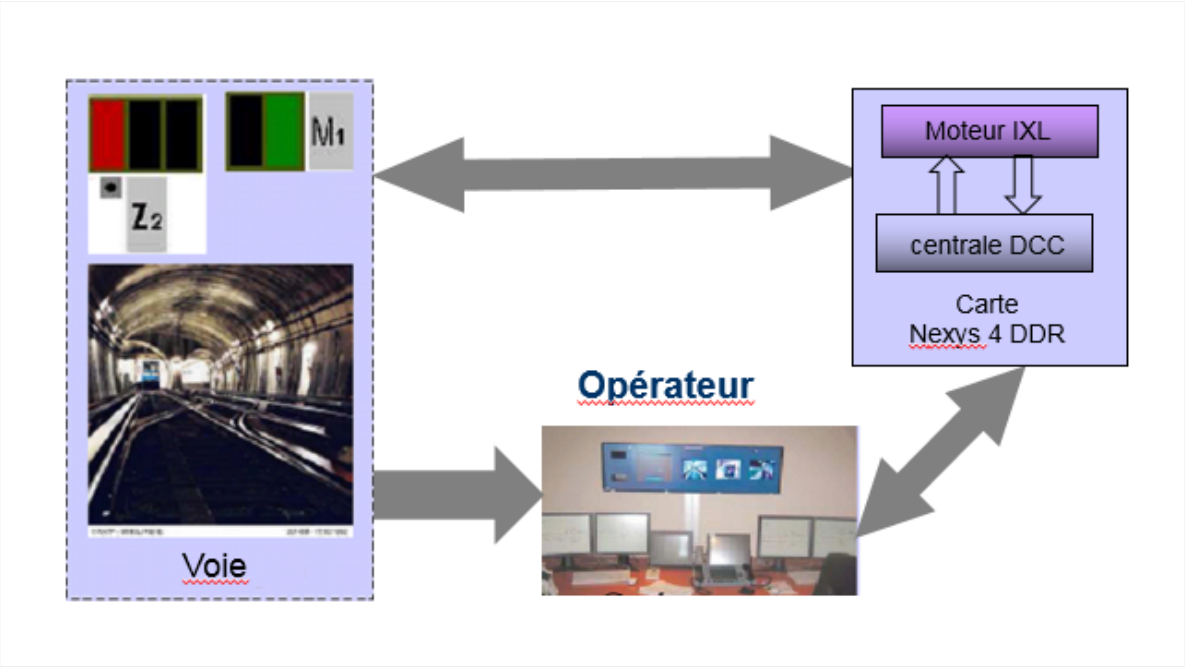
Développement sur FPGA d'un système d'aiguillage générique pour centrale DCC

Encadrant : J. Denoulet

Sommaire

Rappel fonctionnement système ferroviaire

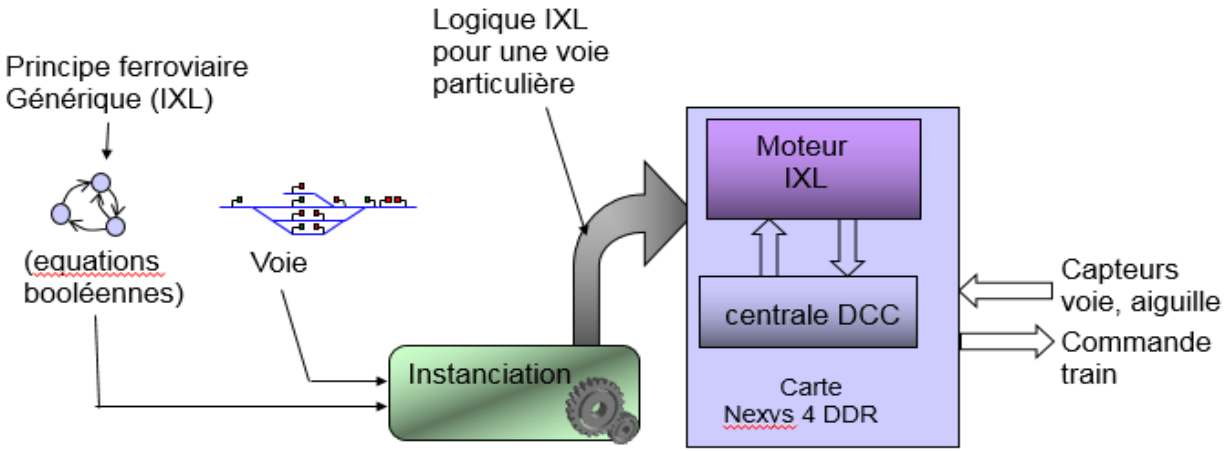
- La voie et les trains
- Le Poste de Commande Centralisé et les opérateurs
- Système d'enclenchement (informatisé ou non)



Sommaire

Architecture générale du projet

- La centrale DCC pour les commandes des équipements à la voie(aiguilles) et aux trains (faire avancer le train)
- Le moteur d'enclenchement (ex : moteur qui vérifie le droit de bouger les aiguilles) généré par les équations logiques
- La traduction des équations logiques (langage eq : gestion des enclenchements)



Définition des différents termes

- **TC : Track Circuit (Circuit De Voie)**

  Plus petite portion de voie entre 2 capteurs.

- **SE : Sensor (capteur)**

  Différents capteurs sur la voie.

- **SW : Switch (aiguillage)**

  Ce qui permet de passer d'un circuit à un autre

Définition des différents termes

UP -> train passé dans le sens UP
(clockwise)

DO -> train passé dans le sens DOWN
(conterclockwise)

ID -> aucun train n'est passé sur le capteur

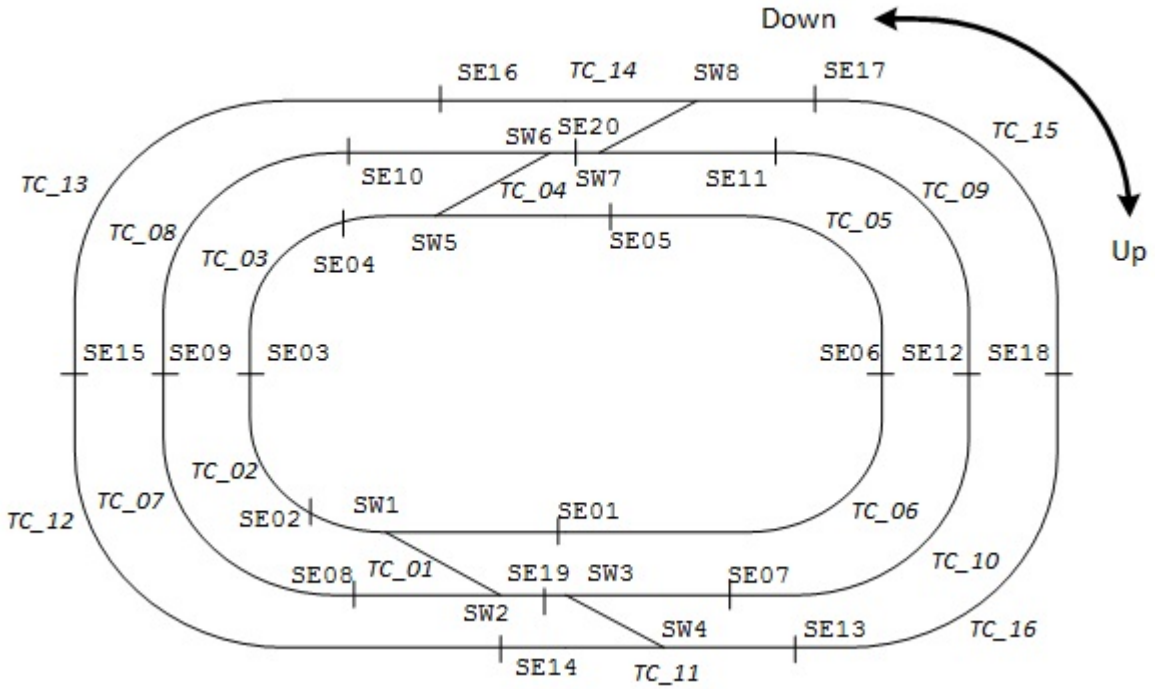
RI : bouger l'aiguille vers la droite

LE : bouger l'aiguille vers la gauche

Architecture du circuit utilisé durant le projet

L'installation est composée de :

- Ø 3 circuits imbriqués
- Ø 4 paires d'aiguillages
- Ø 20 capteurs



Sommaire

Objectifs

- **Porter la centrale DCC de la carte *spartan 6* à la carte *nexys 4 DDR*.**

- **Ajouter la gestion en sécurité des aiguillages et de l'espacement entre trains**

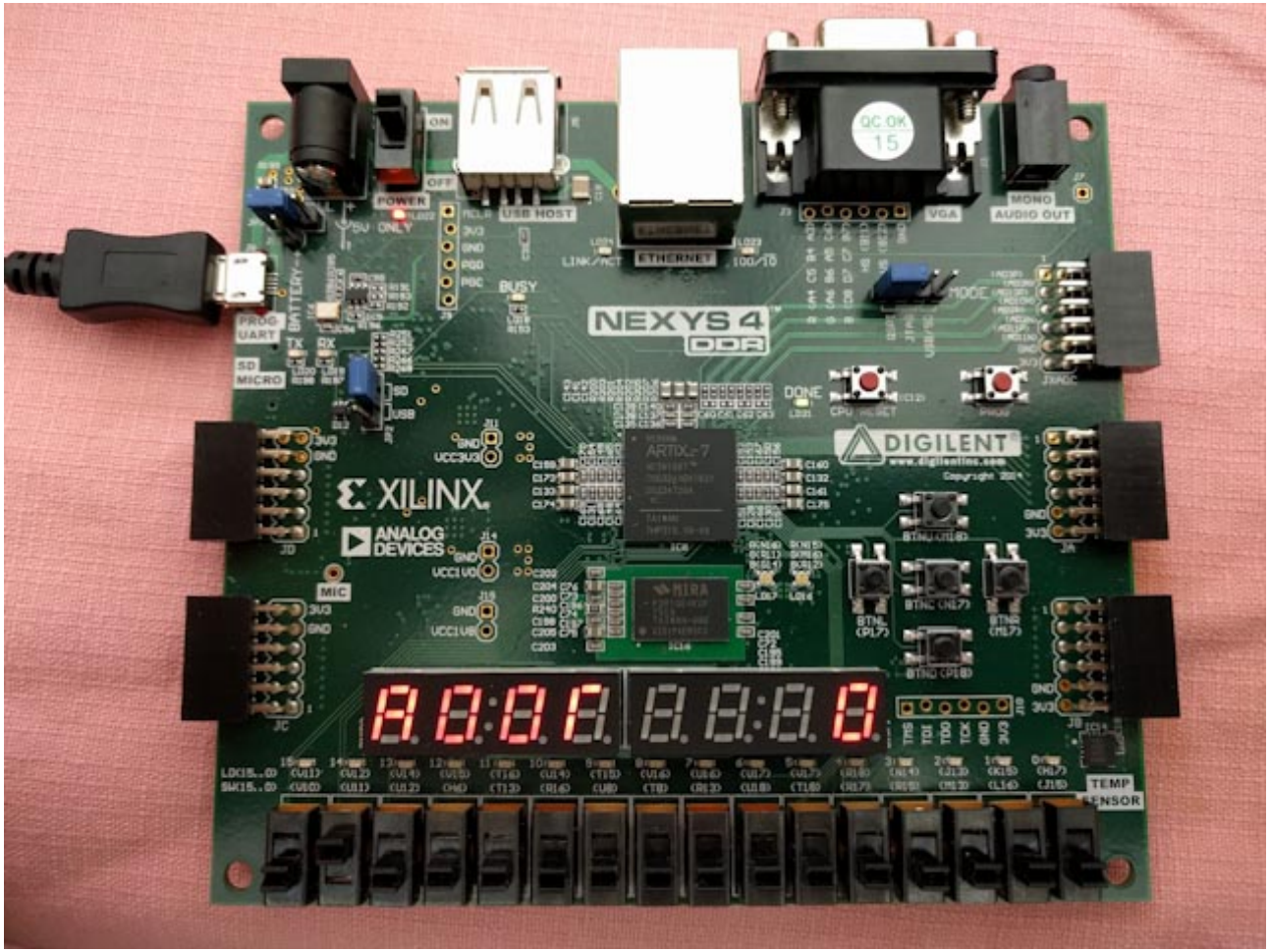
- **Ajouter l'automatisation des trains**

Réalisation

Portage de la centrale DCC de la carte *Spartan 6* à la carte *Nexys 4 DDR*.

- Création d'une nouvelle interface HM
- Portage du code de l'ancienne centrale et de la gestion des capteurs

Réalisation



Réalisation

- **Ajout de la gestion en sécurité des aiguillages et de l'espacement (gestion des enclenchements)**
  - Définition d'une interface entre le centrale DCC et le moteur IXL (communication entre les deux modules)
    - Création d'un langage de définition pour la gestion des enclenchements (« Eq »: Compilateur en Ocaml) et définition des équations génériques de gestion des enclenchements
  - Création d'un langage pour la génération des scénarios de simulation (« Simul »:Compilateur en Ocaml) et écriture des différents scénarios de simulation

Sommaire

Zoom sur la centrale DCC

- **La centrale DCC a été réécrite entièrement en VHDL à partir du code de la carte Spartan 6**

- **La centrale DCC communique avec le moteur IXL à travers l'interface en lui demandant des autorisations pour ensuite pouvoir les appliquer.**

- **La centrale DCC permet de commander plusieurs trains simultanément.**

- **La centrale DCC ne gère pas les capteurs de détection ni les aiguillages.**

Zoom sur le moteur IXL

- **Le moteur IXL est généré automatiquement à partir du langage d'équations Eq (Génération du code VHDL conforme avec ce qu'attend la centrale)**

- **Le moteur IXL communique avec la centrale DCC à travers l'interface en lui fournissant l'état des aiguillages et des capteurs**

- **Les équations sont définies génériquement et instanciées à chaque partie de la voie**

Interface Moteur IXL

```
entity Ixl is

  Port (
        -- synchro
        CLK         : in  STD_LOGIC;
        reset       : in  STD_LOGIC;

        -- input
        valid_in    : in  STD_LOGIC;
        Sw_Cmd_Req  : in  Sw_t;
        Sw_State    : in  Sw_t;
        Sensor      : in  SE_state;

        -- output
        valid_out   : out STD_LOGIC;
        Sw_Cmd_Aut  : out Sw_t;

        --debug output
        TC_out      : out TC_St
        );

end Ixl;
```

`Sw_Cmd_Req` : contient les demandes de changement d'aiguillage voulu

`Sw_State` : L'état actuel des aiguillages

`Sensor` : état des capteurs sur la voie

`Sw_Cmd_Aut` : Commande d'aiguillage autorisée

Equations Génériques

- **Définition des équations génériques d'enclenchement pour**
  - **Les Circuits de voie à partir des capteurs de détection**

$$TC_{nn} \Leftarrow \neg SE\_Up_{nn} \wedge \neg SE\_Do_{mm} \wedge (TC_{nn} \vee SE\_Do_{nn} \vee SE\_Up_{mm});$$

  - **La protection des aiguilles (pas de collision entre trains autour des aiguilles)**

$$SW\_AUT\_RI_{nn} \Leftarrow SW\_CMD\_RI_{nn} \wedge TC_{nn} \wedge TC_{mm};$$

Le langage Eq

- ***TC_02 <= ~SE_UP_02 * ~SE_DO_03 * (TC_02 + SE_DO_02 + SE_UP_03);***

- **Variable booléenne (TC/SW/SE) suivie de « <= »**
- **Suivi d'une équation logique**
  - Ø ~ pour l'opérateur not
  - Ø * pour l'opérateur and
  - Ø + pour l'opérateur or

Le langage Eq

- **Le code généré en VHDL**

```
if
  ((NOT (Sensor(1).dir = "01")) AND
  ((NOT (Sensor(2).dir = "10")) AND
  ((TC(1)='1') OR
    ((Sensor(1).dir = "10") OR
    (Sensor(2).dir = "01"))))) then
 TC(1) := '1';
else
  TC(1) := '0';
end if;
```

```
(~ Se_UP_02) *
(~ Se_DO_03) *
  (TC_02 +
   SE_DO_02 +
   SE_UP_03)
```

Sommaire

Besoins pour la réalisation des tests

- **Automatisation des simulations (répétition des tests)**
- **Utilisation d'un langage de simulation permettant de ne pas avoir à coder en VHDL**
- **Lisibilité des scénarios**
- **Automatisation des résultats des tests**

Le langage Simul

```
Cycle 4 : -- Move Sw 4
Events
  SE_ID_2;
  SE_UP_3;  -- enter into TC3
  SW_CMD_RI_04; -- command SW4 Right
Outputs
  TC_02 = Free; --
  TC_03 = Occ;  --
  SW_AUT_RI_04 = Aut; -- Autorization Sw 04 Right
```

Faire passer le train du secteur 2 au 3 et demander de bouger l'aiguillage 4 vers la droite

Code généré en VHDL

```
CLK <= '0';
  wait for 1 ns;
  report "Cycle 4: -- Move Sw 4";
  Sensor(1).dir <= "00";
  Sensor(2).dir <= "01";
  Sw_Cmd_Req(6) <='1';

  CLK <= '1';
  wait for 1 ns;
  if (TC_out(1) = '1') then report "--" & " : Pass"; else report "--" & " :
Fail."; end if;
  if (TC_out(2) = '0') then report "--" & " : Pass"; else report "--" & " :
Fail."; end if;
  if (Sw_Cmd_Aut(6) = '1') then report "-- Autorization Sw 04 Right" & " : Pass";
 else report "-- Autorization Sw 04 Right" & " : Fail."; end if;
```

Events

SE_ID_02 ;

SE_UP_03 ;

SW_CMD_RI_04 ;

Outputs

TC_02 = FREE ;

TC_03 = Occ ;

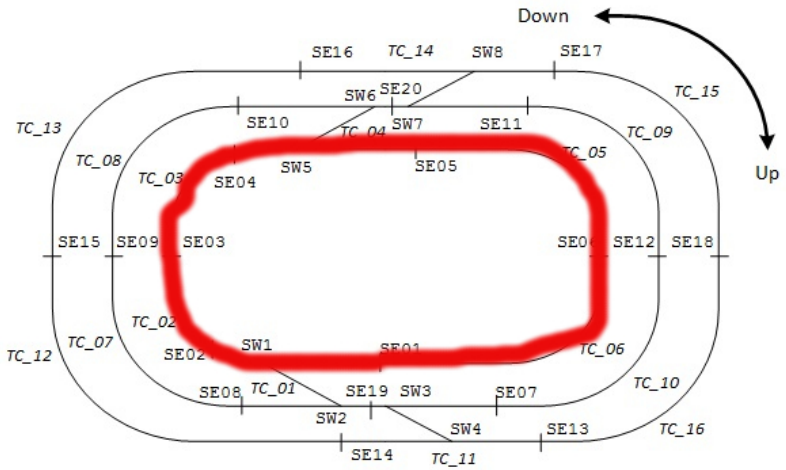
SW_AUT_RI_04 = Aut ;

Sommaire

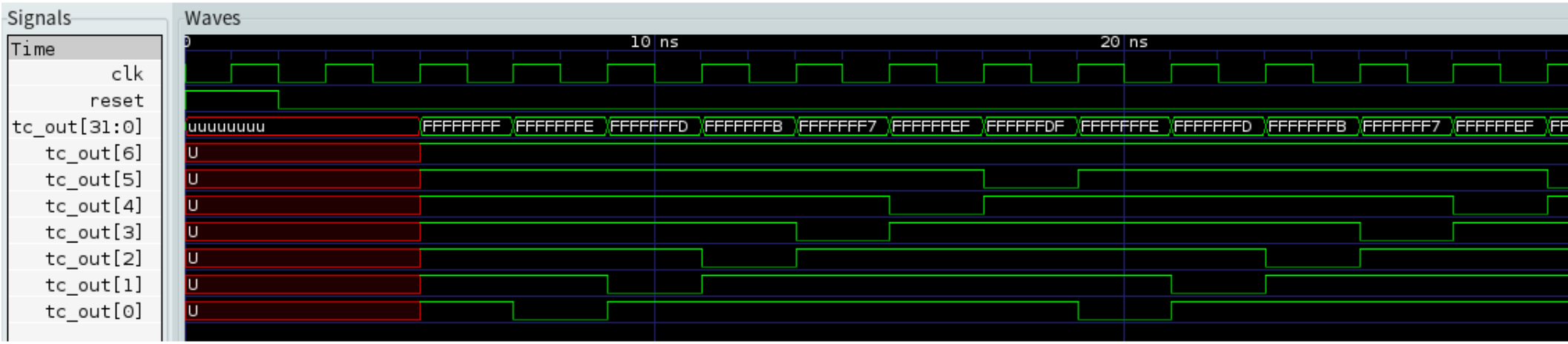
Scénario 1

- **1 train fait 2 tours du circuit interieur**

Ø **Permet de tester la gestion des Circuits de voies**



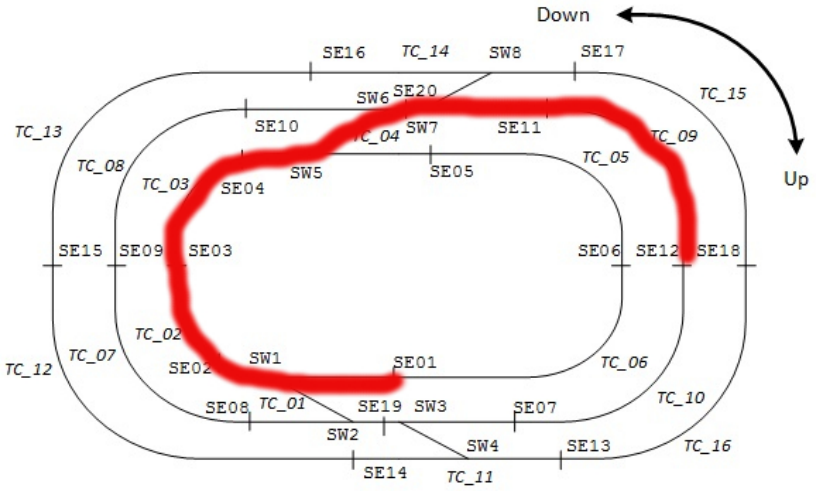
Scénario 1 simulation



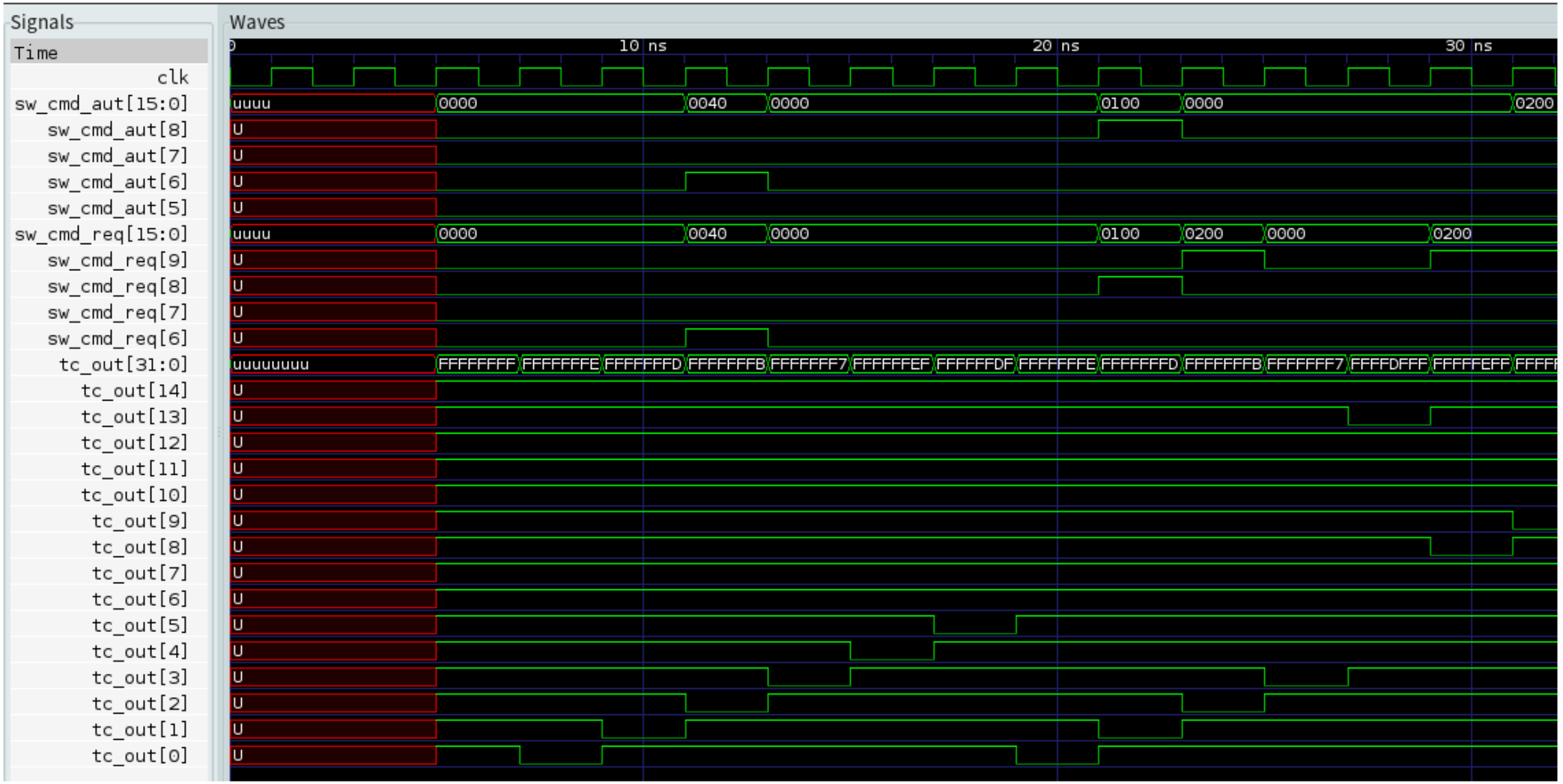
Scénario 2

- **1 train doit passer de la voie A à la voie B.**

Ø **Permet de tester les demandes d'aiguillages**



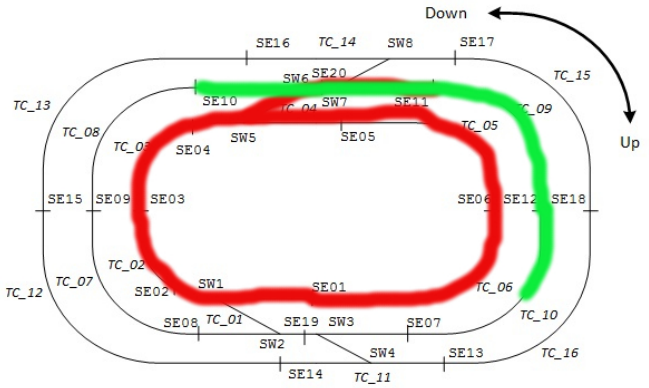
Scénario 2 simulation



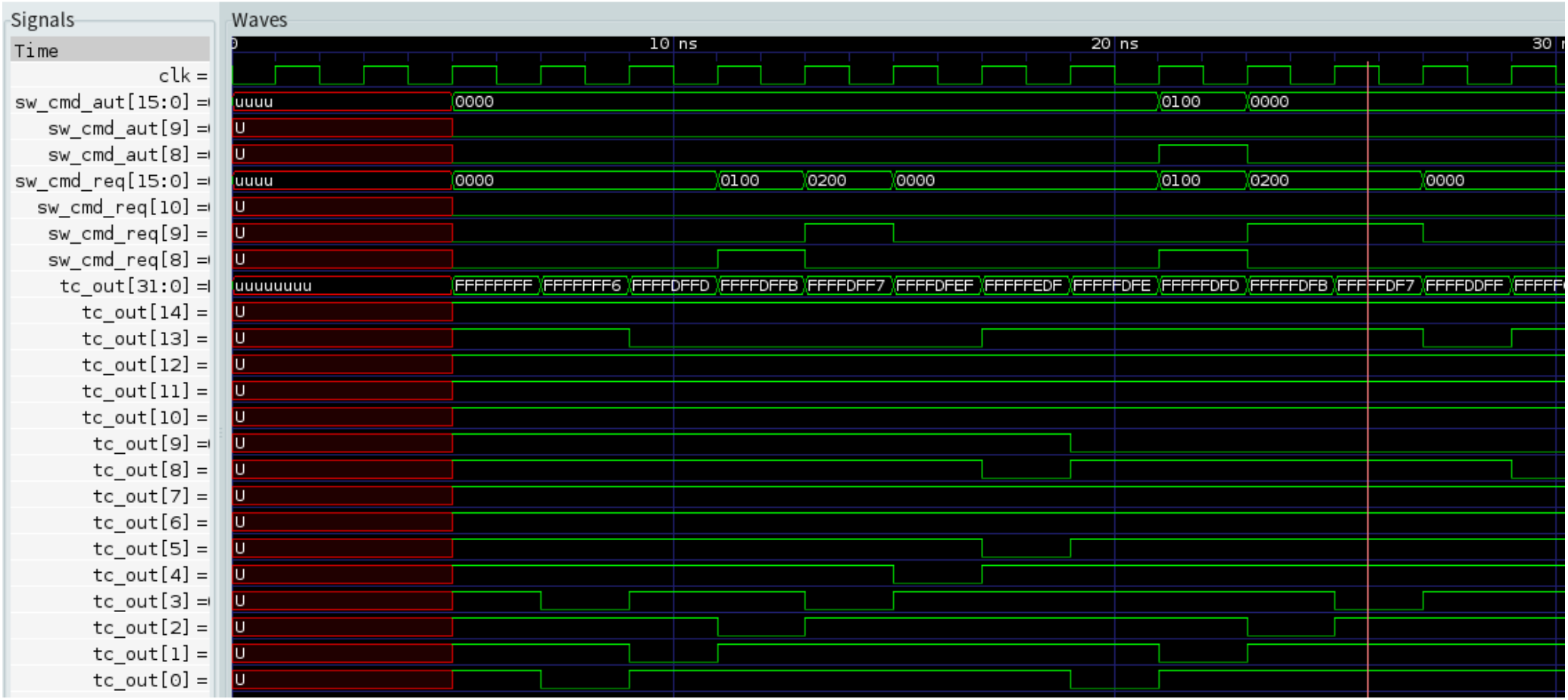
Scénario 3

- **Plusieurs trains en zone d'aiguillage**
  - Train 1 sur la voie A qui s'arrête dans la zone d'aiguillage,
  - Train 2 sur voie B qui tente de passer en voie A,
  - Après un temps le Train 1 redémarre ;
  - Train 2 qui peut passer en voie A.

Ø **Permet de tester le circuit avec plusieurs trains et d'éviter des collisions.**



Scénario 3 simulation



Conclusion - Partie technique

- La centrale DCC permet de commander plusieurs trains
- Le moteur IXL est généré automatiquement et est générique (fichier d'équations)
- Un outil de simulation automatique de scénarios de tests a été créé
- Documentation complète de l'interface et des langages Eq et Simul

Conclusion - Partie projet

- **Ce projet m'a permis d'appronfondir ma connaissance de la gestion en sécurité des trains et de ma connaissance en programmation VHDL.**

- **Utilisation de l'outil de gestion de configuration GIT et du langage Latex**

This is the end

Merci

Question ?