1.4 EDA技术的基础知识

1. 4. 1 VHDL语言基础

VHDL — 超高速硬件描述语言
（Very High Speed Hardware Description Language）

一、VHDL的主要构件

基本设计单元

- 实体（Entity ）--描述设计单元的外部接口信号
- 结构体(Architecture) --描述设计单元内部结构和行为
- 程序包（Package ） -- 存放各设计模块共享的数据类型、常数、子程序等
- 库（Library ） -- 专门存放程序包
- 配置（Configration ）--指定实体所要配置的结构体

1. 实体

功能：实现设计单元的端口说明。

语法结构：

用英文字母赋予每个引脚的名称

```
ENTITY 实体名 IS
    PORT（端口名{，端口名}：端口模式 数据类型；
          端口名{，端口名}：端口模式 数据类型）；
END 实体名；
```

定义引脚上数据传输的方向

常用端口模式

方 向	说 明
IN	输入到实体
OUT	从实体输出
INOUT	双向数据传输
BUHHER	从实体输出（但可反馈到实体内部）

[例1.4.1] 2 输入与门的实体说明。

```
ENTITY and 2  IS
   PORT（a，b：IN   STD_LOGIC；
                y：OUT  STD_LOGIC ）；
END   and2；
```

设计单元端口结果：

2. 结构体

功能：描述设计单元内

语法结构：

```
ARCHITECTURE 结
[结构体说明语句]
BEGIN
[功能描述语句]
END 结构体名;
```

描述结构体内部“功能描述语句”中要用到的的内部信号、常数、数据类型、函数。（无时可省略）

用并行语句形式描述设计单元功能

并行语句类型
- 进程语句（PROCESS）
- 块描述语句（BLOCK）
- 信号赋值语句
- 子程序调用语句
- 元件例化语句

```
ARCHITECTURE 结构体名 OF 实体名 IS
[结构体说明语句]
BEGIN
[功能描述语句]
END 结构体名;
```

[例1.4.2] 2 输入与门的结构体描述。

```
ARCHITECTURE one OF and2 IS
  BEGIN
    y <= a and b
END ARCHITECTUR
```

为并行语句，执行顺序与其书写顺序无关，在实际电路中所有并行语句功能同时实现。

3. 库、程序包和配置

（1） 库

功能： 存储和放置设计单元（元件、程序包等）。

VHDL库
- IEEE库
- STD库
- ASIC库
- WORK库
- 用户自定义

常用资源库

VHDL标准库

资源库

当前作业库
存放

资源库
存放：用户自己创建定义的非

库说明语句格式：

LIBRARY 库名; 如： **LIBRARY IEEE;**

（2） 程序包

功能：存放各设计模块共享的数据类型、常数、子程序等。

语法格式：

```
USE  LIBRARY  库名.程序包名.项目名;
```

[例] 对IEEE库的1164程序包中所有项目的说明。

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

（3） 配置

功能： --为实体指定所要配置的结构体

语法格式：

```
CONFIGURATION 配置名 OF 实体名 IS
    FOR 被选结构体名
    END FOR ;
END 配置名;
```

[例1.4.3] 配置语句举例

```
ENTITY equ2  IS
    PORT(a,b:IN  STD_LOGIC_VECTOR(1 DOWNTO 0);
              equ:OUT  STD_LOGIC );
END   equ2;
ARCHITECTURE equation OF equ2 IS        --结构体一
    ……
END equation ;
ARCHITECTURE con_behave OF equ2 IS   --结构体二
    ……
END con_behave ;
ARCHITECTURE seq_behave OF equ2 IS   --结构体三
    ……
END seq_behave ;
```

实体 **equ2** 拥有三个结构体：**equation** 、**con_behave** 、 **seq_behave**，可以用配置语句选择与实体对应的结构体。

如选用结构体**con_behave**，可用以下语句实现：

```
CONFIGURATION  aequb OF  equ2  IS
    FOR  con_behave
    END  FOR ;
END CONFIGURATION;
```

二、VHDL的数据对象和数据类型

1. VHDL数据对象

VHDL数据对象 { 信号 / 变量 / 常量 }

（1）常量

常量：不变的量，定义时进行赋值，在整个程序中保持不变。

语法格式：

```
CONSTANT 常量名:数据类型： =表达式;
```

[例1.4.4] 常量定义举例

```
CONSTANT V:INTEGER:=8;
```

（2）变量

语法格式：

```
VARIABLE  变量名:数据类型[:=表达式];
```

[例1.4.5] 变量定义举例

```
VARIABLE y:INTEGER;
```

（3）信号

信号：定义电路中的连线和元件的端口的数据对象。

语法格式：

```
SIGNAL  信号名:数据类型;
```

[例1.4.6] 信号定义举例

```
SIGNAL A:INTEGER;
```

2. VHDL数据类型

（1）整数数据类型（INTEGER）

取值范围：-2147483547～ 2147483546

（2）实数数据类型（REAL）

取值范围：-1.0E38～ 1.0E38

（3）位数据类型（BIT）

属于枚举数据类型，取值为‘1’和‘0’。

（4）位矢量数据类型（BIT_VECTOR）

用双引号括起来的一组位数据，如“10011”，通常用来表示数据总线。

（5）布尔数据类型（BOOLEAN）

属于枚举数据类型，取值为‘TRUE’和‘FALSE’。常用来表示关系运算和关系运算结果。

（6）字符数据类型（CHARACTER）

ASCII码的128个字符，书写时用单引号，区分大小写，如‘a’、‘A’等。

（7）字符串数据类型（STRING）

双引号括起来的一串字符，如“abgh”。

（8）STD_LOGIC数据类型

属于枚举数据类型，取值有以下九种：

‘U’	初始值	‘X’	不定
‘0’	0	‘1’	1
‘Z’	高阻	‘W’	弱信号不定
‘L’	弱信号0	‘H’	弱信号1
‘–’	不可能情况		

（9）STD_LOGIC_VECTOR数据类型

用双引号括起来的一组STD_LOGIC数据，如“101011”，通常用来表示数据总线。

注意:使用STD_LOGIC、 STD_LOGIC_VECTOR数据类型时必须在库、程序包说明区进行说明。

二、VHDL的操作符和表达式

（1）算术操作符和算术表达式

操作符	说明
+	加
-	减
*	乘
/	除
**	乘方
mod	求模
rem	求余
abs	绝对值

[例1.4.8] 算术表达式举例

A+B-C

X*Y/Z

（2）逻辑操作符和逻辑表达式

操作符	说明
and	与
or	或
nand	与非
nor	或非
xor	异或
xnor	同或
not	逻辑非

[例1.4.9] 逻辑表达式举例

A AND B

NOT Z

（3）关系操作符和关系表达式

操作符	说明
=	等于
/=	不等于
<	小于
<=	小于等于
>	大于
>=	大于等于

[例1.4.10] 关系表达式举例

Y >= G

A=B

（4）并置操作符和并置表达式

并置操作符“&”主要用来将操作数或数组组合起来,以形成新的操作数。 例“10”&“11”结果为“1011”。

二、VHDL基本语句

- VHDL基本语句
  - 顺序语句
    - 顺序信号赋值语句
    - 变量赋值语句
    - IF 语句
    - CASE 语句
    - LOOP语句
    - NEXT
    - EXIT
    - 子程序和子程序调用语句
    - NULL
  - 并行语句
    - 块语句
    - 进程语句
    - 并行信号赋值语句
    - 并行过程调用语句
    - 元件声明例化语句
    - 生成语句

1. 顺序描述语句

执行顺序与书写顺序一致，只用于进程和子程序中。

（1）顺序信号赋值语句

格式：目标信号<=表达式;

[例1.4.11] 顺序信号赋值语句举例

```
Y<=A AND B ;
```

（2）变量赋值语句

格式：目的变量:=表达式;

[例1.4.12] 变量赋值语句举例

```
Y:=A + B ;
```

（3）IF语句（条件控制语句）

格式一：

```
IF 条件表达式 THEN
    顺序语句;
END IF;
```

[例]

```
IF a='1' THEN
    c<=b ;
END IF;
```

格式二：

```
IF 条件表达式 THEN
    顺序语句;
ELSE
    顺序语句;
END IF;
```

格式三：

```
IF 条件表达式 THEN
    顺序语句;
ELSIF条件表达式 THEN
    顺序语句;
……
ELSE
    顺序语句;
END IF;
```

（3）CASE语句

语法格式：

```
CASE 表达式 IS
    WHEN 选择值=>顺序语句;
    WHEN 选择值=>顺序语句;
     …
    [WHEN OTHERS=>顺序语句;]
END CASE;
```

“选择值”的取值应“选择唯一，覆盖全集”。

“选择值”的具体表示形式有以下四种：

```
WHEN 值=>顺序语句;
WHEN 值|值|…|值=>顺序语句;
WHEN 值 TO 值=>顺序语句;
WHEN OTHERS=>顺序语句;
```

[例1.4.13] 用CASE语句设计4选1数据选择器的程序片段

```
SIGNAL s:STD_LOGIC_VECTOR (1 DOWNTO 0);
…
CASE  s  IS
     WHEN“00”=>z<=a;
     WHEN“01”=>z<=b;
     WHEN“10”=>z<=c;
     WHEN“11”=>z<=d;
     WHEN  OTHERS=>z<=‘X’;
END  CASE;
```

（5）LOOP语句

无条件LOOP语句语法格式：

```
[LOOP标号:]LOO
    顺序语句;
END LOOP [LOO
```

需引入其它控制语句才能退出循环，如exit、next等。

FOR…LOOP语句语法格式

```
[LOOP标号:] FOR 循环变量
    顺序语句;
END LOOP [LOOP标号] ;
```

临时变量，必事先定义

规定顺序语句的执行次数，从循环变量初值开始每执行一次递增1，直至最大值。

WHILE…LOOP语句语法格式：

```
[LOOP标号:] WHILE 条件表达式 LOOP
    顺序语句;
END LOOP [LOOP标号];
```

[例1.4.14] WHILE…LOOP应用举例

```
abcd: WHILE  (i<10)  LOOP
        sun:=i+sum;
        i=i+1;
END  LOOP abcd;
```

（6）NEXT语句

格式：`NEXT [WHEN 条件];`

满足条件时中止当前循环，开始下一次循环。

[例1.4.15] NEXT语句举例

```
loop2:LOOP
      b:=b+1 ;
      NEXT WHEN b<10;
      …
END LOOP loop2;
```

（7）EXIT语句

格式：EXIT [标号][WHEN 条

满足条件时提前退出循环。

[例1.4.16] EXIT语句在比较器中的应用

```
FOR i IN 1 DOWNTO 0 LOOP
    IF (a(i)='1' AND b(i)='0' ) THEN
       a_less_than_b<=false;
       EXIT;
    ELSIF (a(i)='0' AND b(i)='1' ) THEN
       a_less_than_b<=true;
       EXIT;
    ELSE NULL;
    END IF;
END LOOP;
```

（3）子程序和子程序调用语句

VHDL子程序 { 过程（PROCEDURE）
函数（FOUNCTION）

函数定义语句的语法格式：

```
FOUNCTION〈函数名〉(参数表) RETURN〈数据类型〉 IS
BEGIN
    顺序语句;
    RETURN [返回变量名];
END 〈函数名〉;
```

函数调用语句的语法格式：

```
函数名（实际参数表）;
```

函数的参数只能是方式为IN的输入信号，函数只能有一个返回值。

[例1.4.17] 比较器函数形式的程序设计实例

```
FOUNCTION min(x,y:INTEGER) RETURN INTEGER IS
BEGIN
    IF  x<y  THEN
       RETURN x;
    ELSE
       RETURN y;
    END IF;
END min;
```

过程定义语句的语法格式：

```
PROCEDURE <过程名>(参数表) IS
BEGIN
    顺序语句;
END <过程名>;
```

过程调用语句的语法格式：

```
过程名（实际参数表）;
```

过程的参数可以为**IN**、**OUT**和**INOUT**方式，在进行参数说明时除了说明其名称、数据类型，还要说明其端口方式。

[例1.4.18] 比较器过程形式的程序设计实例

```
PROCEDURE swap(data: INOUT  data_array;
                            low,high:in  integer) IS
VARIABLE tmp: data_element;
BEGIN
    IF (data(low) > data(high)) THEN
        tmp:= data(low) ;
        data(low) := data(high);
        data(high) := tmp;
     END IF;
END swap;
```

[例1.4.19] 用过程调用语句调用过程swap

```
swap(datain,1,2);
```

1. 并行语句

当满足条件时，多个语句同时被执行，与书写顺序无关。

（1）BLOCK语句

功能：将一大段并行语句划分为若干子模块，提高了程序的可读性。

BLOCK语句的语法格式：

```
[块标号:]BLOCK[(块保护表达式)]
[块说明语句;]
BEGIN
并行语句;
END BLOCK[块标号];
```

块保护表达式是布尔表达式，当其为真时，该块中的语句被执行。

[例1.4.20] 用BLOCK语句设计2选1数据选择器。

```
ARCHITECTURE connect OF mux IS
SIGNAL tmp1, tmp2, tmp3:BIT;
BEGIN
  BLOCK
  BEGIN
     tmp1 <= d0 AND  not sel;
     tmp2 <= d1 OR  (not sel);
     tmp3<= tmp1 OR  tmp2;
     q <= tmp3;
  END BLOCK;
END connect;
```

（2）进程语句（PROCESS）

进程语句的语法格式：

```
[进程名:] PROCESS (敏感信号表)
[说明语句;]
BEGIN
顺序语句;
END PROCESS [进程名];
```

敏感信号变化启动进程。

说明语句说明数据类型、子程序、变量等，作用范围仅限于本进程。

[例1.4.21] 用语句进程设计“2输入与门”

```
nandx:] PROCESS (a,b)
BEGIN
Y<=a AND b;
END PROCESS nandx;
```

（3）并行信号赋值语句

并行信号赋值语句
- 简单并行信号赋值语句
- 条件并行信号赋值语句
- 选择并行信号赋值语句

简单并行信号赋值语句格式:

```
目标信号<=表达式;
```

[例1.4.22]简单并行信号赋值语句举例

```
ARCHITECTURE one OF sent IS
BEGIN
output<=inb;
END one;
```

条件信号赋值语句格式:

```
目标信号 <= 表达式1 when 赋值条件1 else
            表达式1 when 赋值条件1 else
            ……
            表达式n ;
```

[例1.4.23]条件信号赋值语句举例

```
x <=a WHEN(s="00") ELSE
     b WHEN(s="01") ELSE
     c WHEN(s="10") ELSE
     d;
```

选择信号赋值语句格式:

```
with 选择表达式 select
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n when 选择值n; ;
```

[例1.4.24]选择信号赋值语句举例

```
WITH 表达式 SELECT
x <=a WHEN(s="00"),
      b WHEN(s="01"),
      c WHEN(s="10"),
      d WHEN OTHERS;
```

（4）并行过程调用语句

并行过程调用语句与顺序过程调用语句形式基本相同，只是出现的位置不同。

[例1.4.25] 用并行过程调用语句调用过程swap

```
ARCHITECTURE …
BEGIN
swap(datain,1,2);
……
END;
```