Spécification, optimisation de performance et génération d'exécutif pour application temps-réel embarquée multi-processeur avec **SynDEx**

Christophe LAVARENNE, Yves SOREL
INRIA – Domaine de Voluceau, Rocquencourt B.P.105
78153 Le Chesnay Cedex – France
email : yves.sorel@inria.fr

Real-Time Embedded Processing for Space Applications
CNES International Symposium – Les Saintes-Maries-de-la-Mer, Novembre 1992

1 Objectifs

1.1 Domaines d'application de SIGNAL et SynDEx

On peut distinguer deux classes de systèmes de traitement flots de données temps-réel selon le nombre de processeur utilisés.
• Les petits systèmes où l'on vise la rapidité du développement des algorithmes et des programmes et l'efficacité de l'ordonnancement.
- télécoms : modem, annuleur d'échos ...
- électronique automobile : contrôle moteur, suspension ...
- électronique médicale : traitement d'ECG ...
• Les grands systèmes où l'on vise en plus des mêmes objectifs l'optimisation de la machine cible.
- aérospatial
- contrôle de processus industriels
- télécoms
- systèmes d'armes.

1.2 Implantation d'algorithmes avec SIGNAL et SynDEx

Nous utilisons trois environnements :
• **SIGNAL** pour la spécification et la vérification des algorithmes [1] [2],
• **SynDEx** pour l'Adéquation Algorithme Architecture (AAA) et la génération d'exécutif [3] [4],
• un environnement constructeur pour la compilation, le chargement et l'exécution sur des processeurs tels que **Transputer T800, TMS320C40** ...

1.3 Fonctionnalités de SynDEx

SynDEx est un environnement de programmation graphique interactif pour applications de traitement du signal et d'automatique s'exécutant en temps-réel sur des machines multi-processeur. Il offre les fonctionnalités suivantes :
• interface avec **SIGNAL** pour la spécification et la vérification de l'algorithme d'application afin d'obtenir des programmes fiables,
• spécification et dimensionnement de la machine multi-processeur en vue d'optimiser le matériel,
• placement-ordonnancement optimisé de l'algorithme d'application sur le multi-processeur avec évaluation et visualisation de ses performances,
• génération de l'exécutif distribué pour exécution en temps-réel, déchargeant l'utilisateur au maximum des tâches lourdes de programmation bas niveau.

2 Spécification et vérification avec SIGNAL

SIGNAL est un langage **FLOT DE DONNEES, SYNCHRONE**, permettant de décrire des relations entre des suites d'événements valués, les *signaux*. Ces relations sont définies à partir :
• d'une relation de *précédence* entre les évènements d'un même signal, ce qui permet de les compter et donc de dater leurs valeurs
• de relations de *dépendances fonctionnelles* entre des valeurs de signaux d'entrée de la fonction et des valeurs de signaux de sortie de la fonction
• d'une relation de *simultanéïté* entre évènements de signaux différents, on peut toujours savoir s'ils coïncident ou sont distincts.

L'hypohtèse *synchrone* consiste à supposer que les sorties produites par une dépendance fonctionnelle sont simultannées avec les entrées, ce qui revient à considérer que les calculs et les communications sont instantannées (on ne prend pas en considération à ce niveau les caractéristiques matérielles, le concept de durée n'existe qu'au travers du comptage des évenements d'un signal périodique).

Deux signaux sont *synchrones* si leur événements sont deux à deux simultanés, ce qui définit une relation d'équivalence. La classe d'équivalence d'un signal X est appelée son *horloge*, $h(X)$ et définit ses *présences/absences* relativement à d'autres signaux. A l'aide du calcul ensembliste (implanté par du calcul booléen naturel sur {présent, absent}) on définit les relations temporelles de base sur les horloges des signaux :

$$h(X) = h(Y)\ ;\ h(X) = h(Y) \cap h(Z)\ ;\ h(X) = h(Y) \cup h(Z).$$

Les dépendances fonctionnelles et les relations temporelles s'expriment en **SIGNAL** dans un style *flot de données* ce qui permet de décrire le *parallélisme* de l'application.

On dispose de *quatre processus élémentaires* et d'*un opérateur de connexion* par identité de nom "|".

processus élémentaire et sa syntaxe	relation sur les flots équations d'horloges	dépendance instantanée entre signaux
fonction immédiate $Y := F(X)$	$y = f(x)$ $h(y) = h(x)$	y dépend de x
retard $ZX := X\$1$	$zx(n) = x(n-1)$ $h(zx) = h(x)$	zx ne dépend pas de x
sous-échantillonage $Y := X$ when B	$y = x$ $h(y) = h(x) \cap h(b = \text{true})$	y dépend de x quand b présent et vrai
mélange $Y := X0$ default $X1$	$y = x0$ ou $x = x1$ $h(y) = h(x0) \cup h(x1)$	y dépend de $x0$ quand $x0$ présent, ou de $x1$ quand $x0$ absent et $x1$ présent

Les algorithmes sont représentés par un *graphe orienté* dans lequel :
• les *Sommets* sont caractérisés par des *ports* d'entrée ou de sortie auxquels sont associés des *flots de données*, chaque port est associé à un processus élémentaire
• les signaux sont représentés par les couples (port, horloge)
• les *Arcs* représentent des *dépendances* entre ports *conditionnées* par les horloges (une dépendance est active quand son horloge est présente)
• les horloges vérifient les relations associées aux processus élémentaires.

Lors de la compilation, l'existence de telles horloges et l'absence de cycles actifs sont vérifiés : c'est une *preuve de la correction temporelle du programme*. Le programme généré est aussi un graphe flot de données conditionné (*graphe logiciel*) où chaque sommet et chaque arc est étiqueté par une horloge d'activation exprimée maintenant en fonction de signaux externes.

3 Spécification du multi-processeur avec SynDEx

Le multi-processeur est modèlisé par un hypergraphe (*graphe matériel*) dont chaque sommet représente un *nœud-processeur* et chaque hyper-arc une liaison physique de communication connectant plusieurs nœud-processeurs (deux ou plus). Chaque nœud-processeur est décrit à partir des éléments suivants :
- processeur de calcul
- processeur de communication (un par hyper-arc adjacent)
- processeur d'entrée-sortie

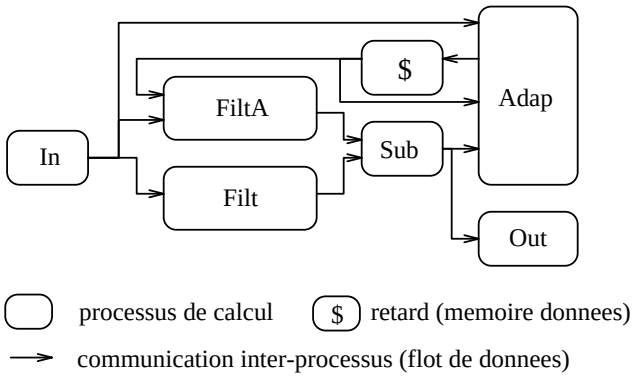


Figure 1: Graphe logiciel, exemple d'algorithme

- mémoire de données partagée entre éléments processeurs

Chaque élément processeur inclut sa mémoire de programme qui recevra les instructions de calcul et l'exécutif généré. La coopération entre processeurs de communication d'un même hyper-arc permet le transfert de données entre les mémoires données des nœud-processeurs connectés. Les processeurs d'entrée-sortie permettent au multi-processeur programmé de communiquer avec son environnement, c'est-à-dire de réagir (sorties) aux stimuli (entrées) de l'environnement.

Un modèle mathématique simple est associé à chacun des éléments d'un nœud-processeur, il est utilisé pour calculer les durées de calcul et de communication nécessaires pour déterminer le placement ordonnancement optimisé de l'algorithme sur le multi-processeur.

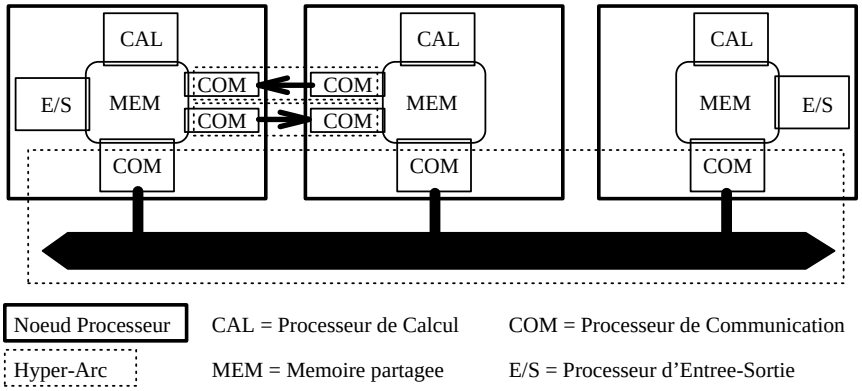


Figure 2: Graphe matériel, exemple d'architecture

4 Placement-ordonnancement optimisé avec SynDEx

Le placement ordonnancement est précalculé et statique (pas de migration ni d'ordonnancement de processus à l'exécution). Il consiste à placer (affecter) et ordonnancer (séquentialiser) les processus de calcul (resp. d'entrée sortie) sur les processeurs de calcul (resp. d'entrée sortie). Ce placement produit des communications inter-processeur qu'il faut également placer et ordonnancer sur les processeurs de communication. Pour cela on choisit un chemin (*route*) dans l'hypergraphe matériel, on crée entre les deux processus (de calcul ou d'entrée sortie) communiquant un *processus de communication* pour chaque hyper-arc de la route et on place chacun de ces processus de communication sur l'hyper-arc correspondant, c'est-à-dire sur chacun de ses processeurs de communication. Cette dernière opération consiste à diviser le processus de communication en sous-processus de communication coopérants, placé chacun sur un processeur de communication. Les sous-processus de communication issus de communications inter-processeur différentes sont ordonnancés sur chaque processeur de communication.

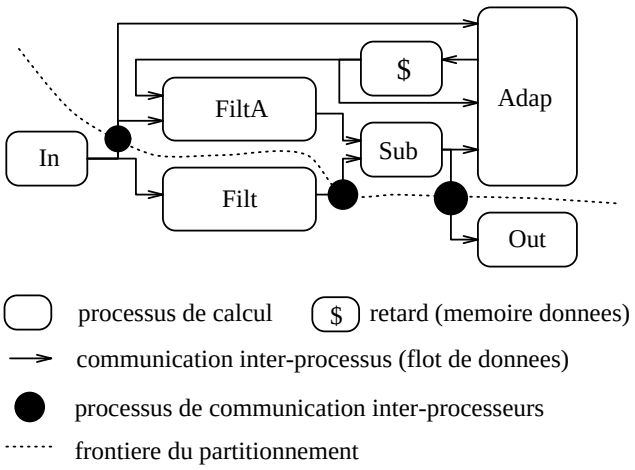


Figure 3: Graphe logiciel après placement

L'optimisation du placement ordonnancement actuellement effectuée consiste à minimiser le temps de réponse, c'est-à-dire la longueur du chemin critique du graphe logiciel valué par les durées d'exécution des processus de calcul et de communication. L'algorithme de placement ordonnancement actuellement implanté dans SynDEx est une heuristique (problème NP-complet) dont la complexité est une fonction linéaire du produit du nombre de processus et de processeurs.

L'heuristique consiste à faire progresser au long du graphe logiciel une coupe séparant les processus déjà placés sur des processeurs de ceux qui ne le sont pas encore. La progression se fait en respectant les précédences du graphe logiciel. De tous les processus à placer sur la coupe et de tous les processeurs, on choisit la paire qui optimise une fonction de coût locale prenant en compte :

- les différences entre dates locales d'exécution au plus tôt et au plus tard,
- l'allongement du temps global d'exécution : temps de réponse (*latence*) et rythme d'entrée (*cadence*),
- la capacité mémoire.

Le diagramme temporel calculé pendant l'optimisation permet une visualisation dans l'environnement graphique SynDEx (voir figure 5) faisant apparaître les communications inter-processeur et pour chaque processeur l'ordonnancement des processus qui y sont placés. Le temps se déroule de haut en bas avec une colonne par processeur. Chaque processus est représenté par une boîte dont la hauteur est proportionnelle à la durée d'exécution du processus. Chaque communication inter-processeur est représentée par un segment de droite dont l'origine se situe dans la colonne du processeur émetteur à la date de début de communication et dont l'extrémité se situe dans la colonne du processeur récepteur à la date de fin de communication. Entre cette date et la date de début du processus récepteur, la donnée est mémorisée dans un tampon. Comme on le constate sur la figure, le modèle choisi autorise la concurrence des calculs et des communications.

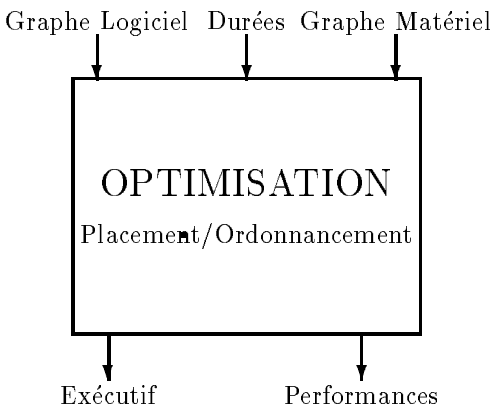
5 Méthode Adéquation Algorithme Architecture (AAA)

L'heuristique d'optimisation calcule les dates d'exécution et les performances à partir des durées d'exécution des éléments du graphe logiciel (processus de calcul ou de communication).

Les calculs des durées élémentaires de communication sont basés sur des modèles mathématiques simples [5], par exemple :

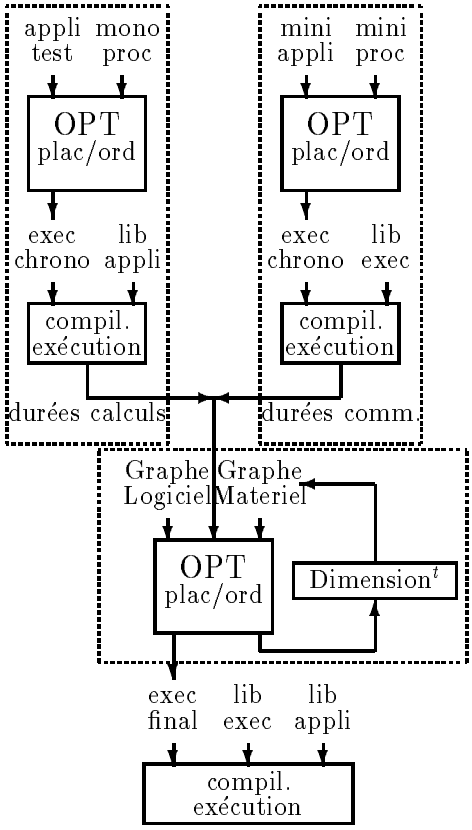
$$dt = \delta * v + \tau$$

- dt : durée de communication
- δ : débit de la liaison physique de communication
- v : volume de la communication (dépendant du type des données)
- τ : temps d'établissement de la communication



Les durées des calculs et les paramètres des modèles de calcul des durées de communications peuvent être obtenues par mesures réalisées sur un exécutif généré à partir de graphes simples (graphes logiciel et matériel génériques).

Le dimensionnement du graphe matériel (choix du nombre de processeurs, de leurs interconnexions ...) est effectué itérativement par l'utilisateur qui compare les performances calculées aux contraintes temps-réel qui lui sont imposées.



6 Génération d'exécutif optimisé avec SynDEx

Un exécutif distribué temps-réel offre des services tels que la gestion des ressources de *calcul*, de *communication* et de *mémoire* et la gestion du *temps*. La gestion des ressources de *calcul*, de *communication* et de *mémoire* est faite statiquement par le placement-ordonnancement des processus de calcul de l'application et des communications inter-processus. La *mémoire* application correspond aux retards SIGNAL. La mémoire système correspond aux tampons de communication intra- et inter-processeurs. La gestion du *temps* est faite statiquement par le placement ordonnancement des processus de calcul (booléen) des horloges produites par le compilateur SIGNAL.

Contrairement aux systèmes d'exploitation distribués standards (du genre de MACH et CHORUS) qui fournissent des services à travers une copie d'un noyau résidant sur chaque processeur, SynDEx produit un exécutif dédié, taillé sur mesure pour l'application [6].

L'utilisateur n'a pas d'autre code à écrire que celui de ses processus de calcul et d'entrée sortie. Tout le reste (ordonnancement et appel des processus de calcul et d'entrée-sortie, allocation de la mémoire nécessaire aux communications inter-processus –intra- ou inter- processeur– et code des processus de communication) est généré *automatiquement* par SynDEx à partir des graphes logiciel et matériel, des résultats du placement ordonnancement et du noyau générique de l'exécutif SynDEx. La simplicité du noyau facilite son portage sur de nouveaux composants matériels.

La méthode de génération de code garantit que le comportement entrées-sorties de l'application sera le même sur une architecture multi-processeur que sur l'architecture mono-processeur sur laquelle l'application aura été préalablement déboguée. Alors que dans un environnement classique, le portage sur un multi-processeur d'une application déboguée sur un mono-processeur demande plusieurs jours (voire plusieurs semaines) de codage et de déboguage, avec l'environnement SynDEx, ce portage est l'affaire de quelques minutes, au plus de quelques heures si le graphe représentant l'algorithme est de taille importante et que plusieurs cycles de placement ordonnancement sont nécessaires pour obtenir des performances satisfaisantes.

L'exécutif généré pour chaque processeur par SynDEx supporte l'ordonnancement et le conditionnement des processus qui y ont été placés, c'est-à-dire les processus d'entrée-sortie et de calcul (tirés d'une bibliothèque fournie par l'utilisateur) et les processus de communication (tirés du noyau générique de l'exécutif SynDEx) assurant le transfert des données entre processeurs par passage de messages. Ces processus réalisent les fonctions suivantes correspondant aux protocoles de communication des principaux niveaux de la norme ISO :

- formatage et déformatage des messages
- routage des messages à l'intérieur des processeurs
- transfert des messages à travers les liaisons physiques de communication.

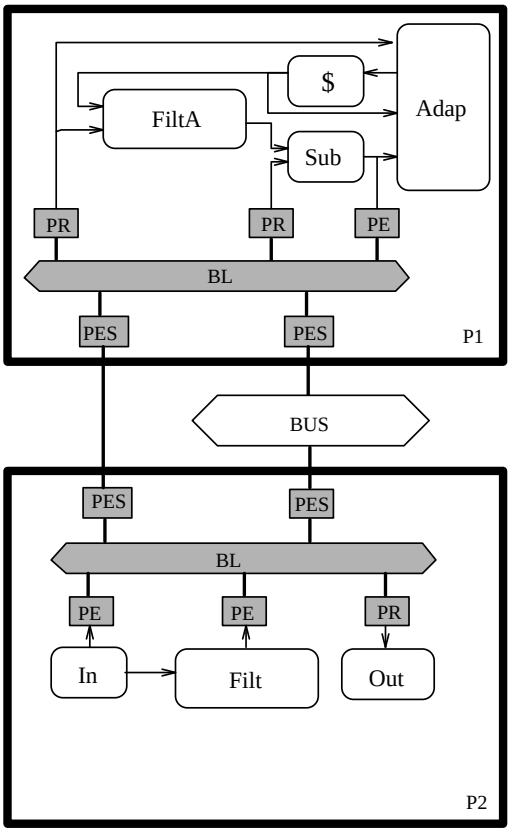


Figure 4: Exécutif généré pour l'exemple sur un bi-processeur

7 Le prototype de SynDEx

L'environnement **SynDEx** est programmé en SMALLTALK. Il comporte une interface interactive graphique et textuelle avec menus dépendant du contexte et zoom [7] permettant :
- l'édition des graphes logiciels et matériels
- la visualisation des performances calculées par l'algorithme de placement-ordonnancement
- la génération d'exécutif (OCCAM, C) pour Multi-Transputer

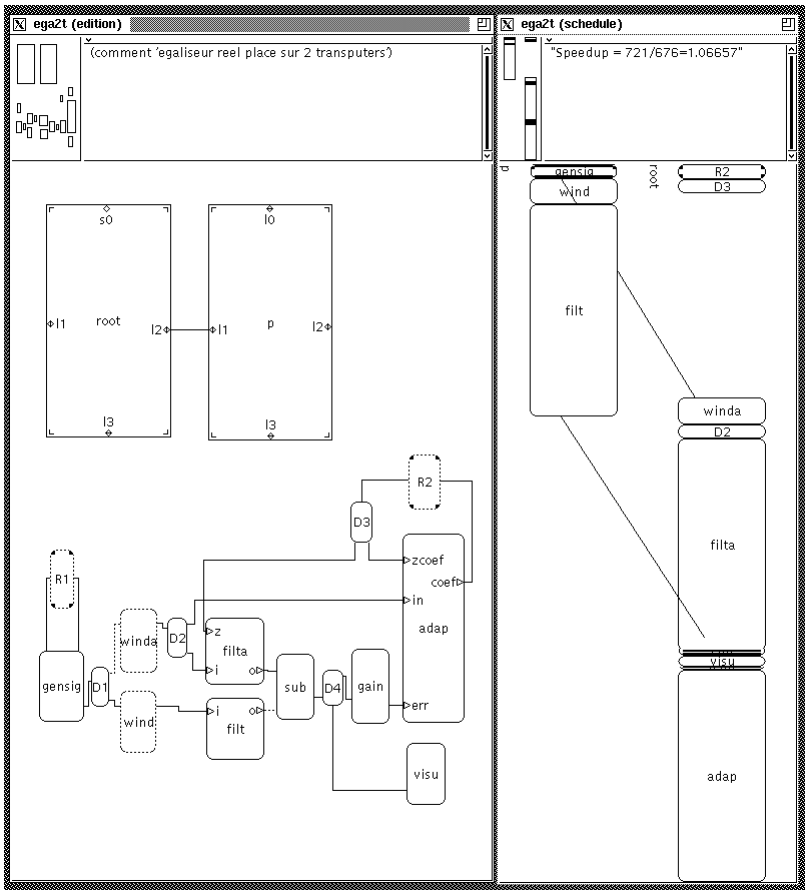


Figure 5: Environnement graphique SynDEx

8 Perspectives

• **SynDEx** et les langages synchrones : Estérel, LUSTRE, SIGNAL
Aujourd'hui SIGNAL est privilégié : une interface avec **SynDEx** a été définie. Demain grâce au travail fait dans les équipes conceptrices des langages synchrones, **un code commun GC**, sera disponible. **SynDEx** sera alors accessible à partir des trois langages à travers **GC**.
• Placement-ordonnancement dans le cas multihorloge
Dans ce cas le graphe flot de données conditionné peut changer d'un instant à l'autre. Aujourd'hui on traite "le mieux possible" le graphe correspondant à la durée d'exécution la plus longue, par une optimisation hors ligne. Des solutions moins conservatives sont envisageables. Elle s'appuieront sur la **duplication de processus** et des décisions d'ordonnancement en ligne.
• Machines cibles hétérogènes
Dans les applications la possibilité de prendre en compte des composants de calcul et de communications de performances diverses et de comportements différents (bus et lien par ex.) est importante. Les problèmes de dimensionnement et de placement-ordonnancement d'une plus grande complexité sont à étudier.

References

[1] A. Benveniste, P. le Guernic, Y. Sorel, M. Sorine
A Denotational Theory of Synchronous Reactive Systems.
Information and Computation, vol. 99, 192-230 (1992)

[2] P. Le Guernic, M. Le Borgne, T. Gautier, C Le Maire
Programming real-time applications with SIGNAL.
Rapports de Recherche INRIA n°1446 (Juin 1991)

[3] C. Lavarenne, O. Seghrouchni, Y. Sorel, M. Sorine
The SynDEx software environment for real-time distributed systems design and implementation.
Proc. of the European Control Conference (Juillet 1991)

[4] C. Lavarenne, O. Seghrouchni, Y. Sorel, M. Sorine
SynDEx, un environnement de programmation pour applications de traitement du signal distribuées.
Treizième Colloque GRETSI (Septembre 1991)

[5] F. Ennesser, C. Lavarenne, Y. Sorel
Méthode chronométrique pour l'optimisation du temps de réponse des exécutifs SynDEx.
Rapports de Recherche INRIA n°1769 (1992)

[6] N. Ghezal, S. Matiatos, P. Piovesan, Y. Sorel, M. Sorine
SYNDEX, un environnement de programmation pour multi-processeur de traitement du signal Mécanismes de communication.
Rapports de Recherche INRIA n°1236 (1990)

[7] C. Lavarenne, Y. Sorel
SYNDEX, un environnement de programmation pour multi-processeur de traitement du signal Manuel de l'utilisateur version v.0.
Rapports Techniques INRIA n°113 (1989)