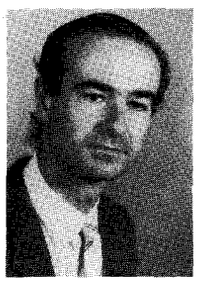
Le calculateur parallèle CAPITAN : 600 MIPS pour l'imagerie temps réel (1)

The CAPITAN parallel processor: 600 MIPS for use in real time imagery

Gérard GAILLAT



MATRA, Division Espace, Produits et Technologie,
37, avenue Louis-Bréguet, 78140 VELIZY

Gérard GAILLAT est ancien élève de l'École Polytechnique et de l'École Nationale Supérieure des Télécommunications. Il a également soutenu une thèse de 3e cycle à l'Université de Paris-VI, sous la direction du professeur Jean-Claude SIMON. Durant 6 ans passés au Laboratoire Central de Recherches de Thomson, il a travaillé principalement dans les domaines de la reconnaissance des caractères, du traitement des images et de l'intelligence artificielle. Il a rejoint le groupe Matra en 1980 où il a dans un premier temps été responsable de l'architecture informatique du centre d'exploitation des images pour le satellite *SAMRO*. Il dirige depuis 1982 le développement du calculateur parallèle CAPITAN qui est décrit dans cet article. Il est également professeur à l'École Nationale Supérieure des Techniques Avancées, chargé de l'enseignement de reconnaissance des formes.

RÉSUMÉ

L'architecture du calculateur parallèle CAPITAN est décrite dans cet article, notamment sa structure d'échange entre processeurs et avec les mémoires. Sa capacité d'accueil des circuits présents et futurs de traitement du signal est en particulier mise en valeur. Ses performances sont ensuite analysées dans le cas de l'utilisation du processeur TMS-320 comme élément de base. Il en ressort que CAPITAN peut soit être couplé à un calculateur hôte et se comporter comme un calculateur vectoriel de très hautes performances, soit constituer, en tant que logique flexible, le cœur d'une chaîne temps réel de traitement d'image. L'article se termine en passant en revue un certain nombre de ses applications potentielles. Pour chacune d'entre elles, une configuration est proposée et analysée. La variété des exemples choisis, empruntés aux domaines du traitement d'image, du signal, de la reconnaissance des formes et de la résolution d'équations aux dérivées partielles illustre la souplesse d'adaptation de la machine et démontre son aptitude à concurrencer les systèmes câblés au regard des critères volume, performance, puissance dissipée.

MOTS CLÉS

Calculateur parallèle MIMD, bus en anneau, reconfigurabilité, structure d'accueil pour microprocesseurs de traitement du signal et pour CITGV, traitement du signal, traitement des images, équations aux dérivées partielles.

SUMMARY

The architecture of the CAPITAN parallel processor is described in this paper, in particular its processor memory interconnection network. Its ability to incorporate present and future signal processing chips is emphasized. Its performances are then analyzed assuming TMS-320 as basic processing element. It comes out that CAPITAN can be either linked to a host computer and act as a very high performances array processor or constitute, as flexible hardware, the heart of a real time image processing device. Finally some potential applications are analyzed. For each one, a configuration is proposed and discussed. The wide range of chosen examples which are taken from image and signal processing, pattern recognition and partial derivative equation fields illustrates machine adaptability and demonstrates that it is a cost and power consumption effective alternative to application dedicated hardware.

KEY WORDS

Parallel MIMD processor, ring bus, reconfigurability, welcome structure for digital signal processors and for VHSIC's, signal processing, image processing, partial derivative equations.

(1) Une version préliminaire de cet article a été publiée dans les actes du 1er *colloque image*, Biarritz, 1984, version corrigée de l'article reçue le 10 octobre 1984.

TABLE DES MATIÈRES

1. Introduction

L'existence aujourd'hui, sur le marché, de processeurs intégrés de traitement du signal permet de réaliser des architectures programmables dont les caractéristiques en performances, en volume ou en puissance dissipée, sont tout à fait comparables à celles d'architectures plus spécialisées. Par contre, leurs capacités d'adaptation sont incomparables. Le calculateur parallèle CAPITAN ([2]) développé par MATRA ([3]) est une structure d'usage général capable d'accueillir un grand nombre de ces processeurs pour les faire coopérer dans le cadre de traitements parallélisables à très haut débit. Une telle structure permet d'abaisser considérablement le coût de développement et d'industrialisation d'un système temps réel de traitement des images, de traitement du signal, de reconnaissance des formes, de résolution d'équations aux dérivées partielles, de calcul matriciel, ...

L'exemple des applications recensées par l'Agence Spatiale Européenne [1] est caractéristique des capacités d'adaptation de CAPITAN : Dans le cadre des missions d'observation de la terre, plusieurs applications nécessitent un traitement plus ou moins élaboré sur un flot de données qui se chiffre en millions, voire en dizaines de millions d'octets par seconde. On peut citer principalement la reconstructions d'images à partir du signal émis par le radar à ouverture synthétique et la compression d'images mono ou multispectrales. Pour chaque application, plusieurs algorithmes et de nombreuses variantes sont possibles. Le choix tient compte des paramètres que l'utilisateur veut optimiser. Le but du calculateur CAPITAN est de fournir une architecture unique, suffisamment générale mais aussi suffisamment efficace pour s'adapter à différentes applications, différents algorithmes, différents débits, différentes caractéristiques de l'instrument d'acquisition, ...

En fait, la généralité de l'architecture la rend utilisable dans des domaines qui dépassent très largement le domaine de l'imagerie spatiale embarquée. La variété des exemples choisis dans la suite de cet article en donnera un premier aperçu.

Nous allons commencer par décrire brièvement l'architecture du calculateur. Le lecteur intéressé par une description plus approfondie ou par une justification des choix architecturaux pourra se reporter à [2] ou à [3]. Nous évoquerons ensuite la mise en œuvre de ce calculateur, soit en temps que logique flexible dans une chaîne de traitement temps réel, soit en temps que calculateur vectoriel couplé à un calculateur hôte. Puis nous terminerons en passant en revue un certain nombre d'applications et en analysant pour chacune d'entre elles les possibilités de mise en œuvre, tant au niveau matériel qu'au niveau logiciel.

2. Description

2.1. Principe d'architecture

Le calculateur parallèle CAPITAN apparaît comme une structure d'accueil, de type MIMD [4] permettant à un nombre éventuellement élevé de processeurs identiques ou différents de coopérer dans le cadre d'un travail nécessitant une forte puissance de calcul. Comme l'indique la figure 1, ces processeurs sont reliés entre eux, avec les mémoires globales et avec les périphériques d'entrée-sortie par une structure de communication. Les abonnés de cette structure peuvent être des microprocesseurs d'usage général tels que le 8086 ou le 68000, des processeurs de traitement du signal tels que le TMS-320 ou le NEC 7720, des circuits ou des logiques spécialisées dans telle opération de traitement du signal (FFT par exemple), des mémoires, des périphériques, voire un calculateur hôte universel. A terme, l'ambition du calculateur CAPITAN est de constituer une véritable structure d'accueil pour les circuits intégrés à grande vitesse qui naîtront des plans VHSIC américain, cinquième génération japonais ou CITGV français.

A titre indicatif, les abonnés aujourd'hui disponibles comprennent notamment :

– un abonné 68000 doté de 256 KO de mémoire propre et d'un opérateur flottant cablé;

– un abonné grappe de processeurs de signal qui regroupe quatre ou huit TMS-320;

([2]) Le sigle CAPITAN signifie : CAlculateur Parallèle pour l'Imagerie, le Traitement du signal et l'Analyse Numérique.

([3]) Le développement a été partiellement financé par l'Agence Spatiale Européenne sous le contrat 4960/81/NL/HP(SC). Le nom initial du projet était HSPP : High Speed Programmable Processor.

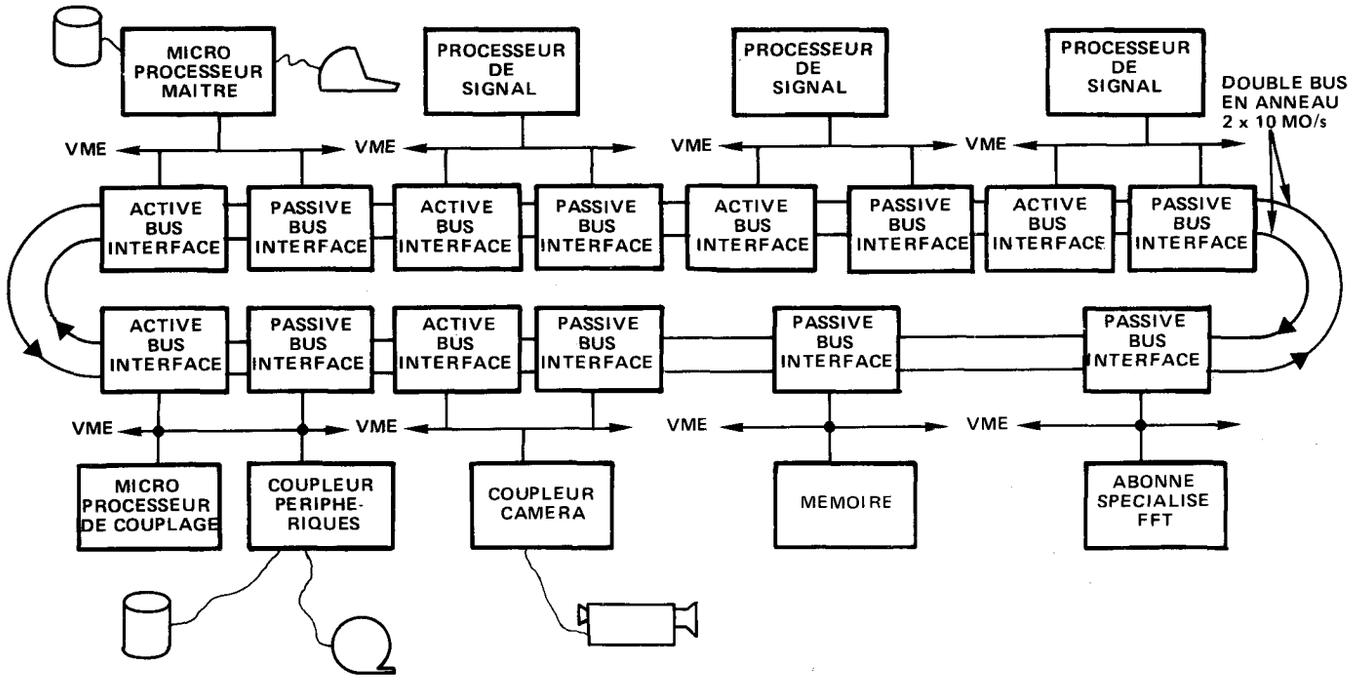


Fig. 1. – Architecture générale du calculateur CAPITAN.

– des abonnés mémoires (toute carte compatible VME);

– un abonné mémoire d'image assurant le couplage à une caméra ou un moniteur;

– des abonnés coupleurs assurant une liaison haut débit avec les calculateurs Vax ou Sel ou avec le bus Ethernet;

– et plus généralement toutes les cartes commercialisées sur le bus VME (*voir* § 2.4).

2.2. Structure de communication

La structure de communication se présente sous la forme de deux bus en anneau reconfigurables. Chacun des deux bus en anneau parcourt tous les abonnés.

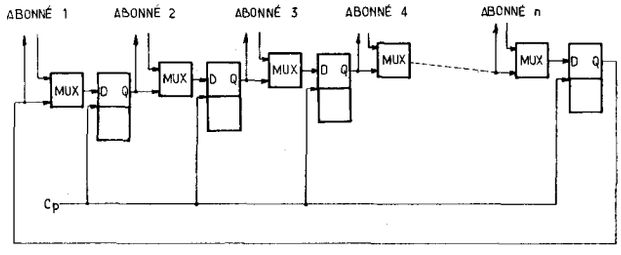


Fig. 2. – Principe d'un bus en anneau.

La figure 2 illustre la notion de bus en anneau. Un tel bus peut être considéré comme un ensemble de cases qui se déplacent toutes d'un cran à chaque coup d'horloge. Si un abonné veut envoyer un message à un autre abonné, il doit attendre le passage d'une case vide et y déposer le message. Au bout de n coup d'horloge, ce message passera devant le destinataire qui pourra le lire. Le protocole retenu sur ce bus est décrit dans [5] et [6]. Il a l'avantage de gérer le flot des échanges de manière à éviter toute collision au niveau des accès mémoires, afin de tirer parti au maximum de la capacité d'un bus en anneau. De ce fait, le débit effectif du bus au voisinage de la saturation est égal à son débit nominal, soit 10 MO/s par bus dans le cas de CAPITAN.

Muni de ce protocole, le bus en anneau apparaît, vu de l'abonné, très semblable à un bus conventionnel trois états. Aux performances près, il est capable d'assurer les mêmes services : ordres de lecture, d'écriture, signal de priorité. Son intérêt essentiel tient à ses performances : avec le protocole choisi, pour une même fréquence d'horloge, un bus en anneau écoule exactement le même débit qu'un bus trois états. Par contre, pour des raisons électriques, il est possible dans une même technologie d'utiliser une horloge environ quatre fois plus rapide sur un bus en anneau. A titre d'exemple, l'horloge qui pilote les deux bus de CAPITAN fonctionne à 50 ns pour 16 abonnés en technologie TTL. En comparaison, la réalisation d'un bus trois états reliant 16 abonnés et fonctionnant à 200 ns nécessite déjà beaucoup de soins et des circuits dissipant bien plus de puissance. En d'autres termes, un bus en anneau permet d'assurer, soit plus de débit sur le même nombre de fils, soit un débit donné sur un moindre nombre de fils. Il se prête en outre bien mieux à une réalisation intégrée (*voir* § 5).

2.3. Disponibilité et reconfiguration fonctionnelle

En apparence, la notion de bus en anneau semble très vulnérable. Le bus est inutilisable si un seul de ses maillons est en panne. Pour éviter cet écueil, nous avons utilisé une technique de fiabilisation d'un bus en anneau qui est décrite dans [6] et que l'on peut, en première approximation illustrer par la figure 3. Elle consiste à doubler le bus et à prévoir des mécanismes permettant de refermer la boucle quel que soit l'abonné défaillant. Ainsi, tout abonné en panne peut-il être totalement isolé électriquement du reste des abonnés. Le système ne sera pas perturbé quoiqu'il émette, ou même s'il met des fils du bus en court-circuit.

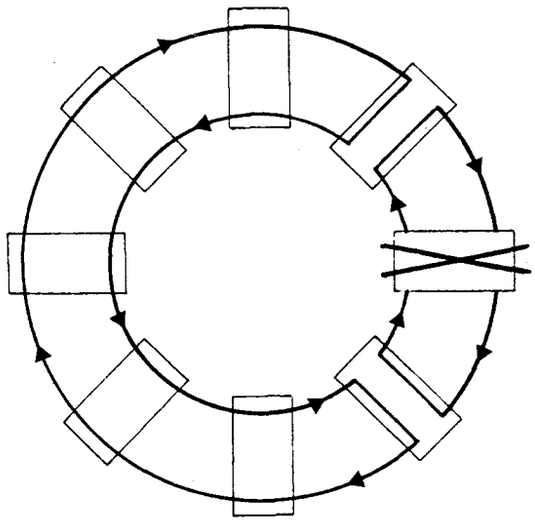


Fig. 3. — Fiabilisation d'un bus en anneau.

La sécurité ainsi offerte est sensiblement plus grande que ce qu'il est possible de réaliser avec un bus trois états. Il est à noter que chacun des deux bus est ainsi fiabilisé de manière indépendante de l'autre. L'utilisateur dispose donc bien de deux bus fiabilisés à 10 MO/s chacun.

Outre son apport sur le plan de la disponibilité des bus, cette technique permet de reconfigurer sans aucune modification matérielle un bus en autant de sous-bus que désiré. Chacun des sous-bus ainsi obtenus achemine le même débit que le bus initial, mais à travers un nombre plus restreint d'abonnés. Ainsi, si le premier bus est cassé en 3 et le deuxième en 4, le processeur est

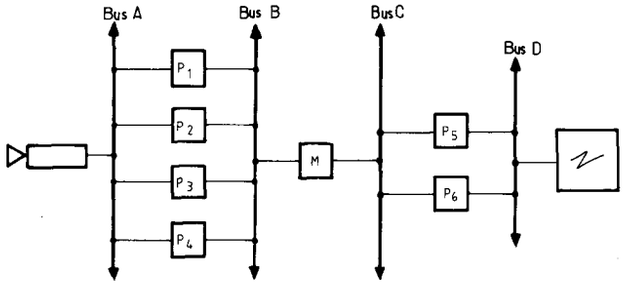


Fig. 4 *a*. — Configuration type pipe-line.

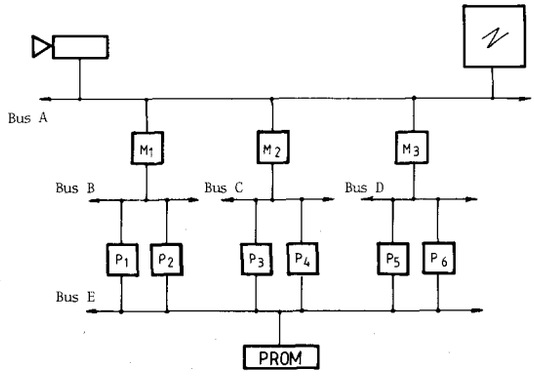


Fig. 4 *b*. — Processeurs partageant une PROM.

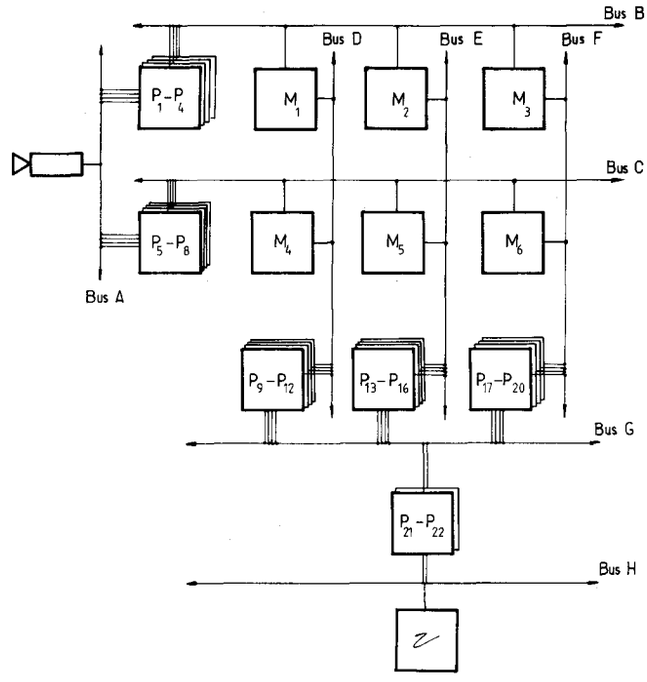


Fig. 4 *c*. — Architecture bidimensionnelle (exemple du SAR).

Légende des figures 4 *a*, 4 *b*, 4 *c* :

— le symbole P représente un processeur avec un ABI et/ou un PBI (*voir* § 2.4);

— le symbole M représente une mémoire avec un PBI;

— tous les bus sont des bus en anneau.

capable d'écouler un débit global d'échange de 70 MO/s, alors que sans cassure les deux bus n'auraient pu écoumer que 20 MO/s. Cela autorise la création d'une grande variété d'architectures spécialisées. Plusieurs d'entre elles ont été décrites dans [5] et sont illustrées par les figures 4 *a*, 4 *b*, 4 *c*. Le lecteur notera que dans chacune d'entre-elles, chaque abonné ne voit que deux bus. Simplement, chaque bus a été coupé en sous-bus et ne parcourt de ce fait plus tous les abonnés.

2.4. Description des modules ABI et PBI

La connexion des différents abonnés aux bus en anneau s'effectue grâce à deux modules. L'un appelé Active Bus Interface (ABI) permet à un abonné d'émettre des ordres de lecture ou d'écriture sur l'un ou l'autre des deux bus. L'autre, appelé Passive Bus Interface (PBI) permet à un abonné de recevoir ces ordres, et de retourner une réponse en cas d'ordre de lecture. Typiquement, les processeurs sont reliés au bus par des ABI, tandis que les mémoires sont reliées au bus par des PBI. Néanmoins, dans certains cas, il peut être intéressant de relier les processeurs aux bus par des PBI ou à la fois par des ABI et des PBI. La figure 1 en est un exemple. Nous en verrons d'autres exemples dans les applications.

De manière plus précise, chaque abonné processeur est organisé autour d'un bus compatible VME [7] indépendant des autres abonnés. Les cartes ABI et PBI contiennent des mémoires à double accès, un accès étant réservé au bus VME, l'autre aux bus en anneau. Ces mémoires sont destinées à contenir aussi bien les ordres de trans-

que les données. Vues du processeur, un module ABI ou PBI apparaît simplement comme un espace mémoire sur le bus VME.

Le rôle d'un ABI consiste à transférer à travers le bus en anneau des données depuis sa propre mémoire vers la mémoire d'un PBI (ordre d'écriture) ou à provoquer le transfert dans sa propre mémoire de données situées sur la mémoire d'un PBI (ordre de lecture). Dans les deux cas, le transfert se fait à l'initiative de l'ABI et selon les ordres de transfert contenus dans sa mémoire. Les transferts se font de préférence par bloc. Un bloc peut être à cheval sur plusieurs PBI. Trois types de transferts de bloc sont offerts par le module ABI :

– des transferts par bloc d'adresses séquentielles;

– des transferts par bloc dont les adresses successives sont distantes de 2^n (colonne d'une image);

– des transferts par bloc dont les adresses sont aléatoires et indiquées dans l'ordre de transfert.

Ainsi, le mécanisme ABI-PBI est-il capable d'effectuer non seulement des transferts, mais aussi un certain réagencement des données lors des transferts. Bien entendu, tout ce travail s'effectue en total parallélisme avec les activités des abonnés processeurs. Ceux-ci sont totalement libérés des tâches d'échanges et peuvent se consacrer à 100 % aux tâches de calcul.

3. Mise en œuvre et performances

Selon la nature des prestations qui lui sont demandées, le calculateur parallèle CAPITAN peut apparaître soit comme un calculateur vectoriel couplable à un calculateur hôte universel, soit comme une logique flexible intégrable dans une chaîne de traitement temps réel.

3.1. Configuration standard type calculateur vectoriel

La configuration standard type calculateur vectoriel se caractérise par le fait qu'elle contient seulement deux bus en anneau et, que chacun de ces bus parcourt tous les abonnés. Elle est illustrée par la figure 1. Un système opératoire est en cours de développement sur cette configuration. Celui-ci suppose en outre que chaque abonné, processeur ou calculateur hôte, est relié aux bus en anneau par un module ABI et un module PBI. De ce fait, ces abonnés peuvent à la fois prendre l'initiative d'envoyer messages ou requêtes et répondre aux initiatives d'autres abonnés. Des abonnés mémoire peuvent également être inclus. Le nombre maximal de modules ABI ou PBI est de 32. Cela limite à 15 le nombre d'abonnés processeurs pouvant être accueillis. Si l'on considère le cas des abonnés grappe de huit TMS-320 précédemment évoqués, la puissance de calcul s'élève alors à 600 Minstructions par seconde (600 MIPS). En effet, chaque TMS-320 exécute une instruction toutes les 200 ns, soit 5 MIPS. Une grappe de 8 fournit 40 MIPS et 15 grappes fournissent 600 MIPS. Bien entendu, cette puissance n'est atteinte que si l'algorithme présente un degré de parallélisme suffisant pour fournir à tout instant à chacun des 120 processeurs une tâche indépendante. C'est heureusement très souvent le cas dans les domaines d'application cités en introduction.

Dans le cas où les 20 MO/s disponibles sur les deux bus en anneau se répartissent équitablement entre données d'entrée et de sortie, les 600 MIPS correspondent à une moyenne de 60 instructions par octet traité. En configuration maximale, la machine est donc adaptée à des calculs de complexité moyenne ou élevée. Pour des calculs de faible complexité, la machine est limitée par ses capacités d'échange si bien que l'adjonction de processeurs supplémentaires n'augmente plus la puissance de calcul. On notera cependant, si l'on tente une comparaison avec d'autres machines, que les 20 MO/s ne concernent que les échanges entre abonnés. La capacité totale d'échange de la machine est en fait beaucoup plus forte si l'on y ajoute les capacités des bus VME abonnés (10 MO/s chacun) et les capacités de dialogue entre les processeurs de signal et leurs mémoires privées.

3.2. Configuration type logique flexible

Si dans le cadre d'une application déterminée, la puissance de calcul ou le volume total d'échange entre abonnés dépassent les limites de la configuration standard, il reste la solution de construire une configuration spécialisée. On peut alors accroître considérablement ces limites et dépasser largement le giga-instruction par seconde ou les 50 MO/s. En effet, les modules de base qui composent CAPITAN peuvent être agencés de multiples manières par simple modification du fond de panier, sans conception de nouvelles cartes logiques. Par exemple, les abonnés processeurs peuvent n'être couplés au bus que par des modules de type PBI. Ils apparaissent alors comme des mémoires intelligentes. L'utilisateur y dépose un bloc de données à traiter, attend un peu, teste un drapeau de fin de tâche et va récupérer les résultats. Ils peuvent aussi n'être couplés au bus que par des modules de type ABI. Le nombre de bus peut également dépasser deux et l'on peut construire des structures pipe-line ou bidimensionnelles ou mixtes ou ... (voir *fig.* 4*a*, 4*b*, 4*c*). Un abonné peut même, au besoin, être relié à plus de deux bus. En effet, les ABI et PBI apparaissant comme des mémoires pour le bus VME, rien n'interdit de placer plusieurs ABI et/ou plusieurs PBI sur le bus VME d'un même abonné. Cela lui permet de dialoguer directement avec plus de deux bus en anneau. L'abonné caméra de la figure 12 en est un exemple.

En outre, il peut être intéressant de construire des abonnés spécifiques d'une application. Par exemple, en traitement SAR, si 75 % des tâches de calcul sont des FFT et si le nombre de processeurs est élevé, il peut être intéressant de construire des abonnés FFT afin de diviser par 4 le nombre de processeurs. Ce point est analysé dans [8].

3.3. Exemple de mise en œuvre d'une application

Pour illustrer ce qui vient d'être dit, prenons l'exemple d'une restauration géométrique d'image et essayons successivement de voir :

– comment les tâches peuvent être réparties et synchronisées entre les différents processeurs;

– quel volume de matériel est nécessaire et pour quelles performances;

– comment la configuration peut être réduite à performances voisines si l'on accepte de spécialiser l'architecture.

La restauration géométrique consiste à partir d'une image ayant subi des distorsions à recréer par rééchantillonnage une nouvelle image aux propriétés géométriques améliorées. Typiquement, en imagerie satellite, il s'agira de construire une nouvelle image superposable à une carte ou à une autre image.

La rotation d'une image est un cas particulier de restauration géométrique.

Pour obtenir une bonne qualité d'image, il faut choisir une interpolation bicubique qui consiste à calculer un point de coordonnées réelles en fonction de ses 16 plus proches voisins de coordonnées entières. Le calcul de chaque point peut se faire en deux passes (voir *fig.* 5).

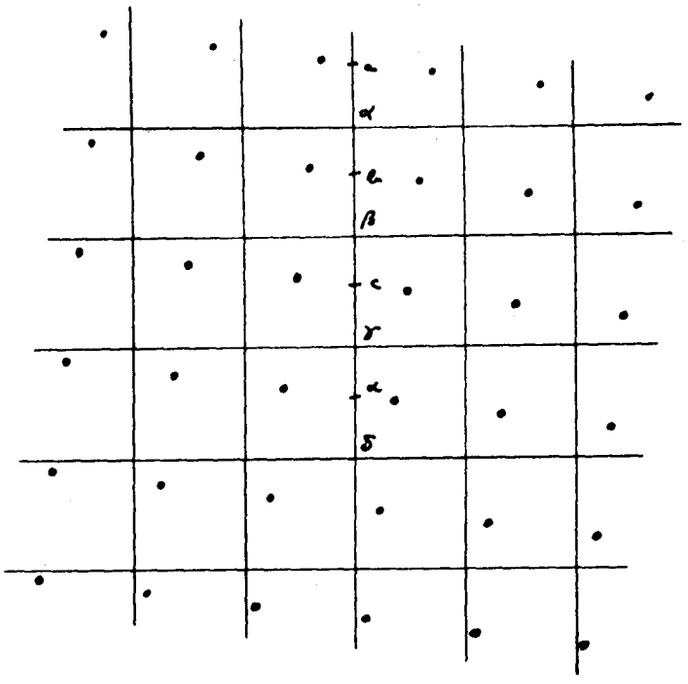


Fig. 5. – Interpolation géométrique : il faut former la nouvelle grille (α, β, γ, ...) à partir de l'ancienne.

Une première passe dans laquelle on rééchantillonne chaque ligne afin d'obtenir les points de type *a*, *b*, *c*, *d* une deuxième passe dans laquelle on rééchantillonne les points de chaque nouvelle colonne pour obtenir les points de type α, β, γ, δ.

Une image de taille $1\,000 \times 1\,000$ peut être traitée sur une configuration standard, telle que celle de la figure 6.

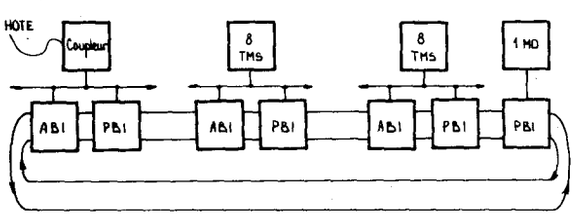


Fig. 6. – Restauration géométrique d'images à 2 MO/s.

La solution la plus élémentaire consisterait à faire charger l'image en mémoire par le calculateur hôte, à lui faire subir le rééchantillonnage ligne, puis le rééchantillonnage colonne et enfin à la faire relire par le calculateur hôte. Il est en fait préférable d'effectuer le rééchantillonnage ligne « au vol » lors du chargement de la ligne, et le rééchantillonnage colonne « au vol » lors du déchargement de la colonne. En effet, si t_e est le temps de chargement, t_s le temps de déchargement et t_c le temps de calcul, la durée totale de l'opération vue du calculateur hôte est de $\sup(t_e+t_s, t_c)$ alors qu'elle serait de $t_e+t_c+t_s$ dans le premier cas.

De plus on économise ainsi du débit sur le bus. Le seul inconvénient de cette technique est qu'une image chargée par ligne est déchargée par colonne ce qui correspond à une transposition de l'image.

On aurait pu éliminer cet effet en conservant un rééchantillonnage ligne au vol, mais en séparant le rééchantillonnage colonne et le déchargement.

Les tâches à accomplir pour le traitement de la *i*-ième ligne sont :

T1*i* : le calculateur hôte écrit la ligne *i* sur son PBI;

T2*i* : l'ABI d'un TMS acquiert la ligne *i*;

T3*i* : ce TMS traite (rééchantillonne) la ligne *i*;

T4*i* : l'ABI de ce TMS écrit la ligne *i* en mémoire.

Les tâches à accomplir pour le traitement de *j*-ième colonne sont :

T5*j* : l'ABI d'un TMS relit en mémoire la colonne *j*;

T6*j* : ce TMS traite (rééchantillonne) la colonne *j*;

T7*j* : l'ABI de ce TMS envoie la colonne *j* au PBI du calculateur hôte;

T8*j* : le calculateur hôte lit la colonne *j* sur son PBI.

Les flèches sur la figure 7 indiquent les relations d'antériorité entre ces tâches. Elles traduisent en particulier le fait que le traitement des lignes est totalement parallélisable, le traitement des colonnes aussi, et que le premier doit être terminé avant que commence le second.

L'allocation des tâches aux processeurs peut être envisagée de manière statique ou dynamique.

Une allocation dynamique suppose une compétition entre processeurs pour s'approprier une nouvelle tâche. Sur CAPITAN, les conflits engendrés par cette compétition peuvent être arbitrés soit à l'aide des sémaphores dont dispose la machine, soit en prévoyant un microprocesseur chef d'orchestre chargé d'allouer les tâches.

Pour l'application présente, dans la mesure où les tâches sont toutes de longueur fixe et identique une allocation statique des tâches est préférable car elle minimise les tâches non productives. Dans ce cas, s'il y a n processeurs, le p-ième d'entre eux exécutera les tâches p, $p+n$, $p+2n$, $p+3n$, ...

Il y aura donc une seule compétition entre processeurs et celle-ci aura lieu à l'initialisation de la machine ou à la reconfiguration après une détection de panne. Elle consistera, pour chacun des processeurs mis sous tension à s'attribuer un numéro p différent. On peut concevoir que le mécanisme ayant décidé la reconfiguration aura également la charge d'attribuer ce numéro. Cela suppose une certaine intelligence de sa part et donc la présence d'un microprocesseur en son sein. Sinon la lecture et l'incrémentation de la variable p doit être assurée par

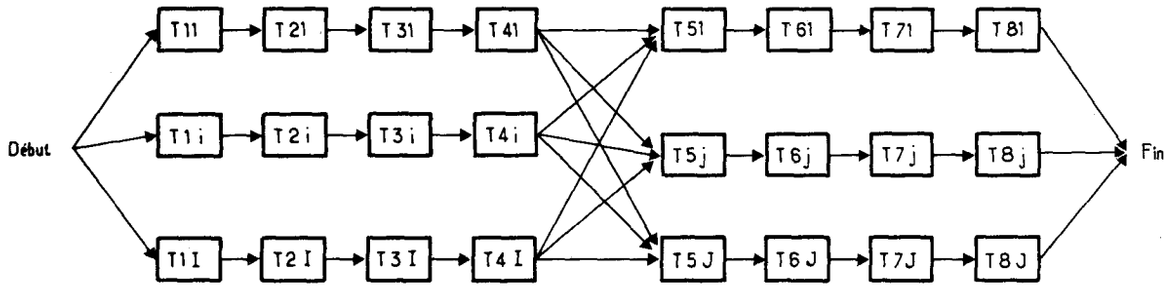


Fig. 7. — Diagramme d'antériorité des tâches.

les processeurs eux-mêmes. Elle doit dans ce cas faire l'objet d'une séquence en exclusion mutuelle, gardée par un sémaphore dont chaque processeur cherchera à s'emparer dès l'initialisation.

La synchronisation entre les tâches peut être faite par un échange des messages adaptés à un processus de type producteur consommateur. Par exemple, la tâche *t3i* enverra à celle qui la précède, soit *t2i*, un message indiquant qu'elle a consommé l'information produite par celle-ci et enverra à celle qui la suit, soit *t4i*, un message indiquant qu'une entité est disponible à son intention.

Les flèches verticales sur la figure 8 *a* matérialisent les messages qui sont échangés entre les différentes tâches pour le traitement de la ligne *i*. Le principe est le même pour le traitement des colonnes. La seule difficulté se situe au niveau de la synchronisation entre T4 et T5 : le traitement des colonnes ne peut commencer que lorsque tous les traitements des lignes sont terminés.

Pour assurer cette antériorité, lorsqu'un processeur a traité toutes ses lignes, il diffuse un message général pour tous les autres processeurs et ne s'autorise à démarrer le traitement des colonnes que lorsqu'il a reçu le message émanant de tous les autres processeurs.

L'efficacité maximale ne peut être obtenue que si l'on prévoit un schéma de recouvrement des tâches qui autorise un taux d'occupation de 100 % des processeurs. Cela peut être obtenu par une technique de pipe-line : pendant qu'un TMS traite la ligne *i* (tâche *t3i*), son ABI décharge la précédente (tâche $t4i-n$), charge la suivante (tâche $t2i+n$) et le calculateur hôte charge la ligne $i+2n$ (tâche $t1i+2n$).

On obtient ainsi, en régime continu une situation telle qu'illustrée par la figure 8 *b*. Cette technique de pipe-line nécessite de doubler tous les tampons mémoire. Ainsi, quatre tampons doivent cohabiter sur l'ABI d'un TMS :

- un tampon E1 où le TMS lit la ligne *i*;
- un tampon S1 où il écrit la ligne *i*;
- un tampon E2 où l'ABI charge la ligne $i+n$;
- un tampon S2 d'où l'ABI décharge la ligne $i-n$.

Après le traitement d'une ligne, et après réception des messages émanant des tâches amont et aval, les rôles de E1, S1 et de E2, S2 sont permutés.

Dans ces hypothèses, l'évaluation des performances de la configuration de la figure 6 peut se faire en évaluant la charge sur les TMS puis en vérifiant qu'il n'y a pas d'autre goulot d'étranglement. Lors du rééchantillonnage ligne, le calcul d'un nouveau point sur un TMS nécessite le calcul d'une fonction de distorsion, le choix d'un jeu de quatre coefficients et d'un produit scalaire par le vecteur de ces quatre coefficients. Cela coûte 4 µs sur un TMS. Comme chaque pixel est l'objet d'un rééchantillon-

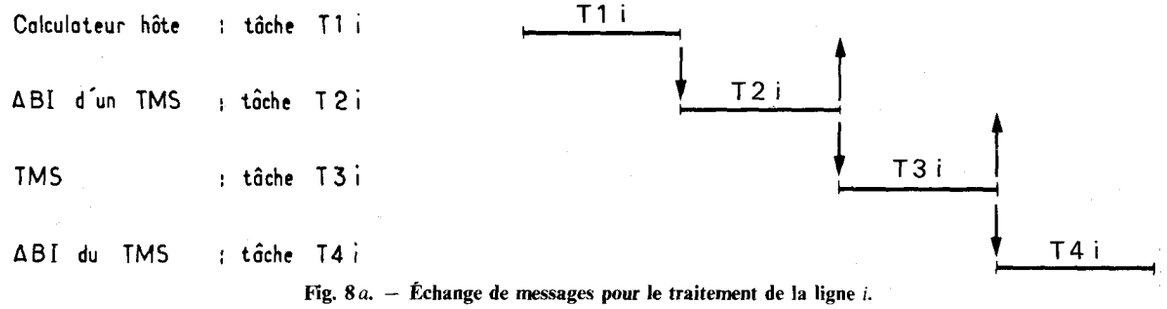


Fig. 8 *a*. — Échange de messages pour le traitement de la ligne *i*.

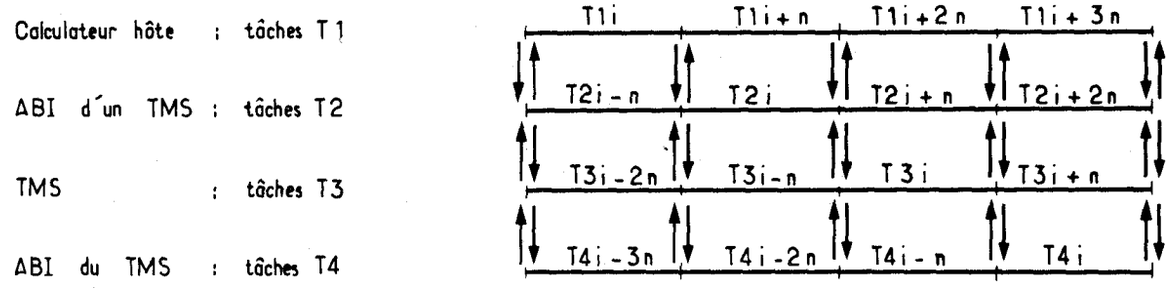


Fig. 8 *b*. — Recouvrement des tâches dans le temps.

nage ligne puis d'un rééchantillonnage colonne, le calcul d'un nouveau point représente 8 µs sur un TMS soit 0,125 Mpixel/s. Les deux grappes de huit processeurs de la figure 8 sont donc capables de traiter 2 Mpixel/s, ... à condition que le calculateur hôte sache les fournir et les récupérer.

Le débit engendré sur le bus de CAPITAN est alors de 8 MO/s. Il est composé de quatre sous-flux à 2 MO/s : calculateur hôte vers TMS, TMS vers mémoire, mémoire vers TMS et TMS vers calculateur hôte. C'est largement inférieur aux capacités d'échange de la configuration. Par contre, une configuration avec cinq grappes de huit processeurs traiterait 5 Mpixel/s et engendrerait un flot d'échange de 20 MO/s ce qui est la limite de la configuration standard. Il ne servirait à rien pour cette application et pour cette configuration de rajouter une sixième grappe.

Le lecteur a sans doute remarqué la dissymétrie qui existe dans cette application entre les rôles joués par le PBI du calculateur hôte et les ABI des TMS d'une part, l'ABI du calculateur hôte et les PBI des TMS d'autre part. Tandis que les premiers supportent la quasi-totalité des flux des échanges, les seconds n'interviennent que pour la transmission de quelques messages de synchronisation.

Dans d'autres applications, les rôles pourraient être inversés ou mieux équilibrés, ce qui incite à prévoir un ABI et un PBI par abonné dans toute configuration standard.

Par contre si l'on admet de spécialiser cette configuration pour cette application on peut aller plus loin et supprimer totalement l'ABI du calculateur hôte et les PBI des TMS. On réduit ainsi le volume et la consommation du matériel ce qui peut être utile dans une chaîne de traitement temps réel. La seule contrepartie est un léger accroissement de complexité pour la transmission de certains messages. Ainsi le calculateur hôte ne peut plus prévenir directement les TMS lorsqu'il a préparé une ligne ou lu une colonne. Il est contraint de laisser le message dans son PBI et les TMS doivent employer une technique de scrutation périodique pour tenter de le lire.

De même, les TMS ne peuvent se prévenir les uns les autres lorsque les rééchantillonnages des lignes sont terminés. Ils doivent envoyer le message dans une zone mémoire accessible de tous, puis tenter périodiquement de la relire. Si l'on compare à la situation d'une configuration standard, cela coûte un trafic supplémentaire sur le bus : sur la configuration standard, dans la mesure où chacun expédie directement ses messages chez le destinataire il n'y a jamais de boucle active de test d'événement sur le bus. Les échanges sont alors réduits à un minimum. Sur la configuration réduite, les messages qui ne peuvent être acheminés sont laissés sur place, à charge pour le destinataire de les scruter.

4. Autres exemples d'application

Nous allons maintenant passer en revue un certain nombre d'applications typiques du calculateur CAPITAN et montrer succintement pour chacune d'elles comment elle peut être réalisée et à quel volume de matériel elle conduit.

Le choix de ces applications cherche avant tout à illustrer la variété des problèmes qui peuvent être traités avec efficacité sur la machine.

4.1. Lecture optique de caractères

L'une des applications de la lecture optique des caractères est la lecture des adresses en tri postal. Il s'agit de lire le code postal, le nom de la ville et de tenter une correction contextuelle en comparant les deux. La zone à balayer pour extraire la dernière ligne de l'adresse est de 6,4 cm de hauteur sur 12 cm en largeur. Avec une résolution de 8 points par millimètre, cela représente une grille de 512×960 pixels. Chaque pixel doit être codé sur 8 bits, de manière à permettre l'usage d'algorithmes de seuillage adaptatif rendus nécessaires par les problèmes de faible contraste sur le courrier imprimé ou les enveloppes à fenêtre. La cadence typique d'un centre de tri est de 10 enveloppes par seconde, soit un volume à traiter de 5 MO/s. En outre, une enveloppe comporte au maximum 20 caractères à reconnaître.

La première étape (prétraitement) consiste à balayer cette zone, à en identifier les points noirs, à faire un filtrage local, et à construire le rectangle circonscrit à chacun des caractères. Un algorithme très simple est possible à ce niveau. Le traitement d'un pixel sur un TMS peut être évalué à 1,6 µs en moyenne. Il faut donc huit processeurs pour assurer ce traitement à la cadence de 5 MO/s.

La deuxième étape consiste à extraire un jeu de caractéristiques sur chaque caractère. Parmi les nombreuses techniques possibles à ce stade (*voir* [9]) nous considérerons la méthode de Komori [10]. Cette méthode préconisée par l'administration japonaise a l'avantage de donner des résultats remarquables, y compris sur les caractères manuscrits. Nos évaluations conduisent à un temps de calcul de 4,5 ms pour le traitement d'un caractère sur un TMS. Les 10 enveloppes par seconde représentant 200 caractères par seconde peuvent donc être traitées par un seul processeur.

La phase de décision peut se faire par une méthode de séparation linéaire. En utilisant la méthode préconisée dans [11], on devra effectuer par caractère environ 50 produits scalaires dans un espace de dimension 100, ce qui représente 2,5 ms sur un TMS. Ici encore, les 200 caractères par seconde peuvent être traités par un seul processeur.

La correction contextuelle consiste à comparer le code et le nom de la ville à une liste de 25 000 bureaux distributeurs. S'il n'y a pas d'erreur, une comparaison suffit. S'il y a un caractère rejeté, 10 ou 26 comparaisons sont nécessaires. S'il y a un caractère erronné, environ 400 comparaisons sont nécessaires. Dans tous les cas, même s'il y a plusieurs rejets, le temps de calcul est très faible, de l'ordre de quelques millisecondes au pire. Le processeur ayant assuré la décision est parfaitement capable d'absorber ce surplus de travail, d'autant plus qu'il n'y a que 10 enveloppes par seconde à traiter. La liste des 25 000 communes ainsi que la table de décision par apprentissage tient aisément dans une mémoire de 1 MO qui peut être locale à l'abonné qui les consulte.

Une configuration à 10 processeurs (voir *fig.* 9) est donc suffisante pour traiter cette application. Afin de ne pas saturer le bus VME de l'abonné prétraitement, nous avons préféré répartir ses huit processeurs en deux abonnés grappe de quatre processeurs au lieu d'une seule grappe de huit. Le dernier abonné est l'abonné extraction de caractéristique, décision et correction contextuelle. Il comprend deux processeurs qui disposent de la mémoire de 1 MO précédemment mentionnée. En outre, un coupleur ligne asynchrone est présent sur cet abonné pour l'envoir des résultats.

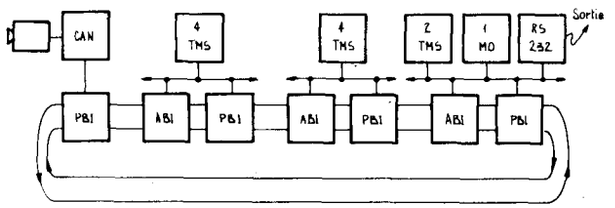


Fig. 9. – **Tri postal à 10 enveloppes par seconde 5 MO/s en entrée.**

4.2. Résolution d'équations aux dérivées partielles

L'exemple que nous choisirons pour illustrer la capacité de CAPITAN à résoudre des équations aux dérivées partielles est emprunté à l'électricité. Un champ de potentiel f sur un isolant est régi par l'équation :

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0.$$

Connaissant les contraintes aux limites, une méthode numérique pour résoudre l'équation consiste à itérer indéfiniment en remplaçant à chaque itération la valeur d'un pixel par la moyenne des valeurs de ses quatre voisins.

L'intérêt de l'exemple est double. D'une part, il implique un type de calcul que l'on rencontre très fréquemment en résolution d'équations aux dérivées partielles ou en traitement d'images : il s'agit de recalculer indéfiniment un pixel en fonction de la valeur de ses quatre (ou huit) voisins. Les méthodes de squelettisation [12] ou de relaxation probabiliste [13] sont basées sur ce schéma. D'autre part, on pourrait penser que ce type de calcul pose problème à CAPITAN. Il y a en effet beaucoup d'échanges de données entre tâches pour peu de calculs.

L'approche pour les traiter est néanmoins fort simple et mérite que nous nous y attardions un peu. Tout d'abord, il faut considérer des tâches qui consistent à traiter non pas un pixel, mais un groupe de pixels. Par exemple, un carré 32×32. Sur une image $1\,024 \times 1\,024$, le degré intrinsèque de parallélisme de l'algorithme est de 10^6, et même en le réduisant ainsi d'un facteur 1 000, il restera bien supérieur au nombre de processeurs d'une configuration. Ensuite, il faut traiter ces carrés comme des damiers composés de cases noires et de cases blanches : chaque itération se fera en deux passes. La première calculera les cases noires de tous les carrés à partir des cases blanches, la deuxième fera l'inverse. Moyennant quoi, les tâches de chaque passe sont totalement parallélisables.

Par contre, une passe doit être achevée avant d'entamer la suivante. C'est une situation très semblable à celle décrite pour le rééchantillonnage (*fig.* 7).

Dans ces conditions, une passe sur un carré 32×32 par un TMS coûte 1 ms. Une configuration de 16 processeurs partageant la mémoire d'image effectue donc une passe sur une image $1\,024 \times 1\,024$ en 64 ms et une itération complète en 128 ms, soit environ huit itérations à la seconde.

Si l'on place la mémoire $1\,024 \times 1\,024$ sur un abonné mémoire du bus, le volume d'échange processeur mémoire s'élève à 16 MO/s environ.

C'est acceptable s'il y a 16 processeurs mais cela signifie qu'il ne servirait à rien d'en mettre plus de 20. La capacité mémoire des PBI ne permet pas non plus de distribuer 1 MO sur deux abonnés. La solution que nous proposons (voir *fig.* 10) consiste à doter chaque abonné d'une mémoire locale de 512 KO sur laquelle nous placerons une demi-image par abonné. Seule une frange d'une ligne à la frontière de chaque demi-image sera dupliquée dans le PBI et régulièrement remise à jour par l'abonné, de manière à être lisible par l'autre abonné. Dans ces conditions, le débit d'échange entre abonnés chute à 32 KO/s. La configuration de la figure 10 comprend un troisième abonné qui est un coupleur haute vitesse avec un calculateur hôte. Les données d'entrée et les résultats en fin de calcul passent par lui.

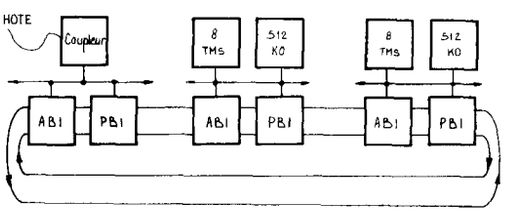


Fig. 10. – Équation aux dérivées partielles : *huit itérations par seconde sur zone* 1 000 × 1 000.

4.3 Compression d'image monospectrale

De nombreux algorithmes de compression d'image ont été publiés à ce jour. Cette variété s'explique pour une part par le fait qu'il s'agit d'un domaine en plein essor. Elle s'explique aussi et surtout par le fait que trois critères difficiles à concilier sont à optimiser : le taux de compression, la qualité image et la complexité algorithmique. Suivant ses objectifs, l'utilisateur privilégie l'un ou l'autre et préfère tel algorithme.

L'intérêt de CAPITAN est de proposer une architecture unique qui s'adapte à de nombreux cas. Reconnaissons cependant que dans le cas de codages très simplistes comme les DPCM qui ont été conçus pour réduire la complexité algorithmique, aux dépens des deux autres critères, les réalisations en matériel câblé conduisent à de meilleures performances.

Nous prendrons comme exemple typique celui de l'algorithme de compression monospectral conçu par l'Agence Spatiale Européenne [14]. Cet algorithme permet, pour une complexité raisonnable, d'obtenir un excellent rapport qualité-taux de compression. Il a notamment le mérite, par rapport au DPCM de ne pas déplacer les fronts radiométriques, ce qui est important pour les usagers utilisant la vision stéréo. Très sommairement, il

consiste à découper l'image en carrés 8×8, à effectuer une transformée cosinus sur chaque carré et à coder chaque composante spectrale sur un nombre variable de bits.

Vu l'indépendance complète des traitements sur chaque carré, la mise en œuvre sur CAPITAN de cet algorithme ne pose pas de problèmes. Les performances sont par contre intéressantes, notamment dans le cadre d'applications spatiales embarquées, dans la mesure où elles démontrent que CAPITAN permet la mise en œuvre d'algorithmes de compression efficaces avec des puissances électriques raisonnables. Le calcul de la transformée cosinus bidimensionnelle d'un carré 8×8 sur un TMS 320 coûte 415 µs. En y ajoutant la partie codage (concaténation des bits) la compression d'un carré 8×8 coûte 580 µs. Un TMS 320 est donc capable de traiter 0,11 Mpixel/s.

Pour assumer le débit d'un satellite tel que SPOT, soit 4,6 Mpixel/s, il faut 42 de ces processeurs. Ceux-ci peuvent être répartis en six grappes de sept processeurs (plus éventuellement une septième pour assurer la disponibilité). Voir *fig.* 11.

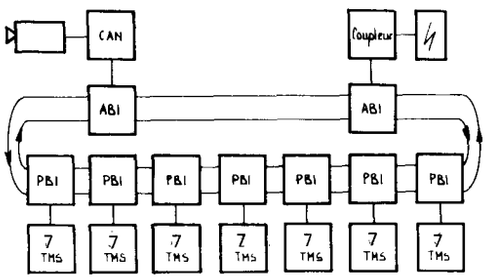


Fig. 11. – Compression monospectrale. Satellite *SPOT* : 4,6 MO/S.

On notera que chaque grappe est seulement dotée d'un PBI : les abonnés processeur apparaissent donc comme une grande mémoire délocalisée écrite par l'organe d'entrée et lue par l'organe de sortie. Lorsque les modules ABI et PBI seront miniaturisés, chaque grappe consommera 9 W (*voir* [8]). L'ensemble des six grappes actives consommera donc 54 W, ce qui est tout à fait envisageable à bord. S'il fallait réduire encore, on pourrait remplacer la transformée cosinus par une transformée de Hadamard, mais au prix d'une détérioration du taux de compression ou de la qualité image.

4.4. Compression d'image de télévision numérique couleur

Si l'on utilise pour la compression d'image de télévision numérique couleur un algorithme de compression d'images multispectrales tel que celui décrit en [15], le traitement de chaque bloc de huit lignes est une tâche indépendante. La mise en œuvre serait donc très proche du cas précédent si le débit d'échange ne posait un problème nouveau. En effet, si l'on respecte la norme CCIR dite 4.2.2 [16], le débit moyen d'entrée est de 20,2 MO/s, réparti à raison de 50 % pour la luminance et 25 % pour chacune des deux valeurs de chrominance. A ceci, il convient d'ajouter, dans le cas d'une compression à 32 Mb/s, un débit de sortie de 4 MO/s. Dans ces conditions, on arrive à un trafic global de 25 MO/s qui dépasse les capacités du double bus en anneau.

La solution que nous proposons (*fig.* 12) consiste à casser l'un des deux bus en anneau en trois sous-bus et à répartir le flot d'entrée sur chacun de ces trois bus. Rappelons que la cassure d'un bus peut se faire par logiciel sur la configuration standard, sans nécessiter le dessin d'un nouveau fond de panier. Le deuxième bus est affecté à la sortie. Il n'a pas été cassé puisqu'il ne doit acheminer que 4 MO/s. Pour varier par rapport à l'exemple précédent, nous avons cette fois relié les processeurs aux bus par des ABI et les organes d'entrée-sortie par des PBI. Les processeurs ont donc la charge de lire les données sur la mémoire d'entrée et d'expédier les résultats vers la mémoire de sortie. Nos études du problème conduisent au chiffre de 96 processeurs pour assumer le temps réel. Nous pouvons les répartir en 12 grappes de 8. L'organe d'entrée écrit huit lignes sur un PBI puis passe au suivant et ainsi de suite. Six PBI sont nécessaires pour l'entrée. La sortie est assurée au travers d'un seul PBI, accessible depuis tous les processeurs à travers le bus qui n'a pas été cassé.

Remarquons que l'on peut transposer cet exemple dans le cas d'un compresseur d'image multispectral embarqué à bord de satellite. Il s'agirait d'un compresseur transformant un flot d'entrée de 166 Mb/s en un flot de sortie de 32 Mb/s. La consommation de la configuration de la figure 12, instrument et antenne non compris peut être estimée à 140 W, si l'on dispose de modules ABI et PBI intégrés. Il s'agit d'un chiffre acceptable dans le cas de missions futures.

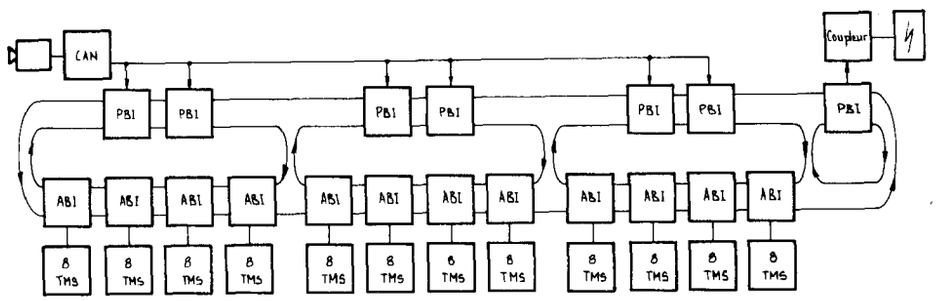


Fig. 12. – Compression télévision couleur : 20,7 MO/s en entrée. 32 MB/s en sortie.

4.5. Traitement Sar en temps réel

L'intérêt d'effectuer à bord la reconstruction d'image à partir du signal émis par le radar à ouverture synthétique tient au fait que la reconstruction est en soi une compression. L'exemple d'une chaîne bord construite à partir de CAPITAN est analysé dans [8] et illustré par la figure 13.

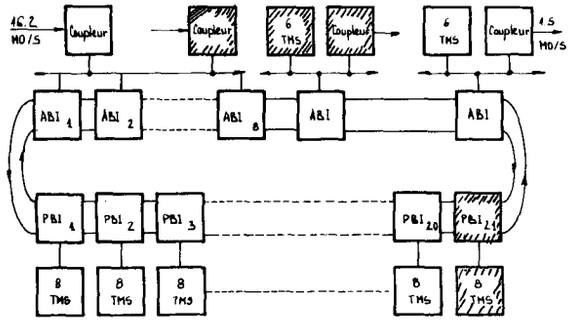


Fig. 13. – Traitement Sar sur satellite *ERS-1* en hachuré les éléments redondants.

Commentons-le brièvement : il suppose que le traitement en distance est effectué sur une chaîne analogique. A la sortie de cette chaîne, le débit d'entrée pour CAPITAN est de 16,2 MO/s. En sortie de CAPITAN, le débit est de 1,5 MO/s. Toutefois, comme pratiquement aucun autre flux d'échange interabonnés n'est nécessaire, la structure à deux bus suffit à écouler le débit. Huit modules ABI dont un de réserve sont nécessaires pour écouler le débit d'entrée. 21 grappes de huit TMS dont une de réserve sont nécessaires pour le traitement en azimuth. Un seul ABI dupliqué pour des raisons de disponibilité est nécessaire pour écouler le débit de sortie. Une grappe de six TMS localisée sur le même abonné assure une restauration géométrique de l'image sortante. Les 21 grappes effectuant les traitements en azimuth ne sont équipées que de PBI. En fait, la main corner turning memory (MCTM) et la re-corner turning memory (RCTM) sont délocalisées sur ces 21 abonnés. Les ABI d'entrée écrivent la première, l'ABI de sortie lit la deuxième. Le traitement en azimuth consiste pour chaque processeur à lire son fragment de MCTM, à le traiter et à écrire les résultats dans son fragment de RCTM. Il est intéressant de noter que le mécanisme ABI-PBI élimine totalement les conflits d'accès à des mémoires lors des transferts : la fonction corner turning memory est réalisable avec les éléments standards de CAPITAN ([3], [8]). La consommation estimée de cette configuration est de 210 W, avec des possibilités de réduction si l'on dispose de circuits FFT câblés plus performants que le TMS [8].

4.6. Élaboration de produits image à bord

L'article [17] indique ce que pourra être dans le futur un système embarqué de traitement d'image et décrit sa réalisation à partir de CAPITAN. La diffusion directe vers les usagers finaux implique en effet l'enchaînement d'un certain nombre de traitements : compensation radiométrique, géométrique, étiquetage des pixels. Une structure comme CAPITAN devient alors particulièrement attractive vu la variété des traitements à effectuer. L'article analyse en outre en détail les mécanismes de synchronisation entre tâches, et plus généralement la mise en œuvre logicielle de l'application sur la configuration proposée.

5. Conclusion

Les différentes configurations et les différentes applications décrites ci-dessus illustrent à la fois la souplesse de CAPITAN et l'étendue de son domaine d'applications. Elles montrent que ce genre de machine est parfaitement capable de s'insérer dans une chaîne temps réel de traitement d'image fonctionnant à cadence vidéo. Si l'on excepte le cas des traitements très élémentaires (nous avons cité le cas du codage DPCM), le volume d'électronique d'une architecture basée sur CAPITAN est tout à fait comparable à celui d'une architecture câblée. La raison en est la suivante : certes, l'utilisation de processeurs programmables conduit indiscutablement à une complexité de matériel bien plus grande exprimée en nombre de transistors; mais cette complexité est largement compensée par l'utilisation de circuits commerciaux d'un niveau d'intégration et de performances auquel bien peu d'utilisateurs ont accès.

A l'avenir, l'aspect « structure d'accueil » de la machine garantit l'évolution de ce produit car elle lui permettra de bénéficier pleinement de l'amélioration technologique de ces circuits. Par exemple, l'arrivée prochaine d'équivalents CMOS du TMS-320 augmentera de 66% sa puissance de calcul et divisera probablement par 2 ou 3 sa consommation. A plus long terme, la machine pourra accueillir des circuits VHSIC, CITGV ou cinquième génération. De plus, l'effort technologique prévu par l'Agence Spatiale Européenne pour la miniaturisation du calculateur en fera un outil encore beaucoup plus attractif. A titre d'exemple, nous avons vu qu'après miniaturisation, l'application Sar embarqué ne consomme que 210 W. 70% de cette puissance est due aux seuls TMS 320 [8]. Tous les frais généraux incluant les modules ABI, PBI, les mémoires et les coupleurs ne représentent que 30%. Quel intérêt économique y aura-t-il dans ce cas à développer une architecture particulière sachant que l'optimisation ne pourra porter que sur une fraction de ces 30%?

(*manuscrit corrigé reçu le* 10 *octobre* 1984).

BIBLIOGRAPHIE

[1] ESTEC, Study of programmable high speed processor for use on board satellite, ESA call for competitive offer n° A0/1/1378/81/NL/HP (SC), septembre 1981.

[2] G. Gaillat, The Design of a parallel processor for image processing on board satellites: An application oriented approach, 10*th Annual Symposium on Computer Architecture*, Stockholm, 1983.

[3] MATRA, HSPP architecture report, MATRA internal report n° EPT/063/339/83.

[4] M. J. Flynn, Some computer organizations and their effectiveness, *IEEE Transactions on Computers*, C-21, septembre 1972, p. 948-960.

[5] G. GAILLAT, CAPITAN : Un calculateur parallèle pour le traitement d'images embarqué à bord de satellites, 4e *congrès Reconnaissance des formes et intelligence artificielle*, Paris, 1984.

[6] G. GAILLAT, J. N. NGUYEN, Procédé d'échange d'informations entre abonnés par bus anneau et dispositif multiprocesseur en comportant application, Brevet n° 84 15 321, 1984.

[7] MOSTEK, MOTOROLA, SIGNETICS et PHILIPS, *VME bus specification manual*, 1981.

[8] R. OKKES, G. GAILLAT et R. SCHOTTER, On board Sar Processing, *ISPRS congress*, Rio de Janeiro, 1984.

[9] G. GAILLAT et M. BERTHOD, Panorama des techniques d'extraction de traits caractéristiques en lecture optique des caractères, 2e *congrès Reconnaissance des formes et intelligence artificielle*, Toulouse, 1979.

[10] K. NOURORI, T. KAWATANI, K. ISHII et T. IIDA, A feature concentration method for character recognition, *IFIP World Congress*, Toronto, 1977.

[11] G. GAILLAT, Méthodes statistiques de reconnaissance des formes, *Cours de l'École Nationale Supérieure des Techniques Avancées*, p. 38-45, 1983.

[12] R. STEFANELLI et A. ROSENFELD, Some Parallel Thinning Algorithms for Digital Pictures, *JACM*, vol. 18, n° 2, April 1971, p. 255-264.

[13] O. D. FAUGERAS et M. BERTHOD, Improving consistency and reducing ambiguity in stochastic labelling: An optimization approach, *IEEE Trans. PAMI*, vol. PAMI, n° 4, July 1981, p. 412-424.

[14] J. C. DE GAVRE, Compression of image data from remote sensing satellites, *ISPRS congress*, Rio de Janeiro, 1984.

[15] G. E. LOWITZ et G. CASSAGNE, CLADYNE : Un nouveau compresseur pour la télévision digitale, 1er *colloque image*, Biarritz, 1984.

[16] CCIR, Codage numérique des signaux de télévision couleur, Rapport CCIR R 629-2, 1982.

[17] J. C. DE GAVRE, R. OKKES et G. GAILLAT, Study of a programmable high speed processor for use on board satellites, 4*th symposium on Computers in Aerospace*, Hartford, 1983.