

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



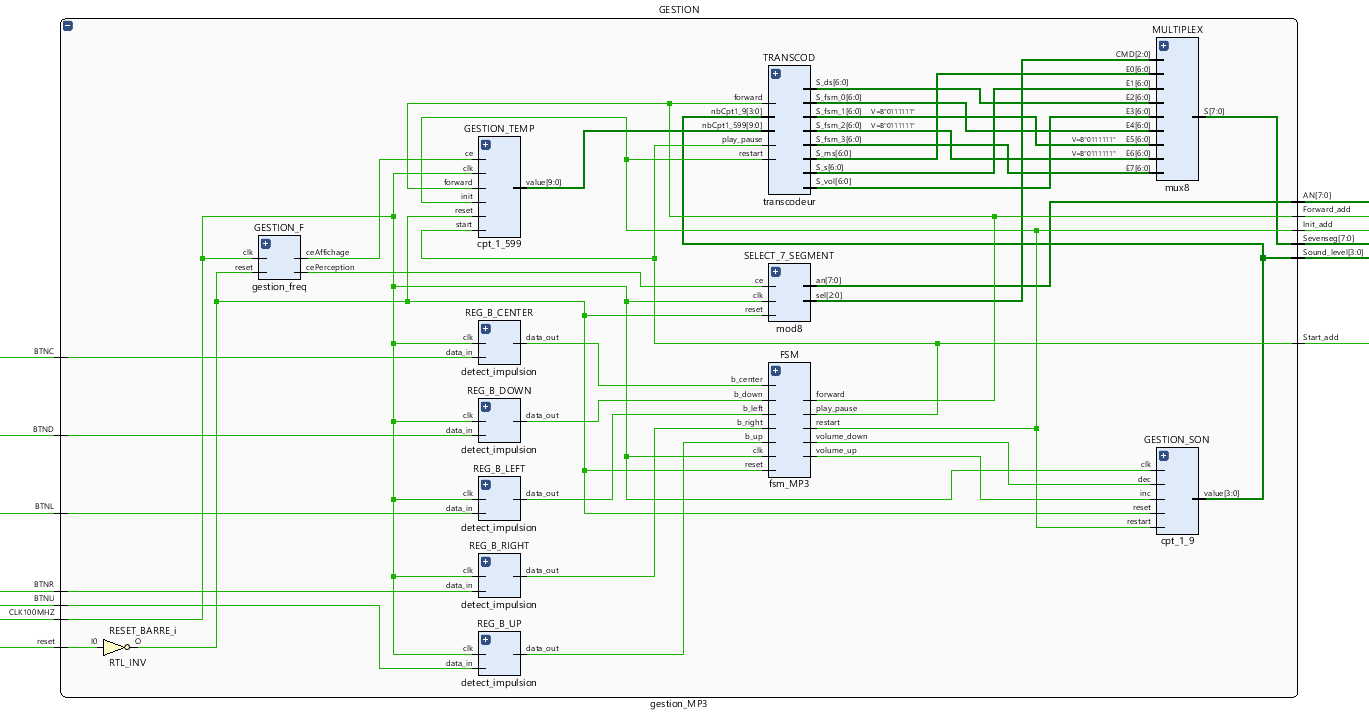
Remis le 02 avril 2021

# 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 :



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 :

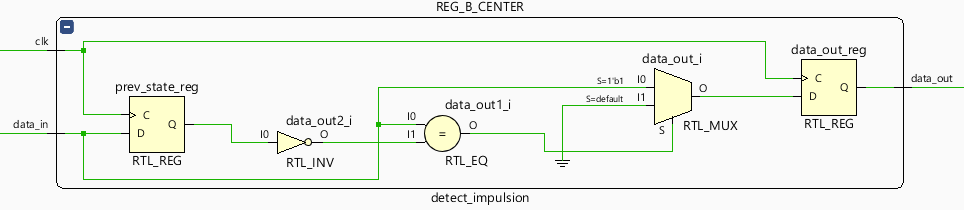


Figure 1 : Schéma RTL du détecteur d'impulsion

## Gestion des fréquences

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

## Compteurs

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

## Machine à état

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

## Transcodeur

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

## Sélecteur

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

## Multiplexeur

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

# Liaison série :

# Contrôle de l’audio

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 :

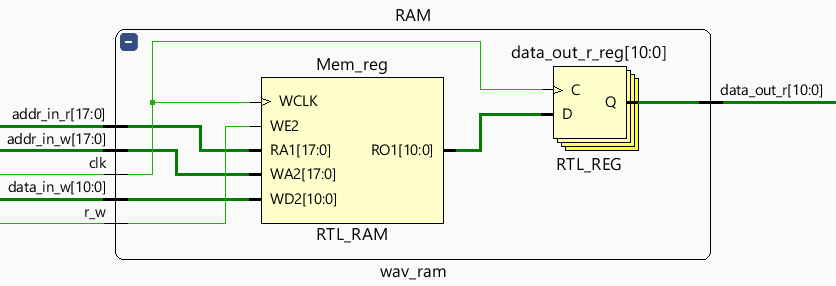


Figure 2 : Schéma RTL de la RAM

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 :

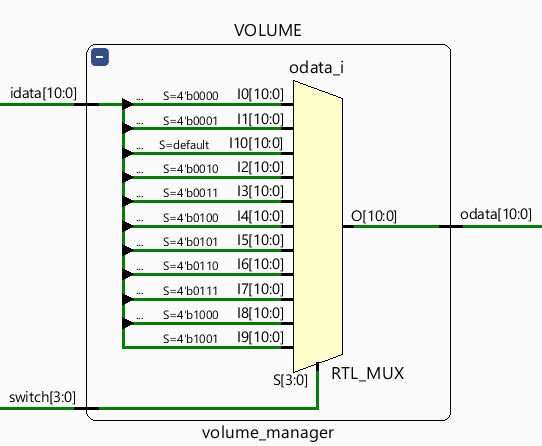


Figure 3 : Schéma RTL de « volume\_manager »

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.

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

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

Test avec le sinus :

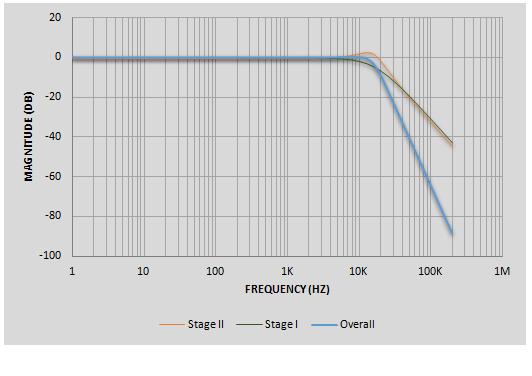


Figure 4 : Réponse du filtre de Butterworth

## Gestion de la fréquence d’échantillonnage

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

## Modulateur numérique

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