Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

GÉNIE ÉLECTRIQUE
ELEKTROTECHNIK

Rapport de Travail de Bachelor

Automne 2024

RF sniffer using USRP

Auteur

Bastien Piguet
bastien.piguet@edu.hefr.ch

Supervision

Daniel Oberson
daniel.oberson@hefr.ch
Philippe Joye
philippe.joye@hefr.ch

Partenaire

Sonova
Loïc Murith
Christophe Ostermann

Date de remise

Fribourg, le 12.07.2024

Remerciements

Un grand remerciement aux personnes suivantes sans qui le projet n'aurait pas pu être réalisé.

Monsieur Christophe Schaer de la fillière Informatique et systèmes de communication qui a mis à disposition le matériel USRP avec son ensemble de développement ainsi que pour son temps et son aide précieuse pour la résolution de problèmes lié au système.

Messieurs Loïc Murith et Christophe Ostermann de Sonova pour leurs expertises tout au long du projet, leurs disponibilités et leurs réponses claires et précises lors de questionnement technique sur le domaine.

Messieurs Daniel Oberson et Phillipe Joye pour le suivi du projet, l'aide à la résolution de problèmes et leurs conseils pour les méthodes de développement et la rédaction de ce rapport.

Table des matières

Liste des figures

Liste des tableaux

Liste des codes

Abréviations

Abréviation	Définition
HEIA-FR	Haute école d'ingénierie et d'architecture de Fribourg
GFSK	Gaussian Frequency Shift Key
CSV	Comma Separreted Value
USRP	User Software Radio Peripheral
CRC	Cyclic Redundancy Check
DM	Digital Modulation
FPGA	Field programmable gate array
MSB	Most Significant Bit
LSB	Less Significant Bit
NI	National Instrument
HDL	Hardware design logic
I	In-phase signal
Q	Quadrature signal
RF	Radio Frequency
DMA	Direct Memormy Access
PCIe	Peripheral Component Interconnect express
FIFO	First In First Out
LO	Local Oscillator
DAC	Digital to Analogue Converter
ADC	Analogue to Digital Converter
IP	Intellectual Property
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
bps	Bit Par Seconde

Symboles

Symbole	Définition	Unité
i	Courant	A
v	Tension	V
f	Fréquence	Hz
T	Temps	s

1. Introduction

L'entreprise Sonova développe des appareils auditif dans de multiple domaine, dans leurs bureau ils prennent à cœur de développer tous les système de A à Z. Pour assurer l'avancement le plus rapide et optimale de ce processus ils ont en place des bancs de test qui mettent à épreuve les nouveaux systèmes mis en place. Des mesures de tension, de communication et autres sont effectués afin de confirmer le fonctionnement correcte, avec l'ensemble du système, des nouvelles capacités produite.

Cependant dans la ligne de qualification des systèmes il n'existe pas de produit commercial qui fait la journalisation des communication sans fils pour les protocole utilisé, tel que le Saleae utilisé pour les signaux électriques analogique et numérique. Pour cause, en dehors du bluetooth et bluetooth low energy, Sonova dispose de deux protocole propriétaire.

1.1. Le projet

Ce projet consiste donc au développement d'un système capable de démodulé simultanément les canaux de communications d'un protocole tel que le bluetooth, en extraire les activités et de décodé les informations reçu pour les traiter ultérieurement lors de déboguage des systèmes.

1.2. Objectifs

D'ici le 12 juillet 2024, le but de ce projet est donc de développer un sniffer des canaux des fréquences ISM 2.4 GHz afin, dans un premier temps de pouvoir déterminer en temps réel le chronogramme d'activités dans chacun des canaux. Puis, selon l'avancement du projet de préparer la synchronisation, la démodulation et le décodage d'un ou plusieurs des 4 protocoles utilisés. Dans la mesure où un ou des décodages ont été implémenté, une validation ainsi qu'une caractérisations des limites du sniffer devront être effectués.

- Déterminer les spécifications techniques selon les besoins de Sonova.
- Mettre en œuvre les USRP pour préparation à l'application visée.
- Déterminer une architecture software (Labview, C/C++ ou autre) pour développement du sniffer.
- Développer et valider le sniffer des canaux et en ressortir leurs utilisations en temps réel.
- Développement de la synchronisation et du décodage d'un protocole.
- Documenter adéquatement la solution proposée.

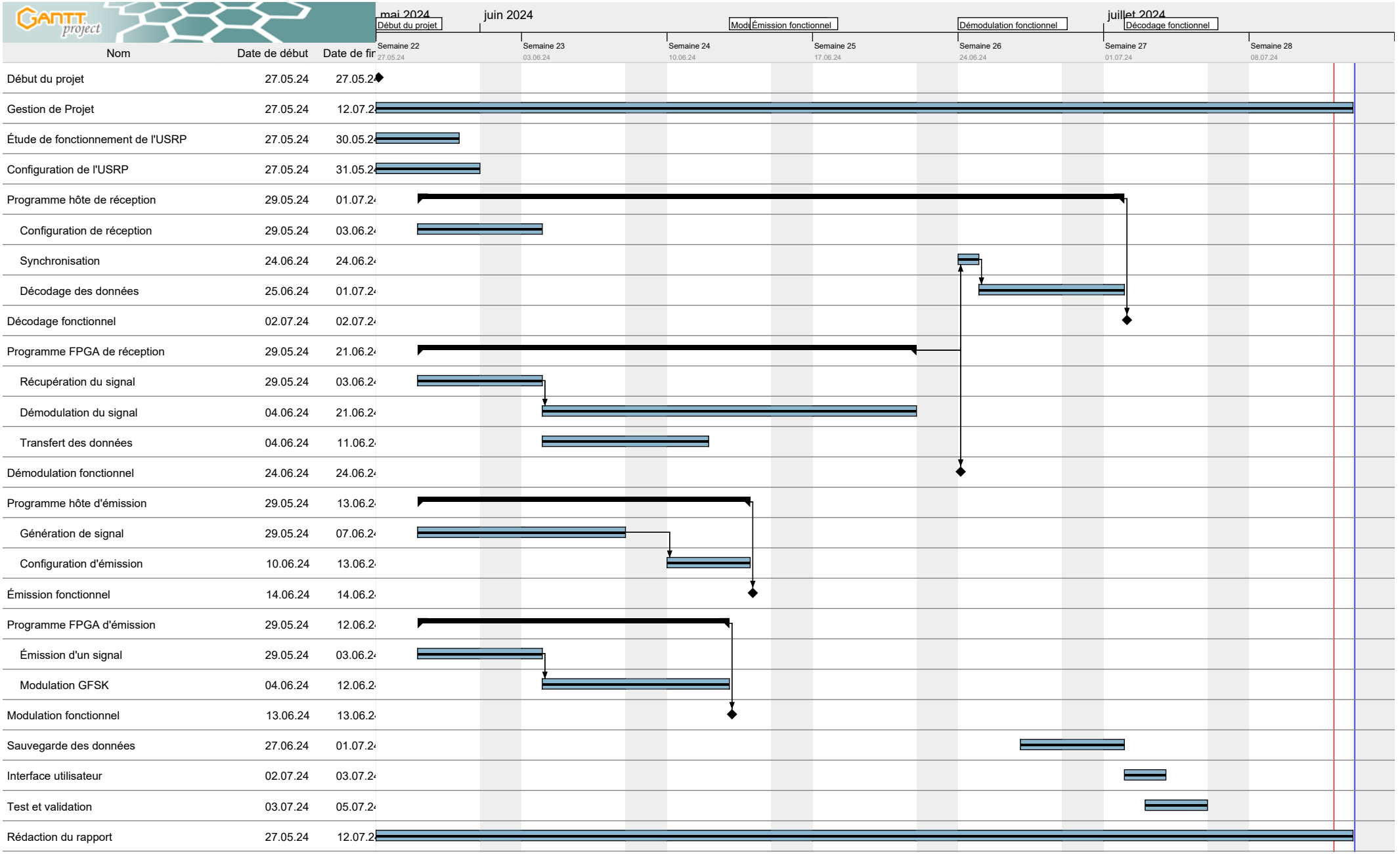
1.3. Sur le travail de Bachelor

Ce travail est réalisé dans le cadre du projet de Bachelor (TB) à la HEIA-FR. Le but du travail de Bachelor est d'évaluer la capacité de l'étudiant à effectuer un travail selon un cahier des charges donné. Le temps disponible est égal à cinq jours de travail par semaine pendant 7 semaines. Le projet est évalué par une table d'évaluation transmise par le corps professionnel et est équivalent à 12 crédits ECTS.

1.4. Travaux antérieurs

Ce projet est la suite du travail du même nom [4], effectuer lors du projet de semestre 6 2024. Celui-ci faisait l'étude du matériel et du logiciel à utilisé ainsi que de la faisabilité d'un tel système. Il en concluat que l'USRP-2953R, mis à disposition par l'école, est un bon candidat pour une première étude malgré ses limitations technique. Le logiciel LabVIEW fu sélectionné comme interface de développement étant étroitement lié avec son hardware USRP.

Diagramme de Gantt



Nom	Date de début	Date de fin
Début du projet	27.05.24	27.05.24
Gestion de Projet	27.05.24	12.07.24
Étude de fonctionnement de l'USRP	27.05.24	30.05.24
Configuration de l'USRP	27.05.24	31.05.24
Programme hôte de réception	29.05.24	01.07.24
Configuration de réception	29.05.24	03.06.24
Synchronisation	24.06.24	24.06.24
Décodage des données	25.06.24	01.07.24
Décodage fonctionnel	02.07.24	02.07.24
Programme FPGA de réception	29.05.24	21.06.24
Récupération du signal	29.05.24	03.06.24
Démodulation du signal	04.06.24	21.06.24
Transfert des données	04.06.24	11.06.24
Démodulation fonctionnel	24.06.24	24.06.24
Programme hôte d'émission	29.05.24	13.06.24
Génération de signal	29.05.24	07.06.24
Configuration d'émission	10.06.24	13.06.24
Émission fonctionnel	14.06.24	14.06.24
Programme FPGA d'émission	29.05.24	12.06.24
Émission d'un signal	29.05.24	03.06.24
Modulation GFSK	04.06.24	12.06.24
Modulation fonctionnel	13.06.24	13.06.24
Sauvegarde des données	27.06.24	01.07.24
Interface utilisateur	02.07.24	03.07.24
Test et validation	03.07.24	05.07.24
Rédaction du rapport	27.05.24	12.07.24

2. Spécifications

2.1. Limites du mandant et leurs appareils

Ce projet ayant pour but d'être utilisé avec les produits de Sonova cela implique donc qu'il doit se référer aux spécifications de ceux-ci. Les informations fournies pas Sonova indique que la puissance minimale de réception de l'appareil doit atteindre -80 dBm et la puissance maximale d'émission 6 dBm.

Les éléments tels que la fréquence d'échantillonnage du signal, la précision des mesures et autres sont à définir selon le matériel et le but est d'avoir la plus grande quantité d'informations possible.

Les paquets démodulés et décodés seront analysé avec le programme Wireshark qui utilise le format *libcap* comme fichier. Ce format étant très complet dans sa structure son implémentation requiert beaucoup de temps. Comme alternative les données seront sauvegardées dans un fichier CSV structuré ainsi.

time	Channel	Preamble	Address	Length	PID	NO_ACK	Payload	CRC
UTF time	*1-10*	*AA|55*	*5 bytes*	*6 bits*	*2 bits*	*1 bit*	*1-32 bytes*	*2 bytes*

TABLE 1 – Structure de sauvegarde des paquets

L'activité dans les différents canaux est une information qui est aussi demandée et est donc à sauvegarder de la même manière. La structure du fichier se présente ainsi.

time	CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8	CH9	CH10
UTF time	*dBm power*	...	...	...	...	...	...	...	...	...

TABLE 2 – Structure de sauvegarde de l'activité

Dans le cas actuel seulement les canaux 1 à 10 sont étudiées, ceci est du aux limitations physiques de l'USRP. Afin de rendre le système capable de démoduler l'ensemble des canaux, veuillez vous référer à la section 7.2 de la conclusion.

2.2. Protocole

Le protocole étudié est porté sur le DM (digital modulation) tel expliqué dans le travail précédent et disponible en annexe .

Le paquet transmit se compose ainsi, Un préambule de 8 bits de synchronisation comportant des bits alternant entre 1 et 0, 3 ou 5 octets d'adresses, 9 bits d'informations sur le paquets, 1 à 32 octet de payload et 2 octets de contrôle.

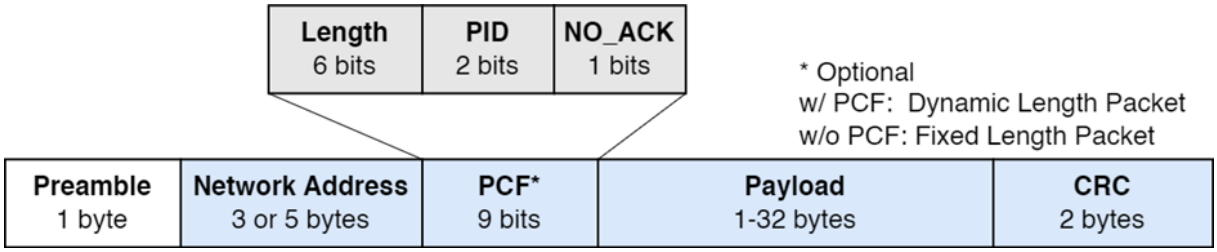


FIGURE 1 – Structure d'un paquet DM.

L'adresse peut se structurer de différente manière selon les besoins de l'utilisateur. Ci-dessous l'ensemble des possibilités que fourni le protocole.

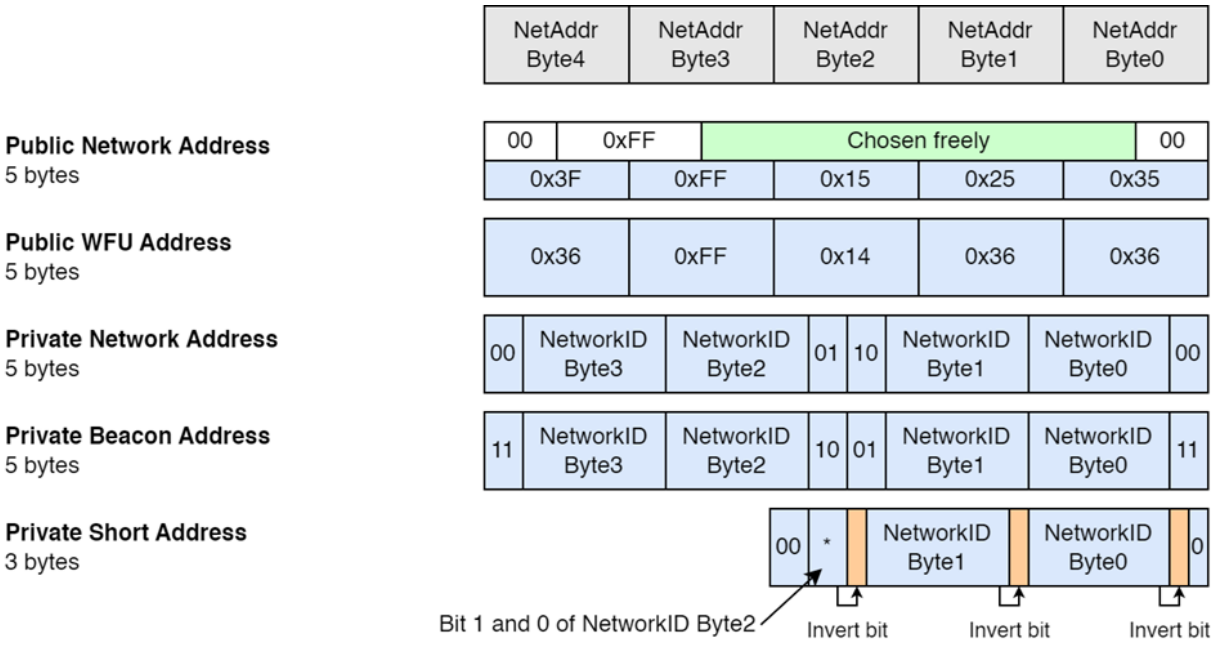


FIGURE 2 – Structure des données d'adresse.

2.3. USRP

Les spécifications de l'USRP ont été définis dans le travail précédent et sont accessibles en annexe section B.3.

2.3.1. Taille de la FPGA

La taille de la FPGA est limité, pour cela tout le long du développement un œil doit être gardé sur la proportion utilisée afin de définir les limites qu'atteindra le projet pour la parallélisation de démodulation des canaux.

2.3.2. Référence de l'ADC

Les documents de NI sur l'USRP ne précise pas la référence que prennent les convertisseurs pour le signal reçu et émis. Pour cela le programme exemple utilisé les informations suivantes ont été définies. La figure (3) transforme le gain du signal en tension, ce qui correspond à l'équation 1. Grâce aux commentaires disséminés dans le projet, il a été possible de déterminer et d'extrapoler cette équation pour obtenir la formule (3) qui indique la référence de 1 *dBm* d'entrée avec une impédance de 50 Ω

$$v_{pp} = 2 \cdot \sqrt{\frac{10^{\frac{gain}{10}}}{10}} \tag{1}$$

$$gain = 10 \cdot \log\left(\left(\frac{v_{pp}}{2}\right)^2 \cdot 10\right) \tag{2}$$

$$gain = 10 \cdot \log\left(\frac{\frac{v_{rms}^2}{50\,\Omega}}{1\,mW}\right) \tag{3}$$

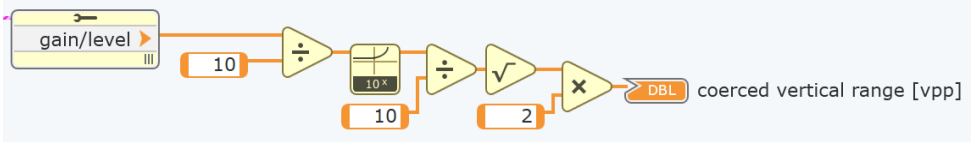


FIGURE 3 – Transformation du gain en tension.

3. Méthodes de démodulation

La démodulation d'un signal GFSK peut se faire de plusieurs méthodes, cette section en fait la liste, l'explication de leur fonctionnement. Ces informations sont tirées des documents *Low-Latency GFSK Demodulation Architecture Comparison and Design for FPGA* [1], *Frequency shift keying demodulators for low-power FPGA applications* [2], *A mixed-signal GFSK demodulator for Bluetooth* [3] et de travaux antérieur sur l'USRP [4].

3.1. Counter

La première méthode utilise un compteur pour mesurer le temps entre les flancs du signal d'entrée, f_{in} . Si la période des fréquences FSK peut être comptée, mesurée et distinguée, les données dans f_{in} peuvent être démodulées. Le nombre de cycles d'horloge entre les flans de f_{in} mesure la période, T_{in} . L'horloge utilisée pour le comptage est l'horloge d'entrée du module démodulateur FPGA. Le signal de sortie du compteur peut être représenté à l'aide de l'équation 4. Lorsque l'entrée varie entre f_H et f_L, la valeur de N_{count} produit une onde pseudo-carrée qui imite les valeurs des données modulées. La valeur de N_{count} est inversement proportionnelle à la fréquence d'entrée. La sortie du compteur devra être un signal numérique réel, donc la sortie sera créée en comparant la forme d'onde de N_{count} à une valeur seuil. Lorsque N_{count} est supérieur ou inférieur au seuil, la sortie sera démodulée en 1 ou 0 respectivement.

$$N_{count} = \frac{f_{clk}}{f_{in}} \tag{4}$$

La précision du démodulateur est strictement liée à la fréquence de travail de la FPGA. Un plus grand nombre de cycles d'horloge permet d'obtenir des valeurs de comptage plus élevées et une plus grande différence de N_{count} pour f_L et f_H . La valeur de N_{count} peut également être augmentée en accumulant le nombre sur plusieurs cycles de f_{in} . N_{count} est accumulé sur plusieurs périodes pour réduire les effets de la distorsion de 1 bit. L'accumulation nécessite des registres supplémentaires dans la conception HDL, et la sortie serait retardée, mais celle-ci aurait des transitions de données plus cohérentes et clairement définies. L'utilisation d'un comptage accumulé sur plusieurs cycles permet également d'augmenter la différence entre les valeurs pour chaque fréquence FSK.

3.2. One-shot

Le démodulateur One-Shot détecte les flancs d'un signal d'entrée FSK. Le nombre de flancs sur une période donnée est proportionnel à la fréquence du signal. Sur une période donnée, le nombre de fronts peut être utilisé pour détecter si l'entrée f_{in} est f_H ou f_L et ainsi sur une période de temps peut être compté et le résultat considéré comme une valeur pseudo-analogique proportionnelle à la fréquence d'entrée.

Lorsqu'un flanc est détecté sur l'entrée, le démodulateur produit une impulsion. La longueur de l'impulsion est conçue pour correspondre à une période de f_H . Lorsque l'entrée est à f_H, les impulsions de sortie se succèdent et produisent une valeur constante jusqu'à ce que l'entrée passe à f_L . Lorsque l'entrée est f_L, les fronts se produisent moins souvent. Les impulsions produites restent de la même longueur et se produisent à chaque front positif, comme auparavant. Comme il y a moins de fronts et que ceux-ci sont plus étalés, la valeur moyenne de la sortie quand l'entrée est f_L sera plus faible.

La valeur moyenne des impulsions de sortie sur un intervalle de temps peut être utilisée pour démoduler l'entrée. Les valeurs moyennes des impulsions de sortie donnent une forme d'onde pseudo-analogique qui représente les données modulées dans l'entrée. La forme d'onde peut être reconvertie en signal binaire en trouvant une valeur seuil appropriée et en la comparant à la sortie One-Shot. Le résultat indique quand le signal doit être démodulé en 1 ou en 0.

3.3. Démodulation par déplacement de fréquence

Les fréquences f_L et f_H étant connu, le signal d'entrée f_{in} est, dans le domaine fréquentiel, déplacé vers la bande de base, qui est de 0 Hz. De ce fait la sortie sera une valeur constante si les fréquences sont identiques ainsi qu'une fréquence de la somme des deux qui sera supprimé à l'aide d'un filtre passe bas, équation (5) . Si les fréquences ne sont pas les mêmes la sortie sera nulle après le filtre. Celui-ci est défini à la fréquence de données, f_L et f_H

$$cos(f_{in}) \cdot \cos(f_{H|L}) = \frac{1}{2}\left(\underbrace{\cancel{\cos(f_{in} + f_{H|L})}}_{\text{filtered}} + \underbrace{\cos(f_{in} - f_{H|L})}_{=1}\right) \tag{5}$$

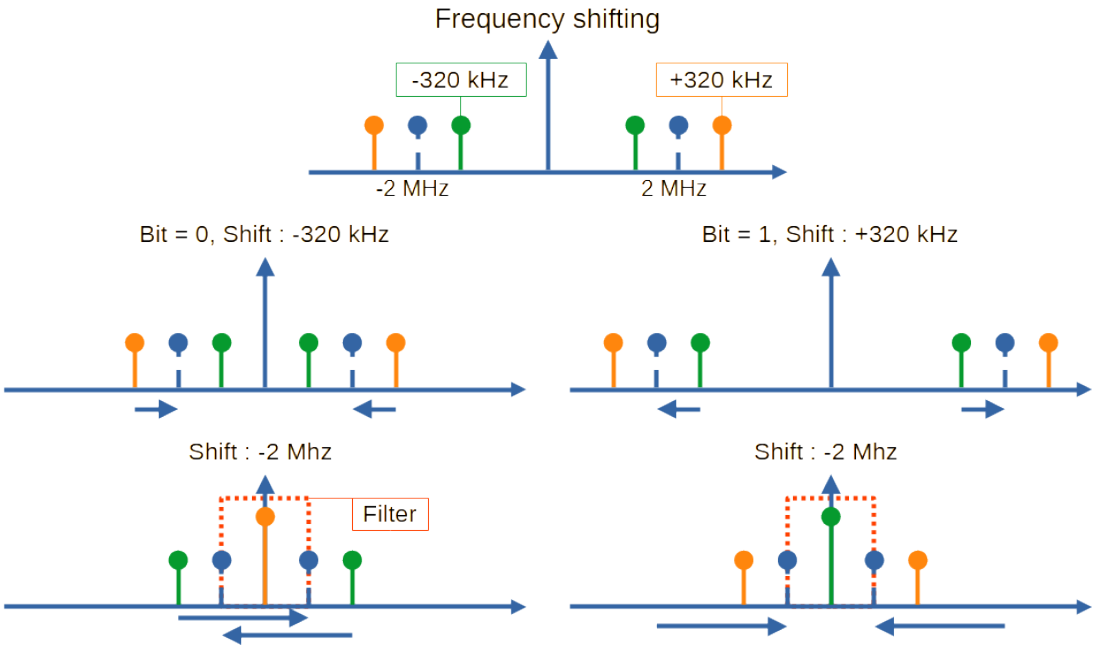


FIGURE 4 – Déplacement de fréquences.

La bitstream est créée en effectuant la comparaison entre les sorties de deux lignes de déplacement de fréquence, une pour f_L et une pour f_H. Si l'amplitude de la ligne de f_L est supérieur à la ligne de f_H ceci indique un bit à 0 et inversement si ligne de f_H est supérieur à la ligne de f_L ceci indique un bit à 1.

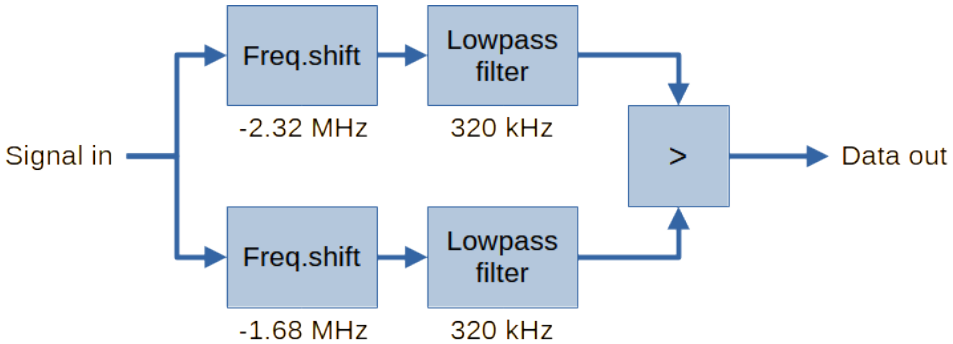


FIGURE 5 – Schéma blocs de démodulation par déplacement de fréquences.

3.3.1. Compensation d'erreur de fréquence

L'appareil émettant les données et l'appareil de réception ne sont pas parfaits et pas synchronisé. De ce fait les fréquences d'horloge ne sont pas identiques et pour la démodulation des données

l'information importante est la fréquence à laquelle le signal se trouve. Une différence minime entre la fréquence d'émission et la fréquence de déplacement en résulte que les données démodulées oscille à une fréquence égale à la différence entre l'émetteur et le récepteur. Afin de résoudre ce problème le signal reçu passe au travers d'un filtre passe-bande qui va isoler les données reçues et ainsi la fréquence à laquelle elles sont modulées. Ces données isolées seront traitées dans le but d'en ressortir la fréquence à laquelle elles sont transmises. Cette fréquence ainsi récupérée et utilisée pour effectuer la démodulation effective des données.

3.4. Quadricorrelator

Le quadricorrelateur est une méthode de démodulation ayant une approche mathématique. Son fonctionnement est basé sur les signaux quadratiques I et Q, équivalent cosinus et sinus. Pour cela le signal RF reçu est multiplié par ces fonctions trigonométriques et en résulte les signaux $s_I(t)$ et $s_Q(t)$.

L'explication du fonctionnement de la génération et de l'extraction des signaux quadratiques ne sera pas abordé dans ce papier, étant un domaine complexe et sortant du domaine d'étude actuel.

Le signal décomposé quadratiquement est défini par les variables suivante

- R_s : Débits de symbole par seconde. [bps]
- h : Facteur de modulation.
- f_c : Fréquence du canal. [Hz]
- f_d : Fréquence d'écart de la modulation. [Hz]
- $\phi(t)$: Données émises du bitstream traité. Valeur entre ± 1.

$$f_d = \frac{hR_s}{2} \tag{6}$$
$$s_I(t) = \cos(2\pi f_c t 2 + \pi f_d \phi t) \tag{7}$$
$$= \cos(2\pi(f_c \pm f_d)t) \tag{8}$$
$$s_Q(t) = \sin(2\pi f_c t 2 + \pi f_d \phi t) \tag{9}$$
$$= \sin(2\pi(f_c \pm f_d)t) \tag{10}$$
$$\tag{11}$$

Lors de la décomposition la fréquence utilisée est celle du canal afin de pouvoir la soustraire et obtenir uniquement en bande de base un signal oscillant entre une fréquence positive et négative représentant les données transmises.

Le cœur de cette méthode est la multiplication croisée des signaux I et Q avec un déphasage dans le temps. La multiplication de deux signaux orthogonaux ayant la même fréquence s'annule, l'introduction d'un déphasage permet d'extraire le signe de cette phase, représentant le sens de rotation entre ces signaux.

$$s_{IdQ}(t) = s_I(t) \cdot s_Q(t - \Delta T) \tag{12}$$
$$= \frac{1}{2}\left(\sin(\pm(2t - \Delta T)) - \sin(\pm(-\Delta T))\right) \tag{13}$$
$$s_{QdI}(t) = s_I(t - \Delta T) \cdot s_Q(t) \tag{14}$$
$$= \frac{1}{2}\left(\sin(\pm(2t - \Delta T)) - \sin(\pm\Delta T)\right) \tag{15}$$

Ces deux signaux sont ensuite soustraits et dans cette opération le sinus ayant le double de la fréquence s'annule et celui constant s'additionne et ressort une valeur variant en -1 et +1 qui

représente donc les bits émient.

$$y(t) = s_{IdQ}(t) - s_{QdI}(t) \tag{16}$$
$$= \sin(\pm\Delta T) = \pm\sin(\Delta T) \tag{17}$$
$$= \phi(t) \tag{18}$$

Afin d'obtenir une amplitude maximale lors de la différence des signaux, il faut qu'ils soient en phase et donc, pour un signal cosinus et sinus, ce retard doit correspondre à un quart de la longueur d'onde. Dans le cas actuel de signaux discret cela représente un retard de N échantillons.

$$\Delta T = \frac{1}{4} \cdot \frac{1}{f_c} \tag{19}$$
$$N = f_{clk} \cdot \Delta T = \frac{f_{clk}}{4 f_d} \tag{20}$$

4. Développement d'une application pour USRP sur LabVIEW

Cette section fait l'explication du fonctionnement et de la marche à suivre pour développer un programme sur l'environnement LabVIEW pour un appareil USRP et le PC auquel il est connecté. Le développement n'est pas exhaustif et contient uniquement les éléments qui ont été utilisés dans le cadre de ce projet.

4.1. Transfert de données dans une FIFO

La première étape dans l'utilisation de l'USRP et la transmission d'information entre l'appareil, qui fait le traitement de signal et le PC hôte qui effectue le traitement des données. Pour ceci des mémoires FIFO sont configurables sur l'USRP, sont mises en commun avec le PC et sont gérées par une DMA qui effectue le transfert au travers du bus de communication établie, qui dans ce cas et le PCIe 4x.

4.1.0.1. Déclaration des mémoires

Les mémoires sont accessibles par l'USRP et sont hôte. Pour cela il faut les déclarer dans des fichiers de ressources qui est en lien avec le programme FPGA. On y défini le type de valeur que la mémoire contiendra, son nombre de bit, sa direction "target to host" ou "host to target" et sa taille qui peut être redéfinie par après dans le programme hôte et qui est aussi appelé sa profondeur.

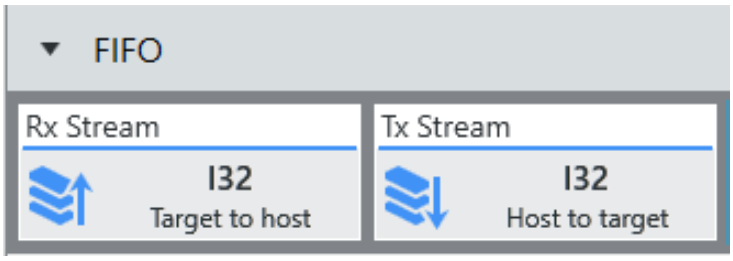


FIGURE 6 – Création des registres.

4.1.0.2. Programme hôte

La configuration de la taille de la mémoire s'effectue dans le programme avec cette fonction. Attention à ne pas modifier cette taille pendant l'utilisation de la mémoire, il faut l'arrêter, la modifier et ensuite la relancer. La taille minimale configurable est de 2048. Afin de modifier ce FIFOs une

variable de type référence contient les informations de la FPGA dont les mémoires, et sur la configuration des fonctions il faut aller chercher quelle mémoire elle doit affecter.

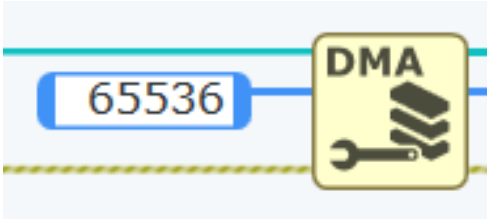


FIGURE 7 – Configuration des FIFOs.

L'écriture se fait en donnant comme paramètre un tableau de valeurs, sa taille ne doit pas dépasser la profondeur de la mémoire. En sorti, le nombre d'emplacements vide est retourné.

La lecture requiert la quantité de valeur voulue ainsi qu'un timeout en milliseconde, qui dans ce cas est à -1 ce qui correspond à une attente infinie.

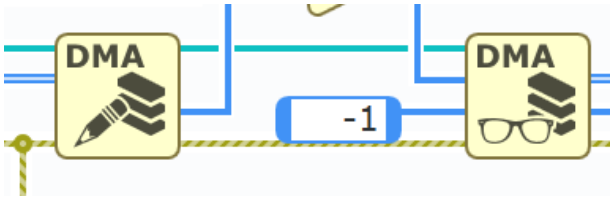


FIGURE 8 – Écriture et lecture de FIFOs.

4.1.0.3. Programme FPGA

Dans une boucle contrôlée par une horloge la mémoire est lue et écrite, chacune de ces actions requiert des signaux de confirmations. La lecture a en entrée "ready for output" et en sortie "output valid" qui informe que la valeur à sa sortie est une nouvelle information. L'écriture à en entré "input valid", qui peut être comparé à un "enable", active le bloc et en sortie "ready for input" qui permet d'indiquer au bloc précédent, avec une boucle de contre-réaction que le bloc est prêt à recevoir une nouvelle donnée.

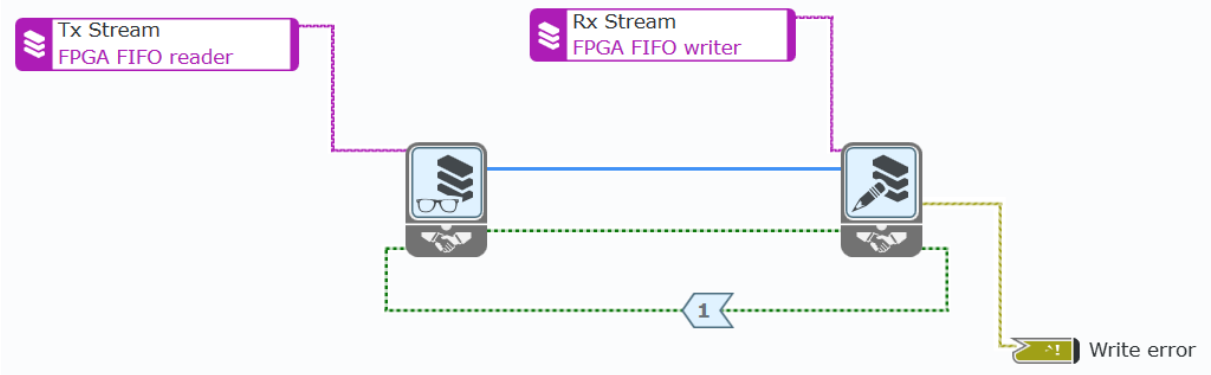


FIGURE 9 – Lecture et écriture de FIFOs.

4.2. Ouverture de la session USRP

Afin de pouvoir configurer les éléments qui ne sont pas reliés à la programmation de la FPGA, il faut ouvrir une session sur l'appareil. Pour cela le bloc "Open", représenté par l'icône d'un nouveau document sur la figure 10, effectue une demande à l'USRP dans le but de récupérer ces informations. Dans la même action la source de l'horloge à utiliser par l'appareil est défini par le bloc de texte "Internal".

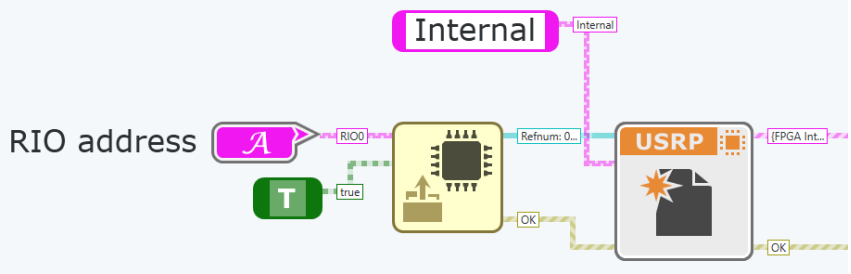


FIGURE 10 – Ouverture de la session.

Afin que cette action réussisse, il est nécessaire que sur la FPGA la gestion des ressources soit mise en place.

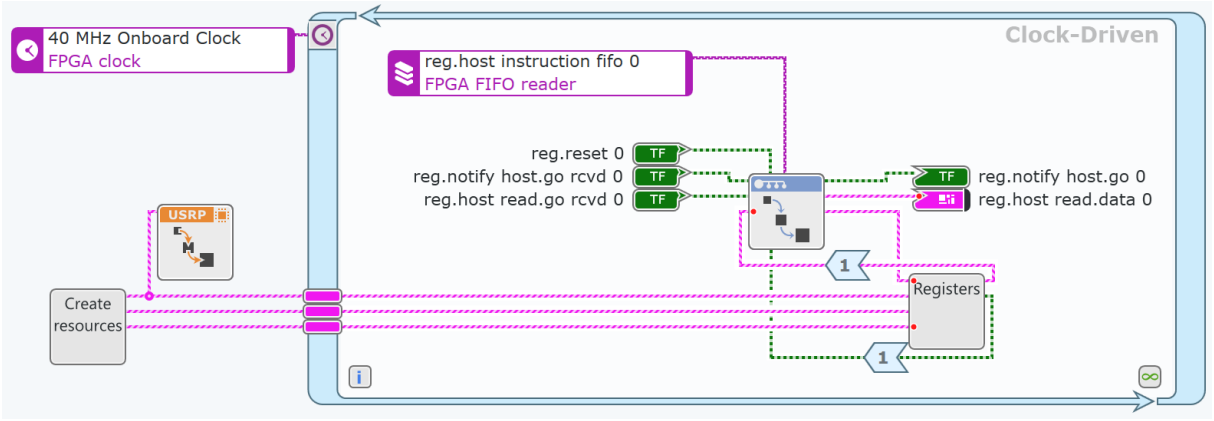


FIGURE 11 – Gestion des ressources FPGA.

La création des ressources se fait dans le bloc "Create resources" qui contient les éléments suivant, visible en figure 12, les ressources de configurations de l'USRP, les ressources de synchronisations, les ressources de temps et 2 ressources d'événements pour les 2 canaux de communication. Puis celles-ci sont inscrites sur les registres de l'USRP dans le bloc "Registers" pour que celles-ci puissent être appliquées.

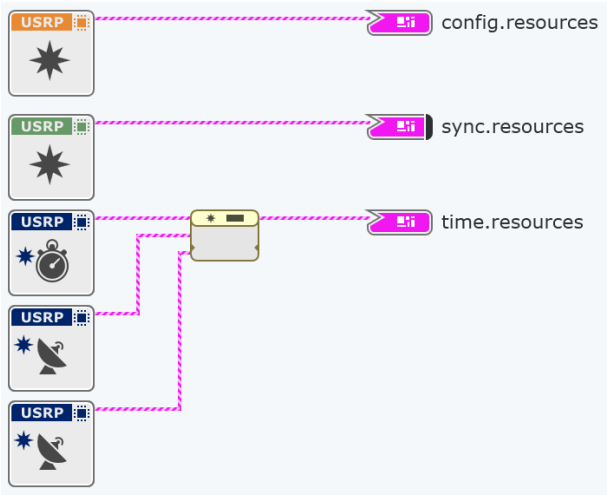


FIGURE 12 – Création des ressources.

L'écriture des valeurs dans les registres se fait de façon périodique grâce à la boucle contrôlé par oscillateur qui à comme source l'oscillateur interne de 40 MHz.

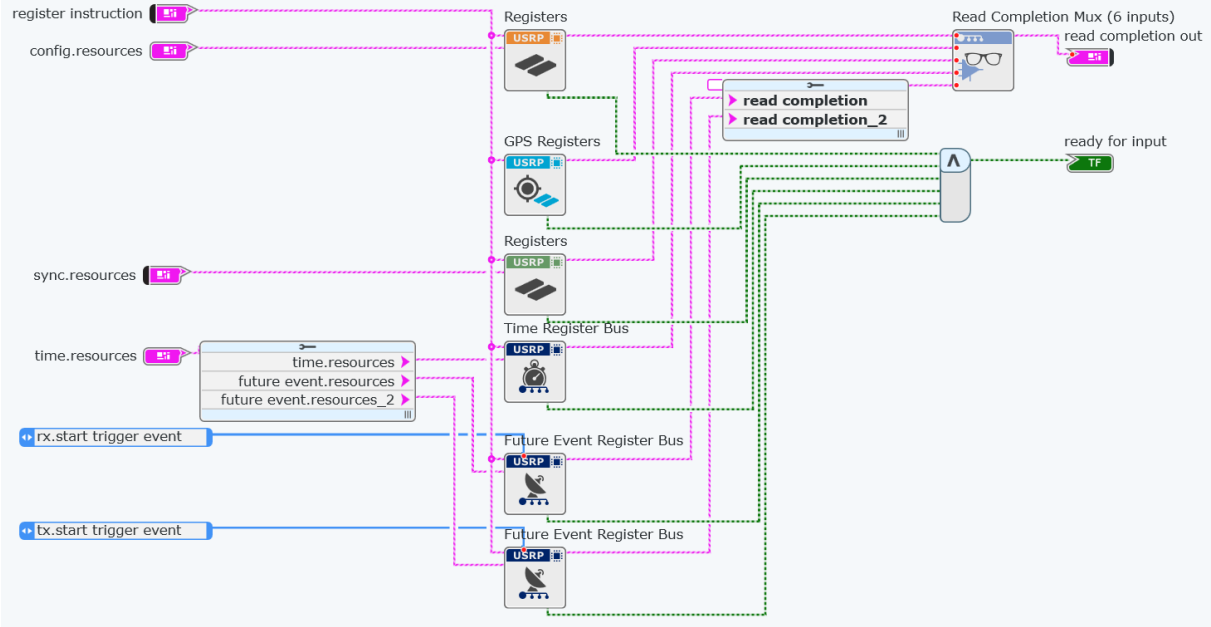


FIGURE 13 – Écriture des registres.

4.3. Émission sur une sortie RF

4.3.1. Configuration

L'USRP peut émettre des signaux sur une grande plage de fréquence et sur 2 canaux physique. Pour cela, il faut configurer les éléments analogiques mis à disposition. Les canaux RF0 et RF1 peuvent émettre sur leurs deux ports, respectivement RX1 et RX2. La fonction "Configure Active Antenna" permet de diriger les signaux vers la sortie voulue. Les paramètres d'entrée sont un string nommant la sortie et la session USRP.

Ensuite la fréquence de porteuse se défini avec la fonction "Configure LO", LO étant l'acronyme de "local oscillator". Comme défini dans les spécifications la fréquence peut être de 1.2 GHz à 6 GHz.

Après l'addition des signaux quadratique se trouve un amplificateur qui est aussi configurable. La fonction "Configure Gain", en lui fournissant l'antenne "tx" ou "rx" et le canal "RF0" ou "RF1", va appliquer le gain défini.

Chacun de ces blocs retourne la valeur appliquée au système. Elle peut varier de la valeur souhaiter dû aux limitations système ou à une correction volontaire de l'imperfection de l'appareil.

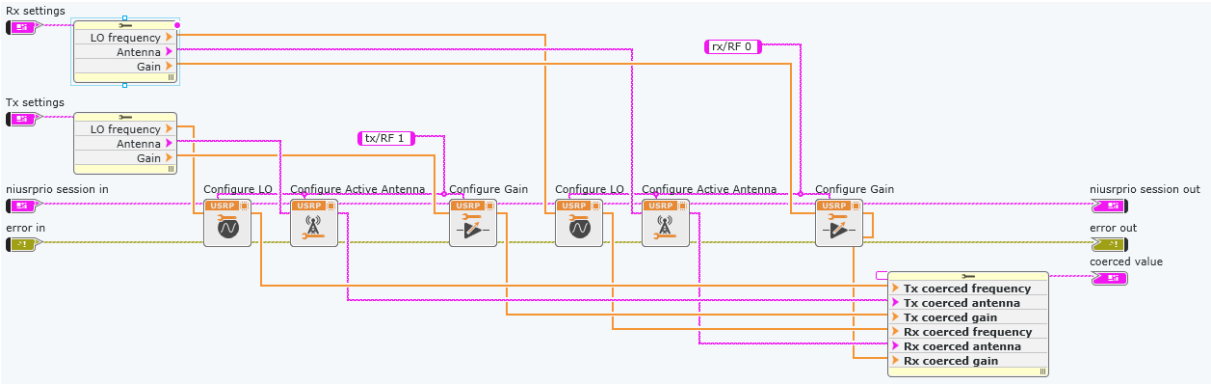


FIGURE 14 – Configuration RF.

4.3.2. Génération d'un signal

Un signal sinus est généré pour être transmis au travers du système RF. L'amplitude est de $\pm 2^{15} - 1$ afin d'utiliser la plage totale du DAC de 16 bits signé.

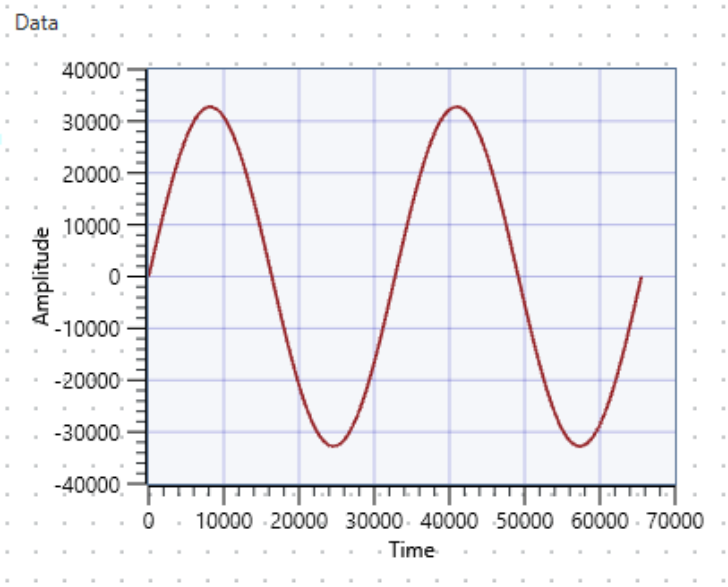


FIGURE 15 – Graphique des données envoyé.

Les données sont transmises au travers d'un FIFO et envoyé sur le canal RF. Le même signal est émis sur les canaux I et Q de l'USRP.

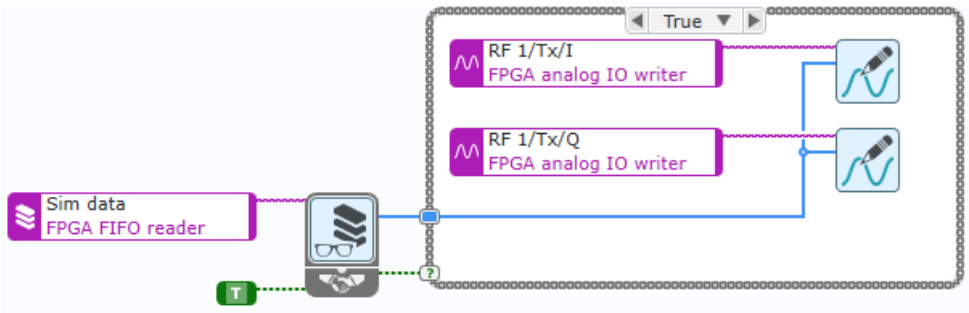


FIGURE 16 – Émission d'un signal reçu par une FIFO.

4.4. Réception sur une entrée RF

Pour la configuration le processus est le même que celui pour l'émission à la section précédente.

Pour limiter le flux d'informations, les données reçues sur le canal RF, sont réunis dans une variable de 32 bits non signée. Puis transmit dans une FIFO pour y accéder sur le PC hôte.

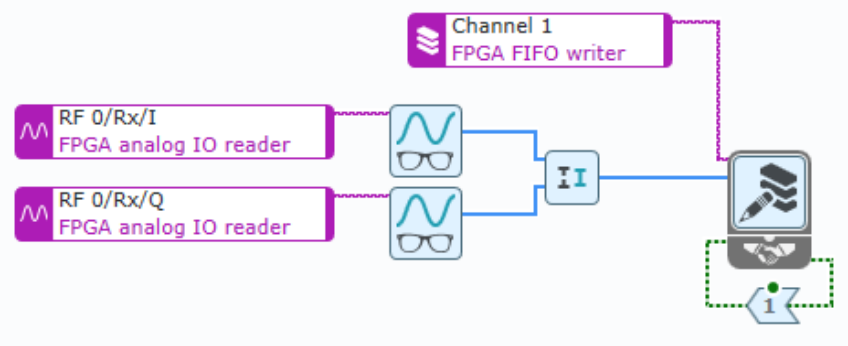
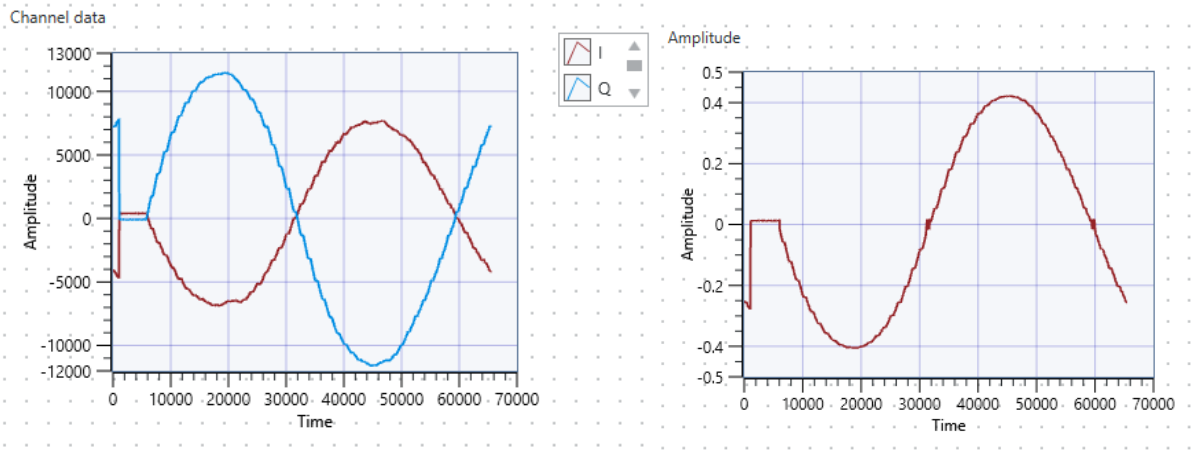


FIGURE 17 – Signal reçu et transmis par FIFO.

Ces données brutes reçues, visible sur la figure 18a, sont traitées afin de retrouver le signal original reçu. Dans le but de retrouver le signal original, il faut additionner les signaux I et Q.

$$s(t) = I(t) + Q(t) \tag{21}$$



(a) Données I et Q brute. **(b)** Données I et Q traitées.

FIGURE 18 – Données I et Q reçu.

Le signal reçu est semblable à celui émis, ce qui confirme que la configuration est correct ainsi que l'émission et la réception fonctionne.

4.4.1. Calcul d'amplitude

Les données reçu étant la sortie directe de l'ADC il faut les transformer pour obtenir des valeurs représentative de la puissance du signal. Pour cela, comme étudié dans la section 2.3.2, les données sont multipliées par le facteur déterminé par le gain appliqué.

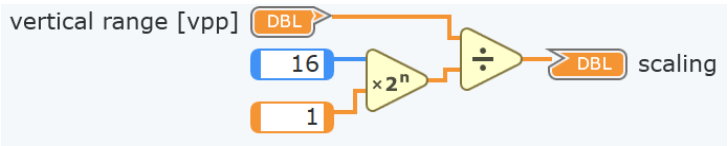


FIGURE 19 – Conversion des données de nombre entier à chiffre à virgule.

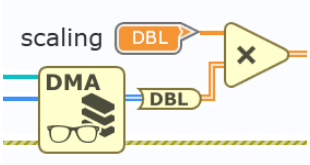


FIGURE 20 – Dimensionnement en tension des données.

4.5. Traitement du signal

4.5.1. Émission

LabView propose une fonction d'interpolation. Les données générées n'étant pas nécessairement à la même fréquence d'échantillonnage, il faut les interpoler afin d'obtenir, en sortie RF, le signal désiré. La valeur de configuration de la fonction est le ratio entre la fréquence désiré et l'horloge de la FPGA.

$$R_s = \frac{f_{in}}{f_{clk}} \tag{22}$$

Dans cet exemple le signal généré est carré avec un échantillonnage à 20 MHz et une horloge à 120 MHz, ce qui donne un facteur de $0.1\overline{6}$.

$$R_s = \frac{\text{data sample rate}}{\text{clock}} = \frac{20M}{120M} = 0.1\overline{6} \tag{23}$$

Dans la figure 21 on observe que la valeur du facteur n'est pas exactement la même que celle calculée. Ceci est du à la précision de 48 bits de la variable.

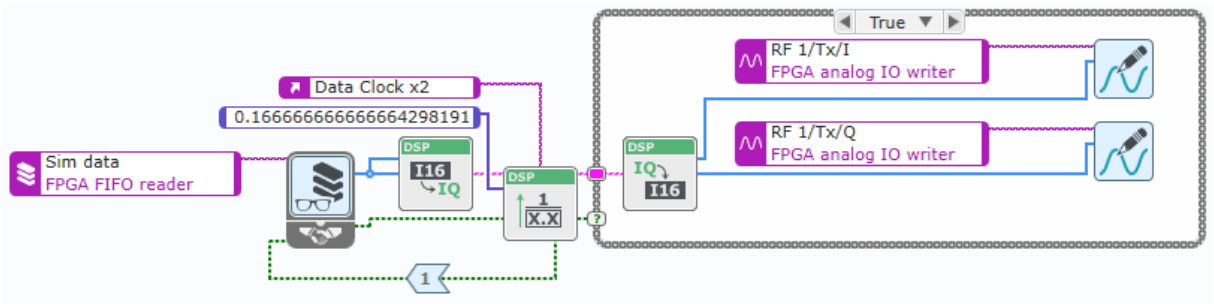
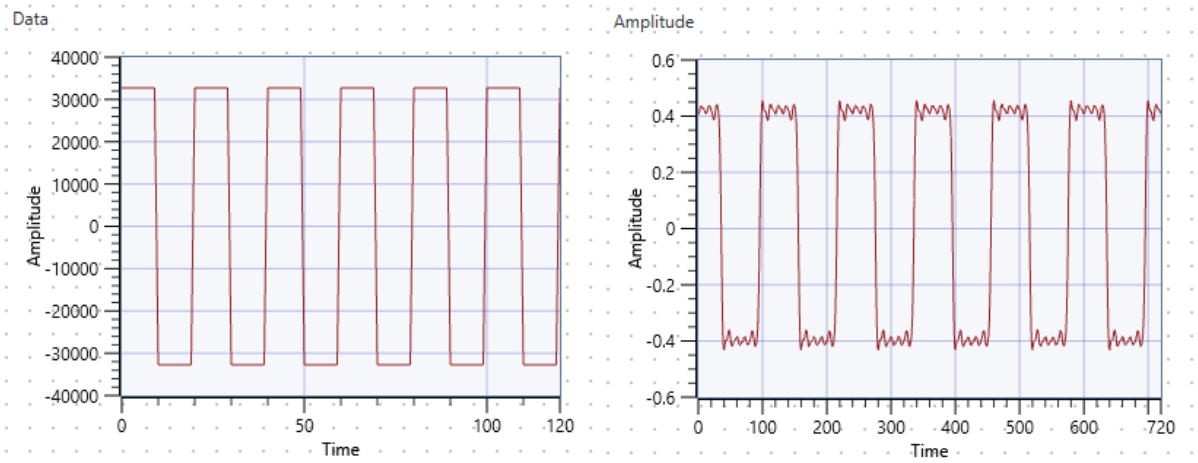


FIGURE 21 – Envoi de données interpolées.

Lors de la réception les données ne sont pas décimées et donc pour le même nombre de période on obtient plus d'échantillon. Un zoom sur une section de 120 échantillons envoyé correspondent aux données reçues qui ont 6 fois plus de données dû à l'interpolation d'émission.

$$N_{\text{sample receive}} = N_{\text{sample send}} \cdot \frac{\text{clock}}{\text{data sample rate}} = 120 \cdot \frac{120M}{20M} = 720 \tag{24}$$



(a) 120 échantillons de données envoyées. **(b)** 720 échantillons de données reçues.

FIGURE 22 – Comparaison de données envoyées avec interpolation et reçues

Malgré un déphasage les données reçues sont visuellement similaires et confirment le transfert du signal.

4.5.2. Réception

De la même manière LabVIEW mets a disposition une fonction de décimation. le calcul de facteur est identique à l'interpolation.

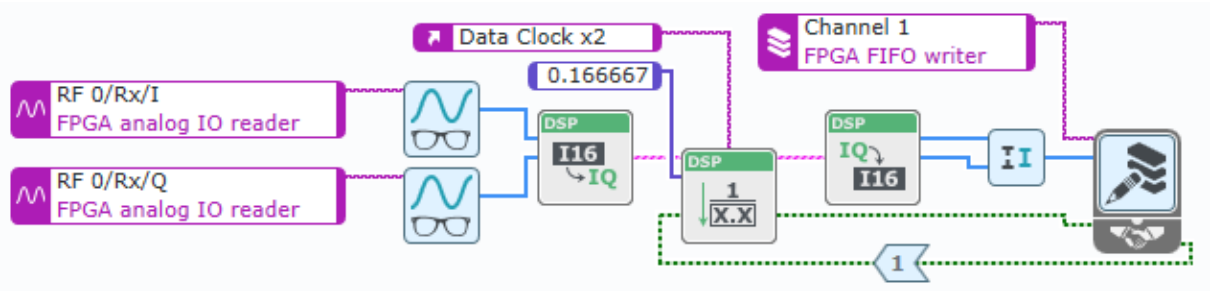


FIGURE 23 – Décimation des données reçues.

Selon la figure 22a, on retrouve cette fois-ci le même nombre de symboles que pour l'émission avec le même nombre d'échantillons grâce à la décimation.

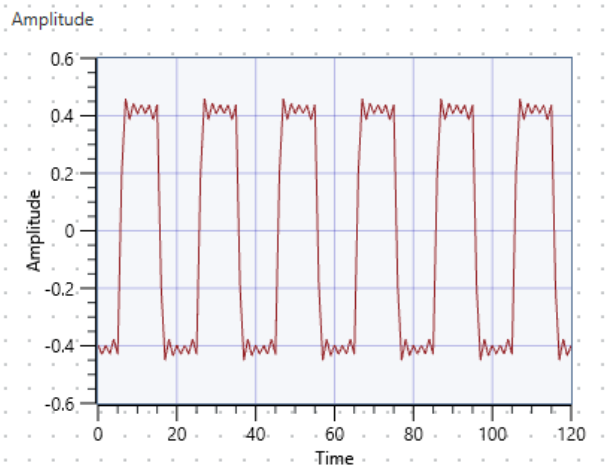


FIGURE 24 – 120 échantillons de données reçues décimées.

4.6. Paramétrage de la FPGA

Afin de rendre le programme sur FPGA le plus polyvalent possible lors de son développement, il est possible de mettre des variables en accès externe et accessible depuis le PC hôte. Pour que celles-ci garde leurs valeurs, il faut mettre en place des registres et leur définir la structure des données qu'ils contiendront.

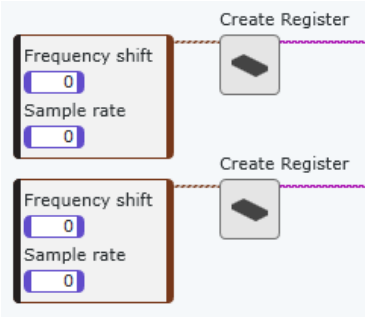


FIGURE 25 – Création des registres.

Pour déclarer ces variables externe, il suffit de les placer comme variable de type "controle" dans le

fichier principal du programme et dans une boucle géré par une horloge. Dans cette même boucle les variables seront écrites sur le registre.

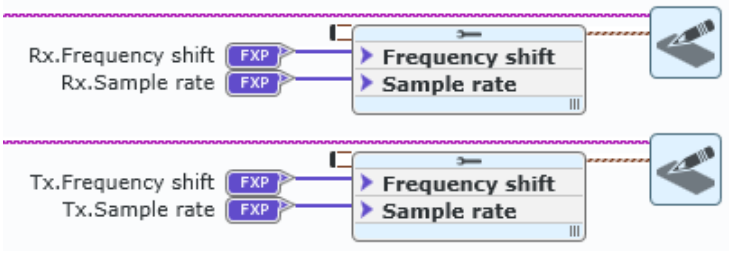


FIGURE 26 – Déclaration des paramètres.

Dans la figure suivante, on y vois l'utilisation d'un registre qui configure l'interpolation et le déplacement de fréquence dans la ligne de traitement de signal.

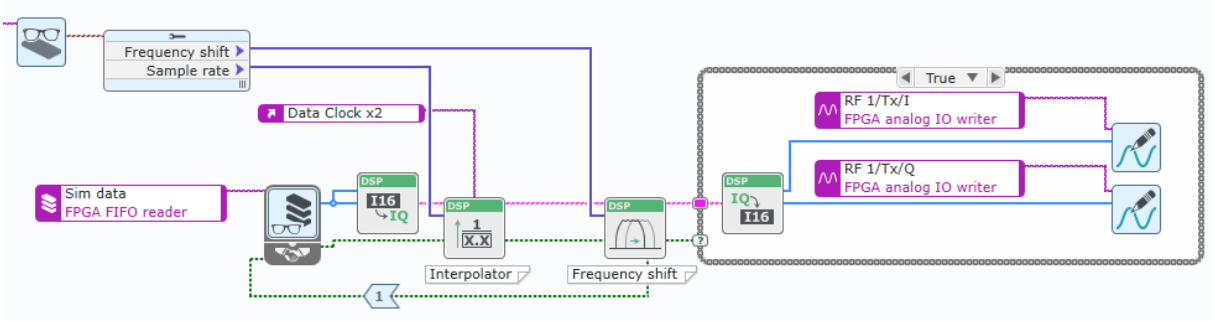


FIGURE 27 – Envoi de données avec déplacement en fréquence.

De même pour la réception, le signal est déplacé en fréquence afin de retirer le signal de porteuse restant et décimé pour atteindre la vitesse de transfert désiré sur le PC hôte.

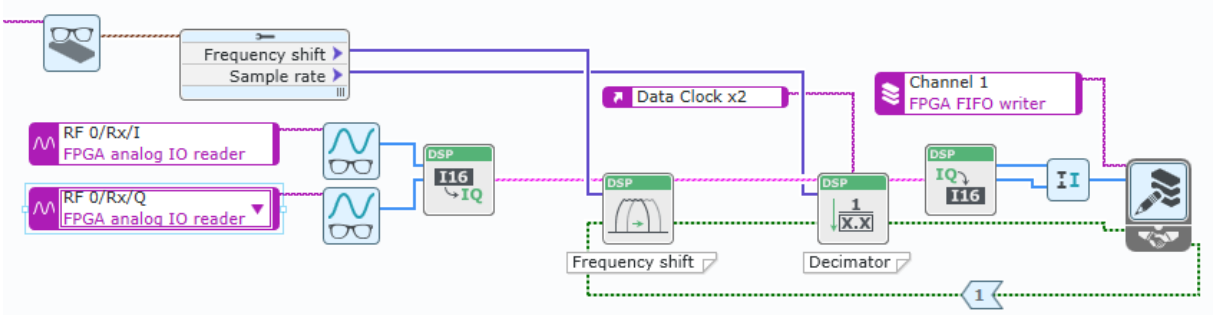


FIGURE 28 – Réception de données avec déplacement en fréquence.

Sur le programme hôte ces paramètre sont modifié au travers de la référence de la FPGA, qui contient les adresses système.

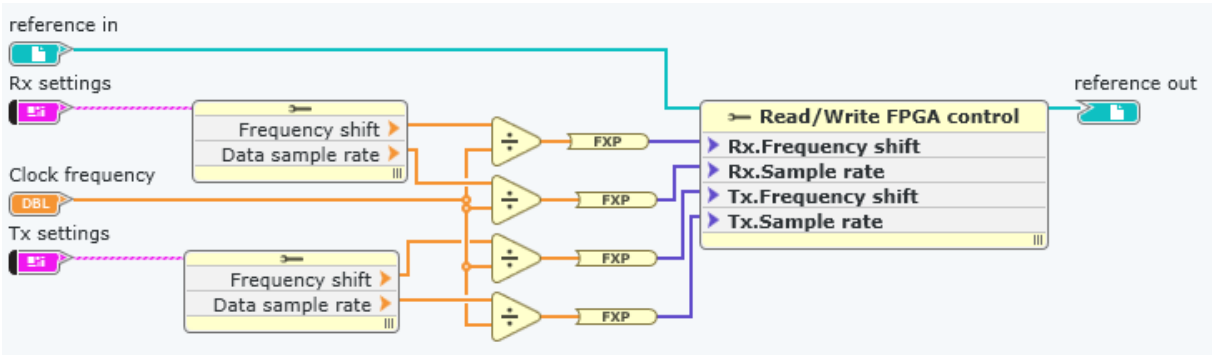


FIGURE 29 – Configuration de la FPGA

4.7. Fin du programme

Une tâche importante est la fermeture de la session en fin de programme, si cela n'est pas fait, lors de la prochaine exécution il y a de grandes possibilités que le programme ait une erreur et ne démarre pas. Pour cela la fonction *Close* assure la déconnexion du système avec l'USRP.

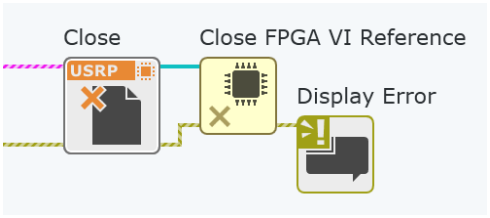


FIGURE 30 – Fermeture de la session.

5. Applications spécifique au protocole DM

5.1. Paramètres sur la FPGA

A des fins de modularité le programme sur FPGA est configurable au travers de variables accessibles depuis le programme du PC hôte. Elles sont sauvegardées dans des registres qui sont inscrits à une fréquence de 40 MHz dans une boucle en dehors de la modulation et de la démodulation. Ici les valeurs de déplacement de fréquence pour les 40 canaux sont calculées pour ne pas entrer dans la ligne de traitement de démodulation du signal.

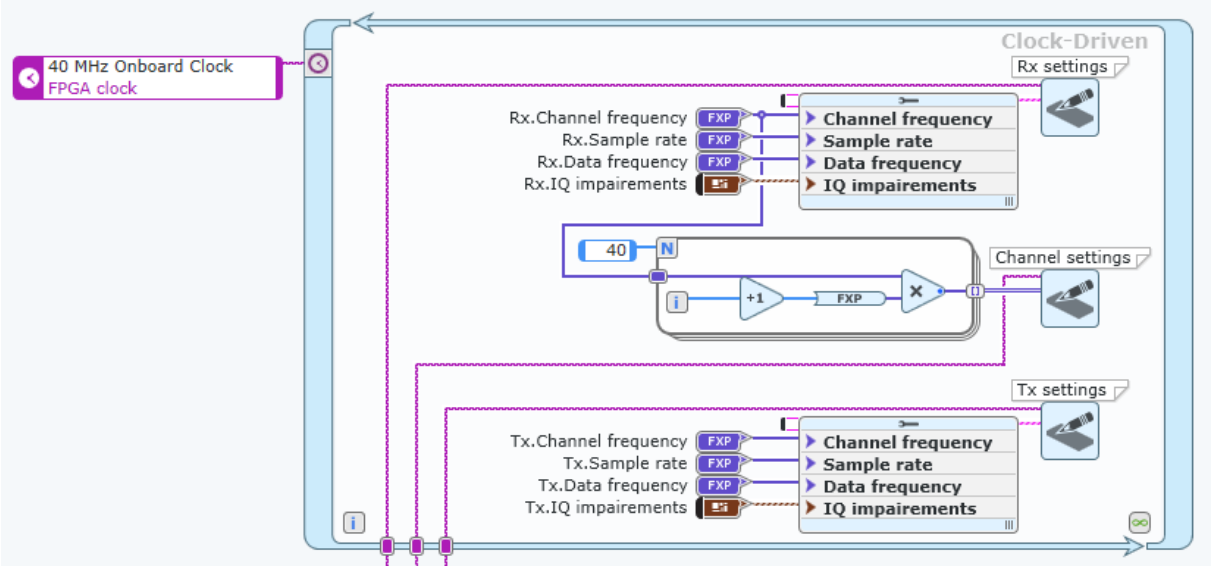


FIGURE 31 – Gestion des registres et des paramètres.

5.2. Modulation

Comme défini la modulation GFSK du protocole DM le bit à 1 est représenté par une fréquence de +320 kHz et le bit à 0 par une fréquence de −320 kHz sur le canal. Les données traitées fournissent à la FPGA des valeurs de −1 pour un bit à 0 et +1 pour un bit à 1. Ces valeurs sont multipliées par le facteur de déplacement de fréquence qui représente 320 kHz selon la formule qui suit.

$$\text{Data frequency} = \frac{f_{in}}{f_{clk}} = \frac{320\,\text{kHz}}{120\,\text{MHz}} = 0.002\overline{6} \tag{25}$$

Pour que les données envoyées simulent le plus correctement le fonctionnement d'un appareil, l'échantillonnage des données est de 2 MHz et les données sont ensuite interpolées sur la FPGA pour correspondre à sa vitesse de traitement de 120 MHz.

L'USRP est imparfait, et cela est connu du constructeur. Pour cela l'appareil contient des facteur de correction qui permette de corrigé les imperfections de l'émetteur et du récepteur.

Pour l'émission, les données à émettre influe uniquement la fréquence, ce qui laisse donc l'amplitude à définir. Elle a été mise à sa valeur maximale pour avoir une puissance de sortie référencée à 0 dBm

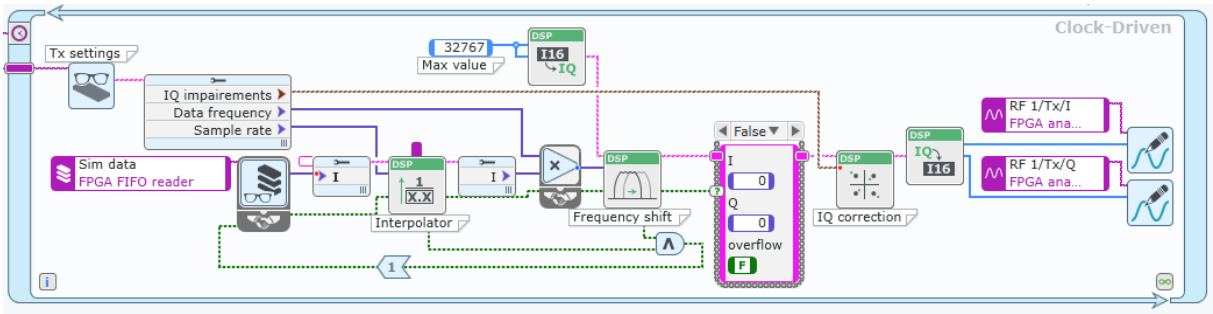


FIGURE 32 – Modulation du signal sur FPGA.

5.3. Démodulation

Selon les choix présentés en section 3, la première option était de continuer la démodulation par déplacement de fréquence avec les fonctions spécifiques à l'USRP, mais la complexité sur la FPGA

étant trop grande empêcha la finalité de cette méthode, son développement est disponible en annexe B.1.

La méthode de démodulation c'est donc porté sur le quadricorrelateur qui à l'implémentation la plus simple et la meilleure performance selon les documents étudiés et cités précédemment.

Le traitement de signal sur la FPGA est représenté en figure 33. Les signaux I et Q sont joints pour convenir au format dont travail les outils mis à disposition par LabVIEW. La correction de phase entre les signaux quadratique est appliquée comme pour la modulation. Cette ligne démodule le premier canal, pour cela dans le tableau des paramètres de déplacement de fréquence celui en position 0 est utilisé. Et après être passé par le quadricorrelateur les données sont écrites sur la FIFO du canal correspondant.

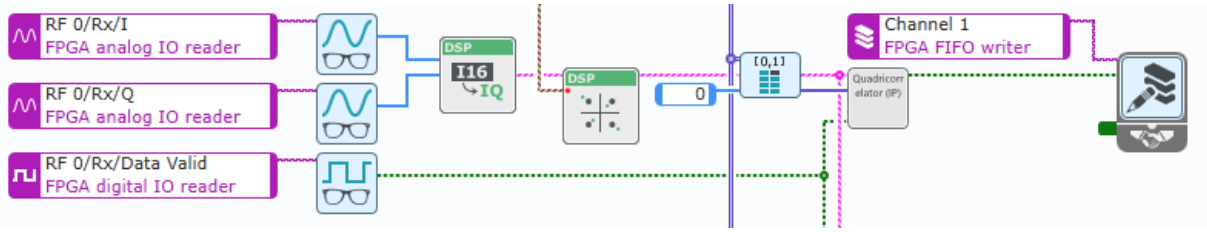


FIGURE 33 – Traitement du signal par le quadricorrelateur.

Les signaux sont mis en bande de base pour convenir au démodulateur. Un filtre passe bas sous la forme d'une moyenne glissante permet de supprimer les fréquences supérieures des autres canaux et des bruits ambiants. Chaque composante quadratique est retardé de 94 coup d'horloge, défini par la formule (20).

$$N = \frac{f_{clk}}{4f_d} = \frac{120\,\text{MHz}}{4 \cdot 320\,\text{kHz}} = 93.75 \approx 94 \tag{26}$$

Cependant, malgré que l'amplitude du signal sois maximale comme prévu, les données reçues ne sont pas stables pour une raison inconnue, en réduisant cette valeur à 30 on retrouve un niveau correct.

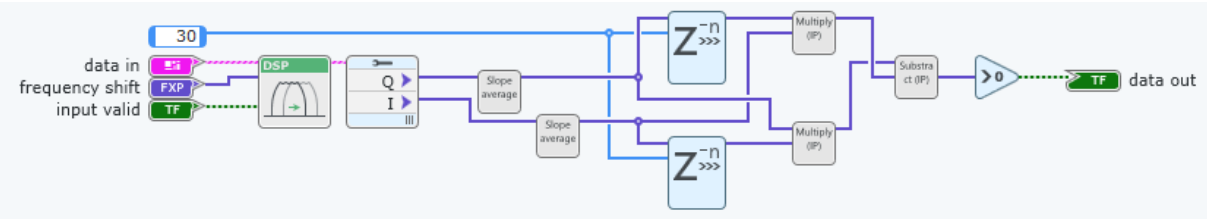
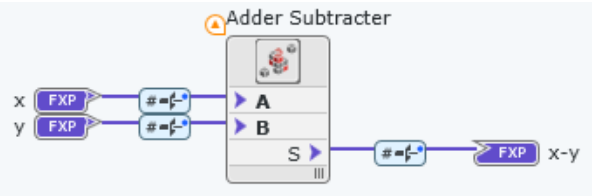
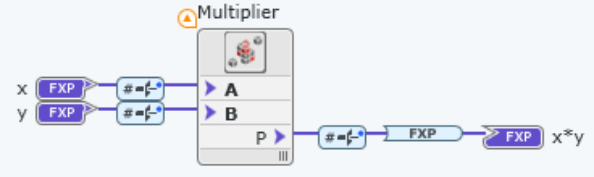


FIGURE 34 – Quadricorrelateur par IP.

Les multiplication et soustraction ne sont pas de simples blocs de math car ne connaissant pas leur fonctionnement interne, il en résulte des problèmes de timing lors de la compilation de la FPGA. Pour résoudre ce problème, des blocs IPs qui utilise des zones DSP sont utilisées.



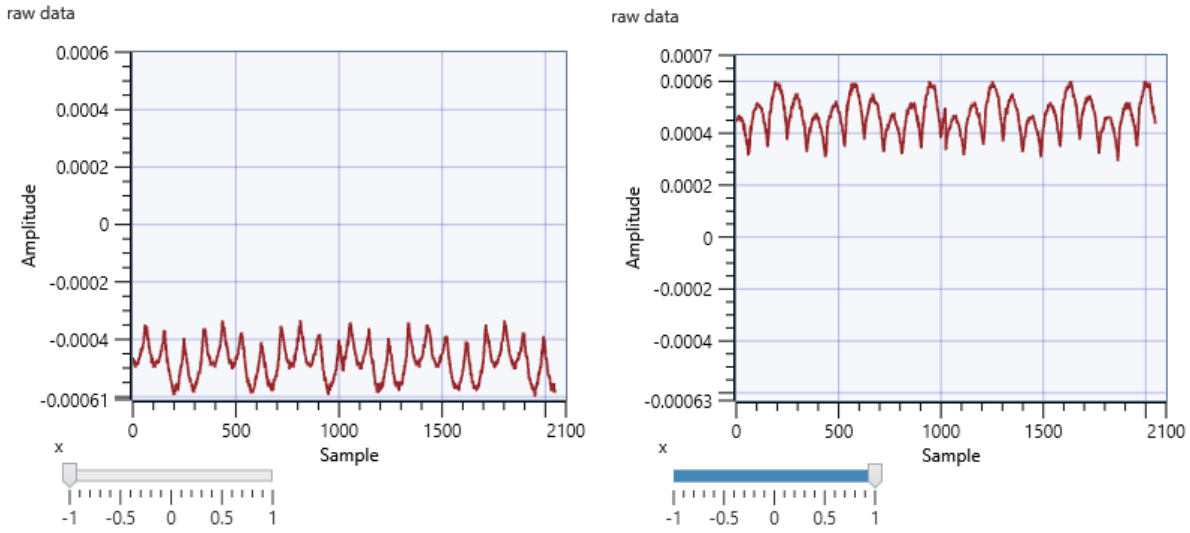
(a) Soustraction.



(b) Multiplication.

Quand un bit à zéro, représenté par une fréquence de -320 kHz, est reçu le quadricorrelateur a en

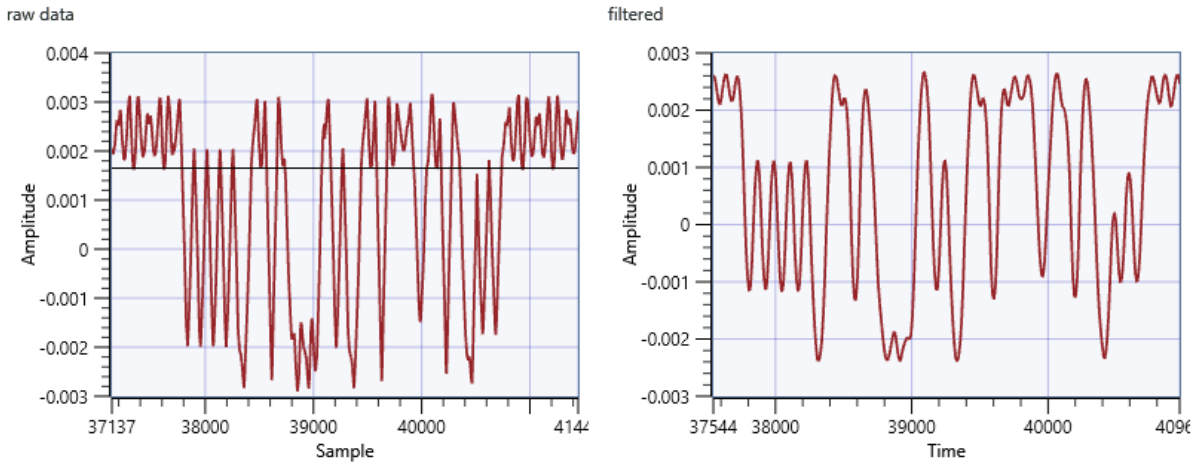
sortie une amplitude négative, exemple en figure 36a et quand le bit reçu est à 1, une fréquence de +320 kHz, l'amplitude est positive, figure 36b.



(a) Bit à 0 démodulé par quadricorrelateur. **(b)** Bit à 0 démodulé par quadricorrelateur.

FIGURE 36 – Bit démodulé par quadricorrelateur.

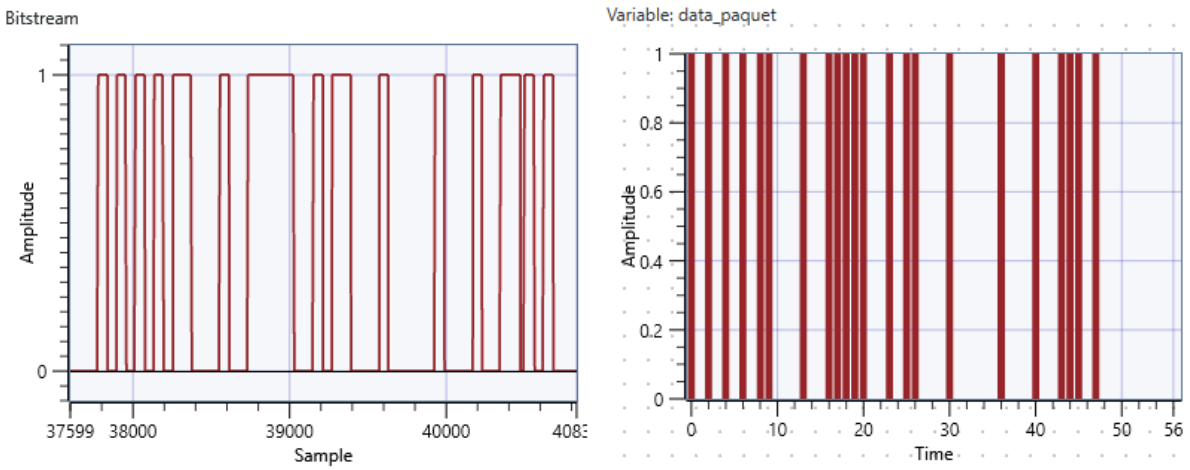
Le signal généré est reçu figure 37a, il passe au travers du démodulateur. Le signal sortant figure 37b est passé au travers d'un filtre passe bas, constitué d'une moyenne glissante de $f_{clk}/R_s = 60$ valeurs.



(a) Données générées, reçues. **(b)** Données générées, reçues et filtrées.

FIGURE 37 – Données générées, reçues manipulées.

Tel expliqué précédemment le bitstream est sorti en comparant si le signal est supérieur à 0 et si cela est vrai la donnée est un bit à 1 et si cela est faux la donnée es un bit à 0. Figure 38a. Ces données sont sur-échantillonnées et doivent être décimé afin d'en sortir le paquet. Grâce à l'algorithme de synchronisation le multiple de sur-échantillonnage est déterminé et appliqué à la décimation. 38b



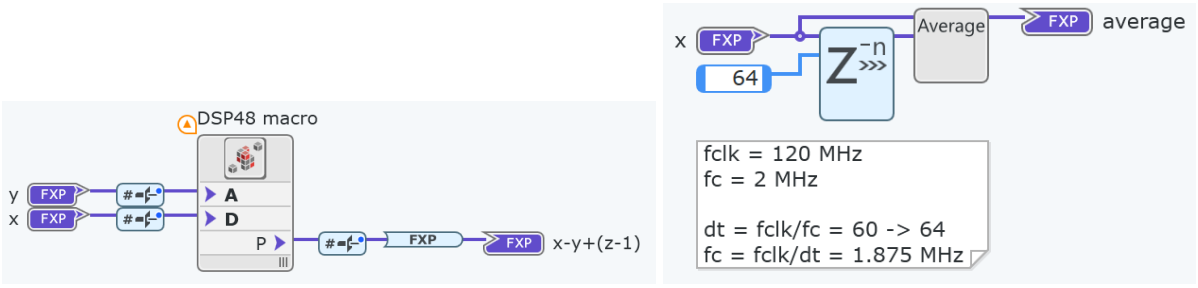
(a) Bitstream des données reçues. **(b)** Bitstream décimé des données reçues.

FIGURE 38 – Bitstream reçu.

5.3.1. Filtre passe bas

La moyenne est effectuée par un bloc DSP48. L'entrée *x* est soustraite par *y* et le résultat de l'opération précédente y est additionné.

L'entrée *y* est retardée de 64 coups d'horloge, théoriquement de 60 tel le démontre la formule en figure 39b, mais monté à 64 pour simplifier la division et obtenir un multiple de 2 qui permet de simplement effectuer un déplacement de bit vers la droite en réinterprétant la donnée de sortie de la fonction de moyenne.



(a) Moyenne. **(b)** Moyenne glissante.

FIGURE 39 – Moyenne glissante sur 64 valeurs.

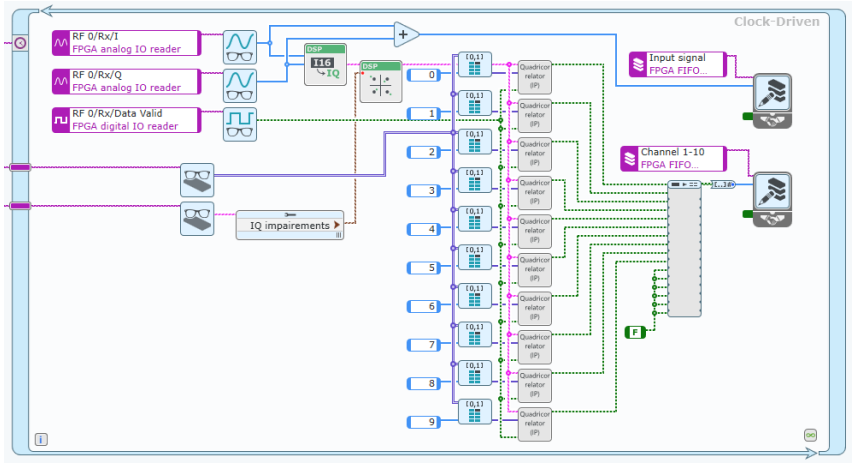


FIGURE 40 – Démodulation de 10 canaux en parallèle.

5.3.2. Parallélisation des canaux

Le signal reçu sur la FPGA contient l'information de plusieurs canaux, afin de les séparer chacun des canaux et le résultat du déplacement en fréquence de la valeur du canal, qui est un multiple de 2 MHz. Puis traiter comme défini précédemment.

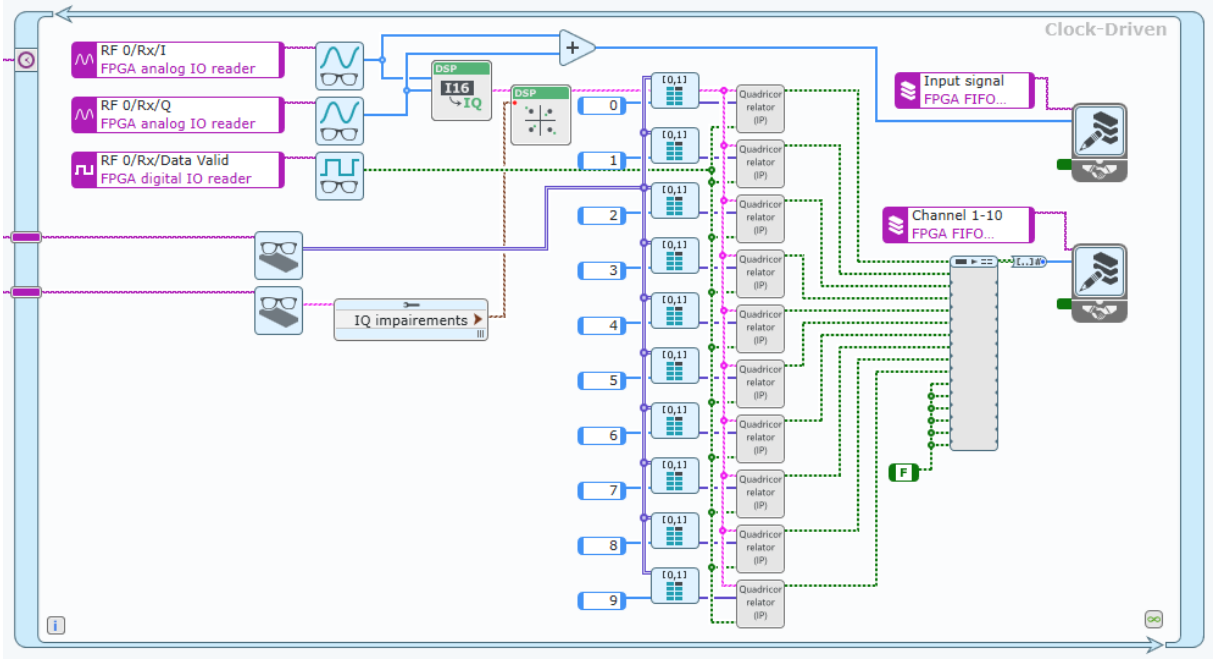


FIGURE 41 – Démodulation de 10 canaux en parallèle.

5.4. Codage

La création du paquet consiste à assembler les données telles que l'adresse et le payload dans le format que le protocole décrit. La façon la plus simple est de créer un tableau de valeurs binaires, attention à l'ordre des données pour qu'à l'émission le MSB soit envoyé en premier.

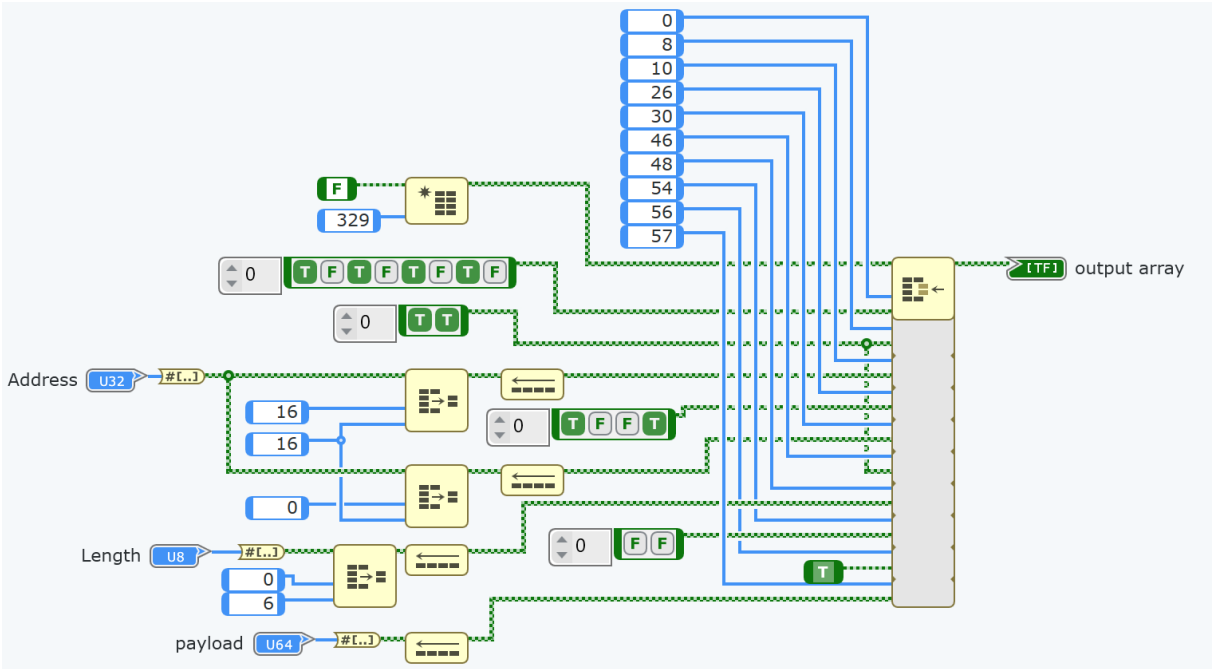


FIGURE 42 – Assemblage des données du paquet.

Ce tableau binaire est transformé en valeurs entières de 16 bits et sur-échantillonnées à la fréquence d'échantillonnage paramétré.

Le filtre gaussien y est ensuite généré et appliqué pour les transmettre à la FPGA.

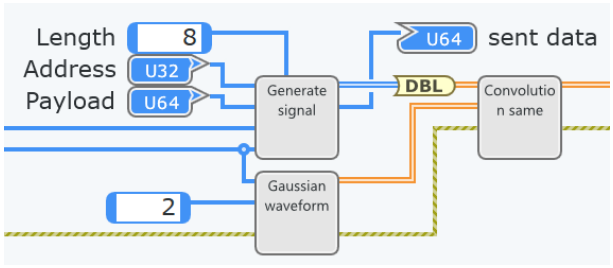


FIGURE 43 – Génération et traitement du paquet à émettre.

5.5. Décodage

Le bitstream reçu de l'USRP est traité afin d'en extraire les paquets reçus.

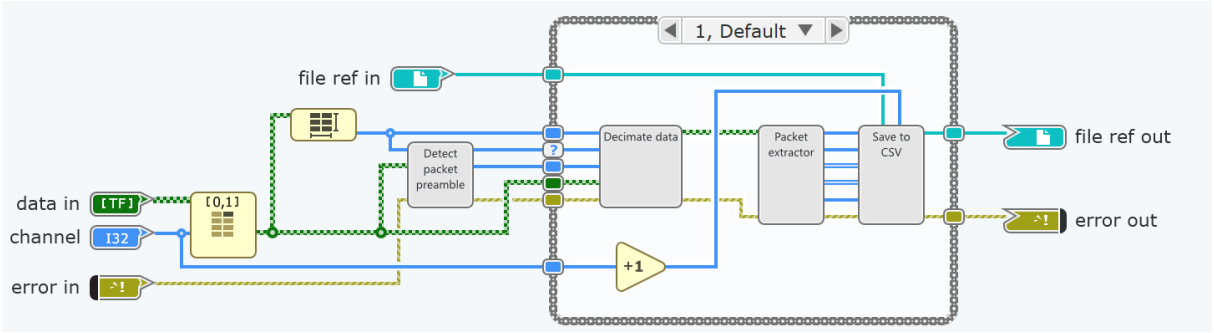


FIGURE 44 – Traitement du bitstream.

5.5.1. Synchronisation

L'horloge d'émission n'étant pas la même que celle de réception les données reçues arrive à un temps t inconnu. Pour récupérer les bits de données au bon moment le protocole défini un octet de synchronisation qui se compose d'une alternance de 1 et de 0, commençant pas un 1 si le premier bit d'information et un 1 et 0 si le premier bit d'information et un 0.

Cette partie étant uniquement du traitement de données binaire l'approche en programmation standard est plus indiqué. LabVIEW permet d'intégrer des blocs de code en C qui sont effectués durant l'exécution du programme. Une première étape est de déterminer la fréquence de sur-échantillonnage que les données contiennent. Pour cela le paquet contient un préambule de bit alternant entre 1 et 0, ceci permet ainsi de déterminer le nombre d'échantillons que contient un seul bit en faisant le compte du nombre d'échantillons entre un bit à 1 et un bit à 0. Sur ces 8 bits de synchronisation 5 sont pris pour effectuer une moyenne de longueur d'échantillons par bit. Cette mesure prend en compte exclusivement les valeurs consécutives qui reste dans une similarité de 25 %. Dans le cas contraire la détection est remise à zéro afin de prendre les données suivantes.

```
1   #define threshold_max 1.25
2   #define threshold_min 0.75
3   #define edge_count 5
4   #define min_delta 45 // (120 MHz / 2 MHz) * threshold_min
5
6   int first_occur = 0;
7   int last_occur = 0;
8   int delta_i0 = 0;
9   int delta_i1 = 0;
10  int delta_sum = 0;
```

```
11  int same = 0;
12  start_paquet = 0;
13  bit_length = 0;
14
15  for(int i = 0; i < data_size-1; i++){
16    if(data_in[i] ^ data_in[i+1]){
17      if(first_occur == 0){
18        first_occur = i+1;
19        last_occur = i+1;
20      }else{
21        if(delta_i0 == 0){
22          delta_i0 = (int)((i + 1) - last_occur);
23          delta_i1 = delta_i0;
24          delta_sum = delta_i0;
25        }else{
26          delta_i0 = (int)((i + 1) - last_occur);
27          if(delta_i0 > delta_i1 * threshold_min &&
28             delta_i0 < delta_i1 * threshold_max &&
29             delta_i0 > min_delta){
30            same ++;
31            if(same >= edge_count){
32              bit_length = (int)(delta_sum / edge_count);
33              start_paquet = first_occur + (int)(bit_length / 2);
34              delta_i0 = 0;
35              delta_i1 = 0;
36              delta_sum = 0;
37              same = 0;
38              first_occur = i + 1;
39              break; //continue;
40            }
41            delta_sum += delta_i0;
42          }else{
43            delta_i0 = 0;
44            delta_i1 = 0;
45            delta_sum = 0;
46            same = 0;
47            first_occur = i + 1;
48            last_occur = i + 1;
49          }
50        }
51        last_occur = i + 1;
52      }
53    }
54  }
```

Code 1 – Code de détection de la fréquence d'échantillonnage.

5.5.2. Décimation

La fréquence d'échantillonnage et l'emplacement de début du paquet défini, les données sont décimées afin d'obtenir les bits effectifs et ressortir le paquet reçu. Le tableau de sorti est limité à 329 valeurs, ce qui est le nombre maximal de bit que peut contenir un paquet.

```
1  #define max_packet_size 329
2  #define min_packet_size 65
3
4  int index = 0;
5  cnode_size_array(data_paquet, max_packet_size);
6
7  for(int i = start_paquet; i < data_size; i += bit_length){
8    data_paquet[index++] = data_in[i];
9    if(index >= max_packet_size) break;
10 }
11 index = 0;
```

Code 2 – Décimation des données d'entrées.

5.5.3. Extraction du paquet

Les données du paquet étant disponible sous forme d'un tableau binaire, il faut découper celui-ci pour extraire les informations utiles formatées. Le document fourni par Sonova explique la structure d'un paquet et permet de définir les informations que représente chaque bit de ce tableau.

Le protocole défini plusieurs types d'adresse, celui utilisé par Sonova étant en dehors des connaissances disponibles, le choix c'est porté sur le *Private Beacon Address* car il s'agit, tel son nom l'indique, d'un formé utilisé pour les adresses privées.

```
#define data_in_length 329
#define payload_length 32
#define PCF_length 3

#define preamble_start 0 // 0
#define address_start preamble_start + 8 // 8
#define PCF_start address_start + 5 * 8 // 48
#define payload_start PCF_start + 9 // 57

cnode_size_array(payload, payload_length);
cnode_size_array(PCF, PCF_length);
preamble = 0;
address = 0;
CRC = 0;

for(int i = 0; i < PCF_length; i++){
  PCF[i] = 0;
}
for(int i = 0; i < payload_length; i++){
  payload[i] = 0;
}
for(int i = 0; i < 8; i++){
  preamble |= data_in[i] << (7-i);
}
for(int i = 0; i < 16; i++){
  address |= data_in[i + address_start + 2] << (31-i);
}
for(int i = 0; i < 16; i++){
  address |= data_in[i + address_start + 22] << (15-i);
}
for(int i = 0; i < 6; i++){
  PCF[0] |= data_in[i+PCF_start] << (5-i);
}
for(int i = 0; i < 2; i++){
  PCF[1] |= data_in[i+PCF_start+6] << (1-i);
}
PCF[2] |= data_in[PCF_start+8];

if(PCF[0] >= payload_length) PCF[0] = payload_length;
for(int i = 0; i < PCF[0]; i++){
  for(int j = 0; j < 8; j++){
    payload[i] |= data_in[(i*8)+j+payload_start] << (7-j);
  }
}

for(int i = 0; i < 16; i++){
  CRC |= data_in[i+payload_start+PCF[0]*8] << (16-i);
}
```

Code 3 – Extraction des informations du paquet.

5.5.4. Fichiers de sauvegarde

Au démarrage du programme les fichiers de sauvegarde sont créés et nommés avec la date et l'heure actuel afin de les retrouver dans un ordre chronologique. Le nom du fichier ainsi que la première ligne, qui représente les nominations des données enregistrées, sont requises comme paramètre.

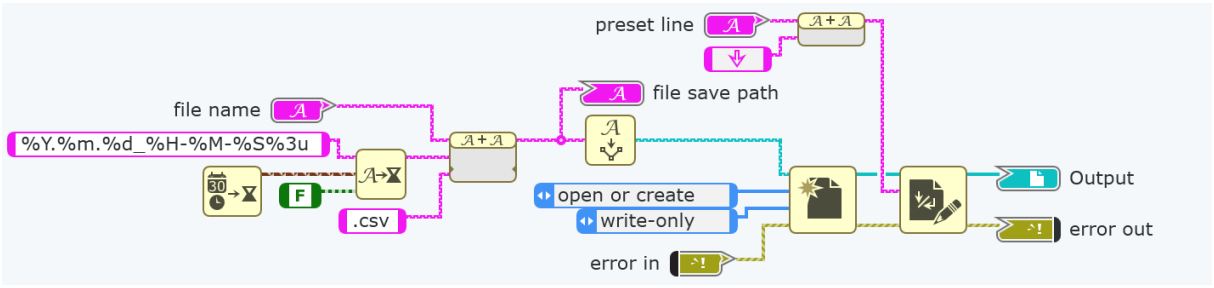


FIGURE 45 – Création d'un fichier de sauvegarde.

5.5.5. Sauvegarde des données

Le paquet ainsi extrait, ces données sont enregistrées dans un fichier CSV, le format de celui-ci est défini dans le tableau 1 de la spécification.

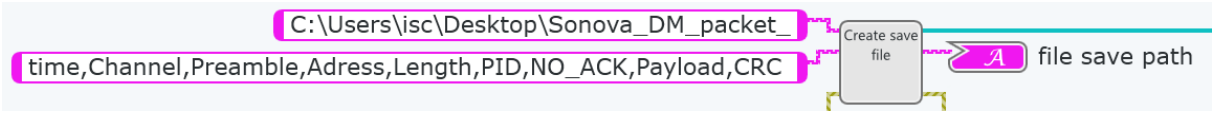


FIGURE 46 – Création du fichier de sauvegarde des paquets.

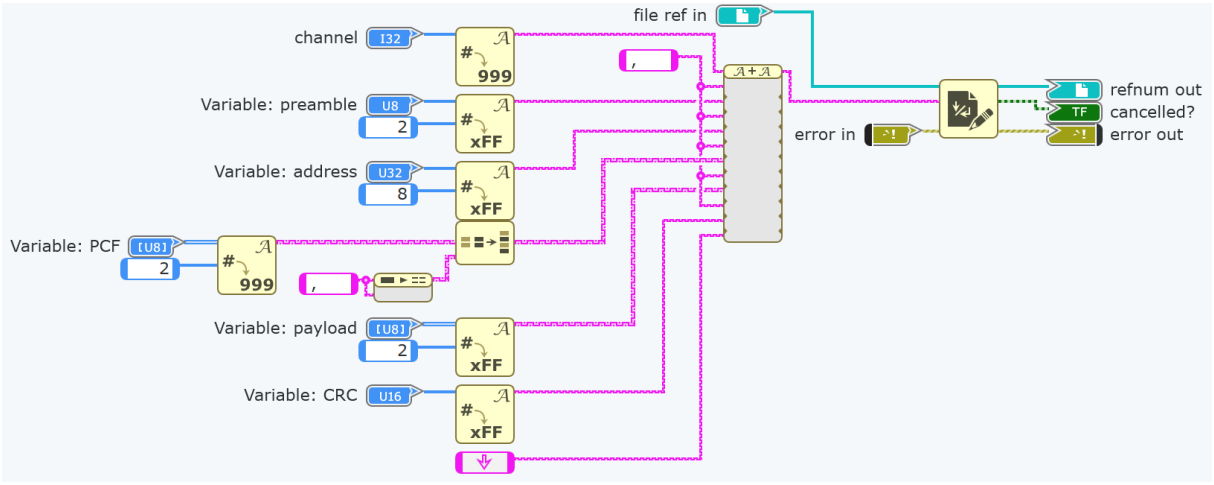


FIGURE 47 – Sauvegarde du paquet en CSV.

5.6. Activité des canaux

Le signal reçu brut de la FPGA s_{in}, a une amplitude relative à l'ADC, pour la convertir en volt les données sont transformées tel défini en section 2.3.2.

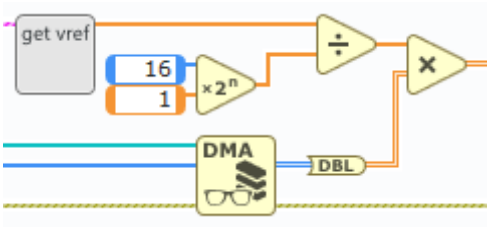


FIGURE 48 – Conversion des données brutes de l'ADC en volts.

Une FFT est effectué sur les données transformées afin d'en ressortir les amplitudes relatives à chaque fréquence que consiste le signal. Finalement les données sont mises en dBm pour avoir une référence générique pour les signaux RF.

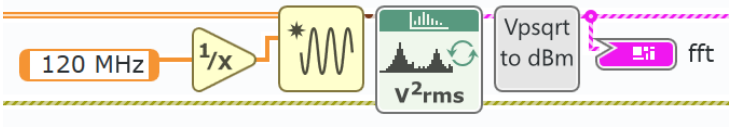


FIGURE 49 – Transformer de Fourier sur le signal brut.

L'ensemble des données sortant de la FFT ne sont pas nécessaires. Pour cela seulement les amplitudes représentants une fréquence d'un canal sont extraits. Le décalage par canal de 2 MHz est connu, la corrélation avec la liste des données est la suivante.

$$A_{canal}(i) = \frac{\Delta f}{(2\,\text{MHz} \cdot i)} \tag{27}$$

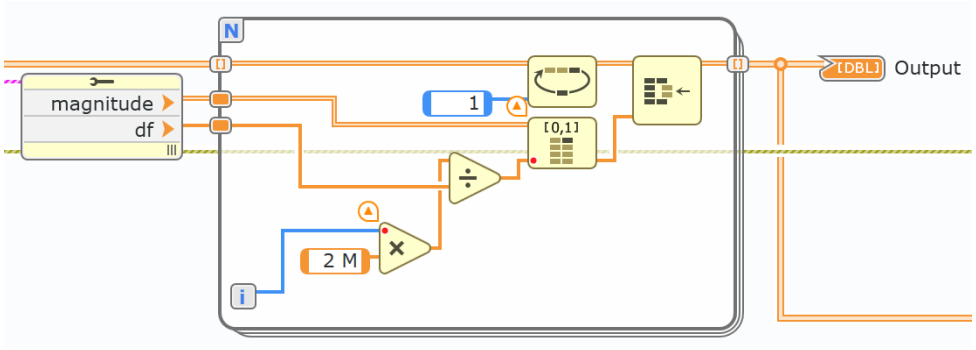


FIGURE 50 – Affichage sur un digramme en waterfall.

En plus de cette opération, les données sont mises dans un tableau tournant afin de les afficher dans l'interface graphique.

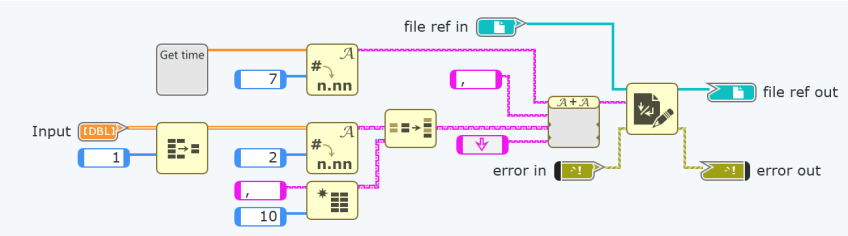


FIGURE 51 – Sauvegarde de l'activité sur un fichier CSV.

5.7. Interface utilisateur

Finalement après tout le développement technique il faut que l'utilisateur puisse interagir avec le programme. LabVIEW à la possibilité de réaliser une interface graphique simple qui permet de paramétrer le système. Avec le protocole utiliser la majorité des éléments sont prédéfinis tels que la fréquence de l'oscillateur RF, l'écart entre les canaux, l'index de modulation et les fréquences d'échantillonnages d'émission et de réception.

À sa disposition l'utilisateur peut choisir.

- Le canal d'émission et de réception.
- L'antenne d'émission et de réception.
- Le canal sur lequel les données simulées sont envoyé.
- Le décalage de canaux reçu pour les 10 démodulés.

— Le gain de l'amplificateur d'émission et de réception en dBm.

— L'activation de l'envoie des données simulées.

— L'adresse du paquet simulé en hexadécimal.

— Les données du paquet simulé sur 64 bits (8 octets) en hexadécimal.

— Le nombre de canaux décodés et sauvegardés.

— Le traitement et la sauvegarde des paquets et de l'activité.

Au lancement de l'application ces paramètres sont appliqués. Ils subissent un traitement et une correction par le système pour les valeurs limites et la résolution des valeurs selon leurs nombres de bit effectifs. Ils sont visibles sur le panneau "Paramètres actif" qui, pour le canal d'émission et de réception, se regroupe ainsi.

— Frequency : Fréquence de l'oscillateur RF.

— RF channel : Le canal RF de l'USRP.

— Antenna : L'antenne du canal RF.

— Gain : Le gain effectif en sortie ou entré du canal RF.

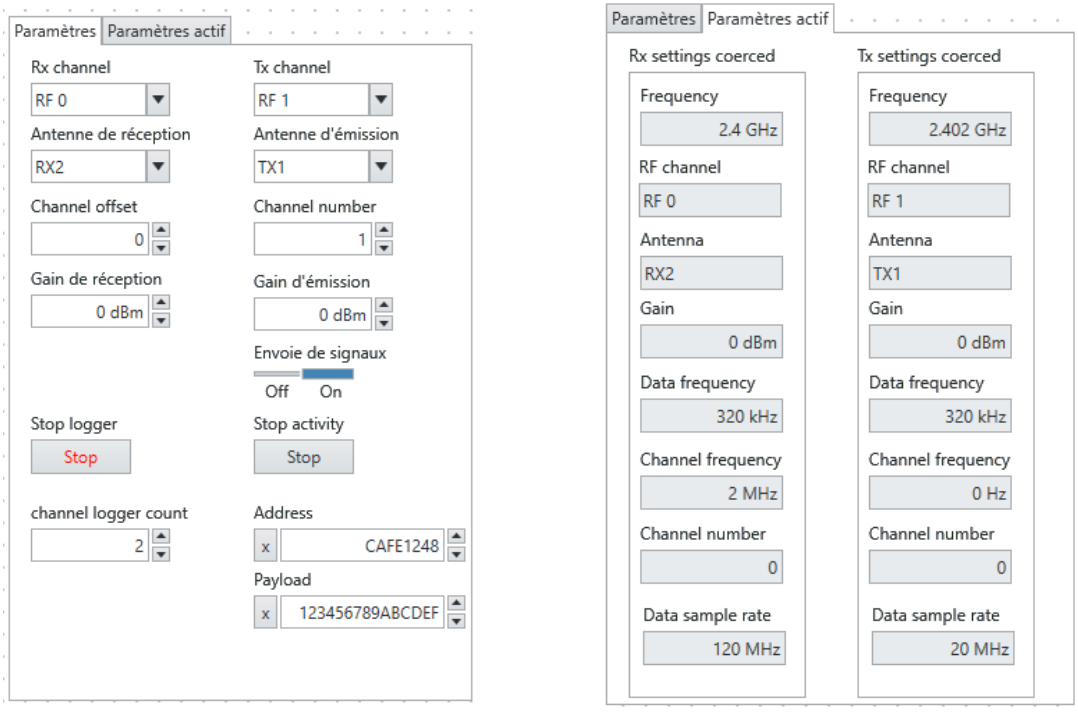
— Data frequency : Résultat représentant l'index de modulation.

— Channel frequency : Espacement entre les canaux.

— Channel number : Numéro du canal actif.

— Data sample rate : Fréquence d'échantillonnage respectivement en entrée ou en sortie de l'USRP.

Le canal démission à comme spécificité que le canal choisi est directement appliqué sur l'oscillateur RF afin de réduire les potentielles déformations du signal que peuvent engendrer la fonction de déplacement de fréquence de la FPGA. Et de ce fait remet à 0 la valeur du numéro de canal et sa fréquence.



(a) Panneau de paramètres. **(b)** Panneau de paramètres appliqués.

FIGURE 52 – Panneau de configuration.

Dans la visualisation des données l'utilisateur retrouve un graphique, figure 53, représentant les données du paquet traité et tel qu'elles seront envoyées à la FPGA pour être modulées et émises. Avec en plus la représentation hexadécimal du paquet avec l'ensemble de ces données de contrôles.

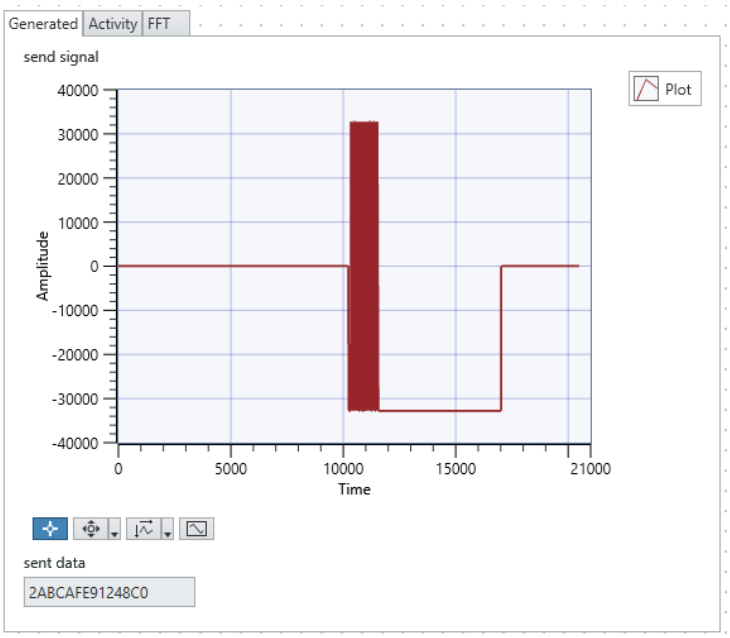
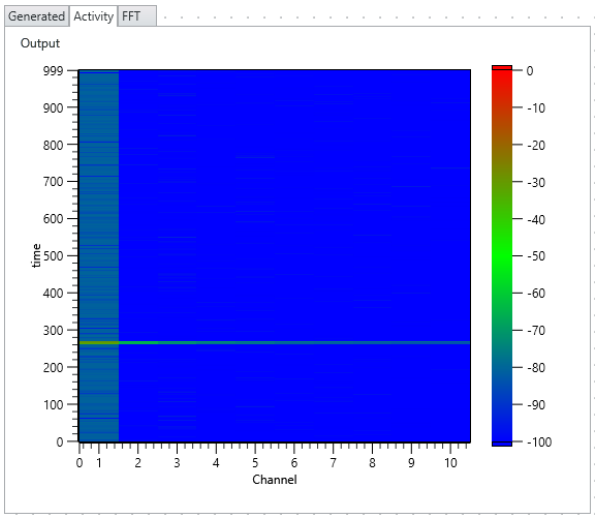
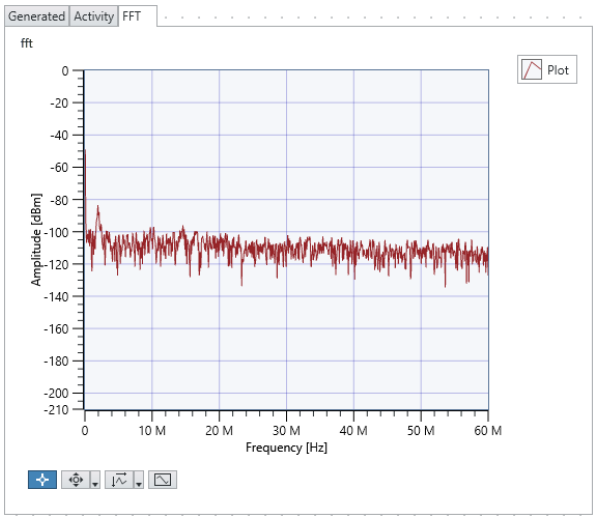


FIGURE 53 – Paquet généré et émis.

La vue de l'activité sur les différents canaux reçu, et non démodulé, est un très bon indicateur des paquets qui sont transmis à quel instant et sur quel canal. Pour cela, comme expliquer dans le projet précédent, un diagramme en waterfall représente l'activité dans le temps, sur quel canal et la puissance du signal. Ce graphique est tiré de la transformer de Fourier du signal brute reçu, cette transformée est visible sur le dernier graphique de cette interface et permet de voir en temps réel le bruit ambiant et les fréquences autres que celles dédiées aux canaux du protocole.



(a) Graphique d'activité sur les canaux.



(b) Transformer de Fourier du signal reçu.

6. Test et validation

Le système étant au terme de son développement il est nécessaire de connaitre ses capacités et ses limites. N'ayant pas connaissance des trames que les appareils de Sonova transmettent il n'est

malheureusement pas possible de faire un essai en condition réel. C'est pour cela que la simulation d'émission a été mise en place.

Le test de fiabilité du système le plus parlant est en fonction de l'amplitude du signal reçu. Il a été mesuré que l'USRP reçoit un bruit de fon d'environ -100 dBm, ce qui sera la base des mesures. Le test se déroule ainsi.

Le gain d'émission est configuré pour qu'en réception le signal ait des puissances de -20, amplitude maximale de -15 dBm de l'USRP, à -80 dBm, sensibilité minimale des appareil de Sonova, par pas de 10. 1000 trames sont envoyé et les données de paquets reçues et sauvegarder le document CSV sont analysé. En voici les résultats.

Amplitude	Detected	Decoded	R_{det}	R_{dec}
-20 dBm	998	836	99.8 %	83.6 %
-25 dBm	1000	986	100 %	98.6 %
-30 dBm	999	904	99.9 %	90.4 %
-40 dBm	1000	281	100 %	28.1 %
-50 dBm	763	21	76.3 %	2.1 %
-60 dBm	4	0	0.4 %	0 %
-70 dBm	0	0	0 %	0 %
-80 dBm	0	0	0 %	0 %

TABLE 3 – Capacité du système selon la puissance.

Pour des puissance de -20 dBm à -30 dBm les résultats sont quasiment parfait, à -40 dBm tout les paquets sont détecté mais seulement 28 % sont décodé. En dessous de -50 dBm plus aucun paquet n'est détecté, le système n'est plus assez performant.

7. Conclusion

Au termes de ce travail, les objectifs demandé ont été atteints. Le systèmes est capable de recevoir les signaux émis par un appareil de Sonova, de les démodulé en temps réel et sur plusieurs canaux en simultané, d'implémenter une synchronisation ainsi que le décodage des paquets afin d'en extraire les informations importantes. L'ensemble de ces éléments ont été valider par le développement conjoint du système d'émission de paquets.

Les connaissances apprises durant ces sept semaines furent très enrichissantes dans le domaine du traitement de signal, les méthodes de modulations et démodulations ainsi que le développement d'applications pour pour PC et FPGA à l'aide de l'outil LabVIEW. Étant été développé depuis zéro et ayant une implémentations réel, la limitations de ressources, tel que sur la FPGA, fu un éléments important à prendre en compte lors de la réalisation du système.

Selon les résultats de la section 6 le projet n'est pas utilisable pour les spécification requise par Sonova, mais il reste un grand pas d'amélioration possible au système, ces points sont abordé dans la section 7.2.

7.1. Problèmes rencontrés

L'USRP est un produit haut de gamme. Son fabricant, NI, dans sa méthode marketing ne met pas à disposition suffisamment d'informations à propos du fonctionnement de ses appareils, des spécificités et de l'utilisation de ses systèmes. Pour cela beaucoup de tests ont dû être effectué afin de déterminer le fonctionnement précis de certaine partie du système.

Tel le bloc "Frequency shift", disponible pour la FPGA, n'effectue pas une simple multiplication du signal avec un cosinus afin de les convolués dans les fréquences et de ce fait déplacé en fréquence les informations. Ce bloc effectue une multiplication croisée avec les signaux quadratique qui ainsi évite les harmoniques négatives produites lors d'une convolution en fréquence.

Les erreurs de compilation de la FPGA apparaissent uniquement à la fin du processus, celui-ci qui prends de 15 min à plus de 40 min suivant la complexité. De plus les erreurs sont mal documentées et donc en trouver la source peut se trouver être une tâche très ardue voir impossible. Un problème de version de librairie est apparue en milieu de projet et à rendu dans l'incapacité d'avancer sur le projet pendant une semaine sans trouver de solutions adéquate. La seule option fut de venir sur une version antérieure du projet.

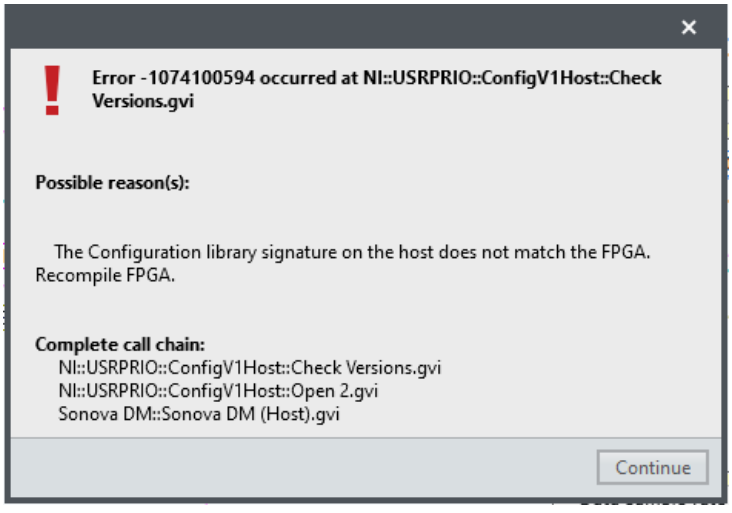


FIGURE 55 – Fenêtre d'erreur de version de librairie

7.2. Améliorations

Ce travail étant une première ébauche au projet de Sonova, beaucoup d'éléments peuvent être améliorés.

Afin de pouvoir décoder l'ensemble des 40 canaux avec un seul appareil plutôt qu'être limité à 10 actuellement , il est déjà possible d'utiliser les 2 canaux RF de l'USRP en mettant de coté la simulation d'envoi d'un paquet ce qui double la capacité du système. De façon optimale , il existe dans la même gamme l'USRP-2945 qui a une plage de fréquence de 80 MHz qui a donc la capacité de recevoir l'ensemble des 40 canaux du protocole et ainsi pouvoir les décoder en simultané en FPGA.

La méthode de décodage des paquets a été développé uniquement sur les connaissances personnelles, il existe donc probablement des méthodes plus efficaces qui prennent moins de ressources et ainsi permettent le décodage de tous les canaux dans un temps réduit sans retarder le système et louper des trames. Développer une application plus performante avec un langage plus adapté et utiliser des systèmes tel que le multithreading seraient une grande amélioration. Ultimement le décodage et la détection d'activité serait à implémenter sur la FPGA au côté de la démodulation dans cet avantage de travaux parallèles.

L'utilisation de l'environnement de développement LabVIEW à des avantages certains lors de l'utilisation des appareils associé , mais son apprentissage est très ardu sans connaissance ou formation ultérieure. Il est possible d'effectuer les mêmes opérations avec les outils de développement standard tel que Vivado de Xilinx pour le programme sur FPGA et ayant comme avantage d'avoir une main prise complète sur les fonctions mise en place et non s'en remettre à celles préfaites et dont on ne connait pas son fonctionnement précis.

Fribourg, le 12 juillet 2024

Bastien Piguet

Références

[1] A. LOPEZ, *Low-Latency GFSK Demodulation Architecture Comparison and Design for FPGA*, juin 2023. adresse : `http://essay.utwente.nl/96205/`.

[2] R. T. HARRINGTON, « Frequency shift keying demodulators for low-power FPGA applications, » Kansas State University, 2017. adresse : `https://core.ac.uk/outputs/77979986/`.

[3] T.-C. LEE et C.-C. CHEN, « A mixed-signal GFSK demodulator for Bluetooth, » *IEEE Transactions on Circuits and Systems II : Express Briefs*, t. 53, n° 3, p. 197-201, 2006, DOI : 10.1109/TCSII.2005.858320.

[4] B. PIGUET, « RF sniffer using USRP, » Haute Ecole d'Ingénierie et d'Architecture de Fribourg, Boulevard de Péroles 80, CH-1700 Fribourg, Rapport de Projet de Semestre 6, avr. 2024.

A. Logiciels utilisés

Nom	Version	Éditeur	Utilisation
TeXstudio	4.8.1	TeXstudio	Éditeur de documents LaTeX
LabVIEW NXG	4.0.0	National Intstruments	Dévellopement d'application PC et FPGA
LanguageTool	6.3	LanguageTooler	Outils de correction orthographique
LibreOffice	24.2.1.2	The Document Foundation	Création de schéma
GanttProject	3.3.33	GanttProject Team	Gestion de planning

B. Annexes

B.1. Démodulation par déplacement de fréquences

B.1.1. Reconstruction du signal

L'USRP travaillant avec des signaux QAM ceci ne correspond pas à l'utilisation actuelle. Pour cela , il faut reconstruire le signal orignal pour ensuite le traiter et effectuer la démodulation dessus. Pour cela , il faut additionner les deux signaux I et Q. Après le déplacement en fréquence à la fréquence de modulation, un signal à 2 fois la fréquence apparaît , mais est filtré par après.

$$1 \cdot \cos(\alpha) = \cos(\alpha) \Rightarrow \cos(\alpha) \cdot \cos(\alpha) = \frac{1}{2}(\underbrace{\cancel{\cos(2\alpha)}}_{\text{filtered}} + \underbrace{\cos(0)}_{1}) = \frac{1}{2}$$

$$\cos(\alpha) \cdot \sin(\alpha) = \frac{1}{2}(\cancel{\sin(2\alpha)} - \underbrace{\sin(0)}_{0}) = 0 \quad \downarrow$$

$$\oplus = 1$$

$$1 \cdot \sin(\alpha) = \sin(\alpha) \Rightarrow \sin(\alpha) \cdot \cos(\alpha) = \frac{1}{2}(\cancel{\sin(2\alpha)} - \underbrace{\sin(0)}_{0}) = 0 \quad \uparrow$$

$$\sin(\alpha) \cdot \sin(\alpha) = \frac{1}{2}(\cancel{\cos(2\alpha)} + \underbrace{\cos(0)}_{1}) = \frac{1}{2}$$

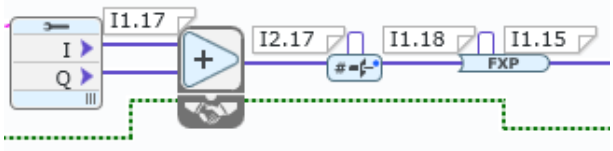


FIGURE 56 – Add I and Q signals

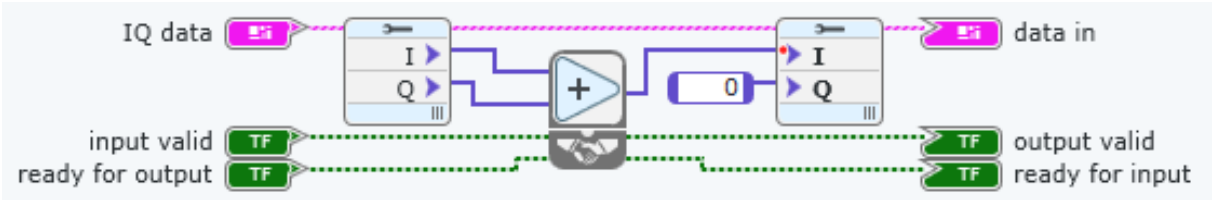


FIGURE 57 – Add I and Q signals

Les notes en dessus des signaux représentent leurs représentations en binaire.

B.1.2. Filtrage du canal

Le signal reçu après avoir été décalé par les signaux I et Q de fréquence de porteuse et décalé afin d'obtenir sur la bande de base les signaux voulu , il faut éliminer les autres fréquences qui crée du bruit sur le signal. Pour cela un filtre passe-bas à une fréquence de 2 MHz est placé.

LabVIEW met à disposition des IPs de Xilinx, dans ceux-ci un filtre FIR est configurable. Les coefficients sont generable par l'outil "Filter designer" de Matlab. Ainsi donc avec cet outil, en appliquant comme paramètre une fréquence d'échantillonnage de 120 MHz, une fréquence de coupure de 2 MHz avec un gain de 1 et une fréquence stop de 3 MHz avec un gain de -40 dB. Un filtre est implémenté pour chacun des signaux I et Q. Il faut faire attention au type de variable d'entrée et de sortie pour ne pas obtenir des valeurs incohérentes.

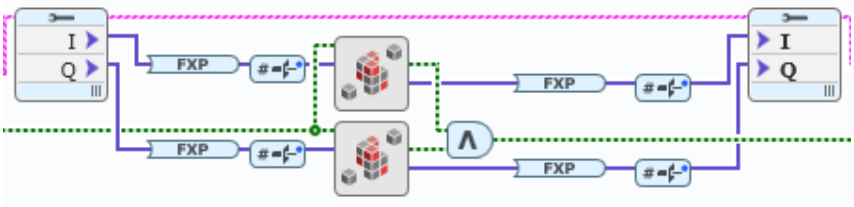
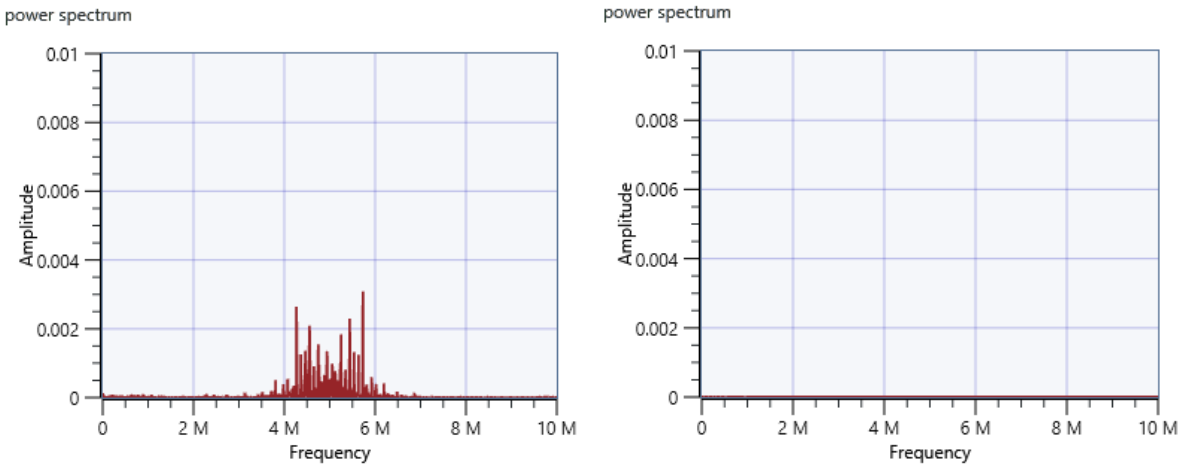


FIGURE 58 – IQ filter FPGA

Dans la figure 59a le signal reçu n'est pas filtré, dans cet exemple l'émission se fait sur une fréquence de 5 MHz, et une puissance non négligeable s'y retrouve. Dans la deuxième figure 59b le filtre est appliqué est on y voit que le bruit précédent a été retiré et ne gène pas la mesure du canal.



(a) Power spectrum 5M **(b)** Power spectrum 5M filtered FPGA

FIGURE 59 – Différence de spectre entre un signal filtré et non filtré

B.1.3. Filtrage en fréquence de données

Après le filtrage du canal quand la démodulation est effectué un signal au double de la fréquence de modulation apparaît, pour ceci le signal reçu est filtré à la moitié de la fréquence de modulation afin de réduire son impact et facilité la restitution des bits. La FPGA dispose d'un filtre FIR, Il est mis en place avec son IP et ses facteurs sont défini avec le filter designer de Matlab.

B.1.4. Déplacement de fréquence

En sortie de la FPGA les données de pour les bit à 1 et les bits à 0 sont récupérés via la FIFO et traité pour en sortir le bitstream, cette opération réalisé sur le PC sera porter sur la FPGA.

Les données récupérées sont mises en forme pour avoir un type de variable "Waveform" qui prend en compte la base de temps, qui est l'inverse de la fréquence d'exécution de la FPGA. Les données étant descendu en bande de base , elles sont filtrées à une fréquence de 160 kHz, qui est la moitié des 320 kHz de la modulation des données en FSK, afin de couper les données déplacé en bande supérieur et tout autres bruits du signal. La méthode est de faire la différence d'amplitude entre les données représentant un bit à 1 et les données représentant un bit à 0. Ainsi l'amplitude la plus élevée indiquera sont bit représentant comme point à un temps t.

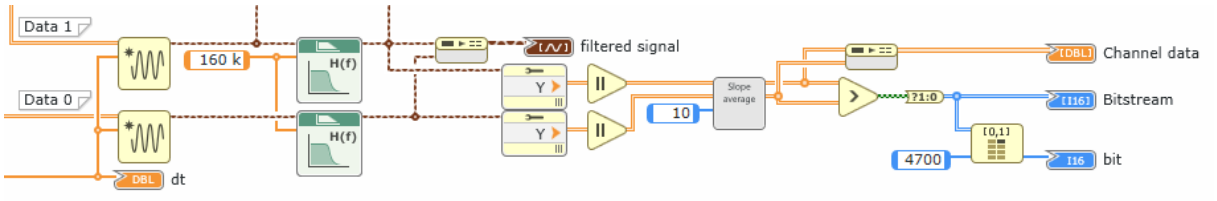


FIGURE 60 – Bitstream generation

Les données reçues sont sur-échantillonnées, connaissant la vitesse d'émission, une moyenne glissante de N valeurs afin de lisser le signal et obtenir une sortie plus lisible.

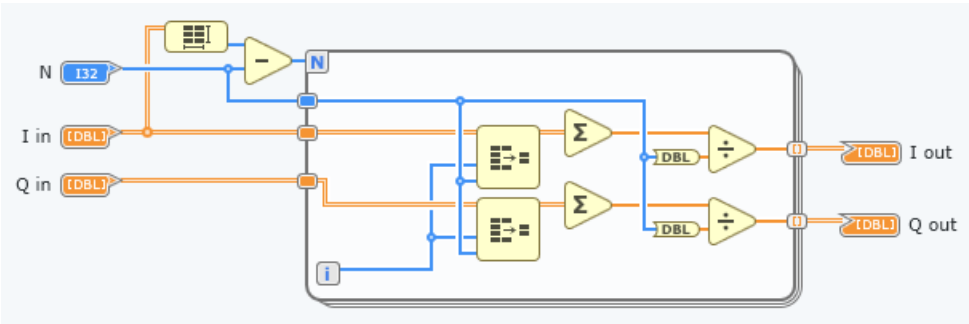


FIGURE 61 – Slope average

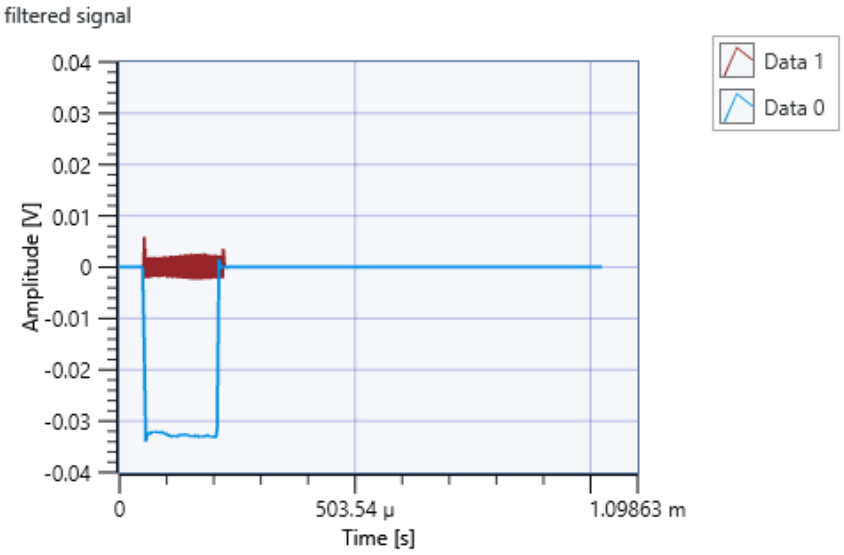


FIGURE 62 – Filtered Data 1 and 0 at 160kHz

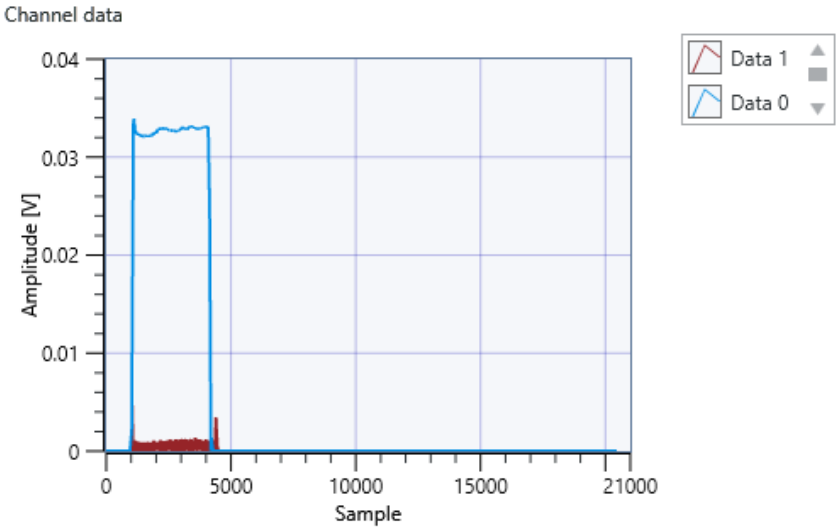


FIGURE 63 – Absolute Data 1 and 0

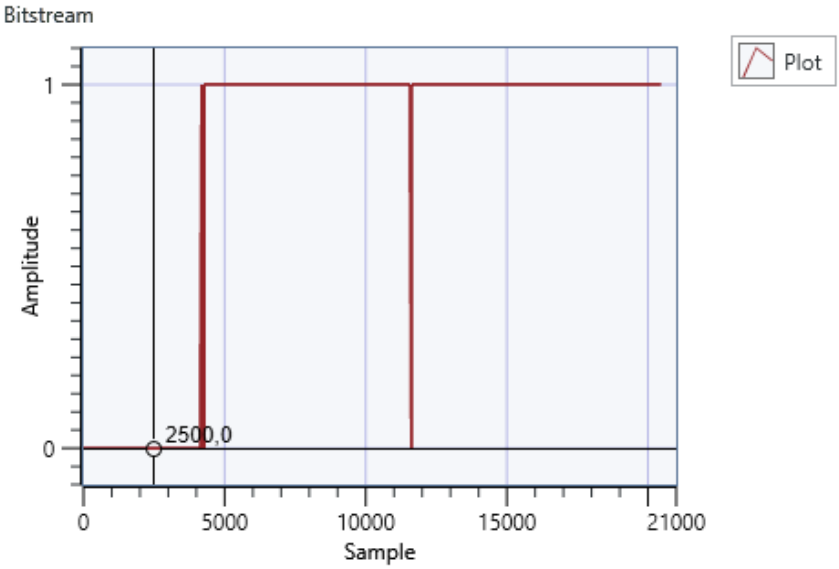


FIGURE 64 – Bitstream

L'émetteur et le récepteur ne sont pas parfaits, cela induit que la fréquence de modulation n'est pas la même et lors de la démodulation le déplacement en fréquence laisse un résidu de cette différence et empêche la restitution correcte des bits. Pour cela , il faut détecter cette fréquence résiduelle et l'appliquer sur le déplacement de fréquence afin d'obtenir une sortie à 0 Hz.

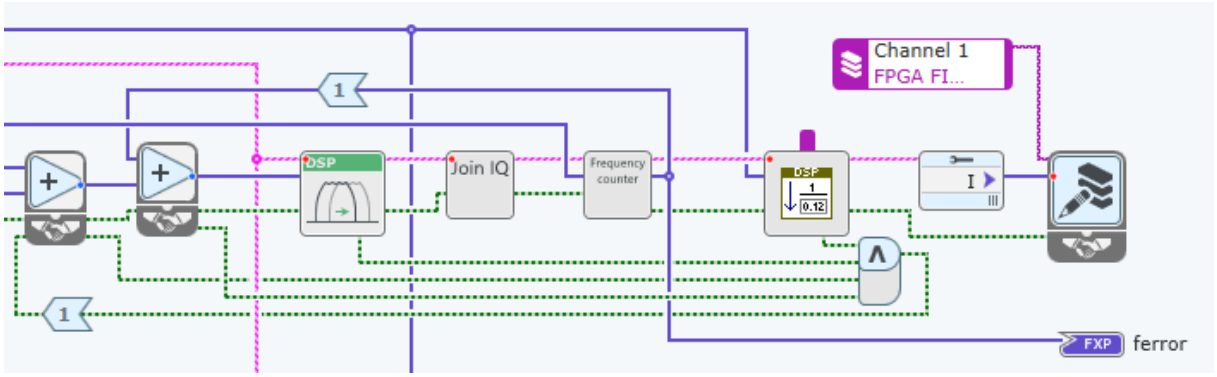


FIGURE 65 – demodulation dual fshift ferror

La fonction qui mesure cette fréquence résiduelle est composé d'une détection de flan et d'un compteur. Le compteur s'incrémente à chaque clock de la FPGA, quand un flan montant est détecté la valeur du compteur est enregistrée, puis remis à zéro, et est utilisée pour calculer la valeur à ajouter au déplacement de fréquence afin de compenser cette erreur. Pour ne pas avoir un déplacement en fréquence trop conséquent lors du démarrage ou en car d'erreur, la valeur est limitée en minimum et maximum. La différence de fréquence pouvant être positive et négative ce compteur ne permet pas d'en déterminé le signe, pour cela si la différence précédente et plus faible que la nouvelle calculée le facteur de déplacement à son signe inversé pour se diriger vers le 0 et non s'en éloigné.

$$f_{clk} = 120\,\text{MHz}$$
$$f_{\Delta max} = 5\,\text{kHz}$$
$$n = \frac{f_{clk}}{f_\Delta} \Rightarrow f_\Delta = \frac{f_{clk}}{n}$$
$$R_{f_\Delta} = \frac{f_\Delta}{f_{clk}} = \frac{\frac{f_{clk}}{n}}{f_{clk}} = \frac{1}{n}$$

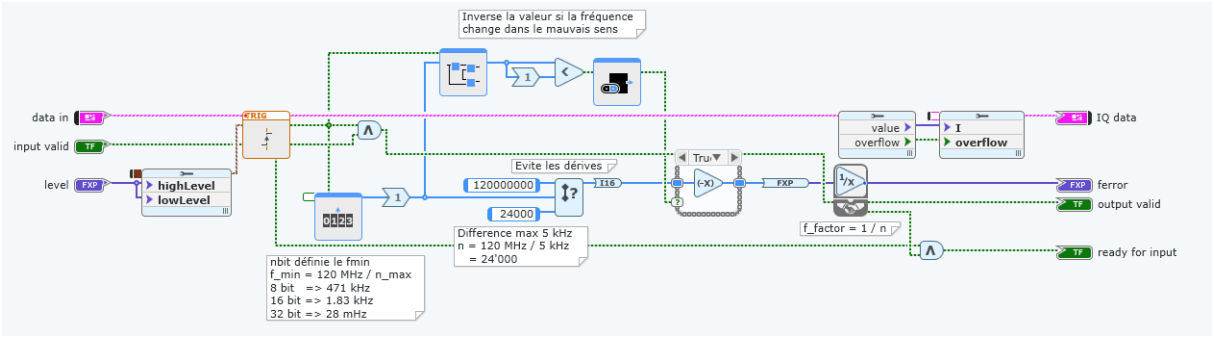


FIGURE 66 – Frequency counter

Cette version à comme problème que la boucle de contre-réaction tant vers l'infini quand la fréquence s'approche de son goal et induit que le circuit devient en boucle ouverte. De ce fait la détection de phase doit donc effectuer le déplacement en fréquence après avoir descendu le signal en une zone facilement mesurable du compteur.

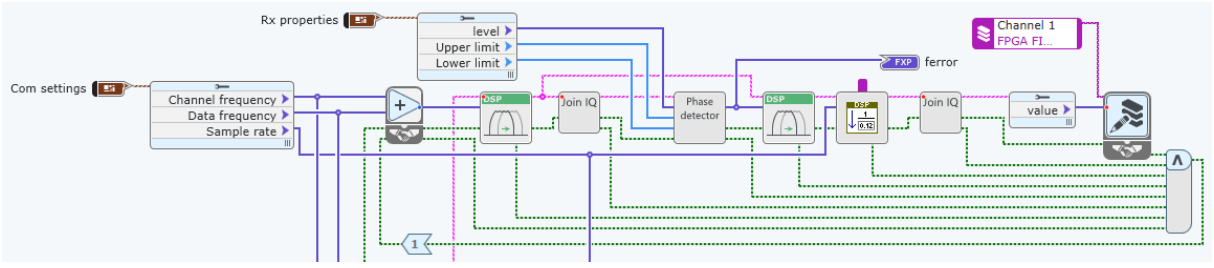


FIGURE 67 – demodulation dual fshift ferror not open

La fréquence d'erreur se trouvant maintenant uniquement dans la bande de 0 à 10 kHz le facteur de déplacement ne requiert plus d'être inversé si la compensation se dirige dans la mauvaise direction.

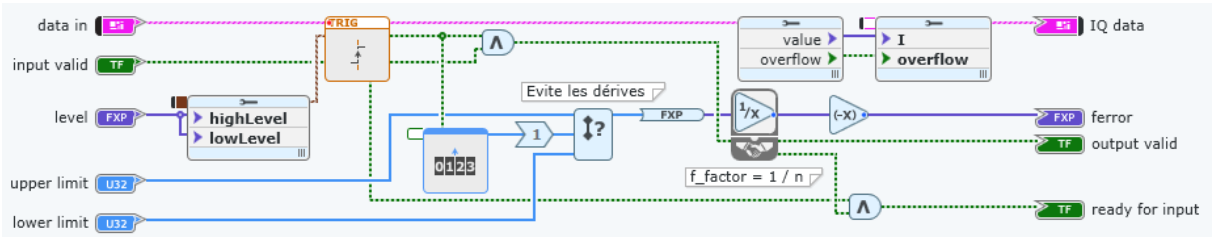


FIGURE 68 – Phase detector

Après de multiple tentative de compilation cette méthode génère des problèmes de timing sur la FPGA et doit donc être abandonné pour une méthode plus légère. Ayant mis en place le compteur la démodulation uniquement par celui-ci, qui est expliqué précédemment, devient une évidence. La sortie de cette démodulation sera la valeur du compteur et qui représente de façon inversement proportionnelle la fréquence que le signal contient.

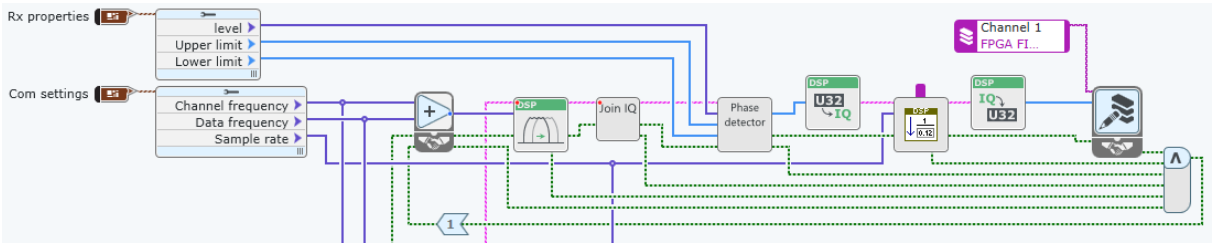


FIGURE 69 – demodulation dual fshift counter

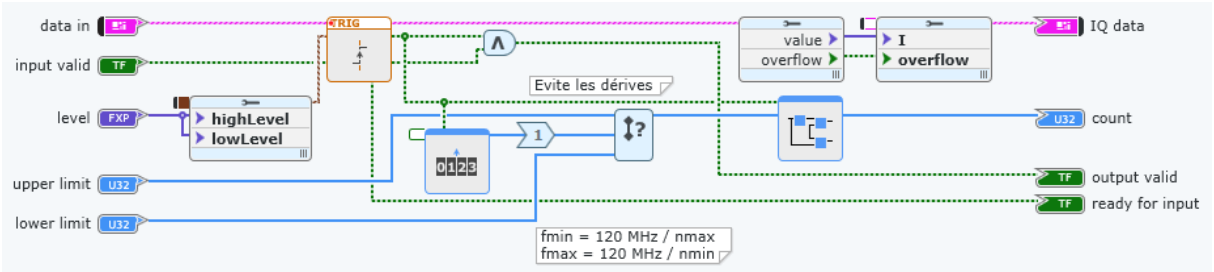


FIGURE 70 – Phase detector only counter

Méthode trop contraignante à cause du timing de la FPGA et de la vitesse de transmission des données. Une fréquence d'échantillonnage plus élevé serait nécessaire

B.1.5. DDS compiler

Le bloc "Frequency shift" n'étant pas adapté à la démodulation en fréquence, la génération d'un signal sinusoidal à la fréquence de démodulation est nécessaire. Pour cela un Direct Digital Synthesizer (DDS) est implémenté, celui générera sur sa sortie "m_axis_data_tdata" les informations d'un sinus et d'un cosinus qui seront à une fréquence déterminée par l'entrée "s_axis_phase_tdata" donc la valeur est déterminée par cette formule.

$$f_{out} = \frac{f_{clk} \cdot \Delta\Theta}{2^{B_{\Theta(n)}}} \tag{28}$$

B.2. Spécification du protocole

L'entreprise Sonova travail avec 4 protocoles de communications différents, Le Bluetooth standard, le Bluetooth low energy, le DM et le FLORA un protocole propriétaire. Le choix de celui qui est

étudié pour ce projet a été défini par les connaissances apprises durant les cours et la quantité de documentation disponible.

Le choix s'est donc porté sur le DM et voici la documentation technique de référence donnée par Sonova pour ce protocole.

Frequency band	2400 - 2483.5 MHz (2.4 GHz ISM Band)
Channels	40 channels1 with a channel spacing of 2 MHz Ch1 : 2402 MHz, Ch2 : 2404 MHz, . . ., Ch40 : 2480 MHz
Modulation	GFSK
Modulation index	0.32 (+/- 320 kHz)
Bandwidth-bit period product (BT)	0.5
Raw bit rate	2 Mbps
Spread spectrum technology	Adaptive Frequency Hopping (AFH) (max. 40 channels, min. 20 channels)
RF output power EIRP (typ. max.)	+20 dBm

Ces informations sont basées sur le chip nRF24L01P de chez Nordic Semiconductor. C'est un transmitter 2.4 GHz qui intègre un protocole appelé le "Enhanced ShockBurst™" qui défini la structure que les données transmises auront et qui seront utilisées dans le prochain travail lors de la partie de décodage des données.

B.2.1. Bande de fréquences

La communication se fait au travers de la bande ISM de 2.4 Ghz. Dans celle-ci se trouve 40 canaux espacés de 2 MHz, partant de 2402 MHz jusqu'à 2480 MHz. Ce sont les fréquences porteuses du signal. Cette bande est la même utilisée par le Bluetooth et les autres protocoles de Sonova précédemment cité, ceci permet au produit de n'avoir qu'un seul type d'antenne et d'émetteur et ainsi simplifie la miniaturisation des circuits.

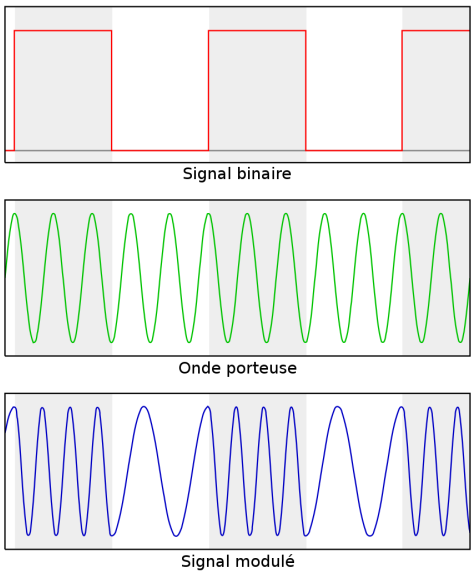
B.2.2. Spread spectrum

Le spread spectrum est une méthode d'étalement du spectre qui permet d'assurer une communication plus fiable entre plusieurs appareils en évitant les collisions de paquets si plusieurs appareils discutent sur les mêmes fréquences en modifiant sa fréquence d'émission dans un autre canal et selon un certain type de multiplexage.

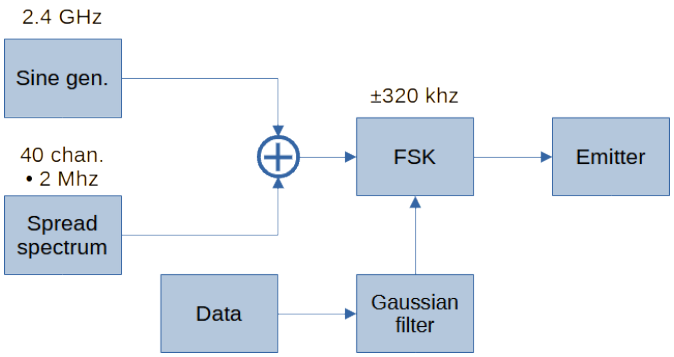
Le DM utilise le type de saut de fréquence adaptatif (Adaptive Frequency Hopping) qui est un type d'étalement de spectre qui va sélectionner la bande de fréquence la moins utilisée afin d'assurer une perte de données minimale lors de la communication entre les appareils.

B.2.3. Modulation

Le type de modulation GFSK (Gaussian Frequency Shift Keying) représente une modification de la fréquence du signal selon la donnée mise en entrée. L'index de modulation est la valeur dont la fréquence est modifiée, dans ce cas la différence est de plus ou moins 320 kHz de la porteuse si la donnée est de valeur 1 ou 0.



(a) Représentation d'un signal modulé en FSK



(b) Traitement et modulation du signal pour l'émission

La spécificité du type Gaussien permet de réduire le spectre de fréquence du signal émis en limitant les changements abrupte de niveau des données et donc de fréquence du signal. Il est effectué en appliquant un filtre Gaussien sur le signal à émettre avant sa modulation. Le filtre est construit suivant la formule 29. Le paramètre BT, spécifié dans la documentation du protocole, représente la largeur sur laquelle la gaussienne s'appliquera sur le signal, une valeur de 1/2 équivaut à une longueur de deux symboles. L'influence de BT sur la gaussienne est visible sur la figure 72 qui donne un exemple pour les valeurs 1, 1/2, 1/3, 1/5 et 1/10.

$$BT = 0.5 \tag{29}$$

$$a = \frac{\pi}{\ln(\sqrt{2})} \cdot BT^2 \tag{30}$$

$$g(x) = \sqrt{\frac{a}{\pi}} \cdot e^{-ax^2} \tag{31}$$

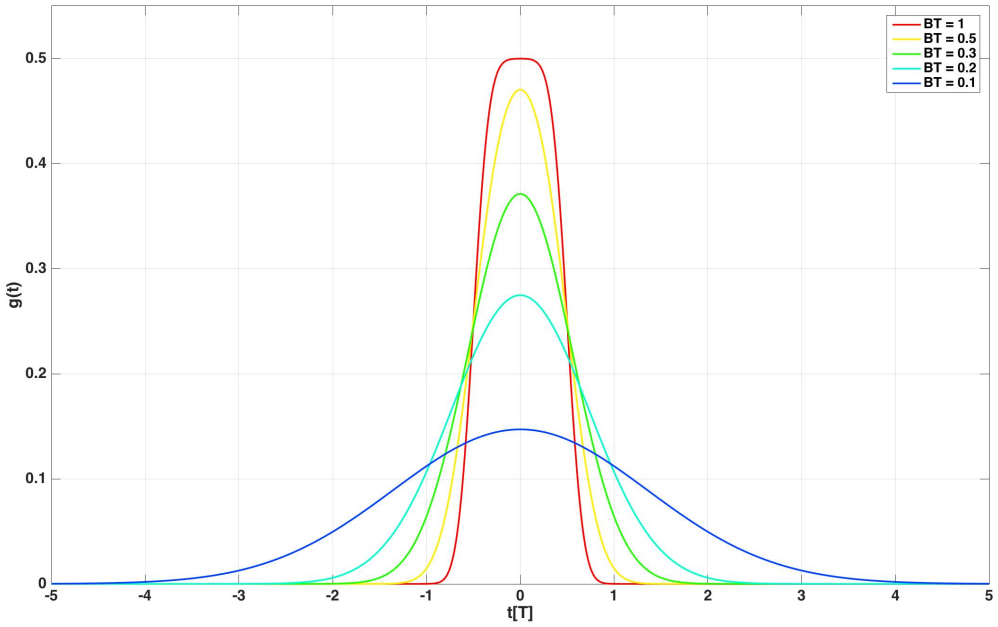


FIGURE 72 – Plot de l'influence de BT sur une gaussienne

B.2.4. Démodulation

Le principe de démodulation est celui d'une PLL, à l'entrée le signal à démoduler passe dans un comparateur de phase qui ressort une tension proportionnelle à la différence de phase, qui correspond aussi à la différence de fréquence, avec la seconde entrée qui dans ce cas est la porteuse du signal que l'on veut démoduler. La contre-réaction passant dans un filtre passe bas qui enlève les fréquences parasites supérieur à celle de référence.

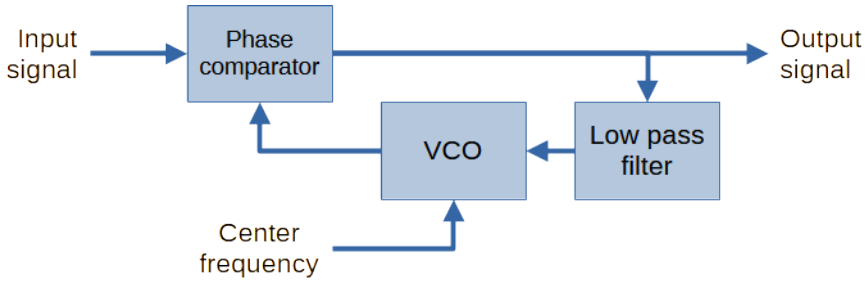


FIGURE 73 – Démodulation FSK avec système de PLL

B.2.5. Synchronisation

Les systèmes d'émission et de réception n'étant pas sur la même appareil les signaux transmis doivent être synchronisé afin de décoder les informations correctement. Pour cela le protocole du nRF24L01P dispose d'un byte de préambule qui est soit 10101010, soit 01010101. Si le premier bit reçu est un 0 le code sera donc 01010101 et s'il est un 1 alors cela sera 10101010.

B.3. Spécification de l'USRP

Dans la description du projet le hardware et le software n'ont pas été définis, seuls des propositions ont étés émis. De ce fait ce choix dépendra des connaissances et de la disponibilité courante des appareils dans le cadre du projet.

B.3.1. Hardware

Pour la Radio fréquence, il existe un modèle amateur grand public nommé RTL-SDR qui fut la première idée de matériel à utilisé pour sa communauté très active et ces outils faciles d'accès, mais cependant sa plage de fréquence ne va pas au-delà de 1.75 GHz et n'est donc pas utilisable pour ce projet. Dans l'école, il y a à disposition des USRP-2953R de National Instrument qui sont utilisés pour des cours de télécommunication sur le WiFi, ce qui est dans la même bande de fréquence que l'utilisation voulue. Le choix s'est alors porté sur cet USRP qui a les capacités nécessaires au développement de ce projet.

B.3.2. Spécification

L'USRP-2953R est un appareil de réception, émission, modulation et démodulation de signaux. Il possède une plage de fréquence de 1.2 GHz à 6 GHz réglable avec un pas minimum de 1 kHz. Chacun des 2 canaux disponibles permettent un échantillonnage de 120 MHz en émission et en réception.

On-board RAM	1'024 MB
Transmitter	
Number of channels	2
Frequency range	1.2 GHz to 6 GHz
Frequency step	<1 kHz
Maximum output power	17 dBm to 20 dBm
Maximum I/Q sample rate	120 MS/s
Resolution	16 bit
Reciever	
Number of channels	2
Frequency range	1.2 GHz to 6 GHz
Frequency step	<1 kHz
Maximum input power	-15 dBm
Maximum I/Q sample rate	120 MS/s
Resolution	14 bit

La structure interne de l'USRP, tel visible en figure 74, consiste en une partie FPGA qui effectue le traitement du signal à une vitesse de 120 MHz. Chaque donnée transmise depuis le PC qu'importe le bitrate sera interpolé afin d'atteindre la fréquence de traitement du système. De façon inverse les données reçues sont décimés pour descendre au bitrate désiré en sortie de l'appareil. En entrée comme en sortie les signaux analogiques sont multipliés par un sinus et un cosinus dans le but de créer une modulation QPSK et passe dans un filtre passe bas de 20 MHz qui supprime les harmoniques de réception ou de le DAC selon la ligne de traitement.

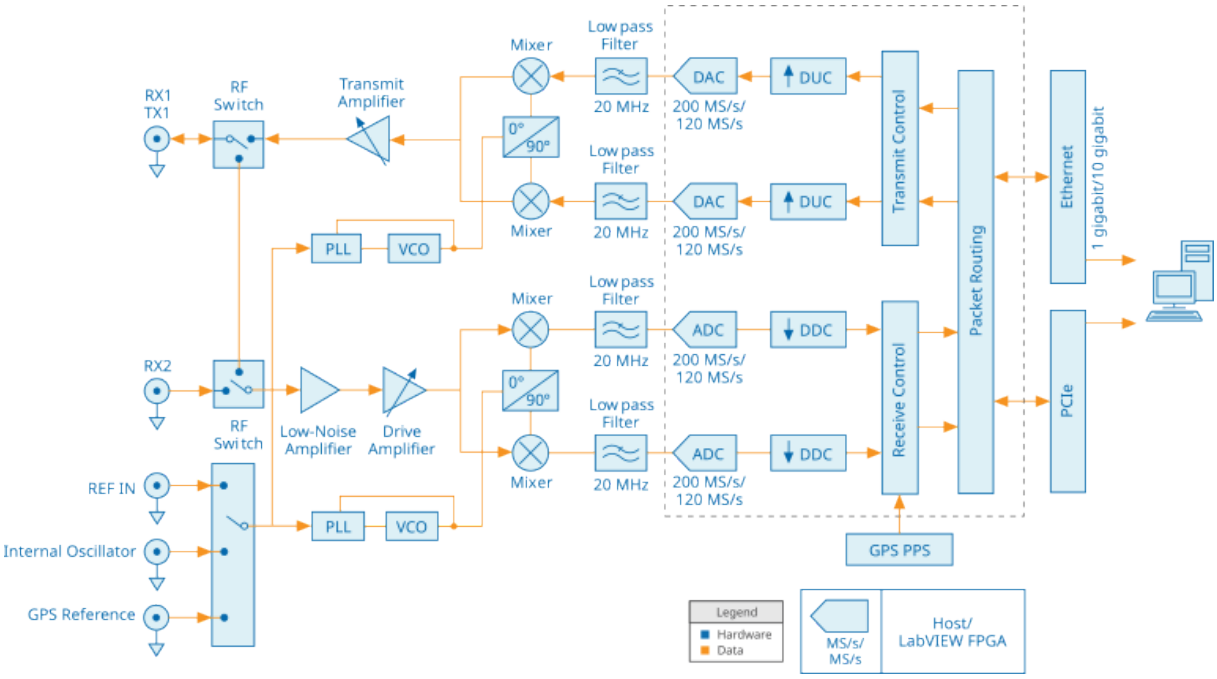


FIGURE 74 – Schéma bloc hardware interne d'un canal de l'USRP

B.3.3. Software

Côté software, il existe de même un programme open-source nommé GNU radio qui permet le développement de programme de traitement de signal et de donnée. Des librairies sont disponibles afin de l'utiliser avec l'USRP, cependant il n'est pas possible de programmer la FPGA de celui-ci. Pour cela, il faut utiliser le programme LabVIEW du même constructeur. Sa méthode de programmation est similaire, en utilisant des blocs et non en ligne de code.

B.4. Fonctionnement du système

Tel expliqué dans la section **??** le système est composé de deux parties, une hardware et une software. La communication entre l'USRP et le PC hôte se fait avec une carte PCIe 4x installé au préalable sur l'ordinateur ainsi que ces drivers fourni avec le programme de développement LabVIEW NXG.

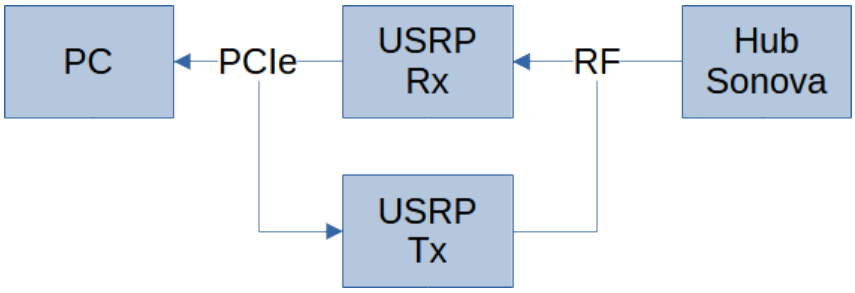


FIGURE 75 – Schéma de connection du système

B.4.1. Transmission du signal et des données

La communication se fait au travers du port PCIe 4x, celui permet d'interfacer les mémoires FIFO configurable, en taille et type de données, de l'USRP de l'hôte vers la cible et inversement, ainsi que la configuration de l'appareil. Les FIFO sont modifiées par le biais d'une DMA.

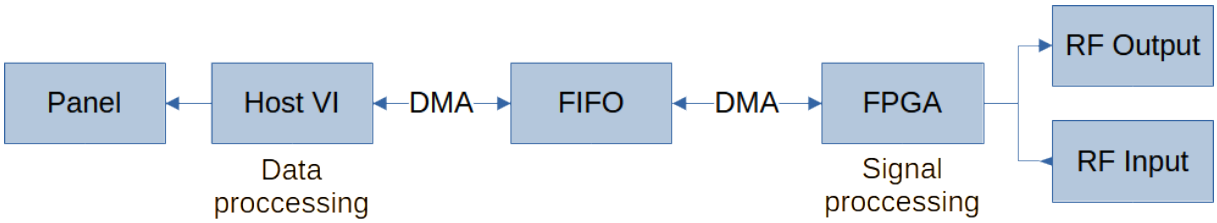


FIGURE 76 – Schéma block de la transmission des données

B.4.2. Traitement du signal (FPGA)

La partie de la FPGA, qui est donc programmable, permet le traitement du signal à une vitesse constante et à la capacité de paralléliser les calculs tel que les informations I, en phase, et Q, déphasé, d'une modulation QPSK et de les transmettre de façon synchrone au PC.

B.4.3. Traitement des données (Host VI)

Le programme se trouvant sur le PC, qui tourne à une vitesse plus rapide que la FPGA, fait le décodage des données reçues ainsi que d'autres calculs tels que la puissance du signal et l'affichage des données sur un graphique ou tout autre besoin. Il s'y trouve aussi la configuration de l'USRP qui lui est envoyé au démarrage du programme.