Crossbus: Flot de Conception et NoC pour MPSoPC

Gabriel Oshiro Zardo, Dominique Houzet, Sylvain Huet

Crossbus: Flot de Conception et NoC pour MPSoPC

Gabriel Oshiro Zardo
UFRGS
Brazil
gabrieloshiro@gmail.com

Dominique Houzet
INPG GIPSA-Lab CNRS
France
dominique.houzet@gipsa-lab.inpg.fr

Sylvain Huet
INPG GIPSA-Lab CNRS
France
sylvain.huet@gipsa-lab.inpg.fr

1. Introduction

Le nombre d'éléments de calcul (PE) à interconnecter dans les systèmes sur silicium (SoC) ne cesse de croître. Aussi, il est particulièrement important de soigner la topologie d'interconnexion entre les PE afin d'en tirer la meilleure efficacité. Cette problématique a motivé la naissance des réseaux sur puce (NoC). Ils offrent un compromis performances/flexibilité non envisageable avec les stratégies classiques d'interconnexion, e.g. bus partagés, liaisons point à point. De nombreux NoC, industriels et académiques ont été proposés depuis les années 2000 [1]. Si une grande majorité de NoC cible une implantation sur ASIC, l'évolution des ressources disponibles dans les composants programmables FPGA rend également pertinent leur utilisation sur ces composants. Les NoC présentés dans [2-3] ont été développés avec cet objectif. Le NoC Crossbus, développé par le GIPSA-lab, cible également une implantation sur FPGA (Xilinx). Son originalité repose d'une part sur la possibilité de générer un NoC permettant d'interconnecter plusieurs FPGA et d'autre part d'être développé en conjonction avec un modèle de programmation de haut niveau, reposant sur SystemC. Le domaine applicatif ciblé par ces travaux est le traitement du signal et de l'image.

2. Architecture matérielle

Le NoC Crossbus permet l'interconnexion de N FPGA en anneau, chaque FPGA contenant une matrice d'interconnexion 2D torique de OxP routeurs. Les FPGA sont interconnectés entre eux à l'aide de liens de type Rocket IO (lien série haut débit Xilinx). Les routeurs sont interconnectés avec des liens de type FSL (FIFO Xilinx). La matrice de OxP routeurs peut être creuse. Chaque routeur peut être connecté à au plus 4 éléments de calculs, logiciels ou matériels. Crossbus utilise une technique de routage câblée de type XY.

La figure 1 présente un exemple de réseau pour N=O=P=3. Les PE connectés aux routeurs ne sont pas représentés.

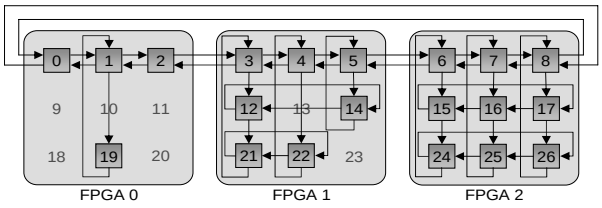

Figure 1: Exemple d'architecture Crossbus

3. Modèle de programmation et flot de conception

Nous avons défini un flot de conception, cf. figure 2, reposant sur le modèle de programmation SystemC qui permet de générer à la fois la description matérielle du système, i.e. NoC Crossbus et PE Hw/Sw connectés aux routeurs, et le logiciel associé.

Le flot de conception commence par la spécification de l'application avec SystemC. Celle-ci est décrite à l'aide d'un ensemble sc_module interconnectés par des canaux de communication de type sc_signal et sc_fifo. La synchronisation est exprimée à l'aide d'un signal d'horloge et de l'appel à la primitive wait.

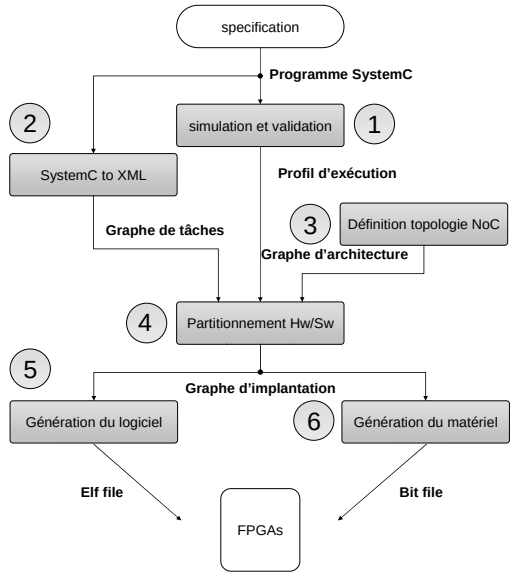

Figure 2: Flot de conception

Dans une première étape ce code est simulé et validé. Il est possible d'en tirer un profile d'exécution.
Une seconde étape consiste à établir le graphe de tâches correspondant à la description SystemC du système. Nous le représentons avec le format XML Spirit 2.0 [4].
Les informations de profil d'exécution, l'expertise du concepteur, permettent de définir la topologie du NoC (étape 3). La description du graphe d'architecture est manuelle. Elle est faite dans un format XML.
Vient ensuite l'étape 4 de partitionnement Hw/Sw. Celle-ci est manuelle. Elle consiste à définir la nature des PE utilisés par chaque tâche (Hw/Sw) et à quels routeurs sont attachés ces PE. Plusieurs tâches pouvant être réalisées par un même PE logiciel, il est également spécifié la répartition des PE logiciels sur les routeurs et l'association tâches/PE logiciels. Il résulte de cette étape un graphe d'implantation.
Le logiciel implanté dans les PE logiciels est généré dans une cinquième étape. Pour des raisons d'efficacité, temps de calcul et taille de code, la spécification SystemC est transformée en un code C qui respecte la sémantique SystemC. Pour cela, d'une part, nous générons un ordonnanceur spécifique à l'application. D'autre part, les primitives de communication sc_fifo, sc_signal et de synchronisation, wait, sont implantées à l'aide des primitives MPI_Init (initialisation), MPI_Comm_Rank (obtention de l'identifiant du PE), MPI_Barrier (synchronisation), MPI_Put (transfert d'information) et MPI_Finalize (fin du programme) du sous ensemble RMA de MPI-2 [5]. Cela permet (1) d'avoir un code générique, i.e. pouvant être implanté sur n'importe quel PE logiciel, pourvu qu'une implémentation de ces primitives soit disponible, et (2) d'optimiser le NoC, e.g. la synchronisation est directement câblée dans le NoC, les transferts d'information ne se font que par des écritures DMA (il n'y a pas de primitive de lecture).
La sixième étape consiste à générer la description matérielle du système. Tout d'abord sont sélectionnés les PE matériels (IP) du graphe d'implantation qui seront instanciés. Pour les PE logiciels cela revient à sélectionner le type de processeur utilisé; pour le moment nous ciblons soit des PowerPC soit des Microblaze. Pour les PE matériels cela nécessite, soit de sélectionner des IP existantes répondant à la fonctionnalité attendue, soit à coder la fonctionnalité manuellement au niveau RTL, ou à utiliser un outil de synthèse de haut niveau, e.g. nous utilisons l'outil ImpulseC qui génère des IP avec interfaces FSL.
Enfin, les fichiers objets .elf et les fichiers de configuration des fpga .bit sont fusionnés puis téléchargés dans le système.

4. Cas d'étude

Le NoC Crossbus et le flot de conception présentés ont été validés sur deux cas d'étude. La première application est composée de 8 producteurs et 8 consommateurs. Chaque consommateur écrit en boucle dans un sc_signal lu par chaque consommateur. Le NoC est constitué d'un FPGA comprenant 4 routeurs. Chaque routeur est connecté à deux Microblaze, chacun réalisant une tâche de production et de consommation. Des résultats d'exécution temporelle sont présentés dans le tableau 1. Une seconde application, plus réaliste, a consisté à implanter un système de radio logicielle de type CDMA. Celle-ci est composée de 7 tâches.

Tableau 1: Producteur/Consommateur durée en nombre de cycle horloge de différentes actions en fonction de la taille des transferts

	4 mots	8 mots	16 mots
Initialisation du système	1385	2281	4057
MPI_init()	426	426	426
MPI_Comm_Rank()	1	1	1
Ordonnancement	60	60	60
Ecriture sc_signal	86	133	175
MPI_Barrier()	1	1	1

5. Conclusion

Les résultats obtenus montrent qu'il est possible d'implanter efficacement, sur une plateforme multi FPGA, une application décrite à haut niveau d'abstraction avec un paradigme multiprocesseur + NoC. Les optimisations permises par la conception conjointe du NoC avec le modèle de programmation associé sont particulièrement encourageantes : les performances obtenues sont meilleures que les autres approches NoC FPGA que nous avons pu évaluer. Pour étayer les résultats obtenus, nous projetons le développement d'un démonstrateur MPEG2.

6. References

[1] Micheli, G.D. & Benini, L. Wolf, W. (ed.) Networks on Chips M. Kaufmann, 2006
[2] T. Marescaux, J-Y. Mignolet, A. Bartic, W. Moffat, D. Verkest, S. Vernalde, and R. Lauwereins. Networks on Chip as Hardware Components of an OS for Reconfigurable Systems. In Field-Programmable Logic and Applications, volume 2778/2003 of Lecture Notes in Computer Science, pages 595-605. Springer Berlin / Heidelberg, 2003.
[3] Riso, S.; Sassatelli, G.; Torres, L.; Robert, M. & Moraes, F. Réseau d'Interconnexion pour les Systèmes sur Puce : le Réseau HERMES SCS'04, 2004
[4] SPIRIT Consortium, "SPIRIT V2.0 Alpha release", 2006.
[5] Gropp, W.; Lusk, E. & Skjellum, A. Using MPI (2nd ed.): portable parallel programming with the message-passing interface MIT Press, 1999.