ENSEIRB-MATMECA

COMPTE RENDU DE PROJET

Mise en œuvre d'un synthétiseur

PROJET EN202

Binôme:
KOLO Ephraïm
YAN Alix

Encadrant:
Christophe JEGO

Table des matières

1 Introduction

L'objectif du projet a été de concevoir et implanter une architecture numérique en utilisant une carte Digilent NEXYS 4.

Pour ce projet, nous avons eu l'idée de réaliser un mini-synthétiseur, c'est-à-dire un instrument de musique électronique capable de créer et de moduler des sons sous forme de signal électrique.

Principe

En prenant en compte les demi-tons d'une gamme musicale classique, il y a 12 "notes" différentes. On souhaite ainsi associer une note à un bouton (d'un clavier).

De plus, on voudra associer des switchs à différents octaves, et d'autres switchs à différentes formes d'onde, pour varier le timbre de la note.

Modules utilisés

Pour réaliser ce synthétiseur, nous utilisons, en plus de la carte Digilent NEXYS 4, un PmodKYPD de 16 touches qui sert de clavier à notre synthétiseur. Son fonctionnement sera détaillé plus loin dans ce rapport.

Quant au son, on utilise la sortie Jack (Mono Audio Out) de la carte où on branche une enceinte / un casque audio.

Cahier des charges

On souhaite que notre synthétiseur respecte les critères suivants :

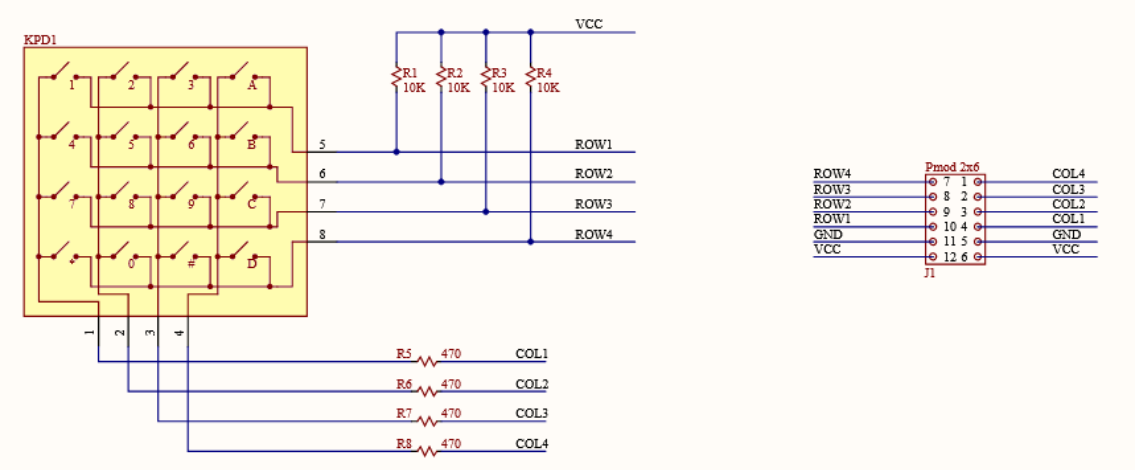
- à chaque appui de bouton, il faut émettre le son correspondant au bouton appuyé, à l'octave choisie et au timbre choisi ;
- lorsqu'on cesse d'appuyer un bouton, le son doit cesser ;
- de plus, l'appui de plusieurs boutons doit entraîner la sortie de toutes les notes correspondantes.

2 Courte présentation du PmodKYPD

Pour notre projet, nous utilisons comme module Pmod un clavier (KeyPad).

Toutes les caractéristiques de ce clavier sont présentées dans sa datasheet (fournie sur le site de Digilent). Les caractéristiques qu'ils seraient primordial d'évoquer ici sont les suivantes:

- Il est composé de 16 boutons-poussoirs momentanés pouvant détecter des appuis simultanés.
- les lignes et colonnes sont isolées.



Le PmodKYPD utilise 4 lignes et colonnes pour créer un tableau de 16 boutons-poussoirs momentanés.
En pilotant les colonnes, les utilisateurs peuvent lire la tension de niveau logique correspondant sur chacune des lignes pour déterminer le bouton, le cas échéant, actuellement enfoncé. Des pressions simultanées sur les boutons peuvent également être enregistrées, bien qu'il soit toujours nécessaire de parcourir chaque ligne et colonne séparément afin de s'assurer que les boutons enfoncés n'interfèrent pas avec chaque mesure.

3 Architecture globale

3.1 Division en modules

Pour ce projet, nous avons utilisé 6 blocs :

- un bloc "Decoder_cor", qui permet de savoir quel(s) bouton(s) est / sont appuyé(s) en pilotant les colonnes ;
- un bloc "DisplayController", qui gère l'allumage des LED selon les touches appuyées, ainsi que l'affichage du volume ;
- un bloc "Generation_carre", qui génère un signal carré aux 24 fréquences différentes (12 notes * 2 octaves) selon le bouton appuyé ;
- un bloc "Generation_triangle", qui génère un signal triangulaire aux 24 fréquences différentes selon le bouton appuyé ;
- un bloc "Mod_PWM", qui récupère les signaux (carré ou triangle) générés, adapte leur volume et les module de manière à pouvoir utiliser la sortie Jack.
- ces 5 blocs constituent les composants du bloc "PmodKYPD" global.

3.2 Formats des données - Communication entre les modules

Les entrées et sorties de notre synthétiseur sont les suivantes :

```
entity PmodKYPD is
    Port (
            clock : in  STD_LOGIC;

            reset : in  STD_LOGIC;

            octave : in STD_LOGIC;
            type_signal : in STD_LOGIC;

            button_up : in STD_LOGIC;
            button_down : in STD_LOGIC;

            JA : inout  STD_LOGIC_VECTOR (7 downto 0);

            an : out  STD_LOGIC_VECTOR (3 downto 0);
            seg : out  STD_LOGIC_VECTOR (6 downto 0);

            sortie_led : out  STD_LOGIC_VECTOR (15 downto 0);

            sortie_son : out STD_LOGIC;
            out_1 : out STD_LOGIC);

end PmodKYPD;
```

Le reset, l'octave choisie et le type du signal choisi sont commandé avec des switchs, d'où l'utilisation de STD_LOGIC.

Les boutons poussoirs haut et bas sont utilisés pour contrôler le volume du son en sortie de la carte.

"JA" correspond à l'entrée / sortie servant à commander le PmodKYPD (les lignes donc JA(7 downto 4) étant des entrées et les colonnes donc JA(3 downto 0) étant des sorties).

"an" correspond, et "seg" correspond aux segments qu'il faut allumer sur l'afficheur, d'où un vecteur de 7 bits.

"sortie_led" correspond aux 16 LEDs de la carte qui s'allument lorsqu'on appuie sur les 16 boutons, d'où un vecteur 16 bits.

Quant aux sorties audio, avec la prise Jack de la carte, elles doivent être sur 1 bit.

Attribution des boutons

Par praticité d'utilisation, nous avons choisi d'attribuer les boutons aux notes de la manière suivante :

- 0 correspond au Do, 7 correspond au Do#
- 4 correspond au Ré, 1 correspond au Ré#
- F correspond au Mi
- 8 correspond au Fa, 5 correspond au Fa#
- 2 correspond au Sol, E correspond au Sol#
- 9 correspond au La, 6 correspond au La#
- 3 correspond au Si

Le 1er switch (le plus à gauche) correspond au type du signal souhaité (triangle ou carré), le 2nd switch correspond à l'octave souhaitée (basse ou haute), et le dernier switch correspond au reset.

Ainsi, les données communiquent entre les différents modules de la manière suivante :

- Les entrées du système issues du clavier sont les entrées du module "Decoder_cor". Celui-ci renvoie un vecteur de 16 bits indiquant l'état de tous les boutons (appuyé ou non);
- ce vecteur est injecté en entrée du "DisplayController", qui fait correspondre une LED de la carte à chaque bit du vecteur ;
- ce vecteur est également injecté en entrée des modules "Generation carre" et "Generation triangle", ceux-ci utilisent alors l'horloge pour générer les signaux de fréquence adéquate. Il en sort donc 2 données : une donnée sur 1 bit "type_donnee" (1 pour un signal carré, 0 pour un signal triangle) et un signal sur 8 bits correspondant au son généré ;
- ces deux signaux sont injectés en entrée de "Mod_PWM" qui va générer la sortie audio sur 1 bit adaptée. Il prend également en entrée les boutons poussoirs haut et bas et génère un vecteur "volume";
- ce vecteur est injecté en entrée du "DisplayController", qui affiche la valeur du volume.

Ainsi, notre système peut-être schématisé de la manière suivante :

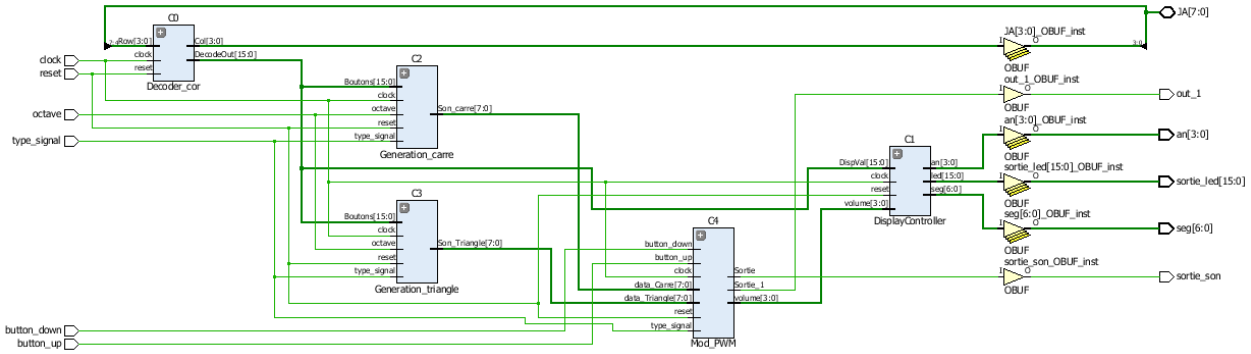


Figure 1: Schematic issu du RTL Analysis

4 Tests et validation des modules

4.1 Exemple de chronogramme

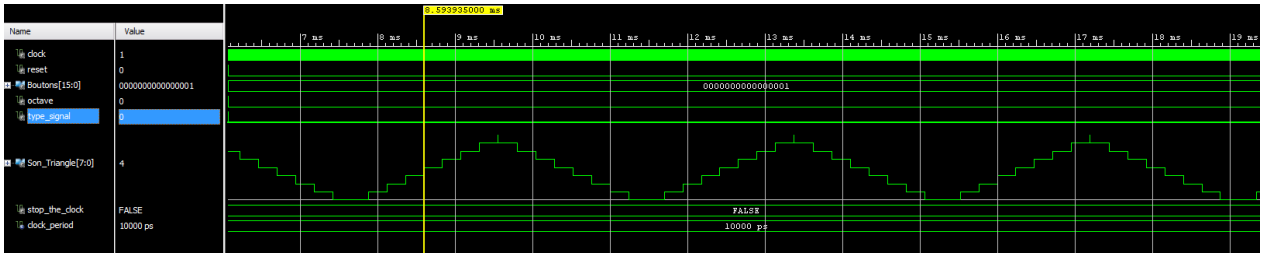


Figure 2: Chronogramme - Simulation "Generation_Triangle"

On constate bien que pour le choix d'un signal triangulaire (type_signal = '0') et de l'octave "grave" (octave = '0'), on obtient en sortie (Son_Triangle) un signal triangulaire de période 3,8 ms, correspondant à une fréquence d'environ 263 Hz soit environ la fréquence du Do (associé au bouton 0, boutons = "0000000000000001").

4.2 Résultats de synthèse

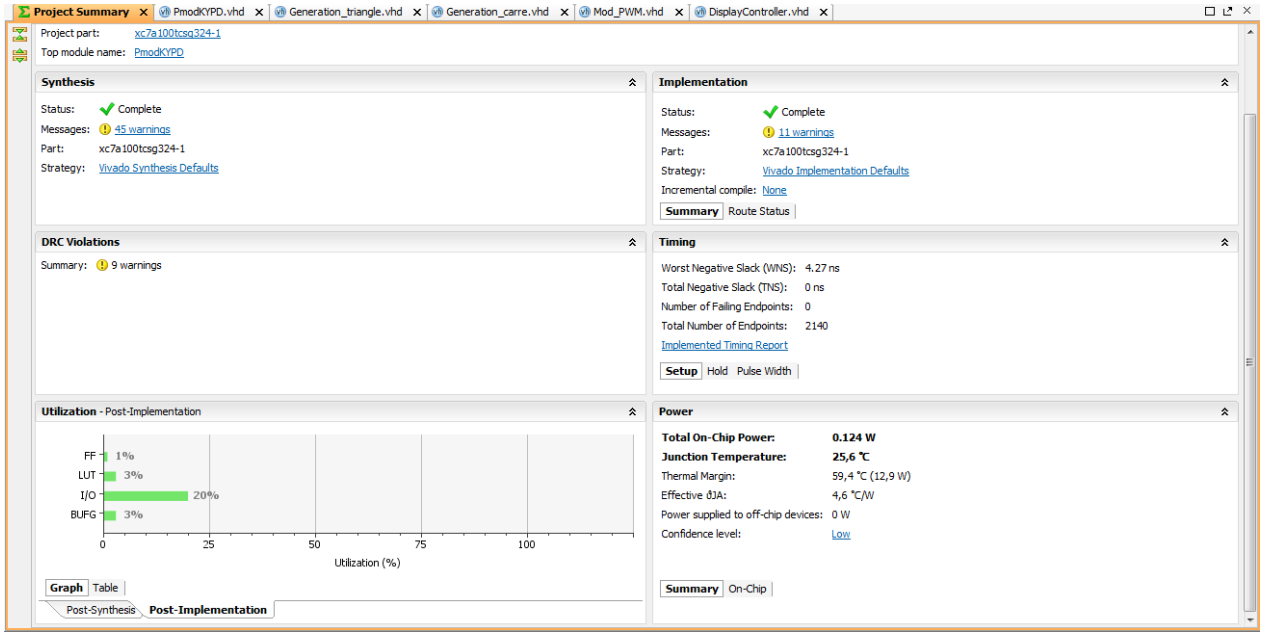


Figure 3: Project Summary

On constate que relativement peu de ressources ont été utilisées.

5 Conclusion

L'objectif de ce projet a été de concevoir et réaliser un synthétiseur, suivant certaines contraintes définies dans le cahier des charges. Notre synthétiseur respecte bien les critères désirés, dans le sens où :

- à chaque appui de bouton, le son émis correspond bien au bouton appuyé, selon l'octave choisie et le timbre choisi ;
- lorsqu'on cesse d'appuyer un bouton, le son cesse ;
- concernant les signaux carrés, il est possible de superposer plusieurs notes et le son final en sortie de la carte correspond bien à l'addition des notes jouées.

Cependant, des problèmes persistent : tout d'abord, la superposition des notes n'est pas bonne avec les signaux triangulaires ; nous n'avons pas encore trouvé d'expression le permettant. Ensuite, la modification du volume n'est pas au point.

Enfin, il existe également de nombreuses pistes à creuser pour compléter ce synthétiseur et lui apporter de nouvelles fonctionnalités. Nous avons songé tout d'abord à ajouter des octaves. De plus, nous aurions souhaité ajouter d'autres types de signaux, tels que le signal sinusoïdal ou rampe. Une fois cela fait, nous pensions également ajouter des modes "démos", c'est-à-dire de petits morceaux pré-enregistrés qui pourraient être joués, ou encore un mode de jeu (par-exemple de mémorisation des touches appuyées,...).

Le synthétiseur est donc fonctionnel mais perfectible.