

Bordeaux INP – ENSEIRB-MATMECA

Filière Systèmes Électroniques Embarqués

**PR209 – Projet expérimental de conception de circuit numérique (S8)**

Module assuré par Christophe JEGO



**Rapport d’activité**

Par

Maxime Dabrowski



Remis le 02 avril 2021

# Cahier des charges

# Module d’affichage et de contrôle

Comme préciser dans le cahier des charges, ce projet ce découpe selon 3 sous parties. Ici je présenterai la partie concernant le contrôle des états du lecteur MP3 ainsi que l’affichage sur les afficheurs 7 segments.

Voici tout d’abord un schéma RTL des blocs :

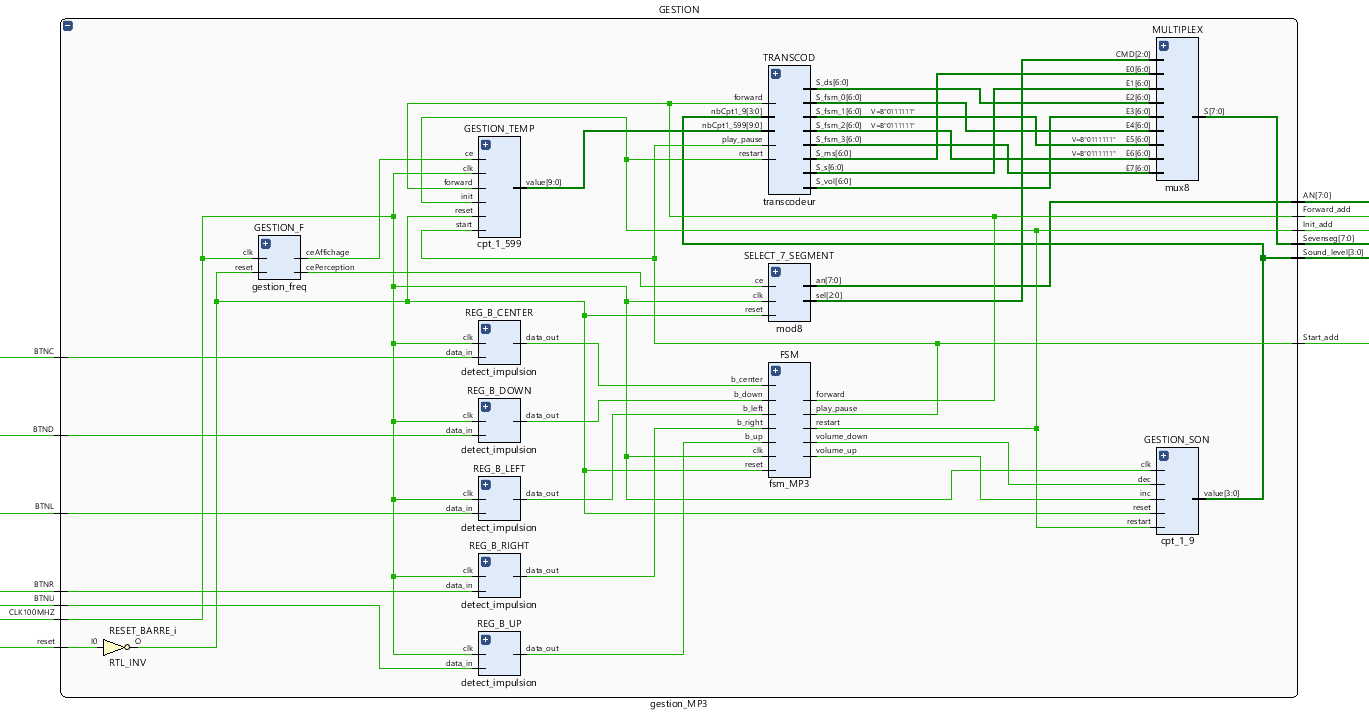


Figure 1 : Schéma RTL du module de « Gestion » du MP3

Nous avons sur cette partie 12 blocs avec une conception de 8 blocs :

* [GESTION\_F] : un synthétiseur de fréquence permettant de générer plusieurs « clock enable »
* [REG\_B\_...] : un détecteur d’impulsion permettant de simuler un front montant lors de l’appui sur un des boutons
* [GESTION\_TEMP] : un compteur allant de 1 à 599 pour compter le nombre de milliseconde qui s’écoule
* [GESTION\_SON] : un compteur allant de 1 à 9 permettant d’ajuster le niveau sonore en sortie
* [FSM] : une machine à état permettant de contrôler les différentes phases de fonctionnement
* [TRANSCOD] : un transcodeur prenant les différentes informations de cette partie pour les remettre en forme afin de les afficher sur les 7 segments
* [SELECT\_7\_SEGMENT] : un sélecteur permettant de sélectionner un des 8 afficheurs 7 segments
* [MULTIPLEX] : un multiplexeur pour sélectionner la donnée à afficher sur les 7 segments

## Détecteur d’impulsion

Voici tout d’abord le schéma RTL de ce bloc :

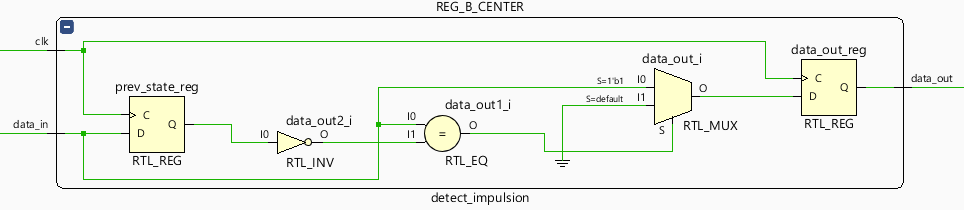


Figure 2 : Schéma RTL du « détecteur d'impulsion »

Ce bloc a pour but de détecter le front montant d’un des boutons poussoirs. La bascule d’entrée sauvegarde la donnée lors du front montant de l’horloge. Si la valeur de cette bascule est différente de l’entrée, alors nous appliquons l’inverse de cette valeur sur la sortie du bloc, sinon nous appliquons un 0 logique. Cela permet de n’avoir la sortie à 1 que sur une période d’horloge. Cela permet de simplifier la FSM puisque sans ces blocs, nous devrions ajouter des étapes intermédiaires pour simuler la détection de front montant ou descendant.

## Gestion des fréquences

Voici tout d’abord le schéma RTL de ce bloc :

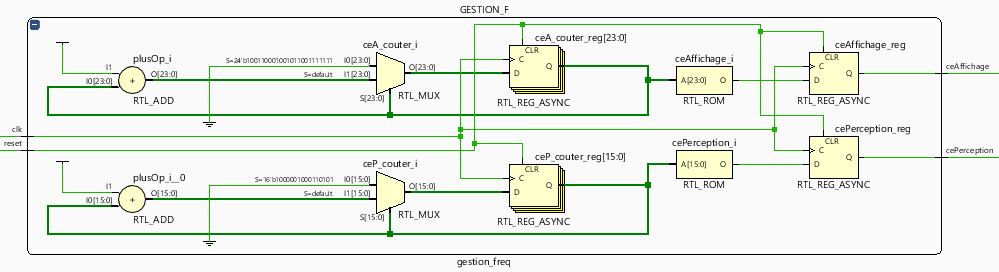


Figure 3 : Schéma RTL de la gestion de fréquence

Notre carte cible fonctionne à une fréquence de 100MHz. Ce bloc génère des « clock enable » qui permette de baisser fictivement la fréquence de fonctionnement de certains blocs. Ici, le bloc comptera le nombre de front de période d’horloge écouler en partant de 0 jusqu’à atteindre un nombre de générer une autre fréquence. Nous avons donc ici 2 « clock enable » permettant de baisser la vitesse à :

* CePerception : 3kHz qui sera utiliser pour la persistance rétienne des afficheurs 7 segments
* CeAffichage : 10Hz qui permettra d’avoir une précision à la dixième de seconde pour l’écoulement du temps de la musique.

Nous avons donc des compteurs allant jusqu’à :

* CePerception : 100 000 000 / 3 000 – 1 = 33 334
* CeAffichage : 100 000 000 / 10 – 1 = 9 999 999

## Compteur temporelle

Voici tout d’abord le schéma RTL de ce bloc :

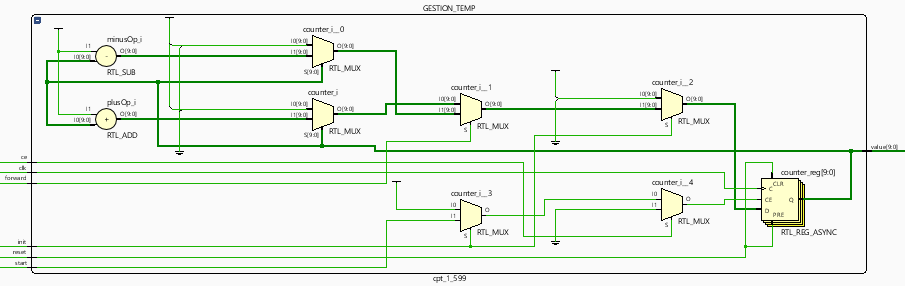


Figure 4 : Schéma RTL du « compteur de temps »

Ce compteur est un compteur permettant d’aller de 1 à 599 pour compter le temps qui s’écoule soit 60 secondes dans notre cas. Lorsqu’il arrive à sa valeur minimale, le compteur recommence à 599 et il recommence à 1 s’il arrive à sa valeur maximale en fonction du sens de comptage. Ce compteur est cadencé par l’horloge et par le CE affichage fonctionnant à 10Hz. Il fonctionne également selon certaines entrées :

* « forward » indique le sens de comptage ;
* « start » indique que le compteur doit compter à la fréquence de 10Hz dans le sens indiqué par « forward » ;
* « restart » indique au compteur qu’il doit se réinitialiser à 1 et rester dans cette configuration.

Ces entrées qui viennent toutes de la machine à état décrites plus loin dans ce rapport.

La sortie de ce compteur est codée sur 10 bits en « STD\_LOGIC\_VECTOR » pour coder la valeur maximum du compteur soit 599.

## Compteur pour le volume sonore

Voici tout d’abord le schéma RTL de ce bloc :

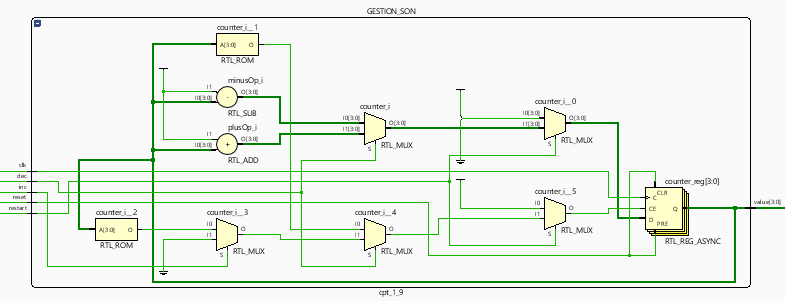


Figure 5 : Schéma RTL du « compteur pour le volume sonore »

Ce compteur est un compteur permettant d’aller de 1 à 9. Ce bloc est cadencé à la vitesse de l’horloge mais fonctionne selon les entrées :

* « inc » pour incrémenter la valeur de 1
* « dec » pour décrémenter la valeur de 1
* « restart » indique au compteur qu’il doit se réinitialiser à 1 et rester dans cette configuration.

L’entrée « restart » vient de la machine à état. Les 2 autres entrées (« inc » et « dec ») viennent de bloc détecteur d’impulsion. Les boutons poussoirs doivent forcément passé par ces blocs car le compteur fonctionnant à 100MHz, il est impossible pour un humain de n’appuyer sur un bouton poussoir que 1ns.

La sortie de ce compteur est codée sur 4 bits en « STD\_LOGIC\_VECTOR » pour coder la valeur maximum du compteur soit 9.

## Machine à état fini

Voici tout d’abord le schéma RTL de ce bloc :

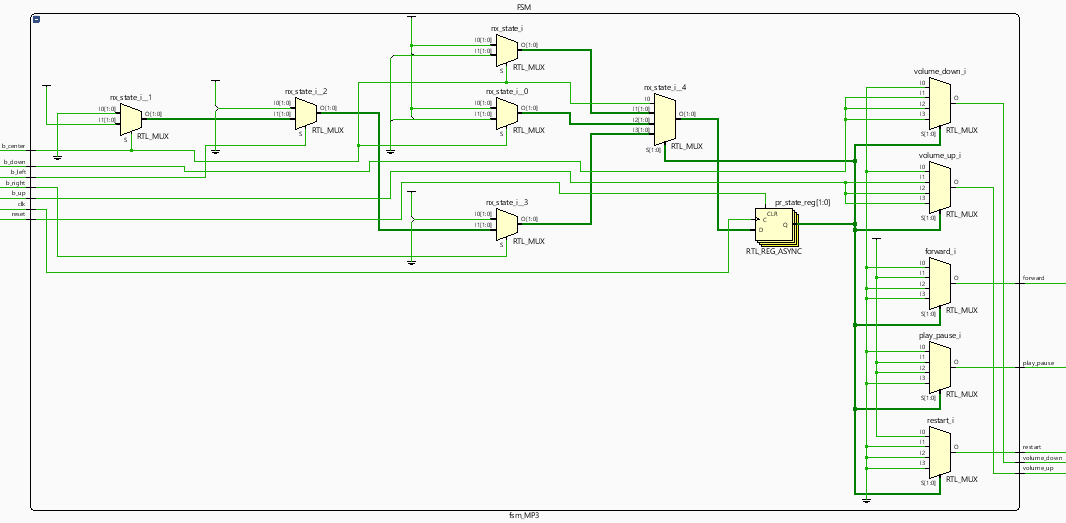


Figure 6 : Schéma RTL de la « Machine à état »

Cette description correspond à ce diagramme de machine à état :



## Transcodeur

Voici tout d’abord le schéma RTL de ce bloc :

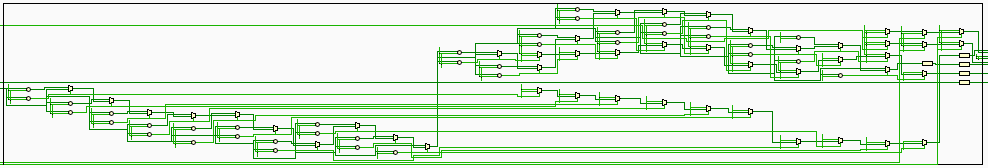


Figure 7 : Schéma RTL du « Transcodeur »

Ce bloc est combinatoire et permet de préparer les données en entrées de manière à les afficher sur les 7 segments.

Voici les entrées :

* « play\_pause » : indique si le lecteur est en pause (à 0) ou en lecture (à 1) ;
* « restart » : indique si le lecteur est à l’état init ;
* « forward » : indique le sens de lecture ;
* « nbCpt1\_9 » : indique le niveau sonore ;
* « nbCpt1\_599 » : indique le temps écoulé depuis le début du morceaux.

Les 3 première entrées sont utilisées pour coder les 4 afficheurs 7 segments sur la gauche de la carte, le compteur de 1 à 9 est affiché sur 1 seul afficheur 7 segments et le compteur 1 à 599 est sur les 3 afficheurs restant. Il est à noté que l’affichage du niveau sonore et du temps doivent être séparé par un point sur les afficheurs.

SIMULATION

## Sélecteur

Voici tout d’abord le schéma RTL de ce bloc :

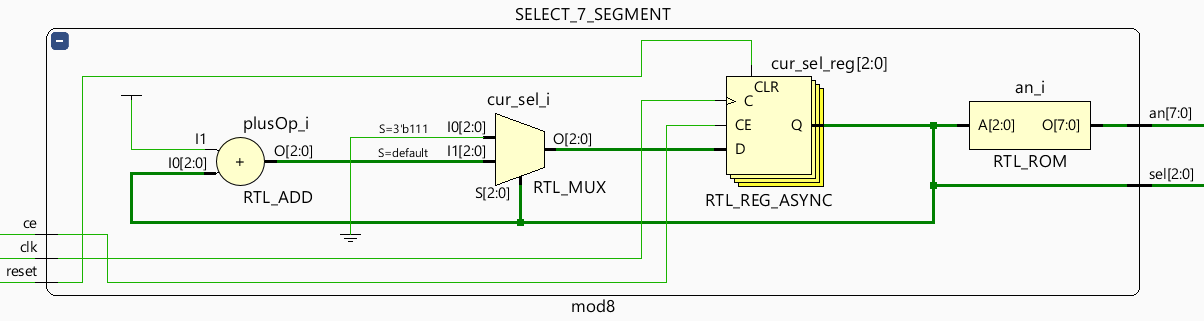


Figure 8 : Schéma RTL du « Sélecteur de 7 segments »

Ce bloque permet de sélectionner les afficheurs 7 segments en propageant un 0 (actif à l’état bas). Nous avons une sortie sur 8 bits pour commander 1 à 1 les 8 afficheurs 7 segments et une sortie « sel » qui permet de transmettre l’information de l’afficheur sélectionné au bloc multiplexeur 8 vers 1 présenté juste après.

Ce bloc fonctionne à 3kHz avec le Ce relié au « CePersistance ». En fonctionnement à cette fréquence, nous pouvons changer d’afficheurs à 3kHz, ce qui permet de bien générer la persistance générique.

## Multiplexeur 8 vers 1

Voici tout d’abord le schéma RTL de ce bloc :

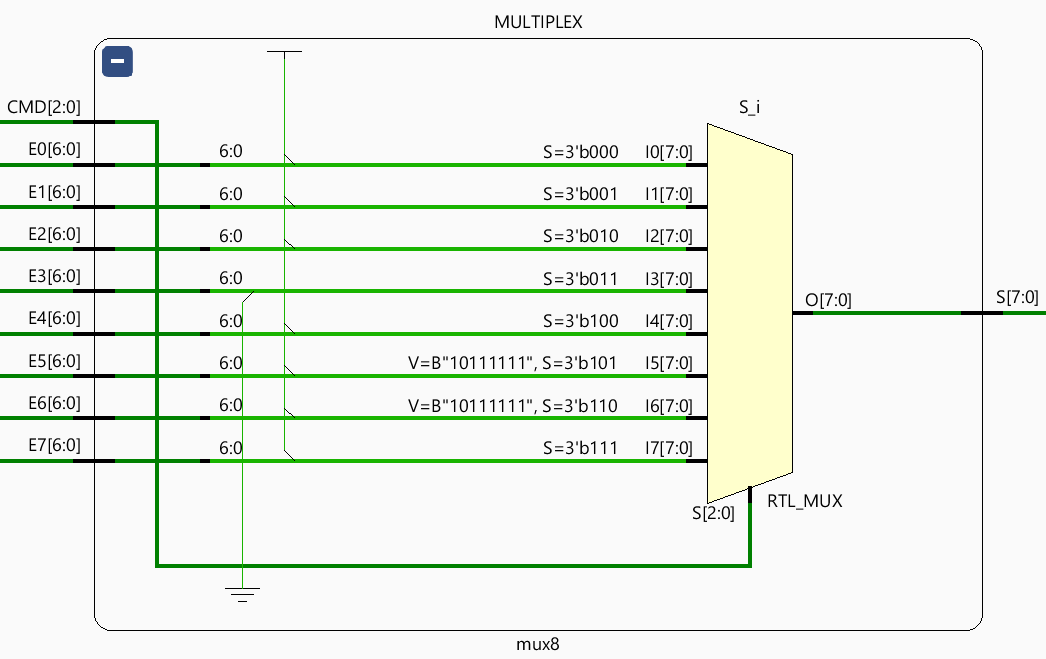


Figure 9 : Schéma RTL du « multiplexeur 8 vers 1 »

# Contrôle de l’audio et production personnelle

Comme préciser dans le cahier des charges, ce projet ce découpe selon 3 sous parties. Ici je présenterai la partie concernant le stockage et la modulation des échantillons audio.

Voici tout d’abord un schéma RTL des blocs :

## Mémoire RAM

Voici tout d’abord le schéma RTL de cette RAM :

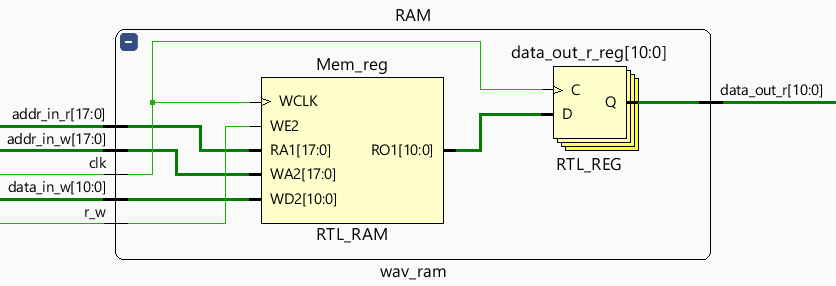


Figure 10 : Schéma RTL de la « RAM »

Cette RAM possède des emplacements mémoire de 11 bits sur 2^18 bits d’adressages soit 262 144 valeurs ce qui représente un total de 88 blocs BRAM sur la carte NEXYS A7. Les valeurs stockées sont signées mais sont transmises aux différents blocs sous forme de « STD\_LOGIC\_VECTOR ». Cette mémoire est remplie à partir d’une liaison série. Elle a été conçue sous forme de mémoire à double accès, elle peut donc être lu et rempli en même temps. Nous avons donc pour son architecture, une adresse de lecture et une adresse d’écriture en entrée du bloc. Nous avons également une entrée permettant de mettre à disposition les données devant être recopier dans la mémoire lorsqu’un signal est à 1 (« RW »). Ce dernier indique à la mémoire lorsqu’une donnée est prête à être copier (mode écriture) ou si la mémoire doit lire les différentes valeurs (mode lecture).

La mémoire est accès séquentiel (cadencé à 100MHz) et ne possède pas de « clock enable » pour accélérer son accès par rapport au reste du système. Elle est également dite à lecture prioritaire puisque peu importe l’état de « RW », la mémoire recopiera à chaque front d’horloge sa valeur stocker à l’adresse de lecture sur sa sortie.

SIMULATION

## Contrôle du son

Voici tout d’abord le schéma RTL de ce bloc :

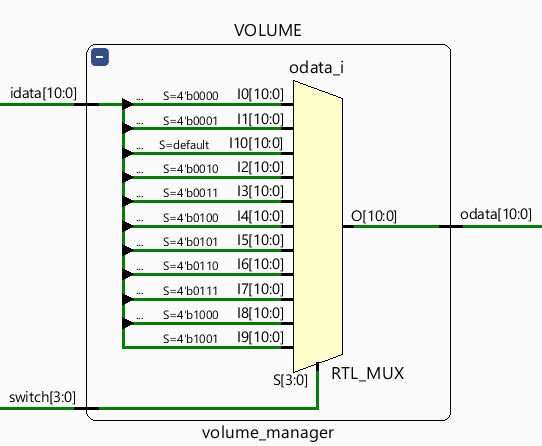


Figure 11 : Schéma RTL de « Contrôleur de volume »

Nous voyons que d’après le schéma RTL, il ne s’agirait que d’un multiplexeur mais son fonctionnement est quelque peu plus compliqué. Cette partie récupère la valeur du compteur présent dans la partie contrôle du lecteur MP3 (valeur allant de 1 à 9) sur son entrée « switch ». En fonction de cette valeur, nous effectuons une division sur les valeurs reçues de la RAM afin de diminuer l’amplitude du signal et donc de diminuer le son en sortie de la prise jack.

Les valeurs reçues sur « idata » sont de type STD\_LOGIC\_VECTOR mais représente en réalité des nombres signés sur 11 bits. Il faut donc prendre en compte le bit de signe lors de la division.

Puisque nous avons 9 niveaux de son et 11 bits pour coder l’information (bit de signe compris), nous pouvons, au lieu d’effectuer une division gourmande en ressource, réaliser un décalage de bit.

Pour ce faire, il suffit de garder tous les bits pour la valeur 9 (qui représente le niveau de son maximum) et de retirer 1 bit de poids faible à chaque diminution de la valeur pour diviser par 2 successivement. Etant donnée qu’il s’agit de nombre signé, nous ne pouvons pas juste mettre un 0 à la place des bits de poids fort mais devons propager le bit de signe pour prendre en compte les nombres négatifs.

with switch select

odata <=

idata(10)&idata(10)& idata(10 downto 2)  when "0111",

    idata(10)& idata(10 downto 1)    when "1000",

    idata   when "1001",

    idata(10 downto 4)&idata(10)&idata(10)&idata(10)&idata(10) when OTHERS;

Nous avons ci-dessus une partie du code utilisé. Pour la valeur maximum (« 1001 » soit 9), nous affectons à la valeur de sortie toute la plage d’entrée. Cependant, lorsque le compteur vaut 8 (« 1000 »), nous ne prenons que les 10 bits de fort en supprimant le bit de faible. Pour avoir toujours une valeur sur 11 bits et toujours signé, nous remettons un bit de poids fort du signe du nombre d’origine devant la nouvelle valeur.

Le même calcul est effectué pour les autres valeurs mais en diminuant toujours le nombre de bit de poids fort pris sur la valeur de départ. Voici une simulation présentant le fonctionnement :

SIMULATION

## Principe de l’augmentation et diminution de la vitesse de lecture

Cette partie concerne l’ajout personnel à ce projet. Le but ici est de pouvoir lire 2 fois plus rapidement ou plus lentement le même échantillon audio. Il ne s’agit pas uniquement d’un bloc mais d’une fonction agissant sur plusieurs blocs du contrôle audio. Voici les blocs impactés en fonction de la vitesse de lecture souhaité :

* Modification de la valeur des échantillons (via un nouveau bloc spécifique)
* Modification de la fréquence d’échantillonnage
* Modification du nombre de période nécessaire pour la PWM

Ces 3 blocs seront décris dans les prochaines parties

Nous souhaitions effectuer une augmentation de la vitesse jusqu’à 4 fois la vitesse de base mais la carte possède un filtre passe bas nous limitant une vitesse 2 fois plus élevé. Ce filtre sera présenté dans le bloc de PWM.

## Gestion de la fréquence d’échantillonnage

Voici tout d’abord le schéma RTL de ce bloc :

## Prise en compte de la vitesse sur la valeur des échantillons

Voici tout d’abord le schéma RTL de ce bloc :

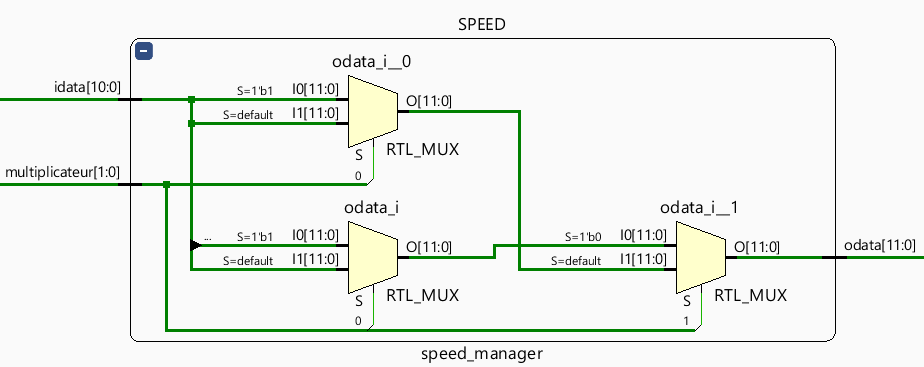


Figure 12 : Schéma RTL du bloc "speed manager"

Ce bloc est construit de la même manière que pour le contrôle du niveau sonore. En fonction de l’entrée multiplicateur sur 2 bits nous augmentons la vitesse ou la diminuons. Nous divisons par 2 la valeur si nous accélérons car la fréquence augmente, nous avons donc pour la PWM moins de période disponible et la valeur de l’échantillon doit alors être adapté aux nombres de période possible. A l’inverse, nous augmenter la valeur de l’échantillon si nous souhaitons diminuer la vitesse car nous aurons plus de période disponible pour coder l’information dans le bloc PWM.

La description de multiplexeur permet de réaliser cette partie tout en prenant en compte la propagation du bit de signe.

## Modulateur numérique

Voici tout d’abord le schéma RTL de ce bloc :

Test avec le sinus :

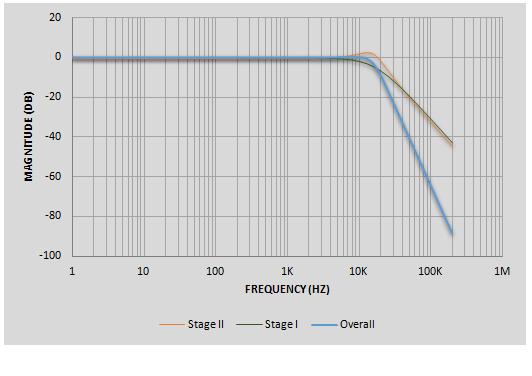


Figure 13 : Réponse du filtre de Butterworth