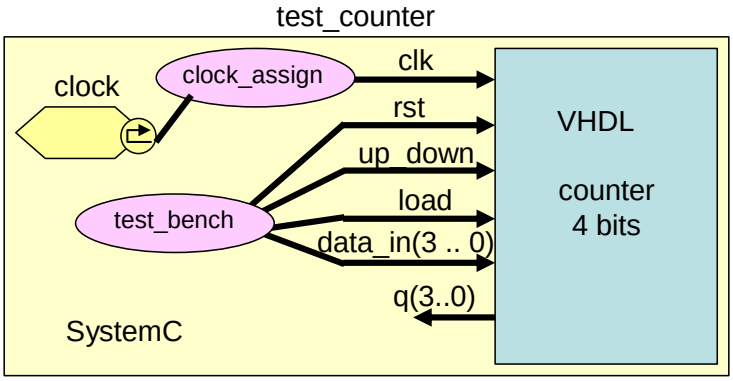
1 - De VHDL à SystemC

1.1 Simulation VHDL/SystemC sous ModelSim

L'objectif est de simuler du code VHDL et du code SystemC sous ModelSim. L'exemple à concevoir est présenté ci-dessous :



a) Créer un projet sous ModelSim

b) Ecrire le compteur « counter » en VHDL dont les caractéristiques sont :
- Le compteur réagit sur front montant
- Le reset est asynchrone
- Si load = '1' alors q = data_in
- Si up_down = '1' alors q = q + 1 autrement q = q – 1
- Les signaux clk,rst,up_down et load sont du type std_logic.
- Les signaux data_in et q sont du type std_logic_vector(3 downto 0)

c) Créer un fichier de commande run.do (File->New->Source->Do). Dans ce fichier, saisir les lignes ci-dessous :

```
# create library
if [file exists work] {
    vdel -all
}
vlib work

# compile the VHDL source files
vcom -93 *.vhd

# generate foreign module declaration
scgenmod counter > counter.h
```

d) A partir de l'entité counter, il faut générer un module counter (héritant de sc_foreign_module) qui fera le lien avec SystemC. La dernière ligne du fichier « run.do » permet de créer ce module. Ainsi, dans la console de commande, saisir la ligne suivante :

```
> do run.do
```

Visualiser le fichier créé « counter.h »

e) Ecrire le module « test_counter » en SystemC
- Ecrire le fichier « test_counter.h » suivant la figure précédente

Attention, l'instanciation du module *counter* a 2 paramètres : le nom de l'instance et la library VHDL où le fichier vhdl a été compilé. Par défaut, c'est dans la librairie work. L'instanciation sera la suivante : *counter("counter_inst", "work.counter")*

– Ecrire le fichier « test_counter.cpp ». Dans ce fichier, il faut mettre au début la ligne suivante qui est propre à ModelSim

```
SC_MODULE_EXPORT(test_counter);
```

f) Ajouter dans le fichier "run.do" les lignes pour la compilation

```
# compile and link C source files
sccom -g test_counter.cpp
sccom –link
```

g) Lancer de nouveau le fichier « run.do » et corriger les erreurs éventuelles

h) Ajouter dans le fichier « run.do » les lignes pour la simulation

```
# start and run simulation
vsim work.test_counter

# Display Wave
add wave *

#Run simulation
set StdArithNoWarnings 1
run 30 ms
```

i) Avant de relancer le « run.do », il faut arrêter la simulation autrement une erreur d'accès au fichier systemc.so est générée (Menu : Simulate->End of Simulation » ou tapez la commande « `quit -sim` »).

j) Ajout le MEMORY_RTL

– Ajouter dans le module « test_counter » votre MEMORY_RTL en connectant le bus d'adresse à la sortie q du compteur.

– Ajouter un autre process « test_memory » qui permet de piloter les autres signaux de la mémoire

1.2 Mémoire

Définir une mémoire RAM synchrone dont les caractéristiques sont données ci-dessous :

- Une entrée horloge synchrone sur front descendant (type booleen)
- une entrée enable (type booleen)
- une entrée rd (lecture) (type boolean),
- une entrée we (écriture) (type booleen)
- une entrée adresse de longueur générique ADDR_SIZE (type sc_uint< ADDR_SIZE>)
- une entrée de donnée (data_in) et une sortie de donnée (data_out) de longueur générique WORD_SIZE (type sc_lv< WORD_SIZE>)
- une profondeur générique (MEM_SIZE)

Travail Demandé

1) Définir le schéma du module mémoire « RTL_memory » avec ses E/S, process
2) Décrire dans un fichier « .h » le module complet (pas de .cpp à cause de la généricité)
3) Définir un module de test « test_memory »

4) Ecrire le programme principal (main) qui capture les signaux clk, en, rw, addr et data pour les sauvegarder dans un fichier au format VCD.

5) L'affichage du fichier VCD peut se faire en utilisant ModelSim. Créer un fichier « view.do » qui vous permettra de transformer un fichier VCD en WLF (format Modelsim) et d'afficher les waveform.

```
echo "SystemC VCD 2 WLF : " $1
echo "Example : open wave.vcd"
echo "> do view.do wave"

vcd2wlf $1.vcd $1.wlf
dataset open $1.wlf $1
add wave *
wave zoomfull
```