Nouvelle méthode de conception de circuits très rapides des filtres numériques linéaires invariants

New design method for fast invariant linear digital filters

Marcel LAPOINTE
Département de Génie électrique
Faculté des Sciences et de Génie,
Université Laval,
Québec, Canada G1K 7P4

Marcel Lapointe a obtenu les diplômes de baccalauréat et de doctorat en Génie électrique à l'Université Laval à Québec, en 1985 et 1990, respectivement. Ses activités de recherches portent sur les architectures rapides en traitement numérique de signal.

Huynh HUU TUÊ
Département de Génie électrique
Faculté des Sciences et de Génie,
Université Laval,
Québec, Canada G1K 7P4

Huu Tuê Huynh est né au Viet Nam en 1943. Il a obtenu les diplômes de baccalauréat et de doctorat en Génie électrique à l'Université Laval à Québec, en 1966 et 1972, respectivement. Il est présentement professeur titulaire au département de Génie électrique de l'Université Laval, où il est responsable des recherches en traitement de signal et en communication numérique.

Paul FORTIER
Département de Génie électrique
Faculté des Sciences et de Génie,
Université Laval,
Québec, Canada G1K 7P4

Paul Fortier est né à Québec en 1959. Il a obtenu les diplômes de baccalauréat et de maîtrise en Génie électrique à l'Université Laval à Québec, en 1982 et 1984, respectivement. Il a par la suite obtenu une maîtrise en Statistiques et un doctorat en Génie électrique à l'Université Stanford en Californie, en 1987 et 1989, respectivement. Il est présentement professeur adjoint au département de Génie électrique de l'Université Laval. Ses activités de recherches portent sur le traitement de signal appliqué aux communications numériques.

RÉSUMÉ

Un nouvel algorithme dit *de l'accumulateur* est utilisé pour réaliser une architecture particulière de multiplieur, faisant appel à une arithmétique redondante. Ce multiplieur est utilisé pour concevoir des circuits très rapides de filtres numériques du second ordre de types tout zéro ou tout pôle. Une architecture en pipeline permet d'atteindre une période d'échantillonnage très courte de l'ordre du temps de traversée de quelques cellules d'addition seulement. Un certain nombre de ces filtres mis en cascade réalisent un système d'ordre quelconque. Fait remarquable, la performance est indépendante de la taille des variables et de l'ordre du système. La réalisation de ces filtres est un candidat idéal pour une implantation sur circuits VLSI.

MOTS CLÉS

Système linéaire récursif, filtre tout pôle, filtre tout zéro, algorithme de l'accumulateur, arithmétique redondante, multiplieur en tableau, pipeline.

ABSTRACT

Using a new algorithm named Accumulator Algorithm, *an architecture for flexible multipliers is proposed, exploiting redundant arithmetic. These new multipliers are then employed to design fast all-pole and all-zero digital filters. The sampling period is reduced to being of the order of a few adder cell propagation delays, because of the pipeline structure. High order filters can be obtained by cascading several lower order modules. Remarkably, the performance does not degrade with the lenght of the variables nor the filter order. The structures obtained are attractive for VLSI implementation.*

KEY WORDS

Recursive linear system, all-pole filter, all-zero filter, accumulator algorithm, redundant arithmetic, multiplier array, pipeline.

1. Introduction

Le filtrage numérique occupe une place importante dans les domaines du traitement du signal et de la commande. Jusqu'à récemment, les filtres numériques étaient réalisés, pour la plupart, par programmation sur des systèmes à base de processeurs, particulièrement des processeurs spécialisés tels les DSP (*« Digital Signal Processors »*). Avec l'avènement des circuits VLSI, qui deviennent de plus en plus populaires en raison des coûts de fabrication toujours plus bas, on peut désormais envisager l'implantation de filtres numériques sur des puces dédiées. Nous abordons ce problème dans cet article.

Une manière courante d'augmenter le débit de calcul d'un système numérique est de faire intervenir un pipeline. Celui-ci est créé en intercalant des registres entre les dispositifs du système, soit sur des liens échangeant des mots (*« world-level pipeline »*), soit sur des liens échangeant des bits (*« bit-level pipeline »*). Le deuxième type de pipeline donne normalement de meilleurs résultats mais présente de sérieuses difficultés d'implantation en ce qui a trait aux systèmes de type récursif. En effet, le fait de placer des registres à l'intérieur de la structure ajoute des délais supplémentaires à l'intérieur de la boucle de récursion, obligeant de ce fait le retard des opérations au voisinage de la boucle.

Certains auteurs résolvent le problème en restructurant l'algorithme de mise en œuvre du système (*« algorithm recasting »*). A titre d'exemple, prenons le cas très simple du filtre tout pôle du premier ordre. Normalement, l'équation aux différences de ce système s'écrit :

$$y(n) = b \cdot y(n-1) + x(n) . \tag{1.1}$$

Visiblement, un seul multiplieur suffit à réaliser le filtre. Dans la situation où un pipeline est ajouté à la structure du multiplieur, très peu de gain au débit peut être escompté. En effet les nombreux délais alors créés à l'intérieur de la structure ont pour effet de retarder la sortie du résultat, ce qui retarde d'autant l'amorce du calcul suivant de l'itération.

La fonction de transfert mise en œuvre ci-dessus peut l'être autrement en itérant plusieurs fois la relation (1.1) [1]-[3]. Ce qui donne :

$$y(n) = b^p \cdot y(n-p) + b^{p-1} \cdot x(n-p+1) + \\ + b^{p-2} \cdot x(n-p+2) + \cdots + x(n) . \tag{1.2}$$

Cette fois, le calcul en cours dépend, non pas du dernier résultat généré, $y(n-1)$, mais du p-ième résultat précédent, $y(n-p)$. On parle ici d'anticipation du calcul (*« look-ahead computation »*). Ceci laisse amplement de temps au délai de s'exécuter, sans que cela nuise globalement au débit. La restructuration de l'algorithme permet donc une augmentation importante du débit, en augmentant arbitrairement p. Mais ce faisant, le matériel nécessaire à la réalisation augmente également d'un facteur p, ce qui peut devenir vite encombrant.

Dans l'approche de la restructuration, on distingue deux voies. La première procède à une anticipation du calcul en ajoutant un îlot de pôles supplémentaires, et la seconde procède à une anticipation en ajoutant et en dispersant des pôles supplémentaires (traduction libre de *« clustered look-ahead »* et *« scattered look-ahead »*, respectivement) [3]. La première voie ne garantit pas la stabilité car des pôles peuvent se situer à l'extérieur du cercle unité. La deuxième voie, en revanche, garantit la stabilité car tous les pôles sont dans le cercle unité. Dans les deux cas, des zéros sont superposés aux pôles qui sont ajoutés, ce qui peut conduire à certaines difficultés lors de l'implantation numérique, sans compter que certains modes s'avèrent ingouvernables.

D'autres auteurs ont résolu le problème en intervenant plutôt à l'intérieur des structures arithmétiques [4]-[7]. Mentionnons que le processus de multiplication se fait habituellement en deux étapes : les produits partiels sont d'abord générés, puis sommés itérativement. Lorsque la sommation des produits partiels se fait dans un ordre croissant, la partie poids faible du résultat se stabilise au fur et à mesure que l'itération progresse. Au contraire, la sommation des produits partiels dans un ordre décroissant pourrait amener l'émergence chiffre par chiffre du résultat, poids fort en tête. Mais ceci n'est envisageable que dans la mesure où une arithmétique redondante est utilisée. En effet la propagation de la retenue dans une telle arithmétique est limitée, typiquement, à une position seulement [8]. Par conséquent la somme des produits partiels qui restent à sommer après un pas donné de l'itération, ne peut affecter la partie déjà extraite du résultat. Il arrive alors que le résultat est restitué en série avec un délai très court et indépendant de la taille des opérandes. A l'aide d'un multiplieur en tableau avec pipeline, réalisé à partir de cette méthode, un filtre tout pôle du premier ordre est réalisé en substituant le résultat du multiplieur à l'opérande multiplicateur. Ce faisant, un délai de deux

périodes d'horloge est créé à l'intérieur de la boucle de récursion, indépendamment de la taille des opérandes [6]. On obtient alors une performance considérable avec un matériel étonnamment réduit par rapport à la méthode précédente. Un autre avantage, non négligeable, est que le système n'est nullement affecté par des modes ingouvernables, comme c'était le cas précédemment.

On doit mentionner que la méthode de réalisation proposée par les seconds auteurs se fait de manière plutôt heuristique. En effet, selon cette dernière, on construit d'abord un multiplieur en tableau en utilisant des cellules spéciales d'addition, conçues en fonction d'une arithmétique redondante. Ensuite le filtre récursif est réalisé en connectant la sortie du multiplieur à l'une de ses entrées. Les caractéristiques de la réalisation du filtre sont alors fonction du multiplieur que l'on a construit. En revanche, la méthode qui est présentée dans cet article est beaucoup plus formelle [9]-[10]. Les caractéristiques du multiplieur sont d'abord décidées en fonction du filtre à réaliser. Ensuite la réalisation du multiplieur est effectuée. Celui-ci a donc plus de chance de mieux répondre aux exigences de la réalisation du filtre. Bref, notre méthode présente une meilleure souplesse de réalisation. Mentionnons de plus qu'elle fait appel à des cellules d'addition plus courantes, celles utilisées dans une arithmétique traditionnelle. Un autre point important à mentionner est que notre réalisation respecte, dans un sens large doit-on dire, les principes du paradigme systolique [11], tout comme les autres réalisations d'ailleurs. C'est un atout majeur pour une implantation sur un circuit VLSI.

Le tour d'horizon de cet article se présente comme suit. La section 2 est consacrée à l'arithmétique redondante. On y trouve la réalisation d'un additionneur faisant appel à une arithmétique redondante pour s'affranchir de la propagation de la retenue. De plus est présenté un convertisseur transformant un nombre binaire redondant dans une numération complément à deux. Un algorithme, dit de l'accumulateur, est élaboré à la section 3, puis mis en œuvre à la section 4 grâce aux dispositifs développés à la section 2. La réalisation de l'accumulateur permet alors la construction d'un multiplieur en tableau à partir duquel un filtre tout pôle du second ordre est réalisé à la section 5. Sur la même lancée, un filtre tout zéro du second ordre est réalisé à la section 6, à partir de la même structure que le filtre tout pôle. Un système d'ordre quelconque, comportant des zéros et des pôles, peut alors être implanté à partir d'une cascade de ces deux types de filtres.

2. Arithmétique redondante

Les numérations redondantes sont caractérisées par un répertoire de chiffres dont la taille est supérieure à la base. De façon générale, un nombre fractionnaire redondant est représenté comme suit :

$$X = \sum_{j=1}^{m} b^{-j} x_j \qquad (2.1)$$

$$x_j \in \{\rho_1, ..., \rho_2\}$$

avec la condition

$$\rho_2 - \rho_1 + 1 > b \,. \qquad (2.2)$$

Le répertoire des chiffres x_j est un ensemble de nombres entiers consécutifs variant entre ρ_1 et ρ_2. La constante b représente la base de la numération. Le paramètre m indique la taille du nombre. Il faut souligner que, dans la plupart des cas rencontrés en pratique, on rend le répertoire symétrique autour de zéro. On a alors :

$$x_j \in \{-\rho, ..., \rho\} \qquad (2.3)$$

avec les conditions suivantes :

$$\frac{b}{2} \leqslant \rho \leqslant b - 1 \,. \qquad (2.4)$$

L'inégalité de gauche établit la redondance tandis que l'inégalité de droite force la représentation du nombre zéro, une exception dans ce seul cas, à être unique [8]. On dit que la redondance est minimale lorsque $\rho = b/2$, et maximale lorsque $\rho = b - 1$.

Une première propriété des numérations redondantes est le fait de pouvoir représenter une valeur donnée à l'aide de plusieurs nombres, ou représentations. Par exemple, la valeur 6 pourra être représentée en base 4 de plusieurs façons, comme suit :

$$6 = (0\ 1\ 2)_4 = (0\ 2\ \bar{2})_4 = (1\ \bar{2}\ \bar{2})_4 \,.$$

Le trait indique un chiffre négatif. On peut constater que plus ρ est grand, plus il y a de façons différentes de représenter une valeur donnée, et ainsi plus la redondance est forte. On a l'habitude de mesurer la redondance par le facteur de redondance, représenté par K et défini comme suit :

$$K = \frac{\rho}{b - 1} \,. \qquad (2.5)$$

Une deuxième propriété des numérations redondantes, fondamentale pour nous, est leur faculté de s'affranchir de la propagation de la retenue lors d'une addition [8]. Un exemple d'additionneur utilisant la redondance est présenté dans ce qui suit. Cette réalisation fait appel à la numération binaire redondante (mieux connue en anglais sous l'expression : *« Signed-Digit Binary Notation »* : SDBN). Sa représentation fractionnaire est la suivante :

$$X = \sum_{j=1}^{m} 2^{-j} x_j \qquad (2.6)$$

$$x_j \in \{-1, 0, 1\} \,.$$

Étant donné qu'il y a trois chiffres dans le répertoire, il est nécessaire d'utiliser deux bits pour leur codage. Parmi toutes les possibilités, celle du tableau 1 a été choisie. Son intérêt réside dans la possibilité d'obtenir la valeur du nombre en faisant une somme pondérée de ses bits. On verra plusieurs exemples dans ce qui suit.

Chiffre b_j	Code r_j	s_j
−1	0	0
0	0	1
0	1	0
+1	1	1

Tableau 1. Code des chiffres de la numération binaire redondante

2.1. ADDITIONNEUR HYBRIDE

Un additionneur est développé pour effectuer la somme d'un nombre complément à deux avec un nombre binaire redondant. Le résultat est restitué en binaire redondant. Le processus est tel que la propagation de la retenue est limitée à une position seulement, faisant de l'additionneur un opérateur parfaitement parallèle. Mentionnons que sa structure emprunte celle d'un additionneur à stockage de la retenue.

On suppose que les variables A et B sont deux opérandes représentés en complément à deux et en binaire redondant, respectivement, comme suit :

$$A = -a_0 + \sum_{j=1}^{m} 2^{-j} a_j \qquad (2.7)$$

$$a_0, a_j \in \{0, 1\}$$

$$B = \sum_{j=1}^{m} 2^{-j} b_j \qquad (2.8)$$

$$b_j \in \{-1, 0, 1\} .$$

Ces représentations sont transformées de manière à pouvoir utiliser un additionneur à stockage de la retenue. Tout d'abord on substitue à $-a_0$ l'expression $\bar{a}_0 - 1$, où $\bar{a}_0$ est l'inverse booléen du bit a_0 :

$$A = \left(\bar{a}_0 + \sum_{j=1}^{m} 2^{-j} a_j\right) - 1 \qquad (2.9)$$
$$= A^* - 1 .$$

Les bits $\bar{a}_0$ et a_j ont été regroupés pour former un vecteur, A*. Ensuite les chiffres b_j codés sur deux bits selon le tableau 1 :

$$B = \sum_{j=1}^{m} 2^{-j} b_j = \sum_{j=1}^{m} 2^{-j}(r_j + s_j - 1) \qquad (2.10)$$

$$r_j, s_j \in \{0, 1\} .$$

Les bits r_j et s_j sont groupés séparément pour former deux vecteurs, R_B et S_B. Ce qui donne :

$$B = \sum_{j=1}^{m} 2^{-j} r_j + \sum_{j=1}^{m} 2^{-j} s_j - (1 - 2^{-m}) \qquad (2.11)$$
$$= R_B + S_B - (1 - 2^{-m}) .$$

Une addition par stockage de la retenue est effectuée sur les trois vecteurs A*, R_B et S_B. A partir de (2.16) et (2.18), on obtient :

$$A + B = A^* + R_B + S_B - (2 - 2^{-m}) \qquad (2.12)$$
$$= R_C + S_C - (2 - 2^{-m}) .$$

Les vecteurs R_C et S_C regroupent les bits retenues et sommes, respectivement, de l'addition par stockage de la retenue. On constate, à ce stade-ci, que la forme prise par le dernier membre est semblable à l'expression rencontrée dans (2.11). Ceci suggère que les vecteurs R_C et S_C codent en fait le résultat de l'addition de A et B en une numération binaire redondante. Le résultat est alors représenté comme suit :

$$C = A + B = R_C + S_C - (2 - 2^{-m}) \qquad (2.13)$$
$$= \sum_{j=0}^{m} 2^{-j}(r_j^{(c)} + s_j^{(c)} - 1)$$
$$= \sum_{j=0}^{m} 2^{-j} c_j$$
$$r_j^{(c)}, s_j^{(c)} \in \{0, 1\}$$
$$c_j \in \{-1, 0, 1\} .$$

Le circuit de l'additionneur hybride est illustré à la figure 1. On constate que la propagation de la retenue est limitée à une position seulement. Le signal noté **Inc** a pour fonction de générer une incrémentation pendant que l'addition s'effectue. On verra son utilité plus tard.

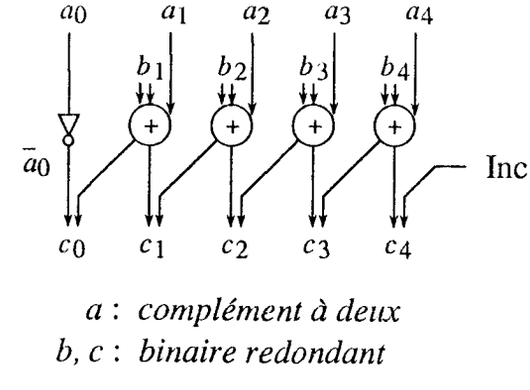

Fig. 1. — Additionneur hybride.

2.2. CONVERTISSEUR

Un autre circuit qui sera utile dans la suite, est un convertisseur qui effectue la conversion d'un nombre binaire redondant en un nombre complément à deux. Soit X, le nombre à convertir :

$$X = \sum_{j=1}^{m} 2^{-j} x_j \qquad (2.14)$$

$$x_j \in \{-1, 0, 1\} .$$

Les chiffres x_j sont codés suivant le tableau 1. Les bits sont ensuite regroupés séparément en deux vecteurs comme suit :

$$(2.15)\quad X = \sum_{j=1}^{m} 2^{-j}(r_j + s_j - 1) = \sum_{j=1}^{m} 2^{-j} r_j + \sum_{j=1}^{m} 2^{-j} s_j + (-1 + 2^{-m})$$

$$r_j, \quad s_j \in \{0, 1\} .$$

Chaque vecteur est transformé en un nombre complément à deux en lui ajoutant une position supplémentaire du côté poids fort, comme ceci :

$$(2.16)\quad X = \left(-r_0 + \sum_{j=1}^{m} 2^{-j} r_j\right) + \left(-s_0 + \sum_{j=1}^{m} 2^{-j} s_j\right) + 2^{-m}$$

$$r_0 = 1, \quad s_0 = 0 .$$

A noter que la constante -1 apparaissant au dernier membre de (2.14) est remplacée ici par le bit négatif $-r_0$. La relation (2.15) indique clairement que la conversion s'effectue en additionnant deux nombres complément à deux. Le circuit est illustré à la figure 2.

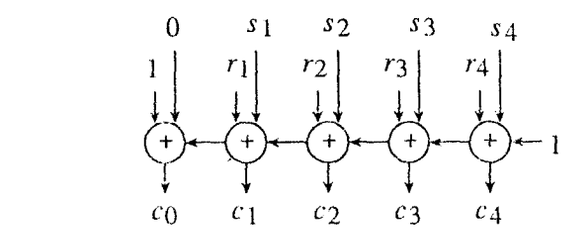

Fig. 2. — Convertisseur de binaire redondant à complément à deux.

3. Algorithme de l'accumulateur

A l'encontre de ce qui se fait traditionnellement, le processus de multiplication est effectué ici en sommant les produits partiels dans un ordre décroissant. Cette procédure permet d'extraire le résultat chiffre par chiffre, poids fort d'abord, à mesure que les produits sont sommés. La bonne marche de cette procédure exige que la numération attribuée au résultat soit redondante, ceci afin de s'affranchir de la propagation de la retenue. Un algorithme a été conçu spécialement à cet effet pour mener à bien cette procédure. Précisons, toutefois, que le concept de sommation des produits partiels est généralisé à celui d'une accumulation de termes décroissants, d'où l'expression *algorithme de l'accumulateur* pour désigner celui-ci.

Définitions

La séquence des termes est définie comme suit :

$$(3.1)\quad \{b^{-i} Y_i : i = 1 \text{ à } \infty\} .$$

La raison b est un nombre entier positif et les termes sont bornés comme suit :

$$(3.2)\quad Y_{min} \leq Y_i \leq Y_{max} .$$

Parce qu'on est dans un contexte d'accumulation, il convient de nommer *ajout* le terme Y_i. L'accumulation totale de la séquence, plus une partie initiale X_0, est représentée par la variable Z :

$$(3.3)\quad Z = X_0 + \sum_{i=1}^{\infty} b^{-i} Y_i .$$

On désire représenter la valeur de l'accumulation totale par la numération suivante :

$$(3.4)\quad Z = \sum_{j=1}^{\infty} b^{-j} d_j$$

$$d_j \in \{\rho_1, ..., \rho_2\} .$$

Il est important de souligner que la base b est égale à la raison en (3.1). Soulignons également qu'aucune supposition n'a été faite pour l'instant sur la symétrie du répertoire, ni sur la redondance ou non de la numération.

Récursion

L'accumulation des ajouts est effectuée itérativement dans un ordre décroissant. L'accumulation partielle, X_k, est alors donnée par :

$$(3.5)\quad X_k = X_0 + \sum_{i=1}^{k} b^{-i} Y_i .$$

L'indice k indique le pas de l'itération. Pendant qu'un ajout est additionné, un chiffre appartenant au résultat est extrait de l'accumulation partielle. Le résultat partiel ainsi formé est représenté par :

$$(3.6)\quad Z_k = \sum_{j=1}^{k} b^{-j} d_j$$

$$d_j \in \{\rho_1, ..., \rho_2\} .$$

Le nombre Z_k est nommé *extraction* et chacun des chiffres le composant est nommé *extrait*. La différence entre l'accumulation partielle et l'extraction est appelée *résidu*. Son expression est la suivante :

$$(3.7)\quad X_k - Z_k = X_0 + \sum_{i=1}^{k} b^{-i} Y_i - \sum_{j=1}^{k} b^{-j} d_j .$$

Comme le résidu est appelé à devenir de plus en plus petit, celui-ci est multiplié par un facteur d'échelle égal à b^k. Cette dernière quantité est représentée par la variable W_k :

$$(3.8)\quad W_k = b^k (X_k - Z_k) .$$

A partir des deux dernières relations, nous déduisons la récursion suivante :

$$W_k = bW_{k-1} + Y_k - d_k \,. \tag{3.9}$$

Cette dernière sera à la base de l'algorithme recherché.

Domaine de convergence

L'algorithme de l'accumulateur procède donc itérativement en additionnant un ajout et en soustrayant un extrait à chaque pas. Mais pour qu'il y ait convergence, il est nécessaire que W_k soit confiné à l'intérieur d'un intervalle fini. C'est ce qui est démontré ici. Supposons que l'itération soit arrêtée au pas k. Il se trouve que le résidu W_k est connu mais le reste des termes à sommer, $\{b^{-i} Y_i : i = k+1 \text{ à } \infty\}$, ne l'est pas. En revanche, nous savons que les Y_i sont bornés par Y_{min} et Y_{max}. De là nous déduisons les inégalités suivantes :

$$W_k + \frac{Y_{min}}{b-1} \leqslant W_k + \sum_{i=k+1}^{\infty} b^{k-i} Y_i \leqslant W_k + \frac{Y_{max}}{b-1} \,. \tag{3.10}$$

Le membre central représente la portion de l'accumulation totale qui n'est pas encore représentée par l'extraction Z_k. Rappelons que l'extraction est composée des chiffres d_1 à d_k, lesquels sont déjà connus. Quant aux chiffres d_{k+1} à d_∞, ils restent à définir en fonction de la valeur qui sera donnée au membre central de (3.10). Cette partie de l'extraction à définir est bornée comme suit :

$$\frac{\rho_1}{b-1} = \sum_{j=k+1}^{\infty} b^{k-j} \rho_1 \leqslant \sum_{j=k+1}^{\infty} b^{k-j} d_j \leqslant \sum_{j=k+1}^{\infty} b^{k-j} \rho_2 = \frac{\rho_2}{b-1} \,.$$

On tient compte ici des bornes ρ_1 et ρ_2 qui limitent le répertoire de chiffres composant l'extraction (3.6). D'autre part, les membres de ces inégalités ont été multipliés par b^k pour convenir au facteur d'échelle introduit dans (3.8). Les limites de ces inégalités sont remplacées comme suit :

$$K_1 \leqslant \sum_{j=k+1}^{\infty} b^{k-j} d_j \leqslant K_2 \tag{3.11}$$

avec :

$$K_1 = \frac{\rho_1}{b-1}, \quad K_2 = \frac{\rho_2}{b-1} \,. \tag{3.12}$$

Le nombre présent dans (3.11) doit absolument représenter la valeur du membre central (3.10). Par conséquent, les limites de (3.10) doivent obligatoirement se situer à l'intérieur des limites de (3.11). Si l'on résout les deux systèmes d'inégalités par rapport à W_k, nous obtenons :

$$K_1 - \frac{Y_{min}}{b-1} \leqslant W_k \leqslant K_2 - \frac{Y_{max}}{b-1} \,. \tag{3.13}$$

Les paramètres b, K_1, K_2, Y_{min} et Y_{max} étant tous fixés d'avance, la variable W_k est par conséquent confinée à l'intérieur d'un intervalle fini bien déterminé, ce qu'il fallait démontrer. Cet intervalle, que l'on désignera dans la suite par l'expression *domaine de convergence,* sera représenté par la variable D :

$$D \equiv \left[K_1 - \frac{Y_{min}}{b-1}, \quad K_2 - \frac{Y_{max}}{b-1} \right] . \tag{3.14}$$

Intervalles de sélection et règle de sélection

Comme on le mentionnait plus tôt, un chiffre d_k est extrait à chaque pas de l'itération. Ainsi au pas k, la sélection de d_k parmi le répertoire de chiffres $\{\rho_1, ..., \rho_2\}$ doit être telle que W_k soit obligatoirement confiné dans le domaine D. Ceci implique que le choix de la valeur donnée à d_k est conditionné par le fait que le résidu ancien W_{k-1} doit appartenir à un intervalle bien précis. Cet intervalle est déduit comme suit. On substitue d'abord la récursion (3.9) dans (3.13) :

$$K_1 - \frac{Y_{min}}{b-1} \leqslant bW_{k-1} + Y_k - d \leqslant K_2 - \frac{Y_{max}}{b-1} \,.$$

Le chiffre d_k a été remplacé ici par le choix particulier d. On sait que l'ajout Y_k prend n'importe quelle valeur entre Y_{min} et Y_{max}. Pour résoudre ces inégalités, nous remplaçons Y_k par la limite appropriée. Prenons, par exemple, l'inégalité de droite. Le terme Y_k est remplacé par Y_{max} et on résout comme suit :

$$bW_{k-1} + Y_{max} - d \leqslant K_2 - \frac{Y_{max}}{b-1}$$

$$bW_{k-1} \leqslant K_2 - \frac{b}{b-1} Y_{max} + d$$

$$W_{k-1} \leqslant \frac{1}{b} \left(K_2 - \frac{b}{b-1} Y_{max} + d \right) \,.$$

De façon similaire, on obtient avec l'inégalité de gauche :

$$W_{k-1} \geqslant \frac{1}{b} \left(K_1 - \frac{b}{b-1} Y_{min} + d \right) \,.$$

Ces deux bornes délimitent l'intervalle recherché. On conviendra de désigner cet intervalle par l'expression *intervalle de sélection,* qui sera représenté par la variable I_d :

$$I_d \equiv \left[\frac{1}{b} \left(K_1 - \frac{b}{b-1} Y_{min} + d \right), \; \frac{1}{b} \left(K_2 - \frac{b}{b-1} Y_{max} + d \right) \right] . \tag{3.15}$$

Ainsi, pour chaque chiffre $d \in \{\rho_1, ..., \rho_2\}$, correspond un intervalle de sélection I_d. Nous pouvons alors énoncer la règle de sélection suivante :

Règle de sélection : *Le choix $d_k = d$ est valide seulement si $W_{k-1} \in I_d$.*

L'importance de cette règle est qu'elle garantit le confinement de W_k dans D.

Condition de chevauchement

Mais pour que la règle soit applicable pour toute valeur de W_{k-1}, il importe que les I_d se chevauchent, ou à tout le moins se touchent. Ce qui nous amène à cette autre condition :

$$\text{Inf}\,(I_{d+1}) \leqslant \text{Sup}\,(I_d)$$

où Inf (.) et Sup (.) sont les bornes inférieures et supérieures, respectivement, des intervalles de sélection. Substituant les limites de (3.15), cette inégalité se transforme comme suit :

$$\frac{1}{b}\left(K_1 - \frac{b}{b-1}Y_{\min} + d + 1\right) \leqslant \frac{1}{b}\left(K_2 - \frac{b}{b-1}Y_{\max} + d\right)$$

$$K_1 - \frac{b}{b-1}Y_{\min} + d + 1 \leqslant K_2 - \frac{b}{b-1}Y_{\max} + d$$

$$\frac{b}{b-1}(Y_{\max} - Y_{\min}) \leqslant K_2 - K_1 - 1$$

$$Y_{\max} - Y_{\min} \leqslant \frac{b-1}{b}(K_2 - K_1 - 1)\,.$$

Si l'on substitue à K_2 et K_1 les expressions (3.12), nous obtenons :

$$(3.16) \qquad Y_{\max} - Y_{\min} \leqslant \frac{\rho_2 - \rho_1 + 1}{b} - 1\,.$$

Cette inégalité sera appelée, dans la suite, *condition de chevauchement* des intervalles de sélection.

C'est ici qu'apparaît la nécessité de la redondance à l'extraction. En effet, l'expression $Y_{\max} - Y_{\min}$ est une quantité strictement positive. Par voie de conséquence, la condition (3.16) nous amène directement à la condition (2.2) qui spécifie la redondance d'une numération, en l'occurrence celle de l'extraction (3.4).

Convergence

Il reste maintenant à vérifier la convergence de l'algorithme. On suppose initialement que $X_0 \in D$ et que la condition (3.16) est vérifiée. Partant de la récursion (3.9), on obtient successivement :

$$\begin{aligned} W_0 &= X_0 \\ W_1 &= bX_0 + Y_1 - d_1 \\ W_2 &= bW_1 + Y_2 - d_2 \\ &= b^2 X_0 + bY_1 + Y_2 - bd_1 - d_2 \end{aligned}$$

et par récurrence :

$$W_k = b^k X_0 + \sum_{i=1}^{k} b^{k-i} Y_i - \sum_{j=1}^{k} b^{k-j} d_j\,.$$

L'itération est effectuée en respectant la règle de sélection, ce qui signifie que $W_k \in D$ est valide en tout temps. En prémultipliant par b^{-k} et en prenant la limite de k, on obtient :

$$\lim_{k\to\infty} X_0 + \sum_{i=1}^{k} b^{-i} Y_i - \sum_{j=1}^{k} b^{-j} d_j = \lim_{k\to\infty} b^{-k} W_k = 0\,.$$

Cette relation est nulle puisque W_k est confiné à l'intérieur de D. Par conséquent, référant à (3.3), on peut écrire :

$$Z = X_0 + \sum_{i=1}^{\infty} b^{-i} Y_i = \sum_{j=1}^{\infty} b^{-j} d_j\,.$$

Le dernier membre est bien la représentation en numération redondante de l'accumulation totale (3.4), ce qui complète la preuve.

L'algorithme de l'accumulateur s'énonce comme suit :

Algorithme de l'accumulateur
Conditions initiales

$$X_0 \in D$$

$$Y_{\max} - Y_{\min} \leqslant \frac{\rho_2 - \rho_1 + 1}{b} - 1$$

$$k = 1$$

Tant que $k \leqslant \infty$, **faire**
Sélection : $d_k \leftarrow$ Sélecte (W_{k-1})
Récursion : $W_k \leftarrow bW_{k-1} + Y_k - d_k$
Itération : $k \leftarrow k + 1$
Fin tant que

La fonction Sélecte (.) symbolise la règle de sélection sur d_k.

4. Réalisation de l'accumulateur

La réalisation de l'algorithme de l'accumulateur consiste essentiellement à mettre en œuvre la récursion (3.9) et la règle de sélection énoncée précédemment. Mais avant de procéder, trois points importants doivent être discutés. Premièrement, on fait remarquer que seule la numération appartenant à l'extraction a été spécifiée jusqu'à maintenant ; rien n'a été dit au sujet des numérations de l'ajout et du résidu. On comprendra que la forme définitive que prendra la réalisation sera liée, pour une large part, au choix qui aura été fait sur les numérations de ces dernières variables.

Deuxièmement, on doit souligner qu'une équivoque existe à l'endroit du chevauchement des intervalles de sélection. En effet, lorsque le résidu ancien W_{k-1} est localisé à cet endroit, il appartient autant à l'intervalle I_d qu'à l'intervalle I_{d+1}, par exemple. Dans ces conditions, un choix unique parmi les valeurs d ou $d+1$ à donner à l'extrait devra être fait par la réalisation.

Troisièmement, la localisation du résidu ancien parmi les intervalles de sélection suppose l'exécution d'une comparaison. Pour éviter un matériel trop important et un temps d'exécution trop long, la comparaison se fera à partir d'une valeur approximative du résidu ancien, en prenant sa troncature. L'incertitude alors créée sera compensée par le chevauchement des intervalles de sélection.

La procédure de réalisation se divise en trois phases. La première est la caractérisation du problème. Elle consiste

principalement à définir les numérations qui seront attribuées aux variables. La seconde phase est la « mise en forme » de l'algorithme. Elle consiste à déterminer le domaine de convergence et les intervalles de sélection, puis à effectuer quelques manipulations sur certaines variables de manière à simplifier la mise en œuvre de la récursion et de la règle de sélection. La dernière phase correspond à la réalisation proprement dite.

4.1. CARACTÉRISATION DU MULTIPLIEUR : 1re PHASE

La souplesse qui caractérise notre méthode apparaît surtout lors de la première phase. En effet il faut d'abord s'intéresser aux caractéristiques du projet à réaliser avant de passer à la réalisation proprement dite du multiplieur. De cette manière il est assuré que la fonction du multiplieur répondra au mieux aux exigences du projet.

Le projet que nous proposons est la réalisation d'un filtre tout pôle du second ordre. Le second ordre permet la mise en œuvre de pôles réels ou de pôles complexes. Ces derniers sont conjugués de manière à ce que les coefficients de l'équation aux différences soient réels. Cette dernière s'écrit :

$$y(n) = b_1 \cdot y(n-1) + b_2 \cdot y(n-2) + x(n) . \quad (4.1)$$

La mise en œuvre de cette équation se fait évidemment à l'aide de deux multiplieurs. La première caractéristique qui est donnée à la réalisation est la suivante : au lieu de faire deux multiplieurs isolés, ceux-ci sont imbriqués l'un dans l'autre pour ne former qu'un seul tableau multiplieur. Les deux premiers termes de l'équation (4.1) génèrent deux séquences de produits partiels. On fait en sorte qu'un produit partiel d'une séquence soit groupé avec le produit partiel correspondant de l'autre séquence pour former une paire. L'accumulateur qui servira à construire le multiplieur aura comme ajout la somme de cette paire. Cette façon de faire est choisie de manière à ce que la réalisation ne comporte qu'un seul bloc. Ainsi on a de meilleures chances que le circuit soit compact, et on évite par ailleurs de longues connexions entre deux blocs.

Une deuxième caractéristique donnée au tableau multiplieur est que les opérandes multiplicateurs soient en base 4 au lieu d'en base 2. De cette façon deux fois moins de produits partiels sont générés, ce qui autorise un matériel deux fois moins important.

On mentionnait précédemment, dans l'introduction, qu'un tableau multiplieur possède un délai, lequel se retrouve à l'intérieur de la boucle de récursion du filtre récursif. Sans faire de démonstration, on affirme que, pour une base donnée, le délai est d'autant plus court que la redondance attribuée à l'extraction est forte. Dans ces conditions, il vaut mieux maximiser la redondance de manière à minimiser le délai à l'intérieur de la boucle de récursion. Sachant que la boucle prend effet lorsque la sortie du tableau multiplieur est reliée aux entrées des multiplicateurs, l'extraction et les deux multiplicateurs partagent, de ce fait, la même numération. Celle-ci est de la forme suivante :

$$Z_k = \sum_{j=1}^{k} 4^{-j} d_j \quad (4.2)$$

$$d_j \in \{-3, ..., 3\} .$$

A noter que la relation (2.4) montre que le répertoire choisi ici autorise bien le maximum de redondance. De cette numération, trois paramètres importants sont à retenir pour la suite :

$$b = 4 , \quad \rho = 3 , \quad K = 1 \quad (4.3)$$

qui sont la base, l'amplitude du répertoire et le facteur de redondance, respectivement.

Une condition nécessaire à la stabilité est que les pôles soient confinés à l'intérieur du cercle unitaire. Cette condition a pour effet de limiter les coefficients comme suit :

$$|b_1| < 2 , \quad |b_2| < 1 . \quad (4.4)$$

Par ailleurs, le complément à deux est parmi les numérations les plus couramment utilisées en électronique numérique. Pour cette raison, elle sera choisie pour représenter les coefficients. En tenant compte des limites, les coefficients sont représentés comme suit :

$$b_1 = -2 b_{-1}^{(1)} + \sum_{j=0}^{m} 2^{-j} b_j^{(1)} \quad (4.5a)$$

$$b_2 = -b_0^{(2)} + \sum_{j=1}^{m} 2^{-j} b_j^{(2)} \quad (4.5b)$$

$$b_{-1}^{(1)}, b_j^{(1)}, b_0^{(2)}, b_j^{(2)} \in \{0, 1\} .$$

Le résidu est continuellement sollicité avec l'addition qui a lieu dans la récursion (3.9). Il est alors souhaitable de lui donner une redondance afin que le processus d'addition soit affranchi de la propagation de la retenue. L'additionneur hybride de la section 2 est utilisé à cet effet. Le résidu sera donc en binaire redondant, comme suit :

$$W_{k-1} = \sum_{j=1}^{m+\sigma} 2^{-j} w_j \quad (4.6)$$

$$w_j \in \{-1, 0, 1\} .$$

Ici se termine la caractérisation du problème. Il peut sembler conceptuellement exagéré de faire appel à autant de numérations différentes à l'intérieur d'un même accumulateur. On remarque, cependant, que chacune d'elles a été choisie en fonction de critères bien précis. Ainsi le résidu est en binaire redondant parce que l'on utilise l'additionneur hybride, de manière à s'affranchir de la propagation de la retenue. Précisons ici qu'un additionneur plus conventionnel, ayant les deux opérandes en binaire redondant, pourrait être tout aussi satisfaisant. Toutefois le matériel à implanter aurait été deux fois plus important [8]. Par ailleurs, le complément à deux pour représenter les coefficients est un choix plus judicieux que le binaire redondant, car ces variables sont externes. Par conséquent, le choix respectif des numérations adoptées pour représenter les coefficients et le résidu s'avère des plus pertinents.

D'autre part, on a expliqué plus haut qu'il était plus avantageux de choisir une base 4 au lieu d'une base 2 pour représenter l'extraction, de manière à simplifier le matériel de moitié. De plus, une redondance maximale est adoptée pour minimiser le délai du multiplieur. Ceci se fait, doit-on signaler, aux dépends d'une complexité plus grande à l'endroit du dispositif qui génère les produits partiels.

4.2. MISE EN FORME DE L'ALGORITHME : 2e PHASE

La mise en forme de l'algorithme consiste, dans un premier temps, à déterminer l'étendue du domaine de convergence et celle des intervalles de sélection. Pour ce faire, quatre paramètres doivent être fixés : b, ρ, K et Y_{opt}. Les trois premiers sont déjà fixés par (4.3). Le quatrième, Y_{opt}, représente l'amplitude absolue des ajouts lorsque $Y_{max} = -Y_{min}$. Ce dernier est déterminé à partir de la condition de convergence.

Dans un deuxième temps, une approximation du résidu ancien W_{k-1} est prise en faisant une troncature. Ceci permet de simplifier grandement la localisation du résidu parmi les intervalles de sélection. Ensuite une simple règle d'arrondissement appliquée à l'approximation permet d'obtenir un extrait unique. Ainsi l'équivoque est levée.

Dans un troisième temps, quelques manipulations sur la représentation du résidu permettent de simplifier la mise en œuvre de la récursion.

Étendue du domaine de convergence et des intervalles de sélection

Un produit partiel est généré par la multiplication d'un coefficient avec un chiffre multiplicateur. Sachant d'une part que $|b_1| < 2$ et $|b_2| < 1$, et d'autre part que les chiffres multiplicateurs appartiennent à $\{-3, ..., 3\}$, une paire de produits partiels est limitée comme suit :

$$|b_1 \cdot q_k^{(1)}| + |b_2 \cdot q_k^{(2)}| < 9 .$$

Les variables $q_k^{(1)}$ et $q_k^{(2)}$ sont les chiffres multiplicateurs appartenant à $y(n-1)$ et $y(n-2)$, respectivement. Cette paire, on le rappelle, tient lieu d'ajout. Mais cette limite doit être reproportionnée de manière à ce que les ajouts respectent la condition de chevauchement. Pour cette raison le facteur $2^{-\sigma}$ est ajouté aux produits partiels pour donner un degré de liberté. On a dans ce cas :

$$|2^{-\sigma} b_1 \cdot q_k^{(1)}| + |2^{-\sigma} b_2 \cdot q_k^{(2)}| < 9 \cdot 2^{-\sigma}$$

ou plus simplement :

$$(4.7) \qquad |Y_{1,k}| + |Y_{2,k}| < Y_{opt} = 9 \cdot 2^{-\sigma} .$$

Les variables $Y_{1,k}$ et $Y_{2,k}$ représentent respectivement les produits partiels reproportionnés. Ceux-ci sont limités en valeur absolue par Y_{opt}. Substituant les paramètres de (4.3), la condition de chevauchement (3.16) devient :

$$Y_{max} - Y_{min} \leq \frac{\rho_2 - \rho_1 + 1}{b} - 1$$

$$2\, Y_{opt} \leq \frac{2\rho + 1}{b} - 1$$

$$18 \cdot 2^{-\sigma} \leq \frac{3}{4}$$

$$2^{-\sigma} \leq \frac{1}{24}$$

$$\sigma \geq 4{,}5850 .$$

L'indice σ est un nombre entier. Par ailleurs, comme l'extraction est en base 4, le facteur $2^{-\sigma}$ doit de plus être un multiple de 4^{-1}. Par conséquent, le plus petit indice permis est $\sigma = 6$, ce qui amène :

$$(4.8) \qquad Y_{opt} = 9 \cdot 2^{-\sigma} = \frac{9}{64} .$$

Substituant les valeurs de (4.3) et (4.8) dans (3.14) et (3.15), on obtient :

$$(4.9) \qquad D = [-0{,}953125, 0{,}953125]$$

$$(4.10) \qquad I_d = \left[\frac{d}{4} - 0{,}203125 , \quad \frac{d}{4} + 0{,}203125 \right] .$$

Il est possible maintenant d'obtenir les représentations exactes des termes $Y_{1,k}$ et $Y_{2,k}$. On débute avec les représentations (4.5). Si on multiplie ces dernières par le chiffre multiplicateur 3, deux positions supplémentaires sont créées à la droite des représentations. Ensuite le proportionnement par le facteur de proportion $2^{-\sigma}$ provoque un décalage de 6 positions vers la gauche du point de numération. On obtient alors :

$$(4.11a) \qquad Y_{1,k} = -2^{-3} y_3^{(1)} + \sum_{j=4}^{m+6} 2^{-j} y_j^{(1)}$$

$$(4.11b) \qquad Y_{2,k} = -2^{-4} y_4^{(1)} + \sum_{j=5}^{m+6} 2^{-j} y_j^{(2)}$$

$$y_3^{(1)}, y_j^{(1)}, y_4^{(2)}, y_j^{(2)} \in \{0, 1\} .$$

Approximation du résidu et règles d'arrondissement

La localisation du résidu ancien parmi les intervalles de sélection se fait à partir d'une approximation de ce dernier, en faisant une troncature. L'incertitude alors générée est compensée par le chevauchement des intervalles de sélection. Prenons, par exemple, une valeur approximative, P, placée à une extrémité de I_d, près de I_{d+1}. La troncature génère un intervalle d'incertitude autour de l'approximation (*fig.* 3). Il peut arriver que cet intervalle déborde de I_d dans I_{d+1}. Dans ce cas, la valeur d donnée à l'extrait peut mener parfois à une erreur. En revanche, si l'incertitude est suffisamment faible, il sera

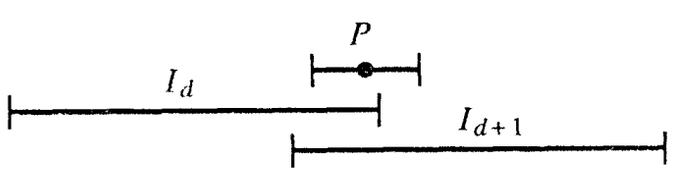

Fig. 3. — Chevauchement des intervalles de sélection.

strictement inclus dans I_{d+1}, en raison du chevauchement des intervalles de sélection. Dans ce cas, la valeur $d+1$ donnée à l'extrait sera en tout temps valide.

Bref, il importe que la troncature comporte assez de chiffres pour que l'intervalle d'incertitude soit suffisamment petit. En contrepartie, une troncature comportant plus de chiffres que nécessaire fait augmenter inutilement le matériel. Dans le cas qui nous intéresse, l'optimum est atteint lorsque 5 chiffres de W_{k-1} sont tronqués. La représentation de l'approximation est la suivante :

$$Q_{k-1} = \sum_{j=1}^{5} 2^{-j} w_j \qquad (4.13)$$

$$w_j \in \{-1, 0, 1\} .$$

L'équivoque qui existe à l'endroit de Q_{k-1} quant au choix de la valeur à donner à l'extrait d_k est facilement levée avec l'application de la règle d'arrondissement suivante :

Règle N° 1 : *La fraction de $4\,Q_{k-1}$ est complétée si celle-ci est* ***plus grande ou égale*** *à une demie. Sinon, elle est soustraite.*

L'arrondissement donne alors automatiquement la valeur de l'extrait à choisir. A noter que cette règle est valable même lorsqu'il n'y a pas d'équivoque. Elle peut donc remplacer la règle de sélection énoncée à la section 3. On profite de l'occasion pour mentionner une seconde règle d'arrondissement, assez semblable à la première et qui sera utilisée plus loin. Elle s'énonce comme suit :

Règle N° 2 : *La fraction de $4\,Q_{k-1}$ est complétée si celle-ci est* ***strictement plus grande*** *que une demie. Sinon, elle est soustraite.*

Pour le moment, seule la première règle sera appliquée.

Un exemple est donné ici sur l'application de la règle d'arrondissement. On suppose l'approximation $Q_{k-1} = 0{,}125$. Référant à (4.13), l'incertitude créée est de 2^{-5} de part et d'autre de cette valeur. L'intervalle d'incertitude est donc : (0,09375, 0,15625). A noter que l'intervalle est ouvert en raison de la taille finie du résidu. Référant à (4.10), on constate alors que l'intervalle d'incertitude est compris autant dans I_0 que I_1 ; d'où l'on a :

$$I_0 = [-0{,}203125, 0{,}203125]$$
$$I_1 = [0{,}046875, 0{,}453125] .$$

Dans ces conditions, les choix $d_k = 0$ et $d_k = 1$ sont valides autant l'un que l'autre. Pour lever l'équivoque, on applique la première règle d'arrondissement. On multiplie par quatre l'approximation : $4\,Q_{k-1} = 0{,}5$; puis l'arrondissement est effectué, ce qui donne $d_k = 1$. De cette manière, le résidu est exclusivement localisé dans I_1, ce qui élimine l'équivoque.

Le tableau 2 regroupe à la colonne 2 les valeurs discrètes de $4\,Q_{k-1}$ pour lesquelles la règle d'arrondissement donne un même choix d. La colonne 3 montre les intervalles ouverts de $4\,W_{k-1}$ qui donnent lieu aux regroupements des valeurs de $4\,Q_{k-1}$ lorsque la troncature à 5 chiffres est effectuée. La colonne 4 montre les intervalles ouverts de $4\,W_k$, une fois la récursion (3.9) effectuée. Les résultats obtenus à la colonne 4 montrent qu'il y a effectivement convergence avec cette forme particulière de l'algorithme. En effet, lorsqu'on divise par 4 les bornes de la réunion des intervalles de la colonne 4, on obtient $W_k \in (-0{,}765625, 0{,}640625)$, ce qui est strictement inclus dans (4.10).

d	$4Q_{k-1}$	$4W_{k-1}$	$4W_k$
+3	{ 2.5, ... , 2.625 }	(2.375, 2.5625)	(-3.0625, -1.1875)
+2	{ 1.5, ... , 2.375 }	(1.375, 2.5)	(-3.0625, 2.5625)
+1	{ 0.5, ... , 1.375 }	(0.375, 1.5)	(-3.0625, 2.5625)
0	{ -0.5, ... , 0.375 }	(-0.625, 0.5)	(-3.0625, 2.5625)
-1	{ -1.5, ... , -0.625 }	(-1.625, -0.5)	(-3.0625, 2.5625)
-2	{ -2.5, ... , -1.625 }	(-2.625, -1.5)	(-3.0625, 2.5625)
-3	{ -3.125, ... , -2.625 }	(-3.0625 , -2.5)	(-0.8125, 2.5625)

Tableau 2. Regroupement des valeurs d'approximation avec mise en œuvre de la règle N° 1

On va donner maintenant quelques explications illustratrices sur la manière de construire le tableau 2. A titre d'exemple, on considère le regroupement correspondant à $d = 1$. La borne gauche donne $4\,Q_{k-1} = 0{,}5$, soit :

$$4\,Q_{k-1} = (0\ 0\ .\ 1\ 0\ 0)_2 = 0{,}5 .$$

La représentation minimale du résidu ancien qui donne lieu à cette approximation est :

$$4\,W_{k-1} = (0\ 0\ .\ 1\ 0\ 0\ \ \bar{1}\ \bar{1}\ \bar{1}\ \ldots\ \bar{1})_2 > 0{,}375 .$$

On rappelle que le résidu est en binaire redondant. L'inégalité est stricte parce que l'on assume que la taille du résidu est finie. L'intervalle occupé par $4\,W_{k-1}$ pour ce regroupement est donc ouvert à gauche. La borne inférieure du résidu nouveau est obtenue en appliquant la récursion (3.9) :

$$4\,W_k = 4(4\,W_{k-1} - Y_{opt} - d) .$$

Le minimum de l'ajout, $-Y_{opt}$, est assumé pour obtenir le minimum de $4\,W_k$. Selon (4.8), cette valeur est $-9/64$, ce qui donne, une fois les valeurs réelles substituées :

$$4\,W_k > 4\left(0{,}375 - \frac{9}{64} - 1\right) = -3{,}0625 .$$

Comme l'inégalité est stricte, l'intervalle occupé par $4\,W_k$ est ouvert à gauche. Une procédure semblable est appliquée pour trouver les bornes supérieures apparaissant au tableau.

Deux exceptions ont lieu aux première et dernière rangées du tableau 2. En effet, comme les intervalles de $4\,W_k$ n'atteignent jamais 2,5625 à droite ($-3{,}0625$ à gauche), il en sera de même avec l'intervalle de $4\,W_{k-1}$ lorsque $d = 3$ ($d = -3$). La valeur maximale de l'approximation de $4\,W_{k-1}$ s'obtient comme suit. On écrit d'abord la

représentation de $4\,W_{k-1}$ en faisant en sorte que la partie comprenant les cinq premières positions du côté poids fort soit la plus grande possible :

$$4\,W_{k-1} = (1\ 0\,.\,1\ 0\ 0\ \ 1\ 0\ 0\ldots 0\ \ \bar{1})_2 < 2{,}5625$$

$$4\,W_{k-1} = (1\ 0\,.\,1\ 0\ 1\ \ \bar{1}\ 0\ 0\ldots 0\ \ \bar{1})_2 < 2{,}5625\,.$$

Lorsque la dernière représentation est tronquée à trois positions après le point, on obtient :

$$4\,Q_{k-1} = (1\ 0\,.\,1\ 0\ 1)_2 = 2{,}625$$

ce qu'il fallait démontrer. Une procédure semblable est appliquée pour obtenir $4\,Q_{k-1} = -3{,}125$ à la dernière rangée.

Conversion de l'approximation et mise en œuvre de la sélection

L'approximation Q_{k-1} représentée par (4.13) comporte 10 bits, soit deux bits par position. On aura ainsi 2^{10} chiffres à comparer. Il est possible de réduire cette complexité en convertissant ce nombre en complément à deux. Dans ce cas, la nouvelle représentation de Q_{k-1} nécessite seulement 6 bits :

$$(4.14) \qquad Q_{k-1} = -q_0 + \sum_{j=1}^{5} 2^{-j}\,q_j$$

$$q_0, q_j \in \{0, 1\}\,.$$

Cette manière de procéder donne l'occasion de simplifier la mise en œuvre de la sélection. Pour s'en convaincre, on réécrit explicitement l'expression (4.14) mais en multipliant les membres par un facteur 4 :

(4.15)

$$4\,Q_{k-1} = -4\,q_0 + 2\,q_1 + q_2 + 2^{-1}\,q_3 + 2^{-2}\,q_4 + 2^{-3}\,q_5\,.$$

On constate alors que l'application de la première règle d'arrondissement implique que la fraction de $4\,Q_{k-1}$ soit complétée seulement lorsque $q_3 = 1$. Par conséquent, l'inspection de ce seul bit suffira à donner la commande d'incrémenter la partie entière de $4\,Q_{k-1}$. La valeur à donner à l'extrait s'écrit donc :

$$(4.16) \qquad d_k = -4\,q_0 + 2\,q_1 + q_2 + q_3\,.$$

Manipulations sur le résidu et mise en œuvre de la récursion

Il s'agit maintenant de faire la mise en œuvre de la récursion. La soustraction de l'extrait est très simple à effectuer puisque, comme nous allons le montrer, elle consiste à tronquer le résidu. Ceci est permis dans la mesure où l'on accepte la numération hybride suivante :

$$(4.17) \quad 4\,W_{k-1} = -4\,q_0 + 2\,q_1 + q_2 + 2^{-1}\,q_3 + 2^{-2}\,q_4 + 2^{-3}\,q_5 + \sum_{j=6}^{m+6} 2^{-j+2}\,w_j$$

$$q_0, \ldots, q_5 \in \{0, 1\}$$

$$w_j \in \{-1, 0, 1\}\,.$$

La partie comprenant les bits q est en complément à deux et le reste est en binaire redondant. Comme on peut le constater, la première partie est identique à la représentation de $4\,Q_{k-1}$ dans (4.15). Si l'on soustrait de (4.17) l'expression (4.16), on obtient :

$$(4.18) \quad 4\,W_{k-1} - d_k = -2^{-1}\,q_3 + 2^{-2}\,q_4 + 2^{-3}\,q_5 + \sum_{j=6}^{m+6} 2^{-j+2}\,w_j$$

ce qui revient effectivement à tronquer les bits q_0, q_1 et q_2 de $4\,W_{k-1}$.

Dans un deuxième temps, l'ajout est additionné à l'expression ci-dessus pour compléter la récursion. Pour ce faire, on doit au préalable convertir l'expression (4.18) entièrement en binaire redondant. On commence d'abord par substituer à $-q_3$ l'expression $\bar{q}_3 - 1$, où $\bar{q}_3$ est l'inverse booléen de q_3 :

$$4\,W_{k-1} - d_k = 2^{-1}(\bar{q}_3 - 1) + 2^{-2}\,q_4 + 2^{-3}\,q_5 + \sum_{j=6}^{m+6} 2^{-j+2}\,w_j\,.$$

Ensuite, on associe au bit $\bar{q}_3(q_4)(q_5)$ un second bit de sorte que le couple ainsi formé code un chiffre $w_3(w_4)(w_5)$ appartenant au répertoire $\{-1, 0, 1\}$, comme ceci :

$$(4.19) \qquad \begin{aligned} &(\bar{q}_3, 0) \rightarrow w_3 : w_3 \in \{-1, 0\} \\ &(q_4, 1) \rightarrow w_4 : w_4 \in \{0, 1\} \\ &(q_5, 1) \rightarrow w_5 : w_5 \in \{0, 1\}\,. \end{aligned}$$

Il convient de noter que le fait d'associer le bit 0 à $\bar{q}_3$, au lieu du bit 1, annule le biais qui était créé par le remplacement de q_3 par $\bar{q}_3$. Substituant les chiffres w_3, w_4 et w_5, dans l'expression ci-dessus, on obtient :

$$4\,W_{k-1} - d_k = \sum_{j=3}^{m+6} 2^{-j+2}\,w_j$$

$$w_j \in \{-1, 0, 1\}\,.$$

L'ajout, constitué de la paire $Y_{1,k}$ et $Y_{2,k}$, additionnée, il vient :

$$(4.20) \quad 4\,W_{k+1} + Y_{1,k} + Y_{2,k} - d_k = \sum_{j=1}^{m+6} 2^{-j}\,w_{j+2}(k-1) - 2^{-3}\,y_3^{(1)}(k) + \sum_{j=4}^{m+6} 2^{-j}\,y_j^{(1)}(k) - 2^{-4}\,y_4^{(1)}(k) + \sum_{j=5}^{m+6} 2^{-j}\,y_j^{(2)}(k)\,.$$

On a ici substitué les expressions (4.11) à l'ajout. Un changement d'indice a eu lieu sur la première sommation pour que les pondérations des trois représentations soient écrites de la même façon. De plus, les chiffres $w_{m+5} = 0$ et $w_{m+6} = 0$ ont été ajoutés pour ajuster la taille du résidu ancien à celles des autres nombres. Le membre de droite

est ainsi composé de trois nombres, l'un représenté en binaire redondant et les deux autres en complément à deux. Ceci autorise donc l'emploi de deux additionneurs hybrides mis en cascade pour effectuer l'addition. Le résultat est alors représenté en binaire redondant, donnant la représentation du résidu nouveau :

$$W_k = \sum_{j=1}^{m+6} 2^{-j} w_j(k) \qquad (4.21)$$

$$w_j \in \{-1, 0, 1\} .$$

A noter que cette numération est semblable à celle du résidu ancien (4.6), hormis l'indice temporel k. L'itération est ainsi bouclée.

4.3. RÉALISATION DE L'ACCUMULATEUR : 3e PHASE

La réalisation s'effectue en mettant en œuvre, d'une part, le processus de sélection et, d'autre part, la récursion. A cela s'ajoute quelques manipulations telles la conversion en complément à deux de l'approximation, ainsi que la conversion en binaire redondant du résidu ancien diminué de l'extrait. La réalisation se fera ici à l'aide de deux dispositifs développés à la section 2, qui sont l'additionneur hybride et le convertisseur.

Biais apportés au résidu

On débute par la mise en œuvre de la récursion, explicitée par (4.20). On a besoin de deux additionneurs hybrides, un pour chaque terme de l'ajout. Quelques modifications doivent cependant être effectuées sur les trois opérandes avant de procéder. Le tableau 3 a montre ces trois opérandes avant modification. Les positions sont juxtaposées de manière à faire correspondre verticalement leurs poids respectifs. Les trois premières positions du résidu diminué sont décomposées en couples de bits comme spécifié par (4.19). Au sujet des termes de l'ajout, le premier bit y_4 de $Y_{2,k}$ a subi l'extension d'une position, et les signes négatifs accompagnant les premiers bits indiquent pour ceux-ci un poids négatif.

Les modifications dont il est question concernent l'allongement des tailles de $Y_{1,k}$ et $Y_{2,k}$. En effet l'additionneur hybride a été conçu de manière à ce que la taille de l'opérande complément à deux soit une position supérieure à celle de l'opérande binaire redondant. Pour ce faire, les bits inversés $\bar{y}_3$ et $\bar{y}_4$ sont substitués aux bits négatifs $-y_3$ et $-y_4$, respectivement. Ceci a pour conséquence que la position de poids fort de chacun des termes est maintenant de poids positif. De cette manière, il est aisé d'allonger les tailles en ajoutant des bits zéros à gauche (tableau 3 b). On signale que la première addition hybride s'effectue avec $Y_{2,k}$.

Considérant que $-y_3 = \bar{y}_3 - 1$ et $-y_4 = \bar{y}_4 - 1$, il est clair qu'un biais de 2^{-3} est créé sur chacun des termes. Cependant, ce biais peut facilement être corrigé lors de l'addition hybride en apportant le même biais sur le résidu, mais avec un signe opposé. A cet effet, la chaîne des bits associés du résidu forme un nombre binaire (.0 1 1). Le biais est alors corrigé si la valeur 2^{-2} est

(a)	$Y_{2,k}$	:				$-y_4$	y_4	y_5	y_6	y_7	⋯
	$Y_{1,k}$	:				$-y_3$	y_4	y_5	y_6	y_7	⋯
	$4W_{k-1} - d_k$	:		$.\bar{q}_3$	q_4	q_5	w_6	w_7	w_8	w_9	⋯
	bits associés	:		0	1	1					
(b)	$Y_{2,k}$	:	-0	$.0$	0	$\bar{y}_4$	y_4	y_5	y_6	y_7	⋯
	$Y_{1,k}$	:	$-0\ 0$	$.0$	0	$\bar{y}_3$	y_4	y_5	y_6	y_7	⋯
	$4W_{k-1} - d_k$	:		$.\bar{q}_3$	q_4	q_5	w_6	w_7	w_8	w_9	⋯
	bits associés	:		0	0	1					
(c)	$Y_{2,k}$	:	-0	$.0$	0	$\bar{y}_4$	y_4	y_5	y_6	y_7	⋯
	$Y_{1,k}$	:	$-0\ 0$	$.0$	0	$\bar{y}_3$	y_4	y_5	y_6	y_7	⋯
	$4W_{k-1} - d_k$	:		$.\bar{q}_3$	q_4	q_5	w_6	w_7	w_8	w_9	⋯
	bits associés	:		0	1	0					
	W_k	:	w_{-1}^*	w_0^*	$.w_1^*$	w_2^*	w_3^*	w_4	w_5	w_6	w_7	⋯
	$4W_k$	:	$-d_3$	d_2	d_1	d_0	$.\bar{q}_3$	q_4	q_5	w_6	w_7	⋯
						extrait	0	1	0		

Tableau 3. Modifications apportées aux opérandes, suivies d'une double addition hybride et d'une sélection de l'extrait

a) Avant modification
b) Ajustement des tailles
c) Addition et sélection

soustraite de ce nombre, ce qui donne (.0 0 1) au tableau 3 b.

Une autre modification reste à effectuer sur le résidu. En effet la relation (4.16) indique que l'extrait est généré en additionnant le bit q_3 à la partie entière de l'approximation. Le même résultat peut être obtenu en additionnant la valeur 1/2 à l'expression (4.17), forçant l'incrémentation de la partie entière lorsque $q_3 = 1$. Cette opération sera devancée en créant dès maintenant au biais de 2^{-3} sur le résidu, en prenant soin de modifier en conséquence la chaîne des bits associés (tableau 3 c).

Le résultat de la double addition hybride, donnant le résidu nouveau W_k, est indiqué à la première rangée sous le trait horizontal (tableau 3 c). On fait remarquer que les chiffres w marqués d'un astérisque sont ceux affectés par le dernier biais apporté au résidu. Une conversion en complément à deux est ensuite effectuée sur les sept premiers chiffres du résultat, jusqu'au chiffre w_5 comme l'indique (4.13). La représentation hybride ainsi obtenue est montrée à la seconde rangée sous le trait. A noter que le point de numération a été déplacé de deux positions vers la droite, ce qui a pour effet de multiplier W_k par un facteur 4. L'extrait d_{k+1} est alors obtenu en tronquant la partie entière de $4\,W_k$, grâce au biais apporté précédemment sur le résidu diminué. La représentation de l'extrait est en complément à deux avec une taille de 4 bits. En fait cette taille pourrait être réduite à 3 bits, compte tenu que le répertoire des extraits est $\{-3, ..., 3\}$. A ce propos, on signale que le quatrième bit sera supprimé plus tard grâce à quelques simplifications amenées sur la réalisation.

Il est clair que le dernier biais apporté au résidu a eu pour effet de faciliter grandement la sélection. D'un autre côté, cependant, il force l'inversion du bit q_3. Ce résultat est heureux puisque c'est justement $\bar{q}_3$ dont il est question lors des manipulations dans (4.19). Rappelons que celles-ci ont pour but de reconvertir en binaire redondant la partie complément à deux de ce qui reste du résidu, une fois l'extrait soustrait. C'est le cas notamment de $4\,W_k$, au tableau 3 c, où les bits q doivent être reconvertis en binaire redondant. Mais au lieu d'utiliser la chaîne de bits associés (. 0 1 1) comme indiqué par (4.19), on anticipe les opérations en associant plutôt la chaîne (. 0 1 0). Cette dernière se retrouve au-dessus du trait horizontal.

A la lumière du tableau 3 c, une première version de la réalisation est illustrée (*fig.* 4 *a*). L'ensemble des cellules d'addition marquées d'un **A**, incluant les inverseurs, représente les deux additionneurs hybrides. La troisième rangée de cellules, marquées d'un **S**, représente un convertisseur, celui développé à la section 2. Il conviendra d'appeler dans la suite ce dispositif *sélecteur*. Les bulles accolées à certaines cellules symbolisent des inversions.

Simplifications

Une première simplification survient lorsque l'addition des bits $\bar{q}_3$ et $\bar{q}_4$ est reportée au deuxième additionneur (*fig.* 4 *b*). En effet, plusieurs des cellules à gauche du circuit ont maintenant des entrées constantes. Ceci autorise la simplification de six cellules, ce qui supprime du même coup le quatrième bit de l'extrait (*fig.* 4 *c*).

Une deuxième simplification se réalise avec l'application de la seconde règle d'arrondissement. Mentionnons d'abord que l'intérêt de la première règle d'arrondissement est qu'elle facilite la mise en œuvre de la sélection par l'inspection du seul bit q_3 pour déterminer l'incrémentation de la partie entière de $4\,Q_{k-1}$. La seconde règle, pourrait-on croire, demanderait l'inspection des trois bits q_3, q_4 et q_5. Il n'en est rien, en réalité, car il suffit d'apporter un biais de -2^{-3} à $4\,Q_{k-1}$ pour ensuite n'avoir à inspecter que q_3. Pour s'en convaincre, un exemple est donné avec quatre représentations de $4\,Q_{k-1}$ prises au voisinage de la demie. Les commandes d'incrémentation sont indiquées à droite, lorsqu'il y a lieu, suivant l'énoncé de la seconde règle

$q_0\ q_1\ q_2 . 0\ 1\ 1$
$q_0\ q_1\ q_2 . 1\ 0\ 0$
$q_0\ q_1\ q_2 . 1\ 0\ 1$ incrémentation
$q_0\ q_1\ q_2 . 1\ 1\ 0$ incrémentation .

Dans cette situation, les trois bits $q_3 - q_5$ doivent être inspectés pour donner la commande juste d'incrémentation. Maintenant, si un biais de -2^3 est ajouté, on obtient :

$q_0\ q_1\ q_2 . 0\ 1\ 0$
$q_0\ q_1\ q_2 . 0\ 1\ 1$
$q_0\ q_1\ q_2 . 1\ 0\ 0$ incrémentation
$q_0\ q_1\ q_2 . 1\ 0\ 1$ incrémentation .

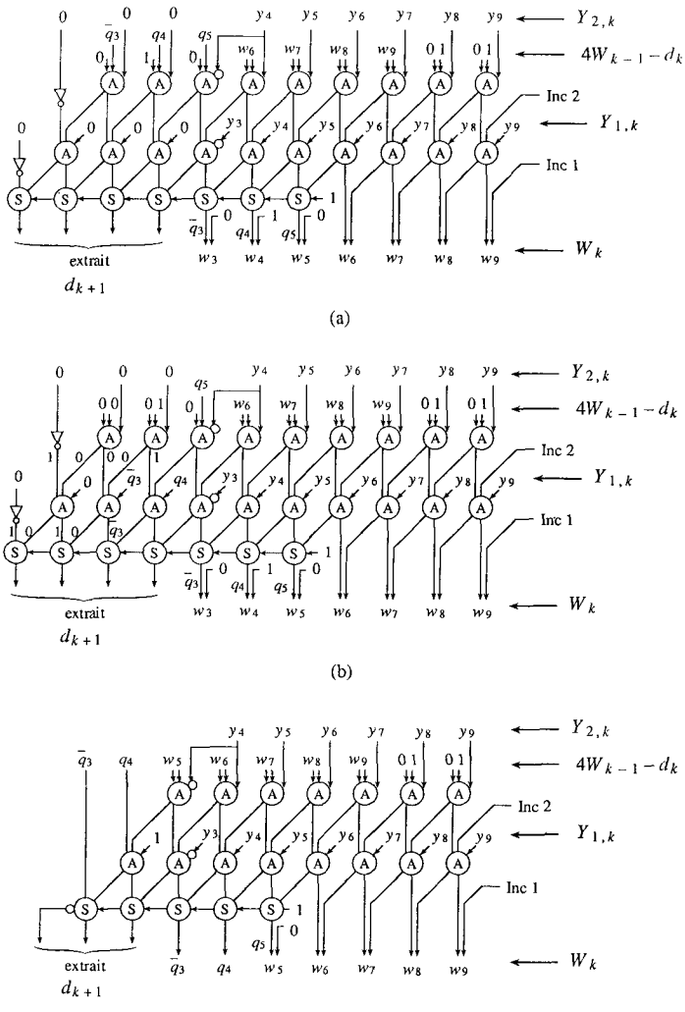

Fig. 4 *a*, *b*, *c*. — Premières versions de l'accumulateur.

Il est évident cette-fois que l'inspection du seul bit q_3 suffit à déterminer l'incrémentation.

Heureusement, aucun matériel supplémentaire n'est requis pour créer ce biais. En effet on peut remarquer à droite du sélecteur (*fig.* 4 c) un bit entrant de valeur 1. Le biais est alors créé en changeant sa valeur pour 0. Mais ce faisant, il faut par la suite corriger ce biais avant de procéder au pas suivant de l'itération. Ceci se réalise en changeant pour la valeur 1, la valeur 0 du bit associé à q_5 (voir *fig.* 4 *c*). Les figures 5 *a* et 5 *b* illustrent la dernière cellule du sélecteur lorsque la première et la deuxième règles d'arrondissement, respectivement, sont appliquées. C'est ici que survient la simplification. A la figure 5 *c*, le signal a est substitué au bit associé à q_5. Les deux entrées de la cellule alors laissées vacantes sont mises respectivement à 0 et 1. Deux situations se présentent. Lorsque $a = 0$, la configuration du circuit correspond exactement à celle de la figure 5 *a*, et lorsque $a = 1$, celle de la figure 5 *b*. Avec un tel circuit les deux règles d'arrondissement sont conjointement appliquées. Et comme il y a deux entrées constantes à la cellule, on peut encore simplifier le circuit (*fig.* 5 *d*).

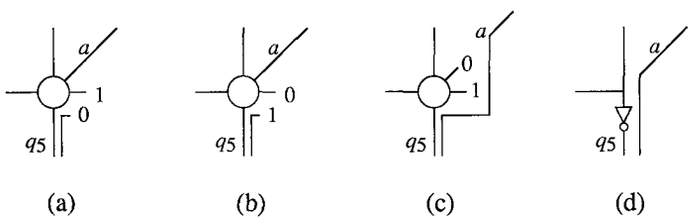

Fig. 5. — Cellule à droite du sélecteur.

(*a*) Mise en œuvre de la règle N° 1.
(*b*) Mise en œuvre de la règle N° 2.
(*c*) Mise en œuvre conjointe.
(*d*) Simplification.

Le tableau 4 présente le nouveau regroupement des valeurs d'approximation avec l'application conjointe des deux règles d'arrondissement.

d	$4Q_{k-1}$	$4W_{k-1}$	$4W_k$
+3	{ 2.5, ... , 3.125 }	(2.375, 3.0625)	(-3.0625, 0.8125)
+2	{ 1.5, ... , 2.375 }	(1.375, 2.625)	(-3.0625, 3.0625)
+1	{ 0.5, ... , 1.375 }	(0.375, 1.625)	(-3.0625, 3.0625)
0	{ -0.5, ... , 0.375 }	(-0.625, 0.625)	(-3.0625, 3.0625)
-1	{ -1.5, ... , -0.625 }	(-1.625, -0.375)	(-3.0625, 3.0625)
-2	{ -2.5, ... , -1.625 }	(-2.625, -1.375)	(-3.0625, 3.0625)
-3	{ -3.125, ... , -2.625 }	(-3.0625, -2.375)	(-0.8125, 3.0625)

Tableau 4. Regroupement des valeurs d'approximation avec mise en œuvre conjointe des règles N° 1 et N° 2

La version finale de la réalisation est montrée à la figure 6. On peut constater avec la réalisation originale (*fig.* 4 *a*) qu'une simplification de sept cellules a eu lieu. De plus, si l'on considère que le parcours critique se situe à l'intérieur du sélecteur, la propagation de la retenue à cet endroit ne traverse maintenant que quatre cellules, soit un gain de trois cellules par rapport à la version originale.

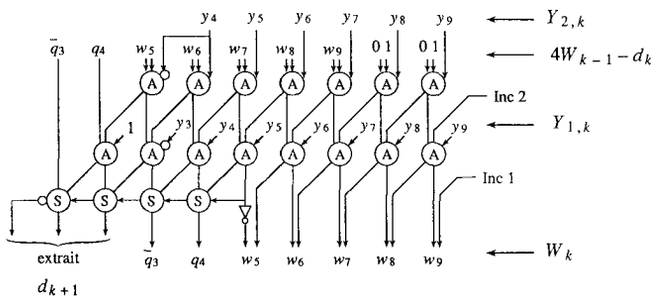

Fig. 6. — Version finale de l'accumulateur.

5. Filtre tout pôle du second ordre

Tout est prêt maintenant pour la rélisation du filtre tout pôle du second ordre. Il s'agit tout simplement d'empiler autant d'accumulateurs qu'il y a de produits partiels à sommer, plus deux accumulateurs pour tenir compte du délai. Ce dernier point sera justifié plus tard. La figure 7 montre un exemple avec six accumulateurs. A noter que le décalage successif des accumulateurs s'explique par la pondération décroissante des produits partiels. Afin de faciliter la discussion autour du tableau multiplieur, nous ouvrons provisoirement la boucle de récursion. L'opération ainsi effectuée devient :

$$(5.1)\quad y(n) = b_1 \cdot q(n-1) + b_2 \cdot q(n-2) + x(n) .$$

La variable excitatrice $x(n)$ joue ici le rôle de résidu initial. Les variables $q(n-1)$ et $q(n-2)$ sont les multiplicateurs. Ceux-ci sont représentés sur le dessin par les chiffres $q_k^{(\cdot)}$. La sortie $y(n)$ est représentée sur le dessin soit par l'ensemble des chiffres d_{k+3}, soit par l'ensemble des chiffres y_k.

On peut remarquer que d'autres dispositifs se greffent à la structure de l'accumulateur. Des registres sont intercalés entre les accumulateurs de manière à créer une structure en pipeline. Un registre est formé d'une rangée de

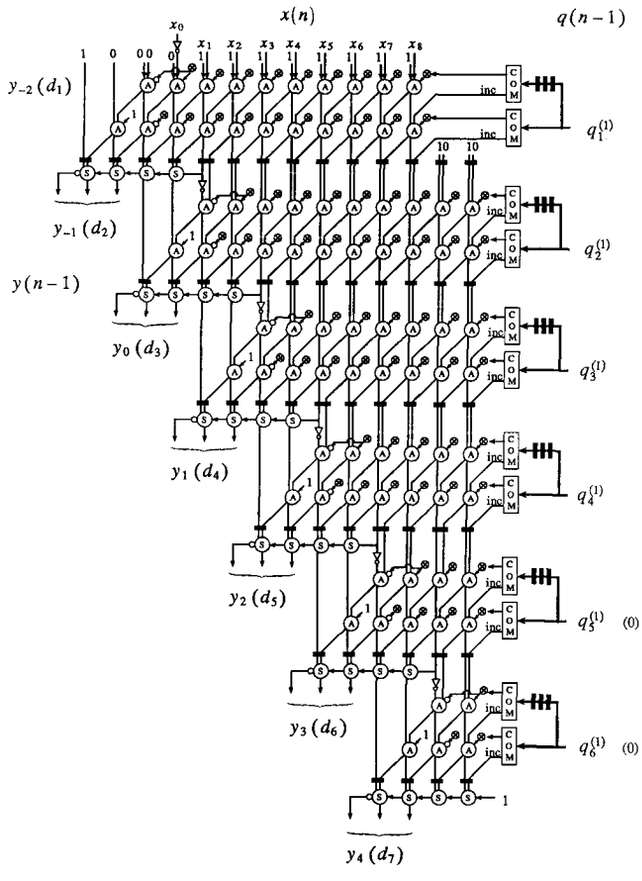

Fig. 7. — Première version du tableau multiplieur.

bascules symbolisées par les petits rectangles noirs. Il est positionné à la suite d'une paire d'additionneurs hybrides, juste avant le sélecteur, de manière à garder une certaine régularité avec les additionneurs. Il est intéressant de souligner qu'un accumulateur est délimité par deux registres. Ainsi, un sélecteur est associé avec la paire d'additionneurs hybrides placés à sa droite. Le sélecteur génère donc l'extrait d_k, et non plus d_{k+1} comme c'était le cas avec les figures 4 et 6. Il est clair que l'extrait d_1 n'est généré par aucun sélecteur à la figure 7, car le premier accumulateur n'en possède pas.

On remarque également que des multiplexeurs sont placés à l'entrée des additionneurs hybrides (ce sont les petits cercles marqués d'un X). Ils ont pour fonction de générer les produits partiels à partir des coefficients. Précisons tout de suite qu'un produit partiel est généré par décalage et complément vrai d'un coefficient. Soit q, un chiffre multiplicateur, et b, un coefficient. Ainsi, lorsque $|q| = 1$, la valeur affichée par le multiplexeur est b. Lorsque $|q| = 2$, un décalage d'une position est en même temps effectué. Enfin, lorsque $|q| = 3$, la valeur $3\,b$ est affichée. On assume ici l'existence de deux bus par coefficient, l'un transportant la valeur directe (b_1 ou b_2) et l'autre transportant la valeur triplée ($3\,b_1$ ou $3\,b_2$). Dans ce dernier cas, la valeur est préalablement calculée avant la mise en opération du filtre. On doit ajouter deux autres situations. Lorsque $q = 0$, la valeur zéro est affichée, et lorsque $q < 0$, les mêmes opérations précédentes sont effectuées mais en ajoutant l'opposé de la valeur affichée. Cette dernière opération est effectuée en inversant les bits du nombre à opposer. Pour compléter, un bit doit être additionné à la position de poids faible. Pour éviter un additionneur supplémentaire, le bit est additionné après le multiplexeur, par un des additionneurs hybrides. Ceci éclaire la fonction du signal **Inc** discuté à la section 2. Ainsi, lorsqu'un bit doit être additionné, le signal **Inc** prend la valeur 1.

Les multiplexeurs reçoivent leurs commandes respectives par les boîtes à droite du tableau, celles marquées COM. Celles-ci interprètent la valeur des chiffres multiplicateurs pour fournir les commandes appropriées. Comme il y a deux opérandes multiplicateurs, il y a deux boîtes par accumulateur. Une des boîtes est précédée de trois registres afin de créer le retard nécessaire pour obtenir le second multiplicateur, $q(n-2)$, à partir du premier, $q(n-1)$.

La figure 7 montre le tableau tronqué à droite. La troncature est faite de telle manière que la partie supprimée a peu ou pas d'influence sur l'issue du résultat. La réduction du matériel qui en résulte est ainsi assez substantielle.

L'utilisation populaire de la numération complément à deux nous encourage en quelque sorte, à la section 4, à représenter les coefficients dans cette numération. Ceux-ci sont en effet des variables externes. La variable excitatrice $x(n)$ est aussi une variable externe et, de ce fait, représentée en complément à deux. Or, la valeur du résidu initial, une variable en binaire redondant, est donnée par $x(n)$. Une conversion a donc lieu. Celle-ci est facilement réalisée en associant à chacun des bits de $x(n)$ un second bit de valeur 1, à l'exception du bit de poids fort qui est inversé puis associé à 0. La validité de cette procédure peut être vérifiée au tableau 1. La figure 7 montre cette conversion effectuée en même temps que pénètre $x(n)$ en haut du tableau.

Avant de fermer la boucle de récursion, il est impératif de connaître exactement le poids respectif de chacun des chiffres que comportent le résultat et les multiplicateurs. Pour commencer, on suppose les représentations suivantes des multiplicateurs :

$$q(n-1) = \sum_{k=1}^{m} 4^{-k}\, q_k^{(1)} \tag{5.2a}$$

$$q(n-2) = \sum_{k=1}^{m} 4^{-k}\, q_k^{(2)} \tag{5.2b}$$

$$q_k^{(\cdot)} \in \{-3, \ldots, 3\}\ .$$

On sait qu'un chiffre multiplicateur multiplie un coefficient pour générer un produit partiel. On sait également que les produits partiels sont sommés par paire. Ainsi la somme d'une paire est :

$$b_1 \cdot q_k^{(1)} + b_2 \cdot q_k^{(2)}\ .$$

Cette expression joue en fait le rôle de l'ajout. Toutefois, la condition de chevauchement impose que cet ajout soit reproportionné avec le facteur 2^{-6}, suivant le résultat de (4.8). On a donc :

$$Y_k = \frac{b_1}{64}\, q_k^{(1)} + \frac{b_2}{64}\, q_k^{(2)}\ . \tag{5.3}$$

L'accumulation des ajouts donne :

$$\begin{aligned}\sum_{k=1}^{m} 4^{-k}\, Y_k &= \sum_{k=1}^{m} 4^{-k} \left(\frac{b_1}{64}\, q_k^{(1)} + \frac{b_2}{64}\, q_k^{(2)} \right) \\ &= \frac{b_1}{64} \sum_{k=1}^{m} 4^{-k}\, q_k^{(1)} + \frac{b_2}{64} \sum_{k=1}^{m} 4^{-k}\, q_k^{(2)} \\ &= 4^{-3}(b_1 \cdot q\,(n-1) + b_2 \cdot q\,(n-2))\ .\end{aligned}$$

On a utilisé (5.2) pour obtenir le dernier membre. L'expression de l'accumulation totale est complète avec l'addition du résidu initial. On sait que celui-ci est représenté par la variable excitatrice $x(n)$. On a alors :

$$\begin{aligned}X_0 + \sum_{k=1}^{m} 4^{-k}\, Y_k &= 4^{-3}(b_1 \cdot q\,(n-1) + \\ &\quad + b_2 \cdot q\,(n-2) + x(n)) \\ &= 4^{-3}\, y(n)\ .\end{aligned} \tag{5.4}$$

On reconnaît au second membre l'expression de l'équation aux différences (5.1). On sait que le résultat d'une accumulation est représenté par la chaîne $(\,.\,d_1\,d_2\,d_3\,d_4 \ldots)_4$. La dernière relation devient :

$$4^{-3}\, y(n) = (\,.\,d_1\,d_2\,d_3\,d_4 \ldots)_4 \tag{5.5}$$

ou mieux encore :

$$y(n) = (y_{-2}\, y_{-1}\, y_0 \cdot y_1 \cdots)_4 \cdot \tag{5.6}$$

En conclusion, on observe à la sortie un décalage de trois positions du point de numération.

Comme les représentations des multiplicateurs commencent avec le chiffre q_1, il faut supprimer d'une manière quelconque les positions occupées par les chiffres y_{-2}, y_{-1} et y_0 avant de pouvoir fermer la boucle. L'astuce consiste à imposer une amplitude limite au résultat de la multiplication de manière à ce que les chiffres avant le point soient nuls. Les chiffres étant nuls, les sélecteurs qui leur sont attachés peuvent être supprimés. Dans ces conditions, trois ajouts sont sommés avant de procéder à une première sélection.

L'amplitude limite se trouve être la limite absolue de la plage occupée par le résidu, celle que l'on trouve à la colonne 4 du tableau 4. La limite absolue est $3{,}0625 \cdot 4^{-1}$, exclusivement. En effet, on suppose, dans un premier temps, que tous les ajouts sont nuls et que le résidu initial est légèrement inférieur à $3{,}0625 \cdot 4^{-4}$. De cette manière, le résultat de la multiplication sera légèrement inférieur à la limite positive. Au quatrième pas de l'itération, la première sélection a lieu. Le résidu ancien $4\,W_3$ est alors légèrement inférieur à 3,0625 et génère l'extrait maximal $d_4 = 3$. Comme le résidu est à l'intérieur des limites de la plage permise au tableau 4, la convergence est assurée. De façon analogue, lorsque le résidu initial est légèrement supérieur à $-3{,}0625 \cdot 4^{-4}$ et que les ajouts sont nuls, le premier extrait est $d_4 = -3$ et la convergence est assurée.

L'hypothèse de départ à l'effet que les ajouts sont nuls peut sembler restrictive mais correspond en fait à la pire situation. En effet, si la valeur de l'accumulation totale $3{,}0625 \cdot 4^{-4}$ est en partie contenue dans les ajouts, ceci signifie que l'amplitude des premiers résidus est inférieure, ou à tout le moins égale, à ceux du cas étudié, ce qui allège la situation. En conclusion, lorsque le résultat de la multiplication est à l'intérieur des limites spécifiées ci-haut, le premier extrait ne déborde jamais du répertoire $\{-3, \ldots, 3\}$ et la convergence est assurée.

La modification proposée consiste à faire la somme des trois premières paires de produits partiels avant de procéder à la première sélection. Ceci a pour effet de supprimer les deux premiers sélecteurs du tableau multiplieur (*fig.* 7). Pour ce faire, quelques modifications doivent être apportées aux représentations des produits partiels et du résidu initial, afin d'ajuster leurs tailles en fonction des additionneurs hybrides utilisés. Le tableau 5 a montre les sept opérandes décomposées en bits. Les deux dernières rangées au-dessus du trait horizontal montrent le résidu initial donné par $x(n)$, ainsi que les bits associés servant à la conversion de ce dernier en binaire redondant. Sous le trait apparaissent les bits à emprunter au résidu pour corriger le biais créé lorsque les bits négatifs des produits partiels sont substitués pour des bits positifs inversés. Également apparaît un biais à donner au résidu pour simplifier la mise en œuvre de la sélection suivante. Il faut mentionner que deux positions supplémentaires sont ajoutées à gauche du résidu pour permettre l'addition des biais. Ces positions sont occupées par des chiffres binaires redondants égaux à 0.

Les ajustements des tailles des premiers opérandes sont effectués au tableau 5 b. De même les biais ont été ajoutés au résidu en modifiant de manière appropriée la chaîne de bits associés. Il est à noter que l'addition hybride des produits partiels s'effectue de haut en bas, avec comme partie initiale le résidu. Afin d'épargner du matériel, les bits imprimés en gras des deux premiers produits partiels sont additionnés plus tard en même temps que les deux derniers produits partiels (tableau 5 c). Les nombreux zéros qui apparaissent alors permettent de nombreuses simplifications.

(a)	$Y_{2,0}$	:				$-y_4$	$.y_5$	y_6	y_7	y_8	y_9	...
	$Y_{1,0}$	:			$-y_3$	y_4	$.y_5$	y_6	y_7	y_8	y_9	...
	$Y_{2,1}$	:					.	$-y_4$	y_5	y_6	y_7	...
	$Y_{1,1}$	:					$.-y_3$	y_4	y_5	y_6	y_7	...
	$Y_{2,2}$	:					.			$-y_4$	y_5	...
	$Y_{1,2}$	:					.		$-y_3$	y_4	y_5	...
	$x(n)$	:		0	0	$\bar{x}_0$	$.x_1$	x_2	x_3	x_4	x_5	...
	bits associés	:		1	1	0	.1	1	1	1	1	...
	emprunts	:			-1	-1	-1	-1	-1	-1		
	biais	:							$+1$			
(b)	$Y_{2,0}$	:	-0	0	0	$\bar{y}_4$	$.y_5$	y_6	y_7	y_8	y_9	...
	$Y_{1,0}$	:	-0	0	$\bar{y}_3$	y_4	$.y_5$	y_6	y_7	y_8	y_9	...
	$Y_{2,1}$	:	-0	0	0	0	.0	$\bar{y}_4$	y_5	y_6	y_7	...
	$Y_{1,1}$	:	-0	0	0	0	$.\bar{y}_3$	y_4	y_5	y_6	y_7	...
	$Y_{2,2}$	:	-0	0	0	0	0	0	0	$\bar{y}_4$	y_5	...
	$Y_{1,2}$	:	-0	0	0	0	0	0	$\bar{y}_3$	y_4	y_5	...
	$x(n)$	:		0	0	$\bar{x}_0$	$.x_1$	x_2	x_3	x_4	x_5	...
	bits associés	:		0	1	1	.0	0	1	0	1	...
(c)	$Y_{2,0}$	:	-0	0	0	0	.0	0	y_7	y_8	y_9	...
	$Y_{1,0}$	:	-0	0	0	0	.0	0	y_7	y_8	y_9	...
	$Y_{2,1}$	:	-0	0	0	0	.0	$\bar{y}_4$	y_5	y_6	y_7	...
	$Y_{1,1}$	:	-0	0	0	0	$.\bar{y}_3$	y_4	y_5	y_6	y_7	...
	$Y_{2,0}$ et $Y_{2,2}$	:	-0	0	0	$\bar{y}_4$	$.y_5$	y_6	0	$\bar{y}_4$	y_5	...
	$Y_{1,0}$ et $Y_{1,2}$	:	-0	0	$\bar{y}_3$	y_4	$.y_5$	y_6	$\bar{y}_3$	y_4	y_5	...
	$x(n)$	:		0	0	$\bar{x}_0$	$.x_1$	x_2	x_3	x_4	x_5	...
	bits associés	:		0	1	1	.0	0	1	0	1	...

Tableau 5. Modifications apportées aux opérandes des trois premiers accumulateurs du filtre du second ordre

a) Avant modification
b) Ajustement des tailles
c) Déplacement de certains bits

La figure 8 illustre la manière d'effectuer ces simplifications. Chaque section représente un accumulateur. Un premier sélecteur apparaît à la fin. Le premier accumulateur fait l'addition de la première paire de produits partiels avec le résidu. Comme souhaité, les premiers bits en gras appartenant à cette paire sont déplacés au troisième accumulateur. Signalons que les premiers bits x_0 et

x_1 du résidu sont aussi déplacés, mais au deuxième accumulateur cette fois. Il résulte de ces changements que la majeure partie des cellules d'addition ont leurs entrées constantes, ce qui autorise leur simplification. On observe que les deux premiers couples de bits à gauche à la sortie de l'accumulateur codent chacun le chiffre zéro en binaire redondant. Pour ne pas allonger inutilement l'accumulateur suivant, ces couples sont mis de côté.

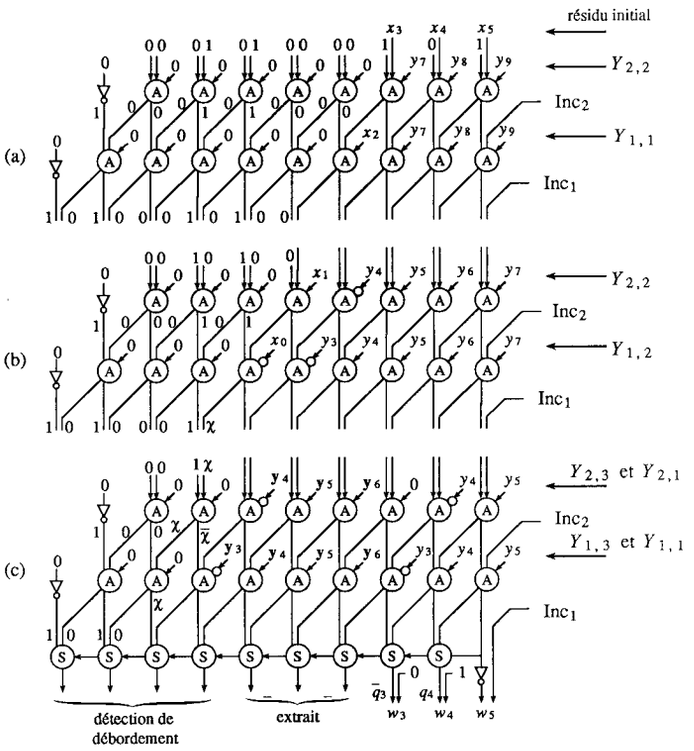

Fig. 8. — Réalisation des trois premiers accumulateurs simplifiés de deux sélecteurs.

Le deuxième accumulateur reprend le résultat du premier et l'additionne à la deuxième paire de produits partiels. Les bits x_0 et x_1 du résidu initial y sont inclus. Encore une fois, une grande partie des cellules d'addition ont leurs entrées constantes ; donc leur simplification. De même, les deux premiers couples de bits à la droite du résultat sont mis de côté.

Le troisième accumulateur additionne la troisième paire de produits partiels, plus les bits de la première paire déplacés précédemment. On constate que le sélecteur à la sortie est exceptionnellement long. Toutefois les deux cellules à gauche peuvent être simplifiées en raison de certaines de leurs entrées qui sont constantes. De tous les bits qui émergent du sélecteur, seulement trois sont utiles en fait pour représenter l'extrait. En effet, s'il n'y a pas débordement du résultat de la multiplication, l'extrait appartient nécessairement au répertoire $\{-3, ..., 3\}$. Dans le cas contraire, les bits à gauche de l'extrait peuvent servir à la détection du débordement.

Les simplifications apportées aux trois premiers accumulateurs sont montrés à la figure 9. Cette figure illustre la réalisation du filtre tout pôle du second ordre. La boucle de récursion est cette fois fermée. Signalons la présence du petit circuit en haut à gauche qui a pour fonction de déplacer les premiers bits de la première paire de produits partiels. Le déplacement dans le temps est simplement créé par deux registres.

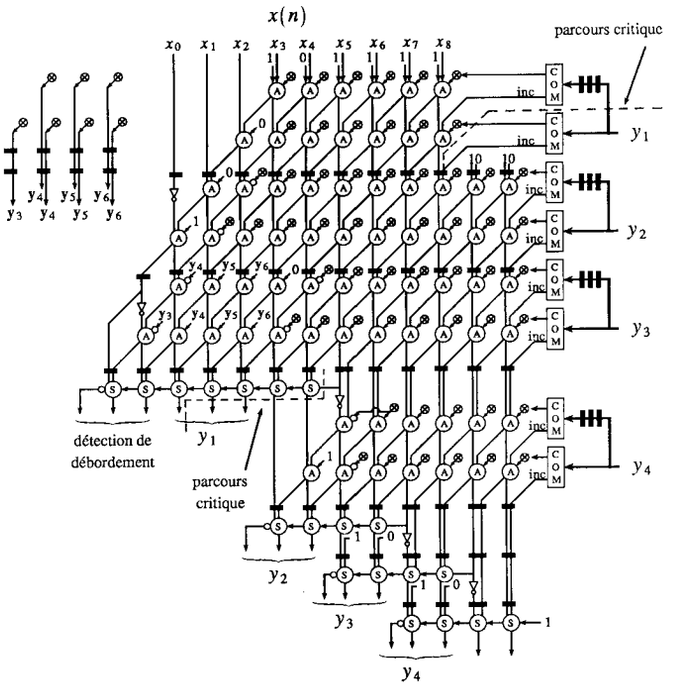

Fig. 9. — Réalisation du filtre tout pôle du second ordre.

Le tableau multiplieur de la figure 9 montre qu'il y a trois registres entre l'entrée du résidu initial et la sortie du premier extrait. Le multiplieur a donc un délai de trois périodes. Ce même délai est aussi contenu dans la boucle de récursion. Pour assurer une synchronisation adéquate, trois registres sont utilisés pour retarder le multiplicateur $y(n-1)$ afin d'obtenir $y(n-2)$. Ces registres sont placés juste avant les boîtes de commandes réservées au deuxième multiplicateur.

Si on revient à la figure 7, on note que le dernier extrait à la sortie est y_4. Par conséquent les deux derniers chiffres du multiplicateur $q(n-1)$, q_5 et q_6, n'ont aucun correspondant à la sortie.

Pour cette raison ils sont mis volontairement à zéro. Mais ce faisant, les deux paires d'additionneurs hybrides placées à la fin du tableau s'avèrent inutiles et peuvent de ce fait être supprimées. C'est ce que l'on montre ici. A titre d'exemple, le tableau 6 a montre la représentation du résidu qui est restitué par le quatrième accumulateur du tableau. A la suite de cet accumulateur aucun ajout n'est bien sûr effectué. Toutefois la sélection suivante exige qu'un biais de 2^{-3} soit additionné au résidu afin de

simplifier l'opération, ce qui se fait par une modification appropriée de la chaîne des bits associés (tableau 6 b). A la figure 9, on peut apercevoir à la fin du circuit les deux avant derniers sélecteurs privés de leurs additionneurs hybrides respectifs. On note à l'endroit de l'avant dernier sélecteur que les deux règles d'arrondissement ont été conjointement appliquées ; d'où la présence d'un inverseur à sa droite.

(a)	résidu :	$\cdot\bar{q}_3$	q_4	q_5	w_6	w_7	w_8
		0	1	1			
(b)	résidu :	$\cdot\bar{q}_3$	q_4	q_5	w_6	w_7	w_8
		1	0	0			
	sélection :	$-d_2$	d_1	d_0	$\cdot\bar{q}_3$	q_4	w_5	w_6
		extrait			1	0	0	

Tableau 6. Exemple de sélection opérée sur un résidu à la fin du tableau multiplieur

La démarche utilisée dans cet article a permis, de manière systématique et sans trop de difficultés, d'obtenir une réalisation très efficace du filtre tout pôle du second ordre. Par efficacité on entend un minimum de matériel et le moins possible de connexions longues. De plus, on a constaté qu'un pipeline a été intégré facilement à l'intérieur de la structure, malgré l'existence d'une récursion. Ajoutons enfin que la structure obtenue fait appel essentiellement à des cellules courantes, à savoir des bascules, des multiplexeurs et des cellules d'addition.

Un fait remarquable attribué à cette structure est que le délai du tableau multiplieur comporte trois périodes seulement, et ce, quelle que soit la taille du résultat de la multiplication. Un autre fait à souligner est que la redondance caractérisant les additionneurs hybrides les affranchit de la propagation de la retenue, ce qui implique un temps d'exécution court et constant, quelle que soit la taille des ajouts. Bref, la structure proposée garde une performance constante, quelle que soit la taille des variables traitées.

Un autre aspect intéressant à souligner est le rapprochement que l'on peut faire avec les structures systoliques [114]. En effet, on observe que notre structure possède une certaine modularité et régularité. De plus, le réseau des chemins de communication est, pour la plupert, à mailles courtes, à l'exception toutefois d'un parcours relativement long qui débute avec la propagation de la retenue traversant un sélecteur, et qui se poursuit de l'autre côté du tableau avec la traversée d'une boîte de commande, d'une cellule de multiplexage et d'une cellule d'addition (voir ligne pointillée, *fig.* 9). Cependant, il est possible de minimiser ce parcours critique en utilisant, par exemple, une technique d'accélération de la retenue à l'endroit du sélecteur (on peut suggérer l'addition conditionnelle), et en jouant sur les dimensions des transistors à l'endroit de la boîde de commande et de la cellule de multiplexage. Le circuit en entier aurait avantage à implanter une cellule d'addition avec traversée rapide de la retenue (par exemple [12, page 318]). On peut conclure que le circuit est facile à concevoir du fait de la modularité et de la régularité, et qu'il subit sans trop de peine les contraintes de la haute intégration du fait d'un réseau de communication à courtes mailles. On obtient de la sorte une métrique élevée sur le rapport de la puissance de calcul sur le débit des entrées/sorties.

Les variables externes, telles les coefficients et la variable excitatrice, sont en complément à deux. Il serait souhaitable de convertir la sortie dans cette même numération. Il existe un algorithme qui effectue très rapidement la conversion d'un nombre redondant [13]. Il procède en série, poids fort en tête. De plus, il possède l'immense avantage de s'affranchir de la propagation de la retenue, ce qui lui confère une performance intéressante pour notre circuit. De cette façon, la conversion peut avoir lieu au fur et à mesure de la génération des extraits et rendre le résultat dès le dernier pas de l'itération.

Le fait que le filtre soit du second ordre permet de mettre en œuvre aussi bien des pôles réels que des pôles complexes. Il est alors possible de réaliser un système tout pôle quelconque d'ordre multiple en plaçant en cascade plusieurs de ces filtres. Mais si on disposait en plus d'un filtre tout zéro, il serait alors possible de créer un système général comportant des pôles et des zéros. Ce type de filtre est l'objet de la section qui suit.

6. Filtre tout zéro du second ordre

La structure obtenue précédemment peut également servir à réaliser, grâce à quelques modifications, le filtre tout zéro du second ordre. Il suffit pour cela d'ouvrir la boucle de récursion et de substituer aux multiplicateurs les variables $x(n-1)$ et $x(n-2)$. L'équation aux différences s'écrit dans ce cas :

$$(6.1) \quad y(n) = x(n) + a_1 \,.\, x(n-1) + a_2 \,.\, x(n-2) \,.$$

La mise en œuvre de cette équation se fait comme avant à l'aide de deux multiplieurs. Cette fois les coefficients a_1 et a_2 remplacent b_1 et b_2, respectivement. Comme on utilise le même tableau multiplieur, les coefficients a_1 et a_2 sont limités de la même manière que b_1 et b_2, comme suit :

$$(6.2) \qquad |a_1| < 2 \,, \quad |a_2| < 1 \,.$$

Il est important de signaler ici que, à cause des limites imposées par (6.2), les zéros sont positionnés au voisinage du cercle unité. On verra plus tard comment déplacer les zéros à l'extérieur du cercle.

La réalisation du filtre tout zéro est illustré à la figure 10. On note que, en plus du tableau multiplieur dont on reconnaît la structure, un bus est placé à droite pour le

transport de $x(n-1)$, et quelques boîtes marquées CONV sont placées entre le bus et le tableau. Trois registres sont ajoutés au début du bus afin de produire $x(n-1)$ à partir de $x(n)$. D'autres sont ajoutés le long du bus pour assurer la synchronisation entre les données qui y circulent et les opérations effectuées dans le tableau.

La variable $x(n-1)$ est représentée en complément à deux à l'instar de $x(n)$. Mais comme les multiplicateurs doivent être dans une numération redondante en base 4 de répertoire $\{-3, ..., 3\}$, une conversion s'impose à l'endroit de $x(n-1)$. Ce qui explique la présence des boîtes CONV qui effectuent cette opération. La procédure de conversion est démontrée comme suit. Soit X, un nombre en complément à deux :

(6.3) $\quad X = (x_0 \,.\, x_1\, x_2\, x_3\, x_4\, x_5\, x_6)_2 \,.$

On suppose dans un premier temps que X est positif, donc $x_0 = 0$. Un nombre en base 4 est formé en groupant par deux les bits de X. On obtient alors :

(6.4) $\quad X = (\,.\, (0\, x_1\, x_2)_2\, (0\, x_3\, x_4)_2\, (0\, x_5\, x_6)_2)_4 \,.$

Un bit supplémentaire de valeur 0 et de poids fort négatif est ajouté à chaque groupe pour que chacun code un chiffre en complément à deux. Les chiffres sont dans le répertoire $\{0, ..., 3\}$. Le résultat de la conversion appartient ainsi à la numération cible dont le répertoire est $\{-3, ..., 3\}$.

On suppose, dans un deuxième temps, que X est strictement négatif, impliquant alors $x_0 = 1$. On exclut pour le moment la situation où $x_1 = 0$ et $x_2 = 0$. Le regroupement des bits donne cette fois :

(6.5) $\quad X = (\,.\, (1\, x_1\, x_2)_2\, (0\, x_3\, x_4)_2\, (0\, x_5\, x_6)_2)_4 \,.$

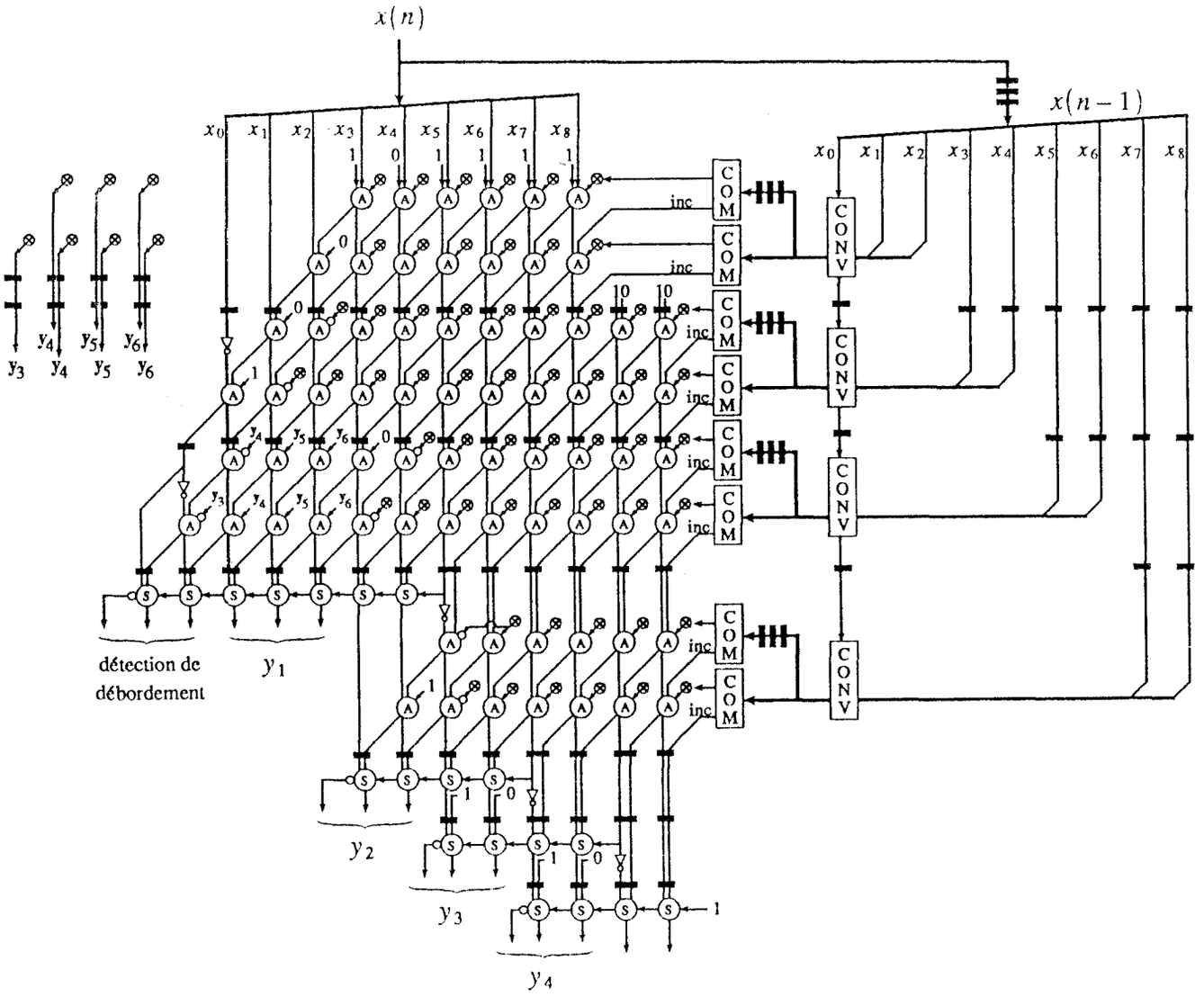

Fig. 10. — Réalisation du filtre tout zéro du second ordre.

Notez la présence du bit 1 à l'intérieur du premier groupe à gauche. Le chiffre que code ce groupe appartient cette fois au répertoire $\{-3, ..., -1\}$. Les autres groupes demeurent dans le répertoire précédent. Le résultat de la conversion appartient encore à la numération cible.

Le cas spécial où $x_1 = 0$ et $x_2 = 0$, a été exclu parce que le premier chiffre aurait alors donné -4, ce qui déborde évidemment du répertoire autorisé. Pour remédier à cette situation, la valeur 1 est additionnée au premier groupe et -4 est additionné au second. L'effet combiné de ces deux opérations s'annule. On obtient dans ce cas :

$$X = (\,.\,(1\ 0\ 1)_2\,(1\,x_3\,x_4)_2\,(0\,x_5\,x_6)_2)_4\,. \tag{6.6}$$

Par cet artifice, les deux premiers chiffres sont amenés parmi $\{-3, ..., -1\}$. Mais avant d'entreprendre une telle action, il est impératif d'exclure la situation où $x_3 = 0$ et $x_4 = 0$, sinon le second chiffre risquerait, comme précédemment, de déborder du répertoire autorisé. Dans le cas contraire, la procédure d'exception est répétée entre le second et le troisième groupe. La procédure d'exception se répète ainsi de suite jusqu'à ce qu'il soit assuré que tous les chiffres du résultat appartiennent au répertoire autorisé. La procédure est alors terminée.

La conversion s'exécute séquentiellement du poids fort au poids faible, ce qui est justement l'ordre dans lequel les chiffres multiplicateurs pénètrent dans le tableau multiplieur. Ce sont les boîtes CONV qui réalisent cette procédure. Il convient de signaler que la connexion qui relie une boîte à l'autre transmet un cas d'exception. Naturellement, une bascule est placée sur cette connexion pour assurer la synchronisation.

Mentionnons au sujet de la réalisation de la conversion qu'elle requiert très peu de matériel car elle se résume, en somme, à la détection des cas d'exception. En effet les additions qui ont lieu requièrent une quantité négligeable de matériel puisqu'elles consistent à remplacer certains bits du nombre par des bits 1 lors des cas d'exception. Par exemple, lorsqu'on passe de (6.5) à (6.6), un bit 1 remplace x_2 dans le premier groupe ainsi que le bit 0 dans le second groupe.

Le problème soulevé au début avec le positionnement des zéros au voisinage du cercle unité peut aisément être résolu. En effet il suffit de permettre le décalage vers le poids fort de la variable $x(n-1)$ à l'intérieur du bus. Ainsi, pour chaque décalage d'une position, il se produit une multiplication par deux sur les coefficients de l'équation aux différences. Pour s'en convaincre, on réécrit l'équation (6.1) comme suit :

$$y(n) = x(n) + (2^{-\alpha}\,a_1^*)(2^{\alpha}\,x(n-1)) + \\ + (2^{-\alpha}\,a_2^*)(2^{\alpha}\,x(n-2)) \tag{6.7}$$

et on fixe α de telle sorte que :

$$|2^{-\alpha}\,a_1^*| = |a_1| < 2\,, \quad |2^{-\alpha}\,a_2^*| = |a_2| < 1\,. \tag{6.8}$$

Les coefficients a_1^* et a_2^* appartiennent à l'équation aux différences à mettre en œuvre, et a_1 et a_2 sont les coefficients du tableau multiplieur.

La performance est ici du même ordre que celle du filtre tout pôle. La période d'échantillonnage est conditionnée, d'une part, par le délai de trois périodes du tableau multiplieur et, d'autre part, par la durée du parcours critique qui est, doit-on signaler, un peu plus longue pour le filtre tout pôle. Mentionnons aussi le fait que, dans les deux cas, le délai de trois périodes oblige à intercaler deux échantillons vides entre deux échantillons vrais, ceci afin d'assurer une synchronisation adéquate. Étant donné un fonctionnement semblable entre les deux types de filtres, il est facile de réaliser un système général d'ordre multiple en cascadant un nombre approprié de ces filtres. Fait surprenant, la performance ne varie pas en fonction ni de la taille des variables, ni de l'ordre du système.

7. Conclusion

La nouvelle méthode développée dans cet article a permis de réaliser, systématiquement et sans trop de difficultés, un filtre tout pôle du second ordre. Malgré la présence de la boucle de récursion, une grande performance est atteinte en adoptant une structure en pipeline. En raison de la modularité et de la régularité de la structure et du réseau de communication à mailles courtes, cette réalisation est un candidat idéal pour l'implantation sur un circuit VLSI. A cet égard, l'emploi exclusif de composants courants, tels des bascules, des multiplexeurs et des cellules d'addition, accélère d'autant l'implantation. Un filtre tout zéro du second ordre a été également réalisé avec une structure similaire. La cascade de ces deux types de filtres permet aisément la réalisation d'un système d'ordre multiple, comportant des zéros et des pôles. Il est remarquable de noter que la performance obtenue ne varie pas en fonction ni de la taille des variables, ni de l'ordre du système.

Manuscrit reçu le 21 novembre 1990.

Remerciements

Ces travaux de recherches ont été effectués avec le support financier du Conseil de Recherches en Sciences naturelles et en Génie du Canada (CRSNG). Les auteurs tiennent à remercier les arbitres de la première version de cet article pour leurs remarques pertinentes qui ont permis d'améliorer sensiblement sa présentation.

BIBLIOGRAPHIE

[1] K. K. Parhi, D. G. Messerschmitt, *A bit-parallel bit level recursive filter architecture,* Proc. IEEE Inter. Conf. on Computer Design, Port Chester, New York, October 6-9, 1986, pp. 284-289.

[2] K. K. Parhi, D. G. Messerschmitt, *Look-ahead computation : Improving iteration bound in linear recursions,* Proc. IEEE Inter. Conf. on Acoustics, Speech and Signal Processing, Dallas, Texas, April 6-9, 1987, pp. 1855-1858, paper 42.18.

[3] K. K. Parhi, D. G. Messerschmitt, *Pipeline interleaving and parallelism in recursive digital filters — Part I : Pipelining using scattered look-ahead and decomposition,* IEEE T. Acoustics, Speech and Signal Processing, Vol. 37, N° 7, July 1989, pp. 1099-1117.

[4] R. F. Woods *et al., Systolic IIR filters with bit level pipelining,* Proc. IEEE Inter. Conf. on Acoustics, Speech and Signal Processing, New York, 1988, pp. 2072-2075, paper V3.8.

[5] S. C. Knowles *et al., Bit-level systolic arrays for IIR filtering,* Proc. IEEE Inter. Conf. on Systolic Arrays, San Diego, May 1988, pp. 653-663.

[6] S. C. Knowles *et al., A bit-level systolic architecture for very high performance IIR filters,* Proc. IEEE Inter. Conf. on Acoustics, Speech and Signal Processing, Glasgow, Scotland, May 23-26, 1989, pp. 2449-2452, paper 17.V2.8.

[7] R. F. Woods *et al., A high performance IIR digital filter chip,* IEEE Intern. Symp. on Circuits and Systems, New Orleans, Louisiana, May 1-3, 1990, pp. 1410-1413.

[8] A. Avizienis, *Signed-digit number representations for fast parallel arithmetic,* IRE T. Electronic Computers, Vol. 10, pp. 389-400, September 1961.

[9] M. Lapointe, *Architecture concurrente et application à des réalisations rapides de filtres numériques invariants et adaptatifs,* Thèse de doctorat, Départ. de Génie électrique, Univ. Laval, Québec, Canada, 1990.

[10] M. Lapointe, P. Fortier, H. T. Huynh, *A new faster and simpler systolic structure for IIR filters,* IEEE Intern. Symp. on Circuits and Systems, New Orleans, Louisiana, May 1-3, 1990, pp. 1227-1230.

[11] H. T. Kung, *Why systolic architectures ?* IEEE Computer, pp. 37-46, January 1982.

[12] N. H. E. Weste, K. Eshraghian, *Principles of CMOS VLSI design : A systems perspective,* Addison-Wesley, VLSI Systems Series, Reading, Massachusetts, 1985.

[13] M. D. Ercegovac, T. Lang, *On the-fly conversion of redundant into conventional representations,* IEEE T. Computers, Vol. 36, N° 7, pp. 895-897, July 1987.