Architecture parallèle basée sur des TMS320C30 pour la comparaison de signaux

Pierre Fréchette,
Laboratoire Scribens et
Les systèmes électroniques Matrox Ltée

Réjean Plamondon,
Laboratoire Scribens,
Département de génie Electrique et de génie Informatique.
École Polytechnique de Montréal.

Abstract

The automatic verification of signatures is a problem that requires much processing power. To make up for this need of power, conmparison algorithms have to be implemented in a parallel processing environment.

The first step toward this goal is to design and build a parallel processor system. The system that we design has four processing elements, each based on a *TMS320C30* digital signal processor. The system is a *MIMD* machine with a dynamic architecture based on a common bus. It is made to operate with a PC/AT. Its peak performance is 133 MFLOPS (million floatting point operations per second) and 66.7 MIPS (million instructions per second).

Two algorithms for signal comparison have been implemented in this environment dynamic programming and the regional correlation. The execution time on the parallel processor is 0.012 second for a signature of an average length (5 seconds) for the regional correlation algorithm and 1.6 second for the dynamic programming, in comparison to 6 seconds and 20 seconds on a standard PC/AT, respectively

Résumé

Ce projet s'inscrit dans le cadre de la comparaison automatique de signaux dans un environnement composé de processeurs parallèles. Les objectifs étaient de concevoir une carte de processeurs parallèles basée sur des processeurs de signaux numériques et d'étudier les performances de deux algorithmes de comparaison de signaux. Ces signaux sont les composantes en X et en Y de la vitesse de la pointe d'un crayon, telles qu'extraite lors de la capture de la signature, l'objectif à moyen terme de ce projet étant la vérification automatique de signatures.

Dans un premier temps, une carte de processeurs parallèles a été conçue et réalisée. La carte possède quatre processeurs de traitement numérique de signaux, des *TMS320C30*. Le système est de type *MIMD* et l'architecture est à connexions dynamiques et basée sur un bus commun. L'environnement du système est le PC/AT. Les performances maximales sont de 133 MFLOPS et 66.7 MIPS.

Les algorithmes de comparaison de signaux qui ont été implantés sont la programmation dynamique et la corrélation régionale. Le temps d'exécution de l'algorithme de corrélation régionale sur la carte de processeurs parallèles est d'environ de 0.012 seconde pour des signatures de longueurs moyennes (5 secondes) et de 1.6 seconde pour l'algorithme de programmation dynamique comparativement à 6 secondes et 20 secondes respectivement sur un PC/AT.

I. Introduction

Afin de vérifier la signature d'un individu, une multitude de méthodes peuvent être utilisées [1]. Dans le cas des techniques dynamiques, la vérification se fait lors de l'action de signer, à partir d'une tablette à numériser permettant d'acquérir en fonction du temps, les coordonnées positionnelles de la pointe du crayon lors de la signature. Les caractéristiques dynamiques d'une signature peuvent alors être représentées par une fonction $F(t)$, par exemple, la vitesse ($v_x(t)$, $v_y(t)$). Ces fonctions sont ensuite comparés avec une référence à l'aide d'un algorithme spécifique. Puisque l'information contenue dans $F(t)$ est souvent complète, les résultats de la comparaison sont généralement meilleurs que ceux obtenus en ne comparant que des paramètres extraits de ces signaux. Un désavantage des techniques dynamiques basées sur la représentation par une fonction $F(t)$ est la longueur du temps d'exécution des algorithmes de comparaison.

Dans ce travail, nous avons conçu une architecture parallèle basée sur des processeurs spécialisés dans le traitement numérique de signaux. Nous nous sommes ensuite servi de cet environnement pour évaluer des algorithmes de comparaison de signaux [2,3]. Les algorithmes expérimentés sont la corrélation régionale et la programmation dynamique. Les performances de ce système ont été évaluées dans le contexte de la vérification automatique des signatures.

II. Architecture parallèle basée sur des TMS320C30

Les classes de processeurs parallèles les plus intéressantes pour notre projet sont les classes *SIMD* et *MIMD*. La classe *SIMD* est légèrement limitée dans ses applications, car elle requiert une même séquence d'instructions à chaque processeur. La classe *MIMD* par contre, possède un mode de fonctionnement asynchrone, donc beaucoup plus versatile. C'est cette dernière qui définit le mode de fonctionnement de la carte que nous avons conçu (ci-après appellée "*RUNNER*").

L'architecture impose le mode de communication entre les éléments de traitement. Les principaux aspects à considérer avant d'arrêter un choix sont: le coût pour réaliser l'architecture, la disponibilité des composants, le nombre de processeurs, l'échéancier pour la conception, la surface disponible, l'environnement de la carte et finalement, sa versatilité.

Une première décision à prendre concerne le nombre de processeurs. Ce nombre influence évidemment les critères coûts, surface utilisée et performances. Étant donné la disponibilité et le coût élevé des processeurs spécialisés en traitement numérique de signaux, leur nombre

a été fixé à quatre dans notre projet.

L'architecture utilisée est une architecture à bus. Elle permet d'avoir des connexions dynamiques entre chaque processeur. Cette dernière utilise le moins de composants, par conséquent, le moins de surface et elle est aussi la moins coûteuse à réaliser. Un désavantage de cette architecture est la diminution de sa largeur de bande avec l'accroissement du nombre de processeurs. Une étude sérieuse de l'impact sur la communication s'imposerait dans notre projet si le nombre de processeurs avait dépassé environ dix [4]. Le dernier avantage important de cette architecture est sa versatilité. Il est possible de simuler d'autres architectures à partir de celle-ci, ce qui permet d'étudier différents types d'algorithmes. De plus, cette architecture permet d'enlever un ou plusieurs processeurs sans toutefois briser la communication. La figure 1 présente un schéma global de cette architecture.

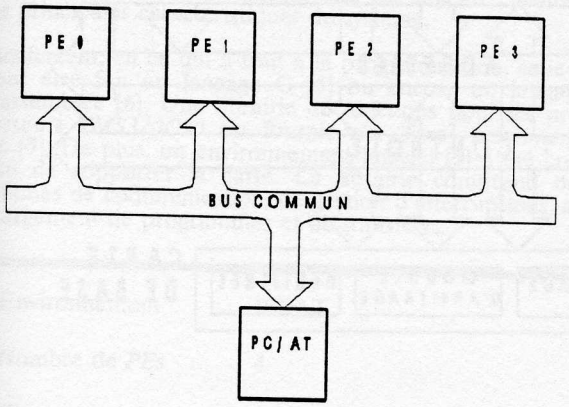


Figure 1: Architecture à bus commun.

A. Description de la carte RUNNER.

La présente section décrit sommairement la carte *RUNNER*. Pour avoir une description détaillée de celle-ci, il faut se référer au rapport technique "*Spécifications matérielles et logicielles d'une carte de processeurs parallèles, carte RUNNER*" [3].

1. Description du processeur.

Puisque l'on traite des signaux, il est intéressant d'utiliser un processeur spécialement conçu pour cette fin, c'est-à-dire ayant une unité arithmétique optimisée en fonction du temps d'exécution pour des calculs arithmétiques. Les processeurs numériques de signaux qui étaient disponibles pour réaliser notre projet, étaient le *TMS320C30* de Texas Instruments, le *DSP32C* de AT&T et le *DSP96002* de Motorola. Étant donné le support technique offert par Texas Instruments pour la compagnie *Les systèmes électroniques Matrox Ltée* et la disponibilité du produit, notre choix s'est arrêté sur le *TMS320C30*.

Les principales caractéristiques du *TMS320C30* [5] sont: opérations en point flottant, bus de 32 bits, 33.3 MFLOPS, 16.67 MIPS, 60 nsec par cycle d'instructions, 16 Megamots d'adressages, 2 Kmots de mémoire vive interne, mémoire cache de 64 mots, multiplication sur 40 bits, instructions parallèles et deux bus. Ces bus sont appelés bus primaire et bus secondaire. Le bus primaire est caractérisé par des lignes de commande permettant de placer le bus en attente (haute impédance). Ces lignes sont utiles pour la communication entre les processeurs.

2. Elément de traitement.

L'élément de traitement, aussi appelé *PE*, est constitué d'un processeur avec son environnement, c'est-à-dire la mémoire vive (RAM), les tampons et le module de contrôle. La carte possède quatre éléments de traitement.

Puisque les processeurs possèdent deux bus, un bus est utilisé pour la communication: le bus primaire qui possède des lignes *(HOLD)* pouvant mettre le bus en attente. Le transfert de données se fait par l'entremise d'une mémoire vive (RAM) interfaçée au bus primaire (32K X 32 bits). Sur le bus secondaire, la mémoire vive (RAM) est limitée à 8K X 32 bits dû à la capacité d'adressage du bus. Un module de contrôle permet de demander l'accès à un autre *PE* ou encore d'indiquer au processeur qu'un autre *PE* veut lui transmettre des données. Ce module possède aussi un système pour éviter les impasses (deadlock).

Comme la figure 2 le démontre, la communication entre les processeurs se fait par l'intermédiaire de la mémoire vive sur le bus primaire. En aménageant le programme et les données dans les mémoires secondaire et interne, le processeur n'est nullement ralenti lorsqu'un autre *PE* accède à sa mémoire.

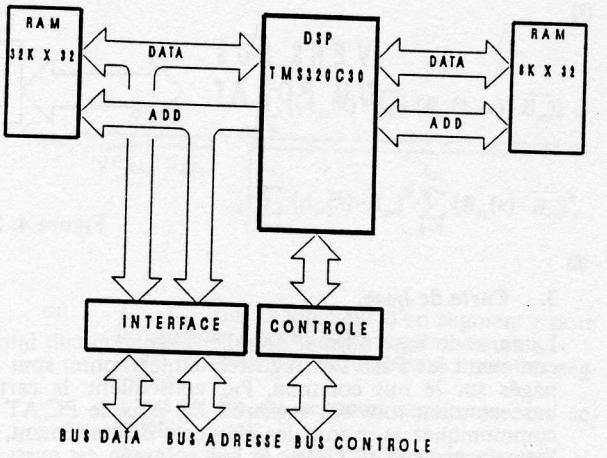


Figure 2: Schéma bloc d'un *PE*.

En ce qui concerne le montage physique, chaque *PE* est fixé sur son propre circuit imprimé. Les éléments de traitement sont reliés par l'intermédiaire d'une carte de base. De cette façon, il est possible de n'utiliser que trois, deux ou encore un seul *PE*, tout dépendant de l'application envisagée. La figure 3 montre l'assemblage de la carte *RUNNER*.

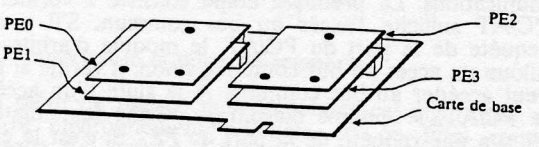


Figure 3: Montage physique du système.

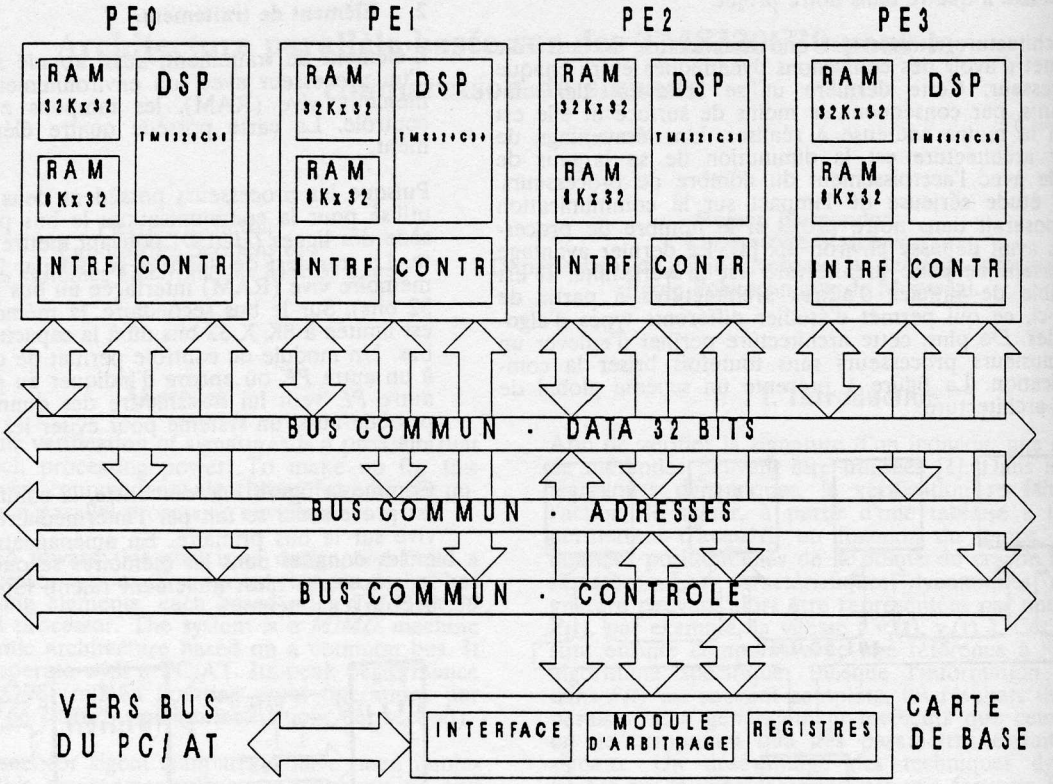


Figure 4: Schéma bloc de la carte parallèle.

3. Carte de base.

La carte de base permet de relier chaque circuit imprimé contenant les PEs. Les registres d'interruption sont aménagés sur le bus commun. Par conséquent, la carte de base contient tous ces registres. De plus, le PC/AT peut communiquer avec tous les éléments de traitement, ainsi l'interface du PC/AT avec le bus commun est aussi aménagée sur cette carte. La figure 4 représente le diagramme bloc de la carte.

Puisque l'architecture est de type bus commun, un module arbitre les accès au bus. Ce module fait la gestion des communications en déterminant quel processeur a accès au bus commun. Certaines priorités sont données concernant l'ordre des accès. Par contre, il faut éviter qu'un processeur prenne possession du bus et vienne ainsi bloquer la communication entre les autres processeurs. Le module d'arbitrage doit donc gérer les compromis. Si plus d'un *PE* veut accéder au bus commun, le module alloue un temps à chaque processeur de façon séquentielle. La figure 5 décrit l'arbitrage des communications. La première étape consiste à vérifier si le PC/AT sollicite l'accès au bus commun. S'il y a une requête de la part du PC/AT, le module d'arbitrage lui alloue un accès au bus commun, sinon, il vérifie si le PE_0 veut accéder au bus commun. A la suite d'un accès par le PC/AT, le module d'arbitrage vérifie immédiatement s'il y a une requête de la part du PE_0. Si c'est le cas, un accès lui est accordé. Cette méthode permet d'accorder un temps égal à chaque PE ainsi qu'au PC/AT.

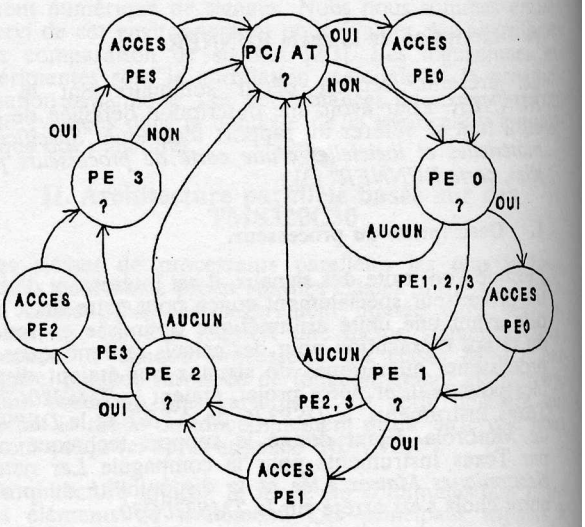


Figure 5: Arbitrage des accès au bus.

Puisque la carte de base est indépendante des PEs, il est possible de la modifier pour avoir une interface avec d'autres systèmes comme les bus VME, multibus, etc... De plus, il est possible de modifier le mode d'arbitrage en modifiant les équations des composants programmables. En modifiant la carte de base, l'architecture du système peut être transformée.

B. Caractéristiques techniques.

La carte fonctionne à la fréquence maximale prévue pour les processeurs, soit 33.3 MHz. Par conséquent, les performances maximales qu'atteint la carte sont de 133.3 MFLOPS et 66.67 MIPS. La performance de 133.3 MFLOPS implique que tous les processeurs fonctionnent en même temps et que les instructions parallèles sont utilisées.

Pour la carte *RUNNER*, la vitesse de communication entre deux *PEs* est de 660 nsec./mots dans le pire cas et de 540 nsec./mots dans le meilleur cas. Un mot a une longueur de 32 bits. Le meilleur cas est obtenu lorsque le *PE* destinataire est déjà en attente. La largeur de bande décrit le nombre de transferts que peut supporter le réseau. Étant donné que le bus ne supporte qu'une seule communication à la fois, cette largeur de bande est de 6 Megaoctets/seconde dans le pire cas et de 7.4 Megaoctets/seconde dans le meilleur cas. Le tableau 1 résume les principales caractéristiques du système.

Finalement, en ce qui a trait à la programmation, celle-ci peut être fait en langage C [5] ou encore en langage d'assemblée [6]. Une librairie de fonctions pour les processeurs *TMS320C30* est fournie par *Texas Instruments inc* [7]. De plus, un environnement logiciel [3] a été créé afin de supporter la carte. La librairie comprend des routines de communication, de gestion d'interruptions, de chargement de programmes et de transferts.

Environnement	PC/AT
Nombre de *PEs*	4
Classe	*MIMD*
Architecture	Dynamique à bus
Fréquence maximale	33.33 MHz
Instructions/seconde	66.67 MIPS
Instructions point flottant par seconde	133 MFLOPS
Vitesse de communication	660 nsec/mot (32 bits)
Largeur de bande	6 Megaoctets/sec.
Espace utilisé	2 fentes d'extension.

Tableau 1.: Caractéristiques de la carte *RUNNER*.

III. Performance des algorithmes de comparaison de signaux

A. Corrélation régionale

La corrélation régionale est basée sur un outil mathématique, soit la corrélation linéaire. On l'appelle corrélation régionale car la corrélation est effectuée par segment sur les fonctions représentant les signaux à comparer [8][9][10]. Cet algorithme est basé sur la corrélation linéaire, une mesure d'asssociation linéaire entre deux variables. Par exemple, le coefficient de corrélation C_{AB} pour deux signaux échantillonnés A(k) et B(k) est défini de la façon suivante:

$$C_{AB} = \frac{\sigma_{AB}}{\sqrt{\sigma_{AA}\cdot\sigma_{BB}}} \qquad (1)$$

où σ_{ij} est la variance entre i et j.

Puisque nos signaux de vitesse ont des composantes en X et en Y, il est important de développer une métrique de comparaison spécifique aux signatures. Pour les deux signaux A(k) et B(k), la corrélation est faite entre $A_x(k)$ et $B_x(k)$ pour un τ donné et aussi entre $A_y(k)$ et $B_y(k)$ pour le même τ, où τ est le décalage entre les segments. Le coefficient de corrélation entre une paire de segments r et pour un décalage τ est donc défini par:

$$C_{ABxr_\tau} = \frac{\sigma_{ABxr_\tau}}{\sqrt{\sigma_{AAxr}\cdot\sigma_{BBxr}}} = \frac{\sum_{k=p}^{q}[(A_{xr}(k)-\overline{A_{xr}})\cdot(B_{xr}(k-\tau)-\overline{B_{xr}})]}{\sqrt{\sum_{k=1}^{L_{A_r}}(A_{xr}(k)-\overline{A_{xr}})^2\cdot\sum_{k=1}^{L_{B_r}}(B_{xr}(k)-\overline{B_{xr}})^2}} \qquad (2)$$

$$C_{AByr_\tau} = \frac{\sigma_{AByr_\tau}}{\sqrt{\sigma_{AAyr}\cdot\sigma_{BByr}}} = \frac{\sum_{k=p}^{q}[(A_{yr}(k)-\overline{A_{yr}})\cdot(B_{yr}(k-\tau)-\overline{B_{yr}})]}{\sqrt{\sum_{k=1}^{L_{A_r}}(A_{yr}(k)-\overline{A_{yr}})^2\cdot\sum_{k=1}^{L_{B_r}}(B_{xr}(k)-\overline{B_{xr}})^2}} \qquad (3)$$

où L_A, L_B sont la longueur d'un segment r pour les signaux A(k) et B(k) respectivement,
p est le début du recouvrement entre les segments,
q est la fin du recouvrement entre les segments.
$\overline{A_{xr}}$, $\overline{A_{yr}}$, $\overline{B_{xr}}$, $\overline{B_{yr}}$ sont les moyennes des signaux pour un segment r.

La plus grande corrélation entre les deux segments r est obtenue lorsque la moyenne géométrique des composantes X et Y des coefficients est maximisée:

$$C_{ABr} = \underset{\tau}{MAX}\left[\sqrt{C_{ABxr_\tau}\cdot C_{AByr_\tau}}\right] \qquad (4)$$

Le coefficient de corrélation final est la somme des coefficients de corrélation pour chaque segment r.

$$C_{AB} = \sum_{r=1}^{N} C_{ABr} \qquad (5)$$

où N est le nombre de segments.

La méthode utilisée pour implanter cet algorithme dans notre environnement parallèle est de faire travailler chaque *PE* sur un τ donné et ceci pour le même segment. Chaque *PE* fait la recherche des coefficients de corrélation maximums pour les τ donnés et pour un segment. Par la suite, un *PE* se charge de trouver le maximum parmi les quatre coefficients.

B. Programmation dynamique

La programmation dynamique consiste à faire une transformation non-linéaire de l'échelle du temps afin de minimiser la distance entre deux signaux [9]. Le résultat que l'on obtient est une fonction déformante w qui indique la correspondance entre les échelles de temps des signaux A et B. La figure 6 représente une fonction déformante avec les signaux à comparer.

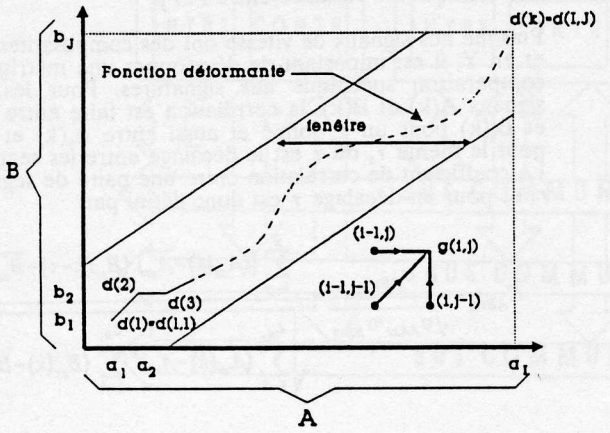


Figure 6:Fonction déformante entre deux signaux.

Puisque les signaux des signatures d'une même personne doivent avoir une certaine ressemblance, il est possible d'imposer des frontières à la fonction déformante. Ainsi, cette frontière spécifie le déphasage maximum permis entre les deux signaux. Il en ainsi résulte une diminution du nombre de calcul à effectuer.

Afin de calculer la fonction déformante, une métrique de distance est utilisée. Dans notre problème, la distance d entre deux échantillons est définie par:

$$d(w(k)) = d((i_x(k), i_y(k)), (j_x(k), j_y(k))) = \sqrt{(i_x(k)-j_x(k))^2 + (i_y(k)-j_y(k))^2} \quad (6)$$

La distance globale et minimale entre les signaux est donnée par:

$$D_r = MIN\left[\sum_{k=1}^{K} d(w(k))\right] \quad (7)$$

L'équation de la programmation dynamique est la suivante:

$$w(i,j) = MIN\begin{bmatrix} g(i,j-1) & + & d(i-j) \\ poids \times g(i-1,j-1) & + & d(i,j) \\ g(i-1,j) & + & d(i,j) \end{bmatrix}$$

Pour calculer les distances à chaque point, nous devons connaître le point précédent. Par conséquent, l'algorithme a la caractéristique d'être sériel et ainsi il s'avère plus difficile de le rendre parallèle que celui de la corrélation régionale.

La méthode choisie pour la mise en parallèle consiste à faire travailler chaque processeur sur une ligne [11][12][13][14]. L'algorithme sous forme parallèle demande beaucoup de communication inter-processeurs. Le mode qui a été choisi est une communication en anneau. Ainsi chaque processeur transmet ses résultats à son voisin et ainsi de suite. La figure 7 montre la communication entre les processeurs ainsi que l'information transmise.

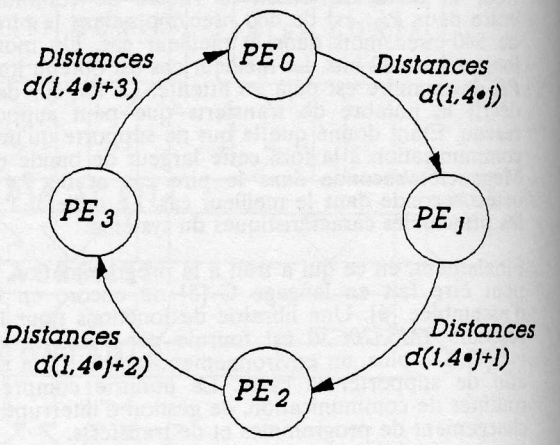


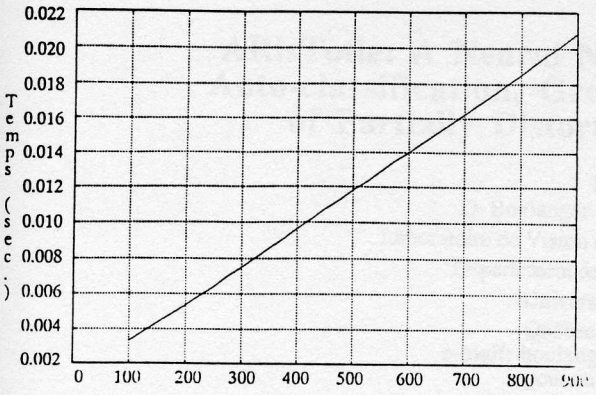
Figure 7:Communication inter-processeurs.

C. Temps d'exécution des algorithmes.

Les performances des algorithmes implantés ont été évaluées en terme de temps d'exécution, c'est-à-dire le temps que prend une comparaison entre deux signatures. Le transfert des coordonnées des signatures est effectué pendant que l'individu appose sa signature, ce qui permet de réduire le temps pour obtenir un résultat. Dans l'estimation des temps d'exécution, la transformation des vecteurs position en vecteurs vitesse est incluse.

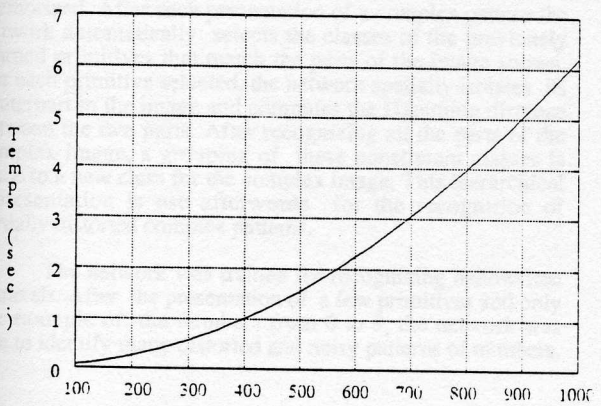
Pour l'algorithme de corrélation régionale, le décalage permis a été fixé expérimentalement à 15%. Le temps d'exécution fut d'environ 0.012 seconde pour une comparaison de deux signatures de longueur moyenne de 5 secondes. Sur un ordinateur personnel de type PC/AT 12 MHz, ce temps est d'environ 6 secondes. Le temps d'exécution de l'algorithme est proportionnel à la longueur des vecteurs. Une courbe représentant le temps d'exécution de l'algorithme en fonction de la longueur des signatures est présentée à la figure 8.

Dans le cas de l'algorithme de programmation dynamique, la contrainte sur la largeur de la fenêtre était de 15% et il n'y avait aucun poids sur la diagonale. Le temps d'exécution fut d'environ 1.6 seconde pour une comparaison de deux signatures de longueur moyenne de 5 secondes. Sur un ordinateur de type PC/AT 12 MHz, ce temps est d'environ 20 secondes. Le temps de comparaison est proportionnel au produit des longueur des signatures. La contrainte sur la largeur de la fenêtre fait diminuer ce temps. La figure 9 présente une courbe du temps d'exécution en fonction de la longueur des signatures.



Longueur des signatures (seconde)
(100 échantillons/seconde)

Figure 8: Temps d'exécution de l'algorithme de corrélation régionale en fonction de la longueur des signatures.



Longueur des signatures (seconde)
(100 échantillons/seconde)

Figure 9: Temps d'exécution de l'algorithme de programmation dynamique en fonction de la longueur des signatures.

IV. Conclusion

Les principaux objectifs de cet article étaient d'étudier une architecture parallèle basée sur des processeurs de signaux numériques et d'évaluer les performances d'algorithmes de comparaison de signaux dans cet environnement. Les signaux que nous avons comparés étaient les composants X et Y du vecteur de vitesse de signatures.

Afin d'atteindre nos objectifs, un environnement de processeurs parallèles a été conçu et réalisé. Ce système est basé sur quatre processeurs de signaux numériques, de type *TMS320C30*, et l'architecture est à connexions dynamiques à bus commun. Deux algorithmes de comparaison de signaux, la programmation dynamique et la corrélation régionale, ont été implantés sous forme parallèle.

Cette étude a fait ressortir quelques points importants. En ce qui a trait à la carte de processeurs parallèles, un des avantages de l'architecture est sa flexibilité. Étant donné que l'architecture est à connexions dynamiques, il est possible de modifier à volonté la configuration des communications. L'autre avantage de l'architecture est que l'ordinateur hôte est utilisé comme cinquième élément de traitement (PE). Celui-ci a accès à presque toutes les mêmes ressources que les quatres éléments de traitement. Le processeur de signal numérique utilisé s'est avèré très puissant. Un seul processeur dépasse les performances d'un processeur 80286, ou même 80386, utilisé dans les ordinateurs personnels (e.g. PC/AT).

Étant donné que l'algorithme de corrélation régionale demande peu de communication et surtout des calculs arithmétiques, les performances de l'algorithme concernant son temps d'exécution ont été largement améliorées, soit un gain de 500 par rapport aux performances d'un PC/AT. L'algorithme de programmation dynamique nécessite beaucoup de communication entre les processeurs, par conséquent, les performances concernant le temps d'exécution ont été limitées par le taux de transfert que supporte la carte de processeurs. Le facteur d'amélioration n'a été que de 12.5.

Cependant, la carte de processeur n'a pas seulement que des avantages. L'étroitesse de la largeur de bande du système est un désavantage. Le canal de communication, soit le bus commun, ne peut supporter qu'un taux de transfert de 6 MegaOctets/sec. Ce taux est suffisant pour le traitement des signaux de signatures, mais insuffisant si l'on vise par exemple du traitement d'images. Le taux est directement limité par le fait q'une architecture à bus commun ne supporte qu'une seule communication à la fois.

Remerciements

Ce projet a été effectué dans le cadre d'une convention de recherche entre l'École Polytechnique et la compagnie *Les systèmes électroniques MATROX Ltée*. Nous désirons remercier tout particulièrement M. Laval Tremblay, directeur du groupe de traitement d'images et M. Donald Connolly pour leur support technique. De plus, nous voulons remercier les gens du Laboratoire Scribens pour leur participation à ce projet. Les devis des différents circuits imprimés ont été réalisé avec l'étroite collaboration de M. Gaétan Décarie, technicien au département de génie Électrique et génie Informatique de l'École Polytechnique. Finalement, nous remercions le CRSNG et l'École Polytechnique pour leur soutien financier.

Références

1. PLAMONDON, R., LORETTE, G., "*Automatic signature verification and writer identification - the state of the art*", Pattern Recognition, vol 22, No 2, 1989, p. 107-131.

2. FRÉCHETTE, P., "*Comparaison de signaux par processeurs parallèles*", Mémoire de maîtrise, École Polytechnique de Montréal, octobre 1990.

3. FRÉCHETTE, P., PLAMONDON, R., "*Spécifications matérielles et logicielles d'une carte de processeurs parallèles, carte RUNNER*", Rapport technique, Ecole Polytechnique de Montréal, EPM/RT-90/14, septembre 1990.

4. ALMASI, G.S., GOTTLIEB, A., *Highly parallel computing*, Benjamin/Cummings, California, 1989

5. TEXAS INSTRUMENTS, "*TMS320C30 C Compiler, reference guide*", Digital signal processor products, U.S.A., 1990.

6. TEXAS INSTRUMENTS, "*TMS320C30 Assembly Language Tools, User's guide*", Digital signal processor products, U.S.A., 1988.

7. TEXAS INSTRUMENTS, "*Third-generation TMS320 user's guide*", Digital signal processor products, U.S.A., 1989.

8. WORTHINGTON, T.K. and al. "*IBM dynamic signature verification*", Computer security, J.B. Grimson and H.J. Kugler, North Holland, 1985, pp. 129-54.

9. PARIZEAU, M., PLAMONDON, R., "*La corrélation régionale, la comparaison par programmation dynamique et la comparaison d'arbres; trois procédures pour mesurer la similitude des signaux*", Rapport technique, Ecole Polytechnique, janvier 1987.

10. PLAMONDON, R., PARIZEAU, M., "*Signature verification from position, velocity and acceleration signals: A comparative study*", Proc. of the 9th international conf. on pattern recognition, Rome, nov. 1988, p. 260-65.

11. EDMISTON, E. WAGNER, R.A. "*Parallelization of the dynamic programming for comparaison of sequence*", Conference on parallel processing, 1987.

12. MALINOWSKI, K. SADECKI, J. "*Dynamic programming: a parallel implementation*", Proceedings of the first European workshop, Manchester England, 1986, pp. 161-70.

13. RYTTER, W. "*On efficient parallel computations for some dynamic programming problems*", Theor. comput. sci., Nertherlands, Vol. 59, no. 3, aout 1988, pp. 297-307.

14. VELDHORST, M. "*Parallel dynamic programming algorithms*", CONPAR 86, Conference on algorithms and hardware for parallel processing, Aachen, West Germany, 1986, pp. 393-402.