Laboratoire 6: Conception d'une table tournante

Département : **TIC**

Unité d'enseignement : **ARE**

Auteur(s) :

- **PILLONEL Bastien**
- **JALUBE Miguel**

Professeur:

- **MESSERLI Etienne**

Assistant:

- **CONVERS Anthony**

Date:

- **Janvier 2024**

Sommaire

- Introduction
- Conception de l'interface
  - Plan d'adressage
  - Canal d'écriture
  - Canal de lecture
  - Générateur de top
  - Diviseur de frequence
  - Masquage des IRQ et commande moteur
  - Machines d'état
    - Machine d'état pour la calibration/initialisation
    - Machine d'état pour le déplacement
    - Machine d'état pour les IRQ
  - Synthèse
- Conclusion

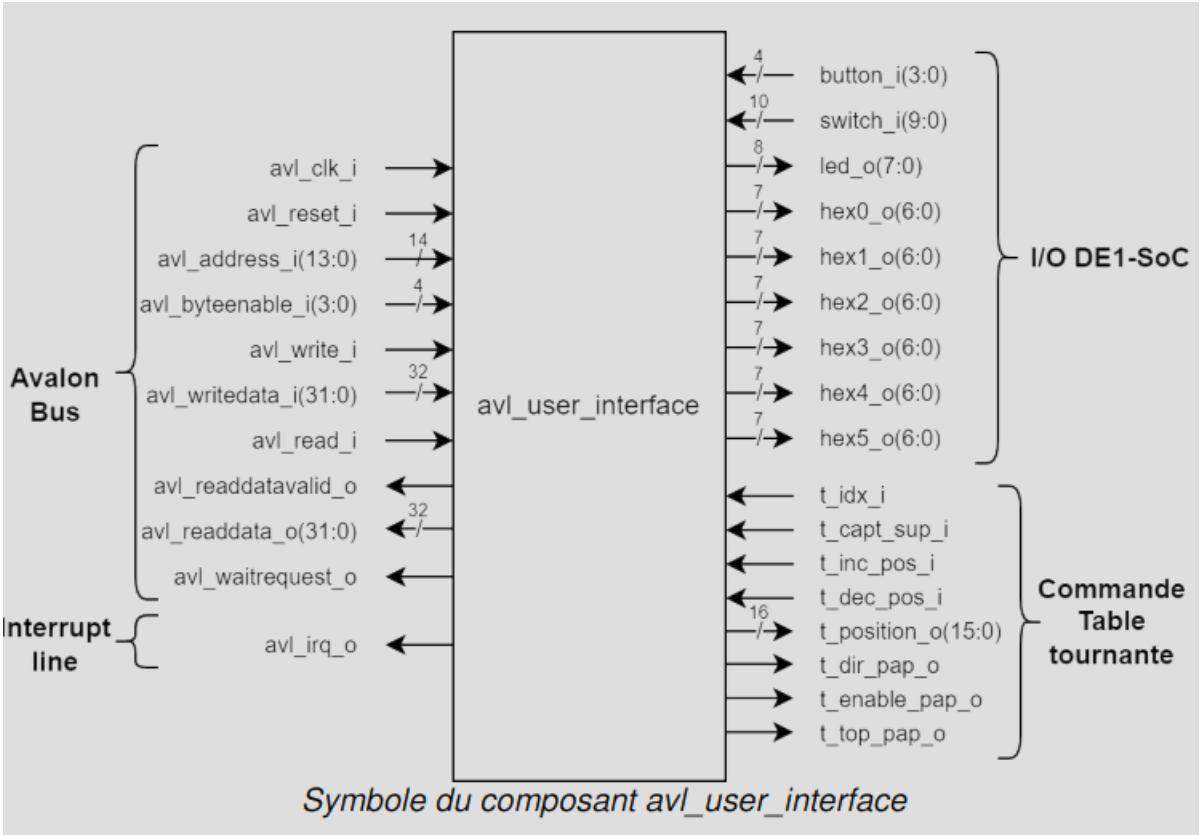
Introduction

Ce laboratoire a pour but de concevoir une application de commande d'une table tournante avec l'utilisation de plusieurs afficheurs externe. Le laboratoire comprendra plusieurs étapes :

- Mise en œuvre d'une interface UART du HPS, afin d'envoyer des messages dans un terminal distant.
- Développement et interfaçage des commandes de la table tournante.
- Développement et interfaçage des capteurs de la table tournante avec compteur de position.
- Implémentation d'un générateur de top pour le moteur afin de pouvoir sélectionner 4 vitesses différentes dans la FPGA.
- Mise en œuvre des interruptions du HPS générées par un dépassement de position de la table depuis la partie FPGA.

Conception de l'interface

Schema bloc



Symbole du composant avl_user_interface

Plan d'adressage

Même fonctionnement global qu'aux précédents laboratoire :

Offset on bus AXI lightweight HPS-to-FPGA (relative to BA_LW_AXI)	Fonctionnalités
0x00_0000 – 0x00_0003	Constante ID 32 bits (Read only)
0x00_0004 - 0x00_FFFF	reserved
0x01_0000 - 0x01_00FF	Zone disponible pour votre interface dans la FPGA
0x01_0100 - 0x01_FFFF	Zone disponible pour d'autres interfaces développées dans la FPGA
0x02_0000 - 0x1F_FFFF	not used

L'adresse de base du bus AXI est 0xFF20_0000 et de ce fait l'adresse de la zone mise à notre disposition pour l'interface est celle de 0xFF21_0000 à 0xFF21_00FF.

Les offsets des différents registres sont définis comme suit :

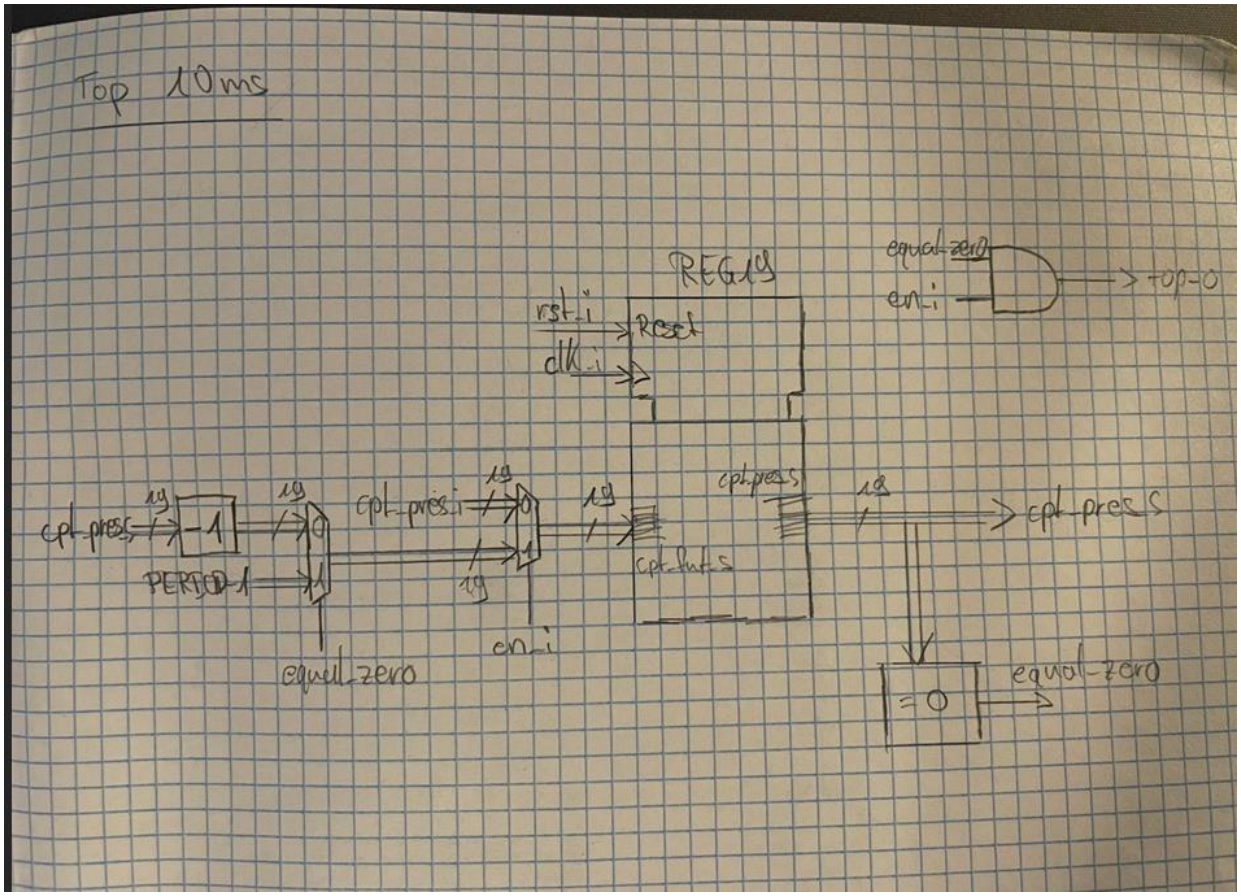
Plan d'adressage		
Adresse (offset)	READ	WRITE
0x00	[31..0] Constante ID interface	not used
0x04	[31..4] "0..0" ; [3..0] buttons	not used
0x08	[31..10] "0..0" ; [9..0] switchs	not used
0x0C	[31..8] "0..0" ; [7..0] leds	[31..8] reserved ; [7..0] leds
0x10	[31..28] "0..0" ; [27-21] hex3 ; [20-14] hex2 ; [13-7] hex1 ; [6-0] hex0	[31..28] reserved ; [27-21] hex3 ; [20-14] hex2 ; [13-7] hex1 ; [6-0] hex0
0x14	[31..14] "0..0" ; [13-7] hex5 ; [6-0] hex4	[31..14] reserved ; [13-7] hex5 ; [6-0] hex4
0x18	[31..16] "0..0" ; [15-0] val_pos;	[31..16] reserved ; [15-0] val_pos;
0x1C	[31..4] "0..0" ; [3-2] speed_s; [1] dir_s; [0] en_pap_s	[31..4] reserved ; [3-2] speed_s; [1] dir_s; [0] en_pap_s
0x20	[31..1] "0..0" ; [0] cal_busy_s;	[31..1] reserved; [0] run_cal_init_s;
0x24	[31..16] "0..0" ; [15-0] pos_end_s;	[31..16] reserved ; [15-0] pos_end_s;
0x28	[31..1] "0..0" ; [0] move_busy_s;	[31..1] reserved; [0] move_busy_s;
0x2C	[31..2] "0..0" ; [1] limit_max_s; [0] limit_min_s;	[31..2] reserved ; [1] mask_irq_s; [0] ack_s;

Précision sur les registres ajoutés :

- 0x18 : lecture et écriture -> position courante
- 0x1c : lecture et écriture -> bits 3…2 : vitesse moteur, bit 1 : direction moteur, bit 0 : enable moteur
- 0x20 : lecture -> bit 0 : calibration/initialisation en cours, écriture -> bit 0 : lancer calibration/initialisation
- 0x24 : lecture et écriture -> bit 0 : position cible du déplacement automatique
- 0x28 : lecture et écriture -> déplacement automatique est en cours
- 0x2c : lecture -> bit 1 : limite maximale atteinte, bit 0 : limite minimale atteinte, écriture -> bit 1 : masquage des irq (1 l'irq est masquée), bit 0 : acknowledge irq (1 on acquitte l'irq)

Générateur de top

Le premier composant que nous avons designé est le générateur de top 10ms. Nous avons choisi de créer un composant générateur de top générique => produit une pulse d'un cycle sur une période donnée (paramètre générique). Le concept est repris du laboratoire pwm réaliser en CSN.

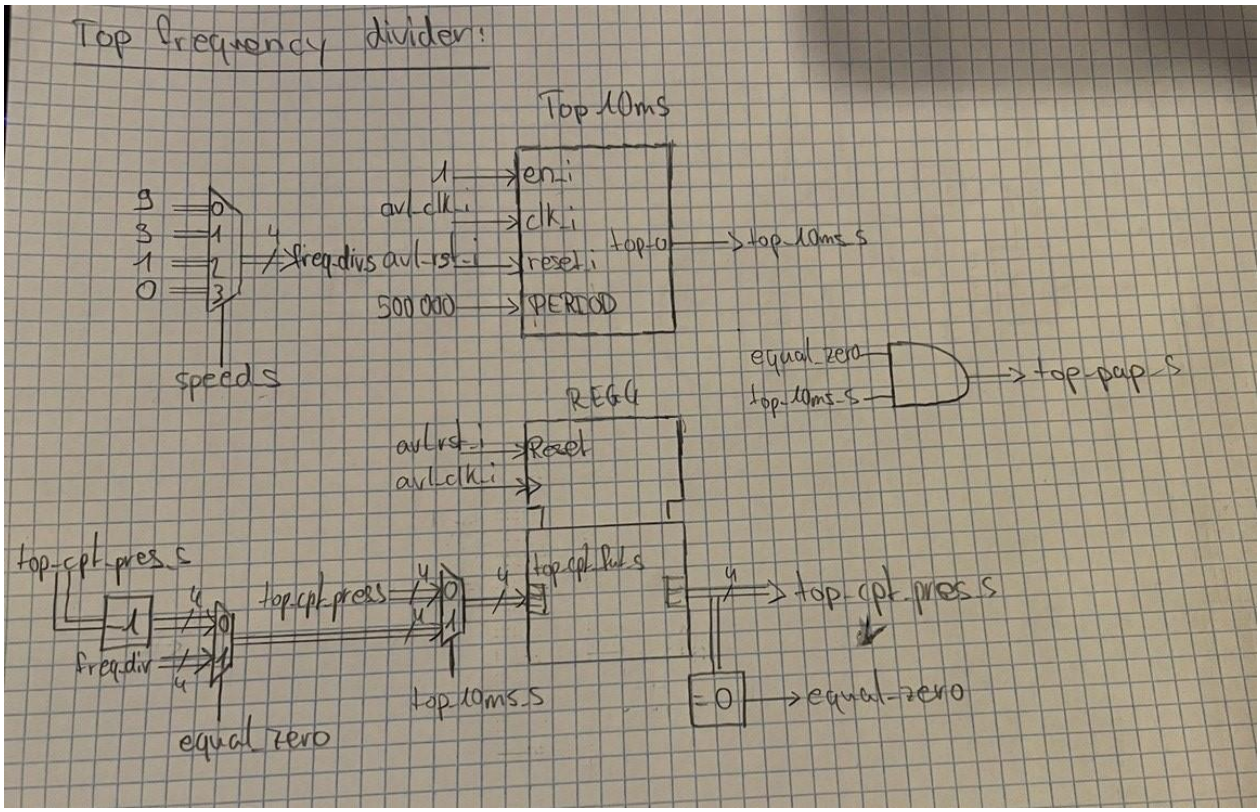


Ce composant sera ensuite instancié avec une période de 10ms (période la plus petite demandée). Nous pourrons diviser cette fréquence afin d'obtenir les trois autres demandées.

Diviseur de fréquence

Comme dis précédemment, nous devons diviser cette fréquence afin d'obtenir les périodes 100, 40 et 20ms demandées pour les différentes vitesses du moteur.

Afin de diviser la fréquence nous réalisons un simple compteur dont la valeur change en fonction de la vitesse demandée.



Afin de vérifier que le générateur nous sort les bonnes fréquences nous pouvons mesurer le signal top sur la borne inférieure de la led9 de la DE1-SoC (Attention les fréquences sont doublé car c'est la pulse du top qui toggle l'état du signal à la sortie du composant cmd_table_tournante)

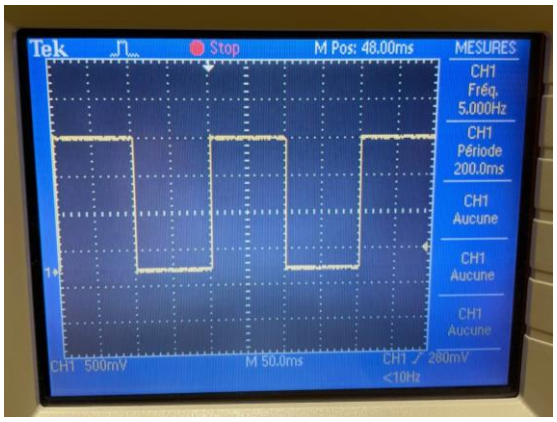


Figure 1Vitesse lente

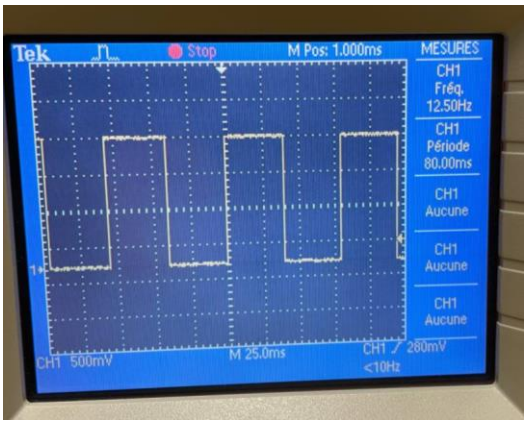


Figure 2Vitesse moyenne

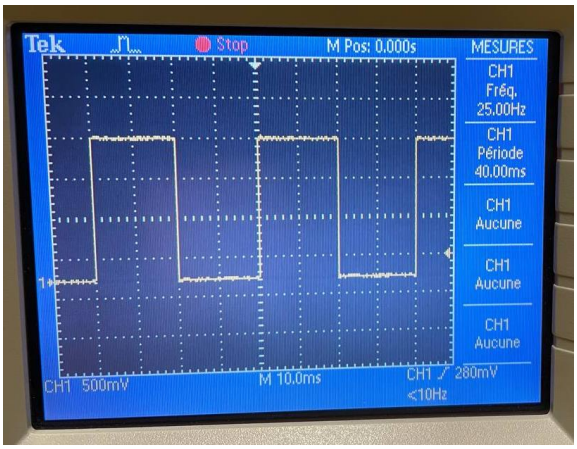


Figure 3Vitesse grande

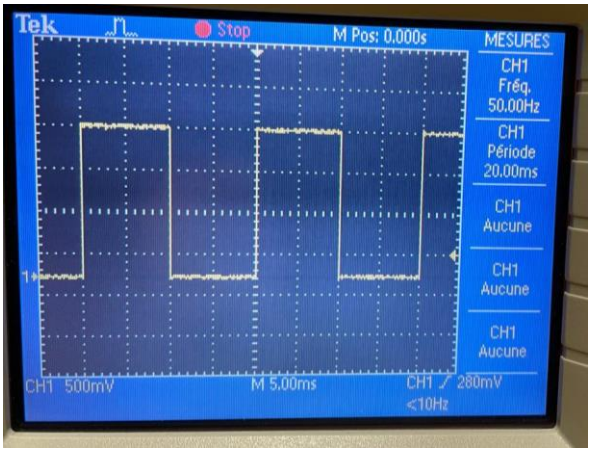
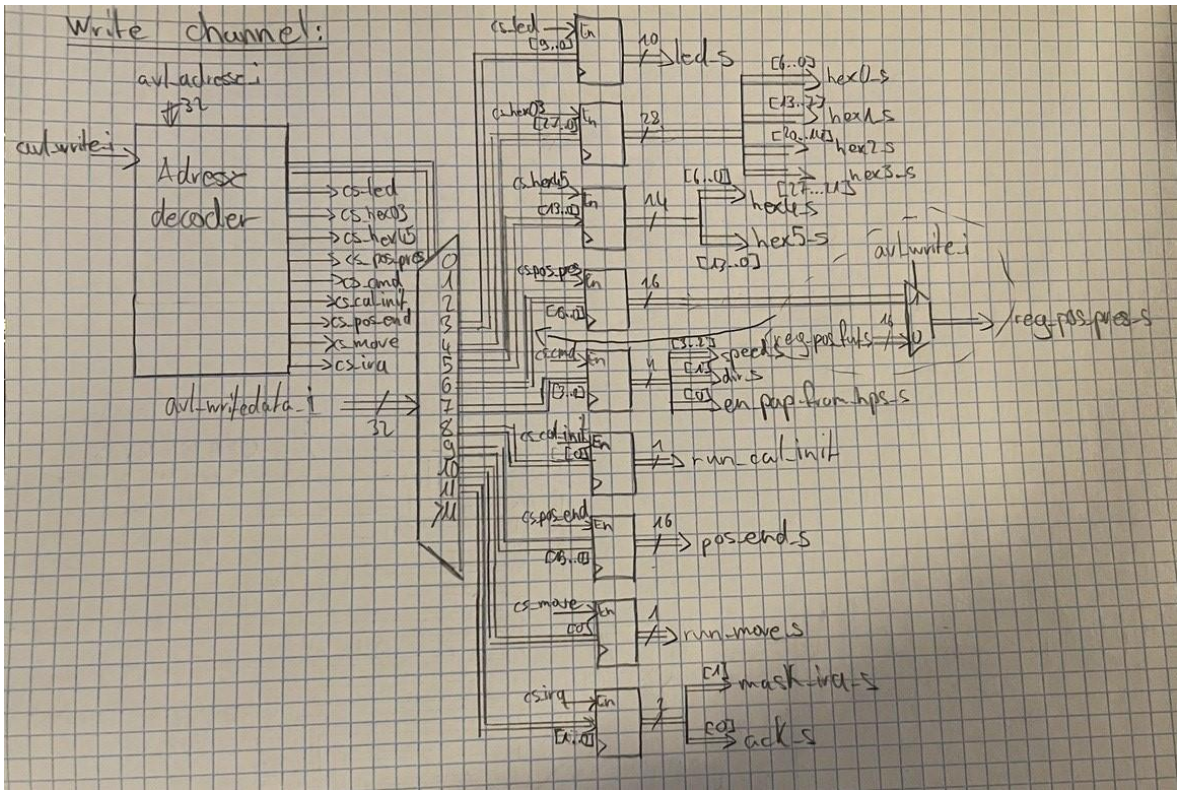


Figure 4Vitesse très grande

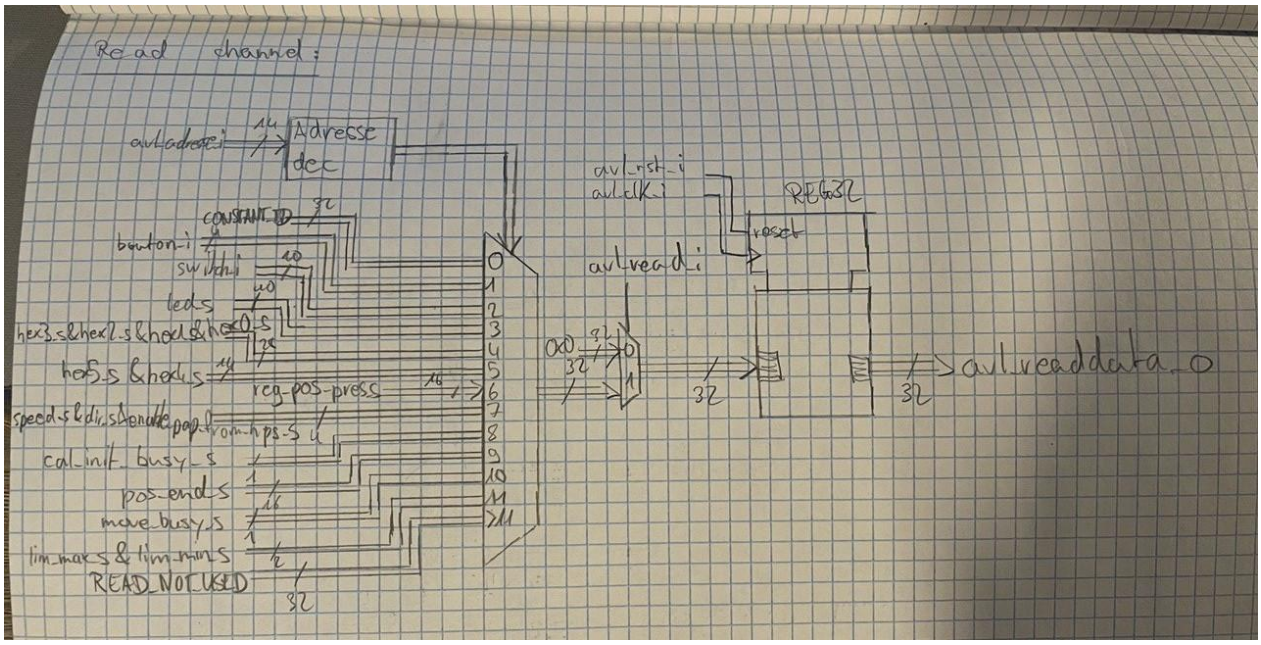
Canal d'écriture



Ici seuls les bits nécessaires du bus de donnée sont enregistrés.

Attention, pour le registre de position. Le multiplexeur se trouve avant le registre et permet de choisir si l'on écrit la position demandée par le hps (avl_write_i = 1) ou si c'est le décodeur d'état future du registre qui écrit la nouvelle valeur (le registre est aussi mis à jour par la table tournante).

Canal de lecture



Une constante de debug est placée lorsque l'on essaie de lire un offset mémoire qui n'est pas attribuer dans notre plan d'adressage => 0xDDEEAADD.

Machine d'état et logique de la table tournante

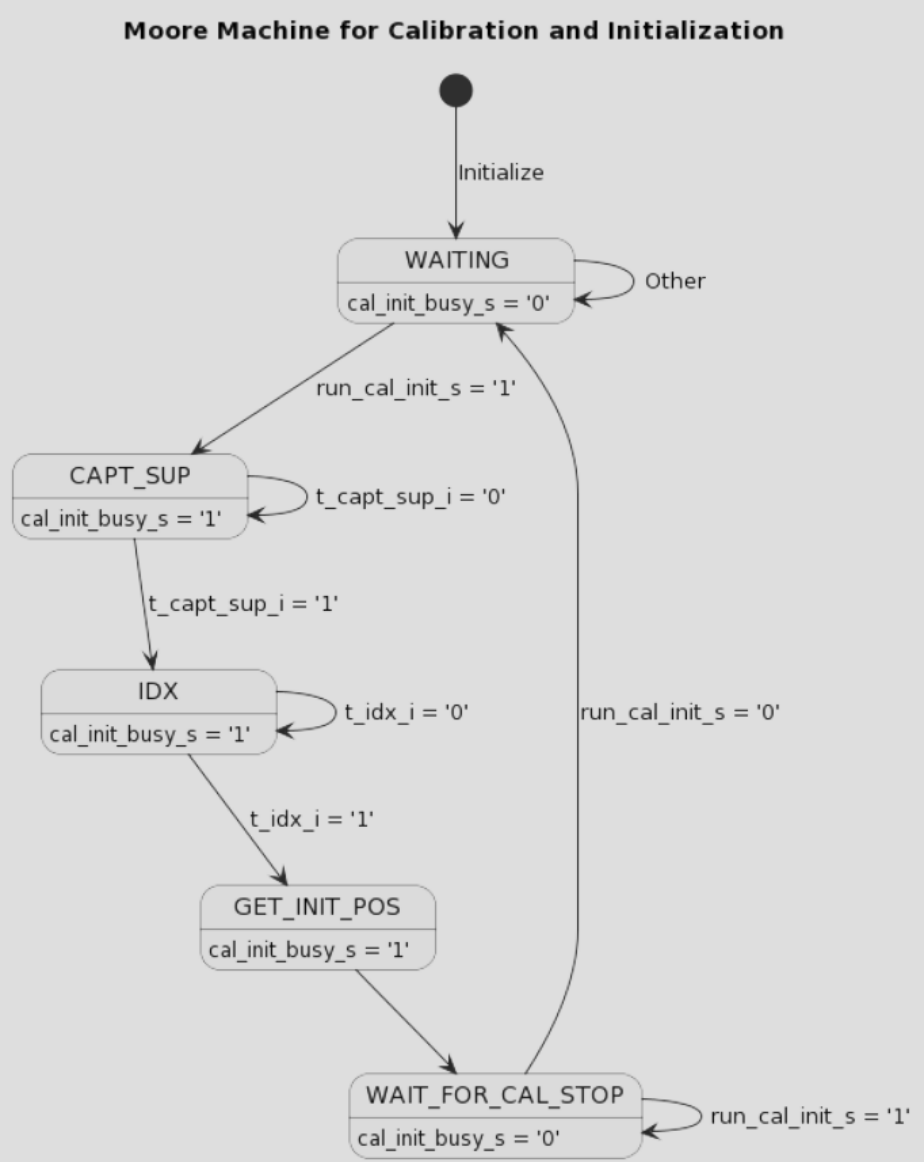
Nous avons décidé d'utiliser 3 machines d'état afin de gérer les actions suivantes :

- Séquence de calibration et d'initialisation
- Mouvement automatique
- Interruption lors d'un dépassement de limite

Séquence de calibration et d'initialisation

Nous avons remarqué que pour la calibration et l'initialisation, la phase pour aller jusqu'à l'index suivant le capteur supérieur était commune. Par soucis de rapidité et précision, nous préférons implémenté l'arrêt du moteur sur l'index dans l'interface. La prise de position pourra se faire dans le soft.

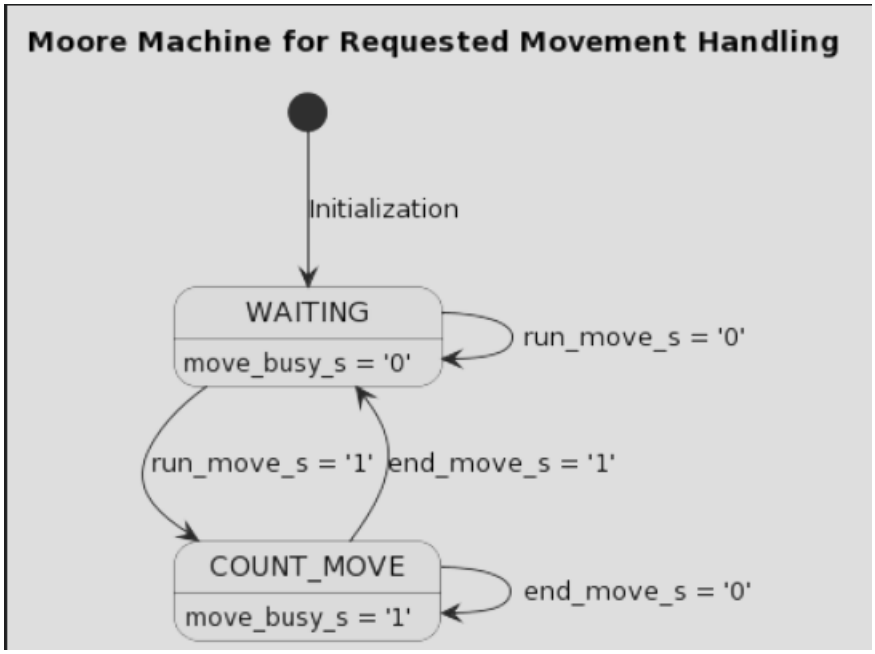
Logique de la séquence :



Mouvement automatique

Le mouvement automatique est demandé comme fonctionnalité à part entière mais elle pourra aussi être réutilisé pour finir la phase de calibration ou d'initialisation lorsqu'il est demandé de revenir jusqu'à la flèche.

Logique de la séquence

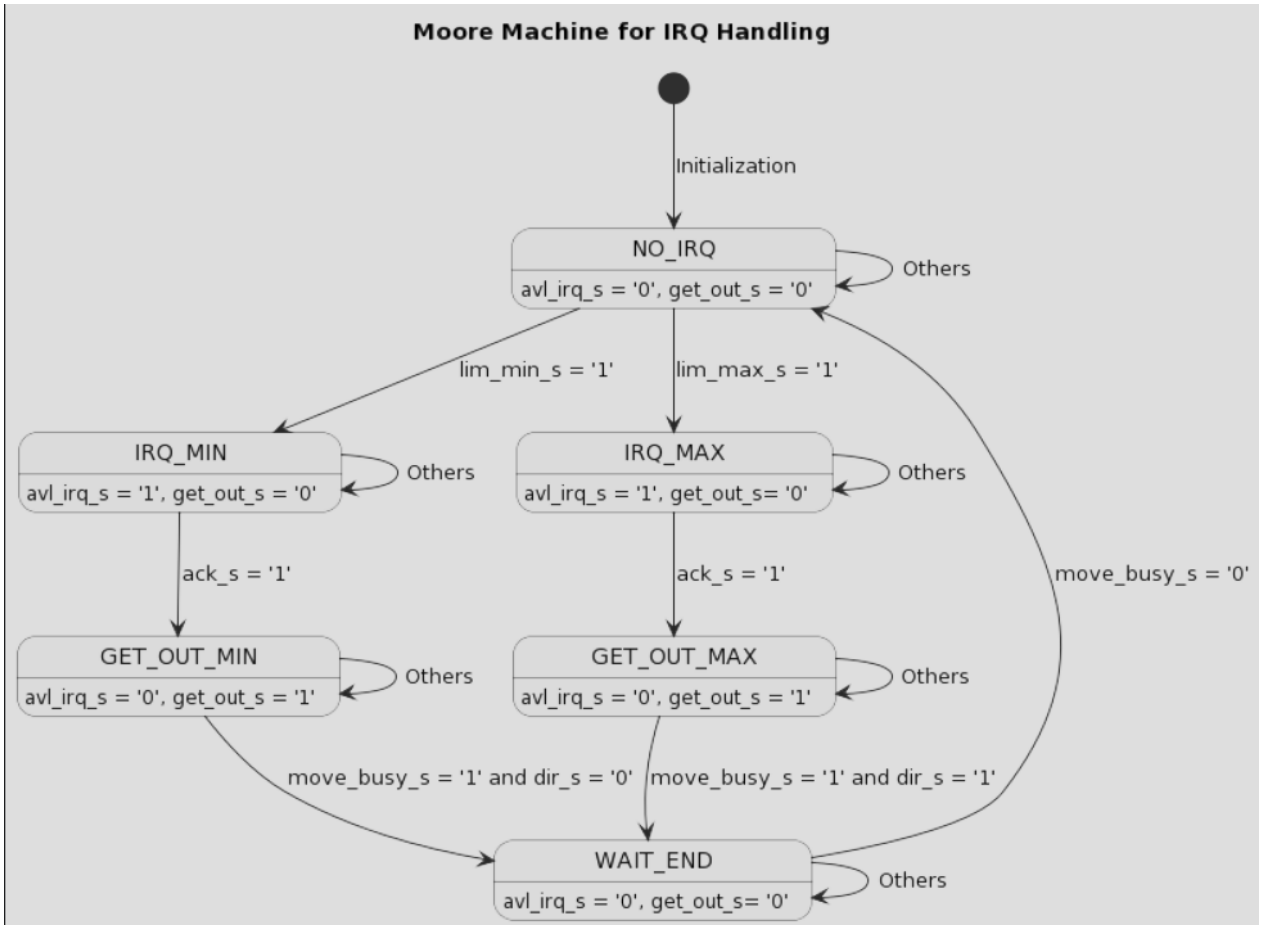


A la base nous voulions que l'utilisateur spécifie un déplacement pour le mouvement automatique mais notre logique ne fonctionnait pas et nous voulions garder une interface simple. L'utilisateur écrit donc une position cible depuis le hps (solution moins fiable si position demandée mal calculé)

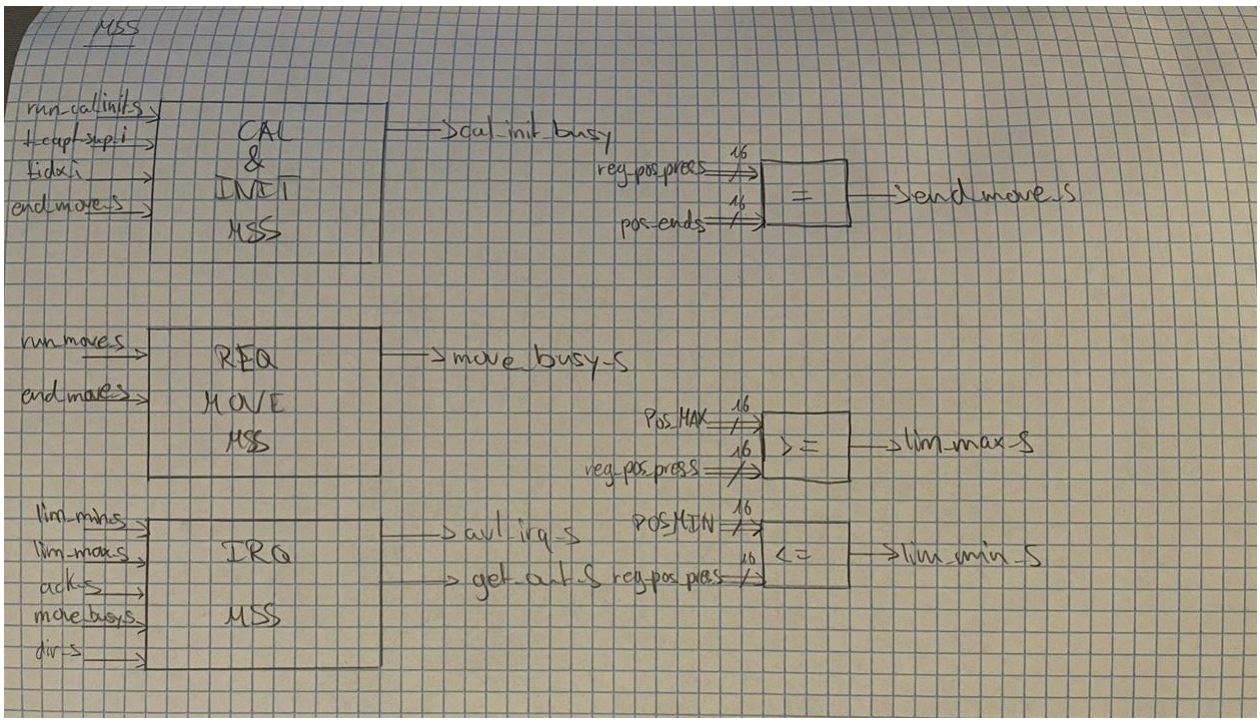
Interruption lors d'un dépassement de limite

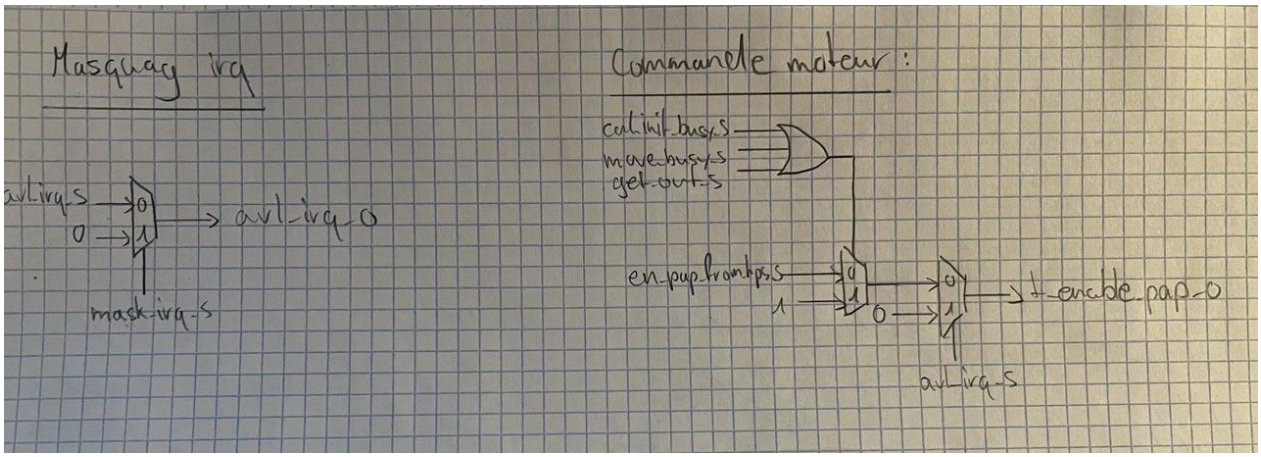
Avl_irq_s permet d'indiquer une interruption sur la ligne avl_irq0 connecté au GIC de notre hps. Permettra au hps de savoir qu'un dépassement des limites s'est produit. Le hps pourra ensuite interroger l'interface sur la limite problématique et acquitter l'irq afin de baisser la ligne irq depuis l'interface. La sortie get_out permet d'indiquer que le moteur doit être activer afin de sortir de la zone de dépassement.

Logique de la séquence :



Schémas associés à la logique des MSS





Séparation des actions HPS/AVL_INTERFACE

Lors de la phase de conception, notre but était de laisser un maximum la main au hps pour la gestion de la logique compliquée.

Le gros point critique étant celui de stopper le moteur à temps lors de certains évènements :

- Arrivé à l'index lors de la calibration
- Arrivé à l'index lors de la calibration
- Arrivé à la position cible lors d'un mouvement automatique
- Dépassement d'une limite

Ces événements sont traités dans l'interface car la latence hps/interface pourrait introduire une imprécision ou même un comportement imprévisible/dangereux si l'interface venait à ne plus répondre sur le bus avalon.

Il est toutefois nécessaire de tenir au courant de l'état de l'interface au hps afin que celui-ci puisse prendre les actions nécessaires. Nous faisons donc du polling sur certains flags dans le code C afin de passer à l'action suivante.

UART

L'UART requiert certains paramètres pour fonctionner correctement. Voici les paramètres utilisés pour ce laboratoire :

1. Baudrate : Le baudrate doit être défini à 9600. Pour calculer les valeurs de Divisor Latch Low (DLL) et Divisor Latch High (DLH), vous pouvez utiliser la formule fournie dans la documentation :

$$\text{baud rate} = \frac{\text{serial clock frequency}}{16 \times \text{divisor}}$$

Puisque l'horloge l4_sp_clk est à 100 MHz, le diviseur à utiliser pour obtenir un baudrate de 9600 est :

$$\text{divisor} = \frac{100{,}000{,}000}{16 \times 9600} \approx 651$$

2. Bit de données : Pour définir le nombre de bits de données à 8, vous devrez vous assurer que le bit DLAB (Divisor Latch Access Bit) du registre LCR (Line Control Register) est réglé sur 0 pour désactiver l'accès aux registres du diviseur. Ensuite, configurez les bits de données du registre LCR pour 8 bits.

3. Bit de parité désactivé: Dans le registre LCR, assurez-vous que les bits de parité sont désactivés, ce qui est généralement le réglage par défaut.

4. Bit de stop: Configurez le registre LCR pour un seul bit de stop. Activer les buffers FIFO en émission et réception: Activez les FIFO en écrivant dans le registre FCR (FIFO Control Register).

Conclusion

En conclusion de ce projet de conception d'une table tournante avec une interface avl_user_interface, les objectifs ont été atteints. La mise en œuvre de l'interface UART0 du HPS a été réalisée avec succès, en s'appuyant sur la documentation du Cyclone V.

Le plan d'adressage de l'interface avl_user_interface a été modifié progressivement tout le long du laboratoire. La partie initialisation/calibration ont été les seules fonctionnalités complexes décrites en VHDL plutôt que codées en C. Malgré la préférence d'intégrer le code complexe en soft plutôt qu'en hard, la latence du bus ne permet pas d'atteindre la précision souhaitée si codée côté soft.

Le générateur de top a été intégré avec succès dans l'interface, et la fréquence du signal top a été vérifiée à l'oscilloscope pour respecter la contrainte de ne jamais dépasser 100 Hz.

L'unité de commande de la table tournante a été définie et conçue avec une interface semblable aux laboratoires précédents.

Les interruptions, gérées par un gestionnaire d'interruption conforme aux spécifications, ont été implémentées avec succès. La routine d'interruption fpga_ISR() a été testée avec le toggle de la Led7 de la DE1-SoC.