IMPLEMENTATIONS VLSI ANALOGIQUES DE RESEAUX DE NEURONES

Michel Verleysen
Chercheur au laboratoire de microélectronique de l'Université Catholique de Louvain (Louvain-la-Neuve, Belgique)
Paul Jespers
Professeur au laboratoire de microélectronique de l'Université Catholique de Louvain (Louvain-la-Neuve, Belgique)

RESUME

Les implémentations analogiques de réseaux de neurones artificiels offrent des perspectives intéressantes en ce qui concerne leur densité d'intégration et leur vitesse de fonctionnement. Ce papier présente une réalisation analogique de réseau d'Hopfield basée sur la sommation de courants synaptiques, et montre comment des algorithmes d'apprentissage appropriés peuvent pallier aux contraintes des réseaux VLSI.

ABSTRACT

Analog implementations of artificial neural networks show interesting properties about their density of integration and their speed. This paper presents an analog realization of a Hopfield's network based on synaptic current summing, and shows how dedicated learning algorithms can cope with the restrictions of VLSI networks.

published in "Journées d'Electronique", Presses Polytechniques Romandes, 1989, pp. 279-289

IMPLEMENTATIONS VLSI ANALOGIQUES DE RESEAUX DE NEURONES

Michel Verleysen
Chercheur au laboratoire de microélectronique de l'Université Catholique de Louvain (Louvain-la-Neuve, Belgique)
Paul Jespers
Professeur au laboratoire de microélectronique de l'Université Catholique de Louvain (Louvain-la-Neuve, Belgique)

1. INTRODUCTION

Si les machines informatiques actuelles offrent des puissances de calcul dépassant de loin tout ce qu'un être humain peut imaginer, il est des domaines où l'ordinateur ne peut rivaliser avec un enfant de 5 ans. La perception, la reconnaissance et l'optimisation sont autant de problèmes qui sont du ressort du cerveau humain: n'importe qui est capable de reconnaître des visages, mais c'est là une tâche quasiment impossible pour un ordinateur classique.

La puissance d'un cerveau humain réside dans sa complexité; il contient en effet plusieurs centaines de milliards de processeurs élémentaires, appelés neurones, et chacun de ceux-ci est en moyenne relié à 10000 autres par l'intermédiaire de synapses [1]. Leur vitesse de réponse est par contre très faible: environ une milliseconde leur est nécessaire pour changer d'état, sans compter les (longs) temps de propagation des signaux à travers les synapses. Néanmoins, un problème complexe de vision peut être résolu par le cerveau en moins de 500 millisecondes; le nombre de cycles effectués est donc très réduit. Afin de résoudre certains problèmes où le cerveau est visiblement plus adapté qu'un ordinateur classique (perception, optimisation, classification,...), des dispositifs artificiels ont été mis au point, où le nombre de processeurs et leur connectivité sont les facteurs les plus importants: ce sont les "réseaux de neurones".

Différents types de réseaux de neurones peuvent être envisagés: réseau d'Hopfield [2], perceptron multi-couches, réseau localement interconnecté,... Afin d'atteindre une fonctionnalité et des performances intéressantes, deux objectifs sont à atteindre quel que soit le type de réseau utilisé: le nombre de neurones qui peuvent être connectés doit être important, et leur vitesse de

fonctionnement élevée, afin de concurrencer avantageusement des méthodes plus classiques. Le réseau d'Hopfield, complètement interconnecté, sera étudié ici car sa régularité se prête parfaitement bien à une réalisation VLSI; les techniques envisagées pourront néanmoins être adaptées simplement à d'autres types de réseaux, par exemple multi-couches, en modifiant seulement la façon d'interconnecter les cellules.

Le modèle de Hopfield consiste en un réseau de N neurones pouvant tous être connectés deux par deux. Chaque neurone contient une non-linéarité et la valeur de sa sortie est 1 ou -1 suivant que son activation dépasse ou non un certain seuil. Le réseau est bouclé sur lui-même, et la sortie de chaque neurone rétroagit donc à travers une des connexions du réseau sur l'entrée de tous les autres neurones. Le fonctionnement d'un tel réseau est simple; premièrement, les valeurs des poids des connexions sont programmées suivant une règle d'apprentissage. Ensuite, chaque neurone est initialisé à une valeur uniquement d'après l'entrée qui y est raccordée. Enfin, de nouvelles valeurs sont calculées pour chaque neurone en évaluant son activation à travers le réseau de rétroaction et en appliquant la non-linéarité. Ce dernier processus est répété jusqu'à ce qu'un état stable du réseau soit atteint.

Ce modèle peut être utilisé comme mémoire associative: on introduit dans le système un vecteur à reconnaître, on laisse converger le réseau et lorsqu'un état stable est atteint, le résultat désiré est déterminé par la sortie des neurones. On peut implémenter aussi bien une mémoire auto-associative, où il faut reconnaître un vecteur d'entrée parmi les différents vecteurs mémorisés, qu'une mémoire hétéro-associative, où un vecteur mémorisé doit être associé à chaque vecteur d'entrée, mais peut en être complètement différent (y compris sa dimension).

2. IMPLEMENTATIONS VLSI A POIDS FIXES

L'idée qui vient la première à l'esprit lorsqu'on veut réaliser un réseau de neurones complètement interconnecté est de réaliser les synapses par des résistances. De cette façon, le poids de la connexion entre deux neurones sera déterminé par la valeur de la résistance. De plus, l'absence de tout circuit logique permet de réaliser facilement un circuit entièrement asynchrone. Le schéma d'un tel circuit est donné figure 1 [3].

On voit tout de suite que ce circuit souffre de plusieurs inconvénients. Tout d'abord, seules des connexions positives sont réalisables. Pour avoir des synapses négatives, il faudrait avoir deux sorties à chaque neurone: une inverseuse et une non-inverseuse. Suivant que la connexion doit être positive ou négative, on raccordera alors l'une ou l'autre de ces sorties. Deuxièmement, les valeurs des résistances doivent être fixées à l'avance (avant la réalisation du circuit). Il est en effet impossible d'intégrer des résistances programmables!

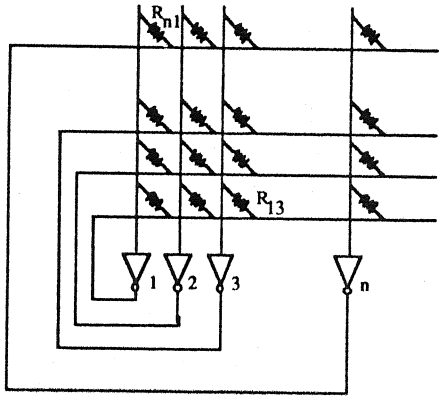

Fig. 1. Réseau à résistances

Troisièmement, si on veut garder une structure régulière au réseau, il ne faut implémenter qu'une seule valeur de résistance; différentes valeurs signifieraient en effet différentes surfaces de silicium, ce qui détruirait la régularité du réseau. La programmation de celui-ci devra alors se réduire à déterminer la présence ou l'absence de cette résistance. Enfin, et ce n'est pas là le plus petit inconvénient, des résistances sont très difficiles à réaliser par un processus technologique classique de circuits intégrés. Il faut souvent des étapes supplémentaires, ce qui augmente le coût de fabrication de manière spectaculaire.

3. IMPLEMENTATIONS DIGITALES

Afin de réaliser des réseaux plus souples, où les poids synaptiques peuvent être programmés, des processeurs digitaux peuvent être employés pour réaliser les différentes fonctions nécessaires (produits, sommes pondérées et non-linéarités). L'avantage des réalisations digitales est bien entendu la précision qui peut être facilement obtenue sur toutes les valeurs mises en jeu. Néanmoins, les processeurs digitaux sont relativement grands, et il est donc difficile d'en intégrer un grand nombre sur un seul circuit. La solution à ce problème est alors d'introduire des multiplexages, afin de remplaces une partie de la complexité spatiale par une complexité temporelle. L'inconvénient est une perte de vitesse de fonctionnement, et les implémentations digitales seront donc surtout utilisées pour de problèmes où la rapidité n'a que peu d'importance.

4. IMPLEMENTATIONS ANALOGIQUES

Afin d'obtenir simultanément un grand nombre de cellules (neurones et synapses) et une vitesse de fonctionnement élevée, une implémentation analogique doit donc être envisagée. L'architecture proposée ici consiste en un réseau complètement interconnecté (N neurones et N^2 synapses), où chaque poids synaptique est restreint aux valeurs +1, 0 et -1; une justification à cette restriction sera présenté au paragraphe suivant.

Lorsqu'on implémente un réseau de neurones par une technique analogique, le problème le plus important est de pouvoir connecter un grand nombre de synapses au même neurone, et ceci sans risque d'erreur. Une solution possible est celle présentée par Graf [4]. Les synapses sont des sources de courant programmables; suivant la valeur du neurone qui y est connecté et du poids synaptique, une des deux sources de courant est enclenchée (ou aucune des deux si le poids synaptique est nul). Dans chaque synapse, un courant peut donc être soit injecté soit retiré de la ligne auquel elle est connectée (fig 2). Tous ces courants synaptiques sont additionnés (et soustraits), et la fonction du neurone sera de déterminer si le courant total à son entrée est positif ou négatif (en admettant que chaque neurone se comporte comme un seuil centré à l'origine).

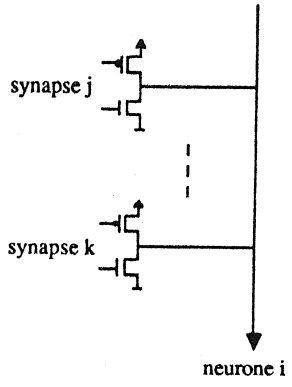

Fig. 2. Synapses avec sources P et N

Le désavantage de cette architecture est qu'il est impossible d'obtenir des courants excitateurs et inhibiteurs exactement égaux; en effet, pour fabriquer des transistors p et n, différents processus technologiques sont mis en oeuvre, et le rapport des mobilités entre les charges p et n et peut être connu que très approximativement avant la fabrication du circuit. Il est donc impossible de compenser exactement cette différence grâce à la taille des transistors, et il y

aura toujours un léger non-appariement entre les deux types de courants. Puisque la fonction de chaque neurone est de déterminer le signe de la somme pondérée des valeurs des autres neurones, l'erreur dûe à la différence entre les courants excitateurs et inhibiteurs s'accroît avec le nombre de synapses qui sont connectées. Pour qu'un neurone soit capable de détecter le signe de son entrée, il faut que cette erreur soit inférieure à un courant synaptique. Celui-ci devra donc être supérieur à n fois l'erreur maximum sur une synapse, où n est le nombre de synapses raccordées au même neurone. Ceci limite considérablement la taille d'un réseau qui peut être implémenté sur un circuit.

Afin de pallier à cet inconvénient, toutes les synapses excitatrices peuvent être rassemblées sur une ligne et toutes les synapses inhibitrices sur une autre (figure 3). La structure employée étant absolument symétrique, le même type de sources de courant (P ou N) pourra alors être utilisé pour les deux sortes de synapses. Les courants excitateurs et inhibiteurs étant sommés sur deux lignes différentes, la fonction du neurone sera alors de comparer ces deux sommes, et de faire basculer sa sortie à 1 si les courants excitateurs sont plus nombreux que les courants inhibiteurs, ou à -1 dans le cas contraire.

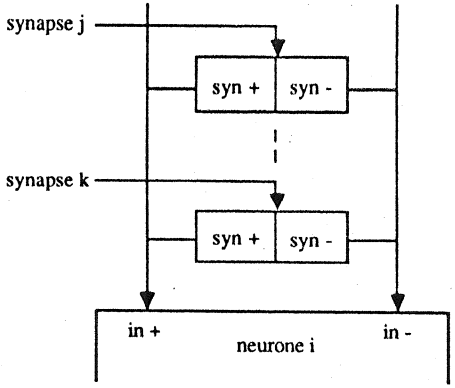

Fig. 3. Synapse symétrique

Chaque synapse est une double source de courant programmable (figure 4). Trois valeurs de connexions sont permises dans chaque synapse. Si "mem1"=0, aucune connexion n'est réalisée entre les neurones i et j. Si "mem1"=1, le signe de la connexion est déterminé par le produit de la sortie du neurone j par la valeur de "mem2". Si ce produit vaut 1, la synapse est excitatrice et un courant est retiré de la ligne "in+" du neurone par l'intermédiaire de T0 et T1. Si le produit vaut -1, la synapse est inhibitrice et le même courant est retiré de la ligne "in-" par l'intermédiaire de T0 et T2.

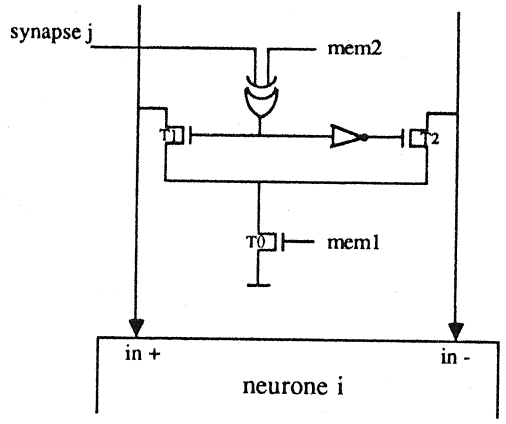

Fig. 4. Schéma d'une synapse

On voit immédiatement sur ce schéma que les courants dérivés par T1 ou T2 seront les mêmes, pour autant que les deux transistors soient identiques. Si le processus technologique peut admettre de très petites variations de dimensions réelles entre deux transistors qui ont été dessinés de façon identique, il est évident que cette erreur sera beaucoup plus petite que si l'on avait affaire à deux transistors de type différents (P et N). Cette solution permet donc d'éliminer ou en tous cas de rendre insignifiant le problème du non-appariement entre les sources de courant.

Tous les transistors de la synapse peuvent être choisis minimaux. Seul T0 est un transistor long (W/L=0.1) afin de diminuer le courant qui y passe. Une autre solution aurait été de diminuer la tension sur sa grille, mais il aurait alors fallu soit stocker une valeur analogique, soit diviser la tension stockée avant de l'introduire sur la grille. Dans les deux cas, ceci aurait nécessité l'emploi de transistors supplémentaires et une perte de surface en aurait résulté.

La fonction du neurone est de comparer les courants synaptiques totaux sur les lignes "in+" et "in-". Si le premier est plus grand que le second, la sortie "out" devra être saturée vers le haut. Si c'est le contraire, elle devra être saturée vers le bas. Le fonctionnement du neurone proposé est très simple (figure 5).

Les deux courants sont tout d'abord convertis en tensions par le réflecteur formé de T3 et T4. Ces deux tensions V+ et V- sont ensuite elles-mêmes comparées par l'amplificateur différentiel formé de T5 à T9. En ajustant correctement les dimensions des transistors, la tension "out" peut être saturée dans un sens ou dans l'autre à partir d'une différence de tension de 0.5V entre V+ et V-. Des tests ont montré que cette différence peut encore être atteinte si

environ 250 synapses actives sont raccordées sur chaque ligne, et avec une différence de seulement un courant synaptique entre les lignes positives et négatives.

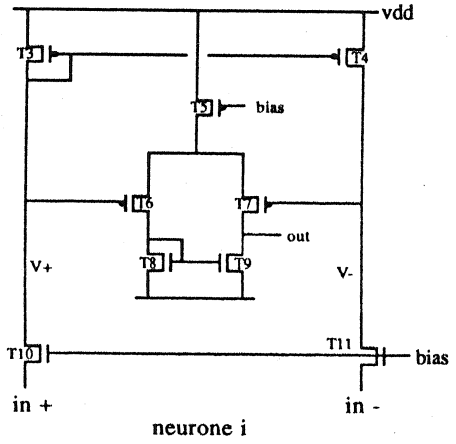

Fig. 5. Schéma d'un neurone

Un seul problème reste à résoudre: la chute de tension sur T3 et T4 augmentant avec le courant qui traverse ces deux transistors, les tensions V+ et V- vont toutes les deux diminuer avec le nombre de synapses qui sont actives. Comme l'amplificateur est différentiel, une chute de tension simultanée sur les deux entrées n'aura qu'une influence minime. Par contre, les tension V+ et V- sont aussi les tensions de drain des transistors T1 et T2. Une variation de ces tensions aura donc un effet indésirable sur les courants dérivés par ces deux transistors. Pour remédier à ce problème, nous pouvons ajouter un transistor en source commune sur chacune des deux lignes, afin d'agir en transformateur d'impédance (transistors T10 et T11). Une boucle de réaction peut également être ajoutée entre la grille et la source de ce transistor, afin d'obtenir une meilleure régulation de la tension.

Un circuit de test a été réalisé suivant cette architecture. Il contient 14 neurones et 196 synapses, pour une surface totale de 9 mm^2, y compris les plots. La technologie employée est 3µm double métal. Le circuit contient, outre les neurones et les synapses, un décodeur permettant la programmation des points mémoires et 28 plots d'entrée-sortie utilisés soit pour la programmation, soit lors du fonctionnement du réseau comme entrées et sorties des neurones

5. ALGORITHME D'APPRENTISSAGE

Au paragraphe précédant, un circuit VLSI pour réseaux d'Hopfield a été décrit. Une restriction de ce circuit était la dynamique des coefficients synaptiques, qui ne pouvaient prendre que les valeurs +1, 0 et -1.

Pour programmer ce type de réseau en tant que mémoire associative, différents algorithmes d'apprentissage ont été proposés. Le plus connu est incontestablement la règle de Hebb [1], qui trouve son fondement dans la biologie du cerveau humain. Si cette règle peut donner de bons résultats lorsque la quantité d'informations à stocker est faible, il est habituellement admis que la capacité limite de cette règle est environ 0.15N vecteurs de N bits, où N est le nombre de neurones dans le réseau. D'autres règles d'apprentissage ont été proposées, parmi lesquelles la règle de la projection (ou règle pseudo-inverse), qui donne de meilleurs résultats que la règle de Hebb. Néanmoins, pour pouvoir utiliser ce type d'algorithme d'apprentissage dans le réseau décrit plus haut, il faudrait n'avoir que trois valeurs possibles pour chaque connexion synaptique. Un moyen simple est de tronquer tous les coefficients, à +1 si ils étaient positifs, et à -1 si ils étaient négatifs. Cette façon brutale de procéder diminue bien entendu les performances du réseau, et donc sa capacité de stockage. Pour la règle de Hebb, bien que les performances varient en fonction du nombre de neurones, la capacité est en moyenne diminuée à environ 0.1N vecteurs de N bits. Par contre, la règle de la projection nécessitant au départ une large dynamique des coefficients, les résultats deviennent très médiocres si cette dynamique est réduite. La nécessité de programmer les réseaux VLSI par des algorithmes appropriés paraît alors évidente.

Dans un réseau d'Hopfield, on peut définir une fonction d'énergie [2], et il peut être démontré que les points fixes du réseau, c'est-à-dire les vecteurs stables, correspondent aux minimums de cette fonction. Afin d'imposer au système ses points fixes, une possibilité consiste à maximiser la stabilité de chaque vecteur-cible; cela revient en fait à donner une forme à la fonction d'énergie, de telle sorte que les vecteurs-cible se trouvent au centre d'un puits le plus profond possible. Ceci peut être formulé mathématiquement en prenant les notations suivantes. Soit:

N : le nombre de neurones dans le réseau
p : le nombre de vecteurs de N bits à mémoriser dans le réseau
T_{rl} : le poids de la connexion entre les neurones r et 1
V_i : la valeur du neurone i (+1 ou -1) à un instant donné
V_{ij} : la valeur du bit i provenant du vecteur j à mémoriser
S_{1k} : la valeur d'entrée du premier neurone lorsque la sortie du réseau correspond au vecteur k à mémoriser

$1 \leq i \leq N$
$1 \leq r \leq N$
$1 \leq j \leq p$
$1 \leq k \leq p$

Nous avons évidemment:

$$S_{1k} = \sum_r T_{rl} . V_{rk} \qquad (5.1)$$

Pour trouver les coefficients d'une colonne du réseau, c'est-à-dire de toutes les synapses raccordées au même neurone, le principe sera d'imposer une différence la plus grande possible entre l'entrée du neurone et son seuil, et ce lorsque le réseau se trouve dans chacun des états stables correspondants aux k vecteurs à mémoriser. Afin d'assurer le signe correct à S_{1k} (le calcul est effectué ici pour la première colonne, mais est identique pour les autres), la quantité à maximiser est $Z_{1k} = S_{1k} . V_{1k}$. Comme ceci est à résoudre simultanément pour toutes les valeurs de k, l'équation à résoudre sera:

$$\text{maximiser } M \quad \text{où } M = \min(Z_{1k}) \text{ pour toutes les valeurs de k.} \qquad (5.2)$$

Pour éviter des solutions non bornées, chaque coefficient synaptique T_{rl} est borné par les inégalités:

$$-1 \leq T_{rl} \leq 1 \qquad (5.3)$$

Un algorithme du Simplex [5] peut facilement résoudre le système des équations 5.2 et 5.3.

La méthode de résolution employée impose qu'au moins N-p valeurs parmi les n coefficients synaptiques prendront une des valeurs des bornes (+1 ou -1). Comme N est généralement nettement plus grand que p, on peut constater par des simulations qu'on ne perd presque rien en termes de performances du circuit si on tronque les p coefficients restants aux valeurs +1, 0 ou -1. L'algorithme présenté peut donc facilement être utilisé dans un réseau de neurones réalisés par un circuit VLSI. De plus, des simulations ont montré que les performances de cette règle d'apprentissage sont nettement meilleures que celles de la règle de Hebb: la capacité de stockage est sensiblement la même, mais une tolérance beaucoup plus large peut être acceptée dans les vecteurs d'entrée (la distance de Hamming, ou nombre de bits différents, entre un vecteur d'entrée et le plus proche des vecteurs enregistrés peut être nettement plus grande, ce qui est parfaitement compréhensible en remarquant que si la stabilité d'un vecteur est maximisée comme dans l'algorithme décrit plus haut, le nombre de synapses produisant un résultat incorrect sans changer la sortie du neurone auxquels ils sont raccordés est également maximisé). De plus, les patterns instables (qui ne convergent jamais), sont pratiquement éliminés. Des résultats quantitatifs sont donnés dans [6].

6. CONCLUSION

Les recherches actuelles menées dans le domaine des réseaux de neurones montrent que ceux-ci peuvent ouvrir des voies prometteuses dans des domaines tels que la classification, l'optimisation, ou la reconnaissance. Deux qualités doivent être présentes dans les réalisations VLSI de tels réseaux afin d'obtenir des dispositifs dont les performances sont meilleures que les méthodes classiques: la densité d'intégration doit être importante afin d'intégrer des réseaux dont la taille est suffisante pour des problèmes réels, et la vitesse de fonctionnement doit être suffisamment élevée pour que l'intégration des réseaux ait un intérêt par rapport à leur simulation. Un réseau analogique tel que celui présenté ici offre ces qualités. Une meilleure densité d'intégration peut encore être obtenue en utilisant des points mémoires dynamiques au lieu des points mémoires statiques utilisés dans le circuit présenté.

Un algorithme d'apprentissage a également été décrit, afin de prouver que les restrictions imposées par les réalisations VLSI de réseaux (par exemple la dynamique réduite des valeurs utilisées) peuvent être compensées par un apprentissage adéquat. Un domaine intéressant d'investigation serait de modifier cet algorithme d'apprentissage afin que les coefficients soient calculés sur le circuit lui-même, afin de tenir compte d'éventuelles imperfections de celui-ci.

Le réseau VLSI présenté plus haut, et programmé grâce à une règle d'apprentissage adaptée, peut être utilisé comme mémoire associative par exemple dans des problèmes de reconnaissance de formes. L'intérêt porté aux réseaux de neurones dans ce type de problèmes sera pleinement justifié lorsque les techniques d'intégration permettront la réalisation de réseaux comportant plusieurs centaines de neurones.

7. BIBLIOGRAPHIE

[1] HEBB D.,*The organization of behavior*,Wiley,New York 1949.

[2] HOPFIELD J.J.,«Neural networks and physical systems with emergent collective computational abilities»,*Proc.Natl.Acad.Sci.USA*,vol.79,pp.2554-2558,1982.

[3] HOWARD R. et al.,«An associative memory based on an electronic neural network architecture»,*IEEE Transactions on Electron Devices*,vol.ED-34,n°7,pp.1553-1556, 1987.

[4] GRAF H. & DE VEGVAR P.,«A CMOS implementation of a neural network model»,*Proceedings of the 1987 Stanford Conference on Advanced Research in VLSI*, MIT Press 1987.

[5] LASDON L., *Optimization theory for large systems*,Collier-Mac Millan,London 1970.

[6] VERLEYSEN M., SIRLETTI B., VANDEMEULEBROECKE A., JESPERS P., «Neural networks for high-storage content-addressable memory: VLSI circuit and learning algorithm»,*IEEE Journal of Solid-State Circuits*,vol.24,n°3,pp.562-569,1989.