Le langage de description VHDL

1 - Présentation

VHDL = **V**hsic **H**ardware **D**escription **L**anguage

Vhsic : Very High Speed Integrated Circuit

Le VHDL est un langage de description du comportement d'un circuit logique programmable.
Les plus anciens sont PALASM, ORCAD PLD et ABEL (un des plus utilisé).
Le VHDL est un langage de haut niveau au meme titre que le C, le pascal,etc….
Il permet de décrire le comportement d'un circuit en faisant abstraction du contenu matériel du circuit.
Ce qui permet la portabilité des réalisations.
Il permet de matérialiser les structures électroniques de circuits complexes.

2 - Description en VHDL :

2.1 CONTENU D'UNE DESCRIPTION :

Une description VHDL est composée de deux parties :
L'entité (Entity) : Elle permet de définir les entrées et sorties du composant, leurs noms et leurs types.
L'architecture (architecture) : elle décrit le comportement attendu du circuit (la partie interne du circuit)

Exemple 1 : Réalisation d'un OU-EXCLUSIF

```
-----------------------------------------
-- Projet : coursvhdl.wpr
-- Fichier : ou_exclu.vhd
-- Auteur : BERNARD Sebastien
-- Date : 03/03/2009
-- Description : realisation d'un Ou Exclusif a 2 entres
-----------------------------------------

-- declaration des librairies
library ieee;
use ieee.std_logic_1164.all;

-- definition des entrees/sorties => l'entite
entity ouexclu is
port (    A : in std_logic;
     B : in std_logic;
     X : out std_logic);
end ouexclu;

-- description du fonctionnemment => l'architecture
architecture description of ouexclu is
begin
x <= (NOT(A) AND B) OR (A AND NOT (B));
end description;
```

Librairies

Entité

Architecture

2.2 SYNTAXE du langage de description VHDL :

Ci-dessous la description de quelques mots réservés pour le langage VHDL.

Une description plus complète du langage VHDL est disponible sur Internet. Se reporter à la syntaxe exhaustive détaillée sur le site d'Alain MOUFLET :

http://www.chez.com/amouf/syntaxe.htm

A/ Mots réservés pour la partie ENTITY

Commande PORT :
Cette commande sert à définir les signaux d'entrées et de sorties du bloc.
Exemple :

```
Port
(
a,b : in std_logic;
c : out integer
);
```

Nom Mode Type

Le mode (sens) :
=> IN : Entrée
=> OUT : Sortie
=> INOUT : Bidirectionnel (entrée/sortie)
=> BUFFER : Nœud interne utilisé pour un feedback

Le Type :
=> Boolean : variable vraie ou faux (utilisé dans les boucles)
=> Bit : 0 ou 1.
=> bit_vector : tableau de bits.
=> Integer : entier de -2^{31} à 2^{31} -1

- La bibliothèque standard IEEE 1164 fournit également /
=> std_ulogic (pour la modélisation)
=> std_logic (pour utiliser en plus de l'état logique « 0 » et « 1 », l'état haute impédance « Z »)
ainsi que des tableaux de ces deux types
Note : On doit signaler à VHDL d'utiliser cette bibliothèque :

```
library ieee;
use iee.std_logic_1164.all;
```

Remarque sur le nom du signal :
- Majuscules/minuscules sont équivalents,
- Le premier caractère doit être une lettre
- Le dernier caractère ne peut être un "underscore"
- Deux "underscores" successifs sont interdits

Remarque sur les vecteurs :
Exemple :

```
E : IN STD_LOGIC_VECTOR (3 downto 0)
```

Dans cette exemple E est un vecteur de bits (un bus de 4 bits). On peut adresser chacun des bits E(3), E(2),E(1),E(O).
Note : on peut aussi définir un vecteur comme celui-ci : `STD_LOGIC_VECTOR (0 to n)`

B/ ARCHITECTURE - DESCRIPTION COMPORTEMENTALE

Il existe trois modèle de description d'une architecture. Toutes les trois utilisables. La première est la description comportementale sous forme de processus.

Une descripton comportementale fournit un algorithme qui modélise le fonctionnement du circuit

Une déclaration de processus contient un algorithme

- Elle commence par une étiquette (optionnel), le mot clé "**process**" et une liste des signaux actifs (sensivity list)

- La liste des signaux actifs indique quels signaux provoqueront l'exécution du processus

Exemple d'architecture comportementale :

```
library ieee;
use ieee.std_logic_1164.all;
entity comp4b is port (
a, b: in std_logic_vector(3 downto 0)
aegalb: out std_logic);
end comp4b;
architecture comportement of comp4b is
begin
comp: process (a, b)
begin
if a = b then
aegalb <= '1';
else
aegalb <= '0';
end if;
end process comp;
end comportement;
```

C/ ARCHITECTURE - DESCRIPTION FLOT DE DONNEES

Une description flot de données spécifie comment la donnée est transférée de signal à signal sans utiliser d'affectations séquentielles (comme dans la description comportementale)

- Cette description ne nécessite pas de déclaration de processus,
-Toutes les déclarations s'exécutent en même temps ("concurrent signal assignment")

Exemple d'architecture Flot de données :

```
architecture FlotDonnees of comp4b is
begin
aegalb <= '1' when (a = b) else '0';
end FlotDonnees;
```

D/ ARCHITECTURE - DESCRIPTION STRUCTURELLE

Méthode de description par interconnexion de composants élémentaires définis dans des bibliothèques.
Les composants sont instanciés et connectés.
- Ils doivent être définis dans un package et compilés dans une bibliothèque
- Les bibliothèques sont attachées par une déclaration

Exemple d'architecture STRUCTURELLE :

```
library ieee;
use ieee.std_logic_1164.all;

entity comp4b is port(
a, b: in std_logic_vector(3 downto 0)
aegalb: out std_logic);
end comp4b;

use work.portespkg.all;
architecture structure of comp4b is
signal x : std_logic_vector(0 to 3)
begin
u0: xnor2 port map (a(0),b(0),x(0));
u1: xnor2 port map (a(1),b(1),x(1));
u2: xnor2 port map (a(2),b(2),x(2));
u3: xnor2 port map (a(3),b(3),x(3));
u4: and4 port map (x(0),x(1),x(2),x(3),aegalb);
end structure;
```

Remarques :
=> xnor2 et and4 sont dans la bibliothèque work.portespkg.
=> le mapping de chaque entrée et sortie des composants est défini par la commande Port mapp
=> x est un signal (signal interne qui sert de lien entres les composants de l'architecture)

E/ ARCHITECTURE - DESCRIPTION SEQUENTIELLE

Les process sont séquentiels, ils possèdent une liste de sensibilité les activant, les signaux ne sont effectivement modifié qu'à la fin du process.

e passage d'un état à l'état suivant se fait sur un événement. Une transition d'un signal d'horloge par exemple.

Exemple d'architecture SEQUENTIELLE :

```
label: process (HORLOGE)
begin
if(rising_edge(HORLOGE)) then
D <= C;
B <= A;
C <= B;
end if;
end process;
```

F/ AUTRES ELEMENTS DU LANGAGE

1/ Les opérateurs :

Classe	Symbole	Fonction	Definit pour
Opérateurs divers Classe de plus haute priorité	not ** abs	complément exponentiel valeur absolue	bit, booléen entier, réel numérique
Opérateurs multiplicatifs	* / mod rem	multiplication division modulo reste	numérique entier
Signe (unaire)	+ -	positif négatif	numérique
Opérateurs additifs (binaire)	+ - &	addition soustraction concaténation	numérique 1 dimension
Opérateurs relationnels	= /= < > <= >=	égal différent inférieur supérieur inférieur ou égal supérieur ou égal	tous les types retournent un booléen Scalaire retourne un booléen
Opérateurs logiques (binaire) Classe de plus faible priorité	and or nand nor xor	et logique ou logique et non logique ou non logique ou exclusif	bit booléen vecteur

2/ Les attributs (des signaux) :

Exemple : If clk'event and clk=1 then ... (vrai si clk vient de monter)

Attributs	Définitions - informations de retour
'high	Placé près d'un nom de tableau, il retourne la valeur du rang le plus haut (la valeur de retour est de type entier).
'low	Placé près d'un nom de tableau, il retourne la valeur du rang le plus bas (la valeur de retour est de type entier).
'left	Placé près d'un nom de vecteur ou tableau, il retourne la valeur du rang le plus à gauche (la valeur de retour est de type entier).
'right	Placé près d'un nom de vecteur ou tableau, il retourne la valeur du rang le plus à droite (la valeur de retour est de type entier).
'range	Placé près d'un signal, il retourne la valeur de l'intervalle spécifié par l'instruction range lors de la déclaration du signal ou du type utilisé.
'reverse_range	Placé près d'un signal, il retourne la valeur de l'intervalle inverse spécifié par l'instruction range lors de la déclaration du signal ou du type utilisé.
'length	Retourne X'high - X'low + 1 (sous la forme d'un entier).
'event	Vrai ou faux si le signal auquel il est associé vient de subir un changement de valeur.
'active	Vrai ou faux si le signal auquel il est associé vient d'être affecté.
'last_value	Retourne la valeur précédente du signal auquel il est associé.
'last_event ou 'last-active	Retournent des informations de temps concernant la date d'affectation ou de transition des signaux auxquels ils sont affectés
'stable(T)	Vrai ou faux si le signal auquel il est associé n'est pas modifié pendant la durée T.
'quiet	Vrai ou faux si le signal auquel il est associé n'est pas affecté pendant la durée T.
'transaction	Retourne une information de type bit qui change d'état lorsque le signal auquel il est associé est affecté.

(tableau d'après T.BLOTIN : le langage de description VHDL)

3/ Les affectations

Affectation simple - La grandeur A prend la valeur de B

```
GrandeurA<=GrandeurB;
```

Affectation conditionnelle - La grandeur A prend la valeur de B sous conditions

```
GrandeurA<=GrandeurB when condition1 else
           GrandeurC when condition2 else
           GrandeurD;
```

Affectation sélective - La grandeur A prend la valeur de B selon la valeur prise par une variable de choix

```
With choix select

GrandeurA<=GrandeurB when valeur1,
          GrandeurC when valeur2,
          GrandeurD when others;
```

4/ Les boucles

Effectuer des instructions n fois. Note « i » ne peut représenté qu'un indice d'une variable exemple E(i)

```
For i in MIN to MAX generate
{instructions}
end generate;
```

5/ Les agrégats

Les agrégats de valeurs peuvent être affectés de la façon suivante . Toutes les écritures sont identiques.

Exemple du vecteur Q :

```
Q : OUT STD_LOGIC_VECTOR (7 DOWNTO 0);
```

Affectations possibles :

```
Q <= "11111011";  -- on spécifie la valeur de chaque bit.
Q <= ('1','1','1','1','1','0','1','1');  -- on spécifie la valeur de chaque bit, bit à bit
dans l'ordre.
Q <= (7=>'1',6=>'1',5=>'1',4=>'1',3=>'1',1=>'1',0=>'1',2=>'0');  -- on spécifie la valeur
de cçhaque bit, en spécifiant le rang du bit dans l'ordre ou le désordre.
Q<=(2=>'0', others=>'1'); -- utilisation du mot clef OTHERS
Q<=(7 downto 3=>'1', 1 downto 0 =>'1', others=>'0'); -- utilisation du mot clef DOWNTO
```

6/ Les processus

Les processus sont activés par une liste de sensibilité. Deux syntaxes possibles :
1ERE SYNTAXE : Liste de sensibilités inclues dans le mot PROCESS :

Process (liste de sensibilité)
BEGIN
-- instructions séquentielles;
END PROCESS;

Le déclenchement ce fait avec l'instructions IF et les attributs des signaux sur font montant .. Voir exemple suivant...

Exemple :

```
architecture ARCH_COMPT_4 of COMPT_4 is
begin
process(H,R)
      begin
      if R='1'
      then Q <= "0000" ;
      elsif (H'event and H='1')
      then Q <= Q+1 ;
      end if ;
end process ;
end ARCH_COMPT_4 ;
```

2EME SYNTAXE : Utilisation de l'instruction WAIT ON et WAIT UNTIL

On utilise l'instruction Wait ON pour énoncer la liste de sensibilités

Process
BEGIN
Wait ON (liste sensibilités);
-- Instructions séquentielles;

END PROCESS

Pour déclencher sur un événement on utilise l'instruction suivante :

Wait until (CLK'event and CLK=1);