Implémentation matérielle de l'interface de communication multiprocesseur de RTEMS

Clément Foucher, Fabrice Muller

Implémentation matérielle de l'interface de communication multiprocesseur de RTEMS

Clément FOUCHER, Fabrice MULLER
Université de Nice-Sophia Antipolis, LEAT-CNRS
foucher@polytech.unice.fr, Fabrice.Muller@unice.fr

1 INTRODUCTION

Le principe de tâche matérielle, disposant de son unité d'exécution dédiée, s'oppose à celui de tâche logicielle, qui s'exécute sur un processeur généraliste à temps partagé.

Une implémentation matérielle des tâches sur des zones reconfigurables permet d'une part une exécution en parallèle, et d'autre part une diminution du temps de traitement.

En s'appuyant sur un système d'exploitation software (SwOS) déjà existant, on se propose de réaliser un système d'exploitation matériel (HwOS) compatible avec celui-ci.

Le choix du SwOS s'est porté sur RTEMS[1] (Real-Time Operating System for Multiprocessor Systems), qui a l'avantage d'être un système open-source.

Ce choix a notamment été guidé par le mode de communication entre les diverses unités d'exécutions (nommées nodes).

Assurée par un ensemble de services regroupés sous le nom de MPCI (MultiProcessor Communication Interface), cette communication inter-nodes a l'avantage d'être transparente pour les tâches exécutées uniquement localement.

En effet, chaque node peut définir des objets (sémaphores, tâches, boites aux lettes, …) comme étant globaux, ce qui signifie que ceux-ci peuvent être manipulés par un autre node.

Le but du travail est de réaliser, en hardware, la partie communication du SwOS, de manière à ce que le HwOS soit capable de s'interfacer avec une ou plusieurs instances software de RTEMS.

On choisit comme unité d'exécution software le processeur LEON3, instancié autour d'un bus AHB, disponible également en open source au sein de la librairie GRLIB[2].

Ces travaux sont réalisés dans le cadre du projet ANR FOSFOR[3] (Flexible Operating System FOr Reconfigurable platform).

2 PRINCIPE DE LA COMMUNICATION

2.1 Structure de la mémoire partagée

L'étude du code source de RTEMS concernant le MPCI dévoile une communication basée sur des messages déposés en mémoire partagée.

En effet, les différents processeurs sont instanciés sur un bus commun auquel est également reliée la mémoire, une partie de celle-ci étant déclarée comme partagée. Chaque node y stocke alors les messages qu'il souhaite transmettre à un autre node, avant de prévenir le node destinataire de la présence d'un message par le biais d'une interruption, comme illustré sur la Figure 1.

Le stockage repose sur le principe d'enveloppes dans lesquelles sont déposés les messages avant d'être « postées » dans une file de réception propre au node destinataire.

La mémoire partagée est découpée en 3 zones distinctes, chacune ayant un rôle dans le processus de communication :

- La zone « Statut des nodes » permet de synchroniser l'initialisation des nodes. Il existe une structure pour chaque node, celles-ci contenant les informations nécessaires au déclenchement des interruptions ainsi que le statut du node.
- La zone « Files de réception » contient N+1 structures, N étant le nombre de nodes dans le système. Il s'agit d'en-têtes de listes chaînées, contenant notamment un pointeur vers la première enveloppe contenue dans la file. Il existe une file pour chaque node, représentant sa boite de réception des messages, ainsi qu'une file supplémentaire contenant les enveloppes non utilisées.
- La zone « Enveloppes » est la zone de stockage des messages à proprement parler. En plus du message lui-même, chaque enveloppe contient un pointeur vers l'enveloppe suivante afin d'implémenter le principe des listes chaînées.

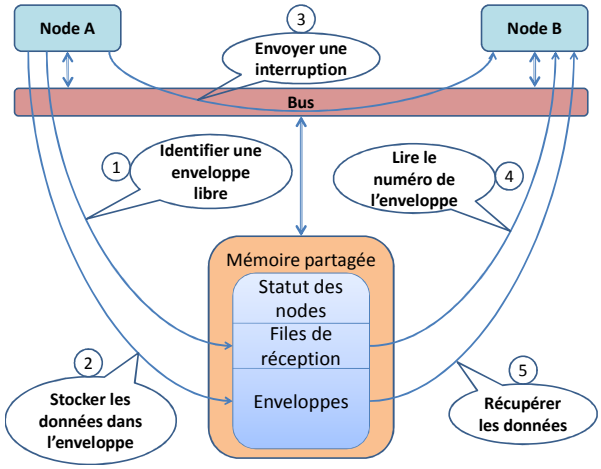

Figure 1 : Processus d'envoi des messages

2.2 Services software

La gestion de la communication software passe par 5 services permettant la gestion des messages :

- Un service d'initialisation, permettant de coordonner les nodes au démarrage par le biais de la zone « Statut des nodes ».
- Un service « get » effectuant la réservation d'une enveloppe, préalable nécessaire à l'envoi d'un message, en récupérant la première enveloppe disponible dans la liste des enveloppes libres.
- Un service « send » permettant d'inscrire une enveloppe dans la file du destinataire une fois le message stocké dans celle-ci, puis de déclencher l'interruption sur le node cible.
- Un service « receive » dont le rôle est d'aller récupérer le message contenu dans la première enveloppe référencée dans la file de réception locale.
- Un service « return », qui effectue l'inscription d'une enveloppe dans la file des enveloppes libres afin de la libérer une fois son contenu consommé.

Hormis l'initialisation, effectuée au reset, la gestion de la communication repose sur des processus. Le service de réception est débloqué par un processus en attente sur une interruption, tandis que les autres services sont appelés par les processus actifs.

3 REALISATION HARDWARE

3.1 Segmentation

En hardware, les étapes du processus devront être implémentées sous forme d'entités concurrentes et interagissant les unes avec les autres comme le montre la Figure 2.

Tout d'abord, il est nécessaire de définir un mode de communication entre le Hardware MPCI (HwMPCI) et les autres services du HwOS. Pour cela, nous allons utiliser une méthode similaire à celle des enveloppes en mémoire partagée, mais de manière locale :

Chaque service du HwOS dispose d'une mémoire tampon, segmentée en blocs de la taille d'un message, dans laquelle seront stockés les différents messages, reçus ou à émettre. Lors de la réception d'un message, et après l'avoir stocké dans la mémoire tampon, le HwMPCI avertit le service destinataire de la présence d'un message reçu à traiter en postant une donnée dans une file fonctionnant sur le mode FIFO. Cette donnée contiendra différents paramètres tels que le numéro du bloc dans lequel le message a été stocké. De la même manière, les services préviendront le HwMPCI de la présence d'un message à émettre en postant une donnée dans la file des tâches du HwMPCI.

Il est à noter que les files de réception des services du HwOS peuvent également être utilisées par les tâches locales.

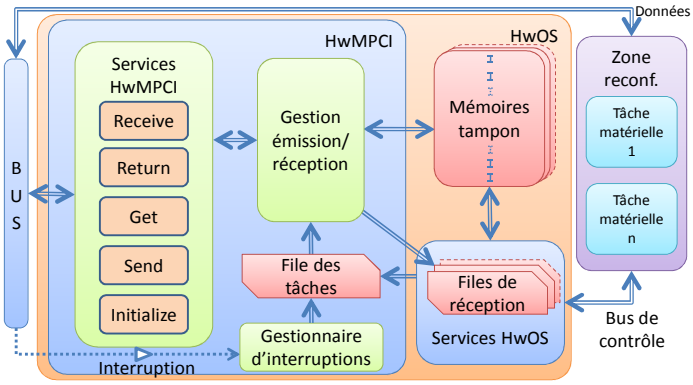

Figure 2 : Raffinement du HwMPCI

Si les services du HwOS peuvent être implémentés de manière concurrente afin de tirer partie de la structure hardware, les services du HwMPCI sont dépendants de l'accès au bus. Pour cette raison, un seul service HwMPCI pourra être exécuté à la fois.

Ces services sont commandés par un gestionnaire, chargés de les activer et de leur transmettre les paramètres nécessaires à leur exécution :

- La partie réception est sensible aux messages postés dans la file des tâches par le gestionnaire d'interruptions lors de la réception d'une interruption. Elle commande les services receive, afin de pouvoir récupérer le contenu d'un message, ainsi que return afin de libérer l'enveloppe utilisée après consommation. Notons que si le message reçu nécessite une réponse, alors l'enveloppe utilisée pour la réception est conservée pour stocker la réponse.
- La partie émission, quant à elle, s'active lorsque la file d'émission délivre une donnée en provenance d'un service du HwOS. Elle commande le service get, afin de récupérer une enveloppe dans laquelle placer le message à envoyer, et le service send afin d'envoyer le message une fois l'enveloppe remplie.

3.2 Implémentation

Après codage du HwMPCI comme illustré sur la Figure 2, on obtient les latences suivantes en cycles d'horloge :

- Initialisation : 97 cycles.
- Emission d'un message (get + send) : 377 cycles.
- Emission d'une réponse à un message précédent (send seul) : 223 cycles.
- Réception d'un message avec réponse demandée (receive seul) : 288 cycles.
- Réception d'un message ne nécessitant pas de réponse (receive + return) : 443 cycles.

Notons que ces chiffres comprennent les accès au bus et les latences induites par celui-ci : attente de l'autorisation d'accès, latence de la mémoire, … et peuvent ainsi varier en fonction du contexte.

Globalement, le HwMPCI tel que décrit ici supporte une fréquence pouvant aller jusqu'à 115 MHz en technologie Virtex5 (SX50T). Après synthèse, la surface utilisée est de 1395 générateurs de fonctions (soit 4,27% du total disponible), 349 CLB (4,28%) et 774 bascules (2,3%).

4 CONCLUSION

Nous avons proposé une implémentation du module de communication du HwOS. Bien sur, celui-ci n'est qu'une partie de ce même HwOS, et serait sans utilité sans les services propres à un OS, seuls capables de traiter les messages, qu'il soit en provenance d'un autre node ou bien d'une tâche locale.

Pour cette raison, les futurs travaux consistent en la réalisation de tels services, qui soient capables de traiter des messages en provenance de RTEMS, ainsi que d'en émettre lorsque l'accès à un objet distant est requis, et offrant la possibilité d'ordonnancement des tâches matérielles.

5 REFERENCES

[1] RTEMS (Real-Time Operating System for Multiprocessor Systems) : rtems.org

[2] Gaisler research : gaisler.com. Disponible également sur le site : RCC (RTEMS LEON/ERC32 GNU cross-compiler system) permettant la compilation RTEMS pour LEON3.

[3] FOSFOR (Flexible Operating System FOr Reconfigurable platform) : www.polytech.unice.fr/~fmuller/fosfor/