Modélisation Transactionnelle des Systèmes sur Puces en SystemC Ensimag 3A — filière SLE Grenoble-INP

Notions Avancé en SystemC/TLM

Matthieu Moy

Matthieu.Moy@imag.fr

2012-2013

Matthieu Moy (Matthieu.Moy@imag.fr)

Modélisation TLM

Sommaire

- Bug, or not Bug?
- Optimisations de Performances
- Questions de Sémantique
- Évaluation de consommation



A Few Kinds of Model/Simulator's Bugs

- Hardware bug :
 - Simulator design bug, corresponding to a real hardware bug
 - Simulator programming error, irrelevant in real hardware
- Software bug : software doesn't run properly on simulator
 - Because software is buggy?
 - Because simulator is not faithful?



Matthieu Moy (Matthieu.Moy@imag.fr)

2012-2013

Another Kind of "Bug"

- Up to now:
 - Hardware Model doesn't work
 - Software doesn't run on hardware model
- What about :
 - ► Software does run on hardware model, but not on real chip

Hiding bugs ≠ Fixing bugs



Planning approximatif des séances

- Introduction : les systèmes sur puce
- 2 Introduction: modélisation au niveau transactionnel (TLM)
- Introduction au C++
- Présentation de SystemC, éléments de base
- Communications haut-niveau en SystemC
- Modélisation TLM en SystemC
- TP1: Première plateforme SystemC/TLM Utilisations des plateformes TLM
- TP2 (1/2): Utilisation de modules existants (affichage)
- TP2 (2/2): Utilisation de modules existants (affichage)
- Notions Avancé en SystemC/TLM
- TP3 (1/3): Intégration du logiciel embarqué
- TP3 (2/3): Intégration du logiciel embarqué
- TP3 (3/3): Intégration du logiciel embarqué
- Intervenant extérieur
- Perspectives et conclusion



Matthieu Moy (Matthieu.Moy@imag.fr)

What is a bug?

- Launch a SystemC/TLM simulation
- It produces incorrect result

Question



Good news or bad news?



What Can we Expect from the Model?

Software runs on TLM ⇒ Software runs on real chip

Software doesn't run on real chip ⇒ Software doesn't run on TLM



Matthieu Moy (Matthieu.Moy@imag.fr)

2012-2013

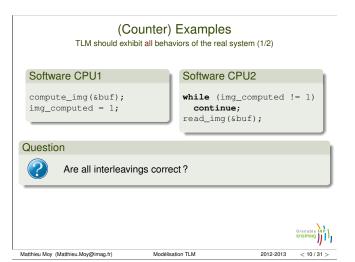
Sets of Behaviors, Faithfulness

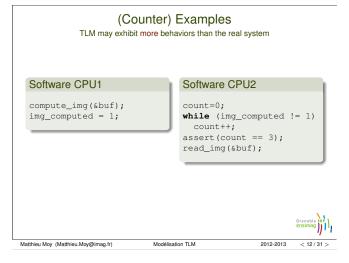
- An ideal TLM model ...
 - ► Should exhibit all behaviors of the real system
 - ► May exhibit more behaviors than the real system
 - ► Should not exhibit "too many" unrealistic behaviors



Matthieu Moy (Matthieu.Moy@imag.fr)

Modélisation TI M

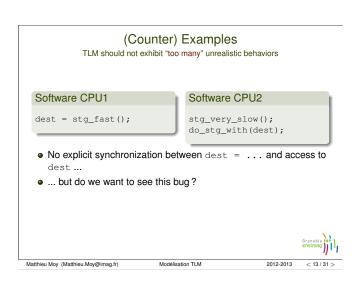


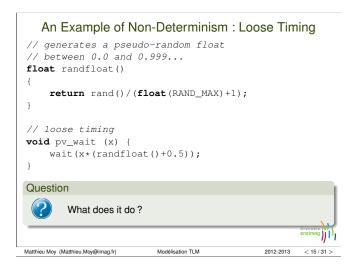


Set of Behaviors and Non-Determinism TLM models should exhibit several behaviors Several possibilities ⇒ non-determinism • Implementing non-determinism : Formal verification approach : exhaustive exploration ► Simulation approach : random Matthieu Moy (Matthieu.Moy@imag.fr) Modélisation TLM 2012-2013 < 14/31 >



(Counter) Examples TLM should exhibit all behaviors of the real system (2/2) (Incorrect) Software CPU1 Software CPU2 img_computed = 1; while (img_computed != 1) compute_img(&buf); continue; read_img(&buf); Question Are all interleavings correct? Question Will we see the bug in a simulation? Matthieu Moy (Matthieu.Moy@imag.fr) Modélisation TLM 2012-2013





Timing approximé et découplage temporel

- Constat : les context-switch sont lents.
- Conséquence 1 : wait coûte cher!
- Conséquence 2 : on évite de mettre des wait.



Matthieu Moy (Matthieu.Moy@imag.fr)

Modélisation TI M

2012-2013

Timing approximé et découplage temporel

- Problème :
 - ► En PVT, granularité de temps fine.
 - ightharpoonup \Rightarrow 1 wait pour chaque avancement du temps.
 - ▶ ⇒ simulation lente.
- Solution proposée en TLM 2 :
 - ► Chaque processus peut être « en avance » sur le temps global.
 - ► (L'avance correspond à l'argument sc_time des méthodes ransport de TLM-2, ignoré dans ENSITLM)
 - Quand faire les wait () (i.e. laisser le reste de la plate-forme rattraper notre avance) ?
 - * Quantum Keeping : Si on est plus « en avance » que le quantum (constante de temps choisie par l'utilisateur)
 - * Synchronisation explicite : Avant (ou après ?) les points de synchronisation



Matthieu Moy (Matthieu.Moy@imag.fr)

Modélisation TLM

2012-2013

Parallélisation de SystemC

- Paradoxe :
 - Les systèmes sur puces sont parallèles
 - SvstemC a une notion de processus
 - SystemC n'exploite qu'un processeur!
- Solution (très) naive :
 - ▶ 1 SC_THREAD \rightarrow 1 pthread
 - On lance tout en parallèle.
 - ⇒ beaucoup de pthreads, ne passe pas à l'échelle.
- Solution moins naive :
 - N processeurs → ≈ N pthreads.
 - ► Gestion du mapping « processus SystemC » ↔ pthread dans le kernel SystemC.

Question



Où est le problème?



Matthieu Moy (Matthieu.Moy@imag.fr)

2012-2013

Parallélisation de SystemC : conclusion

- C'est dur
- La plupart des gens qui le font ne se soucient pas de préservation de la sémantique
- Solution en pratique : lancer N simulations sur < N machines!
- Il faut peut-être un autre langage?



Matthieu Moy (Matthieu.Moy@imag.fr)

Modélisation TLM

2012-2013 < 23 / 31 >

Accélération des simulations SystemC Minimiser le coût du au context-switch

- Utilisation de SC_METHOD à la place des SC_THREAD (pas toujours possible)
- Minimisation du nombre de wait à exécuter
- Scheduling statique (http://www.cprover.org/scoot/: plus de travail à la compilation, simulation 2 à 6 fois plus rapide sur des exemples)



Timing approximé et découplage temporel

Exemple

sc_time t = SC_ZERO_TIME; // local advance over SystemC time
local_computation(); // Instantaneous wrt SystemC
t += sc_time(12, SC_NS); // don't wait() now
other_computation();
t += sc_time(3, SC_NS);
socket.write(addr, data, t); // may update t
wait(t); t = SC_ZERO_TIME; // Catch-up with SystemC time
send interrupt():

Question



Quel sont les problèmes?



Matthieu Moy (Matthieu.Moy@imag.fr)

Modélisation TLM

2012-2013

Parallélisation de SystemC

- Si on veut faire les choses proprement :
 - ► Analyse statique des dépendances de données
 - ► Prise en compte à l'exécution
 - (DRT de Yussef Bouzouzou—2 ans de travail)
 - ⇒ on sort du principe « SystemC, c'est facile à compiler, g++ le fait très bien ».
- Problème restant :
 - Parallélisation « à l'intérieur du δ -cycle », mais exécuter en parallèle des processus censés s'exécuter à différents instants de simu = difficile.
- Solutions envisageables :
 - ▶ Profiter du découplage temporel pour paralléliser
 - ► Tâches avec durée (cf. jTLM)



Matthieu Mov (Matthieu.Mov@imag.fr)

2012-2013

Accélération des simulations SystemC

- Code interne aux composants = C++ ⇒ q++ -03 et le tour est joué (ou pas)
- Context-switch = cher :
 - Scheduling
 - Sauvegarde/restauration de tous les registres
 - Changement du pointeur de pile ⇒ cache-miss
 - ► Changement du compteur programme ⇒ vidage de pipeline
- Transactions = cher:
 - ► Plusieurs appels de méthodes virtuelles (⇒ non inline-ables)
 - Décodage d'adresse
 - ► ⇒ Là où la vraie plateforme fait un load/store, le code TLM exécute du code difficile à optimiser.



Matthieu Moy (Matthieu.Moy@imag.fr)

Modélisation TLM

2012-2013

Accélération des simulations SystemC Minimiser le coût du aux transactions

• DMI = Direct Memory Interface : on récupère un pointeur sur la zone mémoire intéressante, et on fait des accès sans passer par le bus.

Question



Quel est le problème ?

• Techniques de compilation spécifiques : lancer une passe d'optimisations après l'élaboration (prototype basé sur LLVM développé au LIAMA et à Verimag)



Matthieu Moy (Matthieu.Moy@imag.fr)

Modélisation TI M

2012-2013

