HUITIEME COLLOQUE SUR LE TRAITEMENT DU SIGNAL ET SES APPLICATIONS

NICE du 1er au 5 JUIN 1981

REALISATION MATERIELLE ET LOGICIELLE D'UN SYSTEME DE TRAITEMENT DU SIGNAL - APPLICATIONS: CALCUL DE FFT, RESOLUTION DE L'EQUATION DE RICCATI

G. BRUN, D. MOUSSIEGT et A. VINCENT-CARREFOUR

Laboratoire de Signaux et Systèmes, E.R.A. n°835 du C.N.R.S. - Université de Nice
41, boulevard Napoléon III - 06041 NICE CEDEX

RESUME

Les auteurs décrivent un système numérique permettant d'implanter les algorithmes de base du traitement du signal : transformée de Fourier rapide, résolution de l'équation de Riccati, filtrage numérique par exemple.

Les objectifs étaient les suivants: réaliser un processeur rapide grâce à une architecture adaptée et néanmoins facile à programmer.

Le matériel comprend un microprocesseur en tranches, un multiplieur intégré 16 bits et des mémoires rapides. Chaque instruction est exécutée en 200 ns.

La production de programmes est simplifiée par un logiciel implanté sur un calculateur numérique. L'utilisateur dispose d'un assembleur original qui présente tous les avantages d'un langage évolué : adressage symbolique, gestion automatique des bases, détection des erreurs d'assemblage.

Un chargeur - translateur fournit les images des mémoires de données et de programme. Le chargement de processeur est effectué au moyen d'une liaison asynchrone avec le calculateur numérique.

Un simulateur complète cet ensemble et permet de tester les programmes indépendamment du matériel.

Afin de montrer la facilité de mise en oeuvre et la souplesse d'utilisation du logiciel, des résultats sont présentés sur deux exemples.

Le premier concerne la F.F.T. (algorithme de Bruun) sur 256 points ;
le deuxième est relatif à la résolution de l'équation de Riccati dans le cas simple d'ordre 2 à sortie scalaire. La prise en compte d'un ordre plus élevé ne présente aucune difficulté particulière. Les algorithmes programmés sont ceux de quasilinéarisation et de Lindquist en arithmétique double longueur.

SUMMARY

The authors describe a fast digital system which is used for implementing the basic algorithm of digital processing: FFT, Riccati equation, digital filtering.

The aims were to realize a fast, and easy to program processor.

The hardware includes a bit-slice microprocessor, an integrated 16 bit multiplier and fast memories. The execution time of each instruction is 200 ns.

The software is implemented on a computer and includes: an assembler, a loader and a simulator.

Two application examples are given. The first one is a 256-point F.F.T. (Bruun algorithm);
the second one is the resolution of a two-order Riccati equation with scalar output. Quasilinearization and Lindquist algorithms are computed in double precision.

REALISATION MATERIELLE ET LOGICIELLE D'UN SYSTEME DE TRAITEMENT DU SIGNAL - APPLICATIONS: CALCUL DE FFT, RESOLUTION DE L'EQUATION DE RICCATI

INTRODUCTION

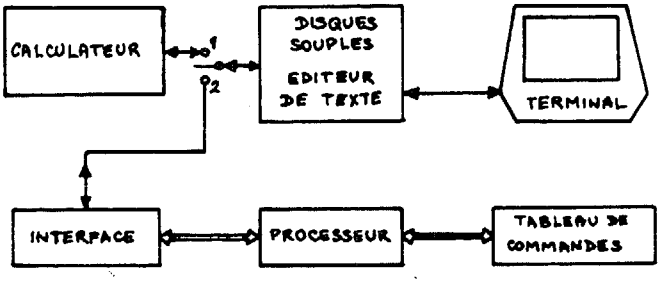
Cette étude a pour but la conception et la réalisation d'un système de laboratoire destiné au traitement du signal en temps réel. Il est conçu afin de mettre au point de manière aisée et d'exploiter les algorithmes de traitement numérique du signal.
La notion de système implique l'existence, en plus du processeur lui-même, d'un certain nombre d'aides à la mise au point matérielle et logicielle, en particulier un logiciel de production de programmes.

L'objectif visé conduit à réaliser un processeur ayant des performances adaptées en vitesse et en puissance de calcul. C'est pourquoi il a été fait appel à un microprocesseur en tranches d'une part pour la vitesse de traitement apportée par les technologies bipolaires, d'autre part pour la souplesse d'emploi (choix de l'architecture, choix du format des mots, choix du jeu d'instructions). A ce dernier est associé un multiplieur intégré : par rapport à une solution programmée, il permet un gain de temps considérable tant dans l'exécution des programmes que dans leur mise au point en raison du très petit nombre d'instructions nécessaires à sa gestion.

La première partie du texte décrit le matériel.
La deuxième expose le logiciel de production de programmes composé d'un assembleur et d'un chargeur-translateur écrit à partir d'un langage symbolique original.
La troisième présente le simulateur aide à la réalisation de programmes indépendamment du matériel.
La quatrième fournit les résultats obtenus sur deux exemples afin de montrer la facilité de mise en oeuvre et la souplesse d'utilisation du logiciel ainsi que les performances du système.
Ces deux applications sont fondamentales l'une dans les aspects filtrage, l'autre dans les aspects stochastiques du traitement du signal. Ce sont, en effet, un algorithme de transformée de Fourier et des algorithmes de résolution de l'équation de Riccati.

I - LE MATERIEL

I-1 Présentation du système :



—— transmission série asynchrone
═══ transmission parallèle

Figure 1 *Synoptique du système*

Mise en oeuvre : 1 - assemblage et chargement du programme
2 - chargement et lecture des mémoires du processeur.

Le logiciel de production de programme est implanté dans le calculateur (Figure 1). Le programme source est édité sur l'unité de disques souples. Celui-ci est alors envoyé au calculateur par une ligne asynchrone afin d'en permettre l'assemblage et le chargement. La liste d'assemblage et le programme objet obtenus sont ensuite renvoyés à l'unité de disques souples. Il est alors possible de charger les différentes mémoires du processeur par l'intermédiaire de l'interface (Fig.2). Le fonctionnement de l'interface est examiné au paragraphe I-2.4.

I-2 Le processeur :

I-2.1 Définition

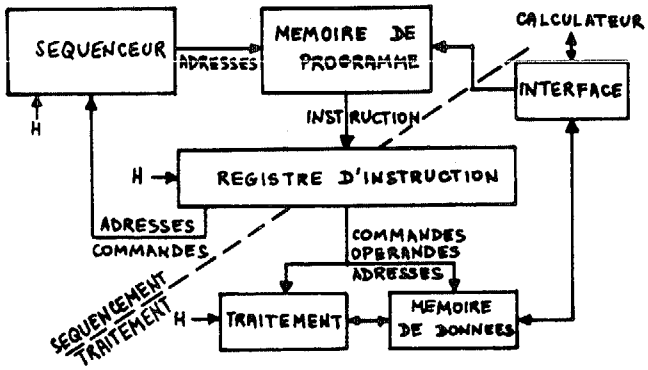


Figure 2 *Architecture générale du processeur*

La volonté de réaliser un processeur rapide grâce à une architecture adaptée et néanmoins facile à programmer a conduit à choisir un microprocesseur bipolaire (vitesse) en tranches (architecture et jeu d'instructions adapté), un multiplieur intégré (vitesse et simplicité de programmation) et des mémoires rapides.

Les données sont sur 16 e.b pour la précision des calculs. L'architecture adoptée (Figure 2) est du type recouvrement à un niveau. Elle constitue, dans la plupart des cas, le meilleurs compromis entre vitesse et simplicité de programmation [1].

En ce qui concerne l'appel de la microinstruction suivante, le choix s'est porté sur une cadence fixe afin de réaliser un fonctionnement de type synchrone. Une microinstruction est ainsi exécutée à chaque coup d'horloge. La période d'horloge est de 200 ns.

I-2.2 Architecture

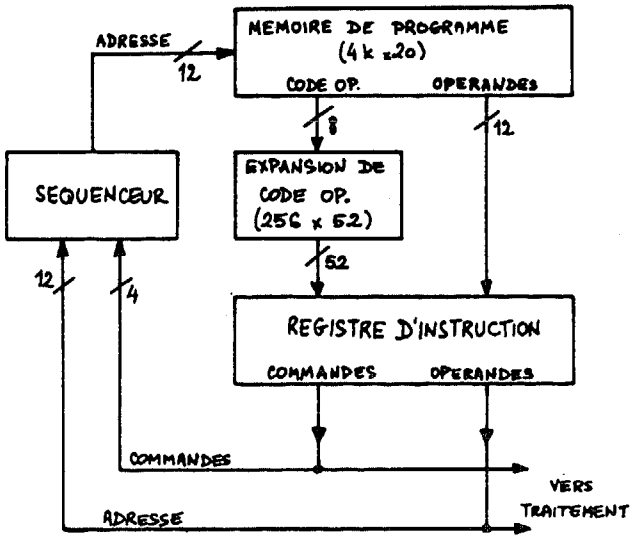


Figure 3 *Partie séquencement*

REALISATION MATERIELLE ET LOGICIELLE D'UN SYSTEME DE TRAITEMENT DU SIGNAL - APPLICATIONS: CALCUL DE FFT, RESOLUTION DE L'EQUATION DE RICCATI

Il est possible de diviser l'architecture en deux parties (Figure 2).
La première correspond à l'unité de commande. C'est la partie séquencement. La seconde est la partie de traitement. La liaison entre les deux est faite par le registre d'instruction.

La partie séquencement gère l'enchaînement des instructions que doit exécuter la partie traitement. Celle-ci fournit donc à celle-là les opérandes nécessaires aux calculs ainsi que les commandes de tous les organes qu'elle contient.

Le processeur construit étant un appareil de laboratoire donc susceptible de servir dans différentes applications, les mémoires sont des mémoires vives. En fonctionnement normal, elles sont à lecture seule. Leur système de chargement, déjà évoqué, sera décrit au paragraphe I-2.4.

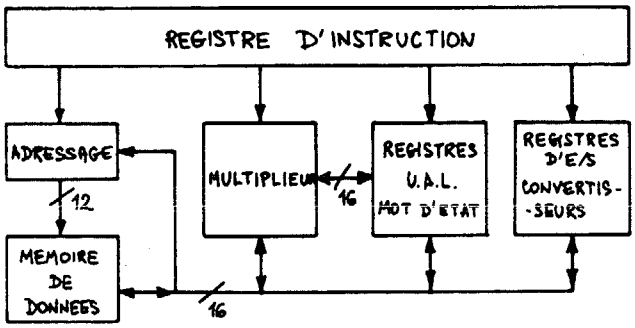


Figure 4 *Partie traitement*

Le rôle de la partie traitement (Figure 4) est d'effectuer des calculs entre les données présentes dans sa mémoire et les opérandes venus de la partie séquencement, donc de l'instruction de programme.
Elle fait également l'entrée du signal et la sortie des résultats (★). Les organes sont liés par un bus de données de 16e.b. Il existe un second bus de données entre le multiplieur et l'U.A.L., afin de permettre le chargement simultané des deux opérandes d'une multiplication.

Afin de faciliter la gestion de diverses zones de données, le système d'adressage de la mémoire est constitué de quatre bases et d'un index d'une part et de deux registres permettant de mémoriser un déplacement d'autre part [2].

$$\text{Adresse effective} = \left\{\begin{matrix}\text{Base}\\\text{Index}\end{matrix}\right\} + \text{déplacement} \left\{\begin{matrix}\text{immédiat}\\\text{calculé}\end{matrix}\right\}$$

I-2.3 Microprogrammation

Le codage choisi pour les différentes instructions est une solution intermédiaire entre le codage par champs indépendants et le codage de type instruction. Cela permet de conserver les avantages du premier (décodage simple, simultanéité des tâches, microprogrammes optimisés) sans se priver de ceux du second (format court, accès aisé, jeu d'instructions et langage assembleur).

L'instruction comprend 20e.b. : 8 pour le code opératoire et 12 pour les opérandes. Dans les structures classiques d'ordinateurs, une instruction correspond à un sous-programme, c'est-à-dire une suite de microinstructions. Dans le système décrit ici, une instruction est décodée en une seule microinstruction de 64e.b.

Ainsi le processeur exécute 5 millions d'instructions par seconde. De plus, chaque instruction bénéficie d'un format relativement court et d'un accès aisé, grâce aux mnémoniques d'un langage assembleur.

Le décodage est réalisé par une mémoire d'expansion de codes contenant 256 mots de 52e.b. car les opérandes sont transmis directement (Figure 3).

Le jeu d'instructions, parfaitement adapté à l'architecture décrite ci-dessus, a été élaboré en tenant compte des besoins rencontrés dans le traitement du signal. Les instructions peuvent être classées en cinq familles :

- Transferts
- Instructions arithmétiques
- Instructions logiques
- Sauts et branchements
- Instructions diverses (adressage, valeur immédiate, compteur de boucles,...).

L'examen de divers types d'instructions montre un choix délibéré de non parallélisme de traitement et de séquencement afin de faciliter la programmation, alors que l'architecture et le type de codage adopté l'auraient permis.
La figure 5 présente les différents formats des instructions.

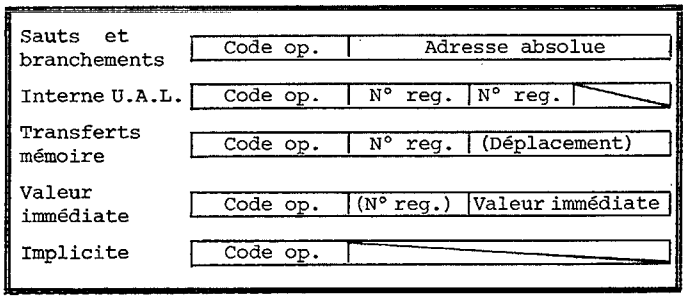


Figure 5 *Format des instructions*

architecture :	recouvrement à un niveau
vitesse :	5 MIPS
mémoire de programme :	4K mots de 20 bits
mémoire de données :	4K mots de 16 bits
microinstruction :	64 bits (champs indépendants)
registres internes :	17

162 circuits intégrés (TTL , HMOS)
3 cartes (format double européen)

Figure 6 *Caractéristiques du processeur*

I-2.4 Liaisons avec l'extérieur

L'utilisateur dispose de deux moyens de dialogue avec le processeur : le tableau de commandes (Fig.1) et le système d'interface de chargement des mémoires (Fig.1, 2).

Matériel d'aide à la mise au point, le tableau de commandes autorise, outre le fonctionnement normal, l'arrêt sur instruction, l'arrêt sur adresse de programme, le fonctionnement en pas à pas avec visualisation des registres de l'U.A.L. et du contenu des mémoires. Il est également possible de charger ces dernières manuellement.
L'interface renferme un microprocesseur monolithique et assure les transferts entre les disques et les mémoires du processeur. Il permet le chargement automatique de toutes les mémoires du processeur et la lecture automatique de la mémoire de données.

(★) par interruption

REALISATION MATERIELLE ET LOGICIELLE D'UN SYSTEME DE TRAITEMENT DU SIGNAL - APPLICATIONS: CALCUL DE FFT, RESOLUTION DE L'EQUATION DE RICCATI

La teneur des messages échangés obéit à des règles précises. Ils débutent par un en-tête identifiant la mémoire, puis les adresses à charger ou à lire. L'interface s'occupe presque en totalité de la gestion de l'unité de disques souples (mise en lecture ou écriture et envoi des caractères de commande).

II - LE LOGICIEL DE PRODUCTION DE PROGRAMMES

II-1 Le langage symbolique :

La volonté de réaliser un processeur facile à programmer impose de développer un logiciel autorisant une production et un stockage commodes des programmes. C'est pourquoi un langage symbolique a été créé. Ses principales caractéristiques sont les suivantes :

- utilisation de codes opératoires mnémoniques
- emploi de symboles réservés pour les registres, index et bases
- désignation des adresses et opérandes par des noms symboliques
- possibilités d'initialiser la mémoire des données, de réserver des tables, d'introduire des variables en équivalence
- écriture de constantes sous forme décimale, hexadécimale ou alphanumérique
- introduction possible de commentaires
- calcul du déplacement et du contenu des bases qui servent à adresser la mémoire de données.

L'instruction se compose d'un code opératoire et de deux champs d'opérandes si nécessaire. Le premier représente (la) ou les sources des opérandes et le second (la) ou les destinations du résultat de l'opération. Le code opératoire a une représentation symbolique constituée d'une racine et éventuellement de préfixes et de suffixes (Figure 7).

II-2 L'assembleur et le chargeur-translateur :

Ecrit en langage FORTRAN pour le Calculateur IRIS 50 du Centre de Calcul, l'assembleur délivre un programme objet translatable à partir du programme symbolique source lu en un seul passage.

Un programme source se compose d'une section de données et d'une section de programme, définies par des directives de sectionnement. L'adressage des éléments de l'une ou l'autre de ces sections ne peut se faire que de façon symbolique, pour assurer la translatabilité des modules.

Quatre directives permettent la génération de données (MOT, TABLE), et la définition de symboles (EQU, VAL).

La directive USE permet à l'utilisateur d'informer l'assembleur qu'il va modifier le contenu d'une base en la chargeant par une valeur immédiate. L'assembleur peut alors gérer correctement les bases

Compte tenu des problèmes qui se sont posés jusqu'à présent, il n'a pas été introduit de références externes, mais cette possibilité a été réservée.

L'assembleur édite une liste d'assemblage en signalant les erreurs éventuelles. En fin d'assemblage, il vérifie que toutes les références ont été résolues et produit :

- l'image translatable de la mémoire de données (4 octets par mot)
- l'image translatable de la mémoire de programme (5 octets par mot)
- la table de références programme.

Le chargeur-translateur effectue le chargement séparé des données et du programme à partir des adresses indiquées par l'utilisateur. Le résultat est un fichier disque qui est directement transféré sur l'unité de disques souples (Figure 1).

III - LE SIMULATEUR

Le simulateur est un programme écrit en langage FORTRAN, réalisé pour faciliter la mise au point de la machine, puis des programmes.

Partant d'un module produit par le chargeur-translateur, le simulateur exécute les instructions du programme.

Comme sur la machine réelle, il est possible de modifier le contenu d'un registre, d'une base, du mot d'état, etc... ou d'un mot de la mémoire de données ou de programme. Il est possible de fonctionner en pas à pas, en éditant le contenu des registres, bases, mot d'état, etc... ou d'effectuer une séquence avec arrêt sur instruction.

Compte tenu des possibilités du calculateur IRIS 50, le simulateur n'est pas conversationnel, mais cette option est prévue.

Le simulateur s'est révélé très utile lors de la mise au point de la machine, car il permet de comparer les résultats obtenus par deux voies différentes.

Accessoirement, il permet de compter les instructions exécutées et de suivre le déroulement d'un programme, ce qui est difficile sur la machine réelle.

IV - EXEMPLES D'APPLICATIONS

IV-1 Transformée de Fourier :

L'algorithme choisi fait appel aux techniques des transformées de Fourier rapides et des filtres numériques. En effet, la transformée de Fourier discrète peut être représentée par un type de filtre qui minimise le nombre de coefficients utiles [3]. Il existe une relation directe entre un tel filtre et les transformées de Fourier rapides classiques. Ceci permet de diviser par deux le nombre de multiplications qui devient alors $N \ell g_2 N$, N étant le nombre de points sur lequel porte la transformée.

L'écriture du programme nécessite 60 instructions en FORTRAN ce qui correspond à 185 instructions en assembleur.

Les résultats obtenus sont les suivants :

N = 8 : t = 42 μs
N = 16 : t = 110 μs
N = 256 : t = 3,25 ms

Ce dernier résultat montre que, si dans une application quelconque le calcul de la transformée de Fourier est seul nécessaire, l'échantillonnage du signal peut atteindre une fréquence de 80 kHz.

Autre façon d'interpréter ce résultat :
dans une application telle que le traitement du signal vocal où l'échantillonnage se fait à 8kHz, le calcul de la transformée de Fourier ne représente que 10% du temps réel. Un traitement complexe peut donc prendre place à la suite de celle-ci.

La figure 6 présente un extrait de la liste du programme de transformée de Fourier.

REALISATION MATERIELLE ET LOGICIELLE D'UN SYSTEME DE TRAITEMENT DU SIGNAL - APPLICATIONS: CALCUL DE FFT, RESOLUTION DE L'EQUATION DE RICCATI

Adresses	Codes	Labels	Code op.	Opérandes	Commentaires
023	DB000		PSQ	Q	initialisation du compteur de boucle
024	1CD00	DEBUT	ADIS	0,RD	Chargement de S; S = RD + 0
025	70100		TMR	T,S R1	Transfert mémoire (T,S) → R1
026	3C100		TMX	R1,COEF	Transferts opérandes dans multiplieur
027	3B000		MP		Multiplication
028	06220		RADX	R2 R2	R2 = (R2+rés. mult)/2
029	18DE0		ADRS	RE,RD	Chargement de S; S = RE + RD
02A	61200		TRM	R2 U,S	Transfert R2 → → mémoire (U,S)
02B	9EE04		ADI	4,RE RE	RE = RE + 4
02C	DED01		BCNE	1,RD RD	RD = RD + 1; Fin de boucle DEBUT
02D	15000		INCY		COEF = COEF+1
02E	8CDA0		ADR	RA,RD RD	RD = RD + RA
02F	84FC0		SUR	RC,RF	RC - RF
030	E4024		JNC	DEBUT	Saut à DEBUT si non nul.

Figure 7 *Extrait du programme de transformée de Fourier*

IV-2 Résolution de l'équation de Riccati :

Le traitement du signal dans ses aspects stochastiques a pour base l'équation de Riccati. Aussi il est intéressant de voir quelles sont les performances réalisées par le processeur dans la résolution de ce problème. Les algorithmes [4] utilisés sont ceux de quasi-linéarisation et de Lindquist avec des matrices d'abord sous forme normale puis sous forme compagne. L'application porte sur le cas simple d'ordre 2 à sortie scalaire mais les calculs sont effectués en arithmétique double longueur. Cette étude permet donc de tester le processeur et le jeu d'instructions dans des conditions totalement différentes de la précédente.
Voici les résultats obtenus :

Algorithme	Quasilinéarisation		Lindquist	
	Forme normale	Forme compagne	Forme normale	Forme compagne
Taille du programme (Nbre d'instructions)	597	507	481 + 138(init)	440 + 129(init.)
Temps d'exécution (ms)	7,32	1,15	22,3	110

CONCLUSION

Les algorithmes de traitement numérique du signal posent le problème de leur exploitation en temps réel. Le but de cette étude est la conception et la réalisation d'un système capable d'apporter une solution à ce problème.

Ce système se compose d'un processeur rapide, d'un logiciel de production de programmes, d'un simulateur et et d'une interface de chargement automatique à partir de mémoires de masse non volatiles.

Le processeur est bâti autour de macrocomposants, d'un multiplieur intégré et de mémoire vives rapides. Grâce à une architecture et un mode de programmation adapté, il a des performances satisfaisantes en particulier en vitesse : exécution de cinq millions d'instructions par seconde. Il y a identité entre instruction et microinstruction. Cela permet à l'utilisateur de disposer d'un jeu d'instructions. A chacune de ces dernières est associé un mnémonique. Ainsi l'écriture du programme source se fait en langage symbolique et le niveau de microprogrammation est totalement transparent pour le programmeur.

Ces outils qui viennent d'être décrits autorisent, à partir d'un algorithme quelconque, l'écriture du programme source en langage symbolique, la traduction en codes, le chargement du programme objet et son exécution soit en temps réel soit sur simulateur.

BIBLIOGRAPHIE

[1] G.BRUN - *Réalisation d'un système de traitement numérique du signal à base de macrocomposants.* Thèse de Docteur-Ingénieur, Laboratoire de Signaux et Systèmes, Nice, novembre 1979

[2] D.MOUSSIEGT, G.BRUN - *Un système de traitement de signal.* MECO'79, Grenoble, juin 1979

[3] G.BRUUN - *Z-transform DFT filters and FFT's.* IEEE Transactions on Acoustics, Speech and Signal Processing, vol. 26, février 1978

[4] A.GIULIERI, C.A.BOZZO - *Analyse des méthodes de résolution numérique de l'équation discrète de Riccati - Application à l'opérateur de Riccati rapide.* GRETSI, 1979.