Une architecture de radio logicielle faible latence et basse consommation pour l'audio sans fil haute qualité

Robin GERZAGUET[1], Olivier ETRILLARD[2], Malo MABON[1], Laurent FEICHTER[2], Antoine COURTAY[1], Olivier BERDER[1].

[1]Univ Rennes, CNRS, IRISA Rue Kérampont, Lannion, 22300, France

[2]Feichter Electronics Technopole Anticipa, Lannion, 22300, France

robin.gerzaguet@irisa.fr;olivier.etrillard@feichter-electronics.com;malo.mabon@irisa.fr;
laurent.feichter@feichter-electronics.com;antoine.courtay@irisa.fr;olivier.berder@irisa.fr

Résumé – Nous proposons une nouvelle architecture de radio logicielle flexible, basse consommation, faible latence. Cette architecture s'appuie sur un microcontrôleur couplé à un coprocesseur réalisant une transformée de Fourier rapide, un FPGA et un émetteur-récepteur radio fréquence. Pour montrer l'intérêt d'une telle structure, nous proposons et caractérisons une couche physique dédiée à la transmission sans fil d'audio haute qualité et montrons que notre système tourne en temps réel avec un bilan énergétique réduit (437 mW pour le récepteur) et une latence de bout en bout très faible (854 μs).

Abstract – In this paper, we present a new architecture software defined radio for low power and low latency applications. By combining an on-the-shelf generic radio component with a low power microcontroller associated to a Fourier transform coprocessor. To prove the benefit of our approach, after describing the key assets of the architecture, we derive a complete physical layer dedicated to audio broadcast applications. This chain is capable of streaming High Definition audio stream in real time with low power (437 mW) and very low latency (854 μs).

1 Introduction

Le concept de radio logicielle date est formalisé par Mitola en 1993 [5]. Il y décrit une architecture mixte composée uniquement des antennes, de convertisseurs analogiques numériques large bande et d'unités de traitement logicielles. Depuis, de très nombreuses architectures ont pu être proposées dans l'objectif de gagner en flexibilité et en versatilité et les radios logicielles sont maintenant utilisées dans de nombreux contextes.

On limite souvent les radios logicielles à des cartes RF sans capacité de calcul (*tuners TNT*) ou des cartes RF interfacées avec des stations de calculs génériques. Il y a en réalité une multitude d'architectures différentes basées sur des processeurs génériques ou des processeurs spécialisés (FPGAs). Plus récemment les avancées sur les architectures embarquées ont permis l'émergence d'une nouvelle génération de radios logicielles autonomes sur systèmes sur puce avec des capacités de calcul importantes. Ceci permet maintenant d'utiliser les radios logicielles pour des applications haut débit dans des contextes très divers (sondage canal pour la radio cognitive, stations de base reconfigurables,...)

De prime abord les radios logicielles semblent tout à fait intéressantes et pertinentes pour le déploiement de l'Internet des Objets (ou *Internet of Things, IoT*). Leur grande flexibilité et leur coût potentiellement réduit sont des atouts précieux pour permettre d'assurer un déploiement massif, de garantir une maintenance légère et de permettre une grande ré-utilisation. Le problème est que ces radios sont généralement large bande, pilotées par des processeurs gourmands en énergie ce qui rend leur utilisation peu adaptée en tant que terminal en bout de chaîne. De fait, la grande majorité des radios logicielles utilisées dans l'IoT le sont comme passerelles où les contraintes énergétiques sont bien moins marquées [4].

Les récentes avancées en architecture basse consommation redistribue les cartes de la viabilité des radios logicielles basse consommation du fait de l'émergence de systèmes où la capacité de traitement n'est pas sacrifiée au profit de la consommation énergétique. Des premiers travaux préliminaires ont montré leur potentielle viabilité. C'est le cas de *TinySDR* [3] qui intègre un FPGA pour réaliser des transmissions LoRa ou Marmote SDR qui se focalise sur l'étude des protocoles [7]. La radio présentée dans [1] permet de réaliser des transmissions satellite et s'appuie également sur un FPGA pour le traitement. C'est ici que l'on peut pointer les limites des architectures proposées : le traitement est réalisé par un FPGA ce qui augmente le coût et complexifie le prototypage. A contrario, les architectures très récentes qui s'appuient sur des microcontrôleurs plus flexibles n'offrent pas de capacité de traitement suffisante [8].

C'est pour ces raisons que l'on présente dans cet article la radio logicielle LOLA pour radio logicielle reconfigurable faible puissance, faible latence (*Low pOwer Low lAtency reconfigurable*). Cette radio se base sur des composants sur étagère dont la combinaison sur une carte dédiée permet une bonne capacité de traitement avec une consommation réduite. L'accent ici est fait sur la capacité de reconfiguration avec un traitement réalisé

en grande partie par le microcontrôleur. Nous avons ajouté une dalle FPGA pour permettre de déporter certains calculs. Pour montrer les capacités de notre architecture, nous y avons porté une couche physique de notre cru dédiée à de la transmission audio sans fil haut débit, dont le cahier des charges nécessite une latence très faible. Notre système est donc capable de supporter le flux audio en temps réel avec une latence inférieure à la milliseconde et une consommation bien plus faible que les architectures basées sur des systèmes sur puces.

L'article est scindé en quatre parties : Dans la partie 2, nous introduisons l'architecture de la radio logicielle LOLA en caractérisant ses éléments architecturaux. Nous justifions le choix des composants clefs et en quoi ils permettent d'obtenir un excellent compromis entre la flexibilité, la consommation et la capacité de traitement. Dans la section 3 nous proposons une couche physique dédiée à la transmission sans fil haut débit faible latence. La partie 4 montre la plateforme ainsi créée et quelques clefs de compréhension sur les performances de traitement obtenues. La dernière partie conclue cet article.

2 Architecture proposée

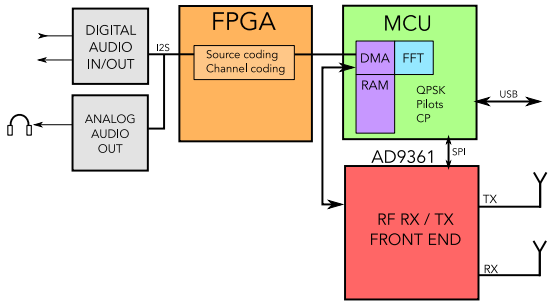

FIGURE 1 – Synoptique de l'architecture proposée

L'architecture de notre radio logicielle est présentée sur la Figure 1 et est composée de plusieurs modules :

- Le terminal radio fréquence (RF) est représenté en rouge et est utilisé pour la transmission et la réception des signaux électromagnétiques. Les étages sont composés de convertisseurs analogiques numériques large bande et d'un oscillateur pour la modulation/démodulation capable d'opérer sur une large gamme de fréquence. Le choix de la carte RF dépend de l'application considérée : une carte RF largement reconfigurable offre d'excellentes capacités de reconfiguration au détriment d'une augmentation du coût et de la consommation énergétique et d'une diminution de la figure de bruit. Le choix d'une carte RF spécialisée permet comme son nom l'indique d'adapter la carte à des scénarios qui ne nécessitent pas une exploration large en bande de fréquence et bande de travail.
- Le microcontrôleur (μC), en vert, est l'unité de traitement principale de l'architecture. L'utilisation d'un μC offre plusieurs avantages : la consommation énergétique est très inférieure à celle des processeurs génériques et les nombreuses interfaces permettent de relier l'unité de traitement avec des périphériques ou des coprocesseurs. Enfin, leur fréquence de fonctionnement est adaptée à la cadence de traitement permettant de traiter en temps réel une bande de l'ordre de quelques MHz.
- Le coprocesseur spécifique est représenté en bleu. Certains μC intègrent des coprocesseurs matériels tels que des systèmes de cryptographie ou des outils de transformée de Fourier rapide (TFR). L'utilisation d'un μC offre deux avantages majeurs : les opérations sont effectuées par un coprocesseur spécialisé de manière efficace et le μC peut effectuer d'autres tâches pendant que le coprocesseur réalise son traitement.
- La RAM est en violet et est placée dans le μC. La mémoire à accès direct (DMA) utilisée pour permettre de simplifier l'acquisition des signaux par la radio qui seront traités par le μC et par le coprocesseur.
- Un petit FPGA, en orange, est utilisé pour deux objectifs : permettre la mise en place d'une colle logique entre les différents composants et interfaces, et plus particulièrement les sorties audios ; et déporter quelques tâches de traitement en matériel, en fonction de la taille du FPGA.

L'architecture est proposée dans le but d'offrir un compromis très intéressant entre la simplicité et la basse consommation mais une capacité de traitement conséquente. Nous détaillons dans la partie qui suit une couche physique dédiée à l'audio haute qualité qui permet un traitement temps réel des 2.3MBits/s du flux audio.

3 Chaine de traitement

3.1 Description des paramètres et de l'émetteur

Nous proposons une chaîne à base de multiplexage fréquentiel orthogonal (OFDM) avec modulation à changement de phase à 4 état (QPSK), qui permet une égalisation simple dans le domaine fréquentiel. La chaîne s'appuie donc sur plusieurs paramètres (cf Table 1) que l'on peut scinder en deux groupes : les paramètres définis (soulignés) et les paramètres calculés. Les paramètres définis correspondent à ce que nous impose la norme audio ou certains choix arbitraires. Le détail est donné dans [2].

TABLE 1 – Paramètres définis, où SC correspond à Symbole Complexe

Définition	Valeur
Fréquence d'échantillonnage audio	96kHz
Nombre de bits par mot audio	24 bits
Débit binaire	2.304 MBits/s
Nombre de mots audio par paquet	10
Paramètrage du Hamming	(4,7) bits
Taille de la TFR	512 SC
Taille du préfixe cyclique	36 SC
Débit de sortie	5.2608 MS/s

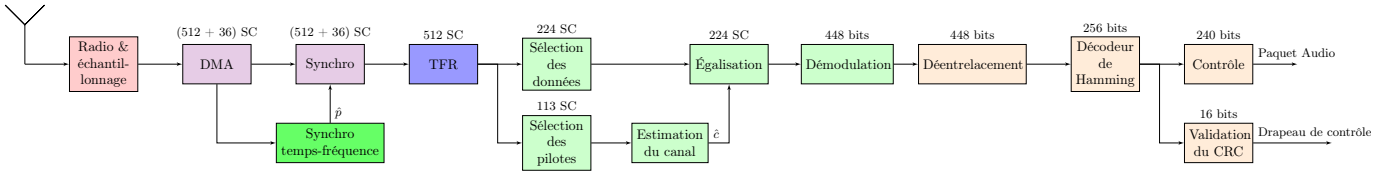

FIGURE 2 – Chaine de réception proposée.

3.2 Étage de réception

La réception se fait en deux étapes i) l'étape de synchronisation et ii) l'étape de décodage.

La première étape vise à positionner finement dans la grille temps fréquence le signal reçu. Ceci se fait par l'intermédiaire d'une corrélation entre le signal reçu et une séquence pilote. Cette séquence est définie comme l'inverse de la transformée de Fourier des pilotes positionnés en fréquence avec des zéros pour les données. La position du maximum donne le début du symbole OFDM et l'argument permet de calculer le décalage de fréquence qui est retro-compensé au niveau de l'oscillateur de la radio.

Le synoptique du récepteur est présenté sur la Figure 2. Chaque couleur correspond à un domaine de traitement différent et suit le même code couleur que la Figure 1. En violet les opérations de la DMA qui aligne les signaux à partir des données de synchronisation. Comme ce positionnement peut bouger au cours du temps, le second coeur du μC (en vert foncé) est dédié au suivi de l'erreur de synchronisation qui s'obtient par calcul de la corrélation au voisinage de l'instant de synchronisation. Si un décalage est détecté, la rétro-action se fait directement au niveau du DMA. Le passage temps-fréquence est matérialisé par le coprocesseur TFR, en bleu. Le gros du traitement est fait par le μC : nous séparons les données des pilotes et estimons le canal directement à partir de la position des pilotes. Les données sont enfin égalisées et une décision dure est faite, ce qui conduit à un bloc binaire de 448 bits. Ces données sont ensuite déentrelacées dans le FPGA puisque c'est un traitement aisé à porter sur architecture matérielle. Le décodage canal est également porté sur le FPGA avec une approche parallèle [6]. L'ensemble des traitements matériels est présenté en orange. La dernière étape est le rendu audio vers les sorties jack.

4 Performances de la carte

4.1 Discussion autour de la création de la carte

Nous avons donc synthétisé et créé une carte fonctionnelle qui s'appuie sur les clefs architecturales présentées dans la partie 2. Pour cette carte, nous avons choisi une carte RF large bande, l'AD9361 d'Analog Devices car c'est une carte qui est largement répandue et qui permet de synthétiser des bandes larges (jusqu'à 61MHz) et fonctionne sur une très large gamme de fréquences (jusqu'à 6 GHz). Ceci nous permet d'être très flexible et de pouvoir configurer l'intégralité des étages numériques de la carte RF (contrôleur automatique de gain, coefficients des filtres de sélection ...). Nous avons intégré un FPGA faible consommation de petite taille, un Lattice MachXO2. Pour le μC nous avons opté pour un LPC55S69 de NXP : c'est un μC dual core ARM M33 qui dispose d'un coprocesseur de TFR.

TABLE 2 – Sommaire des ressources FPGA utilisées

	Émetteur		Récepteur	
	Ressources	%	Ressources	%
Registres	488/2352	21%	632/2352	27%
Slices	413/1056	39%	513/1056	49%
Tables	817/2112	39%	1011 / 2112	48%

Les différents blocs ont donc été intégrés dans leur domaines respectifs. Nous présentons sur la table 4.1 un sommaire des ressources utilisées sur le FPGA en termes de nombre de registres, nombres de slices et nombre de tables de correspondances. Nous montrons ainsi que seulement la moitié du FPGA a été utilisée et que plus de ressources sont nécessaires pour la réception que la transmission.

4.2 Mesures dans un environnement contrôlé

Nous validons les choix techniques et les fonctionnalités de la carte dans une chambre anéchoique. Ceci permet d'avoir une calibration des mesures et de contrôler finement les performances. Nous plaçons l'émetteur et le récepteur à une distance de 3 mètres 80. L'émetteur sans fil est amplifié par un amplificateur de puissance de 26 dBm auquel on adjoint un atténuateur variable pour simuler une perte d'espace libre. Nous avons également positionné une plaque de métal pour émuler une propagation multi-trajet et générer une interférence constructive et un évanouissement important. Les performances sont présentées sur la Figure 3. Pour les trois scénarios (ligne de vue, et les deux configurations avec la plaque de métal), nous évaluons le taux d'erreur paquet en fonction du niveau d'atténuation. Nous observons alors le bon niveau de performance dans les différents scénarios. Par ailleurs, un point important est la décroissance observée de -25dB d'atténuation pour la ligne de vue (et respectivement -27dB et -23dB pour les scénarios avec la plaque) : on remarque que celle ci est progressive ce qui démontre la bonne capacité de synchronisation de notre dispositif même lorsque le niveau de bruit est important.

4.3 Consommation énergétique et latence

Nous mesurons la latence de bout qui vaut 854 μs. Celle ci est partagée équitablement entre l'émetteur et le récepteur.

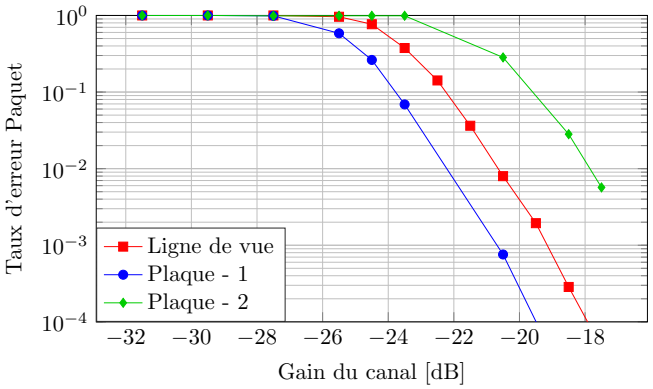

FIGURE 3 – Taux d'Erreur Paquet en fonction du niveau d'atténuation

TABLE 3 – Consommation énergétique du récepteur

Partie du récepteur	Courant	Tension	Puissance
FPGA	23 mA	3.3 V	76.9 mW
Micro-controleur	35 mA	1.8V	63 mW
Récepteur Radio (AD9361)	165 mA	1.8V	297 mW

TABLE 4 – Comparaison entre LOLA et un USRP e310

	Carte RF	Processeur	Conso
LOLA SDR	AD-9361	LPC55S69 @ 150 MHz	437 mW
USRP E310	AD-9361	Zynq 7010 @ 866 MHz	3.25 W

Les sources de latence sont fonction de la durée d'un mot audio (10.4 μs) et de la durée d'un paquet de 10 mots audios (104.1 μs).

On décrit dans la table 3 la consommation énergétique (en tension, courant et puissance) de la chaîne de réception puisque la majorité des étapes de traitements sont réalisées du côté récepteur. La consommation totale du récepteur est de 437 mW, principalement due au récepteur RF en lui même. L'utilisation du μC permet justement de faire chuter le budget de consommation de manière conséquente. La consommation du FPGA est principalement due au choix du Lattice qui requiert un niveau de tension élevé mais une demande en courant faible.

4.4 Comparaison avec un USRP e310

Nous avons implémenté la même chaine de réception sur une radio logicielle basée sur une architecture classique Zynq 7020 et nous comparons la consommation énergétique dans la table 4. La consommation énergétique est grandement réduite (d'un facteur 7) ce qui prouve l'intérêt de notre choix d'architecture.

5 Conclusion

Dans cet article nous avons présenté une nouvelle architecture de radio logicielle faible consommation, faible latence et flexible. Cette architecture s'appuie sur une nouvelle génération de microcontrôleurs faible consommation avec de bonnes capacités de calcul notamment via coprocesseurs. Nous avons intégré sur une même carte ce microcontrôleur ainsi qu'un émetteur récepteur radio-fréquence et un FPGA faible consomation.

Nous avons montré l'intérêt de ce type de plateforme en proposant une couche physique dédiée à de l'audio sans fil haute qualité. Notre plateforme est capable de traiter en temps réel avec une latence inférieure à la milliseconde est une consommation energétique inférieure à 500 mW.

Une radio logicielle n'est pas nécessairement un objet purement générique et arbitrairement reconfigurable : c'est avant tout une plateforme flexible, capable de se spécialiser pour une application. LOLA prouve que l'on peut allier radio logicielle flexible, faible consommation et latence maitrisée.

6 Remerciements

Ce projet de recherche a été soutenu par la région Bretagne et Lannion Trégor Communauté (APP PME HAD-OC labellisé par le Pôle Image et Réseaux)).

Références

[1] Xin Cai, Mingda Zhou *& al.* Low-power SDR design on an FPGA for intersatellite communications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 26(11) :2419–2430, 2018.

[2] Olivier Etrillard, Robin Gerzaguet *& al.* LOLA SDR : Low power low latency software defined radio for broadcast audio applications. *IEEE Transactions on Circuits and Systems II : Express Briefs*, pages 1–1, 2022.

[3] Mehrdad Hessar, Ali Najafi *& al.* TinySDR : Low-Power SDR Platform for Over-the-Air Programmable IoT Testbeds. In *Proc. USENIX Symposium on Networked Systems Design and Implementation*, pages 1031–1046, 2020.

[4] Yong Hua Lin, Qing Wang *& al.* Wireless IoT Platform Based on SDR Technology. In *Proc. IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 2245–2246, 2013.

[5] J. Mitola. Software radios : Survey, critical evaluation and future directions. *IEEE Aerospace and Electronic Systems Magazine*, 8(4) :25–36, 1993.

[6] M. Sprachmann. Automatic generation of parallel CRC circuits. *IEEE Design Test of Computers*, 18(3) :108–114, 2001.

[7] Sándor Szilvási, Benjámin Babják *& al.* Marmote SDR : Experimental platform for low-power wireless protocol stack research. *Journal of Sensor and Actuator Networks*, pages 631–652, 2013.

[8] Mathieu Xhonneux, Jérôme Louveaux *& al.* Implementing a LoRa Software-Defined Radio on a General-Purpose ULP Microcontroller. In *Proc. IEEE Workshop on Signal Processing Systems (SiPS)*, pages 105–110, 2021.