SORBONNE UNIVERSITÉ

TP - FPGA

Central DCC

Auteur : Yong LI
Marc ZHAN

19 mai 2023

Table des matières

1 Introduction

Ce projet consiste à réaliser la partie pour générer le signal qui contrôle le Train. Donc, ce que notre objet est de générer le signal spécifique en entrant les informations par les interrupteurs sur la carte FPGA. avec les informations recueillies, nous devons produire le signal qui obéit au DCC protocole afin que le système de train puisse comprendre. mais le signal de sortie de la carte FPGA n'est pas suffisant en puissance, nous devons envoyer le signal à un supérieur pour obtenir le signal final et l'envoyer au système de train par les locomotives.

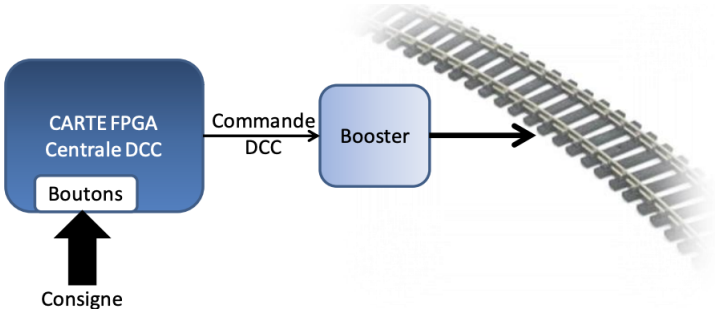

FIGURE 1 – Synoptique du système de commande

2 Protocole DCC

Pour commencer le projet, nous devons comprendre la fonction du protocole DCC. Protocole DCC est un standard utilisé dans le modélisme ferroviaire pour commander individuellement des locomotives ou des accessoires de voie en modulant la tension d'alimentation de la voie.

2.1 Représentation de bit

Un bit est représenté par une impulsion à 0 du signal de sortie, suivie d'une impulsion à 1. La durée des impulsions permet de différencier un bit à 0 d'un bit à 1. L'ordre des impulsions (à 0 puis à 1) doit être obligatoirement respecté

Bit	Représentation
0	Impulsion à 0 de 100 µs puis Impulsion à 1 de 100 µs
1	Impulsion à 0 de 58 µs puis Impulsion à 1 de 58 µs

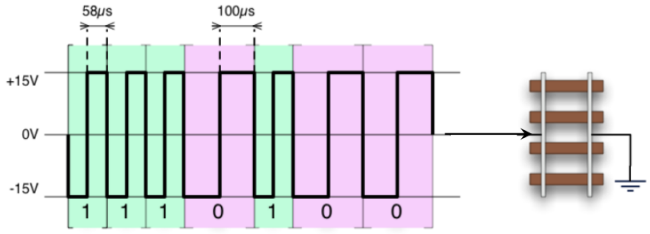

FIGURE 2 – Synoptique du système de commande

Les trames sont générées en permanence. Donc entre chacun des deux trames, il y a un espace d'un Intervalle de 6 ms.

2.2 La composition de la trame

Chaque trame est composée de 4 champs : Dans le détail, cela donne :

Composition de la Trame	Détail
Préambule	Suite d'au moins 14 bits à 1
Start Bit	1 bit à 0
Champ d'adresse	Adresse de la locomotive à commander (1 octet)
Start Bit	1 bit à 0
Champ de commande	Commande envoyée au train (de 1 à 2 octets*)
Start Bit	1 bit à 0
Champ de contrôle	XOR entre les octets des deux champs précédents, (1 octet). Permet de détecter d'éventuelles erreurs de transmission
Stop Bit	1 bit à 1

* Si le champ de commande comprend plus d'un octet, chaque octet sera suivi d'un bit à 0.

Table 1 – Composition de la trame

Ici, une figure qui donne l'allure d'une trame comprenant un champ de commande sur 1 octet.

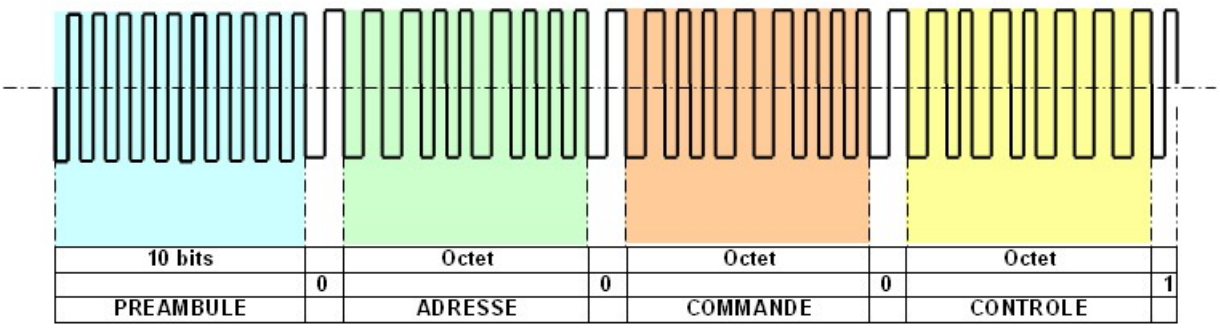

Figure 3 – Exemple d'une trame DCC

2.3 Champ de commande

Le champ de commande permet de contrôler la vitesse d'un train, mais aussi d'activer ou de désactiver les fonctions de la locomotive (phares, signaux sonores divers)

2.3.1 Contrôler la vitesse d'un train

La commande permettant de gérer la vitesse d'un train est codée sur 1 octet, avec le format suivant :

- 01DXXXXX
- Le bit D fixe la direction du train : 0 pour une marche arrière, 1 pour une marche avant.
- Les 5 bits de poids faible indiquent la vitesse de la locomotive, selon le codage suivant (le Step 28 correspond à la vitesse maximale) :

$CS_3S_2S_1S_0$	speed	$CS_3S_2S_1S_0$	speed	$CS_3S_2S_1S_0$	speed	$CS_3S_2S_1S_0$	speed
00000	Stop	00100	speed 5	01000	speed 13	01100	speed 21
10000	Stop (I)	10100	Step 6	11000	Step 14	11100	Step 22
00001	E-stop*	00101	Step 7	01001	Step 15	01101	Step 23
10001	E-stop* (I)	10101	Step 8	11001	Step 16	11101	Step 24
00010	Step 1	00110	Step 9	01010	Step 17	01110	Step 25
10010	Step 2	10110	Step 10	11010	Step 18	11110	Step 26
00011	Step 3	00111	Step 11	01011	Step 19	01111	Step 27
10011	Step 4	10111	Step 12	11011	Step 20	11111	Step 28

TABLE 2 – Configuration du vitesse

2.3.2 Les fonctions de la locomotive

Les locomotives possèdent 22 fonctions, étiquetées de F0 à F21 et détaillées dans le tableau ci-dessous, issu de la documentation (en traduction approximative de l'allemand vers le français...).

KEY	FUNCTION	KEY	FUNCTION
F0	Lumière on / off	F11	Court Cor Française #1
F1	Son on / off	F12	Court Cor Française #2
F2	Cor Française #1	F13	L'Annonce station française #1
F3	Cor Française #2	F14	L'Annonce station française #2
F4	Turbo off	F15	Signal d'alerte française #1
F5	Compresseur	F16	Signal d'alerte française #2
F6	Accélération / freinage temps, ...	F17	Porte chauffeur ouvrir / fermer
F7	Courbe grincement	F18	Valve
F8	Ferroviaire Clank	F19	Attelage
F9	Ventilateur	F20	Sable
F10	Conducteur de signal	F21	Libération des freins

TABLE 3 – Les fonctions de la locomotive

La gestion de ces fonctions s'effectue de la façon suivante :

- **Fonctions F0 à F4** - Champ de commande sur 1 octet, de la forme suivante
  - o 100XXXXX
  - o Le bit 4 sert à gérer la fonction F0 (Bit à 1 : Phares allumés, Bit à 0 : Phares éteints)
  - o Les bits 3-0 gèrent les fonctions F4 à F1, dans cet ordre (Bit à 1 : Fonction ON, Bit à 0 : Fonction OFF)

- **Fonctions F5 à F12** - Champ de commande sur 1 octet, de la forme suivante
  - o 101SXXXX
  - o Le bit S sert à sélectionner le groupe de fonctions F5-F8 (Bit à 1) ou F9-F12 (Bit à 0).
  - o Si S=1, les bits 3-0 gèrent les fonctions F8 à F5, dans cet ordre (Bit à 1 : ON, Bit à 0 : OFF)
  - o Si S=0, les bits 3-0 gèrent les fonctions F12 à F9, dans cet ordre (Bit à 1 : ON, Bit à 0 : OFF)

- **Fonctions F13 à F20** - Champ de commande sur 2 octets, de la forme suivante
  - o 11011110 XXXXXXXX
  - o Les 8 bits du 2ème octet gèrent chacun une fonction (Bit 7 pour F20, Bit 0 pour F13)
  - o Comme pour les autres groupes de fonctions, la fonction est ON si son bit est à 1, elle est OFF sinon

Attention, pour certaines fonctions générant un son ponctuel (les coups de klaxon F11-F12, les messages d'annonce F13-F14), il est nécessaire, pour les générer plusieurs fois consécutivement, de désactiver la fonction avant chaque nouvelle activation.

Ainsi pour émettre deux coups de klaxon (fonction F11), il faudra émettre 3 trames : 1 trame pour activer F11 (1er coup), 1 trame pour désactiver F11, et enfin 1 trame pour activer F11 (2ème coup)

2.4 Le champ de contrôle

Le champ de contrôle est un octet qui est le résultat d'un XOR entre tous les octets des champs précédents.

- Par exemple, si on souhaite activer la fonction F14 du train d'adresse 2, les octets des différents champs de la trame seront

Octet	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Adresse	0	0	0	0	0	0	1	0
Commande (1er octet)	1	1	0	1	1	1	1	0
Commande (2ème octet)	0	0	0	0	0	0	1	0
	↓	↓	↓	↓	↓	↓	↓	↓
Contrôle (XOR)	1	1	0	1	1	1	1	0

TABLE 4 – Le champ de contrôle

On ajoute que :

- La trame (et les différents octets) est transmise du poids fort au poids faible
- Il n'est pas possible d'insérer plusieurs commandes dans une même trame (par exemple une trame qui inclurait une commande de vitesse d'un train tout en allumant ses phares). Il faut envoyer deux trames pour cela.
- Le protocole DCC prévoit une tolérance de 3 µs sur les timings indiqués plus haut. Il est cependant important de les respecter au mieux.
  - o En cas de dépassement, le décodage des trames par les locomotives n'est plus garanti !

3 Architecture de la centrale DCC

L'architecture de la Centrale DCC est présentée en figure ci-dessous :

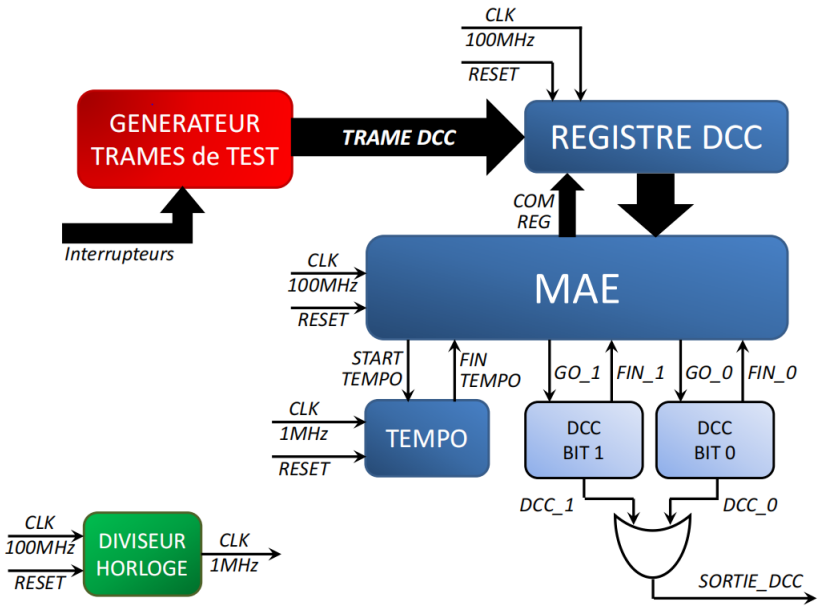

FIGURE 4 – Architecture de la Centrale DCC

3.1 DIVISEUR HORLOGE

Ce module génère un signal d'horloge de fréquence 1 MHz, à partir de l'horloge 100 MHz de la carte. Cette horloge **CLK_1MHz** sert à générer les délais requis par le protocole **DCC**.

3.1.1 Portes

porte d'entrée

- Clk_in : Horloge 100 MHz de la carte Nexys/Basys3
- Reset : Reset Asynchrone, réinitialiser le module pour que le signal de sortie et le compteur à l'intérieur du module soient réinitialisés

porte d'sortie

- Clk_out : Horloge 1 MHz de sortie, qui se sont connectés au signal interne

signal interne

- Div : compteur à l'intérieur du module chaque fois que front montant du signal d'horloge, il va soit augmenter 1 ou réinitialiser à 0, qui va également contrôler le changement du signal interne Clk_temp
- Clk_Temp : 1 ou 0 qui est connecté au port de sortie Clk_out

3.1.2 principe de fonctionnement

ce module fonctionne assez simple, il a un compteur à l'intérieur et quand le compteur atteint la valeur 49 il changera la valeur du port de sortie de sorte que chaque deux fois de commutateur est une horloge.

3.1.3 Simulation

nous avons fait une simulation pour consolider le module fonctionne bien :

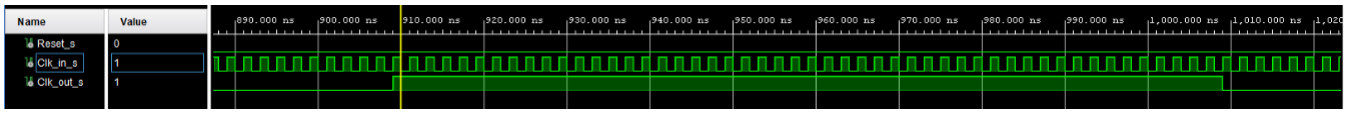

FIGURE 5 – Simulation du module DIVISEUR HORLOGE

3.2 TEMPO

Ce module, cadencé par l'horloge **CLK_1MHz** sert à mesurer l'intervalle de temps qui doit séparer deux trames DCC (soit 6 ms).

- La commande **Start_Tempo**, générée par la **MAE** (Machine à Etats), doit permettre à un compteur de compter le délai de temporisation.
- Lorsque ce délai est écoulé, un signal **Fin_Tempo** est mis à 1..
  - o Ce signal est automatiquement remis à 0 dès que la commande **Start_Tempo** est remise à 0.

3.2.1 Portes

porte d'entrée

- Clk : Horloge 100 MHz de la carte Nexys/Basys3
- Reset : Reset Asynchrone
- Clk1M : Horloge 1 MHz
- Start_Tempo : Drapeau de Fin de la Temporisation

porte d'sortie

- Fin_Tempo : Horloge 1 MHz de sortie, qui se sont connectés au signal interne

3.2.2 principe de fonctionnement

ce module contient deux parties principales : l'une est un compteur, l'autre est un MAE.

Compteur

le compteur est contrôlé par 2 signaux : Raz_Cpt et Inc_Cpt et dispose d'un signal de sortie Fin_Cpt

- Quand le compteur reçoit le signal raz_cpt il réinitialisera le compteur.
- Quand le compteur reçoit le signal inc_cpt il comptera une fois chaque front montant d'horloge
- le signal de sortie Fin_Cpt sera égal à 1 lorsque le compteur atteint la valeur 5999

MAE

Pour le MAE, il fonctionne comme la figure ci-dessous :

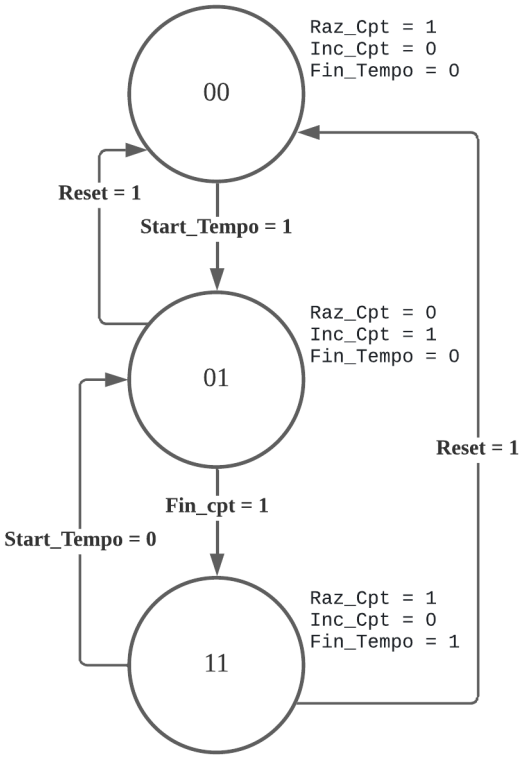

FIGURE 6 – MAE du module TEMPO

3.2.3 Simulation

nous avons fait une simulation pour consolider le module fonctionne bien :

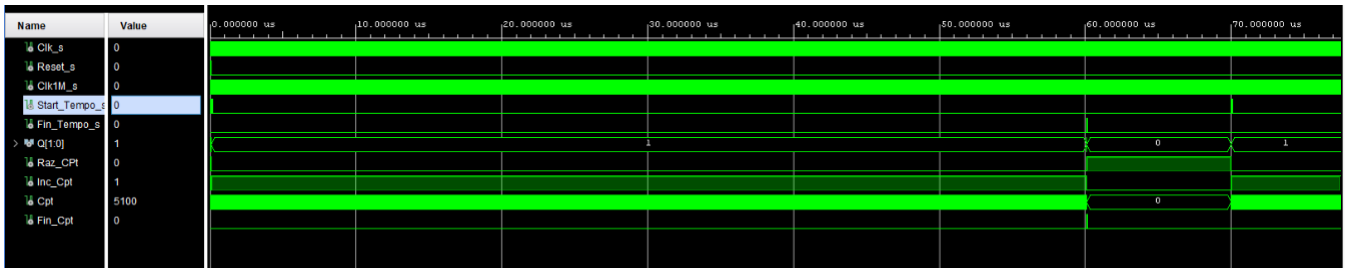

FIGURE 7 – Simulation du module TEMPO

3.3 DCC_Bit_1 / DCC_Bit_0

- Ces deux blocs permettent de générer respectivement un bit à 1 ou à 0 au format DCC (cf. plus haut)
  - o Ils sont tous les deux cadencés par les 2 horloges (CLK_100MHz et CLK_1MHz) et sont connectés au Reset du système (ce n'est pas représenté sur le schéma pour plus de clarté)
- Chaque module est piloté par la MAE globale du système.
  - o La commande Go active la génération du bit.
  - o Le signal Fin indique que la transmission du bit est terminée.
- Chaque bloc possède une machine à états et un compteur
  - o La machine à états est cadencée par l'horloge CLK_100MHz. Son rôle est de :
    - Réaliser l'interaction avec la MAE globale du système.
    - Positionner le signal de sortie au niveau logique opportun.
  - o Le compteur est cadencé par l'horloge CLK_1MHz.
    - Réaliser l'interaction avec la MAE globale du système.
    - Son rôle est de compter les temps pendant lesquels le signal de sortie doit être à 1 ou 0, conformément au protocole DCC (voir plus haut).
- Lorsque le module n'est pas activé, le signal de sortie doit être au niveau bas.

On trouvera en sortie des modules DCC_Bit_0 et DCC_Bit_1. une porte OU qui joint les signaux de sortie de ces deux modules.

3.3.1 Portes

porte d'entrée
- Clk : Horloge 100 MHz de la carte Nexys/Basys3
- Reset : Reset Asynchrone
- Clk1M : Horloge 1 MHz
- GO_0/GO_1 : Commande pour commencer à envoyer un signal bit

porte d'sortie
- FIN_0 : Drapeau fin de l'envoi d'un signal bit
- DCC_0/DCC_1 : Volume de tension de signal du port de sortie

3.3.2 principe de fonctionnement

ce module contient deux parties principales : l'une est un compteur, l'autre est un MAE.

Compteur

le compteur est contrôlé par 2 signaux : Raz_Cpt et Inc_Cpt et dispose d'un signal de sortie Fin_Cpt

- Quand le compteur reçoit le signal raz_cpt il réinitialisera le compteur.
- Quand le compteur reçoit le signal inc_cpt il comptera une fois chaque front montant d'horloge
- le signal de sortie Fin_Cpt sera égal à 1 lorsque le compteur atteint la valeur 5999

MAE

Pour le MAE, il fonctionne comme la figure ci-dessous :

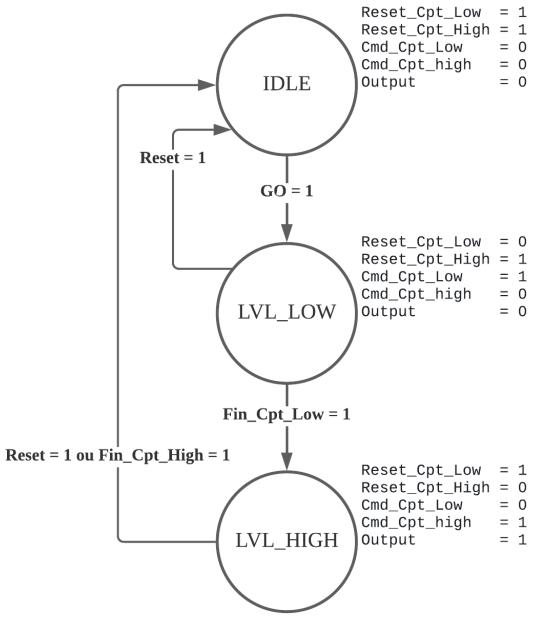

FIGURE 8 – MAE du module DCC0/DCC1

Une fois que le MAE a reçu une commande GO pour envoyer un bit il va commencer le premier compteur pour envoyer 58 us de bit 0 ou 100 us de bit 0 et le compteur va conduire le MAE à la prochaine etat où le second compteur va commencer et il va commencer à envoyer le même temps de bit 1.

3.3.3 Simulation

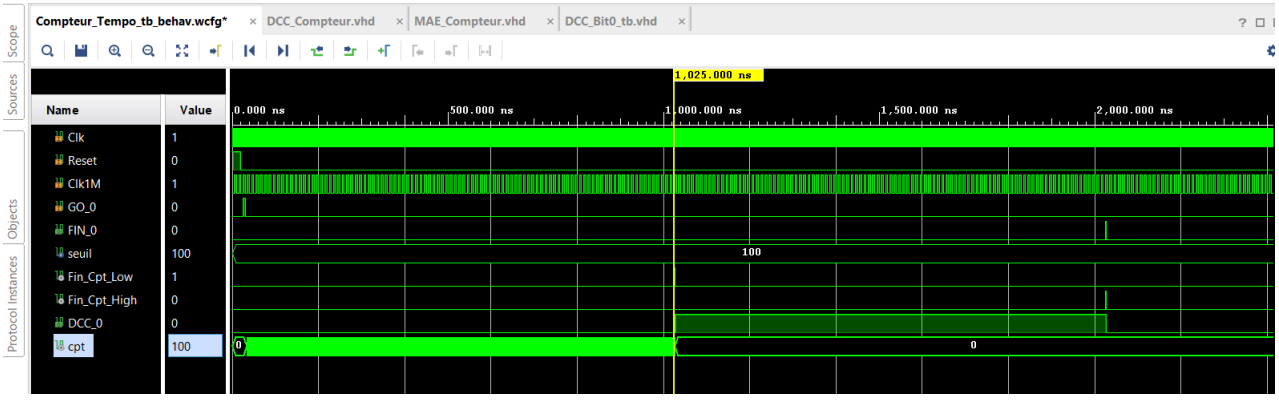

FIGURE 9 – Simulation du module DCC_0

3.4 Registre DCC

Ce module est un registre à décalage qui charge la trame DCC préparée dans Générateur Trames de Test, puis effectue des décalages au fur et à mesure que les bits de la trame sont transmis.

— La taille du registre correspond à la taille de la plus longue trame dans le protocole DCC.

— La commande du registre (chargement et décalage) est assurée par la MAE (Machine à Etats)

3.4.1 Portes

Porte d'entrée

- Clk : Horloge 100 MHz
- Reset : Reset Asynchrone
- Trame_DCC : Une trame sous le protocole DCC, sous forme de signal de 51 bits
- Com_REG : commande envoyer au registre, dans notre conception il y a 3 options ne rien faire ou déplacer un bit ou recharger la trame

Porte d'sortie

- Bit_out : le bit envoyer à la MAE

3.4.2 principe de fonctionnement

il y a un processus où nous faisons les 3 types d'opération :

- charger le trame de l'entrée jusqu'au signal interne
- incrémenter le signal qui stocke le bit déplacé
- ne rien faire

3.4.3 Simulation

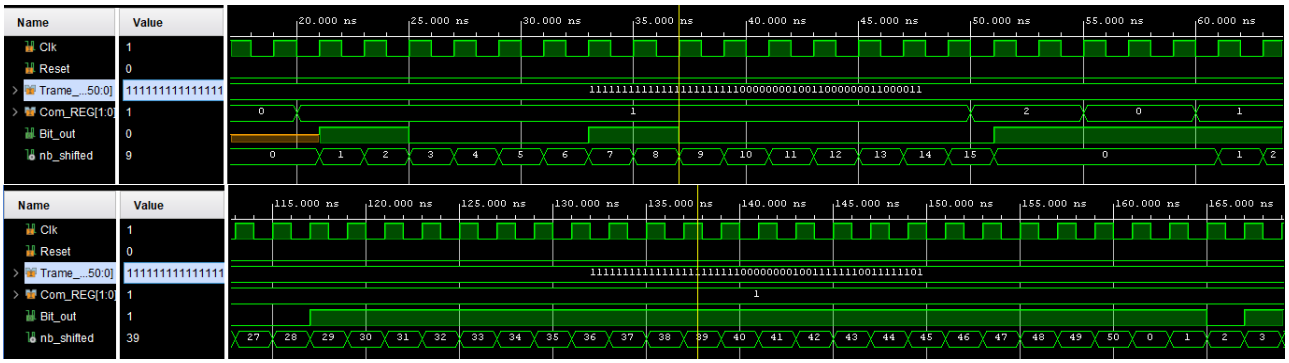

FIGURE 10 – Simulation du Registre DCC_0

Au début, le signal CMD est à 00 et le registre ne fait rien, il envoie simplement le signal précédent, ce qui fait qu'il est toujours indéfini. Lorsque le signal CMD est égal à 1, le registre commence à envoyer les bits suivants sur chaque front d'horloge. Lorsque le signal CMD est égal à 2, le registre recharge la nouvelle valeur et réinitialise le nombre de bits décalés, ce qui signifie également qu'il recommence à envoyer le premier bit de la trame. Lorsque le CMD est égal à 0, le registre ne fait rien mais la sortie reste la même qu'auparavant. 161 ns plus tard, le registre redémarre en envoyant une nouvelle trame. Comme on peut le voir, la trame se termine par 23 '1' et recommence par "10...".

3.5 MAE (Machine à Etats)

Ce module réalise la commande des autres blocs du système, comme indiqués précédemment. Il permet ainsi de générer sans interruption la trame de commande DCC préparée par l'utilisateur sur le signal Sortie_DCC.

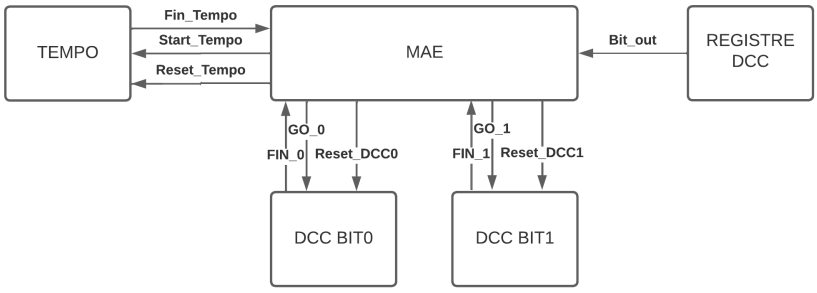

FIGURE 11 – Connection MAE

Donc, il relie donc tous les blocs entre eux, comme le montre la figure 4.

3.5.1 Le grahe de l'État

Pour le MAE, il fonctionne comme la figure ci-dessous :

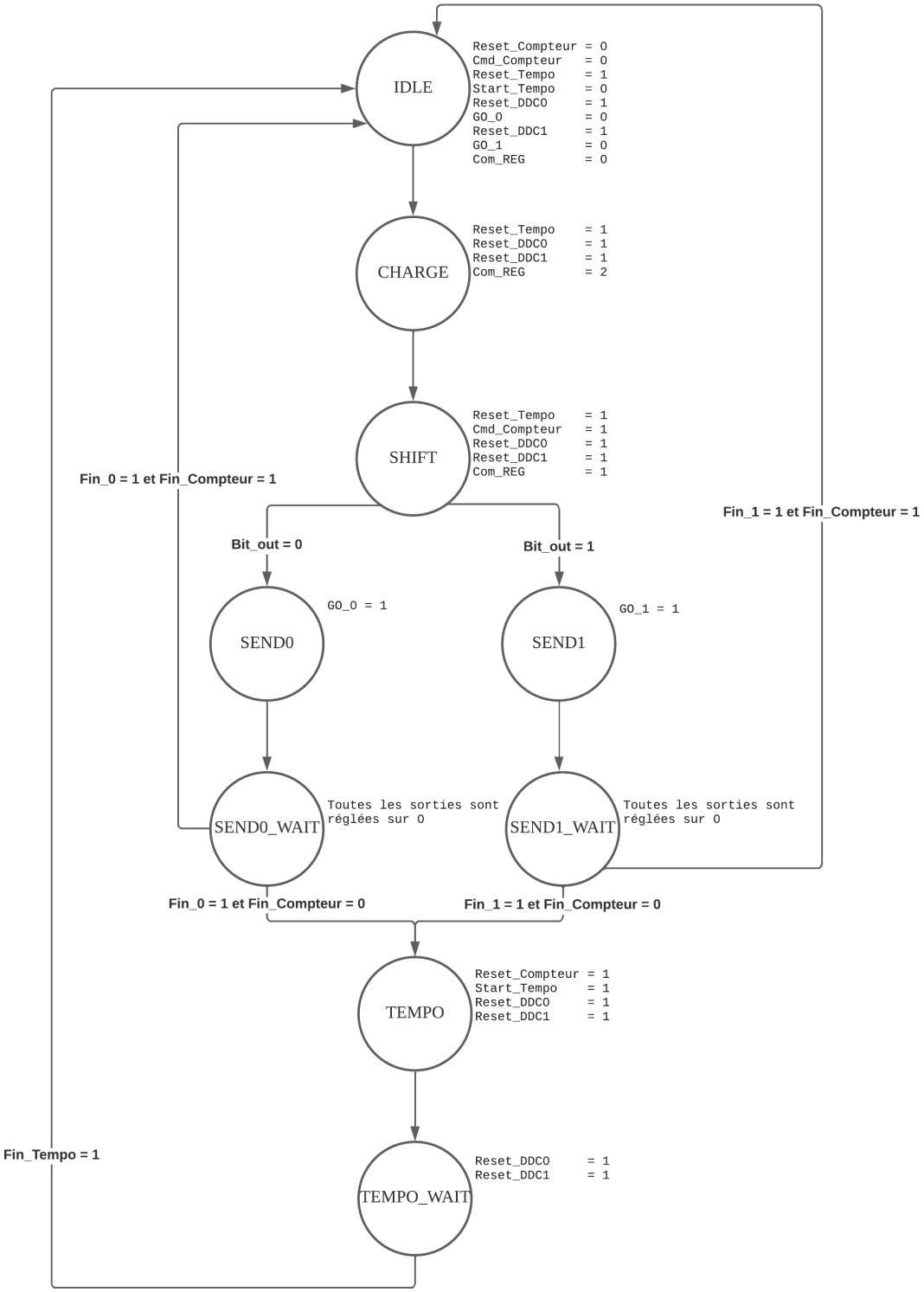

FIGURE 12 – Le graphe de l'état du MAE

3.6 Simulation du TOP

La simulation consiste donc à simuler la commande de changement de vitesse du train à l'adresse 1 et avec une vitesse en arrière-plan et un pas de 16.

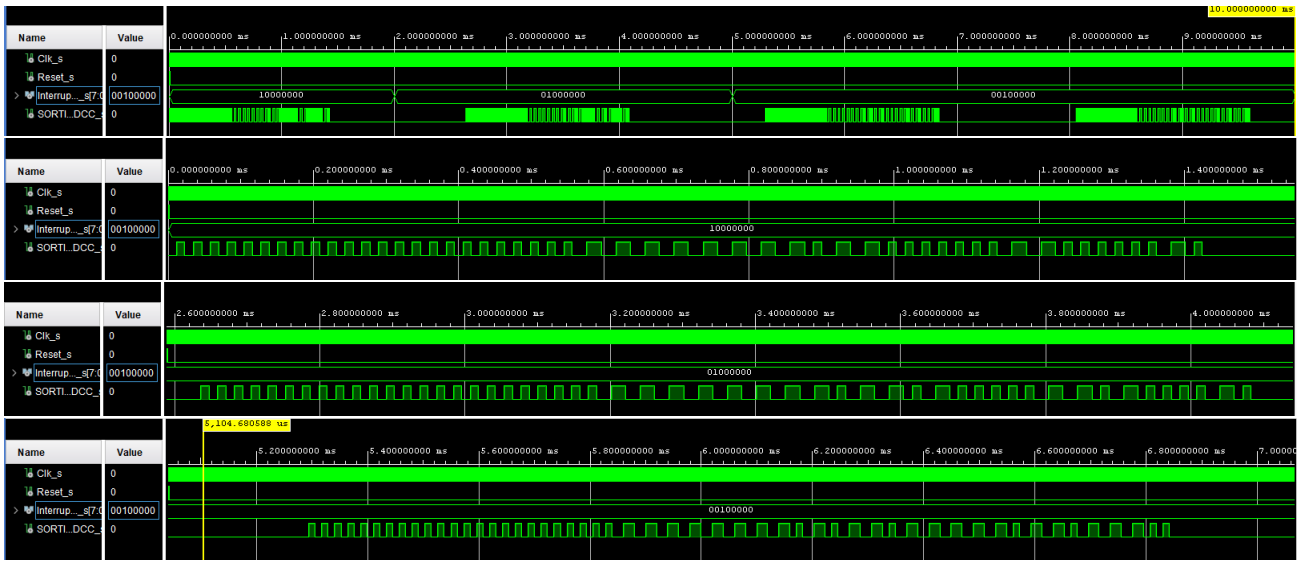

FIGURE 13 – Les simulations du Module TOP

Comme tous les modules testés fonctionnent bien, nous les avons assemblés et testés avec la simulation, puis testés avec l'oscillateur et enfin testés avec le modèle.

Voici quelques simulations que nous avons faites :

- la première fonction est d'envoyer l'ordre au train avec l'adresse 1 de se déplacer avec la vitesse maximale en avant
- la deuxième fonction est d'envoyer la commande au train avec l'adresse 1 pour qu'il se déplace à la vitesse maximale vers l'arrière.
- la troisième fonction est d'envoyer la commande d'allumer les phares du train à l'adresse 1

4 Ajout de la Centrale DCC comme IP dans un système Microblaze

Nous allons à présent intégrer la **Centrale DCC** au sein d'un système **Microblaze**. Pour cela, la centrale devra être packagée sous forme d'**IP** pour interagir avec le **Microblaze** via le bus **AXI**. L'architecture du système est présentée dans la Figure 14. La gestion des entrées/sorties de la carte **Nexys4 (boutons, interrupteurs, LED)** est à présent assurée par le **Microblaze** à l'aide de contrôleurs **GPIO**. Cela va impliquer de légères modifications sur la **Centrale DCC** (voir plus bas).

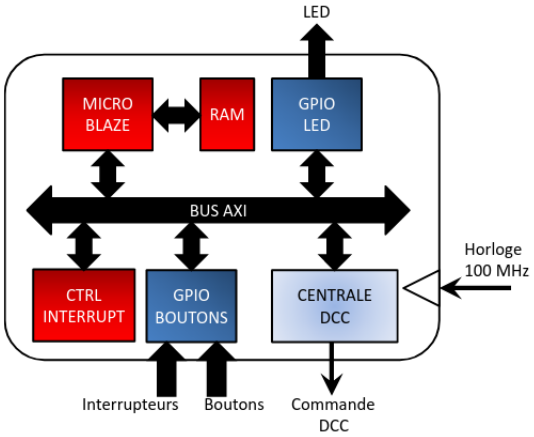

FIGURE 14 – Synoptique du système Microblaze

Le code exécuté par le **Microblaze** doit analyser l'état des **interrupteurs** et des **boutons**, afin de constituer la commande **DCC** souhaitée par l'utilisateur. Cette commande est ensuite transmise à l'**IP Centrale DCC**. Pour éviter d'envoyer des trames partiellement constituées, le **Microblaze** ne transmettra de trame à l'IP que si l'utilisateur appuie sur un bouton de validation de trames (A votre convenance, ce bouton pourra ou non être traité par interruption dans le code C). Une fois la trame reçue, l'**IP Centrale DCC** enverra le signal DCC aux trains.

La Figure 15 présente l'architecture interne de l'**IP Centrale DCC**.

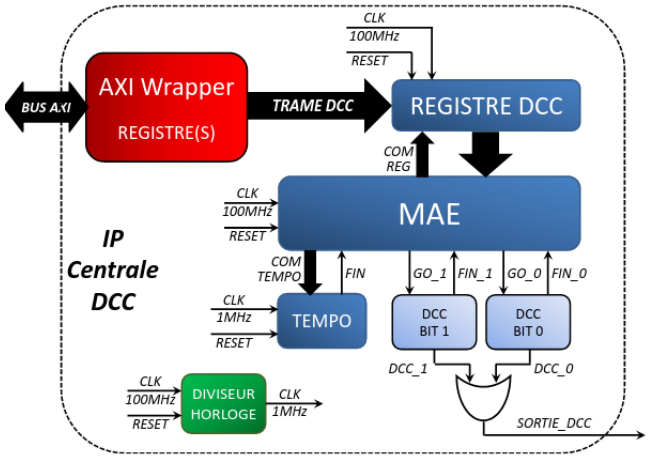

FIGURE 15 – L'IP Centrale DCC

Dans notre conception, nous avons ajouté un autre IP pour l'affichage des 7 segments. L'architecture à l'intérieur de l'IP est comme la figure suivante :

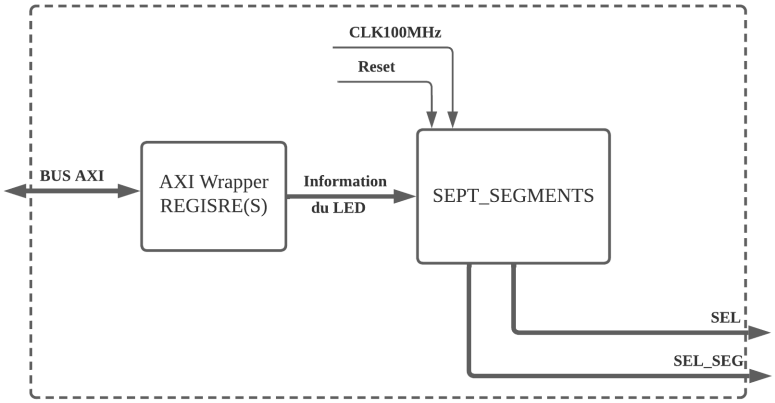

FIGURE 16 – L'IP Sept Segments

Voici donc notre conception globale :

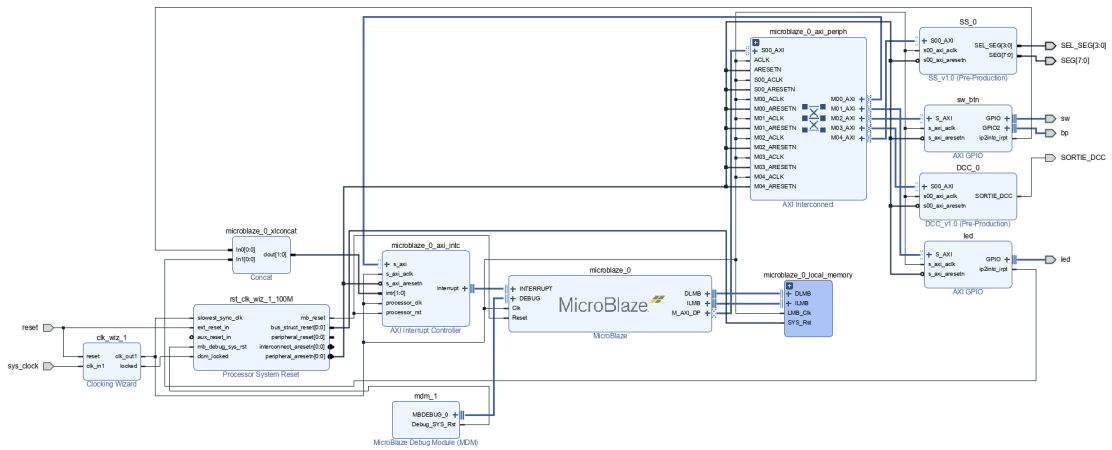

FIGURE 17 – MicroBlaze Design

4.0.1 Organisation des registres d'esclaves

Nous avons donc 4 registres de 32 bits pour le transfert de données entre le Microblaze et nos IP.

- L'IP DCC doit recevoir le trame qu'il doit générer et envoyer, ce qui fait 51 bits de données que nous utilisons tous les bits du registre esclave numéro 0 et les 19 derniers bits du registre esclave numéro 1.
- L'IP des 7 segments doit recevoir toutes les informations pour afficher et nous avons utilisé le registre esclave numéro 3. Vous trouverez plus de détails sur la façon dont nous procédons dans la partie suivante.

4.0.2 Les ports de sortie des IPs

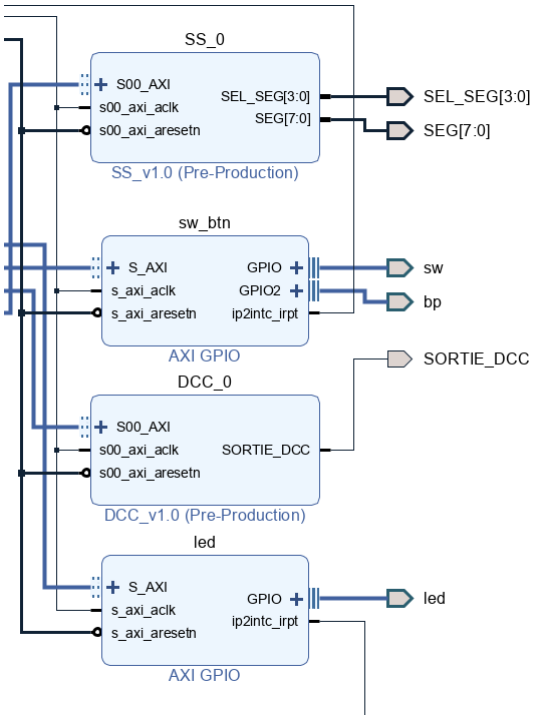

FIGURE 18 – Les Ports sortie du IPs

Tous les ports GPIO sont reliés aux ports du fichier *.xdc* comme nous l'avons fait dans les TPs précédents. le port SORTIE_DCC est connecté au port JC4 du port Pmod et le SEL_SEG et SEG sont connectés exactement selon les indications de la documentaire du carte Basys3.

Voici une partie du documentaire de la carte Basys3 :

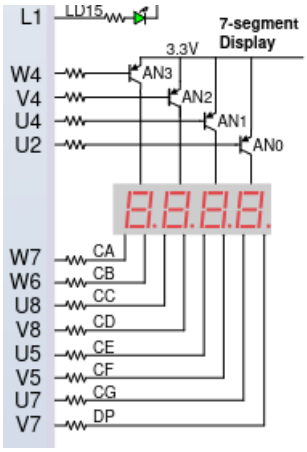

FIGURE 19 – La doc pour Basys3 7 segments

5 Conception de l'IP DCC

5.0.1 Stockage et organisation des informations de demande utilisateur avec des LED

Je vais présenter comment je stocke les informations de la demande des utilisateurs. Toutes les informations de la trame que je suis disposé à envoyer sont stockées sous forme de valeurs qui seront affichées par les LED. Ainsi, lors de la configuration de mon code ou lors du débogage, cela sera facile car je saurai comment toutes les informations ont changé en regardant les informations affichées par les LED. Au total, nous avons 16 LED réparties en 4 parties :

- La première partie est la fonction que je vais envoyer, indiquée par le bit le plus élevé de la LED, et elles sont dans l'ordre, en commençant par la fonction de changement de vitesse d'un train, puis les 21 autres fonctions. Pour cela, nous devons utiliser 5 bits.
- La deuxième partie est l'adresse du train que nous allons envoyer, qui est réinitialisée à l'adresse 1.
- La troisième partie n'est utilisée que lorsque la fonction de changement de vitesse est sélectionnée et indique si nous voulons configurer le train pour avancer ou reculer.
- La quatrième partie est utilisée soit pour le changement de mode de vitesse que nous utilisons pour configurer la vitesse du train, soit pour indiquer ON ou OFF pour la fonction. Ainsi, cette partie a également besoin de 5 bits car il y a 32 paramètres de vitesse différents.

Voilà comment j'organise fondamentalement les données qui ont été configurées en une fois. Il y a quelques petits détails, comme la réinitialisation du quatrième champ lorsque je change la fonction que je vais envoyer, et la troisième partie de la formation des LED est à 0 lorsque nous ne sommes pas prêts à envoyer un changement de vitesse, etc.

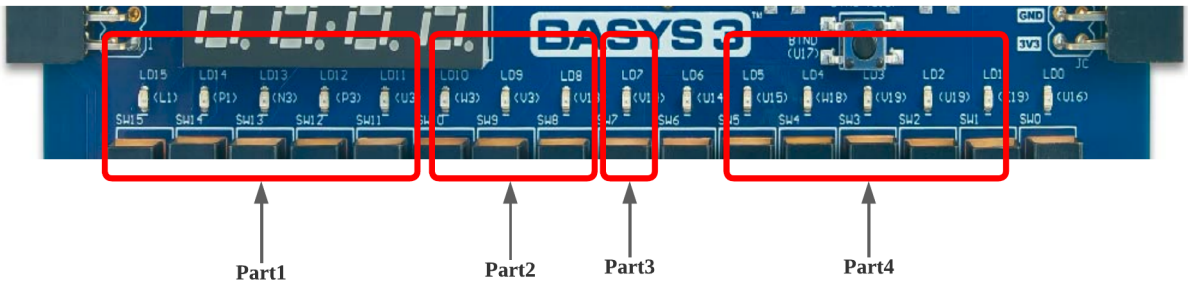

FIGURE 20 – Partitionnement et affichage des données sur des LED

5.1 Conception du système de configuration de données avec boutons de contrôle

Pour faciliter la configuration des données stockées dans les 4 parties, notre système propose une interface conviviale. Les utilisateurs peuvent utiliser les boutons gauche et droite pour naviguer entre les différentes parties, tandis que les boutons haut et bas permettent d'ajuster les paramètres de chaque partie. Lorsque les réglages sont terminés, il suffit d'appuyer sur le bouton central pour générer une nouvelle trame de données, qui sera ensuite envoyée au registre esclave.

Pour garantir une utilisation optimale, certaines limitations ont été mises en place. Par exemple, lorsque la valeur atteint déjà le maximum ou le minimum de chaque partie, il n'est pas possible de déplacer le curseur vers le haut ou vers le bas. De même, si nous sommes déjà sur la dernière partie, il n'est pas possible de passer à la partie suivante.

Grâce à cette interface intuitive et aux fonctionnalités de contrôle, les utilisateurs peuvent configurer facilement et rapidement les données à envoyer, tout en s'assurant que les changements sont clairement reflétés sur les LED. Voici un schéma de base illustrant le fonctionnement de notre conception :

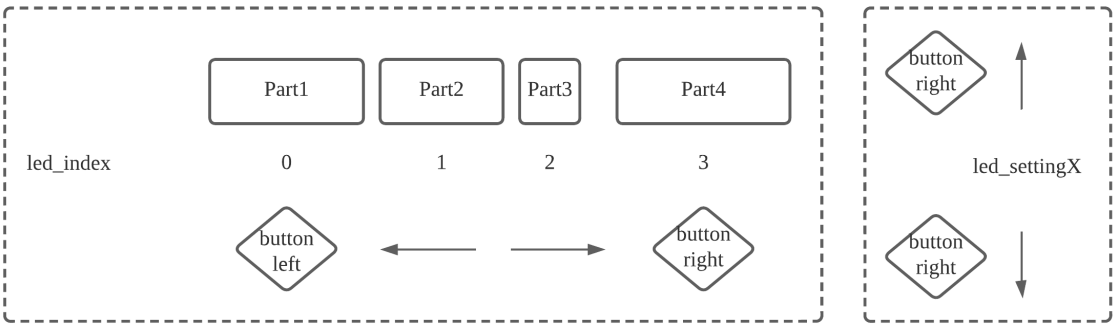

FIGURE 21 – Conception du contrôle

6 Conception de l'IP Afficheur 7 segments

En plus des quatre parties d'information, le module IP à afficheur 7 segments reçoit également un bit supplémentaire qui agit comme un classifieur. Ce classifieur permet de distinguer si la fonction et l'index de la partie sont liés au changement de vitesse ou non.

Le module IP à afficheur 7 segments fonctionne comme une sorte de structure de cas sélectifs complète, prenant en compte les valeurs de toutes les entrées, y compris la fonction, l'index de la partie et le bit classifieur. En fonction de la combinaison spécifique de ces entrées, le module génère des signaux pour afficher les caractères correspondants sur l'afficheur 7 segments.

L'afficheur 7 segments lui-même fonctionne en affichant séquentiellement un caractère après l'autre à une fréquence très élevée. Cette rotation rapide crée l'illusion que les quatre caractères sont visibles simultanément. En mettant constamment à jour les caractères affichés en fonction des valeurs changeantes des parties d'entrée, l'afficheur 7 segments transmet de manière efficace une représentation dynamique des informations transmises.

En utilisant le module IP à afficheur 7 segments avec le bit classifieur, le système peut communiquer clairement les détails de configuration aux utilisateurs. Cela garantit une interprétation et une compréhension faciles des informations affichées, améliorant ainsi l'expérience globale de l'utilisateur.

Voici quelques photos à titre d'exemple

FIGURE 22 – Exeple du 7 segments

7 Simulation

Donc, une fois que chaque partie fonctionne correctement individuellement, nous les assemblons et les testons à l'aide de l'oscilloscope. Voici quelques-uns des tests que nous avons effectués pour vérifier leur fonctionnement :

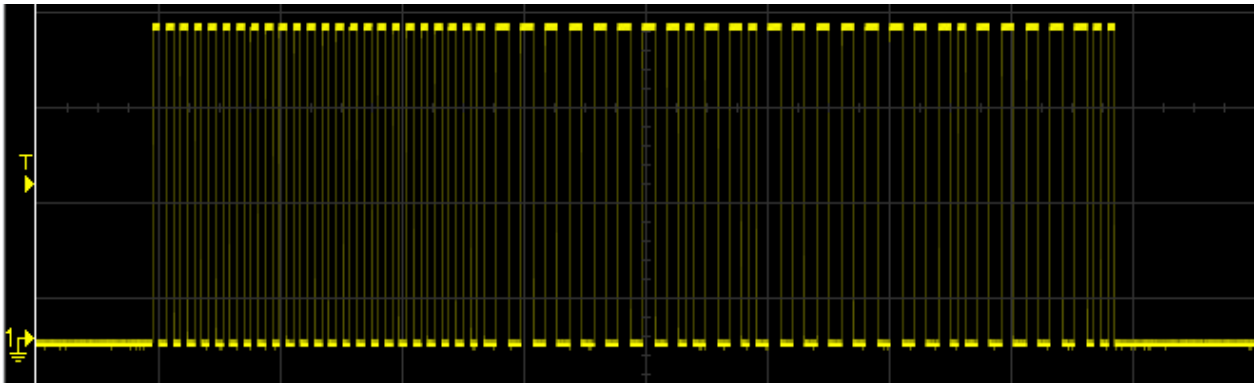

FIGURE 23 – Trame Speed0 Dir0 Adr1

Adresse	Command	XOR
00000001	01000000	01000001

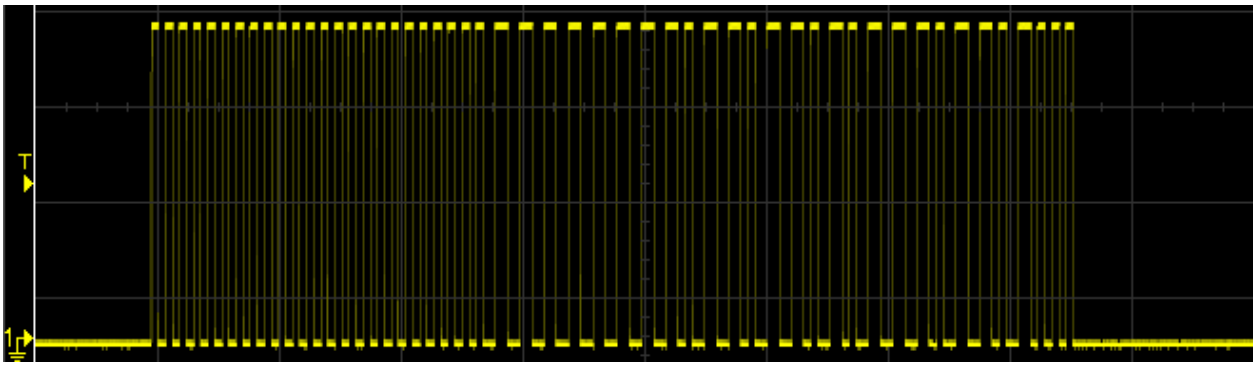

FIGURE 24 – Trame Speed10 Dir0 Adr1

Adresse	Command	XOR
00000001	01001010	01001011

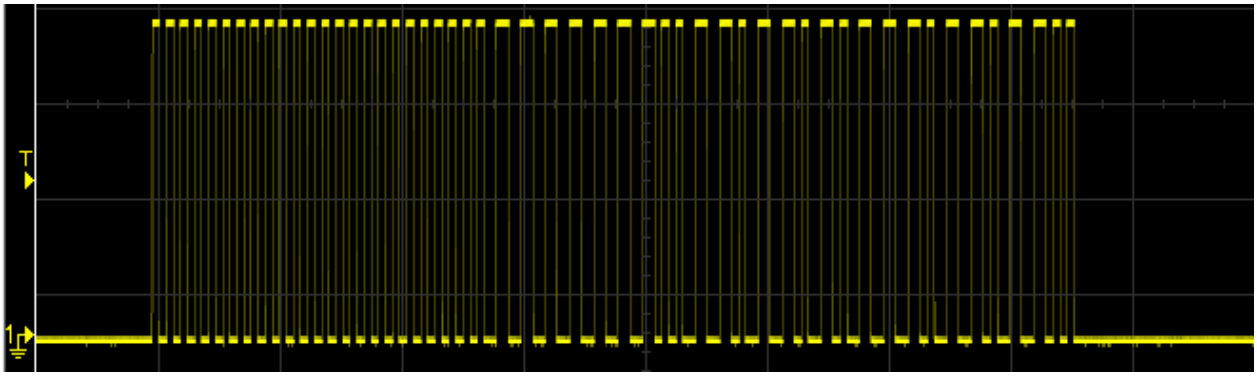

FIGURE 25 – Trame Speed10 Dir0 Adr3

Adresse	Command	XOR
00000011	01001010	01001001

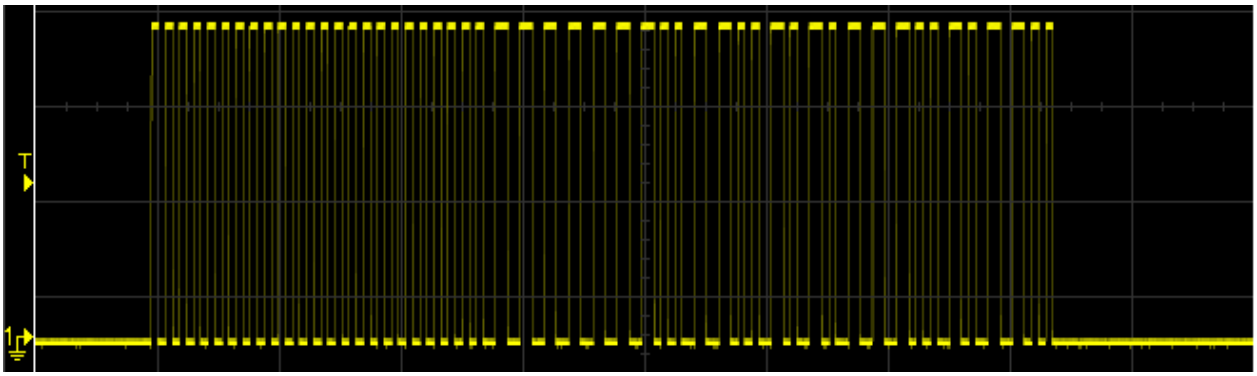

FIGURE 26 – Trame Speed10 Dir1 Adr3

Adresse	Command	XOR
00000011	01101010	01101001

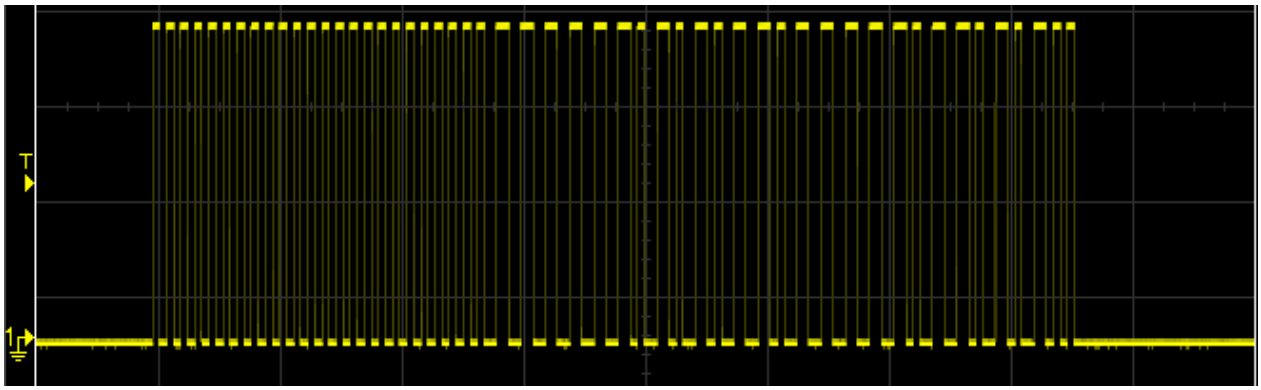

FIGURE 27 – Trame Fonct0 Adr5 ON

Adresse	Command	XOR
00000101	10010000	10010101

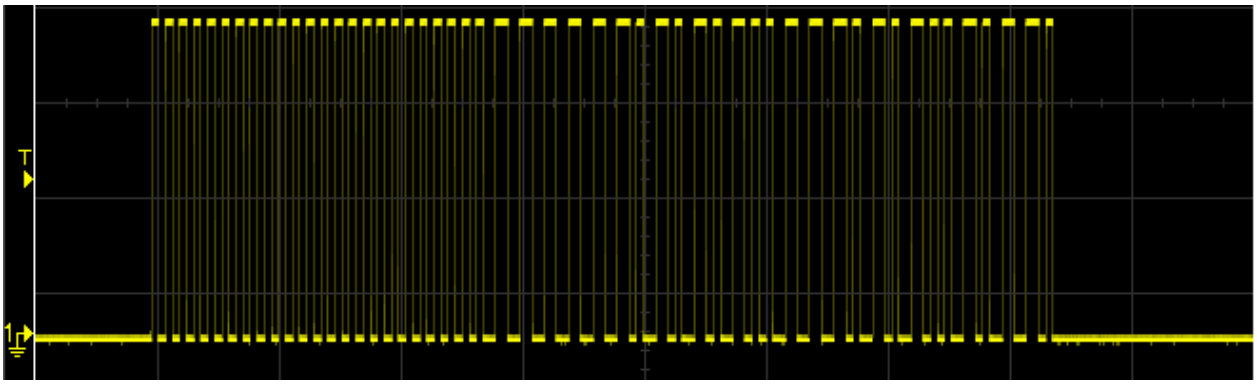

FIGURE 28 – Trame Fonct5 Adr5 ON

Adresse	Command	XOR
00000101	10110001	10110100

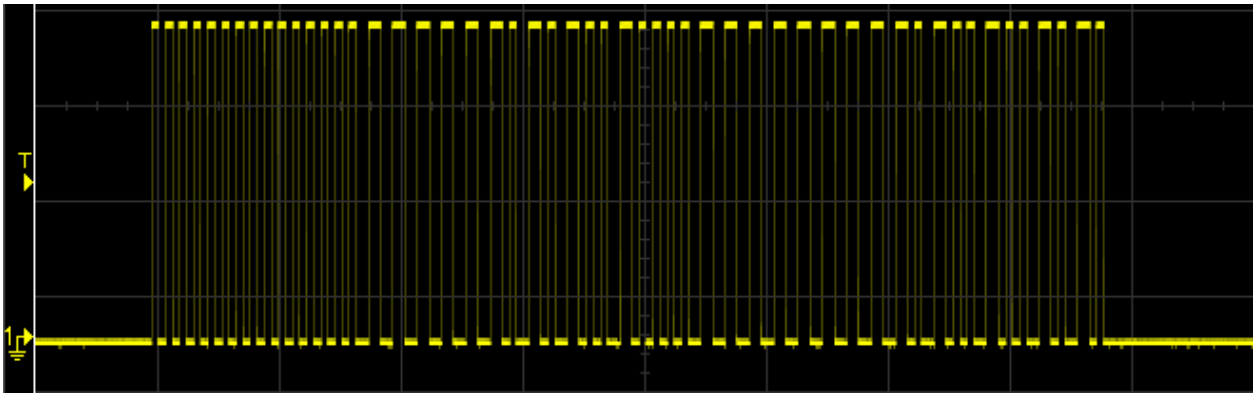

FIGURE 29 – Trame Fonct13 Adr5 ON

Adresse	Command0	Command1	XOR
00000101	11011110	00000001	11011010