HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Reconfigurable & embedded
Digital Systems

Laboratoire 6

Mesure du temps de réaction

Départements : TIC
Unité d'enseignement ARE

Auteurs : **Rodrigo Lopez Dos Santos**
Urs Behrmann
Professeur : **Etienne Messerli**
Assistant : **Anthony Convers**
Classe : **ARE**
Salle de labo : **A07**
Date : **27.01.2025**

Contents

1 Introduction

2 Analyse et conception

2.1 Plan d'adressage

Address (offset)	Read	Write
0x00	[31..0] Interface user ID	reserved
0x04	[31..4] "0..0" [3..0] buttons	reserved
0x08	[31..10] "0..0" [9..0] switches	reserved
0x0C	[31..10] "0..0" [9..0] leds	[31..10] reserved [9..0] leds
0x10	[31..28] reserved [27..21] hex3 [20..14] hex2 [13..7] hex1 [6..0] hex0	[31..28] reserved [27..21] hex3 [20..14] hex2 [13..7] hex1 [6..0] hex0
0x14	[31..2] reserved [1] interrupt [0] statut flanc montant	[31..1] reserved [0] clear interrupt
0x18	[31..1] reserved [0] interrupt mask	[31..1] reserved [0] set interrupt mask
0x1C	[31..1] reserved [0] Max10 status	[31..0] reserved
0x20	[31..0] reserved [0] Max10 busy	[31..0] reserved
0x24	[31..0] reserved	[31..1] reserved [0] Max10 CS
0x28	[31..0] reserved	[31..0] reserved [0] Max10 data

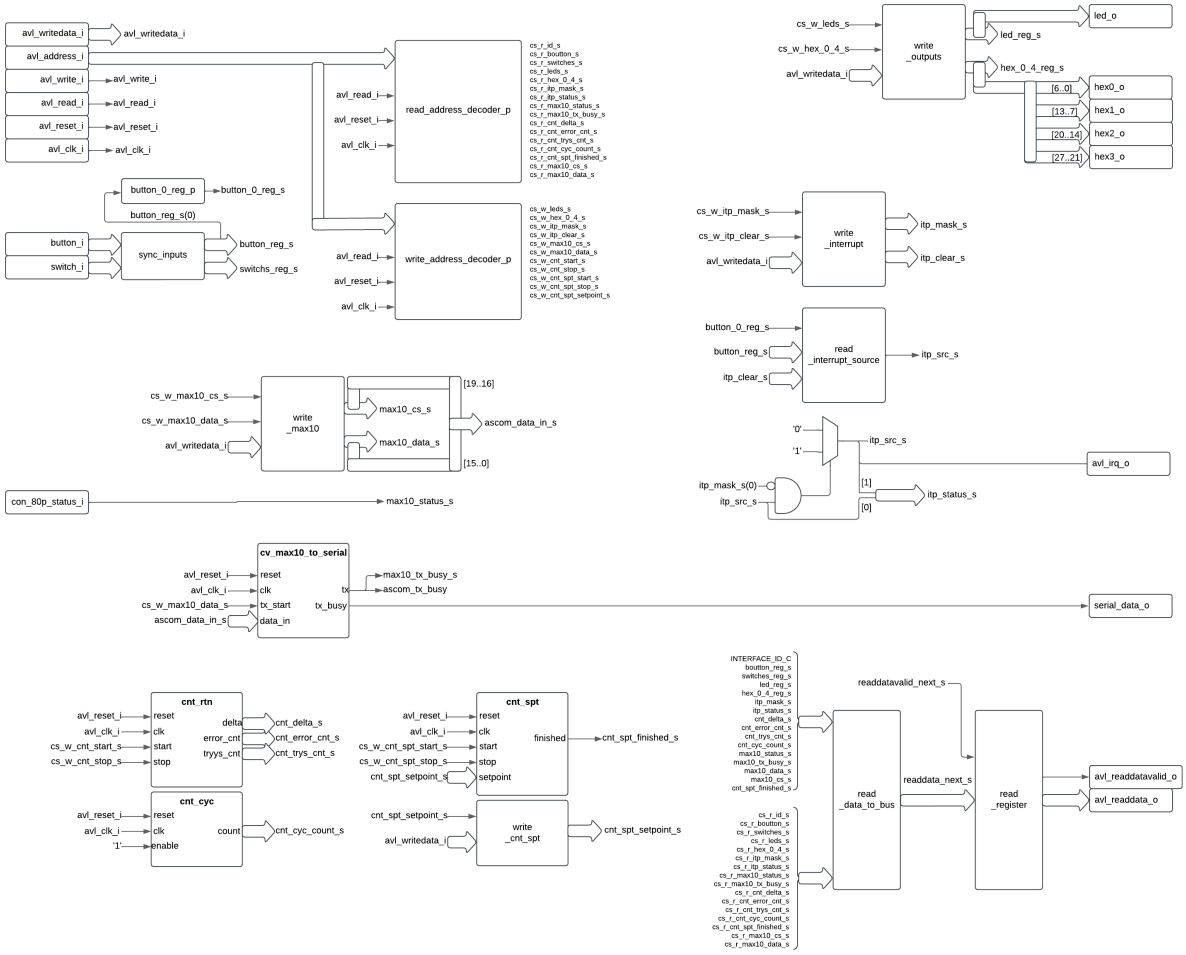
Address (offset)	**Read**	**Write**
0x2C	[31..0] reserved	[31..0] reserved [0] Counter start
0x30	[31..0] reserved	[31..0] reserved [0] Counter stop
0x34	[31..0] reserved [0] Counter delta	[31..0] reserved
0x38	[31..0] reserved [0] Counter error	[31..0] reserved
0x3C	[31..0] reserved [0] Counter cycle count	[31..0] reserved

2.2 Emetteur série asynchrone

Pour l'émetteur série asynchrone, nous avons dû utilsé une baudrate de 9600. Mais la vitesse de l'horloge étant de 50 MHz, nous avons dû utiliser un timer pour diviser cette fréquence. Nous avons donc utilisé un timer qui compte jusqu'à 5208 pour obtenir une fréquence de environ 9600 Hz.

3 Interface (VHDL)

3.1 Schéma



3.1.1 Composants

counter_cycle.vhd

Implémente un compteur cyclique utilisé pour mesurer des périodes définies, probablement à 20 ns de précision.

counter_reaction.vhd

Spécifique à la mesure du temps de réaction, il démarre et arrête le comptage en fonction des signaux d'entrée.

counter_setpoint.vhd

Permet de définir un seuil ou un point d'arrêt pour le comptage.

serial_async_transmitter.vhd

Implémente l'émetteur série asynchrone de 20 bits, sans parité, pour communiquer avec la carte Max10_leds. Prend en charge le protocole spécifié (bit de départ, 20 bits de données, bit d'arrêt).

3.1.2 Processus

write_outputs

Gère les sorties des LEDs et des afficheurs 7 segments (HEX) en écrivant les données reçues depuis le bus Avalon.

read_address_decoder

Décode les adresses Avalon pour diriger les lectures vers les périphériques appropriés, comme les boutons et interrupteurs.

write_interrupt

Permet de masquer, activer ou acquitter les interruptions générées par KEY0, en interagissant avec les registres d'interruption.

read_interrupt_source

Renvoie l'état des interruptions et des détections de flanc pour une gestion correcte des signaux.

sync_inputs

Synchronise les entrées des boutons et interrupteurs pour garantir leur stabilité face aux métastabilités.

write_max10

Prend en charge la conversion des données pour les transmettre à la carte Max10_leds via l'émetteur série.

write_cnt_spt

Permet de configurer le seuil du compteur 'cnt_spt' et de démarrer ou arrêter son fonctionnement en fonction des besoins.

read_data_to_bus

Lit les données des registres internes (par exemple, valeurs des compteurs ou état des périphériques) et les transmet via le bus Avalon pour qu'elles soient accessibles au processeur.

4 Programme (C)

4.1 fichiers & utilisation

4.1.1 UART (uart.c / uart.h)

- Fournit des fonctions pour configurer et gérer la communication série (baudrate, envoi de messages, gestion FIFO).
- Utilisé pour afficher des messages dans un terminal distant via UART.

4.1.2 Interruptions (exceptions.c / exceptions.h)

- Configure le contrôleur générique d'interruptions (GIC).
- Définit les vecteurs d'exceptions pour gérer des interruptions (notamment pour le bouton KEY0).
- Gère l'interruption principale avec la fonction fpga_ISR.

4.1.3

- Avalon (avalon.h, avalon_functions.c / avalon_functions.h)
- Encapsule l'accès au bus Avalon pour interagir avec les registres des périphériques (LEDs, switches, afficheurs).
- Fournit des macros de lecture/écriture pour manipuler les données efficacement.

4.1.4 Application principale (app.c / app.h)

- Implémente la logique du jeu en utilisant une machine à états :
- Initialisation (APP_INIT).
- Attente (APP_WAIT).
- Début du jeu (APP_START_GAME).
- Erreur (APP_ERROR).
- Gère les entrées (boutons, interrupteurs) et affiche les résultats sur l'UART et les LEDs.
- Implémente l'ISR de la FPGA (KEY0), ISR utilisé pour la gestion de fin de jeu lorsqu'une jeu est commencé (APP_START_GAME)

4.2 Logique principale

L'objectif du programme est de mesurer le temps de réaction de l'utilisateur en utilisant les composants de la carte DE1-SoC et MAX10. Voici la logique étape par étape :

4.2.1 Initialisation

- Configure l'UART, le GIC, et les périphériques via Avalon.
- Désactive les interruptions jusqu'à ce que l'application soit prête.
- Affiche les instructions utilisateur sur l'UART.

4.2.2 Machine à états

- APP_INIT : Configure les LEDs, afficheurs, et active les interruptions.
- APP_WAIT : Surveille les entrées utilisateur (boutons, interrupteurs) pour déterminer l'état suivant.
- APP_START_GAME : Affiche le symbole de début et démarre le compteur.
- APP_INIT_GAME : Configure un temps aléatoire pour démarrer la mesure.
- APP_ERROR : Gère les erreurs (ex. : configuration invalide de la carte MAX10).

4.2.3 Gestion des interruptions

- Lorsqu'un utilisateur appuie sur KEY0, une interruption déclenche :
- Le calcul du temps de réaction via un compteur.
- L'affichage des résultats (UART, LEDs, HEX).
- La mise à jour des statistiques (meilleur/pire temps, erreurs, essais).

4.2.4 Affichage des résultats :

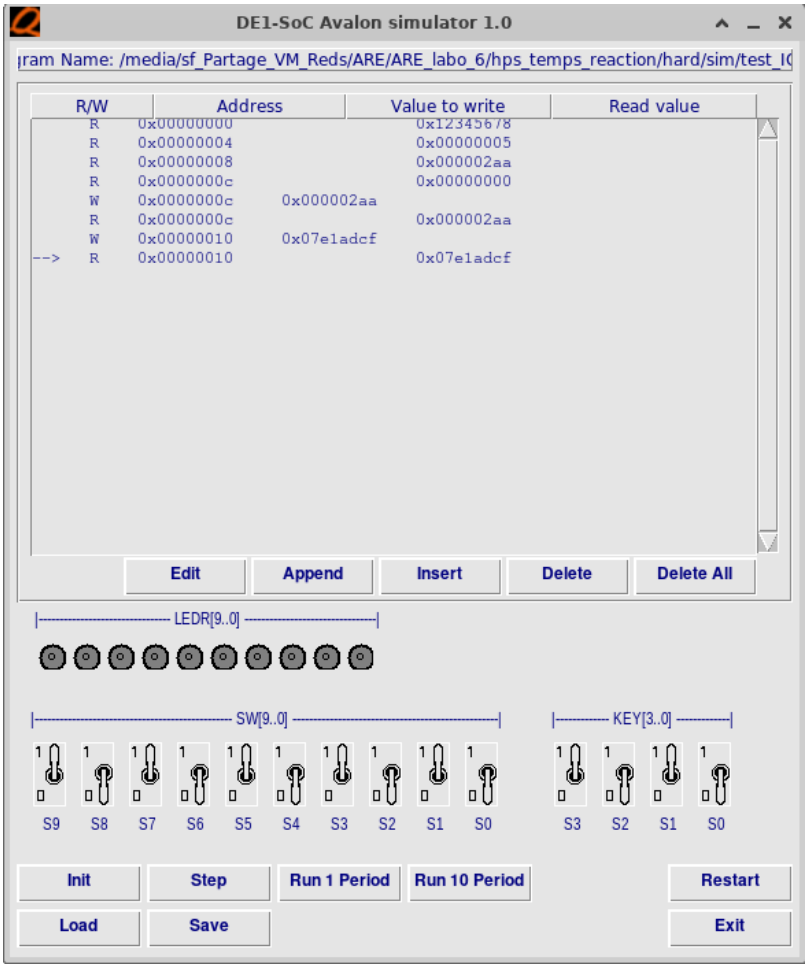
- Les afficheurs HEX montrent les données (dernier temps, meilleures performances, erreurs, essais) selon la configuration choisie via SW0 à SW3.
- L'UART fournit un résumé complet des performances.

5 Tests

5.1 Simulation (interface)

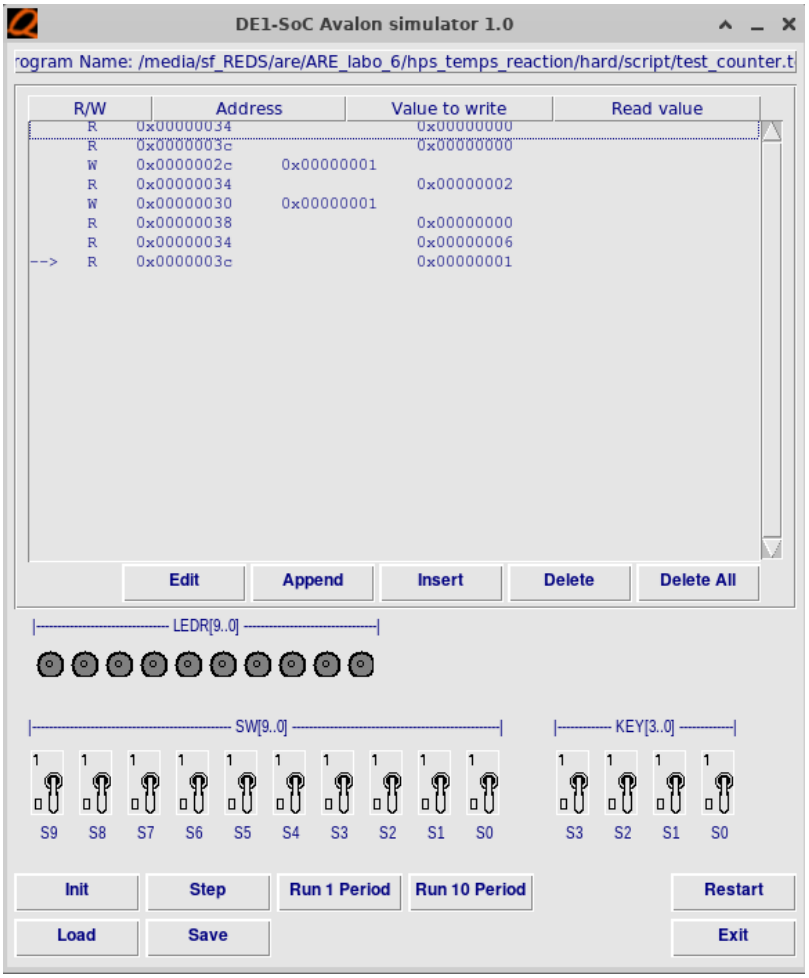
5.1.1 IOs

On a fait une première simulation pour vérifier les IOs. On a fait une suite de R/W sur les adresses de la mémoire. On a vérifié que les valeurs lues correspondaient aux valeurs écrites.

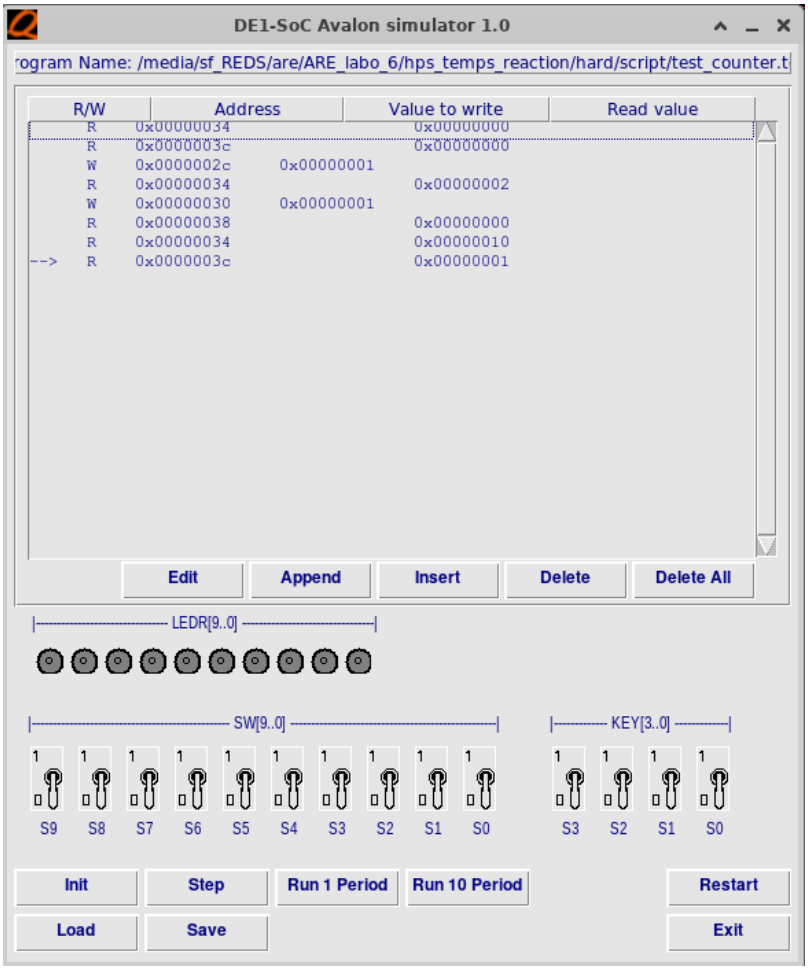


5.1.2 Compteurs

On a fait une deuxième simulation pour vérifier le compteur. On a fait un test avec 0 périodes supplémentaires. On voit bien que le compteur s'incrémente de 2 avec chaque instruction. On a un total de 6 périodes. entre le début et la fin du comptage.



Avec 10 périodes supplémentaires, on voit qu'on a un total de 16 périodes entre le début et la fin du comptage.

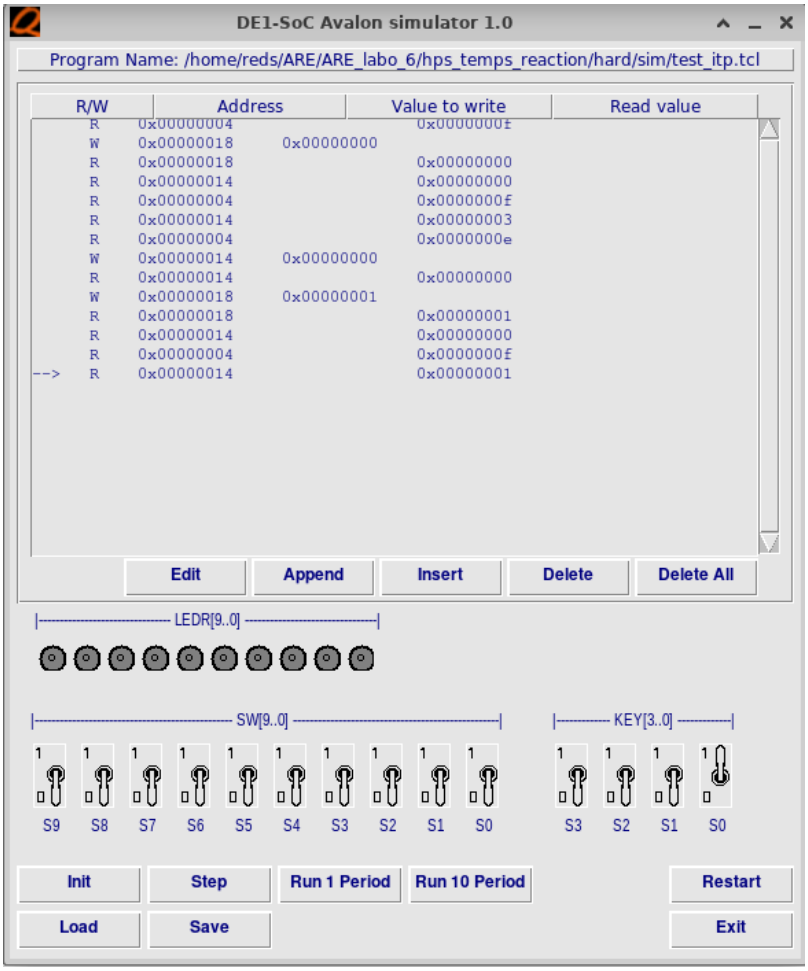

DE1-SoC Avalon simulator 1.0
rogram Name: /media/sf_REDS/are/ARE_labo_6/hps_temps_reaction/hard/script/test_counter.t
R/W
Address
Value to write
Read value
R 0x00000034 0x00000000
R 0x0000003c 0x00000000
W 0x0000002c 0x00000001
R 0x00000034 0x00000002
W 0x00000030 0x00000001
R 0x00000038 0x00000000
R 0x00000034 0x00000010
--> R 0x0000003c 0x00000001
Edit
Append
Insert
Delete
Delete All
LEDR[9..0]
SW[9..0]
KEY[3..0]
S9 S8 S7 S6 S5 S4 S3 S2 S1 S0
S3 S2 S1 S0
Init
Step
Run 1 Period
Run 10 Period
Restart
Load
Save
Exit

5.1.3 Interruptions

Pour les interruptions, on a fait un premier test pour vérifier que l'interruption se déclenche bien. On a changé la position du bouton 0 de 0 à 1. On voit bien que l'interruption se déclenche.

On peut bien voir qu'on n'a pas d'interruption à la première lecture du registre 0x14 et qu'on a une interruption à la deuxième lecture.

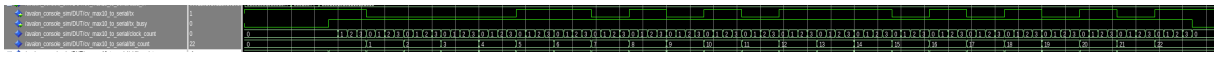
Dans la deuxième partie de la simulation, on a activé le masque d'interruption. On voit bien que l'interruption ne se déclenche pas, mais que le bit d'indication de flanc montant est à 1.



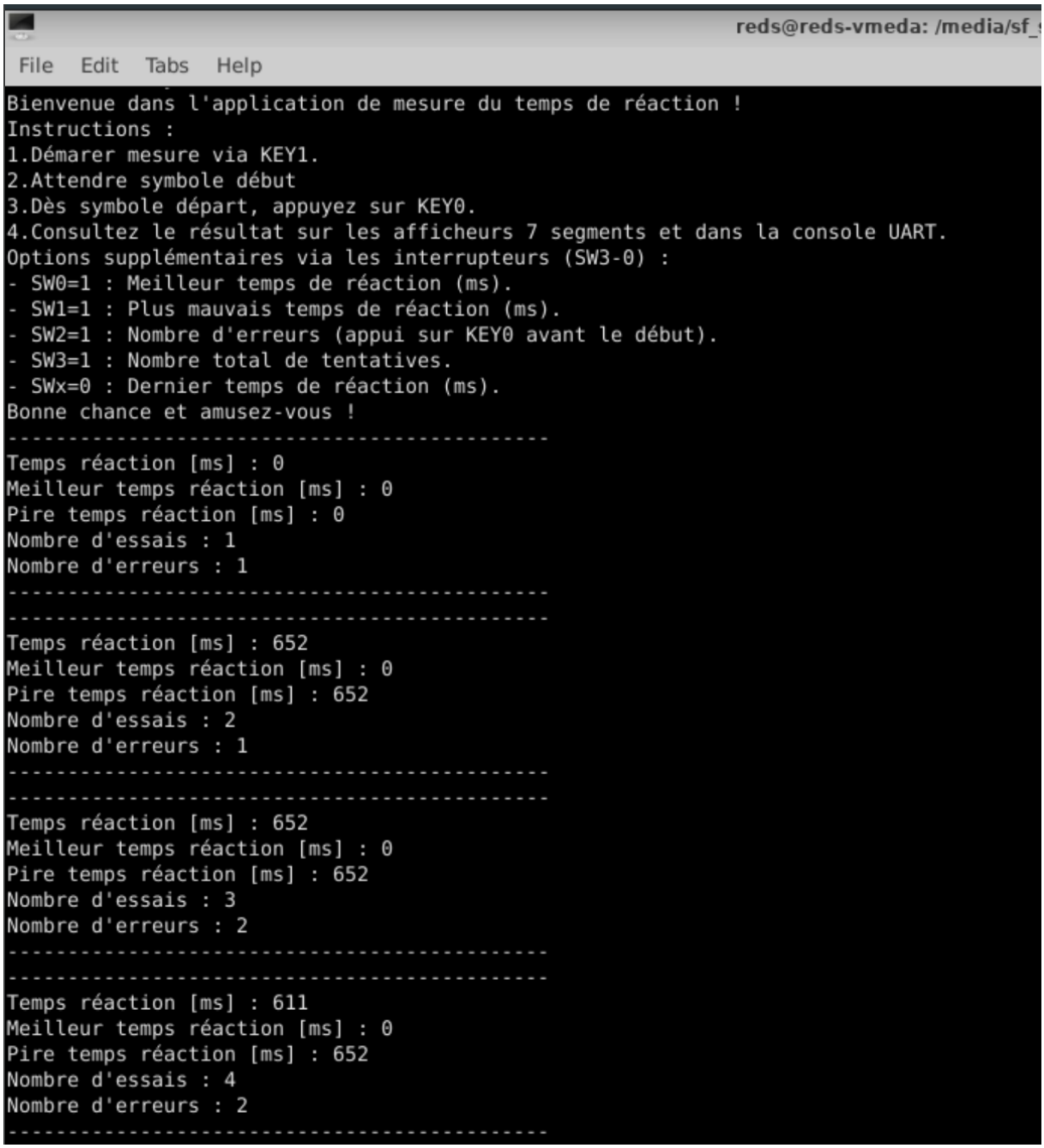
5.1.4 Test serial transmission to max10

Pour tester la transmission série vers le max10, on a fait un test avec une valeur de 0xabaa et un chip select de 0x01. Cela nous donne une data de 0x01abaa ou bien 0b0001_1010_1011_1010_1010.

On a aussi changé la vitesse de la baudrate pour avoir une pulse tous les 4 cycles.



5.2 Visuel via UART (Programme)



6 Conclusion

Le projet a permis de développer une application fonctionnelle respectant les exigences du cahier des charges, malgré des défis rencontrés tout au long de sa réalisation. Voici une synthèse des principaux points :

6.1 Points marquants et réussites

- Conformité au cahier des charges : Toutes les fonctionnalités spécifiées, comme la gestion des LEDs, les afficheurs 7 segments, la communication série avec la MAX10 et la mesure du temps de réaction, ont été implémentées et validées.
- Code C complet et fonctionnel : La logique de l'application est robuste et a permis de gérer les différents états du système tout en intégrant efficacement les interruptions, les entrées utilisateur et les affichages.
- Gestion des aléas : Un problème d'allumage aléatoire des LEDs de la MAX10 a été corrigé grâce à un "patch" introduisant une attente de 1 ms entre chaque envoi, garantissant ainsi un comportement stable.
- Approche modulaire : La séparation claire entre le VHDL et le C a facilité l'intégration et le dépannage des différentes parties.

6.2 Défis rencontrés et axes d'amélioration

- Communication série initiale : Une première conception imprécise pour la communication série avec la MAX10 a entraîné des retards importants (2 à 3 semaines) en raison d'une phase de débogage prolongée. Cela a limité le temps disponible pour perfectionner le projet.
- Problèmes liés au VHDL : Certains comportements inattendus, comme l'interruption déclenchée au démarrage ou le problème d'allumage des LEDs, semblent liés à des lacunes dans notre conception VHDL. Une version non finalisée du fichier avl_user_interface_not_finished.vhdl est incluse, illustrant nos efforts pour améliorer cette partie, bien que non testée faute de temps.
- Pression temporelle : Les dernières semaines ont été marquées par un rythme intense, limitant nos capacités à itérer sur certaines solutions et à approfondir les tests.

6.3 Bilan global

Malgré ces défis, nous sommes fiers de présenter une solution fonctionnelle. Bien qu'elle ne soit pas exempte de limitations, elle répond aux attentes principales du laboratoire et démontre notre capacité à surmonter des obstacles complexes dans un délai restreint. Ce projet nous a permis de renforcer nos compétences en conception embarquée, en gestion des interruptions, et en communication série.

Date: 27.01.25
Noms des étudiants: *Rodrigo Lopez Dos Santos, Urs Behrmann*

7 Annexe