TUTORIAL ModelSim
VHDL

D. Chillet, E. Casseau

Le 14 novembre 2008

ENSSAT - Technopole Anticipa - BP 80518 - 6 rue de Kérampont - 22305 LANNION Cedex - Siret 193 509 361 00278 - APE 9215
Téléphone (33) 02 96 46 50 30 - Télécopie (33) 02 96 37 01 99 - Email ...@enssat.fr - Minitel 36 15 ENSSAT - Serveur WWW http://www.enssat.fr/

1 Tutorial ModelSim

Cette première section a pour objectif de vous faire prendre en main un compilateur et un simulateur pour le langage VHDL. Il s'agit du logiciel *ModelSim* de *MentorGraphics*. A l'Enssat, la version ModelSim SE 6.3ade cet outil est installé sur les machines de type PC.

Après un bref rappel concernant le langage *VHDL*, ce tutorial présente la manière de créer un projet ainsi que la façon de reprendre un projet en cours.
La première section de ce tutorial présente les différentes manipulations de l'outil ModelSim.
La deuxième section vous guide dans la conception d'une entité de base.

Dans la troisième section, nous listons les éléments de base que vous pourrez développer afin de compléter votre compréhension du langage et de l'outil.

Bonne lecture et bons développements ...

1.1 Rappel : Qu'est ce que le langage VHDL ?

VHDL est un langage de description matérielle. À la différence des langages informatiques classiques, VHDL ne vise pas une exécution, son but est de permettre la description de tout système électronique, d'en valider le fonctionnement avant de passer à la mise en œuvre matérielle.

La figure 1 illustre les différentes utilisations du langage VHDL.

‘ L'objectif de ce document concerne les aspects description et simulation. Les autres facettes du langage seront abordées par l'utilisation d'autres outils plus spécifiquement dédiés à la conception matérielle de circuits.

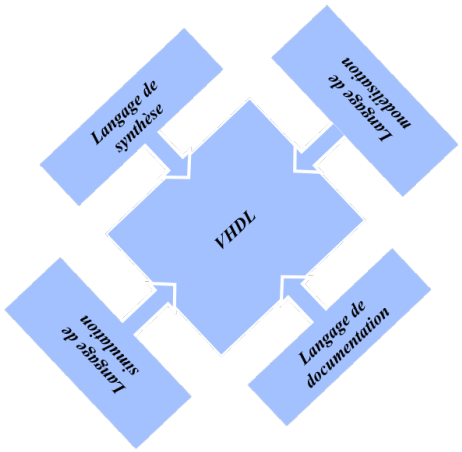

FIG. 1 – VHDL, à quoi ça sert ?

La conception d'un système passe par sa description. Cette description est toujours réalisée en deux étapes au minimum. La première étape consiste à décrire le système comme une *boite noire*, alors que la seconde s'intéresse à la description interne de la boite noire. Si la description de la vue externe (boite noire) ne pose généralement pas de problème, la vue interne (l'architecture) peut quant à elle être réalisée selon plusieurs modèles de description.
Rappelons brièvement les trois types de description utilisables en VHDL :
– description comportementale : il s'agit d'une description indiquant le comportement d'un système. Généralement réalisée sous la forme de processus, elle s'apparente à du code procédural classique ;
– description structurelle : il s'agit d'une description schématique d'un système. S'appuyant sur des composants disponibles dans une bibliothèque et sur des signaux. Cette description est l'exacte représentation du schéma électrique du système ;
– description flot de données : il s'agit d'une description indiquant comment un flot de données traverse un système. Le flot des sorties est exprimé en fonction du flot des entrées.

2 Manipulation de l'outil ModelSim

L'outil ModelSimest disponible sur les PC windows. Il peut être nécessaire de configurer l'accès au serveur de licence, pour cela référez à la note [1].
Une fois l'outil ModelSimlancé, vous pouvez soit reprendre un projet en cours soit créer un nouveau projet. Nous allons dans un premier temps expliquer les principales fonctionnalités de l'outil. Cette présentation sera ensuite accompagnée d'un petit guide qui vous permettra de compiler et simuler votre première entité.

2.1 Débuter un projet (initialisation du projet)

Lorsque vous débutez votre projet, vous devez configurer l'outil *ModelSim* afin qu'il organise correctement les différents fichiers que vous allez créer et manipuler.

1. À l'aide de *Windows*, créez un répertoire de travail sur votre espace disque `Z :/VHDL/Projet`
2. Dans ModelSim, modifier le répertoire de travail par le menu :
   File —> Change directory
   Sélectionnez le répertoire que vous venez de créer `Z :/VHDL/Projet`.

[1] Sur PC, avant de pouvoir lancer pour la première fois l'outil ModelSim, vous devez tout d'abord activer la licence, pour cela lancer le `Licensing wisard`, cliquez sur **Continue**, vérifiez que le serveur de licence est bien adressé `1763@mark`, validez puis répondez **Yes** à la fenêtre suivante. Une fois cette manipulation effectuée, vous pouvez lancer ModelSim à partir du menu "Démarrer".

3. Pour pouvoir stocker correctement vos unités de conception VHDL, vous devez créer une bibliothèque de travail :

   File —> New —> Library

   Laisser les noms `work` comme librairie de travail et physique, et cocher `a new library and a logical mapping to it`. C'est dans cette bibliothèque que seront placées toutes les entités compilées sans erreur.

4. Vous devez ensuite créer un projet :

   File —> New —> Project

   Tapez un nom de projet, par exemple `projet`, laisser coché `copy library mappings` et valider.

5. L'outil vous demande d'ajouter des `items` dans le projet. Pour l'instant, cliquer sur `Close`.

Vous pouvez ensuite passer au développement de votre système et aux compilations des différentes spécifications.

2.2 Reprise d'un projet

Lorsque vous reprenez votre travail sur un projet que vous avez déjà initialisé et sur lequel vous avez déjà travaillé, vous devez vous ouvrir le projet souhaité.

1. Après le lancement du logiciel *ModelSim*, placez vous dans votre répertoire de travail sur votre espace disque par le menu :

   File —> Change directory

   Sélectionnez le répertoire de travail `Z :/VHDL/Projet`.

2. Ouvrez votre projet :

   File —> Open —> Project

   Sélection du projet.

   Recherchez le fichier `.mpf` qui correspond à votre projet (normalement, ce fichier doit être dans le répertoire que vous avez sélectionné par le menu précédent).

2.3 La compilation

Le lancement du compilateur s'effectue par le menu :

Compile —> Compile | **Compile**

du logiciel. Une fenêtre s'ouvre alors et vous donne accès aux fichiers de votre répertoire de travail. Sélectionnez un fichier et cliquez sur **Compile.** La phase de compilation est alors activée.

2.3.1 Remarques

– Utilisez intelligemment les indentations et les commentaires pour que vos fichiers soient lisibles et facilement maintenables.
– Donnez l'extension **.vhd** à tous vos fichiers VHDL (fichiers de déclaration d'entités, d'architectures, de paquetages, de corps de paquetage, etc) ;
– Afin d'éviter une surabondance de fichiers dans votre répertoire de travail, nous vous conseillons de placer le corps d'un paquetage dans le même fichier que la spécification de ce même paquetage. De même, regroupez dans un même fichier, la déclaration d'entité (*entity, generic, port*) et son ou ses architectures.
– Écrivez un fichier par entité ou par paquetage.

2.3.2 Ordre de compilation de vos fichiers

L'ordre de la compilation de vos fichiers est important. Il faut toujours que toutes les unités utilisées (par l'unité à compiler) soient présentes dans la bibliothèque. Ce qui signifie que vous devez respecter l'ordre de compilation suivant :

1. compilation de vos paquetages. Si vous en avez plusieurs, compilez d'abord ceux qui n'utilisent aucun autre paquetage, ensuite vous compilerez les autres ;
2. compilation de vos entités feuilles. On entend par entité feuille, toute entité décrite de façon comportementale ;
3. compilation de vos entités décrivant un sous-système, c'est à dire des entités décrites sous la forme d'une structure de composants ;
4. compilation de votre entité système, c'est à dire de l'entité du plus haut niveau regroupant l'ensemble des sous-systèmes (reliés par des signaux) ;

2.3.3 Résultats de compilation

Dès que la compilation d'une unité est réalisée sans erreur, l'unité en question apparaît dans la bibliothèque de travail. Vous pouvez la visualiser en cliquant sur l'onglet `Library` du `workspace` ;
Dans cet onglet, vous voyez alors tous les paquetages et entités présents dans la bibliothèque `Work`, et pour chaque entité vous pouvez obtenir la liste de toutes les architectures décrites et compilées sans erreur. À partir de cette fenêtre, vous pouvez supprimer un paquetage ou une entité ou une architecture. Vous observerez que les paquetages `IEEE` sont présents dans la librairie.

2.3.4 Tests des unités réalisées

N'attendez pas que tout votre système soit réalisé pour effectuer des tests sur vos entités. La démarche idéale consiste à tester chaque entité dès que sa compilation est correcte. Pour cela, décrivez une entité de test. Compiler alors cette entité et lancer une simulation.

2.4 La simulation

2.4.1 Lancez une simulation

1. Pour lancer le simulateur, sélectionnez le menu **Simulate —> Start simulation**
2. Une fenêtre est alors ouverte et vous propose l'ensemble des unités présentes dans la bibliothèque. Sélectionnez une entité SIMULABLE (ou une configuration), c'est à dire une entité autonome qui instancie l'entité à simuler et qui gère les signaux d'entrée de l'entité à simuler. Les entités que vous allez développer seront stockées dans la librairie `Work` ;
3. Ouvrez ensuite les fenêtres de simulation par le menu **View —> Wave**
4. Par défaut, l'intervalle de simulation est de 100 ns, vous pouvez changer cette valeur en choisissant le menu :

   Simulate —> Runtime options
5. Pour visualiser les signaux de l'entité de test, sélectionnez `TestPorteET` dans l'onglet `sim` du `workspace`, cliquez sur **Add**, puis **Add to wave**. La fenêtre des chronogrammes doit alors faire apparaître la liste des signaux de l'entité.
6. Pour lancer un pas de simulation (de 100 ns), choisissez **Simulate —> Run —> Run**
7. Pour recommencer une simulation, choisissez **Simulate —> Run —> Restart**.

3 Conception d'une entité de base

Dans cette section, nous allons modéliser et simuler en VHDL un élément de base qui nous permettra de prendre en main l'outil ainsi que le langage.

3.1 Initialisation du projet

- À l'aide de *Windows*, créez un répertoire de travail sur votre espace disque `Z :/VHDL/Tutorial/`
- Choisissez le menu **File —> Change Directory** et sélectionnez le répertoire `Z :/VHDL/Tutorial` ;
- Choisissez le menu **File —> New —> Library** et tapez `WORK` (nom de la bibliothèque de travail dans laquelle toutes vos unités seront stockées) ;
- Choisissez le menu **File —> New —> Project** et tapez `tutorial` (nom du projet) ;

3.2 Compilation d'une unité fonctionnelle

- Créer un nouveau fichier VHDL dans votre projet **Project —> Add to project —> New File**. Saisir le nom de fichier `and`, le fichier sera créé avec l'extension `.vhd` et laissez l'option `Add file as type` sur `VHDL` et l'option `Folder` sur `TopLevel`.
- En double cliquant sur le nom du fichier dans le `workspace`, un éditeur de texte vous permettra de saisir la description de l'entité `PorteET`. Faite la saisie et sauvegarder ;

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

ENTITY PorteET IS
    PORT    (
             entree1  : IN  Std_Logic ;
             entree2  : IN  Std_Logic       ;
             sortie   : OUT  Std_Logic
    ) ;
END PorteET ;
```

- Dans ModelSim, choisissez le menu **Compile —> Compile selected** ;
- Lorsque la compilation est correcte, vous devez voir un message `Compile of <nom fichir vhdl> was successful` ;
- En double cliquant sur cette ligne, vous devez voir apparaître un fenêtre vous indiquant que les actions suivantes ont été réalisées :

```
vcom -work work -2002 -explicit -vopt Z:/VHDL/VONNEUMAN/AND.VHD
ModelTechnology ModelSim SE vcom 6.3a compiler 2007, 06 Jun 25 2007
# - Loading package standard
# - Loading package std_logic_1164
# - Compiling entity PorteET
```

- Visualisez le contenu de la librairie de travail dans l'onglet `Library` du `workspace`. Développez la librairie `Work`, vous devez voir apparaître l'entité `PorteET` dans cette librairie ;
- Créer un nouveau fichier VHDL qui contiendra la description de l'architecture de l'entité. Pour cela, **Project —> Add to project —> New File**.
  Saisissez le nom de fichier `ArchCompAnd`, le fichier sera créé avec l'extension `.vhd` et laissez l'option `Add file as type` sur `VHDL` et l'option `Folder` sur `TopLevel`. Dans ce fichier, placez le code VHDL suivant :

```
LIBRARY IEEE ;
```

```
USE IEEE.STD_LOGIC_1164.ALL ;

ARCHITECTURE Comportementale OF PorteET IS
BEGIN
    ProcessPorteET : PROCESS (entree1, entree2)

         BEGIN
             sortie <= entree1 AND entree2 AFTER 1 ns;
         END PROCESS ProcessPorteET ;
END Comportementale ;
```

– Sélectionnez le fichier `ArchCompAnd.vhd` dans le `workspace` puis cliquez droit et choisissez **Compile —> Compile selected** ;
– Lorsque la compilation est correcte, vous devez voir un message `Compile of <nom fichir vhdl> was successful` ;
– En double cliquant sur cette ligne, vous devez voir apparaître un fenêtre vous indiquant que les actions suivantes ont été réalisées :

```
 vcom Z:/VHDL/VONNEUMAN/ARCHCOMPAND.VHD
 ModelTechnology ModelSim SE vcom 6.3a compiler 2007, 06 Jun 25 2007
# - Loading package standard
 # - Loading package std_logic_1164
 # - Compiling architecture Comportementale of PorteET
 # - Load entity PorteET
```

– Visualisez le contenu de la librairie de travail dans l'onglet `Library` du `workspace`. Développez la librairie `Work`, vous devez voir apparaître l'entité `PorteET` et l'architecture `Comportementale` dans cette librairie ;

3.3 Simulation de l'unité fonctionnelle

Pour parvenir à tester / simuler / valider notre entité `PorteET` il faut décrire un *testbench*. Le *testbench* doit instancier un composant de *type* `PorteET` et faire évoluer les signaux d'entrée de la porte ET.

– Décrivez une entité de test pour la porte ET `TestAnd` ainsi qu'une architecture pour cette entité. L'architecture instanciera une porte ET et fera évoluer les signaux d'entrées de la porte ET. Stocker le tout dans le fichier `TestAnd.vhd` ;

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

ENTITY TestPorteET IS
END TestPorteET ;

ARCHITECTURE Test OF TestPorteET IS

        COMPONENT PorteET
             PORT    (
                 entree1  : IN  Std_Logic ;
                 entree2  : IN  Std_Logic ;
                 sortie   : OUT  Std_Logic
         ) ;
        END COMPONENT;

        SIGNAL s_entree1 : Std_Logic := '0';
```

```
        SIGNAL s_entree2 : Std_Logic := '0';
        SIGNAL s_sortie : Std_Logic := 'Z';

BEGIN
        porte : PorteET
                PORT MAP (s_entree1, s_entree2, s_sortie);

        ProcessSimulation : PROCESS
        BEGIN
                WAIT FOR 10 ns;

                s_entree1 <= '0';
                s_entree2 <= '0';
                WAIT FOR 10 ns;

                s_entree1 <= '0';
                s_entree2 <= '1';
                WAIT FOR 10 ns;

                s_entree1 <= '1';
                s_entree2 <= '1';
                WAIT FOR 10 ns;

                s_entree1 <= '1';
                s_entree2 <= '0';
                WAIT FOR 10 ns;

                s_entree1 <= '0';
                s_entree2 <= '0';
                WAIT FOR 10 ns;

                WAIT ;
        END PROCESS ProcessSimulation ;
END Test;
```

- Sélectionnez le fichier `TestAnd.vhd` dans le `workspace` puis cliquez droit et choisissez **Compile —> Compile selected** ;
- Lorsque la compilation est correcte, vous devez voir un message `Compile of <nom fichir vhdl> was successful;`
- En double cliquant sur cette ligne, vous devez voir apparaître un fenêtre vous indiquant que les actions suivantes ont été réalisées :

```
vcom Z:/VHDL/VONNEUMAN/TESTADD.VHD
ModelTechnology ModelSim SE vcom 6.3a compiler 2007, 06 Jun 25 2007
# - Loading package standard
# - Loading package std_logic_1164
# - Compiling entity TestPorteET
# - Compiling architecture test of TestPorteET
```

- Lancez ensuite la simulation **Simulate —> Start simulation**.
- Choisissez alors l'entité `TestPorteET` disponible dans la librairie `work` ;
- Demandez l'affichage de la vue `wave`, pour cela, choisissez **View —> Wave** ;

– Ajoutez ensuite les signaux de l'entité de test, pour cela, choisissez
View —> Objects
– Ajouter des signaux dans la fenêtre d'observation des chornogrammes (wave), pour cela, sélectionnez un signal dans la fenêtre *Objects* et cliquez droit sur ce signal, puis
Add to wave —> Selected signal.
– Vous pouvez aussi ajouter d'un coup tous les signaux de l'entité de test en choisissant
Add to wave —> Signals in region ;
– Fnalement vous pouvez aussi afficher tous les signaux de tout le système (toutes les profondeurs de la hiérarchie), en sélectionnant
Add to wave —> Signals in design ;
– Vérifiez le bon fonctionnement de l'entité à l'aide des chronogrammes. Pour cela lancez la simulation. Celle-ci se lance par pas de $100ns$ (valeur par défaut qui est modifiable dans la barre de boutons). Pour lancer une simulation, choisissez
Simulate —> Run ;

4 Liste des éléments à concevoir

Procédez comme indiqué dans la section précédente pour modéliser et simuler les éléments suivants :
– Porte OU ;
– A l'aide des deux entités précédemment réalisées, décrivez une entité structurelle assurant le calcul `S = A . B + C;`
– Porte inverseuse ;
– Elément de mise en haut impédance ;
– Multiplexeur 2 vers 1 ;
– Multiplexeur N vers 1 ;
– Décodeur 1 parmi 8 ;
– Décodeur 1 parmi N ;
– Registre 1 bit ;
– Registre N bits ;