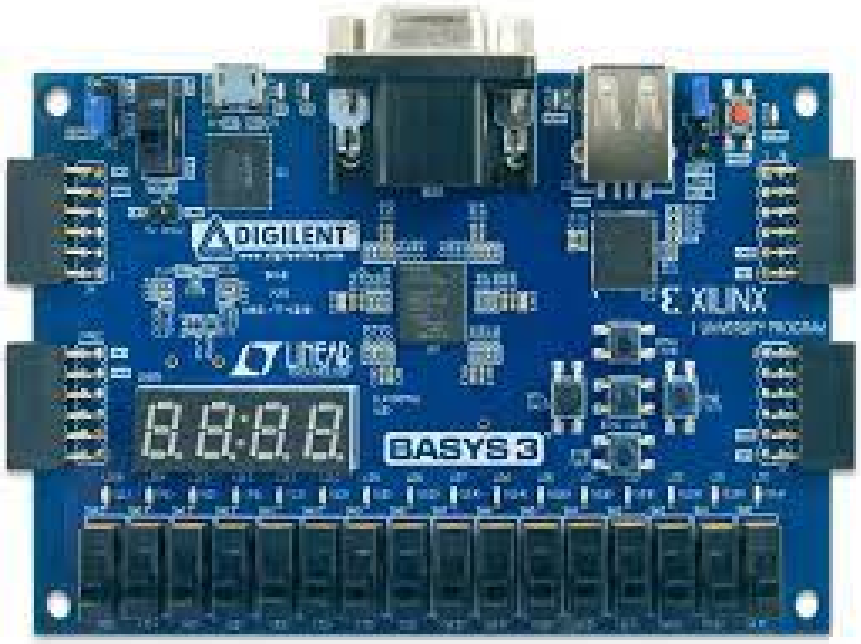
SORBONNE UNIVERSITÉ

Rapport TP : Mini Projet Centrale DCC



Systèmes Communicants

DOUADI Khaled
AL HARISS Nivine
FERRAH Lisa

July 28, 2025

Introduction

Dans le cadre de notre module **Systèmes Programmables** du parcours, nous avons réalisé un travail pratique visant à implémenter une centrale DCC (Digital Command Control) sur une carte FPGA Basys3. Le protocole DCC est largement utilisé dans le modélisme ferroviaire pour contrôler de manière individualisée les locomotives et les accessoires en modulant la tension d'alimentation des rails.

Ce projet a pour objectif de développer un système mixte matériel/logiciel capable de générer des signaux de commande DCC à partir d'une interface utilisateur simplifiée, utilisant les boutons de la carte Basys3. Le signal généré est ensuite amplifié via une carte Booster pour atteindre une puissance suffisante pour être transmis sur les rails et décodé par les locomotives.

Contexte et objectifs

Le TP est structuré en plusieurs parties, commençant par la compréhension et la mise en œuvre du protocole DCC. Les commandes DCC sont transmises sous forme de trames de données numériques, et chaque bit est représenté par une série spécifique d'impulsions. Nous avons étudié et programmé la structure de ces trames, incluant des champs tels que le préambule, le champ d'adresse, le champ de commande, et le champ de contrôle.

Architecture du système

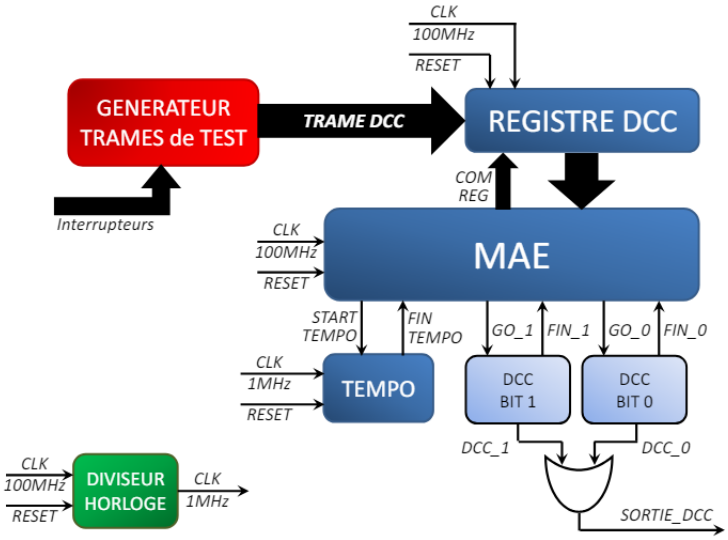


Figure 1: Architecture de la Centrale DCC

L'architecture de notre centrale DCC repose sur plusieurs modules clés implémentés en VHDL :

- **Diviseur d'horloge** : Génère un signal d'horloge de 1 MHz à partir de l'horloge de 100 MHz de la carte FPGA.
- **Module Tempo** : Mesure les intervalles de temps de 6ms entre la lecture des trames DCC .
- **Modules DCC_Bit_0 et DCC_Bit_1** : Génèrent respectivement les bits à 0 et à 1 selon le format DCC.
- **Registre DCC** : Gère le décalage des bits de la trame DCC à transmettre.
- **Machine à États (MAE)** : Coordonne les opérations des autres modules pour générer la trame DCC continue.

Développement et validation

I Présentation générale des codes

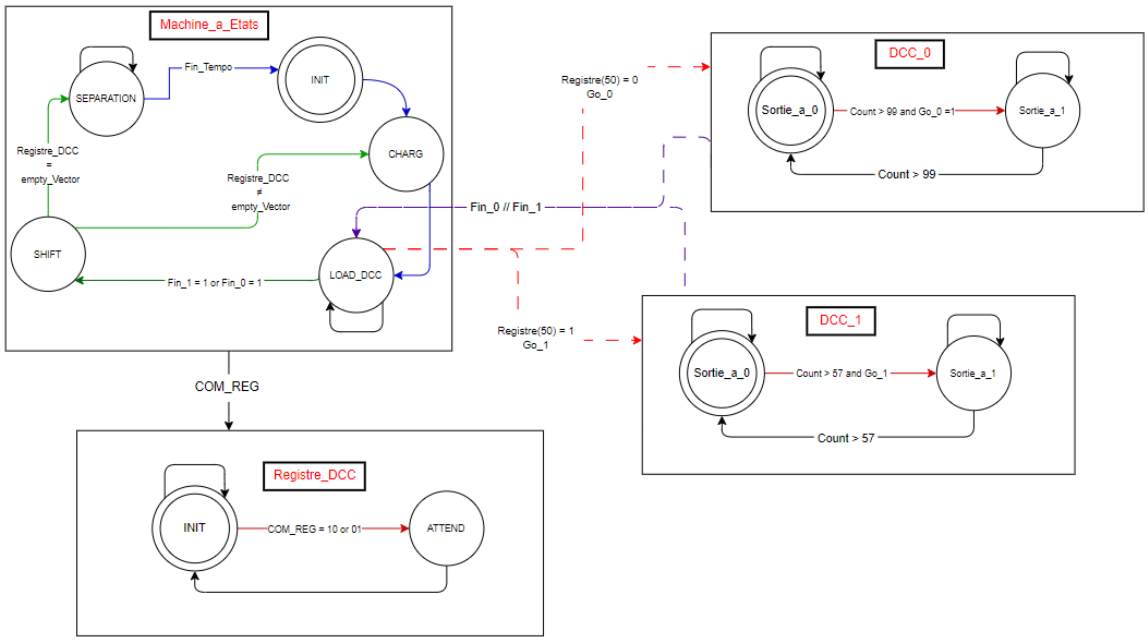


Figure 2: Présentation générales des machine à états des codes qu'on a écris et leurs interconnexion

L'architecture de notre centrale DCC repose sur plusieurs modules clés implémentés en VHDL :

Machine à états

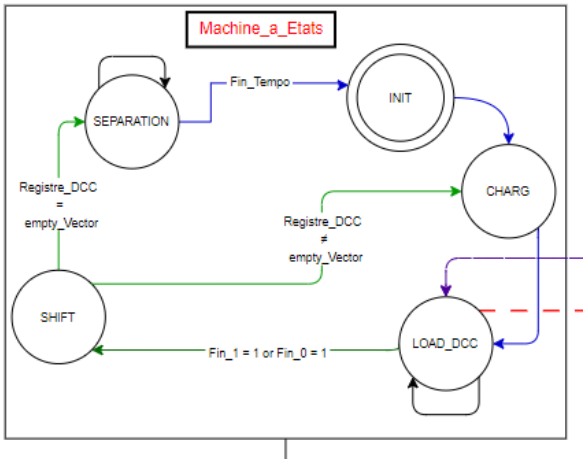


Figure 3: schéma bulles du Module Machine à états

Description: La machine à états (MAE) est un module crucial dans notre système DCC. Elle contrôle la séquence d'envoi des bits DCC en orchestrant la coordination entre les différents sous-modules responsables de la génération des signaux de commande. Ce module utilise une horloge à 100 MHz et un signal de réinitialisation asynchrone pour assurer une synchronisation précise et une récupération fiable en cas de réinitialisation.

Rôle: La machine à états joue un rôle central dans la gestion des opérations de génération de trames DCC. Elle suit une séquence d'états bien définis pour charger les trames dans un registre, déterminer le type de bit à générer (0 ou 1), et assurer que les trames sont transmises correctement avec les délais nécessaires entre les bits. En orchestrant ces opérations, la machine à états garantit que les trames DCC sont conformes aux spécifications du protocole.

États de la Machine à États: La MAE passe par plusieurs états pour accomplir sa tâche :

- **INIT (Initialisation)** : Dans cet état, la machine initialise les paramètres et prépare le chargement du registre DCC avec une nouvelle trame. Les signaux de commande et de démarrage sont réinitialisés pour s'assurer que le système commence dans un état propre.
- **CHARGER (Chargement)** : La machine attend que la trame DCC soit complètement chargée dans le registre. Durant cette phase, elle maintient les signaux de génération des bits à un niveau bas pour éviter toute génération prématurée.
- **LOAD_DCC (Chargement de la Trame DCC)** : La machine lit le bit le plus significatif (MSB) du registre DCC pour déterminer s'il faut générer un bit DCC_0 ou DCC_1. Si le traitement d'un bit précédent est terminé, elle passe à l'état suivant pour décaler le registre. Sinon, elle commande la génération du bit approprié.
- **SHIFT (Décalage)** : Dans cet état, la machine vérifie si le registre DCC est vide. Si c'est le cas, elle passe à l'état de séparation. Sinon, elle recharge la prochaine trame en revenant à l'état CHARG et continue le processus.
- **SEPARATION (Séparation)** : Cet état est utilisé pour insérer un délai de 6 ms entre les trames DCC. La machine attend que le délai soit écoulé avant de revenir à l'état INIT pour commencer à générer une nouvelle trame.

Connexions: La machine à états interagit avec plusieurs autres modules du système :

Zero_DCC

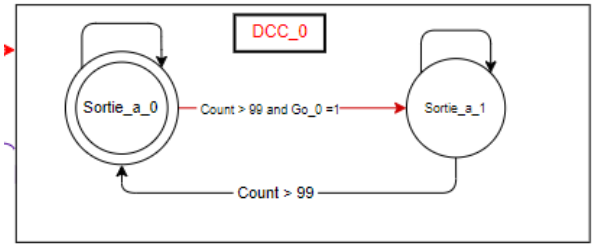


Figure 4: schéma bulles du Module Zero DCC

Description: Le module *Zero_DCC* est responsable de la génération d'un signal DCC représentant un bit '0'. Ce module fonctionne en produisant une pulsation spécifique qui respecte les spécifications du protocole DCC pour les bits '0'.

Rôle: Le rôle principal de ce module est de générer une pulsation de 100 microsecondes à l'état bas (0) suivie de 100 microsecondes à l'état haut (1), ce qui constitue un bit '0' selon le protocole DCC. Cette précision est essentielle pour que les trames DCC soient correctement interprétées par les locomotives.

États de la Machine à États: Le module *Zero_DCC* utilise une machine à états pour gérer la génération de la pulsation :

- **sortie_a_0** : Dans cet état, la sortie est maintenue à 0 pendant 100 microsecondes.
- **sortie_a_1** : Dans cet état, la sortie est maintenue à 1 pendant 100 microsecondes.

Fonctionnement:

- Lorsqu'un front montant de l'horloge à 1 MHz est détecté, le module commence à compter les cycles.
- Si le signal *go_0* est activé, la machine à états passe à l'état *sortie_a_0* et maintient la sortie à 0 pour 100 cycles (microsecondes).
- Une fois les 100 cycles écoulés, la machine passe à l'état *sortie_a_1* et maintient la sortie à 1 pour les 100 cycles suivants.
- À la fin de cette séquence, le signal *fin_0* est activé pour indiquer que la génération du bit '0' est terminée.

Connexions: Le module *Zero_DCC* interagit avec d'autres modules et signaux du système :

One_DCC

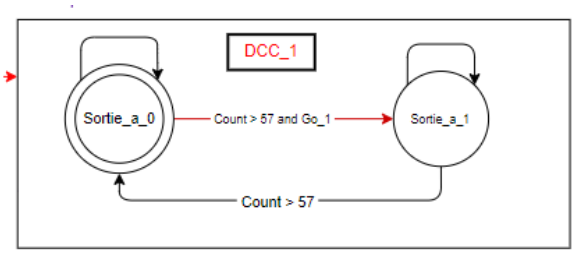


Figure 5: schéma bulles du Module One DCC

Description: Le module *One_DCC* est responsable de la génération d'un signal DCC représentant un bit '1'. Ce module produit une pulsation spécifique qui respecte les spécifications du protocole DCC pour les bits '1'.

Rôle: Le rôle principal de ce module est de générer une pulsation de 58 microsecondes à l'état bas (0) suivie de 58 microsecondes à l'état haut (1), ce qui constitue un bit '1' selon le protocole DCC. Cette précision est essentielle pour que les trames DCC soient correctement interprétées par les locomotives.

États de la Machine à États: Le module *One_DCC* utilise une machine à états pour gérer la génération de la pulsation :

- **sortie_a_0** : Dans cet état, la sortie est maintenue à 0 pendant 58 microsecondes.
- **sortie_a_1** : Dans cet état, la sortie est maintenue à 1 pendant 58 microsecondes.

Fonctionnement:

- Lorsqu'un front montant de l'horloge à 1 MHz est détecté, le module commence à compter les cycles.
- Si le signal *go_1* est activé, la machine à états passe à l'état *sortie_a_0* et maintient la sortie à 0 pour 58 cycles (microsecondes).
- Une fois les 58 cycles écoulés, la machine passe à l'état *sortie_a_1* et maintient la sortie à 1 pour les 58 cycles suivants.
- À la fin de cette séquence, le signal *fin_1* est activé pour indiquer que la génération du bit '1' est terminée.

Connexions: Le module *One_DCC* interagit avec d'autres modules et signaux du système :

Registre_DCC

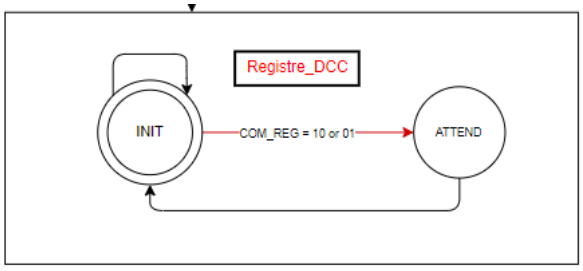


Figure 6: schéma bulles du Module Registre DCC

Description: Le module *Registre_DCC* est conçu pour stocker et gérer la trame de données DCC à envoyer. Ce registre permet de charger une trame complète et de la décaler bit par bit pour la transmission séquentielle.

Rôle: Le rôle principal de ce module est de recevoir une trame DCC complète, de la stocker, puis de la décaler à chaque cycle pour transmettre les bits successivement. Cela permet à la machine à états et aux générateurs de bits de traiter et d'envoyer chaque bit dans l'ordre correct.

États de la Machine à États: Le module *Registre_DCC* utilise une machine à états pour gérer le chargement et le décalage de la trame :

- **Init** : État initial où le registre est prêt à recevoir une nouvelle trame.
- **Attend** : État d'attente où le registre attend les commandes suivantes après avoir chargé la trame ou effectué un décalage.

Fonctionnement:

- Lorsque le signal *COM_REG* indique "01", la trame DCC est chargée dans le registre.
- Lorsque le signal *COM_REG* indique "10", le registre effectue un décalage à gauche, ajoutant un '0' au bit de poids faible.
- Dans l'état *Attend*, le module revient à l'état *Init* après avoir exécuté la commande de chargement ou de décalage.

Connexions: Le module *Registre_DCC* interagit avec d'autres modules et signaux du système :

TOP

Description: Le module *TOP* est le module principal qui intègre tous les sous-modules pour former le système complet de la centrale DCC. Il orchestre les interactions entre les différents composants pour assurer la génération correcte des trames DCC et leur transmission sur les rails.

Rôle: Le rôle principal de ce module est de coordonner tous les sous-modules (Machine à états, Zero_DCC, One_DCC, Registre_DCC, et autres) pour générer un signal DCC conforme. Il gère l'horloge, les commandes et les signaux de fin pour synchroniser les opérations de génération et de transmission des trames.

Fonctionnement:

- **Horloge et Réinitialisation** : Le module utilise une horloge principale à 100 MHz et un signal de réinitialisation asynchrone pour synchroniser tous les sous-modules.
- **Division de l'horloge** : Un diviseur d'horloge génère une horloge de 1 MHz à partir de l'horloge principale pour les besoins de temporisation précise des trames DCC.
- **Coordination des sous-modules** : Le module TOP active et coordonne les modules Zero_DCC et One_DCC pour générer les bits appropriés en fonction de la trame DCC chargée dans le registre.
- **Contrôle de la Machine à États** : La machine à états contrôle les séquences de chargement, de décalage et de génération des bits en fonction des commandes et des signaux de fin.

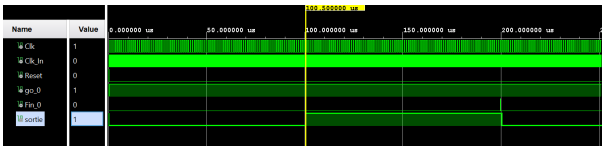
- **Interface Utilisateur** : Les interrupteurs sur la carte Basys sont utilisés pour définir les commandes à envoyer dans les trames DCC.

Connexions: Le module *TOP* interagit avec tous les sous-modules et les signaux externes pour former un système complet.

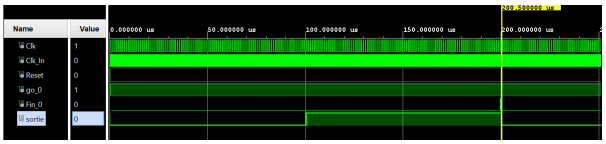
Simulation et Validation

Après plusieurs essais et plusieurs débogages et corrections des codes, nous sommes arrivés à la version finale qui ne contient aucune erreur. Nous présentons en fin de cette section les erreurs les plus importantes que nous avons pu rencontrer et comment nous les avons corrigées.

Simulation des DCC_0 et DCC_1 et vérification de période



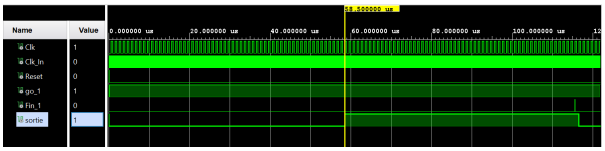
(a) Passage de sortie à 1



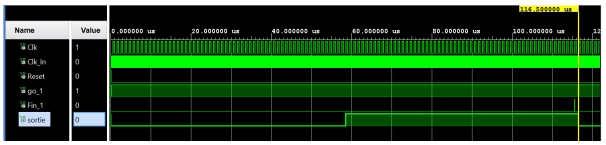
(b) Fin de cycle et génération de Fin_0

Figure 7: Résultats de la simulation TestBench du module Zero_DCC

Comme illustré dans la Figure 7, le module Zero_DCC génère une période de 200µs, suivie d'une courte pulsation de Fin_0.



(a) Passage de sortie à 1

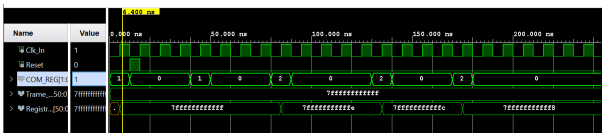


(b) Fin de cycle et génération de Fin_1

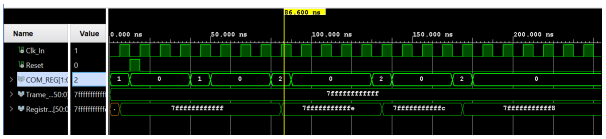
Figure 8: Résultats de la simulation TestBench du module One_DCC

Comme illustré dans la Figure 8, le module One_DCC génère une période de 116µs, suivie d'une courte pulsation de Fin_1.

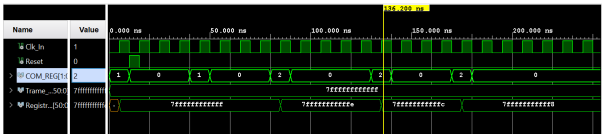
Simulation du Registre_DCC et du décalage



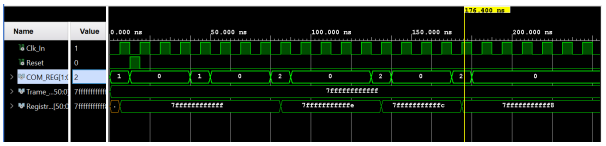
(a) Ordre de chargement du Registre_DCC avec le contenu de Trame_DCC



(b) Ordre de décalage du Registre_DCC pour la première fois



(c) Ordre de décalage du Registre_DCC pour la deuxième fois



(d) Ordre de décalage du Registre_DCC pour la Troisième fois

Figure 9: Résultats de la simulation TestBench du modules Registre_DCC

Lors de la simulation, comme illustré dans la Figure 9, nous avons initialement réglé le signal COM_REG à 01, ce qui a permis de charger correctement la trame DCC (Trame_DCC) dans le registre (Registre_DCC) (Figure 9a). La trame a été transférée sans erreur, et le contenu du registre a été mis à jour en conséquence. Ensuite, en réglant COM_REG à 10, nous avons observé le décalage correct de la trame dans le registre à chaque cycle d'horloge. Chaque bit a été décalé vers la gauche, ajoutant un '0' au bit de poids faible, comme montré dans les Figures 9b, 9c, et 9d. Les photos de la simulation montrent clairement ces transitions, confirmant que le module Registre_DCC fonctionne comme prévu.

Simulation totale de la machine à états avec le module TOP

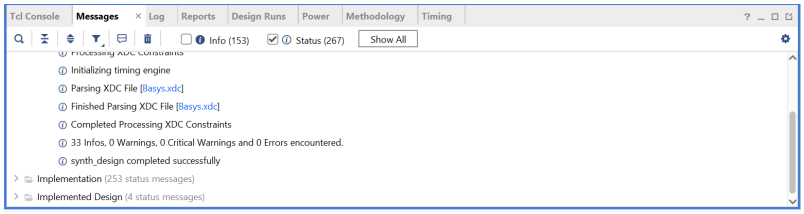


Figure 10: Message du résultat de simulation

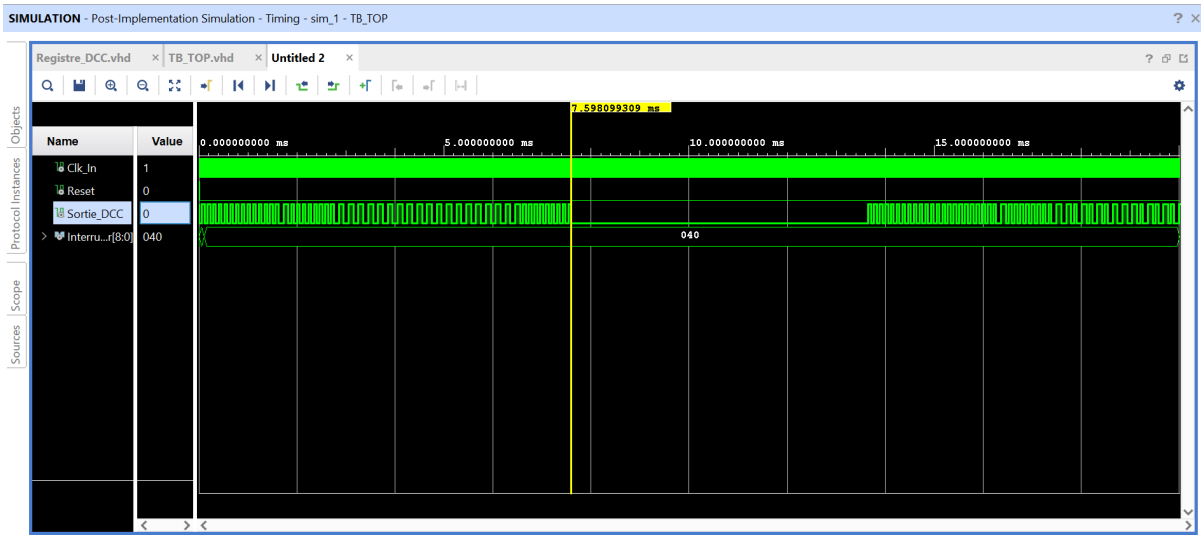


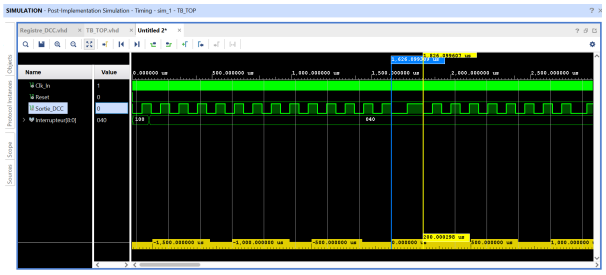
Figure 11: Simulation avec Post implementation Simulation timing

Comme illustré dans la Figure 10, notre code a été **correctement synthétisé sans erreurs ni avertissements**. Dans notre simulation post-implémentation (Figure 11), nous avons réglé le signal de Reset à 0 après 2µs.

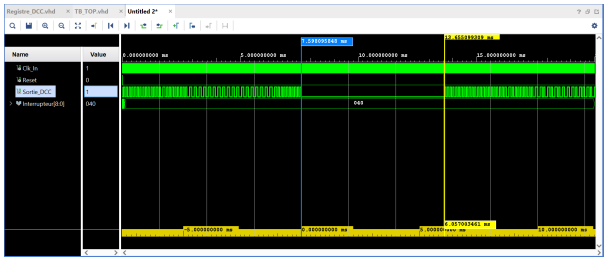
Le système commence alors à générer la trame DCC avec le registre initial **Trame_DCC = "11111111111111 0 11111111 0 00000000 0 00000000 0 11111111 1"**, car **l'interrupteur 8** est activé au démarrage.

Malgré le changement d'interrupteur, le registre n'est mis à jour qu'après le délai d'attente de 6 ms, moment où une nouvelle lecture de la trame DCC est effectuée. Cette mise à jour du registre donne **Trame_DCC = "11111111111111111111111 0 11111111 0 01010100 0 ("11111111" XOR "01010100") 1"**, ce qui se traduit par **Trame_DCC = "11111111111111111111111 0 11111111 0 01010100 0 (10101011) 1"**, comme montré dans les Figures 12a, 12b, 12c, 12d, et 12e.

(a) Vérification de la période du 1 généré

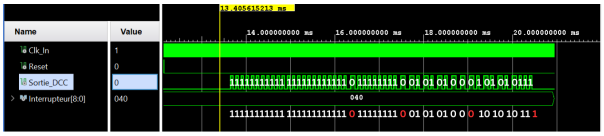


(b) Vérification de la période du 0 généré



(c) Vérification de la période d'attente 6ms

(d) le registre généré pour interrupteur(8) = 1



(e) le registre généré pour interrupteur(6) = 1

Figure 12: Résultats de la simulation TestBench du modules Registre_DCC

I.1 Problèmes rencontrés et solutions proposées

Durant notre travail, nous avons rencontré plusieurs erreurs de difficulté variable qui nous ont pris beaucoup de temps. Nous allons souligner les principaux problèmes qui nous ont poussés à rechercher et à documenter sur le web afin de trouver leurs solutions et de les résoudre :

1 **Le choix des valeurs des compteurs** Comme vu dans les codes de *Zero_DCC* et *One_DCC*, nous avons choisi les valeurs 57 et 99 pour nos compteurs. Ces valeurs n'ont pas été choisies au hasard : notre logique fonctionne avec une horloge de 1 MHz, ce qui signifie que chaque transition entre états ajoute une microseconde. Cela est dû à la simplicité de notre machine à états, composée de seulement deux états. Sinon, nous aurions dû calculer les délais en fonction du nombre d'états et des délais entre chaque état. Ce problème a été relativement facile à gérer comparé aux autres.

2 **Les inferring latches** Au début, nous pensions que les inferring latches étaient simplement dues à l'absence

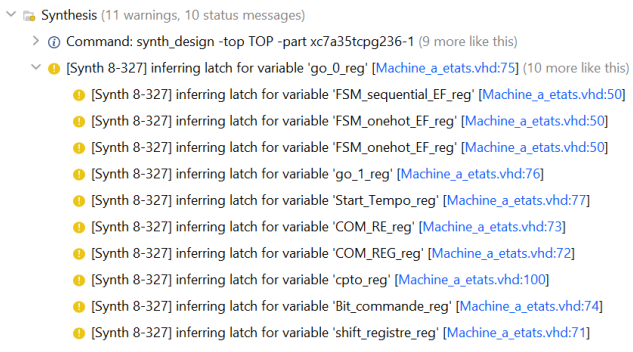


Figure 13: Message du résultat de simulation

de conditions *else* ou *others* dans les instructions *case* et *if* de nos codes. Cependant, après plusieurs ten-

tatives infructueuses, nous avons cherché la définition exacte des inferring latches : "Les latches inférés se produisent lorsqu'un synthétiseur de matériel déduit la nécessité de latches à partir de descriptions de code HDL ambiguës, souvent en raison de conditions de contrôle incomplètes dans les blocs de processus, entraînant un comportement de stockage non souhaité". Cela expliquait bien les problèmes rencontrés lors du débogage, où nous avions assigné des numéros d'étape à notre simulation comportementale pour identifier les problèmes. Nous avons remarqué que certaines valeurs, jamais définies dans notre code, apparaissaient dans la simulation. Nous avons donc dû revoir tout le code de notre machine à états et assigner des valeurs explicites pour éviter que le processeur ne se perde et n'assigne des valeurs aléatoires à nos variables ou signaux. Ces erreurs ont pris la majeure partie de notre temps.

3 **Le problème de synchronisation d'horloge** Ce problème nous a sans aucun doute causé le plus de difficultés. Notre simulation comportementale fonctionnait avec le code donné, mais une fois implémenté sur la carte, aucune sortie n'était visible. Après de multiples essais et débogages, nous avons réalisé que la synchronisation entre nos différents modules (notamment la machine à états et le registre) était incorrecte. Nous essayions d'imposer des changements sur le registre en continu, sans les contrôler par le *rising_edge* de notre horloge, ce qui causait des décalages multiples et le remplissage du registre avec des valeurs parasites. Ce problème nous a poussé à revoir toute la logique de synchronisation de nos codes. Nous avons dû réécrire le code de notre registre et de notre machine à états pour les rendre synchrones.

Conclusion

Ce TP nous a permis de comprendre et de maîtriser les concepts de base des systèmes programmables appliqués à un cas concret de modélisme ferroviaire. La réalisation de la centrale DCC sur FPGA nous a offert une expérience pratique précieuse en conception de systèmes numériques complexes, en programmation VHDL, et en utilisation d'outils de développement tels que Vivado et Vitis.

Nous avons rencontré divers défis tout au long du projet, notamment dans le choix des valeurs de compteur, la gestion des inferring latches et la synchronisation des horloges. Chaque problème a nécessité une analyse approfondie et une solution méthodique, nous permettant d'améliorer notre compréhension des systèmes embarqués et de la conception de circuits logiques.

La mise en œuvre finale a démontré que notre système pouvait générer et transmettre des trames DCC conformes aux spécifications, grâce à une coordination efficace entre les différents modules. Les simulations et validations ont confirmé le bon fonctionnement de notre design, offrant une base solide pour des développements futurs dans le domaine des systèmes de contrôle ferroviaire numériques.

Ce projet nous a également appris l'importance de la documentation et de la recherche de solutions en ligne, soulignant l'importance de rester informé des meilleures pratiques et des nouvelles technologies dans le domaine de l'ingénierie électronique.