5.5.3 VHDL的行为描述

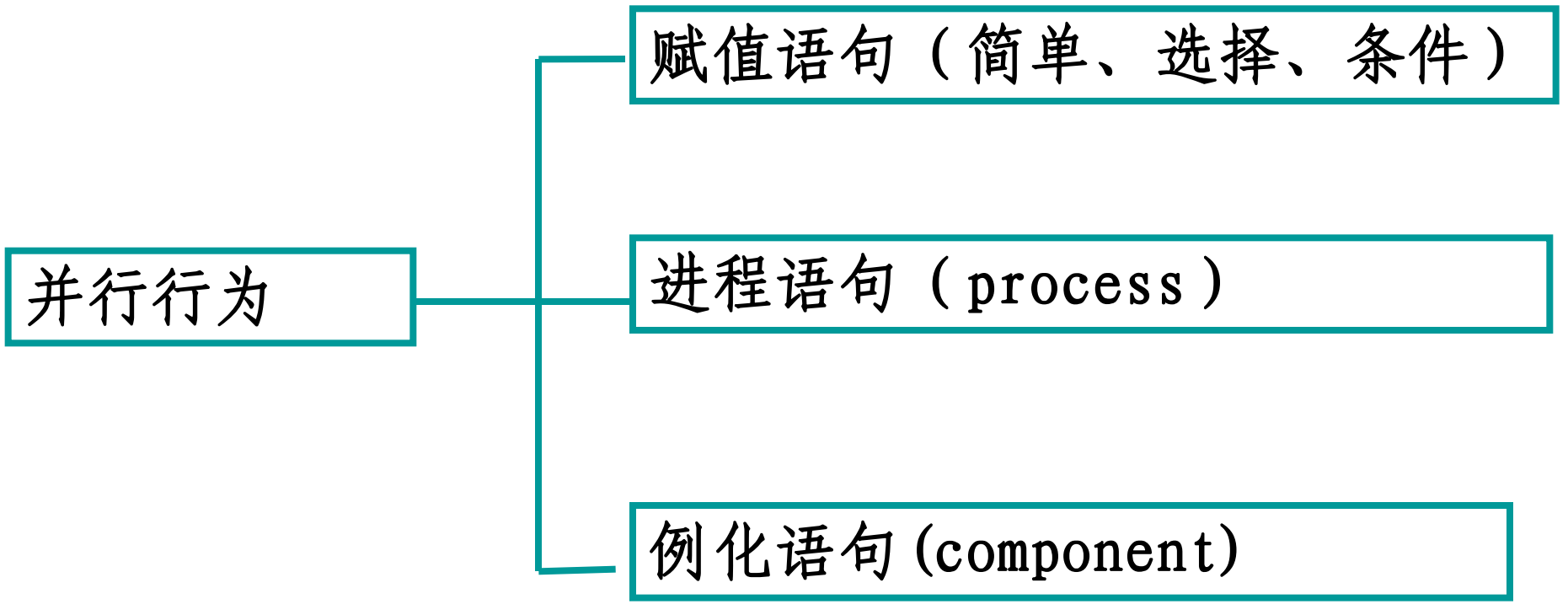
行为描述语句
- 并发（Concurrent）描述语句
- 顺序（Sequential）描述语句

顺序语句：描述逻辑关系、算法

并行语句：硬件并行的特点

一、VHDL的并行语句

描述硬件最基本的本质特性——并行行为。



进程行为之间并行关系，进程内部是顺序行为。

1. 赋值语句：

特点：

- 执行与书写顺序无关。
- 每一个赋值相当于一个进程。

（1） 简单并行赋值：

语法格式

```
信号 <= 表达式；
```

赋值的原则：相同位宽，相同数据类型。

```
ENTITY exe IS
      port  ( a, b: IN bit;
                 y: OUT bit );
 END exe;

 ARCHITECTURE art OF exe  IS
      SIGNAL c :  bit ;
 BEGIN
      y <= c ;
      c <= a and b ;
 END  art1;
```

(2) 选择赋值 with-select-when

语法格式

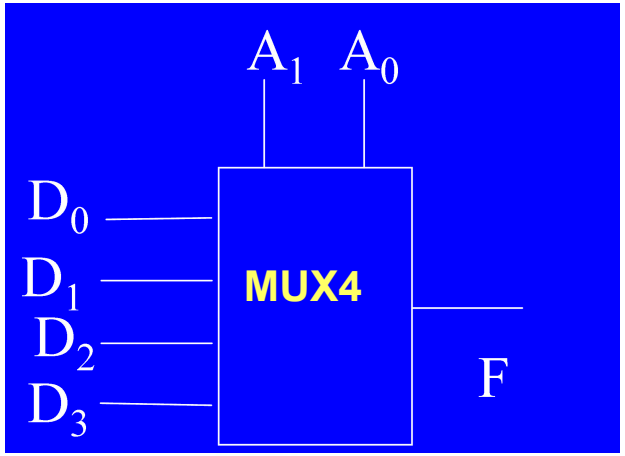
```
WITH 选择表达式 SELECT
信号名<=表达式1 WHEN 选择值1,
        表达式2 WHEN 选择值2,
            ...
        表达式n WHEN others ;
```

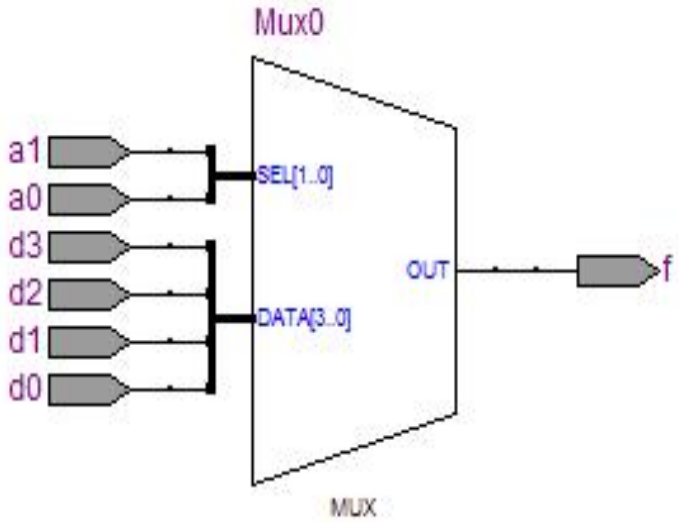
结束为“,”

- 不能有重叠的条件分支。
- 最后条件为 others。选择值必须覆盖所有取值可能。
- 没有优先级判断。
- 只能描述简单的组合逻辑。

例：用选择赋值语句描述四选一电路

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

entity mux4 is
    port (d0, d1, d2, d3, a0, a1 : in std_logic;
                                f : out std_logic);
end  mux4;

architecture  rtl  of  mux4  is
    signal  sel : std_logic_vector (1 downto 0);
begin
    sel <= a1 & a0;
    with sel select
       f <= d0  when “00” ,
            d1  when “01” ,
            d2  when “10” ,
            d3  when others;
end  rtl;
```





编译之后，点击
Tool>>Netlist Viewers>>RTL Viewers

(3) 条件赋值when-else

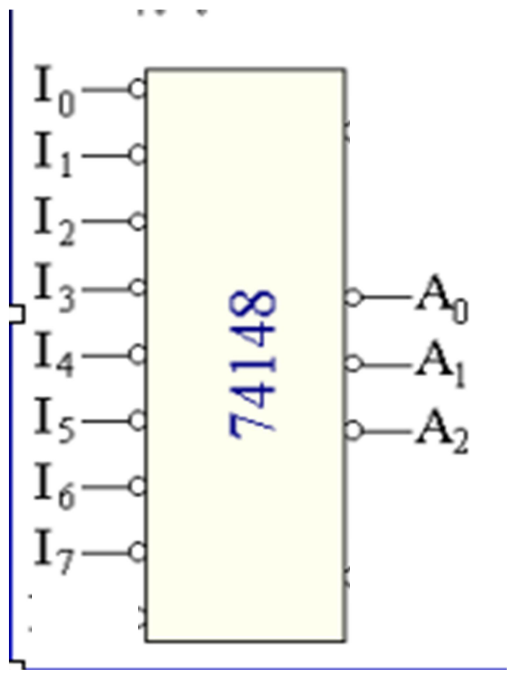
语法格式

```
信号名<=表达式1  WHEN 条件式1 ELSE
        表达式2  WHEN 条件式2 ELSE
          .   .       .条件式n-1 ELSE
        表达式n;
```

注意:

- 最后的Else 项是必须的;
- 有优先级逻辑关系，先判断第一条件;
- 类似 if （在process中使用）的嵌套语句;
- 只能描述简单的组合逻辑。

例：8－3优先编码器。

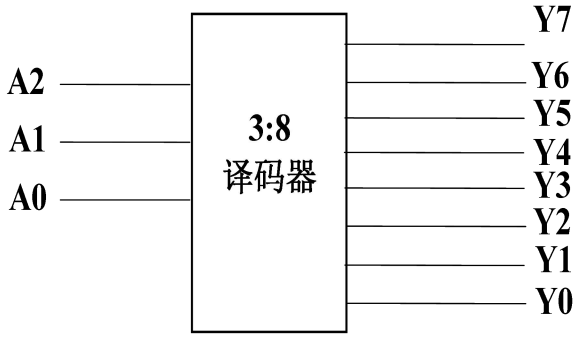



```
library ieee;
use ieee.std_logic_1164.all;

entity coder is
  port (
          I : in std_logic_vector(7 downto 0);
          A : out std_logic_vector(2 downto 0)
        ) ;
end coder;
```

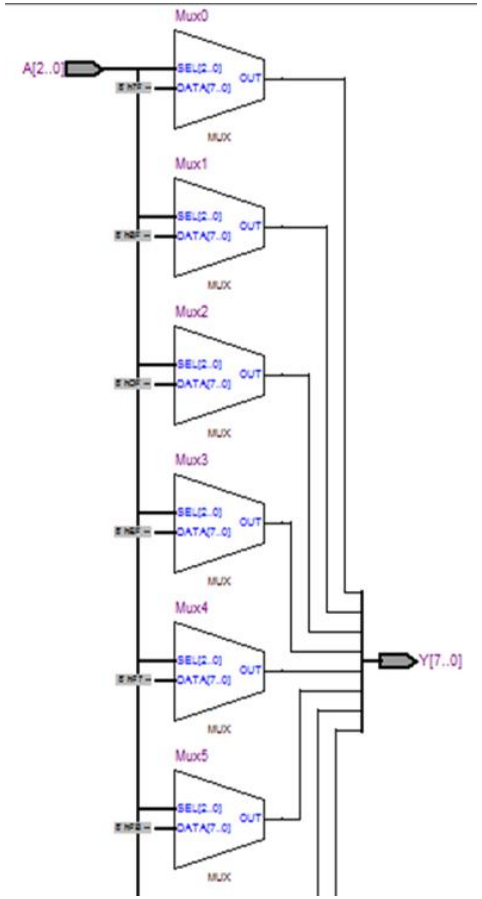
```
architecture rtl of coder is
begin
        A <= "000"  when  I(7) = '0'  else
             "001"  when  I(6) = '0'  else
             "010"  when  I(5) = '0'  else
             "011"  when  I(4) = '0'  else
             "100"  when  I(3) = '0'  else
             "101"  when  I(2) = '0'  else
             "110"  when  I(1) = '0'  else
             "111" ;
End  rtl;
```

语句	With-select-when	When-else	If-else	Case-when
选择条件	一个信号的不同值，互斥	多个信号多种组合，不必互斥	多个信号多种组合，不必互斥	一个信号的不同值，互斥
语句属性	并行	并行	顺序	顺序
用途	编码、译码、多路选择器	优先编码器，地址译码器	优先编码器，地址译码器	编码、译码、多路选择器 状态机

设计3:8译码器

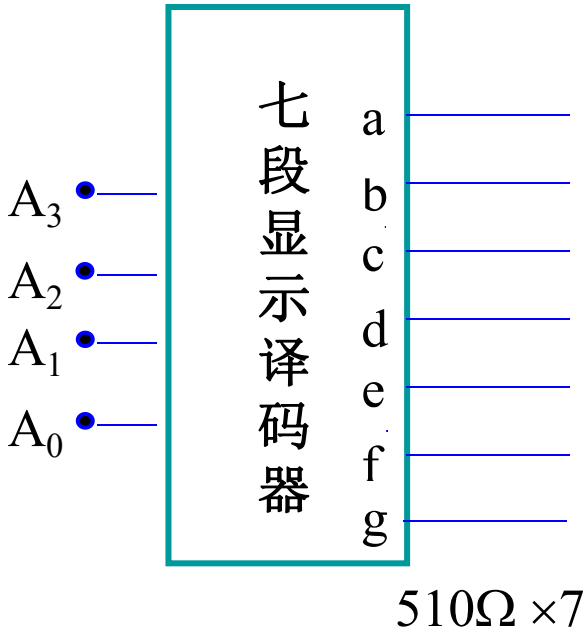


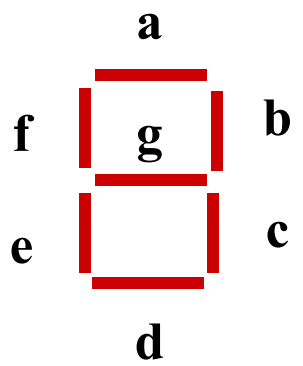

```
library ieee;
use ieee.std_logic_1164.all;

entity  dec38 is
  port(
    A : in std_logic_vector(2 downto 0);
    Y : out std_logic_vector(7 downto 0)
      );
end dec38;

architecture m1 of dec38 is
begin
   with A select
         Y<= "11111110" when "000",
             "11111101" when "001",
             "11111011" when "010",
             "11110111" when "011",
             "11101111" when "100",
             "11011111" when "101",
             "10111111" when "110",
             "01111111" when "111",
             "11111111" when others;
 end m1;
```



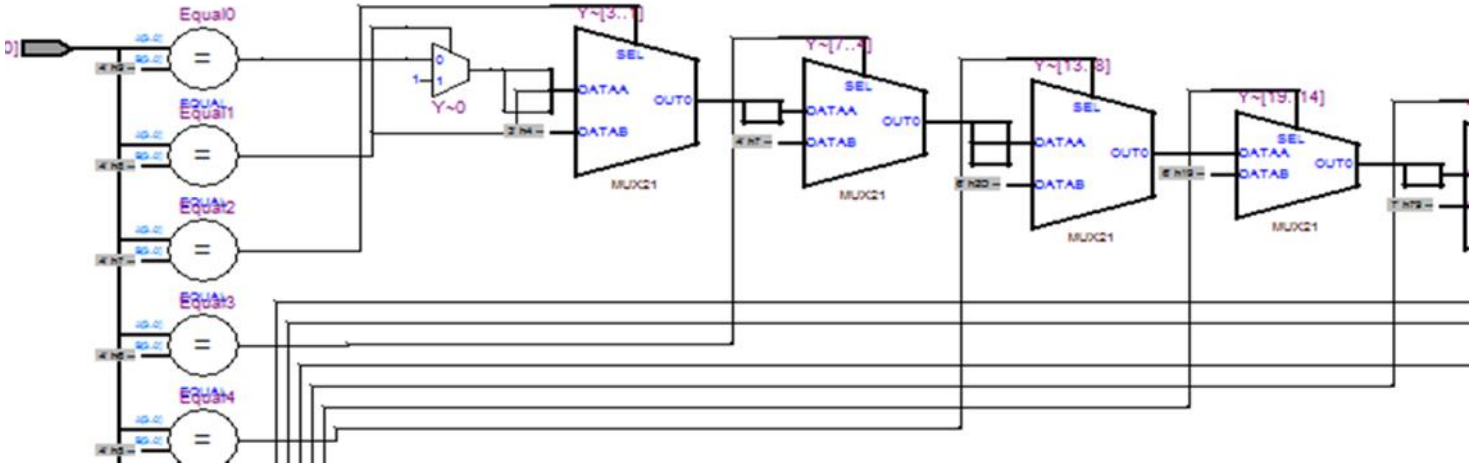
七段显示译码器

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

entity  bcd is
   port(
        A : in std_logic_vector(3 downto 0);
        Y : out std_logic_vector(6 downto 0)
       );
end bcd;
```





```
architecture m1 of bcd is
Begin
        y  <=  "1111110" when A="0000" else  --0
               "0001100" when A="0001" else  --1
               "1101101" when A="0010" else  --2
               "1111001" when A="0011" else  --3
               "0110011" when A="0100" else  --4
               "1011011" when A="0101" else  --5
               “0011111" when A="0110" else  --6
               "1110000" when A="0111" else  --7
               "1111111" when A="1000" else  --8
               "1110011" when A="1001" else  --9
               "0000000" ;
End m1;
```



3、进程语句

提供了一种用算法描述硬件行为的方法。几个进程语句之间是并行行为。外部并行，内部顺序。

语法格式

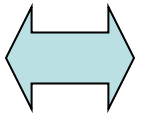
```
[进程名：] PROCESS （敏感信号表）
        [变量说明语句];
        BEGIN

          顺序说明语句;

        END PROCESS ;
```

敏感信号表:

- 进程中要读取的信号。
- 敏感信号的变化都将启动进程。
- 组合逻辑中，所有输入都作为敏感信号。

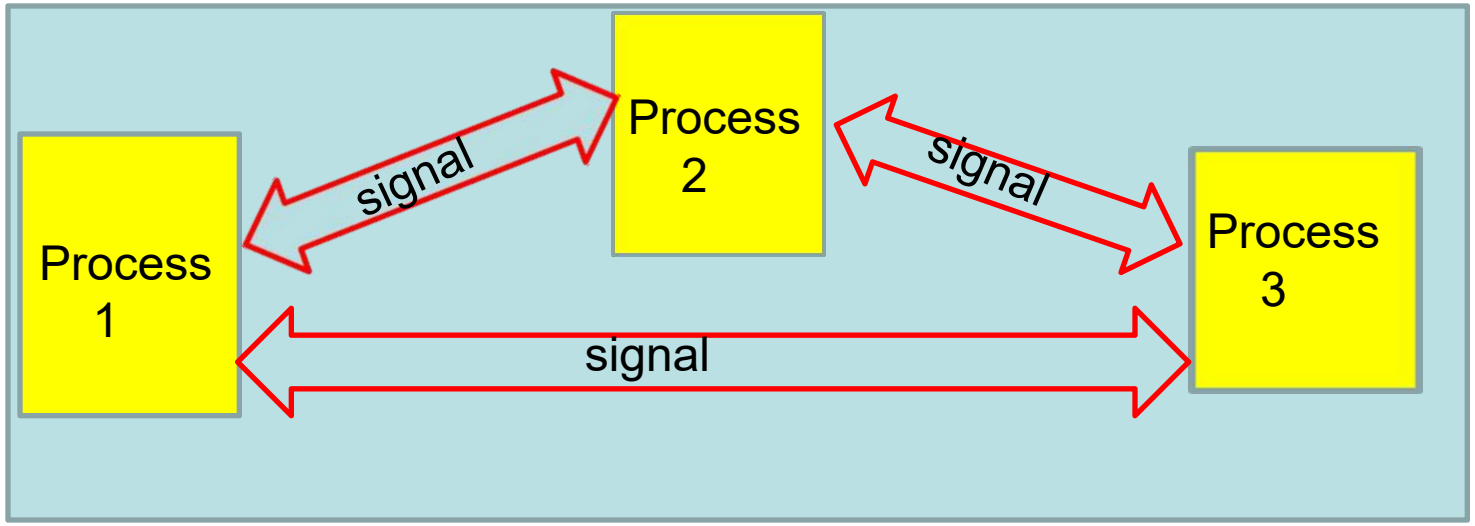
进程的并发特点：

```
architecture behav of a_var is
begin
    output<=a(i);
end behav;
```



```
architecture behav of a_var is
begin
    process(a, i)
    begin
        output<=a(i);
    end  process;
end behav;
```

一个简单并行信号赋值语句是一个进程的缩写。



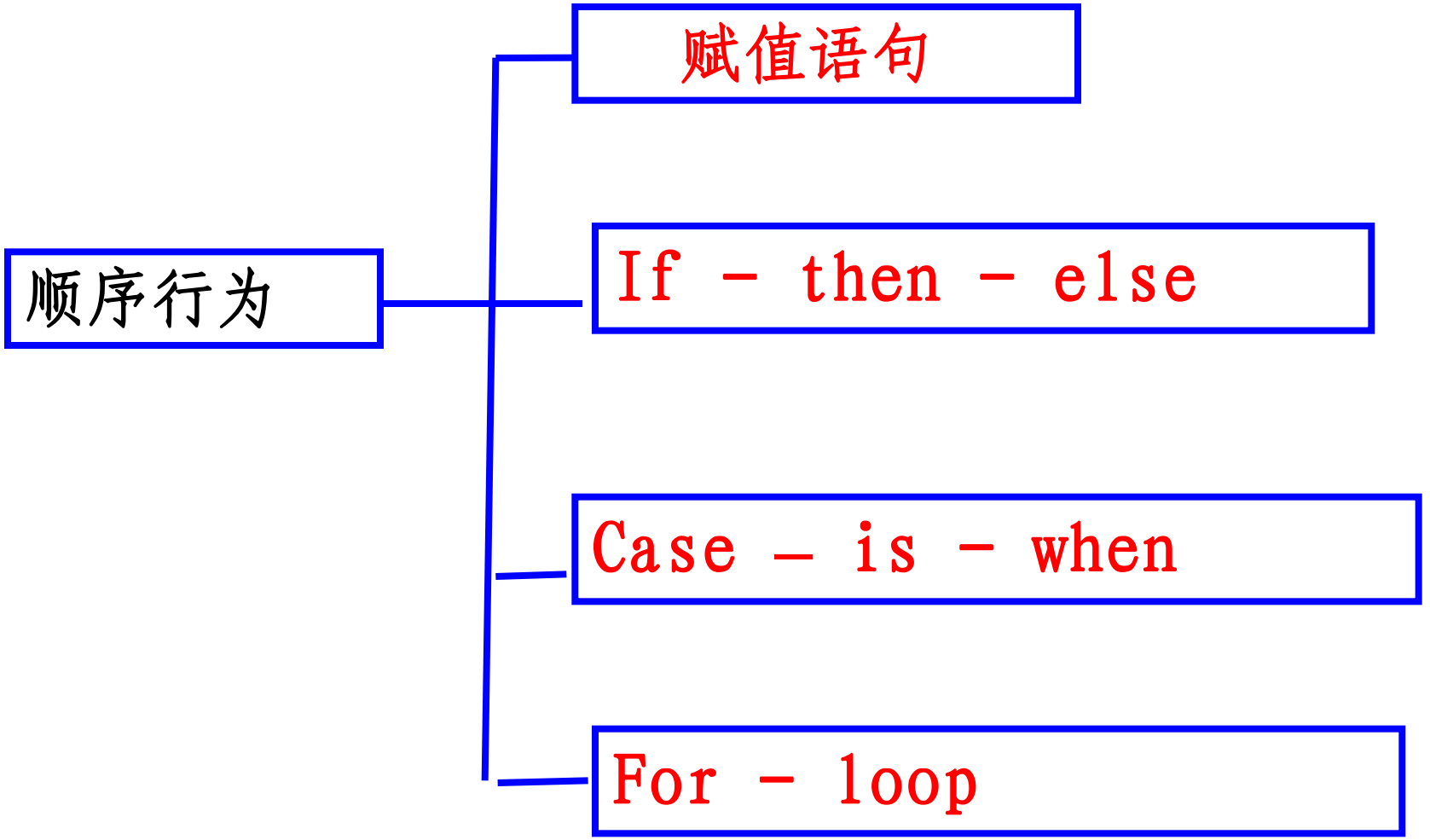
等效:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY ex1 IS
   PORT(a, b : IN STD_LOGIC;
            y : OUT STD_LOGIC);
END ex1;

ARCHITECTURE rtl OF ex1 IS
    SIGNAL c : STD_LOGIC;
BEGIN
    y <= c;
    c <= a and b;
END rtl;
```

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
ENTITY ex2 IS
    PORT(a,b : IN STD_LOGIC;
             y : OUT STD_LOGIC);
END ex2;

ARCHITECTURE rtl OF ex2 IS
    SIGNAL c : STD_LOGIC;
BEGIN
    PROCESS(a, b)
    BEGIN
        c <= a and b;
    END PROCESS;
    PROCESS(c)
    BEGIN
        y <= c;
    END PROCESS;
END rtl;
```

二、VHDL顺序语句

执行顺序与书写顺序一致。只能出现在进程（process）和子程序中。
描述逻辑关系，具体算法（类似C）。



（1）简单顺序赋值：

赋值对象可为信号、变量

语法格式

信号 <= 表达式 ；

变量 := 表达式 ；

赋值的原则：相同位宽，相同数据类型。

变量赋值与信号赋值的差异:

- 硬件实现的功能不同

  信号：实际的硬件连线;

  变量：电路单元内部的操作，代表暂存的临时数据。

- 赋值行为的不同

  信号赋值：延迟更新数值;

  变量赋值：立即更新数值。

- 信号的多次赋值

  a. 一个进程：最后一次赋值有效

  b. 多个进程：多源驱动

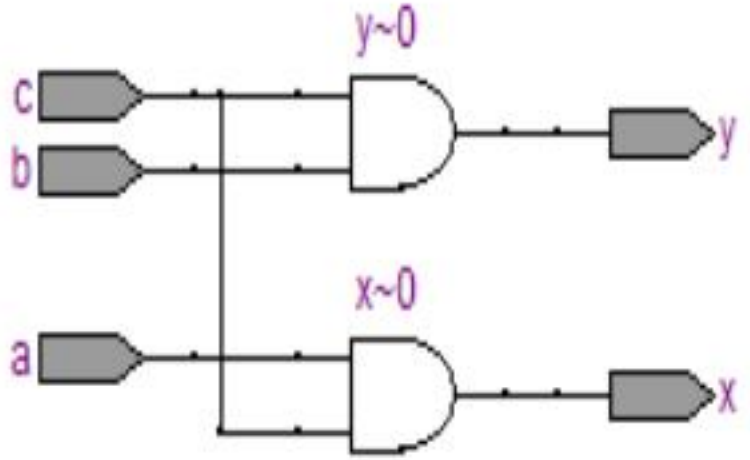
  线与、线或、三态总线

信号**signal**	变量**variable**
代表电路中的连接	代表局部的一个暂存值
全局变量	局部变量
赋值符号‘**<=**’	赋值符号‘**:=**’
赋值在一个过程结束后才会变化	赋值没有延迟，立即变化

例：信号赋值与变量赋值的比较

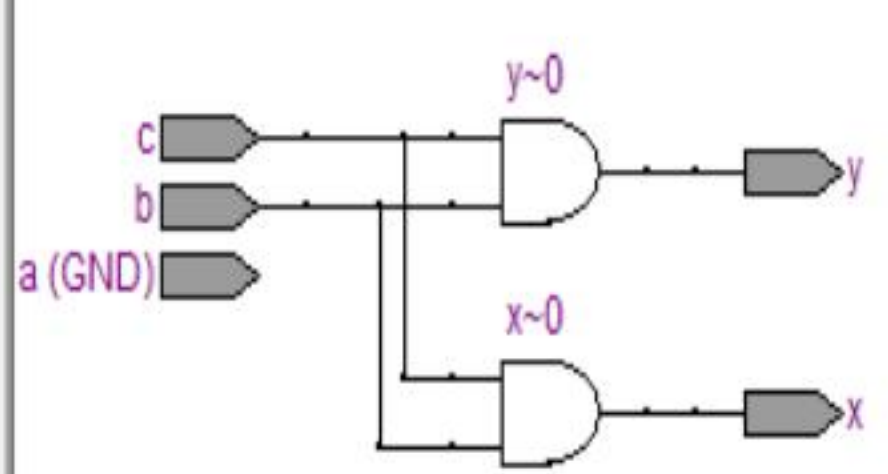
变量赋值：

```
architecture  rtl  of  var  is
begin
    process  (a,b,c )
    variable d :std_logic;
    begin
        d := a ;
        x <= c and d;
        d := b ;
        y <= c and d;
  end  process ;          -- 结果: x = ca
end rtl;                        y = cb
```



信号赋值：

```
architecture  rtl of sig  is
signal d : std_logic;
begin
  process( a, b,c )
  begin
      d <= a ;
      x <= c and d ;
      d <= b ;
      y <= c and d;
  end  process ;
end  rtl ;
```



读出（process中）

更新（end process后）

不要在一个process中对同一信号多次赋值

-- 结果: x = cb
y = cb

（2）转向控制语句：

通过条件控制决定是否执行一条或几条语句，或重新执行一条或几条语句，仿真时顺序进行。

主要有：

if 语句、case 语句、loop 语句

if 、case、loop语句必需在Process语句中。

1) if 语句

- if 语句的门闩控制（不完全if ）

```
if  条件式  then
    顺序处理语句;
end if ;
```

对于不完全的if语句，VHDL综合器将引进一个时序元件保持当前状态值。用于锁存器或触发器。

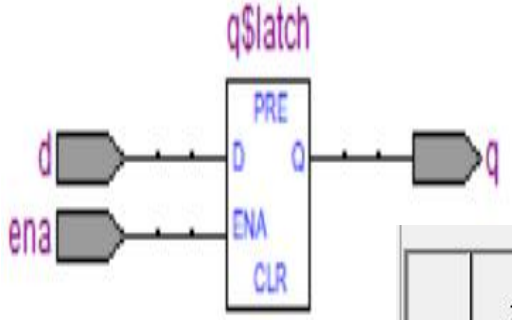
* if 语句的电平触发

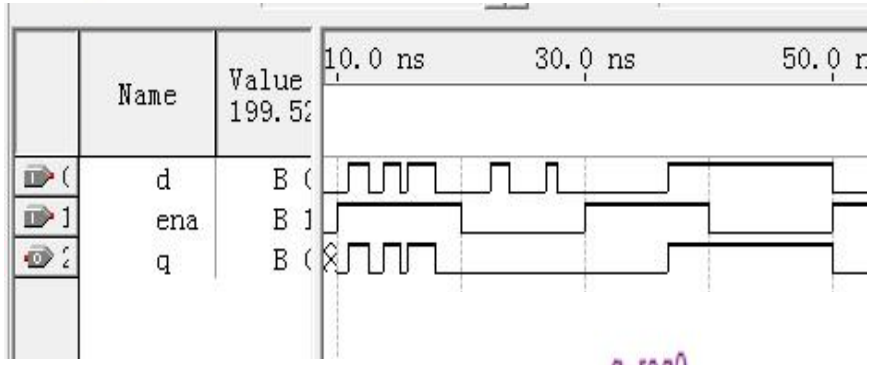
综合后生成锁存器（latch）

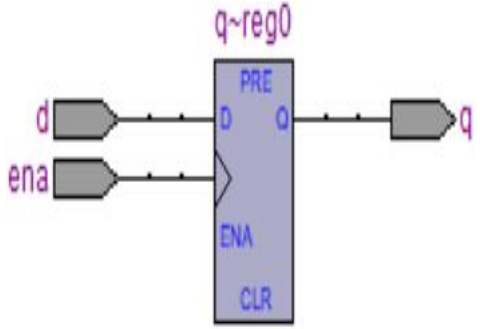
```
LIBRARY  IEEE;
USE IEEE.STD_LOGIC_1164.all;

ENTITY latch IS
     port ( ena, d: IN std_logic;
                    q: OUT std_logic );
END latch;

ARCHITECTURE rt1 OF latch  IS
BEGIN
   PROCESS ( ena ,d )
   BEGIN
      IF (ena= '1') then
            q <= d;
      END  IF;
   END PROCESS;
END  rt1;
```

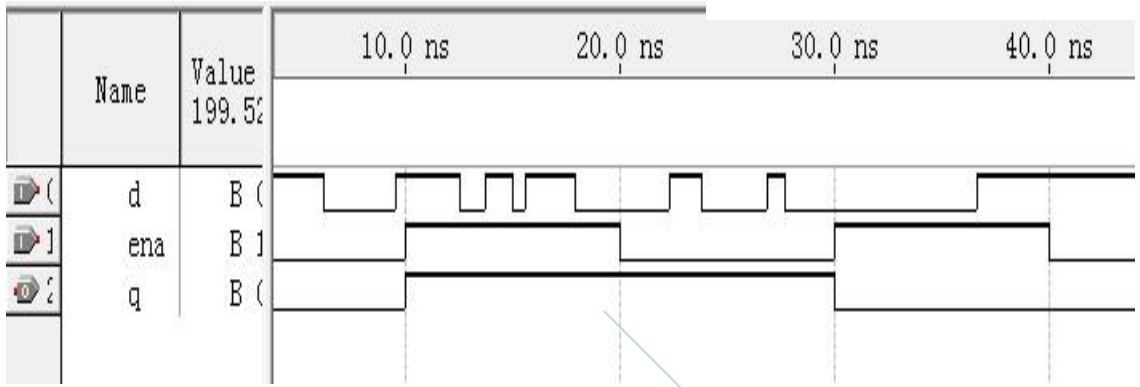
敏感信号表

敏感信号表无d的波形





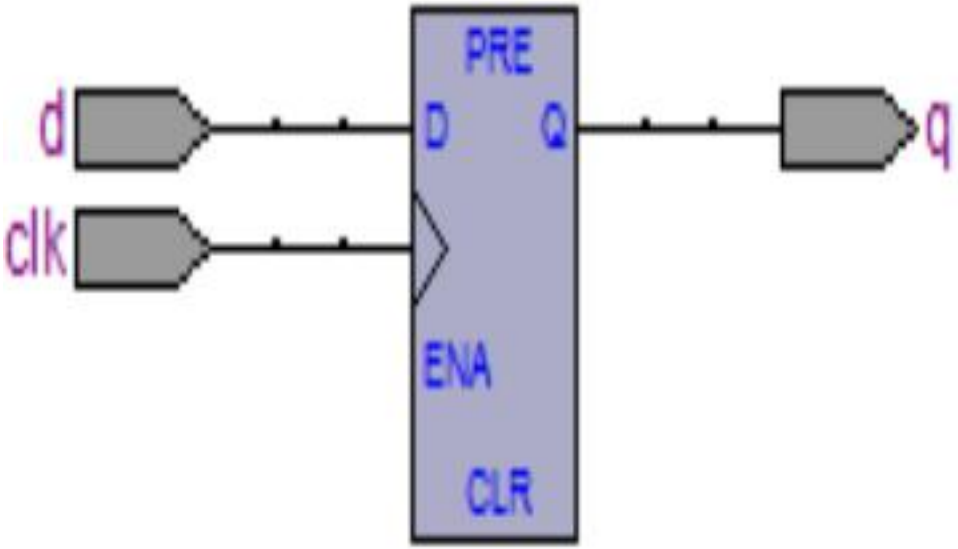




* if 语句的时钟沿触发

综合后生成寄存器（register）

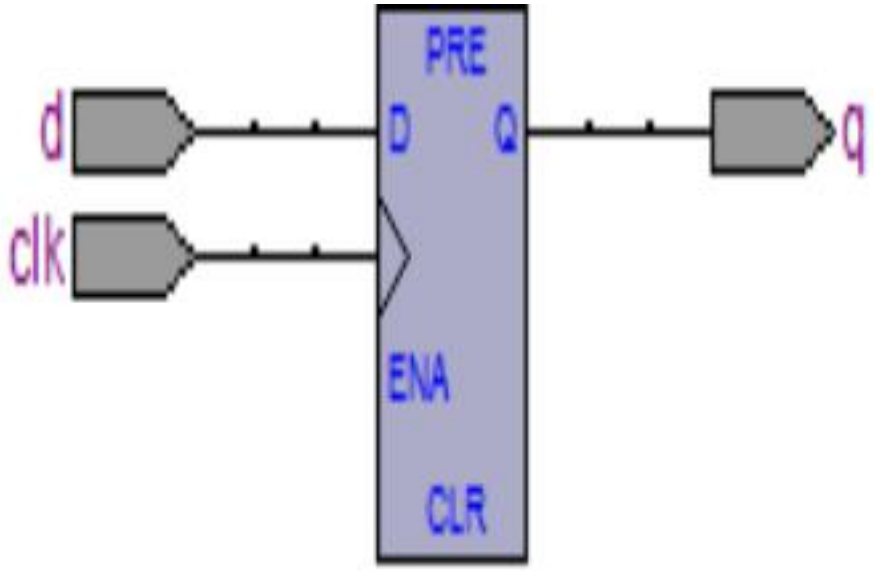
```
LIBRARY  IEEE;
USE IEEE.STD_LOGIC_1164.all;

ENTITY fd IS
      port ( clk, d: IN std_logic;
                   q: OUT std_logic );
END fd;

ARCHITECTURE rt1 OF fd  IS
BEGIN
    PROCESS (clk)
    BEGIN
       IF (clk'event and clk= '1') then
            q <= d;
       END  IF;
    END PROCESS;
END  rt1;
```



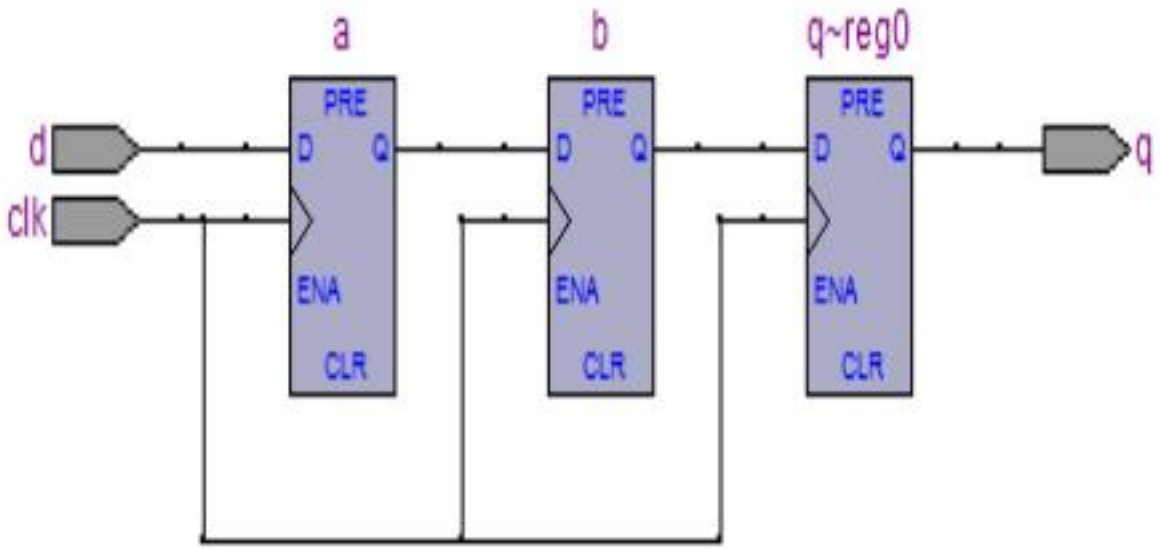
敏感信号表

时钟上升沿触发

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

 ENTITY dff IS
      PORT ( d : in std_logic ;
                clk : in std_logic ;
                  q : out std_logic ) ;
  END  dff;

  ARCHITECTURE  arc  OF  dff IS
  BEGIN
      PROCESS (clk)
            variable  a , b : std_logic ;
       BEGIN
          IF ( clk'event  AND clk='1' )  THEN
             a := d;
             b := a;
             q <= b;
         END IF;
    END PROCESS;
  END arc;
```



```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY reg3 IS
    PORT ( d : in std_logic ;
           clk : in std_logic ;
             q : out std_logic ) ;
END  reg3;

 ARTHITECTURE  arc  OF  reg3 IS
    SIGNAL  a , b : std_logic ;
 BEGIN
   PROCESS (clk)
     BEGIN
        IF ( clk'event  AND clk='1' ) THEN
           a <= d;
           b <= a;
           q <= b;
       END IF;
   END PROCESS;
 END arc;
```



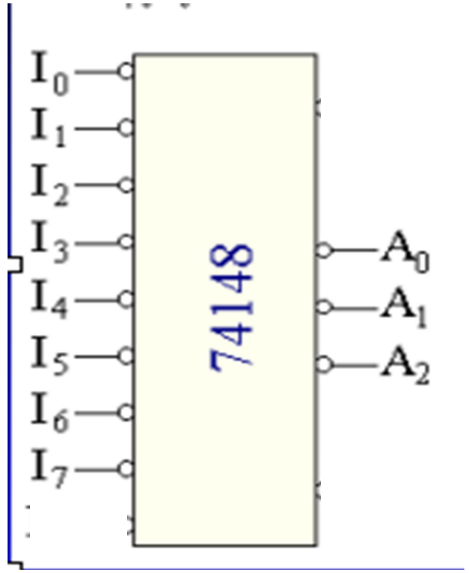
- if 多选择语句

语法格式

```
IF 条件表达式1 THEN
        顺序语句11；顺序语句12；
ELSIF 条件表达式2 THEN
        顺序语句21；顺序语句22；
ELSIF
        …

 ELSE
        顺序语句n1；顺序语句n2；
 END IF;
```

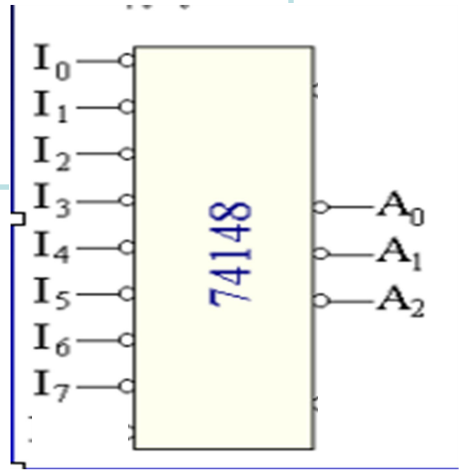
- if_then_elsif 语句中最先出现的条件优先级最高（自上而下优先）。
- 可以有多个elsif，但只能有一个else（组合逻辑）。

例：8－3优先编码器。




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

entity coder is
  port (
      input: in std_logic_vector(7 downto 0);
      output: out std_logic_vector(2 downto 0)
        );
end coder;
```

```
architecture art of coder is
begin
    process(input)
    begin
        if input(7)='0' then
            output<="000";
        elsif input(6)='0' then
            output<="001";
        elsif input(5)='0' then
            output<="010";
        elsif input(4)='0' then
            output<="011";
        elsif  input(3)='0' then
            output<="100";
        elsif  input(2)='0' then
            output<="101";
        elsif  input(1)='0' then
            output<="110"';
        else
            output<="111";
        end if;
    end process;
end art;
```



If 语句是顺序语句	When语句是并行语句
只能在进程中使用	必须用在进程之外
Else语句可有可无	Else语句是必须有的
可以嵌套	不能嵌套
高级描述，与硬件无关	与硬件电路十分贴近

组合逻辑一定有else，否则综合为锁存器

（2）CASE 语句

CASE语句根据某个表达式的值来选择执行体。 无优先级。

语法格式

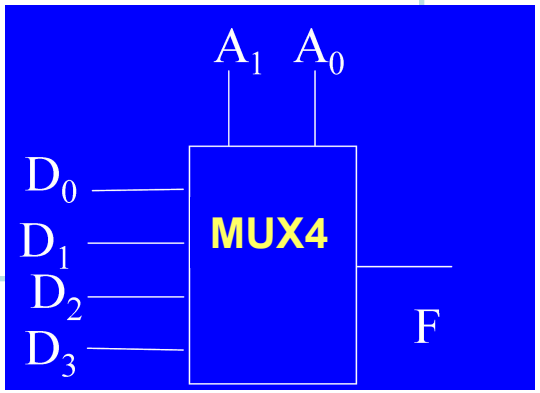
```
CASE  选择表达式  IS
    WHEN 分支值1 => 顺序处理语句11; 语句12;
    WHEN 分支值2 => 顺序处理语句21; 语句22;
    WHEN OTHERS => 顺序处理语句32; 语句32;
END CASE;
```

- 分支条件须在表达式范围内，且不能重合。
- 执行时必须选中且只能选中一个分支。
- 所有值必须列举穷尽，对sted_logic等必须用others。

例： 4选1多路选择器描述方式

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

ENTITY test_case IS
PORT (
   a1, a0: IN STD_LOGIC;
  d0, d1, d2, d3:  IN STD_LOGIC;
    f: OUT STD_LOGIC
    ) ;
END test_case;
```

```
ARCHITECTURE behave OF test_case IS
 SIGNAL s: STD_LOGIC_VECTOR (1 DOWNTO 0) ;
BEGIN
   PROCESS (a1, a0, d0, d1, d2, d3)
   BEGIN
        s<=a1 & a0;
        CASE s IS
            WHEN “00” => f<=d0;
            WHEN “01” => f<=d1;
            WHEN “10” => f<=d2;
            WHEN others => f<=d3;
        END CASE;
      END PROCESS;
  END behave;
```



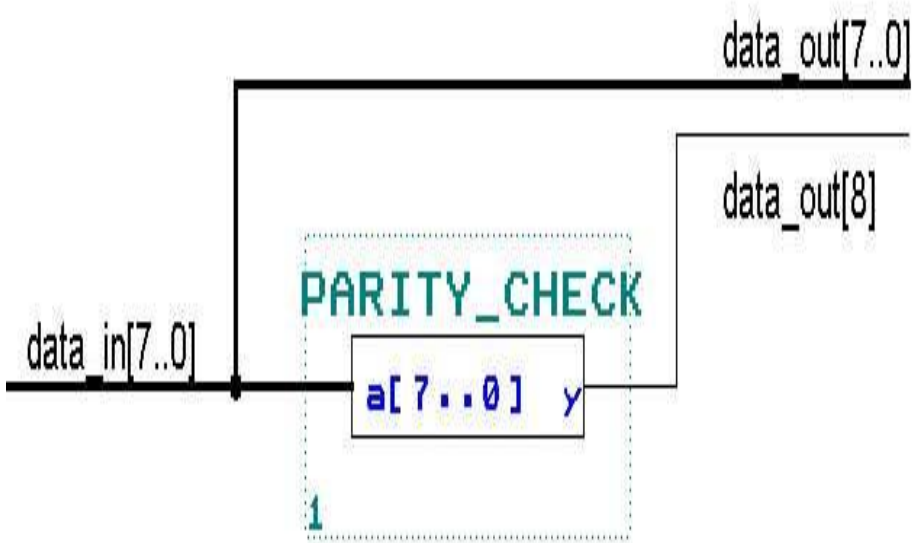
3、循环语句

语法格式

```
FOR 循环变量 IN 循环范围 LOOP
    顺序处理语句1；
    顺序处理语句2;
    …  ..  ..  …
END LOOP；
```

- 循环变量是 loop 内部自动声明的局部量，仅在 loop 内可用;
- 循环范围为可计算的整数范围:　　*m*（小）　*to*　*n*（大）

【例】　　--8位奇偶校验位产生电路



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


ENTITY p_check IS
PORT (
      d: IN STD_LOGIC_VECTOR
(7 DOWNTO 0);
      y: OUT STD_LOGIC);
END p_check;


ARCHITECTURE behave OF
p_check IS
```

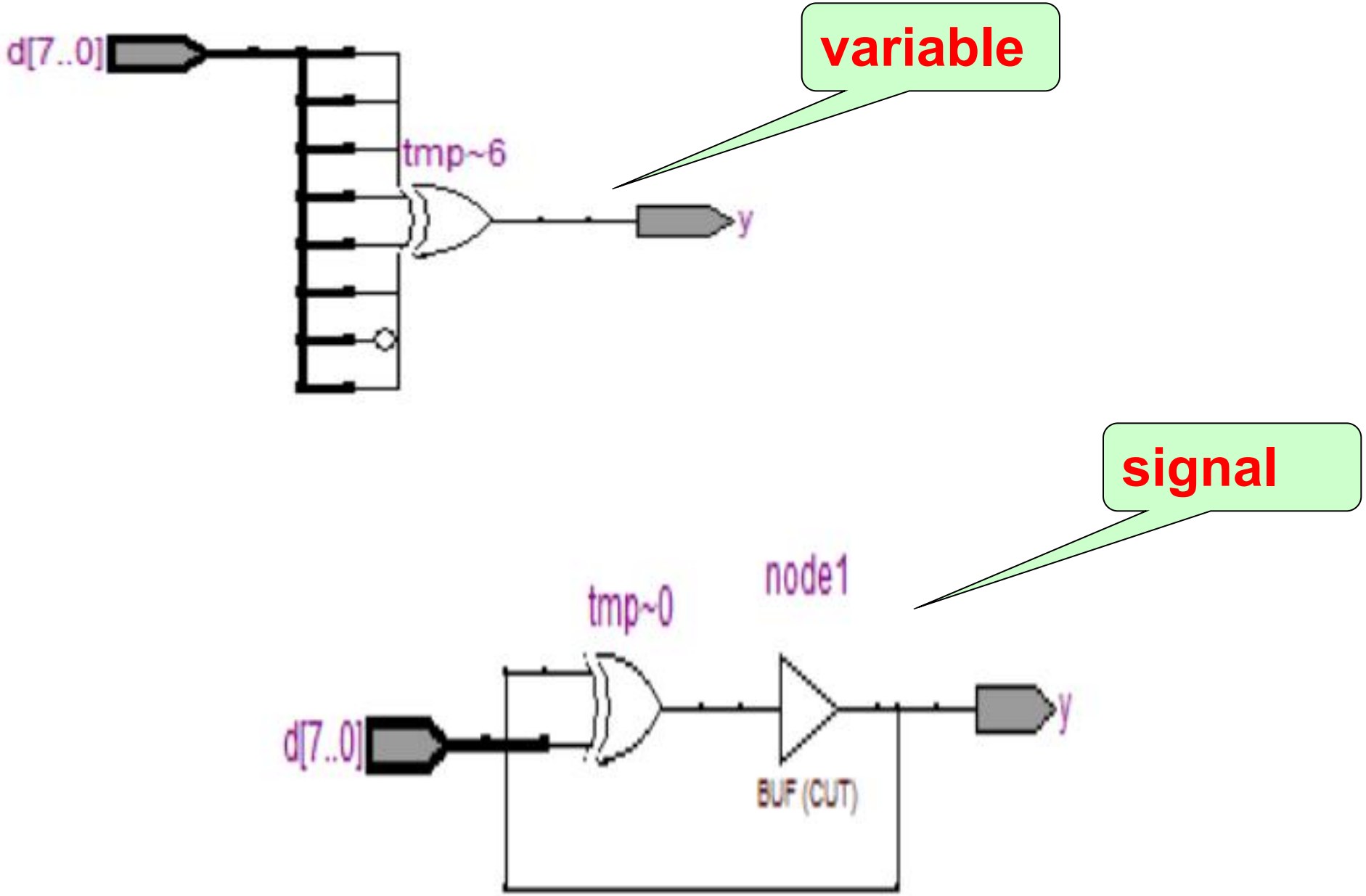
```
BEGIN
   PROCESS (d)
    VARIABLE  tmp: STD_LOGIC;
    BEGIN
          tmp := '1';
          FOR n IN 0 TO 7 LOOP
                    --FOR循环语句
               tmp := tmp XOR d(n);
          END LOOP;
          y <= tmp;
   END PROCESS;
END behave;
```

signal?

奇偶？

只做校验？

信号多次赋值，只最后一次赋值有效


d[7..0]
tmp~6
y
variable
signal
tmp~0
node1
d[7..0]
y
BUF (CUT)