SETIT 2005
3rd International Conference: **S**ciences of **E**lectronic,
Technologies of **I**nformation and **T**elecommunications
March 27-31, 2005 – TUNISIA

Intégration d'un FPGA dans un système sur puce

Hayder MRABET , Wahid BAHROUN , Zied MARRAKCHI
et Habib MEHREZ

Laboratoire LIP6/ASIM Université Paris 6, 4 place Jussieu 75005 Paris. France.
Hayder.mrabet@lip6.fr
Wahid.bahroun@lip6.fr
Zied.marrakchi@lip6.fr
Habib.mehrez@lip6.fr

Résumé: Avec l'évolution des systèmes sur puce (SoCs) et l'augmentation de leur complexité, la possibilité d'introduire des changements après la fabrication devient intéressante. L'intégration d'un FPGA embarqué(eFPGA) dans un SoC permet de réutiliser une portion de la puce. Grâce à sa reconfigurabilité facile et illimitée un eFPGA est capable d'implanter des circuits spécialisés qui aide à améliorer les performances du système. Dans cet article nous présentons une méthode de conception d'un eFPGA . Nous présentons aussi l'environnement d'exploration et le mécanisme de fonctionnement de ce nouveau composant matériel comme étant une cible VCI.

Mots clés: eFPGA, SoC.

1 Introduction

Depuis deux décades les FPGAs (Filed Programmable Gate Array) sont devenus indispensables dans les systèmes numériques. Grâce aux cellules SRAM il est facile de reconfigurer un FPGA autant de fois pour implanter les fonctionnalités désirées. Les circuits reconfigurables ont évolués depuis leur apparition au début des années 1980. Leur capacité logique a énormément augmenté par rapport aux premières versions. Les FPGAs actuels sont capables d'implanter la totalité d'un processeur de traitement de signal.

De nos jours nous assistons à une grande évolutions des systèmes sur puces (SoC). Suivant ce courant, les fabricants des FPGAs sont arrivés à produire une nouvelle famille de circuits reconfigurables connue sous le nom de système reconfigurable sur puce (CsoC : configurable SoC). C'est un mélange de IP matériel tel que des processeurs, des blocs RAM etc. L'ensemble est noyé dans une structure programmable[1,2].

Face aux technologies ASIC les technologies reconfigurables ont des performances plus faibles et consomment plus d'énergie. A cause de ses inconvénients les FPGAs et les CsoCs sont toujours réservés aux prototypages et aux applications de volume réduit.

C'est la technologie des ASICs qui est la plus performante sur tous les niveau quant il s'agit d'applications à grand volume. Le seule obstacle devant les SoCs en technologie ASIC c'est le manque de souplesse, et l'impossibilité de faire des changement après la fabrication. Une erreur de conception ou un changement dans un modèle de communication standard nécessitera une nouvelle conception du système. La souplesse cherchée par les concepteurs des ASICs ne peut exister qu'en utilisant les circuits reconfigurables.

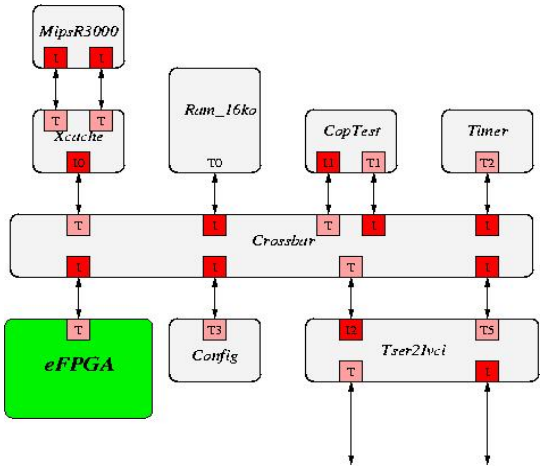

Figure 1 . *Système sur puce intégrant un eFPGA*

Récemment des vendeurs de IP [3,4,5] proposent des FPGAs embarqués pour systèmes sur puces (eFPGA). Intégrer un eFPGA dans un système sur puce permettra de réutiliser une portion de la puce après la fabrication afin d'implanter une variété de fonctions comme celles de traitement de signal et d'images. Un eFPGA peut être aussi considéré comme accélérateur de performance permettant d'implanter en matériel les parties critiques du code logiciel qui s'exécutent sur le processeur du système.

Dans cet article nous présentons une approche d'intégration de eFPGA cible sur système sur puce. Le système est présenté dans la figure 1.

Dans une première partie nous nous intéressons à l'architecture globale du eFPGA, la modélisation, et les phases de développement du modèle. La deuxième partie décrit l'environnement d'exploration de l'eFPGA et le mécanisme de fonctionnement global avant et après la configuration.

La dernière partie résume les réalisations effectuées et conclut le travail présenté avec une introduction des perspectives futures.

2 Architecture du eFPGA

2.1 La structure reconfigurable

La structure reconfigurable est un ensemble de blocs logiques connectés entre eux par des canaux de routage programmables. La structure adoptée pour notre eFPGA est de type matricielle comme le montre la figure 2. C'est une structure à base de cellules mémoires SRAM qui sauvegardent la configuration du circuit à implanter.

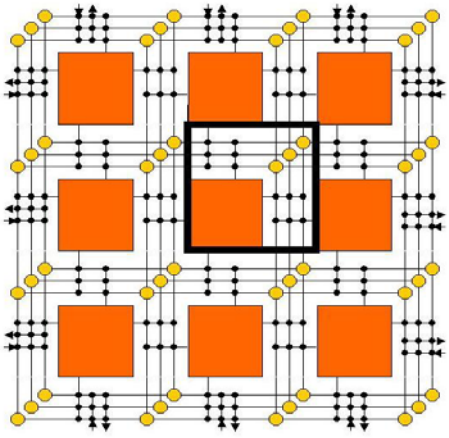

Figure 2 . *Architecture eFPGA matricielle*

La figure 3 montre l'élément logique de base (BLE). Il est composé de :

- une Look-Up-Table (LUT) capable d'implanter toute fonction booléenne à 4 entrées .
- Une bascule connectée à une horloge globale. Elle est utilisée dans les applications séquentielles.
- Un multiplexeur programmable de sortie pour choisir la sortie combinatoire du LUT ou la sortie séquentielle à travers la bascule.

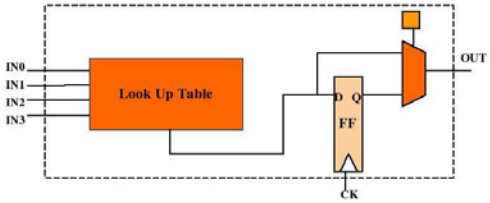

Figure 3 . *Elément Logique de Base (BLE)*

Les entrées du BLE sont accessibles depuis les canaux de routage qui entourent le bloc logique. L'accès se fait grâce à des multiplexeurs programmable qui permettent de choisir un fil parmi ceux du canal adjacent.

Les sorties sont aussi connectés aux canaux de routage et peuvent piloter un fil à travers des portes à 3 états (tri-states) programmables.

Les canaux de routages sont constitués de fils bidirectionnels connectés à la fois aux entrées/sorties des blocs logiques, et connectés entre eux grâce à des boites de commutation programmables.

Les boites de commutation programmables permettent de connecter les canaux sur les quatre faces et dans toutes les directions.

2.2 Modélisation en SystemC

Pour modéliser notre eFPGA nous avons adopté une méthodologie hiérarchique ascendante. La première étape consiste à développer les modules élémentaires de base en SystemC : le bloc logique, la boite de commutation et les boites de connexions qui permettent la connexion entre les entrées/sorties du bloc logique et les canaux de routage.

La création de la structure programmable consiste à rassembler les modules élémentaires pour former les groupes de base (encadré en trait fort dans la figure 2). Une simple duplication de ces groupes de base permettra de construire la structure programmable globale avec les dimensions désirées pour notre système. Les dimensions de la matrice correspondent au nombre de blocs logiques par ligne et par colonne. Ces dimensions sont fixées en fonction de la surface du silicium réservée pour l'eFPGA.

Un eFPGA englobe en plus de la structure programmable deux autres modules :

- Un module d'Entrée/Sortie qui assure l'interface entre la matrice programmable et l'interconnect du SoC. Ce module est développé pour communiquer à travers le protocole VCI standard[6]. Ce choix permettra d'intégrer notre eFPGA comme étant un nouveau composant matériel (IP core) réutilisable dans une plate-forme à base de composants SOCLIB[11] . SOCLIB est une bibliothèque "libre" de composants virtuels réutilisables pour la

conception de systèmes sur puce. Tous les composants de cette librairie communiquent sur VCI.

- Un module pour charger le fichier de configuration dans la structure programmable. Ce module récupère une série de mots de 32 bits qui correspondent aux données de configuration avec leurs adresses correspondantes. Ces données sont récupérées à travers le module d'E/S. Puis chargées dans la mémoire de configuration selon le protocole de configuration décrit dans le paragraphe 3.2 et présenté par l'automate de la figure 8.

La figure 4 montre la totalité du eFPGA composé de trois modules. Tous les modules sont décrits en SystemC.

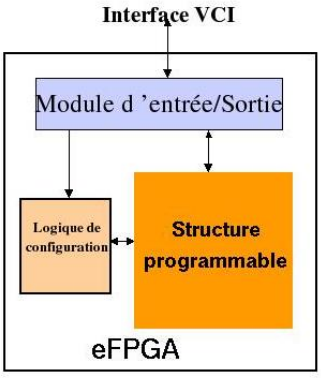

Figure 4 . *eFPGA comme étant IP VCI*

Le eFPGA est ensuite intégré dans le système de la figure 1 composé d'un bloc « RAM » (16 kO), un processeur « MIPS R3000», un module de « config », un « timer », un coprocesseur de test et un module qui assure l'interface vers un port série externe. L'interconnexion entre tous les composants est assurée grâce à un « crossbar ».

2.3 Du SystemC au Layout

La régularité de la structure programmable facilite la structuration de la matrice par simple aboutement des groupes de base dupliqués sur les colonnes et les lignes de la matrice. Un générateur automatique [8] permet la construction de la matrice programmable selon les dimensions désirées. La figure 5 montre le dessin de masque d'un exemple de matrice programmable placée et routée. La matrice de dimensions 4x4 a une capacité logique de 32 BLE sur une surface de dimension 3770λx3800λ (dessin symbolique) ce qui correspond à 0.453mm x 0.476mm en technologie 0.13µm.

Les modèles SystemC du chargeur et du module d'E/S sont transportés en VHDL comportementale puis synthétisés sur la librairie de cellules standards Sxlib[7].

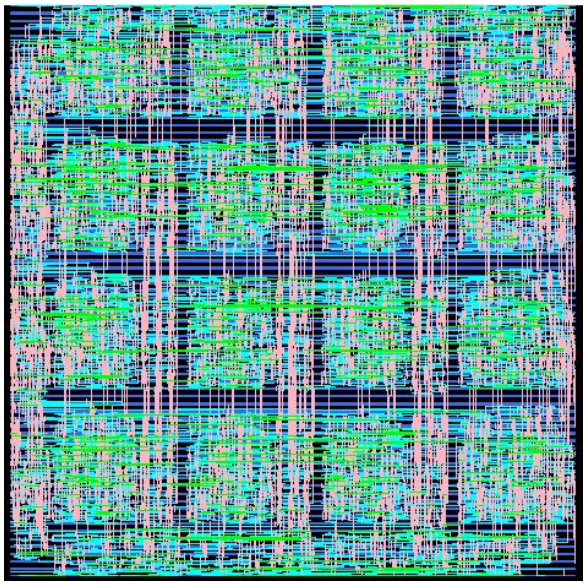

Figure 5 . *Dessin de masque routé d'une matrice programmable 4x4*

3. Exploration du eFPGA

3.1 Génération de fichier de configuration

L'utilisation et l'exploitation du eFPGA nécessite un ensemble d'outils automatiques pour générer le fichier de configuration qui sera chargé dans la matrice programmable afin de réaliser la fonction désirée. Le fichier de configuration contiendra la configuration des cellules SRAM de l'eFPGA.

A partir d'une description haut niveau en VHDL comportementale ou en machine d'états finis nous parvenons à générer le fichier de configuration qui correspond à cette description. Nous avons construit le flot complet (voir figure 6) qui inclut la synthèse et l'optimisation logique, le placement/routage sur la structure programmable et l'extraction du fichier de configuration.

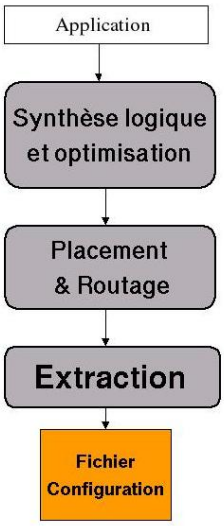

Figure6. *Flot de configuration du eFPGA*

La particularité de ce flot de configuration est qu'il n'utilise que des outils universitaires (open source) . La synthèse logique permet de générer des net-listes en portes logiques à partir du VHDL ou d'une description machine d'états finis et elle est effectuée grâce à l'outil BOOG[7]. Le résultat de la synthèse est optimisé et transformé en net-liste de blocs logiques grâce à SIS[9] et T-Vpack[10]. La net-liste de blocs logiques est ensuite placé/routé sur l'architecture cible grâce à VPR[10] comme le montre la figure 7.

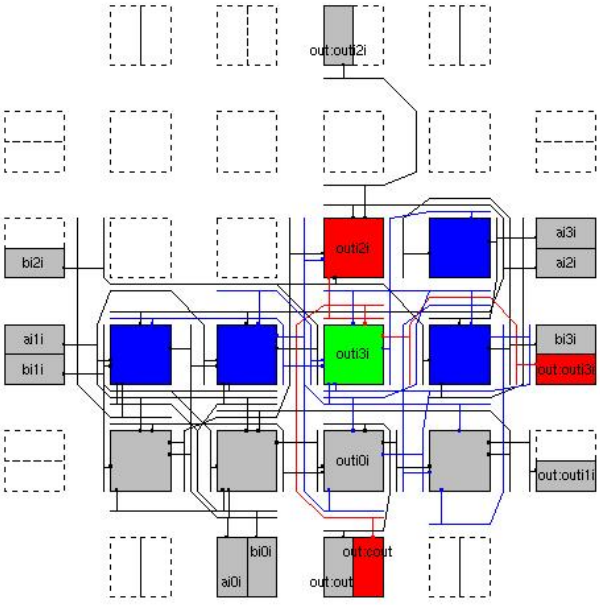

Figure 7. ***Résultat du placement et du routage d'une fonction combinatoire arithmétique qui sera implantée sur une matrice 4x4***

Finalement un extracteur générique que nous avons développé analyse tous les résultats précédents et génère un fichier binaire de configuration qui contient la donnée qui va être chargée dans la mémoire de configuration.

3.2 Mode configuration et mode fonctionnel

En mode configuration la matrice programmable est vue comme un bloc mémoire[8]. Ce bloc est décomposé en mot de 16 ou 32 bits répartis sur tous les groupes de la matrice. Chaque mot a sa propre adresse dont l'accès est aléatoire. Ceci permet une configuration rapide de l'eFPGA. Cette technique de chargement permet aussi de faire des configurations partielles et dynamiques de l'eFPGA.

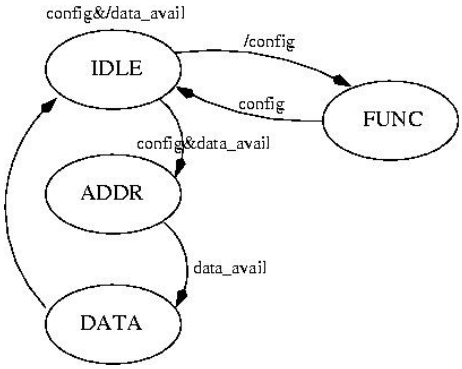

Figure8. *Automate du chargement de la configuration*

Pour programmer la matrice vierge, nous avons défini un protocole de configuration simplifié dans la figure 8. Le chargeur reçoit les données de configuration en mots de 32 bits précédés de leur adresses en mots de 32 bits.

Initialement tout le fichier de configuration est stocké dans la mémoire interne de notre système ou dans une mémoire externe accessible à travers l'interface série. Ensuite c'est le processeur MIPS qui active le **mode configuration** et joue le rôle de DMA. Des mots de 32 bits sont transférés de la mémoire jusqu'au chargeur. Ce dernier décode l'adresse et charge le mot correspondant.

Ce protocole n'est pas le plus performant en temps nécessaire pour le chargement (avec accés aléatoire), mais il est optimisé en terme de surface occupée par le chargeur (vu le nombre réduits d'états de l'automate). La surface nécessaire pour le chargeur est équivalente à la surface d'un seul groupe de base.

Une fois la configuration est réalisée le processeur MIPS établit la connexion eFPGA/VCI fonctionnelle en basculant la valeur du registre de configuration vers celle du **mode fonctionnel**. A partir de ce moment l'eFPGA est considéré comme un coprocesseur cible spécifique pour une fonctionnalité bien déterminée .

4. Test fonctionnel et validation

Le modèle SystemC de l'eFPGA est testé dans le système de la figure 1 complètement modélisé en SystemC.

Les simulations consistent généralement à charger différentes fonctions et applications sur l'eFPGA et à tester leur bon fonctionnement. Ces applications sont les fonctions qui prennent beaucoup de temps à s'exécuter sur le MIPS. Généralement ce sont les boucles qui correspondent à des fonctions arithmétiques et logiques . Selon la taille de la matrice, on peut implanter des multiplieurs 16 ou 32 bits par exemple. La figure 7 montre l'exemple d'un additionneur 4 bits sur une matrice 4x4.

Grâce au flot de configuration, il suffit de décrire n'importe quelle application en VHDL pour avoir rapidement la configuration adéquate à implanter.

Les mêmes simulations ont été effectuées sur le modèle eFPGA totalement en VHDL. Ce modèle qui remplace son équivalent en SystemC dans le même SoC de la figure 1.

La dernière étape consiste à valider le modèle en matériel. En effet nous avons intégré la totalité de l'eFPGA sur un FPGA Apex d'Altera. Le module d'E/S, le chargeur et la matrice programmable sur Apex , le reste du système est sur PC. Une liaison virtuelle entre la maquette Apex et le PC assure la communication entre l'eFPGA et le reste du système. Les mêmes simulations ont été réalisées sur cette dernière plate-forme.

Au niveau transistor pour toute dimensions de la matrice, la technique de génération du Layout est la même. La matrice est générée par aboutement des groupes de base. Le chargeur et le module d'E/S sont toujours synthétisés sur la librairie de cellules Sxlib, puis placés et routés automatiquement.

Les dimensions de ces modules en utilisant la librairie de cellules symboliques Sxlib sont : 550λx600λ pour le chargeur et 4500λx1700λ pour le module d'E/S avec l'interface VCI standard.

Conclusion

Dans cet article nous avons décrit l'implantation d'une IP VCI cible réutilisable. Le concepteur peut facilement générer l'eFPGA qui correspond à ses contraintes physiques et à la surface qu'il veut lui réserver sur silicium.

Nous avons présenté aussi le flot complet pour explorer l'eFPGA grâce aux différents outils existants et que nous avons ajoutés. Un flot automatique et flexible partant du VHDL jusqu'au fichier de configuration.

Nous avons défini aussi le mécanisme de configuration et de fonctionnement du eFPGA. Ce mécanisme est testé et validé sur un système sur puce complet. Le modèle eFPGA est validé en SystemC, en VHDL synthètisable et en matériel sur FPGA commercial.

Les perspectives de ce travail seront concentrées sur l'amélioration de l'interface d'un eFPGA dans un système sur puce et la réalisation d'un IP programmable qui peut être cible ou initiateur après la configuration.

Remerciements

Ce travail a été effectué en collobaration avec le CEA-DAM-DIF sous la direction de Monsieur André TISSOT. Les auteurs tiennent à remercier tous les responsables du CEA ainsi que François PECHEUX (ASIM/LIP6) pour leur soutien continu.

Références

[1] *Excalibur Device Overview* http://www.altera.com/literature/ds/ds_arm.pdf

[2] *PowerPC Embedded Processor Solution* http://www.xilinx.com

[3] Actel Corp, *Varicore Embedded Programmable Gate Array Core (EPGA) 0.18µ Family* . Datasheet, December 2001

[4] eASIC Corp, *FlexASIC 0.13µ Core*, http://www.eASIC.com .

[5] M2000, Inc, FLEXOStm Configurable IP Core.

http://www.m2000.fr

[6] *Virtual Socket Interface Alliance*, Virtual Component Interface Standard – Draft Specification, *v.2.2.0*, http://www.vsia.com, August 1997.

[7] ALLINCE CAD, http://www-asim.lip6.fr

[8] Mrabet & al., Automatic layout technique for scalable embedded programmable gate array , *ICEEC04, September 2004.*

[9] www-cad.eecs.berkley.edu/Software/software.html

[10] www.eecg.toronto.edu/vaughn/vpr/vpr.html

[11] http://soclib.lip6.fr