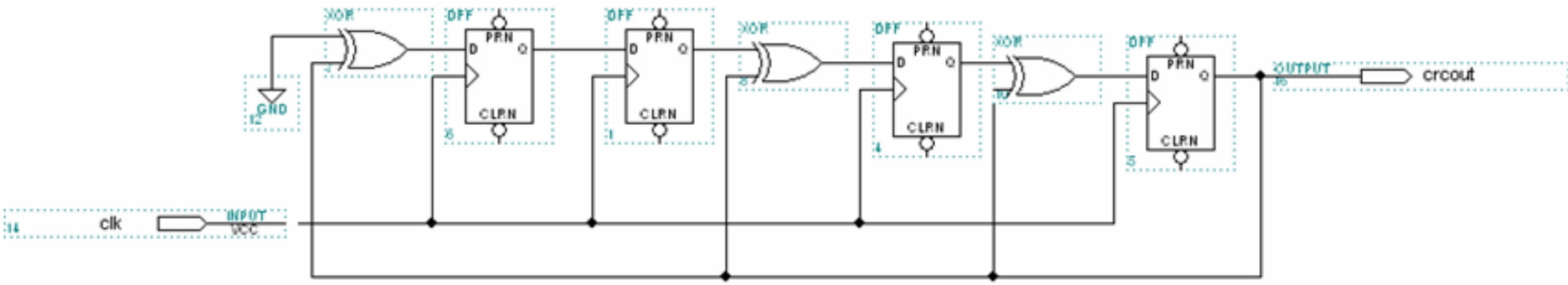

图 8-21　LFSR 举例

（3）实验任务：

实验与设计

8-2 线性反馈移位寄存器设计

（4）思考题1:

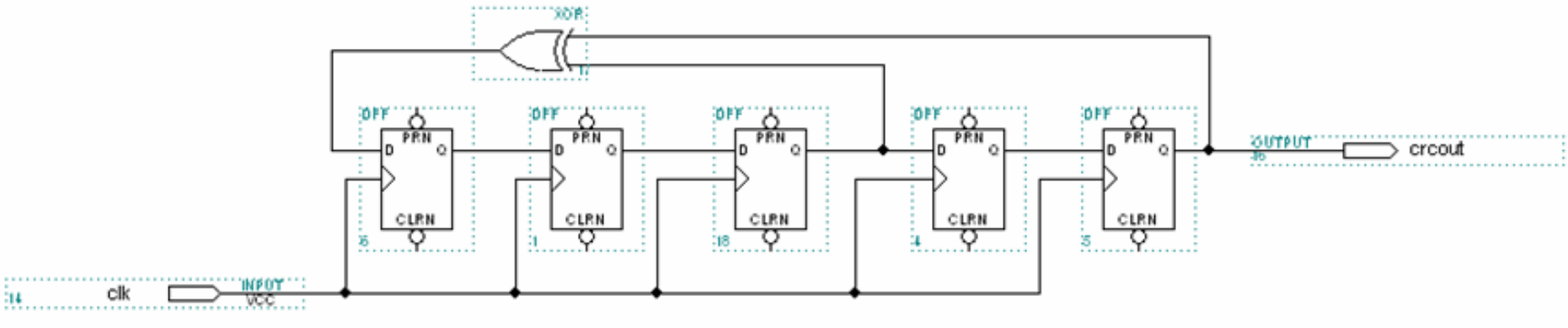

图 8-22 另一种 LFSR 结构

（5）思考题2：

（6）实验报告：

实验与设计

8-3 循环冗余校验（CRC）模块设计

（1）实验目的：

（2）实验原理：

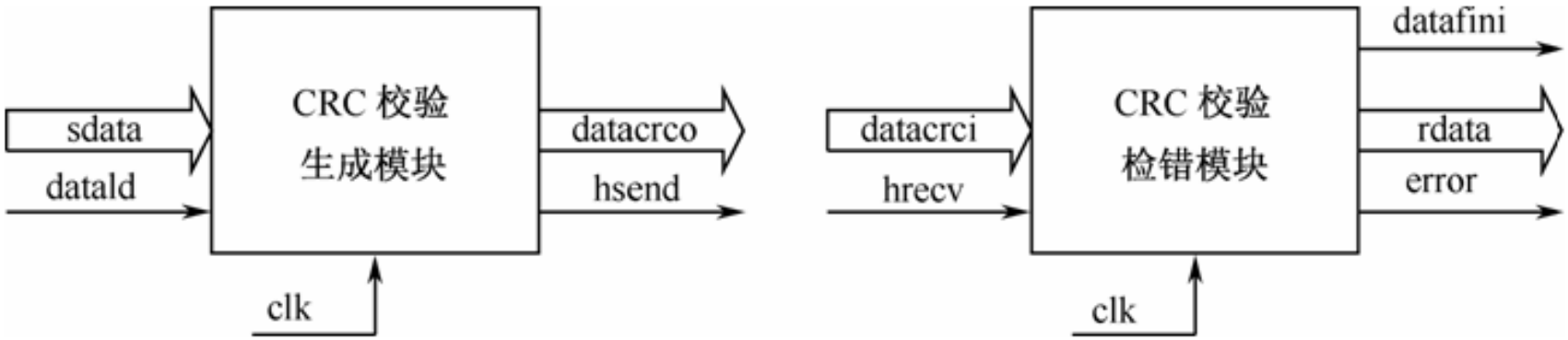

图 8-23　CRC 模块

【例 8-11】

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.ALL;
USE ieee.std_logic_arith.ALL;
ENTITY crcm IS
    PORT (clk, hrecv, datald : IN std_logic;
          sdata     : IN std_logic_vector(11 DOWNTO 0);
          datacrco  : OUT std_logic_vector(16 DOWNTO 0);
          datacrci  : IN std_logic_vector(16 DOWNTO 0);
            rdata   : OUT std_logic_vector(11 DOWNTO 0);
          datafini  : OUT std_logic;
          ERROR0, hsend  : OUT std_logic);
END crcm;
ARCHITECTURE comm OF crcm IS
    CONSTANT multi_coef : std_logic_vector(5 DOWNTO 0) := "110101";
           -- 多项式系数, MSB 一定为'1'
    SIGNAL  cnt,rcnt : std_logic_vector(4 DOWNTO 0);
    SIGNAL  dtemp,sdatam,rdtemp   : std_logic_vector(11 DOWNTO 0);
    SIGNAL  rdatacrc: std_logic_vector(16 DOWNTO 0);
    SIGNAL  st,rt  : std_logic;
BEGIN
```

接下页

接上页

```
PROCESS(clk)
    VARIABLE crcvar : std_logic_vector(5 DOWNTO 0);
BEGIN
    IF(clk'event AND clk = '1') THEN
        IF(st = '0' AND datald = '1') THEN  dtemp <= sdata;
  sdatam <= sdata; cnt <= (OTHERS => '0');  hsend <= '0';  st <= '1';
        ELSIF(st = '1' AND cnt < 7) THEN  cnt <= cnt + 1;
        IF(dtemp(11)='1')THEN crcvar:=dtemp(11 DOWNTO 6)XOR multi_coef;
           dtemp <= crcvar(4 DOWNTO 0) & dtemp(5 DOWNTO 0) & '0';
            ELSE  dtemp <= dtemp(10 DOWNTO 0) & '0';   END IF;
        ELSIF(st='1' AND cnt=7) THEN datacrco<=sdatam & dtemp(11 DOWNTO 7);
            hsend <= '1';  cnt <= cnt + 1;
        ELSIF(st='1' AND cnt=8) THEN   hsend<= '0';    st<='0';
        END IF;
    END IF;
END PROCESS;
PROCESS(hrecv,clk)
    VARIABLE rcrcvar : std_logic_vector(5 DOWNTO 0);
BEGIN
    IF(clk'event AND clk = '1') THEN
      IF(rt='0' AND hrecv = '1') THEN  rdtemp <= datacrci(16 DOWNTO 5);
       rdatacrc <= datacrci;  rcnt <= (OTHERS => '0');
```

接下页

实验与设计

接上页

```
    ERROR0 <= '0';    rt <= '1';
     ELSIF(rt='1' AND rcnt<7) THEN  datafini<='0';  rcnt <= rcnt + 1;
         rcrcvar := rdtemp(11 DOWNTO 6) XOR multi_coef;
         IF(rdtemp(11) = '1') THEN
          rdtemp <= rcrcvar(4 DOWNTO 0) & rdtemp(5 DOWNTO 0) & '0';
         ELSE  rdtemp <= rdtemp(10 DOWNTO 0) & '0';
         END IF;
      ELSIF(rt = '1' AND rcnt = 7) THEN  datafini <= '1';
         rdata <= rdatacrc(16 DOWNTO 5);  rt <= '0';
         IF(rdatacrc(4 DOWNTO 0) /= rdtemp(11 DOWNTO 7)) THEN
           ERROR0 <= '1'; END IF;
      END IF;
   END IF;
END PROCESS;
END comm;
```

实验与设计

8-3 循环冗余校验（CRC）模块设计

（3）实验任务1：

（4）实验任务2：

（5）思考题1：

（6）思考题2：

（7）思考题3：

（8）实验报告：

实验与设计

8-4 设计3级流水线16位加法器

实验任务：
根据8.2.1介绍的方法，设计具有3级流水线的16位加法器。在Quartus II上仿真验证，并通过Quartus II的相关编译报告比较无流水线（可以加一级锁存器以利比较）和有3级流水线的16位加法器的数据处理速度及资源占用情况。

实验与设计

8-5 基于DES数据加密标准的加解密系统设计

（1）实验原理：

（2）实验任务：

图 8-24　DES 加解密模块示意

实验与设计

8-6 SPWM脉宽调制控制系统设计

（1）实验原理：

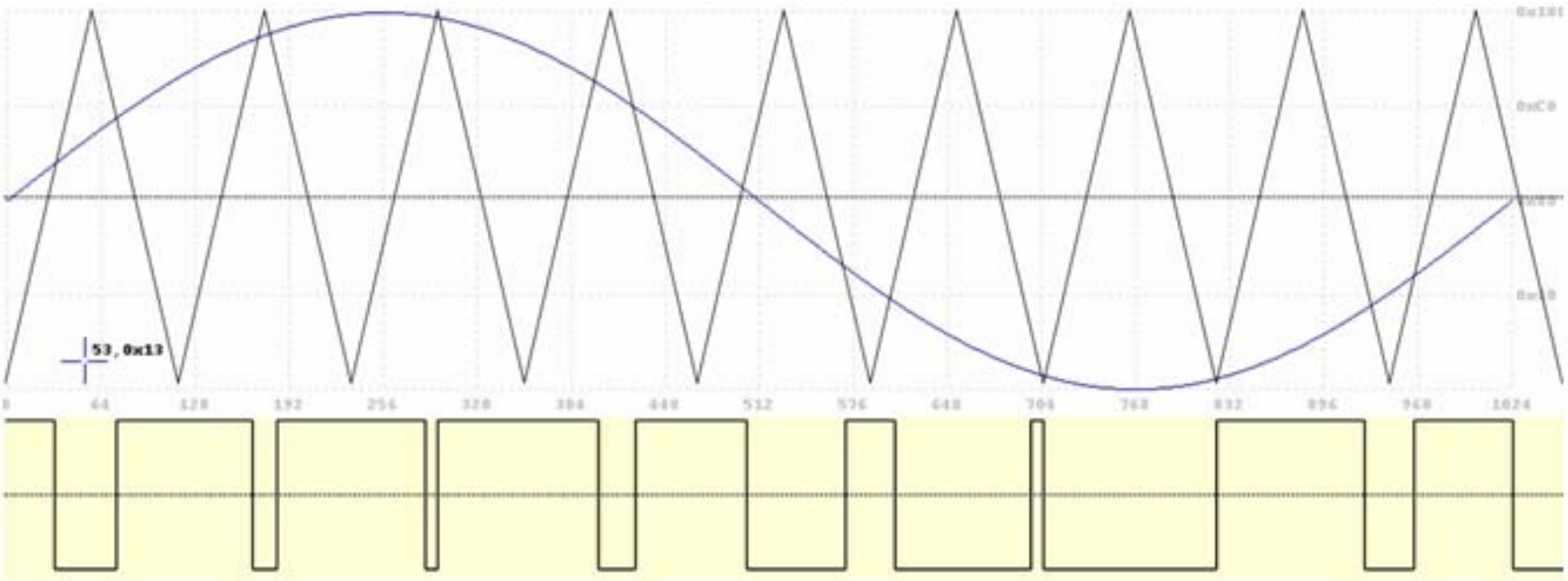

图 8-25 SPWM 波生成原理图

【例 8-12】三角波发生模块

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
USE ieee.std_logic_arith.all;
ENTITY TRANG3 IS
    PORT ( ADR : IN std_logic_vector(9 downto 0);
         OUTD : OUT std_logic_vector(9 downto 0));
END TRANG3 ;
ARCHITECTURE rtl OF TRANG3  IS
SIGNAL OT1 : std_logic_vector(9 downto 0);
SIGNAL CC  : std_logic_vector(10 downto 0);
BEGIN
 process(ADR,CC)   begin
     IF (ADR<"1000000000") THEN
       OT1(9 downto 1) <= ADR(8 downto 0); OT1(0)<='0';
       ELSE CC<="10000000000" + (NOT ADR) ;
           OT1(9 downto 1)<=CC(8 downto 0); OT1(0)<='0';  END IF;
   end process;
   OUTD<=OT1;
 END rtl;
```

实验与设计

8-6 SPWM脉宽调制控制系统设计

（2）实验内容1：演示示例：

/KX_7C5EE+/EXPERIMENTs/EXP33_PWM_GENERATOR/ 。

（3）实验内容2：

设计示例：**/KX_7C5EE+/EXPERIMENTs/EXP37_SPWM_Basic/**

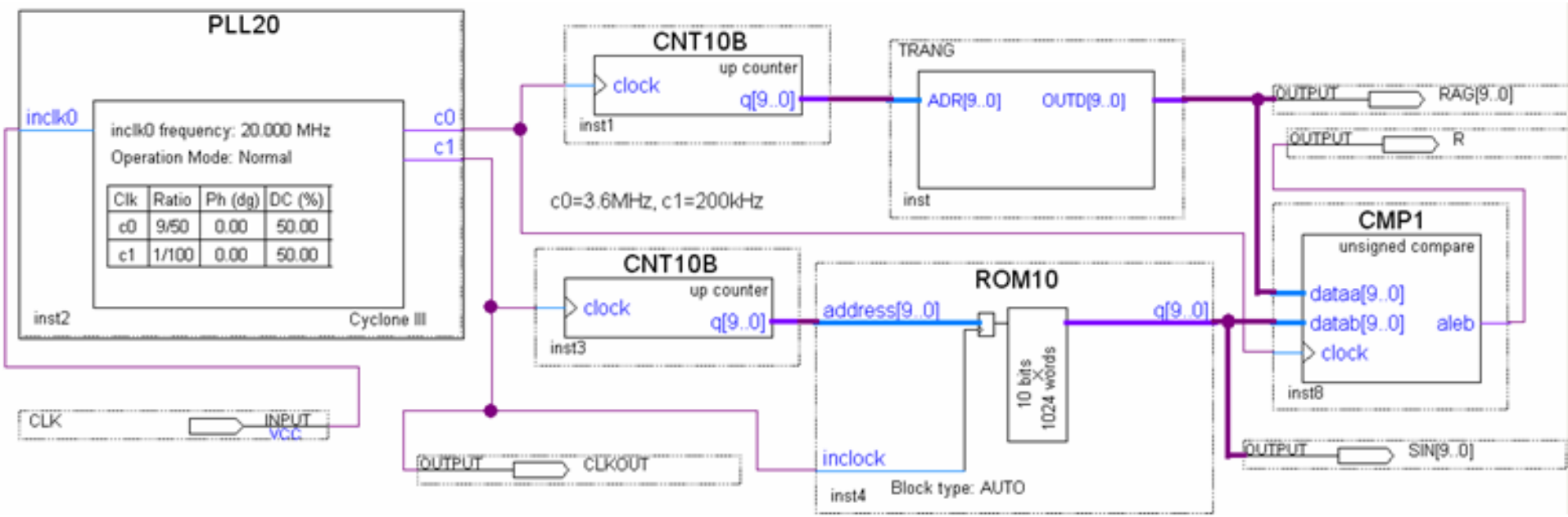

图 8-26　SPWM 波发生器基本电路图

实验与设计

8-6 SPWM脉宽调制控制系统设计

（2）实验内容1：演示示例：
/KX_7C5EE+/EXPERIMENTs/EXP33_PWM_GENERATOR/ 。

（3）实验内容2：
设计示例：/KX_7C5EE+/EXPERIMENTs/EXP37_SPWM_Basic/

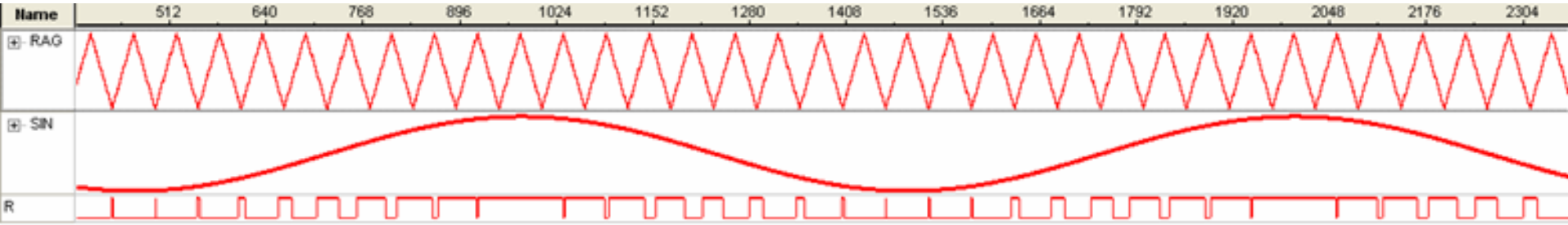

图 8-27 图 8-26 电路的 SignalTap II 实测波形

（4）实验内容3：
（5）实验内容4：

实验与设计

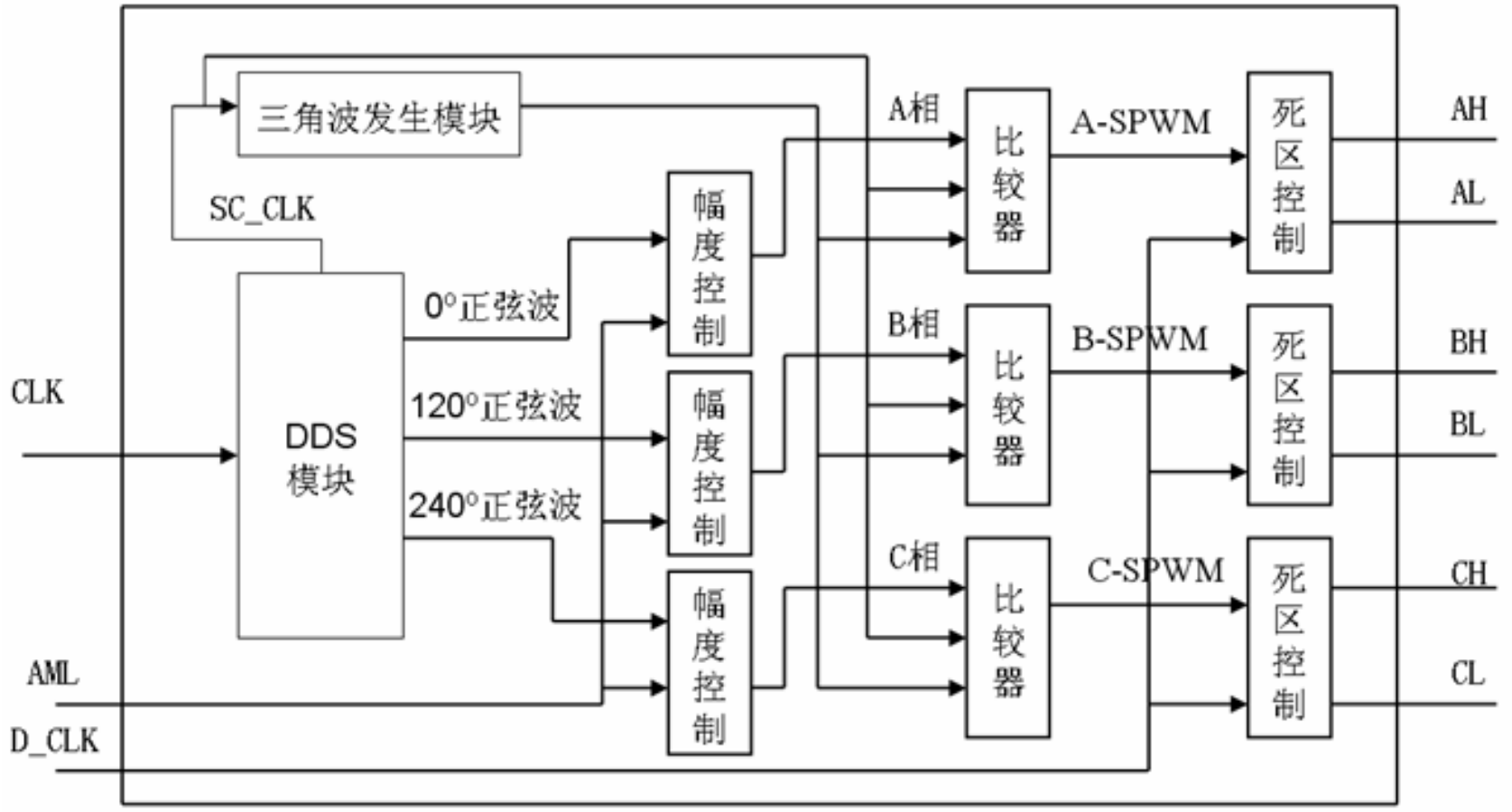

图 8-28 三相 SPWM 控制器电路模块图

实验与设计

8-7 步进电机细分控制电路设计

（1）实验目的：

（2）实验原理：

1. 步进电机细分驱动原理
2. 步距细分的系统构成
3. 细分驱动性能的改善

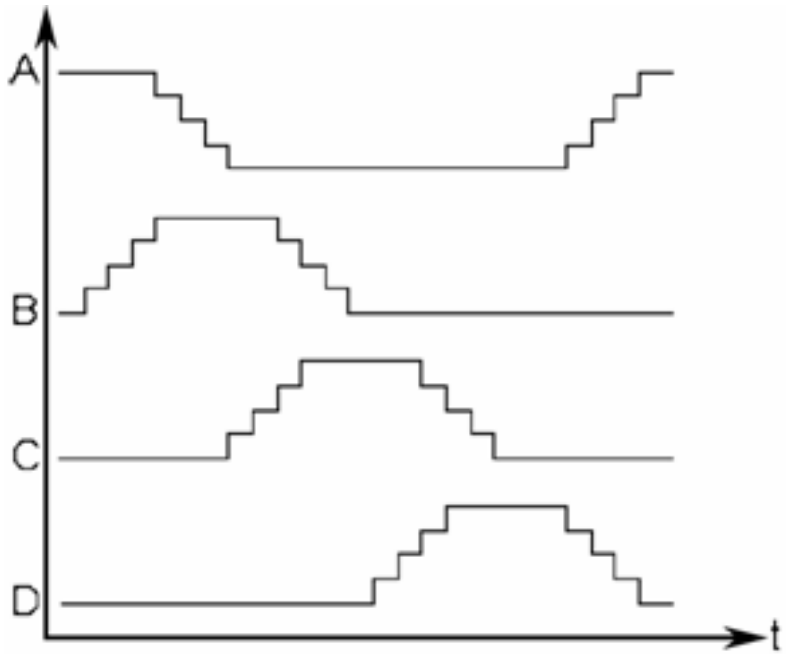

图 8-29 四相步进电机八细分电流波形

实验与设计

8-7 步进电机细分控制电路设计

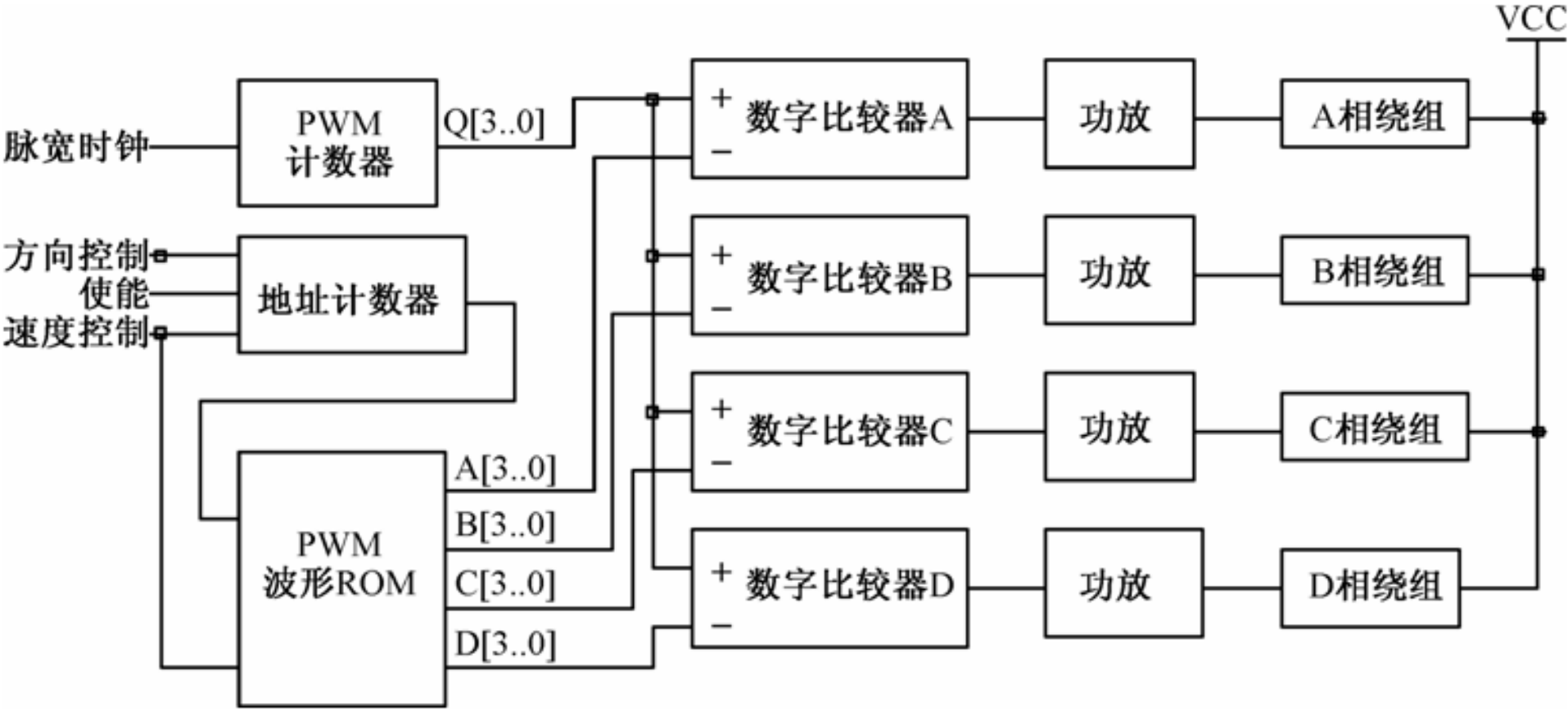

图 8-30 步进电机细分驱动电路结构图

实验与设计

8-7 步进电机细分控制电路设计

（1）实验任务1：

（2）实验任务2：

（3）实验任务3：

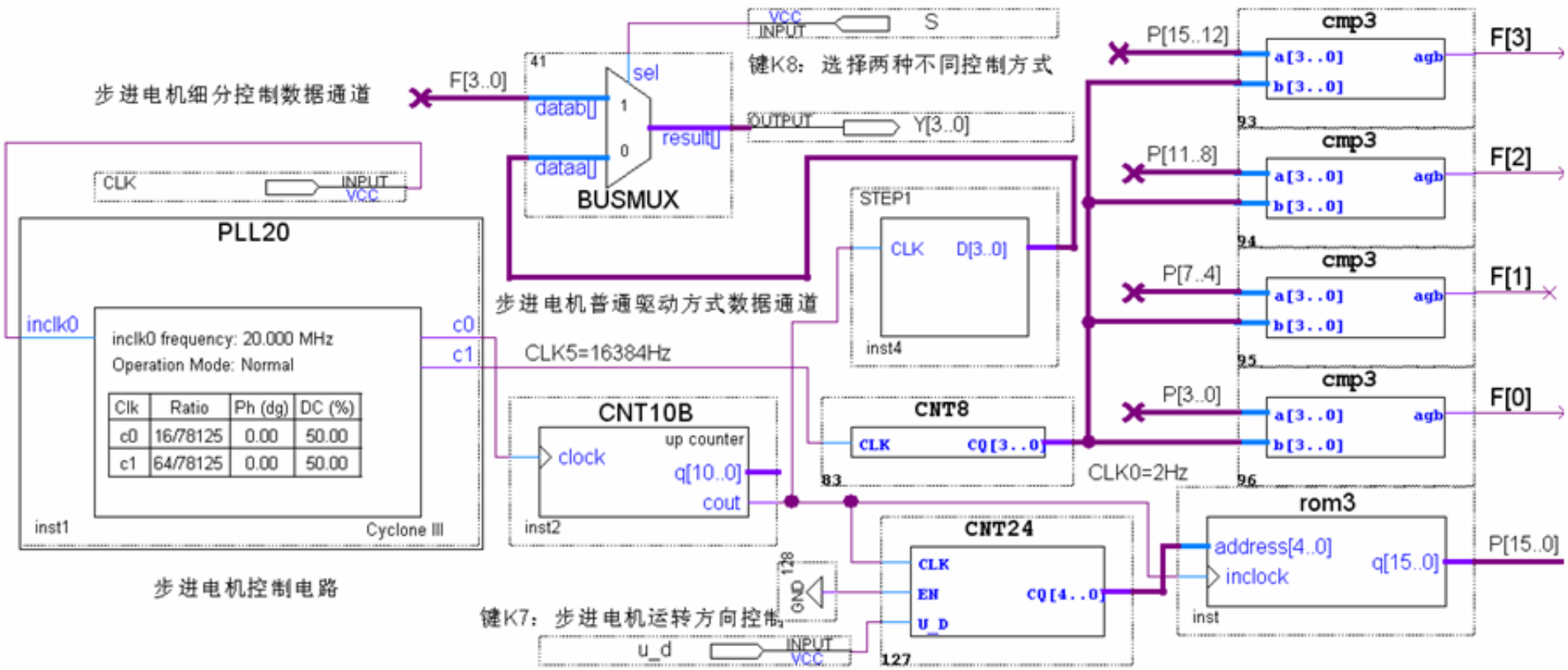

图 8-31 步进电机细分控制电路