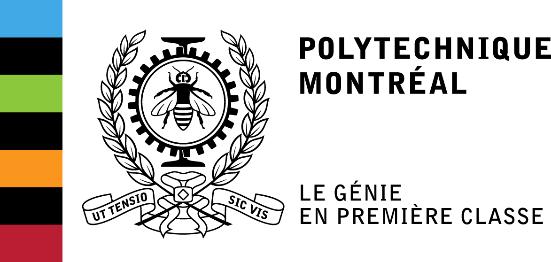
****

**INF3610 – Systèmes embarqués**

**Rapport du laboratoire 2  
Simulateur d’une tour de contrôle d’aéroport**

**Groupe 02**

**Simon Nadeau – 1791314  
Mathieu Châteauvert – 1846732**

**Travail présenté à  
Eva Terriault**

**Rapport remis le 25 octobre 2018  
École Polytechnique de Montréal**

Question 1

*Lors de l’implémentation matérielle, vous avez instancié un module GPIO, 2 FIT Timers et un contrôleur AXI. Donner une brève définition et une explication du rôle de chaque module dans ce laboratoire.*

Un module GPIO (*General Purpose Input/Ouput*) est un ensemble de port permettant de communiquer avec des composantes électroniques externes comme des capteurs, des interrupteurs, etc. Dans ce laboratoire, le module GPIO est utilisé pour capter les signaux provenant d’un changement d’état d’une interruption sur la carte Zynq.

Un FIT (*Fixed Interval Timer*) est une composante permettant de générer un signal à intervalles fixes. Dans ce laboratoire, deux FITs sont utilisés : un générant un signal à chaque seconde (pour la génération d’avion) et l’autre générant un signal aux trois secondes (pour la vérification des files d’attente d’atterrissage).

Un contrôleur d’interruption AXI est un contrôleur matériel permettant de gérer les interruptions. Son rôle n’est pas d’interrompre le processeur, mais d’activer les routines d’interruption et de s’assurer que les interruptions ont bien été servies. Dans ce laboratoires, le contrôleur AXI permet de multiplexer les trois signaux d’interruption provenant des deux FITs et du GPIO.

Question 2

*Combien de périphériques (minuteries, switches, etc.) aux maximum votre design pourrait-il supporter?*

Selon la documentation du contrôleur d’interruption AXI[[1]](#footnote-1), le nombre d’entrée d’interruptions maximum est de 32, donc notre design pourrait supporter un maximum de 32 périphériques.

Question 3

*Au cours no 4 (pages 23 à 25), on a présenté le déroulement générique des interruptions uC, alors qu’au cours no 5 (pp. 25 à 34) on a présenté la gestion des interruptions propres au Zynq. À partir de cette information et des fichiers du portage du lab 2 (e.g. bsp\_init.c et os\_cpu\_a.S) ainsi que votre code d'application (simAvion.c) décrivez de manière la plus précise possible le déroulement complet (différentes étapes) d'une interruption provenant de fit timer 0 jusqu’à son service ISR.*

En premier, le signal d’interruption provenant du FIT arrive au Contrôleur d’interruption AXI. Ensuite, le General Interrupt Controller (GIC) s’occupe d’interrompre le processeur. Puis, le processeur enregistre le contexte en sauvegardant tous les registres. Il fait appel à OSIntEnter() qui permet d’entrer dans la routine d’interruption. Une fois entré, il exécute le code de l’interruption reliée au FIT (génération d’avion). Après, il ressort de l’ISR en appelant OSIntExit() et restaure les registres précédemment sauvegardés avant de sortir de l’interruption.

Aspects concernant l’implémentation

Premièrement, pour ce qui est de la priorité des tâches et des mutex, nous n’aurions pu arriver à un résultat satisfaisant sans modifier la priorité de ceux-ci. Le code nous retournait des erreurs mentionnant que les tâches utilisant des mutex possédaient une priorité supérieur au mutex lui-même. Nous avons donc dû décaler les priorités de toutes les tâches vers le bas afin de pouvoir insérer les mutex de manière plus prioritaire. Nous n’avons pas modifier l’ordre de priorité des tâches.

Deuxièmement, avec un délai au terminal de 160, le terminal 1 a pratiquement toujours le temps de redevenir libre avant la fin du prochain atterrissage. C’est pourquoi il est rare que le terminal 2 accueille des avions. En augmentant le délai d’attente (360 par exemple) au terminal, nous avons pu confirmer que le terminal 2 peut recevoir des avions lui aussi.

Appréciation générale du laboratoire

Le sujet de ce laboratoire était intéressant et permettait de mettre convenablement en application les concepts vus en classes. Sa complexité était suffisante. Nous y avons consacré environ 22 heures (2h Vivado, 18h code, 2h rapports ). Nous avons trouvé que le fichier détaillant la création plateforme sur Vivado était très bien expliquée et facile à suivre.

1. <https://www.xilinx.com/support/documentation/ip_documentation/axi_intc/v4_1/pg099-axi-intc.pdf> (page 10-11). Consulté le 20 octobre 2018. [↑](#footnote-ref-1)