Aide mémoire de la syntaxe du langage VHDL

CHILLET Daniel & COUSIN Jean-Gabriel

23 octobre 2001

1 Les opérations de base

1.1 Opérations logiques

AND, OR, NAND, NOR, XOR

1.2 Opérations relationnelles

=, /=, ¡, ¡=, ¿, ¿=

1.3 Opérations d'addition

+, -
& (concaténation de chaines, exemple : "concat"&"énation"

1.4 Opérations de multiplication

*, /, mod, rem

1.5 Opérations diverses

**, abs, not

2 Inclure une librairie

```
LIBRARY IEEE ;
USE IEEE.Std_Logic_1164.all ;
```

3 Déclaration de types

SIGNAL
VARIABLE
CONSTANT
FILE
ACCESS

4 Types du package standard

BOOLEAN	(FALSE, TRUE)
BIT	('0', '1')
CHARACTER	
TIME	(fs, ps, ns, us, ms, sec, min, hr)
INTEGER	
NATURAL	(0, INTEGER'high)
POSITIVE	(1, INTEGER'high)
REAL	
STRING	array (POSITIVE RANGE <>) OF CHARACTER ;
BIT_VECTOR	array (NATURAL RANGE <> OF BIT)

5 Attributs

5.1 Attributs sur les types

NomType'HIGH	Rend la valeur la plus grande du type NomType
NomType'LOW	Rend la valeur la plus petite du type NomType
NomType'RIGHT	Rend la valeur limite de droite du type NomType
NomType'LEFT	Rend la valeur limite de gauche du type NomType
NomType'POS(X)	Rend la position de l'élément X dans l'énumération du type NomType
NomType'VAL(X)	Rend l'élément se trouvant à la position X dans l'énumération du type NomType
NomType'SUCC(X)	Rend l'élément successeur de l'élément X dans l'énumération du type NomType
NomType'PRED(X)	Rend l'élément prédecesseur de l'élément X dans l'énumération du type NomType

5.2 Attributs sur les tableaux

NomTableau'LEFT(N)	Rend la valeur de la borne gauche du N ième indice du tableau NomTableau
NomTableau'RIGHT(N)	Rend la valeur de la borne droite du N ième indice du tableau NomTableau
NomTableau'HIGH(N)	Rend la valeur de la borne la plus haute du N ième indice du tableau NomTableau
NomTableau'LOW(N)	Rend la valeur de la borne la plus basse du N ième indice du tableau NomTableau
NomTableau'RANGE(N)	Rend l'intervalle de variation du N ième indice du tableau NomTableau
NomTableau'REVERSE_RANGE(N)	Rend l'inverse de l'intervalle de variation du N ième indice de NomTableau
NomTableau'LENGTH(N)	Rend la taille de l'intervalle de variation du N ième indice de NomTableau

5.3 Attributs sur les signaux

NomSignal'DELAYED(T)	Rend un signal identique à NomSignal mais décalé d'un temps T
NomSignal'STABLE(T)	Rend TRUE si le signal NomSignal n'a pas eu d'évènement pendant le temps T
NomSignal'QUIET(T)	Rend TRUE si le signal NomSignal est resté tranquille (ni évènement, ni transaction) pendant le temps T
NomSignal'TRANSACTION	Rend un signal de type BIT qui change de valeur à chaque fois que le signal NomSignal devient actif
NomSignal'EVENT	Rend TRUE si le signal NomSignal vient d'avoir un évènement durant le cycle de simulation
NomSignal'ACTIVE	Rend TRUE si le signal NomSignal est actif durant le cycle de simulation
NomSignal'LAST_EVENT	Rend le temps écoulé depuis le dernier évènement arrivé sur le signal NomSignal
NomSignal'LAST_ACTIVE	Rend le temps écoulé depuis que le signal NomSignal a été actif pour la dernière fois
NomSignal'LAST_VALUE	Rend la valeur du signal NomSignal immédiatement avant le dernier changement de celui-ci

6 Déclarations

6.1 Déclarations de types et sous types

```
TYPE boolean IS (TRUE, FALSE);
TYPE tableau IS ARRAY (0 TO 10) OF type;
TYPE tableau IS ARRAY (10 DOWNTO 0) OF type;
TYPE tableau2x2 IS ARRAY (0 TO 20) OF tableau;
TYPE tableau2x2 IS ARRAY (0 TO 20, 15 DOWNTO 6) OF type;
TYPE tableau IS ARRAY (type RANGE <>) OF type;
SUBTYPE nom IS tableau (3 TO 8);
```

6.2 Déclarations d'alias

```
VARIABLE mot : BIT_VECTOR (15 DOWNTO 0);
ALIAS PoidsFort : BIT_VECTOR (15 DOWNTO 8) IS mot (15 DOWNTO 8);
ALIAS PoidsFaible : BIT_VECTOR (7 DOWNTO 0) IS mot (7 DOWNTO 0);
```

6.3 Déclarations de fonctions de résolution

```
TYPE bit IS ('0', '1', 'Z', 'X');
TYPE tabBit IS ARRAY (INTEGER RANGE <>) OF bit;
FUNCTION ResoudBit (src : IN tabBit) RETURN bit;
SUBTYPE bitResolu IS ResoudBit bit;


FUNCTION ResoudBit (src : IN tabBit) RETURN bit IS
        VARIABLE Result : bit := 'Z';
        BEGIN
                FOR i IN src'RANGE LOOP
                        CASE src(i) IS
                                WHEN '0' =>
                                        IF Result = '1' THEN RETURN 'X';
                                        ELSE Result := '0';
                                        END IF;
                                WHEN '1' =>
                                        IF Result = '0' THEN RETURN 'X';
                                        ELSE Result := '1';
                                        END IF;
                                WHEN 'X' =>
                                        RETURN 'X';
                                WHEN OTHERS =>
                        END CASE;
                END LOOP;
        END ResoudBit;
```

6.4 Déclarations de variables

```
CONSTANT nom : type := valeur;
VARIABLE nom : type := valeur;
SIGNAL nom : type := valeur;
```

6.5 Déclarations de packages

```
PACKAGE nom IS
        déclaration des types;
        déclaration des constantes;
        déclaration des variables;
        déclaration des entétes des fonctions;
END nom;


PACKAGE BODY nom IS
        écriture des corps des fonctions;
END nom;
```

6.6 Déclarations de procédures et fonctions

```
FUNCTION nom (param1 : IN type := vpd; param2 : OUT type := vpd) RETURN type;


FUNCTION nom (param1 : IN type := vpd; param2 : OUT type := vpd) RETURN type IS
        déclaration de variables;
        BEGIN
                corps de la fonction;
        END nom;

PROCEDURE nom (param1 : IN type := vpd; param2 : OUT type := vpd);


PROCEDURE nom (param1 : IN type := vpd; param2 : OUT type := vpd) IS
        déclaration de variables;
        BEGIN
                corps de la fonction;
        END nom;
```

6.7 Déclarations d'entitée

```
ENTITY Nom IS
        GENERIC (param1 : type := vpd ; param2 : type := vpd);
        PORT (port1 : IN type ; port2 : OUT type);
END Nom;
```

6.8 Déclarations d'architecture

```
ARCHITECTURE NomArchi of Nom IS
        déclaration de signaux;
        déclaration de composants;
        instanciation de composants;
        écriture de processus;
        écriture de blocs;
END NomArchi;
```

7 Les blocs

```
ARCHITECTURE Bloc OF MultAdd IS
BEGIN
        NomBloc: block (condition de garde)
                déclarations locales;
                BEGIN
                        instructions concurrentes (instanciation de signaux);
                        NomSignal <= GUARDED SignalSource AFTER Temps;
                END BLOCK NomBloc;
END Bloc;
```

8 Boucles

```
LOOP
        corps de la boucle;
END LOOP;


FOR i IN 0 TO 10 LOOP
        corps de la boucle;
END LOOP;


FOR i IN 100 DOWNTO 10 LOOP
        corps de la boucle;
END LOOP;


WHILE (condition) LOOP
        corps de la boucle;
END LOOP;
```

9 Conditions

```
IF condition1 THEN
        statement;
ELSIF condition2 THEN
        statement;
ELSIF condition3 THEN
        statement;
ELSE
        statement;
END IF;


CASE nom IS
        WHEN valeur1 =>
                statement;
        WHEN valeur2 =>
                statement;
        WHEN valeur3 =>
                statement;
        WHEN OTHERS =>
                statement;
END CASE;
```

10 Processus

```
label: PROCESS (liste de sensibilité)
        déclaration de variables;
        BEGIN
                corps du processus;
        END PROCESS label;


label: PROCESS
        déclaration de variables;
        BEGIN
                corps du processus;
                WAIT tempsI, signal, condition
        END PROCESS label;
```

11 La clause WAIT

```
WAIT;
WAIT ON liste de sensibilité;
WAIT UNTIL condition;
WAIT FOR temps;
WAIT ON liste de sensibilité UNTIL condition FOR temps;
```

12 Instanciations

12.1 Instanciation de composants

```
op: NomEntite GENERIC MAP ( Generic1, Generic2, ... )
                PORT MAP ( Port1, Port2,... );


op: NomEntite GENERIC MAP ( NomGeneric1 => Generic1, NomGeneric2 => Generic2, ... )
                PORT MAP ( NomPort1 => Port1, NomPort2 => Port2,... );
```

12.2 Instanciation de signaux

```
NomSignal <= Valeur;
NomSignal <= Valeur1 AFTER Temps1, ···, ValeurN AFTER TempsN;
NomSignal <= transport S1 AFTER Temps;
```

(permet de filtrer les impulsions dont la durée est inférieur à Temps)

13 Exemple : le multiplieur additionneur

13.1 L'entitée MultAdd

```
ENTITY MultAdd IS
        GENERIC (TempsTraverse: TIME := 50 ns);
        PORT (entree1: IN type;
                entree2: IN type;
                entree3: IN type;
                sortie: OUT type);
END MultAdd;
```

13.2 Description comportementale

```
ARCHITECTURE comportementale OF MultAdd IS
        BEGIN
        PROCESS (entree1, entree2, entree3)
                BEGIN
                sortie  <= entree1 * entree2 + entree3 AFTER TempsTraverse;
        END PROCESS;
END nom archi;
```

13.3 Description structurelle

```
ARCHITECTURE structurelle OF MultAdd IS


        SIGNAL intermediaire: type;


        COMPONENT Mult
                GENERIC (TpsMult  : TIME);
                PORT (in1, in2  : IN type; out  : OUT type);
        END COMPONENT;


        COMPONENT Add
                GENERIC (TpsAdd  : TIME);
                PORT (in1, in2  : IN type; out  : OUT type);
        END COMPONENT;


BEGIN
        Mult1 : Mult GENERIC MAP (30 ns)
                PORT MAP (entree1, entree2, intermediaire);


        Add1 : Add GENERIC MAP (20 ns)
                PORT MAP (in1 => entree3, in2 => intermediaire, out => sortie);
END nom archi;
```

13.4 Description flot de données

```
ARCHITECTURE DataFlow OF MultAdd IS
BEGIN
        intermediaire <= entree1 * entree2;
        sortie <= intermediaire + entree3;
END DataFlow;
```

13.5 Description à l'aide de blocs guardés

```
ARCHITECTURE Bloc OF MultAdd IS
        SIGNAL intermediaire: type;
BEGIN
        NomBloc: block (NOT entree1'STABLE OR NOT entree2'STABLE
                        OR NOT entree3'STABLE)
                BEGIN
                        intermediaire <= GUARDED entree1 * entree2 AFTER TpsAdd;
                        sortie <= GUARDED intermediaire + entree3 AFTER TpsMult;
                END BLOCK NomBloc;
END Bloc;
```

14 Description d'un fichier de test

```
ENTITY SimulMultAdd IS
        GENERIC (TempsTraverse: TIME := 50 ns);
END SimulMultAdd;


ARCHITECTURE Comportement OF SimulMultAdd IS


SIGNAL S1, S2, S3, Result: type  ;


        COMPONENT MultAdd
                GENERIC (TempsTraverse  : TIME);
                PORT (entree1, entree2, entree3  : IN type; sortie  : OUT type);
        END COMPONENT;


BEGIN
        MultAdd1: MultAdd GENERIC MAP (TempsTraverse)
                PORT MAP (S1, S2, S3, Result);


        S1 <= ValeurS11 AFTER TempsS11, ···, ValeurS1N AFTER TempsS1N;
        S2 <= ValeurS21 AFTER TempsS21, ···, ValeurS2N AFTER TempsS2N;
        S3 <= ValeurS31 AFTER TempsS31, ···, ValeurS3N AFTER TempsS3N;


END Comportement;
```

15 Manipulation des packages IEEE

15.1 Types du package Std_Logic_1164

STD_ULOGIC	('U', 'X', '0', '1', 'Z', 'W', 'L', 'H', '-') type non résolu
STD_ULOGIC_VECTOR	tableau de STD_ULOGIC
STD_LOGIC	TYPE A MANIPULER type résolu associé à la fonction de résolution resolved
STD_LOGIC_VECTOR	TYPE A MANIPULER type tableau de Std_Logic
X01	sous type restreint aux valeurs de X à 1
X01Z	sous type restreint aux valeurs de X à Z
UX01	sous type restreint aux valeurs de U à 1
UX01Z	sous type restreint aux valeurs de U à Z

15.2 Types du package Std_Logic_arith

UNSIGNED	tableau non contraint de Std_Logic
SIGNED	tableau non contraint de Std_Logic
SMALL_INT	entier valant 0 ou 1

15.3 Fonctions du package Std_Logic_1164

Redéfinition de l'ensemble des fonctions logiques sur le type Std_Logic
Redéfinition de l'ensemble des fonctions logiques sur le type Std_Logic_Vector
Définition de quelques fonctions de convertions

15.4 Fonctions du package Std_Logic_arith

Redéfinition de l'ensemble des fonctions arithmétiques sur le type Std_Logic Attention, ce package est écrit de façon à manipuler des vecteurs signés ou non signés. Le choix (signé ou non signé) s'effectue en incluant dans les descriptions soit le package Std_Logic_Signed soit le package Std_Logic_Unsigned

Définition de quelques fonctions de convertions

15.5 Fonctions du package Std_Logic_Signed

Redéfinition de l'ensemble des fonctions arithmétiques sur le type Std_Logic. Les fonctions tiennent compte du signe du ou des vecteurs

15.6 Fonctions du package Std_Logic_Unsigned

Redéfinition de l'ensemble des fonctions arithmétiques sur le type Std_Logic. Les fonctions ne tiennent pas compte du signe du ou des vecteurs

16 Quoi déclarer, et où?

	Spécification d'entité	Architecture	Configuration	Spécification de paquetage	Corps de paquetage	Bloc	Processus	Sous programme
Déclaration sous programme			■					
Corps de sous programme			■	■				
Déclaration de type			■					
Déclaration de sous type			■					
Déclaration de constante			■					
Déclaration de variable	■	■	■	■	■	■		
Déclaration de signal			■		■		■	■
Déclaration de fichier			■					
Déclaration d'alias			■					
Déclaration de composant	■		■		■		■	■
Déclaration d'attribut			■		■			
Spécification d'attribut					■			
Spécification de déconnection			■		■		■	■
Spécification de configuration	■		■	■	■		■	■
Déclaration d'entité	■	■	■	■	■	■	■	■
Déclaration de configuration	■	■	■	■	■	■	■	■
Déclaration de paquetage	■	■	■	■	■	■	■	■
Clause USE								

TAB. 1 - *Quoi, où? (si la case (ligne i, colonne j) est grisée, cela indique qu'il n'est pas po de déclarer l'objet de la ligne i dans l'objet de la colonne j)*