第三章 VHDL程序语句

* 1. VHDL基本语句

1. 简单赋值语句

目标信号名<=表达式

1. 选择信号赋值语句

WITH 选择表达式 SELECT

赋值目标信号<=表达式1 WHEN 选择值1，

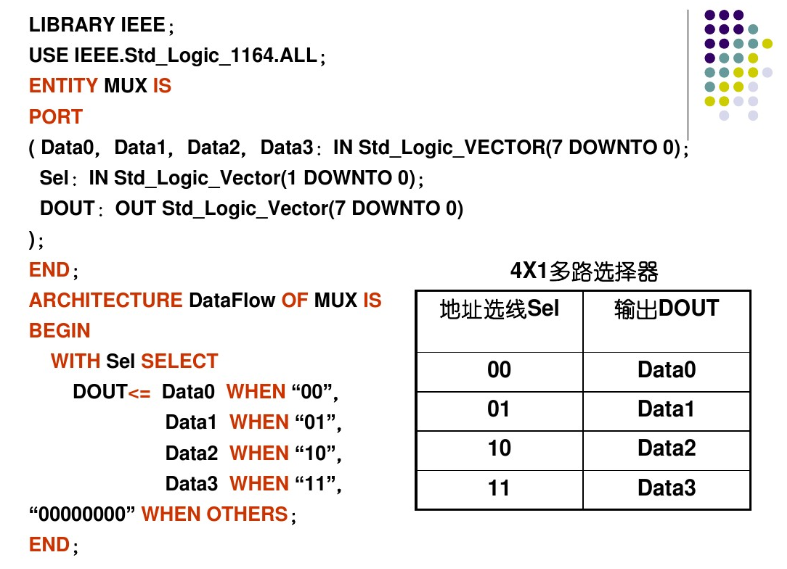
表达式2 WHEN 选择值2，

表达式3 WHEN 选择值3，

......

表达式n WHEN OTHERS;

选择值要覆盖所有可能情况，若不能一一指定，用OTHERS为其他情况找个出口；

选择值必须互斥，不能出现条件重复或重叠的情况。

1. 条件信号赋值语句

赋值目标信号 <= 表达式1 WHEN 赋值条件1 ELSE

表达式2 WHEN 赋值条件2 ELSE

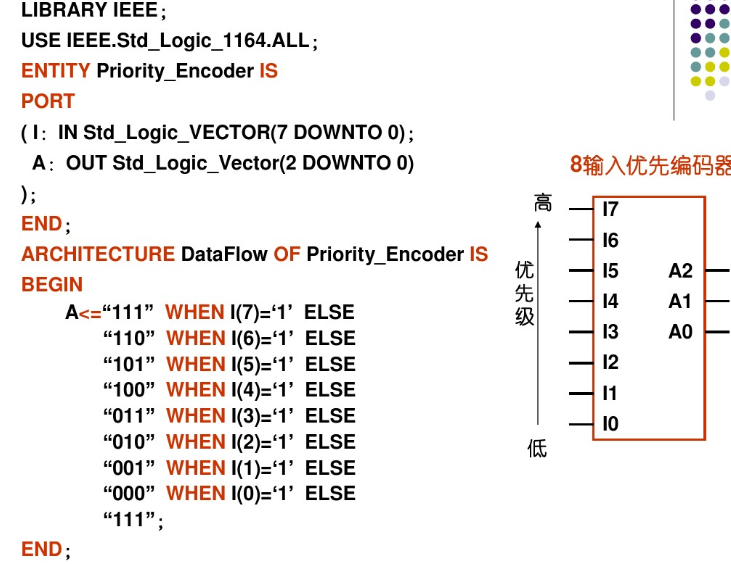
......

表达式n WHEN 赋值条件n ELSE

表达式；

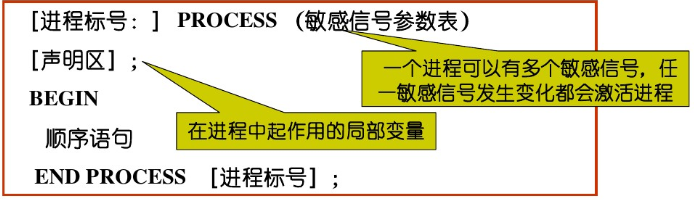
各赋值语句有优先级的差别，按书写顺序从高到低排列；

各赋值条件可以重叠。

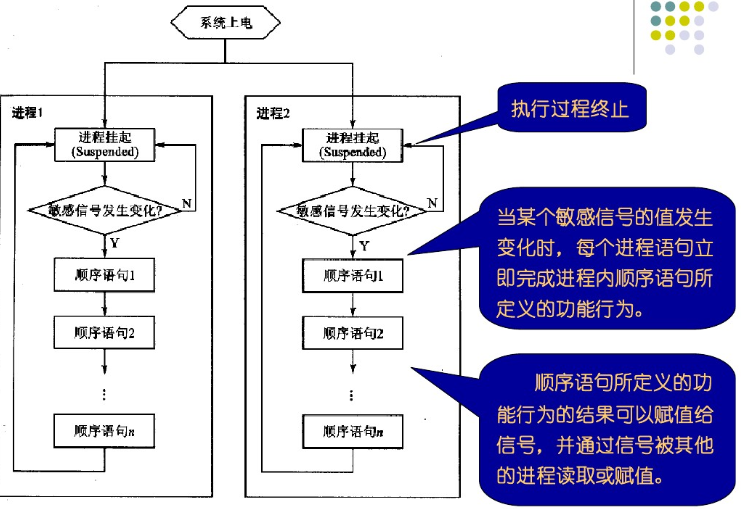


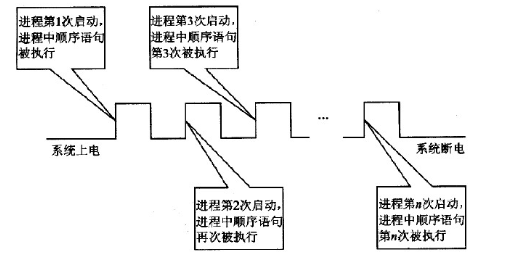
1. 进程语句

进程语句定义顺序语句模块，用于将从外部获得的信号值，或内部的运算数据向其他的信号进行赋值。

* 1. 进程本身是并行语句，但其内部是顺序语句；
  2. 进程只有在特定的时刻（敏感信号发生变化）才能被激活。

进程的工作原理：

进程与时钟

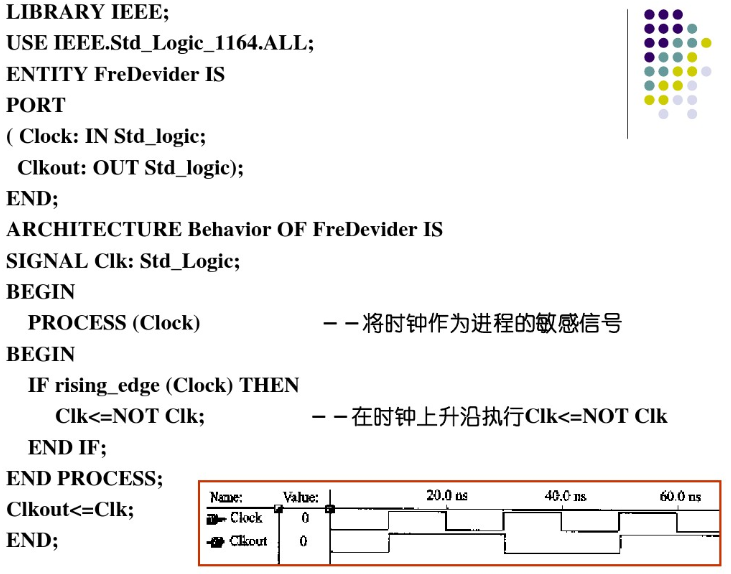
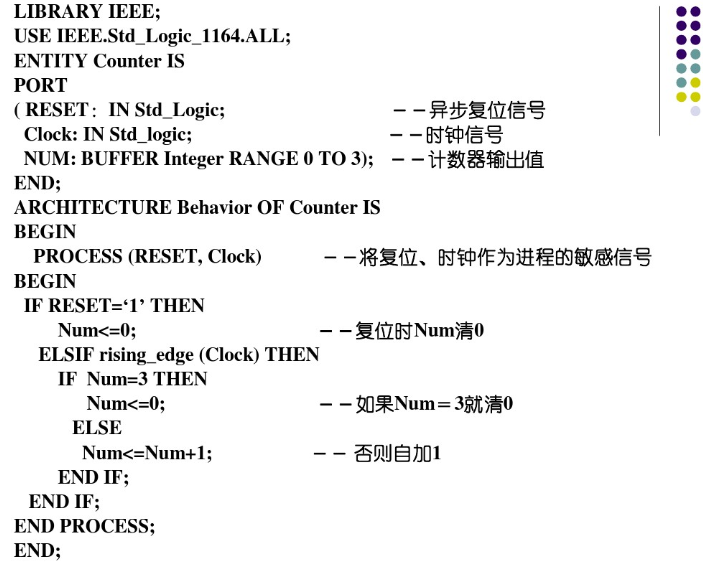
在每个上升沿启动一次进程（执行进程内所有的语句）

上升沿描述：Clock’ EVENT AND Clock=’1’

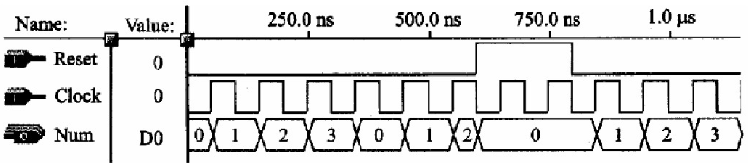
下降沿描述：Clock’ EVENT AND Clock=’0’

上升沿描述：rising\_edge(Clock)

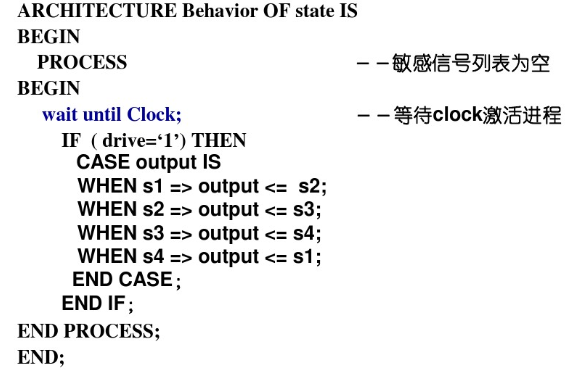
下降沿描述：falling\_edge(Clock)



仿真波形：



1. 进程的启动

当process的敏感信号参数表没有列出任何敏感信号时，进程通过wait语句启动。

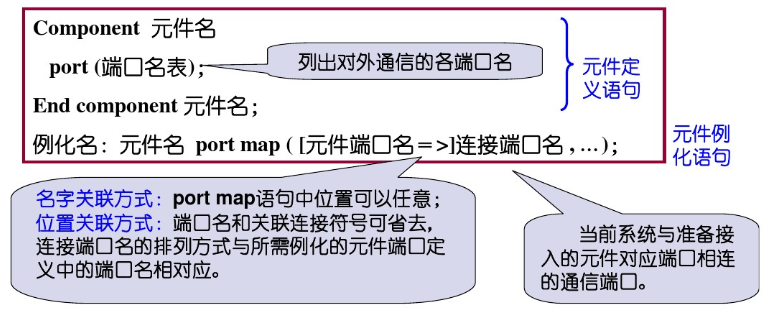
进程注意事项：  
在同一进程中对同一信号多次赋值，只有最后一次生效；

在不同进程中，不可对同一信号进行赋值；

一个进程不可同时对时钟上、下沿敏感；

* 1. 元件例化语句

元件例化引入一种连接关系，将预先设计好的实体定义为元件，并将此元件与当前设计实体中的端口相连接，从而为当前设计实体引入一种新的低一级的设计层次。

元件例化：LIBRARY IEEE;

USE IEEE.Std\_Logic\_1164.ALL;

ENTITY ord41 IS

PORT(a1,b1,c1,d1: IN Std\_Logic;

Z1: out std\_logic);

END;

ARCHITECTURE ord41behv OF ord41 IS

BEGIN

COMPONENT nd2

PORT(a,b: IN Std\_Logic;

c:out std\_logic);

END COMPONENT;

SIGNAL x,y:Std\_Logic;

BEGIN

u1:nd2 PORT MAP(a1, b1, x);

u2:nd2 PORT MAP(a=>c1, c=>y, b=>d1);

u3:nd2 PORT MAP(x, y, c=>z1);

END;

设计2输入与非门：

LIBRARY IEEE;

USE IEEE.Std\_Logic\_1164.ALL;

ENTITY nd2 IS

PORT

(a, b: IN Std\_Logic;

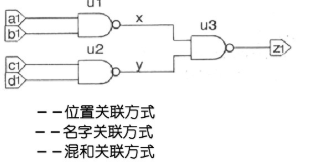
c:out std\_logic);

END;

ARCHITECTURE nd2behv OF nd2 IS

BEGIN

y<=a NAND b;

 END;

* 1. 顺序语句

顺序语句仅出现在进程和子程序中。

顺序语句综合后，映射为实际的门电路，系统上电，门电路开始工作。电路可实现逻辑上的顺序执行，实际上所有门电路是并行工作的。

流程控制语句  
 IF语句：IF 条件式 THEN

顺序语句

END IF;

IF 条件式 THEN

顺序语句

ELSE

顺序语句

END IF;

IF 条件式 THEN

顺序语句

ELSEIF 条件式2

THEN

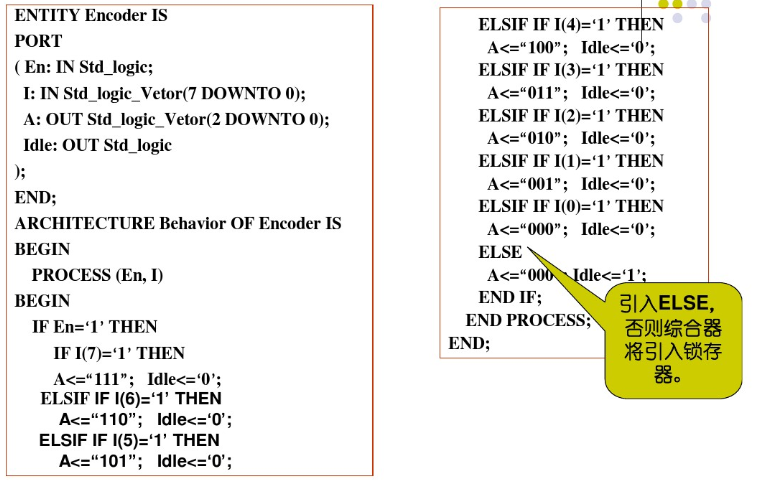
顺序语句

...

ELSE

顺序语句

END IF;

用IF语句描述组合逻辑电路时，务必涵盖所有的情况，否则综合后将引入锁存器。

不完整的条件语句与时序电路

ENTITY COMP\_BAD IS

PORT(a1: IN BIT;

b1: IN BIT;

q1: OUT BIT);

END;

ARCHITECTURE one OF COMP\_BAD IS

BEGIN

PROCESS(a1, b1)

BEGIN

IF a1>b1 THEN q1 <= ‘1’;

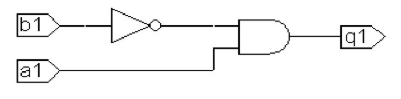
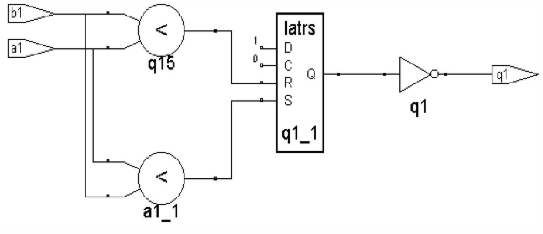
ELSIF a1<b1 THEN q1 <= ‘0’; --未提及当a1=b1时，q1作何操作

END IF;

END PROCESS;

END;

综合结果：

完整条件语句的综合结果为：  
 CASE语句

CASE 表达式 IS

WHEN 选择值[|选择值] => 顺序语句；

WHEN 选择值[|选择值] => 顺序语句；

......

WHEN OTHERS => 顺序语句；

END CASE;

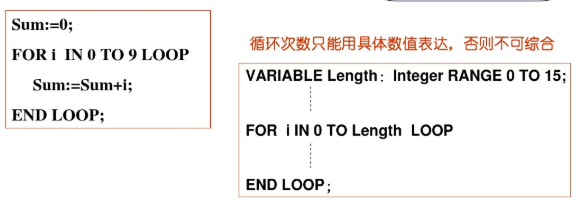
选择值不可重复；

当CASE语句的选择值无法覆盖所有的情况时，要用OTHERS指定未能列出的其他所有可能情况的输出值；

LOOP语句

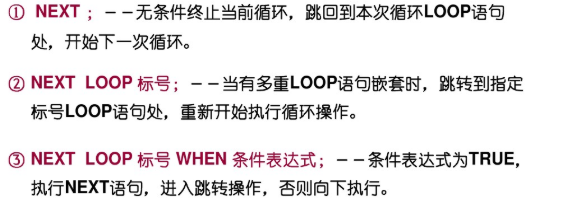
[LOOP标号:]FOR 循环变量 IN 循环次数范围 LOOP

顺序语句

 END LOOP[LOOP标号];

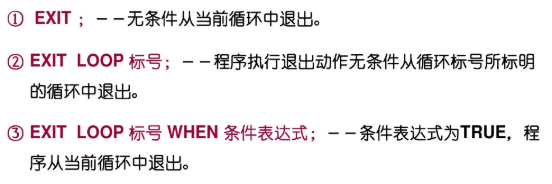
NEXT语句

NEXT语句主要用在LOOP语句执行中有条件或无条件转向控制，跳向LOOP语句的起点。

NEXT[循环标号] [WHEN 条件];

EXIT语句

EXIT语句主要用在LOOP语句执行中有条件或无条件内部转向控制，跳向LOOP语句的终点，用于退出循环。当程序需要处理保护、出错和警告状态时，语句能提供一个快捷、简便的方法。

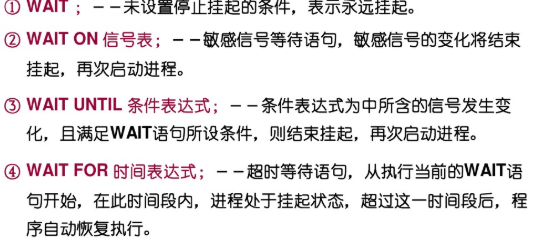
EXIT[循环标号] [WHEN条件];

NULL语句

NULL为空操作语句，一般用于CASE中，表示在某些情况下对输出不做任何改变，隐含锁存信号。不能用于纯组合逻辑电路。

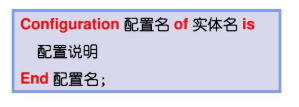
WAIT语句

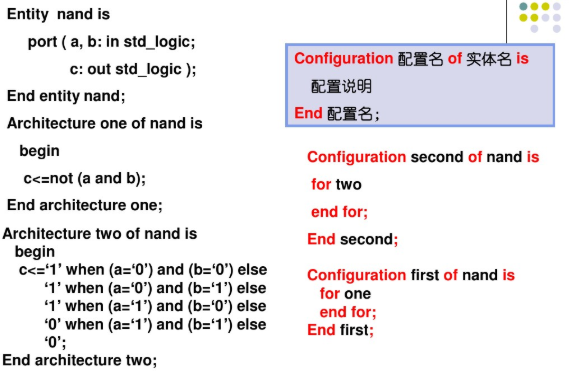
在进程或过程中执行到WAIT语句时，程序将被挂起，并设置好再次执行的条件。

 WAIT [ON信号表][UNTIL条件表达式][FOR时间表达式];

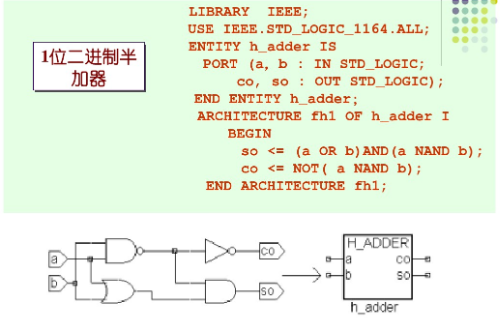
* 1. 配置语句

配置主要为顶层设计的实体指定结构体，或为参与例化的元件实体指定所希望的结构体，以层次方式来对元件例化做结构配置。





例：1位二进制半加器的设计



三态门：

LIBRARY IEEE;

USE IEEE.Std\_Logic\_1164.ALL;

ENTITY buffer IS

PORT(OE: IN std\_logic;

DATAIN: IN std\_logic;

DATAOUT: IN std\_logic);

END buffer;

ARCHITECTURE Behavioral OF buffer IS

BEGIN

PROCESS(OE, DATAIN);

IF OE=’0’ THEN

DATAOUT<=’Z’;

ELSE DATAOUT<=DATAIN;

END IF;

END PROCESS;

END Behavioral;

移位寄存器：  
