# Caractérisation d'une IP VHDL de réseau sur puce en SystemC

S Jovanovic, Y. Berviller, Serge Weber

# Caractérisation d'une IP VHDL de réseau sur puce en SystemC

S. Jovanovic, Y. Berviller et S. Weber

Institut Jean Lamour (UMR7198) et Pôle CNFM MIGREST – Nancy, Université de Lorraine, Nancy, France

Contact email : slavisa.jovanovic@univ-lorraine.fr

Nous présentons un ensemble de travaux pratiques qui sont dispensés au sein du Master Ingénierie Électrique, Électronique et Informatique Industrielle (I2E2I) spécialité Électronique Embarquée et Microsystèmes (EEM) à l'université de Lorraine dans le cadre du module « Modélisation SystemC ». Ces TP sont destinés à initier les étudiants à la modélisation de systèmes et circuits numériques en SystemC et sont organisés autour de la suite logicielle *Modelsim* de *MentorGraphics* pour la simulation, le test et la vérification.

## 1. Introduction

Pour initier les étudiants de la formation Master I2E2I-EEM à l'université de Lorraine à la modélisation de systèmes et circuits en SystemC, nous avons mis en place une série de travaux pratiques autour des logiciels de simulation MentorGraphics consistant à caractériser un bloc de propriété intellectuelle (IP) décrit en VHDL, notamment un réseau sur puce NoC, en utilisant des modules de test décrits en langage SystemC. Cette expérience pédagogique a lieu depuis 2 ans au sein de la formation I2E2I-EEM à l'université de Lorraine. La modélisation d'un système en SystemC permettant de caractériser une IP VHDL avancée s'inscrit dans une unité de 30 heures comprenant :

- introduction à la modélisation SystemC (flot de conception, architecture du langage, modules, canaux primitifs et hiérarchiques, interfaces, processus, types de données, scheduler, noyau, temps): 10h de cours/TD,
- introduction à la modélisation TLM: 4h de cours,
- introduction à la simulation SystemC et co-simulation SystemC-VHDL avec Modelsim: 2h de TP,
- modélisation d'un module (processeur ou PE) de test en SystemC: 4h de TP
- validation du module PE en SystemC sur un modèle SystemC de réseau sur puce : 4h de TP,
- adaptation et validation de modules de test décrits en SystemC à l'IP VHDL d'un réseau sur puce: 6h de TP.

Cette unité s'adresse aux étudiants de niveau Master 2 qui ont déjà suivi des formations de conception numérique en VHDL et sur FPGA (au moins 30h), de conception de cellules numériques de base au niveau microélectronique (60h) et de conception de circuits numériques de complexité moyenne avec la suite logicielle Cadence (30h). On peut donc considérer que les étudiants ont une expérience suffisante et non négligeable pour mener à bien le projet proposé.

## 2. Réseau sur puce NoC

Les réseaux sur puce (NoC) apparaissent comme une alternative pour connecter des modules intégrés sur une même puce vis-à-vis d'une approche d'interconnexion basée sur

Article publié par EDP Sciences et disponible sur le site https://www.j3ea.org ou https://doi.org/10.1051/j3ea/20171019

une structure de communication de type bus partagé ou hiérarchique [1,2]. Ils représentent une solution de transposition et de réduction d'échelle (au niveau d'un même substrat) des concepts des grands réseaux de communications (*large-scale networking*) vers le domaine des systèmes sur puce embarqués (SoC, MPSoC). Les NoC reposent sur une paquetisation de données pour transporter les données d'une source vers une cible à travers un réseau d'aiguilleurs (routeurs) et de canaux d'interconnexion (*interconnection link*). Un exemple d'une structure de NoC couramment utilisée est illustré Figure 1. Il s'agit d'une structure NoC de type maillé 2D (mesh), composée de plusieurs éléments de calcul PE (*Processing Element*) connectés les uns aux autres via les routeurs et les fils d'interconnexion d'une taille fixe. Un élément de calcul PE (souvent appelé un nœud), peut être un processeur, une fonction spécifique de calcul (IP), un bloc de mémorisation ou une combinaison de tous ces composants reliés ensemble. Un PE est généralement connecté à un routeur via un module d'interface NI (*Network Interface*) dont la fonction principale est la mise en paquets de toutes les données générées et à transmettre par un PE vers un autre PE à travers le réseau de communication. Un routeur est composé de buffers d'entrée, de sortie ou d'entrée-sortie dont le rôle est d'accepter temporairement les paquets à transmettre de la part du PE qui lui est associé ou de ses routeurs voisins dans le but de faire transiter vers une destination des paquets qui ne lui sont pas destinés. Pour cela, un routeur dispose généralement d'un module crossbar utilisé pour l'aiguillage de paquets depuis les ports d'entrée vers les ports de sorties du routeur selon l'adresse de destination généralement contenue dans le premier paquet (paquet d'en-tête - *header packet*).

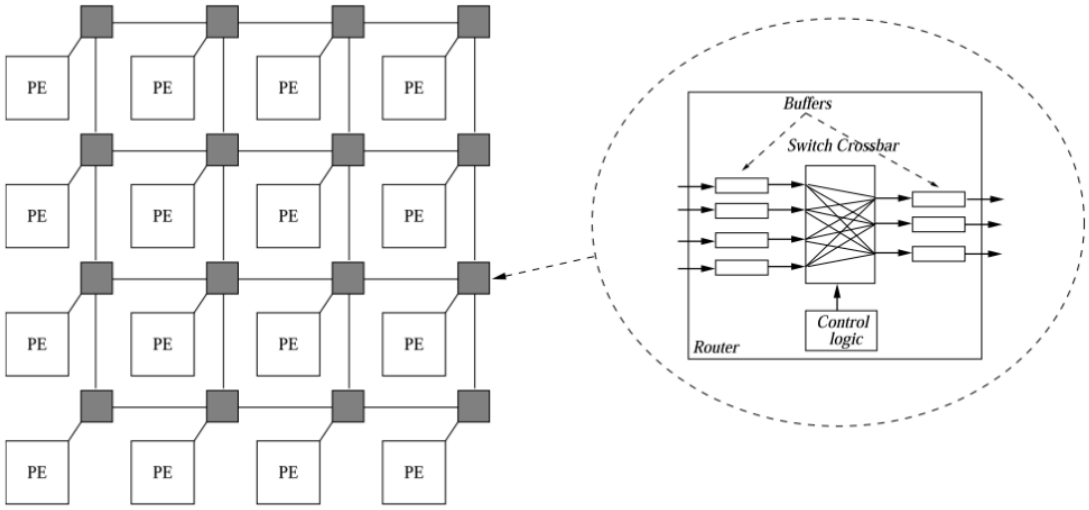


Fig.1 Réseau sur puce

Les données qui transitent dans un NoC sont sous forme de messages comme présenté Figure 2. Chaque message est composé de plusieurs paquets que l'on appelle *flits* et qui peuvent être de type :

- *header* - les flits d'en-tête comportant l'adresse de la destination à laquelle le message doit être envoyé - commencent par la séquence binaire "01",
- *body* - les flits comportant les données à transmettre vers la destination désignée par le *header* flit - commencent par la séquence binaire "00",
- *tail* - les flits de fin comportant les données à transmettre et permettant de fermer la communication initiée entre 2 PE - commencent par la séquence binaire "11".


Fig.2 Exemple de structure de message dans un NoC

Le transfert de données entre deux routeurs ou entre un PE et un routeur s'effectue en utilisant le protocole ABP (*Alternate Bit Protocole*) qui est présenté Figure 3. Chaque paquet est envoyé sur 2 cycles d'horloge, suivi d'un signal de contrôle (*signal _req_*). Le paquet suivant n'est envoyé que si le signal d'acquittement pour le paquet précédent est reçu (le *signal _ack_*).

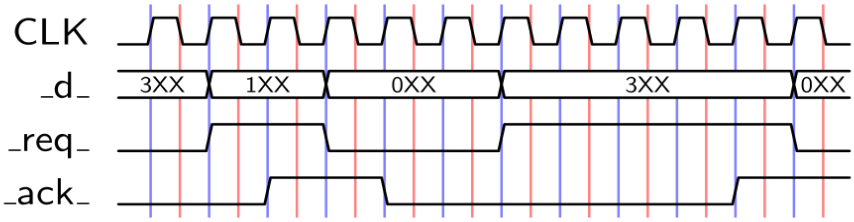


Fig.3 Alternate Bit Protocole [1,2]

## 3. Démarche pédagogique

La modélisation d'un système de test en SystemC permettant de caractériser une IP VHDL de type NoC passe par les étapes suivantes :

1. Introduction à la simulation SystemC et co-simulation SystemC-VHDL avec *Modelsim,*
2. Modélisation d'un module (processeur ou PE) de test d'un NoC en SystemC et sa validation,
3. Adaptation et intégration de modules de test décrits en SystemC à l'IP VHDL d'un réseau sur puce.

## 4. Introduction à la (co-)simulation SystemC(-VHDL) avec Modelsim

Dans la première étape qui est programmée sur 2h de TP, l'étudiant est amené à se familiariser avec l'outil de modélisation Modelsim de MentorGraphics [3,4]. L'objectif principal de cette étape est de voir les principales différences dans la modélisation d'un système numérique en SystemC en utilisant l'outil Modelsim par rapport au flot de modélisation classique vu en cours/TD, consistant à utiliser l'outil Eclipse et la chaîne de compilation gcc [5]. Dans l'outil Modelsim, il n'y pas de fonction principale, tout doit être décrit sous forme de modules. Le module qui sera au niveau hiérarchique le plus élevé (et qui fera le rôle de la fonction principale) est désigné par la macro SC_MODULE_EXPORT(nomDeModule). Les figures 4 et 5 présentent un exemple d'adaptation d'une fonction principale décrite en SystemC pour son utilisation dans la suite logicielle Modelsim. Nous pouvons constater que la fonction principale de la Figure 4 est remplacée par le module new_top de la Figure 5, qui contient une fonction nommée sc_main_body(), qui jouera le rôle de la fonction principale au sein du module. Nous pouvons également constater que les différences entre ces deux descriptions sont minimes et les étudiants franchissent rapidement cette étape d'adaptation à la simulation SystemC avec l'outil Modelsim.

```
int sc_main(int, char**)
{
        sc_signal<bool> reset;
        counter_top top("top");
        sc_clock CLK("CLK",10,SC_NS,        );
        top.reset(reset);
        reset.write(1);
        sc_start(5, SC_NS);
        reset.write(0);
        sc_start(100, SC_NS);
        reset.write(1);
        sc_start(5, SC_NS);
        reset.write(0);
        sc_start(100, SC_NS);
}
```

```
SC_MODULE(new_top)
{
        sc_signal<bool> reset;
        counter_top top;
        sc_clock CLK;
        void sc_main_body();
        SC_CTOR(new_top)
        :reset("reset"),top("top"),
         CLK("CLK", 10, SC_NS)
        {
                top.reset(reset);
        SC_THREAD(sc_main_body);
        }
};
void new_top::sc_main_body()
{
        reset.write(1);
        wait(5, SC_NS);
        reset.write(0);
        wait(100, SC_NS);
        reset.write(1);
        wait(5, SC_NS);
        reset.write(0);
        wait(100, SC_NS);
}
SC_MODULE_EXPORT(new_top);
```

Fig.6 Co-simulation SystemC-VHDL sur l'exemple d'une bascule D décrite en VHDL

Dans un second temps, le flot de co-simulation SystemC-VHDL leur est présenté sur un exemple simple: une bascule D décrite en VHDL est utilisée comme exemple pour l'initiation à la co-simulation SystemC-VHDL (voir Figure 6). La bascule D décrite en VHDL ne peut pas être utilisée directement dans un fichier de *testbench* décrit en SystemC. Pour ce faire, une interface SystemC-VHDL est créée à l'aide de la commande scgenmod pour une utilisation en SystemC, le nom étant donné de la façon suivante :

```
scgenmod dff > dff.h
```

Ce fichier d'interface est présenté sous forme d'un fichier d'en-tête .h facilement utilisable dans une description SystemC (voir Figure 7). Ensuite, les étudiants réalisent un fichier de test (*testbench*) en SystemC permettant de générer les stimuli pour simuler et tester l'entité fournie en VHDL.

```
#include "systemc.h"

class dff : public sc_foreign_module
{
public:
   sc_in<sc_logic> d;
   sc_out<sc_logic> q;
   sc_in<bool> clk;

   dff(sc_module_name nm, const char* hdl_name)
    : sc_foreign_module(nm),
      d("d"),
      q("q"),
      clk("clk")
   {
      elaborate_foreign_module(hdl_name);
   }
   ~dff()
   {}
};
```

Fig.7 Le fichier "dff.h" représentant l'interface de la bascule décrite en VHDL pour son utilisation dans une simulation SystemC

## 5. Modélisation d'un module de test (PE) d'un NoC en SystemC et sa validation

Dans cette étape, qui est programmée sur deux séances de TP de 4 heures, l'étudiant est amené à décrire en SystemC le module de test PE communicant, permettant d'envoyer et de recevoir des paquets de données en format reconnaissable par le réseau NoC (voir Figure 8). Avant de passer à la modélisation SystemC, l'étudiant analyse les mécanismes d'échanges de données entre deux routeurs NoC décrits dans la documentation fournie. Le module de test ou PE à modéliser en SystemC doit comprendre les propriétés et fonctions suivantes:

- un identifiant (de 0 à 3) qui sera initialisé au moment de l'instanciation du module et qui servira de base pour l'adressage des messages,
- une entrée et une sortie au format reconnaissable par le NoC (34 bits) permettant d'envoyer ou de recevoir des paquets vers d'autres PE ou depuis les autres PE,
- tx_function() permettant d'envoyer des messages d'une taille fixe (3 flits) vers un autre PE choisi de manière aléatoire. Cette fonctionnalité doit être contrôlable par un drapeau (flag). Chaque *flit* généré doit être sur 34 bits, les deux premiers bits sont utilisés pour déterminer le type de *flit* ("01" - *header*, "00" - *body* et "11" - *tail*). Le *header flit* a la structure suivante: les bits de 31 à 28 sont utilisés pour la coordonnée x, les bits de 27 à 24 sont utilisés pour la coordonnée y. Les autres peuvent être utilisés pour décrire le PE source et le numéro du *flit* courant (0 pour *header*, 1 pour le suivant, 2 pour *tail*),
- rx_function() permettant de recevoir tous les messages envoyés depuis les PE voisins. Cette fonction est toujours opérationnelle (pas de drapeau de contrôle).

Le module PE doit, dans un premier temps, avoir uniquement une entrée de réception et une sortie d'émission de *flits* comme présenté Figure 8.

```
#include "systemc.h"
#include "Main.h"

SC_MODULE(pe){
        sc_fifo_out<sc_lv<N> > data_tx;
        sc_fifo_in<sc_lv<N> > data_rx;
        void tx_function();
        void rx_function();
        void setSentFlag(bool f);
        unsigned int getId();

        SC_HAS_PROCESS(pe);
        pe(sc_module_name mn, unsigned int _id,bool f);
        unsigned int id;
        int flitCounter;
        int headCounter;
        bool sentFlag;
        bool finished;
        bool message_end;
};
```

Fig.8 Le fichier "pe.h" décrivant les entrées/sorties et les fonctions principales du module de calcul PE.

Une fois le module PE modélisé en SystemC avec les propriétés et fonctionnalités présentées ci-dessus, l'étudiant passe à l'étape de validation. Pour ce faire, il utilisera le réseau sur puce d'une taille 2x2 décrit en SystemC de manière comportementale et fourni par l'enseignant (voir Figure 9). L'objectif de cette étape est de vérifier uniquement si toutes les fonctions réalisées dans cette étape ont le comportement voulu, notamment les phases d'envoi et de réception de données. Un extrait des résultats de simulation à l'issue de cette étape de validation est présentée Figure 10. Nous pouvons voir dans l'onglet de simulation de l'outil Modelsim la présence de 4 instances de module PE et d'une seule instance de module NoC. De plus, les résultats de simulation sont affichés dans la console de l'outil Modelsim et pas sous forme de chronogrammes pour la simple raison que les données de communication entre le réseau sur puce comportemental et les éléments de calcul PE transitent via les canaux de communication de type *sc_fifo*, qui malheureusement ne peuvent pas être visualisés autrement que dans la console à l'aide des commandes d'affichage cout de la library *stdout*.

```
#include "systemc.h"
#include "Main.h"
#include <vector>
#include <stdlib.h>
using namespace std;
SC_MODULE(crossbar)
{
        sc_fifo_in<sc_lv<N> > *inputs;
        sc_fifo_out<sc_lv<N> > *outputs;
        void crossbar_func();
        SC_HAS_PROCESS(crossbar);
        crossbar(sc_module_name _name, unsigned int _inputNbr,
unsigned int _outputNbr);
private:
        unsigned int inputNbr;
        unsigned int outputNbr;
        vector<int> crossbarVect;
};
```

Fig.9 Le fichier d'entête du réseau sur puce comportemental utilisé pour valider les PE

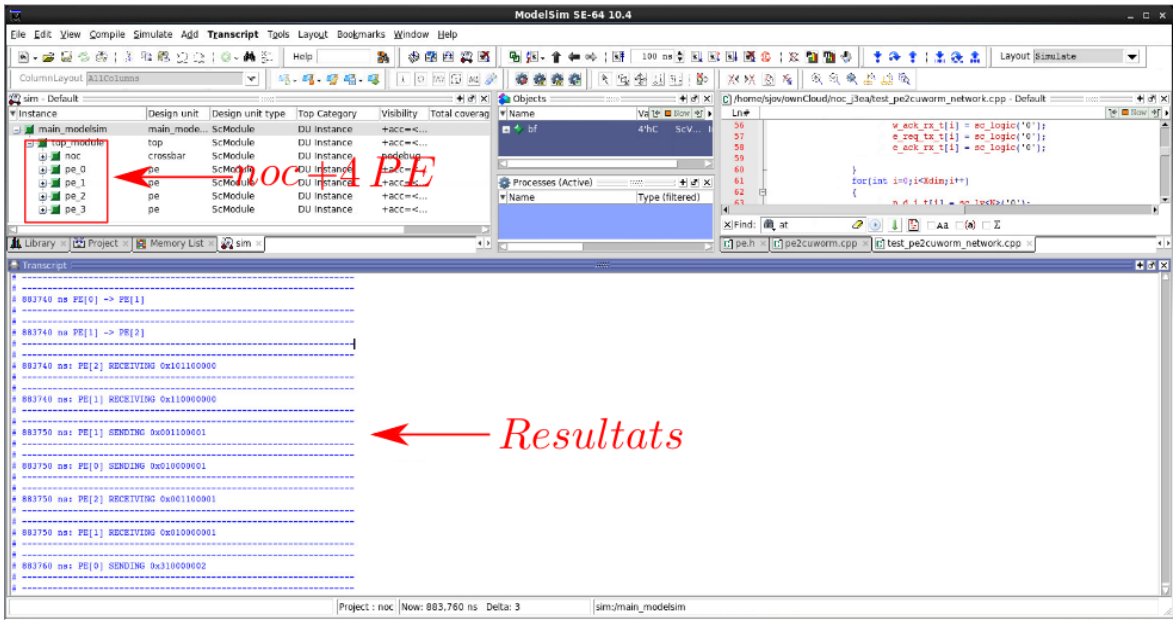


Fig.10 Extrait de résultats obtenus lors de l'étape de validation des PE à l'aide d'un réseau sur puce comportemental.

## 6. Adaptation et validation de PE décrit en SystemC à l'IP VHDL

La dernière étape de ces TP, programmée sur une durée de 6h, consiste à adapter à l'IP VHDL de NoC fournie le module PE décrit précédemment et effectuer une validation finale. Avec les fichiers d'en-tête générés par l'outil Modelsim, l'étudiant doit adapter les entrées/sorties du module PE à celles de l'IP VHDL en respectant les directions de chaque signal (voir Figure 11). Dans cette étape, les connexions entre les PE adaptés et l'IP VHDL de NoC fournie sont réalisées. De cette manière, les modules PE validés dans l'étape précédente sur un NoC comportemental en SystemC peuvent être utilisés pour caractériser un NoC synthétisable fourni sous forme d'une IP protégée.

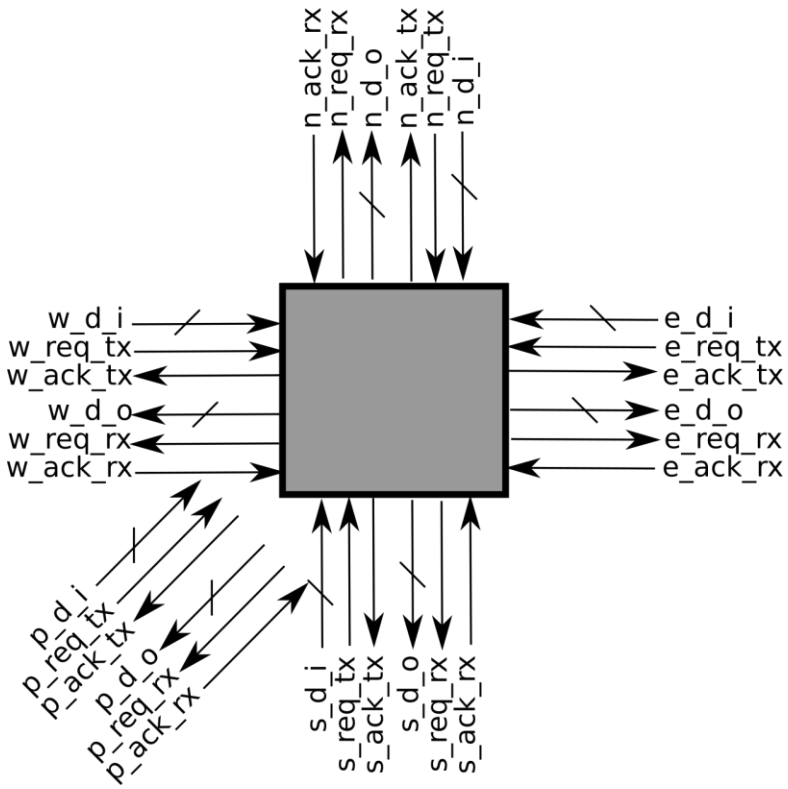


Fig.11 Les signaux d'interconnexion d'un routeur

En observant les interconnexions entre un élément de calcul PE et un routeur (tous les signaux commençant par la lettre p_ en Figure 11), nous pouvons remarquer que non seulement les bus de données sont présents (p_d_i et p_d_o) mais également les signaux de contrôle. Dans la version comportementale d'un élément de calcul PE décrit dans les sections précédentes, seules les entrées et sorties de type *sc_fifo_in* et *sc_fifo_out* utilisant les données sur 34 bits sont utilisées. Pour pouvoir connecter ces PE comportementaux à l'IP de réseau sur puce décrite en VHDL, ces interconnexions de bas niveau doivent également être réalisées. Pour ce faire, chaque élément de calcul PE doit se connecter au réseau NoC VHDL à travers un adaptateur nommé « pe2cuworm » présenté Figure 12. Cet adaptateur comprend également deux fonctions « tx_function() » et « rx_function() » comme le module PE décrit précédemment. Le rôle principal de ces deux fonctions est tout simplement de récupérer les valeurs envoyées de ou vers le module PE et de les transférer vers le réseau NoC VHDL tout en générant les signaux de contrôle de bas niveau nécessaires pour la réalisation du protocole ABP décrit dans la partie introductive de ce papier.

Une fois les adaptateurs réalisés pour tous les PE, l'ensemble des éléments de calcul PE avec leurs adaptateurs et le réseau sur puce VHDL peut être simulé avec la suite logicielle Modelsim. Ces résultats sont présentés en Figure 13. Dans l'onglet de simulation de l'outil Modelsim, nous pouvons observer non seulement les 4 instances des éléments de calcul PE validés dans l'étape précédente, mais également le réseau sur puce VHDL (l'IP en bleu) et les adaptateurs entre les PE et le réseau. Contrairement à la validation comportementale des PE dans l'étape précédente où les résultats étaient affichés uniquement dans la console de l'outil Modelsim, dans cette étape les résultats sont visibles dans la console et dans les chronogrammes comme présenté Figure 13. La console affiche toujours les résultats

générés par les PE tandis que les chronogrammes affichent les signaux de très bas niveau propres au réseau sur puce VHDL.

```
#include "systemc.h"
#include "Main.h"

SC_MODULE(pe2cuworm){
        sc_fifo_in<sc_lv<N> > data_fifo_tx;
        sc_fifo_out<sc_lv<N> > data_fifo_rx;
        sc_out<sc_lv<N> > data_tx;
        sc_out<sc_logic> req_tx;
        sc_in<sc_logic> ack_tx;
        sc_in<sc_lv<N> > data_rx;
        sc_in<sc_logic> req_rx;
        sc_out<sc_logic> ack_rx;
        sc_in<bool> reset;
        sc_in<bool> clock;
        void tx_function();
        void rx_function();
        SC_HAS_PROCESS(pe2cuworm);
        pe2cuworm(sc_module_name mn, unsigned int _id);
        bool current_level_rx;
        bool current_level_tx;
};
```

Fig.12 Le fichier d'en-tête de l'adaptateur entre l'élément de calcul PE et le réseau sur puce VHDL

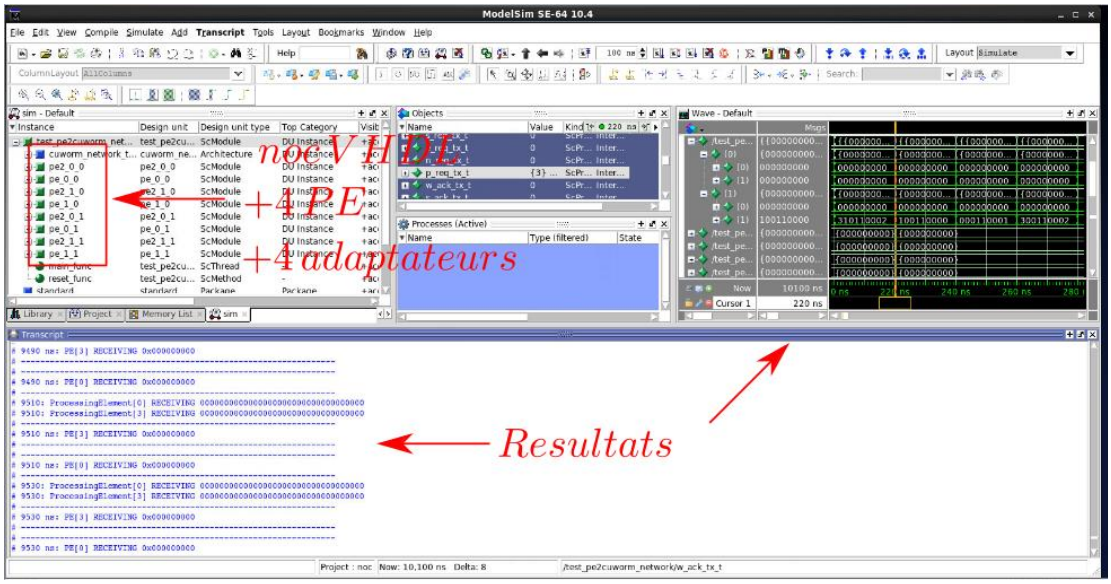


Fig.13 Extrait de résultats obtenus lors de l'étape de validation des PE, adaptateurs et du réseau sur puce VHDL

## 7. Conclusion et bilan

Nous avons présenté une suite de travaux pratiques destinés à familiariser les étudiants du master I2E2I-EEM de l'université de Lorraine à la modélisation SystemC et la co-simulation SystemC-VHDL permettant de tester et caractériser des IP en VHDL de

complexité élevée. Les travaux pratiques décrits dans cet article ont déjà eu lieu en 2015 et 2016. Le bilan de ces premières expériences est plutôt positif, malgré des difficultés rencontrées pour aller jusqu'au bout du projet durant les 16 heures de TP à disposition. L'étape qui pose le plus de problèmes aux étudiants est l'étape de modélisation d'un PE selon les spécifications énoncées. Malgré une relativement bonne expérience en programmation C/C++ acquise dans le cadre d'autres matières, les étudiants perdent beaucoup de temps dans cette étape et débordent sur la séance destinée à adapter le PE conçu pour l'IP NoC en VHDL, compromettant ainsi le bon déroulement du projet jusqu'à sa phase finale. Pour pouvoir finaliser le projet dans les temps, une séance de TP supplémentaire de 4 heures est à prévoir dans les prochaines accréditations.

## Remerciements

Les auteurs souhaitent remercier l'ensemble des membres des Services Nationaux qui distribuent les logiciels ainsi que le GIP-CNFM et l'IDEFI FINMINA (ANR 2011-IDFI-0017) pour les co-financements apportés aux Services Nationaux pour développer ces nouvelles approches pédagogiques.

## Références

1. S. Jovanovic, Architecture reconfigurable de système embarqué auto-organisé, Université de Lorraine (2009).
2. W.J. Dally and B. Towles. Route packets, not wires: on-chip inteconnection networks. In *Proc. of the 38th annual Design Automation Conference (DAC '01). ACM,* New York, NY, USA, 684-689. (2001).
3. MentorGraphics, SystemC verification with Modelsim, Application Note, (2007).
4. MentorGraphics, Modelsim SE User's manual, (2010).
5. K. Bjerge, *Guide for getting started with SystemC development*, Danish Technological Institute (2007).