VHDL语言基础（二）

中国传媒大学信息与通信工程学院

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
  - 顺序语句必须在进程PROCESS内
  - 进程本身是并行语句
- 并行执行的进程（PROCESS）
  - 进程语句是并行语句，进程语句之间是并行关系
  - 进程语句内部则是由一组在整个模拟期间连续执行的顺序语句构成，是顺序执行的
  - 进程之间通过信号或共享变量进行通信
- PROCESS是描述硬件并行工作的最常用、最基本的语句

进程语句（PROCESS）

进程名可以有也可以省略

进程语句从PROCESS开始

```
[进程名]： PROCESS（敏感信号表）
        说明部分；
BEGIN
        顺序语句1；
        顺序语句2；
        ……
        顺序语句n；
END  PROCESS；
```

敏感信号表是进程启动、触发的条件

可以说明数据类型、常量、变量和子程序等

是顺序语句，是一段程序

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
    PROCESS(A,B)
    BEGIN
        SUM <= A XOR B;
        CO <= A AND B;
    END PROCESS;
END RTL;
```

这个进程是一个电路模块，实现了一个半加器

敏感信号表中的A和B是半加器的输入，每当输入信号变换时，电路翻转，输出发生变化

信号赋值语句在进程内就是顺序语句

顺序语句

1. IF语句

- 根据所指定的条件来确定执行哪些语句
- 书写格式通常可以分为三种类型

格式 1

```
IF   条件   THEN
     顺序语句1;
     顺序语句2;
     ……
     顺序语句n;
END  IF;
```

- 如果条件成立，则执行IF语句所包含的顺序处理语句
- 如果条件不成立，程序将跳过IF语句所包含的顺序处理语句，而向下执行IF语句后的语句
- 这种描述可以用来描述D触发器

实例二：D触发器

输入端口：时钟CLK；数据输入：D；

输出：Q

- 时钟信号为驱动信号
- 只是在时钟的边沿到来时，其状态发生变化,因此时钟信号通常是描述时序电路程序的执行条件。
- 总是以时钟进程形式来进行描述，作为PROCESS的敏感量 PROCESS（CLK）

时钟边沿的描述（属性描述）

上升沿 **CLK'EVENT AND CLK = '1'**

下降沿 **CLK'EVENT AND CLK = '0'**

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY D_FF IS
PORT(CLK, D : IN STD_LOGIC;
     Q : OUT STD_LOGIC);
END D_FF;
ARCHITECTURE RTL OF D_FF IS
BEGIN
    PROCESS(CLK)
    BEGIN
        IF CLK'EVENT AND CLK = '1' THEN
            Q <= D;
        END IF;
    END PROCESS;
END RTL;
```

实例三：移位寄存器

输入端口：时钟CLK；数据输入：Din；

输出：Q

- 时钟信号为驱动信号
- 在IF…END IF 结构中，每个信号赋值语句产生一个触发器

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY REG_SHIFT IS
PORT(CLK, Din : IN STD_LOGIC;
     Q : OUT STD_LOGIC);
END REG_SHIFT;
ARCHITECTURE RTL OF REG_SHIFT IS
    SIGNAL D : STD_LOGIC_VECTOR(2 DOWNTO 0);
BEGIN
    PROCESS(CLK)
    BEGIN
        IF CLK'EVENT AND CLK = '1' THEN
            D(0) <= Din;
            D(1) <= D(0);
            D(2) <= D(1);
            Q <= D(2);
        END IF;
    END PROCESS;
END RTL;
```

实例四：带异步复位的D触发器

输入端口：时钟：CLK；数据输入：D；

异步复位输入：rst；输出：Q

- 复位信号：同步复位，异步复位
- 同类控制信号：复位信号、置位信号、使能信号
- IF语句优先级
- IF语句嵌套

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY D_FF IS
PORT(CLK, D, RST : IN STD_LOGIC;
     Q : OUT STD_LOGIC);
END D_FF;
ARCHITECTURE RTL OF D_FF IS
BEGIN
    PROCESS(CLK, RST)
    BEGIN
        IF RST = '1' THEN
           Q <= '0';
        ELSIF CLK'EVENT AND CLK = '1' THEN
            Q <= D;
        END IF;
    END PROCESS;
END RTL;
```

实例五：带同步复位的D触发器

输入端口：时钟：CLK；数据输入：D；

异步复位输入：rst；同步置位输入：PRESET

输出：Q

- 复位的作用：是电路在外部复位信号的作用下，从一个正确（期望）的状态开始执行。
- 全局复位信号：是设计中每一个模块都在同一个复位信号的控制下，从正确状态开始执行。
- 触发器的初始状态应该由复位信号来设置
- 复位信号：同步复位，异步复位
- 同类控制信号：复位信号、置位信号、使能信号

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY D_FF IS
PORT(CLK, D, RST, PRESET : IN STD_LOGIC;
     Q : OUT STD_LOGIC);
END D_FF;
ARCHITECTURE RTL OF D_FF IS
BEGIN
    PROCESS(CLK, RST)
    BEGIN
        IF RST = '1' THEN
           Q <= '0';
        ELSIF CLK'EVENT AND CLK = '1' THEN
           IF PRESET = '1' THEN
                Q <= '0';
           ELSE
                Q <= D;
           END IF;
        END IF;
    END PROCESS;
END RTL;
```

1. IF语句

格式 2

```
IF   条件   THEN
     顺序语句;
ELSE
     顺序语句;
END  IF;
```

- 指定的条件满足时，将执行THEN和ELSE之间所界定的顺序处理语句
- 当IF语句所指定的条件不满足时，将执行ELSE和END IF之间的顺序处理语句
- 用条件来选择两条不同程序执行的路径
- 描述的典型电路是二选一电路

实例五：二选一选择器

输入端口：输入A，B；输入选择：SEL；输出：Y

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY MUX IS
PORT(A, B, SEL : IN STD_LOGIC;
     Y : OUT STD_LOGIC);
END MUX;
ARCHITECTURE RTL OF MUX IS
BEGIN
    Y <= A WHEN SEL = '1' ELSE
         B;
END RTL;
```

两者等价

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

格式 3

```
IF   条件   THEN
     顺序语句;
ELSIF   条件   THEN
     顺序语句;
ELSIF   条件   THEN
     顺序语句;
ELSE
     顺序语句;
END   IF;
```

- 设置了多个条件，当满足所设的多个条件之一时，就执行该条件后跟的顺序处理语句
- 如果所有设置都不满足时，则执行**ELSE**和**END IF**之间的顺序处理语句
- 描述的典型电路是多选一电路，优先编码器

实例六：优先编码器

4路输入I[3..0]；编码输出：Y[1..0]

```
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 <= "11" WHEN I(3) = '0' ELSE
         "10" WHEN I(2) = '0' ELSE
         "01" WHEN I(1) = "0' ELSE
         "00";
END RTL;
```

两者等价

```
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
    PROCESS(I)
    BEGIN
        IF I(3) = '0' THEN
            Y <= "11";
        ELSIF I(2) = '0' THEN
            Y <= "10";
        ELSIF I(1) = '0' THEN
            Y <= "01";
        ELSE
            Y <= "00";
        END IF;
    END PROCESS;
END RTL;
```

顺序语句

2. CASE语句

- 从许多不同语句的序列中选择其中之一执行
- 常用来描述总线或编码译码的行为

CASE语句的格式

```
CASE  表达式  IS
   WHEN  条件表达式1  => 顺序语句1;
   WHEN  条件表达式2  => 顺序语句2;
    ……
   WHEN  条件表达式n => 顺序语句n;
   WHEN  OTHERS => 顺序语句
END  CASE;
```

- 当CASE和IS之间的表达式的取值满足指定的条件表达式的值时，执行后跟的由符号 => 所指的顺序处理语句

实例七：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;
```

两者等价

```
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
    PROCESS(A)
    BEGIN
        CASE A IS
            WHEN "00" => Y <= "1110";
            WHEN "01" => Y <= "1101";
            WHEN "10" => Y <= "1011";
            WHEN "11" => Y <= "0111";
            WHEN OTHERS => Y <= "1111";
        END CASE;
    END PROCESS;
END RTL;
```

时序逻辑

•时序电路通常用时钟进程的形式来描述

时序电路常见的两种描述方式

```
PROCESS(CLK)
BEGIN
   IF（时钟边沿条件）THEN
         顺序语句;
   END IF;
END PROCESS;
```

```
PROCESS
BEGIN
   WAIT ON CLK UNTIL 时钟边沿条件
      顺序语句;
END PROCESS;
```

前面的触发器和寄存器的描述，时序电路进程中最外层一定是IF……END IF结构

实例八：模16计数器

输入端口：时钟输入CLK；异步复位输入：RST，计数输出：Q[3..0]

- 是对脉冲进行计数的电路，可以用于如事件计数、定时、分频和控制等
- 同步计数器在时钟脉冲的控制下，构成计数器的各触发器的状态同时发生变化
- 异步计数器是把上一级计数器的输出作为下一级计数器的时钟输入，多个计数器串联起来就构成了异步计数器
- 结构设计方法：设计加减法器，电路连接，一般描述计数器不必描述到这一层次
- 最简单常用的方法：＋、－算术运算符

复位输入	时钟输入	输出端			
RST	CLK	Q3	Q2	Q1	Q0
0	X	0	0	0	0
1	0	不变	不变	不变	不变
1	1	不变	不变	不变	不变
1	下降沿	不变	不变	不变	不变
1	上升沿	计数值加1			

时序电路进程中最外层一定是IF……END IF结构

计数器结构，代码中信号反馈的问题

STD_LOGIC_VECTOR类型进行算术运算

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;
ENTITY COUNTER IS
PORT(
      CLK : IN STD_LOGIC;
      RST : IN STD_LOGIC;
      Q   : OUT STD_LOGIC_VECTOR(3 DOWNTO 0));
END COUNTER;
ARCHITECTURE RTL OF COUNTER IS
      SIGNAL TEMP : STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
      PROCESS(CLK, RST)
      BEGIN
            IF RST = '0' THEN
                  TEMP <= "0000";
            ELSIF CLK'EVENT AND CLK = '1' THEN
                  IF TEMP = "1111" THEN
                        TEMP <= "0000";
                  ELSE
                        TEMP <= TEMP + 1;
                  END IF;
            END IF;
      END PROCESS;
      Q <= TEMP;
END RTL;
```

实例九：异步清零/同步使能 模12计数器

- 功能
  - 异步清零：对计数值清零
  - 计数使能：**EN**打开，允许计数
  - 同步计数：在时钟控制下，计数值加**1**
- 端口
  - 输入控制： **clr**、**en**、**clk**、
  - 输出数据： **Qa**、**Qb**、**Qc**、**Qd**

- 四位二进制计数器
- 每一位，占用**1bit**寄存器
- 每一位寄存器在**CLK**作用下，同时（同步）变化

输入			输出			
CLR	EN	CLK	Qd	Qc	Qb	Qa
1	X	X	0	0	0	0
0	0	↑	不变			
0	1	↑	计数值+1（MOD12）			

定义寄存器CNT，用于状态记忆

异步复位的描述

同步使能的描述

计数器模值的描述

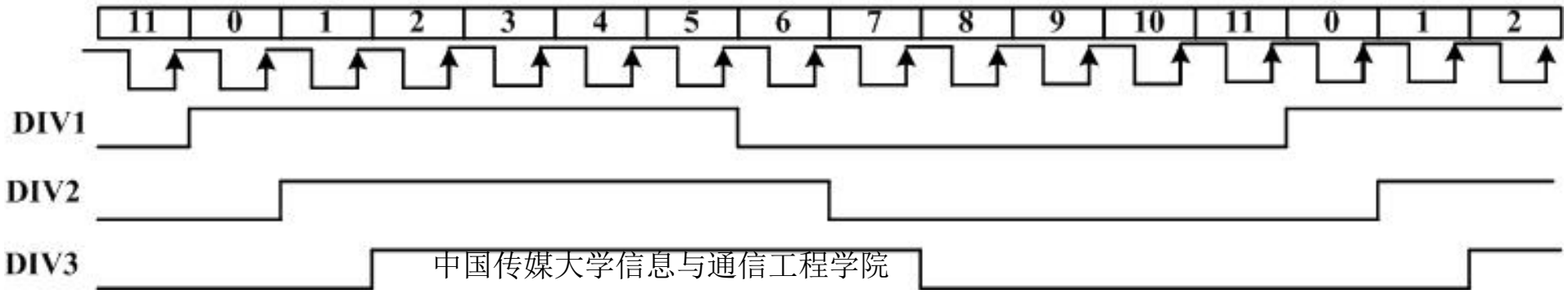
```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_ARITH.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL;

ARCHITECTURE RTL OF COUNT12 IS
SIGNAL  CNT: STD_LOGIC_VECTOR(3 DOWNTO 0);
BEGIN
QD<=CNT(3);QC<=CNT(2);QB<=CNT(1);QA<=CNT(0);

PROCESS(CLK)
BEGIN
        IF CLR='1' THEN
                CNT<="0000";
        ELSIF CLK'EVENT AND CLK='1' THEN
                IF EN='1' THEN
                        IF CNT="1011" THEN
                                CNT<="0000";
                        ELSE
                                CNT<=CNT+'1';
                        END IF;
                END IF;
        END IF;
END PROCESS;
END RTL;
```

实例十：模12分频器

- 功能
  - 对输入时钟进行12分频
  - 注意分频的占空比
  - 多路输出信号之间的相位关系
- 端口
  - 输入控制 ： **clk**
  - 计数状态信号量 ： **counter**
  - 分频输出 ： **div1/div2 /div3**

  （相位相差一个时钟周期）

- 时序要求，如下图所示
- 调整输出信号 --- 占空比、频率
- 调整多路输出信号 --- 相位差



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

ENTITY DIV12 IS
PORT(  CLK: IN       STD_LOGIC;
       DIV1,DIV2,DIV3:  OUT  STD_LOGIC);
END DIV12 ;
ARCHITECTURE RTL OF DIV12 IS
SIGNAL REG : INTEGER  RANGE 0 TO 15;
BEGIN
       PROCESS(CLK)
       BEGIN
         IF CLK'EVENT AND CLK='1' THEN
           IF REG = 11 THEN
               REG <=0;
           ELSE
              REG <=REG+1;
           END IF;
         END IF;
       END PROCESS;
```

计数器描述：分频比

```
PROCESS(REG)
BEGIN
  CASE REG IS
  WHEN 0 TO 5 => DIV1 <='1';
  WHEN 6 TO 11 => DIV1 <='0';
  WHEN OTHERS => DIV1 <='0';
  END CASE;
END PROCESS;
PROCESS(REG)
BEGIN
  CASE REG IS
  WHEN 1 TO 6 => DIV2 <='1';
  WHEN 7 TO 11|0 => DIV2 <='0';
  WHEN OTHERS => DIV2 <='0';
  END CASE;
END PROCESS;
PROCESS(REG)
BEGIN
  CASE REG IS
  WHEN 2 TO 7 => DIV3 <='1';
  WHEN 8 TO 11|0|1=> DIV3 <='0';
  WHEN OTHERS => DIV3 <='0';
  END CASE;
END PROCESS;
END RTL;
```

输出逻辑描述

占空比，相位关系