COLLOQUE NATIONAL SUR LE TRAITEMENT DU SIGNAL
ET SES APPLICATIONS

NICE du 26 au 30 AVRIL 1977

# Techniques de filtrage numérique applicables à des microprocesseurs sans multiplication câblée

O. MAUREL

LASSY LA 190 CNRS
Université de Nice
06000 NICE

J. PATURET
D. ESTEBAN

Laboratoire IBM
06610 La Gaude (France)

## RESUME

Une approche temps réel des problèmes posés par la simulation des algorithmes de filtrage numérique est décrite. Elle résulte de la programmation dynamique sur microprocesseurs de type séquentiel - ne possédant pas la multiplication câblée - d'algorithmes d'optimisation de somme de multiplications.

Une simulation temps réel de filtre numérique appliqué au signal vocal a été réalisée sur une machine construite en technologie TTL MSI standard opérant avec un temps d'instruction moyen de 122 ns et travaillant sur des registres de 16 e.b. Le temps de multiplication moyen observé est de l'ordre de 500 ns, compte tenu de la circulation des mémoires du filtre considéré.

## SUMMARY

In this paper, a real time approach of digital filtering algorihms is presented. It is obtained thanks to the use on sequential type microprocessors -without hardwared multiplication- of dynamic assembling which allows to generate time efficient instruction set to perform any sum of multiplications.

A real time simulation of digital filter applied to voice processing has been achieved on a machine build with standard TTL MSI technology designed to operate with a basic cycle of 122 ns with 16 bits registers. The average multiplication time has been observed in the range of 500 ns taking into account the data circulation of the filter.

# Techniques de filtrage numérique applicables à des microprocesseurs sans multiplication câblée

## *Introduction*

L'apparition d'éléments fonctionnels de microprocesseurs rapides de type séquentiel exécutant jusqu'à dix millions d'instructions par seconde permet d'aborder d'une façon nouvelle les techniques de filtrage numérique temps réel.

Une des opérations le plus souvent exécutée dans le traitement du signal, la somme de multiplications, peut-être optimisée en supprimant les tests redondants, en utilisant une décomposition optimale des coefficients et en indexant la circulation des données.

Une architecture originale utilisant la technologie standard TTL MSI fonctionnant avec un temps d'instruction moyen de 122 ns sur des mots de 16 éléments binaires en a résulté. Le temps d'exécution moyen de multiplication observé est de l'ordre de 500 ns pour un produit correspondant à une donnée de 16 e.b multipliée par un coefficient de 12 e.b compte tenu de la circulation des mémoires du filtre considéré.

## *Rappel sur les techniques d'optimisation de la somme de multiplications:*

Pour formuler le problème, considérons le filtrage d'un signal d'entrée IX contenu dans une mémoire de longueur L:

$$L:\{IX(1),..IX(L)\}$$

par un filtre IC dont la réponse impulsionnelle est représentée par la séquence finie de quatre coefficients IC(1), IC(2), IC(3), IC(4);
IX et IC étant des variables entières.

A ce filtrage correspond l'opération de convolution exprimée par la relation

$$IR(1)=\sum_{J} IC(J)xIM(J) \qquad 1\leq J\leq 4 \quad (1)$$

où IR est une mémoire de même dimension que IX contenant le signal filtré.

La relation (1) peut-être évaluée par la séquence d'instructions de type PL/1 donnée dans la figure 1.a, où ACC est l'accumulateur du processeur et IM une mémoire de travail de même dimension que IC utilisée pour la circulation des données et la mémorisation des échantillons de bloc à bloc.

Chaque instruction PL/1 de la figure 1.a nécessite si on suppose un microprocesseur sans multiplication câblée la répétition et l'exécution d'une séquence d'instructions correspondant à un test, une addition et un décalage, séquence dont un exemple est décrit dans la figure 1.b.

Le nombre d'instructions exécuté est directement fonction du nombre d'éléments binaires du multiplicateur et la séquence est entièrement réexécutée pour chaque nouvel appel de la boucle de filtrage.

Par exemple avec 8 e.b par coefficient, chaque multiplication nécessitant par e.b dix cycles élémentaires, le produit sommé de quatre termes en demandera 320.

Pour optimiser le programme évaluant la convolution, l'ensemble des coefficients peut être analysé en dehors de la boucle de convolution de manière à supprimer les tests superflus. D'autre part les opérations arithmétiques équivalentes à la somme des produits résultant des coefficients seront obtenues en générant une séquence minimum d'instructions spécifiques de la valeur de ces coefficients /1/.

```
DO I=1 TO L;
  IM(1)=IM(2);                 circulation des données
  IM(2)=IM(3);                 et mémorisation des
  IM(3)=IM(4);                 échantillons bloc
  IM(4)=IX(I);                 à bloc

  ACC=IC(1)xIM(1);             évaluation de la
  ACC=ACC+IC(2)xIM(2);         somme de
  ACC=ACC+IC(3)xIM(3);         multiplications
  ACC=ACC+IC(4)xIM(4);

  IR(I)=ACC;                   mémorisation du résultat
END;
```

*a) Instructions de type PL/1*

| | | | Nombre de cycles |
|---|---|---|---|
| * IM(1)=IM(2); | | | |
| L | IM(2) | ACC=IM(2) | 2 |
| ST | IM(1) | IM(1)=ACC | 2 |
| * ACC=ACC+IC(2)xIM(2); | | | |
| ST | SAVE | SAVE=ACC | 2 |
| L | IC(2) | ACC=IC(2) | 2 |
| STR | XR1 | XR1=ACC | 1 |
| L | IM(2) | ACC=IM(2) | 2 |
| STR | XR2 | XR2=ACC | 1 |
| MULT | XR1,XR2 | XR1xXR2 | 60-80 |
| A | SAVE | RESULTAT+SAVE | 2 |
| | | | 74-94 |

où Mult est une macro instruction de multiplication standard

*b) Exemple d'instructions machine équivalentes*

Figure 1

Un exemple de cette méthode est donné à la figure 2.a pour des coefficients de 8 e.b. Ils ont été décomposés sous forme canonique minimale, cette forme permet un gain moyen de 30% d'opérations par rapport à la représentation en complément à 2 /2/.

On peut constater sur l'exemple de la figure 2 que l'évaluation de cette somme de multiplications demande environ 30 cycles machines, ce qui correspond à un gain de 1 à 10 comparé à la même opération exécutée avec une macro-instruction multiplication standard.

```
                    2^7 2^6 2^5 2^4 2^3 2^2 2^1 2^0
IC(1)=   -8 :   1   1   1   1   1   0   0   0
IC(2)=   14 :   0   0   0   0   1   1   1   0
IC(3)=  -34 :   1   1   0   1   1   1   1   0
IC(4)=   81 :   0   1   0   1   0   0   0   1
```

Représentation binaire des coefficients

```
    0   0   0   0-1   0   0   0
    0   0   0   1   0   0-1   0
    0   0-1   0   0   0-1   0
    0   1   0   1   0   0   0   1
```

Représentation sous forme canonique minimale

*a) Exemple de représentation des coefficients sous forme canonique minimale*

```
DO I=1 TO L;

  IM(1)=IM(2);                 Circulation des données
  IM(2)=IM(3);                 et mémorisation des
  IM(3)=IM(4);                 échantillons bloc
  IM(4)=IX(I);                 à bloc

  ACC=IM(4);                   évaluation de la
  ACC=ACC+ACC-IM(3);           somme
  ACC=ACC+ACC+IM(4)+IM(2);
  ACC=ACC+ACC-IM(I);
  ACC=ACC+ACC;
  ACC=ACC+ACC-IM(2)-IM(3);
  ACC=ACC+ACC+IM(4);

  IR(I)=ACC;                   mémorisation du résultat

END;
```

*b) Exemple des opérations équivalentes*

| *ACC=ACC+ACC-IM(3); | | | nombre de cycles |
|---|---|---|---|
| SLL | ACC | ACC=ACC+ACC | 1 |
| S | IM(3) | ACC=ACC-IM(3 | 2 |
| | | | 3 |

Figure 2

# Techniques de filtrage numérique applicables à des microprocesseurs sans multiplication câblée

### *Architecture de simulation en temps réel de filtrage numérique appliqué au signal vocal:*

a) **Description générale**

Cette architecture a été réalisée pour mettre en œuvre un filtre microprogrammé à coefficients entiers ou ICMF (Integer Coefficients Microprogrammed Filter).

L'ICMF effectue les opérations nécessaires à l'émulation de filtres numériques dont les coefficients sont décomposés sous la forme canonique minimale signée /1/.

L'architecture de l'ICMF se distingue par trois principales caractéristiques: (voir figure 3).

- L'utilisation d'une unité arithmétique et logique ou ALU réalisée à l'aide d'éléments standards de technologie TTL en couches de 4 e.b par 4 e.b associée à un registre général ou GR de 1024 positions de 16 eb.
- L'utilisation d'un adressage indexé permettant de générer et faire circuler les adresses absolues associées au registre général sans modifier l'adresse ou le déplacement fourni par le programme de génération des filtres numériques.
- L'utilisation de trois programmes de contrôle indépendants et asynchrones:
  - le programme d'entrée/sortie ou IOP (Input-Output program) qui assure la gestion des mouvements des échantillons sur les bus d'entrée/sortie,
  - le programme arithmétique ou AP (arithmétic program) qui traite les opérations évaluant les filtres,
  - le programme superviseur ou SP (supervisor program) qui gère l'enchaînement des différents programmes arithmétiques.

Cette architecture spécialement définie pour évaluer les multiplications rapides nécessaires dans le traitement du signal permet d'exécuter une multiplication d'une donnée de 16 e.b par un coefficient de 12 e.b en un temps moyen de 500 ns par l'utilisation des techniques précédemment décrites.

D'autre part, elle a été organisée de manière à ce que les différents programmes de contrôle travaillent de façon asynchrone et indépendante sous le contrôle du superviseur afin d'obtenir le maximum d'efficacité.

b) **Description fonctionnelle**

L'opération de filtrage exprimée par la relation précédente (1), nécessite:

- la mémorisation des échantillons et leur circulation en mémoire conformément à la période d'échantillonnage,
- la mémorisation de la séquence d'opérations résultant de la décomposition sous forme canonique minimale équivalente aux coefficients des multiplications.

L'ICMF est organisée de façon telle que:

- les échantillons correspondant à un filtre donné, auquel est assigné un numéro de voie, sont mémorisés dans le registre général soit sous le contrôle de l'IOP soit sous celui de l'AP à des adresses déterminées à l'aide d'un pointeur spécifique du filtre considéré, l'adresse relative d'un échantillon ou déplacement est fournie par l'IOP ou par l'AP.

  L'ICMF peut gérer jusqu'à 32 pointeurs differents qui sont mémorisés dans le générateur dynamique d'adresse ou DAG (Dynamic Address Generate).
- La circulation et le décalage des échantillons sont effectués sans modification ni des pointeurs ni des déplacements (/3/,/4/)

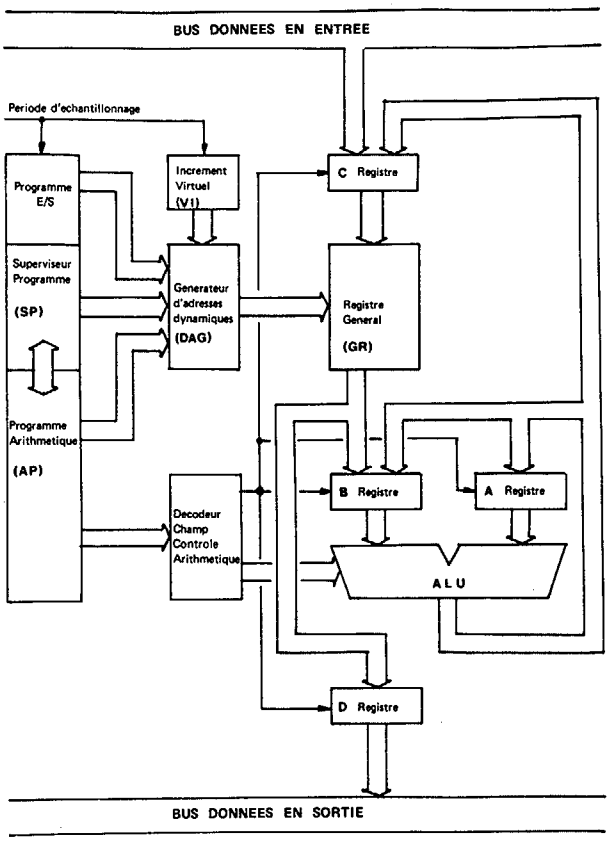


Figure 3

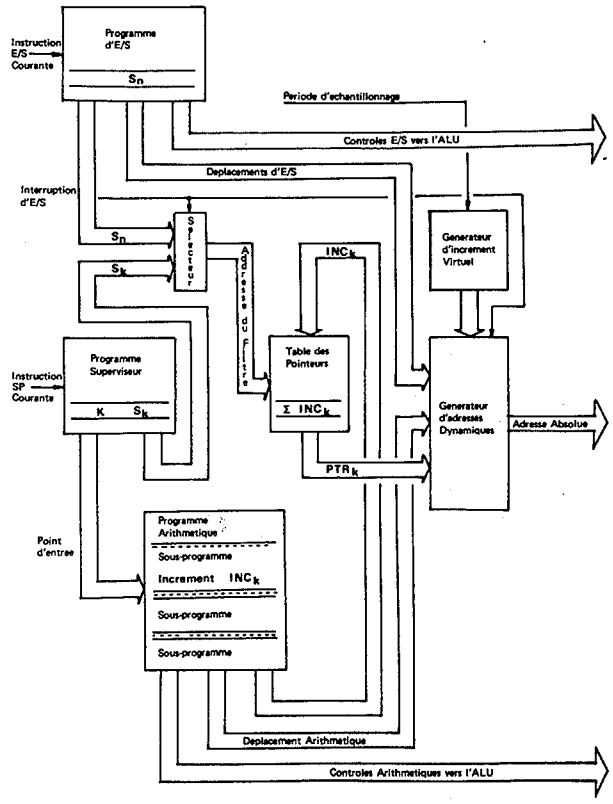


Figure 4

# Techniques de filtrage numérique applicables à des microprocesseurs sans multiplication câblée

grâce à l'utilisation d'un incrément virtuel on VI qui est incrémenté sur la base d'une période d'échantillonnage. Une adresse absolue dans le GR est déterminée à l'aide de la relation suivante:

$$AA(K)=[PTR(K)+DPL(K)+VI]_{MOD} \quad (2)$$

où

PTR(K) est la valeur du pointeur du filtre "K"

DPL(K) le déplacement dans le filtre "K"

VI l'incrément virtuel géré par le DAG et incrémenté à chaque période d'échantillonnage

MOD la valeur du modulo utilisé pour la circulation des adresses

AA(K) l'adresse absolue dans le GR.

- les échantillons sont lus et écrits de manière synchrone par rapport à la période d'échantillonnage et sous le contrôle de l'IOP qui opère selon la technique du vol de cycles mais de manière asynchrone par rapport à l'unité de traitement arithmétique, un échantillon de 16 e.b étant traité en un seul cycle machine.

  Chaque échantillon est atteint au moyen de son déplacement et par l'utilisation d'un pointeur donné pour le filtre (K):

  XIO(K,DPL) pour un ordre de lecture

  ou

  d'écriture

- les pointeurs associés au numero de filtre sont contenus dans une table, dite table des pointeurs, sont determinés en séquence et sont mis à jour à la fin d'une exécution d'un A.P, par l'addition au pointeur courant d'un incrément INC fourni par l'AP et correspondant au nombre de positions du registre général GR occupé par le filtre considéré, voir figure 4:

  $$PTR(K)=PTR(K-1)+INC(K) \quad (3)$$

- la generation dynamique de la somme des multiplications par les coefficients est exécutée par un groupe d'instructions rangé dans l'AP représentant la décomposition des coefficients en une séquence minimale d'opérations élémentaires d'additions, de soustractions et de décalages.

  Cette décomposition est obtenue grâce à un assembleur qui a été spécialement étudié à cet effet, un exemple est donné dans la table 1.

c) **Description des programmes de contrôle**

- IOP (Programme d'entrée-sortie)

  L'IOP est rangé dans une mémoire programmable accessible en lecture seulement (PROM) et contrôle les transferts d'échantillons en entrée/sortie du registre général, une instruction de l'IOP et exécutée suivant la technique du vol de cycle en un seul cycle machine.

  Le format d'une instruction IOP est donné dans la table suivante:

| Numéro de voie | Champ de contrôle | Déplacement |
|---|---|---|
| 4 e.b | 5 e.b | 4 e.b |

  L'amplitude du déplacement a été limitée à 16 pour des raisons pratiques.

- AP (programme arithmétique)

  La taille de l'AP est limitée à 1024 instructions de base rangées dans une PROM. Chaque procédure spécifique à un filtre est réentrante en série, elle est accessible par un point d'entrée associé au filtre fourni par le programme superviseur.

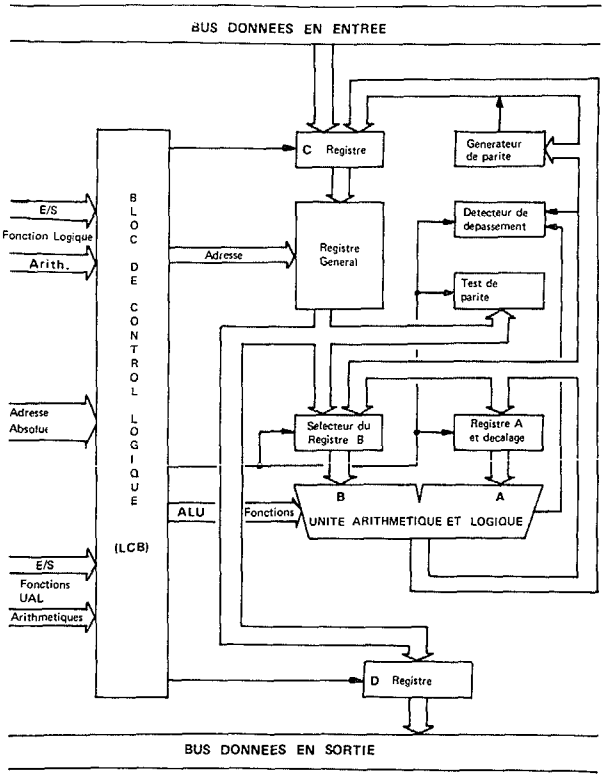


Figure 5

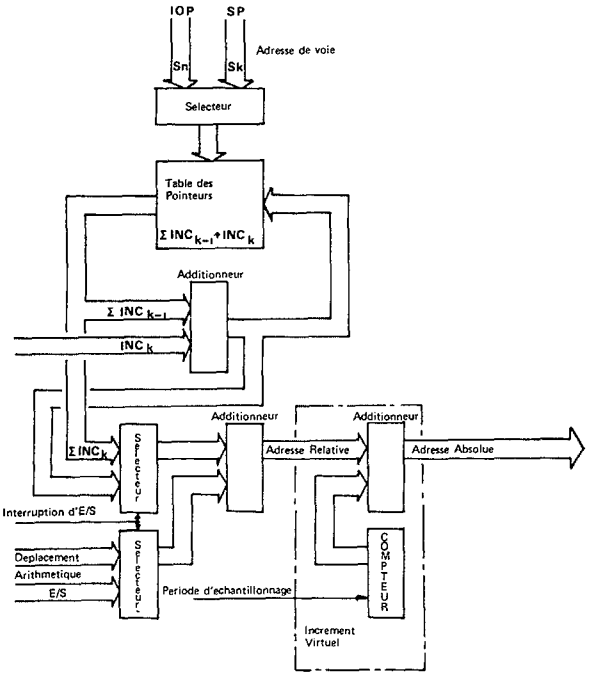


Figure 6

# Techniques de filtrage numérique applicables à des microprocesseurs sans multiplication câblée

Le format d'une instruction de l'AP est le suivant:

| Deplacement/ Increment | Contrôle de l'opération arithmétique |
|---|---|
| 10 e.b | 9 e.b |

Le premier champ sert soit comme déplacement ou incrément suivant le champ de contrôle.

Le champ de contrôle arithmétique est décodé par le bloc de contrôle logique ou LCB et génère toutes les lignes de contrôle d'unité arithmétique et logique ALU (voir fig. 5).

- SP (programme superviseur)

Le programme superviseur est rangé dans une PROM et distribue séquentiellement les filtres comme spécifié par l'utilisateur en associant à un filtre donné un numéro de voie.

Le format des instructions SP est donné par la table suivante:

| Point d'entrée | Numéro de voie | Champ de contrôle |
|---|---|---|
| 10 e.b | 4 e.b | 1 e.b |

Au début de chaque période d'échantillonnage le SP est reinitialisé afin de synchroniser le traitement.

d) **Description des blocs fonctionnels**

DAG (générateur d'adresse dynamique)

Le DAG est construit autour d'une table qui contient les pointeurs associés aux différentes voies.

Le pointeur, pour une voie sélectée, est additionné au déplacement fourni soit par l'IOP, soit par l'AP pour déterminer l'adresse relative ou RA (voir figure 6).

$$RA(K)=PTR(K)+DPL(K) \quad (4)$$

Cette adresse est ajoutée au contenu du compteur de l'incrément virtuel (VI) de façon à déterminer l'adresse absolue (AA) à l'aide de la relation (3).

La circulation des échantillons contenus dans le registre général est obtenue au moyen de l'incrément virtuel auquel est ajouté 1 au début de chaque période d'échantillonnage.

ALU (unité arithmétique et logique)

Le diagramme d'ensemble de l'ALU est décrit dans la figure 5, l'unité est composée de:

* une unité arithmétique et logique de 16 e.b
* une mémoire de 16 e.b à accès aléatoire de 1024 positions (GR)
* un bloc de contrôle logique LCB qui décode le champ de l'instruction.
* différents registres permettant la mémorisation des échantillons en entrée/sortie et le contrôle du flot de données
* des circuits supplémentaires permettant la génération et le test de la parité des échantillons, le contrôle du dépassement de capacité.

L'ALU travaille avec un cycle de base de 122 ns et exécute le jeu d'instructions dont un exemple est donné en annexe.

## *Conclusion*

Les techniques d'optimisation des algorithmes de filtrage numérique décrites ont permis de réaliser une architecture spécialisée organisée autour d'un microprocesseur ne possédant pas de multiplication câblée.

Le temps moyen de multiplication observé étant de l'ordre de 500 ns alors que le cycle d'instructions moyen est de 122 ns, il a été possible de réaliser les applications suivantes:

* multiplexage d'un maximum de 16 filtres
* décimation et interpolation d'une transmission M.I.C pour modifier sa fréquence d'échantillonnage
* réalisation d'un groupe de filtres numériques pour décoder un signal de réception à fréquence multiple (Multifrequency Receiver MFR).

| Coef. / Poids | 7 | ¯15 | 23 | ¯15 | 7 |
|---|---|---|---|---|---|
| 16 | 0 | ¯1 | 1 | ¯1 | 0 |
| 8 | 0 | 0 | 1 | 0 | 0 |
| 4 | 1 | 0 | 0 | 0 | 1 |
| 2 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | ¯1 | 1 | 1 |
| Adresses | 1 | 2 | 3 | 4 | 5 |

```
     ∇EXAMPLE[⎕]∇
    ∇ R←EXAMPLE
[1]   START
[2]   A 1..... 'ADD ADDRESS 1'
[3]   A 2......'ADD ADDRESS 2'
[4]   S 3......'SUBTRACT ADDRESS 3'
[5]   A 4......'ADD ADDRESS 4'
[6]   ASH 5....'ADD AND SHIFT ADDRESS 5'
[7]   A 2......'ADD ADDRESS 2'
[8]   ASH 5....'ADD AND SHIFT ADDRESS 5'
[9]   A 1......'ADD ADDRESS 1'
[10]  ASH 5....'ADD ADDRESS 5'
[11]  ASH 3....'ADD AND SHIFT ADDRESS 3'
[12]  S 2......'SUBTRACT ADDRESS 2'
[13]  A 3......'ADD ADDRESS 3'
[14]  S 4......'SUBTRACT ADDRESS 4'
[15]  R←END
    ∇
```

Table 1

## *Annexe*

1) Liste des instructions.
2) Un exemple d'un programme de filtre par décimation:
   tableau des coefficients
   listing des programmes

## *Table des instructions*

Nous ne présentons ici que quelques-unes des instructions les plus usitées.

*Note:* + et - ont ici le sens de plus et de moins arithmétiques, X signifie non applicable, addr une adresse mémoire. Quand Addr apparaît dans une instruction, cela implique que deux opérations sont exécutées dans une seule instruction: le contenu de la mémoire à l'adresse 'Addr' est chargé dans un registre B, l'opération définie par le mémonique de l'instruction est executée entre A et B (A = accumulateur, B, F registres).

| | | |
|---|---|---|
| CLA | X | mise à 0 de l'accumulateur A avec F=0 |
| A | Addr | addition de B à A (A+B) |
| S | Addr | soustraction de B à A (A-B) |

# Techniques de filtrage numérique applicables à des microprocesseurs sans multiplication câblée

| | | |
|---|---|---|
| AB1 | Addr | correspond à (A+B+1) |
| AB1 | Addr | correspond à (A-B-1) |
| A1 | X | addition de 1 à A (A+1) |
| S1 | X | soustraction de 1 à A (A-1) |
| NOP | X | pas d'opération, le contenu de A est sauvegardé [un registre de sortie est connecté à l'Alu F=A] |
| LB | Addr | chargement de la mémoire d'adresse 'Addr' de la mémoire au travers de l'Alu (F=B) |
| SH | X | décalage d'une position de l'accumulateur à droite |
| ASH | Addr | addition de B à A dans A puis décalage d'1 position à droite ((A+B)/2) |
| SSH | Addr | soustraction de B à A dans A puis décalage d'1 position ((A-B)/2) à droite |
| ST | Addr | rangement de A à l'adresse mémoire Addr en passant par l'Accu (F=A) |
| WRI | | écriture de l'index mémoire RAM sans opération arithmétique (F=A) |
| INC | In | transmission de l'incrément In du filtre selecté Fn au superviseur sans opération arithmétique (F=A) |
| STOP | X | incrémentation du compteur adresse superviseur (SAC) changement de l'index (F=A). |

*LIST EXTRAPOLATOR*

| | ADDRESS | OP. | CTL | S. | MNEMONICS | |
|---|---|---|---|---|---|---|
| [1] | 1111111111 | 10111 | 111 | 11 | CLA | 0 |
| [2] | 0000011001 | 11100 | 001 | 11 | S | 25,UDS |
| [3] | 0000010111 | 11100 | 011 | 11 | S | 23,WRI |
| [4] | 0000011011 | 10110 | 100 | 11 | NOP | 27,INC |
| [5] | 0000010110 | 11100 | 111 | 11 | S | 22 |
| [6] | 0000000110 | 11101 | 111 | 11 | A | 6 |
| [7] | 0000010001 | 11100 | 111 | 11 | S | 17 |
| [8] | 0000010000 | 11100 | 111 | 11 | S | 16 |
| [9] | 0000001110 | 01001 | 111 | 11 | SSH | 14 |
| [10] | 0000000100 | 10100 | 111 | 11 | LBK | 4 |
| [11] | 0000011010 | 10000 | 111 | 11 | STRA | 26 |
| [12] | 0000011001 | 11100 | 111 | 11 | S | 25 |
| [13] | 0000011000 | 11101 | 111 | 11 | A | 24 |
| [14] | 0000010100 | 11101 | 111 | 11 | A | 20 |
| [15] | 0000010011 | 11101 | 111 | 11 | A | 19 |
| [16] | 0000001111 | 11101 | 111 | 11 | A | 15 |
| [17] | 0000001110 | 11100 | 111 | 11 | S | 14 |
| [18] | 0000001101 | 01010 | 111 | 11 | ASH | 13 |
| [19] | 0000011000 | 11101 | 111 | 11 | A | 24 |
| [20] | 0000010101 | 11101 | 111 | 11 | A | 21 |
| [21] | 0000010010 | 11101 | 111 | 11 | A | 18 |
| [22] | 0000001111 | 01010 | 111 | 11 | ASH | 15 |
| [23] | 0000010111 | 11100 | 111 | 11 | S | 23 |
| [24] | 0000010000 | 01001 | 111 | 11 | SSH | 16 |
| [25] | 0000010110 | 11101 | 111 | 11 | A | 22 |
| [26] | 0000010100 | 11101 | 111 | 11 | A | 20 |
| [27] | 0000010011 | 11101 | 111 | 11 | A | 19 |
| [28] | 0000010001 | 01010 | 111 | 11 | ASH | 17 |
| [29] | 0000010101 | 11100 | 111 | 11 | S | 21 |
| [30] | 0000010010 | 01001 | 111 | 11 | SSH | 18 |
| [31] | 0000010100 | 11101 | 111 | 11 | A | 20 |
| [32] | 0000010011 | 01010 | 111 | 11 | ASH | 19 |
| [33] | 0000000110 | 01010 | 111 | 11 | ASH | 6 |
| [34] | 1111111111 | 10110 | 111 | 11 | NOP | 0 |
| [35] | 0000000001 | 10011 | 111 | 11 | ST | 1 |
| [36] | 0000000010 | 10101 | 111 | 11 | LB | 2 |
| [37] | 0000001101 | 01101 | 111 | 11 | STR | 13 |
| [38] | 1111111111 | 11110 | 010 | 10 | T | 0,STOP |

| CK | ADDR. | MATRICE: | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 13 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 0 |
| -3 | 14 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -1 |
| 6 | 15 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| -9 | 16 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | -1 |
| 15 | 17 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | -1 |
| -28 | 18 | 0 | 0 | -1 | 0 | 0 | 1 | 0 | 0 |
| 82 | 19 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 129 | 6 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| -82 | 20 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| -28 | 21 | 0 | 0 | -1 | 0 | 0 | 1 | 0 | 0 |
| 15 | 22 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | -1 |
| -9 | 23 | 0 | 0 | 0 | 0 | -1 | 0 | 0 | -1 |
| 6 | 24 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| -3 | 25 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -1 |
| 2 | 26 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

## *Références*


(1) C.V. Smith
"Electronic Digital Computers" chap. I, p. 20
Mc Graw Hill, New-York, 1959

(2) A. Peled
"On the Hardware Implementation of Digital Signal Processors"
IEEE Trans. on Assp, vol. Assp-24, N°1, Feb 76

(3) D. Esteban, S. Hajek, M. Tubiana,
"Microprogrammed Digital Filter",
IBM Technical Disclosure Bulletin, Vol. 18,
N°10, March 1976

(4) D. Esteban
"Shift Register Implemented by Indexing a Random Access Memory"
IBM Technical Disclosure Bulletin, vol. 18 N°10, March 1976