5.5.3 VHDL语言的描述风格

三种描述风格:

行为描述: 使用功能描述

数据流（寄存器传输）: 使用布尔代数式描述

结构描述: 模块间的连接关系描述

5.5.4 基本逻辑电路的VHDL设计

一、组合电路

原则1：在process中用到的所有输入信号都出现在敏感信号列表中；

原则2：电路的真值表必须在代码中完整的反映出来。（否则会生成锁存器）

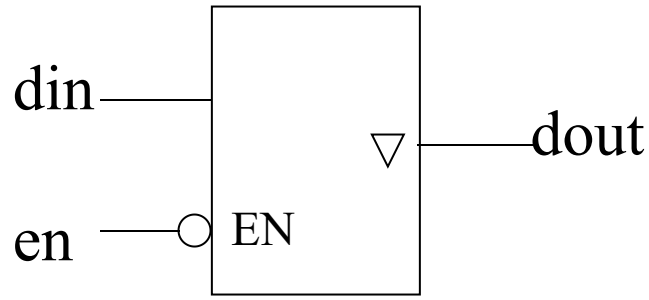
原则3：使用if‥ else‥，必须完全互斥。

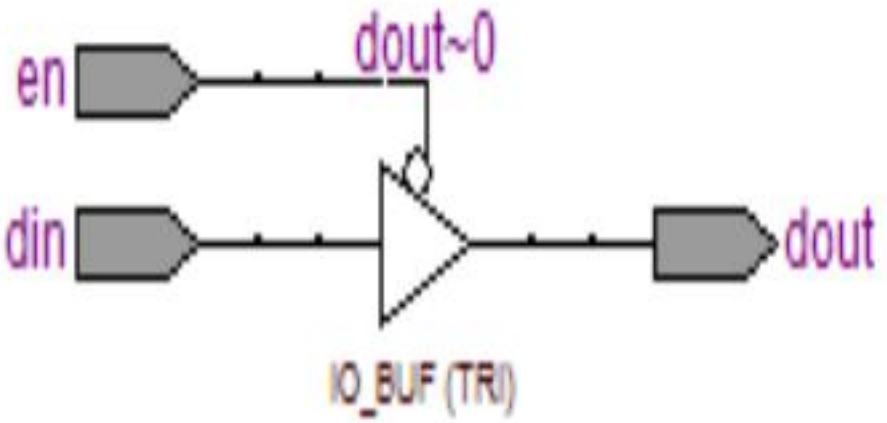
1. 三态门及总线缓冲器

```
——指定大写“Z”表示高阻态
        a <= ‘Z’ ;
        a_bus <= “ZZZZZZZZ” ;
```

```
LIBRARY  IEEE;
USE IEEE.STD_LOGIC_1164.all;

ENTITY tri_gate IS
      port ( din, en: IN std_logic;
                  dout: OUT std_logic );
END tri_gate;

ARCHITECTURE art OF tri_gate IS
BEGIN
    PROCESS (din,en)
    BEGIN
       IF (en=  '0' ) then
            dout <= din;
      ELSE
            dout<= 'Z' ;
      END  IF;
    END PROCESS
END  art;
```

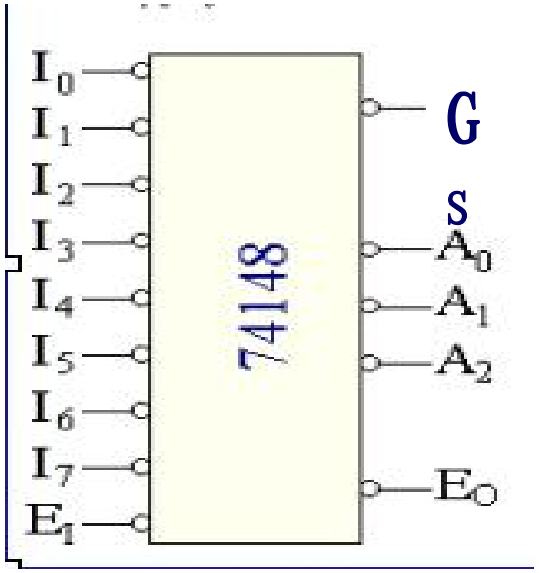
8位数据总线?





2. 8:3优先编码器

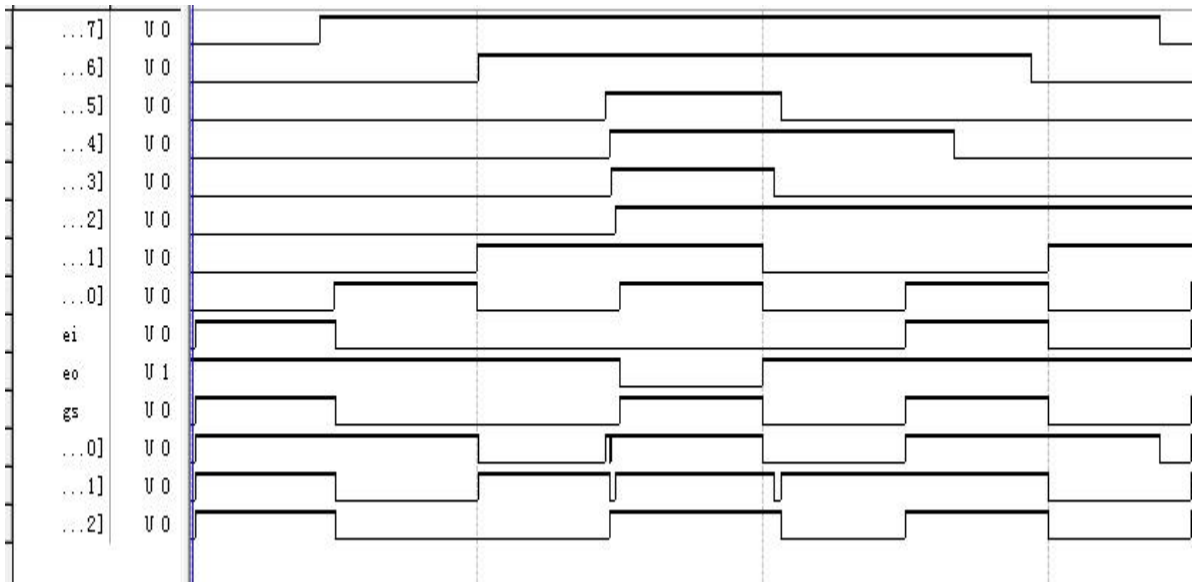
```
library ieee;
use ieee.std_logic_1164.all;

ENTITY  priencoder  is
PORT (input:IN STD_LOGIC_VECTOR(7 downto 0);
        ei:IN STD_LOGIC;
       yout:OUT STD_LOGIC_VECTOR(2 downto 0);
        eo,gs:OUT STD_LOGIC);
END  priencoder;

ARCHITECTURE  cod74148  OF  priencoder  IS
BEGIN
  PROCESS(ei,input)
  BEGIN
    IF(ei='1')THEN
      yout <= "111";
      eo <= '1';
      gs <= '1';
    ELSE
```



输入									输出				
$\overline{E_I}$	$\bar{I}_7$	$\bar{I}_6$	$\bar{I}_5$	$\bar{I}_4$	$\bar{I}_3$	$\bar{I}_2$	$\bar{I}_1$	$\bar{I}_0$	$\bar{Y}_2$	$\bar{Y}_1$	$\bar{Y}_0$	$\overline{G_s}$	$\overline{E_O}$
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	×	×	×	×	×	×	×	0	0	0	0	1
0	1	0	×	×	×	×	×	×	0	0	1	0	1
0	1	1	0	×	×	×	×	×	0	1	0	0	1
0	1	1	1	0	×	×	×	×	0	1	1	0	1
0	1	1	1	1	0	×	×	×	1	0	0	0	1
0	1	1	1	1	1	0	×	×	1	0	1	0	1
0	1	1	1	1	1	1	0	×	1	1	0	0	1
0	1	1	1	1	1	1	1	0	1	1	1	0	1

输入									输出				
$\overline{E_I}$	$\bar{I}_7$	$\bar{I}_6$	$\bar{I}_5$	$\bar{I}_4$	$\bar{I}_3$	$\bar{I}_2$	$\bar{I}_1$	$\bar{I}_0$	$\bar{Y}_2$	$\bar{Y}_1$	$\bar{Y}_0$	$\overline{G_s}$	$\overline{E_O}$
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	×	×	×	×	×	×	×	0	0	0	0	1
0	1	0	×	×	×	×	×	×	0	0	1	0	1
0	1	1	0	×	×	×	×	×	0	1	0	0	1
0	1	1	1	0	×	×	×	×	0	1	1	0	1
0	1	1	1	1	0	×	×	×	1	0	0	0	1
0	1	1	1	1	1	0	×	×	1	0	1	0	1
0	1	1	1	1	1	1	0	×	1	1	0	0	1
0	1	1	1	1	1	1	1	0	1	1	1	0	1

```
    IF (input(7)='0')THEN
            yout <= "000";
            eo <= '1' ;
            gs <= '0' ;

    ELSIF (input(6)='0')THEN
            yout <= "001";
            eo <= '1' ;
            gs <= '0' ;
    ELSIF (input(5)='0')THEN
            yout <= "010";
            eo <= '1' ;
…………….……
    ELSIF  (input(0)='0')THEN
            yout <= "111";
            eo <= '1' ;
            gs <= '0' ;
    ELSE   yout <= "111";
            eo <= '0' ;
            gs <= '1' ;
    END IF;
  END IF;
 END PROCESS;
END cod74148;
```



输入									输出				
$\overline{E_I}$	$\overline{I}_7$	$\overline{I}_6$	$\overline{I}_5$	$\overline{I}_4$	$\overline{I}_3$	$\overline{I}_2$	$\overline{I}_1$	$\overline{I}_0$	$\overline{Y}_2$	$\overline{Y}_1$	$\overline{Y}_0$	$\overline{G_s}$	$\overline{E_O}$
1	×	×	×	×	×	×	×	×	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	×	×	×	×	×	×	×	0	0	0	0	1
0	1	0	×	×	×	×	×	×	0	0	1	0	1
0	1	1	0	×	×	×	×	×	0	1	0	0	1
0	1	1	1	0	×	×	×	×	0	1	1	0	1
0	1	1	1	1	0	×	×	×	1	0	0	0	1
0	1	1	1	1	1	0	×	×	1	0	1	0	1
0	1	1	1	1	1	1	0	×	1	1	0	0	1
0	1	1	1	1	1	1	1	0	1	1	1	0	1

二、 时序逻辑电路设计

触发器、寄存器、计数器、分频器、节拍发生器、状态机等。

进程的敏感信号是时钟信号，在进程内部用if语句描述时钟的边沿条件。

时钟上升沿：

（clock’event and clock = ‘1’）

时钟下降沿：

（clock’event and clock = ‘0’）

敏感信号表的特点：

1）敏感信号表中只有时钟信号——同步复位等。

如：

```
process (clk)
begin
    if (clk’ event and clk =  ‘1’ )
then
        if  reset =  ‘1’   then
            data <=  “00” ;
        else
            data <= in_data;
        end  if;
    end  if;
end  process;
```

最先判断时钟

同步复位

同步复位2D触发器

2）敏感信号表中除时钟外，还有其它信号——异步操作

例：

```
process (clk, reset)
begin
    if   reset = '1'   then
         data <= "00";
    elsif (clk'event and clk = '1') then
         data <= in_data;
    end  if;
end  process;
```

最先判断异步复位

对于判断时钟电路边沿，后没有“else”语句。

同步、异步区别：敏感信号表、语句顺序

2）计数器

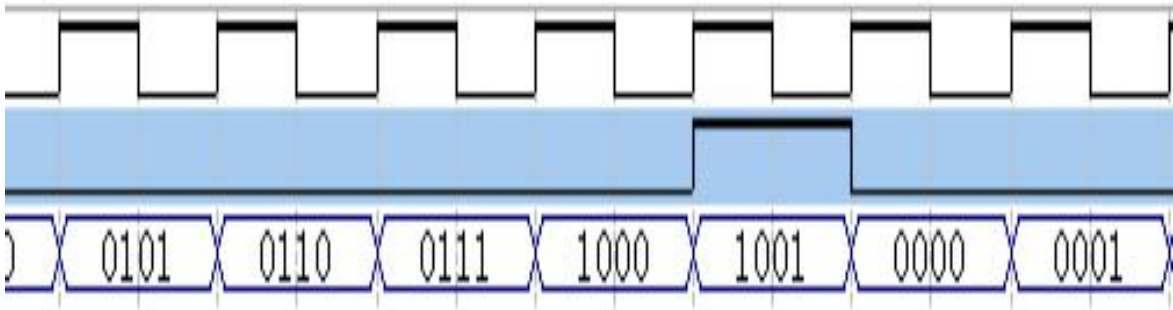
描述计数器（std_logic_vector），要用到中间信号和标准的程序包：IEEE.STD_LOGIC_UNSIGNED

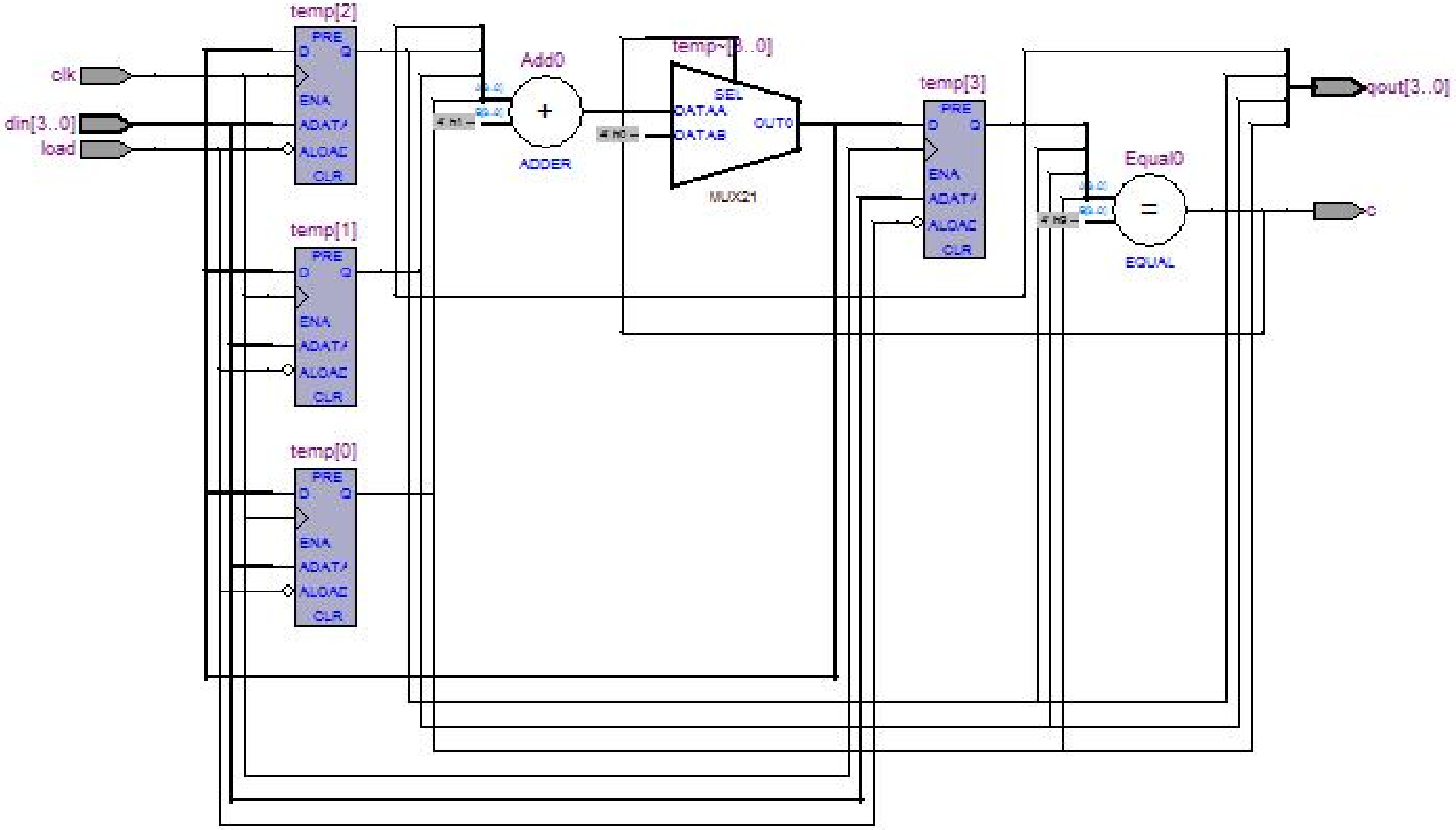
例：异步置数(低有效）的10进制计数器。

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity counter10 is
    Port ( clk : in std_logic;
           load : in std_logic;
           din : in std_logic_vector(3 downto 0);
           qout : out std_logic_vector(3 downto 0);
            c: out std_logic);
end counter10;
```

```
architecture art of counter10 is
    signal temp : std_logic_vector(3 downto 0);
begin
    process (clk, load, din)
    begin
        if (load= '0' ) then
            temp <= din;
        elsif (clk'event and clk= '1')  then
            if (temp = "1001")  then          --if (temp=9) then
                temp <= "0000";
            else
                temp <= temp+1;
            end if;
        end if;
    end process;
    qout <= temp;
    c <= '1' when temp="1001" else
         '0';                                 --when temp=9 else
end art;
```

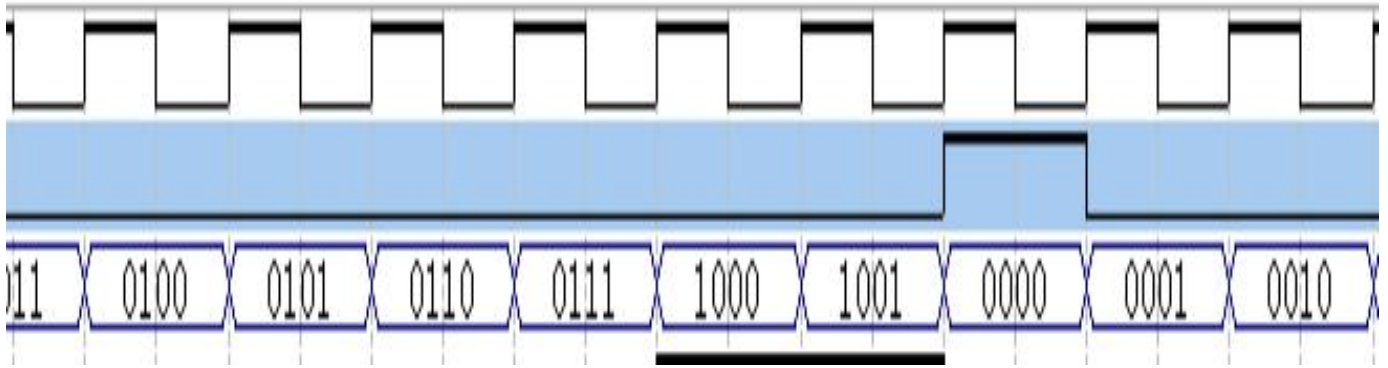
为描述计数过程，
定义中间信号。

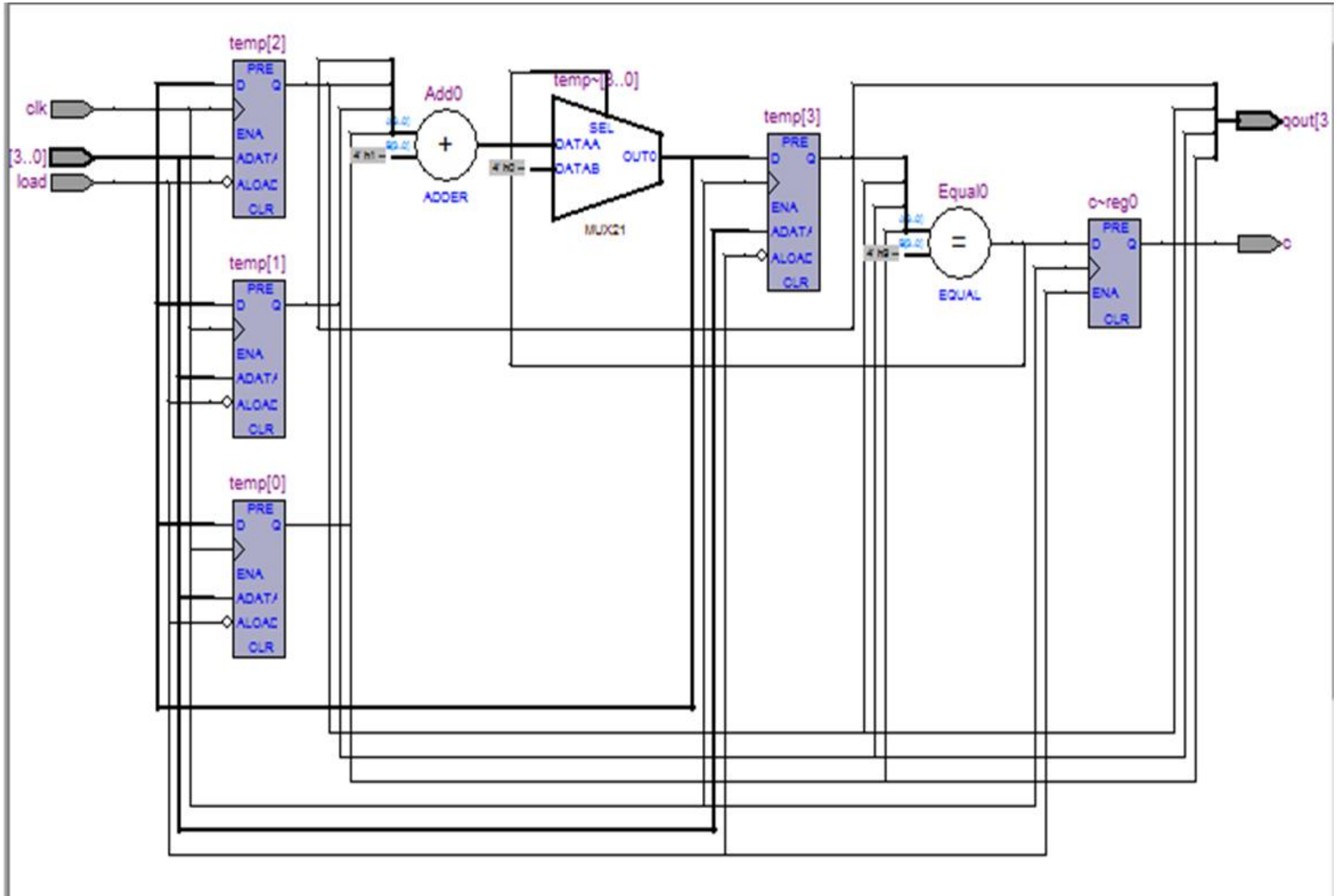



clk
din[3..0]
load
temp[2]
temp[1]
temp[0]
PRE
D
Q
ENA
ALOAD
CLR
Add0
ADDER
temp~[3..0]
SEL
DATAA
DATAB
OUTO
MUX21
temp[3]
Equal0
EQUAL
qout[3..0]
=

```
. . . . . .
process (clk, reset, din)
   begin
      if (load= '0' ) then
          temp <= din;
      elsif (clk'event and clk= '1')  then
          if (temp = 9)  then
              temp <= "0000";
              c <= '1';
          else
              temp <= temp+1;
              c <= '0';
          end if;
      end if;
end process;
  qout <= temp;
End art
```

```
if (clk'event and clk= '1')  then
      if (temp = m-1)  then
          temp <= "00···00";
      else
          temp <= temp+1;
       end if;
 end if ;
end process ;

q <= temp;
 c <= '1' when  temp=m-1 else
      '0';
```




temp[2]
clk
[3..0]
load
PRE
D
Q
ENA
ALOAD
CLR
Add0
ADDER
temp~[3..0]
SEL
DATAA
DATAB
OUT0
MUX21
temp[3]
Equal0
EQUAL
c~reg0
qout[3
c
temp[1]
temp[0]

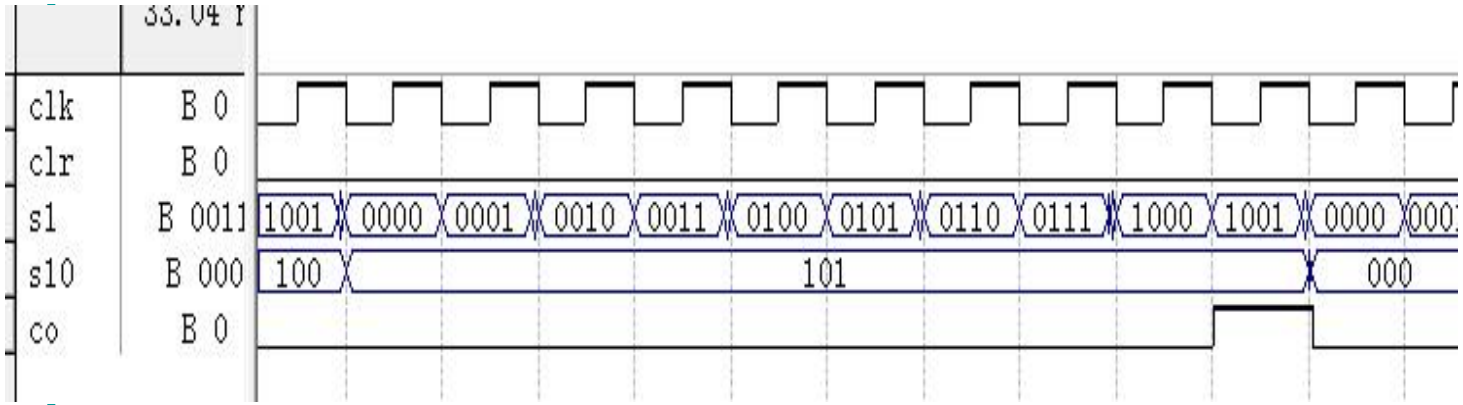
例：六十进制（分、秒）计数器

与一般二进制计数器不同，分
个位十位显示，为分、秒计数

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity clock60 is
    Port ( clk : in std_logic;
           clr : in std_logic;
           s1 : out std_logic_vector(3 downto 0);
           s10 : out std_logic_vector(2 downto 0);
           co : out std_logic );
end clock60;

architecture art of clock60 is
    signal s1_temp : std_logic_vector(3 downto 0);
    signal s10_temp : std_logic_vector(2 downto 0);
```

```
begin
  process (clk, clr)
   begin
        if (clr= ‘1‘) then
            s1_temp <= “0000”;
            s10_temp <= “000”;
       elsif (clk’event and clk= ‘1’)  then
          if (s1_temp= 9) then
                s1_temp<= "0000";
                if (s10_temp = 5) then
                    s10_temp<= “000”;
                else
                    s10_temp <= s10_temp+1;
                end if;
           else
                s1_temp <= s1_temp+1;
           end if;
       end if;
  end process;
```

```
    s1 <= s1_temp;
    s10 <= s10_temp;
   co <= ‘1’ when ( s10_temp=5 and s1_temp=9)   else
          ‘0’;
  End art;
```

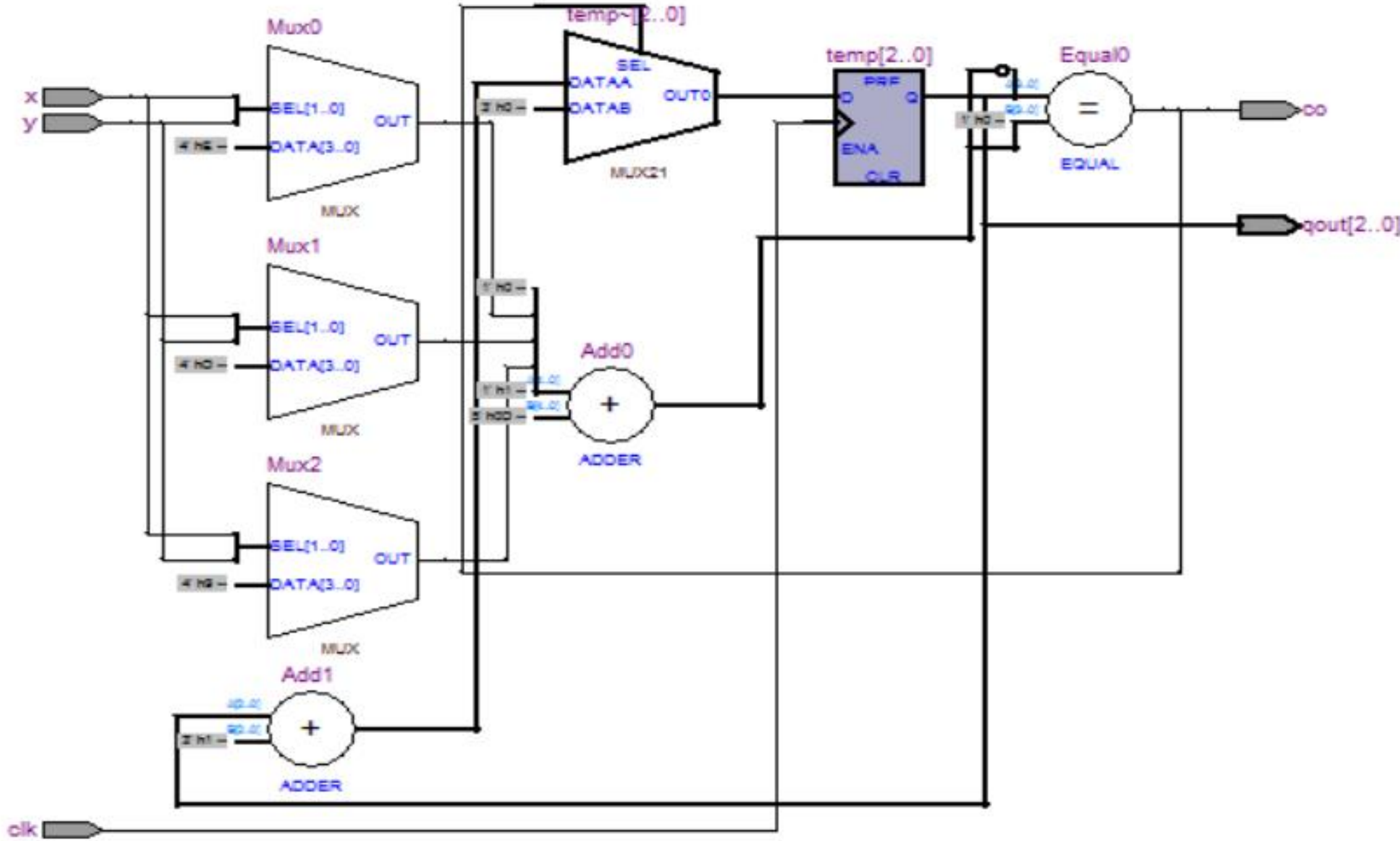


模可控计数器（由输入x, y控制）

M=3,4,6,7

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

ENTITY add IS
  PORT(clk, x,y:IN STD_LOGIC;
         qout: OUT STD_LOGIC_VECTOR (2 DOWNTO 0);
             co:OUT STD_LOGIC);
END add;

ARCHITECTURE behv OF add Is
     SIGNAL con : std_logic_vector (1 downto 0);
     SIGNAL temp : std_logic_vector (2 downto 0);
    SIGNAL m : integer range 7 downto 0;
BEGIN
     PROCESS(x,y)       -- 组合逻辑
     begin
      con<=x&y;
      case con is
          when "00“  =>  m<= 3 ;
          when "01"  =>  m<= 4;
          when "10"  =>  m<= 6;
          when others  =>  m<= 7;
       END case;
    END process;
```

```
PROCESS (clk, m) – 时序逻辑
 begin
      if (clk'event and clk='1') then
           if  (temp = m-1)  then
               temp <= "000";
          else
               temp <=temp+1;
           END IF;
     END IF;
  END PROCESS;

 qout <= temp ;
  co <= '1' when temp=m-1 else
              '0';

END behv;
```

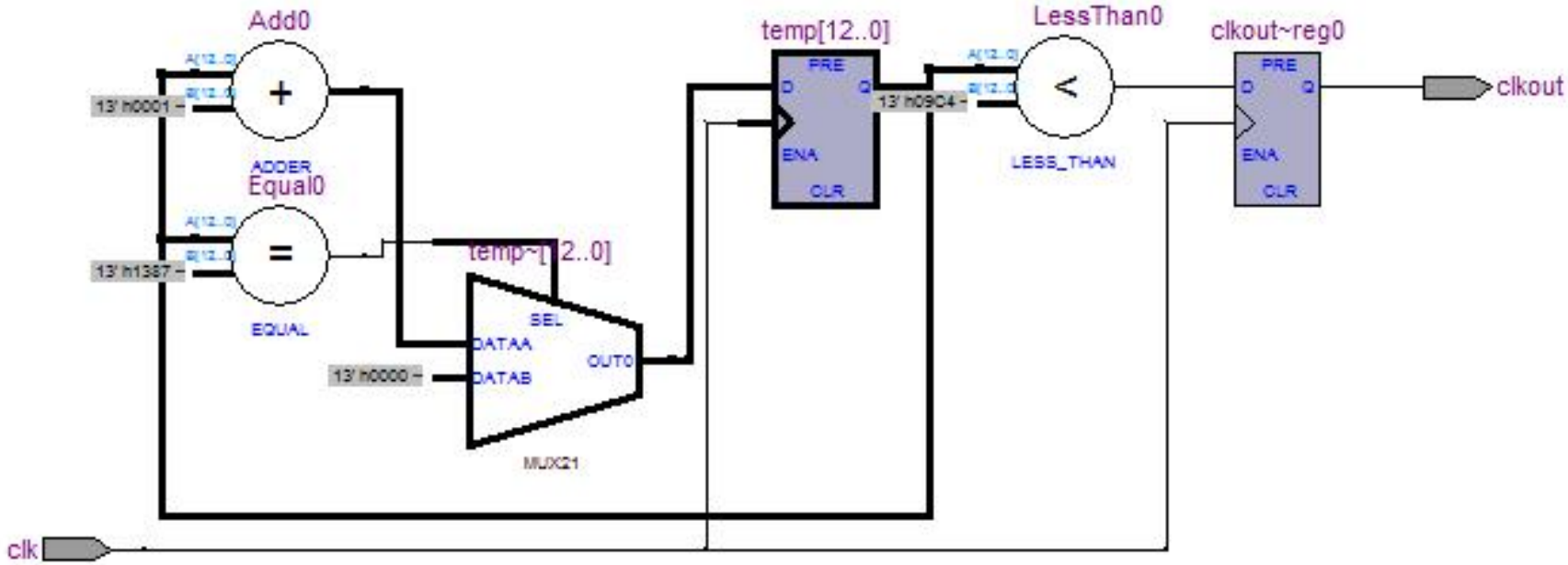

Mux0
temp~[2..0]
temp[2..0]
Equal0
x
y
SEL[1..0]
DATA[3..0]
OUT
MUX
SEL
DATAA
DATAB
OUT0
MUX21
PRE
D
Q
ENA
CLR
=
EQUAL
CO
qout[2..0]
Mux1
Add0
+
ADDER
Mux2
Add1
clk

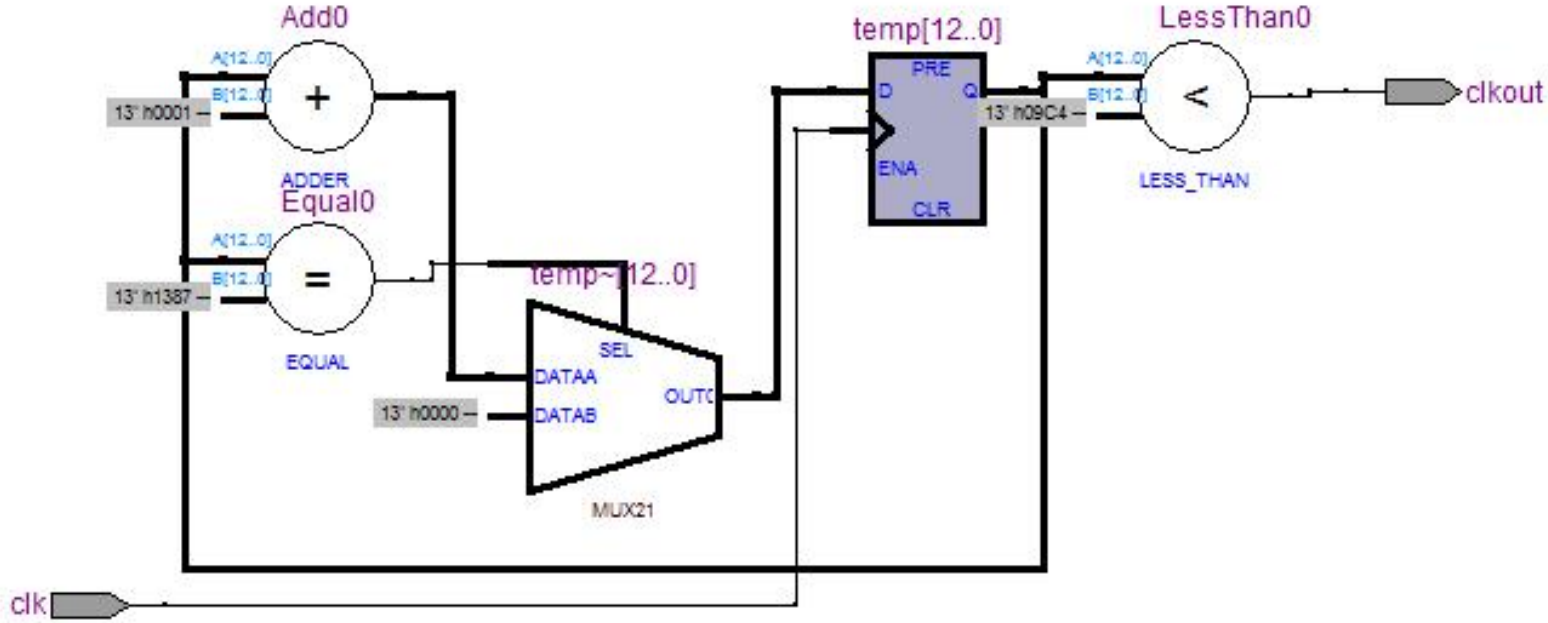
例：设计5000分频器

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fp is
    Port ( clk : in std_logic;
           clkout : out std_logic);
end fp;


architecture art of fp is
    signal temp: integer range 0 to  4999;
Begin
```

```
process (clk)
begin
       if (clk’ event and clk= ‘1’ ) then
             if ( temp=4999) then
                  temp <= 0 ;
             else
                  temp <= temp+1;
             end if;
       end if;
end process;
process ( clk , temp)
begin
     if (clk’ event and clk= ‘1’ ) then
        if( temp<2500)  then  clkout <= '1' ;
        else                  clkout <= '0';
        end if;
    end if;
End process;
end art;
```

（方波）


Add0
A[12..0]
B[12..0]
13' h0001
+
ADDER
Equal0
A[12..0]
B[12..0]
13' h1387
=
EQUAL
temp~[12..0]
SEL
DATAA
DATAB
OUT0
13' h0000
MUX21
temp[12..0]
PRE
D
Q
ENA
CLR
13' h09C4
LessThan0
A[12..0]
B[12..0]
<
LESS_THAN
clkout~reg0
PRE
D
Q
ENA
CLR
clkout
clk


Add0
A[12..0]
B[12..0]
13' h0001
+
ADDER
Equal0
A[12..0]
B[12..0]
13' h1387
=
EQUAL
temp~[12..0]
SEL
DATAA
DATAB
OUT0
13' h0000
MUX21
temp[12..0]
PRE
D
Q
ENA
CLR
13' h09C4
LessThan0
A[12..0]
B[12..0]
<
LESS_THAN
clkout
clk

例：8位环形移位寄存器（左、右移、异步置数）

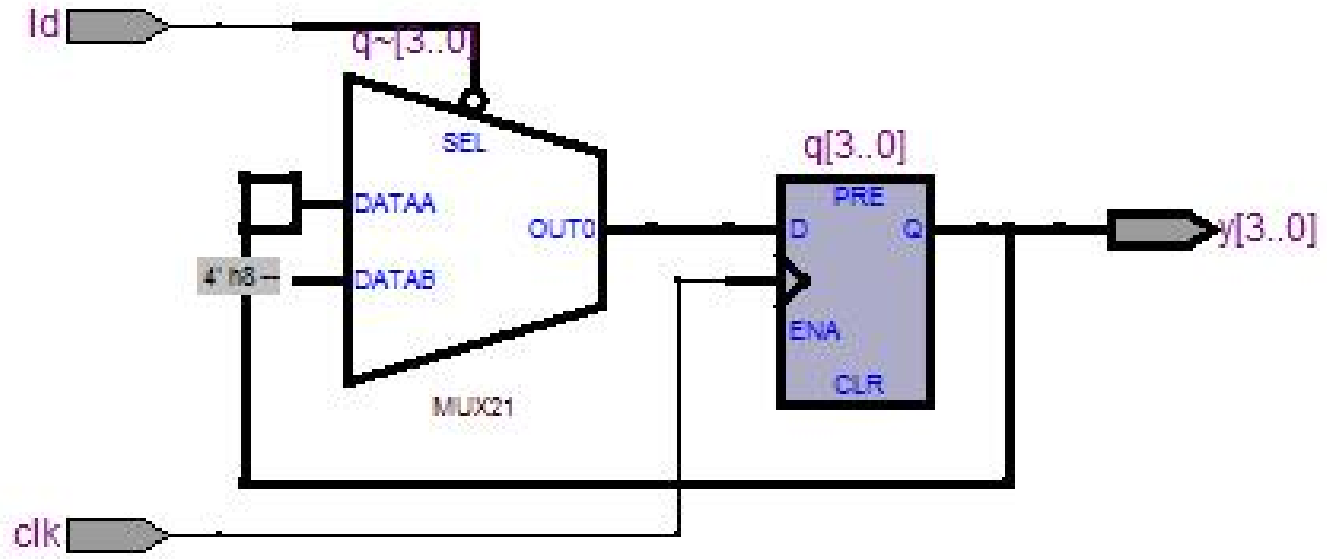
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity shfrt is
    Port ( clk ,load,con: in std_logic;
                  din:   in std_logic_vector (7 downto 0);
                 qout : out std_logic_vector (7 downto 0)
               );
end shfrt;

architecture art of shfrt is
    signal q_temp: std_logic_vector (7 downto 0);
begin
```

```
process (clk, load, din, con)
begin
       if load= ‘1’  then
           q_temp <= din;
       elsif (clk’ event and clk=  ‘1’ )  then
            if con= ‘1’  then
                q_temp <= q_temp (6 downto 0) & q_temp(7) ;
            else
                 q_temp <= q_temp(0) & q_temp (7 downto1);
            end if;
      end if;

 End process;
  qout <= q_temp;
End art;
```

例：环型顺序脉冲发生器(4位）

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity index is
    Port ( clk ,ld: in std_logic;
           y : out std_logic_vector (3 downto 0)
          );
end index;

architecture art of index is
  signal q: std_logic_vector (3 downto 0);
begin
```

```
process (clk)
begin
      if (clk’ event and clk=  ‘1’ )   then
            if ld= ‘0’  then
                q <= “1000” ;
            else
                 q <= q (0) & q (3 downto 1);
            end if;
      end if;
End process;

            y <= q  ;

End art;
```



3) 状态机

- 用户自定义数据类型

利用用户自定义数据类型－－枚举类型实现。

语法格式：

```
type 数据类型名 is 数据类型定义（枚举）;
```

综合器自动实现枚举类型元素的编码

```
type week is (sun, mon, tue, wed, thu,   fri, sat);
Signal day : week ;
                ……
              day <= sun;
```

```
type statetype is (s0, s1, s2, s3);
signal present_state, next_state : statetype;
              ……
           present_state <= S0 ;
```

- 状态转移的描述

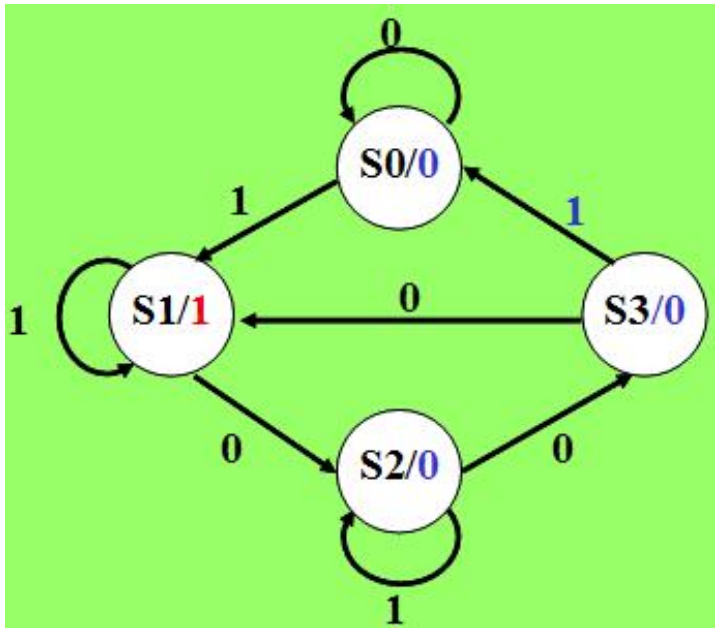
```
CASE   现态    IS
  WHEN   表达式值1 => 顺序处理语句11; 语句12;
                .....
  WHEN 表达式值3 => 顺序处理语句31; 语句32;
 END CASE;
```

- 状态机整体描述结构和输出的描述

组合进程：描述状态逻辑、输出逻辑；

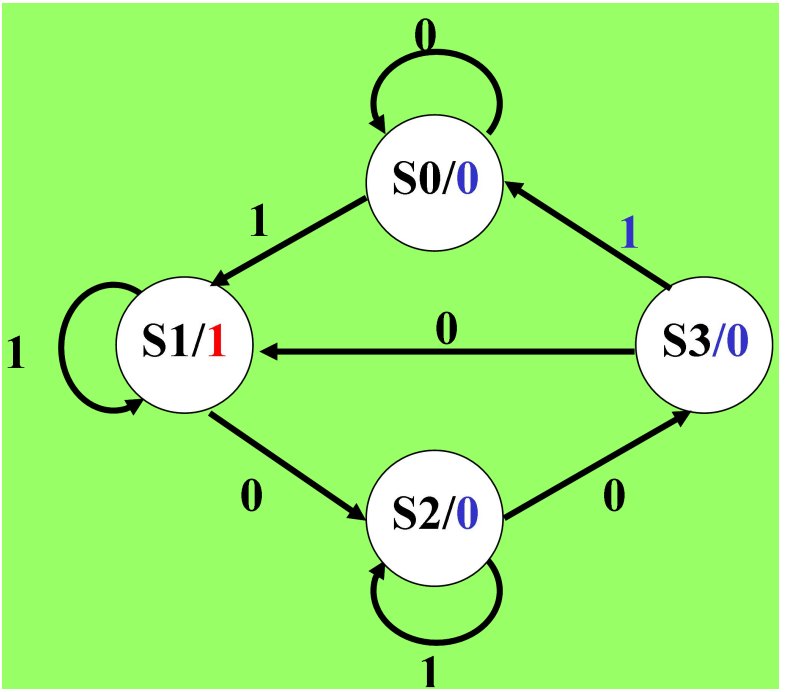
时序进程：描述从次态到现态转换；

—— *两段式有限状态机*



Moore 型状态机的描述

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY moore IS
  PORT ( clk,x  : IN STD_LOGIC;
                y: OUT STD_LOGIC );
END moore;

ARCHITECTURE behv OF moore IS
      TYPE State IS (s0, s1, s2, s3);
     SIGNAL current_state, next_state: State;
BEGIN
    PROCESS (clk) -- 时序逻辑
          BEGIN
             IF (clk'EVENT and clk='1' ) THEN
                  current_state <= next_state;
             END IF ;
          END PROCESS;
```

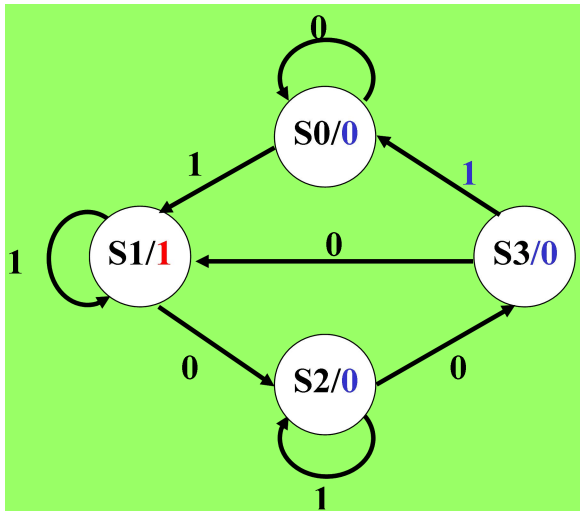


第一进程，完成状态转换，必须饱含时钟敏感信号

第二进程：输出和状态逻辑间关系，必须包含现态、输入等敏感信号

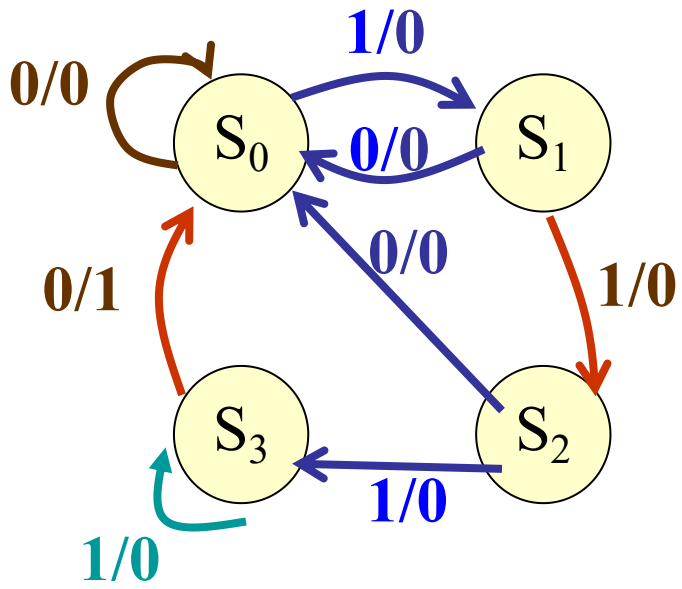
```
PROCESS (current_state, x) —— 组合逻辑
    BEGIN
      CASE current_state IS
        WHEN s0 => y <= '0';
                IF x = '0' THEN
                    next_state <= s0;
                ELSE
                    next_state <= s1;
                END IF;
        WHEN s1 => y <= '1';
                IF x = '0' THEN
                    next_state <= s2;
                ELSE
                    next_state <= s1;
                END IF;
        WHEN s2 => y <= '0';
                IF x = '0' THEN
                    next_state <= s3;
                ELSE
                    next_state <= s2;
                END if;
        WHEN s3 => y <= '0';
                IF x = '0' THEN
                    next_state <= s1;
                ELSE
                    next_state <= s0;
                END IF;
      END CASE;
    END process;

END behy;
```

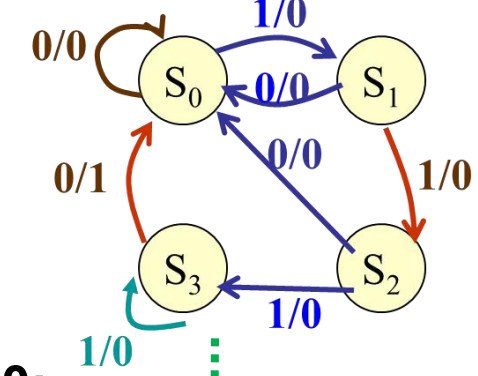
摩尔型



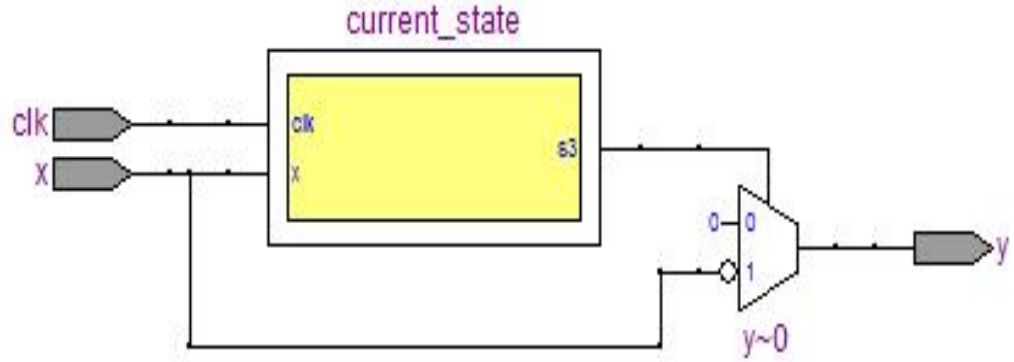
Mealy型有限状态机的设计

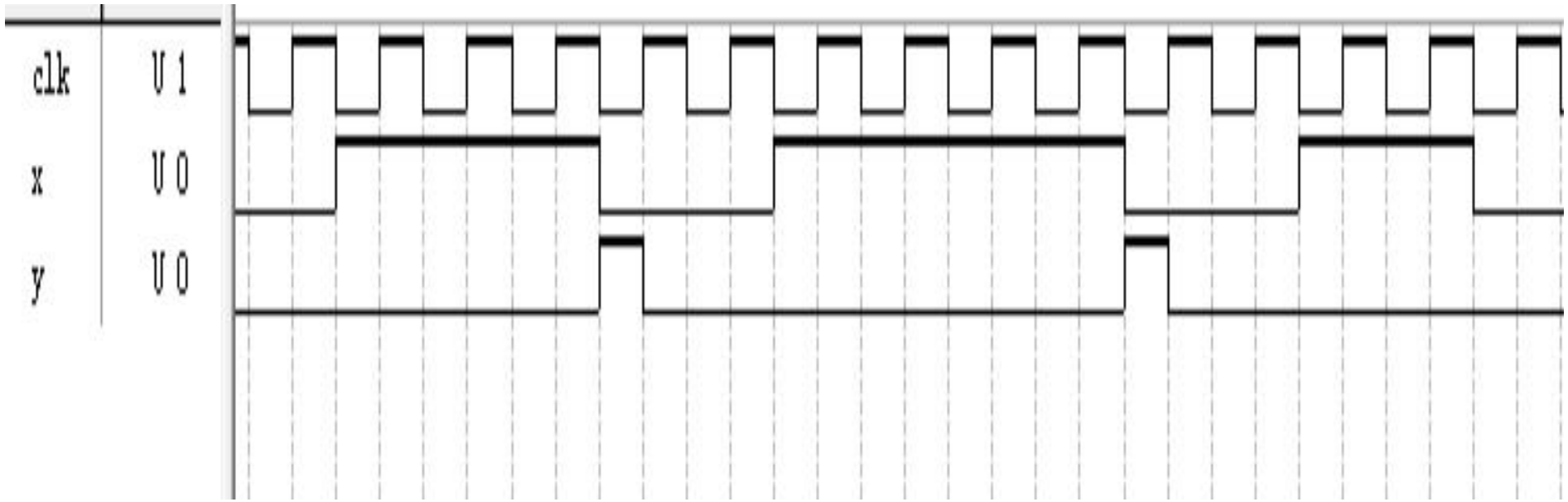
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY mealy IS
  PORT ( clk , x  : IN STD_LOGIC;
                y: OUT STD_LOGIC );
END mealy;

ARCHITECTURE behv OF mealy IS
      TYPE state IS (s0, s1, s2, s3);
     SIGNAL current_state, next_state: state;
BEGIN
  PROCESS (clk)
      BEGIN
         IF (clk'EVENT and clk='1' ) THEN
              current_state <= next_state;
         END IF ;
      END PROCESS;
```






```
PROCESS (current_state, x)
    BEGIN
      CASE current_state IS
        WHEN s0 => IF x = '0' THEN
                       next_state <= s0;
                       y <= '0';
                   ELSE
                       next_state <= s1;
                       y <= '0';
                   END IF;
        WHEN s1 => IF x = '0' THEN
                       next_state <= s0;
                       y <= '0';
                   ELSE
                       next_state <= s2;
                       y <= '0';
                   END IF;
        WHEN s2 => IF x = '0' THEN
                       next_state <= s0;
                       y <= '0';
                   ELSE
                       next_state <= s3;
                       y <= '0';
                   END IF;
        WHEN s3 => IF x = '0' THEN
                       next_state <= s0;
                       y <= '1';
                   ELSE
                       next_state <= s3;
                       y <= '0';
                   END IF;
      END case;
    END PROCESS;
END behv;
```

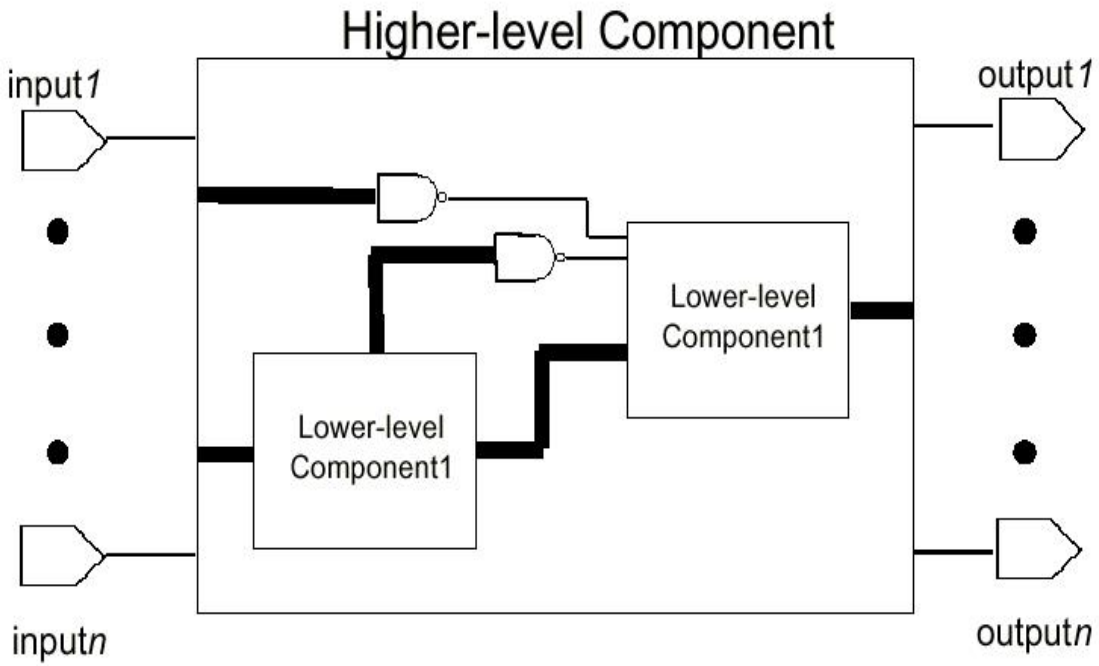

current_state
clk
x
clk
x
s3
0
0
1
y
y~0


clk
U 1
x
U 0
y
U 0

5.5.4 VHDL的层次化（结构化）设计与元件例化(component)语句

在多层次的设计中，高层次的设计模块调用低层次的设计模块，构成模块化的设计。重点描述模块间的连接关系。

由元件声明、元件例化两部分关键语句组成。



被例化元件的来源：

- VHDL设计实体；
- 其它HDL设计实体；
- 厂商提供的工艺库中的元件、IP核。

层次化设计的优点：

- 分工
- 共享
- 移植
- 周期短

一个元件是一段结构完整的常用代码，包括声明，实体和结构体，使用component可以使代码具有层次化的结构。

1. 元件声明：在结构体说明区中对所调用的低层次的实体模块（名称、端口类型等）声明为元件。

语法格式：

```
COMPONENT 元件名
    PORT (端口名表);
END COMPONENT;
```

2. 元件例化：将低层次元件调用、安装到当前层次。给出连接映射。

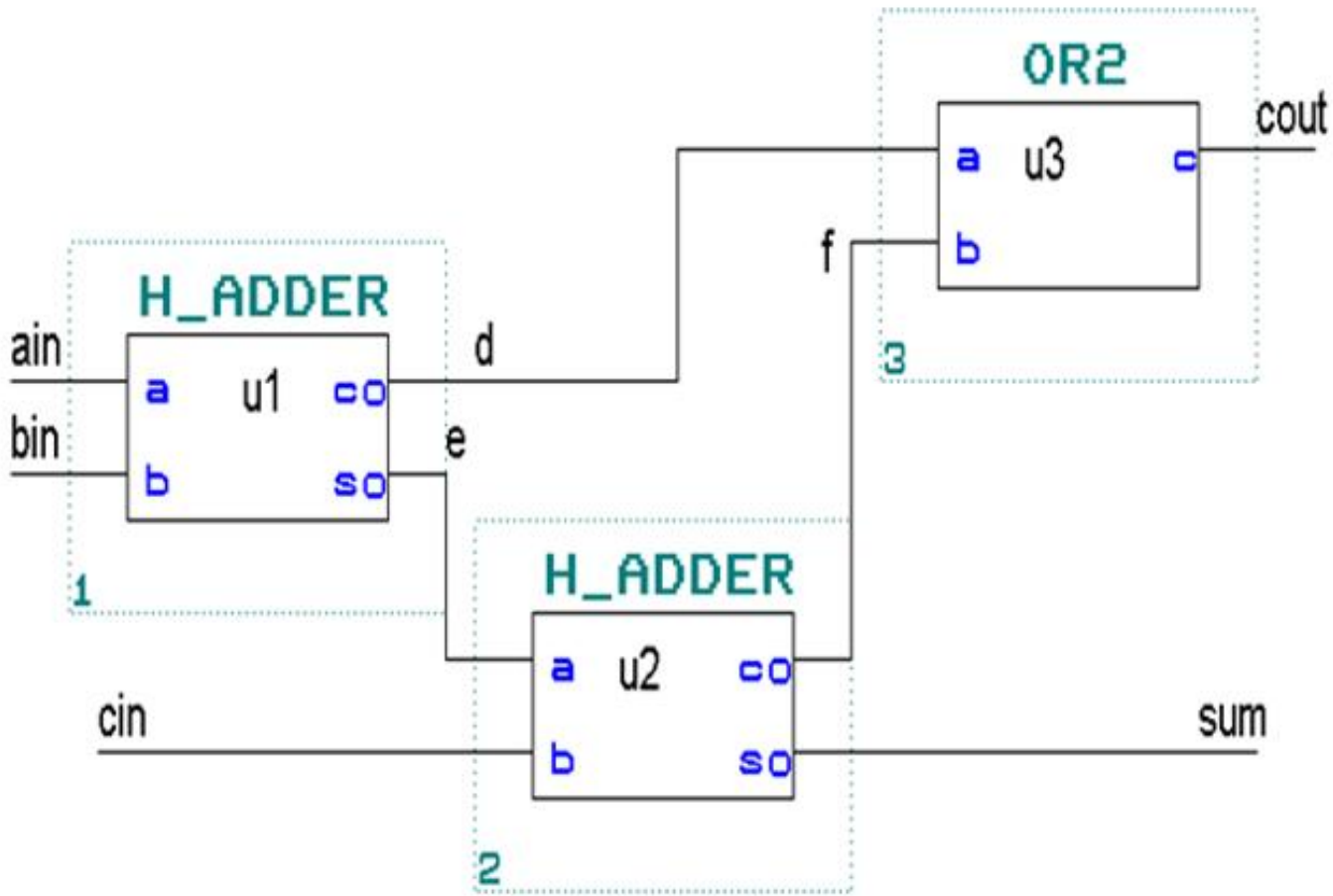
语法格式：

```
例化名：低层元件名 PORT MAP ( 端口列表）;
```

端口列表：低层端口名=> 当前名称

（或，直接位置映射）

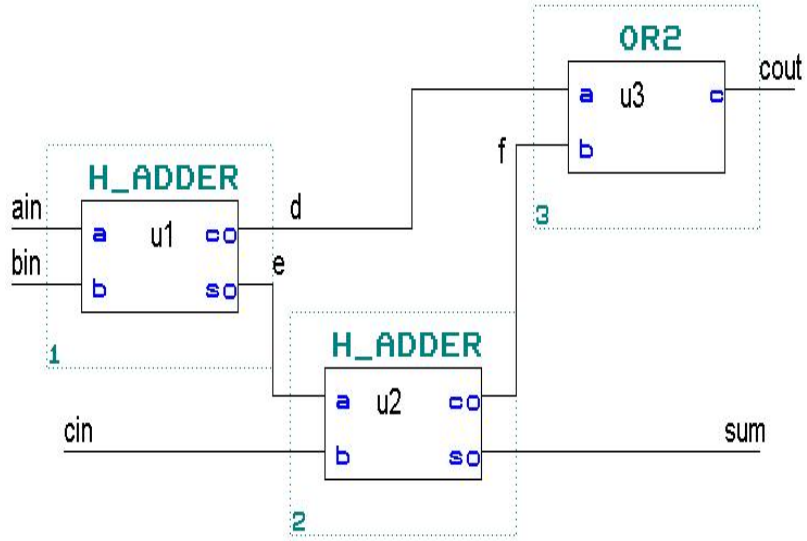
一位全加器原理图


OR2
u3
a
b
o
cout
f
3
H_ADDER
u1
ain
bin
a
b
co
so
d
e
1
H_ADDER
u2
cin
a
b
co
so
sum
2

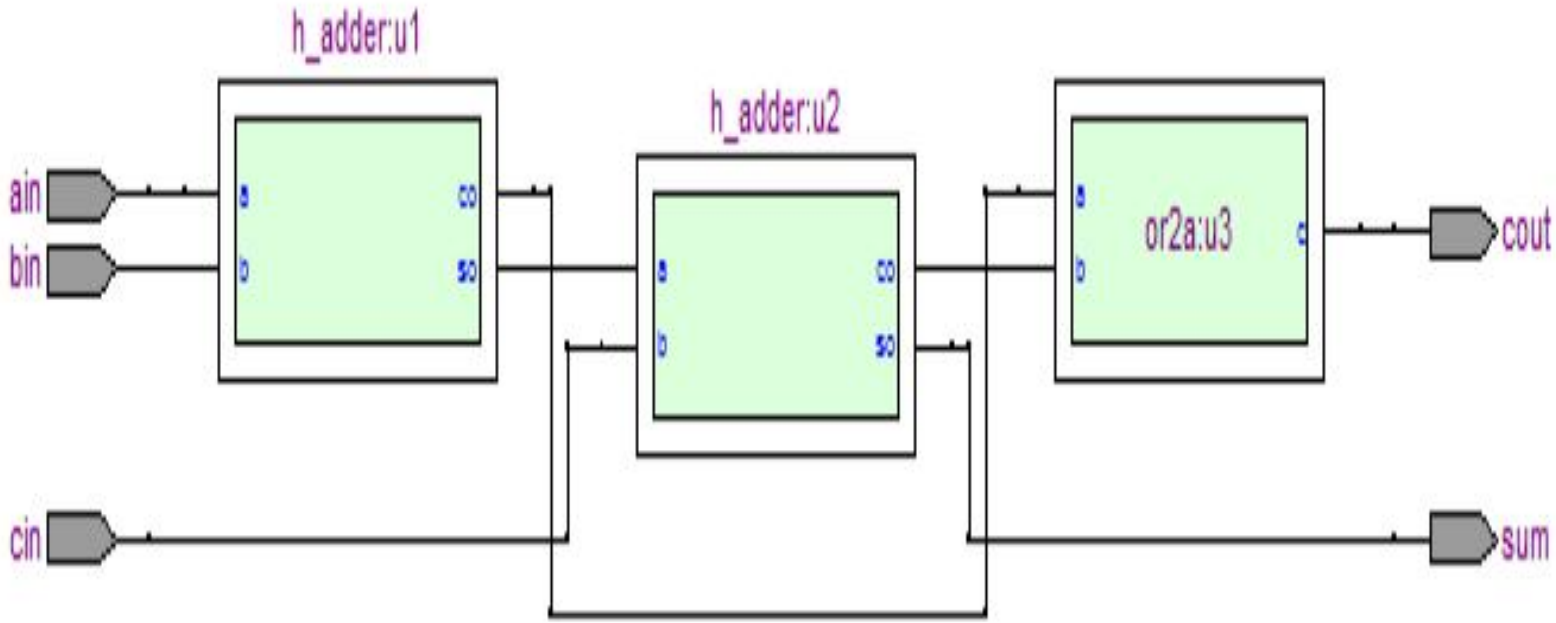
```
LIBRARY  IEEE ;--或门逻辑描述
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY or2a IS
  PORT (a, b :IN STD_LOGIC;
           c : OUT STD_LOGIC );
END or2a ;
ARCHITECTURE art1 OF or2a IS
  BEGIN
  c <= a OR b ;
END art1;
```

```
LIBRARY  IEEE; --半加器描述
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY h_adder IS
  PORT (a, b : IN STD_LOGIC;
        co, so : OUT STD_LOGIC);
END h_adder;
ARCHITECTURE art2 OF h_adder  is
BEGIN
  so <= a XOR b ;
  co <= a AND b ;
END art2;
```

```
LIBRARY  IEEE;    --1位二进制全加器顶层设计描述
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY f_adder IS
  PORT (ain, bin, cin  : IN STD_LOGIC;
                cout, sum   : OUT STD_LOGIC );
END f_adder;
ARCHITECTURE art3 OF f_adder IS
  COMPONENT h_adder
    PORT (  a, b :  IN STD_LOGIC;
            co, so :  OUT STD_LOGIC);
  END COMPONENT ;  ——元件声明
  COMPONENT or2a
     PORT (a, b : IN STD_LOGIC;
              c : OUT STD_LOGIC);
  END COMPONENT;
 SIGNAL d, e, f  :  STD_LOGIC;
 BEGIN
  u1 : h_adder PORT MAP (ain, bin, d, e);
  u2 : h_adder PORT MAP (a=>e, b=>cin, co=>f, so=>sum);
  u3 :   or2a  PORT MAP (d, f, cout);——元件例化
END art3;
```

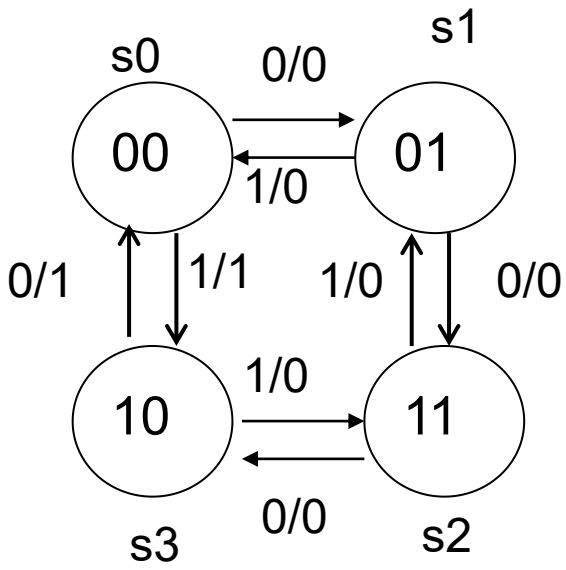


必须与元件例化中的端口顺序一致


h_adder:u1
h_adder:u2
or2a:u3
ain
bin
cin
cout
sum
a
b
co
so
a
b
c

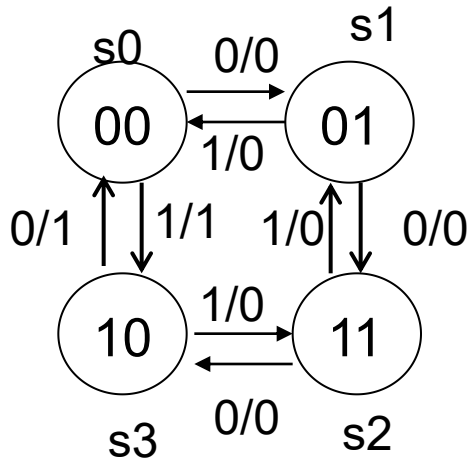
VHDL　举例

4进制格雷码可逆计数器 X=0时顺时针

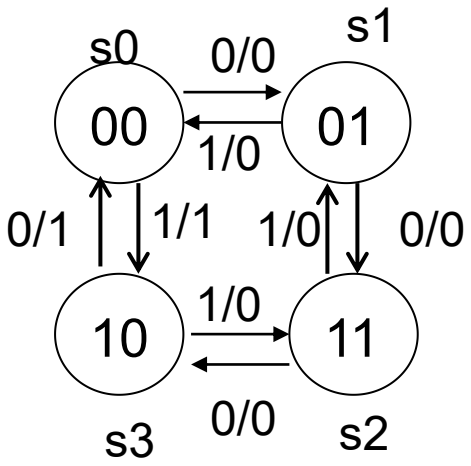


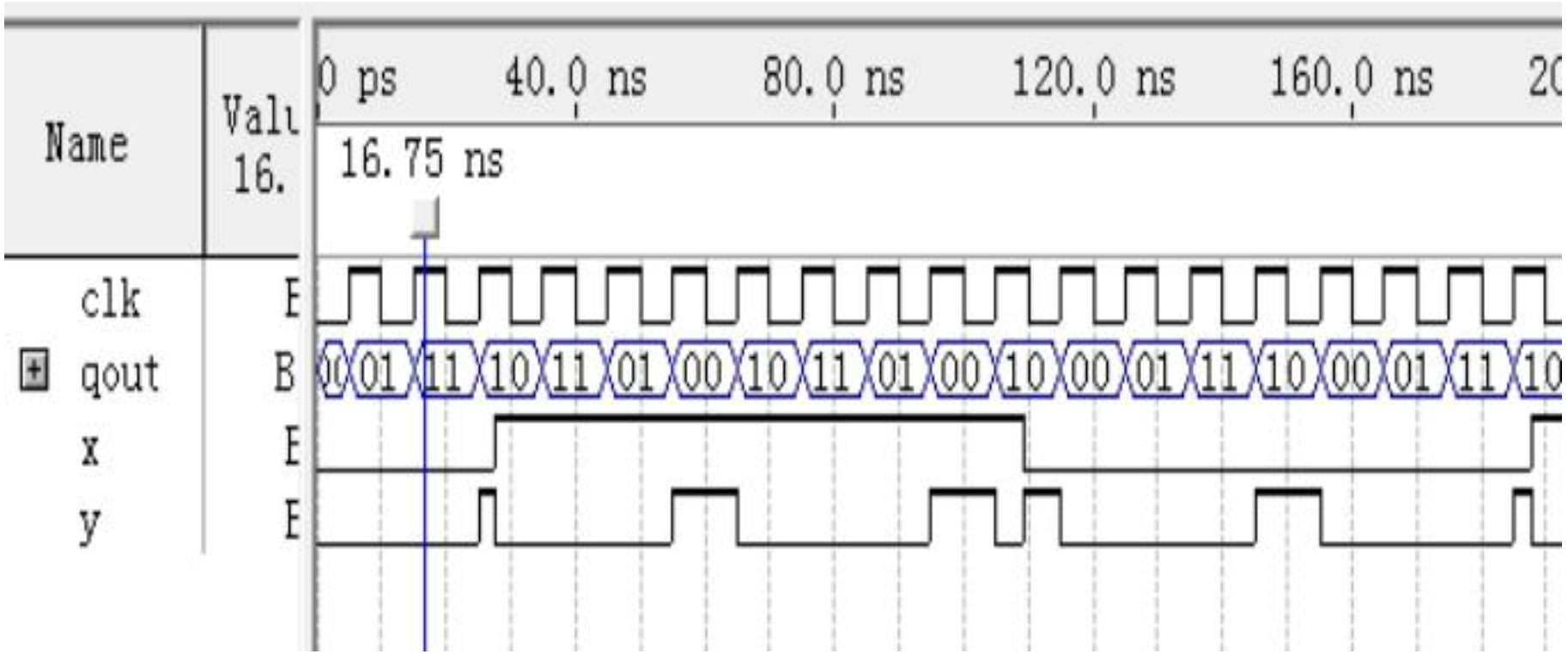

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY count4 IS
  PORT ( clk , x  : IN STD_LOGIC;
                qout : OUT STD_LOGIC _VECTOR (1 DOWNTO 0);
               y: OUT STD_LOGIC );
END count4;

ARCHITECTURE behv OF count4 IS
      TYPE state IS (s0, s1, s2, s3);
     SIGNAL c_state, n_state : state;
BEGIN
 REG: PROCESS (clk)
      BEGIN
         IF (clk'EVENT and clk='1' ) THEN
              c_state <= n_state;
         END IF ;
      END PROCESS;
```

```
COM:PROCESS (c_state, x)
    BEGIN
      CASE c_state IS
        WHEN s0 => qout <= "00" ;
                   IF x = '0' THEN
                      n_state <= s1;
                      y <= '0';
                    ELSE
                      n_state <= s3;
                      y <= '1';
                   END IF;
        WHEN s1 => qout <= "01" ;
                   IF x = '0' THEN
                      n_state <= s2;
                      y <= '0';
                    ELSE
                      n_state <= s0;
                      y <= '0';
                   END IF;
```



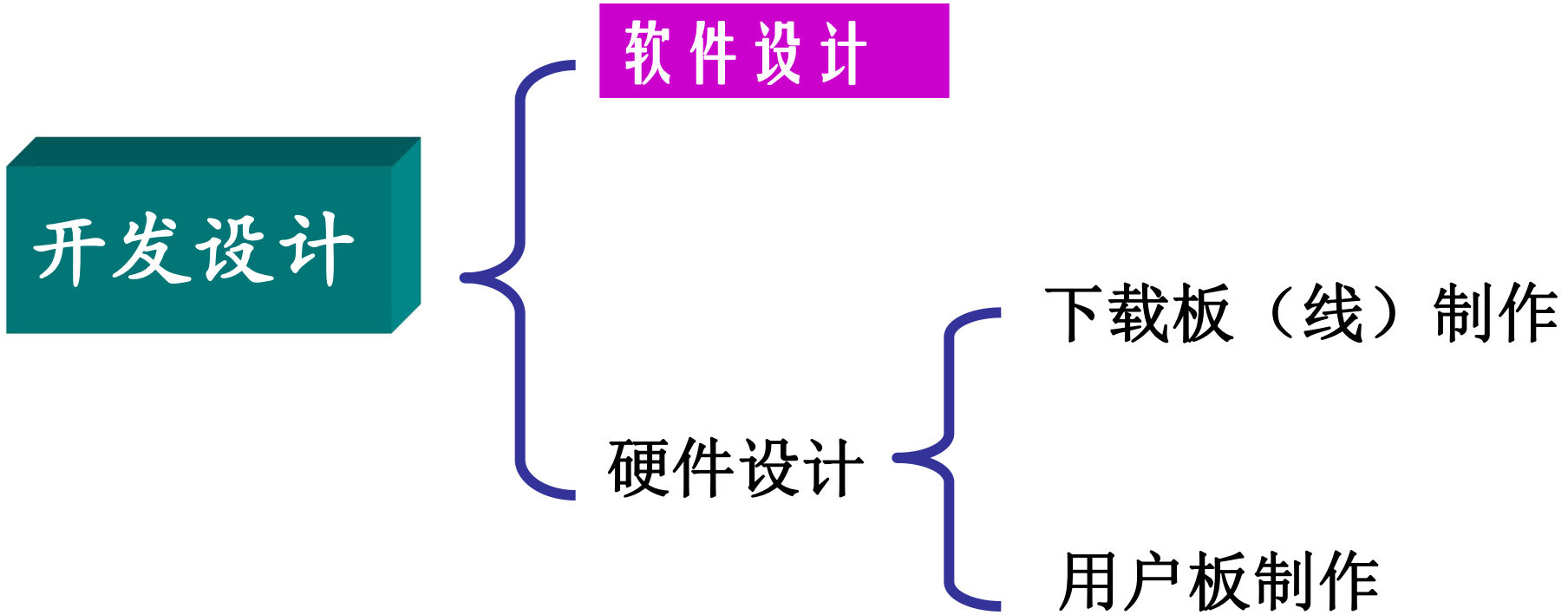

```
        WHEN s2 => qout <= "11" ;
                   IF x = '0' THEN
                      n_state <= s3;
                      y <= '0';
                    ELSE
                      n_state <= s1;
                      y <= '0';
                   END IF;
        WHEN s3 => qout <= "10" ;
                   IF x = '0' THEN
                      n_state <= s0;
                      y <= '1';
                    ELSE
                      n_state <= s2;
                      y <= '0';
                   END IF;
      END case;
    END PROCESS;
END behv;
```


s0
0/0
s1
00
01
1/0
0/1
1/1
1/0
0/0
1/0
10
11
0/0
s3
s2

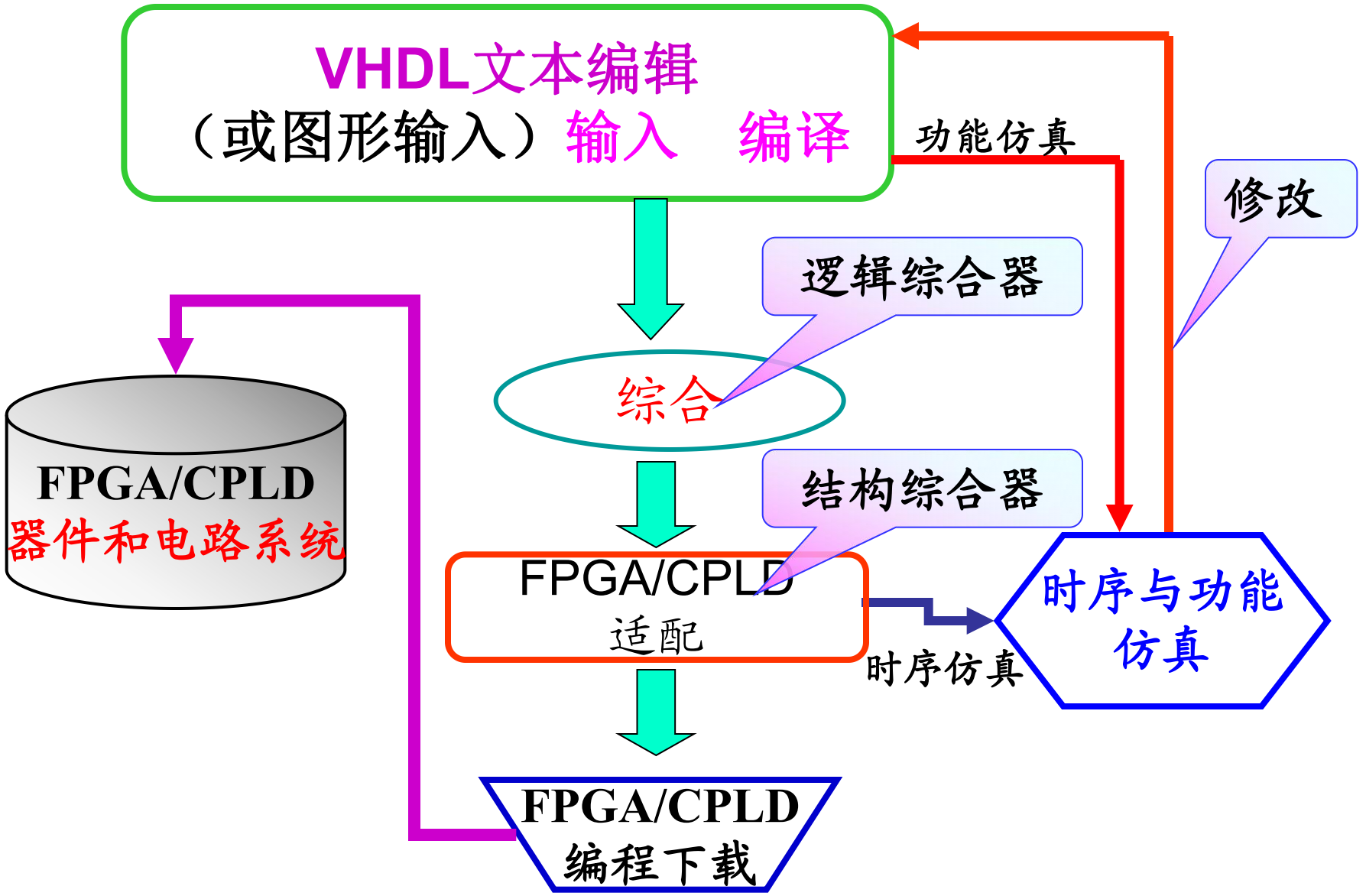

Name
Valu
16.
0 ps
40.0 ns
80.0 ns
120.0 ns
160.0 ns
16.75 ns
clk
qout
x
y
01 11 10 11 01 00 10 11 01 00 10 00 01 11 10 00 01 11 10

可编程逻辑器件器件的开发流程

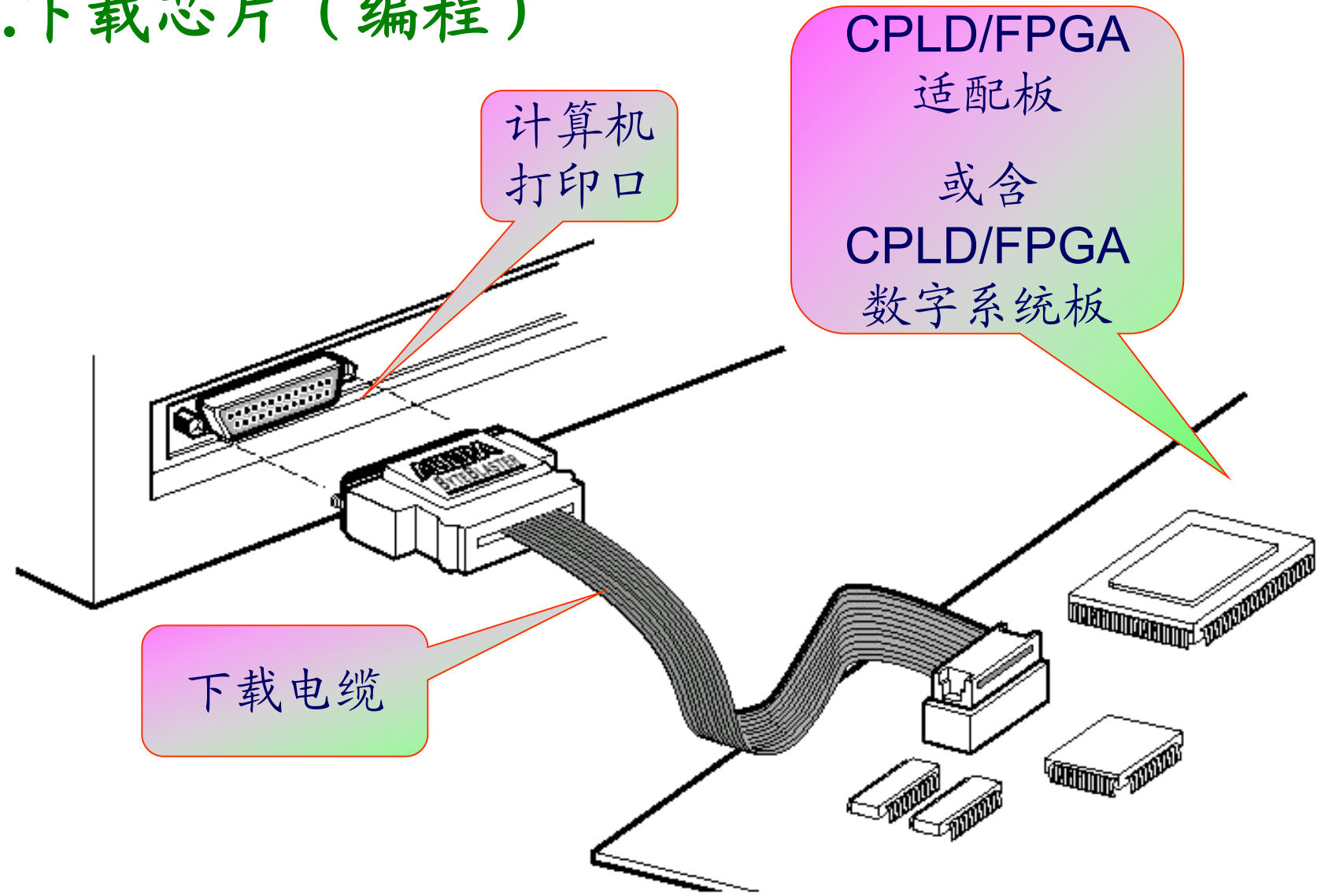
开发工具



三、软件设计流程


VHDL文本编辑
（或图形输入）输入　编译
功能仿真
修改
逻辑综合器
综合
结构综合器
FPGA/CPLD
器件和电路系统
FPGA/CPLD
适配
时序仿真
时序与功能
仿真
FPGA/CPLD
编程下载

四.下载芯片（编程）



8个彩灯，4花样自动切换的彩灯控制器：
第1：从右到左，然后从左到右逐次点亮，再全黑，再全亮；
第2：两边同时亮1个逐次向中间移动再散开；
第3：两边同时亮2个逐次向中间移动再散开；
第4：每三个亮、每四个亮、间隔亮。
共32个种变换，循环往复。

```
Library ieee;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_unsigned.ALL;

entity caideng is
Port (clk:in std_logic;
      q:out std_logic_vector(7 downto 0));
end;

architecture art of caideng is
signal s:std_logic_vector(4 downto 0);
```

```
Begin

  process (clk)
  begin
    if clk'event and clk='1' then
          if s = 31 then
              s<="00000";
         else
               s<=s+1;   --不用判断
          end if;
     end if;
  end process;

 process (s)
  begin
case s is
   when "00000"=>q<="00000001";
   when "00001"=>q<="00000010";
   when "00010"=>q<="00000100";
   when "00011"=>q<="00001000";
   when "00100"=>q<="00010000";
   when "00101"=>q<="00100000";
   when "00110"=>q<="01000000";
   when "00111"=>q<="10000000";
   when "01000"=>q<="10000000";
   when "01001"=>q<="01000000";
   when "01010"=>q<="00100000";
   when "01011"=>q<="00010000";
   when "01100"=>q<="00001000";
   when "01101"=>q<="00000100";
   when "01110"=>q<="00000010";
   when "01111"=>q<="00000001";
   when "10000"=>q<="00000000";
    when "10001"=>q<="11111111";
   when "10010"=>q<="10000001";
   when "10011"=>q<="01000010";
   when "10100"=>q<="00100100";
   when "10101"=>q<="00011000";
   when "10110"=>q<="00100100";
   when "10111"=>q<="01000010";
   when "11000"=>q<="10000001";
   when "11001"=>q<="11000011";
   when "11010"=>q<="00111100";
```

```
   when "11011"=>q<="11000011";
   when "11100"=>q<="11100111";
    when "11101"=>q<="11110000";
   when "11110"=>q<="00001111";
   when "11111"=>q<="01010101";
   when others=>null;
 end case;
end process;
End art;
```

序列信号发生器如何实现？

顺序脉冲发生器如何实现？