

# Utilisation de SystemC pour la conception des SoC

**Daniela Dragomirescu<sup>1,2</sup>, Roberto Reyna<sup>3</sup>**

*1 - Université de Toulouse : INSA Toulouse, 135 Av. de Rangueil Toulouse cedex 4  
2-LAAS-CNRS ; Université de Toulouse, 7, Av. du colonel Roche, F-31077 Toulouse  
3-Freescale Toulouse*

## Résumé :

Nous présentons ici une approche de co-design et de co-simulation des systèmes matérielles logicielles en utilisant le langage SystemC. Le Transaction Level Model est présenté et utilisé. Plusieurs exemples sont proposés aux étudiants en partant des systèmes basiques tel que le registres à décalage vers des systèmes complexes tel que les calculatrices ou encore les bus CAN. Ces cours et travaux pratiques sont dédiés aux étudiants de Master de Toulouse.

## INTRODUCTION

L'enseignement du SystemC, présenté dans cet article, est composé des quelques séances de cours pour introduire les langages suivi des séances de travaux pratiques. Cet enseignement a une durée d'environ 15h et s'adresse à plusieurs filières d'enseignement toulousain, au niveau master 2 :

- aux étudiants du Master Pro CAMSI (Concepteur en Architecture de Machines et Systèmes Informatiques)
- aux étudiants de Master 2 de l'IUP AISEM (Architecture et Ingénierie de Systèmes électroniques et Microélectroniques)
- aux étudiants de 5<sup>ème</sup> année Automatique - Electronique, option Systèmes Electroniques de l'INSA de Toulouse – seulement le cours d'introduction au langage SystemC

Les promotions d'étudiants dans ces filières comptent entre 15 et 36 étudiants, avec des variations d'une année à l'autre.

L'utilisation des langages de description matérielle pour la modélisation, conception et synthèse de circuits et systèmes électroniques est très développée. L'enseignement dans ce domaine concerne les deux principaux langages de description : le VHDL et Verilog. Les élèves de master CAMSI et AISEM ainsi que les étudiants de l'INSA ont des solides connaissances en description VHDL. L'enseignement des langages de modélisation à un niveau système et non plus seulement matérielle est tout naturel dans le profil de ces formations.

De nos jours, les applications devient de plus en plus complexes et inclut presque toujours une partie matérielle et une partie logicielle. Pour arriver à atteindre des grandes vitesses de calcul l'intégration des différents circuits sur la même puce est fortement

souhaitée. Dans un système on chip (SoC) nous avons aujourd'hui la cohabitation sur silicium des nombreuses fonctions déjà existantes en elles-mêmes au par-avant: processeurs, traitement de signal, mémoires, bus, convertisseurs, communication. En vue des applications et de par la présence de microprocesseurs sur les SoC, le logiciel est toujours présent dans ces systèmes. Deux questions se posent :

- en partant d'une nouvelle application, comment faire le partitionnement software hardware ?
- une fois le partitionnement software hardware fait, les deux cycles de conception matériel et logiciel vont être développés en parallèle. Est-ce que à la fin, lors de l'intégration du système, le software fonctionnera-t-il sur le hardware ? C'est pour répondre à cette question qu'il ne faut pas développer les cycles de conception matériel et logiciel de manière disjointe, mais tout au contraire, conjointement. C'est ce que nous appelons le co-design et la co-simulation.

Pour apporter une réponse concrète à ces deux questions, nous avons besoin des logiciels de simulation système tel que SystemC (voir figure 1).

De plus, les systèmes matériels devenant de plus en plus complexes, le temps de conception devient très long, ce qui est incompatible avec le time to market (de plus en plus court). Donc il faut un savoir faire nécessaire très grand, mais le risque de "bug" reste très important. Des circuits entièrement dédiés "full custom" on est passé à l'utilisation dans les SoC des blocs virtuels avec des fonctionnalités relativement complexes (microprocesseurs), conçus, testés et fabriqués au par avant qu'on appelle des blocs IP (Intellectual Property)

### **OBJECTIFS PEDAGOGIQUES :**

- apprendre aux étudiants une méthodologie de conception des systèmes matériels – logiciels, ainsi que les langages disponibles pour développer cette méthodologie, en occurrence SystemC et son Transaction Level Model
- apprendre aux étudiants le co-design et la co-simulation des systèmes matériels – logiciels

### **LE FLOT DE CONCEPTION SystemC**

Le langage SystemC a été proposé comme solution pour la modélisation des composants logiciels et matériels d'un système en utilisant le langage C++. Le langage C++ est un langage logiciel qui possède néanmoins trois principales limitations pour la modélisation des systèmes :

- la notion du temps. En C++ aucune notion d'événements temporaires n'existe ;
- le comportement concurrentiel du matériel : le matériel est de nature concurrentielle, les différentes parties d'un système matériel fonctionnent généralement de manière parallèle et donc naturellement concurrentes.
- Les types C++ ne sont pas adaptés aux spécifications matérielles : par exemple il n'y a pas un type pour déclarer un objet qui serait en état de haute impédance.
- 

Ces trois limitations du langage C++ se trouvent tout naturellement gérées dans les langages de description matérielle comme VHDL ou Verilog. Mais ces derniers ne peuvent évidemment modéliser facilement les composants logiciels d'un système ou éventuellement pourraient le faire mais de manière peu optimisée.

Le langage SystemC a été proposé pour palier aux limitations du pur C++ pour la modélisation système. SystemC est une extension de C++, la syntaxe reste donc la même et on utilise les modules pour l'encapsulation et la gestion de la hiérarchie (l'équivalent des entités en VHDL), voir figure 1.

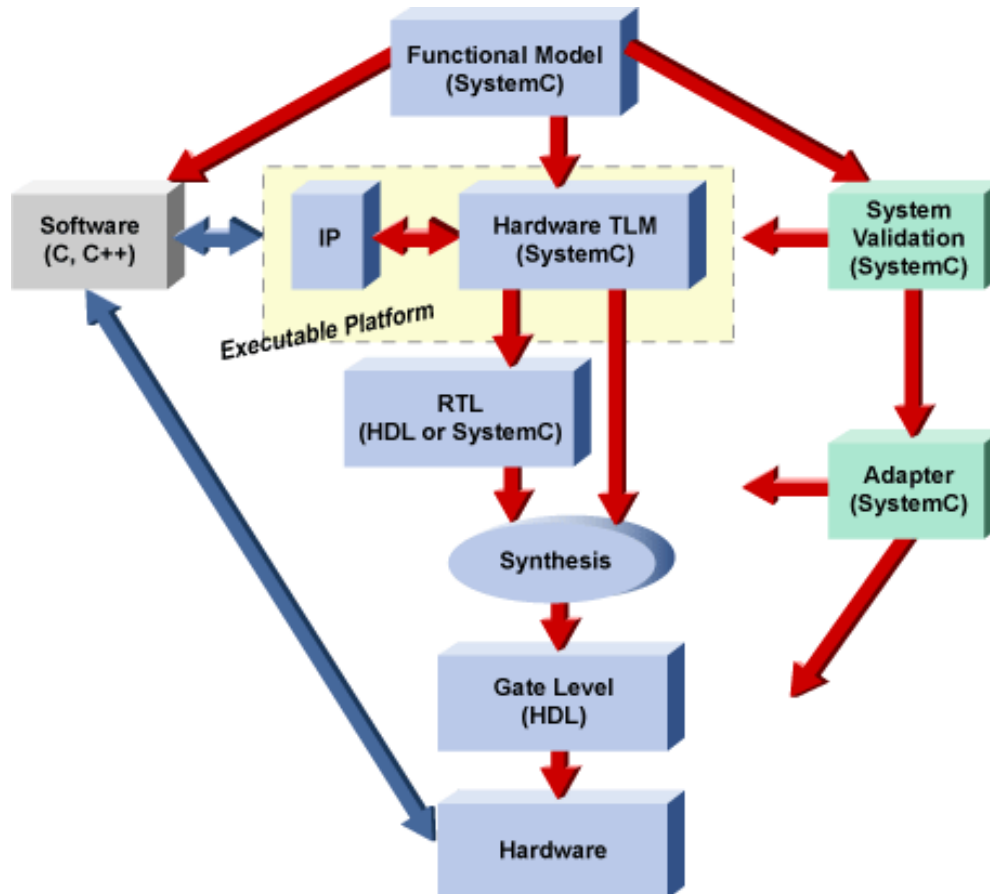


Figure 1. Le flot de conception d'un système complet (H/S) (image appartenant à Forte Design Systems)

## TRAVAUX PRATIQUES : DE LA MODELISATION DE REGISTRES A DECALAGE A LA MODELISATION DES CALCULATRICES

Après avoir acquis les notions théoriques de conception des SoC et les particularités du langage SystemC, les étudiants vont passer à l'implémentation en travaux pratiques effectué à l'AIME. Ils utiliseront le simulateur NCSC de CADENCE. La compilation de fichiers SystemC se fait avec le compilateur C++ de la machine. Lors de l'élaboration, les bases des données pour la simulation seront créés. Ensuite, la simulation est lancé en utilisant le même environnement de simulation que pour VHDL.

Les étudiants commencent par la modélisation d'un registre à décalage de 8 bits avec un signal de sélection pour pouvoir décaler à gauche, à droite, pour charger la donnée en entrée ou pour retenir la donnée contenue dans le registre. Le signal de remise à zéro Raz est actif bas tel qu'indiquer sur la figure et synchrone par rapport à l'horloge. Les 4 opérations sur le registre sont aussi synchrones par rapport à l'horloge principale.

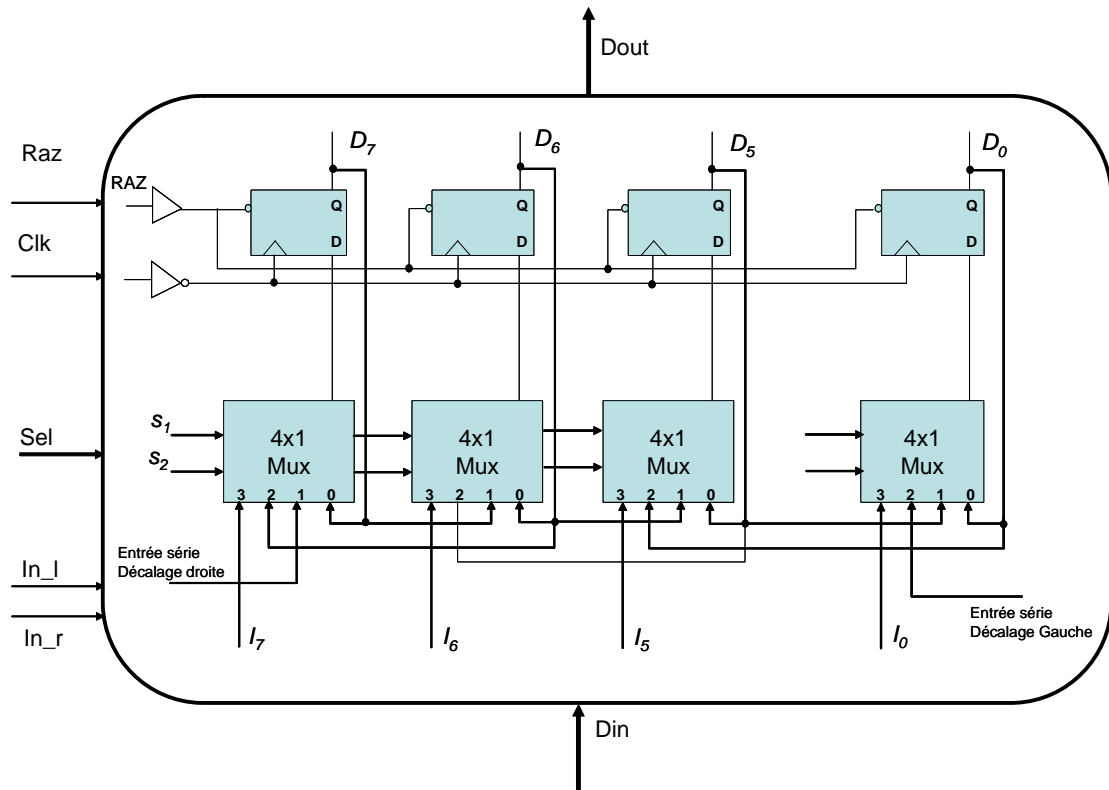


Figure 2. Registre à décalage

Les étudiants vont décrire ce registre en VHDL dans un premier temps (fichier **entity\_reg.vhd** pour décrire uniquement l'entité et fichier **archi\_reg.vhd** pour l'architecture) et le compiler avec la suite classique NC-VHDL. Ils vont également faire la description d'une séquence de test (un fichier pour l'entité et un fichier pour l'architecture). Une fois que votre simulation VHDL soit terminée correctement, les étudiants vont réaliser une traduction de ces différents fichiers en SystemC et lancez une simulation avec NCSC. Ils vont traduire le fichier **entity\_reg.vhd** dans un fichier d'entête C++ **registre\_dec.h** et l'architecture dans un fichier **registre\_dec.cpp** et simuler avec NCSC de CADENCE.

Voici le code SystemC de l'architecture du registre à décalage réalisé par les étudiants :

```
//-----
//-- ARCHITECTURE : Registre a décalage 8 bits
//-----

#include "registre_dec.h"
void registres::decaler()
{
if (!raz) {
sortie = "00000000";
} else if (sel==0) {
sortie = din;
} else if (sel==1) {
sortie.range(7,1) = sortie.range(6,0);
sortie[0] = in_l;
} else if (sel==2) {
sortie.range(6,0) = sortie.range(7,1);
sortie[7] = in_r;
}
}
dout = sortie;
}
```

Nous évoluons ensuite vers la conception des systèmes plus complexes. L'exemple choisi est une calculatrice en notation polonaise inversée également connue sous le nom de notation post-fixée. Dans un premier temps les étudiants vont programmer la calculatrice en utilisant toutes les ressources du langage informatique (C++), particulièrement les piles avec des pointeurs. La deuxième partie consistera à modéliser une pile en version « matérielle » c'est-à-dire une LIFO en SystemC que l'on intégrera à l'application.

### *L'APPLICATION : CALCULATRICE*

La notation polonaise inverse (NPI), permet de noter les formules arithmétiques sans utiliser de parenthèses. Dérivée de la notation polonaise présentée en 1920 par le mathématicien polonais Jan Lukasiewicz, elle s'en différencie par l'ordre des termes : les opérandes y sont présentés avant les opérateurs et non l'inverse. Les avantages : en programmation, chaque paire de parenthèses correspond à un calcul intermédiaire qu'il faudrait stocker quelque part. La NPI permet de gérer ces calculs intermédiaires sous forme de pile (alias LIFO pour last in- first out). Le passage par la NPI permet donc d'organiser plus facilement le travail d'optimisation du compilateur. Dans le cas d'une calculette, elle diminue le nombre de caractères à frapper, au prix d'un léger effort d'abstraction (écrire  $2\ 2\ +$  au lieu de  $2 + 2 =$ ). Accessoirement, à l'époque des premiers circuits intégrés, cela en diminuait aussi la complexité.

Les données seront rentrées au clavier sous forme de chaînes de caractères séparées par un espace ou un retour chariot. On devra tester si l'utilisateur a rentré un nombre (opérande) ou un caractère d'opération (opérateur). Dans le premier cas on empilera simplement la donnée, dans l'autre cas on appliquera l'opérateur sur les deux derniers opérandes en tête de la pile.

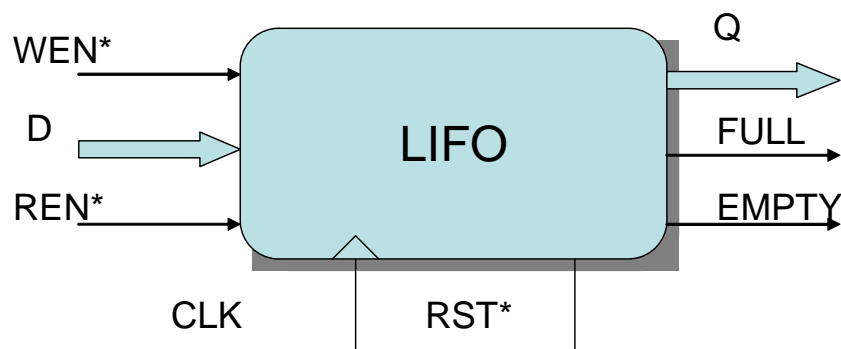
Les seuls opérateurs que l'on implémentera seront des opérateurs binaires : l'addition (+), la soustraction (-), la multiplication (\*) et la division (/).

### *IMPLEMENTATION DE LA PILE EN C++*

L'application envisagée consiste à empiler des opérandes (nombres réels) jusqu'à rencontrer un opérateur. L'opérateur dépile les opérandes dont il a besoin et empile le résultat. La pile à implémenter est donc une pile de nombres réels. La pile peut être caractérisée par sa taille, l'indice de son sommet et un tableau stockant les valeurs qu'elle contient. On utilisera une structure pour la représenter. Une fonction de création permettra d'affecter les valeurs des champs (taille et sommet) et d'allouer dynamiquement de la mémoire pour le tableau. La pile créée étant vide au départ, le seul paramètre à passer à la fonction (qui renverra un pointeur vers une structure de type pile) est sa taille. Ecrire le code correspondant et vérifier qu'il est correct en créant une pile de 10 éléments. Les deux primitives principales pour gérer une pile sont les fonctions d'empilage et de dépilage qui doivent être écrites.

### *LE MODELE MATERIEL.*

On se propose, dans un deuxième temps, de modéliser en SystemC une pile LIFO (figure 3) qui sera **intégrée à l'application de la calculatrice et qui remplacera le pointeur et le tableau de données.**



**Figure 3. La LIFO à implémenter en SystemC et à intégrer à l'application**

On fera la modélisation générique par rapport à la taille de la LIFO. Une pile LIFO possède un seul pointeur sur le tableau et un seul bus d'entrée-sortie. Le signal RST\* efface le contenu du tableau et remet la valeur de pointeur à zéro. A chaque activation de WEN\* la LIFO doit écrire la valeur présente sur le bus D. Quand REN\* est actif, la LIFO présentera une donnée sur le bus de données sur le front montant d'horloge. Si l'on suppose que la pile sera raccordée à d'autres unités, la sortie passe en haute impédance quand REN\* est inactif. Les signaux EMPTY et FULL représentent respectivement l'état vide et plein, qui seront mis à jour sur le front montant d'horloge. Les signaux REN\* et WEN\* ne peuvent pas être actifs simultanément. Pour que le modèle soit complet il faut supposer qu'en cas que l'application active ces deux signaux c'est l'écriture qui sera prioritaire. Les cas particuliers d'écriture avec LIFO pleine et lecture avec LIFO vide seront implémentés également.

Après la modélisation en SystemC de la pile LIFO et sa simulation avec le fichier de test, celle-ci sera intégrée à l'application calculatrice.

Pour certains binômes, en fonction de leur avancement, un mini-projet est proposé concernant la modélisation du protocole CAN en utilisant SystemC.

## CONCLUSION

Lors de cet enseignement, les étudiants ont découvert le monde des systèmes on chip et ils ont appris le co-design et la co-simulation des systèmes on chip software hardware en utilisant le langage SystemC. Les étudiants ont vu l'intérêt du passage de VHDL à SystemC pour les applications complexes nécessitant du matériel et logiciel en même temps.