Etudiants

Delsinne Anthony
Quinet Loic
Vitulli Michael

Professeurs

Jojczyk Laurent
Triquet Fabrice

Année académique 2016-2017

Table des matières

1 Introduction

Beaucoup d'application embarquées, lowcost ou avec des contraintes temps réelles extrêmement exigeante ne peuvent pas se permettre de réaliser des tâches de traitement du signal en utilisant un ordinateur.

L'objectif de ce projet est la mise en place d'une architecture flexible permettant de réaliser des opérations de traitement du signal sur des signaux analogiques. L'architecture mise en place se compose de 2 modules différents : une carte d'acquisition « low noise » déjà préconçue et un module de calcul FPGA qui réalisera la partie signal processing qui pourra être dimensionnée en fonction des besoins.

L'objectif principal du projet est donc de réaliser une chaine de communication entre ces différents modules, permettant l'acquisition d'un signal sonore pour des entrées audio génériques.Plusieurs modules sont mis à disposition du projet :

- Un module codec d'acquisition de signaux sonore déjà conçu
- Un module de transmission de données pour un futur traitement

Le module d'acquisition sera la base du projet et permettra après analyses et mesures de ses performances de dimensionner un nouveau module d'acquisition permettant un interfaçage plus rapide avec un module de contrôle. L'interfaçage de contrôle sera dans un premier temps réalisé à l'aide d'un microcontrôleur utilisant des boutons poussoirs. L'interfaçage sera par la suite modifié et amélioré pour permettre un contrôle direct à l'aide d'un pc.

Le module de transmission de données sera remanié en version plus performante et servira ainsi de base pour le futur traitement des données au sein du module de calcul. Le module de transmission sera développé sur FPGA et permettra d'interagir avec plusieurs modules d'acquisition.

Ce rapport se divise en plusieurs parties :

- La description des différents objectifs de travail
- L'explication des différents Work Package abordés lors des travaux de cette année
- Une conclusion permettant de synthétiser les différents travaux avec un comparatif des objectifs initiaux. La conclusion permettra également de mettre en avant le cahier des charges pour la bonne continuité du projet.

2 Définition des objectifs de travail

Les objectifs de travail du projet DSPOB sont définis à l'aide des différents Work Package ou WP ci-dessous :

Dénomination	Description
DSPOB_01	Etude du codec Wolfson. Prise en main du kit de développement Mikroelectronika
DSPOB_02	Recherche et analyse d'un nouveau kit de développement et comparaison avec le kit précédent
DSPOB_03	Modification du kit de développement afin d'utiliser des entrées et sorties plus génériques que celles proposées sur la carte
DSPOB_04	Modification du code sur µ-contrôleur ATMEGA pour le contrôle du codec à partir d'un PC
DSPOB_05	Tests et analyses des performances du kit modifié
DSPOB_06	Implémentation du traitement numérique sur FPGA ou sur µ-contrôleur. Choix du système à utiliser pour le traitement
DSPOB_07	Réalisation d'un PCB sur base des résultats obtenus après les tests du kit modifié. Le PCB comprendra une modification des entrées pour des entrées génériques et une compatibilité des connectiques du kit pour l'introduire facilement sur une carte FPGA ou µ-contrôleur
DSPOB_08	Rédaction du rapport final
DSPOB_09	Analyse du shieldArduino
DSPOB_10	Analyse du codec Wolfson de la carte Altera DE2
DSPOB_11	Prise en main et analyse de l'architecture FPGA implémentée

La conclusion en fin de rapport permettra de mettre en évidence les différents points ayant été réalisés dans leur ensemble.

3 Analyse du codec Wolfson

Cette partie permet de mettre en avant les différentes analyses du module d'acquisition. Le module d'acquisition utilisé pour le projet est le codec Wolfson WM8731. L'analyse du module s'est faite en deux temps :

- Une analyse du codec présent sur la carte de développement Mikorelectronika
- Une analyse du codec présent sur le kit de développement FPGA DE2-70

Les architectures analogiques présentent sur ces deux cartes de développement sont totalement différentes. Ainsi, des analyses poussées suivies de mesures permettront d'identifier les éléments importants sur chacune des cartes. Ces analyses et mesures permettront de dimensionner au mieux une nouvelle carte de développement.

Sur cette nouvelle carte viendront s'implémenter le codec Wolfson WM8731 ainsi qu'une toute nouvelle architecture analogique comprenant des filtres analogiques pour les entrées et sorties du codec mais aussi une interface de contrôle pour le codec.

3.1 Analyse des entrées du codec Wolfson WM8731

L'objectif a été de définir si le passage par les entrées « LINE IN » améliorerait la qualité du son reçu en sortie du codec audio. Afin de pouvoir comparer des situations semblables, des paramètres ont été fixés.

Pour définir si le changement d'entrée a une répercussion sur la qualité du signal, la campagne de test peut s'effectuer soit en mode « ANALOGIQUE » ou en mode « DIGITAL ». Les tests réalisés et présentés dans ce document ont uniquement été réalisés en mode « ANALOGIQUE ».

Les tests consistent ont une mesure à l'analyseur de spectre du signal en sortie de la carte MikroElektronika. La manipulation réside en un balayage en fréquence jusqu'à l'atteinte de 24 kHz. Cette valeur maximale est fixée car l'exercice peut être reproduit en choisissant le mode « DIGITAL ». Ayant utilisé une fréquence d'échantillonnage de 48 kHz, et en tenant compte du **théorème de Shannon**, la fréquence à ne pas atteindreet dépasser en mode « DIGITAL » est de 24 kHz.

Théorème de Shannon :

« La représentation discrète d'un signal exige des échantillons régulièrement espacés à une fréquence d'échantillonnage supérieure au double de la fréquence maximale présente dans ce signal. »

En mode « ANALOGIQUE », la mesure à 24 kHz a été prise pour vérifier le comportement de la carte lorsqu'elle dépasse la gamme de fréquence audible par l'homme (20 Hz – 20 kHz).

➢ Test A : Codec non-modifié, entrée MIC IN utilisée

Dans un premier temps, une observation sur le passage d'un son au travers du module a été faite. Les premières constatations montrent un son sans parasite mais avec un filtrage des basses.

Conditions de fonctionnement	
Entrée	MICIN
Filtre	Composants de base

Tableau de mesures A						
Mesure	**Trace Average**	**Tension IN [Vpp]**	**Fréquence IN [Hz]**	**Puissance OUT [W]**	**Puissance bruit [W]**	**SNR [dB]**
1	10	1	2.52 k	78 µ	203 p	55.85
2	10	1	5.04 k	1.68 m	831 p	63.06
3	10	1	9.96 k	10.6 m	56.7 p	82.72
4	10	1	15 k	21.4 m	246 p	79.39
5	10	1	20.04 k	13.1 m	62.6 p	83.21
6	10	1	24 k	4.03 n	13.2 p	24.85

➢ Test B : Codec modifié sans composant en entrée, entrée LINE IN utilisée

Les premières observations faites sur l'écoute d'un son passant au travers du module montrent la présence de beaucoup de parasites. Le son est très bruité et de mauvaise qualité.

Conditions de fonctionnement	
Entrée	LINEIN
Filtre	Aucun

Tableau de mesures B						
Mesure	**Trace Average**	**Tension IN [Vpp]**	**Fréquence IN [Hz]**	**Puissance OUT [W]**	**Puissance bruit [W]**	**SNR [dB]**
1	10	1	2.52 k	2.41 n	125 p	12.85
2	10	1	5.04 k	6.81 n	80.3 p	19.28
3	10	1	9.96 k	6.54 n	33.1 p	22.96
4	10	1	15 k	7.19 n	48.5 p	21.71
5	10	1	20.04 k	10.8 n	53.6 p	23.04
6	10	1	24 k	12 n	34.7 p	25.39

La prise de mesure permet de voir que la qualité du signal en sortie a été plus altérée en modifiant le codec. La valeur du SNR a considérablement baissé. La puissance utile du signal a énormément baissée pour se rapprocher des valeurs du bruit.

➢ Test C : Codec modifié avec filtre externe, entrée MIC IN utilisée

Le test suivant a comme objectif de montrer l'importance d'un filtre en entrée du codec. Le filtre a été installé en externe de la plaquette de développement du codec Wolfson 8731.

Les premières observations faites sur l'écoute d'un son ayant passé au travers du module montrent la présence des basses. Le son est de bonne qualité.

Conditions de fonctionnement	
Entrée	MICIN
R [Ohm]	120
C [Farad]	33 µ

Le choix du filtre a été fait de telle sorte à obtenir une fréquence basse du filtre passe-bande la plus proche de zéro. La fréquence de coupure est à environ 0.5Hz. Ainsi, les basses fréquences sonores en entrée auront la possibilité d'être entendue en sortie.

La modification de l'étage d'entrée avec l'insertion d'un filtre engendre également une modification de la dynamique d'entrée du codec. Ainsi, les signaux en entrée du codec seront des signaux possédant une plus faible amplitude.

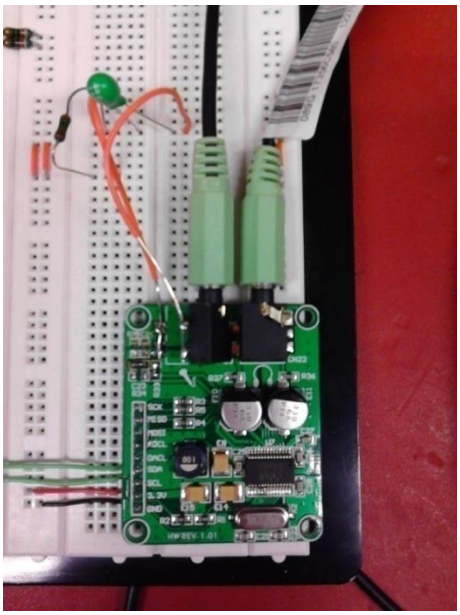


Figure 1: Filtre en externe du kit de développement Mikroelectronika

Tableau de mesures C						
Mesure	**Trace Average**	**Tension IN [Vpp]**	**Fréquence IN [Hz]**	**Puissance OUT [W]**	**Puissance bruit [W]**	**SNR [dB]**
1	5	600 m	2.56 k	1.14 m	160 p	68.53
2	5	600 m	4.96 k	4.82 m	122 p	75.97
3	5	600 m	10 k	8.1 m	187 p	76.37
4	5	600 m	15.04 k	15.6 m	386 p	76.07
5	5	600 m	20.08 k	18 m	195 p	79.65
6	5	600 m	24.04 k	16.4 m	38.3 p	86.32

➢ Test D : Codec modifié avec filtre externe, entrée LINE IN utilisé

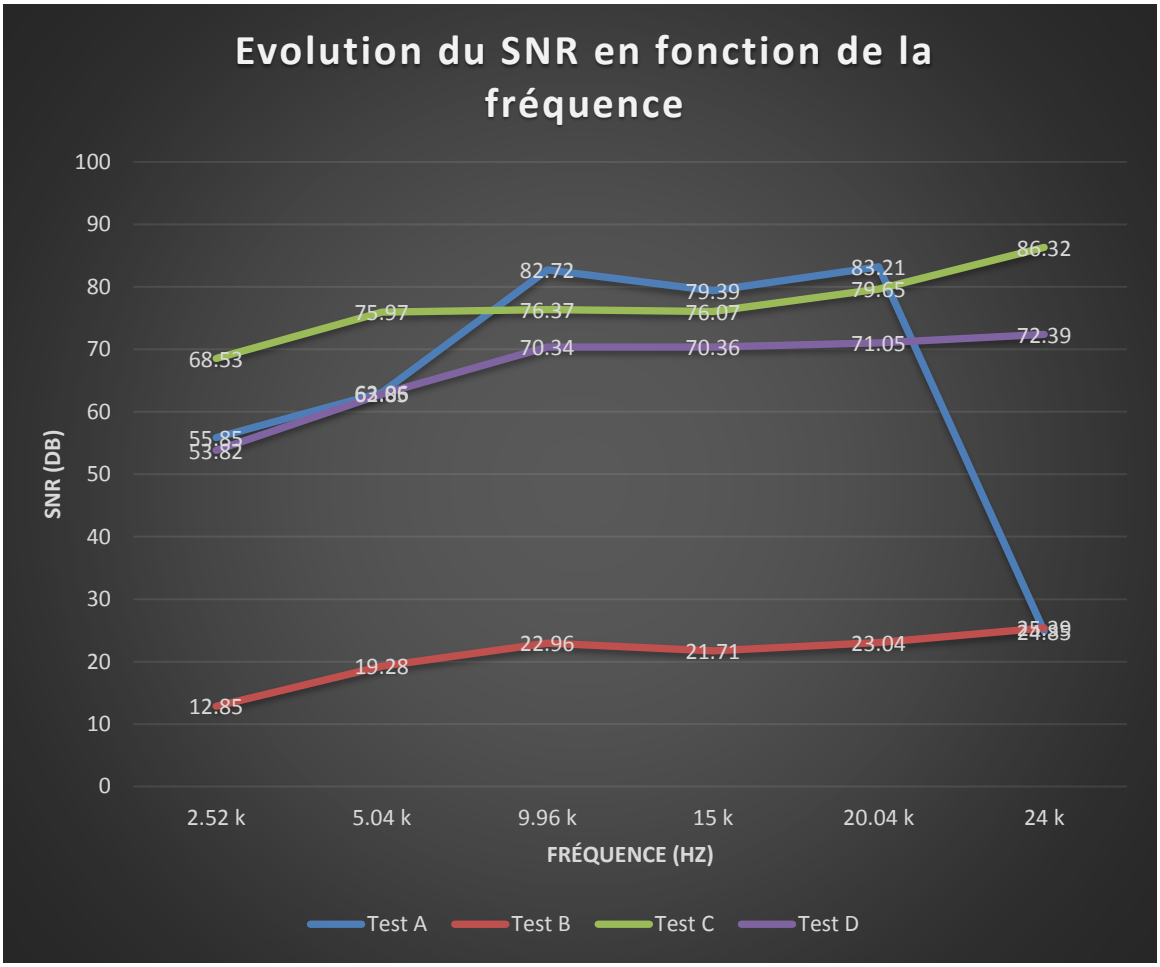
Conditions de fonctionnement	
Entrée	LINEIN
R [Ohm]	120
C [Farad]	33 µ

Ces dernières mesures représentent le codec avec l'entrée LINEIN active. Le filtre utilisé a permis de mettre en exergue l'importance d'un filtre.

Les premières observations faites sur l'écoute d'un son ayant passé au travers du module montrent la présence des basses. Le son est de bonne qualité.

Tableau de mesures D						
Mesure	**Trace Average**	**Tension IN [Vpp]**	**Fréquence IN [Hz]**	**Puissance OUT [W]**	**Puissance bruit [W]**	**SNR [dB]**
1	5	600 m	2.56 k	56.2 µ	233 p	53.82
2	5	600 m	4.96 k	174 µ	90.2 p	62.85
3	5	600 m	10 k	494 µ	45.7 p	70.34
4	5	600 m	15.04 k	641 µ	59 p	70.36
5	5	600 m	20.08 k	613 µ	48.1 p	71.05
6	5	600 m	24.04 k	608 µ	35.1 p	72.39

➢ **Synthèse des résultats expérimentaux**



➢ Critiques des résultats expérimentaux

- *Blocage de la composante DC du signal*

Les résultats proposés montrent qu'il est obligatoire de placer des composants en entrée afin d'avoir un blocage de la composante DC du signal entrant.

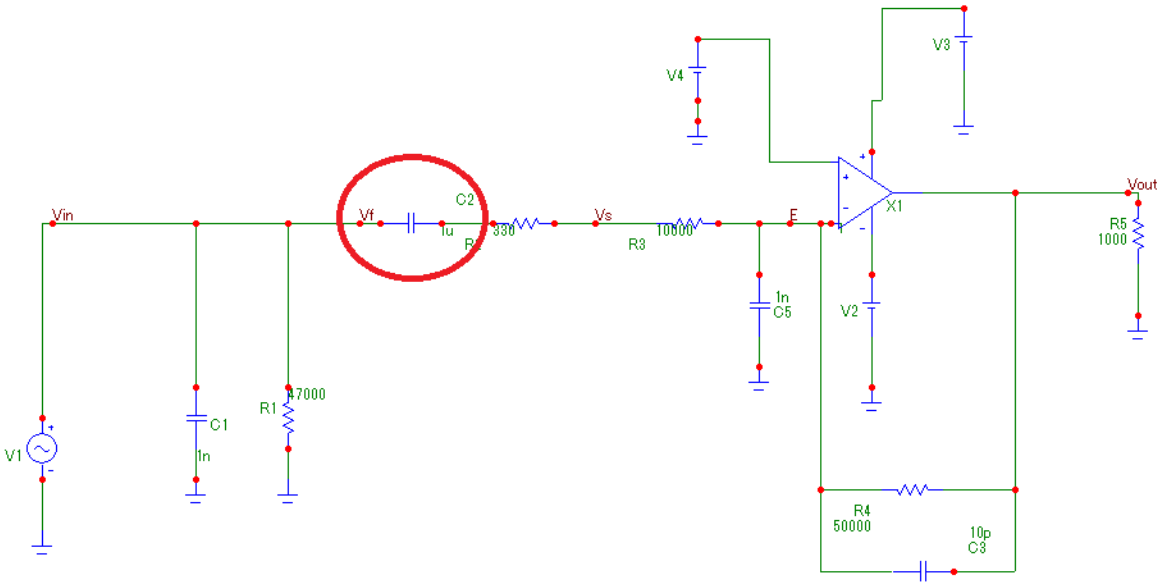


Figure 2: Filtre interne du codec

Le condensateur C2 n'a pas un impact dans la composition du filtre passe bande. Ce condensateur est là pour agir en tant que condensateur de liaison. C2 va bloquer la composante DC du signal d'entrée. Expliquons le comportement du circuit. L'AOP est en contreréaction négative. V4 représente la valeur de VMID (VCC/2). Avec le cas présent, V+ = V-.

Avec le condensateur C2, seule la composante alternative du signal va passer. L'entrée V- de l'AOP voit donc la composante alternative du signal d'entrée additionnée à la tension VMID. L'amplitude d'un signal audio oscille entre +0.45 et -0.45 volt. VMID étant égale à 1.65 volt, la tension résultante se voit être dans la fourchette des 1.2 volt et 2.1 volts.

Ce signal se trouve dans la plage de fonctionnement de l'AOP (0-3.3 volts).

- *Modification de la dynamique d'entrée*

Le choix des composants influencent l'amplitude des signaux à appliquer en entrée. Plus la fréquence de coupure basse se rapproche de zéro, plus l'amplitude maximum à appliquer en entrée sera basse. La conséquence de l'application d'une amplitude trop élevée se répercutera sur le THD.

Sur l'analyseur de spectre, l'observation d'un grand nombre d'harmoniques a été faite dans le cas où le signal d'entrée est trop grand. Ce phénomène a aussi pu être observé par une saturation du signal de sortie.Voici la forme des signaux de sortie appliqués au codec avec le filtre apposé en externe de la plaquette de développement.

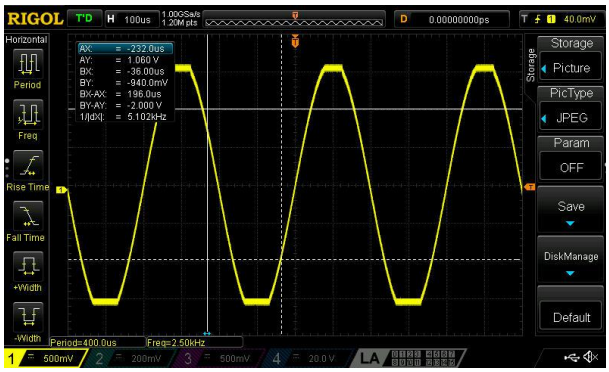


Figure 3: Signal 700mV pic à pic

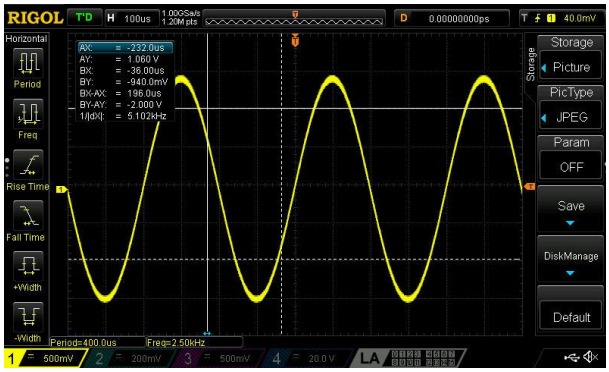


Figure 4: Signal 600mV pic à pic

Le seuil de non-saturation des signaux en entrée du codec se situe aux environs de 600mV pic à pic.

- *Capacités du codec*

Une modification de l'étage d'entrée du codec permet une amélioration des capacités du codec. En effet, la modification des composants extérieurs au codec abaisse la fréquence de coupure du filtre passe haut composant le filtre passe bande. Des fréquences plus basses ne sont plus filtrées.

La qualité du son s'en voit améliorée. D'un point de vue résultat, des signaux sinusoïdaux ne sont plus atténués. Le SNR est supérieur lors de l'expérience C à 2.56 kHz comparé à l'expérience A à 2.56 kHz. Les figures 13 et 15 permettent de montrer le filtrage des fréquences.

Est-il préférable de passer par l'entrée LINE IN ou MICIN ? Une fois modifiée, l'entrée MICIN présente de meilleurs résultats, soit une présence de moins de bruit. Cependant, on ne peut affirmer que l'entrée LINE IN est « mauvaise ». Les deux sortes d'entrée présente des résultats très correcte en termes de SNR. L'observation des courbes du test C et du test D illustre ce propos.

3.2 Analyse d'autres kit comportant le codec WM8731

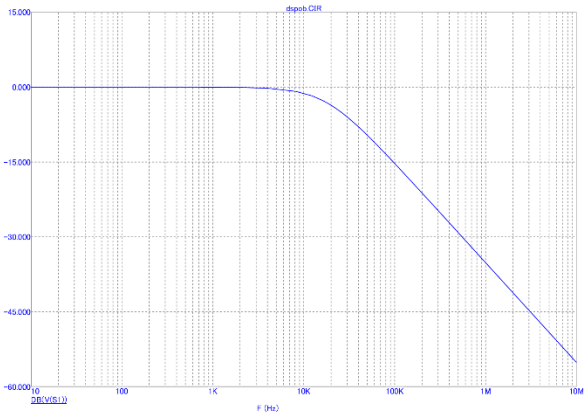


Figure 5: Filtre RC Line In

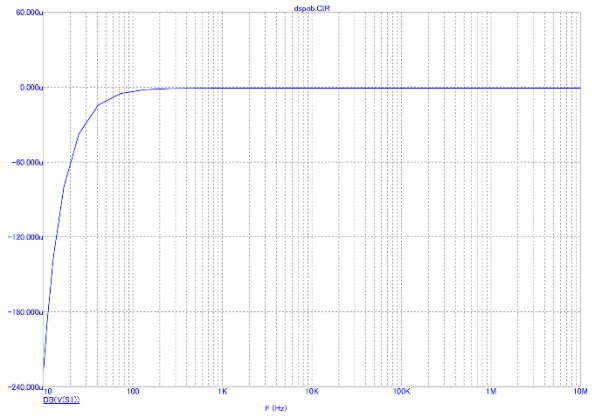


Figure 6: Filtre CR HPOUT

Le schematic d'une carte Arduino comportant un codec Wolfson a été étudié. Le signal d'entrée se propage par les entrées LINE IN. Les LINE INPUTS sont des entrées haute impédance et faible capacitance.

Les composants externes sont similaires aux composants de la carte MikroElektronika. C'est une capacité de liaison suivi d'un montage RC. Ce passe bas a une fréquence de coupure à 17.75 kHz. Les sorties utilisées sont les sorties HPOUT, ou High Pass OUT. La sortie est un filtre passe haut avec une fréquence de coupure 0.0723 Hz.

L'agencement des composants pour les entrées/sorties sont similaires à la carte MikroElektronika.

3.3 Analyse du codec présent sur la carte de développement FPGA DE2-70

L'objectif est d'analyser le fonctionnement d'un second module comprenant le même codec Wolfson 8731. Ce codec est présent au sein d'une carte de développement Altera DE2-70.Les différences entre la carte de développement Mikroelectronika et le codec au sein de la FPGA se placent au niveau des filtres des étages d'entrées et de sorties.

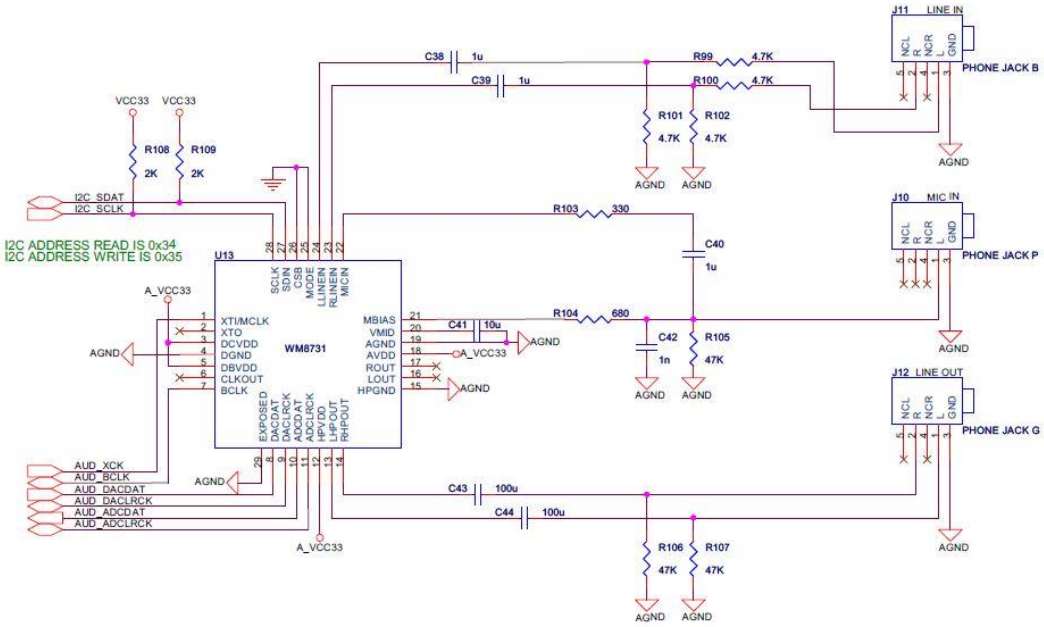


Figure 7: Schématique du codec WM8731 présent sur la carte de développement FPGA

La carte dispose des entrées LINEIN déjà disponibles. Les filtres en entrées permettront théoriquement de laisser passer les basses fréquences au sein du codec (environ 15Hz). L'entrée Line In comporte un pont diviseur.

Le but est d'empêcher la surcharge du canal. En règle générale, l'amplitude provenant d'un CD atteint les 2 Vrms. Le pont diviseur permet le passage à 1 Vrms. L'entrée MICIN est aussi disponible et est en tout point semblable au circuit MICIN de la carte MikroElektronika. Les sorties HPOUT sont utilisées. La fréquence de coupure du filtre passe haut est de 0.0339 Hz.

L'utilisation du FPGA pour faire passer du son présente un très bon résultat en sortie. Le son de sortie est de bonne qualité.

➢ Test A : Analyse de l'entrée MICIN, seuil de tension en entrée à 1V pic à pic

Le signal en sortie du codec possédait un certain nombre d'harmoniques. Ces harmoniques sont dues à la différence de valeur prise pour le filtre de l'étage d'entrée, modifiant la dynamique d'entrée du codec

Tableau de mesures A						
Mesure	**Trace Average**	**Tension IN [Vpp]**	**Fréquence IN [Hz]**	**Puissance OUT [W]**	**Puissance bruit [W]**	**SNR [dB]**
1	5	1	2.52 k	2.38 m	185 p	71.09
2	5	1	5.04 k	11.5 m	93.7 p	80.89
3	5	1	9.96 k	23.1 m	45.9 p	87.02
4	5	1	15 k	28 m	70.9 p	85.97
5	5	1	20.04 k	30 m	70 p	86.32
6	5	1	24 k	31.3 m	33.6 p	89.69

➢ Test B : Analyse de l'entrée MICIN, seuil de tension en entrée à 500mV pic à pic

Tableau de mesures B						
Mesure	**Trace Average**	**Tension IN [Vpp]**	**Fréquence IN [Hz]**	**Puissance OUT [W]**	**Puissance bruit [W]**	**SNR [dB]**
1	5	500 m	2.56 k	826 µ	87.9 p	69.73
2	5	500 m	5.04 k	3.64 m	50 p	78.62
3	5	500 m	10 k	8.03 m	32.4 p	83.94
4	5	500 m	15.05 k	10.5 m	241 p	76.39
5	5	500 m	20.08 k	10 m	177 p	77.52
6	5	500 m	24.04 k	11.6 m	34.9 p	25.22

L'étude du FPGA nous a permis de nous rendre compte que la dynamique d'entrée était un paramètre à prendre en compte. Celle-ci est dépendante de l'entrée analogique.

Comme précédemment, voici les formes des signaux de sortie du codec avec le filtre d'entrée proposé sur la carte de développement FPGA :

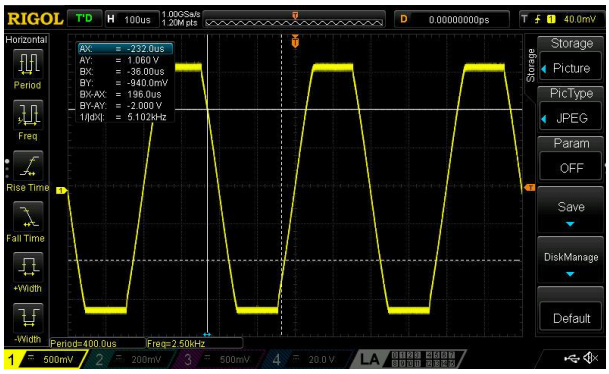


Figure 8: Signal 1V pic à pic

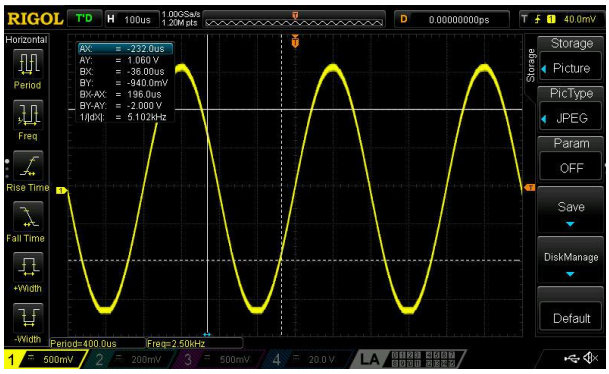


Figure 9: Signal 700mV pic à pic

Pour le filtre du kit de développement, le seuil de non-saturation des signaux se trouve désormais aux alentours de 700mV pic à pic pour les signaux en entrée.

3.4 Design de la schématique du nouveau kit

Le schematic présente l'allure et le placement des composants pour la fabrication du PCB. Dans un premier temps, quelques règles de bonnes pratiques vont être énoncées. Le texte se poursuivra par la présentation des blocs principaux constituants la future carte électronique. Enfin, un aperçu du schematic et du schéma de rooting sera exposé.

➢ Règles pour le développement d'une carte électronique

Les règles qui vont suivre sont des règles de bonnes pratiques. Cette liste ne reprend pas tous les critères possibles car ils seraient trop nombreux. Ceux explicités sont ceux pris en compte dans l'élaboration de notre carte électronique. Le but est de s'en rapprocher le plus pour respecter la compatibilité électromagnétique.

Identifions d'abord les grands contributeurs de bruit. Ils sont au nombre de trois. Premièrement, les circuits intégrés. Les I/O pins ont des pistes dites « longues » pouvant être assimilées à des antennes. En seconde place se trouvent les alimentations de puissance, et ce incluant les régulateur de tension. Les derniers gros contributeurs sont les oscillateurs.

Les lignes directrices à suivre :

- Minimiser au maximum la longueur des pistes, surtout ceux portant des signaux digitaux d'horloge. Dans cette optique, la carte réalisée utilise une petite superficie (60 mm × 60 mm). Les éléments ont été placés afin de rapprocher les composants nécessitant des connections entre-eux.

- Les circuits sur un PCB doivent être groupés par type. Les circuits de puissance devraient être les plus près possibles des connecteurs. Les circuits digitaux à haute vitesse(high speed digital circuits) devraient être les plus éloignés des connecteurs.

- Les composants en surface-mount sont à préférer aux through-hole. Ils permettent un placement plus rapproché et ont une inductance plus réduite.

- Dans le cas où il y a des I/O pins, les inputs devraient être reliées à la plus basse impédance possible (la masse). L'inductance du transistor de sortie non utilisé transfert du bruit du rail de puissance à la pin.

- Placer le microprocesseur juste à côté du régulateur de tension et le régulateur de tension à côté du point d'entrée de Vbatt.

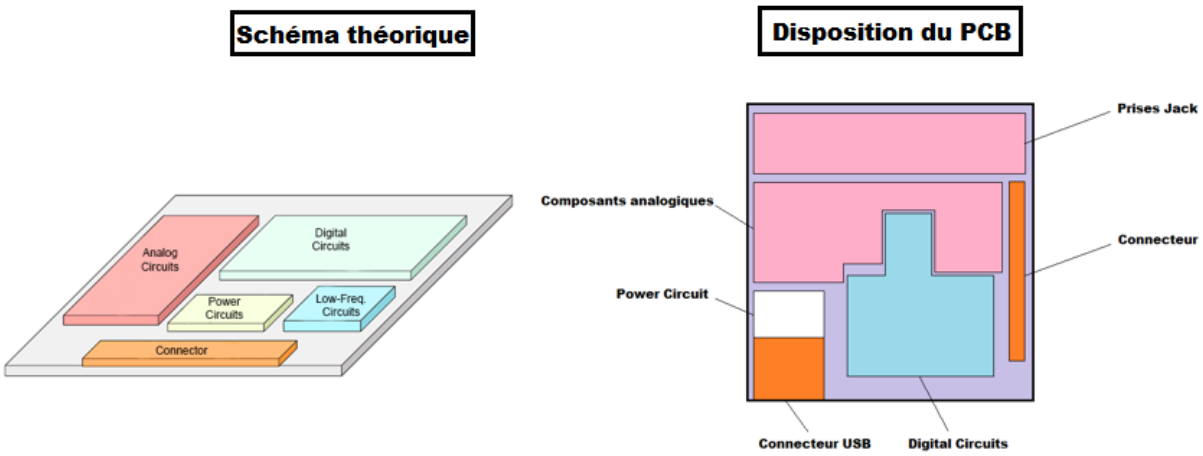


Figure 10: Schéma théorique et appliqué du PCB

➢ Présentation de blocs composant le PCB

- *Alimentation*

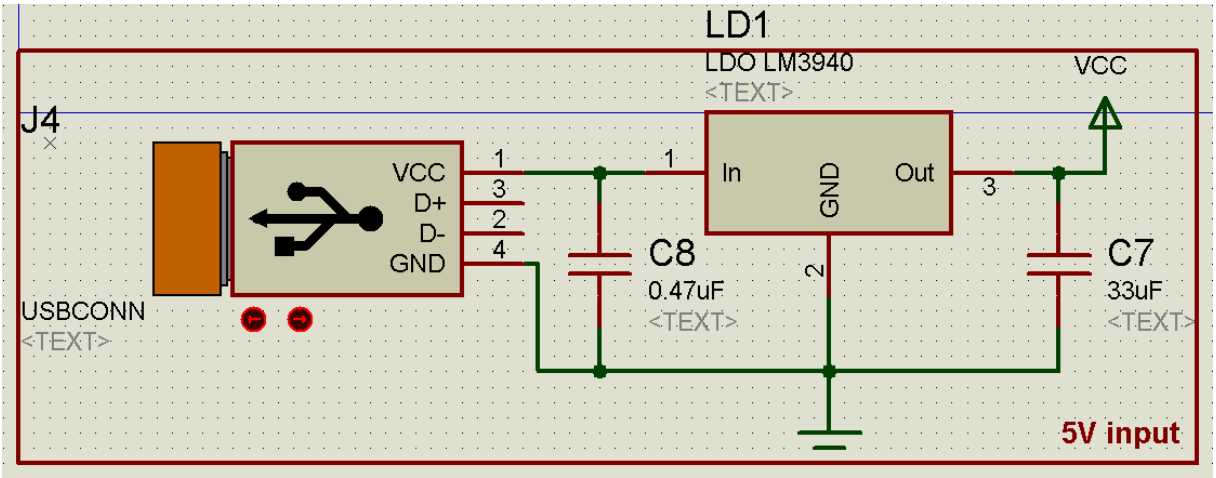


Figure 11: Bloc alimentation de la carte

Afin de pouvoir se libérer de la contrainte d'avoir une alimentation stabilisée à proximité, un circuit permettant l'alimentation de la carte a été pensé. Le circuit transforme 5 volts provenant d'un connecteur USB en du 3.3 Volts. Le choix du connecteur est simple : c'est une connectique très répandue.

La carte pourrait être alimentée soit par un port USB provenant d'un ordinateur ou par celui d'un transformateur chargeant un smartphone. Le courant mis à disposition par un port USB d'ordinateur est de l'ordre des 500 mA. Le chargeur d'un smartphone peut délivrer 550 mA.

Au grand maximum, la carte consomme 100 mA. Le composant permettant l'acquisition d'une alimentation stable est un LDO, pour Low Drop Out. Le LM3940 va assurer une tension de 3.3 volts même si sa tension varie tant qu'elle ne descend pas en dessous du seuil limite des 3.5 volts.

- *MICIN*

L'entrée MICIN est similaire aux entrées standards des cartes étudiées. Elles présentent en plus un jumper permettant la connexion entre deux configurations de filtres pour visualiser les effets d'un filtre avec une fréquence de coupure plus basse que l'autre.

- *LINE IN*

Elles sont similaires aux entrées Line In étudiées sur la carte FPGA Altera DE2-70.

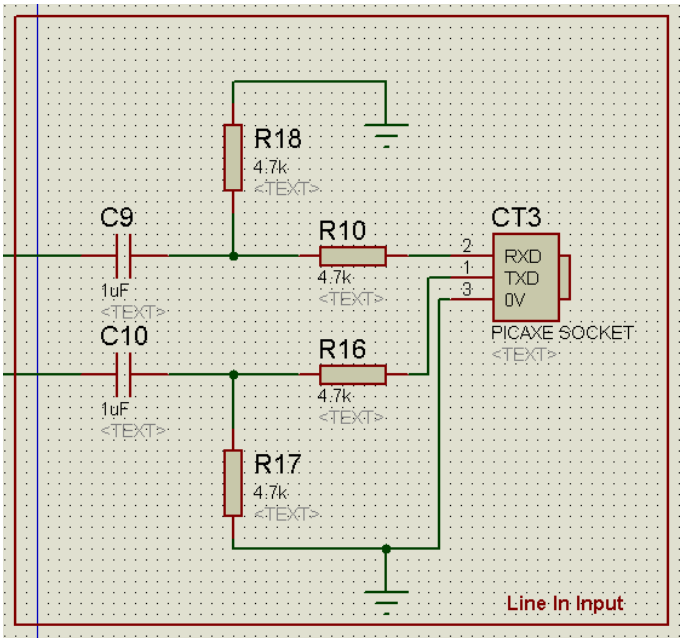


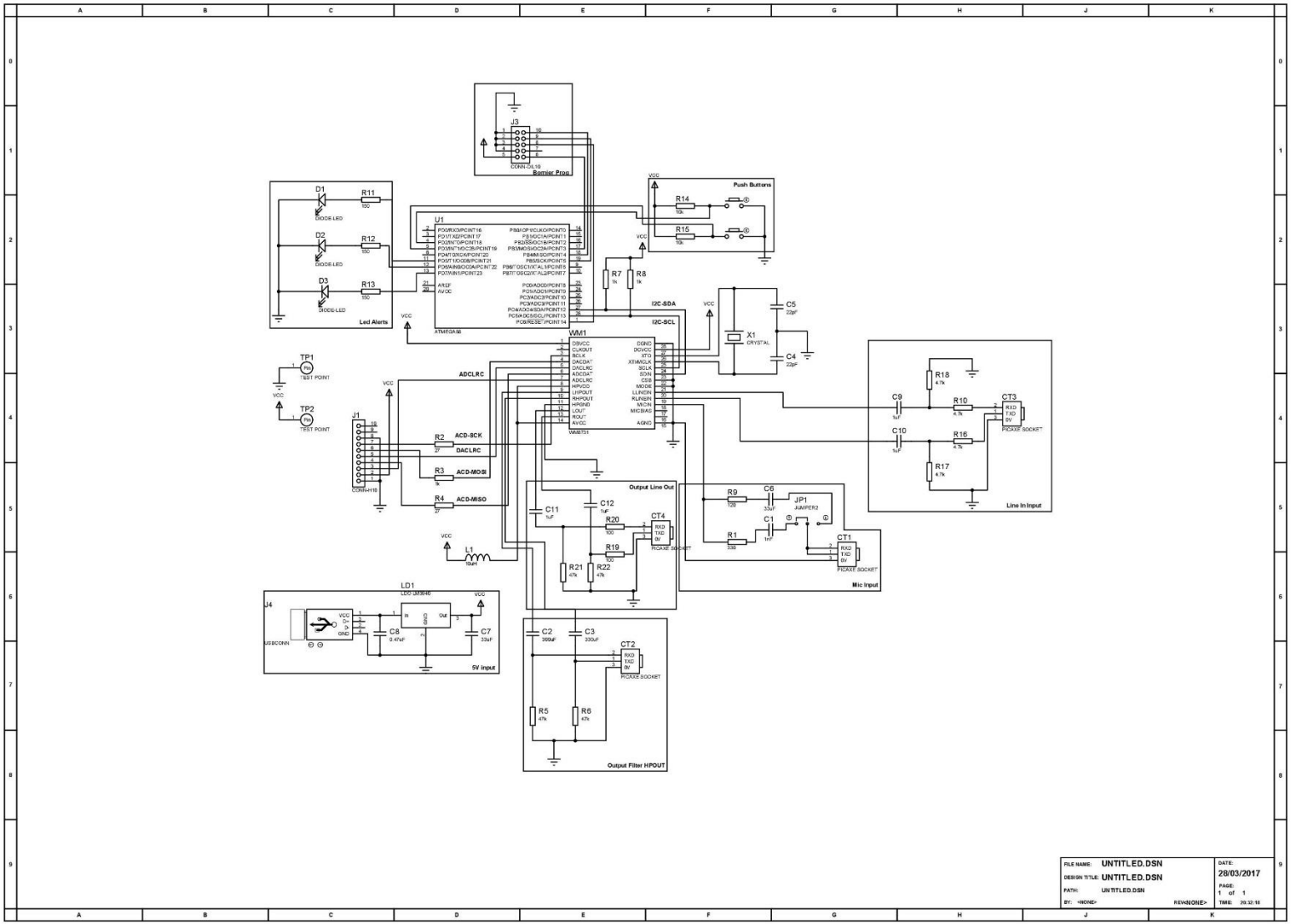
Figure 12: Bloc des entrées LINE IN

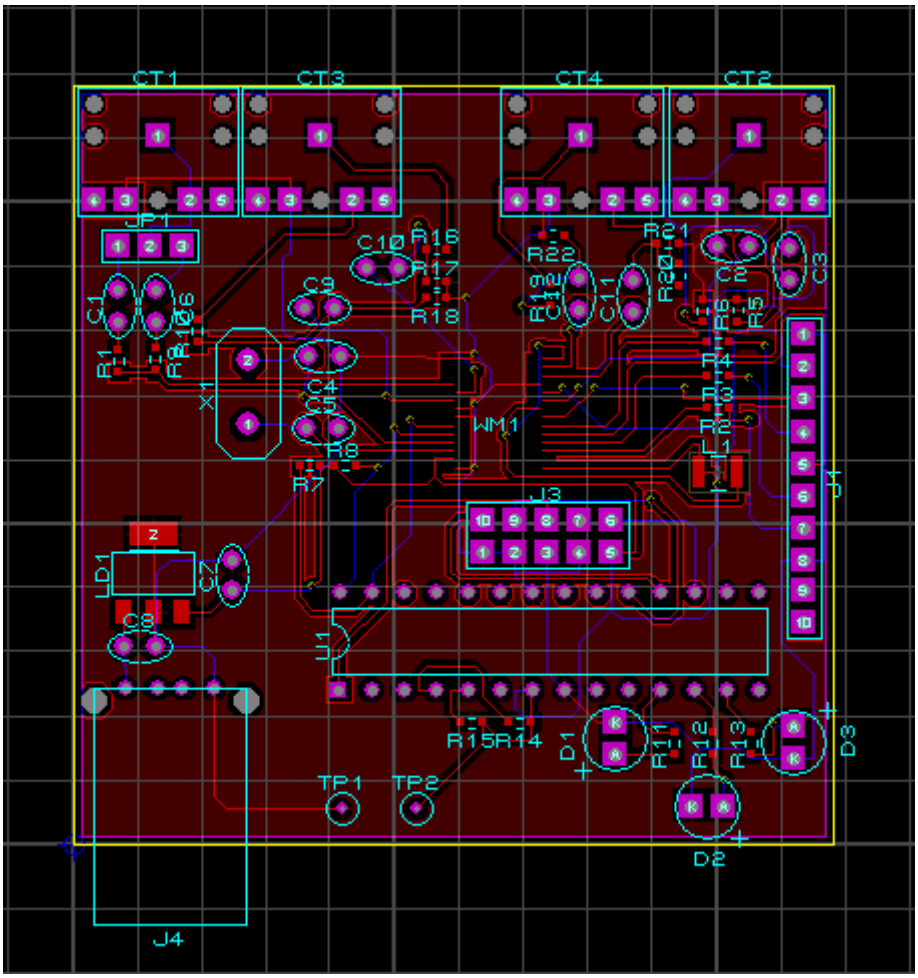
- *LINE OUT&HPOUT*

Ces sorties sont les copies conformes des circuits utilisés sur la carte FPGA pour les Line OUT et de la carte Mikroelektronika pour les HPOUT.

➢ Vue des schémas

Le PCB est un PBC double couche. Un plan de masse est placé sur la couche bottomcopper.


J3
CONN-SIL10
Bomier Prog
D1
D2
D3
R11
R12
R13
DIODE-LED
Led Alerts
U1
ATMEGA168
Push Buttons
R14
R15
10k
R7
R8
1k
I2C-SDA
I2C-SCL
C5
C4
22pF
X1
CRYSTAL
WM1
WM8731
TP1
TP2
TEST POINT
ADCLRC
J1
CONN-H10
R2
R3
R4
ACD-SCK
DACLRC
ACD-MOSI
ACD-MISO
Output Line Out
C11
C12
1uF
R20
100
R19
33
R21
R22
47k
CT4
CT1
PICAXE SOCKET
L1
10uH
R9
R1
120
330
C6
C1
33uF
1nF
JP1
JUMPER2
Mic Input
C9
C10
R18
R10
R16
R17
4.7k
CT3
Line In Input
LD1
LDO 1A/3648
J4
USBCONN
C8
0.47uF
C7
33uF
5V Input
C2
C3
330uF
CT2
R5
R6
47k
Output Filter HPOUT
FILE NAME: UNTITLED.DSN
DESIGN TITLE: UNTITLED.DSN
PATH: UNTITLED.DSN
BY: <NONE>
REV:<NONE>
DATE: 28/03/2017
PAGE: 1 of 1
TIME: 20:32:16


CT1
CT3
CT4
CT2
JP1
C10
R16
R17
R18
C9
C1
R1
X1
C4
R8
R7
WM1
R21
R22
R20
C3
R5
J4
J3
LD1
C7
C8
U1
R15
R14
D1
R11
R12
R13
D3
TP1
TP2
D2
J4

3.5 Explication des filtres du codec

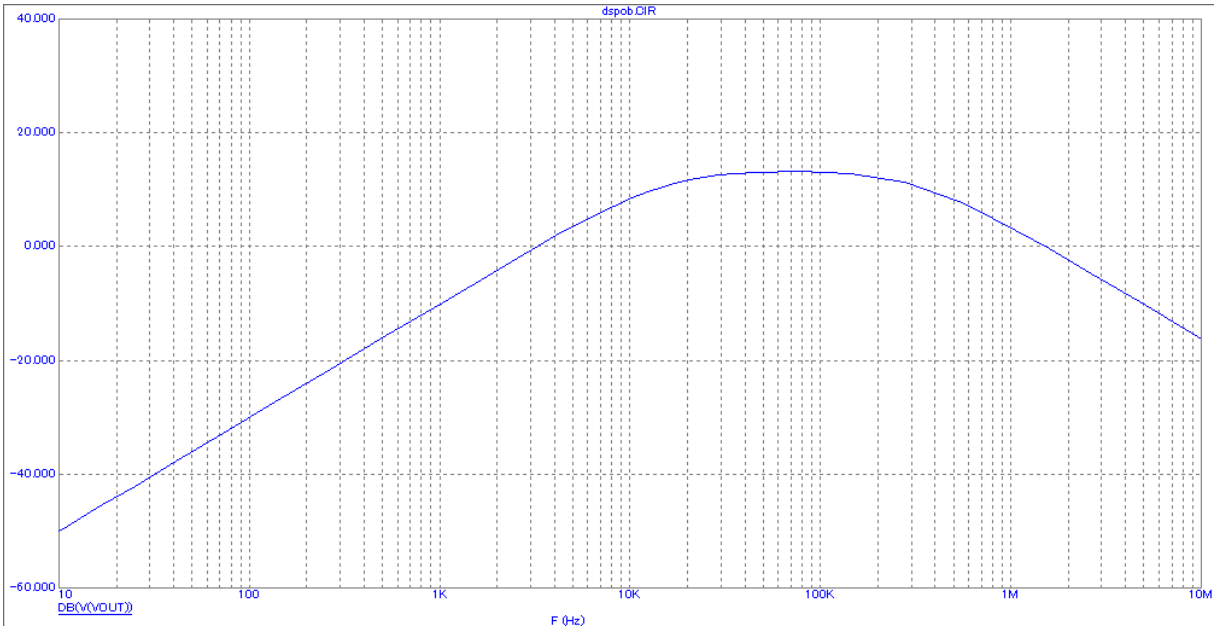


Figure 13: Réponse en fréquence du filtre de l'entrée micro du codec audio proto non modifié

La fréquence de coupure basse se trouve être aux environs de 15kHz

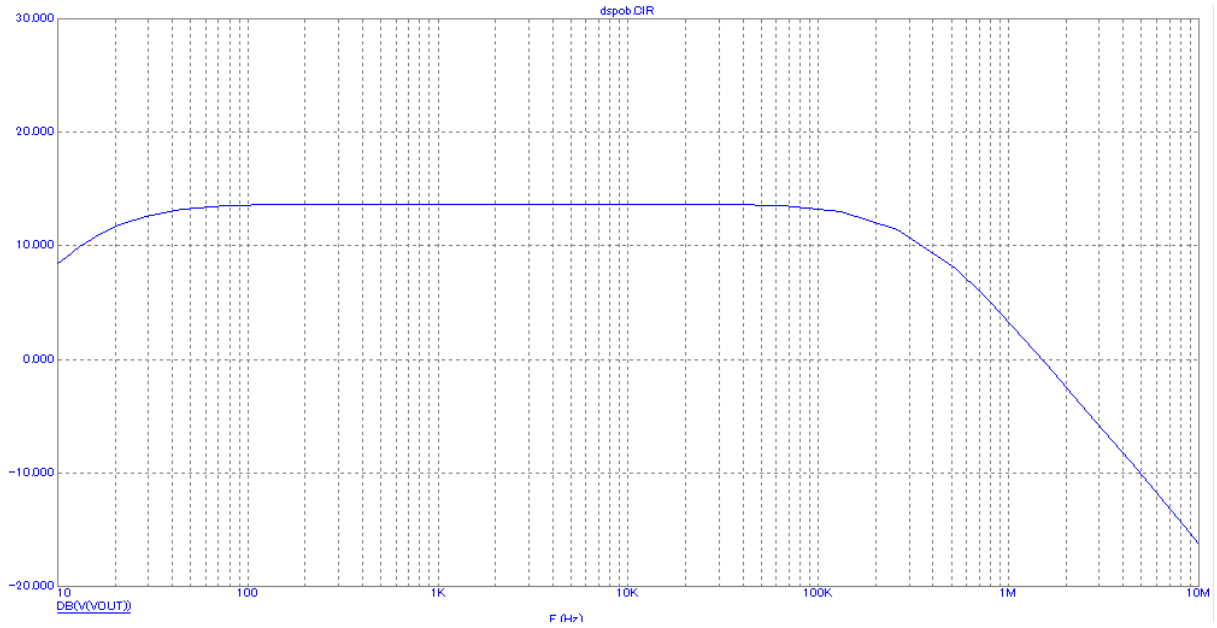


Figure 14: Réponse en fréquence du filtre de l'entrée micro du FPGA

La fréquence de coupure basse se trouve cette fois-ci aux environs de 15Hz. Cette fréquence de coupure permet ainsi de laisser passer les basses fréquences sonores.

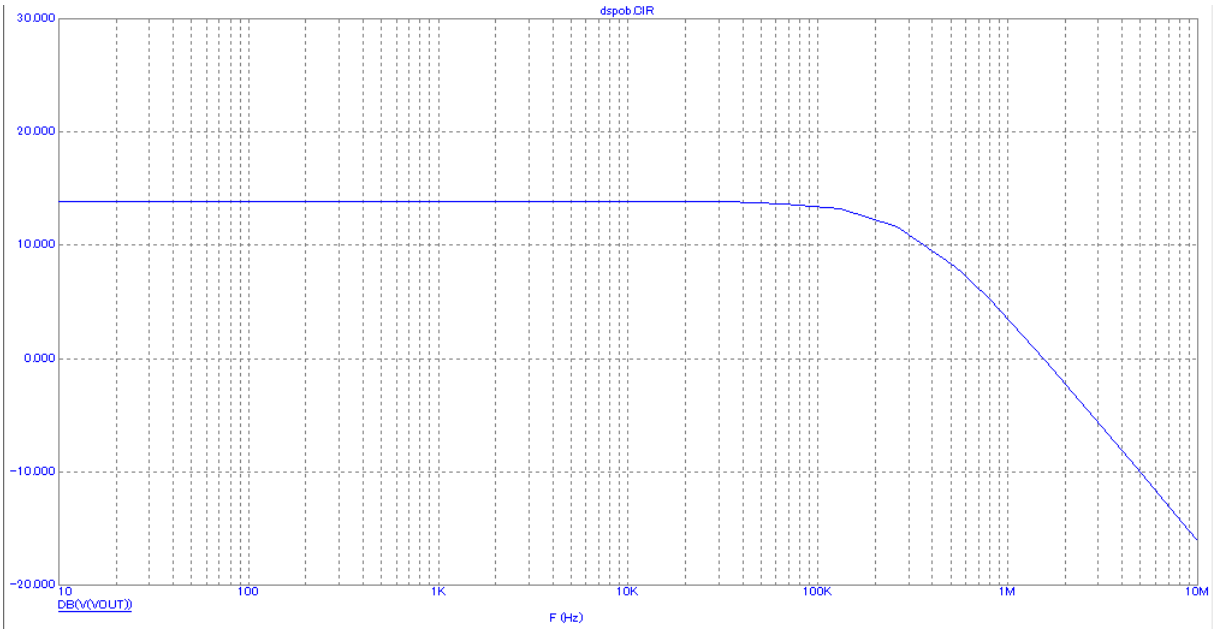


Figure 15: Réponse en fréquence du filtre de l'entrée du codec modifié

La fréquence de coupure basse du filtre proposé en externe du kit de développement Mikroeletronika se trouve être aux environ de 0.5Hz. Comme pour le filtre FPGA, la fréquence de coupure laisse passer les basses fréquences sonores.

4 Traitement de signal audio sur FPGA

Cette partie est réservée à l'explication de l'architecture sur FPGA permettant de mettre en place un module de transmission de données audio avec le codec et de faire du traitement de signal audio. Il est à noter que le traitement de signal audio s'est limité à un effet d'écho sur le son envoyé par le codec et n'est pas encore fonctionnel.

Pour la transmission des données audio nous avons utilisé le protocole I²S car c'est un standard pour la transmission audio, ce qui permettra éventuellement d'utiliser d'autres systèmes de traitement audio dans le futur.

Le programme sur FPGA a été réalisé en VHDL sur le kit de développement DE2-70. Le kit contient une multitude de composants. Les composants qui nous intéresseront ici seront principalement le FPGA de la marque Altera Cyclone II 2C70, le codec Wolfson WM8731 ainsi que des ports d'entrées/sorties, des boutons poussoirs, des LEDs et des switches.

4.1 Rappel sur le codage en VHDL

Le VHSIC Hardware Description Language (VHDL) est un langage de description matérielle. Il permet de concevoir au niveau « Register Transfer Level » (RTL), qui permet de plus facilement décrire le fonctionnement du système sans se soucier de la manière dont il sera implémenté réellement, au niveau des transistors ou des portes logiques.

Le système digital va être décris par une entité associée avec une architecture. L'entité ne décrit que l'interface avec l'extérieur, au moyen des ports d'entrées/sorties. Elle peut être vue comme une boîte noire, dont on ignore comment se comporte le système digital. L'architecture quant à elle va décrire le contenu du système, va décrire comment il se comporte au niveau RTL. La figure ci-dessous illustre comment est décrit un système digital en VHDL.

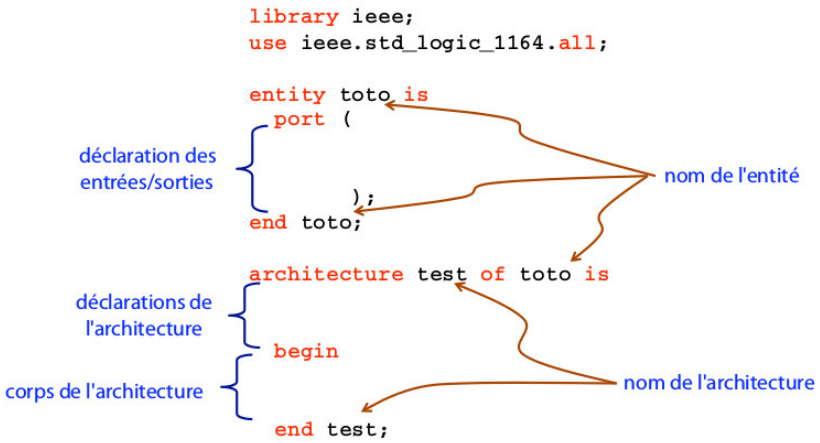


Figure 16: Exemple de code VHDL

Le VHDL permet aussi de créer des composants, qui sont des sous-systèmes du système digital. Ils sont aussi décrits par une entité et une architecture, et permettent de décomposer le système digital afin de simplifier le développement. En effet cela permet de développer des composants séparément, qui sont plus simples et qui sont aussi vu comme des boîtes noires, et ensuite de les connecter ensemble dans un « top level », formant le système digital.

4.2 Analyse de l'existant

Un programme sur FPGA avait été déjà été réalisé l'année passée afin de d'**assurer la réception et la transmission de données audio avec le codec**. La première chose que nous avons faite a donc été de lire la documentation réalisée sur le rapport de l'année passée et de compiler et téléverser le programme dans le FPGA afin de tester sa fonctionnalité. Cela a posé quelques problèmes, notamment parce qu'il y avait deux versions du programme et qu'on ne savait pas laquelle était la bonne, et aussi parce que les ports d'entrées et sorties étaient mal configurés. Nous avons donc dû analyser le code afin de pouvoir faire fonctionner le programme réalisé l'année passée.

➢ Restructuration du programme

Le code était très peu documenté et il était difficile de s'y retrouver. Nous devions nous baser sur le programme développé, permettant d'échanger les données audio avec le codec, pour ensuite pouvoir faire des opérations de traitement de signal audio. Il fallait donc pouvoir avoir un programme avec une structure claire et bien organisée afin de se retrouver plus facilement dans le code pour faciliter le développement.

Le programme VHDL ne comportait qu'un seul fichier ne comportant qu'une seule entité. Nous avons donc décomposé le système digital en plusieurs composants qui ont chacun une fonction particulière, et un « top level » connectant tous les composants ensemble. La figure ci-dessous illustre l'architecture retenue :

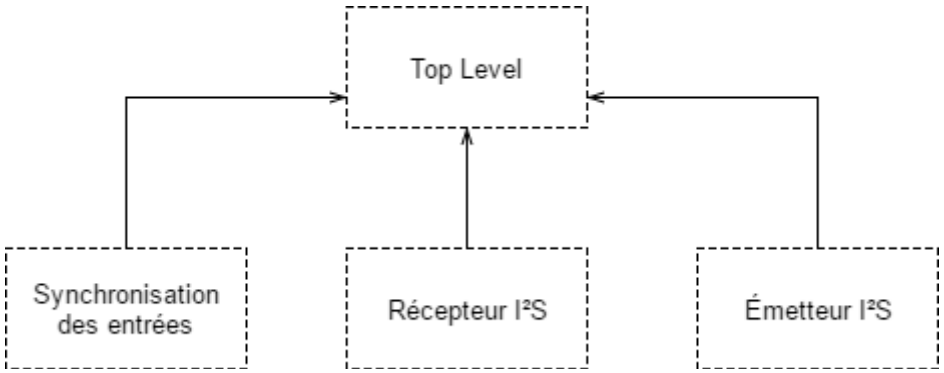


Figure 17: Structure de l'architecture VHDL pour la communication I2S

Les signaux d'entrée, qui sont les signaux ADCLRC, DACLRC, BCLK et ADCDAT vont passer dans le composant « synchronisation des entrées ». Celui-ci va**échantillonner les signaux** au rythme de la clock interne de laFPGA(50MHz) et va **détecter les flancs montant et descendant des signaux**.

Le composant « récepteur I²S » va utiliser les signaux en sortie du premier composant. Le récepteur I²S va les utiliser pour décoder les données envoyées en I²S par le codec. Le récepteur I²S est en fait composé de registres à décalage, et va stocker les données audio dans deux registres (un registre pour le canal gauche et un registre pour le canal droit).

Le composant « émetteur I²S » va directement utiliser les données audio décodées par le récepteur pour les renvoyer au codec par I²S à l'échantillon suivant. Il va aussi utiliser les signaux I²S du premier bloc.

➢ Analyse de la communication

En analysant les trames de la communication en I²S, nous nous sommes aperçus qu'il y avait des problèmes de timing. En effet si on regarde une trame I²S on doit lire les données sur le flanc montant de la bit clock et on doit écrire le bit suivant au flanc descendant. Dans le programme développé il y avait un décalage de quelques coups d'horloge de l'horloge interne du FPGA (50 MHz), ce qui provoquait parfois un mauvais échantillonnage des données lorsque le FPGA lisait les données envoyées par le codec et lorsque le codec lisait les données envoyées en retour par le FPGA. Cela provoquait du bruit sur le signal audio.

Nous avons essayé le corriger le problème en modifiant le code VHDL, mais le problème est que la manière de gérer la communication sur FPGA n'est pas optimale et il est très difficile de détecter la source des erreurs car la communication n'a pas été gérée par une machine d'états. Cela a pour conséquence qu'il y a beaucoup de signaux qui sont interdépendants et qu'il faut analyser le moment où chaque signal change d'état.

Par exemple le composant qui synchronise les entrées va détecter une transition d'un signal digital à chaque flanc montant de l'horloge interne du FPGA (50 MHz). Cela a pour conséquence que la transition du signal ne sera vue qu'au coup d'horloge suivant. Si plusieurs signaux sont en cascade, ces délais vont s'ajouter et provoquer des délais plus importants.

Nous aurions pu continuer à chercher les sources du problème, mais cela nous aurait peut-être pris plus de temps que de complètement redévelopper le système de communication. En redéveloppant le système de communication avec des machines d'états, cela nous permettra d'avoir un programme qui sera plus optimisé, plus simple à maîtriser et qui servira de base plus solide pour les années à venir.

4.3 Architecture FPGA actuelle

L'architecture du programme en VHDL qui a été redéveloppé comporte quatre entités :

- Le top level
- Le composant récepteur I²S
- Le composant transmetteur I²S
- Le composanteffet d'écho

Le « top level » du programme sur FPGA va se charger de connecter les signaux des différents composants ensemble. La figureci-dessousprésente l'architecture du programme :

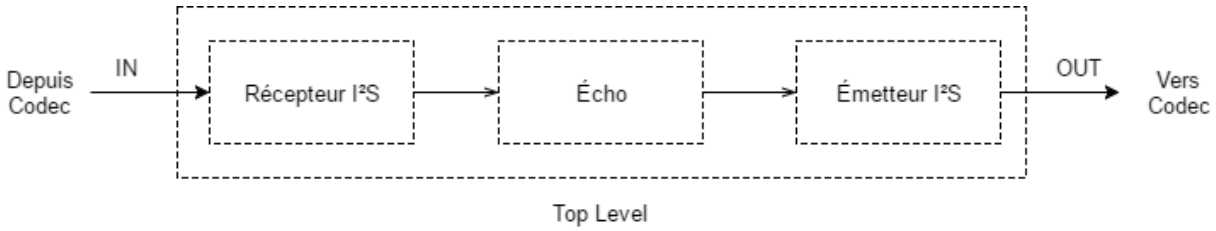


Figure 18: Chaine de traitement de la communication I2S

Le récepteur I²S va se charger la réception des données audio envoyées par le codec en I²S et de les stocker en mémoire. Le composant « écho » va faire un traitement sur les données envoyées par le récepteur pour faire un écho sur le son, et enfin l'émetteur I²S va renvoyer les données ayant subis un traitement au codec en I²S.

A. Communication en I²S sur FPGA

L'effet d'écho n'étant pas encore fonctionnel, nous n'expliquerons en détail que l'architecture permettant de communiquer en I²S.

La communication du FPGA avec le codec est effectuée en mode « master », c'est-à-dire que c'est le codec qui génère les signaux pour la communication I²S, et le FPGA n'a plus qu'à renvoyer un seul signal qui est le DACDAT. Le signal DACDAT renvoie les données ayant subis un traitement vers le codec. La communication se fait selon le schéma ci-dessous :

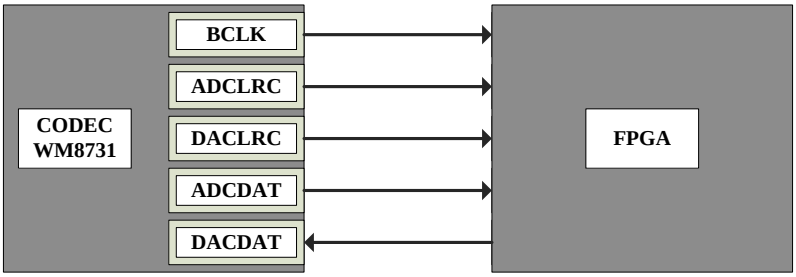


Figure 19: Sens des signaux de contrôle en mode "Master"

La longueur des données audio est de 24bits pour chaque canal.

Le top level va connecter les signaux des différents composants ainsi que les ports d'entrées/sortie du système ensemble. La figure ci-dessous présente les différentes connexions :

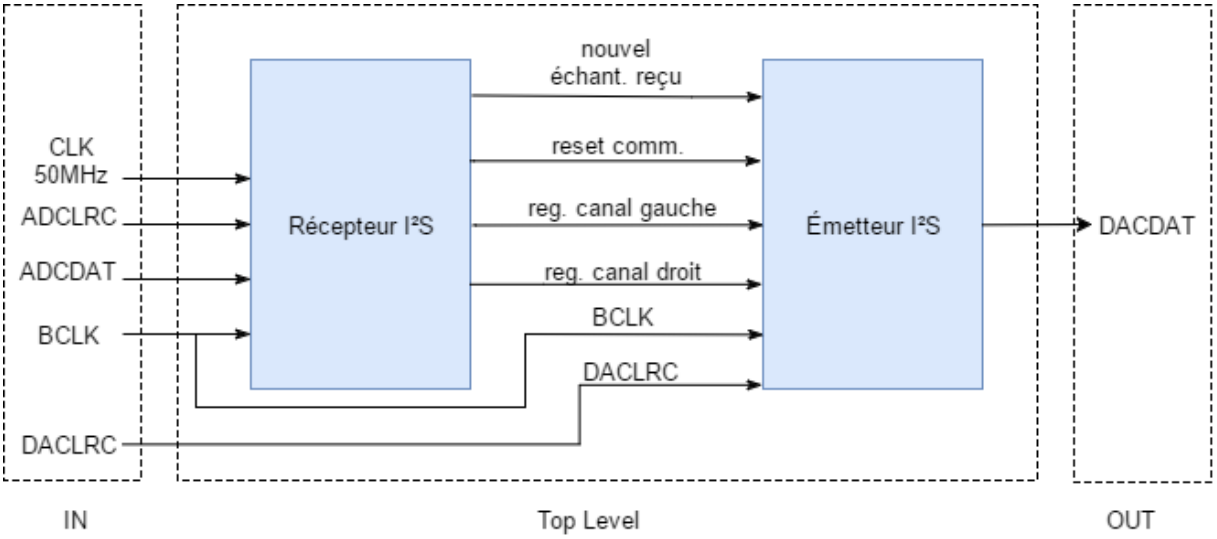


Figure 20: Structure en version composants de l'architecture VHDL

Les signaux d'entrées du système sont l'horloge interne du FPGA à 50Mhz, et les signaux I²S envoyés par le codec, à savoir BCLK, ADCLRC, ADCDAT et DACLRC. Le signal de sortie est le signal DACDAT envoyé au codec.

Le récepteur I²S utilise les signaux BCLK, ADCLRC et ADCDAT pour décoder les trames I²S. Il stocke ensuite les données décodées dans un registre pour le canal gauche et un registre pour le canal droit. Le signal « nouvel échant. reçus » sert à indiquer lorsqu'il est mis à 1 qu'un nouvel échantillon a été reçu et a été stocké dans les deux registres précédents.

Le récepteur I²S va aussi vérifier que le codec communique toujours avec le FPGA. Si ce n'est pas le cas, par exemple si le codec n'est plus sous tension ou si on est passé en mode « by-pass » (le codec ne communique plus avec le FPGA), le récepteur va mettre à 1 le signal « reset com. » pour réinitialiser la communication.

L'émetteur I²S utilise les signaux BCLK, DACLRC et DACDAT pour encoder les trames I²S. Le signal qui contient les données envoyées au codec est le DACDAT. L'émetteur utilise directement les registres du canal gauche et droit utilisé par le récepteur pour stocker l'échantillon reçu.

Il va commencer à émettre lorsque le signal « nouvel échant. reçus » passe à 1, indiquant qu'un nouvel échantillon a été réceptionné par le récepteur I²S. Si le signal « reset com. » est mis à 1, l'émetteur I²S doit réinitialiser sa communication.

➢ Récepteur I²S

Le récepteur I²S est implémenté sous forme d'une machine d'états. Son but est de décoder une trame I²S et de stocker les données dans des registres.

Reprenons la figure décrivant une trame I²S :

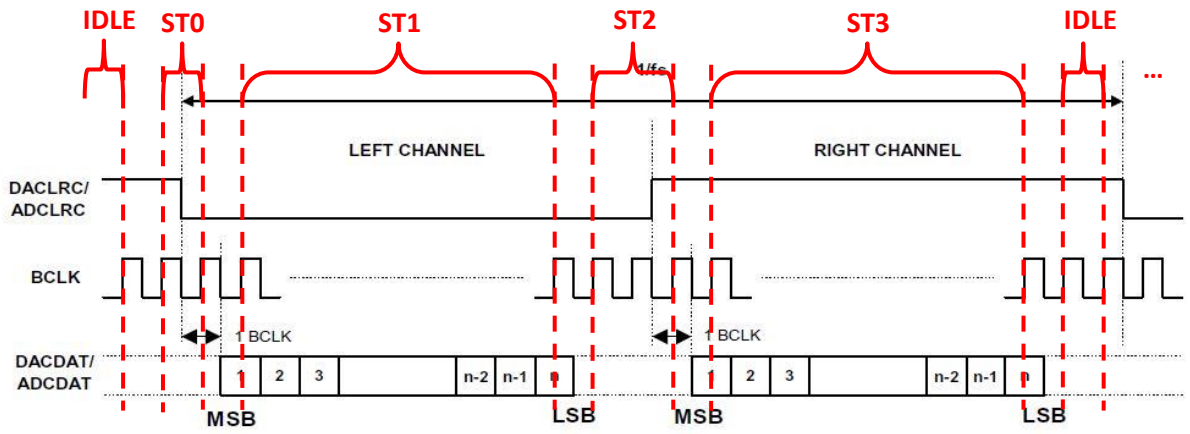


Figure 21: Détails d'une trame de communication avec étapes

Sur base de cette trame I²S on construit la machine d'état illustrée à la figure suivante, permettant de décoder la trame. Remarquons que nous avons indiqué sur la trame I²S les états auxquels se trouve la machine d'état.

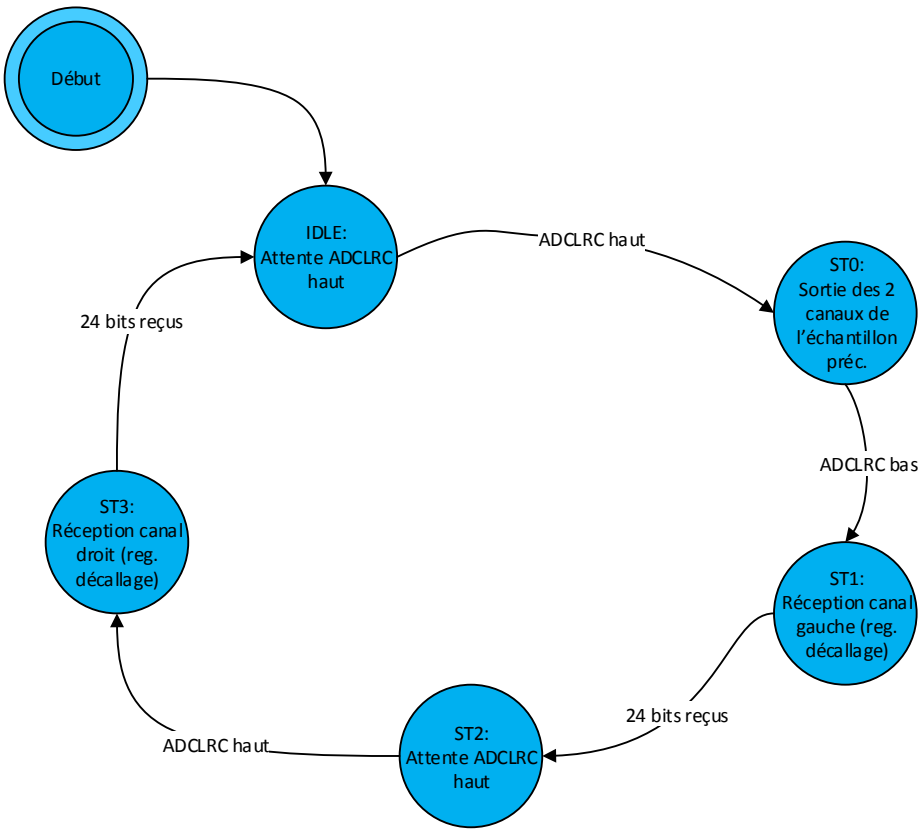


Figure 22: Machine d'état du récepteur I2S

Puisque nous devons lire les données au flanc montant de la bit clock BCLK, la machine d'état va être mise à jour au flanc montant du signal BCLK. Cela implique qu'il y aura un délai d'une période de BCLK entre chaque changement d'état, comme on peut le voir sur la figure précédente.

La machine d'état possède 4 états :

- **IDLE**

Dans cet état on va attendre que le signal ADCLRC soit haut afin d'être synchronisé avec le début de la trame.

Lorsque le signal ADCLRC est haut on passe à l'état suivant ST0.

- **ST0**

Dans cet état on va sortir les données du canal gauche et droit réceptionnées à l'échantillon précédent afin qu'elles puissent être utilisées par d'autres composants (dans notre cas afin qu'elles soient utilisées par l'émetteur I²S pour les retransmettre au codec).

Lorsque le signal ADCLRC est bas on passe à l'état suivant ST1. Cela indique qu'on a atteint le début de la trame.

- **ST1**

Cet état va servir à réceptionner les données du canal gauche envoyées sur le signal ADCDAT à l'aide d'un registre à décalage.
Lorsqu'on a reçu 24 bits, qui est la taille de la donnée audio d'un canal, on passe à l'état suivant ST2 servant à attendre la réception du canal droit.

- **ST2**

Dans cet état on va attendre que le signal ADCLRC soit haut afin de recevoir le canal droit.

Lorsque le signal ADCLRC est haut on passe à l'état suivant ST3.

- **ST3**

Cet état va servir à réceptionner les données du canal droit envoyées sur le signal ADCDAT à l'aide d'un deuxième registre à décalage.

Lorsqu'on a reçu 24 bits, on revient à l'état de départ IDLE afin d'attendre l'arrivée de la trame suivante et d'être synchronisé avec le début de trame.

➢ Transmetteur I²S

Le transmetteur I²S est aussi implémenté sous forme d'une machine d'états. Son but est d'encoder une trame I²S et de sortir le signal DACDAT pour l'envoyer au codec.

Reprenons encore la figure décrivant une trame I²S dans laquelle on indique les états de la machine d'états :

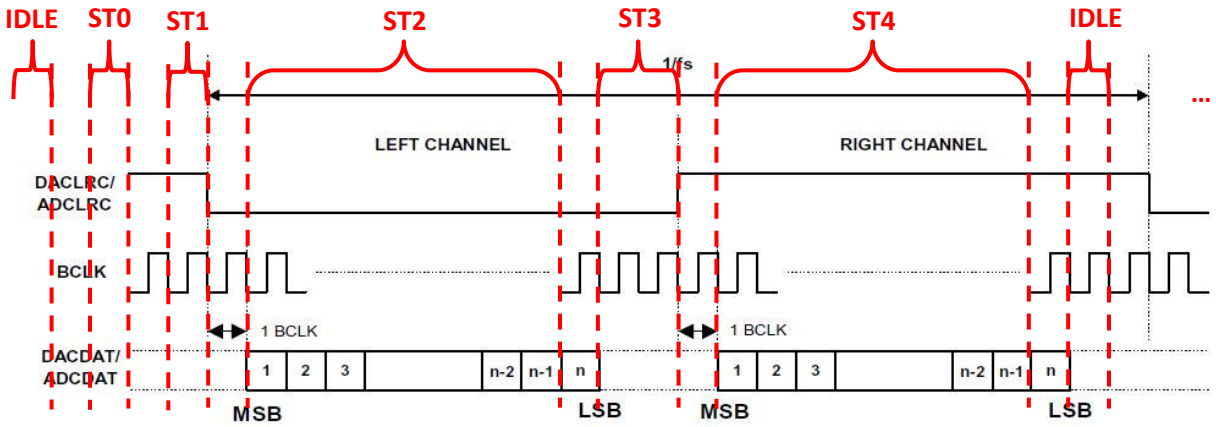


Figure 23: Détails d'une trame de communication avec étapes

La machine d'état qui permet d'encoder la trame I²S est présentée à la figure suivante. Elle est assez similaire à la machine d'état du récepteur I²S.

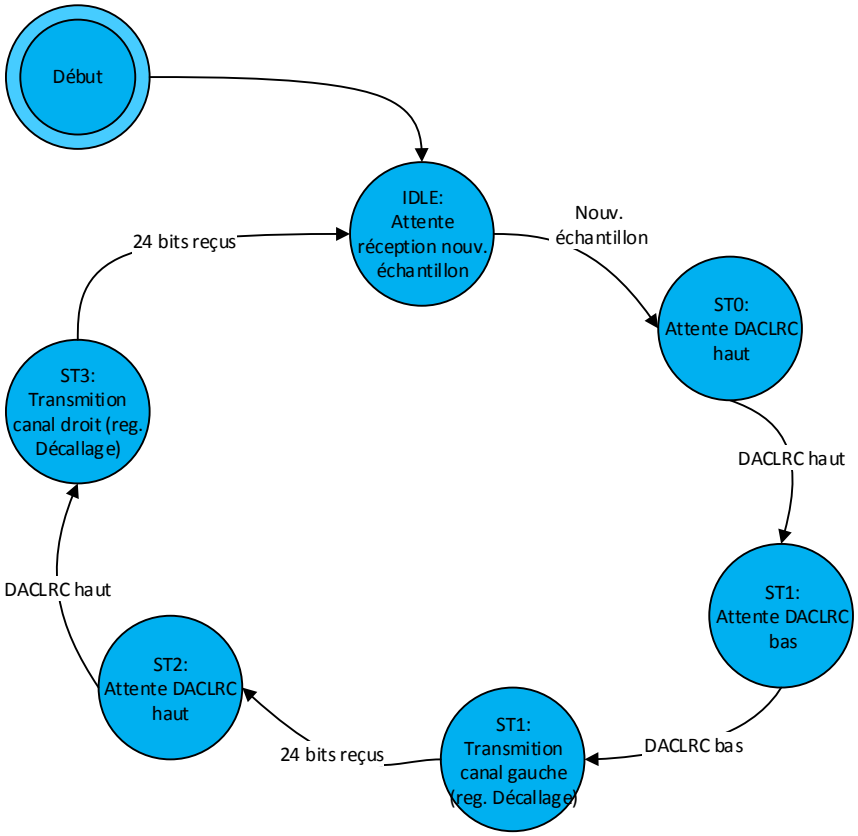


Figure 24: Machine d'état du transmetteur I2S

Puisque nous devons changer les données au flanc descendant de la bit clock BCLK, la machine d'état va être mise à jour au flanc descendant du signal BCLK cette fois-ci. Cela implique qu'il y aura un délai d'une période de BCLK entre chaque changement d'état, comme on peut le voir sur la figure précédente.

La machine d'état possède 5 états :

- **IDLE**

Dans cet état on va attendre l'arrivée d'un nouvel échantillon réceptionné par le récepteur I²S.

Lorsqu'un nouvel échantillon est réceptionné on passe à l'état suivant ST0.

- **ST0**

Dans cet état on va attendre que le signal DACLRC soit haut afin d'être synchronisé avec le début de la trame.

Lorsque le signal DACLRC est haut on passe à l'état suivant ST1.

- **ST1**

Dans cet état on va attendre que le signal DACLRC soit bas, ce qui veut dire qu'il y aura eu un flanc descendant du signal DACLRC donc que nous sommes au début d'une nouvelle trame.

Lorsque le signal DACLRC est bas on passe à l'état suivant ST2.

- **ST2**

Cet état va servir à transmettre les données du canal gauche sur le port DACDAT à l'aide d'un registre à décalage.

Lorsqu'on a transmis 24 bits, qui est la taille de la donnée audio d'un canal, on passe à l'état suivant ST3 servant à attendre la transmission du canal droit.

- **ST3**

Dans cet état on va attendre que le signal DACLRC soit haut afin de pouvoir transmettre le canal droit.

Lorsque le signal DACLRC est haut on passe à l'état suivant ST4.

- **ST4**

Cet état va servir à transmettre les données du canal droit sur le port DACDAT à l'aide d'un deuxième registre à décalage.

Lorsqu'on a transmis 24 bits, on revient à l'état de départ IDLE afin d'attendre l'arrivée d'un nouvel échantillon à transmettre.

B. Effet d'écho

Le traitement de signal audio s'est limité à un effet d'écho sur le son. Il n'est malheureusement pas encore fonctionnel mais nous expliquons comment l'implémenter sur FPGA.

L'écho est réalisé selon le schéma bloc représenté sur la figure ci-dessous :

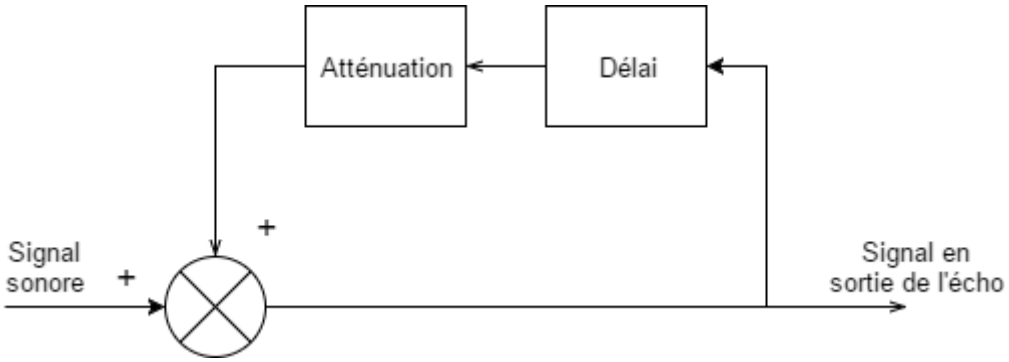


Figure 25: Schéma-bloc de l'effet d'écho

L'écho consiste donc à un délai suivi d'une atténuation du signal de sortie, que l'on va ajouter au signal sonore en entrée. Cela va permettre d'avoir une prolongation de l'effet d'écho, qui s'atténue avec le temps.

La partie la plus lourde à implémenter dans l'écho est le délai. En effet le délai demande de sauvegarder une grande quantité d'échantillons. Ici la fréquence d'échantillonnage du signal est de 48kHz. Si on veut un délai d'une seconde, il faudra stocker 48000 échantillons.

Or on a deux canaux dont la taille des données est de 24 bits, ce qui fait $48000.2.24 = 2{,}304\ Mb$. C'est une grande quantité de données, et le FPGA ne possède pas assez de mémoire RAM interne. La quantité maximum de RAM indiquée dans ladatasheet est de 1,152 Mb. On pourrait passer par de la RAM externe mais cela implique d'implémenter un driver spécifique pour contrôler la RAM, ce qui est assez coûteux en temps.

Une première solution est de diminuer la durée du délai (donc la durée de l'écho), à 100 ms par exemple. Cela demanderait de stocker 4800 échantillons et donc$230\ Kb$ de donnée, ce qui est faisable avec la RAM interne du FPGA.

Si la durée du délai de 100 ms est trop courte, on pourrait alors diminuer la taille des données, dont la taille maximum serait de 12bits pour un délai d'une seconde, ce qui utiliserait toute la RAM interne du FPGA ($48000.2.12 = 1{,}152 Mb$).

Cependant cela diminuerait la qualité du signal sonore et si on utilise l'entièreté de la RAM interne du FPGA cela demanderait un long temps de compilation et pourrait augmenter de manière critique les délais des signaux internes dans le FPGA. On pourrait aussi sous-échantillonner le signal audio pour l'écho, mais cela pourrait provoquer une distorsion du signal en sortie et réduirait aussi la qualité du signal.

➢ Utilisation de la RAM

Pour implémenter l'écho, on passe donc par de la RAM. On va en fait utiliser la RAM comme un buffer circulaire ; c'est-à-dire que lorsque l'on arrive à la dernière adresse, on va revenir au début et réécrire sur les anciennes données :

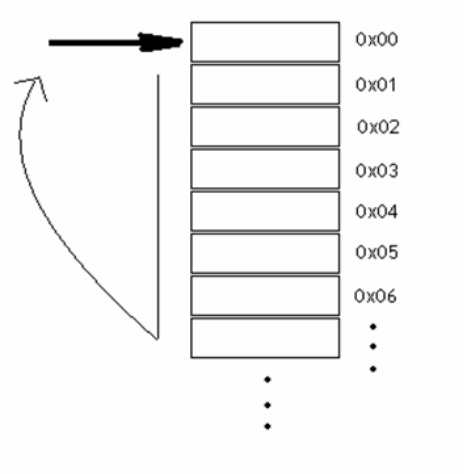


Figure 26: Visualisation du fonctionnement de la RAM

La RAM va être organisée en blocs de 48 bits divisés en deux sous-blocs de 24 bits pour les deux canaux. Chaque bloc contiendra donc un échantillon. La taille de la mémoire RAM utilisée sera de 4800 blocs de 48 bits, pour faire un délai de 100ms puisque la fréquence d'échantillonnage fe est de 48kHz. L'organisation de la mémoire est décrite ci-dessous :

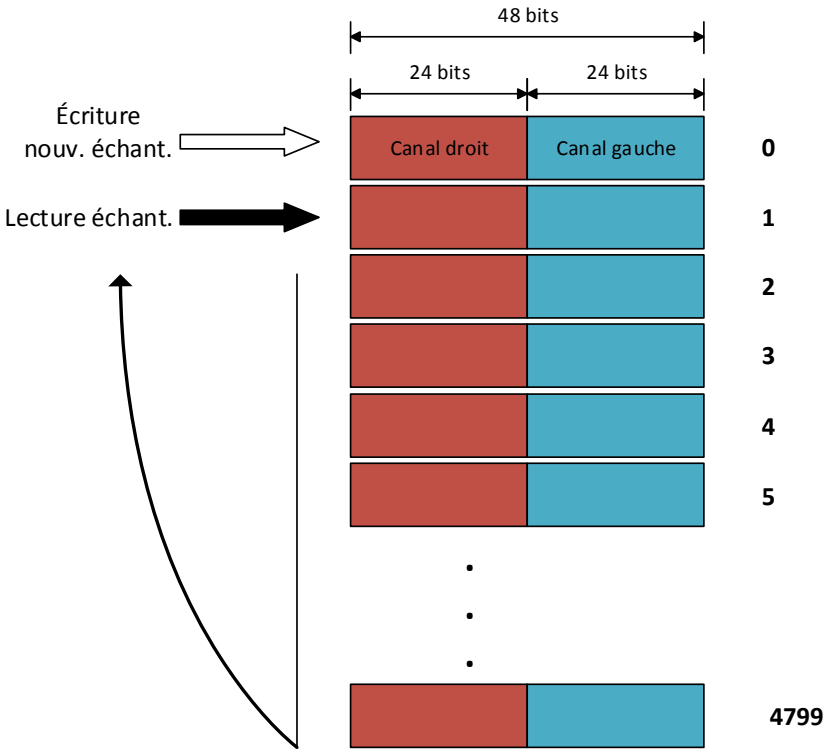


Figure 27: Visualisation du fonctionnement de la RAM - Deux canaux

Considérons que la mémoire est initialisée à 0 et que l'on démarre notre délai. Lorsqu'un nouvel échantillon est réceptionné par le récepteur I²S, on va commencer à stocker le nouvel échantillon à l'adresse 0.

Il faut aussi sortir un échantillon du bloc « délai », on va donc lire l'échantillon à l'adresse suivante, ici l'adresse 1. Lorsque l'échantillon suivant est réceptionné, on va écrire à l'adresse 1 et lire à l'adresse 2, et ainsi de suite jusqu'à la lecture de l'échantillon à l'adresse 4799.

Lorsqu'un nouvel échantillon est encore réceptionné, nous allons lire de nouveau l'échantillon à l'adresse 0, échantillon qui avait été stocké 4800 échantillons avant le nouvel échantillon réceptionné, c'est-à-dire il y a $4800 \cdot \frac{1}{fe} = 100\ ms$! Nous avons donc bien créé notre délai de 100ms sur la lecture des échantillons. Ensuite au prochain échantillon nous réécrirons à l'adresse 0, écrasant ainsi l'ancien échantillon réceptionné il y a plus de 100ms.

Notons que la RAM utilisée est une RAM « single port » : il ne sera pas possible d'écrire et de lire simultanément. Ce type de RAM permet l'utilisation de blocs de 48 bits.

5 Interfaçage du module de transmission de données

Cette partie permet de mettre en avant les différentes architectures de transmission de données implémentées. Deux architectures de transmission ont été développées pour ce projet. Chacune de ces architectures ont été développée pour pouvoir travailler en parallèle sur les deux codecs présents sur la carte Mikroelectronika et le kit DE2-70.

Ces deux codecs ne fonctionnant pas de la même manière, les architectures de communication seront également différentes. Une explication des différences entre les deux codecs est cependant utile pour la suite du rapport.

5.1 Explication générale d'une transmission de données

Le codec WM8731 dispose de quatre modes de communication bien distincts dans le cadre d'une communication pour envoyer et recevoir les données audio digitalisées. Ces modes de communication permettent un interfaçage entre le codec et un module externe, ce qui permettra au son d'être traité de manière digitale.

La chaine de transmission est la suivante :

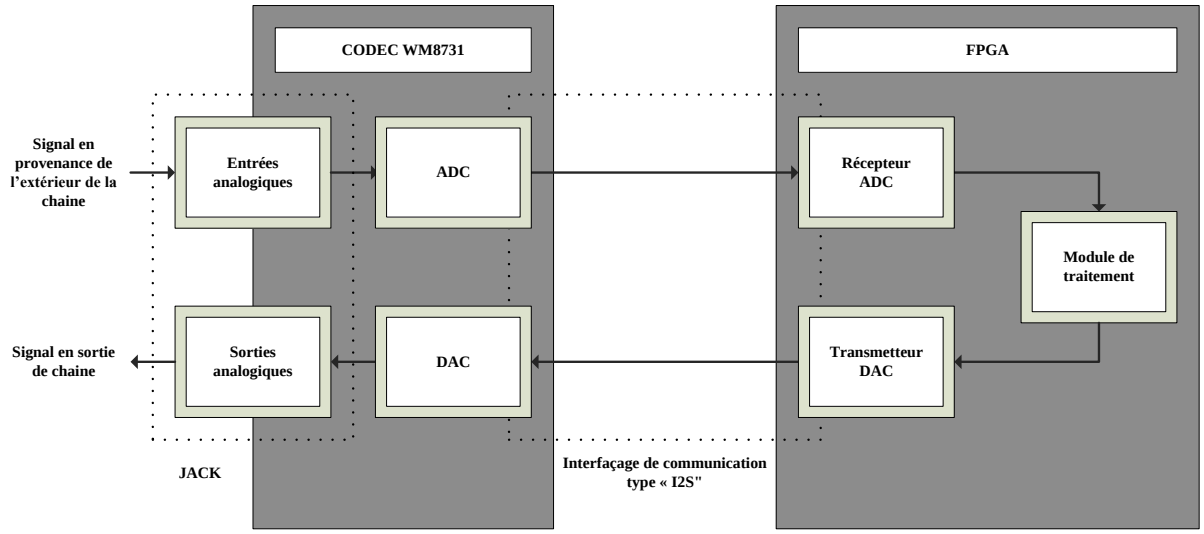


Figure 28: Interfaçage codec/FPGA

Le signal pénètre au sein de la chaine de communication par les entrées jacks mises à disposition sur les cartes de développement. Le signal passe ensuite au travers une série de blocs internes au codec. Ces blocs ont deux objectifs pour la réalisation d'une communication digitale :

- Rediriger le signal en fonction des commandes interprétées par le codec
- Réaliser un premier filtrage pour le bloc ADC

Le bloc ADC s'occupe ensuite d'échantillonner le signal analogique en signal digital suivant un certain format. Ce format est imposé en fonction de la commande envoyée préalablement au codec. L'annexe « *Annexe - Mode d'emploi WM873.docx* » permet de mettre en évidence les différents registres de commande.Le format d'échantillonnage permet de gérer à la fois la fréquence d'échantillonnage du bloc ADC mais également le nombre de bits sur lequel les données vont être encodées.

Les données échantillonnées sont ensuite transmisses vers le module de traitement ou de calcul se trouvant au sein de la FPGA au travers une interface de communication. En fonction de la carte de développement, cette interface sera filaire ou non.

Les données envoyées sont reçues au niveau d'un récepteur ADC de communication. Le récepteur ADC permet de correctement traduire les données entrantes pour les rediriger vers le module de calcul de la FPGA.

Après traitement, les données sont envoyées vers le transmetteur DAC qui s'occupera de replacer correctement les données pour gérer la communication. Les données sont pour finir retransmisses vers le DAC du codec qui aura comme objectif de repasser les données digitalisées sous un format analogique.

Comme pour le bloc ADC, le bloc DAC dispose de commande pour paramétrer le format des données à convertir, toujours en fonction de la fréquence d'échantillonnage et du format de données.

En fonction du mode de communication choisi, les blocs de réception ADC et de transmission DAC implémentés au sein de la FPGA seront différents.

5.2 Explication des différents modes de communication

➢ Introduction aux modes de communication

Comme déjà explicité, il est possible d'interfacer le codec WM8731 avec un module externe à l'aide de quatre types de communication distincts. Ces quatre modes de communication fonctionnent sous le même format d'interfaçage. Cela signifie que le câblage nécessaire pour les quatre modes de communication sera les mêmes.

Pour établir une communication, cinq signaux sont nécessaires. En voici la liste et leur description :

- ***BCLK*** ou *« **B**it **C**lo**CK** »*:

Signal d'horloge de contrôle de données. En fonction de la communication utilisée, un flan montant ou descendant de cette horloge correspond à la disponibilité d'une donnée.

- ***ADCLRC***ou*« **ADC L**eft/**R**ight **C**hannel »*:

Signal d'horloge de contrôle de l'ADC. Ce signal d'horloge permet de cadencer correctement l'échantillonnage des données en provenance du codec. La fréquence de ce signal correspond donc à la fréquence d'échantillonnage du bloc ADC du codec WM8731.

En outre, en fonction de la communication utilisée, un niveau haut ou bas de cette horloge correspond au canal gauche ou au canal droit pour la transmission des données.

- ***DACLRC*** ou « ***DAC Left/Right Channel*** » :

Signal d'horloge de contrôle du DAC. Ce signal d'horloge permet de cadencer correctement l'échantillonnage des données en provenance du codec. La fréquence de ce signal correspond donc à la fréquence d'échantillonnage du bloc DAC du codec WM8731.

En outre, en fonction de la communication utilisée, un niveau haut ou bas de cette horloge correspond au canal gauche ou au canal droit pour la transmission des données.

- ***ADCDAT*** ou « ***ADCDATa*** » :

Ce signal correspond aux données transmisses par le bloc ADC du codec et reçues par le récepteur ADC de l'architecture FPGA. Le nombre de données présentes sur ce signal dépendra du nombre de bit sur lequel les données ont été codées.

- ***DACDAT*** ou « ***DACDATa*** » :

Ce signal correspond aux données transmisses par le transmetteur DAC de l'architecture FPGA et reçues par le bloc DAC du codec. Le nombre de données présentes sur ce signal dépendra du nombre de bit sur lequel les données ont été codées.

En fonction de la manière dont est utilisé le codec, soit en mode « Master », soit en mode « Slave », le sens des signaux sera différent.

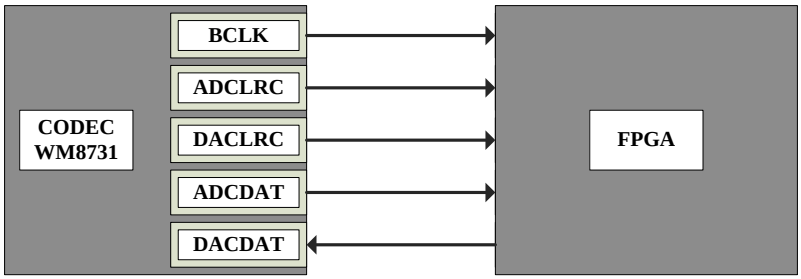


Figure 29: des signaux de contrôle en mode "Master"

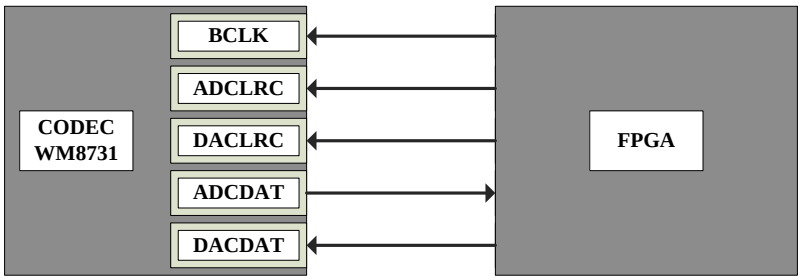


Figure 30: des signaux de contrôle en mode "Slave"

Dans le cadre de ce projet, les deux manières d'interfacer le codec ont été réalisées. En effet, le codec disponible sur la carte de développement FPGA DE2-70 ne disposait pas d'un oscillateur externe comme pour celui de la carte de développement Mikroelectronika.

Le codec présent sur la carte FPGA ne peut donc pas fonctionner en temps que « Master » mais uniquement en mode « Slave ». L'architecture de communication entre le codec et la FPGA doit donc disposer d'un générateur d'horloges pour contrôler totalement la communication et la « Master Clock » du codec Wolfson.

➢ Explication des modes de communication

Les quatre modes de communication du codec WM8731 avec un module externe sont expliqués ci-dessous à l'aide du schéma représentatif des trames. A savoir que chacun de ces modes de communication sont dit « MSB First » c'est-à-dire, que le MSB d'une donnée échantillonnée sera toujours la donnée envoyée en premier lors d'une communication.

Les schémas explicatifs représentent les trames de communication par défaut. Certains paramètres concernant les différents signaux peuvent être modifiés à l'aide de commande à envoyer au codec.

- ***MSB LeftJustified*** :

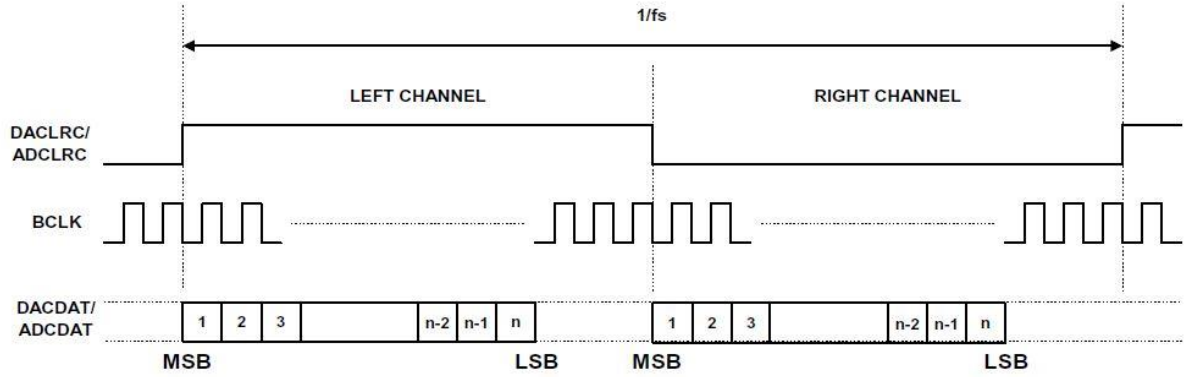


Figure 31: Communication MSB LeftJustified

Le « MSB LeftJustified »est le mode de communication le plus simple et le plus basique à comprendre. «LeftJustified » signifie que les données seront disponibles au premier flanc montant de BCLK suivant une transition des horloges ADCLRC ou DACLRC, respectivement en transmission ou réception de données de la part du codec.

Chaque coup d'horloge de BCLK correspond à une donnée échantillonnée, du MSB au LSB. Pour cette communication, un niveau haut sur ADCLRC ou DACLRC correspond aux données disponibles pour le canal audio gauche et inversement pour le canal audio droit.

Niveau	Canal
1	Gauche
0	Droit

Le signal BCLK est dimensionné en fonction de la fréquence d'échantillonnage du bloc ADC et du bloc DAC du codec et est donc fonction de la fréquence des signaux ADCLRC et DACLRC. Le signal BCLK est également dépendant du nombre d'échantillons ou bien du nombre de bits sur lequel les données sont échantillonnées.

Pour mieux comprendre comment le signal BCLK est dimensionné, prenons l'exemple suivant :

Donnée	Valeur
Fréquence d'échantillonnage [kHz]	48
Nombre de bits d'échantillonnage	16

Le signal BCLK doit être dimensionné au minimum pour permettre la gestion des données échantillonnées sur 16 bits. Un flanc montant de BCLK correspondant à une donnée échantillonnée, 16 flancs montants doivent donc être présents pour échantillonner complètement une donnée.

Il faut cependant prendre en compte la présence de deux canaux audio. Les données échantillonnées gauche et droite sont identiques mais cela représentant donc le double de données à faire transiter.

Ainsi, 16 données échantillonnées seront présentes pour le canal gauche, de même que pour le canal droit. Cela représente donc 32 données au total sur une seule période des signaux ADCLRC et DACLRC. 32 flancs montants du signal BCLK doivent donc être présents au sein d'une période des signaux ADCLRC et DACLRC pour pouvoir transiter les 32 bits de données.

La formule suivante permet de calculer la fréquence minimale du signal BCLK :

$$F_{BCLK} = 2 * Bit_{Ech} * F_{LRC}$$

Dans notre exemple nous avons donc :

Donnée	Valeur
Fréquence d'échantillonnage [kHz]	48
Nombre de bits d'échantillonnage	16
Fréquence minimale du signal BCLK [MHz]	***1.536***

Si le codec est configuré en tant que « Master », les différentes horloges de contrôle seront générées en sortie du codec. Dans ce cas-là, le signal BCLK sera surdimensionnée par rapport au nombre de bits sur lequel les données sont échantillonnées. Ainsi, plus aucune donnée ne transitera pendant un cours instant.

Si dans le cas contraire le codec est configuré en mode « Slave », libre choix à l'utilisateur de configurer un signal BCLK avec la fréquence minimale ou bien une fréquence surdimensionnée.

- ***I2S*** :

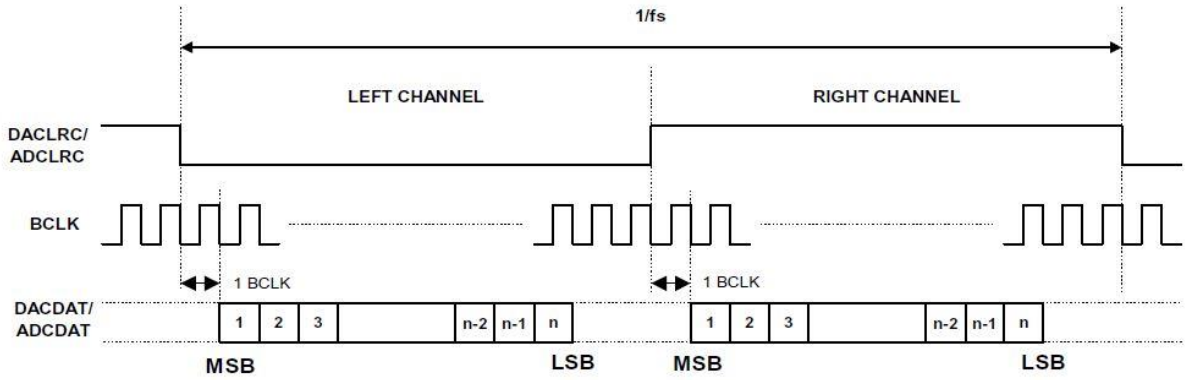


Figure 32: Communication I2S

Le mode « I2S » est une version plus standardisée du mode basique « MSB LeftJustified ». L'I2S (IntergratedIntership Sound) est un protocole de communication standard pour le transfert de données audio. Ce mode de communication est à peu près identique au mode de communication précédent à l'exception de quelques paramètres.

Comme montré sur la figure, les données sont toujours disponibles sur un flanc montant du signal BCLK. Le MSB est cette fois-ci disponible non plus directement après une transition des signaux LRC mais bien au 2ème flanc montant après une transition des signaux LRC.

Ce coup d'horloge de latence permet aux signaux LRC de correctement s'établir pour permettre une communication optimale.

Autre changement, les niveaux par défaut des signaux LRC pour la gestion des canaux gauche et droit. Un niveau haut des signaux LRC correspond cette fois-ci au canal droit et invesément pour le niveau bas.

Niveau	Canal
1	Droit
0	Gauche

Le canal gauche est cependant toujours le premier lors de la communication. Les signaux LRC débutent donc sur un niveau bas, ce qui permet une synchronisation parfaite entre eux et le signal BCLK.

Le dimensionnement du signal BCLK s'effectue de la même manière que pour la communication « MSB LeftJustified ». A noter que dans le cadre d'une configuration en mode « Slave », si le signal BCLK est dimensionné à la fréquence minimale, les données échantillonnées ne seront plus unilatéralement correspondantes aux canaux gauche et droit des signaux LRC.

Cet effet est dû à la latence produite par la disponibilité des données échantillonnées au 2ème flanc montant du signal BCLK. Ainsi, les données seront décalées d'un coup d'horloge du signal BCLK, le LSB étant décalé vers la demi-période suivante des signaux LRC.

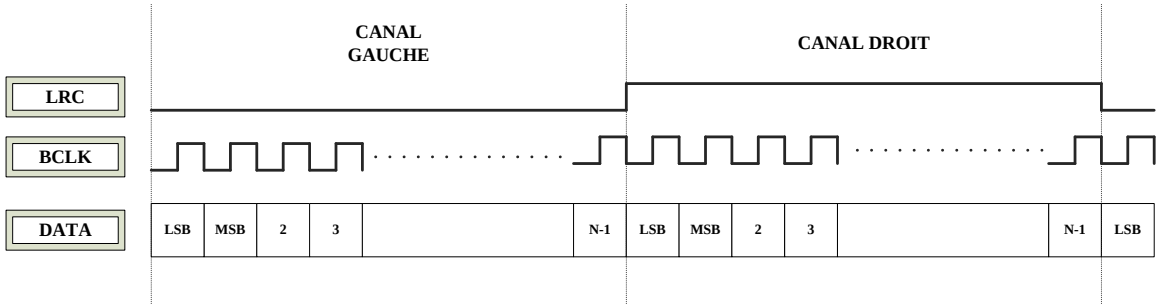


Figure 33: Communication I2S - Fréquence BCLK minimum

En fonction de la manière dont les données ont été échantillonnées (de 16 à 32 bits), une perte d'information plus ou moins négative sera perçue en sortie de chaine.Pour éviter cet effet, le signal BCLK doit être surdimensionné pour permettre aux données de transiter correctement sur le canal convenu.

- ***MSB Right Justified*** :

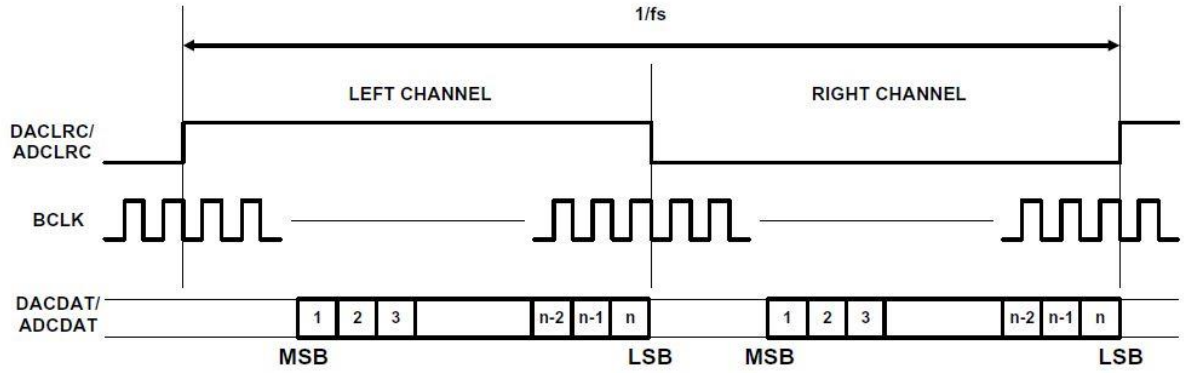


Figure 34: Communication MSB Right Justified

Ce mode de communication est identique au mode de communication « MSB LeftJustified », la différence étant la disponibilité des données échantillonnée qui cette fois-ci dépendent du LSB et non du MSB.

Le LSB d'une donnée transmisse est disponible au dernier flanc montant du signal BCLK juste avec une transition des signaux LRC.

Le dimensionnement du signal BCLK s'effectue de la même manière que pour la communication « MSB LeftJustified ». Par défaut, les signaux LRC sont également configurés comme pour la communication « MSB LeftJustified ».

La communication « MSB Right Justified » ne diffère de la communication « MSB LeftJustified » que si le signal BCL est surdimensionné. Si ce signal est dimensionné à la fréquence minimale, les trames des deux types de communication seront identiques.

- ***DSP*** :

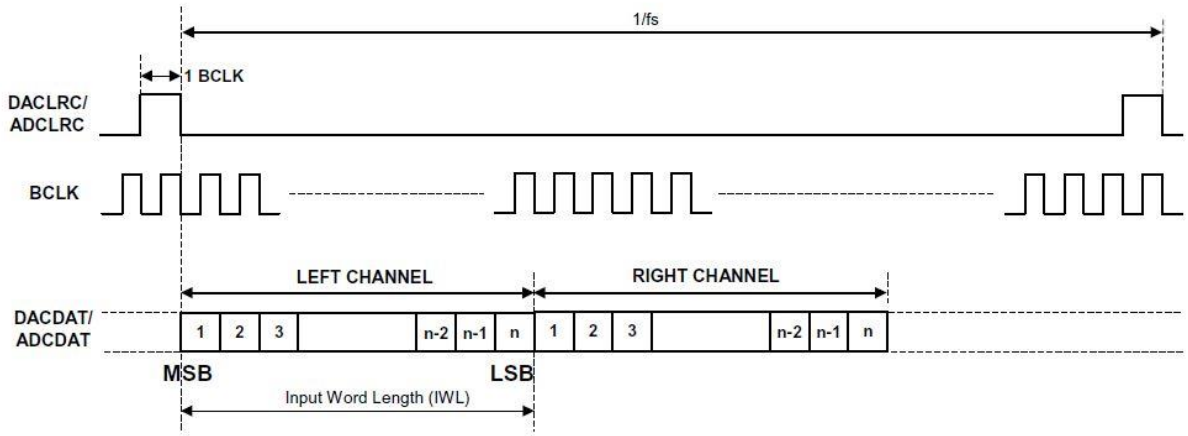


Figure 35: Communication DSP

Le mode de communication « DSP » est celui qui diffère le plus des trois modes de communications précédents. Un coup d'horloge du signal BCLK fait toujours correspondre à une donnée échantillonnée.

Le MSB est ici disponible au premier ou au second flanc montant (configurable) du signal BCLK suivant un flan descendant des signaux LRC. Les données correspondantes à chaque canal transitent ensuite directement l'une après l'autre, le canal gauche toujours en premier.

Les signaux LRC ne servent donc plus ici pour permettre de différentier quelle donnée doit transiter en fonction du canal convenu mais sert de « START » pour la transmission des données.

Les signaux LRC doivent rester à un niveau haut pendant une période complète du signal BCLK. Le niveau bas des signaux LRC est quant à lui dimensionné en fonction de la fréquence d'échantillonnage des données.

Le dimensionnement du signal BCLK est quant à lui toujours dépendant de la fréquence d'échantillonnage et du nombre de bits sur lequel sont codées les données. La fréquence du signal BCLK peut toujours être surdimensionnée ou correspondre à la fréquence minimale.

Les deux modes de communication implémentés pour le projet sont les modes de communication « MSB LeftJustified » et « I2S ». Les codes en VHDL de ces deux communications seront mis à disposition en annexe de ce rapport de projet.

A noter que l'architecture de communication pour le « MSB LeftJustified » n'est totalement opérationnelle et demandera donc un travail supplémentaire à réaliser. L'architecture « I2S » est quant à elle totalement utilisable pour la suite du projet et permettra d'utiliser sans encombre le codec WM8731 sur le kit de développement Mikroelectronika et celui sur la carte de développement dimensionnée et décrite dans ce rapport.

➢ Registre de commande

***Note** : L'ensemble des registres de contrôle seront complètement explicités au sein de l'annexe « Annexe - Mode d'emploi WM8731.docx ». Se place ici une brève explication sur les registres important à configurer pour la commande du codec en digital.*

Pour pouvoir contrôler le codec et l'utiliser avec un module externe de traitement de données, il faut préalablement le configurer en fonction de la communication.Un certain nombre de registre sont à utiliser pour permettre au codec d'échantillonner des données et de pouvoir communiquer avec un module externe.

Les bits à modifier par registre seront surlignés en « *vert* » avec leur valeur de configuration. Une explication sera ensuite donnée sur la modification du bit en cours. :

- *Le registre « **Analog Path** »* :

Bit	7	6	5	4	3	2	1	0
Name	SIDEATT1	SIDEATT0	SIDETONE	DACSEL	BYPASS	INSEL	MUTEMIC	MICBOOST
Default	0	0	0	0	1	0	1	0
Change	-	-	-	1	-	1 ou 0	-	-

Bit	**Valeur**	**Description**
DACSEL	-	-
	1	**Permet de connecter le bloc DAC du codec à ses sorties audio**
INSEL	**0**	**Permet de connecter l'entrée LINE IN au bloc ADC du codec**
	1	**Permet de connecter l'entrée MIC IN au bloc ADC du codec**

Le bit « DACSEL » est le bit le plus important à sélectionner pour ce registre. Le bit « INSEL » sera configuré en fonction de l'entrée à utiliser avec le codec.

- *Le registre « **Digital Path** »* :

Bit	7	6	5	4	3	2	1	0
Name	X	X	X	HPOR	DACMU	DEEMP1	DEEMP0	ADCHPD
Default	x	x	x	0	1	0	0	0
Change	-	-	-	-	1 ou 0	-	-	-

Bit	**Valeur**	**Description**
DACMU	**0**	**Désactive le bloc MUTE pour le bloc DAC du codec**
	1	**Active le bloc MUTE pour le bloc DAC du codec**

Le bit « DACMU » permettra d'activer ou non un « soft mute » pour le bloc DAC du codec. Ce bit est à sélectionner pour permettre soit d'activer ou non ce « soft mute », permettant au bloc DAC d'être toujours activé mais de ne plus rien sortie comme donnée.

La sélection de bit est également envisageable lorsqu'une commande pour passer d'un état à un autre pour le codec est utilisée (par exemple, passer du mode analogique au mode digital et inversement). Cela permet d'éviter des « pop » audio au niveau des sorties audio du codec.

- *Le registre « **Power Down Control** »* :

Bit	7	6	5	4	3	2	1	0
Name	POWEROFF	CLKOUTPD	OSCPD	OUTPD	DACPD	ADCPD	MICPD	LINEINPD
Default	1	0	0	1	1	1	1	1
Change	0	0	1 ou 0	0	0	0	1 ou 0	1 ou 0

Bit	**Valeur**	**Description**
POWEROFF	**0**	**Active l'ensemble du codec**
	-	-
CLKOUTPD	**0**	**Active l'horloge en sortie du codec**
	-	-
OSCPD	**0**	**Active l'oscillateur attribué au codec**
	1	**Désactive l'oscillateur attribué au codec**
OUTPD	**0**	**Active les sorties audio du codec**
	-	-

DACPD	**0**	**Active le bloc DAC du codec**
	-	-
ADCPD	**0**	**Active le bloc ADC du codec**
	-	-
MICPD	**0**	**Active l'entrée « MIC IN » du codec**
	1	**Désactive l'entrée « MIC IN » du codec**
LINEINPD	**0**	**Active l'entrée « LINE IN » du codec**
	1	**Désactive l'entrée « LINE IN » du codec**

Avant toute chose :

Bit du registre Power Down Control	**Description**
1	**DESACTIVATION**
0	**ACTIVATION**

Le registre ci-dessus est l'un des registres les plus importants à configurer puisqu'il permet d'alimenter chacun des blocs internes du codec WM8731. De manière logique, la description de l'activation des blocs du tableau fait ainsi sens.

Le bit « OSCPD » est à activer ou non en fonction de la présence d'un oscillateur externe au codec. Un oscillateur est présent sur la carte de développement Mikroeletronika ainsi que sur la carte dimensionné dans ce rapport. La carte de développement FPGA DE2-70 ne dispose cependant pas d'un oscillateur externe directement connecté au codec.

- *Le registre « **Digital Interface** » :*

Bit	7	6	5	4	3	2	1	0
Name	BCLKINV	MS	LRSWAP	LRP	IWL1	IWL0	FORMAT1	FORMAT0
Default	0	0	0	0	1	0	1	0
Change	1 ou 0	1 ou 0	1 ou 0	Variable	Variable	Variable	Variable	Variable

Bit	Valeur	Description
BCLKINV	**0**	**N'inverse pas le signal BCLK**
	1	**Inverse le signal BCLK**
MS	**0**	**Configure le codec en mode « Slave »**
	1	**Configure le codec en mode « Master »**
LRSWAP	**0**	**Canal gauche en premier**
	1	**Canal droit en premier**
LRP	-	**Bit de formatage en fonction du type de communication adopté. Voir annexe « *Annexe - Mode d'emploi WM8731.docx* »**
	-	
IWL1	-	**Bits de formatage des données à échantillonner. Configure le nombre de bits sur lequel les données vont être codées. Voir annexe « *Annexe - Mode d'emploi WM8731.docx* »**
	-	
IWL0	-	
	-	
FORMAT1	-	**Bits de formatage du type de communication. Configure le format de la communication à utiliser entre le codec et un module externe. Voir annexe « *Annexe - Mode d'emploi WM8731.docx* »**
	-	
FORMAT0	-	
	-	

Le registre « Digital Interface » permet de configurer en partie l'interface de communication entre le codec et le module externe de traitement. L'annexe « *Annexe - Mode d'emploi WM8731.docx* » permet de mettre des données sur les bits de contrôle « LRP », « IWL » et « FORMAT ».

En fonction du bit « MS », les bits « BCLKINV » et « LRSWAP » permettent au codec de lui-même inverser le signal BCLK ou de changer l'ordre des canaux des signaux LRC (« Master » uniquement).

- *Le registre « **Sampling Control** » :*

Bit	7	6	5	4	3	2	1	0
Name	CLKODIV2	CLKIDIV2	SR3	SR2	SR1	SR0	BOSR	USB_NORM
Default	0	0	0	0	0	0	0	0
Change	1 ou 0	1 ou 0	Variable	Variable	Variable	Variable	Variable	1 ou 0

Bit	Valeur	Description
CLKODIV2	**0**	**L'horloge de sortie vaut l'horloge interne du codec**
	1	**L'horloge de sortie vaut l'horloge interne du codec/2**
CLKIDIV2	**0**	**L'horloge interne du codec vaut la « Master Clock »**
	1	**L'horloge interne du codec vaut la « Master Clock »/2**
SR3	**-**	**Bits de formatage pour les données à échantillonner. Configure le nombre de bits sur lequel les données vont être échantillonnées. Voir annexe « *Annexe - Mode d'emploi WM8731.docx* »**
	-	
SR2	**-**	
	-	
SR1	**-**	
	-	
SR0	**-**	
	-	
BOSR	**-**	**Bit de formatage pour le sur-échantillonnage des données. Voir annexe « *Annexe - Mode d'emploi WM8731.docx* »**
	-	
USB_NORM	**0**	**Active le mode NORMAL**
	1	**Active le mode USB**

Le registre « Sampling Control » travaille de pair avec le registre précédent. Il permet de finaliser la configuration du mode de communication en réglant les paramètres des signaux de l'interface. Les bits « CLKODIV2 » et « CLKIDIV2 » permettre de configurer les blocs internes du codec.

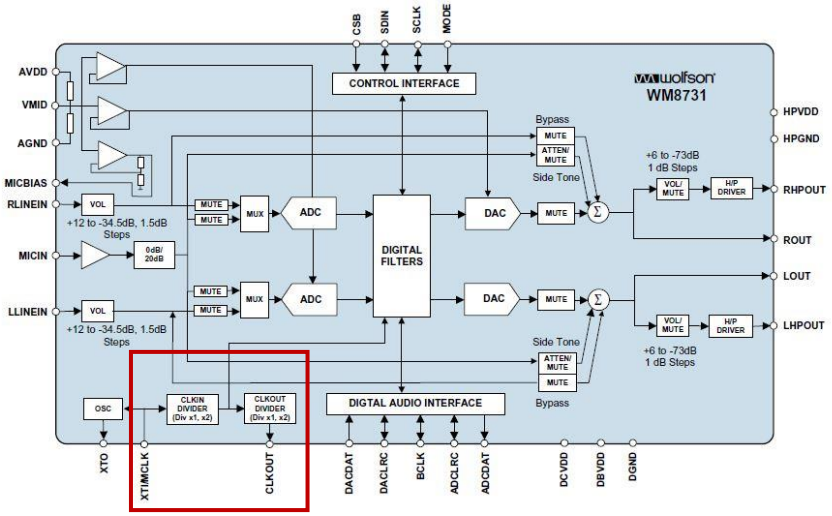


Figure 36: Bloc internes du codec - Visualisation des sorties d'horloge

Ces bits permettent ainsi de diviser les horloges de fonctionnement du codec, que cela soit la « Master Clock » dans le cadre d'une configuration en mode « Master » ou bien l'horloge interne du codec.

Les bits « SR » et « BOSR » fonctionnent ensemble pour paramétrer les fréquences des signaux LRC, soit les fréquences d'échantillonnage des blocs ADC et DAC du codec. Un exemple du fonctionnement de ces bits sera donné lors de l'explication de l'architecture FPGA pour le fonctionnement du codec en mode « Slave ».

Le bit « USB_NORM » permetquant à lui de passer du mode NORMAL au mode USB. Ce bit aura une incidence sur la fréquence atteinte par la « Master Clock » du codec lorsque ce dernier fonctionne en mode « Master ».

Le mode NORMAL permet au codec de configurer sa « Master Clock » à la fréquence de son oscillateur externe dédié (12.248 ou 18.432 MHz pour les formats audio standards). Le mode USB stabilise la fréquence de la « Master Clock » du codec à une fréquence de 12Mhz, quel que soit la fréquence de l'oscillateur externe dédié.

- *Le registre « **Active Control** »* :

Bit	7	6	5	4	3	2	1	0
Name	X	X	X	X	X	X	X	ACTIVE
Default	x	x	x	x	x	x	x	0
Change	-	-	-	-	-	-	-	1

Bit	**Valeur**	**Description**
ACTIVE	-	-
	1	**Active l'interface de communication entre le codec et un module externe**

Ce registre de contrôle permet d'activer l'interface de communication entre le codec et un module externe de traitement. A noter que l'activation ou non du bit « ACTIVE » aura certaines répercussions sur le codec lorsqu'il passe d'un mode analogique à un mode digitale (ou inversement).

Pour pouvoir passer à un mode digital, le bit « ACTIVE » doit être mis à 0. Les différents paramètres de configuration sont alors à envoyer avant de replacer le bit « ACTIVE » à 1.

6 Architecture FPGA pour la carte de développement DE2-70 – Projet DSPOB_COM

Cette partie à comme objectif d'expliquer l'architecture FPGA mise en place pour permettre de communiquer et de contrôler le codec WM8731 se trouvant sur la carte de développement FPGA DE2-70. Le codec présent sur la carte FPGA ne possédant pas d'oscillateur externe dédié, le codec n'a pas d'autre choix que d'être configurer en mode « Slave ».

Cette architecture contient les composants principaux suivant :

- Un composant de communication I2C
- Un certain nombre de composants permettant d'établir les différents signaux nécessaire à l'interfaçage codec/FPGA en mode « Slave »

6.1 Composition de l'architecture

L'architecture FPGA Top Levelest représentée à la figure 37 de la page suivante. Cette architecture comprend comme déjà explicité :

- Un composant de communication I2C permettant l'interfaçage entre le codec WM8731 de la FPGA et un microcontrôleur externe
- Un composant permettant d'établir la « Master Clock » du codec
- Un composant permettant d'établir la transmission des données échantillonnées par le codec
- Un composant GPIO permettant d'accéder aux différents signaux et de pouvoir les visualiser correctement sur oscilloscope

Composant	**Nom VHDL**
I2C	I2C_COM
« Master Clock »	PLL_MFILE
Transmission de données	DSP_BLOCK
GPIO	GPIO_COMP

Chacun des composants sera décrit en détails dans la suite de ce rapport avec une description des entités et architectures de chacun.

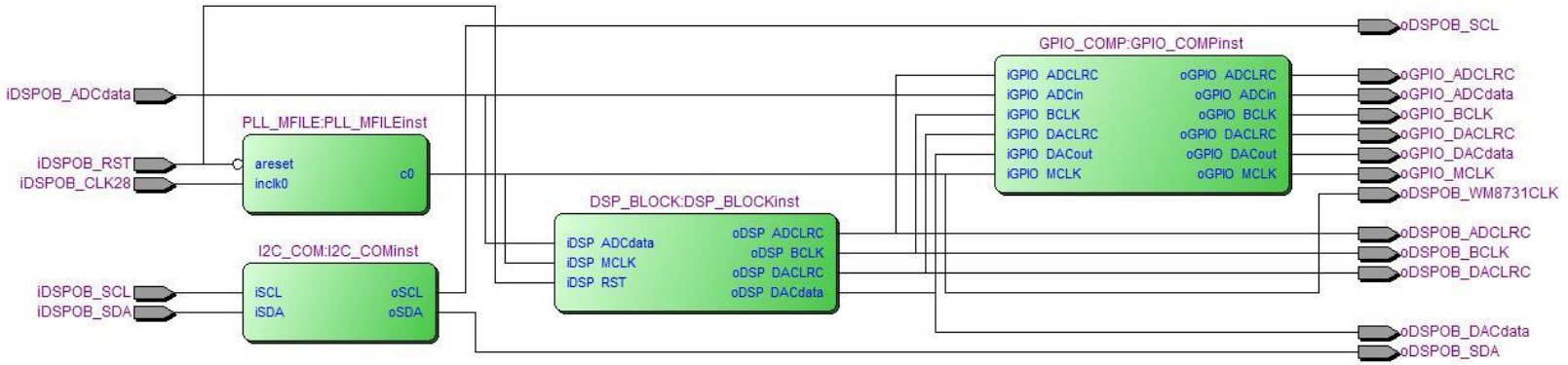


Figure 37: Vision composant de l'architecture VHDL du projet DSP_COM

6.2 Mise en place de la communication I2C – Composant I2C_COM

La première partie de l'architecture consiste à mettre en place un interfaçage filaire de contrôle du codec avec un microcontrôleur en externe de la carte de développement.

Une première interface de contrôle avait été mise en place pour pouvoir communiquer avec le codec de la carte de développement Mikroelectronika. Cette interface à donc été adaptée pour pouvoir utiliser le codec de la carte de développement DE2-70.

Voici un schéma explicatif de l'interface de contrôle µc/codec :

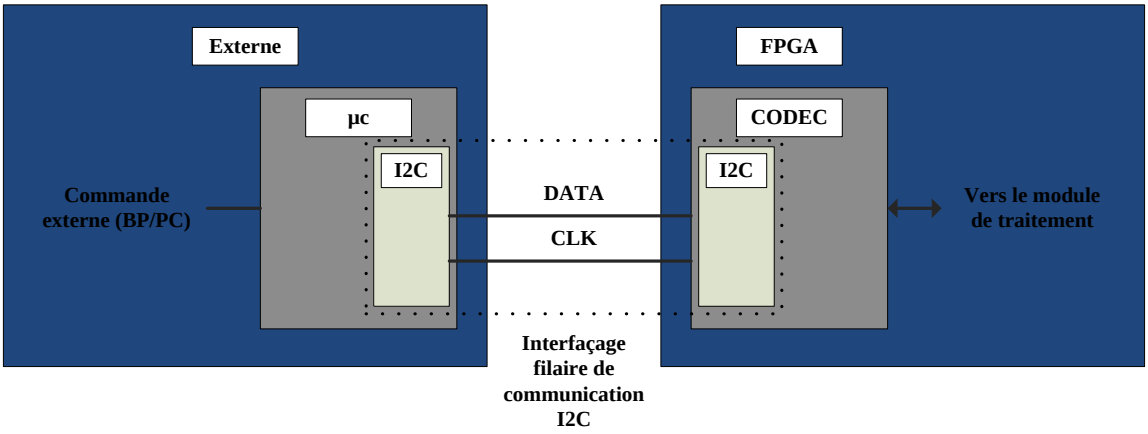


Figure 38: Interface entre le codec et le microcontrôleur de commande

Le microcontrôleur peut être commandé via deux boutons poussoirs ou bien une commande série par ordinateur.

Ainsi, le composant VHDL mis en place pour permettre la communication µc/codec ne reprend qu'un simple mapping de ports de la carte FPGA.

Les entrées et les sorties de ce composant VHDL sont les suivant :

Entité

Entrées	**Description**
iSDA	Connecté directement vers le GPIO de la FPGA. Permet de transiter les données I2C entrantes
iSCL	Connecté directement vers le GPIO de la FPGA. Permet de faire transiter l'horloge de synchronisation des données I2C

Sorties	**Description**
oSDA	Connecté directement vers le GPIO de la FPGA. Permet de transiter les données I2C vers l'interface I2C du codec WM8731
oSCL	Connecté directement vers le GPIO de la FPGA. Permet de faire transiter l'horloge de synchronisation des données I2C

6.3 Mise en place de la « Master Clock » - Composant PLL_MFILE

Le codec de la FPGA ne possédant pas d'oscillateur externe dédié, une « Master Clock » externe doit lui être injecté pour pouvoir contrôler les flux de données entrants et sortants pour l'interfaçage codec/module.

Le composant PLL_MFILE permet de générer un « Master Clock » pour le codec en utilisant un oscillateur de 28.63MHz présent sur la carte FPGA. Un composant PLL peut facilement être généré au sein d'un projet Quartus grâce à l'outil « MegaWizard Plug-In Manager »

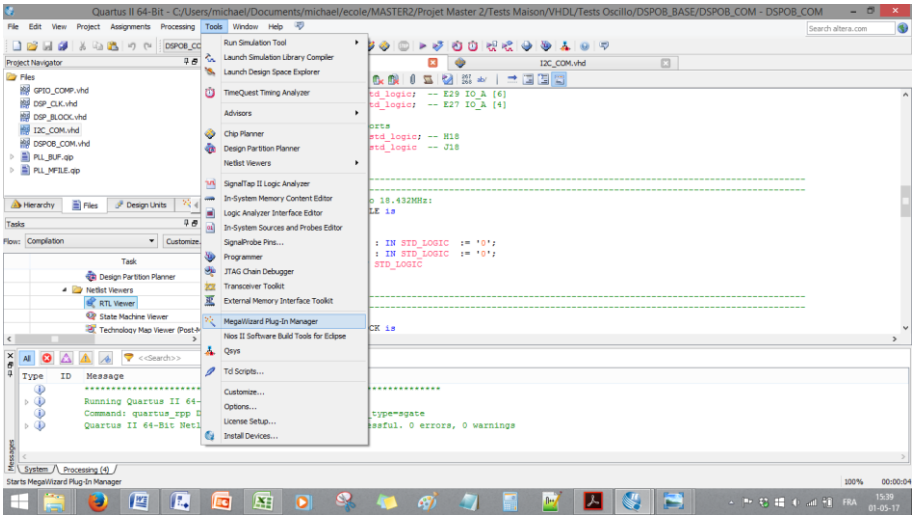


Figure 39: Outil "MegaWizard"

Cet add-on à Quartus permet de générer un certain nombre de composants génériques en VHDL. L'assistant de configuration permet de paramétrer le composant comme bon nous semble en modifiant certains paramètres au sein de l'entité et de l'architecture du composant PLL.

Entité

Entrées	**Description**
areset	Permet de reset le composant. Directement relié au reset global du Top Level
inclk0	Horloge d'entrée en provenance de l'oscillateur de 28.63MHz

Sorties	**Description**
c0	Horloge de sortie. Configuré à 18.432MHz (standard audio)

En plus de servir d'horloge de contrôle pour le codec, l'horloge générée par la PLL permettra également de générer les fréquences des différents signaux de communication (LRC et BCLK).

6.4 Mise en place des signaux de contrôle – Composant DSP_BLOCK

Le composant DSP_BLOCK est un composant MidLevel. Cela signifie qu'il contient aussi d'autres composants VHDL. Pour l'instant, le composant DSP_BLOCK ne contient que le composant suivant :

- DSP_CLK

L'objectif final est de placer au sein du composant DSP_BLOCK, les composants nécessaires pour un traitement de données complet, soit :

Composants	Description
DSP_CLK	S'occupe de générer les différents signaux de contrôle de la communication en mode « Slave »
ADC_RECV	Bloc de réception des données ADC en provenant du codec
DAC_TRSM	Bloc de transmission des données échantillonné à rediriger vers le codec
Composant de traitement	Composant permettant le traitement des données échantillonnées

Le code en annexe mettra en avant les composants non-fonctionnels ADC_TEST et DAC_TEST dans le cadre d'une transmission « MSB LeftJustified ». Cette partie du rapport ne contient ainsi que l'architecture fonctionnelle pour une communication.

En plus de contenir le composant DSP_CLK, l'architecture du composant DSP_BLOCK contient un process implicite permettant de rediriger directement les données échantillonnées par le bloc ADC du codec vers son bloc DAC. Ainsi, les signaux ADCDAT et DACDAT sont directement reliés au sein de ce composant.

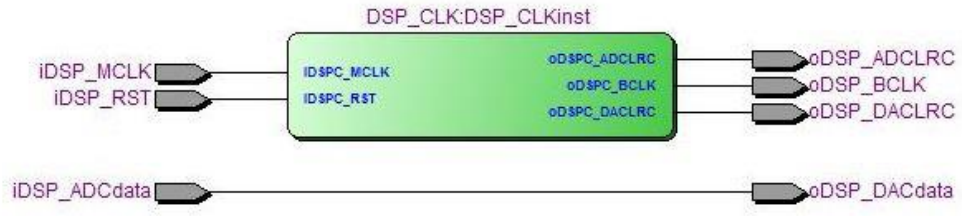


Figure 40: Architecture interne du composant DSP_CLK

L’entité du composant DSP_BLOCK est la suivante :

Entité

Entrées	**Description**
iDSP_RST	Permet de reset le composant. Directement relié au reset global du Top Level
iDSP_MCLK	« Master Clock » en provenance du composant PLL_MFILE. Permet de générer les signaux de contrôle de la communication
iDSP_ADCdata	Equivalent du signal ADCDAT. Comprend les données échantillonnées par le bloc ADC du codec

Sorties	**Description**
oDSP_BCLK	Signal de sortie BCLK généré à l’aide de la « Master Clock ». Connecté au composant DSP_CLK pour sa génération
oDSP_ADCLRC	Signal de sortie ADCLRC généré à l’aide de la « Master Clock ». Connecté au composant DSP_CLK pour sa génération
oDSP_DACLRC	Signal de sortie DACLRC généré à l’aide de la « Master Clock ». Connecté au composant DSP_CLK pour sa génération
oDSP_DACdata	Equivalent du signal DACDAT. Comprend les données échantillonnées à transmettre au bloc DAC du codec

➢ Génération des signaux de contrôle – Composant DSP_CLK

Le composant DSP_CLK est le composant le plus important de l'architecture. Ce composant permet de générer les signaux de contrôle pour une communication. Le composant a été créé dans le cadre d'une communication « MSB LeftJustified » mais peut très bien être adapté pour une communication « I2S ».

Etant donné la nature des signaux de contrôle, ce composant peut aussi être utilisé tel quel pour la communication « MSB Right Justified ».

Le composant DSP_CLK comprend l'entité suivante :

Entité

Entrées	**Description**
iDSPC_RST	Permet de reset le composant. Directement relié au reset global du MidLevel DSP_BLOCK
iDSPC_MCLK	« Master Clock » qui permettra de générer les différents signaux de contrôle

Sorties	**Description**
oDSPC_BCLK	Signal de sortie BCLK généré à l'aide de la « Master Clock »
oDSPC_ADCLRC	Signal de sortie ADCLRC généré à l'aide de la « Master Clock »
oDSPC_DACLRC	Signal de sortie DACLRC généré à l'aide de la « Master Clock »

Valeurs génériques	**Description**
midBclkCnt	Valeur générique permettant d'établir la valeur moyenne du compteur pour la génération du signal BCLK
maxBclkCnt	Valeur générique permettant d'établir la valeur maximale du compteur pour la génération du signal BCLK
midLrcCnt	Valeur générique permettant d'établir la valeur moyenne du compteur pour la génération des signaux LRC
maxLrcCnt	Valeur générique permettant d'établir la valeur maximale du compteur pour la génération des signaux LRC

Le composant DSP_CLK comprend deux process permettant chacun de générer soit le signal BCLK, soit les signaux LRC.

Pour comprendre comment les différents process fonctionnent, une lecture du code VHDL doit être réalisée en parallèle de l'explication avec exemple. Le code sera bien sûr commenté pour permettre une meilleure compréhension.

Comme déjà explicité, le composant a été créé dans le cadre de la communication « MSB LeftJustified ». Les paramètres de configuration du codec pour cette communication sont les suivants :

Paramètres	Description
Fréquence de la « Master Clock » [MHz]	18.432
Fréquence d'échantillonnage [kHz]	48
Nombre de bits de codage (bit SR)	16
Sur-échantillonnage (bit BOSR)	384

Les données ci-dessus vont permettre d'établir les valeurs des compteurs (valeurs génériques dans l'entité) pour diviser correctement la « Master Clock » et la dériver pour la génération des signaux de contrôle.

Le bit BOSR permet ici de déterminer le facteur entre la « Master Clock » et les signaux LRC. Dans notre cas, ce facteur est de 384 car la « Master Clock » est à une fréquence de 18.432MHz.

***Note** : Si nous avions généré une horloge de 12.288MHz, le bit BOSR aurait dû être configuré autrement pour permettre d'obtenir un facteur de 256 entre la « Master Clock » et les signaux LRC.*

Les bits SR permettent d'identifier le nombre de bits de codage, soit 16 bits. La communication étant « MSB LeftJustified », nous avons besoin de deux canaux de contrôle. Cela correspond donc à 32 bits au total en transfert.

Pour rappel, un coup d'horloge du signal BCLK correspond à une donnée en transite. Il faut donc au minimum 32 coups d'horloge de BCLK pour faire transiter les 32 bits de données. Ces paramètres permettent de déterminer la fréquence minimale du signal BCLK.

$$F_{BCLK} = 2 * Bit_{Ech} * F_{LRC}$$

Donnée	Valeur
Fréquence d'échantillonnage [kHz]	48
Nombre de bits d'échantillonnage	16
Fréquence minimale du signal BCLK [MHz]	***1.536***

Nous pouvons désormais calculer le rapport entre la fréquence de la « Master Clock » et la fréquence du signal BCLK :

Donnée	Valeur
Fréquence de la « Master Clock » [MHz]	18.432
Fréquence minimale du signal BCLK [MHz]	1.536
Facteur	***12***

Toutes ces données permettent de déterminer les différentes valeurs des compteurs génériques. Nous serons ainsi en mesure de générer nos signaux de contrôle :

- Pour générer le signal BCLK :

Donnée	Valeur
Facteur entre la « Master Clock » et BCLK	12
midBclkCnt	***6***
maxBclkCnt	***12***

- Pour générer les signaux LRC :

Donnée	Valeur
Facteur entre BCLK et LRC	32
midLrcCnt	***16***
maxLrcCnt	***32***

Voici un rappel des trames de transfert des signaux en communication « MSB LeftJustified » :

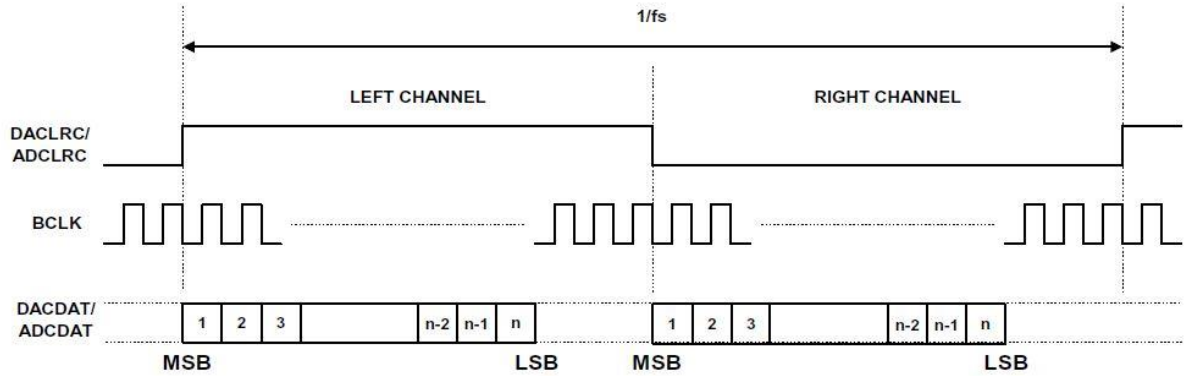


Figure 41: Communication MSB Left Justified

A l'aide du schéma des trames de la figure 41 et des valeurs génériques des compteurs, déterminons la génération des signaux de contrôle :

- Pour générer le signal BCLK :

Les signaux en VHDL suivants ont été utilisés pour la génération de BCLK :

Signaux	Description
sBCLK	Signal tampon BCLK
sBCLK_CNT	Signal compteur pour la génération de BCLK

Pendant 6 flancs montant de la « Master Clock », le signal sBCLK devra être égal à 0. Pendant 6 autres flancs montant de la « Master Clock », le signal sBCLK devra être égal à1.

Le signal sBCLK_CNT permet d'établir correctement les conditions pour le signal sBCLK en tant compris entre les valeurs génériques des compteurs. Le signal sBCLK est constamment envoyé vers la sortie oDSPC_BCLK.

- Pour générer les signaux LRC :

Les signaux en VHDL suivants ont été utilisés pour la génération de LRC :

Signaux	Description
sLRC	Signal tampon LRC
sLRC_CNT	Signal compteur pour la génération de LRC

Pendant 16 flancs montant de BCLK, le signal sLRC devra être égal à 1.
Pendant 16 autres flancs montant de BCLK, le signal sLRC devra être égal à0.

Le signal sLRC_CNT permet d'établir correctement les conditions pour le signal sLRC en tant compris entre les valeurs génériques des compteurs. Le signal sLRC est constamment envoyé vers les sorties oDSPC_ADCLRC et oDSCP_DACLRC.

Le paramétrage de ce composant est assez facile une fois le mode de communication correctement assimilé. Voici les paramétrages possibles du composant :

- *Surdimensionnement de la fréquence du signal BCLK* :

Le surdimensionnement de la fréquence du signal BCLK est assez facile à mettre en œuvre puisqu'il suffit de modifier les valeurs des compteurs génériques présents dans l'entité.

- *Modification du composant pour la communication « I2S »* :

Pour pouvoir passer du mode de communication « MSB LeftJustified » au mode de communication « I2S », le process de génération des signaux LRC doit être modifié. Le signal sLRC doit être mis à 0 ou à 1 de manière opposé à ce qui est proposé.

6.5 Visualisation des trames de communication

Voici deux exemples de trames de communication en utilisant la communication « MSB LeftJustified ». La Figure 42 représente des trames de communication avec un signal BCLK dimensionné avec la fréquence minimum :

Figure 42: Trames de communication- Fréquence BCLK minimum

On remarque ici que les données sont correctement synchronisées entre elles puisqu'aucun bloc de traitement ne vient s'insérer entre l'envoi et la réception des données échantillonnées. Chaque donnée est ainsi présente sur les deux canaux.

La figure 43 représente quant à elle la même communication mais cette fois-ci avec le signal BCLK surdimensionné par rapport à sa fréquence minimale :

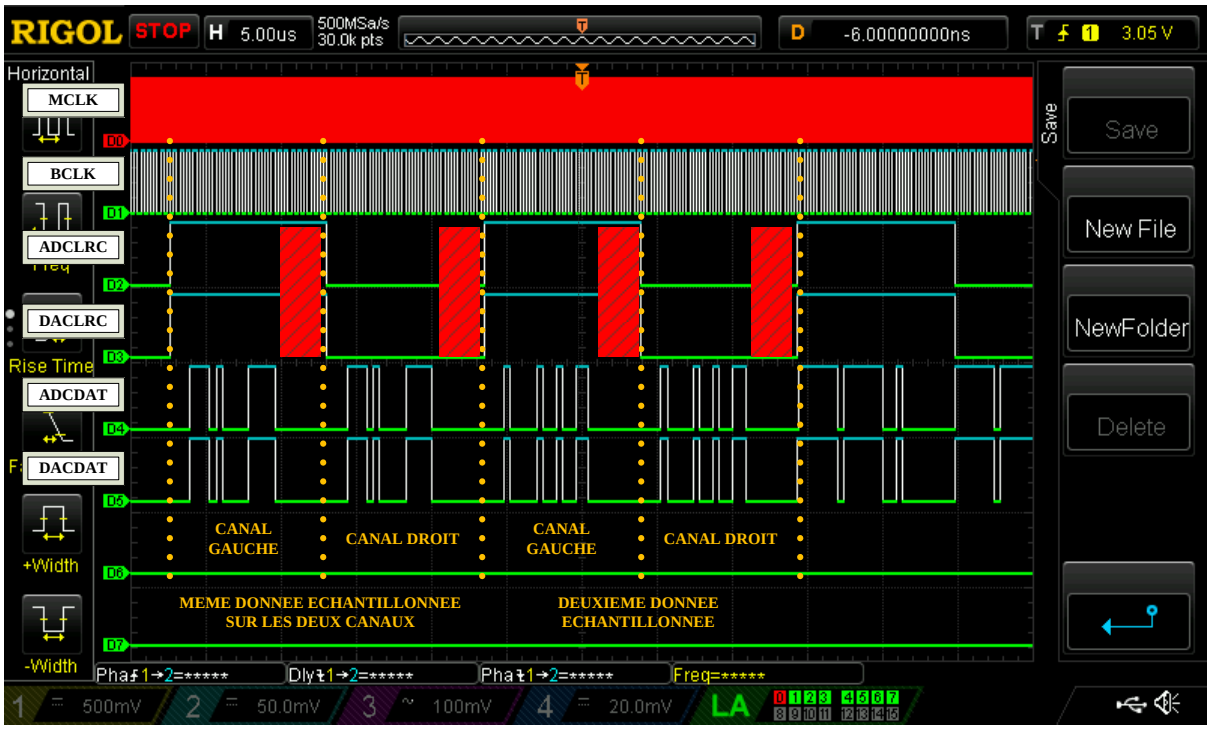


Figure 43: Trames de communication- Fréquence BCLK surdimensionnée

Les données échantillonnées sont également synchronisées correctement. Le signal BCLK est dimensionné de telle sorte à avoir une fréquence correspondante non plus à un codage des données sur 16 bits mais sur 24 bits.

$$F_{BCLK} = 2 * Bit_{Ech} * F_{LRC}$$

Donnée	Valeur
Fréquence d'échantillonnage [kHz]	48
Nombre de bits d'échantillonnage (fictif)	24
Fréquence surdimensionnée du signal BCLK [MHz]	***2.304***

24 coups d'horloge du signal BCLK sont donc présents au sein d'une demi-période des signaux LRC. On remarque ainsi sur chacun des canaux, sur 8 coups d'horloge du signal BCLK, plus aucune donnée n'est transmise par le bloc ADC du codec.

7 Conclusion et perspectives

Notre groupe de projet étant exclusivement composé d'étudiants en Master 2, notre objectif principal de travail était de mettre en place un lissage complet de la structure du projet pour permettre aux futurs étudiants de Master 1 et de Master 2 de reprendre le plus convenablement le projet DSPOB.

Ainsi, nous avons sélectionné parmi les Work Package proposés ceux permettant d'atteindre au mieux cet objectif. Par cela, les objectifs de projet suivant ont été atteint :

- Une analyse complète du codec WM8731 proposé comme étant la base du projet
- Une remise en forme du programme de commande pour un meilleur contrôle du codec
- La mise en place d'un système complet et fonctionnel de communication codec/module de traitement

Ces objectifs ont permis de parcourir un certain nombre de point concernant le projet DSPOB.

Les analyses des différentes architectures analogiques entourant le codec disposé sur les différentes cartes de développement ont permis de mettre en place la première schématique « home-made » du projet.

La mise en place des différentes architectures FPGA permet de communiquer entre le codec WM8731 et un module de traitement a permis d'établir les différents chemins à prendre et à ne pas prendre pour la suite de la partie digitale du projet.

Ces différents travaux permettent ainsi de définir au mieux les perspectives du projet DSPOB au travers un cahier des charges.

Sur le plan analogique :

- Réaliser la carte de développement conçue durant les travaux réalisés
- Mettre en place un plan de tests du nouveau kit de développement et analyser ses performances avec comparaison des analyses déjà effectuées.

Sur le plan numérique :

- Une mise en place de différents traitements numériques possibles. Cet objectif consiste à réaliser la partie la plus ludique du projet, partie que nous avons déjà commencé à implémenter.

La suite du projet s'étend donc autant sur le plan analogique que numérique au point de vue électronique, permettant ainsi au projet d'être aussi varié que possible et de s'étendre sur une grande partie du monde électronique.