VHDL语言基础（一）

中国传媒大学信息与通信工程学院

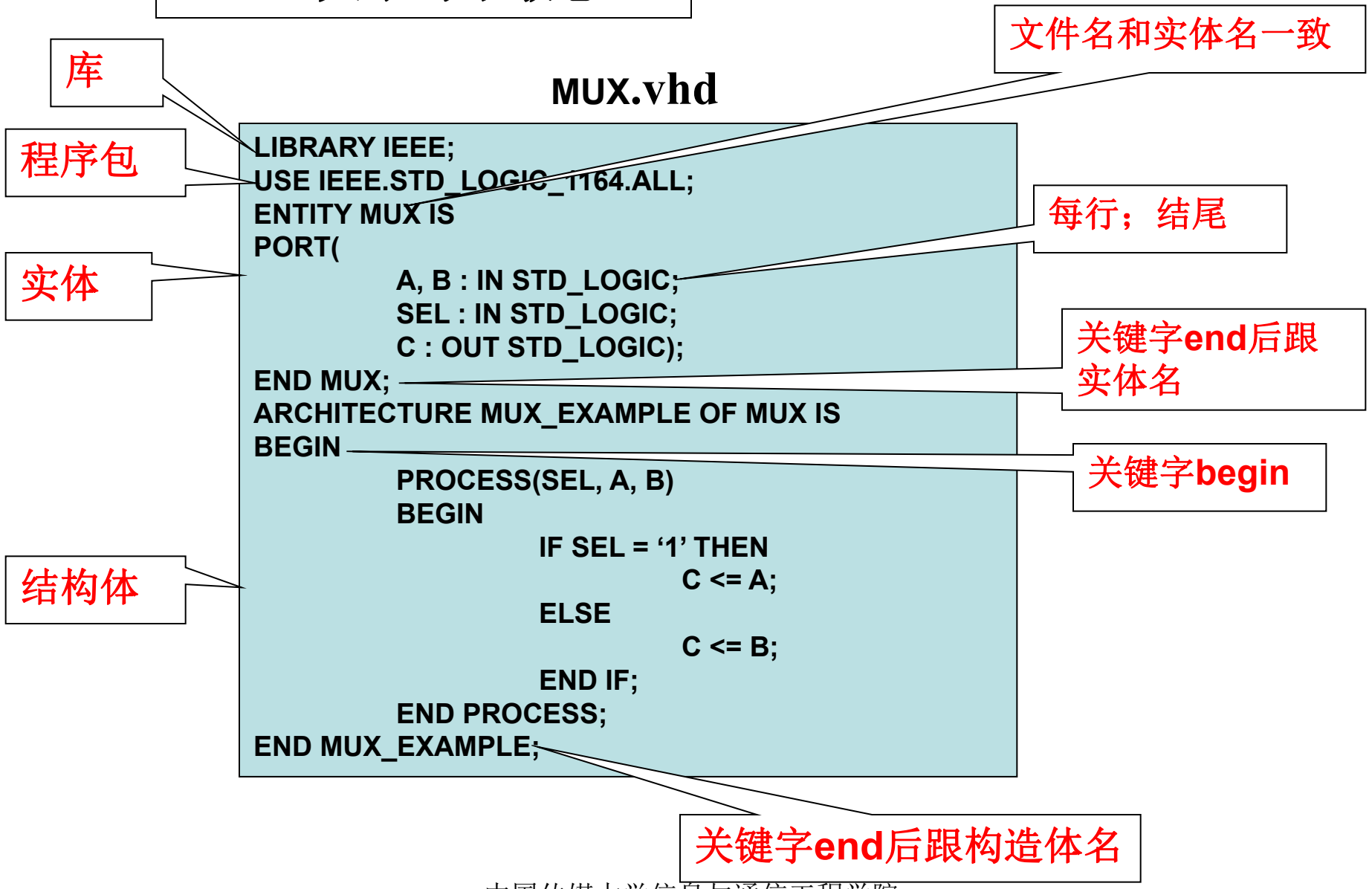
VHDL语言基础

- 简介
- **VHDL**程序基本结构
- 数据对象、数据类型、运算符和属性
- 并行语句
- 进程
- 顺序语句

VHDL程序的基本结构

- VHDL把任意复杂度的电路模块看作一个单元
  - 一个单元又可以分为接口部分和设计描述部分
- 一个VHDL程序包括五个部分
  - 实体（Entity）
    - 描述设计的外部接口信号和该设计单元的公共信息
  - 结构体（Architecture）
    - 描述该设计单元的行为、数据的流程或结构
  - 库（Library）
    - 存放已编译的实体、结构体、程序包等，用户可以直接引用
  - 程序包（Package）
    - 存放设计可以共享的数据类型、常数和子程序等
  - 配置（Configuration）

VHDL 大小写不敏感




```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX IS
PORT(
        A, B : IN STD_LOGIC;
        SEL : IN STD_LOGIC;
        C : OUT STD_LOGIC);
END MUX;
ARCHITECTURE MUX_EXAMPLE OF MUX IS
BEGIN
        PROCESS(SEL, A, B)
        BEGIN
                IF SEL = '1' THEN
                        C <= A;
                ELSE
                        C <= B;
                END IF;
        END PROCESS;
END MUX_EXAMPLE;
```

实体声明

```
ENTITY 实体名 IS
PORT（
        端口名1：模式 数据类型；
        端口名2：模式 数据类型；
        ……
        端口名n：模式 数据类型）；
END 实体名；
```

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX IS
PORT(
        A, B : IN STD_LOGIC;
        SEL : IN STD_LOGIC;
        C : OUT STD_LOGIC);
END MUX;
ARCHITECTURE MUX_EXAMPLE OF MUX IS
BEGIN
        PROCESS(SEL, A, B)
        BEGIN
                IF SEL = '1' THEN
                        C <= A;
                ELSE
                        C <= B;
                END IF;
        END PROCESS;
END MUX_EXAMPLE;
```

端口说明

- 每一个I/O信号都被称为端口，其功能对应于电路图符号的一个引脚
- 每个端口必须有
  - 一个名字
  - 通信模式
    - 说明数据通过该端口的流动方向
    - **IN模式：输入模式，只允许信号流入实体**
    - **OUT模式：输出模式，只允许信号流出实体**
    - **INOUT模式：既可以流入，也可以流出**
    - **BUFFER模式：缓冲模式，和输出模式的端口类似**
  - 数据类型
    - 说明流过该端口的数据类型
    - bit, bit_vector, std_logic, std_logic_vector, integer

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX IS
PORT(
        A, B : IN STD_LOGIC;
        SEL : IN STD_LOGIC;
        C : OUT STD_LOGIC);
END MUX;
ARCHITECTURE MUX_EXAMPLE OF MUX IS
BEGIN
        PROCESS(SEL, A, B)
        BEGIN
                IF SEL = '1' THEN
                        C <= A;
                ELSE
                        C <= B;
                END IF;
        END PROCESS;
END MUX_EXAMPLE;
```

结构体

- 描述实体的内在
  - 描述一个实体的功能
  - 规定实体的数据流程
  - 定义实体中内部单元的连接关系
- 结构体的一般格式

```
ARCHITECTURE 结构体名 OF 实体名 IS
        [说明部分]
BEGIN
        并行语句1;
        并行语句2;
        ……
        并行语句3
END 结构体名;
```

对结构体内部所使用的信号、常数、数据类型、元件和子程序等进行声明和定义

用于具体地描述结构体的行为及其连接关系

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX IS
PORT(
        A, B : IN STD_LOGIC;
        SEL : IN STD_LOGIC;
        C : OUT STD_LOGIC);
END MUX;
ARCHITECTURE MUX_EXAMPLE OF MUX IS
BEGIN
        PROCESS(SEL, A, B)
        BEGIN
                IF SEL = '1' THEN
                        C <= A;
                ELSE
                        C <= B;
                END IF;
        END PROCESS;
END MUX_EXAMPLE;
```

标识符

- **VHDL语言中符号书写的基本规则**
- **VHDL’87版标识符的语法规则**
  - 短标识符

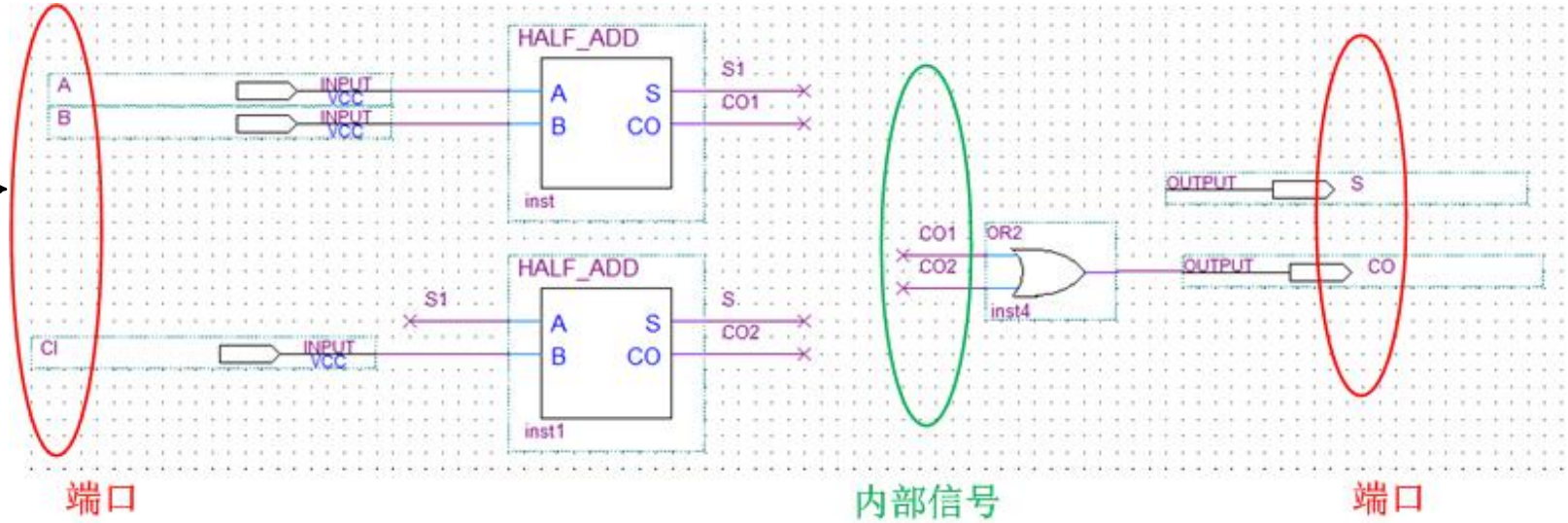
- 有效字符为：英文字母（’a～z’、’A～Z’），数字（’0～9’）和下划线（’_’）
- 第一个字符必须是字母。
- 下划线前后都必须有英文字母或数字。
- 最后一个字符不能是下划线。
- 不允许连续两个下划线。
- VHDL的保留字不能用于标识符。
- 在标识符中大写字母和小写字母是等效的

_txclk -- 标识符必须起始于字母　　8B10B -- 标识符必须起始于字母
large#number -- 只能是数字、字母和下划线
link__bar -- 不能有两个连续的下划线
rx_clk_ -- 最后字符不能是下划线

数据对象

- 凡是可以赋予一个值的客体就称为对象
- 数据对象
  - 常数
  - 信号
  - 变量
  - 文件
- 数据对象在使用前必须给予声明

- 是电子电路内部硬件连接的抽象，代表连线
- 在结构体的说明部分声明
- 可以是逻辑门的输入或输出
- 也能表达存储元件的状态



SIGNAL CO1, CO2：STD_LOGIC;

数据类型

- **VHDL**语言中的对象：信号、变量和常数都要指定数据类型
- 数据类型
  - 标准的数据类型
  - 用户自定义数据类型
- 数据类型的定义严格
  - 不同类型之间的数据不能直接赋值
  - 数据类型相同，位长不同也不能直接赋值

标准的数据类型

1. 整数（INTEGER）

- 与数学中的整数的定义相同
  - 表示范围为－2147483647～2147483647
- 任何带有小数点的数字都被认为是实数
- 整数不能看作是位矢量
  - 不能按位来进行访问
  - 对整数不能用逻辑操作符

SIGNAL COUNT：INTEGER；

2. 位（BIT）

- 位值的表示方法是用‘0’或‘1’表示
- 可以用来描述数字系统中总线的值

SIGNAL CO：BIT；

3.位矢量（BIT_VECTOR）

- 是用双引号括起来的一组数据
  - “001100”
- 用位矢量数据表示总线状态最形象也最方便

SIGNAL SUM：BIT_VECTOR(3 DOWNTO 0)；

IEEE标准的“STD_LOGIC”和“STD_LOGIC_VECTOR”类型

使用时必须写出下列
库说明语句和使用程序包说明语句

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
```

‘U’, —— 未初始化（Uninitialized）
‘X’, —— 未知（Forcing Unknown）
‘0’, —— 强0（Forcing 0）
‘1’, —— 强1（Forcing 1）
‘Z’, —— 高阻（High Impedance）
‘W’, —— 弱未知（Weak Unknown）
‘L’, —— 弱0(Weak 0)
‘H’, —— 弱1（Weak 1）
‘-’, —— 不可能情况（Don’t care）

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX IS
PORT(
        A, B : IN STD_LOGIC;
        SEL : IN STD_LOGIC;
        C : OUT STD_LOGIC);
END MUX;
ARCHITECTURE MUX_EXAMPLE OF MUX IS
BEGIN
        PROCESS(SEL, A, B)
        BEGIN
                IF SEL = ‘1’ THEN
                        C <= A;
                ELSE
                        C <= B;
                END IF;
        END PROCESS;
END MUX_EXAMPLE;
```

运算操作符

- 运算操作符
  - 逻辑运算
  - 关系运算
  - 算术运算
  - 并置运算
- 操作数的类型应该和操作符所要求的类型相一致
- 运算操作符是有优先级

逻辑运算符

NOT ——取反；
AND ——与；
OR ——或；
NAND ——与非；
NOR ——或非；
XOR ——异或；

```
S <= a xor b;

Co <= a and b;
```

- 可以对“STD_LOGIC”、“BIT”和“STD_LOGIC_VECTOR”型数据进行逻辑运算
- NOT的优先级最高

算术运算符

+ ——加；

－ ——减；

* ——乘；

/ ——除；

MOD ——取模；

REM ——取余；

＋ ——正；

－ ——负；

** ——指数；

ABS ——取绝对值；

关系运算符

= ——等于；
/= ——不等于；
< ——小于；
<= ——小于等于；
> —— 大于；
>= ——大于等于；

- 关系运算符的左右两边是运算操作数
- 左右两边操作数的类型必须相同，但位长度不一定相同
- 等号“=”和不等号“/=”可以适用于所有的数据类型

并置运算符

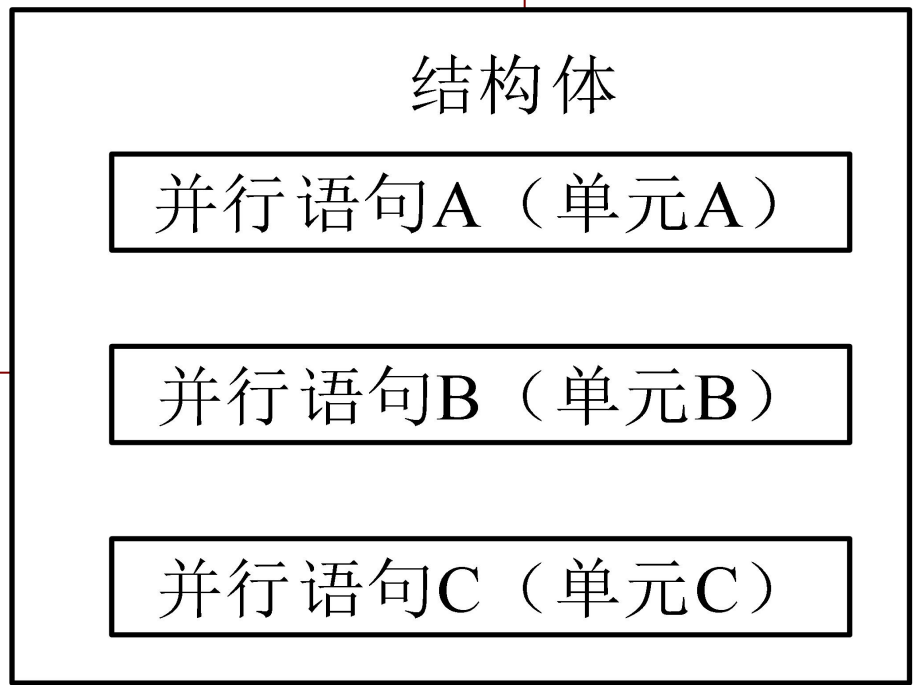
- 并置运算符“&”用于位的连接，形成位矢量
- 可以把两个位矢量连接起来形成更大的位矢量

```
SIGNAL DATA_A : STD_LOGIC_VECTOR(3 DOWNTO 0);
        .....
DATA_C <= D1 & D2 & D3 & D4;
DATA_D <= DATA_C & DATA_A;
```

VHDL语言的基本语法

- VHDL语言提供了一系列顺序语句和并行语句
  - 顺序语句的用来实现模型的算法部分
  - 并行语句用来表示黑盒子的连接关系
- 结构体是由一系列的并行语句构成
  - 每个并行语句表示一个功能单元
  - 多个功能单元构成一个结构体

```
ARCHITECTURE 结构体名 OF 实体名 IS
        ——说明部分
BEGIN
        ——并行语句A
        ——并行语句B
        ——并行语句C
END 结构体名；
```



并行语句

1.并行信号赋值语句

普通并行信号赋值语句

条件信号赋值语句

选择信号赋值语句

普通并行信号赋值语句

信号量 <= 信号量表达式;

实例一：半加器

输入端口：加数 A，B；求和位：SUM；进位输出：CO

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY HALF_ADDER IS
PORT(A, B : IN STD_LOGIC;
     SUM, CO : OUT STD_LOGIC);
END HALF_ADDER;
ARCHITECTURE RTL OF HALF_ADDER IS
BEGIN
        SUM <= A XOR B;
        CO <= A AND B;
END RTL;
```

实例二：译码器

输入端口：输入 A[1..0]；译码输出：Y[3..0]

2-4译码器真值表

高电平有效/低电平有效

A1	A0	Y3	Y2	Y1	Y0
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

逻辑表达式： y0=a0 or a1 ;

y1=not(a0) or a1; y2=a0 or not(a1) ;

y3=not(a0) or not(a1)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY DECODER IS
PORT(A : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
     Y: OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END DECODER;
ARCHITECTURE RTL OF DECODER IS
BEGIN
        Y(0) <= A(0) OR A(1);
        Y(1) <= NOT(A(0)) OR A(1);
        Y(2) <= A(0) OR NOT(A(1));
        Y(3) <= NOT(A(0)) OR NOT(A(1));
END RTL;
```

并行语句

1.并行信号赋值语句

条件信号赋值语句

```
目的信号量 <= 表达式1 WHEN 条件1 ELSE
              表达式2 WHEN 条件2 ELSE
              表达式3 WHEN 条件3 ELSE
                    ……
              ELSE 表达式n;
```

SEL(1)	SEL(0)	Y
0	0	A
0	1	B
1	0	C
1	1	D

实例三：多路选择器

输入端口：输入 A，B，C，D；选择控制：SEL；选择输出：Y

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX IS
PORT(A, B, C, D : IN STD_LOGIC;
     SEL : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
     Y: OUT STD_LOGIC);
END MUX;
ARCHITECTURE RTL OF MUX IS
BEGIN
     Y <= A WHEN SEL = "00" ELSE
          B WHEN SEL = "01" ELSE
          C WHEN SEL = "10" ELSE
          D;
END RTL;
```

实例四：编码器

输入端口：4路输入I[3..0]；编码输出：Y[1..0]

真值表

I3	I2	I1	I0	Y1	Y0
0	1	1	1	1	1
X	0	1	1	1	0
X	X	0	1	0	1
X	X	X	0	0	0

- 理解优先级
- 合理选择描述语句

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY ENCODER IS
PORT(I : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
     Y: OUT STD_LOGIC_VECTOR(1 DOWNTO 0));
END ENCODER;
ARCHITECTURE RTL OF ENCODER IS
BEGIN
    Y <= "00" WHEN I(0) = '0' ELSE
         "01" WHEN I(1) = '0' ELSE
         "10" WHEN I(3) = "0' ELSE
         "11";
END RTL;
```

并行语句

1.并行信号赋值语句

选择信号赋值语句

```
WITH  表达式  SELECT
  目的信号量 <= 表达式1 WHEN 条件1，
                表达式2 WHEN 条件2，
                 ……
                表达式n WHEN 条件n;
```

A1	A0	Y3	Y2	Y1	Y0
0	0	1	1	1	0
0	1	1	1	0	1
1	0	1	0	1	1
1	1	0	1	1	1

实例一：译码器

输入端口：输入A[1..0]；译码输出：Y[3..0]

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY DECODER IS
PORT(A : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
     Y : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END DECODER;
ARCHITECTURE RTL OF DECODER IS
BEGIN
    WITH A SELECT
        Y <= "1110" WHEN "00",
             "1101" WHEN "01",
             "1011" WHEN "10",
             "0111" WHEN "11";
END RTL;
```

实例六：

输入端口：输入I[3..0]；输出：Y

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MY_DESIGN IS
PORT(I : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
     SEL : IN STD_LOGIC;
     Y: OUT STD_LOGIC);
END MY_DESIGN;
ARCHITECTURE RTL OF MY_DESIGN IS
    SIGNAL M0, M1 : STD_LOGIC;
BEGIN
    M0 <= I(0) AND I(1);
    M1 <= I(0) XOR I(1);
    Y <= M0 WHEN SEL = '0' ELSE M1;
END RTL;
```

并行语句

- 进程语句（PROCESS）
- 信号赋值语句（SIGNAL ASSIGNMENT）
- 块语句（BLOCK）
- 元件调用语句（COMPONENT）
- 端口映射语句（PORT MAP）
- 生成语句（GENERATE）
- 过程调用语句（PROCEDURE CALL）