SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Master Informatique

M1 Parcours SESI

UE MU4IN108 - J.Denoulet

TP – FPGA1

SYSTEMES PROGRAMMABLES

4EME PARTIE – CENTRALE DCC SUR FPGA

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.

Les commandes vers les trains sont générées par une ***Centrale DCC*** que nous allons implémenter dans le ***FPGA*** de la carte ***Nexys***, sous la forme d'un système mixte matériel/logiciel

Grâce à une interface utilisateur réalisée à l'aide des boutons de la carte ***Nexys***, le ***FPGA*** doit générer un signal numérique de commande. Cette commande doit ensuite être amplifiée en courant à l'aide d'une carte Booster, afin d'obtenir un signal suffisamment puissant pour être envoyé sur les rails puis être décodé par les locomotives.

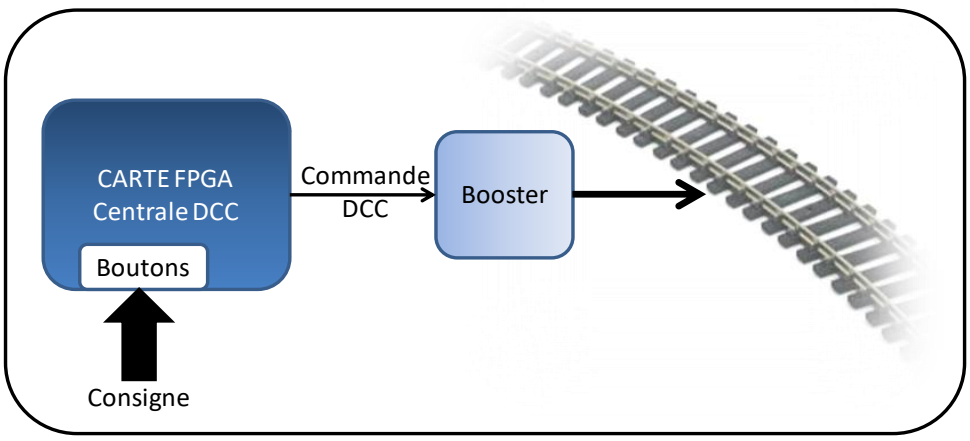


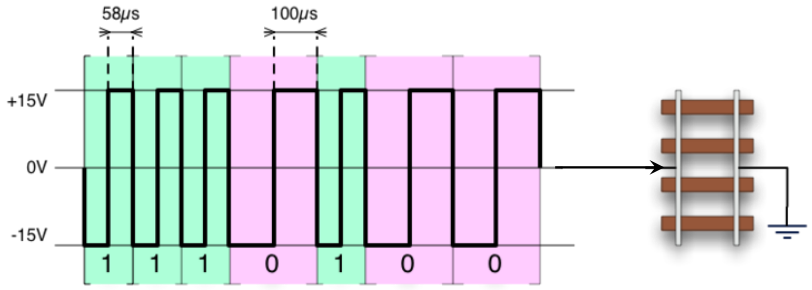
Figure 1 – Synoptique du système de commande

I) Protocole DCC

Les commandes sont transmises sous la forme d'une trame de données numériques.

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



La commande d'un train s'opère en envoyant sur les rails une trame de configuration. L'alimentation des trains étant fournie par le courant circulant sur les rails, il est nécessaire de générer des trames en permanence, espacées par un intervalle de 6 ms.

Le protocole ***DCC*** prévoit des trames IDLE que l'on peut utiliser lorsque l'on ne souhaite pas modifier les configurations des trains, mais pour simplifier notre système, on émettra en permanence la dernière trame de configuration qui aura été validée.

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.*

La Figure 2 donne l'allure d'une trame comprenant un champ de commande sur 1 octet.

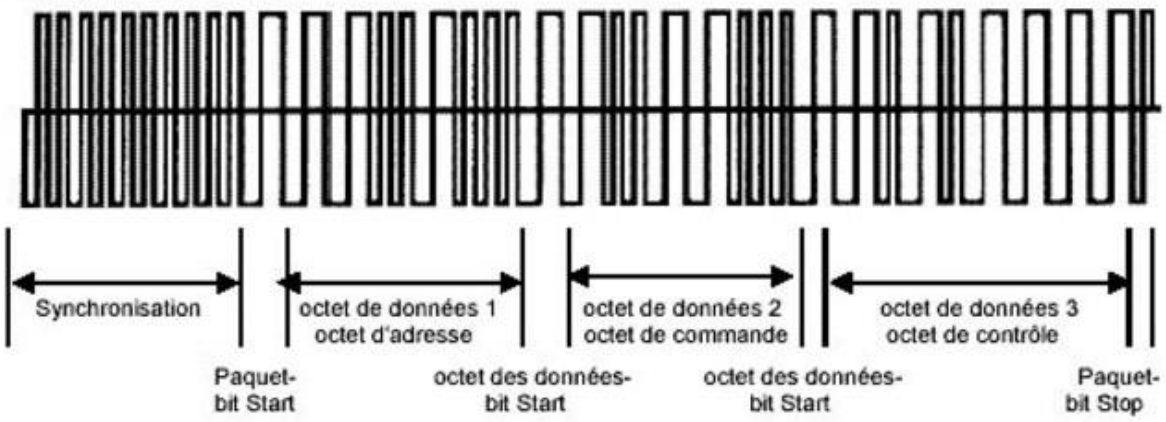


Figure 2 – Exemple d'une trame DCC

Chaque train dispose d'une adresse ***DCC*** unique qu'il faut donc indiquer dans le champ d'adresse de la trame de configuration.

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)

- 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	Step 5	01000	Step 13	01100	Step 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

- 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
F0	Lumière on / off
F1	Son on / off
F2	Cor Française# 1
F3	Cor Française# 2
F4	Turbo off
F5	Compresseur
F6	Accélération / freinage temps, bouchant mode / vitesse de manœuvre
F7	Courbe grincement
F8	Ferroviaire Clank
F9	Ventilateur
F10	Conducteur de signal
F11	Court Cor Française # 1

KEY	FUNCTION
F12	Court Cor Française # 2
F13	L'Annonce station française #1
F14	L'Annonce station française #2
F15	Signal d'alerte française #1
F16	Signal d'alerte française #2
F17	Porte chauffeur ouvrir / fermer
F18	Valve
F19	Attelage
F20	Sable
F21	Libération des freins

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
  - **100XXXXX**
  - Le bit 4 sert à gérer la fonction F0 (Bit à 1 : Phares allumés, Bit à 0 : Phares éteints)
  - 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
  - **101SXXXX**
  - Le bit S sert à sélectionner le groupe de fonctions F5-F8 (Bit à 1) ou F9-F12 (Bit à 0).
  - Si S=1, les bits 3-0 gèrent les fonctions F8 à F5, dans cet ordre (Bit à 1 : ON, Bit à 0 : OFF)
  - 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
  - **110 11110 XXXXXXXX**
  - Les 8 bits du 2ème octet gèrent chacun une fonction (Bit 7 pour F20, Bit 0 pour F13)
  - 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)

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

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.
  - En cas de dépassement, le décodage des trames par les locomotives n'est plus garanti !

II) Architecture de la centrale DCC

L'architecture de la ***Centrale DCC*** est présentée en Figure 3.

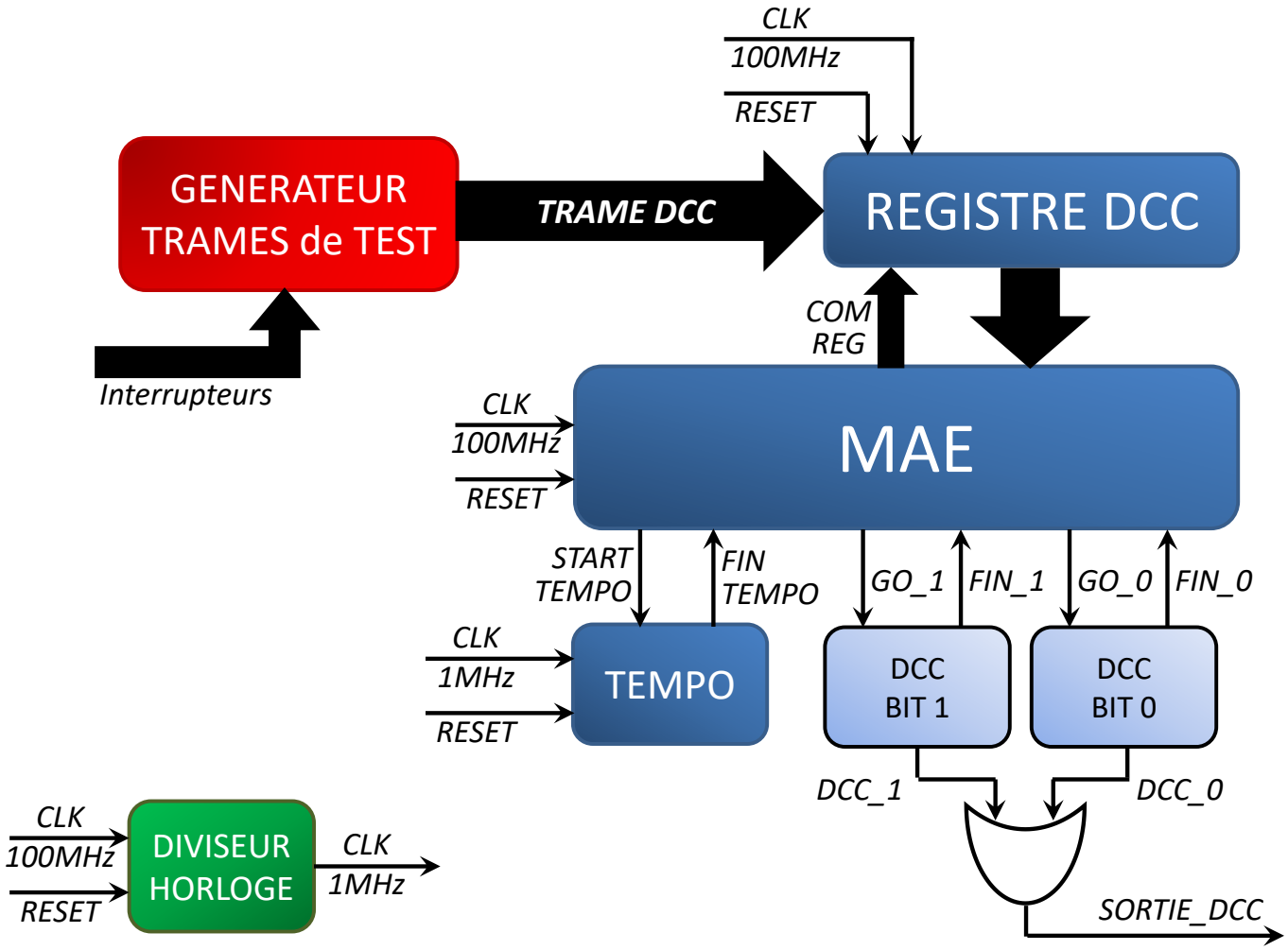


Figure 3 – Architecture de la Centrale DCC

- **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***

- **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..
    - Ce signal est automatiquement remis à 0 dès que la commande ***Start_Tempo*** est remise à 0.

- **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)
        - 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.
        - La commande ***Go*** active la génération du bit.
        - Le signal ***Fin*** indique que la transmission du bit est terminée.
    - Chaque bloc possède une machine à états et un compteur
        - 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.
        - Le compteur est cadencé par l'horloge ***CLK_1MHz***.
            - 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.

- **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)

- **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***

- **Générateur Trames de Test**
    - Ce module sert uniquement à valider le reste de l'architecture, en générant des trames de test qui vont permettre de vérifier le bon fonctionnement du système
    - Les trames de test sont mémorisées dans le module et sont positionnées en sortie en fonction de la configuration des interrupteurs.
    - Ce module sera retiré du système dans la partie suivante (et sera remplacé par le Microblaze).

- Une partie de ces modules a déjà été codée. Il s'agit du ***Diviseur d'Horloge*** et du ***Compteur de Temporisation***. Le ***Générateur de Trames*** est lui à compléter à partir d'une squelette de base.

- Les autres modules sont à développer en intégralité.

- ***Récupérer sur Moodle les codes VHDL des modules déjà écrits. Décrire les modules restants de la Centrale DCC en VHDL.***

- ***Pour chaque module procéder ainsi :***
  - ***Ecrire le code VHDL***
  - ***Faire une simulation comportementale à l'aide d'un testbench. Corriger les éventuelles erreurs.***
  - ***Faire une synthèse afin de vérifier que la description est bien implémentable (regarder les warnings et les erreurs éventuellement indiqués par Vivado)***

- ***Concernant le module de Génération des Trames de Test***
  - ***Compléter le code VHDL en spécifiant les trames que vous voulez envoyer pour tester votre architecture.***

- ***Assembler tous ces modules dans un module Top_DCC***
  - ***Vérifier par simulation le bon fonctionnement du système***
  - ***Implémenter le système et vérifier à l'aide d'un oscilloscope que la carte génère bien une trame DCC en sortie.***
  - ***Valider le fonctionnement du système sur la plate-forme trains.***
  - ***La sortie DCC sera reliée à la broche 4 du connecteur PMOD A (broche JA[4] sur les fixhiers XDC)***

- ***NB : Dans l'évaluation du TP, une attention particulière sera portée à la qualité de vos testbenchs***
  - ***N'hésitez pas à vous servir des notions vues en cours (comme les ASSERT) pour vous aider à contrôler le bon fonctionnement de vos modèles VHDL***

III) 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 4. 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).

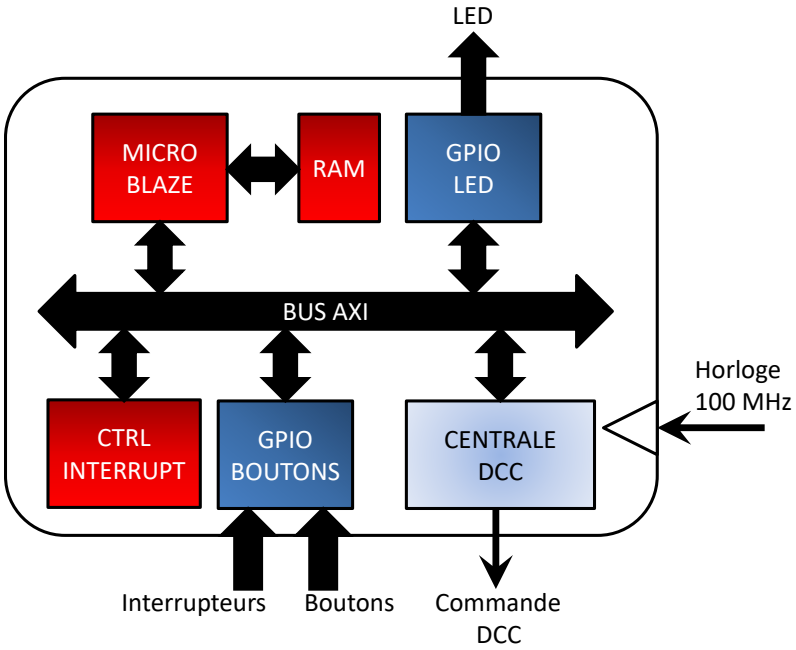


Figure 4 – Synoptique du système Microblaze

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 5 présente l'architecture interne de l'***IP Centrale DCC***.

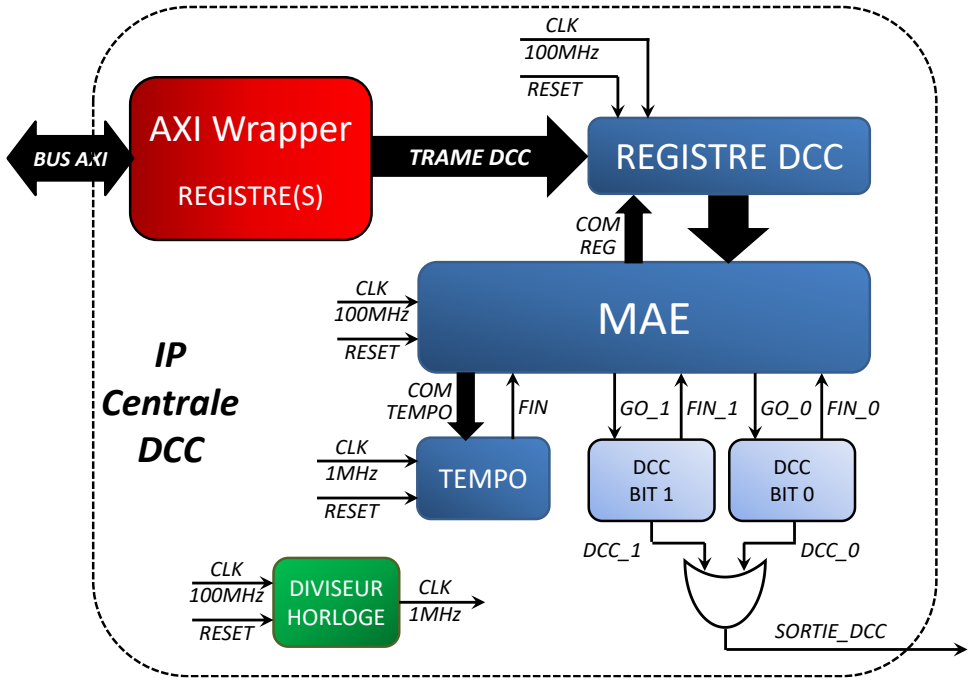


Figure 5 –l'IP Centrale DCC

- Le seul changement par rapport à l'architecture précédente de la ***Centrale DCC*** est que le module de ***Génération Trames de Test*** est remplacé par un ***AXI Wrapper*** qui permet de rattacher la ***Centrale DCC*** au bus AXI du ***Microblaze***. On rappelle que le template de ce ***Wrapper*** est généré automatiquement par ***Vivado*** (cf. TP 3ème partie) et contient les registres de configuration (***slv_reg***) de l'***IP***.

- Dans le ***AXI Wrapper***, il vous faut définir combien de registres vous sont nécessaires et quel va être leur rôle. Pour cela, deux approches sont possibles :
    - Le code C du ***Microblaze*** va préparer la trame exacte à envoyer au bit près (avec toutes les caractéristiques du protocole DCC comme les bits de Start, l'octet de contrôle, etc…) et transférer cette trame dans un ou plusieurs registres d'interface de l'***IP***. Le contenu de ces registres est ensuite redirigé vers le cœur de la ***Centrale DCC***.
    - Le code C du ***Microblaze*** se contente de fixer les paramètres généraux de la trame à envoyer (adresse du train, type de commande) et envoie ces paramètres à l'***IP*** dans un ou plusieurs registres d'interface. Il faut alors ajouter dans l'***AXI Wrapper*** de l'***IP*** une architecture qui décode les paramètres reçus pour générer la trame DCC correspondante puis la transférer au cœur de l'***IP***.

- Dans tous les cas, on rappelle qu'il ne faudra transmettre au cœur de l'***IP*** que des commandes validées par l'utilisateur.

- ***Créer dans Vivado une IP Centrale DCC pour le Microblaze (cf. TP3).***
    - ***L'horloge 100 MHz de votre Centrale sera connectée au signal S_AXI_CLK du bus AXI***
    - ***Le Reset Asynchrone de votre Centrale sera connectée au signal S_AXI_ARESETN du bus AXI***
        - ***Attention le signal S_AXI_ARESETN est actif au niveau bas.***

- ***Créer dans Vivado un block design (cf. TP2) pour réaliser le système de la Figure 5. Implémenter le design et générer le bitstream.***

- ***Dans Vitis, créer un projet puis développer le code C permettant de piloter les trains via le Microblaze et l'IP Centrale DCC.***
    - ***Vous pourrez utiliser pour cela le driver de l'IP que Vivado aura généré.***

- ***Valider votre système sur la plate-forme trains.***