Plan du tutorial sur SystemC Compiler

1 - Introduction

Ce tutorial a pour objet de montrer par un exemple simple (i.e. compteur/décompteur 4 bits) comment les outils du logiciel Synopsys permettent d'utiliser le langage SystemC comme n'importe quel HDL.

Ce tutorial présuppose que le logiciel Synopsys a été installé conformément au tutorial sur l'utilisation du VHDL avec Synopsys.

2 - Configuration

Un fichier tutorial_SystemC.tar comprend tous les fichiers nécessaires pour le bon fonctionnement des commandes expliquées dans ce tutorial.

compteur.cpp le fichier d'implémentation du module
compteur.h le fichier de déclaration du module

display.h le fichier de description de la sonde permettant de lire les résultats de simulation sur un terminal
main.cpp le fichier de simulation du module logique.

Make_compteur le makefile du compteur
makefile.defs un script pour que le makefile interprète du SystemC (à vérifier pour être sur que SystemC est installé au bon endroit
run le script de compilation pour la simulation (lancer **run compteur**)

cp_comteur le script de compilation et de synthèse du compteur pour **dc_shell**
sim_compteur.vhd le fichier VHDL contenant le benchmark et la configuration pour la simulation après synthèse.
umc18_typ_dc.setup le script de description de la bibliothèque de synthèse (inclus dans .synopsys_dc.setup)
.synopsys_dc.setup le fichier de configuration de **Design Compiler** et de **VHDL Analyzer**
.synopsys_vss.setup le fichier de configuration du simulateur VHDL

Décompressez le fichier tutorial_SystemC.tar dans le répertoire de travail et créez y deux répertoires

- **lib** : il contiendra par la suite les fichiers .sim et .mra générés par **vhdlan.**
- **rtl_work** : il contiendra les rapports et les différents fichiers générés par **dc_shell**

Vérifiez que vous avez bien les permissions nécessaires pour l'utilisation des fichiers (sinon : **chmod *commande nom_de_fichier***)

Vous êtes prêt à travailler.

3 - Simulation du modèle

La simulation d'un modèle SystemC peut se faire par deux méthodes :
- rediriger les signaux vers la sortie du terminal en utilisant les flux C++,
- tracer un chronogramme des signaux.

L'utilisation des chronogrammes est conseillée dans le cas de processus synchrones dont le fonctionnement temporel est à vérifier alors que la redirection vers la sortie du terminal est plus appropriée pour un fonctionnement asynchrone ou pour mettre l'accent sur un problème particulier.

3 - 1 Chronogrammes

On obtient les chronogrammes en exécutant :

run compteur -- ceci compile le modèle et crée le fichier compteur.ow

waves -- on lance le visionneur de chronogrammes (celui de vhdldbx)

File->Open->compteur.ow -- on ouvre le fichier contenant les résultats

Vous pouvez ensuite voir le résultat de la simulation comme pour n'importe quelle simulation VHDL.

La partie de **main.cpp** qui concerne le tracé des chronogrammes est la suivante :

main.cpp

```
/* création du fichier externe contenant les chronogrammes */
sc_trace_file *CPT_trace;
CPT_trace = sc_create_wif_trace_file(DESIGN);

/* définition des signaux à tracer */
sc_trace(CPT_trace,CLK,"clk");  // sc_trace(fichier,signal,nom)

sc_trace(CPT_trace,RESET,"Reset");
sc_trace(CPT_trace,UP,"Up");
sc_trace(CPT_trace,LOAD,"Load");
sc_trace(CPT_trace,VAL,"Val");
sc_trace(CPT_trace,OUT,"Out");

/* simulation */
/* fin de simulation */

/* fermeture du fichier */
sc_close_wif_trace_file(CPT_trace);
```

Remarque : le fichier est exporté ici sous le format WIF. Vous pouvez exporter sous d'autres formats (par exemple VCD) si vous disposez du visionneur approprié.

3 -2 Flux C++

Un module supplémentaire a été défini dans le fichier **display.h**. Il permet de retourner une liste de signaux avec leurs états vers la sortie du terminal selon sa liste de sensibilité.

```
/* display.h*/

SC_MODULE(DISPLAY)
{
        sc_in<bool> reset;          //entrées de la sonde
        sc_in<bool> clk;
        sc_in<bool> up;
        sc_in<bool> load;
        sc_in<sc_uint<4> > val;
        sc_in<sc_uint<4> > out;

    void sonde()
    {
       int tmp = val.read();              //conversion de type pour l'affichage
       printf("clk = %d",clk.read());
       printf("  reset = %d",reset.read());
       printf("  up = %d",up.read());
       printf("  load = %d",load.read());
       printf("  val = %d",tmp);
       tmp = out.read();                  //conversion de type pour l'affichage
       printf("  out = %d\n",tmp);
    }

    SC_CTOR(DISPLAY)
    {
       SC_METHOD(sonde);
       sensitive_pos<<clk;                        //sensibilités
       sensitive<<reset<<up<<load<<val<<out;
    }
};
```

Le branchement de ce module supplémentaire se fait comme suit dans **main.cpp**.

```
/*main.cpp*/

DISPLAY D("Affichage flux");                    // declaration du module comme variable
D<<RESET<<CLK<<UP<<LOAD<<VAL<<OUT;   // branchement des signaux
```

La redirection des flux, avec les fichiers du tutorial, se fait automatiquement. Il suffit de lancer une simulation/compilation par **run compteur**. Pour des raisons pratiques, le résultat est redirigé dans le fichier **test.txt**.

Remarque : lorsqu'on utilise la deuxième méthode de simulation, la taille des résultats est souvent trop importante pour être lue dans une fenêtre simple du terminal et est souvent redirigée dans un fichier pour des commodités de lecture, comme c'est le cas ici.

4 - Synthèse RTL

Tous les fichiers SystemC ne sont pas synthétisables, en particulier ici le fichier **display.h**. Seuls les fichiers **compteur.h** et **compteur.cpp** de ce tutorial sont synthétisables.

La synthèse SystemC se fait à partir du shell de Design Compiler (dc_shell). Il faut ensuite entrer un suite d'instructions permettant l'analyse, la vérification et la synthèse du module.

On obtient en général après ce passage un ficher .db (netlist Synopsys) et un fichier VHDL utile pour une simulation après synthèse.

Procédure de synthèse :

dans une fenêtre de terminal, lancer **dc_shell**.

depuis l'invite `dc_shell` : **include cp_compteur**

Le résultat sera normalement les netlist générées et les rapports de taille et de *timing*.

Le script *cp_compteur* est décrit ci après :

```
compile_systemc -rtl -cpp "g++ -E" compteur.cpp -rtl_format db;
      -- elabore le design du compteur
uniquify;
      -- elimine toutes les instances du design courant
      -- dans la hierarchie en creant un design unique
      -- pour chaque instance, inutile dans le cas d'un
      -- design sans hierarchie comme ici

check_design;
      -- verification de la validite logique du design
      -- (en cas d'erreurs presence de warning)

create_clock clk -period 20;
      -- definition des contraintes ; ici la periode
      -- de l'hologe. Voir les autres commandes dans la
      -- documantation du dc_shell

compile  -map_effort high;
      -- optimisation du design
      -- rem: il y a 3 modes "d'effort de compilation":
      -- low, medium et high

write -hier -f db -o uc.db;
      -- sauvegarde de la netlist sous forme de fichier DB
      -- pour la reutilisation dans d'autres designs

change_names -rules vhdl;
      -- change le mode d'ecriture des noms pour une sortie
      -- ulterieure en vhdl

write -f vhdl -hier -o compteur_c.vhd;
      -- sauvegarde de la netlist sous forme de fichier VHDL
      -- pour les test apres synthese
```

Une fois ce script exécuté, votre design est synthétisé et il ne reste plus qu'à le simuler.

5 - Simulation RTL.

La simulation du module synthétisé se fait à partir du fichier **compteur_c.vhd** issu de la compilation de dc_shell.

Ce fichier est une transcription de la netlist et est donc difficile à comprendre mais il se simule comme n'importe quel fichier VHDL issu de simulation.

Un fichier VHDL a besoin d'un *BenchMark* et d'une configuration pour pouvoir être simulé. Tout ce qui est nécessaire est présent dans le fichier **sim_compteur.vhd**.

Il suffit alors d'analyser les fichiers compteur_c.vhd et sim_compteur.vhd pour ensuite lancer la simulation, ce qui se fait comme suit :

```
vhdlan -nc compteur_c
vhdlan -nc sim_compteur
vhdldbx
```

Vous pouvez procéder ensuite comme pour n'importe quelle simulation vhdl.
L'utilisation de **design_analyzer** et des autres outils est possible mais la compilation et l'optimisation doit se faire par **dc_shell**, la seule partie de Synopsys compatible avec SystemC.

Remarque : L'écriture de fichiers VHDL de simulation peut se faire comme si vous testiez le même modèle écrit en VHDL. Si vous comparez **sim_compteur.vhd** au fichier de simulation du compteur du Tutorial Synopsys VHDL, la seule différence se situe au niveau du PORT MAP qu'il faut souvent aller chercher dans la netlist car la transformation de types SystemC en types VHDL est parfois imprévisible.