PFE : Réalisation d'un Synthétiseur

Rapport intermédiaire

UNTEREINER Antoine

Polytech Lille

Encadrants :

Boe Alexandre

Vantroys Thomas

POLYTECH®

INTRODUCTION

Afin de compléter ma formation d'ingénieur en option IMA-SC j'ai décidé de me lancer dans la réalisation d'un synthétiseur. En effet, ce PFE s'inscrit dans une volonté de poursuite de mon projet professionnel. Voulant travailler dans l'industrie de la musique, il me permettra d'avoir un premier contact avec les notions techniques qu'elle implique.

Fort heureusement, mon sujet à été accepté en me laissant une grande liberté. Le synthétiseur pourra alors évoluer assez librement lors de l'année. Il me faut cependant réfléchir à toutes les étapes, afin de passer des premières idées sur papier au produit final. Cela n'est pas tout le temps évident, nous le verrons.

Le projet mêle électronique analogique et numérique, leur proportion pourra évoluer selon les idées ou les difficultés rencontrées en cours de route. Mon premier choix de microcontrôleur se porte sur la famille **Atmega** qui, par sa simplification de codage, permet de réaliser de la DSP assez efficacement.

Un travail très important de recherche à été réalisé en amont afin d'assurer une base de connaissances suffisante et nécessaire au bon déroulement du projet.

I/ Décomposition du projet

Un projet complexe et complet comme celui-ci nécessite d'être décomposé en tâches à effectuer. C'était donc ma première mission, m'ayant permis de prévoir un agenda et d'être efficace. Ci dessous sont exposées les différentes parties que j'ai pu découper.

1. Définition des fonctionnalités de l'instrument

2. Recherches bibliographiques

3. Réalisation du VCO

4. Codage de la DSP numérique (ADSR)

5. Réalisation d'un filtre

6. Réalisation du VCA

7. Réalisation de différents effets intégrés

8. Gestion du MIDI

9. Ajout de fonctionnalités (en fonction du temps restant)

10. Réalisation du boîtier et mise en place du circuit

En gras: Tâche effectuée.
En italique: Tâche en cours.

II/ Définition du cahier des charges

Lorsque j'ai commencé à réfléchir à quoi mon synthétiseur allait ressembler, je dois avouer que j'étais légèrement perdu dans la multitude de possibilités. Les idées venaient par centaines les unes au dessus des autres, toutes plus motivantes les unes que les autres, mais souvent bien incompatibles..

Il me fallait trouver une méthodologie de gestion de projet et découper mon objectifs en sous-catégories. D'une autre part il me fallait des gardes fous afin de ne pas m'emballer et partir sur des idées interminables. J'ai décidé de contacter mon tuteur de stage à Montréal qui travaille dans l'industrie des synthés modulaires. Après plusieurs heures de discussion, voici les principales questions que je me suis posées et auxquelles je me suis appliqué à répondre.

a) Desktop Synth VS Keyboard

A quoi va ressembler le synthétiseur ? Il a plusieurs types de synthétiseurs et parmi eux les Keyboards et les Desktop synths. Alors que le premier incorpore les touches à son boîtier, l'autre et une version plus petite et compact. C'est vers cela que je vais me tourner.

Fig 1. Le légendaire Juno 106 (en haut) et sa version desktop (en dessous)

► Quels sont les avantages ?

1. **Le prix:** Dans le commerce, les synthétiseurs Desktop sont généralement moins cher. Ici, la conception étant entièrement réalisée par mes soins, cela me permet d'oublier la gestion des touches et d'éviter leur achat.

2. **La taille** : Mon but, au delà du projet scolaire, est de garder le synthétiseur et de pouvoir l'utiliser pour mes projet musicaux personnels. Un synthétiseur ainsi réalisé permet de gagner de la place lorsque qu'il est installé dans une configuration d'instruments.
3. **Facilement séquençable** : Un séquenceur est un outil capable d'enregistrer et exécuter une séquence de commandes permettant de piloter des instruments de musique électronique. Il ne produit aucun son par lui-même, mais sert à automatiser l'exécution d'une séquence musicale.

► Qu'est ce que cela implique ?

Le synthétiseur comporte un ou plusieurs VCO qui vont générer les notes. Cependant, sans clavier il faut ajouter au synthétiseur Desktop un moyen de contrôle des notes. Ceci se fera grâce à l'ajout d'un port **MIDI IN**. Cela implique un code de gestion des messages midi et de conversion de ces derniers en tension pour la commande du/des VCO.

b) Analogique ou Digital ? Mono/Polyphonique ?

Lors de l'apparition des premiers synthétiseurs, ces derniers se ressemblaient beaucoup. Tout les sons étaient générés analogiquement, c'est à dire qu'il reposaient tous sur un signal électrique analogique. Les circuits étaient complexes et en résultait principalement des synthétiseurs monophoniques (ne pouvant jouer qu'une note à la fois).

Avec l'arrivée du digital la polyphonie s'est grandement développée. Les synthétiseurs digitaux ont permis une approche simplifiée de la polyphonie (Réduction de la complexité et du coût). En effet, un synthétiseur analogique doit, pour être polyphonique, ajouter chaque voix individuellement à la chaîne du signal et dans l'ordre pour créer le son de l'accord joué.

Dans la mesure où je voulais réaliser un synthétiseur analogique (par envie d'étendre mes connaissance en électronique analogique et par amour du son analogique), mon choix s'est naturellement porté vers la monophonie.
Je précise que l'aspect analogique du synthétiseur que j'entreprends de réaliser concerne la génération du son (VCO) et d'autres blocs tels que le VCA ou le filtre par exemple. Une partie digitale sera en effet présente pour la gestion de différents éléments (ADSR, MIDI IN).

c) ***Quelles seront les fonctionnalités ?***

Une fois le type de synthétiseur défini il m'a fallu réfléchir aux fonctionnalités qu'il apportera. En effet jusqu'ici, si l'on suppose avoir un VCO et un VCA, on a simplement le son d'un oscillateur que l'on contrôle de manière externe via un port MIDI.
Que peut-on faire pour modifier ce signal et offrir à l'utilisateur un minimum d'interaction ? Les idées et possibilités sont infinies. C'est pour cela que la réponse à cette question risque de changer tout au long de l'année. Pour le moment j'aimerai pouvoir :

- Coder une enveloppe ADSR sur le microcontrôleur
- Concevoir un filtre analogique
- Ajouter des effets analogiques (Delay, Reverb, Chorus)
- Réaliser un LFO assignable à plusieurs paramètres (Fréquence de coupure du filtre, paramètres d'effets etc..)
- Rendre le synthétiseur semi-modulaire.

d) ***Décomposition en bloc fonctionnels***

Une fois les bases du projet posées, il m'a fallu découper le synthétiseur en blocs fonctionnels. Cela me permet de travailler sur chacun d'eux séparément, tout en réfléchissant à comment les rassembler en terme de niveaux de signal et de contrôle. Le microcontrôleur sera alors le cerveau sous-jacent qui contrôlera l'ensemble :

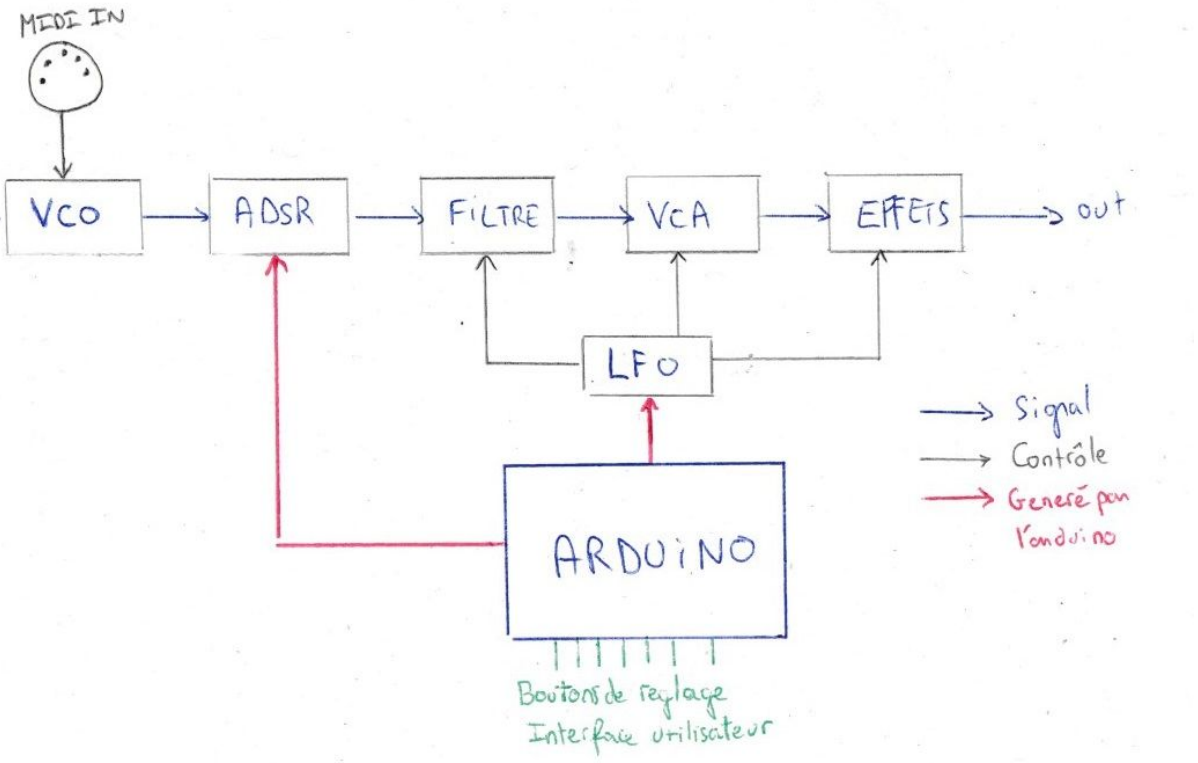

Fig 2. Blocs fonctionnels du synthétiseur.

.

III/ Gestion du Midi et génération du CV pour le VCO

a) Qu'est-ce que le CV ?

Le CV (Control Voltage) est une méthode analogue permettant de contrôler les synthétiseurs. Il trouve son origine dans les premiers synthétiseurs modulaires qui, à l'aide de câbles manipulés par l'utilisateur, faisaient circuler un voltage à travers plusieurs composants (VCO, filtre, effets..). Sur la plupart des modules on retrouvait alors une prise "CV" permettant de moduler un paramètre, comme la fréquence de coupure du filtre par exemple.
Dans notre cas, il va permettre de jouer la note voulue. En effet, comme nous l'avons expliqué plus haut, le VCO à besoin d'une tension afin d'être commandé. Nous communiquerons avec le synthétiseur via MIDI. Le micro-contrôleur se chargera alors d'effectuer la conversion MIDI->CV à l'aide d'un DAC. Il y a deux implémentations majeures du CV :

Implementation / Note	A1	A2	A3	B3	C4	D4	E4	A4	A5
Volts/octave (V)	1.000	2.000	3.000	3.167	3.250	3.417	3.583	4.000	5.000
Hertz/volt (V)	1.000	2.000	4.000	4.491	4.745	5.345	6.000	8.000	16.000

Fig 3. Les standards du CV.

- **Le Volt/Octave** : Comme son nom l'indique, 1V équivaut à une octave. Ce standard a été popularisé par Bob Moog dans les années 60 et est utilisé dans de nombreux appareils.
- **Le Hz/Volt** : Dans cette configuration, augmenter d'une octave revient à doubler la tension. Ce mode est majoritairement utilisé par les compagnies **Korg** ***et*** *Yamaha'*.

b) Le langage MIDI

Le langage MIDI (abréviation de Musical Instrument Digital Interface) est un langage inventé dans les années 80. C'est un protocole de communication dédié à la musique permettant l'interaction entre les instruments électroniques, et les différents contrôleurs, séquenceurs.
Les messages MIDI sont généralement composés de 2 octets (mais peuvent aller jusqu'à 4). On les distingue en deux catégories majeures définies par leur premier bit.

- Si le premier bit est de poids **fort**, il s'agit d'un *status byte*. Eux-mêmes se décomposent en nibble (demi-octets). Le premier nibble indique la commande. Dans notre cas, nous nous concentrerons les octets **Note On** et **Note Off** qui, comme leur nom l'indique, indiquent lorsqu'une note est jouée ou arrêtée. Le second indique le canal.

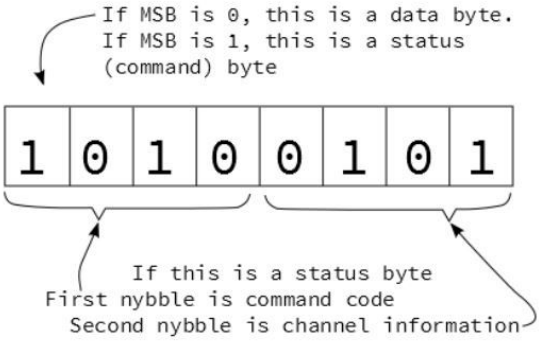

Fig 4. Les messages MIDI.

- Si le premier bit est de poids **faible**, il s'agira alors d'un *data byte* qui apportera des informations complémentaires à l'octet de statut. Les octets **Note On** et **Note Off** sont accompagnés de deux octets de données. Le premier indique la note jouée, le deuxième la vélocité (force avec laquelle la touche à été pressée).

Donc, lorsque l'utilisateur joue un LA440 (A4 en anglais) cela produira le message: **0x90(NoteOn) 0x69(A4) 0x64(Velocity)** Le langage midi comporte un grand nombre d'octet de statuts permettant toute sorte de choses (contrôle de séquence, changement de preset etc..) mais nous n'aurons besoin que de ces deux pour le moment.

c) ***MIDI INPUT : Circuit de réception des données***

Toutes les informations sur le MIDI peuvent être facilement trouvées sur le site de la *MIDI manufacturers association*[1]. En cherchant sur le site, j'ai pu établir sur logiciel ce schéma de réception MIDI:

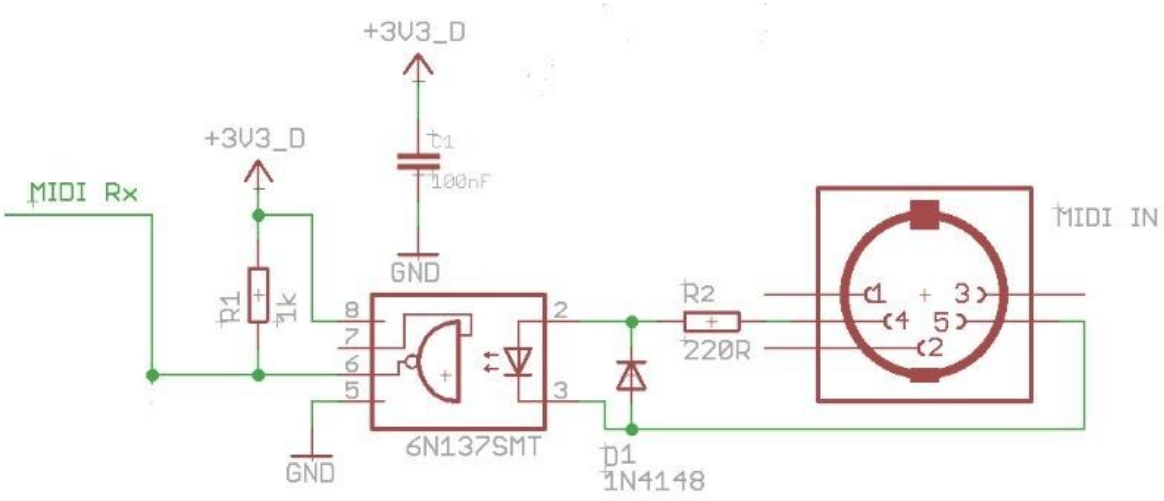

Fig 5. MIDI IN.

Ce circuit, assez simple, utilise un optocoupleur. Un optocoupleur permet la transmission de donnée sans connexion électrique. Le signal en entrée fait clignoter une DEL et la lumière produite est captée par un phototransistor qui la convertit à nouveau en signal électrique.

Les pins 4 et 5 sont reliée respectivement à l'anode et la cathode de la LED. Lorsque aucune information est transmise, les pins 4 et 5 sont au même voltage et la DEL n'est pas allumée. Donc l'UART reçoit une tension en Pull-up et il en résulte un signal logique à 1. Et inversement lorsqu'une donnée est transmise.

C'est l'idée brillante des concepteurs du MIDI. En effet, cela permet au différents fabricants de faire communiquer leurs synthétiseurs entre eux, sans prendre en compte leur standard de

[1] https://www.midi.org/

tension. En plus de cela, cela permet d'éviter les pertes importantes de tensions dans les longs câbles, car ils utilisent des boucles de courant.

Une fois le circuit mis en place j'ai pu tester grâce à un analyseur logique que les données étaient effectivement reçues. Pour cela j'ai connecté le port midi à mon ordinateur via un boîtier spécialisé et j'ai envoyé des données via un logiciel nommé MidiOx. N'ayant pas apporté de clé USB lors du test je n'ai pas de capture d'écran mais cela ressemblait à ceci (exemple des messages Note_On et Note_Off) :

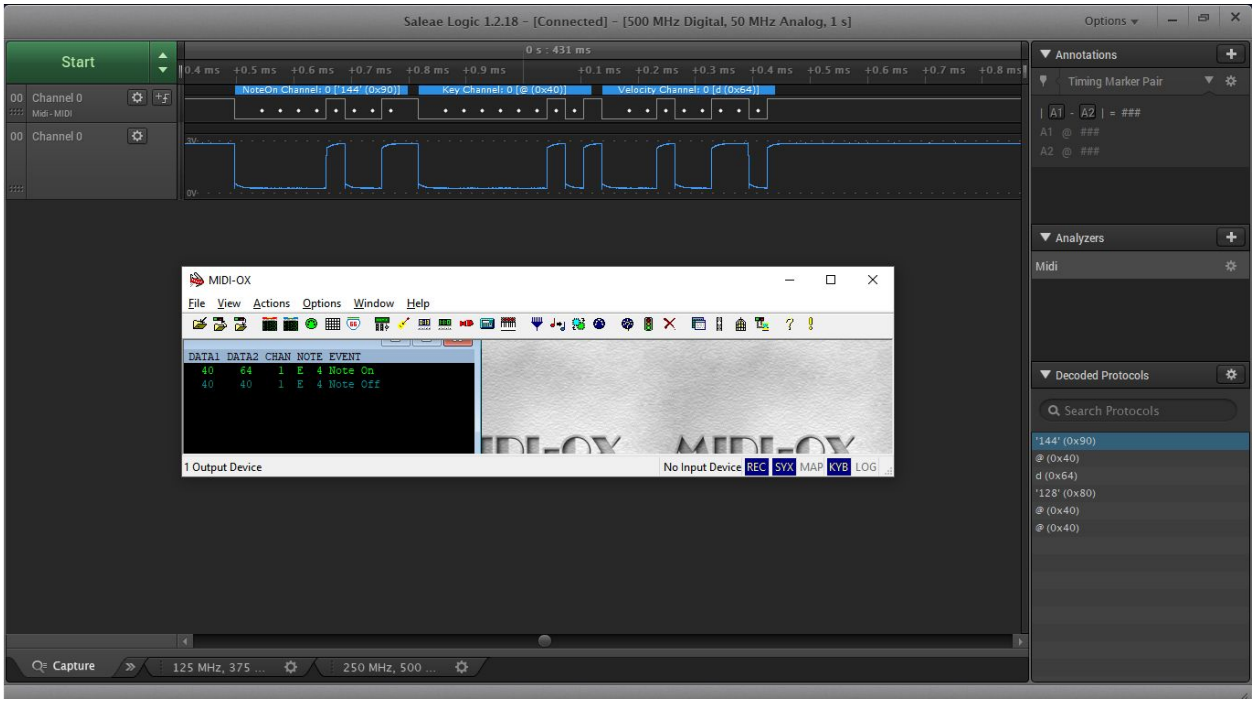

*Fig 6. Messages **Note_On** et **Note_Off**.*

Ces messages vont donc être transmis à l'Arduino afin d'être traités. Ils sont transmis via le port Rx avec un baudrate peu conventionnel et propre au midi valant **31250**. A partir de la, les données doivent être analysées. Pour cela je voulais d'abord ré-implémenter la machine à état finis que j'avais créé lors de mon stage de 4ème année. Cependant, cette machine à état finis fait parti d'un système bien plus large, et donc lourde pour cette application, et est codée pour les noms de registres et les paramètres d'un autre système (Système ARM).

Remarque : Il faudra ajouter un switch entre la réception et le port Rx. En effet sans cela l'opto-coupleur interférera avec le port Rx même quand il n'y a pas de messages et empêcherait l'upload de nouveaux programmes sur l'arduino.

Afin de nous simplifier le travail la base de librairies arduino propose une librairie Midi développée par *Forty Seven Effects*[2]. En regardant la documentation de cette librairie on se rend compte qu'elle permet l'usage de callbacks. Un callback est une fonction que l'on écrit et qui va appeler la librairie uniquement lorsqu'un message arrive. Cela évite de faire tourner le programme en continu et d'utiliser tout le CPU. Pour les utiliser c'est simple, lors de l'initialisation (**void setup()**) on active le callback voulu. Ainsi, **MIDI.setHandleNoteOn(NoteOnHandler)** appellera la fonction NotOnHandler lorsque qu'une note sera jouée.

Nous pourrons donc traiter indépendamment les différents messages (si on veut faire évoluer le programme). Avant cela nous devons régler le problème de la sortie. L'arduino seul n'est pas capable de sortir directement des voltages stables.

A la place il utilise des pulsations de périodes variées (PWM). Il nous faut donc un convertisseur digital vers analogique (DAC). Il y a plusieurs manières de réaliser un DAC, comme une échelle R-2R par exemple, ou un filtre RC. Cependant, pour contrôler un VCO nous avons besoin d'un DAC assez précis dans la mesure où un écart entre deux semi-tons est environ égal à 0.083V. J'ai donc décidé d'utiliser un DAC dédié, le **MPC4725** d'Adafruit qui a une librairie dédiée pour les modules arduino.

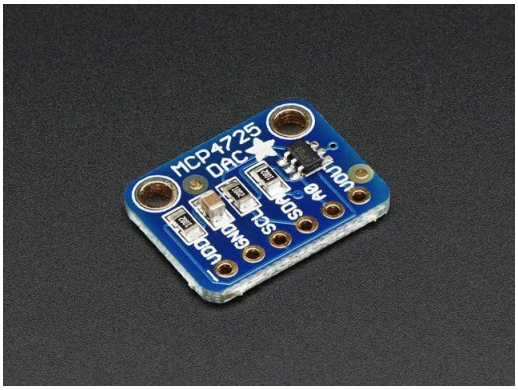

Fig 7. DAC Adafruit MPC4725.

[2] https://github.com/FortySevenEffects/arduino_midi_library

Nous allons maintenant décrire les différentes parties du programme :

► Variables:

- ➢ **MIDI_CHANNEL** : Canal d'écoute des messages midi.
- ➢ **Voltage** : Voltage en mV correspondant à la note jouée.
- ➢ **dacValue** : Valeur transmise au DAC.
- ➢ **VLinCoeff** : Variable permettant un tuning lors de la mise en pratique.
- ➢ **Vshift** : Permet d'augmenter ou diminuer d'octave.
- ➢ **LastNote** : Stockage de la note jouée pour vérification dans la fonction Note_Off

► Setup():

- ➢ **MIDI_CREATE_DEFAULT_INSTANCE** : Créé et lie l'interface MIDI au port Série du matériel.
- ➢ **TCCR1B = TCCR1B & B11111000 | B00000001** : Augmente la fréquence de la PWM du timer 1 (Pin D9 et D10) en réglant son diviseur à 1. On obtient alors une fréquence de 31372.55 Hz, ce qui permet de réduire les ondulations de tension en sortie.
- ➢ **MIDI.setHandleNoteOn/Off(Note_On/Off_Handle)** : Initialisation des callbacks.
- ➢ **MIDI.begin(MIDI_CHANNEL)** et **dac.begin(0x60)** : Initialisation de la connexion MIDI et DAC.

La fonction **MIDI.read(MIDI_CHANNEL)** sera placée dans la boucle pour lire les messages en continu.

► La fonction Note_On_Handle(byte channel, byte note, byte velocity):

➢ Tout d'abord la fonction sauvegarde la note dans la variable **LastNote**

➢ On calcule ensuite le Voltage à l'aide de la formule:

Voltage = 1000*((note*VoctLinCoeff)+ VoctShift)

➢ Le Voltage à été multiplié par 1000 car les valeurs décimales sont tronquées par la fonction constrain(). Or, nous appelons cette fonction pour le calcul de la valeur du DAC:

dacValue = constrain[3](map(Voltage, 0, 5000, 0, 4095), 0, 4095)

➢ Finalement on applique au dac la valeur souhaitée avec la fonction **dac.setVoltage(dacValue, false).**

Remarque: Pour la fonction Note_Off_Handle teste si la note reçue est la note jouée auparavant. Si c'est le cas, elle met à 0 la valeur envoyée au DAC.

[3] La fonction Constrain ré-étalonne un nombre d'une fourchette de valeur vers une autre fourchette. Cette fonction ne contraint pas les valeurs à rester dans les limites indiquées, d'où l'utilité de la fonction constrain.

IV/ Conception du VCO

a) *1er Circuit : Miss10*[4]

Après avoir passé beaucoup de temps à lire des documentations techniques et des articles sur les différentes marques/modèles de synthétiseurs j'avais décidé dans un premier temps d'essayer de reproduire le VCO du Korg MS-10[5]. J'ai finalement choisi de changer de modèle. En effet, mon ancien tuteur de stage était pris et n'a pu eu le temps de m'aider suffisamment pour que j'ai une compréhension complète du circuit. De plus certains composants nécessaires à la réalisation de ce VCO sont obsolètes et donc difficiles et onéreux à trouver. J'ai donc décidé de me tourner vers une version plus moderne. Je vais tout de même rendre compte ici de mon analyse du circuit dans la mesure où elle m'a permis de comprendre un certain nombre d'éléments.

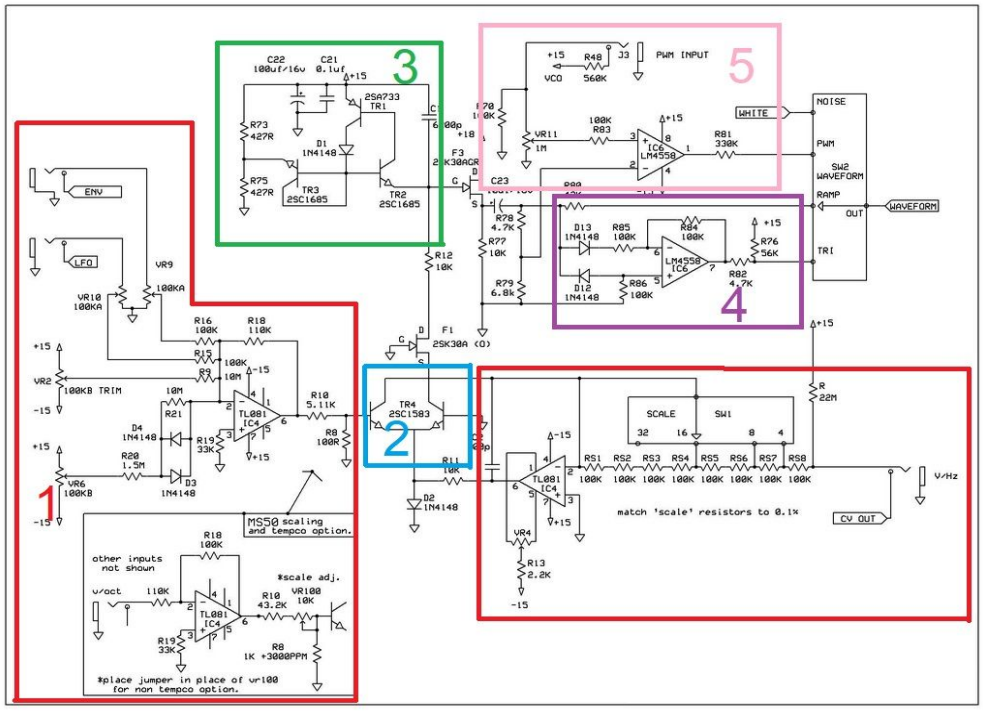

Fig 8. Le VCO et sa Décomposition

[4] Projet trouvé sur https://www.muffwiggler.com/forum/index.php
[5] Manuel de référence : http://synthmanuals.com/manuals/korg/ms-10/service_manual/

On peut découper ce circuit en plusieurs parties fonctionnelles. Il faut savoir avant tout que le cœur même de cet oscillateur se base sur un générateur de signal en dent de scie. Ce signal est après modifié afin de créer les autres formes de signaux.

► Partie 1 : Le sommateur de CV (Control Voltage) exponentiel.

L'Amplificateur opérationnel IC4 (un TL081) permet de sommer les voltages correspondants à l'enveloppe, au LFO (Low Frequency Oscillator) afin de contrôler avec une réponse exponentielle la fréquence. Alors qu'un oscillateur classique ajoute un autre nœud pour le CV (Conversion depuis le midi comme vu dans la partie précédente), les oscillateurs à la base des synthétiseurs MS-XX de Korg gèrent cela sur un autre nœud de TR4, le couple de transistors. Je ne suis pas sûr de l'utilité de ceci mais je crois en conclure que comme ces synthétiseurs fonctionnent en V/Hz et non en V/oct, cela permet d'avoir une réponse linéaire plutôt qu'exponentielle. Le TL081 à droite est donc l'entrée V/Hz associée à un réseau R2R et un sélecteur rotatif qui permet de contrôler la "scale", autrement dit l'octave.

► Partie 2 : La source Voltage->Courant.

La combinaison des deux TL081 directement connectés à la paire de transistors TR4 permet de créer un source de courant qui permet de donner au cœur du VCO la fréquence à laquelle il doit se charger. En d'autres mots: Plus le voltage est élevé, plus il y a de courant à travers la paire TR4, donc un plus grand taux de charge pour l'intégrateur et donc une fréquence plus élevée.

► Partie 3 : Le cœur de l'oscillateur.

TR1, TR2 et TR3 forment le cœur de l'oscillateur. Comme je l'ai dit je n'ai pas compris les détails précis du fonctionnement. Mais il que la capacité de 6200 pF, avec les composants autour, forme un intégrateur qui se charge à un voltage donné. Quand il atteint un certain seuil, qui sera le point le plus haut du signal rampe, un transistor est activé et la capacité se décharge et le cycle est répété. Ainsi, de manière périodique, la capacité subit une série de charges et de décharges qui forment le signal en dent de scie. La fréquence dépendra alors des éléments de la partie 1.

F3 me semble être un simple buffer/Amplificateur pour leur cœur du VCO. A la sortie de la capacité C23, qui est une capacité de découplage pour probablement centré la rampe sur 0V, on trouvera le signal en dent de scie.

► Partie 4 et 5 : Les transformations du signal

Comme je l'ai dit plus haut, l'oscillateur se base sur ce signal rampe pour créer les deux autres signaux. En effet, dans la partie 4, le signal est injecté dans IC6 à travers deux diodes. Selon moi, ce circuit découpe le signal rampe en 2 moitié : celle qui augmente et n'est pas inversée et une miroir qui est inversé. Il y aura plus de détails la dessus plus tard.

Dans la partie 5, on a IC6 qui agit comme un simple comparateur. En comparant le signal rampe à une certaine valeur (donnée par l'entrée PWM) on obtient le signal carré.

b) 2ème Circuit

i) Coeur du VCO

► Fonctionnement:

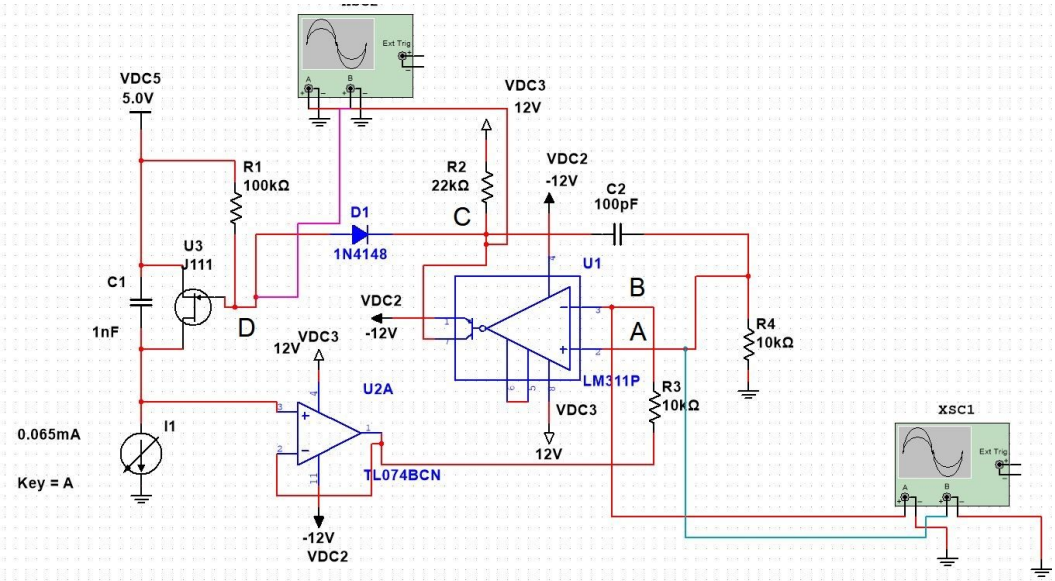

Fig 9. Coeur de l'oscillateur.

L'idée sous-jacente à cet oscillateur est simple : Considérons la capacité C1 déchargée. L'amplificateur opérationnel U2 sert de buffer pour la tension aux bornes de C1; la tension au point B est donc de 5V. La source de courant force un courant à passer à travers C1 qui se charge alors, faisant diminuer la tension en B de manière linéaire.
Le composant U1 est un comparateur, comparant alors la tension en B avec celle de l'autre borne, qui est égale à 0V. Quand A descend en dessous de 0, le comparateur active brièvement U3. U3 est un transistor JFet, qui lorsqu'il est activé permet à la capacité de se décharger, jusqu'à que la tension en a est à nouveau égale à 5V. Ainsi, ce principe se répète de manière cyclique et crée le signal rampe.

► Simulation et problèmes:

J'ai donc simulé ce circuit. Lorsque j'ai lancé la simulation avec un oscilloscope au point B, je n'avais rien. J'ai alors perdu une journée entière à débugger le circuit. Après quelques temps et de nombreux essai je me suis rendu compte que Multisim définit par défaut au lancement de la simulation ses propres conditions initiales basées sur un calcul que je n'ai pas vraiment compris.
J'ai donc changé les conditions initiales pour que la tension en B soit bien de 5V au départ. J'ai donc constaté la diminution linéaire de la tension. Cependant lorsqu'elle passait en dessous de 0V elle se bloquait à -500mV. Après divers essais et des visualisations à différents points du circuit, j'en ai conclu que le transistor JFet ne fonctionnait pas.

Après quelques heures de tests et de recherches j'ai trouvé la source du problème. C'est à deux doigts d'abandonner que j'ai remarqué que le transistor était le seul élément qui, schématiquement, était affiché en vert. J'ai donc cherché ce que cela signifiait sur des forums et c'est à ce moment que je me suis rendu compte que le composant était placé en **PCB LAYOUT ONLY** ce qui signifie qu'il est placé pour avoir l'empreinte lors du passage en pcb mais ne comporte aucun modèle de simulation.

A l'aide d'un tutoriel[6] fourni par *National Instruments* j'ai pu créer le composant. Pour le modèle de simulation j'ai du chercher ce qu'on appelle le modèle **Spice**[7].

J'ai donc rechargé le schéma, relancé la simulation et tout fonctionnait correctement :

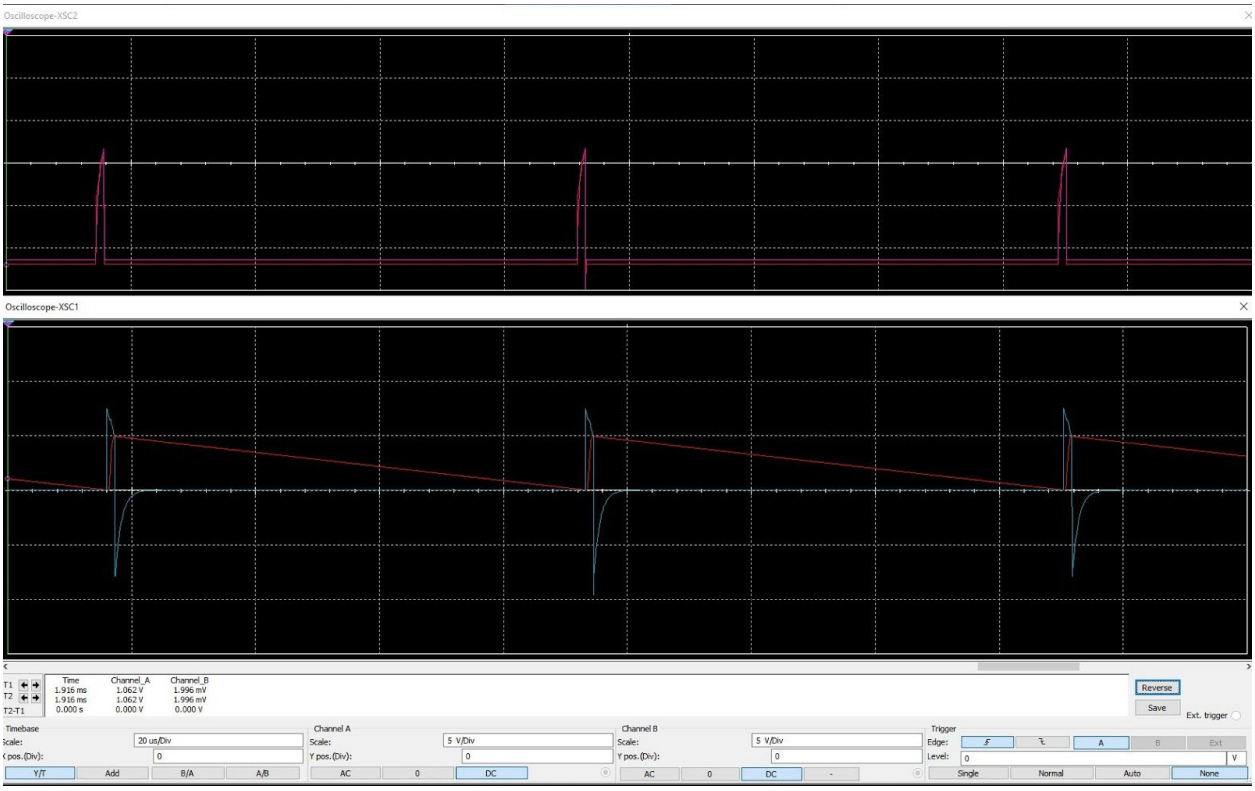

Fig 10. Simulation du coeur.

Dans un premier temps on a une tension positive au point B, et la tension au point A est égale à 0V. La sortie du comparateur est donc basse et la tension au point C vaut -12V. Cette tension se propage à travers la diode et le transistor n'est donc pas activé.

Ensuite, lorsque la valeur de tension au point B est négative, la sortie du comparateur devient haute. La tension au point C vaut donc +12V et grâce à la diode et la résistance R1 elle vaut +5V au point D. Le transistor devient alors passant et la capacité C1 se décharge, jusqu'à que la tension au point B revienne à 5V. Et ainsi de suite.

[6] https://forums.ni.com/t5/National-Instruments-Circuit/Component-Creation-101/ba-p/3486929
[7] http://web.rfoe.net:8000/ziliaoxiazai/philips/models/spicespar/data/j111.html

ii) La source de courant

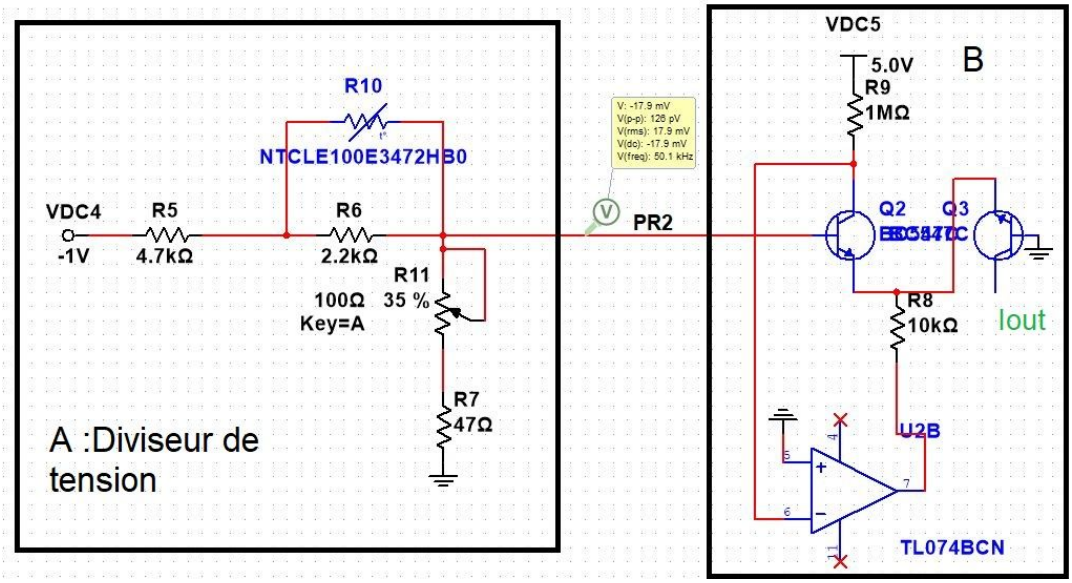

Fig 11. Source de courant.

Une fois que le cœur du VCO était en place, il m'a fallu travailler sur la source de courant qui permet d'avoir un courant tiré proportionnel au CV appliqué, pour avoir pas extension un lien de proportionnalité avec la fréquence. Afin d'avoir la propriété 1V/Octave il faut que le courant ait une relation exponentielle avec le CV d'entrée[8].

Si nous prenons un transistor classique, son courant à l'émetteur s'exprime par la formule :

$$IE = IS(e^{\frac{VBE}{VT}} - 1) \approx ISe^{\frac{VBE}{VT}}$$

L'approximation peut se faire car Ie >> Is. Le courant de collecteur Ic, courant que nous utiliserons, est quasi-égal au courant de l'émetteur Ie. Le problème cependant que l'on peut observer dans cette formule est la dépendance forte à la température (facteurs **VT** et **Is** liés à la température). Afin de contrer ce phénomène on utilise deux transistors en miroir.

[8] Pour plus de détails : https://www.schmitzbits.de/expo_tutorial/index.html
https://www.allaboutcircuits.com/projects/diy-synth-series-vco/

Si l'on part du principe que Vb1 = Vb2 = 0V, comme les émetteurs des deux transistors sont connectés, ils ont leur tension VBE et leur courant d'émetteur égaux. L'amplificateur opérationnel sert de convertisseur tension->courant et force le courant de référence Iref défini par R1 à travers Q2. On a un courant de 5µA avec une résistance de 1 MOhms. Le fait d'avoir configuré les transistors en miroir permet d'avoir le même courant à travers Q3 et on récupère alors le courant indépendamment de la température.
On veut maintenant changer le courant de sortie de ce sous-circuit B à l'aide d'une tension. Ceci s'effectue lorsque Vb1 et Vb2 ne sont pas égaux. Le courant que l'on récupère en sortie s'exprime par la formule $Ic = I_r e f e^{\frac{Vb2-Vb1}{VT}}$

Le CV alimente Vb1 et Vb2 est connecté à la masse, valant alors 0V. Lorsque l'on calcule l'inverse de la fonction on peut voir que lorsque Vb1 diminue de 17.8 mV, le courant Ic double. Nous avons un circuit à -17.8mV/octave. Ainsi, pour atteindre les 1V/octave il nous faut, à l'aide du sous-circuit A, un diviseur de tension et un inverseur (non présent dans le schéma, voir schéma global).

Il reste cependant la dépendance en température de VT qui s'exprime par $V_T = \frac{k_B T}{q}$.
Pour contrer ceci, on utilise une thermistance CTN (Coefficient de Température Négatif) dont la résistance diminue de façon uniforme quand la température. Plus de détails sur cette partie sur le site de René Schmitz.

Remarque sur la paire de transistors : Lorsque nous réalisons un amplificateur différentiel avec les deux transistors en configuration miroir, il faut que ceux ci soit assortis. Pour assurer la linéarité de l'amplificateur, il faut que les transistors aient la même caractéristique tension vers courant et approximativement le même Is. Des circuits de tests se trouvent en faisant quelques recherches[9].

[9] http://www.dragonflyalley.com/synth/images/TransistorMatching/ianFritz-transmat0011_144.pdf

iii) Génération du signal rampe

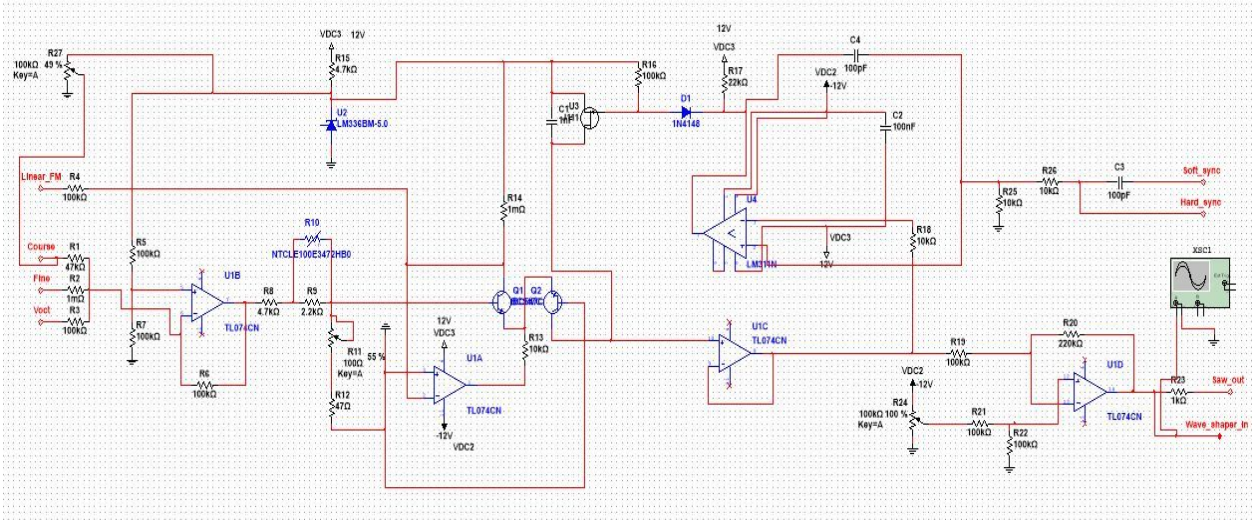

Fig 12. Génération du Signal rampe.

►<u>Tension de référence 5V :</u>

Il est essentiel que la fréquence du VCO ne change pas s'il y a du bruit dans l'alimentation. L'utilisation d'une Diode Zener LM336/5 permet de générer la tension de 5V stable. Cette tension est utilisée pour :

- ✓ Iref comme expliqué dans la partie **source de courant**
- ✓ L'amplitude du signal rampe comme vu dans la partie **coeur de l'oscillateur**
- ✓ Le sommateur de CV : Somme les entrées CV et inverse le signal pour donner à la thermistance un suivi -1V/octave. Pour avoir une plage de réglage convenable un offset négatif est nécessaire. On ajoute donc 5V à l'entrée positive du sommateur.

►Entrée FM, Soft et Hard Sync:

La FM, frequency modulation est une caractéristique très présente dans les modules VCO. L'entrée ajoutée change Iref. En effet l'amplificateur opérationnel ajoute la modulation de fréquence via la résistance R14.

L'option **sync** est très présente sur les synthétiseur. On y ajoute un second oscillateur. Lorsqu'un oscillateur fini son cycle, il réinitialise la période de l'autre, le forçant à opérer sur la même base de fréquence. Cela permet de créer un son riche entre les deux oscillateurs. L'oscillateur qui ré-initialise le second s'appelle le **maître** et l'autre **l'esclave**. Il y a deux méthode de synchronisation : le **hard sync** et le **soft sync**.

- ✓ **Hard Sync:** Ici la hauteur de l'oscillateur esclave peut être changée ou non. A chaque fois que le cycle de l'oscillateur maître est fini, celui de l'esclave recommence, peut importe sa position. Si l'esclave opère à une fréquence inférieur au maître, il sera forcé de se ré-initialiser avant de finir un cycle. Dans le cas contraire, il se réinitialisera au milieu du second ou troisième cycle.

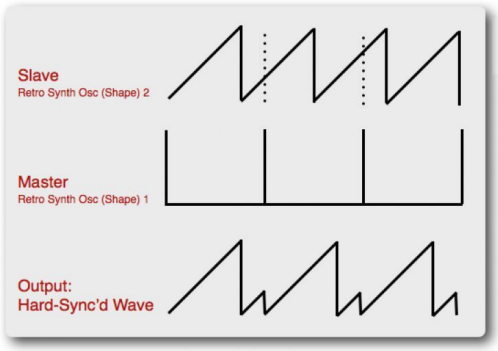

Fig 13. Exemple du Hard Sync.

- ✓ **Soft Sync:** Tandis que dans la synchronisation *dure* l'esclave est forcé de se ré-initialiser peu importe de la position ou la direction de l'onde, créant parfois des formes asymétriques, la synchronisation *douce* se base sur les harmoniques des signaux.

iv) Modelage du signal

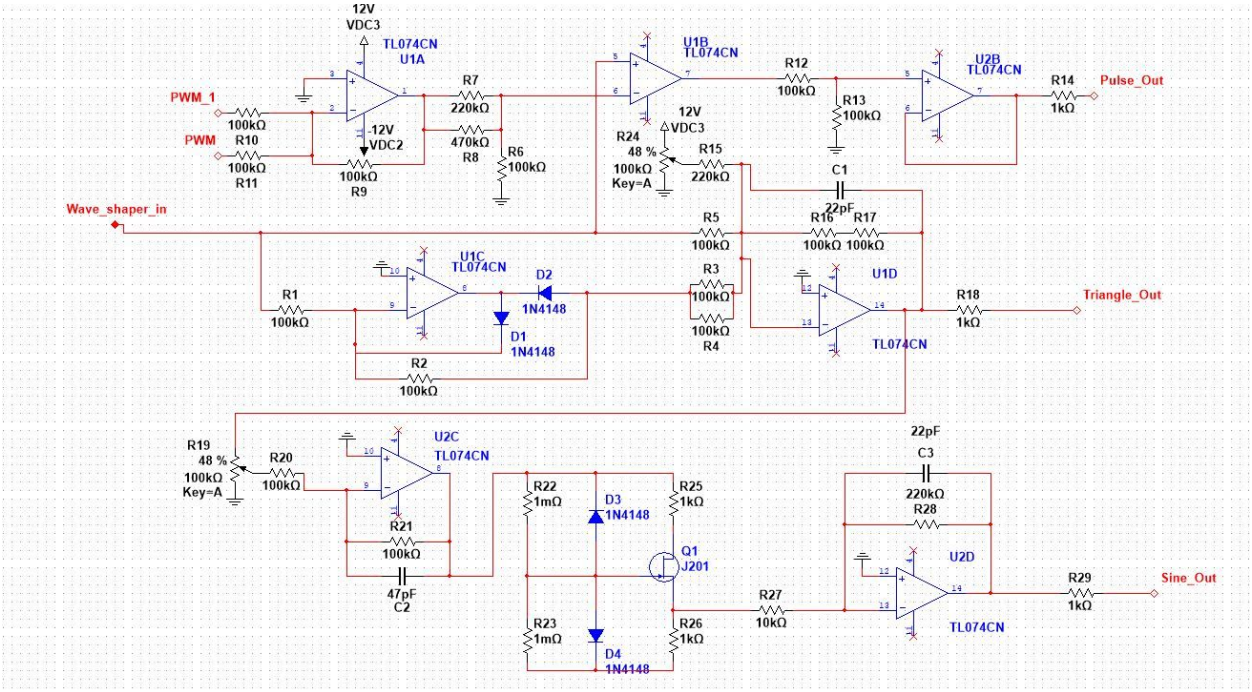

Fig 14. Circuit de modelage du signal.

Une fois le signal rampe généré, il nous faut une autre partie de circuit pour transformer ce signal et générer d'autres formes (Sinusoïde, Carré, Triangle):

► <u>Le signal Carré:</u>

Afin de créer un signal carré avec une modulation du rapport cyclique, il suffit d'un simple circuit comparateur. Le signal en dent de scie est comparé avec une entrée d'un certain voltage. Ceci est effectué grâce à l'amplificateur opérationnel U1B. R6, R7 et R8 forment un pont diviseur de tension pour ajuster l'amplitude du signal à environ +/- 5V, ce qui correspond au signal rampe que nous avons. L'amplificateur opérationnel U2B sert de buffer.

► Le signal Triangle:

Le signal triangle est créé à partir du signal rampe par un **redressement à double alternance**. L'amplificateur U1C prend la partie positive du signal et l'inverse. La partie négative est "remplacée" par 0V. Ce signal la est ensuite ajouté deux fois au signal rampe original, ce qui crée un signal triangle. A l'aide d'un offset introduit par le potentiomètre R24, ce signal pourra être centré sur 0. L'amplificateur opérationnel U1D va alors doubler le gain pour avoir le signal entre +/- 5V.

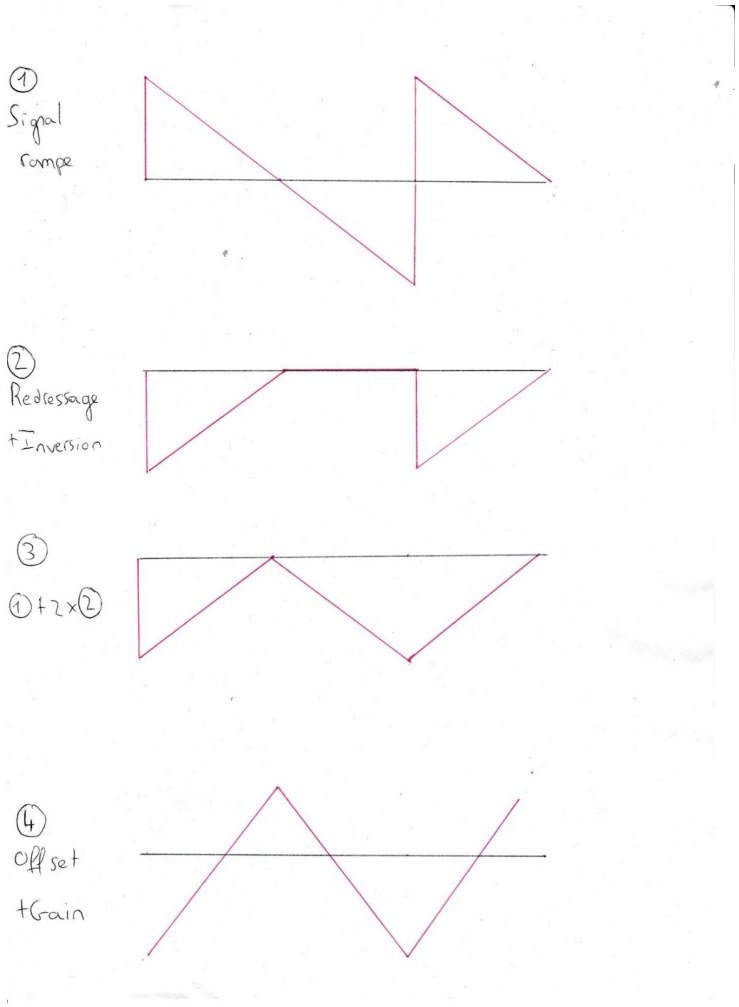

Fig 15. Signal Triangle.

► Le signal Sinusoïdal:

Le signal sinusoïdal est, quant à lui, créé à partir du signal triangle. Je n'ai pas compris le fonctionnement de cette partie du circuit en détail. Ce que j'ai pu comprendre c'est que selon la polarité de l'entrée, les diodes D3 et D4 limitent la tension à la gate de Q1. La saturation de ce transistor JFET donne la conversion en sinusoïde. Les différents amplificateurs opérationnels servent à amplifier le signal.

J'ai choisi ce design car il permettait d'obtenir un taux de distorsion harmonique inférieur à 1% et se retrouvait dans de nombreux synthétiseurs classiques de l'époque[10].

v) PCB

Une fois le circuit complet mis en place, et les différents test réalisés, il était temps de construire le pcb. Comme l'intégralité de mes tests étaient réalisés sur **Multisim** j'ai décidé d'utiliser le software associé développé par National Instrument : **UltiBoard**.

J'ai dû préalablement choisir un **package** pour chacun de mes composants. J'ai donc cherché les composants utilisés un par un, afin d'avoir les **footprints** correspondants. Un circuit de cette taille était très compliqué à mettre en œuvre. N'ayant pas fait de PCB depuis le module de CCE de 3A, je n'avais plus les réflexes et était perdu. Après avoir demandé conseil à Mr. Flammen, j'ai pu commencer un peu plus sereinement. Tout d'abord j'essayais de mettre tout en place en reliant les masses, alors que celles-ci sont fondues dans le plan de masse à la toute fin. Ensuite j'ai utilisé mon schémas en parallèle pour suivre le placement des composants.

L'élaboration du PCB a pris plusieurs jours de mon programme. Plusieurs versions sont sorties de cela, avec à chaque fois des changements mineurs à appliquer (piste avec un angle aigus, orphelins etc..). Une fois que tout était correctement en place j'ai pu lancer en production la carte.

[10] Pour plus d'informations : http://www.timstinchcombe.co.uk/index.php?pge=trisin
https://wiki.analog.com/university/courses/electronics/electronics-lab-12sg

Remarque: J'avais oublié de mettre des capacités de découplage autour de chaque amplificateur opérationnel. Multisim permet de les rajouter et de mettre à jour le PCB sans perdre les placements effectués auparavant'.

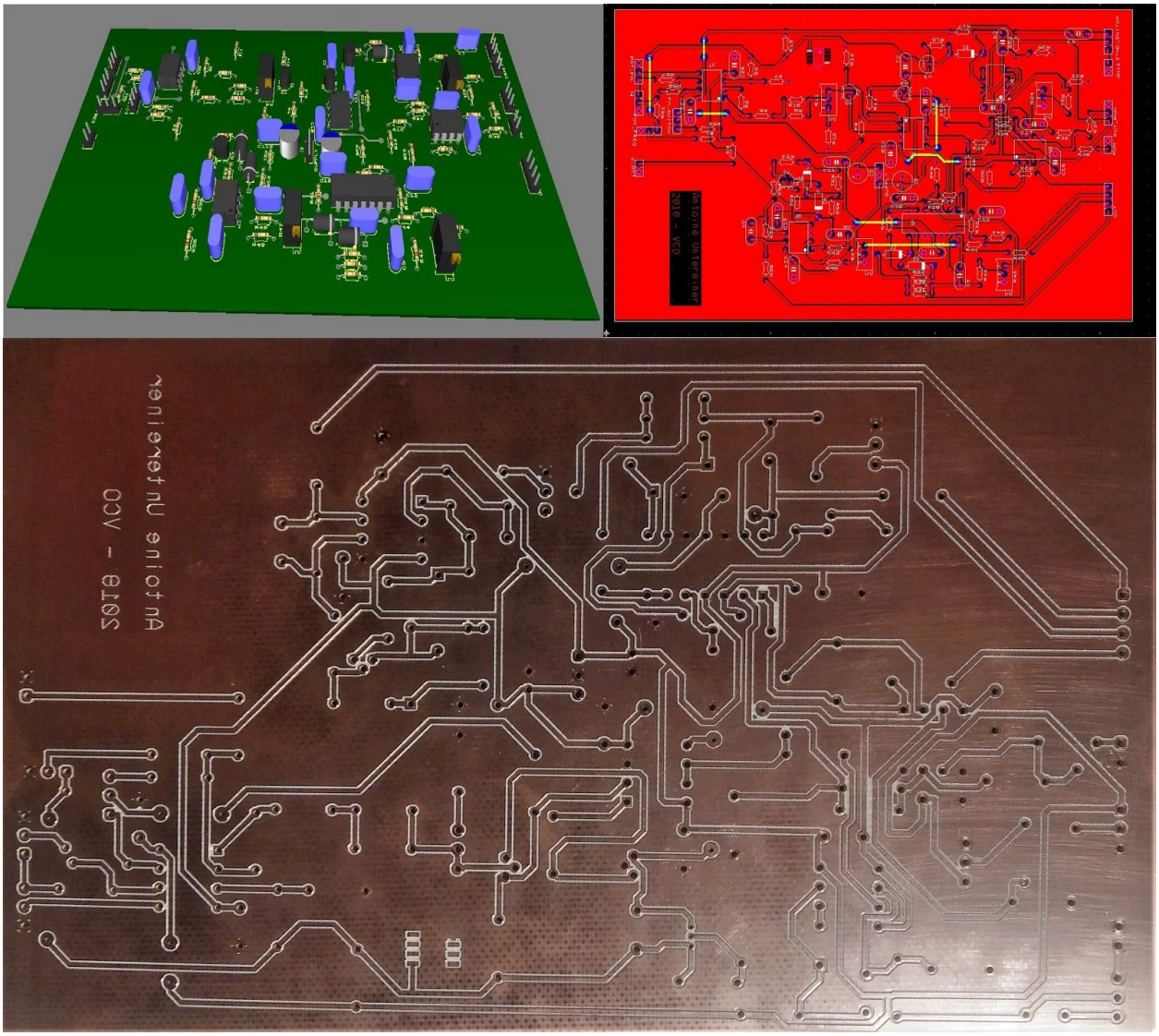

Fig 16. Elaboration du PCB.

IV/ Travail effectué / Travail restant

A l'heure où j'écris ce rapport, plusieurs des fonctionnalités essentielles au synthétiseur ont été réalisées:

- ➢ Gestion du Midi : Le DAC est commandé. J'ai cependant effectué le test avec un autre dac et cela fonctionne.
- ➢ Conception du VCO : Le PCB est imprimé. Les composants sont commandés et ne devraient pas tarder à arriver. Je les ai fait arriver chez mes parents pour pouvoir souder le tout pendant les vacances.
- ➢ Codage de l'ADSR : Le code est quasiment terminé, il me manque que quelques éléments pour pouvoir lancer la phase de test.
- ➢ Réalisation du filtre : De même cette partie est presque finie. Je ne la présente pas encore je ne me juge pas encore en mesure d'expliquer les notions théorique nécessaire. Celui-ci se base sur un type de filtre. Le schéma est déjà mis en place et simulé, le PCB est presque fini.

► Il me reste donc à souder les composants pour les parties déjà effectuées. Côté conception, je dois encore me pencher sur le VCA. Mais j'ai déjà des pistes de réflexion conséquentes.
► Il n'y aura plus qu'à mettre les différents éléments en lien et tester le tout. Je ne m'attend pas à un essai réussi dès la première fois. Il me faudra donc un temps pour débugger le circuit.
► Coté boitier et design, j'ai envoyé les différents éléments à mon frère qui va réfléchir à une disposition ergonomique et esthétique des potentiomètres/switchs.
► Selon le temps restant je pourrais travailler sur des effets intégrés à synthétiseur. Le fait d'avoir séparé les composants du synthétiseurs en différents blocs, cela me permet d'ajouter des blocs fonctionnels. Ce qui, à l'avenir, me permettra de faire évoluer sans cesse ce synthétiseur.