Compte Rendu FPGA1 Projet DCC

——Centrale DCC sur FPGA

JIANG HONGBO 3602103

LI SONGLIN 3770906

I. Présentation du projet :

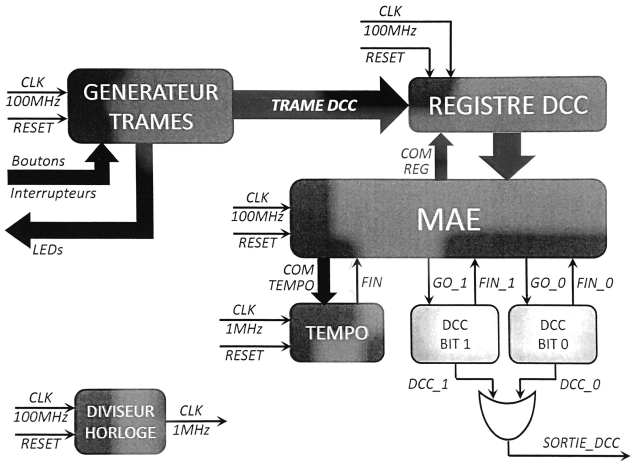
Le but de ce projet est d'envoyer les commandes à un modélisme ferroviaire.

La réalisation : On implémente une Centrale DCC dans le FPGA de la carte Nexys4-DDR. Après avoir envoyé les consignes via les boutons de la carte Nexys4-DDR, des commandes vont être amplifié par une carte Booster (amplificateur de signal) afin que le signal soit suffisamment puissant d'être envoyé sur les rails puis d'être décodé par les locomotives.

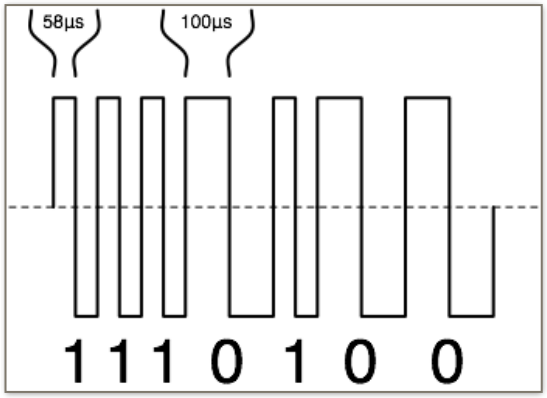
On a implémenté le code sur le FPGA de la carte Nexys4 DRR, et on a appuyé les boutons correspondants pour tester de différents fonctions avec différent vitesse sur les 2 trains. Et puis les LEDs affiche les valeurs d'adresse, de vitesse et de fonctions, aussi les trains roulent comme on a désigné. Donc on a réussi à envoyer les commandes au ferroviaire.

II. Implémentation de Protocole DCC

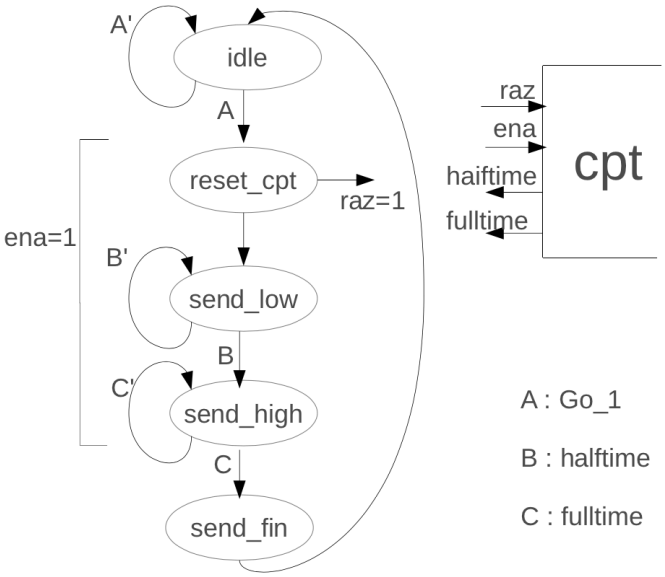
Le protocol DCC(Digital Command Control) est un standard défini par le NMRA (National Model Railroad Association), qui permet le contrôl individuel des locomotives ou des accessoires en modulant la tension d'alimentation de la voie.



1. Les bits :

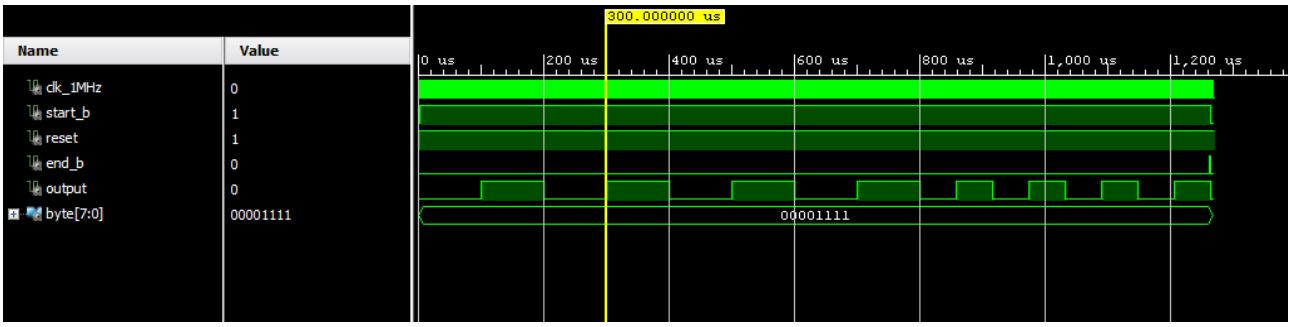


La tension de la voie est un signal continu bipolaire qui donne une forme de courant alternatif. Les inversions de tensions sont instantanées ce qui donne un signal pulsé. La durée de l'impulsion dans chaque sens fournit le codage du signal :Pour définir un bit de '1', la durée est courte (58µs pour un demi-cycle) alors qu'un bit '0' est représenté par une période longue (100µs pour un demi-cycle).

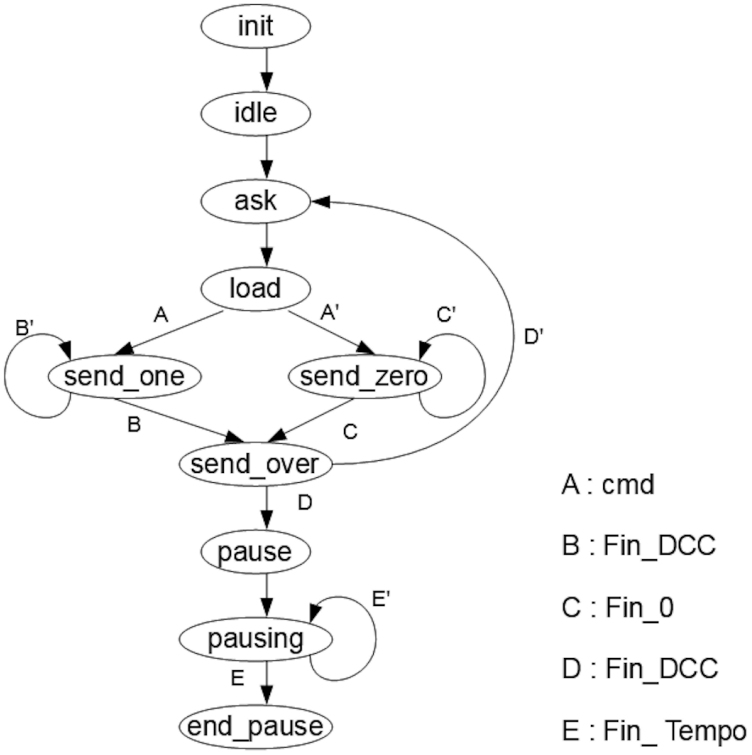


Automate et compteur de **send_one/send_zero** module

Ce sont les deux modules qui réalisent l'envoi d'un signal DCC correspondant à un 1 logique(SEND ONE) représenté par une impulsion à 0 de 58 µs puis une implusion à 1 de 58 µs, ou un 0 logique(SEND ZERO) représenté par une impulsion à 0 de 100 µs puis implusion à 1 de 100 µs.



Chronogramme de **send_one/send_zero** module première Post-synthesis simulation



Automate fini (**FSM**) module

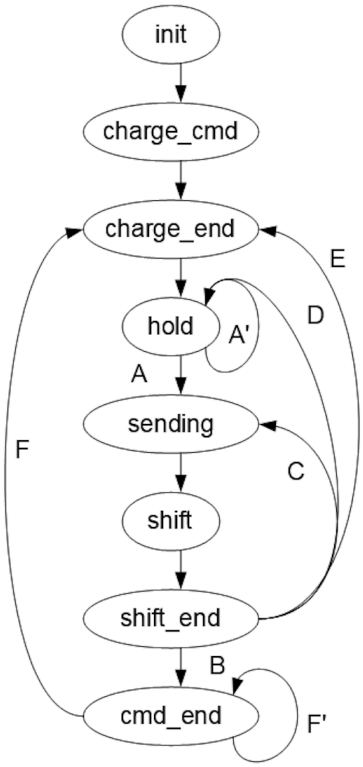
Si on utilise le language VHDL à simuler le matériel, il y a beaucoup de contraintes. Par exemple, chaque “if” dans un état de la machine à état doit avoir un “else”. C’est-à-dire que on ne peut pas laisser un signal sans donner une valeur chaque horloge, sinon, cela pose un problème pour le post-syntaxe.

(les dessins de simulation sont dans le compte de l'Esclangon, on a oublié à les récupérer… Quand on a trouvé cela et a revenu à l'Esclangon, toutes les salles de TP étaient fermées)

Simulation post- synthesis de Module send

2. Les trames :

Les locomotives et leurs accessoires (feux, effets sonores des engins moteurs) ainsi que les accessoires du réseau (aiguillages, itinéraires) possèdent chacun une adresse unique et décodent les instructions qui leur sont envoyées, et seules les instructions des paquets avec l'adresse correspondante seront activées par le décodeur. Le signal codé envoyé sur la voie donne des ordres aux équipements tout en fournissant la puissanc



A : com_reg = '01'

B : cpt = DATA_WIDTH

C : (cpt != DATA_WIDTH)
& (com_reg = '01')

D : (cpt != DATA_WIDTH)
& (com_reg = '00')

E : (cpt != DATA_WIDTH)
& ((com_reg = '11')
| (com_reg = '10'))

F : com_reg = '11'

Com_reg	Fonction
"01"	Shift
"00"	Hold
"11"	Charge Frame

Automate fini de **SHIFT REGISTER** module

Le shifter register est controlé par le **FSM**.

Les informations envoyées consistent en une séquence de bits (une trame) qui est utilisé pour coder l'un d'un ensemble d'instructions que le décodeur numérique fonctionnera sur. Les paquets doivent être précisément définis pour assurer le bon encodage et le décodage des instructions.

3. Les commandes (une séquence de 4 trames):

La composition de trame est la suivante :

<table>
<tr><th colspan="2">Trame DCC</th></tr>
<tr><td colspan="2">"11111111111111"</td></tr>
<tr><td colspan="2">'0'</td></tr>
<tr><td colspan="2">DCC_Param_Address (8 bits address 1/2)</td></tr>
<tr><td colspan="2">'0'</td></tr>
<tr><td>—si la function est dans (F13-F20)</td><td>—si la function cmd est 8 bits (F1-F12)</td></tr>
<tr><td rowspan="2">DCC_Param_Funct_plus</td><td>DCC_Param_Speed</td></tr>
<tr><td>DCC_Param_Funct</td></tr>
<tr><td colspan="2">'0'</td></tr>
<tr><td colspan="2">DCC_Param_Control(7 downto 0)</td></tr>
<tr><td colspan="2">‘1'</td></tr>
</table>

Pour donner les DCC_Param (8 bits), on a distribué 3 bits pour indiquer l’adresse, 2 bits pour indiquer la VITESSE en proposant 4 choixs et 2 bits pour indiquer la FONCTION en représentant 4 foncitions différentes :

L’octet de donnée de la trame de VITESSE est de la forme suivante : 01DXXXXX. Le bit D fixe la direction du train : le bit 0 pour une marche arrière et le bit 1 pour une marche avant. Les 5 bits de poids faible indiquent la vitesse de la locomotive. On a choisi 14 mode de vitesse, comme indiqué dans le tableau ci-dessous:

C	speed_step bits	Vitesse choisie (Speed)
1	“0001”	Stop (Arrêt)
1	“0010”	Speed_Step 2
1	“0011”	Speed_Step 4
1	“0100”	Speed_Step 8
…	…	…
1	“1111”	Speed_Step 28

Et l'interface pour contrôler le VITESSE:

Interface	Changement	Fonction
BTNC	speed_step = 0/1	stop/go
BTNU	speed_step + 2	Speed up
BTND	speed_step - 2	Speed down
BTNR/L	Change Address	Change train

- Pour stabiliser, on a introduit lock pour chaque bouton avec un compteur de 160 µs

L'octet de donnée de la trame de FONCTION permet d'activer ou de désactiver certaines fonctionnalités des trains, selon le tableau ci-dessous. Chaque fonction est activée en mettant el bit correspondant à 1, et est désactivée en mettant ce même bit à 0. Comme dans le tableau ci-dessous, on a distribué 2 bits pour représenter 4 fonctions. La composition de trame est la suivante :

Fonction	7 down 5	bit 4	3 down 0
F0-F4	100	XXXXX	
F5-F7	101	1	XXXX
F8-F12		0	XXXX

La composition de DCC_Param_Fonct (7 downto 0)

Et l'interface de controller des fonction est des interrupteur SW0 - SW14

Détail du fonctionnement des LED

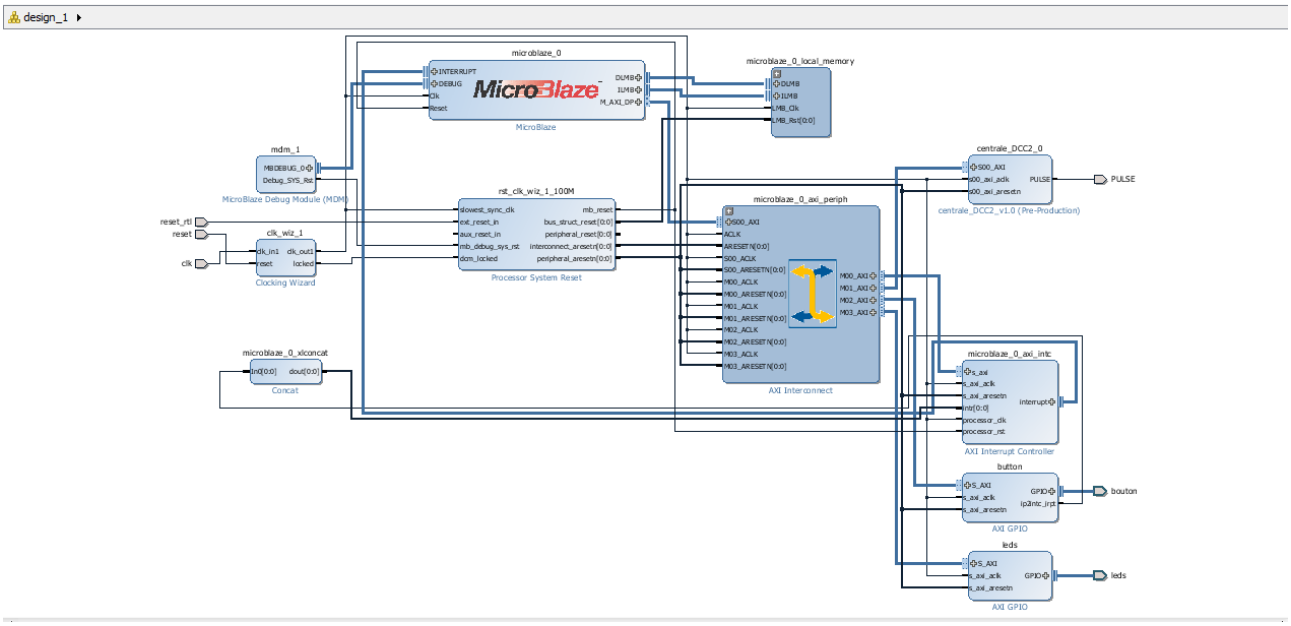
LED	Rôle	Function
LED(15)	DCC ON /OFF	Activer GO
LED(14 :12)	Valeur FONCTION/ SPEED/ADDRESS	- ADRESSE : 10 - VITESSE : 01 - FONCTION : 00
LED(6 :0)	Valeur FONCTION/ SPEED/ADDRESS	- FONCTION : 2 bits- LED(1:0) - VITESSE : 2 bits – LED(3 :2) - ADRESSE : 3 bits – LED(6 :4)

(les dessins de simulation sont dans le compte de l'Esclangon, on a oublié à les récupérer… Quand on a trouvé cela et a revenu à l'Esclangon, toutes les salles de TP étaient fermées)

III) Intégration de l'IP Centrale DCC dans un système Microblaze

Dans cette partie on va intégrer l'IP Centrale DCC dans un système mixte métériel/ logiciel organisé autour d'un Microblaze. Il faut donc remplacer l'interface de la partie précédente par un wrapper AXI afin que la Centrale DCC puisse échanger des informations avec le processeur.

Le wrapper AXI est créé automatiquement par Vivado, qui ajoute un certain nombre de registres de configuration (slv_reg). Ces registres sertent à faire le lien entre le code C exécuté par le Microblaze et la DDC_Frame qui donne les trames à envoyer aux trains.



1. Spécification et génération de la plate-forme matérielle

Créer une plate-forme Microblaze sous Vivado :

1.1 Dans la partie IP Integrator du Flow Navigator, cliquer sur Create Block Design. Une fenêtre Diagram se crée à droite de l'écran.

1.2 Pour instancier un Microblaze, cliquer sur Add IP dans le bandeau vert en haut de la fenêtre Diagram, taper les premières lettres de Microblaze et sélectionner Microblaze dans la liste filtrée des IP disponibles.

1.3 Un module Microblaze est ajouté au Diagram. Cliquer sur Run Block Automation puis /microblaze_0 pour configurer le processeur. Modifier les paramètres suivants :

a) Local Memory : Passer de 8 à 32KB.
b) Interrupt Controller : Cocher cette case.

2. Créer un IP Centrale DCC:

2.1 Aller dans le menu Tools, sélectionner Create and Package IP.

Une fenêtre s'ouvre : cliquer sur Next puis dans la fenêtre suivante choisir Create a new **AXI4** peripheral.

Dans la fenêtre suivante, donner un nom (**centrale_DCC**) et éventuellement une description à l'IP.
Dans la fenêtre suivante, choisir comme interface **bus AXI4-Lit**e.
Dans la fenêtre suivante, vérifier les paramètres sont bien ceux qu'on veut.
Dans la fenêtre suivante, terminer en cliquant sur Edit IP puis Finish.

2.2 Créer ou modifier la fonctionnalité de votre IP.
Aller dans le menu File et cliquer sur New File et créer un fichier **top_DCC.vhd**, dans ce fichier décrire l'architecture de l'IP Centrale DCC.

2.3 Ouvrir d'abord le fichier **central_DCC_v1_0.vhd**. Dans l'entité, après la ligne -- Users to add ports here, ajouter : **Out_to_train**: out std_logic;
Chercher la ligne – component declaration. Dans la liste des ports, ajouter avant S_AXI_ACLK la ligne suivante : **Out_to_train**: out std_logic;

Chercher la ligne port map(et ajouter dans la liste des instanciations la ligne suivante : **Out_to_train** => **Out_to_train**.

2.4 Ouvrir ensuite le fichier **centrale_DCC_v1_0_S00_AXI.vhd**.
Dans l'entité, après la ligne -- Users to add ports here, ajouter les I/O port de **top_DCC** et l'entité de lui .

2.5 Après sauvegarde des fichiers, l'entité led est à présent rattachée au module **centrale_DCC_v1_0_S00_AXI**.

2.6 Dans le fichier **centrale_DCC_v1_0_S00_AXI.vhd**, chercher la déclaration des signaux **slv_reg0,1,2,3** , ces 4 signaux correspondent aux 4 registres de configuration de votre IP. slv_regs <= **DCC_Param_REG**.

2.7 Cliquer sur IP File Groups puis sur Merge Changes from IP File Groups Wizard dans le bandeau jaune. Faire de même pour les autres IP Packaging Steps jusqu'à ce qu'il n'y ait plus que des verts devant chaque Packaging Step. Cliquer alors sur Review and Package puis sur Repackage IP. Vivado ferme alors la fenêtre du projet de l'IP.

3. Ajouter de l'IP Centrale DCC au système :

- Cliquer avec le bouton droit sur le Diagram et choisir Add IP. Chercher puis ajouter l'IP centrale_DCC. Cliquer sur Run Connection Automation. Vérifier que le paramètre Auto est sélectionné puis valider. Cliquer sur le Diagram avec le bouton droit et choisir Create Port. Ajouter les ports suivants : PULSE / Reset / clk. Connecter manuellement ce port aux sorties corrélatives.

-Cliquer sur Run Block Automation puis /centrale_DCC_0 pour configurer le processeur. Cliquer sur OK. Vivado ajoute alors automatiquement d'autres modules au Diagram.

- Cliquer sur Validate Design pour vérifier qu'il n'y a pas d'erreurs dans le Diagram.

Référence :

https://fr.wikipedia.org/wiki/Digital_Command_Control https://www.picotech.com/library/oscilloscopes/digital-command-control-dcc-protocol-decoding