Compte rendu de Projet, Système Programmable [MU4IN108]

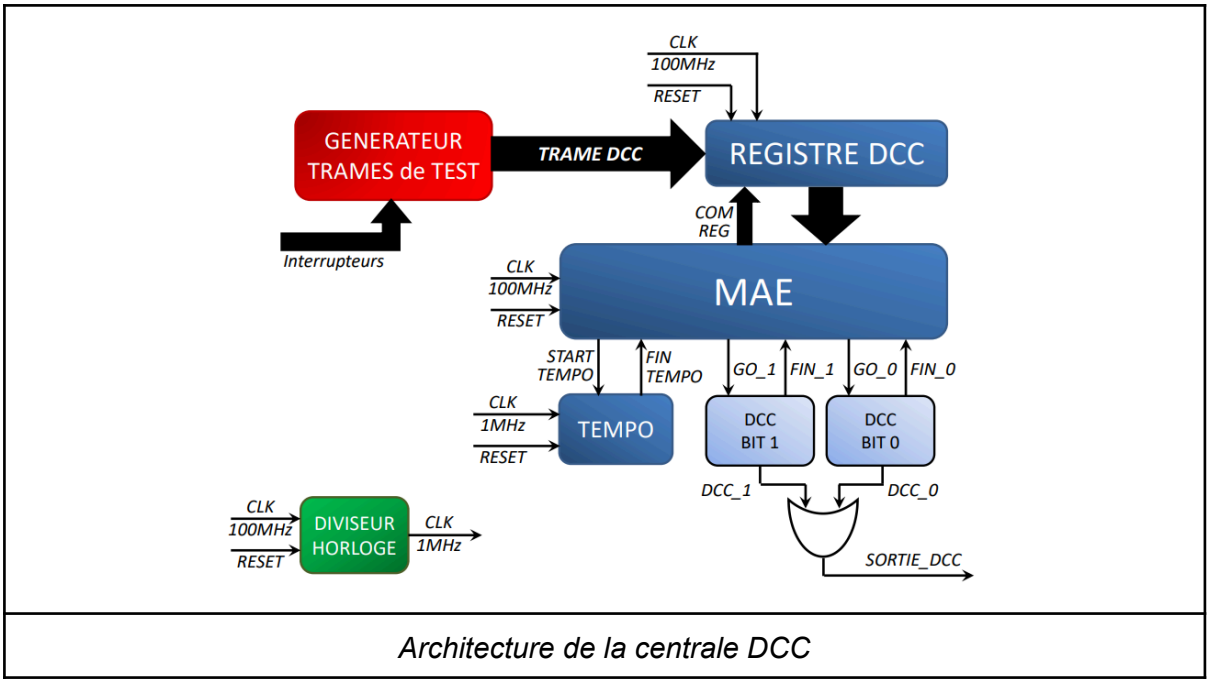
Haron DAUVET (M1 SAR)

Kavish RAGHUBAR (M1 SESI)

Introduction

Maintenant que nous avons exploré les bases de conception d'un projet en utilisant une carte FPGA, nous pouvons désormais réaliser notre premier projet concret. Le protocole DCC (Digital Command Control) est un standard utilisé dans le modélisme ferroviaire pour commander individuellement des locomotives ou des accessoires en modulant la tension d'alimentation de la voie. Le but de ce projet est donc de comprendre et modéliser une architecture capable d'envoyer une trame de commande lisible par une locomotive, nous synthétiserons alors cette IP **Centrale DCC** qui se chargera de générer des trames, enfin nous créeront une implémentation logiciel qui pilotera notre matériel via des commandes simples, finalement nous aurons un matériel et une abstraction logiciel pour piloter une locomotive. Vous trouverez dans ce rapport des explications simples pour tous les modules de l'IP avec leurs schémas et leurs testbench.

IP Centrale DCC



Architecture de la centrale DCC

Le principe de fonctionnement de notre IP est la suivante :

Une Trame DCC écrite en dur est décodé bit par bit par le module **Registre DCC,** celui-ci communiquera avec le module **MAE (globale)** qui est en quelques sorte le chef d'orchestre de l'IP, celui-ci récupère le bit à écrire fournit par **Registre DCC**, et le transmet à **DCC_BIT_0** ou **DCC_BIT_1** qui générera le bit voulue en sortie de l'IP, entre chaque trame le module **Tempo** temporisera notre sortie comme le spécifie le protocole DCC.

Le module **Diviseur Horloge** offre un signal de fréquence 1 MHz au module **Tempo** et aux modules générateur **DCC_BIT_0** et **DCC_BIT_1**.

Le module **Tempo** communique avec la **MAE** pour temporiser entre les trames, la **MAE** envoie un signal de lancement et le module envoi un signal en retour pour indiquer que la temporisation est finie.

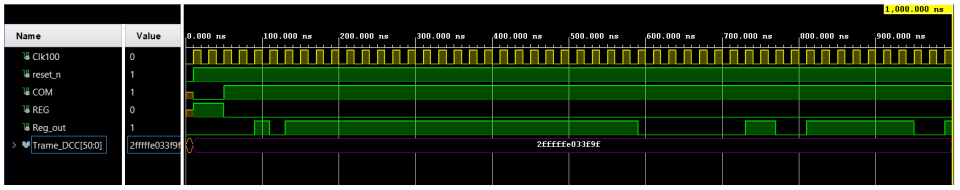
Le module **Générateur Trames de Test** permet de tester l'entièreté de notre IP, avec ce module on passera une trame en entrée de **Registre DCC**. Ce module n'est que temporaire dans cette partie car il sera ensuite remplacé par un pilote **Microblaze,** qui nous permettra d'envoyer nos propres trames en temps réel à notre IP.

Tous les modules sont reliés à un reset asynchrone et sont cadencés par l'horloge global du système (100 MHz).

Registre DCC

Ce module prend en entrée *TRAME_DCC* qui est un vecteur de 51 bits (c'est le nombre de bits maximum dans une trame, 42 est le nombre de bits minimum), 2 bits de contrôle *COM* et *REG* fourni par la **MAE global** : si REG est à 1, la MAE demande à charger une trame si

COM est à 1, la MAE demande à sélectionner le bit suivant dans la trame chargé, chaque bit est envoyé à travers un signal *Reg_out.*



TestBench de Registre DCC

MAE

Le Module **MAE globale** du système permet de contrôler et faire communiquer tous les modules de l'IP entre eux, on définit 9 états où *load* est le point d'entrée :

- load : État d'entrée, chargement d'une trame et initialisation des signaux internes (compteurs)
- shift : Décalage (lecture de la trame)
- compare : sélectionne le bit à généré
- bit1 : Le bit à généré est un 1
- bit0 : Le bit à généré est un 0
- wait1 : On attend que la génération d'un bit 1 soit fini, si il reste des bits dans la trame on shift
- wait0 : On attend que la génération d'un bit 0 soit fini, si il reste des bits dans la trame on shift
- tempo : La trame à été généré, on démarre la temporisation
- wait_tempo : On attend la fin de la temporisation

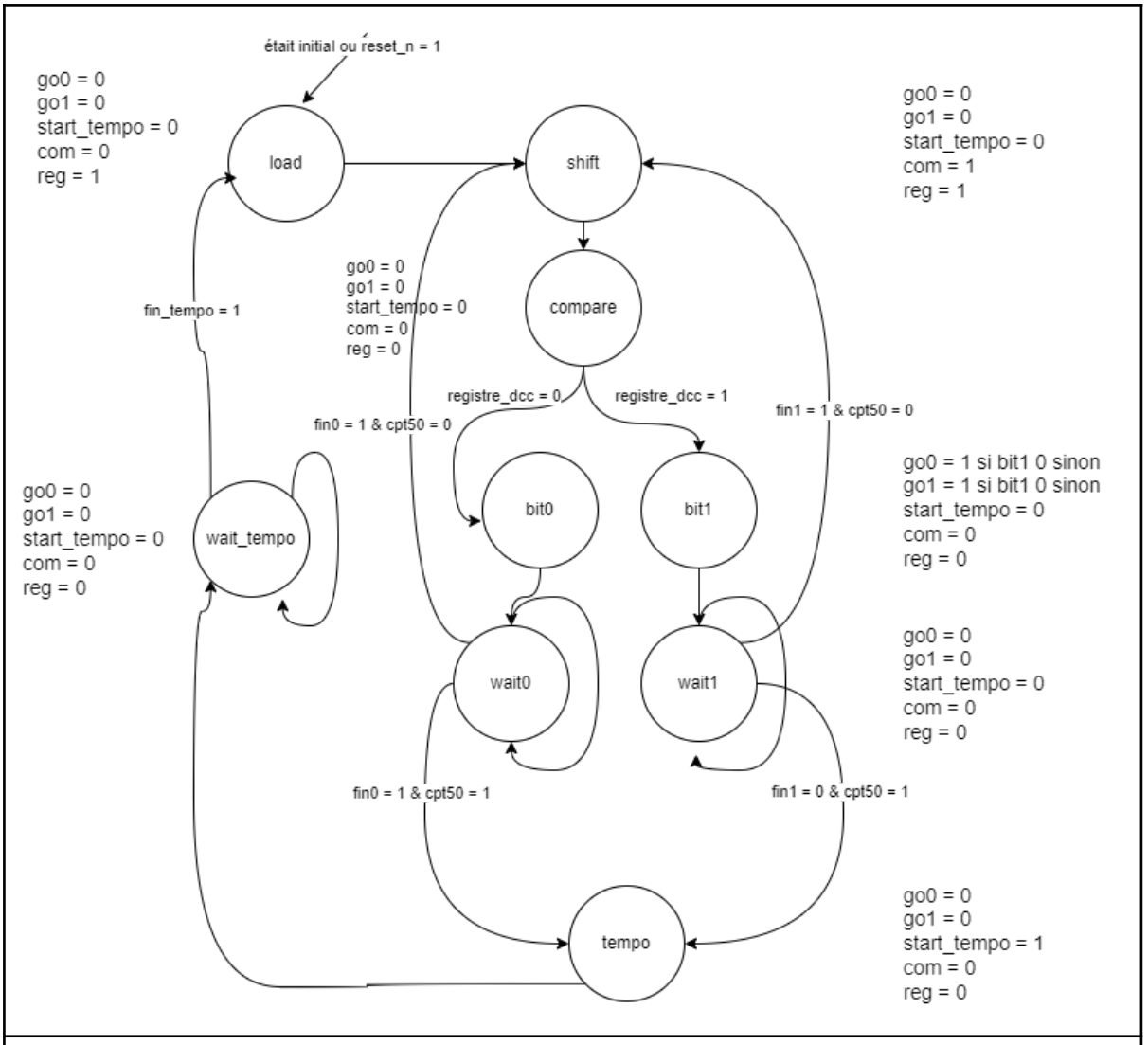
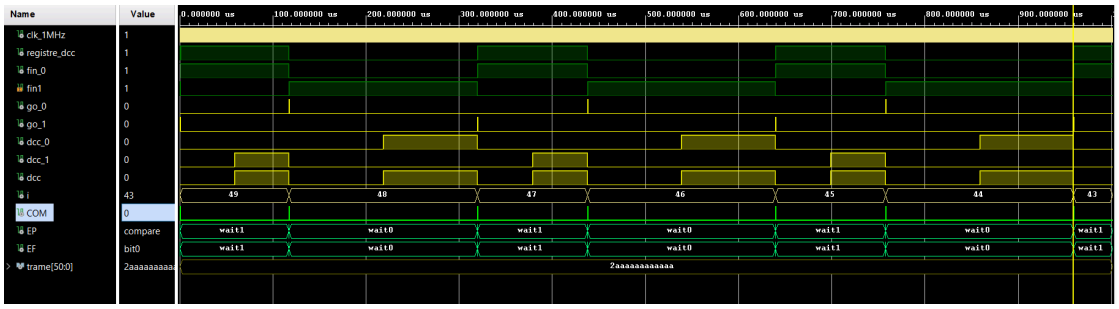
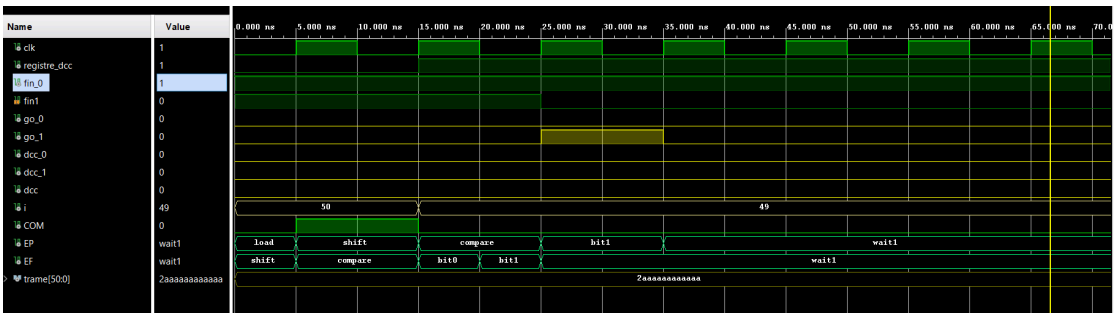


Schéma de la MAE global de l'IP





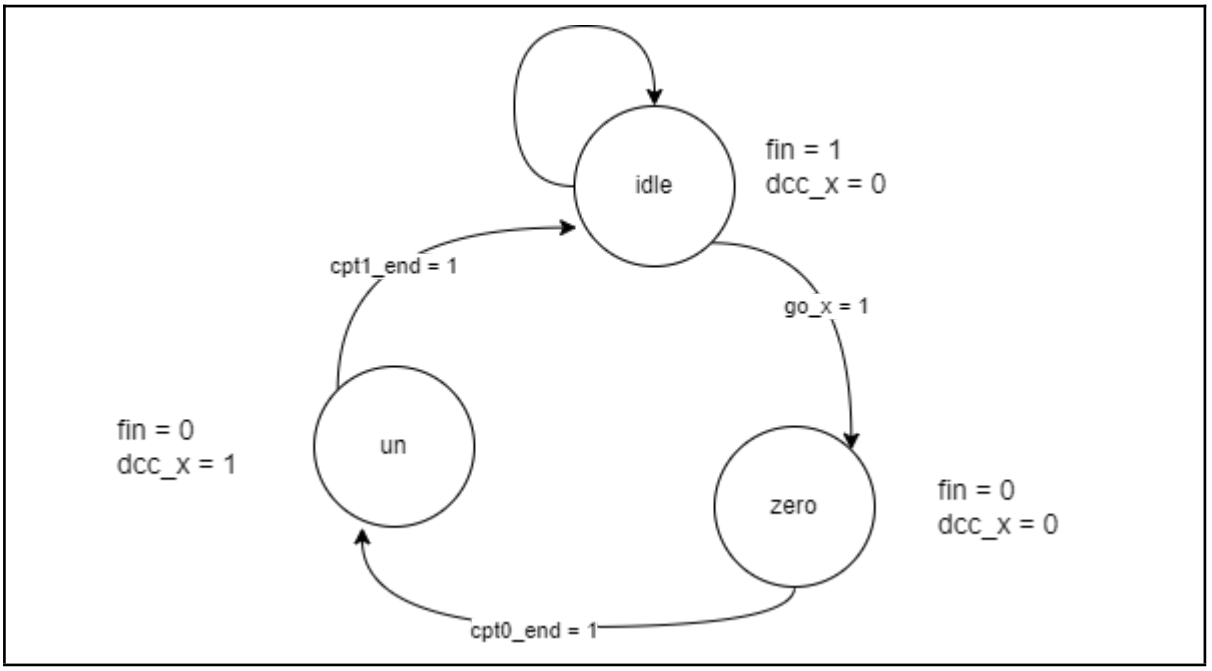
TestBench et zoom sur la testbench de la MAE global

DCC_BIT_0 et DCC_BIT_1

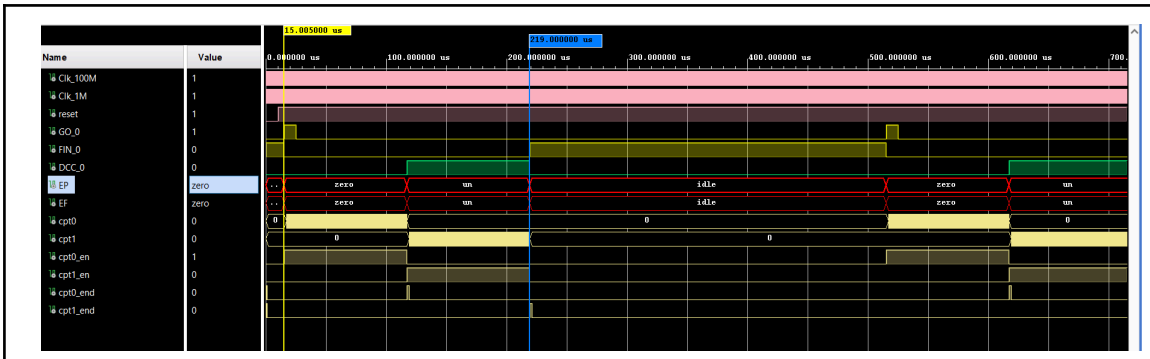
Ces modules génèrent en direction de la sortie de l'IP le bit de la trame lu par **Registre DCC** et acheminé jusqu'ici par la **MAE.** Ces modules communiquent avec la **MAE global** par 2 signaux, ils reçoivent *go* pour leur indiquer de générer un bit. D'après la documentation DCC, générer un bit à 0 consiste à créer un front montant et descendant de 100 µs chacun et pour un bit à 1 des fronts de 58 µs chacun. une fois le bit émis les modules répondent par *fin*.

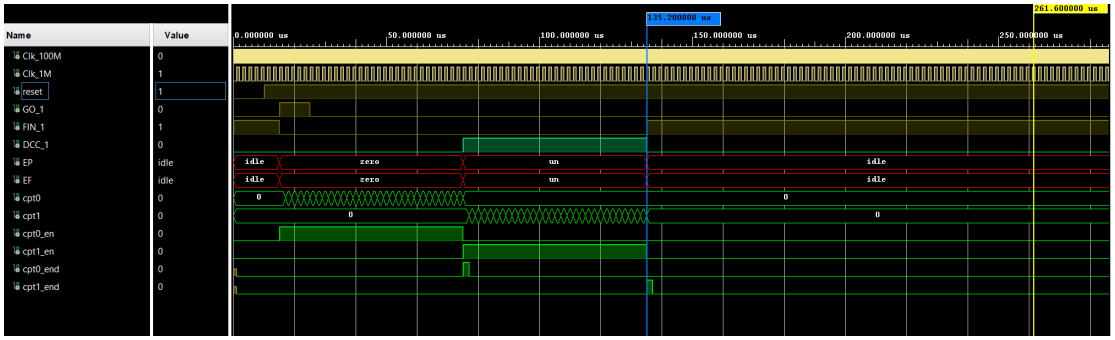
Pour ces modules (parfaitement analogue l'un l'autre) il est préférable de faire une machine à état que nous décrivons ci dessous :

- idle : état initial, reste dans ce état temps qu'il n'a pas reçu de départ de la **MAE global** et initialisations des compteurs
- zero : génération du front d'onde descendant, un compteur se lance
- un : génération du front montant, un compteur se lance



MAE des modules DCC_BIT





TestBench de DCC_BIT_0 et DCC_BIT_1

Générateur de Trames de Test

Ce module à été légèrement complété pour s'adapter à notre utilisation :
A ce stade il n'était pas important de sélectionner l'adresse de la locomotive, on a donc stocké l'adresse dans une variable *adr*

```
signal adr : std_logic_vector(7 downto 0) := x"02";
```

Pour le reste, pour chaque interrupteur (activé dans le fichier de contraintes) on complète un signal de sortie *Trame_DCC* qui ira en entrée de **Registre DCC**.

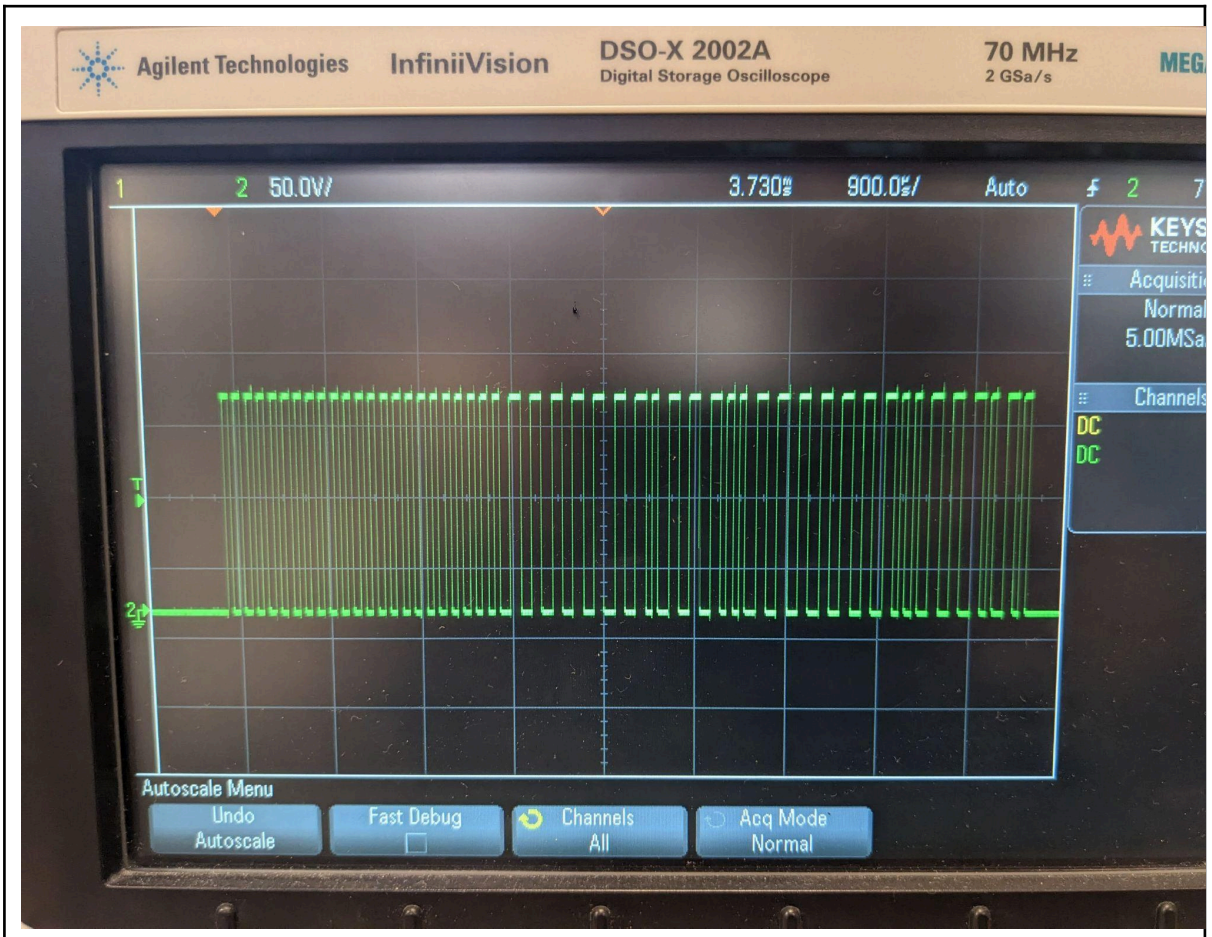
Une trame se compose de :

1. Un préambule de 14 à 23 bits à 1
2. Un bit start à 0
3. Une adresse sur 1 octet
4. Un bit start à 0
5. 1er octet de commande
6. Un bit start à 0
7. 2eme octet de commande
8. Un bit start à 0
9. L'octet de contrôle étant le XOR de 2 5 et 7

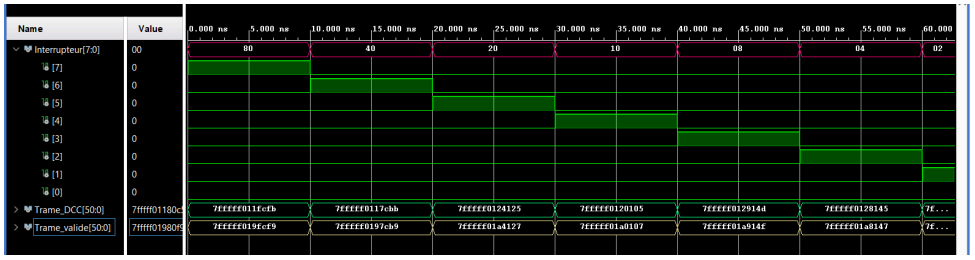
10. Un bit de stop à 1

Notre IP ne différenciant pas les trames de 1 octets de commande ou 2 octets de commande, nous avons rusé en transmettant un préambule plus grand de 1 octet, comme ca toutes les trames ont une longueurs de 51 bits.

Avant de tester notre IP sur les trains, nous nous sommes assuré qu'on avait bien une trame généré correctement sur l'oscilloscope :



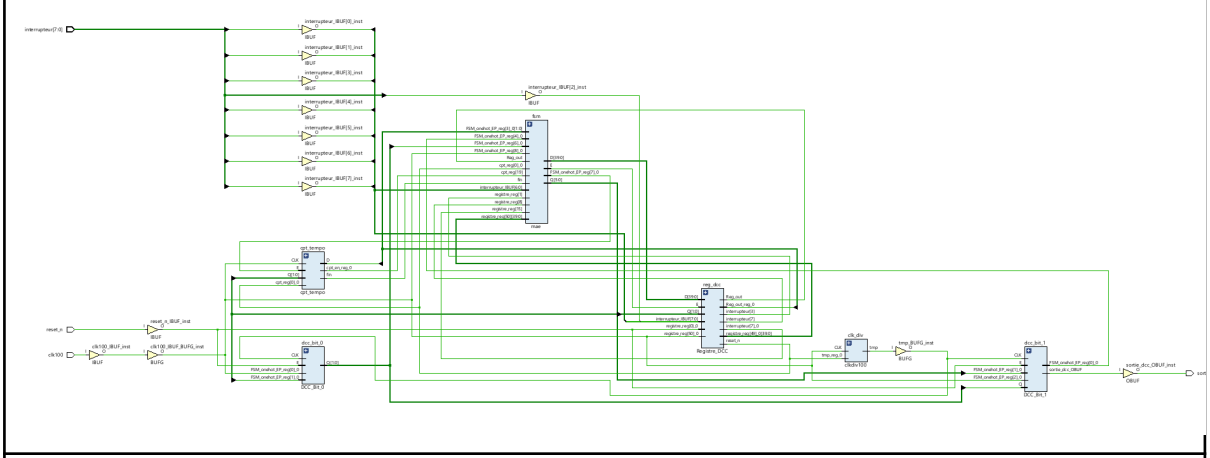
Trame généré sur l'oscilloscope par la broche J4



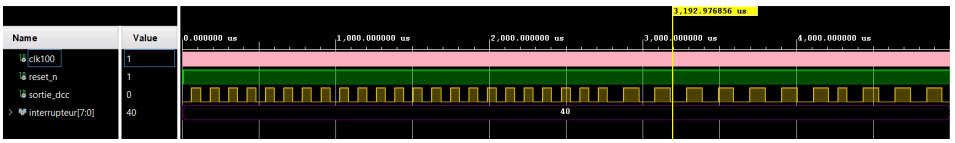
TestBench du générateur de trame

Top_DCC

TOP_DCC est le module qui encapsule tous les autres, il permet de mapper les port de chaque module pour les relier entre eux et activer les périphériques (leds, switchs et broches) , une fois le bit stream effectué, on remarque que le schéma de l'architecture généré ressemble bien au schéma du design présenté plus haut :



Le schéma de l'architecture synthétisé

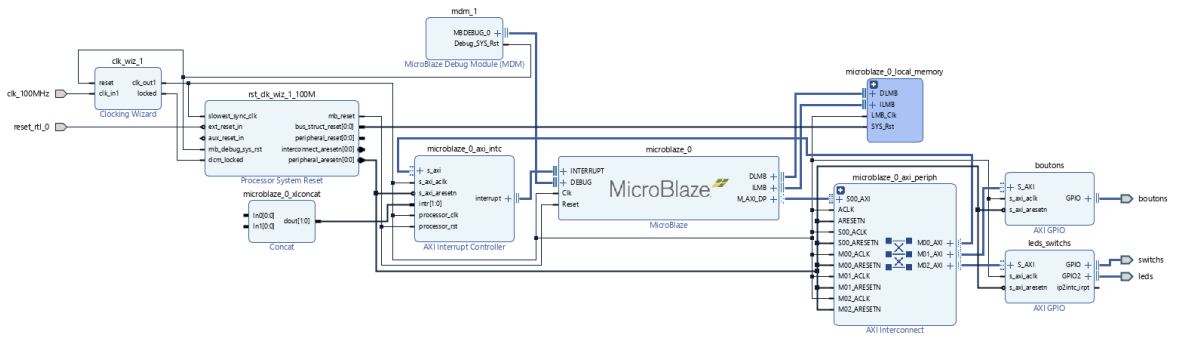


TestBench de TOP DCC

Centrale DCC comme IP dans un système Microblaze

Intégration de l'IP au système Microblaze

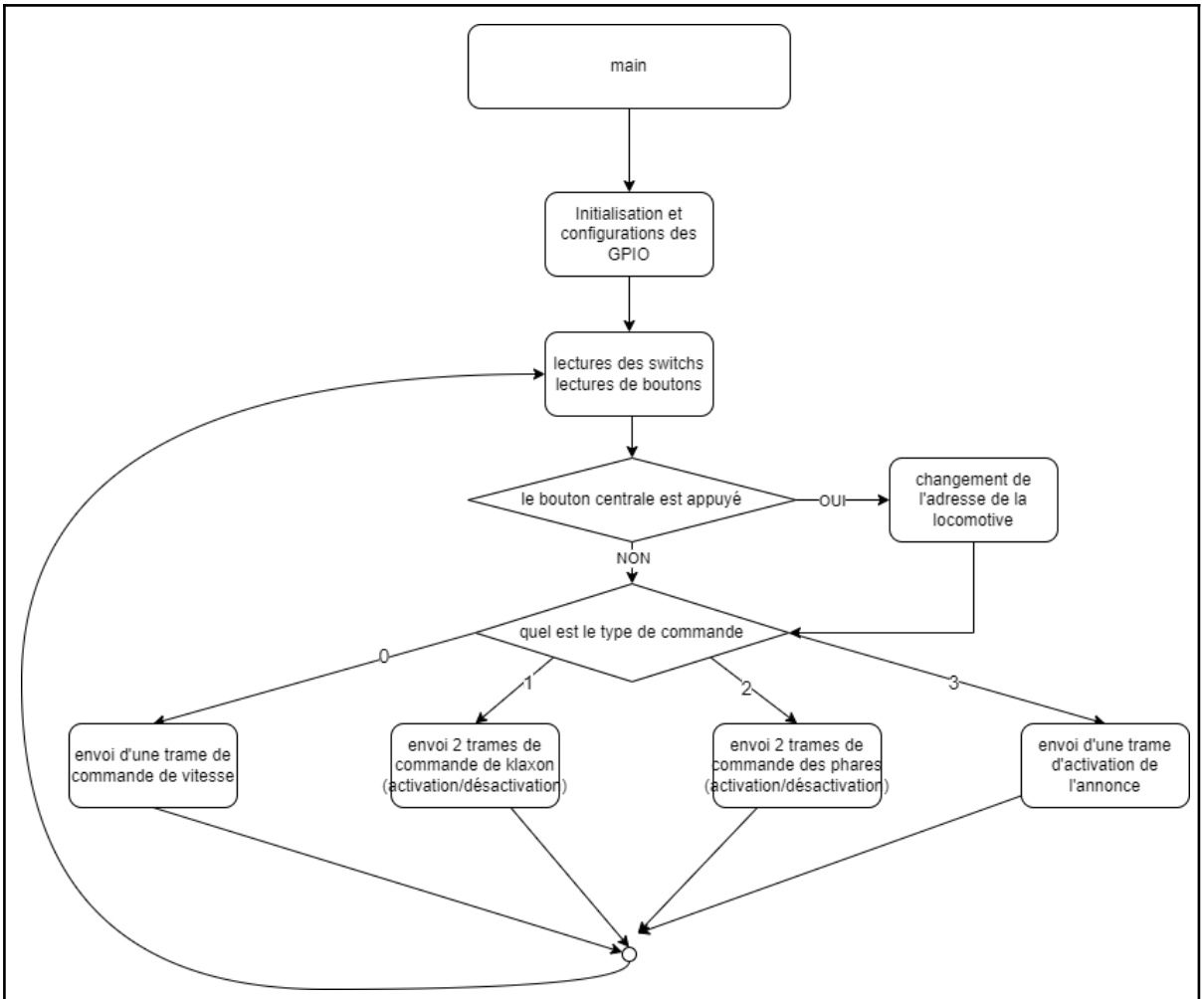
Nous créons un projet VHDL dans lequel nous ajoutons l'IP ***Microblaze,*** que nous configurons pour ajouter tous les modules nécessaires à notre processeur.
On ajoute ensuite deux IP ***AXI GPIO*** que nous configurons pour le contrôle des led/switch et des boutons left center et right.



l'IP Microblaze avec l'IP DCC

Développement logiciel

Pour le développement sur Vitis, notre programme *main.c* permet de générer une trame en fonction de comment l'utilisateur interagit avec les switchs et boutons, les leds permettent à l'utilisateur de visualiser les commandes qu'il a rentré.
Pour cela nous avons défini plusieurs macro pour l'utilisation des masques pour les slaves register et les types de commandes (avec les préambule de 14 à 23 bits) .
Ensuite nous avons défini plusieurs fonctions pour envoyer des commandes en fonction de l'état des switchs.
Dans le corps principale du programme nous initialisons nos périphérique et démarrons une boucle infini qui lit (en permanence) l'état des switchs et des boutons, en fonction de *type* la variable globale de la commande actuel (mis à jour par nos fonctions) nous composons nos trames et l'écrivons aux adresses des slaves register de notre IP. Cette fois ci les switchs permettent de sélectionner l'adresse de la locomotive, l'architecture du code permet aussi de régénérer la dernière trame entré comme le préconise la documentation DCC.



Organnigramme du code principal

Conclusion

Le sujet proposé à pu mettre en expérience toutes les compétences nécessaires à la confection d'un contrôleur simple. Nous sommes parti d'un schéma d'IP et d'un cahier des charges, afin de mettre en pratique la gestion d'horloge, les machines à état et les entrées sorties. Grâce à la manipulation de logiciel comme Vivado, nous avons expérimenté un nouveau workflow de design matériel et logiciel, intégré d'autre IP comme Microblaze, simuler le comportement des descriptions et soumis notre carte à des contraintes matérielles. Pour aller jusqu'au bout de ce projet, nous avons suivi les directives pour développer un pilote sous le logiciel Vitis. Enfin, nous avons pu expérimenté nos travaux, en salle de TP, avec le matériel nécessaire aux tests et vérifications de notre implémentation. La centrale DCC est certainement le premier projet qui nous initiera dans l'électronique des grandes industries.