Récepteur radio-logicielle efficace pour la démodulation de trames DCSS

Bertrand LE GAL[1] Guillaume FERRE[1]

[1]Laboratoire IMS, 351 Cours de la libération, 33405 Talence cedex, France

Résumé – Nous présentons dans cet article les premiers résultats de la mise en œuvre d'une forme d'onde de type LoRa intégrant un codage de données différentiel [1] qui rend l'estimation du décalage temporel indépendant du décalage fréquentiel. Cela permet également d'améliorer l'efficacité du récepteur LoRa en réduisant la sensibilité aux erreurs de synchronisation temporelle, mais également de rendre la démodulation insensible aux variations Doppler. Cette dernière caractéristique est particulièrement intéressante pour les communications par satellite en orbite basse. Les résultats d'implantation sur une cible ARM Cortex A72 montrent qu'il est encore possible d'améliorer le récepteur proposé en augmentant la complexité calculatoire du récepteur et en gardant un aspect temps réel. Cet article décrit les évolutions algorithmiques, les gains obtenus et les conséquences sur l'implémentation de la station de base en termes de (a) stations de base autonomes (ARM A72) et (b) stations de base centralisées de type Cloud-RAN (INTEL Xeon).

Abstract – In this paper, we present the first results of the implementation of a LoRa-like waveform incorporating a DCSS data encoding that allows an estimation of the time offset independently of the frequency offset. This improves receiver efficiency by reducing sensitivity to time synchronization errors. Implementation results on an ARM Cortex A72 target show that it is still possible to improve the proposed receiver by increasing the computational complexity of the receiver and keeping a real-time aspect. This paper describes the algorithmic evolutions, the gains obtained and the consequences on the base station implementation in terms of (a) autonomous base stations (ARM A72) and (b) centralized Cloud-RAN type base stations (INTEL Xeon).

1 Introduction

L'Internet des objets permet à un objet de se connecter à Internet, peu importe où il se trouve sur Terre. Afin de satisfaire à ce besoin, il doit pouvoir se connecter à des stations de base sur Terre, dans les zones urbaines, ou pouvoir utiliser des constellations de satellites en orbite basse (LEO pour *Low Earth Orbit*). Parmi les technologies de communication existantes qui répondent à ces besoins, il y a LoRa, qui utilise des signaux vobulés (*chirp* en anglais). En effet, LoRa est basée sur la modulation *Chirp Spread Spectrum* (CSS). Cette dernière permet sous certaines configurations de concevoir des récepteurs ayant des sensibilités jusqu'à -140 dBm permettant des communications sur de longues distances. Cependant, les modes de communication qui offrent de telles sensibilités sont très sensibles aux variations de fréquence pouvant apparaître lorsque l'objet ou le récepteur se déplacent.

Lors des communications avec les satellites LEO, l'effet Doppler entraîne des pertes de sensibilité significatives. Pour démoduler ces signaux, les récepteurs doivent suivre la variation Doppler. Une solution pour réduire efficacement cet effet lors de la réception d'un paquet consiste à utiliser un mode différentiel. Un récepteur prenant en charge cette approche, nommée DCSS pour *Differential CSS*, a été détaillé d'un point de vue théorique dans [1, 2]. Cet article analyse les capacités temps réel du récepteur décrit dans la littérature sur une cible logicielle (ARM), et propose aussi des évolutions algorithmiques permettant d'améliorer significativement les performances de réception.

L'article est organisé comme suit : dans la section 2, les traitements mis en œuvre dans le récepteur DCSS original ainsi que des améliorations algorithmiques sont présentés. Ensuite, les gains en termes de pouvoir de réception sont évalués dans la section 3. Enfin, la section 4 présente les caractéristiques temps réel des récepteurs sur différentes plateformes.

2 Récepteur de signaux DCSS

2.1 Proposition originale

Préambule : il ne s'agit pas dans cet article de détailler le principe de fonctionnement d'un modulateur CSS. Nous rappelons simplement les grandes étapes où à l'émission les symboles sont des entiers obtenus par codage de Gray de SF bits. Ces symboles servent ensuite à moduler un *raw chirp* linéaire qui parcourt une bande B pendant le temps symbole. L'enveloppe complexe transmise consiste donc en une succession de *chirps* générés de façon à ce que la phase soit continue. Au niveau du récepteur, une fois synchronisée, l'enveloppe complexe est démodulée par *dechirping*, opération consistant à multiplier chaque tranche de signal de durée symbole par le *raw chirp* conjugué. Après cette opération, le symbole est estimé par recherche de *argmax* dans une FFT en mode cohérent ou non cohérent. Le lecteur peut se référer à [1].

Le récepteur proposé dans [1] rend la détection des symboles CSS reçus robuste aux erreurs de synchronisations grâce à un mode différentiel. Les symboles ne sont alors pas transmis directement. Il s'agit plutôt d'utiliser un intégrateur pour qu'au niveau du récepteur, ils puissent être récupérés par différenciation. L'émetteur DCSS génère séquentiellement des *chirps* basés sur les symboles D_p obtenus comme suit :

$$D_p = (S_p + D_{p-1}) \mod M \text{ for } p \geq 0, \qquad (1)$$

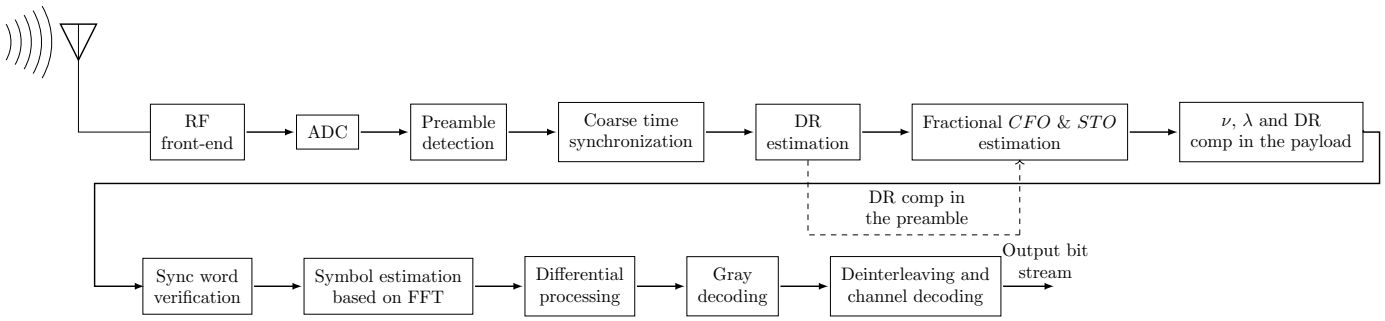

FIGURE 1 : Architecture du récepteur proposé dans [1, 2].

avec S_p a été défini comme le symbole LoRa transmis à l'instant pT. Du côté du récepteur, l'estimation de $\left\{\hat{S}_p\right\}_{p\geq 0}$ est obtenue comme suit :

$$\hat{S}_p = (\hat{D}_p - \hat{D}_{p-1}) \mod M \tag{2}$$

L'émetteur DCSS est fondamentalement semblable à l'émetteur LoRa, puisque les mêmes structures de signaux à phases linéaires et continues sont utilisées. Du côté du récepteur, il est uniquement nécessaire d'estimer et de compenser le *carrier frequency offset* (CFO) fractionnaire ν et le *sample time offset* (STO) fractionnaire λ pour éviter une dégradation des performances. Un processus de synchronisation mettant en œuvre les six étapes principales (Fig. 1) a été détaillé dans [1, 2] :

la détection d'un préambule (PD) - cette opération multiplie les blocs de M échantillons entrants par le conjugué des *chirps* utilisés par l'émetteur. Ensuite, une FFT est calculée sur chaque bloc de M échantillons qui ne se chevauchent pas. Cela permet de calculer la moyenne des amplitudes FFT des blocs successifs avant d'appliquer la fonction $argmax$. En effet, dans le préambule les symboles sont identiques, cela permet de détecter une trame à $\pm M$ échantillons. la synchronisation temporelle gros grain (CTS) - avant de commencer la démodulation, il est nécessaire de procéder à un affinement de la synchronisation temporelle afin d'éviter les interférences entre symboles. Pour ce faire, une recherche dichotomique sur l'espace des $2 \times M$ échantillons est réalisée avec comme objectif de maximiser l'énergie. l'estimation de l'effet Doppler (DR) - après avoir effectué cette synchronisation temporelle, le récepteur estime le DR pour éliminer le décalage de fréquence dans le temps. l'estimation fractionnaire du STO - un alignement précis nécessite d'estimer le STO fractionnaire (lié au suréchantillonnage d'un facteur α) après les calculs du CFO fractionnel et du DR sont estimés. la compensation du CFO fractionnaire et du STO fractionnaire - après les estimations du CFO et du STO, l'ensemble des échantillons de la trame sont compensés avant de réaliser un sous-échantillonnage à $f_s/\alpha = f_{s_{min}}$ et d'entamer l'étape de démodulation.

Une description plus détaillée des algorithmes impliqués est fournie dans [1, 2]. La conception initiale du récepteur donnait la priorité à une faible complexité calculatoire, en réduisant la quantité de calculs FFT à réaliser lors du processus de synchronisation temporelle grossier. Cependant, ces simplifications réalisées pour répondre aux exigences temps réel ont un impact sur les performances du récepteur. Cependant, après avoir exploité, par exemple, les unités SIMD qui permettent aux CPU de traiter plusieurs données à chaque cycle d'horloge [5] par souci d'efficacité, avec la majeure partie du code source décrite à la main à l'aide de SIMD intrinsèques [3], le système s'avère temps réel. Cela est possible, car le récepteur dépend fortement du traitement FFT, pour lequel nous avons tiré parti de la bibliothèque FFTW3 [4].

2.2 Évolutions algorithmiques

Le récepteur proposé dans [1, 2] fourni des niveaux de performances de réception intéressantes. Cependant, son implantation efficace sur des cibles basse consommation a permis d'envisager une augmentation de la complexité des algorithmes originaux pour améliorer ses performances de détection et de démodulation.

Par conséquent, les choix algorithmiques effectués dans [1, 2] ont été réévalués. Tout d'abord, l'approche de synchronisation temporelle gros grain a été remplacée par une approche hiérarchique sous la forme de grille (*grid*), afin d'améliorer la synchronisation temporelle au prix de $\approx 1,5 fois$ calculs de FFT. De plus, le système de traitement qui était initialement séquentiel à partir de (1) $\rightarrow$ (5) a été modifié. Le système actualisé recalcule certaines tâches après les avoir toutes exécutées une seule fois, ce qui réduit encore les erreurs. En d'autres termes, après l'étape (5), les étapes (2), (3) et (5) sont exécutées à nouveau. Cela améliore les performances de réception au prix d'un doublement de la complexité calculatoire.

3 Évaluation des performances

Cette section présente une comparaison des performances des récepteurs en fonction des algorithmes sélectionnés par rapport à l'approche originale [1, 2]. Afin d'identifier les meilleures solutions d'un point de vue des performances de démodulation et ensuite d'identifier les meilleurs compromis au regard des contraintes temps réel, différentes configurations du récepteur ont été testées. Les spécificités des récepteurs ainsi que leurs dénominations ($\mathcal{C}_i$) sont présentées ci-dessous :

- $\mathcal{C}_1$, il s'agit de la configuration initiale [1]. La détection (PD) est opérée par pas de M échantillons et l'estimation fine (CTS) est exécutée de manière dichotomique.
- $\mathcal{C}_2$ étend la configuration $\mathcal{C}_1$, mais la détection (PD) est opérée par pas de $M/2$.
- $\mathcal{C}_3$, cette configuration est équivalente à la configuration $\mathcal{C}_2$, mais la synchronisation (CTS) refaite après l'estimation et correction du CFO.

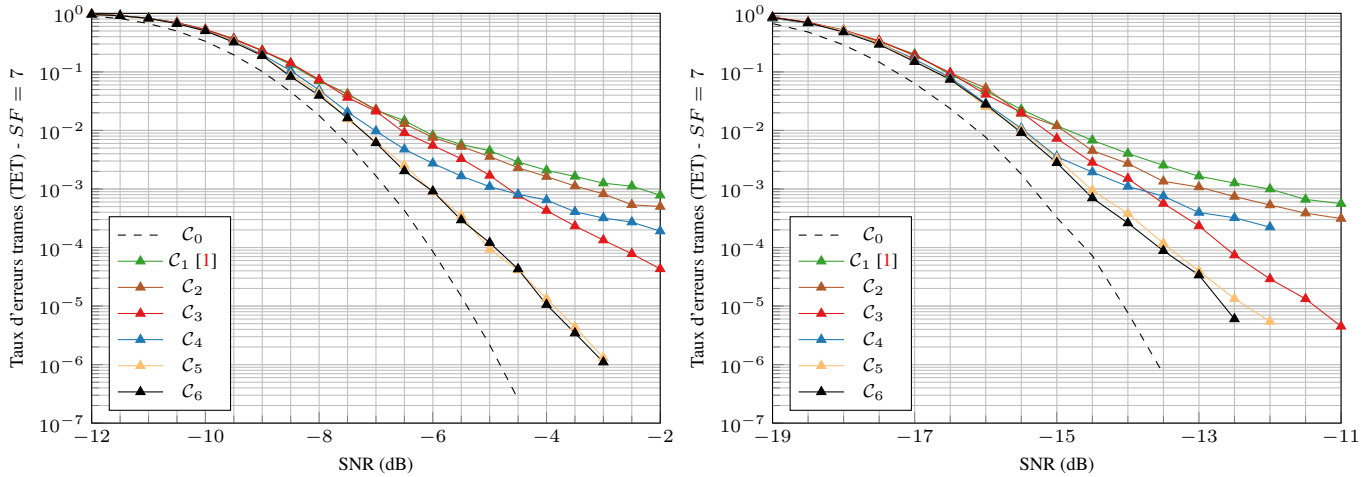

FIGURE 2 : Performances des décodeurs DCSS avec $SF = 7$ et $SF = 10$ en fonction des algorithmes exécutés. Bruit AWGN associé à des erreurs de CFO et de STO - $B = 125$ kHz, $\alpha = 8$, $N_p = 10$, $N_{sw} = 2$, $SFD = 1$, charge utile = 10 symboles.

- $\mathcal{C}_4$ reprend la configuration $\mathcal{C}_2$, mais la synchronisation (CTS) est réalisée hiérarchiquement par pas de 16 évaluations.
- $\mathcal{C}_5$ reprend la configuration $\mathcal{C}_4$, mais la synchronisation temporelle (CTS) est refaite une seconde fois après l'estimation et correction du CFO.
- $\mathcal{C}_6$ reprend la configuration $\mathcal{C}_5$, mais l'estimation (CTS) est faite de manière exhaustive sur une fenêtre de $2 \times M$. Cela correspond à l'optimal en termes de synchronisation.

Pour cette évaluation, nous avons fixé les paramètres applicatifs de la manière suivante : le nombre de *chirps* montant dans le préambule (N_p) est égal a 10, la bande (B) est fixée à 125 kHz et le facteur de sur échantillonnage (α) a été fixé a 8. La charge utile dans les trames $Nsymb = 10$. Lors des simulations, nous avons considéré que $\Delta\tau$, la désynchronisation temporelle, est uniformément distribuée dans l'intervalle $[-\frac{T}{2}, +\frac{T}{2}]$. La désynchronisation fréquentielle (Δf) est quant à elle comprise dans l'intervalle $[-B, +B]$. Le bruit est de type AWGN. La figure 2 présente les performances en termes de taux d'erreurs trames (TET) des récepteurs DCSS lorsque $SF = 7$ et $SF = 10$ en fonction de la valeur du SNR. Dans les simulations logicielles effectuées, les valeurs de CFO et STO évoluent de trame en trame.

Les résultats présentés dans la figure 2 démontrent que l'augmentation de la complexité calculatoire du récepteur permet d'améliorer les performances vis-à-vis de [1]. Le passage d'un pas de recherche $M \rightarrow \frac{M}{2}$ pour la synchronisation grossière ($\mathcal{C}_1 \rightarrow \mathcal{C}_2$) permet de gagner de 0.10 à 0.50 dB en fonction de la valeur de SF et de la valeur du SNR. Dans un second temps, la seconde synchronisation temporelle après compensation du CFO estimé ($\mathcal{C}_2 \rightarrow \mathcal{C}_3$) permet de regagner 0.5 à 3 dB. Remplacer la méthode de synchronisation fine réalisant un parcours hiérarchique d'un intervalle de $2 \times M$ associé à une bonification liée au nombre d'*argmax* identiques améliore de $1.0 \sim 1.5$ dB les performances du récepteur par rapport à la méthode originale ($\mathcal{C}_2 \rightarrow \mathcal{C}_4$). L'application de la double synchronisation à l'approche ($\mathcal{C}_4$) augmente notablement les performances du récepteur et annule l'apparition de planchers d'erreurs comme cela est visible ($\mathcal{C}_5$). Ainsi, la solution ($\mathcal{C}_5$) tend vers la solution optimale ($\mathcal{C}_6$), qui elle est fondée sur une évaluation exhaustive des désynchronisations temporelles possibles en effectuant 2×2^{SF} FFTs, et cela, pour un coût calculatoire moindre.

Le coût en termes de complexité calculatoire sera estimé dans la section suivante d'un point de vue pratique. Comme cela sera démontré, l'ensemble des approches présentées ici, à l'exception de la configuration $\mathcal{C}_6$, sont implantables sous contrainte en temps réel sur une architecture *low-power*.

4 Expérimentations temps réel

Pour évaluer l'efficacité du récepteur et la pertinence des algorithmes retenus dans un contexte temps-réel, le système complet a été implanté et déployé sur deux plateformes composées, d'un cœur ARM (Cortex-A72) et d'un cœur INTEL Xeon. L'expérimentation sur cœur ARM évalue la capacité du récepteur à fonctionner dans un contexte de type station de base autonome. En effet, la plateforme choisie intègre un processeur ARM Cortex-A72 (Broadcom BCM2711B0 SoC de la carte Raspberry Pi 4) qui fonctionne à une fréquence figée de 1.5 GHz. Il offre une "faible" puissance de calcul ainsi qu'une faible consommation d'énergie. À l'opposé, un processeur INTEL (Xeon Gold 6148) fonctionnant à 3.7 GHz est représentatif des systèmes de traitement déportés inclus dans des stations de type *Cloud-RAN*.

Afin d'atteindre des performances temps réel, des implémentations logicielles flexibles et efficaces en C/C++ de l'émetteur [6] et du récepteur ont été développées. Pour obtenir des performances optimales sur les supports d'exécution sélectionnés (ARM/INTEL), leurs fonctionnalités avancées, par exemple, les unités SIMD [5], ont été exploitées. Par souci d'efficacité, la majorité du code source a été écrit spécifiquement à l'aide des intrinsèques SIMD [3] offertes par chaque plateforme (NEON pour la cible ARM et AVX512 pour la cible INTEL). Ces optimisations manuelles ont permis d'accélérer notablement, par exemple, le calcul des convolutions complexes, la génération de la forme d'onde pour la compensation du CFO (sin/cos), la recherche d'*argmax*, etc. De plus, le récepteur nécessitant l'exécution d'un nombre important de FFT, nous avons évalué les bibliothèques open-source réalisant ce traitement. Ainsi, pour l'implantation sur cœur ARM, la bibliothèque sélectionnée est pfft tandis que pour la plate-

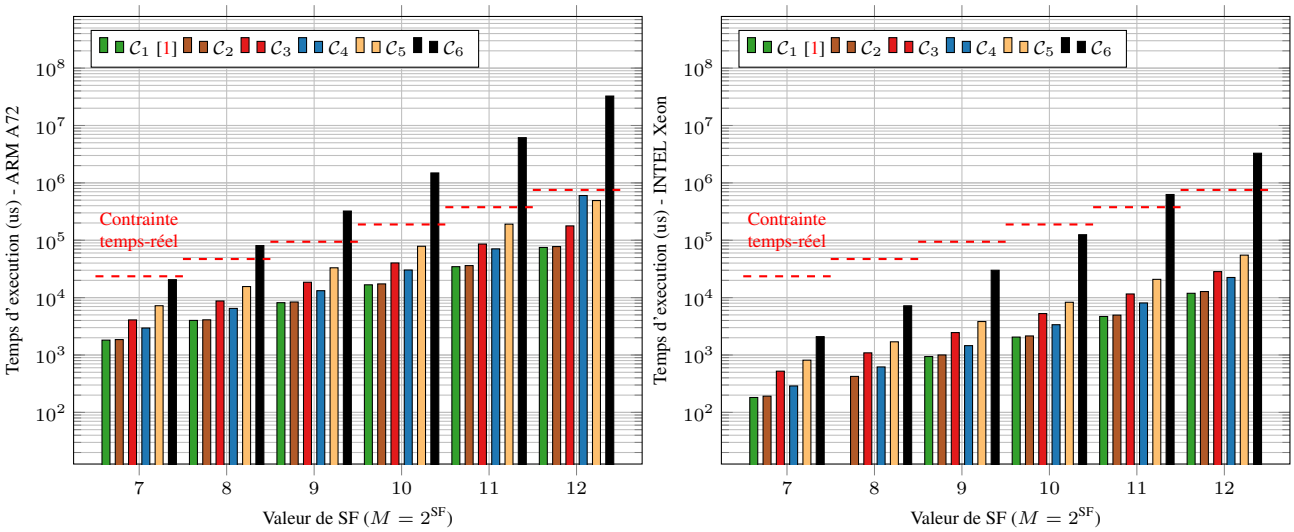

FIGURE 3 : Temps d'exécution pour traiter une fenêtre d'échantillons I/Q en fonction de l'algorithme et de la plateforme.

forme INTEL, la bibliothèque FFTW3 [4] a été retenue. Il est à noter que l'ensemble des algorithmes manipulent des valeurs flottantes en simple précision.

Les récepteurs ont été validés à l'aide de modules radiologicielle ETTUS B205 et HackRF ainsi que d'antennes adaptées aux bandes ISM utilisées. Les temps d'exécution nécessaires au traitement d'une fenêtre d'échantillons IQ de la taille d'une trame sont fournis dans la figure 3. Ces courbes fournissent les temps d'exécution des différents récepteurs en fonction de la valeur de SF. Il convient de noter que les temps de démodulation de la charge utile sont inclus dans les mesures. Les résultats montrent que les techniques de parallélisation et d'optimisation appliquées associées à la faible complexité calculatoire du récepteur $\mathcal{C}_1$ issu de [1] a permis sa mise en œuvre en temps réel sur le cœur ARM A72 dont il n'exploite que 7% à 10% des capacités de calcul en fonction de la valeur de SF. La version améliorée de ce récepteur (C3) respecte aussi les contraintes temps réel sur la plateforme ARM et consomme environ 17% à 24% des ressources, ce qui est cohérent vis-à-vis de l'augmentation de la complexité algorithmique induite par la synchronisation faite en deux temps. Le meilleur récepteur établi à partir une estimation exhaustive de la synchronisation temporelle ($\mathcal{C}_6$) quant à lui n'est pas temps réel. Il a un temps d'exécution trop long ($> 4300\%$ pour $SF = 12$). Le récepteur $\mathcal{C}_5$ qui offre un niveau de performance proche d'optimal ($\mathcal{C}_6$) respecte les contraintes temps réel pour l'ensemble des valeurs de SF avec un taux d'utilisation variant de 34% à 65%. Le gain en réception notable obtenu grâce à $\mathcal{C}_5$ (cf. figure 2) vis-à-vis de $\mathcal{C}_1$ [1, 2] se traduit donc par une augmentation d'environ $6\times$ au niveau du temps d'exécution.

La comparaison des temps d'exécution sur la plateforme INTEL offre des conclusions similaires. Cependant, comme cela était prévisible, le taux d'utilisation du cœur de processeur est bien plus faible. Ainsi, le récepteur $\mathcal{C}_5$ ne nécessite que $4\% \approx 8\%$ des ressources pour atteindre un fonctionnement temps réel. Ainsi, sur ce serveur possédant 40 cœurs physiques, il pourrait en théorie être possible de traiter jusqu'à ≈ 200 flux différents lorsque $SF = 12$ en considérant la réduction de la fréquence de fonctionnement.

5 Conclusion

Dans cet article, nous avons présenté un récepteur adapté aux signaux DCSS. Ce récepteur est une évolution de celui proposé dans [1] améliorant ses performances de réception tout en conservant une complexité calculatoire faible. Les performance démontrent que le récepteur proposé est plus résilient aux erreurs de démodulation en présence de CFO et de STO. De plus, ils évitent l'apparition de plancher d'erreurs permettant des gains de plusieurs dB lorsque des taux d'erreurs trame de 10^{-5} sont ciblés. Les résultats d'implantation des récepteurs sur des processeurs (ARM et INTEL) démontrent la faisabilité de ces derniers pour des applications temps réel grâce aux fonctionnalités avancées de ces processeurs (SIMD). Les travaux futurs se concentreront sur un raffinement des algorithmes de réception en virgule fixe et développement d'une implantation conjointe sur plateforme FPGA de type Zynq.

Références

[1] M.A. BEN TEMIM, G. FERRÉ et R. TAJAN : A novel approach to enhance the robustness of lora-like PHY layer to synchronization errors. *In Proc. of GLOBECOM*, 2020.

[2] M.A. BEN TEMIM, G. FERRÉ, R. TAJAN et B. LAPORTE-FAURET : A novel approach to process the multiple reception of non-orthogonal LoRa-Like signals. *In Proc of ICC*, 2020.

[3] P. ESTÉRIE, M. GAUNARD, J. FALCOU, J.T. LAPRESTÉ et B. ROZOY : Boost.SIMD : Generic programming for portable SIMDization. *In Proc. of PACT*, 2012.

[4] M. FRIGO et S.G. JOHNSON : The design and implementation of FFTW3. *Proceedings of the IEEE*, 2005.

[5] H. TANAKA et AL. : A new compilation technique for SIMD code generation across basic block boundaries. *In Proc of ASP-DAC*, 2010.

[6] L. VOLPIN, B. LE GAL et G. FERRE : Efficient LoRa-like transmitter stacks for SDR applications. *In Proc of ICECS*, 2022.