NEUVIEME COLLOQUE SUR LE TRAITEMENT DU
SIGNAL ET SES APPLICATIONS

NICE du 16 au 20 MAI 1983

EVALUATION D'ARCHITECTURES DE MICROPROCESSEURS DE TRAITEMENT DU SIGNAL

Y. SOREL P. WOLF

INRIA - Domaine de Voluceau - B.P. 105 - 78153 LE CHESNAY

RESUME

Nous avons étudié l'implantation d'applications typiques en traitement du signal tel un modem 1200 Bps et une FFT temps réel 64 points complexes, sur les microprocesseurs spécialisés suivants : le µPTS du CNET, le µPD 7720 de NEC et le TMS 320 de TI.

Cette étude nous a amené à effectuer une classification à différents niveaux des architectures considérées et de mettre en évidence les qualités et défauts respectifs dans le cadre limité d'une classe d'applications.

La méthode utilisée consiste en une phase de simulation suivie d'une réalisation matérielle qui sert de validation. La mise au point des algorithmes a été réalisée en Pascal. Pour leur simulation, le logiciel ISPS qui permet de décrire la structure interne et le comportement matériel et logiciel du microprocesseur s'est avéré être un outil souple et puissant.

Cela nous a fourni de nombreuses informations sur les points essentiels d'une architecture de microprocesseurs de traitement du signal et de son jeu d'instructions.

Nous avons plus particulièrement mis en évidence l'importance du mécanisme d'adressage, du contrôle des débordements, des instructions de répétition de séquence , compteur de boucle, registres retard, contrôle des entrées sorties. La méthodologie dégagée est utilisée actuellement pour la réalisation d'un modem 4800 Bps.

SUMMARY

The purpose of this paper is to present the implementation of typical applications in digital signal processing, as a 1200 Bps modem and a 64 complex points FFT in real time, on three specialized microprocessors : the µPTS of CNET, µPD 7720 of NEC and the TMS 320 of TI.

This work allows us to classify them at different architecture levels and determines their adequacy to a limited class of algorithms.

Our method consits of a software simulation using the ISPS Computer Description Language and a hardware implementation on a µPD 7720 evaluation kit.

We gather a lot of informations about the significant points of their architecture and instruction set. More specifically, we emphasize the importance of addressing modes, overflow control, delay registers, loop counter and input-output control.

The tools developed are now being used to realize a 4800 Bps modem.

EVALUATION D'ARCHITECTURES DE MICROPROCESSEURS DE TRAITEMENT DU SIGNAL

INTRODUCTION.

Dans une première partie nous présentons brièvement les architectures matérielles des trois microprocesseurs et dégageons les particularités de chacun. Nous décrivons ensuite les deux applications retenues et détaillons en particulier le modem 1200 Bps. En ce qui concerne la FFT, le choix des algorithmes dépend du microprocesseur utilisé pour leur implémentation.

La seconde partie précise la méthodologie employée en insistant plus particulièrement sur l'efficacité du logiciel de simulation ISPS.

Enfin, nous décrivons les résultats obtenus en mettant en évidence les relations entre les algorithmes liés à chaque application et les unités matérielles qui les implémentent.

I. PRESENTATION DES ARCHITECTURES MICROPROCESSEURS ET DES APPLICATIONS.

1. *Microprocesseurs de traitement du signal étudiés.*

Parmi les microprocesseurs actuellement disponibles sur le marché, nous avons retenu le µPD7720 de NEC [Ne81] et le TMS 320 de TI [Te82] qui semblent les plus performants. Le µPTS du CNET [Cn82] dont la structure et le jeu d'instructions sont actuellement figés et qui est sur le point d'être réalisé matériellement présente des qualités qui justifient sa présence dans cette étude. Les schémas associés à leur description succincte font ressortir une philosophie de conception identique pour chaque architecture ; seuls les chemins de données reliant les différentes unités matérielles entre elles et la richesse du jeu d'instructions les distinguent.

Ainsi, on retrouve des parties fonctionnelles communes ; une unité arithmétique et logique à précision étendue, un multiplieur rapide intégré, des mémoires de données sur 16 bits, une large mémoire de programme interne, une entrée-sortie parallèle

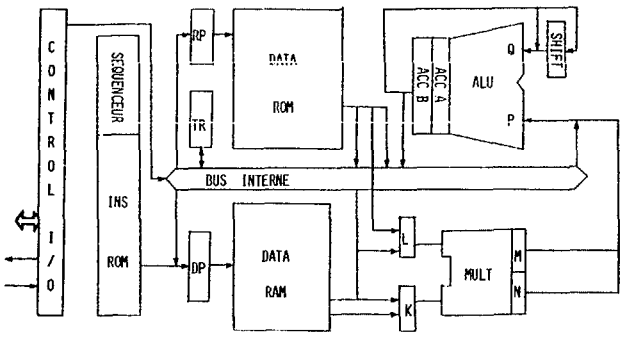


Fig. 1 : µPD7720.

- unité arithmétique et logique ALU à virgule fixe avec 2 ACCU 16 bits (double précision possible) ce qui donne une dynamique de 2^3 à 2^7 pour la représentation interne des nombres ; un registre temporaire.
- multiplieur câblé 16x16 → 31 bits séparés en deux registres de 16 bits.
- RAM données 128x16, ROM données 512x13 avec adressage indirect par un pointeur RAM(DP) et un pointeur ROM(RP) et post incrémentation possible des pointeurs.
- mémoire de programme PROM 512x23 ; 4 niveaux de sous-programmes.
- données circulant sur un bus unique multiplexé dans le temps.
- un port entrée-sortie parallèle sur 8 bits plus 2 lignes d'adresse, une entrée série et une sortie série qui permettent le chaînage de plusieurs boîtiers.

En une instruction, on est capable de réaliser : opération d'ALU, modification des pointeurs d'adresse, déplacement source destination, la multiplication s'effectuant en parallèle. Grâce à 3 bits de garde, on peut contrôler les débordements.

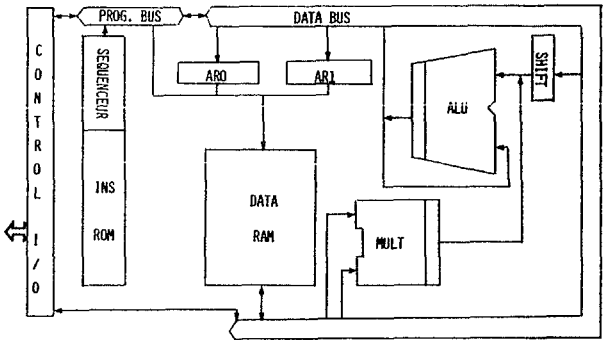


Fig. 2 : TMS 320.

- ALU virgule fixe, 32 bits avec possibilité de décalage des entrés et ACCU 32 bits accessibles en 2x16.
- multiplieur câblé 16x16 → 31 bits.
- RAM données 144x16 avec adressage direct ou indirect et paginé grâce à deux registres pointeur ARO et AR1 et un pointeur de page.
- PROM 1536x16 ; 4 niveaux de sous-programmes ; registres d'adresse pouvant servir de compteur de boucle.
- un bus programme 16 bits et un bus données de même taille qui communiquent.
- un port d'entrée-sortie parallèle 16 bits avec extension externe posssible de la PROM (mode émulation) et de la RAM.

L'instruction autorise un parallélisme très réduit : opération d'ALU plus modification des pointeurs d'adresse d'où sa simplicité d'emploi. Une opération conditionnelle facilite l'implantation de la division.

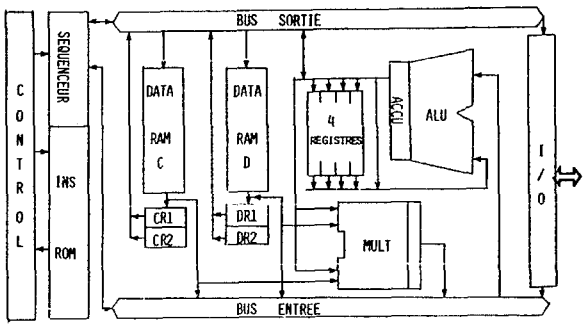


Fig. 3 : µPTS

- ALU à virgule fixe 32 avec ACCU 32 bits et 4 registres brouillon de même taille d'où une dynamique élevée 2^7 à 2^{-24} ; possibilité de précision étendue sur 25 bits.

- multiplieur câblé 16×16 → 25 bits transmis sur 26 fils pour régler les problèmes de débordement.
- deux RAM données 128×16 avec adressage direct sur une des mémoires ou adressage indexé modulo 2^n (n=0 à 6) sur les deux mémoires ; deux registres retard par RAM.
- PROM 512×26 ; 4 niveaux de sous-programmes ; un registre compteur de boucle et un registre compteur de répétition.
- 2 bus principaux ; un bus sortie 16 bits, un bus entrée 26 bits.
- un port entrée-sortie parallèle 16 bits et un port de contrôle qui permet notamment une extension de la PROM.

Il permet de nombreuses possibilités de cadrage des données. Le répertoire d'instructions est très riche : opérations conditionnelles, renversement automatique des bits d'adresse de RAM ("bit reverse"), contrôle élargi des débordements ...
En une instruction on peut faire : modification des pointeurs mémoires, opération d'ALU, déplacement source-destination et multiplication en parallèle ou aussi rupture de séquence avec déplacement source-destination.

2. *Description fonctionnelle des applications.*

2.1. *Modem 1200 Bps.*

Il utilise la modulation à déplacement de fréquence (F.S.K.) et fonctionne à 1200 Bps sur le réseau commuté (avis V23 du CCITT) [DU73].

Les fréquences caractéristiques de la voie de transmission représentant les symboles 1 et 0 sont respectivement :

$$F_z = 1300\ H_z \quad \text{et} \quad F_A = 2100\ H_z$$

La chaîne complète modulation-ligne-démodulation est représentée par le schéma suivant :

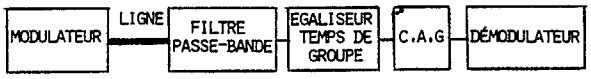
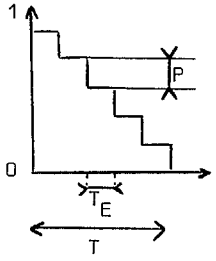


Fig. 4

2.1.1. *Modulation.*

Elle s'effectue à l'aide d'un oscillateur numérique échantillonné à F_E = 13020 H_z qui délivre les sinusoïdes à 1300 et 2100 H_z dépendant du bit de donnée en entrée.

La sinusoïde est obtenue à partir d'un signal en dent de scie décalé, redressé puis écrêté. La période de la dent de scie prend deux valeurs et est égale à n fois la période d'échantillonnage.



$$T = nTe$$

$$\text{il faut } p \leq \frac{1}{n-1}$$

Fig. 5

Cette méthode à l'avantage de permettre un meilleur raccordement entre deux morceaux de sinusoïde de fréquences différentes.

2.1.2. *Démodulation.*

Elle se compose dans l'ordre de :

- un filtre passe-bande 1300-2100 Hz avec un zéro de transmission à 450 Hz ; filtre Cauer du 6e ordre constitué de 3 cellules du 2e ordre.
- un égaliseur de temps de groupe ; filtre passe-tout du 6e ordre constitué de 3 cellules du 2e ordre. Son rôle est de rendre constant $Tg(f) = \frac{1}{2\pi}\frac{d\phi}{df}$.
- une C.A.G. qui rend l'amplitude constante à la sortie des filtres précédents. Le signal est redressé puis divisé par son enveloppe extraite par filtrage passe-bas.
- un démodulateur

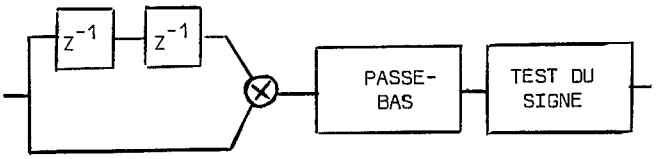


Fig. 6.

Les fréquences de modulation sont :

$$f_z = f_o - \Delta f \quad \text{avec } f_o = 1700 \text{ Hz et } \Delta f = 400 \text{ Hz}$$
$$f_A = f_o + \Delta f$$

Si y(t) est le signal à l'entrée :

$$y(t) = A \cos [2\pi (f_o + \Delta fa_i)t + \Phi] \text{ avec } a_i = \pm 1$$

le signal d'entrée est décalé de τ :

$$y(t - \tau) = A \cos [2\pi (f_o + \Delta fa_i)(t - \tau) + \Phi]$$

en multipliant y(t) par y(t - τ) on a :

$$y(t).y(t - \tau) = \frac{A^2}{2} \cos [2\pi(f_o + \Delta fa_i)\tau] +$$
$$+ \frac{A^2}{2} \cos[2\pi(f_o + \Delta fa_i)(\partial t - \tau) + 2\Phi]$$

en filtrant par un passe-bas on élimine les fréquences doubles correspondant à la partie droite de la somme ; en prenant $\tau = \frac{1}{4f_o}$ on obtient :

$$s(t) = \frac{A^2}{2} \cos [2\pi(f_o + \Delta fa_i)\frac{1}{4f_o}]$$

soit pour

$$a_i = -1 \quad s(t) = \frac{A^2}{2} \sin(-\pi \frac{\Delta f}{2f_o})$$

$$a_i = +1 \quad s(t) = \frac{A^2}{2} \sin(\pi \frac{\Delta f}{2f_o})$$

s(t) prend deux valeurs opposées. Il suffit de tester le signe en sortie pour déterminer la valeur 0 ou 1. Pour f_E = 13020 Hz le retard τ est proche de 2 échantillons. Le filtre passe-bas est un Batterworth du 4e ordre coupant à 100 Hz et conservant 600 Hz.

2.2. *Transformée de Fourier rapide.*

Rappelons que la TDF (transformée de Fourier discrète) d'une séquence de N valeurs complexes f_n est la séquence

$$(1) \qquad F_m = \sum_{n=0}^{N-1} f_n \, e^{-j(2\pi m\, n/N)} \qquad m=0,1,\ldots,N-1$$

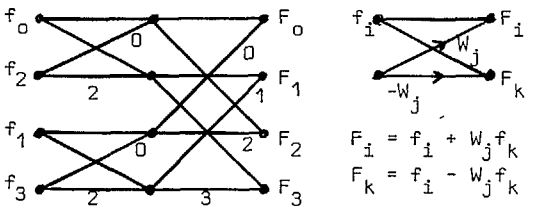
A cause de la périodicité de la fonction exponentielle, on voit qu'un certain nombre de ces produits sont redondants. Si $W_N = e^{-j\frac{2\pi}{N}}$ on a :

$$(2) \qquad F_m = \sum_{n=0}^{N-1} W_N^{mn} \, f_n$$

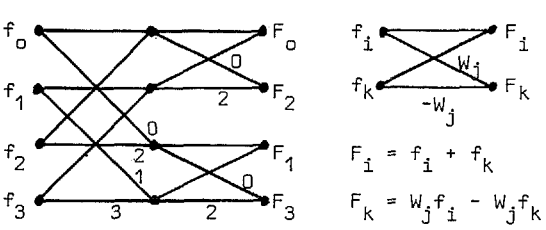
L'équation (1) peut être réécrite sous la forme :

$$F_m = \sum_{n=0}^{E(\frac{N-1}{2})} f_{2n} \, W_N^{2mn} + W_N^m \sum_{n=0}^{E(\frac{N-3}{2})} f_{2n+1} \, W_N^{2mn}$$

chacune des sommes peut se mettre sous la forme d'une TDF que l'on peut à son tour décomposer. On choisit $N=2^k$ pour faciliter la dichotomie. Nous avons pris N=64 pour saturer les mémoires internes des microprocesseurs étudiés. Ces observations conduisent à deux types de décomposition : la décomposition en temps qui correspond à un entrelacement des f_k et celle en fréquence qui correspond à un entrelacement des F_k. Dans les deux cas, il faut effectuer un réarrangement des indices des échantillons en entrée ou sortie ("bit reverse"). Ces deux décompositions s'illustrent par un diagramme du type flot de données dont on peut extraire un motif de base noté "papillon".



Décomposition en temps
avec bit reverse en entrée



Décomposition en fréquence
avec bit reverse en sortie

Fig. 7

2.3. *Description des algorithmes.*

Pour le modem 1200 Bps, l'algorithme principal utilisé est celui qui permet de réaliser une cellule de filtrage du 2e ordre. La cellule est mise comme ci-dessous sous une forme permettant de minimiser les cases mémoires :

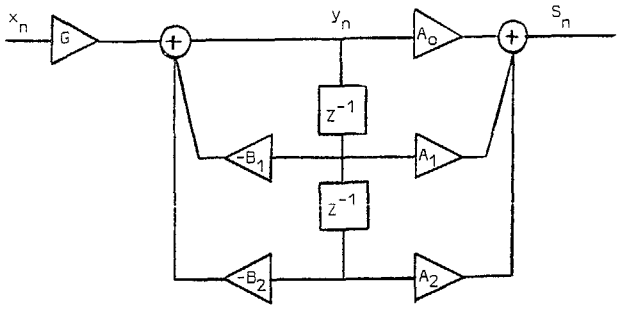


Fig. 8

Les équations récurrentes sont alors :

$$\begin{cases} y_n = Gx_n + B_1 y_{n-1} + B_2 y_{n-2} \\ S_n = A_o y_n + A_1 y_{n-1} + A_2 y_{n-2} \\ y_{n-2} = y_{n-1} \text{ et } y_{n-1} = y_n \end{cases}$$

La FFT sur 64 points complexes a été implantée de deux façons différentes (selon le microprocesseur) :

- un seul sous-programme général représentant le papillon de base, les calculs étant faits après avoir déterminé les adresses de chaque entrée et sortie ainsi que du coefficient W. les calculs progressent papillon après papillon sur une colonne, puis d'une colonne à l'autre (cf. Fig. 7) sachant qu'il y a k colonnes pour 2^k points à traiter

$F(i) = f(i) + W_j \, f(k)$ — i, j, k sont calculés à chaque appel

$F(k) = f(i) - W_j f(k)$ — de l'algorithme papillon.

- plusieurs sous-programmes décrivant un papillon par colonne, la progression des calculs étant la même que ci-dessus. On économise ainsi le calcul d'adresse sur k : $k = i + 2^q$ sur la q-ième colonne de calcul, d'où l'intérêt d'un adressage indexé modulo 2^q.

La première méthode économise de la place mémoire PROM mais impose un calcul d'adresses sophistiqué et la seconde méthode conduit à un programme plus long mais plus simple au niveau de l'adressage.

3. *Description du langage ISPS.*

Le langage de description d'ordinateurs ISPS (Instruction Set Processor Specifications) a été développé à Carnegie Mellon Univ. et utilisé comme outil de conception par DEC par exemple pour le PDP11 et la famille VAX. C'est un langage du type RTL [Le80] (Register Transfer Language) qui constitue une étape vers la formalisation de la conception d'architectures au niveau supérieur du fonctionnement.

Il permet la description des ressources matérielles de la machine vues par l'utilisateur (mémoires, registres, bus, unités de calcul ...) et des cycles d'interprétation des instructions utilisant ces ressources (aspect dynamique). Pour décrire le jeu d'instructions d'une machine, il faut définir le nombre et le type de supports qui stockent et véhiculent l'information entre les unités, les opérations sur les contrôles et données de la machine ainsi que les règles d'interprétation utilisées. On les introduit successivement dans des parties bien définies du programme :

- une partie déclaration qui précise la structure des mémoires et du processeur
- une partie procédure qui décrit l'étape d'interprétation i.e. le mécanisme par lequel le processeur charge, décode et exécute les instructions.

Ce logiciel est utilisable pour des applications diverses : simulation et synthèse de matériel, génération de logiciel, vérification de programmes, évaluation d'architectures et analyse des comportements. Dans notre étude, nous nous sommes servis d'ISPS pour décrire les trois microprocesseurs, pour simuler le fonctionnement des algorithmes définis ci-dessus et mesurer leurs performances respectives.

L'exploitation des descriptions ISPS s'effectue en 3 phases. Un premier programme effectue l'analyse syntaxique et sémantique de la description ISPS et la traduit sous une forme arborescente dite "Base de Données Globales" (GDB). Un deuxième programme traduit celle-ci en code objet pour une machine virtuelle appelée "Machine à transfert de Registres" (RTM). Un troisième programme utilise ce code objet et un simulateur de la machine virtuelle RTM permet de générer un simulateur de la machine cible.

Dans le cas d'un microprocesseur spécialisé, la simulation ISPS permet d'évaluer des structures particulières : pipelines, mécanismes d'adressage, registres retard, contrôle des débordements.

Nous donnons dans la suite des exemples d'application du langage qui montrent sa souplesse et facilité d'emploi (voir en Annexe un abrégé de la description en ISPS du µpD7720).

II - METHODOLOGIE.

1. *Simulation.*

Avant d'effectuer la programmation des algorithmes dans l'un ou l'autre des assembleurs, il est important de se faire une idée de leur sensibilité aux problèmes de précision à l'aide d'une simulation en PASCAL. Cette dernière prend en compte la dynamique des données (simple ou double précision, fixe ou flottant), contrôle les débordements et reproduit le cheminement des données. Pour le modem 1200 Bps elle permet notamment de calculer les gains inter-cellules qui optimisent la dynamique et maîtrisent les débordements.

Pour la FFT elle permet de suivre l'évolution des adresses pour les différentes méthodes de programmation.

Après traduction des algorithmes en assembleur, on regroupe dans un fichier de simulation ISPS [Ba80] un certain nombre de commandes qui testent ces programmes. Grâce à ces commandes on peut, entre autres, charger les registres et mémoires de la description ISPS, lancer l'exécution d'un programme d'un point quelconque à un point quelconque, suivre au cours de cette exécution l'évolution des divers registres et supports de données, simuler les entrées-sorties par des fichiers de valeurs ... La simulation donne, en outre, la valeur de compteurs qui représentent le nombre de lectures et écritures dans les registres et mémoires (l'unité de comptage choisi est l'octet) et d'exécutions des procédures. Ces paramètres mesurent le nombre d'octets circulant sur un bus ou entre les ressources matérielles du circuit pendant l'exécution du programme ainsi que la fréquence d'utilisation des divers types d'instructions ou d'utilisation de certaines instructions spéciales.

La souplesse d'utilisation du simulateur ISPS nous apporte les moyens d'optimiser la programmation en assembleur des algorithmes. Dans le cas du µPD7720 et µPTS, on utilise au mieux les possibilités du parallélisme. On gère au mieux les mémoires et les registres, ce qui est particulièrement important dans l'application FFT.

2. *Résultats.*

Dans le cadre d'une évaluation d'ordinateurs militaires pour le Ministère de la Défense aux Etats Unis le projet CFA [Ba81] a proposé les paramètres suivants, tirés de l'exploitation des compteurs générés par la simulation ISPS de certains programmes de base :

- M le nombre d'octets transférés entre la mémoire et le processeur.
- R le nombre d'octets transférés entre les divers registres.
- S le nombre d'octets utilisés pour écrire un programme test.

En ce qui concerne les microprocesseurs étudiés, le paramètre S reste singificatif, mais on lui rajoute les paramètres calculés par intervalle d'échantillonnage :

- E le nombre d'instructions effectivement exécutées lors du déroulement d'un programme : le temps d'exécution est le produit de ce paramètre par le temps de cycle.
- D qui mesure le "flot de données" entre mémoires et registres et entre les registres : il faut faire un choix judicieux des compteurs retenus (nous détaillerons pour le TMS 320).
- A le nombre d'opérations effectuées par l'unité arithmétique et logique.
- P le nombre de produits "efficaces" : dans le cas où la multiplication s'effectue en parallèle à chaque cycle avec l'opération d'ALU, tous les produits ne sont pas exploités.

Les paramètres caractérisant les transferts d' d'entrées-sorties, les sauts et appels de sous-programmes sont moins significatifs dans le cas des applications traitées (les chiffres donnés pour les algorithmes ne tiennent pas compte des entrées-sorties).

Par exemple, pour le TMS320, le paramètre D est la somme des compteurs associés aux lectures et écritures en RAM, aux écritures dans AR, la paire de registres auxiliaires, et T et aux lectures dans P, T et P étant respectivement les registres d'entrée et de sortie du multiplieur. Pour pouvoir effectuer une comparaison à partir de cette mesure, on ramène les compteurs qui sont donnés en nombre d'octets, à la taille des registres concernés.

Le tableau qui suit donne les paramètres définis ci-dessus pour les algorithmes :

- partie démodulation du modem 1200 Bps sur µPD7720 et µPTS.
- FFT 64 pts et 32 pts avec contrôle des débordements sur µPD7720 et µPTS.
- FFT 32 pts (limite imposée par la taille de la mémoire interne de données) sur TMS320 sans contrôle des débordements (mal commode sur ce microprocesseur).

Paramètres	*µP / Applications*	*µPD7720*	*TMS320*	*µPTS*
S	FFT32	240	101	101
	FFT64	297		133
	MODEM	48		28
E	FFT32	3746	4429	1258
	FFT64	8306		2996
	MODEM	146		115
D	FFT32	7952	4675	3680
	FFT64	17400		8852
	MODEM	357		266
A	FFT32	2009	1092	773
	FFT64	4681		1501
	MODEM	88		87
P	FFT32	256	356	320
	FFT64	640		768
	MODEM	65		57

Les temps de cycles sont actuellement : 250 ns pour µPD7720, 200 ns pour TMS 320, 300 ns pour µPTS.

3. *Réalisations matérielles.*

Nous avons implanté les programmes FFT et modem 1200 Bps sur un simulateur temps réel composé d'un système d'évaluation du µPD7720 fourni par NEC associé à une carte d'interface comportant, entre autres, des convertisseurs A.N et N.A. Le comportement des maquettes est conforme aux prédications du simulateur.

III - EXPLOITATION DES RESULTATS.

A partir des résultats tirés de la méthode définie précédemment, nous dégageons les points importants concernant les microprocesseurs de traitement du signal en mettant en parallèle les ressources matérielles, le jeu d'instructions et les algorithmes utilisés. Examinons successivement l'adressage, l'arithmétique, le séquencement et les entrées-sorties.

1. *Adressage.*

Les algorithmes de traitement du signal manipulent fréquemment des variables indicées ce qui se traduit au niveau de la programmation par de nombreuses manipulations d'adresses. On a besoin d'un pointeur d'adresses pour les coefficients implantés en ROM ou RAM et d'un pointeur pour les données en RAM. Un pointeur de RAM supplémentaire augmenterait l'efficacité d'un calcul du type papillon FFT qui manipule deux données et un coefficient : par exemple le fait de n'avoir qu'un pointeur RAM pour le µPD7720 oblige le stockage des pointeurs données en RAM ce qui explique en partie les différences au niveau du paramètre d'évaluation D. L'adressage indirect des mémoires à partir de registres pointeurs associé à une possibilité de calcul sur ces registres (accès à l'ALU) facilite les calculs d'indices sophistiqués (FFT avec un seul sous-programme papillon, treillis, filtrage multidimensionnel...). L'adressage direct éventuellement indexé s'avère indispensable lorsque les adresses des données sont fixes. Le µPTS possède ce mode d'adressage avec un index multiple de 2 qui est très efficace pour la FFT à plusieurs sous-programmes "papillon" (ceci explique ses bonnes performances pour les critères S et E).

L'algorithme associé à la cellule de filtrage du 2^e ordre fait intervenir deux mémoires retard. Ces dernières sont implantées directement sous forme de registres à la sortie des RAM C et D du µPTS. Ils sont remis à jour à chaque lecture de RAM. Cela économise deux accès supplémentaires à la mémoire. Dans le cas des applications choisies une profondeur de deux retards suffit.

Nous avons apprécié l'instruction spéciale du µPTS qui effectue un renversement matériel des bits d'adresse jusqu'à désactivation ("bit reverse").

2. *Arithmétique.*

Les 3 microprocesseurs étudiés intègrent un multiplieur cablé 16 x 16. Le paramètre P montre la fréquence d'utilisation de cette unité et indique l'utilité d'un tel choix qui facilite la programmation et le choix des algorithmes car la multiplication ne prend pas plus de temps qu'une addition. Pour les calculs du type $\Sigma A_i x_i$ son fonctionnement en parallèle avec la séquence opération d'ALU, modification des pointeurs, déplacement mémoire s'impose : cette option n'a pas été retenue sur le TMS 320 (instructions courtes sur 16 bits) et ceci ralentit les calculs (en particulier pour le filtrage). D'autres microprocesseurs comme IBM RSP [Mi80] n'intégrant pas cette fonction (gain de place sur la puce) imposent un choix judicieux des algorithmes et des coefficients (ex : méthode Winograd pour la FFT [Mc79]).

Pour les applications étudiées, nous n'avons pas utilisé les possibilités de précision étendue sur 32 bits. Le problème des débordements doit pouvoir se régler par plusieurs bits de garde (3 pour le µPD7720, 8 pour le µPTS). Une solution complémentaire consiste à positionner en fonction de drapeaux de débordement un registre à décalage à gauche qui recadre les données (un système à deux drapeaux fonctionne sur le µPTS et permet la division par 2 ou 4 des données).

Une arithmétique flottante ou pseudo-flottante qui ne se justifie pas ici peut être envisagée ainsi qu'une architecture différente basée sur un algorithme unifié CORDIC de calcul des fonctions élémentaires (multiplication, division, racine, sin, cos ...) [Wa71].

3. *Séquencement.*

Les possibilités de rupture de séquence (branchements et appels sous-programme éventuellement conditionnés par des drapeaux) sont analogues dans les 3 processeurs : le µPTS possède de plus un retour de sous-programme conditionnel ainsi qu'une instruction spéciale de répétition de l'instruction suivante. L'utilisation d'un registre compteur de boucle est souhaitable (efficacité du code : voir paramètre S pour µPTS et µPD7720).

Le µPD7720 et le µPTS traitent les instructions en pipelinant la recherche de l'instruction suivante avec

le décodage et l'exécution de l'instruction. L'exécution s'effectue en plusieurs étapes successives (instruction de format long à plusieurs champs) en parallèle avec une multiplication. Cette dernière donne le rythme d'exécution de l'instruction. Les paramètres P et A montrent que les unités de calcul ne sont pas utilisées à plein rendement (ex : pour le µPD7720 : P/E = 8% pour la FFT 64 et P/E = 44% pour le modem). Une solution consisterait en un format variable d'instructions avec éventuellement l'introduction d'un pipeline à plusieurs niveaux (4 niveaux de pipeline au total pour le RSP d'IBM [Mi81]). Afin de régler avec souplesse ces problèmes l'introductior de possibilités de microprogrammation au niveau du séquenceur semble souhaitable.

4. Entrées-sorties.

Les organes d'entrées-sorties proposés par les 3 microprocesseurs (port entrée-sortie parallèle sur 8 ou 16 bits, port série en plus sur le µPD7720) sont satisfaisants pour des applications à un seul processeur. Pour celles nécessitant un stockage de données et programmes plus important il est utile de pouvoir accéder à de la RAM et de la ROM externe : le TMS 320 offre ces possibilités de manière transparente pour l'utilisateur.

Pour des applications où la fréquence d'échantillonnage est élevée pour une taille de programmes importante (modem rapide, traitement audio et parole Hi-Fi) on est amené à découper les algorithmes ; ceci conduit à une configuration multiprocesseur du type série parallèle [Ru81], réseau en étoile, etc... Ce type d'organisation impose au niveau du microprocesseur un système d'entrée-sortie efficace avec par exemple : arbitre d'accès à un bus commun (cf le µPTS), possibilités de transferts asynchrones, d'émission simultanée d'adresses et de données (cf le TMS 320), tampons d'entrées-sorties...

Nous étudions actuellement les structures souhaitables d'entrées-sorties ainsi que les possibilités d'introduction de pipeline dans le type d'architectures étudiées à partir d'une description plus fine à l'aide de ISPS. La réalisation en cours d'un modem 4800 Bps impose de répartir sur plusieurs microprocesseurs les ressources en calcul ; elle permettra d'évaluer une configuration multiprocesseur du type série-parallèle.

BIBLIOGRAPHIE

[Ba80] Barbacci, M.R., et al. : "The Symbolic Manipulation of Computer Descriptions : An ISPS Simulator". Technical Report, Department of Computer Science, Carnegie-Mellon University.

[Ba81] Barbacci, M.R. : "ISPS : The Notation and Its Applications". IEEE Transactions on Computers, Vol. C-30, N° 1 Janvier 1981, pp. 24-40.

[Be81] Bellanger M. : Traitement Numérique du Signal. CNET ENST, ed. Masson.

[Cn82] Spécifications du µPTS. Note Technique NT/CNS/CCI/09, Juillet 1982.

[Du73] Dupraz J. : Théorie de la Communication. Signaux, Bruits et Modulation. ed. Eyrolle.

[Le80] Lewin D. : "Computer aided Design for microcomputer systems". Department of electrical engineering and electronics. Brunel University, Uxbridge.

[Mc79] Mc Clellan J.H., Rader C.M. : "Number Theory in Digital Signal Processing". Prentice Hall Processing series. Alan V. Oppenheim Series Editor.

[Mi81] Mintzer F., Pelled A. : "A microprocessor for signal processing : the RSP". IBM Research Report RC 9081.

[Ne81] µPD7720 Signal Processing Interface Product Description.

[Ra75] Rabiner L.R., Gold B. : "Theory and Application of Digital Signal Processing". Prentice Hall.

[Ru81] Ruiz A. : "On the hardware implementation of a multiprocessor environment for several digital processing applications". IBM Research Report RC 8669.

[Te82] TMS 320 : Preliminary Data Manual.

[Wa71] Walther J.S. : "A unified algorithm for elementary functions". Proc. AFIPS 1971. Spring Taint Computer Conference, pp. 379-385.

ANNEXE

```
        !EXTRAITS DE LA DESCRIPTION ISPS DU MPD7720
MPD7720:=
BEGIN
**MEMORY.STATE**                        !ETAT DES MEMOIRES:
PROM[0:511]<22:0>,                      ! PROGRAMME
DROM[0:511]<12:0>,                      ! COEFFICIENTS
RAM[0:127]<15:0>,                       ! DONNEES

**PROCESSOR.STATE**                     !ETAT DU PROCESSEUR:
PC<8:0>,                                !COMPTEUR DE PROGRAMME
ACC[0:1]<15:0>,                         !ACCU A ET B
K<15:0>,                                !ENTREE MULT.
L<15:0>,                                !ENTREE MULT.
REMUL<31:0>,                            !PRODUIT PRECEDENT
M<15:0>,                                !SORTIE MULT.
N<15:0>,                                !SORTIE MULT.
IR<22:0>,                               !REGISTRE D'INSTRUCTION
IDB<15:0>,                              !BUS DONNEES INTERNES
.............
.............

**INSTRUCTION.INTERPRETATION**         !JEU D'INSTRUCTIONS
RUN:=
  BEGIN
  REPEAT
    BEGIN
    FETCH() NEXT                        !CHARGEMENT INSTRUCTION
    DECODE IR<22:21>=)                  !TYPES D'INSTRUCTIONS:
      BEGIN
      '0?:=TYPEOPRT(),                  ! OPERATION
      '10:=TYPEJP(),                    ! RUPTURE DE SEQUENCE
      '11:=TYPELD()                     ! CHARGEMENT
      END
    END
  END,
FETCH:=
  BEGIN
  IR=PROM[PC] NEXT
  PC=PC+1                               !INCREMENT DU POINTEUR
  END,
TYPELD:=
  BEGIN
  IDB:=IR<20:5> NEXT                    !VALEUR A CHARGER
  DECDST() NEXT                         !RECHERCHE DESTINATION
  MULTI()                               !PROCEDURE MULT. EN //
  END,
.............
.............
MULTI:=
  BEGIN
  M<15:0>@N<15:0>=REMUL<30:0>@'0 NEXT
  REMUL=K*L                             !NOUVEAU PRODUIT
  END,
.............
.............
END                                     !FIN DE MPD7720
```