Plate-forme de Conception d'Architectures Reconfigurables Dynamiquement pour le Domaine du TSI

Julien LALLET, Sébastien PILLEMENT, Olivier SENTIEYS

IRISA-CAIRN
6 rue de kerampont, BP 80518, 22305 Lannion, France
lallet@irisa.fr,pillemen@irisa.fr,sentieys@irisa.fr

Résumé – Le travail présenté ici se place dans le contexte de l'aide à la conception et à l'exploration d'architectures reconfigurables dynamiquement. Nous présentons la plate-forme de développement MOZAÏC qui permet la conception d'architectures reconfigurables dynamiquement. Plus particulierement, cette plate-forme permet l'introduction automatique de ressources matérielles dédiées et adaptées à la mise en œuvre de l'ensemble des mécanismes et du contrôle de la reconfiguration dynamique.

Abstract – The area of this work is the architecture description, computer-aided design (CAD) and exploration of dynamically reconfigurable architectures. This document presents the development framework MOZAÏC which aims at designing dynamically reconfigurable architecture. More especially, this framework aims at automatically generating the specific hardware resource needed to applicate dynamic reconfiguration and to control it on any kind of architectures.

1 Introduction

L'évolution constante des applications et le besoin toujours croissant de performances imposent le développement de nouvelles architectures compétitives et évolutives au sein de systèmes reconfigurables dynamiquement sur puces. Ces contraintes ont amené à une augmentation de la complexité des architectures, de leurs mécanismes de reconfiguration et de leur conception. De manière à répondre efficacement à ce problème, des plate-formes de développement ont été conçues [9] [11] et permettent ainsi d'automatiser certains processus constituant la chaîne de conception d'une architecture. Cela est rendu possible par l'intermédiaire de langages de description haut niveau (ADL) qui permettent, par une spécification rapide de certains paramètres matériels, de procéder rapidement à la génération d'une architecture et de ses outils de développement adaptés tels que des outils de simulation, de compilation ou encore de synthèse. Le travail présenté ici se place dans le contexte de l'aide à la conception et à l'exploration d'architectures reconfigurables dynamiquement. Pour cela, la plate-forme de développement MOZAÏC permet la conception d'architectures reconfigurables dynamiquement par l'introduction automatique de ressources matérielles dédiées et adaptées. La figure 1 montre l'intégration de MOZAÏC avec une plate-forme hôte de développement d'application. L'architecture ciblée pour une application donnée nécessite la validation de contraintes temporelles et architecturales. La prise en compte de ces contraintes par MOZAÏC permet la génération automatique des ressources matérielles nécessaire à la mise en œuvre d'une implémentation dynamique de l'application ainsi que la génération du flots de données de configuration.

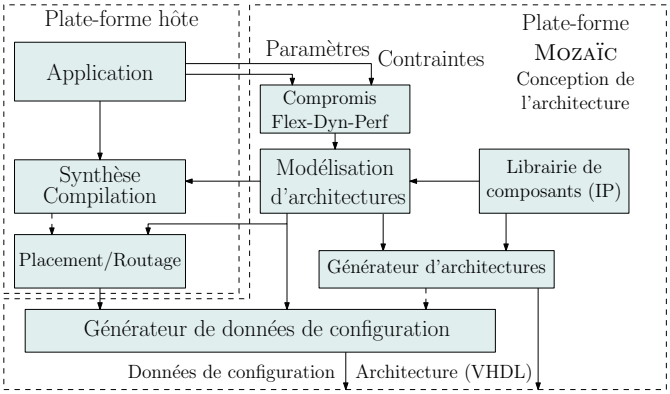

FIGURE 1 – Intégration de MOZAÏC avec une plate-forme de conception hôte.

Dans une première partie, nous présentons les concepts du modèle générique de reconfiguration dynamique qui ont été développés et mis en œuvre dans MOZAÏC après avoir présenté un état de l'art des architectures reconfigurables dynamiquement et de leurs outils associés de conception. Dans une deuxième partie, nous présentons l'ADL xMAML qui permet la spécification de l'architecture et de l'exploitation efficace des mécanismes précédemment présentés. Enfin, dans une dernière partie, nous présentons l'implémentation d'un décodeur WCDMA par reconfiguration dynamique sur un modèle xMAML d'un FPGA.

TABLE 1 – Classement des architectures reconfigurables dynamiquement en fonction de leur critères de granularité et de méthodologie de reconfiguration.

ARCHITECTURES RECONFIGURABLES DYNAMIQUEMENT		GRANULARITÉ	
		FIN	ÉPAIS
MÉTHODE DE RECONFIGURATION	MONO-CONTEXTE	AT40K [2] Virtex [1]	DART [3] Systolic Ring [4] WPPA [Kissler06]
	MULTI-CONTEXTE	DPGA [5] ispXPGA [6] PiCoGa [7]	XPP [8] Adres [9] NEC-DRP [12]

2 État de l'art

Les architectures reconfigurables dynamiquement se différencient selon deux critères distincts :

– selon leur granularité,

– selon leur méthode de reconfiguration.

La granularité d'une architecture définie la taille des données traitées par les ressources de calcul de l'architecture. Les ressources de calcul traitant des données de un bit caractérisent les architectures *grains fin*, alors que les architectures traitant des données de plusieurs bits caractérisent les architectures *grain épais*. Les architectures grains fin sont réputées plus flexibles de part la possibilité de regrouper plusieurs ressources de calcul entres elles de manière à pouvoir traiter des données de plusieurs bits. En revanche, pour un même calcul, les architectures grains fin nécessiteront davantage de bits de configuration que les architectures grains épais, allongeant par conséquence les temps de reconfiguration.

Afin d'accélérer les phases de reconfiguration, certaines architectures permettent la sauvegarde locale de plusieurs configurations. Ainsi, le passage d'un contexte de calcul à un autre est réalisé en sélectionnant la configuration concernée. Cette méthode de reconfiguration est appelée reconfiguration multi-contexte et permet d'accélérer les phases de reconfiguration mais présente l'inconvénient de nécessiter davantage de ressources de mémorisation. Le tableau 1 présente une liste non exhaustive de quelques architectures reconfigurables dynamiquement et les classifie en fonction de leur méthode de reconfiguration et de leur granularité.

Le tableau 2 présente deux classes différentes d'outils dédiés aux architectures reconfigurables. La première classe recense quelques outils dédiés spécifiquement à une seule architecture. Cependant, ces outils présentent l'avantage de prendre en compte l'aspect dynamique de la reconfiguration. La deuxième classe recense les outils qui ne prennent pas en compte l'aspect dynamique de la reconfiguration. Cependant, ces outils ne se limitent pas spécifiquement à une architecture précise.

3 Modèle de reconfiguration dynamique

MOZAÏC permet l'intégration de ressources de calcul reconfigurables et hétérogènes. Suite à une analyse des ressources intégrées, MOZAÏC génère automatiquement l'ensemble des ressources matérielles nécessaires à la mise en œuvre de la dynamicité de la reconfiguration et de son contrôle. Cette mise en œuvre passe par l'intégration de ressources locales supplémentaires permettant une reconfiguration dynamique de type multi-contexte [12] présenté plus en détail dans [14]. La figure 2 montre de quelle manière les ressources de reconfiguration supplémentaires générés appellées DUCK (*Dynamical Unifier and Configurable blocK*) agissent comme interface entre la ressource à reconfigurer et le contrôleur de reconfiguration dynamique. Les ressources reconfigurables sont aussi bien des ressources d'entrée-sortie (IO_u), des ressources d'interconnexions (I_u) ou encore des ressources de calculs (C_u). L'ensemble des DUCK de l'architecture forment un chemin de données de configuration parallèle qu'il est possible de venir modifier pendant les phases de calcul. Cette configuration est effectuée par l'intermédiaire des ports $ConfigIn(i)$ aussi nombreux que nécessaire pour une configuration rapide, et récupérés par l'intermédiaire des ports $ConfigOut(i)$. Les différentes configurations contenues dans les différents DUCK sont ensuite échangées avec les configurations des ressources devant être reconfigurées.

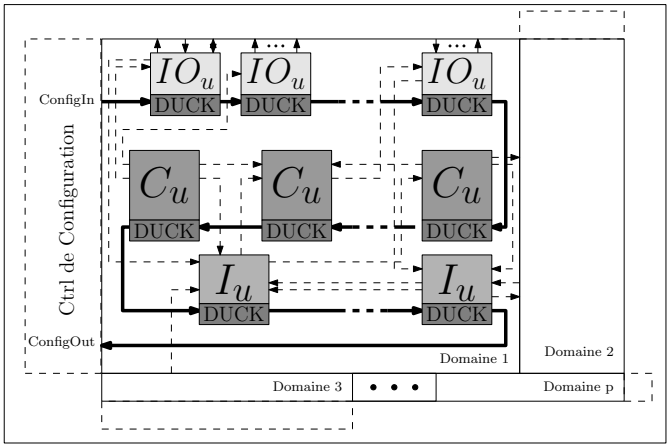

FIGURE 2 – Modèle d'architecture et de reconfiguration générique.

4 Modélisation xMAML

Afin de générer les DUCK adaptés aux ressources reconfigurables, il est nécessaire de procéder à la description des différentes ressources de calculs implémentées dans l'architecture. Cela est réalisé par l'intermédiaire du langage de description xMAML. xMAML est le fruit de l'introduction de deux nouveaux concepts et constitue une évolution du langage de description MAML [15]. Tout d'abord, xMAML introduit

TABLE 2 – Classement des outils dédiés aux architectures reconfigurables dynamiquement en fonction de la granularité et de la dynamicité de la reconfiguration.

OUTILS DÉDIÉS AUX ARCHITECTURES RECONFIGURABLES	GRANULARITÉ	
	FIN	ÉPAIS
RECONFIGURATION DYNAMIQUE - OUTILS DÉDIÉS	Xilinx (PlanAhead [10])	DART Dresc [Mei07]
RECONFIGURATION STATIQUE - OUTILS GÉNÉRIQUES	MADEO [11] VPR [16]	-

l'utilisation d'un réseau de connexion flexible permettant les communications entre les ressources de calcul quelque soient leur type. La specification xMAML des ressources d'interconnexion permet la génération complète d'un réseau d'interconnexion synthétisable. Le deuxième concept concerne la possibilité de procéder à une division de l'architecture modélisée en plusieurs espaces de reconfiguration (domaines) de manière à permettre la reconfiguration partielle et rapide de l'architecture. Les spécifications xMAML ne sont pas présentées ici. La specification de l'interface de reconfiguration d'une ressource de calcul implémentée avec un DUCK adapté et généré automatiquement se fait par l'intermédiaire de la partie xMAML appellée `PEInterface`. Dans l'exemple de la figure 3, une ressource de calcul de type bloc logique de FPGA à base de LUT est implémentée dans l'architecture. La reconfiguration de ce bloc nécessite 16 bits de configuration et est réalisée en 16 cycle d'horloge. La spécification des ports *ConfigIn*, *ConfigAddr* et *RW* nécessaires à l'adressage de la mémoire de configuration du bloc logique permet la génération matérielle du DUCK capable de procéder à l'interfaçage de cette ressource avec l'ensemble des ressources de reconfiguration dynamique de l'architecture.

```
1  <PEInterface name="clb">
2   <Reconfiguration cycle="16" bits="16" preemption="no
       "/>
3   <IOPorts>
4    <Port name="luti0" bitwidth="1" direction="in" type="
       data" />
5    ...
6    <Port name="ConfigIn" bitwidth="1" direction="in"
       type="RAMConfIn"/>
7    <Port name="RW" bitwidth="1" direction="in" type="
       RAMConfEn"/>
8    <Port name="ConfigAdre" bitwidth="4" direction="in"
       type="RAMConfAddr"/>
9   </IOPorts>
10 </PEInterface>
```

FIGURE 3 – Exemple d'une description xMAML de l'interfaçage d'une ressource de calcul

5 Validation

Cette section présente l'implémentation d'un décodeur WCDMA *Wideband Code Division Multiple Access* [13] sur une architecture reconfigurable dynamiquement de type FPGA embarqué. Trois fonctions principales, implémentées successivement par reconfiguration dynamique, sont utilisées par l'application WCDMA : *FIR (Finite Impulse Response) filter, Searcher*, et *Rake Receiver*. La fonction *Rake Receiver* est elle même décomposée en 4 sous fonctions *finger* et une sous fonction d'estimation de symbole. Les phases d'acquisition et de traitement se superposent de manière à ce que l'acquisition d'une série d'échantillon $n+1$ se fasse pendant le traitement de la série d'échantillons n. Une période d'acquisition d'un ensemble d'échantillon complet est réalisée en $t = 66.6\mu s$.

L'application a été synthétisée par les outils ABC Berkeley [17] (Tableau 3) et le placement routage a été effectué par l'outils VPR de Toronto [16]. À partir des résultats de ces deux outils, nous avons pu procéder à la génération des données de configuration adaptées à l'application WCDMA et à l'architecture des ressources de calculs que nous avons choisi d'implémenter. La fonction qui nécessite le plus de bloc logiques pour son implémentation est la fonction *searcher* avec 4953 blocs logiques. Par conséquent, 4953 blocs logiques sont suffisants pour l'implémentation dynamique des trois fonctions du décodeur WCDMA soit un temps d'implémentation par fonction et un temps de propagation de contexte disponible de $22.2\mu s$. Sachant que chaque bloc logique nécessite 30 bits de configuration, interconnexions comprises, sachant que la taille du bus de configuration reliant l'ensemble des DUCK est de 8 bits, alors $4953 \times 30/6 = 18574$ mots de 8 bits sont nécessaires pour l'implémentation de chaque fonction. Considérant que les différentes configurations sont stoquées dans une mémoire permettant une vitesse de lecture de 110 MHz, alors il en résulte que le nombre de domaines N_D nécessaire au respect des contraintes temporelles imposées par l'application est :

$$N_D = \left\lceil \frac{18574/110E^6}{22,2E^{-6}} \right\rceil = 8\ domaines \qquad (1)$$

Chaque domaine est composé de 620 blocs logiques. L'ensemble des 8 domaines sont nécessaires à l'implémentation des fonctions *FIR* et *Searcher*, alors que seul 5 domaines sont nécessaires à l'implémentation de la fonction *Rake Receiver* (1 do-

TABLE 3 – Résultats de synthèse des différentes fonctions du décodeur WCDMA

	FIR	*Searcher*	*Rake Receiver*	
			1 *Finger*	Complet
blocs logiques	3475	4953	561	4488
Total	12916			

maine par *finger* et un domaine pour l'estimation de symbole).

Si l'on compare la solution présentée avec une implémentation statique sur un FPGA commercial de type Xilinx, celle-ci aurait nécessité l'utilisation de 12926 blocs logiques. L'implémentation dynamique que nous présentons permet une économie de 7966 blocs logiques.

6 Conclusions

Dans cet article, nous avons présenté un modèle de reconfiguration dynamique flexible permettant un gain en terme de ressource d'implementation. Grâce au langage de description xMAML il est possible de définir l'interaction d'une ressource de calcul implémentée avec le modèle de reconfiguration dynamique. Les temps de reconfiguration sont également accélérés grâce à l'introduction du concept de ressource de reconfiguration DUCK qui permet une mise en œuvre d'une méthodologie de reconfiguration dynamique homogène. Ce concept est basé à la fois sur la séparation des chemins de configuration et des chemins de données permettant la propagation des contextes futures pendant les phases de calculs et à la fois sur une méthodologie de reconfiguration dynamique de type multi-contexte. Enfin, la gestion partitionée des différents chemin de configuration permet un contrôle partagé et accéléré des processus de reconfiguration effectués en parallèle. Les travaux futures visent l'intégration de la plate-forme Mozaïc à des architectures cible de granularité et de topologies différentes de celles présentées ici.

Références

[1] Xilinx, *XC4000E and XC4000X Series Field Programmable Gate Arrays*, May 1999.

[2] ATMEL, "AT40K FPGAs," ATMEL Inc., Tech. Rep., 1999.

[3] R. David, D. Chillet, S. Pillement, and O. Sentieys, "DART : A Dynamically Reconfigurable Architecture Dealing with Future Mobile Telecommunications Constraints," in *Procedings of IEEE Reconfigurable Architectures Workshop, (RAW),2002*.

[4] G. Sassatelli, L. Torres, P. Benoit, T. Gil, C. Diou, G. Cambon, and J. Galy, "Highly Scalable Dynamically Reconfigurable Systolic Ring-Architecture for DSP Applications," in *Procedings of the conference on Design, automation and test in Europe (DATE)*, 2002, pages 553–559.

[5] A. DeHon, "Dynamically programmable gate arrays : A step toward increased computational density," in *Procedings of the Fourth Canadian Workshop of Field Programmable Devices,1996*,

[6] Lattice, "ispXPGA Family," Lattice Semiconductor Corporation, Tech. Rep., 2007.

[7] A. Cappelli, A. Lodi, C. Mucci, M. Toma, and F. Campi, "A dataflow control unit for c-to-configurable pipelines compilation flow," in *Procedings of the 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2004, pages 332–333.

[8] V. Baumgarte, G. Ehlers, F. May, A. Nückel, M. Vorbach, and M. Weinhardt, "PACT XPP—A Self-Reconfigurable Data Processing Architecture," *Journal of Supercomputing*, vol. 26, no. 2, pages 167–184, 2003.

[9] B. Mei, A. Lambrechts, D. Verkest, J. Y. Mignolet, and R. Lauwereins. Architecture Exploration for a Reconfigurable Architecture Template. *IEEE Journal of Design and Test*, March.

[10] Xilinx, "Planahead software as a platform for partial reconfiguration," Xilinc Inc., Tech. Rep., 2008.

[11] L. Lagadec and B. Pottier. Object-oriented meta tools for reconfigurable architectures. *Reconfigurable Technology : FPGAs for Computing and Applications II*, volume 4212, pages 69–79, 2000.

[12] M.Suzuki, Y.Hasegawa, V. M. Tuan, S.Abe, and H. Amano. A Cost-Effective Context Memory Structure for Dynamically Reconfigurable Processors. In *Procedings of IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[13] T. Ojanpera and R. Prasad. *Wideband CDMA For Third Generation Mobile Communication*. Artech House Publishers, 1998.

[14] J. Lallet, S. Pillement, and O. Sentieys. Efficient Dynamic Reconfiguration for Multi-context Embedded FPGA. In *Procedings of Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2008.

[15] A. Kupriyanov, F. Hannig, K. Dmitrij, J. Teich, J. Lallet, O. Sentieys, and S. Pillement. Modeling of Interconnection Networks in Massively Parallel Processor Architectures. In Procedings of the *20th International Conference on Architecture of Computing Systems (ARCS)*, 2007, pages 268–282.

[16] V. Betz and J. Rose. VPR : A new packing, placement and routing tool for FPGA research. In Wayne Luk, Peter Y. K. Cheung, and Manfred Glesner, editors, in *Procedings of the Conference Field-Programmable Logic and Applications*, Lecture Notes in Computer Science (LNCS), vol. 1304, 1997, pages 213–222.

[17] J. Pistorius, M. Hutton, A. Mishchenko, and R. Brayton. Benchmarking Method and Designs Targeting Logic Synthesis for FPGAs. In *Procedings of the International Workshop on Logic and Synthesis (IWLS)*, 2007.