SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Master Informatique

M1 Spécialité SESI

UE 4I108 - J.Denoulet

TP – FPGA1

SYSTEMES PROGRAMMABLES

5EME 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 ***Nexys4***, sous la forme d'un système mixte matériel/logiciel

A l'aide d'une interface utilisateur réalisée à l'aide des boutons de la carte ***Nexys4***, 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.

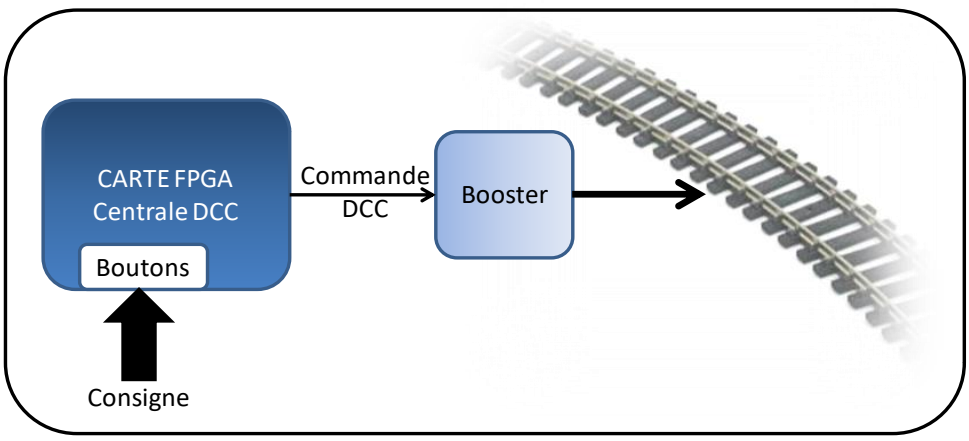


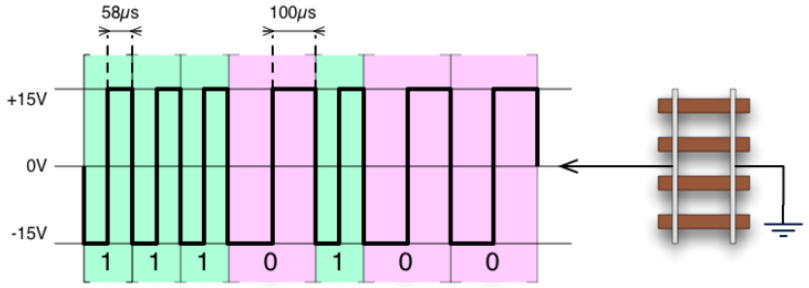
Figure 1 – Synoptique du système de commande

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.

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 3 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.

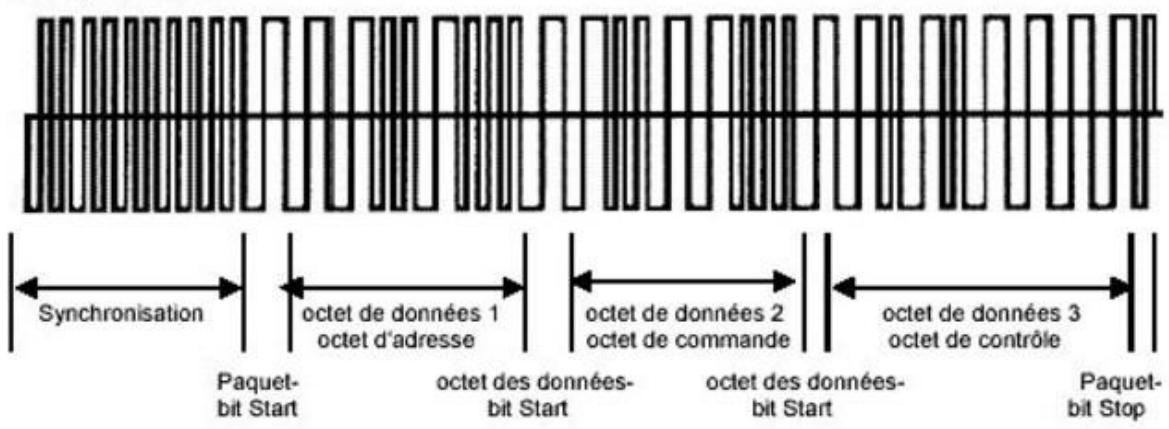


Figure 2 – Exemple d'une trame DCC

Chaque train disposera d'une adresse ***DCC*** unique qu'il faudra 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 9 : 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
- 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.

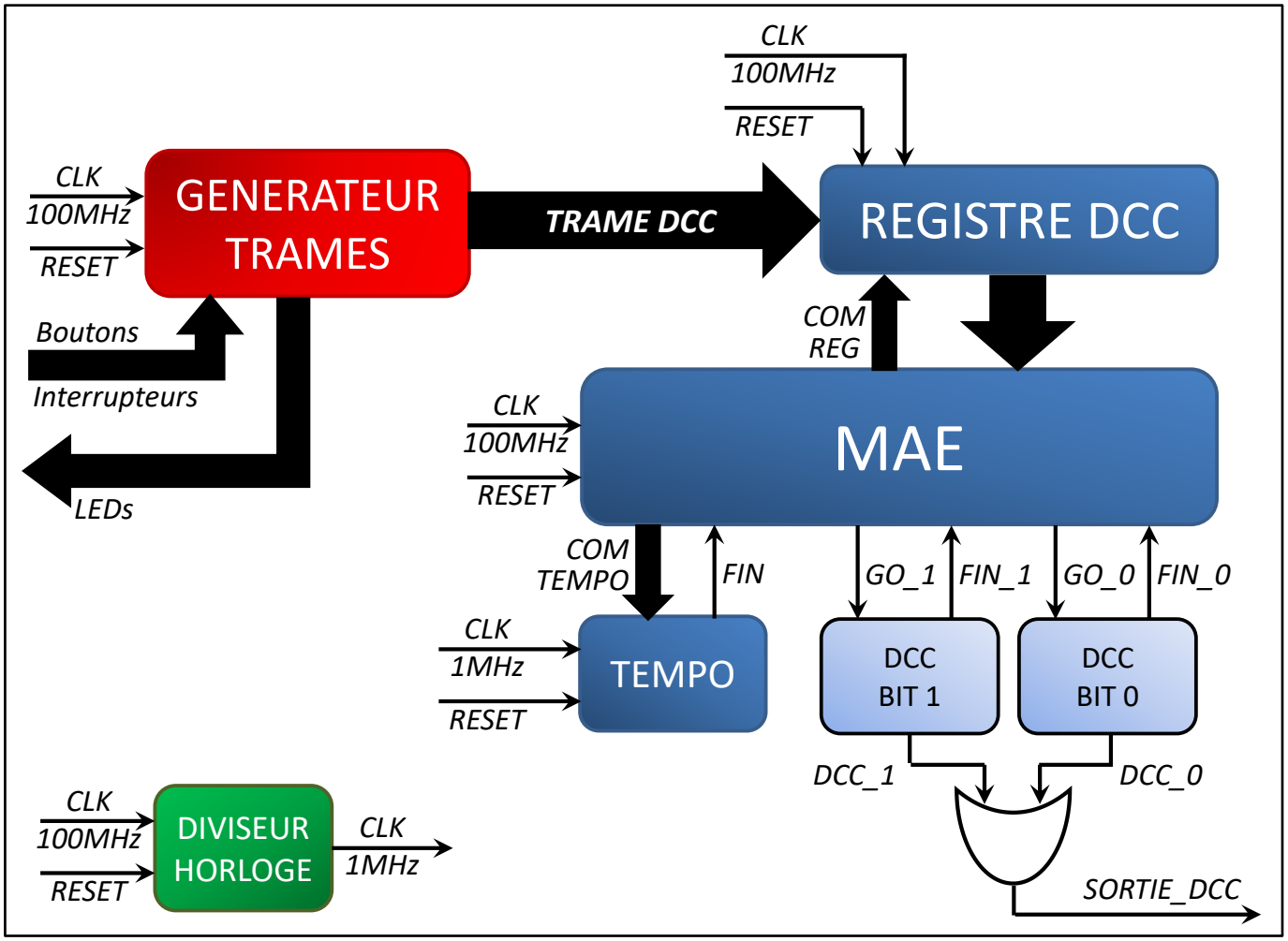


Figure 3 – Architecture de la Centrale DCC

- **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 de ce module est assurée par la ***MAE*** (Machine à Etats)

- **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***.
    - 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***
      - 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 doit charger la trame ***DCC*** préparée dans le module précédent, puis effectuer des décalages au fur et à mesure que les bits de la trame sont transmis.
  - 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**
  - Ce module permet à l'utilisateur de constituer la commande ***DCC*** pour piloter le train de son choix.
  - Son architecture est présentée sur la Figure 4 ci-dessous.
    - Cette organisation permettra de passer plus facilement d'une version purement matérielle de la ***Centrale DCC*** à une version mixte HW/SW avec le ***Microblaze*** (voir partie III, page 8)
    - Le module est composé de 3 sous-blocs.
      1) Une ***Interface Utilisateur*** pour faire le lien entre les actions des boutons poussoirs et interrupteurs et les commandes ***DCC*** que l'on souhaite mettre en œuvre
      2) Des ***Registres de Configuration*** qui mémorisent les paramètres souhaités pour la trame ***DCC***.
         - Dans la version ***IP Microblaze*** de la ***Centrale DCC***, ces registres correspondront aux ***slv_reg*** (cf. le sujet de la 3ème partie des TP)
      3) Un ***Constructeur de Trames*** qui reconstitue l'intégralité de la trame ***DCC*** à partir des paramètres des ***Registres de Configuration***.

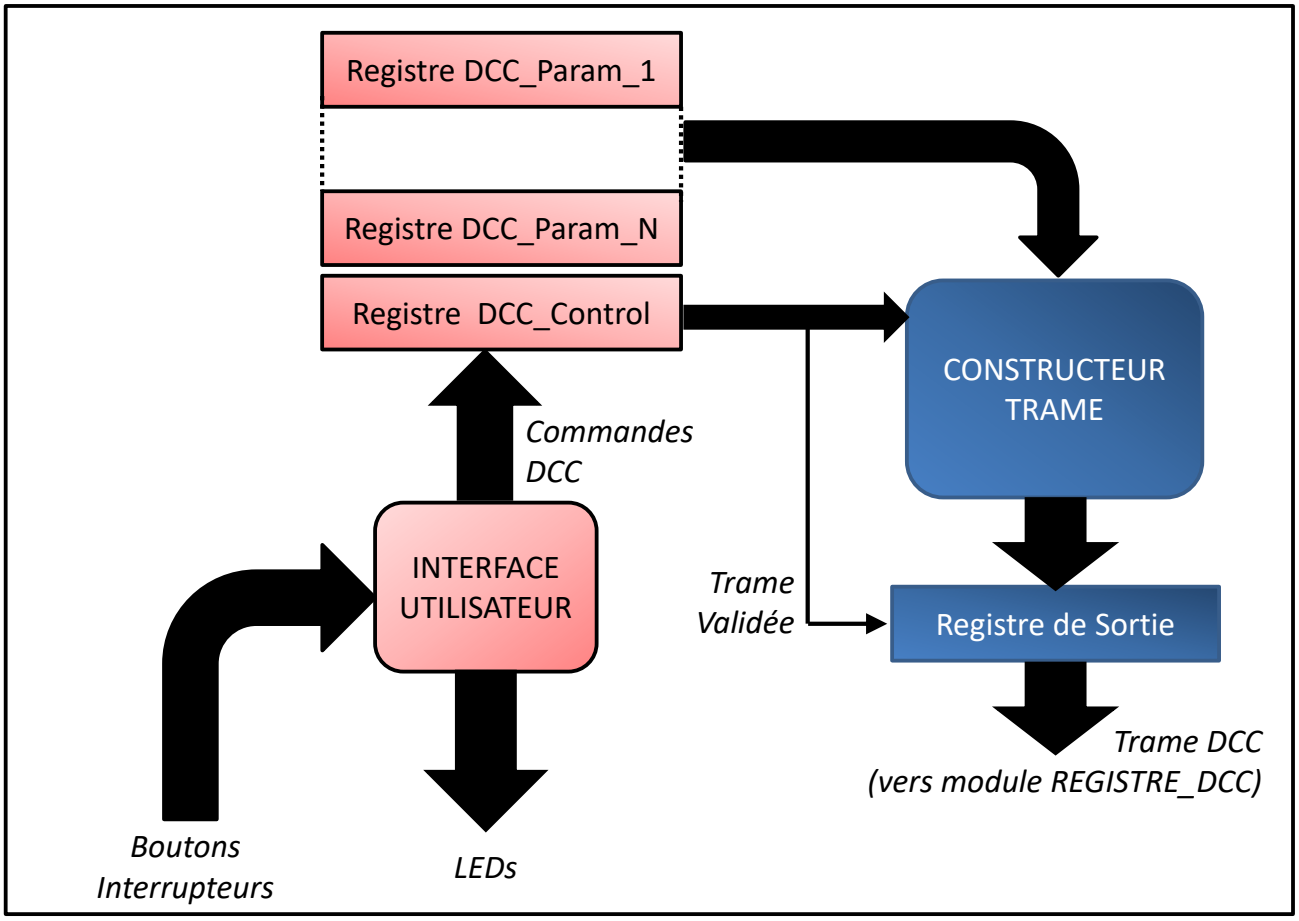


Figure 4 – Architecture du module de génération de trames DCC

- L'***Interface Utilisateur*** permet donc de constituer la trame ***DCC*** grâce aux ***boutons*** et ***interrupteurs*** de la carte. Les ***LEDs*** peuvent aussi être utilisées pour aider à visualiser la trame préparée.
  - Vous pouvez choisir d'implémenter seulement un sous-ensemble des vitesses et des fonctions des locomotives, ou bien de toutes les implémenter (sachant que c'est un peu plus compliqué…)
  - Il faudra réserver un ***bouton poussoir*** pour valider la trame préparée. Cela aura pour effet de générer en sortie la trame ***DCC*** correspondant à votre sélection.

- Les ***Registres de Configuration*** ont une taille de 32 bits (comme les registres des IP du ***Microblaze***).
  - On trouvera un ou plusieurs registres de type ***DCC_Param*** qui stockeront les paramètres de la trame ***DCC*** en train d'être constituée. Par l'utilisateur
    - Le nombre de registres et la répartition de leurs 32 bits dépendent des commandes ***DCC*** que vous voulez implémenter.
  - On trouvera aussi un registre 32 bits, ***DCC_Control***, qui permettra notamment de savoir si la trame est validée, c'est-à-dire si elle peut être envoyée en sortie.

- Le module de ***Construction de Trame*** est un décodeur qui interprète la valeur des ***Registres de Configuration*** pour générer les différents champs de la trame DCC à envoyer.
  - On trouvera en sortie du décodeur un registre qui se rechargera à chaque fois qu'une trame sera validée par l'utilisateur.

- ***Décrire la Centrale DCC en VHDL. (Commencez plutôt par les blocs Diviseur, Tempo et DCC_Bit)***

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

- ***Assembler tous ces modules dans un module Top_DCC***
  - ***Vérifier par simulation le bon fonctionnement du système***
  - ***Implémenter le système et valider son fonctionnement 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)***

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 5.

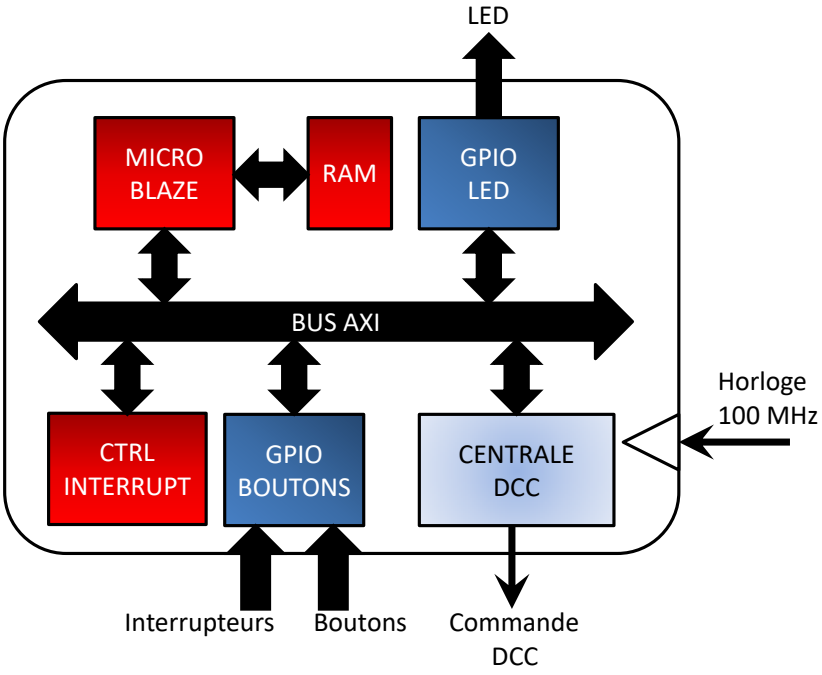


Figure 5 – Synoptique du système Microblaze

La gestion des entrées/sorties de la carte ***Nexys4*** (***boutons***, ***interrupteurs***, ***LED***) sont à présent gérées par le ***Microblaze*** à l'aide de contrôleurs ***GPIO***, ce qui impliquera de légères modifications sur la ***Centrale DCC*** (voir plus bas).

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. Lorsque la commande est validée par ce dernier, le processeur doit transmettre une commande à l'***IP Centrale DCC*** pour qu'elle puisse générer la trame ***DCC*** et ainsi configurer le train, conformément au souhait de l'utilisateur.

La Figure 6 présente l'architecture interne de l'***IP Centrale DCC***.

- L'***IP*** dispose d'un ***Wrapper AXI*** pour se connecter au bus. On rappelle que ce ***Wrapper*** est généré automatiquement par ***Vivado*** (cf. TP 3ème partie) et contient les registres de configuration (***slv_reg***) de l'***IP***.

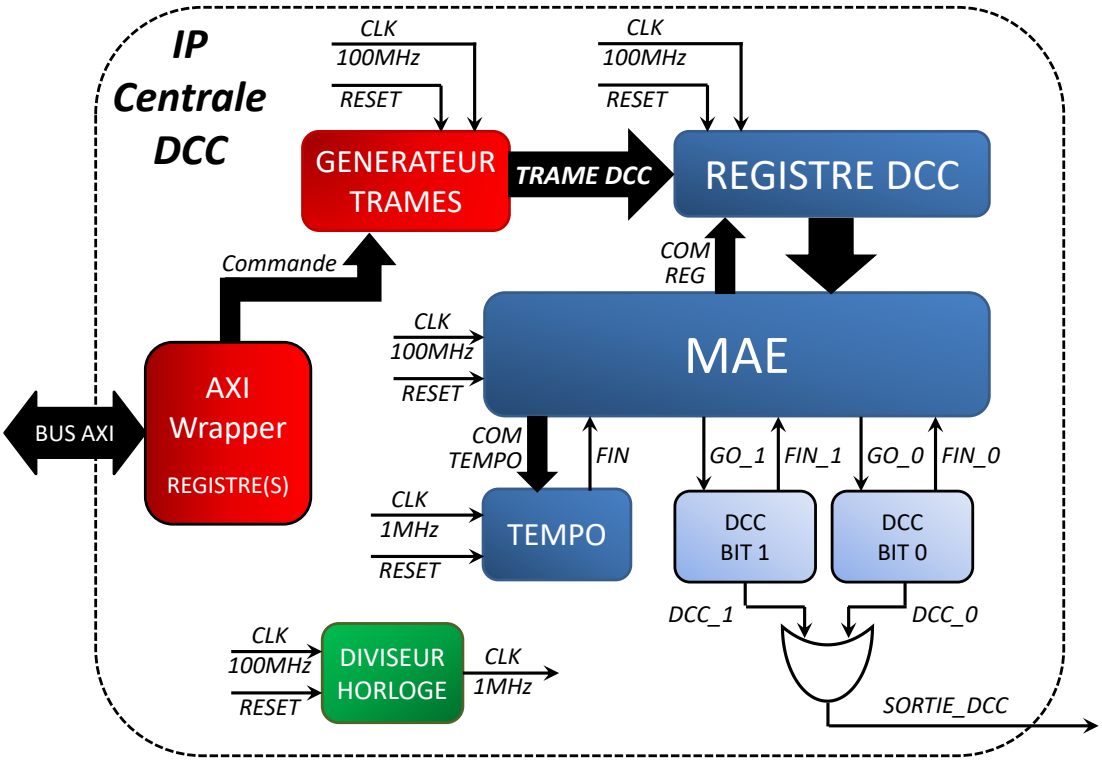


Figure 6 –l'IP Centrale DCC

- Par rapport à la version précédente de la ***Centrale***, le seul changement concerne le bloc ***Générateur Trames***.
  - Celui-ci ne gère en effet plus les ***boutons***, ***interrupteurs*** et ***LEDs*** de la carte, mais se contente de recevoir une ***Commande*** de la part du ***Microblaze***.
  - Il faut donc supprimer les sous-blocs ***Interface Utilsateur*** et ***Registres de Configuration*** (remplacés par les ***slv_reg*** du ***Wrapper AXI)***
  - Le sous-bloc ***Constructeur de Trame*** prendra donc en entrée la ***Commande*** issue des ***slv_regA*** et peut retser inchangé.

- ***Créer dans Vivado une IP Centrale DCC pour le Microblaze.***
  - ***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 4. Implémenter le design et générer le bitstream.***

- ***Dans SDK, 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.***