Evaluation d'une spécification synchrone pour une application embarquée.

Frédéric BIZOUERNE, Samir BOUAZIZ, Thierry MAURIN

IEF, Bâtiment. 220, Université de Paris-Sud, 91405 ORSAY Cedex
e-mail : {bizouern, bs, tm}@ief.u-psud.fr

Résumé : La plupart des applications complexes temps réel impliquent des volumes de calcul suffisamment important pour que leur implantation soit effectuée sur une architecture multiprocesseur. On connaît bien les avantages des outils de spécification temps-réel s'appuyant sur l'approche des langages synchrones dans des cadres applicatifs spécifiques. Cette approche permet d'assurer un déterminisme accru avec notamment SIGNAL et SynDEx. Nous avons testé l'extension de cette approche pour gérer l'asynchronisme de capteurs répartis et du réseau auquel ils sont reliés.

Abstract : As most of real time applications need more and more computing power, their implementation must be done on a multiprocessor architecture. We all know the advantage brought by real time specification tools used with a synchronous approach, on specific application. With this approach, SIGNAL and SynDEx can guaranted an improved determinism. We have tested this asynchronous approach on a particular application, in order to control the asynchronous data flow of the sensors and the network which link them together.

1. Introduction

Certaines applications complexes exigent, de plus en plus, l'emploi d'une puissance de calcul élevée. Cette puissance ne peut être atteinte que sur des architectures multiprocesseurs. Cette caractéristique est d'autant plus vraie lorsque l'application considérée impose des contraintes temporelles strictes mettant en danger des vies humaines. Afin de mieux exploiter le parallélisme et mieux répartir les différentes tâches de l'application, l'utilisation de méthodes et d'outils de conception s'impose. Ceci permet d'assurer un déterminisme et un minimum de sûreté de développement. Il existe dans le domaine du temps-réel, des outils de spécification algorithmique et matérielle, basés sur l'approche des langages synchrones permettant d'atteindre ce déterminisme souhaité.
Le but de cet article est de présenter l'intégration d'un réseau sur une architecture multiprocesseur à l'aide d'outils de spécification synchrone : SIGNAL et SynDEx. La première partie exposera l'application supportant cette méthodologie. Ensuite nous verrons comment gérer l'asynchronisme latent, en vue d'une implantation sur une architecture répartie.

2. Application

2.1. Architecture matérielle

L'application visée, possède des capteurs délivrant des données devant être traitées par une machine multiprocesseurs à base de DSPs. Les données sont acheminées entre les différents modules grâce à un réseau embarqué de type VAN[1] [1]. Les capteurs intègrent un microcontrôleur 80C51, qui effectue la récupération des données provenant d'un module analogique, et les transmet sur le réseau, via un contrôleur de protocole VAN de type RCP2. Les données sont récupérées par un autre microcontrôleur sur le bus (voir figure n°1), dont la principale fonction est d'interfacer le bus à une architecture multi-TMS320C40 [2]. Celle-ci réalise des opérations de traitement du signal et de perception puis d'analyse de situation à travers un système expert. L'application est répartie sur différents processeurs, entraînant une gestion parallèle de l'ensemble.

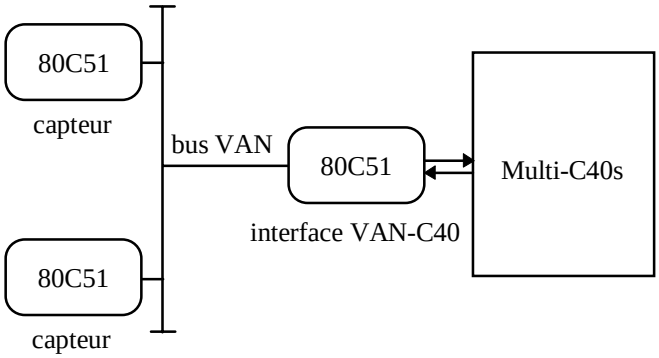

Figure n°1 : architecture matérielle

[1] Vehicle Area Network

2.2. Contraintes temporelles

Deux principales contraintes temporelles sont présentes dans notre application. La première concerne les capteurs qui délivrent des données avec une périodicité de 100 ms : cette période correspond au temps maximal pendant lequel le système expert doit réaliser son diagnostique avant qu'un nouveau cycle ne commence. Les données sont constituées de deux tableaux de 120 bits. Chaque bit représente une information sur la présence d'un écho infra rouge, dans un secteur angulaire de 120°. Nous pouvons considérer le système comme un flot-de-données, où le traitement n'est effectué que si toutes les données sont présentes.

L'autre contrainte, non négligeable, est l'asynchronisme des capteurs ; ils délivrent leurs informations selon leur propre horloge et non de manière synchrone. Ce phénomène est accentué par l'utilisation du réseau qui introduit des aléas temporels lors de la transmission des données.

Afin de mener à bien le développement de ce projet, nous avons choisi d'évaluer l'efficacité des outils de spécification synchrone. En effet, l'approche des langages synchrones doit permettre le développement d'algorithme de manière sûre, et le placement/ordonnancement optimisé de l'ensemble des traitements sur l'architecture cible. Nous avons utilisé pour cela SIGNAL et SynDEx, qui sont des logiciels destinés essentiellement au domaine du traitement du signal et des images en temps-réel et se basent sur un modèle de traitement flots-de-données synchrone.

3. Les outils de spécification synchrone

SIGNAL[2] est un langage synchrone flots-de-données, permettant de spécifier et vérifier des algorithmes sans prise en compte de l'aspect matériel. L'algorithme est décrit sous forme d'un graphe flots-de-données où les nœuds représentent les opérations et les arcs sont les transferts de données entre les opérations. Les langages synchrones [3] effectuent une vérification de cohérence entre les événements des signaux, sans aucune considération matérielle. Les durées d'exécution des opérations et de communication ne sont pas prises en compte. Seules les occurrences des signaux sont considérées : on ne parle pas de temps physique, mais de temps logique. Les sorties produites par une opération sont simultanées avec les entrées. SIGNAL produit un fichier de description du graphe vérifié, pour le logiciel SynDEx.

SynDEx[3] est un logiciel supportant la méthodologie Adéquation Algorithme Architecture (A^3) d'aide à l'implantation d'algorithmes soumis à des contraintes temps-réel, sur des architectures multicomposants [4]. Ce logiciel permet de spécifier l'architecture matérielle devant exécuter l'algorithme vérifié par SIGNAL. SynDEx réalise une implantation au moyen d'une heuristique, en distribuant les opérations sur l'ensemble des processeurs. Il effectue aussi l'ordonnancement de ces opérations sur chacun des processeurs. Une évaluation des performances est fournie afin de procurer une idée du nombre de processeur à utiliser pour atteindre les contraintes temporelles souhaitées. L'évaluation fournie par SynDEx repose sur la durée du chemin critique du graphe logiciel sur le graphe matériel. Après avoir proposé une adéquation algorithme architecture, SynDEx génère un exécutif pour chaque processeur, sous forme de macro-instructions. L'exécutif intègre des primitives de communications de type réseau et de type point-à-point, sans inter-blocages.
En ce qui concerne le code généré, SynDEx produit un exécutif sous forme de macro-instructions, qui est ensuite transcrit en langage de bas niveau. Ceci permet à l'utilisateur de créer ses propres macro-instructions, notamment en ce qui concerne les communications par bus, qui varient selon le protocole et le matériel choisi.

Il est possible de décomposer l'approche menée en trois couches (voir figure n°2). L'algorithme est développé à l'aide de SIGNAL (le matériel et l'aspect temporel n'interviennent pas). Ensuite, l'exécutif généré par SynDEx est dédié à une architecture optimisée pour cet algorithme. Enfin, en bas niveau, il est possible d'utiliser un micro-noyau spécifique.

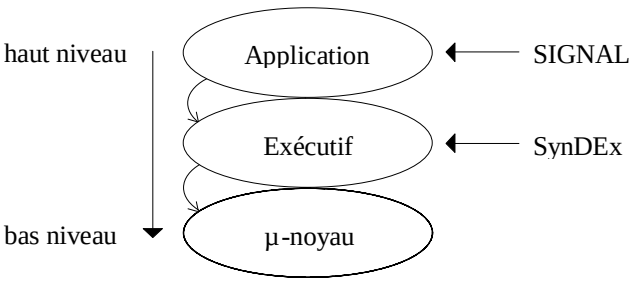

Figure n°2 : les couches de spécification

4. Spécifications

4.1. Architecture matérielle hétérogène

L'application est constituée d'une architecture hétérogène, ayant deux types de processeurs totalement différents : les microcontrôleurs gérant les transferts, les

[2] un projet de l'IRISA - Rennes

[3] un des objectifs du projet SOSSO de l'INRIA - Rocquencourt

DSP gérant les traitements. Ce problème d'hétérogénéité est résolu par le modèle de processeur générique utilisé par SynDEx. Cette généricité se répercute, aussi au niveau de l'exécutif par les macro-instructions, dont le code exécutable est spécifique au processeur considéré.

4.2. Aléas dans les transferts

De par la répartition géographique des différents microcontrôleurs, le placement de l'algorithme de récupération des données capteurs est fortement contraint. Chaque capteur est géré par un microcontrôleur afin d'effectuer le transfert des données sur le réseau, néanmoins, il est possible de faire un pré-traitement de ces données avant transmission. Le pré-traitement est un filtrage binaire sur les échos infra rouges. La spécification SIGNAL du filtrage peut être évaluée séparément, à l'aide de SynDEx par le calcul de la durée du chemin critique du graphe correspondant.

Notre architecture matérielle est fortement contrainte par l'utilisation du bus VAN, et de ses aspects asynchrones qu'il est nécessaire de gérer. Le bus VAN entraîne, comme tous les réseaux, des aléas au niveau du transfert des trames, (erreurs de transmission...). Néanmoins, il est possible de prendre en compte ce problème en le spécifiant. Nous présentons deux possibilités pour spécifier l'algorithme de transfert.

4.3. Première solution

La première solution consiste à intégrer un protocole en maître/esclave, entraînant un déterminisme des échanges de données.

Ce cas utilise la présence du réseau comme médium de communication. En effet, afin de rester déterministe, nous pouvons implanter un protocole maître/esclave. Comme tous les capteurs sont connectés sur le bus, et reçoivent toutes les trames y circulant, l'idée est de spécifier l'algorithme en ne considérant qu'un seul capteur. Le maître envoie une requête à un capteur, qui répond par des données. Comme nous le voyons sur la figure n°3, cet échange peut être décrit sous forme de graphe flots-de-données. Il suffit de dupliquer le code sur chaque capteur, et de changer le numéro d'identification du destinataire. Cette spécification prend en compte la valeur du flot-de-données en plus du flot lui même. Ainsi, la donnée transmise doit contenir le numéro du capteur.

L'intérêt de cette spécification est qu'elle ne dépend pas du nombre des capteurs. Par contre, le protocole maître/esclave implique une interrogation des différents

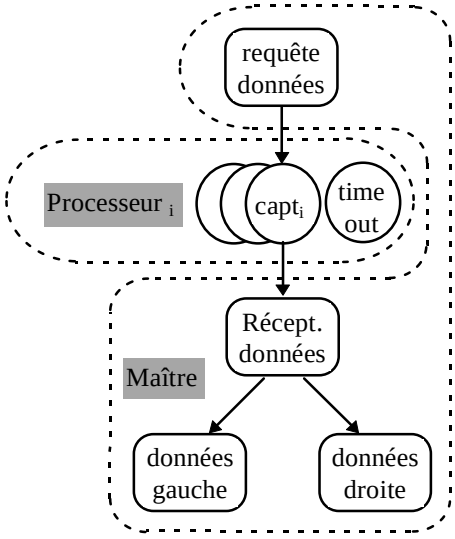

Figure n° 3 : placement des nœuds

capteurs de la part du maître, d'où un nombre élevé de messages sur le bus, pour le volume des données capteurs.

4.4. Deuxième solution

La seconde solution consiste à spécifier l'algorithme sans aucune considération du matériel, en laissant SynDEx se charger du déterminisme des échanges de données.

Dans ce cas, les flots d'entrée du graphe sont constitués des données des deux capteurs qui parcourent le graphe logiciel. Il est toutefois nécessaire de spécifier l'asynchronisme des capteurs au niveau de SIGNAL [5], afin de faire un appariement temporel des données capteur gauche et capteur droit.

Ensuite, le but est alors de placer les différents nœuds du graphe logiciel sur le graphe matériel, laissant à SynDEx le soin de prendre en charge les primitives de communications. SynDEx offre un protocole de communication effectuant une synchronisation par pré-ordonnacement entre émetteur et récepteur (voir figure n°4), permettant de s'abstenir du mode de fonctionnement en maître/esclave. Dans le cadre de notre application, lorsqu'un des capteurs émet un message au récepteur, l'autre capteur note l'émission de cette trame (nop), et peut donc émettre sa trame ensuite. Ceci permet une synchronisation des communications.

4.5. Granularité des flots

Il est aussi nécessaire de raisonner sur la granularité des flots d'entrée du graphe SIGNAL. Au niveau des capteurs, ils peuvent être considérés, soit au niveau bit, soit au niveau vecteurs :

• au niveau du bit. La spécification est alors fastidieuse, mais le filtrage binaire peut être réalisé entre chaque acquisition du flot. Cette solution n'est réaliste, en terme de performances, que sur le premier cas de spécification précédent (cf §4.4.), car sur le second cas, une requête bit à bit par trame est totalement inadaptée ; l'envoi des données est conditionné par une requête de trame, de la part du maître.

• au niveau tableau de bits complet. Cette solution pose des problèmes, car il y a une perte de temps dû au fait que le microcontrôleur doit avoir concaténé tous les bits des échos capteurs avant de les transmettre.
Pour optimiser les performances, il faut donc tenir compte des contraintes du contrôleur VAN RCP2 qui impose des limitations à 8 octets de données utiles maximum par trame transmise. Il faut donc découper les tableaux des échos capteurs, en plusieurs trames (un capteur envoie 120 bits échos). De ce fait, dès qu'une trame VAN d'un capteur est prête, elle est bufferisée dans le contrôleur, en attente de son envoi.

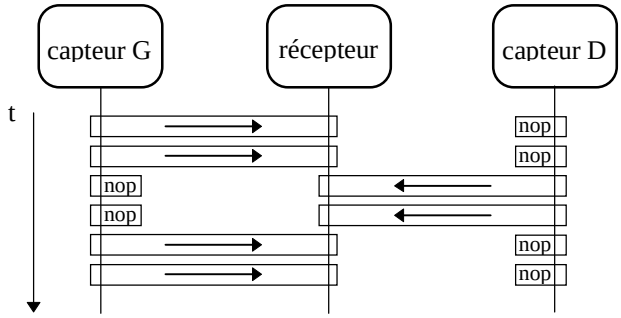

Figure n°4 : échange de messages

4.6. Gestion des aléas temporels du réseau

L'approche des langages synchrones se situe dans un cadre formel de temps logique, ne prenant pas en compte les défaillances matérielles, notamment ceux des médiums de communication. Lors du passage au temps physique, il est nécessaire de gérer ces aléas au niveau de l'application et de SynDEx, sinon le système risque de rester en attente active sur une donnée.

Il est possible d'intégrer dans les primitives de communication de SynDEx, la notion de "time-out", qui fournit une trame pour prévenir d'un problème de transmission. Bien sûr, cette fonction ne peut pas faire partie du graphe logiciel, mais elle est définie au niveau d'une interface de communication PBL (Processor to Bus Link) sous SynDEx. Cette notion de chien de garde est tout à fait réalisable, sur le microcontrôleur 80C51 grâce à la présence de timers.

Mais cette spécification n'est rendue possible que par l'existence des fonctions simultanées qui au sens du langage SIGNAL, fournissent des données en sortie de manière synchrone aux données en entrées. La spécification ne tient compte que de l'interface de cette fonction, et ne vérifie en aucun cas le contenu, qui est laissé au soin du programmeur. A l'aide de ces fonctions simultanées, il est aussi possible de spécifier une application avec différents grains. Une fonction simultanée peut regrouper tout un traitement décomposable en primitives du noyau SIGNAL.

Conclusion

Nous vous avons présenté une application mettant en jeu des capteurs connectés à un réseau VAN. Nous avons utilisé l'approche des langages synchrone afin de spécifier l'application en vue d'une implantation sur une architecture multiprocesseur.

Bibliographie

[1] M. Dang, A. Lagreze, I. Sabouni ″Communication Circuit for IntraVehicle Information Exchange″, Prometheus-Prochip Research Activities, Stuttgart, 1992.

[2] S. Bouaziz, R. Reynaud, T. Maurin ″Parallel Architecture for an Embedded Real-Time Application″, ICSPAT, October 1993.

[3] A. Benveniste, G. Berry, "The synchronous approach to reactive and real time systems", Proc. of the IEEE vol79, n°9, pp 1270-1282, 1991.

[4] C. Aiglon, C. Lavarenne, Y. Sorel, A. Vicard, "Utilisation de SynDEx pour le traitement d'images temps-réel", rapport de recherche INRIA n° 2968, thème 4, septembre 1996.

[5] R. Reynaud, Y. Sorel, C. Lavarenne, "Spécification et validation à l'aide d'un langage synchrone d'un protocole d'appariement de données asynchrones", GRETSI, septembre 1993.