Décodeurs de codes polaires déclenchés par transfert de donnée

Mathieu LÉONARDON[1,2], Camille LEROUX[1], Pekka JÄÄSKELÄINEN[3], Christophe JÉGO[1], Yvon SAVARIA[2]

[1]Univ. Bordeaux, Bordeaux INP, IMS Lab, UMR CNRS 5218, France

[2]École Polytechnique de Montréal, QC, Canada,

[3]Tampere University of Technology, Tampere, Finland

mathieu.leonardon@ims-bordeaux.fr,camille.leroux@ims-bordeaux.fr,
pekka.jaaskelainen@tut.fi, christophe.jego@ims-bordeaux.fr, yvon.savaria@polymtl.ca

Résumé – Cet article présente un décodeur de codes polaires d'architecture déclenchée par transfert de donnée (TTA : Transport Triggered Architectures). Il supporte l'algorithme de décodage par annulation successive. Il a été validé sur cible FPGA et synthétisé pour une technologie ASIC 28 nm. Sa fréquence de fonctionnement est de 800 MHz et il atteint un débit de 352 Mbps pour un code polaire (1024,512). En comparaison de travaux précédents, l'énergie consommée est réduite d'un ordre de grandeur (0.14 nJ/bit) et le débit est multiplié par 5. En comparaison d'une implémentation logicielle sur un processeur généraliste (architecture x86), le débit est supérieur de 37 % et la consommation énergétique est réduite de deux ordres de grandeur. Le modèle de conception TTA peut être considéré comme un moyen de réduire l'écart entre les implémentations sur processeur généraliste et les implémentations matérielles dédiées.

Abstract – In this paper, a processor based on transport triggered architectures (TTA) is customized for the decoding of polar codes is proposed. It supports the Successive Cancellation (SC) decoding algorithm. It was prototyped and validated on FPGA. It was also synthesized in 28nm ASIC technology. It runs at a frequency of 800 MHz and reaches a throughput of 352 Mbps for a (1024, 512) polar code decoded with the SC algorithm. Compared to previous work, the energy consumption is reduced by one order of magnitude (0.14 nJ / bit) and the throughput is increased fivefold. Compared to an optimized software implementation on a general purpose processor (x86 architecture), the throughput is 37 % higher and the energy consumption is two orders of magnitude lower. TTA can be seen as a way to reduce the gap between programmable and dedicated polar decoders.

1 Introduction

Les codes polaires sont une famille de codes correcteurs d'erreurs inventée il y a une dizaine d'années [1]. Cette famille de codes correcteurs d'erreurs particulière est intégrée dans la nouvelle norme de communications mobiles 5G [2], pour les canaux de contrôle de certains scénarios de communication. Peu de temps après leur invention, les premières implémentations matérielles dédiées de décodeurs de codes polaires ont été proposées [3, 4]. De tels décodeurs, implémentés sur cible ASIC ou FPGA, constituent un ensemble de solutions efficaces en termes de débits, latences et consommation énergétique. À l'autre bout de l'espace de conception, certains travaux ont proposé des implémentations de ces algorithmes de décodage sur des architectures programmables, que ce soit des processeurs généralistes (GPP : General Purpose Processors) ou des GPU [5]. Si les débits atteints peuvent être très élevés, en particulier en ciblant des architectures multicœurs, la latence est supérieure aux architectures dédiées et la consommation énergétique est très élevée.

Entre ces deux approches, les processeurs à jeu d'instructions spécialisé (ASIP : Application Specific Instruction-set Processors) sont une approche intermédiaire pour trouver un meilleur compromis entre l'efficacité des architectures dédiées et la flexibilité des architectures programmables. Ces processeurs incluent des unités matérielles spécialisées très efficaces pour l'application visée, ici le décodage de codes polaires. De plus, les unités matérielles inutilisées sont supprimées pour limiter leur complexité matérielle.

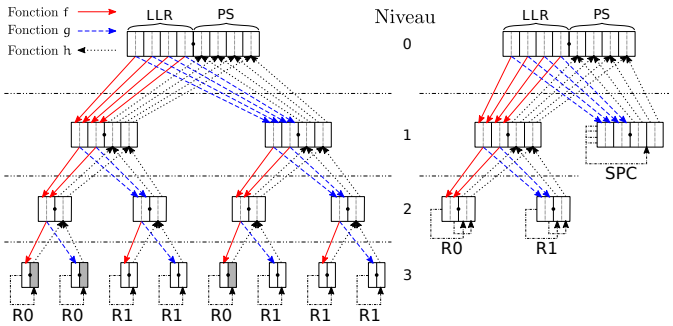

FIGURE 1 – L'arbre de décodage SC.

L'architecture du processeur est une architecture déclenchée par transfert de donnée (TTA : Transport Triggered Architectures). Un ASIP basé sur l'approche TTA est conçu et paramétré pour le décodage de codes polaires tout en conservant un jeu d'instructions généraliste. Un processeur TTA est consti-

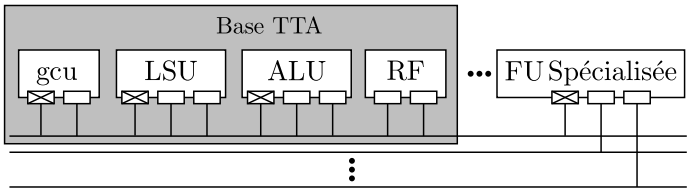

FIGURE 2 – Processeur TTA basique.

tué de nombreuses unités fonctionnelles (UF) connectés par un ensemble de bus configurables [6].

Les résultats d'implémentations montrent que les décodeurs proposés améliorent le débit et la consommation énergétique en comparaison avec les décodeurs programmables précédents, réduisant ainsi l'écart entre ceux-ci et les architectures matérielles dédiées. La suite de l'article est organisée de la façon suivante : la Section 2 détaille les algorithmes de décodage considérés. La Section 3 présente l'architecture du décodeur proposé. Les résultats d'implémentations sont présentés et analysés dans la Section 4. Enfin, la Section 5 conclue l'article.

2 Décodage de codes polaires

2.1 Décodage par annulation successive

L'algorithme de décodage SC peut être décrit comme la traversée d'un arbre binaire commençant par le nœud racine. Pour un code polaire de taille $N = 2$, l'arbre contient $\log_2 N + 1$ niveaux. Ces niveaux sont indexés de $l = 0$ (niveau le plus haut) à $l = \log_2 N$ (niveau le plus bas). Un niveau l contient 2^l nœuds. Chaque nœud contient 2^{n-l} rapports de vraisemblance logarithmiques (LLR : Log-Likelihood Ratios) notés L_i et 2^{n-l} sommes partielles (PS : Partial Sum), s_i. Les sommes partielles du niveau le plus bas contient $N - K$ bits gelés, représentés avec des cellules grisées dans la figure 1a, et K bits d'information. Les LLR d'un nœud sont mis à jour lorsqu'il est traversé dans le sens descendant. Les sommes partielles sont quant à elles mises à jour lors de la traversée dans le sens ascendant. Les règles de mise à jour sont listées dans les Eq. (1).

$$\begin{cases} f(L_a, L_b) & = & sign(L_a \times L_b) \times \min(|L_a|, |L_b|) \\ g(L_a, L_b, s) & = & (1-2s)L_a + L_b \\ h(s_a, s_b) & = & (s_a \oplus s_b, s_b) \\ \text{R1}(L_a) & = & \begin{cases} 0 \text{ if } L_a > 0 \\ 1 \text{ else} \end{cases} \end{cases} \quad (1)$$

2.2 Élagage de l'arbre

La figure 1b illustre la technique d'élagage qui a été proposée dans [7] et plus tard améliorée dans [8]. Cette technique réduit la complexité calculatoire du décodage de l'algorithme SC. Selon le sous-ensemble de bits gelés, certains sous-arbres peuvent être remplacés par un seul nœud. À ces nœuds particuliers sont associées des fonctions spécialisées. Par exemple, les nœuds de rendement 0 (`R0`) et de rendement 1 (`R1`) correspondent à des nœuds ne contenant respectivement que des bits gelés et des bits d'information. Les nœuds spécialisés appelés nœuds à répétition (`REP`) et nœuds de test de parité sur un bit (`SPC` : Single Parity Check) permettent également de réduire la complexité calculatoire de l'algorithme, sans effet sur les performances de décodage.

3 Décodeurs de codes polaires déclenchées par le transfert de données

3.1 Architectures déclenchées par le transfert de données

Les processeurs d'architectures TTA [6] sont un modèle original de processeur. La chaîne de co-conception matérielle/logicielle appelée TCE facilite la conception et le développement de tels systèmes [9]. Un processeur TTA est composé de blocs élémentaires appelés unités fonctionnelles (FU) connectées à un nombre de bus choisi par le concepteur comme montré dans la figure 2.

Une particularité des architectures TTA est qu'elle ne dispose pas d'un jeu d'instructions classique. En effet, le contenu d'une instruction TTA est seulement la description d'un transfert de données, depuis le port de sortie d'une FU vers le port d'entrée d'une autre. Les opérations de chaque FU sont seulement déclenchées lorsque ce port d'entrée est un port de déclenchement, auquel cas un traitement est déclenché au niveau de l'unité fonctionnelle correspondante. L'opération effectuée est une conséquence indirecte du transfert de données. La structure d'un TTA est essentiellement parallèle dans le sens où de nombreux transferts de données peuvent être réalisés en parallèle : À chaque bus peut être associé un transfert pour chaque cycle d'instruction.

3.2 Architecture du décodeur proposée

L'architecture proposée est illustrée dans la figure 3. La **Base TTA** permet à architecture proposée de pouvoir exécuter n'importe quel algorithme simple en fournissant un jeu d'instructions généraliste. Cette base est étendue avec différentes unités fonctionnelles de traitement parallèle. Un niveau de parallélisme $P = 64$ a été choisi, ce qui veut dire qu'un maximum de 64 opérations peut être réalisé en parallèle. Les LLR sont représentés sur $Q = 8$ bits afin de garantir des pertes de décodages négligeables par rapport à la version flottante. Des unités de chargement et sauvegardes **LSU** permettent des accès parallèles aux mémoires de données. Les **Unités de Calcul Polaire** implémentent les fonctions listées dans les Eq. 1 et les fonctions associées à l'élagage. L'**Unité SC déroulée** est un décodeur polaire de taille 8 permettant de décoder des sous arbres dans les niveaux inférieurs, et très peu parallélisables, de l'arbre de décodage comme proposé dans [10]. Enfin, une **File de Registres Vectoriels** est ajoutés pour stocker temporairement des variables vectorielles.

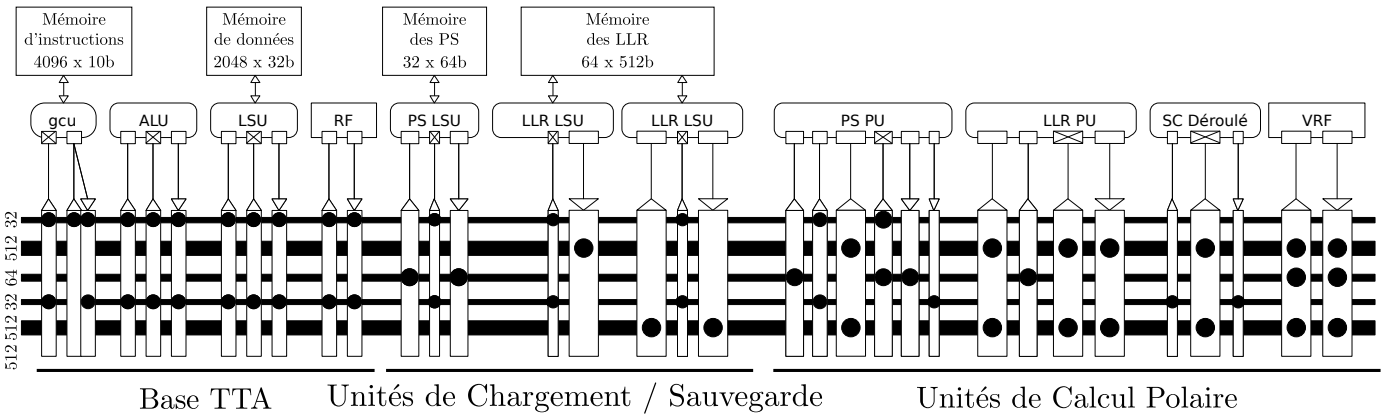

FIGURE 3 – Architecture du décodeur proposé.

3.2.1 Unités de chargement et sauvegarde

Dans l'architecture proposée, la taille maximum du code polaire est $N_{max} = 1024$ qui est la taille maximum de code polaire définie dans la norme 5G [2]. Comme mentionné dans la section 1, l'efficacité du décodeur est fortement corrélée au temps d'accès à la mémoire. Pour cette raison, les LSU du décodeur proposé sont des versions modifiées des LSU proposées par défaut par l'environnement TCE. Tout d'abord, la latence de chaque opération de chargement et de sauvegarde a été réduite de trois à un cycle en enlevant des registres internes. Ensuite, de la logique a été ajoutée afin de réaliser des chargements et des sauvegardes non alignées. L'ajout de ces instructions spécialisées n'affectent pas la latence de l'unité matérielle.

3.2.2 Unités de Calcul Polaire

Les unités de calcul polaire (PS PU et LLR PU dans la figure 3) réalisent les opérations spécifiques au décodage de codes polaires de manière parallèle. Elles peuvent être vues comme des instructions SIMD (Single Instruction Multiple Data) dédiées au décodage polaire. Une version étendue de la chaîne d'outils TCE, appelée TCEMC [11] a été utilisée afin d'en faciliter le développement.

Dans [10], les niveaux inférieurs de l'arbre de décodage sont réalisés par des décodeurs de sous-arbres déroulés dans lesquels des registres ont été ajoutés pour diviser le chemin critique. Dans le processeur TTA proposé, deux décodeurs déroulés similaires sont inclus dans une des FU. Elles permettent de décoder un sous-arbre de taille 8 en 6 cycles d'horloge seulement. Contrairement a ce qui est décrit dans [10], les sous-arbres déroulés n'incluent pas de registre mais une stratégie multi-cycles est utilisée. Ceci est rendu possible dans TCE par la définition de FU ayant plusieurs cycles de latence.

4 Résultats d'implémentation

Il est tout d'abord nécessaire de préciser qu'une comparaison fine avec des décodeurs existants est difficile puisque beaucoup de paramètres d'implémentation changent d'une implémentation à l'autre, que ce soit la technologie, les paramètres d'alimentation, les algorithmes supportés, les stratégies d'élagage, la quantification, les ensembles de bits gelés. Le but de cette section est de prouver que l'architecture TTA proposée est un compromis naturel entre architectures programmables et dédiées dans le cas du décodage polaire. Nous montrons également que ce travail est une amélioration par rapport à nos travaux précédents [12] en termes de débit et d'efficacité énergétique. Le décodeur polaire déclenché par le transfert de donnée (`TTPD` : Transport Triggered Polar Decoder) a été synthétisé avec la bibliothèque de cellules standard ST-28nm FD-SOI, 0.9V, 125°C. Le tableau 1 compare le décodeur `TTPD` avec l'état de l'art des implémentations de l'algorithme SC sur des processeurs généralistes fournie par la chaîne d'outil AFF3CT [13]. L'implémentation sur Intel i7-4712HQ est une solution atteignant un haut débit au prix d'une consommation énergétique importante. La solution sur ARM réduit la consommation mais les débits sont inférieurs. La solution LX7 utilisant les outils Tensilica [12] permet de réduire la consommation d'un ordre de grandeur par rapport à l'ARM mais les débits atteints sont faibles. Le décodeur proposé `TTPD` atteint un débit de 352 Mbps, ce qui est supérieur d'environ 40 % au processeur i7 considéré tandis que la consommation énergétique est inférieure de deux ordres de grandeur.

Le décodeur `TTPD` peut également être comparée à des implémentation dédiées sur cible ASIC et FPGA. 1161 cycles sont nécessaires en comparaison aux 222 nécessaires à l'architecture proposée dans [8]. Les deux facteurs principaux sont i) un élagage plus complet avec utilisation des nœuds REP et SPC et ii) dans le décodeur `TTPD`, le chemin de données depuis le port de sortie de la mémoire, passant par les unités de calcul et revenant vers le port d'entrée de la mémoire, est fractionné par de nombreux registres, ce qui n'est pas le cas dans les architectures dédiées. En termes de complexité, le décodeur `TTPD` est compétitif. L'empreinte mémoire supérieure est due en majeure partie à la taille de la mémoire d'instructions qui pourrait être réduite, ceci ayant cependant un impact sur la flexibilité de l'architecture.

Le tableau 2 montre quelques résultats d'implémentations

TABLE 1 – Comparaisons des débits, des latences et de la consommation énergétique de processeurs programmables exécutant l'algorithme SC ($R = 0.5$)

Architecture	N	Latence [μs]	Débit [Mb/s]	E_b [nJ/bit]
i7-3.3GHz	1024	2.0	257	41
	512	1.2	210	49
(GPP)	256	0.7	179	59
	128	0.4	143	73
A57-1.1GHz	1024	10.7	48	17
	512	5.3	48	17
(GPP)	256	2.8	46	17
	128	1.6	41	20
LX7-835MHz	1024	7.2	71	1.6
	512	3.9	66	1.7
(ASIP)	256	1.9	65	1.7
	128	1.0	62	1.8
TTPD-800MHz	1024	1.4	352	0.14
	512	0.8	313	0.15
(ASIP)	256	0.4	304	0.16
	128	0.2	284	0.17

TABLE 2 – Implémentations ASIC de décodeurs SC dédiées comparées au décodeur TTA pour un code polaire (1024,512)

	TTPD	[14]	[4][1]
Cible	28nm	28nm	28nm
Cycles d'horloge	1161	1833	1568
IT/P [Mb/s]	352	94	436
Freq [MHz]	800	336	1335
Puissance [mW]	48	18	5
$\mathbf{E_b}$ [nJ/bit]	0.14	0.19	0.011
Surface [mm^2]	0.16	0.44	0.04
Élagage'	R0 & R1	Premier R0	Aucun

[1] La mise à l'échelle de 180nm à 28nm de [4] est issue de [14].

de décodeurs SC implémentés sur cible ASIC. La comparaison est rendue difficile car il existe peu d'implémentations comparables dans la littérature. En effet, le décodeur proposé dans [14] est la base d'un décodeur liste, et sa consommation énergétique et sa somplexité pourraient être réduites si l'algorithme SC seulement était supporté. De plus, une stratégie d'élagage très partielle est utilisée. Cela explique le débit inférieur obtenu. Le décodeur SC de [4] a été réalisé en utilisant une technologie ASIC moins récente et les résultats sont mis à l'échelle. Ainsi, le tableau 2 n'a comme seule ambition que de montrer que la complexité du processeur TTPD est raisonnable en comparaison de ces deux implémentations de la littérature. Il permet d'atteindre des débits de plusieurs centaines de Mb/s tout en conservant la grande flexibilité d'un processeur programmable.

5 Conclusion

Dans ce papier, le premier décodeur de codes polaires d'architecture TTA est proposé. Les débits obtenus sont supérieurs à ceux obtenus sur un cœur de processeur généraliste, tandis que l'énergie consommée est réduite de deux ordres de grandeurs. En comparaison d'un ASIP précédemment conçus, les débits sont augmentés d'un facteur 5 (352 Mbps) et la consommation énergétique réduite d'un ordre de grandeur (0.14 nJ / bit). Le décodeur conçu peut être considéré comme un bon compromis entre la performance des décodeurs dédiés et la flexibilité des implémentations logicielles sur processeurs généralistes. De futurs travaux s'intéresseront à l'implémentation de l'algorithme de décodage de codes polaires à liste.

Références

[1] E. Arıkan. Channel Polarization : A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels. *IEEE Transactions on Information Theory*, 2009.

[2] 3GPP. TS 38.212, Multiplexing and Channel Coding, 2017.

[3] C. Leroux et al. Hardware Architectures for Successive Cancellation Decoding of Polar Codes. In *ICASSP*, 2011.

[4] A. Mishra et al. A Successive Cancellation Decoder ASIC for a 1024-bit Polar Code in 180nm CMOS. In *IEEE A-SSCC*, November 2012.

[5] B. Le Gal et al. Multi-Gb/s Software Decoding of Polar Codes. *IEEE TSP*, Jan 2015.

[6] H. Corporaal. *Microprocessor Architectures : From VLIW to TTA*. John Wiley & Sons, Inc., 1997.

[7] A. Alamdar-Yazdi and F. R. Kschischang. A Simplified Successive-Cancellation Decoder for Polar Codes. *IEEE Comm. Letters*, 2011.

[8] G. Sarkis et al. Fast Polar Decoders : Algorithm and Implementation. *IEEE JSAC*, 2014.

[9] O. Esko et al. Customized Exposed Datapath Soft-Core Design Flow with Compiler Support. In *IEEE FPL*, 2010.

[10] B. Le Gal et al. A scalable 3-phase polar decoder. In *ISCAS*, May 2016.

[11] P. Jääskeläinen, T. Viitanen, J. Takala, and H. Berg. *HW/SW Co-design Toolset for Customization of Exposed Datapath Processors*, pages 147–164. Springer International Publishing, 2017.

[12] M. Léonardon et al. Custom Low Power Processor for Polar Decoding. In *ISCAS*, May 2018.

[13] A. Cassagne et al. Fast Simulation and Prototyping with AFF3CT. In *SiPS*, 2017.

[14] P. Giard et al. PolarBear : A 28nm FD-SOI ASIC for Decoding of Polar Codes. *IEEE JESTCS*, 2017.