Plateforme multi-ASIP reconfigurable dynamiquement pour le turbo décodage dans un contexte multi-standard

Vianney LAPOTRE[1], Purushotham MURUGAPPA[2], Guy GOGNIAT[1], Amer BAGHDADI[2], Jean-Philippe DIGUET[1]

[1]Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance (Lab-STICC) CNRS UMR 6285
Université de Bretagne-Sud, Centre de recherche Christiaan Huygens, BP 92116 - 56321 Lorient Cedex, France

[2]Institut Telecom, Telecom Bretagne, CNRS UMR 6285 Lab-STICC
Electronics Department, Technopôle Brest Iroise CS 83818, 29238 Brest, France

[1]prénom.nom@univ-ubs.fr,[2]prénom.nom@telecom-bretagne.eu

Résumé – La multiplication des standards de communication sans fils introduit le besoin de récepteurs multi-standards reconfigurables. Afin d'adresser cette problématique et faire face à la demande croissante en débit des applications sur les terminaux mobiles, des architectures multi-ASIP ont été développées ces dernières années. De plus, l'évolution dynamique des paramètres de communication ainsi que la réduction du délai entre deux trames de données imposent la mise en oeuvre de mécanismes de reconfiguration optimisés. Dans ce contexte, nous proposons d'optimiser un turbo-décodeur multi-ASIP multi-mode et multi-standard dans un but d'optimisation globale des mécanismes de reconfiguration. Les résultats présentés montrent que les optimisations apportées à l'ASIP engendrent un faible surcoût en surface (0.004 mm^2 pour une technologie CMOS 65 nm) et permettent de réduire significativement la quantité de données nécessaire à la reconfiguration de la plateforme. En effet, pour une plateforme implémentant 8 ASIP, la quantité de données devant être diffusée aux ASIP lors d'un changement de configuration est divisée par 10 grâce aux optimisations proposées couplées à une infrastructure de configuration efficace.

Abstract – The emergence of many wireless standards is introducing the need of flexible multi-standard baseband receivers. To address this issue and to face the increasing demand of higher throughput for new greedy applications on mobile devices recent works propose multi-ASIP platforms for decoding algorithms. Furthermore dynamic evolution of communication parameters combined with the reduction of latency between two data frames imposes the need for an efficient reconfiguration management of such systems. In this context, we propose to tackle reconfiguration optimizations of a multi-standard and multi-mode ASIP for turbo decoding in order to improve the global reconfiguration management of a multi-ASIP platform. A comprehensive analysis concerning the area impact and dynamic reconfiguration performance is presented. Proposed ASIP configuration optimizations lead to a low area overhead of 0.004 mm^2 in 65 nm CMOS technology. For a multi-ASIP platform in which 8 ASIPs are implemented on a same device the configuration load is divided by ten thanks to both ASIP optimizations and an efficient configuration infrastructure.

1 Introduction

Les évolutions des derniers standards de communication sans fils visent à accroître les exigences en termes de débit, de robustesse et du niveau de convergence des services dans un terminal intelligent. A titre d'exemple, la norme de quatrième génération (4G) de téléphonie cellulaire sans fils vise à fournir une solution haut débit pour les modems d'ordinateur portable, les téléphones intelligents et autres appareils mobiles. Divers services tels que l'accès Internet à haut débit, la téléphonie sur IP, les jeux en réseaux et la diffusion en streaming de contenus multimédia sont visés.

Dans ce contexte de mobilité accrue et de haut débit, le codage correcteur d'erreurs est une technique clé des standards de communication sans fils. Les turbo codes sont fréquemment utilisés dans ce contexte afin d'atteindre un faible taux d'erreur et ainsi garantir les niveaux de performance requis. Par ailleurs, la contrainte de haut débit impose souvent une exploitation efficace des différents niveaux de parallélisme intrinsèques aux algorithmes de décodage. Dans ce contexte, une architecture multi-ASIP (Application-Specific Instruction-set Processor) [1, 2, 3] pour le turbo décodage correspond à une approche prometteuse afin d'atteindre à la fois une flexibilité élevée et un débit élevé. Le niveau de flexibilité de ces architectures multi-ASIP est assuré par la possibilité de charger dynamiquement une nouvelle configuration pour chaque ASIP de la plateforme et ainsi basculer d'un standard à un autre ou bien modifier les paramètres de décodage au sein d'un standard.

Plusieurs travaux proposent des solutions efficaces afin d'atteindre les niveaux de performance requis par les nouveaux standards. Cependant, les propriétés de reconfiguration dynamique de ces plateformes ne sont que partiellement adressées. En effet, bien que toutes ces plateformes puissent être reconfigurées dynamiquement à travers la mise à jour des mémoires de programmes et de configurations associées à chaque ASIP, les mécanismes de configuration ne sont pas optimisés et leur mise en oeuvre n'est pas suffisamment efficace pour une architecture multi-ASIP. Par ailleurs, l'augmentation continue des

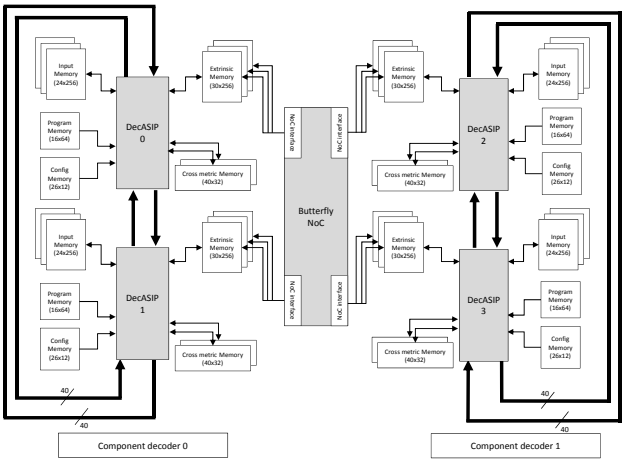

FIGURE 1 – UDec system architecture example with 2x2 ASIPs

débits de communication réduit le temps disponible entre deux trames de données afin d'effectuer une reconfiguration. Afin de faire face à ces contraintes antagonistes ; à savoir un nombre d'ASIP à reconfigurer croissant et un temps de reconfiguration décroissant du fait des débits très élevés, il est indispensable de proposer des plateformes reconfigurables dynamiquement plus efficaces. Dans cet article, nous proposons plusieurs optimisations afin d'adresser cette problématique dans un contexte multi-ASIP. L'approche proposée est illustrée à travers le processeur DecASIP et la plateforme multi-ASIP UDec présentés dans [3].

La suite de l'article est organisée de la façon suivante. La section 2 introduit la plateforme multi-ASIP UDec. La section 3 présente différentes optimisations permettant d'atteindre une reconfiguration efficace du DecASIP. La section 4 propose plusieurs résultats d'implémentation et discute les performances obtenues.

2 Plateforme multi-ASIP pour le turbo décodage

La plateforme multi-ASIP UDec, présentée en Fig. 1, est composée de deux rangées de processeurs DecASIP interconnectés par un réseau sur puce multi-étage d'une topologie de type papillon [3]. Chaque rangée de DecASIP correspond à un composant de décodage. Une rangée traite les données dans le domaine naturel et l'autre rangée traite les données dans le domaine entrelacé (parallélisme de décodage combiné, en plus du parallélisme de sous-bloc). La flexibilité du turbo décodeur provient de différents paramètres tels que le type de code : simple binaire (e.g. LTE) ou double binaire (e.g. WiMAX), la loi d'entrelacement, la structure du treillis, la taille des trames de données, le nombre d'itérations de décodage. Tous ces paramètres peuvent être mis à jour en fonction des performances à atteindre et des standards à exécuter. Les DecASIP et la plateforme UDec doivent donc être capables de supporter différentes configurations de ces paramètres en fonction du contexte d'exécution.

Afin de garantir une reconfiguration dynamique efficace de la plateforme UDec deux points fondamentaux doivent être adressés : 1) l'utilisation de DecASIP reconfigurables dynamiquement et 2) l'utilisation d'une architecture de communication efficace permettant de charger les nouvelles configurations (le détail de cette architecture de reconfiguration sort du cadre de cet article). En ce qui concerne la reconfiguration des DecASIP, plusieurs optimisations ont été proposées et sont présentées dans la suite de cet article.

3 DecASIP reconfigurable dynamiquement

Plusieurs optimisations ont été proposées afin d'aboutir à une reconfiguration dynamique efficace du DecASIP. La première optimisation concerne la sauvegarde des paramètres de configuration. En effet, dans la version initiale du DecASIP certains paramètres sont sauvegardés dans la mémoire de configuration alors que d'autres sont directement définis dans le jeu d'instructions du DecASIP. Une telle solution n'est pas efficace et implique de reconfigurer à la fois la mémoire de programme et la mémoire de configuration à chaque fois que de nouveaux paramètres doivent être utilisés. Une seconde optimisation concerne l'organisation des paramètres dans la mémoire de configuration. La solution actuelle n'est pas adaptée à une reconfiguration rapide du DecASIP. Une optimisation de la taille et de l'organisation mémoire permet de réduire significativement le nombre de cycles d'horloge nécessaires afin de reconfigurer un DecASIP et de façon plus large la plateforme UDec. La troisième optimisation concerne le développement d'un programme générique supportant plusieurs configurations. Cette approche permet de réduire le temps de basculement d'une configuration à une autre dans la mesure où le programme ne doit pas être rechargé.

3.1 Sauvergarde des paramètres de configuration

Afin de garantir une reconfiguration dynamique efficace, les paramètres présents dans la mémoire de programme sont déportés dans la mémoire de configuration. Cette solution permet de reconfigurer une seule mémoire (la mémoire de configuration) au lieu de deux. De plus, une fois le DecASIP configuré il est possible de charger en temps masqué les paramètres de la prochaine configuration dans la mesure où le programme n'accède pas à la mémoire de configuration mais directement aux registres internes lors de l'éxécution du programme. En effet, une fois la configuration terminée, les paramètres sont présents dans des registes internes du DecASIP. Cette solution permet de réduire le temps de reconfiguration mais induit une pénalité en surface (davantage de registres internes sont nécessaires afin de stocker tous paramètres liés à une nouvelle configuration). Ces pénalités sont analysées dans la partie résultats.

TABLE 1 – Mémoire de configuration optimisée

bit	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
@0	-																					Tail	ASIPId			
@1	Turbo Seed 0														Turbo Seed 1											
@2	-						TurboInitIteration				Maxiteration								State				NumSteps			
@3	Turbo Step 0														Turbo Step 1											
@4	Turbo Step 2														Turbo Step 3											
@5	Turbo Step 4														Turbo Step 5											
@6	Turbo Step 6														Turbo Step 7											
@7	-		@ Tail bits													Scaling Factor									Mode	
@8	Turbo PrevStep														Blocklength in bits											
@9	-			NumASIPs				StepIndex			WindowSize								LastWindowSize							
@10	-		CurrentWindowN_norm								CurrentWindowID_tail								WindowN_tail							

3.2 Organisation de la mémoire de configuration

Les paramètres présents dans la mémoire de configuration (Table 1) sont très spécifiques. Quatre types de paramètres peuvent être identifiés. Certains sont dépendants d'un domaine (liés aux DecASIP dans le domaine naturel ou aux DecASIP dans le domaine entrelacé), d'autres paramètres sont identiques pour tous les DecASIP, d'autres paramètres sont spécifiques à chaque DecASIP et enfin certains paramètres sont uniquement liés aux DecASIP traitant les bits de fin de trames. La taille de la mémoire de configuration ainsi que son organisation ont donc été analysées et optimisées afin de permettre une reconfiguration efficace des paramètres de décodage. En effet, l'organisation mémoire a un impact direct sur les mécanismes de diffusion des paramètres lors d'une reconfiguration. Par exemple, pour des paramètres identiques pour tous les DecASIP (@7 à @10 de la mémoire de configuration présentée en Table 1) un mécanisme de broadcast permet de réduire significativement le nombre de cycles nécessaires au chargement des paramètres dans les mémoires de configuration associées aux DecASIP. De même, un mécanisme de multicast peut être employé pour diffuser les paramètres dépendants d'un domaine (@2 à @6). Finalement, la diffusion des paramètres spécifiques à chaque DecASIP peut être réalisée via un mécanisme d'unicast. L'impact des choix retenus est détaillé dans la partie résultats.

La Table 2 présente différents choix d'implémentation de la mémoire de configuration en terme de largeur et de profondeur. Dans le but de prendre en compte l'implémentation de cette mémoire, la profondeur et la largeur considérées doivent être un multiple de 2 (ceci correspond à une contrainte minimale dépendante de la technologie considérée). Des solutions de mémoire avec une largeur de 14 à 32 bits ont été évaluées. L'étude des différentes architectures a permis de déterminer le meilleur compromis entre l'optimisation du temps de configuration, le taux de remplissage de la mémoire et l'impact global sur la plateforme. En effet, le temps de configuration de la plateforme est proportionnel à la profondeur de la mémoire de configuration puisqu'il dépend du nombre de lectures et du nombre d'écritures à réaliser, la largeur de la mémoire influence la taille des bus de communication entre la mémoire et l'ASIP ainsi qu'entre la mémoire et l'interface de configuration chargée de diffuser les configurations. La solution retenue est une mémoire

TABLE 2 – Mémoire de configuration : alternatives architecturales

Largeur (en bits)	Profondeur	Taux de remplissage (en %)	Nombre de lectures et écritures par configuration
32	10	79,1	12
28	12	75,3	13
26	**12**	**97,3**	**13**
24	16	65,9	17
20	18	70,3	19
16	20	79,1	22
14	22	82,1	24

d'une profondeur de 12 mots de 26 bits de large. Cette solution permet de maximiser l'utilisation de l'espace mémoire en proposant un taux de remplissage de 97,3% tout en proposant un faible nombre d'accès lors de la configuration des DecASIP.

3.3 Programme de turbo décodage générique

Afin de ne pas avoir à reconfigurer la mémoire de programme lors de chaque changement de configuration, un programme générique supportant les différents modes d'exécution a été proposé. Dans le cas de cette étude trois modes ont été considérés : deux modes pour du turbo décodage simple binaire et un mode pour du turbo décodage double binaire. Afin d'aboutir à la définition d'un programme générique le jeu d'instructions du DecASIP a été étendu afin de supporter notamment certaines instructions de test. D'autres instructions ont également été modifiées afin de ne plus considérer des paramètres en dur dans les attributs de l'instruction. Dans ce cas les paramètres liés à l'instruction sont présents dans des registres internes. L'optimisation du jeu d'instructions ainsi que la définition d'un programme générique permettent de réduire le temps de configuration tout en offrant un haut niveau de flexibilité.

4 Résultats d'implémentation

L'ensemble des optimisations présentées dans les sections précédentes ont été implémentées afin d'aboutir à la proposition d'un DecASIP reconfigurable. Le DecASIP a été modélisé avec le langage LISA en utilisant l'environnement de conception Processor Design tool de Synopsys. Les synthèses ont ensuite été réalisées pour une technologie CMOS 65 nm avec un objectif de fréquence d'horloge de 500MHz. Le DecASIP est constitué de 10 étages de pipeline (Fig 2). Les optimisations présentées dans ce papier n'affectent pas tous les étages du pipeline, seul trois étages ont été impactés. En effet, les étages du

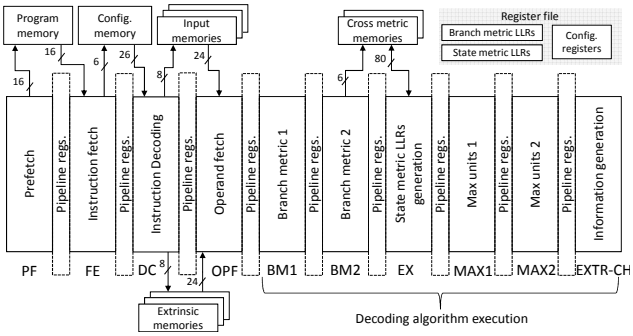

FIGURE 2 – DecASIP Pipeline

TABLE 3 – Comparaison des surfaces des étages du pipeline des ASIP en mm^2

	FE	DC	OPF	Total pipeline
DecASIP	0.001	0.05	0.003	0.078
Nouvel ASIP	0.0011	0.055	0.0037	0.080
Diff.	0.0001 +10%	0.005 +10%	0.0007 +23%	0.002 +2.6%

BM1 au *EXTR-CH* sont dédiés au décodage des données et ne sont donc pas directement impactés par les optimisations apportées à l'ASIP. De plus, l'étage pre-fetch n'a pas été affecté par la nouvelle implémentation. Les étages fetch (*FE*), decode (*DC*), et operand fetch (*OPF*) ont vu leur surface augmenter comparé à la version original du DecASIP. La Table 3 présente les résultats de synthèse de ces différents étages pour les deux version du DecASIP. Les différentes optimisations ont été implémentées de la façon suivante :

- L'étage FE assure la généricité du programme en incluant la gestion d'instructions de test.
- L'étage DC est impacté par le transfert des paramètres précédemment contenus dans les instructions vers la mémoire de configuration. En effet, ces paramètres doivent maintenant être lus depuis des registres internes créant de nouvelles connexions vers la file de registre.
- L'étage OPF est impacté par la nouvelle mémoire de configuration. En effet, les paramètres de configuration à gérer sont plus nombreux. Par conséquent, le nombre de connexions avec la file de registre a été augmenté.

TABLE 4 – Comparaison des surfaces des ASIP en mm^2

	Pipeline	file de registre	Total ASIP
DecASIP	0.078	0.076	0.175
Nouvel ASIP	0.080	0.087	0.179
Diff.	0.002 +2.6%	0.011 + 14.5%	0.004 +2.3%

La Table 4 présente les résultats de synthèse pour les ASIPs complets. L'impact des optimisations proposées est réduit puisque l'accroissement de la surface total du DecASIP est de 2.3% (soit 0.004 mm^2). Ce surcoût est principalement lié à l'ajout de registres internes afin de sauvegarder les paramètres de configuration. Du fait des modifications précédemment décrites, la surface du pipeline du DecASIP a également augmenté de 2,6%.

La Table 5 analyse le nombre de bits devant être reconfigurés afin d'obtenir une nouvelle configuration. Dans la solution initiale du DecASIP il était nécessaire de reconfigurer à la fois la mémoire de programme et la mémoire de configuration ce qui impliquait près de 1 kbits pour chaque ASIP. Dans les travaux de [4] qui proposent également une plateforme reconfigurable, le nombre de bits à charger est de 1463 bits auxquels s'ajoutent les tables d'entrelacement. Pour la nouvelle version du DecASIP, seuls 286 bits doivent être chargés pour chaque ASIP. Les optimisations proposées permettent ainsi de réduire fortement la charge lors de la reconfiguration d'un système multi-ASIP en utilisant des mécanismes de diffusion adaptés (broadcast, multicast, unicast).

TABLE 5 – Comparaison entre les nombres de bits à reconfigurer pour les mémoires de configuration et de programme

	Mem. Config	Mem. Prog	1 ASIP	n ASIPs
Nouvel ASIP	286	-	286	n.52+260+104
DecASIP	336	640	976	n.976
Gain	14%	100%	70%	90% ($n = 8$)
[4]	383 + Entrelac.	$\sim$1080	1463 + Entrelac.	n.1463 n.Entrelac.
Gain	25%	100%	80%	93% ($n = 8$)

L'ensemble des optimisations proposées permet pour une plateforme UDec composée de 4 DecASIP (2 dans le domaine naturel et 2 dans le domaine entrelacé) de garantir un temps de reconfiguration de $2\mu s$ via un bus de reconfiguration cadencé à 62,5 MHz. Un tel niveau de performance permet d'imaginer le déploiement de systèmes de communication sans fils haut débit qui s'adapteraient en temps réel à l'environnement d'exécution et aux besoins des utilisateurs.

5 Conclusion

Les architectures multi-ASIP pour le turbo-décodage sont une approche prometteuse pour atteindre un haut degré de performance et de flexibilité. Au vu de la montée en débit et de la réduction des délais séparant deux trames de donnée, l'optimisation de la configuration de telles architectures devient un enjeu important. Dans ce papier, nous proposons d'optimiser un ASIP multi-mode et multi-standard pour une reconfiguration efficace dans un contexte multi-ASIP. Les résultats montrent que le nombre de bits devant être configurés a été réduit de 70%. De plus, lorsque que 8 ASIP sont implémentés, le nombre total de bits à charger a été divisé par 10.

Références

[1] C. Brehm, T. Ilnseher, and N. Wehn, "A scalable multi-ASIP architecture for standard compliant trellis decoding," in *International SoC Design Conference (ISOCC)*, 2011, pp. 349 –352.

[2] T. Vogt, C. Neeb, and N. Wehn, "A reconfigurable multi-processor platform for convolutional and turbo decoding," in *Proc. of International Workshop on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, 2006, pp. 16–23.

[3] P. Murugappa, A.-K. R., A. Baghdadi, and M. Jézéquel, "A Flexible High Throughput Multi-ASIP Architecture for LDPC and Turbo Decoding," in *Proc. of Design, Automation and Test in Europe Conference & Exhibition (DATE)*, 2011.

[4] T. Vogt and N. Wehn, "A reconfigurable asip for convolutional and turbo decoding in an sdr environment," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 16, no. 10, pp. 1309 –1320, oct. 2008.