Franck Paugnat
CIME Nanotech
franck.paugnat@minatec.grenoble-inp.fr

Mini guide de prise en main de ModelSim

4 novembre 2013

Sommaire

Objet

ModelSim est un outil de CAO de Mentor Graphics destiné à la simulation et la vérification pour des descriptions matérielles en VHDL, Verilog, SystemVerilog, SystemC ou une combinaison de ces différents langages.

Le présent document indique les principales commande de l'outil ModelSim pour commencer à travailler sur un projet dont les modules sont décrits en VHDL, et uniquement dans ce langage de description matériel. Il s'agit plus d'un pense-bête que d'un guide de l'utilisateur ou un tutoriel.

Pour obtenir de l'aide sur ModelSim, consulter les manuels suivants (de préférence dans l'ordre indiqué) :

- ModelSim® SE Tutorial
- ModelSim® SE User's Manual
- ModelSim® SE Reference Manual

Ces documentations au format PDF sont disponibles dans le répertoire "`/softslin/modelsim10_1d/modeltech/docs/pdfdocs/`"

Le moyen le plus simple d'accéder à ces fichiers d'aide est d'ouvrir le fichier "`/softslin/modelsim10_1d/modeltech/docs/pdfdocs/_bk_modelsim_se.pdf`" puis de cliquer les liens correspondants.

L'aide est également accessible depuis l'interface graphique menu

- en HTML, Menu « Help | SE Documentation – InfoHub »
- en PDF, Menu « Help | SE Documentation - PDF bookcase »

I) Principes de base et définitions

ModelSim utilise un fichier d'initialisation pour enregistrer les paramètres et personnalisations requis pour votre projet. Ce fichier est appelé `modelsim.ini`. Son emplacement est laissé libre. Son emplacement par défaut est connu par ModelSim grâce à la variable d'environnement `MODELSIM`. Cette variable est définie dans le script de configuration qui est à sourcer[1] avant de lancer d'utiliser l'outil ModelSim.

Chaque module au sens VHDL, c'est-à-dire entité plus architecture(s), est compilée dans une bibliothèque par ModelSim afin de pouvoir être simulé. Une bibliothèque est désignée dans le code VHDL par un nom logique et est associé à un ensemble de fichiers consignés dans un répertoire.

Bibliothèque physique :

Répertoire sur le disque dur regroupant les fichiers des modules compilés

Nom logique d'une bibliothèque :

Nom de bibliothèque utilisés dans les fichiers source VHDL (instruction "library")

Avant d'effectuer une simulation, il faudra donc au préalable procéder aux étapes suivantes :

- créer une bibliothèque physique
- associer un nom logique à la bibliothèque physique
- compiler les modules requis dans la bibliothèque en la désignant par son nom logique.

Les commandes pour effectuer ces opérations sont indiquées ci-après.

La bibliothèque par défaut est appelée `work` et son utilisation n'a pas besoin d'être déclarée dans un fichier source VHDL. Cependant, afin de mener un travail structuré et réutilisable, il est indispensable de répartir les modules du projets dans différentes bibliothèques en les regroupant par unité fonctionnelle ou type de traite ment.

1 Dans le contexte d'une utilisation des outils de CAO dans un environnement Linux, l'expression « sourcer un script » désigne (par contamination du nom de la commande) le lancement du script à l'aide de la commande Linux `source` plutôt que par une exécution simple sur la ligne de commande.

II) Principales commandes de ModelSim

Les commandes indiquées ici peuvent être exécutées directement à la console ou au sein d'un script. Ces commandes possèdent de nombreuses options. Seules celles les plus courantes sont données ici. Le lecteur se reportera au manuel de référence de ModelSim pour connaître en détail les options des commandes.

Remarque : pour les commandes prenant un chemin en argument, il est conseillé d'encadrer le chemin d'une paire d'accolades de façon à ce que les noms de chemin contenant des espaces soient correctement interprétés. Les accolades peuvent être omises si le nom du chemin ne contient ni espace ni caractère d'échappement (caractère précédé d'un '\').
De même pour les chemins construits avec une variable d'environnement, il est recommandé d'encadrer le nom de la variable d'une paire d'accolades afin de garantir un développement correct de cette variable.
Ex : vmap LIB_Filtre {${TP_PATH}/Mes bibliothèques/libMonFiltre}

Remarque : les commandes et leurs options sont sensibles à la casse (majuscules et minuscules sont distinguées).

II.1) Création, association, suppression des bibliothèques

Pour illustrer le propos, nous supposerons que nous voulons créer et utiliser une bibliothèque ayant pour nom logique `LIB_FiltreComport` et dont la bibliothèque physique (le répertoire) est `libs/libFiltreComport`.

Créer une bibliothèque physique : commande `vlib`

```
vlib libs/libFiltreComport
```

Créer un nom logique et l'associer à une bibliothèque physique : commande `vmap`

```
vmap LIB_FiltreComport ${TP_PATH}/libs/libFiltreComport
```

La commande `vmap` modifie le fichier d'initialisation `modelsim.ini`. Elle connait l'emplacement du fichier `modelsim.ini` grâce à la variable d'environnement MODELSIM définie dans le fichier de configuration.

Il est possible d'indiquer à la commande `vmap` un fichier `modelsim.ini` différents de celui par défaut, désigné par la la variable MODELSIM, en utilisant l'option `-modelsimini <chemin/modelsim.ini>`

Il est tout à fait possible d'associer plusieurs noms logiques à une même bibliothèque physique.

Connaitre la bibliothèque physique associée à un nom logique donné

Pour connaitre la bibliothèque physique associée à un nom logique donné, il suffit d'évoquer la commande `vmap` suivie du nom de la bibliothèque logique mais sans nom de répertoire

```
vmap LIB_FiltreComport
```

Changer la bibliothèque physique associée à un nom logique

Pour changer la bibliothèque physique (répertoire) associée à un nom logique, il suffit de exécuter à nouveau la commande `vmap` avec le même nom logique mais en donnant le nom de la nouvelle bibliothèque physique

```
vmap LIB_FiltreComport ${TP_PATH}/libs/libComportNouvelle
```

Dissocier nom logique et bibliothèque physique

Pour rompre l'association entre un nom logique et une bibliothèque physique, utiliser la commande `vmap` suivie de l'option `-del` et du nom de la bibliothèque logique.

```
vmap -del LIB_FiltreComport
```

Le bibliothèque physique existe toujours, son répertoire et ses fichiers demeurent sur le disque dur.

Remarque : Une gestion astucieuse des associations entre nom logique et bibliothèque physique permet de changer de version de bibliothèque sans modifier les fichiers source VHDL.

Supprimer partiellement ou totalement une bibliothèque physique : commande `vdel`

ATTENTION cette commande supprime des fichiers sur le disque dur sans demander de confirmation. Son **action** est **irréversible**.

L'option `-lib` indique la bibliothèque sur laquelle la commande `vdel` doit agir. La bibliothèque physique peut être désignée par le chemin de son répertoire sur le disque dur ou par le nom logique qui lui est associé. Dans ce dernier cas, le chemin du répertoire est lu dans le fichier `modelsim.ini`. En l'absence de l'option `-lib`, la commande `vdel` agit sur la bibliothèque `work`.

L'option `-all` commande de supprimer entièrement la bibliothèque physique. Le répertoire de la bibliothèque est alors effacer du disque dur. Dans le cas de notre exemple où le nom logique `LIB_FiltreComport` est associé à la bibliothèque physique `${TP_PATH}/libs/libFiltreComport`, les deux commandes suivantes auront le même résultat : suppression du répertoire `${TP_PATH}/libs/libFiltreComport`.

```
vdel -lib LIB_FiltreComport -all
vdel -lib ${TP_PATH}/libs/libFiltreComport -all
```

La commande `vdel` sert aussi à enlever d'une bibliothèque, sans la supprimer, un module complet (entité et toutes ses architectures) ou seulement une architecture d'un module .

En supposant que la bibliothèque `LIB_FiltreComport` contiennent un module `Buffer` qui possède une architecture `behavioural`, la commande

```
vdel -lib LIB_FiltreComport Buffer behavioural
```

retirera l'architecture de `behavioural` de la bibliothèque `LIB_FiltreComport` tout en conservant l'entité `Buffer` et ses éventuelles autres architectures. Tandis que la commande suivante

```
vdel -lib LIB_FiltreComport Buffer
```

supprimera l'entité `Buffer` et toutes ses architectures de la bibliothèque `LIB_FiltreComport`.

II.2) Compilation et Simulation

Avant de pouvoir accomplir une simulation, il fat compiler les modules et les placer dans une bibliothèque. Ensuite le simulation doit être chargé avec le projet et enfin la simulation déroulée.

Compilateur VHDL : `vcom`

Le compilateur VHDL de ModelSim est appelé `vcom`.

L'option `-work` permet de spécifier le nom de la bibliothèque logique qui recueillera les unités de conception compilée.

L'option `+acc` est une option nécessaire au débogage. Elle donne l'accès aux variables, constantes et alias utilisés dans les *process* qui ne sont pas accessibles sans cette option du fait des optimisations de vitesse de simulation. L'ajout de cette option ralentit donc la simulation mais donne plus de visibilité lors de la simulation.

Exemple : compiler les entités définies dans le fichier source `MonModule.vhd` (lui-même placé dans le sous-répertoire `src`) en gardant l'accès aux variables utilisées dans les processus et les placer dans la bibliothèque logique `LIB_FiltreComport`

```
vcom +acc -work LIB_FiltreComport src/MonModule.vhd
```

Remarque : les unités ne sont ajoutées à la bibliothèque que si la compilation s'est déroulée sans erreur.

Dans le cas où il y a plusieurs fichiers source, il est possible de les spécifier sur une même ligne de commande

```
vcom -work LIB_FiltreComport src/MonModule.vhd src/fichier1.vhd src/fichier2.vhd …
```

Toutefois répartir la compilation sur plusieurs commandes `vcom` enchainé successivement peut améliorer la lisibilité des script de compilation (une commande par ligne, un fichier par commande).

```
vcom -work LIB_FiltreComport src/MonModule.vhd
vcom -work LIB_FiltreComport src/fichier1.vhd
vcom -work LIB_FiltreComport src/fichier2.vhd
…
```

Simulateur `vsim`

Le simulateur est démarré simplement en tapant la commande `vsim`.

Pour démarrer le simulateur avec un projet directement chargé, il suffit d'indiquer le nom du module englobant (*top-level module*) en paramètre. Spécifier la bibliothèque contenant ce module englobant à l'aide de l'option `-lib`

```
vsim -lib LIB_FiltreComport MonFiltre_top
```

Si le projet utilise des unités dans des bibliothèques autres que les bibliothèques compilées pour le projet courant (cf. `vcom`) par la directive `uselib`, ces bibliothèques supplémentaires doivent être listées sur la ligne de commande en faisant le nom logique de chacune des bibliothèques de l'option -L. Le nom du module englobant doit être le dernier argument de la ligne de commande.

```
vsim -lib LIB_FiltreComport -L LIB_QuelqueChose -L LIB_AutreChose MonFiltre_top
```

La recherche des instances se fait dans l'ordre de la liste des bibliothèques (donc de gauche à droite sur la ligne de commande). Dans le cas d'un projet utilisant des cellules ayant le même nom mais appartenant à des bibliothèques différentes (fréquemment le cas pour les portes logiques) , il faut faire précédé la liste des bibliothèques avec l'option `-L work` afin de résoudre les conflits. Avec cette option, le simulateur cherche l'instance de la cellule dans la bibliothèque dans laquelle a été compilé le modules qui instancie cette cellule et non plus dans la première bibliothèque de la liste qui contient une instance de ce nom.

```
vsim -lib LIB_FiltreComport -L work -L LIB_QuelqueChose -L LIB_AutreChose MonFiltre_top
```

Dérouler une simulation : commande `run`

La façon la plus commune d'utiliser la commande `run` est d'indiquer une durée de simulation en paramètre. Les unités de temps autorisées sont fs, ps, ns, us, ms, et sec (pour les secondes). Il est possible d'indiquer à la simulation de se dérouler jusqu'à un temps données en faisant précéder le paramètre d'une arobase @.

`run 10.4 ms` => avance la simulation d'une durée de 10,4 ms

`run @20.4 ms` => avance la simulation jusqu'au temps simulé 20,4 ms

`run 8 ms` => avance la simulation d'une durée de 8 ms à partir du temps courant simulé issue de la précédente exécution de la commande `run`. La simulation s'arrête donc quand le temps simulé atteint 28,4 ms.

Il est également possible d'avancer la simulation jusqu'au prochain évènement avec l'option `-next`

```
run -next
```

Contrôler l'affichage des courbes : commandes `add wave` et `wave`

Pour définir les signaux à faire apparaitre dans la fenêtre des courbes (*wave window*) choisir le format d'affichage, sélectionner la couleur des courbes, définir des zoom ou placer des curseur automatiquement à l'aide script, se reporter aux commandes `add wave` et `wave` dans le manuel de référence ModelSim® SE Reference Manual.

III) Interface graphique

L'interface graphique de ModelSim est très bien présentée au chapitre 2 « Graphical User Interface » du manuel utilisateur ModelSim® SE User's Manual.

Son utilisation bien expliquée dans le tutoriel de ModelSim. Se reporter notamment aux chapitres suivants du document ModelSim® SE Tutorial :

- Chapter 3 Basic Simulation
- Chapter 4 Projects
- Chapter 5 Working With Multiple Libraries
- Chapter 7 Analyzing Waveforms