Contrôle de connaissances SE207 “SystemC”

28 juin 2017

Instructions

Ce contrôle de connaissances est strictement individuel. Vous devez modifier ce fichier pour y inclure vos réponses puis l'ajouter à votre dépôt dans un dossier `CC` à la racine de ce dernier.

Seules les parties entre `---` sont à modifier.

Questions

Question 1

En SystemC, pour modéliser des calculs sur des entiers signés, plusieurs types peuvent être utilisés. Donnez la liste de ces types (les grandes familles) en expliquant dans quels cas il est préférable d'utiliser l'un plutôt que l'autre.

Ceci est un exemple de réponse. ***Merci d'effacer ce paragraphe*** *(mais de* ***laisser les groupes de trois tirets et les lignes vides avant et après eux****) lorsque vous y écrirez la vôtre.*

- X ceci est X
- Y ceci est Y

```
// ceci est un exemple de code
int main() {
   return 0;
}
```

Question 2

Pourquoi peut-on connecter *directement* la sortie (`sc_out<>`) d'un module à la sortie d'un autre module mais pas à une entrée (`sc_in<>`) ?

Question 3

- Que se passe-t-il si une boucle infinie existe dans une `SC_METHOD` ?
- Que se passe-t-il si la fonction `wait()` est appelée dans cette boucle infinie ?

Question 4

Nous voulons modéliser un bloc matériel synchrone (à une horloge `clk`) dans lequel une étape de traitement doit attendre la fin d'une autre étape avant de commencer.

Pour ce fait, nous utilisons deux `SC_THREAD`, sensibles au front montant de l'horloge, pour modéliser chaque étape.

Dans une première implémentation, nous utilisons un `sc_mutex` pour synchroniser les de `SC_THREAD`.

```
// Thread 1
// étape 1 du traitement
step1_end_mutex.lock();
…
step1_end_mutex.unlock();
wait();


// Thread 2
wait();
// attente de la fin de l'étape 1
step1_end_mutex.lock();
// passage à l'étape suivante
…
```

Dans une seconde implémentation nous utilisons un `sc_signal` dont nous examinons la valeur à chaque front de l'horloge.

```
// Thread 1
// étape 1 du traitement
end_step1 = false;
…
end_step1 = true;
wait();

// Thread 2
wait();
// attente de la fin de l'étape 1
while(!end_step1) wait();
// passage à l'étape suivante
…
```

- Expliquez brièvement le fonctionnement.
- En justifiant votre réponse :
  - Voyez-vous des différences en termes de ressources utilisées pour la simulation (temps de calcul, mémoire…) ?
  - Voyez-vous des différences quant à la précision temporelle des deux implémentations ?

Question 5

- Pouvons-nous modéliser au niveau RTL en utilisant un `SC_THREAD` ?
- *Justifiez votre réponse*
