Architecture VLSI faible consommation pour le traitement numérique du signal

A. Heubi, M. Ansorge, F. Pellandini

Institut de Microtechnique
Université de Neuchâtel
Rue de Tivoli 28
CH-2003 Neuchâtel-Serrières, Suisse

RÉSUMÉ

Cet article présente une nouvelle famille d'architectures permettant d'implanter sur silicium une grande variété d'algorithmes de traitement numérique du signal avec une consommation électrique très faible. La régularité structurelle des algorithmes concernés est exploitée afin de simplifier le séquencement, et les blocs sont définis de manière à diminuer leur fréquence de travail. A titre d'exemple, un filtre à réponse impulsionnelle finie d'ordre 20 avec un calcul sur 20 bits ne consomme que 1 mW avec une fréquence d'échantillonnage de 10 kHz et une tension d'alimentation de 5V.

ABSTRACT

This paper presents a novel family of architectures previewed for the VLSI implementation of a large class of digital signal processing algorithms with a very low electrical power consumption. The structural regularity of these algorithms is exploited to simplify the scheduling, and the functional modules are defined such that their processing rate is reduced. As an example, the power consumption of a Finite Impulse Response filter of order 20 with a data wordlength of 20 bits is as low as 1 mW at a sampling rate of 10 kHz and with a supply voltage of 5V.

1. Introduction

Il existe incontestablement une grande demande pour le développement de nouvelles générations d'appareils portatifs à hautes performances dans les domaines des télécommunications, des prothèses auditives et de l'instrumentation électronique.

Les besoins croissants en miniaturisation et en autonomie d'utilisation des appareils portatifs exigent une faible consommation électrique. Dans ces derniers, les fonctions de traitement du signal sont de plus en plus réalisées numériquement et sous forme intégrée vu l'important potentiel qu'offrent ces techniques. La réduction de la consommation électrique des fonctions de traitement numérique du signal implantées sur silicium demande une optimisation aux niveaux de l'algorithme, de l'architecture du processeur, des circuits ainsi que des plans de masques.

Cet article propose une contribution à l'optimisation de l'architecture dont l'objectif est de permettre l'implantation de filtres à réponse impulsionnelle finie ou infinie, de filtres adaptatifs, de transformées de Fourier et autres, sous différentes structures de réalisation tout en assurant une consommation électrique aussi faible que possible.

Les principes de base de l'architecture proposée sont donnés au chapitre 2 alors que le chapitre 3 est consacré à la présentation des principaux blocs fonctionnels utilisés. Ces derniers ont été réalisés jusqu'aux plans de masques pour disposer d'une évaluation précise de leurs caractéristiques essentielles.

On obtient ainsi une collection de blocs fonctionnels permettant d'implanter sous forme de circuits ASIC une grande variété de filtres numériques en procédant à un assemblage et une paramétrisation adéquats.

Finalement le chapitre 4 décrit succinctement un exemple d'application.

2. Architecture

En technologie CMOS, la consommation dynamique est pratiquement toujours prépondérante. Celle-ci s'exprime en première approximation comme suit [Pow91, Pigu92]:

$$P = f \cdot C_{éq} \cdot Vdd^2 \qquad \text{(EQ 1)}$$

où f est la fréquence de travail du circuit considéré, Céq sa capacité électrique équivalente et Vdd la tension d'alimentation.

Les objectifs sont donc, conformément à l'équation 1, d'abaisser autant que possible la fréquence de travail, la capacité équivalente et la tension d'alimentation.

La réduction de la fréquence de travail et de la capacité équivalente est obtenue en ne sollicitant que les ressources qui contribuent directement aux calculs requis. On en déduit les principes suivants:

a) Lorsqu'un circuit est inactif il convient de le mettre en veilleuse (mode STAND-BY).
b) Les différents blocs d'une architecture doivent être définis de manière à limiter le trafic d'information. Il faut privilégier le trafic d'information local.
c) Les grandes mémoires sont subdivisées en plusieurs petites dont une seule est active à la fois.

Par ailleurs, il importe de tirer parti de la régularité structurelle du problème pour simplifier le séquencement des opérations ainsi que le matériel requis.

Enfin, il existe un optimum de puissance entre l'abaissement de la tension d'alimentation et l'augmentation de la capacité équivalente consécutive au plus fort degré de parallélisme requis pour maintenir un temps de calcul compatible avec le problème [Chan92]. Dans ce contexte, il importe que l'architecture se prête à un degré variable de parallélisme.

2.1 Solution proposée

Un calcul série-parallèle est choisi parce qu'il permet une vitesse de calcul suffisamment élevée pour la plupart des applications de filtrage dans le domaine audio (fréquences atteignant jusqu'à quelques dizaines de kHz) tout en offrant une complexité convenable.

Le séquencement se fait de manière hiérarchique de façon à limiter au strict nécessaire la fréquence de travail de chaque partie [Pigu87].

L'ordonnancement des opérations du graphe de fluence des algorithmes à implanter permet souvent un accès séquentiel aux données (coefficients et variables d'état). Pour exploiter cette régularité, on peut construire un ensemble de mémoires à accès séquentiel, leur nombre étant fixé par l'application. Le calcul des adresses se trouve ainsi extrêmement simplifié. Cette approche permet également d'effectuer des décalages virtuels pour la mise à jour des données et permet une simplification extrême des décodeurs d'adresse. La logique spécifique à l'application est simple et peut facilement être générée automatiquement.

Le principe peut être appliqué à des cas plus complexes tels que filtrage adaptatif et transformées de Fourier en adaptant la flexibilité de l'adressage mémoire si nécessaire.

3. Réalisation

Les principaux blocs fonctionnels de l'architecture proposée sont constitués d'une mémoire séquentielle, d'une unité arithmétique ainsi que d'un séquenceur.

3.1 Mémoire à accès séquentiel

Il existe principalement deux possibilités de réaliser une mémoire séquentielle. La première consiste à utiliser un banc de registres à décalage. Dans la seconde, on recourt à une matrice de cellules mémoires, les cellules d'une même ligne étant sélectionnées par un registre séquentiel. Cette dernière réalisation n'implique aucun déplacement global des données comme c'est le cas avec les registres à décalage.

Le choix de la meilleure solution dépend principalement de deux facteurs à savoir la consommation et la surface occupée. Ces deux facteurs sont eux-mêmes fonction de la dimension de la mémoire (nombre de bits · nombre de mots) et ce de manière différente pour les deux solutions. La Figure 1 présente les tableaux comparatifs obtenus à l'aide des plans de masque et de la simulation de ceux-ci.

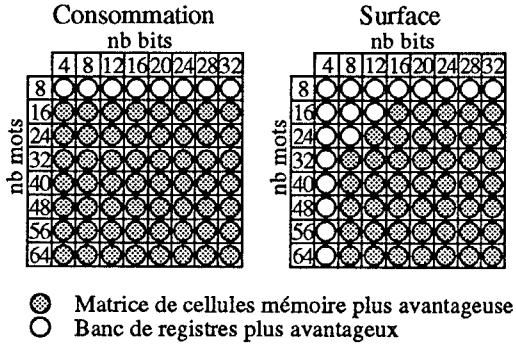

FIGURE 1. Comparaison des solutions pour la mémoire séquentielle

Une solution à matrices de cellules est toujours plus avantageuse pour de grandes dimensions alors que la solution à banc de registre l'est pour de faibles dimensions.

Pour le cas d'une mémoire de 4 bits · 8 mots, la solution à matrice de cellules consomme 1.5 fois plus et est 1.9 fois plus grande qu'une solution à banc de registres. A l'inverse, pour une mémoire de 32 bits · 64 mots, la solution à matrice de cellules consomme 2 fois moins et est 1.5 fois plus petite qu'une solution à banc de registres.

3.1.1 Adressage séquentiel

Pour le système d'adressage séquentiel, la solution classique consiste à implanter un compteur, suivi d'un décodeur d'adresse. Une solution plus efficace, au niveau matériel (sauf pour un espace d'adressage très faible) et au niveau de la consommation est de recourir à un registre à décalage. Mais il importe de prendre quelques précautions pour qu'il fonctionne sans problème. La solution proposée se trouve à la Figure 2.

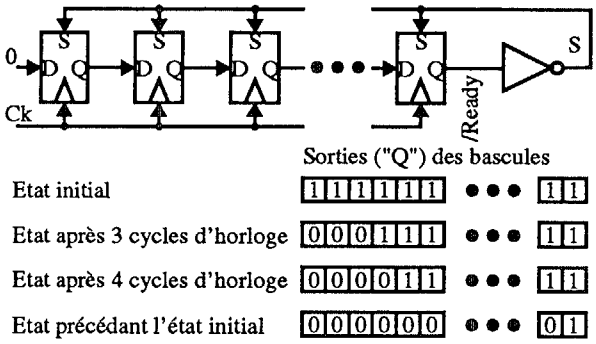

FIGURE 2. Principe de fonctionnement de l'adressage séquentiel

A l'état initial, tous les registre sont à "1" et passent à zéro au rythme du signal d'horloge. Le passage à zéro du dernier registre provoque une remise à un de tous, ce qui permet de retrouver l'état initial. Cette solution est sûre mais exige une logique supplémentaire pour sélectionner la ligne mémoire correspondant à la transition "0" "1" dans la liste des états des registres.

On peut remarquer que seul le passage à zéro doit être synchrone avec le signal d'horloge, le passage à l'état "1" pouvant se propager de manière asynchrone. Outre la simplification qui en résulte, cela augmente la sécurité de fonctionnement vu que le premier "1" depuis la gauche se propage jusqu'à la fin de la structure indépendamment du signal d'horloge.

Il ne peut donc exister qu'*une seule transition stable* "0" "1" dans la liste des états des registres.

3.2 Unité arithmétique

L'une des opérations les plus fréquentes dans le domaine du traitement numérique du signal est celle du calcul de produits scalaires. L'unité arithmétique a ainsi été optimisée pour cette opération.

Dans l'ensemble, on a choisi de représenter les nombres en complément à deux, car ce mode de représentation convient bien au calcul de produits scalaires, malgré les irrégularités qu'il implique lors de la quantification des nombres (troncature, arrondi, ...).

Les coefficients des filtres numériques sont codés selon l'algorithme de Booth afin de réduire d'un facteur deux au moins le nombre d'additions élémentaires requises pour le calcul d'un produit. La solution retenue permet de recoder deux bits à la fois avec un recouvrement d'un seul bit [Sam90] (Tableau 1).

Nombre à encoder			Nombre encodé		Opération correspondante
a_{i+1}	a_i	a_{i-1}	b_{i+1}	b_i	
0	0	0	0	0	+ 0
0	0	1	0	1	+ 1
0	1	0	0	1	+ 1
0	1	1	1	0	+ 2
1	0	0	-1	0	- 2
1	0	1	0	-1	- 1
1	1	0	0	-1	- 1
1	1	1	0	0	- 0

TABLEAU 1. Algorithme de recodage de Booth modifié Avec $a_{-1} = 0$

Par rapport au calcul du produit scalaire par regroupement des produits partiels selon leur poids, l'approche consistant à évaluer une somme de produits nécessite un minimum de transferts mémoire, mais pose le problème de la largeur de l'additionneur. Il est toutefois possible, en décalant les produits partiels de l'additionneur dans un registre, de garder la pleine précision jusqu'à la fin du calcul, comme illustré à la Figure 3. L'additionneur a

une largeur de W_d bits et possède un registre de même largeur alors qu'un registre auxiliaire de largeur W_c bits sauvegarde les bits de poids faible sortants et réintroduit les bits de poids fort du produit précédent.

Au début du calcul d'un produit scalaire, le registre auxiliaire et celui de l'additionneur sont initialisés à 0. Durant la première multiplication les bits de poids faible sortants sont introduits dans le registre auxiliaire. Au début de la multiplication suivante, un échange entre les deux registres permet d'additionner les poids faibles de la première multiplication et de sauvegarder simultanément les poids forts. Ces derniers sont réintroduits au cours de la deuxième multiplication et ainsi de suite.

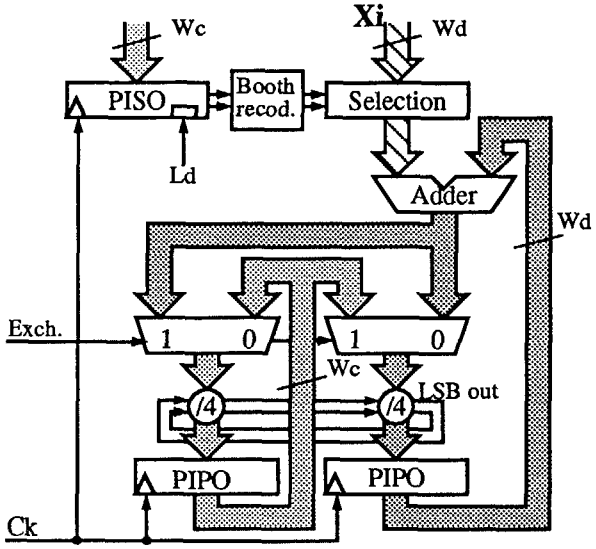

FIGURE 3. Calcul de produits scalaire avec registres auxiliaires

Lorsque ces opérations sont exécutées sur des nombres signés, le bit de poids fort de la donnée x_i exige un traitement particulier pour que le résultat soit correct. En effet, une extension normale du signe engendre des produits partiels pouvant atteindre Wd +Wc bits et l'utilisation du schéma de la Figure 3 devient ainsi impossible pour les nombres signés. Heureusement, ce problème peut être contourné par un recodage du bit de signe en tenant compte de l'extension des bits de signe des produits partiels de poids plus faible selon l'équation 2.

Il est clair que ce procédé ne peut s'appliquer qu'à un calcul commençant par les poids faibles.

$$\tilde{s}_i = s_i \oplus \bigvee_{j=0}^{j-1} s_j$$

s_i: bit de signe du produit partiel i (EQ 2)

L'additionneur utilisé fonctionne avec une sauvegarde du report. La fréquence d'horloge est ainsi indépendante du nombre de bits des opérandes.

Les registres sont réalisés en logique séquentielle à deux phases. Afin de réduire au maximum le surplus de complexité dû aux registres tout en répartissant au mieux les calculs sur les deux phases, il a été décidé d'additionner un produit partiel lors de chacune de ces deux phases. Cette approche permet de calculer les produits à raison de 4 bits par cycle (chaque cycle comprenant deux phases)!

Le temps de calcul d'une multiplication exprimé en fonction du nombre de bits W des opérandes vaut ainsi:

$$T_{calc.\ mult.} = [int(W/4) + 1] \cdot T_{cycle} \quad \text{(EQ 3)}$$

Le temps de latence étant quant à lui:

$$T_{latence\ mult.} = 2 \cdot T_{calc.\ mult.} \quad \text{(EQ 4)}$$

3.3 Séquencement

Pour limiter la fréquence de travail de chaque bloc fonctionnel, le séquenceur a été conçu de manière à permettre un fonctionnement hiérarchique. Il s'agit en fait d'une machine à états finis réalisée par un adressage séquentiel de mémoires ROM. Ce choix a été déterminé par la régularité obtenue.

4. Evaluation des performances

Le "layout" des différents blocs a été dessiné pour avoir une bonne évaluation et pour la vérification fonctionnelle. Ce travail a été exécuté dans l'environnement CAO de la maison COMPASS Design Automation (anciennement VLSI Technology Inc.).

Tous les blocs fonctionnels (mémoire, unité arithmétique, séquenceur) sont paramétrables; à partir d'un assemblage de cellules de base, il est possible, dans certaines limites, de varier le nombre de bits d'un additionneur, d'une mémoire etc.

L'estimation de la rapidité des circuits a été faite par simulation. En général, l'outil utilisé est VTIsim. Pour des cellules particulières telles que celles de la mémoire, le recours à VTIspice peut être obligatoire.

	Mémoire vive	Unité de calcul	Séquenceur
Fréquence de travail max.	25 MHz (32 · 64 bits)	20 MHz (32 · 32 bits)	25 MHz (32 · 64 bits)
Consommation (énergie, donnée en pJ)	8 + nbmots·0.75 + W·14.5 + nbmots·W·0.7 (par cycle lecture + écriture)	110 + W·62 + W^2·7.8 (par multiplication)	8 + nbmots·0.75 + nbbits·1.75 + nbmots · nbbits· 0.1 (par cycle lecture)
Surface (λ = 0.6 µm en technologie CMN12)	(120λ + W·23λ) · (280λ + nbmots·28λ)	1200λ · (300λ + W·35λ)	(215λ + nbbits·12λ) · (70λ + nbmots·12λ)

TABLEAU 2. Résumé des caractéristiques des circuits

Les différentes caractéristiques sont résumées dans le Tableau 2 dans le cas où Wc = Wd = W.

Les résultats sont donnés pour une tension d'alimentation de 5 V et la technologie CMN12 de VLSI Technology Inc. Les simulations ont été faites en tenant compte de toutes les capacités associées au circuit réel et avec les paramètres "slow-slow" des transistors.

5. Exemple d'application

L'exemple choisi est celui d'un filtre RIF (à réponse impulsionnelle finie) d'ordre 20 avec des nombres représentés sur 20 bits.

Pour un filtre de ce type, le séquencement peut être simplifié, seule l'unité arithmétique ayant besoin d'une unité de séquencement.

Une multiplication de 20·20 bits se fait en 6 cycles (équation 3) avec l'unité arithmétique décrite ci-dessus. Les signaux générés par le séquenceur sont au nombre de 8 (6 pour l'unité arithmétique + 2 pour les signaux "wait" des mémoires).

A chaque période d'échantillonnage, 20 lectures en ROM, 20 lectures + 1 écriture en mémoire vive, et 20 multiplications sont effectuées.

La consommation du filtre évaluée sur une période d'échantillonnage vaut alors:

$$E_{éch.} \approx 20 \cdot (E_{mém\ 20\cdot20} + E_{ROM\ 20\cdot20} + 6 \cdot E_{ROM\ 8\cdot6} + E_{mult\ 20\cdot20}) \approx 20 \cdot (593 + 98 + 188 + 4470)\ pJ = 103\ nJ. \quad \text{(EQ 5)}$$

Ou, si l'on exprime la puissance consommée en fonction de la fréquence d'échantillonnage:

$$P/f_{éch} \approx 10^{-4} \text{ W/kHz} \qquad \text{(EQ 6)}$$

ou, avec Vcc = 5V: $$\mathbf{I/f_{éch} \approx 20\ \mu A/kHz} \qquad \text{(EQ 7)}$$

Le temps de calcul correspondant à un temps de cycle de 50 ns est:

$$\mathbf{Tc = 20 \cdot 6 \cdot 50\ ns = 6\ \mu s} \qquad \text{(EQ 8)}$$

ce qui équivaut à une fréquence d'échantillonnage maximale de 167 kHz.

Les surfaces occupées en technologie CMN12 sont:

mémoire vive:	$0.52 \cdot 0.36\ mm^2$
mémoire morte:	$0.21 \cdot 0.27\ mm^2$
séquenceur:	$0.10 \cdot 0.20\ mm^2$
multiplicateur:	$0.76 \cdot 0.61\ mm^2$

Ces blocs peuvent être assemblés selon le plan directeur de la Figure 4:

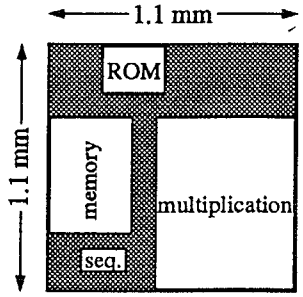

FIGURE 4. Plan directeur du filtre RIF d'ordre 20, 20 bits (sans les plages de contact)

En première approximation, les temps de propagation de la logique CMOS varient en fonction de la tension d'alimentation avec le facteur $Vdd/(Vdd-Vt)^2$ [Pigu92]. Pour Vt = 1 V et une fréquence d'échantillonnage de 20 kHz, on peut réduire la tension d'alimentation à 2 V. Ceci entraîne une réduction de la consommation d'un facteur supérieur à 6 en supposant que la composante de la puissance dissipée due au courant direct soit négligeable.

6. Conclusions

Les résultats obtenus sont intéressants tant au niveau de la consommation que de la vitesse et de la surface. Il serait évidemment utile de comparer ces prévisions avec un circuit réel.

Le fait que tous les blocs constitutifs soient paramétrables permet de les utiliser pour un grand nombre d'applications.

Cette architecture peut également convenir à des problèmes de filtrage adaptatif.

L'utilisation de mémoires séquentielles donne des solutions compactes et le séquencement s'en trouve simplifié.

L'unité arithmétique a un débit très élevé pour une surface raisonnable.

La flexibilité, tout en étant bien moindre que celle des processeurs de traitement du signal du commerce, est suffisante pour répondre aux besoins d'un grand nombre d'applications.

Références

[Chan92] A. P. Chandrakasan, S. Sheng, and R. Brodersen, "Low-Power CMOS Digital Design" , IEEE J. of Solid-State Circuits, Vol. SC-27, no. 4, april 1992, pp. 473-484.

[Sam90] H. Sam & A. Gupta, "A Generalized Multibit Recoding of Two's Complement Binary Numbers and Its Proof with Application in Multiplier Implementations" , IEEE Trans. on Computers, Vol. C-39, no. 8, Aug. 1990, pp. 1006-1015.

[Lee89] E. A. Lee, Wai-Hung Ho, E. E. Goei, J. C. Bier, S. Bhattacharyya, "Gabriel: A Design Environment for DSP" , IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-37, no. 11, Nov. 1989, pp. 1751-1762.

[Pigu87] C. Piguet, E. Dijkstra, A. Theys, M. Stauffer, J-F. Perotto, "A Design Methodology of Micro-programmed Controllers for Custom IC's" , Proc. EUROMICRO'87, Portsmouth, UK, Sept. 1987, pp. 463-470.

[Pigu92] C. Piguet, V. von Kaenel, J-M. Masgonty, J-F. Perotto, R. Klootsema, "Basic Design Techniques for both Low-Power and High-Speed ASIC's" , Proc. EUROASIC ' 92, Paris, France, 1992, pp. 220-225.

[Pow91] S. R. Powell, P. M. Chau, "Estimating Power Dissipation of VLSI Signal Processing Chips: The PFA Technique" , VLSI Signal Processing IV, S. Moscovitz, K. Yao, R. Jain, Eds; IEEE Press, New York, USA, 1991, pp. 250-259.

[They87] A. Theys, E. Dijkstra, C. Piguet, "Discrete Transforms on a Chip: Design and VLSI Implementation" , Proc. Int'l Symp. on Applied Control, Filtering, and Signal Processing' 87, IASTED' 87, Geneva, CH, June 1987, pp. 241-245.

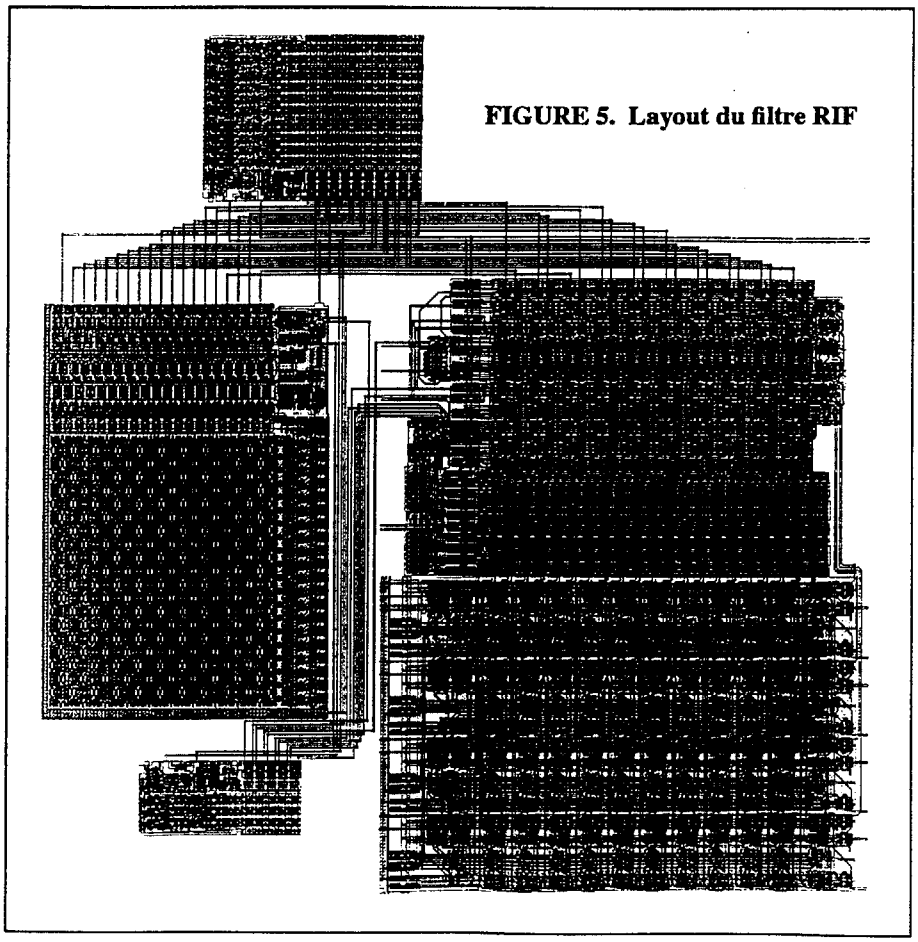

FIGURE 5. Layout du filtre RIF