INF3610 - TP2

Présenté à

Jeff Falcon, chargé de laboratoire

Par

Michael Chidiac,

Olivier Cotte, 1538493

École Polytechnique de Montréal

30 octobre 2013

**Barème correction**

**EXÉCUTION**

 Fonctionnement sur carte FPGA /4

**TOTAL EXÉCUTION** /4

**CODE SOURCE**

 Contenu des tâches /3

 Mécanismes de communication et de synchronisation /1

 Respect de l’énoncé, commentaire et clarté du code /1

 Fonctionnement des fit\_timer /1

**TOTAL RAPPORT** /6

**RAPPORT**

 Présentation générale, introduction, conclusion /1

 Fonctionnement du système /2

**QUESTIONS**

 Question 1 /1

 Question 2 /1

 Question 3 /1

 Question 4 /1

 Question 5 /1

 Question 6 /1

**TOTAL RAPPORT** /10

**NOTE GLOBALE /20**

INTRODUCTION

Afin de nous familiariser avec les systèmes a temps réel, ce second laboratoire vise à nous faire développer un système embarqué sur un processeur μBlaze sur les cartes FPGA du laboratoire. Nous allons devoir utiliser le RTOS μC.

Nous allons concevoir un système qui simule l’échange de paquets informatiques à travers plusieurs routeurs. Ces paquets vont devoir se rendre à la bonne destination indiquée dans leur structure en passant par les bons routeurs sur le chemin. Chaque routeur doit regarder la destination finale du paquet et déterminer le prochain routeur où envoyer le paquet. Si un routeur reçoit un paquet qui n’est pas dans sa plage d’adresse à traiter, le paquet est alors rejeté. Le routeur doit aussi vérifier la validité du paquet grâce à un calcul CRC. La troisième et dernière fonction de nos routeurs sera de trier les paquets selon leur type, c’est-à-dire des paquets audio, vidéo ou autre.

FONCTIONNEMENT DU SYSTÈME

Toutes les tâches sont initialiser et créer dans le main qui appelle une fonction \_CreateTask(). Nous avons aussi la création de trois Semaphore : SemPrint, SemStop et SemVerification; cinq queues: ptrFifoIn, ptrFifoVideo, ptrFifoAudio, ptrFifoOtherwise et ptrFifoAuxilary; et finalement 4 mailbox: Int1Mbox, Int2Mbox, Int3Mbox, Int4Mbox. La tâche main initie le BSP et commence le multitasking.

Chaque paquet est généré à l’intérieur d’une tâche nommée TaskInjectPacket. Cette tâche crée des structures de type Paquets définis ainsi :

typedef struct {

unsigned int src;

unsigned int dst;

unsigned int type;

unsigned int crc;

unsigned int data[12];

}Packet;

Chacun des unsigned int de la structure représente les 4 octets définis dans la structure d’un vrai paquet. Seul le unsigned data représente 48 octets de données aléatoirement généré par la tâche. Le CRC est calculé en dernier par la tâche.

Ils sont ensuite introduits dans la FifoIn. Si la fifo est pleine, le paquet est rejeté. Un OSTimeDly(50) est ensuite effectué afin de ne produire que 2 paquets par seconde.

Le paquet est alors recueillis par de la FifoIn par la tâche TaskComputing. Sa première fonction est de rejeter les paquets qui ne sont pas dans son espace d’adresse en vérifiant les bordure REJECT\_LOW et REJECT\_HIGH de 1 à 4.

Si la source se retrouve bel et bien dans l’espace d’adressage, on vérifie la validité CRC. Pour ce faire, nous sauvegardons la valeur du CRC du paquet dans une variable temporaire puis nous réinitialisons la variable CRC du paquet a 0. Nous passons ensuite le paquet complet à la fonction computeCRC qui nous retourne le bon CRC. Nous comparons cette valeur de retour avec celle transmise avec le paquet. Si les valeurs sont égales, nous continuions le traitement, autrement nous rejetons le paquet. Il ne faut pas oublier de remettre la valeur du CRC dans le paquet.

TaskComputing met alors le paquet dans l’une des fifos selon le type du paquet, c’est-à-dire High, Medium ou Low. Les trois fifos respectives sont FifoVideo, FifoAudio et FifoOtherwise.

Le paquet TaskForwarding se reveille ensuite afin de lire à l’interieur des trois fifo, et ceci en ordre de priorité. Si le message reçu n’est pas vide, nous procédons à la vérification de l’interval de destination, comme dans la tâche TaskComputing. Nous déponsons par la suite le paquet dans la mailbox correspondant à sa destination. Si la boite est pleine alors le paquet est rejeté. Autrement, nous incrémentons la valeur à être affiché sur les LEDs de la planche FPGA.

TaskPrint à ensuite la tâche d’aller lire dans les mailbox et d’imprimer à l’écran l’information sur les paquets.

La tache TaskVerification pour sa part, vérifie si les fifos High, Medium et Low sont pleines et si elles le sont, augmente le nombre de paquets rejeté. Elle est réveillé grâce à une interruption faisant appel à fitTimer2Handler qui débloque la sémaphore SemVerification après laquelle TaskVerification attend.

La fonction fitTimerHandler est pour sa part appelé par une interruption. Elle sert a vérifié si le nombre de paquets rejeté est supérieur à 15 et si elle l’est, débloque le sémaphore SemStop qui lui-même débloque la tâche TaskStop. Cette dernière fait appelle à une fonction nommée BSP\_IntDisAll() qui éteint les interruptions ainsi qu’une fonction \_DeleteTask() qui détruit toutes les tâches en cours et ferme ainsi le routeur.

RÉPONSES AUX QUESTIONS

QUESTION 1

Le BSP sert à initialisé les drivers pour qu’ils soient en mesure de communiquer avec le système d’exploitation.

QUESTION 2

Aucune section critique n’est présente dans le code. Il ne peut donc pas y avoir d’inversion de priorité, car aucune tâche n’attend les ressources possédées par une autre. Les tâches sont synchronisées unilatéralement par des sémaphores binaires.

QUESTION 3

OSTimeDly() délai la tâche courant par le nombre de ticks passé en paramètre. Le temps de délai dépend donc du nombre de ticks par seconde du système. On peut trouver cette valeur en examinant OS\_TICKS\_PER\_SEC. Dans notre cas, OS\_TICKS\_PER\_SEC = 100. Donc pour attendre une demi-seconde on doit effectivement attendre 50 ticks.

QUESTION 4

Non, le tick de l’OS est générer par une timer hardware indépendant du processeur avec sa propre horloge. C’est une ISR de l’OS qui actualise la valeur du tick de l’OS à chaque fois que le timer lance une interruption.

Dans notre cas, la valeur de l’horloge du timer et du MicroBlaze sont différentes : l’horloge du timer est de 100Hz (100ticks/secondes) et l’horloge du MicroBlaze est cadencée à 100 MHz.

QUESTION 5

On réalise un appel bloquant à une fifo à l’intérieur de TaskVerification, ce que l’on n’aurait pas pu faire à l’intérieur d’une ISR.

QUESTION 6

Puisque l’on devait simuler la table de routage en ajoutant un délai d’une seconde, les paquets sont effectivement traités à un débit de 1 paquets/second. Peut-importe l’état des queues des paquets, on ne pourra jamais faire mieux puisque l’on devra toujours attendre au moins une second pour traiter les paquets.

La tâche TaskComputing ordonne les paquets par leur priorité en les plaçant dans trois files différentes. Alors il devrait y avoir au moins autant de tâche TaskFowarding qu’il y a de file de paquets.

CONCLUSION