Contribution à l'étude de l'impact des nanotechnologies sur les Architectures : Apprentissage d'inspiration neuronale de fonctions logiques pour circuits programmables

Michel He

N° D'ORDRE 9270

THÈSE DE DOCTORAT

SPECIALITE : PHYSIQUE

Ecole Doctorale « Sciences et Technologies de l'Information des Télécommunications et des Systèmes »

Présentée par :

Michel Haiyu He

Sujet :

Contribution à l'étude de l'impact des nanotechnologies sur les Architectures : Apprentissage d'inspiration neuronale de fonctions logiques pour circuits programmables

Soutenue le 17 Décembre 2008 devant les membres du jury :

M.	V. BEIU	Professeur, United Arab Emirates University
M.	E. BELHAIRE	Ingénieur, Thales Optronique
M.	P. GARDA	Professeur, Université Pierre et Marie Curie
M.	C. GAMRAT	Ingénieur, CEA-LIST
M.	B. GRANADO	Professeur, Universités Cergy-Pontoise
MME.	C. MANEUX	MdC, Université Bordeaux 1
M.	J .O KLEIN	MdC, IUT de Cachan, Université Paris XI
M.	R. REYNAUD	Professeur, IUT d'Orsay, Université Paris XI

Remerciements

Les travaux de thèse que j'ai réalisés se sont déroulés au laboratoire Institut d'Electronique Fondamentale de l'Université Paris Sud XI Orsay, entre 2005 et 2008. Durant cette période, les efforts ont été concentrés dans l'évaluation de nouvelles technologies de semi-conducteur de l'échelle nanométrique, plus particulièrement sur l'élément de calcul logique par réseau de neurones.

D'emblée, je voudrais remercier mon directeur de thèse, et mon directeur de l'équipe d'accueil, pour avoir offert un cadre formidable pour les études doctorales. Sans oublier évidemment la direction de l'école doctorale et du laboratoire d'accueil j'ai apprécié à chaque moment l'attention d'écoute de chacun et l'assistance apportée pour faire face à toutes les situations. L'université a été un facteur déterminant pour mener la quête scientifique. La recherche est nécessairement importante pour la croyance personnelle. J'aimerais remercier les personnes qui m'ont donnés les moyens pour approfondir mes connaissances. Je remercie également les services bibliothécaires qui m'ont aidé à trouver les œuvres scientifiques inestimables. Je témoigne tant de gratitude envers tous les membres du jury, mes co-encadrants, mes collègues et mon ancien directeur de master pour tout ce qui a été les plus précieux.

J'adresse l'amitié envers toutes les personnes que j'ai croisées aux conférences, ainsi qu'aux séminaires et colloques. Enfin je remercie ma famille, ma fiancée, mes amis.

Ce serait égoïste de ma part de dédier ce manuscrit uniquement à ma sphère. Alors je dirais…

A tous les chercheurs

Avant toute fin, il y eut un début, ensuite un chemin d'évolution.

Sommaire

INTRODUCTION

Les découvertes et inventions récentes dans le domaine des nanotechnologies permettent d'ouvrir continuellement de nouveaux champs d'applications.

La question centrale qui se pose dans ce travail de thèse est la suivante : Comment pourrait-on saisir l'importance des nanocomposants pour imaginer les changements avérés dans les architectures de calcul ? A priori, on sait que les systèmes sur puce rencontreront une limite à la réduction de la taille du transistor. Pour cela une synthèse et une analyse critique des différentes alternatives doivent être faite. En effet, les procédés de fabrication commencent à atteindre les plafonds de leur performance. Par conséquent, les différentes pistes doivent être explorées afin de maintenir une perspective qui fait face aux énormes défis technologiques.

Dans le contexte de recherche et développement du semi-conducteur, on a vu par le passé quelques étapes importantes. Parmi celles-ci, on peut citer d'abord le remplacement de substrat de germanium par le silicium pour une raison de coût. Ensuite, on peut citer le passage du calcul analogique au calcul numérique. Le calcul numérique par rapport à l'analogique permet de réduire la sensibilité aux bruits de l'environnement. Puis, on peut constater l'évolution du transistor, de bipolaire au CMOS [Chen06] pour un meilleur ratio performance versus puissance. Enfin, on peut retracer l'évolution de la technologie CMOS, notamment les changements de matériaux de grille et de diélectrique, pour diminuer l'effet des courants de fuite. Quant aux interconnexions métalliques, on note l'évolution de l'aluminium vers le cuivre, pour diminuer la résistance… Ces nombreux changements sont à l'origine des prédictions quelque fois erronées [Denn74]. Néanmoins on peut dire que la logique Booléenne s'adapte bien aux architectures basées sur le CMOS, et ce n'est donc pas un hasard qu'aujourd'hui les portes logiques de ce genre constituent le processeur de système sur puce. En sommes, cette rétrospective montre que

l’évolution de la technologie suit un chemin rationnel, c'est-à-dire la recherche de l’optimum entre le coût, la performance, et la consommation.

Ce processus d’évolution par sauts continuels est relativement rapide à l’échelle de la civilisation. La fin supposée de la loi de Moore [Moor65] est toutefois proche, disait-on vers l’an 2020. Alors que va-t il se passer, je me force d’établir trois options :

Option 1 : Pas de développement futur en semi-conducteur. Après avoir atteint le dernier nœud, on disposera d’un transistor optimisé poussé à l’ultime, ou encore de plusieurs types de transistors améliorés et spécifiques. On s’en servira pleinement car un optimum de la technologie sera atteint.

Option 2 : L’inversion de tendance. Le dernier nœud existe mais la majorité de production de système sur puce ne se fera pas sur celui-ci, mais sur une génération de technologie moins coûteuse. Seule une minorité de puces seront fabriquées à cet ordre pour faire des calculs intensifs. On peut citer la cause de l’élévation de coût de fabrication du transistor ultime.

Option 3 : Nouvelle ère des architectures. L’industrie du semi-conducteur va s’orienter vers l’emploi d’autres éléments non utilisés actuellement, et inventer de nouvelles chaînes de conception pour de nouvelles architectures. Elles seront régies par de nouvelles contraintes, d’où la recherche des alternatives vis-à-vis de l’architecture des portes logiques Booléenne.

Ces options ne seront pas mutuellement exclusives, ni pour les choix stratégiques, ni dans le temps. Surtout, des causes comme la dispersion de caractéristiques et les défauts de plus en plus présents obligent à revoir les méthodes de conceptions actuelles. De cela vient la nécessité d’imaginer une conception d’architecture innovante, et tolérante aux désavantages cités. Notamment par la recherche de nouveaux paradigmes de calcul logique, on pourra accéder à un nouveau point de vue sur les systèmes de traitement d’information et calculs. L’investigation doit montrer la relation entre l’objectif et les moyens les plus efficaces pour réaliser un système, qui puisse aussi s’incorporer dans la tendance "More than

Moore", c'est-à-dire l'extension de fonctionnalités. Quoiqu'il en soit, l'industrie du semi-conducteur recherchera à se réinventer pour échapper à la crise. En fait, depuis un certain temps l'apparition des nanocomposants dans l'industrie du semi-conducteur suscite à la fois l'euphorie et le désintéressement. Par ailleurs, on remarque une accumulation de nouvelles technologies, parfois sans débouché. Pourra t-on alors voir un jour une technologie concurrente menacer les positions du CMOS dans les systèmes sur puces, et avec quels impacts ?

La première partie de mon manuscrit s'intéresse aux problèmatiques de la technologie du semi-conducteur traditionnelle. Ensuite dans la deuxième partie je vais m'intérésser aux propriétés des nanocomposants. Ils se distinguent du CMOS classique selon plusieurs critères. D'emblée, j'ai dressé les descriptions des nanocomposants, puis discuté des critères de choix. Les données sont un indicateur de performance de ces derniers, en termes du coût additionnel, de la densité d'intégration, de la vitesse de circuit, de la consommation, et de la puissance de calcul. Dans cet état de l'art on peut dire que la motivation réelle est de tirer partie des opportunités qu'offrent les nanocomposants, plutôt que de les voir comme un remplaçant du CMOS. Cette analyse est une façon de montrer les antipodes de l'optimisme portée par un remplacement intégral du CMOS. L'idée serait de trouver une combinaison adaptée pour l'intégration hétérogène des technologies émergentes.

Ayant une connaissance globale des architectures, j'ai choisi de développer plus amplement les réseaux de neurones en seconde partie. En effet, des fonctions logiques peuvent être émulées par les réseaux de neurones. C'est assurément dans le cadre de circuits reconfigurables que j'ai essayé de mettre en valeur cette architecture, l'objectif est différent des réalisations de simulations de réseaux de neurones par les circuits numériques [Zhu03] dans les années précédentes. Pour cela j'ai étudié les différents algorithmes de l'apprentissage des réseaux de neurones. Je suis amené à trouver une modélisation de nanocomposants pour implémenter la synapse utilisée dans les connexions de neurones. Cela m'a amené à dessiner une architecture pour le bloc de la logique associative, et son module d'apprentissage. Néanmoins, avant la simulation j'ai porté mes attentions sur les modèles des nanocomposants utilisés, ici les nanotubes de carbones semi-

conducteurs et la mémoire CBRAM. Ce type de mémoire est assimilé à la conductance multi-niveau programmable. Enfin je vais présenter la simulation de l’apprentissage neuronale de fonction logique.

Pour la dernière partie, la robustesse d’une architecture de réseaux de neurones est évaluée par simulation. Il est question d’étudier la redondance des réseaux, qui est inhérente à ce type d’architecture. Alors je vais déterminer et expliquer le taux de succès d’apprentissage pour différents niveaux de redondance en me basant sur le modèle de fautes prises en compte dans le réseau. Cette simulation fonctionnelle montre la possibilité de construire un circuit robuste grâce à l’apprentissage.

La conclusion revient sur le résumé de mes études, ma contribution à l'étude de l'impact des nanotechnologies sur les architectures, et l’apprentissage d’inspiration neuronale pour réaliser des circuits programmables.

CHAPITRE 1 : PROBLEMATIQUE

Dans les systèmes électroniques sur puce, on a observé la course à la miniaturisation. La surface de composant est réduite. Le nombre d'opérations réalisées par seconde augmente tout en réduisant la consommation en énergie par opération. On constate l'augmentation de volume de production, et en même temps le coût unitaire de la puce a diminué drastiquement. Cette tendance prédite par [Moor65] n'est pas sans limite. Ainsi, en 2007 sur les feuilles de route de l'ITRS (International Technology Roadmap for Semi-conducteur) [ITRS – Executive Summary], on prévoit la fin de la course en 2022 après avoir atteint le nœud de 11 nm (le nœud étant défini par la distance minimale séparant deux lignes Metal1 adjacentes), à ce moment la longueur minimale de la grille du transistor serait de 4 nm. Hormis l'aspect de la taille physique ultime, la dispersion de caractéristiques va devenir de plus en plus critique. Mais avant tout, les études sur l'évolution de l'industrie de semiconducteurs montrent qu'elle va se heurter à deux défis majeurs. Le premier défi est celui de la consommation. Le deuxième concerne les coûts de la fabrication.

Selon [Lloy00], les limites de l'ordinateur ultime peuvent être calculées à partir des équations quantiques. L'entropie régit la quantité d'informations que le système peut enregistrer, et la température régit le nombre d'opérations par bit par seconde. D'après [Lloy00], celui-ci pourra effectuer au total 10^{51} opérations par seconde sur 10^{31} bits. Dans ces conditions, il pourrait atteindre 10^{9} degrés Kelvin, il consommerait 4.10^{26} W, et dégagerait 10^{17} Joule en une nanoseconde. Mais, il n'existe aucune technologie pour l'alimenter, ni pour évacuer la chaleur. En considérant les projections actuelles du nœud final du CMOS [ITRS], les performances seront limitées par la puissance maximale que pourront dissiper ces circuits (entre 100 et 200 W par cm^2 d'après [ITRS]), donc loin de ces chiffres faramineux.

Par ailleurs, [Cavi05] soutient que le concept de barrière d'énergie reste universel pour tout type de composant et ne diffère que par les grandeurs physiques

mises en jeu. La conclusion de son analyse montre que le CMOS est la solution optimale à charge commutée pour le calcul logique en termes de densité, de vitesse et d'énergie. À plus long terme, une nouvelle approche pour le traitement de l'information peut être exploitée, l'auteur de [Cavi05] suggère qu'elle serait basée sur la fusion logique-mémoire universelle. En effet, la séparation entre la mémoire et la logique des circuits numériques nécessite le transfert fréquent des données entre les deux modules, ce qui augmente la consommation énergétique.

Pour autant, la loi de Moore risque de rencontrer une limite financière bien avant d'approcher les limites de la physique. En effet, l'explosion des coûts de fabrication de la microélectronique se traduit d'abord par la multiplication des coûts de recherche développement et de production. Selon les informations recueillies de la session parlementaire du Sénat en 2008 [Saun08], les coûts liés à la conception architecturale des circuits intégrés augmentent de 50 % pour chaque nouvelle génération technologique. On peut regarder l'évolution de ces dernières années, (tableau 1, selon les chiffres de Thales repris dans [Saun08]).

Finesse de la gravure, nœud technologique	Nombre de portes logique/circuit	Coût de développement du circuit	Part du coût des logiciels dans le développement
130 nm	9 M	9 M€	30%
90 nm	16 M	18 M€	50%
65 nm	30 M	46 M€	66%

Tableau 1. [Saun08] Comparaison de coût de développement du circuit en fonction de la finesse de gravure

Toujours d'après le rapport parlementaire [Saun08], « *Le coût des unités de production... Il a ainsi doublé depuis 2002, passant de 1,5 à 3 milliards de dollars.* » [Saun08] Les facteurs expliquant cette tendance sont d'origines technologiques. « *D'abord, la fabrication de circuits intégrés exige des environnements de production extrêmement propres pour éviter des contaminations fatales aux circuits. L'air des salles blanches est ainsi filtré et entièrement renouvelé 10 fois par minute. Il contient 100.000 à 1 million de fois moins de poussières que l'air extérieur »* [Saun08]. Pour le degré de propreté de la salle blanche, on évoque d'ors et déjà la classe 1 pour la fabrication du puce, dans la Fab 32 de *Intel*, Chandler, Arizona, USA, nœud technologique 45nm. On peut constater que les sites de fabrication sont sous

contrôle anti-sismique. Les installations nécessitent des stabilisateurs. *« Ensuite, la poursuite de la miniaturisation implique l'utilisation des machines de production précises, fiables, difficiles à mettre au point et à entretenir. Elles ne sont fabriquées qu'en petites séries »* [Saun08]. On peut également remarquer qu'un masque de gravure varie entre 100 kilos et 1 Méga dollars par unité. Les masques servant à la photolithographie doivent être changés périodiquement, et la fabrication d'un circuit intégré peut nécessiter jusqu'à 30 masques différents (cela dépend du nombre de niveaux). La machine à lithographie extrême ultra violet (EUV) coûte 60 millions de dollars. Si en 1965, le wafer mesurait 25 mm de diamètre, aujourd'hui elles font 300 mm de diamètre et le développement de 450 mm sera pour 2012 [Hutc06]. Le développement de la fabrication des wafers de diamètre 300 mm à lui seul absorberait près de 12 milliards de dollars [Hutc06]. En ce qui concerne le développement du 450 mm, les spécialistes tablent sur le chiffre de 100 milliards de dollars [Hutc06]. En outre, le recours à des matériaux spéciaux et à des solutions techniques complexes entraîne la multiplication des étapes de fabrication, nous sommes arrivés à plus de 700 par circuit, cela entraîne des répercussions sur le coût de production. Le passage à un nouveau noeud technologique environ tous les deux ans implique également le changement d'une partie des équipements au profit des machines plus modernes, plus chères. L'explosion du coût des unités de production s'explique enfin par la taille croissante des usines. Afin de baisser les coûts unitaires de production, les circuits intégrés sont fabriqués dans des volumes de plus en plus importants. A titre illustratif, la société TSMC dispose de deux usines gigantesques, capables de produire 135.000 wafers par mois. Chaque usine coûte *8 milliards de dollars* [Saun08].

« *Dans certains domaines, le cycle de vie des produits se raccourcit. Ainsi, les gammes de produits électroniques changent tous les six mois »* [Saun08]. Les fabricants de semiconducteurs sont donc soumis à des pressions très fortes en matière de la conception, le « time-to-market », l'augmentation rapide des volumes de production, car ce sont les facteurs prépondérants pour l'obtention du marché. La moindre marge par unité de produit doit être impérativement compensée par un fort volume de production et un rendement très élevé. Néanmoins les spécialistes estiment que *« ce marché serait en train d'atteindre sa maturité, ce qui engendrerait des taux de croissance plus faibles (entre 6 et 8 % actuellement contre plus de 15 %*

entre les années 70 et 90) » [Saun08]. Les entreprises de semiconducteurs continuent à consacrer entre 15 et 20 % de leur chiffre d'affaires à la recherche et développement, compte tenu du coût de développement d'une technologie et le coût de logiciels (on table sur 750 millions dollars pour le 45 nm, 1 milliard dollars pour le 32 nm), les efforts financiers sont importants. En conséquence, certaines entreprises ont renoncé à cette course, préférant concentrer leurs efforts et leurs investissements sur les services à forte valeur ajoutée pour des technologies moins agressives mais matures. Pour d'autres, les alliances ont été crées afin de partager les coûts du développement technologique.

« *Le raccourcissement des cycles de vie des produits exigeait une montée rapide des volumes de circuits intégrés fabriqués, ce qui nécessite une très bonne maîtrise des procédés de production* » [Saun08]. Les erreurs de fabrication sont pénalisantes (cf. figure 1). Afin de limiter l'explosion de leurs coûts, [Deho05], [Mori05], [Likh03] prônent le développement des puces universelles. Pour l'instant, on sait que des puces reconfigurables, qui peuvent être reprogrammées en « post-process », pourrait s'adapter à un certain nombre d'applications.

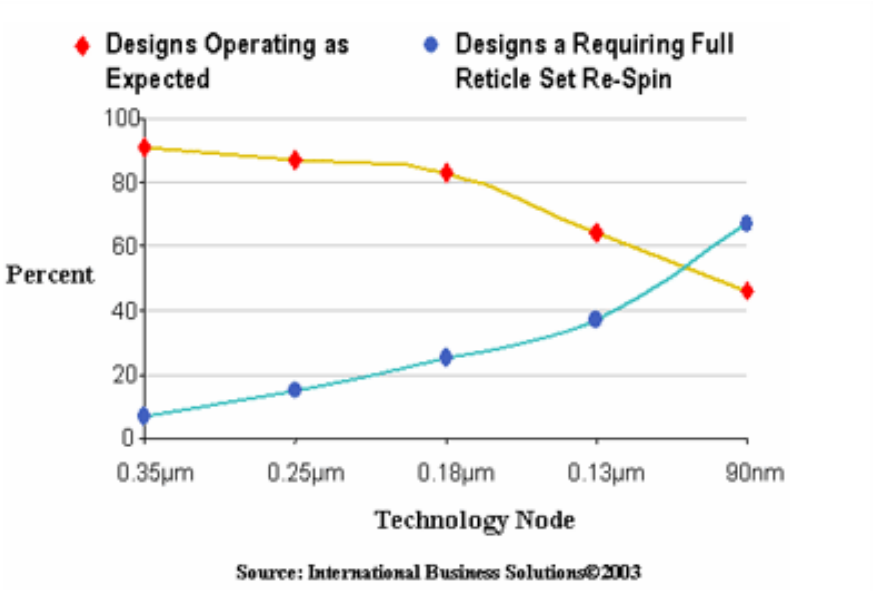


Figure 1. [Levi06] Ces courbes montrent le taux de Full-Respin (renvoi de dessins de masques à la fonderie suite aux erreurs de conception) par rapport au taux de bon design. La baisse du rendement serait du aux variations de process, non prise en compte par certains outils de conception.

Pour une plus grande robustesse, il conviendrait alors de s'appuyer sur la technique de redondance, ou sur le fonctionnement de code correcteur d'erreur (ECC). Pour la configuration de la table de correspondance, il faut au préalable identifier les défauts. La localisation des défauts sur la puce est une étape consommatrice de temps dans la mesure où le nombre de nanocomposants dépasse 10^{12} par centimètre carré [Gree07]. Peut-on l'éviter ? Il me semble plausible qu'une possibilité technologique existe en fondant une architecture qui rassemble le test (découverte de défauts), le calibrage, la configuration et l'auto-correction (compensation de dispersions et/ou de défauts) en une étape. Pour y répondre, une piste originale me semble intéressante, c'est l'architecture des réseaux de neurones. En effet, ils peuvent s'inviter à devenir une solution de rendement fiable, où la simplification est à l'horizon.

Selon [Fors04], la méthode de reconfiguration nécessite un nombre de modules redondants inférieur à toutes les autres méthodes sur la figure 2. Un FPGA (Field Programmable Gates Array) est une architecture des portes reconfigurable, elle fait partie de cette catégorie de circuits. Son avantage est la synthèse « post-process » des blocs combinatoires et séquentiels. Le FPGA s'appuie sur les blocs LUT (Look-Up Table), élément de la logique associative. On peut étudier une conception de LUT tolérante aux défauts. Ce bloc est une mémoire associative, or on sait qu'un réseau de neurones peut également remplir ce rôle.

Au regard de l'intérêt porté envers les nanocomposants, l'objectif de ce travail de thèse est d'étudier l'assemblage des nanocomposants pour la réalisation d'un bloc de mémoire associative en guise du bloc de calcul logique universel. Alors la question plus précisément est, comment réaliser ces blocs à base des nanocomposants de la façon la plus efficace en prenant en compte leurs spécificités.

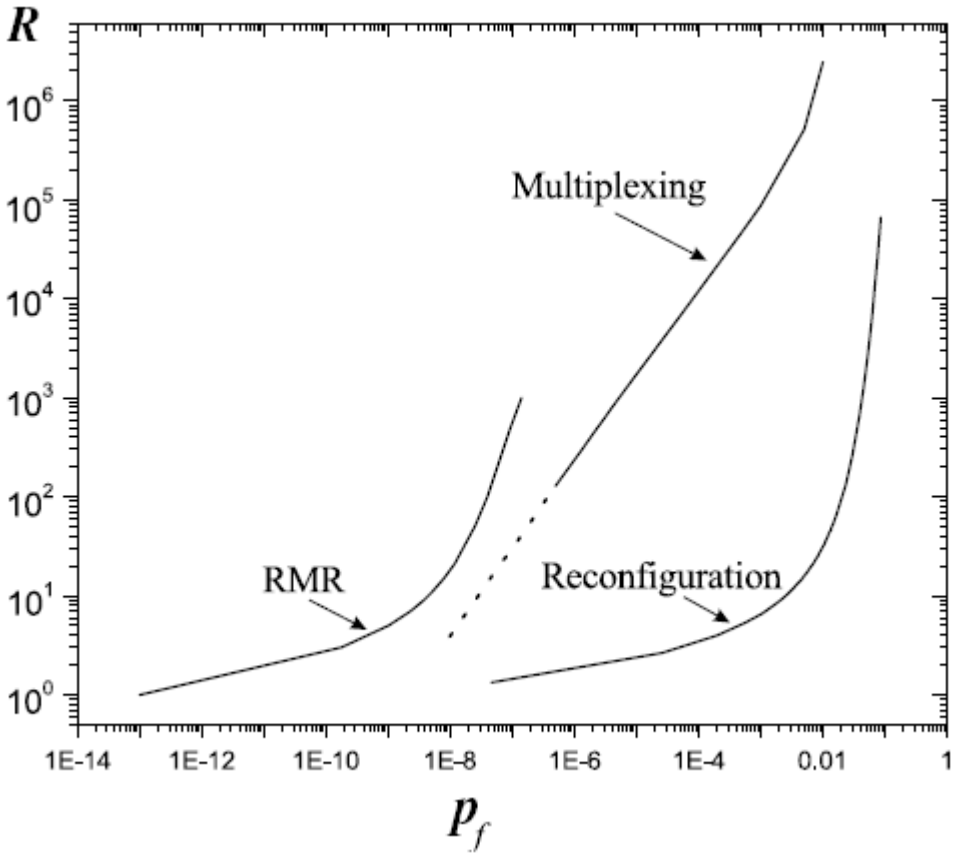


Figure 2. [Fors04] Ces courbes comparent trois stratégies pour la tolérance aux fautes. La redondance modulaire, le multiplexage de von Neumann, et la reconfiguration. *R* est le nombre de redondance requis pour avoir une probabilité de fonctionnement à 90%, P_f est la probabilité de défaut par composant. La meilleure solution préconisée est la reconfiguration.

CHAPITRE 2 : ETAT DE L'ART

1. Introduction

La loi de Moore prévoit un doublement approximatif de la densité d'intégration à un rythme régulier entre 18 à 24 mois. C'est en réalité devenu un objectif fixé par les industriels du domaine du semi-conducteur depuis ces quarante dernières années. On peut faire la corrélation entre le renouvellement de génération des appareils électroniques et le cycle d'évolution technologique. La réduction de taille de transistor induit l'augmentation de la richesse des fonctions intégrables, ce qui requiert des études partant des matériaux jusqu'à l'amélioration de la finesse de gravure par la photolithographie. Cependant, la réduction est limitée. Cette limite annoncée est d'abord d'ordre physique, mais aussi économique, et enfin technologique. Il convient ensuite de s'interroger sur ce qui va se passer : les nanotechnologies vont-elle vraiment surpasser la limite ultime de la technologie CMOS ? Pour contribuer à répondre à cette question, j'ai fait une synthèse, en m'appuyant sur les rapports de l'ITRS, de différentes technologies alternatives. Ces rapports font état des investigations menées jusqu'ici par de nombreux scientifiques, et ce à partir des analyses bibliographiques couvrant un large spectre de la recherche contemporaine. Je propose de traiter les thèmes liés à ce chapitre en cinq sections. D'emblée, je vais parler des composants électroniques que l'on peut classer dans la catégorie des nanotechnologies, puis, je vais détailler les critères de choix concernant ceux-ci. Ensuite, je vais consacrer une section pour décrire des réalisations expérimentales, et des architectures alternatives. Enfin, il est naturellement important de parler de l'architecture que j'ai choisi de développer dans mon travail de thèse : les réseaux de neurones.

2. Les nanocomposants

Les nanocomposants sont des éléments de base en nanoélectronique. Ils atteignent la taille déca-nanomètrique (10^{-8} mètre), ou nanométrique (10^{-9} mètre).

Les phénomènes physiques à la base des comportements des nanocomposants sont différents des uns aux autres. Il en résulte des avantages et des inconvénients dans l'usage des nanocomposants dans des domaines différents. Je vais maintenant montrer par les tableaux suivants les caractéristiques attendues de chaque nanocomposant. A partir de ces éléments, je vais dresser une analyse comparative des nanocomposants. Les points forts et les points faibles résultant de mon analyse sont montrés ci-dessous.

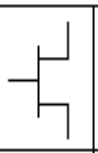
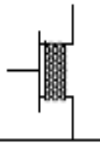
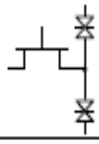
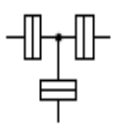
Device								
		FET [B]	*1D structures*	*Resonant Tunneling Devices*	*SET*	*Molecular*	*Ferromagnetic logic*	*Spin transistor*
Types		*Si CMOS*	*CNT FET* *NW FET* *NW hetero-structures* *Crossbar nanostructure*	*RTD-FET* *RTT*	*SET*	*Crossbar latch* *Molecular transistor* *Molecular QCA*	*Moving domain wall* *M: QCA*	*Spin transistor*
Supported Architectures		*Conventional*	*Conventional and Cross-bar*	*Conventional and CNN*	*CNN*	*Cross-bar and QCA*	*CNN Reconfigure logic and QCA*	*Conventional*
Cell Size (spatial pitch)	*Projected*	100 nm	100 nm [C]	100 nm [C]	40 nm [L]	10 nm [Q]	140 nm [U]	100 nm [C]
	Demonstrated	590 nm	~1.5 μm [D]	3μm [H]	~700 nm [M]	~2μm [R]	250 nm [V, W]	100 μm [X]
Density (device/cm^2)	*Projected*	1E10	4.5E9	4.5E9	6E10	1E12	5E9	4.5E9
	Demonstrated	2.8E8	4E7	1E7	2E8	2E7	1.6E9	1E4
Switch Speed	*Projected*	12 THz	6.3 THz [E]	16 THz [I]	10 THz [M]	1 THz [S]	1 GHz [U]	40 GHz [Y]
	Demonstrated	1 THz	200 MHz [F]	700 GHz [J]	2 THz [N]	100 Hz [R]	30 Hz [V, W]	Not known
Circuit Speed	*Projected*	61 GHz	61 GHz [C]	61 GHz [C]	1 GHz [L]	1 GHz [Q]	10 MHz [U]	Not known
	Demonstrated	5.6 GHz	220 Hz [G]	10 GHz [Z]	1 MHz [F]	100 Hz [R]	30 Hz [V]	Not known
Switching Energy, J	*Projected*	3E-18	3E-18	>3E-18	1×10^{-18} [L] [$>1.5\times10^{-17}$] [O]	5E-17 [T]	~1E-17 [V]	3E-18
	Demonstrated	1E-16	1E-11 [G]	1E-13 [K]	8×10^{-17} [P] [$>1.3\times10^{-14}$] [O]	3E-7 [R]	6E-18 [W]	Not known
Binary Throughput, GBit/ns/cm^2	*Projected*	238	238 [C]	238 [C]	10	1000	5E-2	Not known
	Demonstrated	1.6	1E-8	0.1	2E-4	2E-9	5E-8	Not known
Operational Temperature		RT	RT	4.2 – 300 K	20 K [L]	RT	RT	RT
Materials System		Si	CNT, Si, Ge, III-V, In_2O_3, ZnO, TiO_2, SiC,	III-V Si-Ge	III-V Si	Organic molecules	Ferromagnetic alloys	Si, III-V, complex metals oxides
Research activity [A]			171	88	65	204	25	102

Tableau 2. [ITRS, ERD 2005, Tab 59] Tableau des nanocomposants actifs. Les sigles : CNT pour Carbon Nanotube, NW pour Nanowire, SET pour Sigle Electron Transistor, CNN pour Cellular Neural Network, QCA pour Quantum Cellular Automata, RTT pour Resonant Tunnel Transistor, RTD pour Resonant Tunnel Diode, RT signifie Room Temperature

Nanocomposant	structures 1D	FET de canal III-V	SET	Moléculaire	Ferromagnetic	Spin transistor
Points forts	Extension du FET	Extension du FET	Bonne densité d'intégration $2e^{8}$ Cell / cm^{2}	Meilleure densité d'intégration en projection $1e^{12}$ / cm^{2}	Démontrée 250 nm / CellSize. $6e^{-18}$ Joule / commutation	Consommation faible en projection $3e^{-18}$ Joule / commutation
Points faibles	Résistance de contact très importante, fréquence d'opération faible démontrée 220 Hz	Coût supérieur	Sensible aux bruits thermiques, vitesse de circuit à 1Mhz	Consommation importante, 3^{e-7} / commutation	Fréquence de fonctionnement faible à 30 Hz	Rapport Ion/Ioff faible, moins de 100

Tableau 3. Comparaison de spécificité des nanocomposants actifs

Je vais maintenant décrire les nanocomposants présentés dans le tableau 2, afin d'en analyser les spécificités.

Les nanotubes de carbones offrent une haute conductivité thermique et disposent d'un transport électronique balistique pour une longueur inférieure à 100nm. Parmi les nanotubes de carbones on trouve le type multi-paroi (multi-wall) et le type mono-paroi (single-wall). Parfois, le mot *feuillet* est employé à la place de *paroi*. Les nanotubes de carbones peuvent être métalliques [Sait92], [Tans97] ou semiconducteurs. Typiquement, les nanotubes multiparois sont métalliques, conducteurs. Ils peuvent donc être employés pour l'interconnexion à condition d'avoir une faible résistance de contact. Le recuit (annealing) [Lee00] peut être utilisé pour la réalisation de faible résistance de contact électrode - nanotubes de carbone. Les chercheurs explorent les possibilités d'exploiter des nanotubes métalliques pour l'interconnexion, notamment dans le contexte du packaging 3D. Parmi les nanotubes monoparoi, un tiers est de type métallique, les autres sont semi-conducteurs. Ces derniers peuvent servir de canal dans les transistors à effet de champs, principalement de type P-FET. La transformation d'un nanotube semiconducteur de type P en N a aussi été démontrée. L'architecture complémentaire des portes CMOS

est donc extensible aux transistors à nanotubes de carbones [Dai02]. On parle alors de CN(T)FET, carbon nanotube field effect transistor. Le délai de propagation (C*V/I) des CNTFET prévu par [Wong05] est au moins deux à trois fois meilleur qu'un transistor en silicium. Cependant, la dispersion de caractéristique liée à leur nature (métallique ou semi-conducteur), leur diamètre, leur longueur, leur type (P ou N), et leur résistance de contact peut compromettre l'essor en tant que technologie alternative au CMOS. Il existe une forte corrélation entre le catalyseur et les propriétés structurelles. Le diamètre des nanotubes formés dépend du catalyseur initial, une forte dispersion est constatée systématiquement après la synthèse de nanotubes [Ishi04, Ago05, Kond06]. La variation de celui-ci peut conduire à la dispersion du courant I_{ds}. Le tri des nanotubes devient une étape rédhibitoire lors de la fabrication de la puce. Outre la dispersion de ces propriétés intrinsèques, l'orientation et le positionnement des nanotubes dans un circuit posent un problème considérable. La résistance de contact peut induire un changement de comportement au nanocomposant, on parle alors de « Contact-Dominated Devices ». Dans ce cas, le nanotube adopte le comportement d'une résistance linéaire. Il existe deux techniques de synthèse de nanotube de carbone. La première technique consiste à vaporiser du graphite en le chauffant à très haute température, au-dessus de 3000℃, puis on laisse le carbone se condenser. Pour atteindre cette température, on utilise l'arc électrique, ou le laser. Si l'on vaporise du graphite isolé dans son environnement, on obtient des nanotubes multi-parois. Sinon pour fabriquer des nanotubes mono-paroi, un catalyseur métallique est indispensable. La deuxième technique, dite de dépôt en phase vapeur (CVD, Chemical Vapor Deposition) est aussi courante. Des hydrocarbures comme le méthane ou l'éthylène sont décomposés à la surface de particules métalliques servant de catalyseur. Le processus se déroule dans un four chauffé. Suivant la température, on obtient des nanotubes multifeuillets (500-900℃) ou monofeuillets (750-1200℃).

Un autre élément 1D connu est le nanofil. Les nanofils peuvent être de natures différentes. Les nanofils métalliques sont fabriqués à partir de métaux comme le nickel, le platine ou l'or. Les nanofils semi-conducteurs sont fabriqués à partir des matériaux semi-conducteurs comme le silicium, le germanium ou encore le phosphure d'indium. Les nanofils isolant sont faits de matériaux d'oxides tels que le dioxyde de silicium ou du dioxyde de titane. L'approche courant pour sa synthèse est

le processus appelé vapeur-liquide-solide (VLS). La synthèse fait appel à une source de gaz silane SiH_4 et à un catalyseur nanoparticule métallique. L'auto-assemblage consiste à casser les atomes de silicium liés à des atomes d'hydrogène de silane pour les assembler en une nouvelle chaîne de nanofil. Quand la sursaturation est atteinte durant la « solidification » le nanofil commence à croître. La longueur et la structure des nanofils sont ainsi définies par la synthèse pendant la croissance de ceux-ci. Durant le processus, en alternant le gaz (composition d'autres matériaux) à intervalles données les nanofils hétérostructures peuvent être crées.

Maintenant on peut aussi se tourner vers le graphène qui est une monocouche de graphite. On sait qu'un nanotube monoparoi est un feuillet de graphène replié sur lui-même. Tout en étant un composé carbonique, il semblerait que certaines caractéristiques électroniques soient proches du nanotube de carbone [Berg06]. L'intérêt ici est que le graphène planaire peut être gravé par les techniques de la photolithographie conventionnelle. De plus, on peut le faire croître en épitaxie sur wafer. Il supporte également de hautes températures allant jusqu'à 1500℃, ce qui le rend compatible avec la technologie CMOS. Il est possible de contrôler leurs propriétés électroniques en choisissant leur dimension [Li08]. Par opposition des nanotubes, la structure planaire du graphène est supposée assurer plus d'interactions avec le substrat sur lequel il est déposé, d'où une forte sensibilité au substrat, le but étant le même, améliorer la conductivité du canal.

Parmi les autres nanocomposants répertoriés, la diode à résonance par effet tunnel (RTD, Resonant Tunneling Diode) utilise des effets quantiques pour reproduire l'effet de résistance différentielle négative (NDR, Negative Differential Resistance). Il s'agit d'une structure de boîte quantique entourée par une couche mince. Elle est appelée structure double barrière. Elle est capable de générer des oscillations térahertz à température ambiante. Elle peut être utilisée dans des circuits à ultra-haute vitesse. L'effet tunnel est sensible à la variation de l'épaisseur de la couche mince, ce qui conduit à une difficulté de contrôle. En disposant deux RTD en contre-sens, on peut créer une bi-stabilité de niveaux. La logique basée sur la bi-stabilité de RTD est sensible au bruit thermique.

Les SETs (Single-Electron Transistors) sont des éléments, de faible dimension, dont le principe repose sur le blocage de Coulomb. Le but est de contrôler individuellement les électrons dans un îlot quantique et ainsi de réduire à l'extrême le transfert de charge. Il peut se comporter comme une mémoire non volatile, mais le temps de rétention est fortement perturbé par le bruit thermique. Outre ce fait, on peut dire que le SET est sensible à l'instabilité de charge. La résistance d'accès est généralement grande. Son principal défaut est sa sortance (fan-out) limitée, il ne peut pas charger de capacité de sortie importante. Pour pouvoir fonctionner à la température ambiante, la taille de l'îlot quantique doit être inférieure à 10 nm, sinon le système doit être refroidi à la très basse température à seulement quelques degrés Kelvin.

Les molécules de type Rotaxane [Coll99] présentent un comportement bistable et une non linéarité de leur caractéristiques I(V). Ces caractéristiques ont d'abord été attribuées à la modification de la conformation de la molécule. Cependant, un doute persiste dans l'explication du phénomène physique observé. En fait, le comportement ne se situerait pas uniquement au niveau de la molécule mais résulterait également de l'interaction entre la molécule et l'électrode de contact [Stew04]. En effet, la variation de résistance est observée dans les expériences sans prendre en compte les barrières de potentiel élevées de l'électrode de contact. Les auteurs envisagent de les utiliser comme des switch moléculaires. Néanmoins le plus grand défi reste la reproductibilité de contacts qui permettent de contrôler l'état de chaque molécule (seulement 15 % des switchs [Gree07] sont exploitables). On observe également une forte variation de comportement entre les composants de la même puce. Enfin, on peut noter l'instabilité des molécules à hautes températures, qui peut les rendre incompatibles avec le process classique CMOS lors de la fabrication.

Les automates cellulaires quantiques (abrégé en QCA, pour Quantum Cellular Automata) se réfèrent à l'analogie entre la transition d'état quantique d'automates cellulaires et les modèles classiques d'automates cellulaires de von Neumann. Traditionnellement une cellule QCA est représentée par 4 points quantiques, positionnés aux quatre coins du carré et isolé l'une après l'autre. Le principe proposé à l'origine par [Lent93], consiste à utiliser un couplage électrostatique et l'effet de

blocage de Coulomb appliqués à une matrice de boîtes quantiques pour réaliser des opérations binaires. Ce fonctionnement n'est possible qu'en dessous du degré Kelvin (70 mK). Les portes logiques avec un domaine unique magnétique à la taille de 10 à 100 nm (au-dessus de la limite supra-paramagnétique, mais dans la limite du domaine unique) sont censées fonctionner à température ambiante. A titre illustratif [Imre06] un réseau de cellules magnétique a été réalisé pour effectuer des opérations logiques. Le champ dipolaire peut induire le retournement d'aimantation de la cellule de la sortie. Le changement d'états de cellules se propage dans un sens, poussé soit par un champ magnétique [Pari03], soit par un champ électrique [Orlo00].

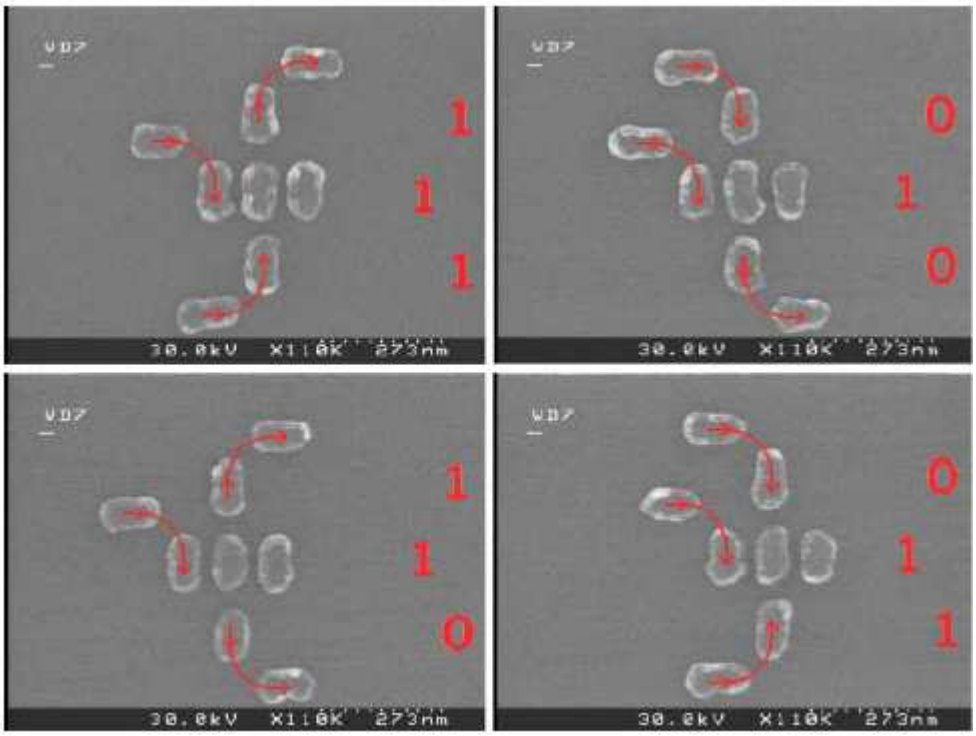


Figure 3. [Imre06] Les ilots magnétiques allongée en vertical (AFC) sont couplés antiferromagnetiquement. Les ilots allongés en horizontal (FC) sont couplés ferromagnetiquement. Cette structure est une porte majoritare conçue pour tester tous de combinaisons logiques d'entrée. Les flèches indique les directions d'aimantation quand un champ horizontal externe est appliqué. Les ilots AFC and FC se sont attribués la même valeur logique pour des aimantations opposées. L'état logique de bit 0 correspond à l'aimantation verticale vers le bas, et l'état logique 1 correspond à l'aimantation verticale vers le haut.

Dans les QCA magnétiques, l'opérateur de base est une porte majoritaire inverseuse, l'inversion étant due au couplage dipolaire. Chaque polarisation correspond à une valeur logique 0 ou 1. En les disposant dans une géométrie

particulière (cf. figure 3) une fonction logique peut être synthétisée. La dissipation de puissance dans les circuits QCA est supposée faible par rapport au CMOS. Elle est de 0.1W à 100 Mhz pour 10^{10} portes. Cependant, le signal de sortie est aussi extrêmement faible, on doit d'abord le détecter, puis le restaurer. Un champ magnétique oscillant appliqué au circuit sert d'apport d'énergie et aussi d'horloge. La fréquence de fonctionnement est de 30 Hz dans [Cowb00], mais théoriquement les QCAs peuvent atteindre une fréquence de fonctionnement de l'ordre du 10 MHz selon [ITRS, 2007 Edition – Emerging Research Devices]. La vitesse de calculs restera modeste au regard de la fréquence de fonctionnement du CMOS.

Le spin transistor (SpinFET) exploite la capacité spintronique de l'électron. Ce principe est décrit dans [Datt90]. Deux états différents proviennent de la discrimination sur le spin, propriété quantique fondamentale. Le composant est fait de contact source et drain ferromagnétique - semi métallique. Les électrons sont injectés depuis la source avec leurs spins polarisés. Il s'ensuit un parcours suivant l'axe du canal, où la précession du spin est alors contrôlée par la tension de grille. Du côté drain, la probabilité de la transmission de l'électron dépend du composant du vecteur de spin. Le courant source-drain est modulé selon l'angle de précession. Il peut exister des problèmes d'interfaces, et les problèmes d'impuretés non magnétiques [Pram04], de ces faits l'injection de la polarisation n'est pas encore très bien maîtrisée. De plus, la faiblesse du rapport de courant Ion-Ioff du spin transistor (environ 100 selon [Denn03]) est un obstacle à la performance, particulièrement en comparaison du transistor MOS (10^8). A titre d'exemple, un prototype de spin-transistor a été fabriqué récemment, de structure GaMnAs fonctionnant à T=2.6 K [Mizu07].

Les points de mémoires connus en nano-électronique sont basés sur les principes des phénomènes physiques comme la modification de la charge électrique d'une capacité, et aussi comme la variabilité de résistance du nanocomposant. Je vais maintenant résumer les principaux phénomènes physiques exploités :

- Mémoire à barrière tunnel : C'est le type de mémoire le plus couramment utilisé aujourd'hui. On ajoute une barrière multicouche de diélectrique empilée sur la grille. La charge sur la barrière modifie le comportement de

conduction électronique du transistor, ce qui se traduit par un décalage du seuil.

- Mémoire ferroélectrique FeFET : La polarisation de la charge stockée est prise en compte dans ce type de mémoire. Comme la mémoire à barrière tunnel, les charges stockées se trouvent sur la grille, on utilise une couche composée de type métal-ferroélectrique-diélectrique-semiconducteur.

- Mémoire de résistance filamentaire : La résistance entre deux électrodes est modifiée par une tension (courant) appliquée, car un filament de conduction peut être détruit ou recrée.

- Mémoire nano-électromécanique : La résistance du nanocomposant est modifiée selon le déplacement mécaniquement d'un élements, par exemple la flexion d'une poutre. Le contact peut être alors ouvert ou fermé.

- Mémoire à changement de phase : Elle utilise le comportement du verre de chalcogenide. A l'application de la chaleur par impulsion, selon la température la molécule transite entre deux états, amorphe et cristallin. Sa résistance varie suivant l'état dans lequel elle se trouve.

- Mémoire moléculaire : La modification de la résistance est basée sur le principe de la variation de conductance par rapport à la conformation de la molécule.

Je viens d'évoquer la famille des nanocomposants, où la forte dispersion de fabrication est omni-présente. Les RTD sont beaucoup trop sensibles aux bruits d'environnement. Actuellement, les SET nécessitent un système de refroidissement à très basse température pour fonctionner. La fréquence de fonctionnement de circuit basée sur les QCA magnétiques sera à fortiori très faible, incomparable avec le CMOS. Les transistors spin ont un rapport Ion/Ioff faible. En effet, améliorer les caractéristiques des nanocomposants est un challenge. Quand les nanocomposants seront affranchis de leurs contraintes, on pourra réellement parler de leur émergence. Maintenant je vais évoquer les principes des critères de choix.

3. Critères de choix des nanocomposants

Je vais comparer les nanocomposants, dans l'objectif de concevoir un système à composants nanoélectroniques. Pour cela, j'ai choisi les critères suivants, la température de fonctionnement, le coût du matériau, la densité d'intégration, et la consommation.

Les nanocomposants dotés du comportement électronique attendu doivent être destinés à fonctionner à la température ambiante « Room Temperature » (aux alentours de 300 K). La puissance pour l'évacuation de chaleur est évaluée à : 2 W pour maintenir le système dans les conditions opérationnelles par simple ventillation air cooling, 300 W pour maintenir un système à la température de 77 degré Kelvin (azote liquide), et 7 kW pour 4,2 degré Kelvin (hélium liquide) par cryogénie. Cela entraîne non seulement un surcoût énergétique exorbitant, mais aussi des problèmes d'intégration de système de refroidissement dans un volume d'intégration limité. Bien que les étapes technologiques accomplies soient prodigieuses, l'adéquation énergétique doit rester logique. Jusqu'à présent le défi à relever est de faire fonctionner l'ensemble des nanocomposants à la température ambiante, suffisamment tolérantes aux bruits thermiques.

A partir des matériaux de base, j'ai essayé de mettre en évidence quelques-unes des questions de substitution de la matière de silicium utilisée jusqu'alors. Je m'intéresse ici seulement à la part du coût de la matière première dans le système sur puce. J'ai pris en compte l'interconnexion métalliques (Aluminium, Cuivre) et le substrat (matériaux III-V et IV, puis Carbones). En effet, si les équipements de fabrication sont amortissables à terme, si l'énergie de la transformation physico-chimique dans les procédés est maitrisable, il restera donc la matière première utilisée qui sera un facteur déterminant. La quantité des éléments mis en jeu à l'échelle industrielle est en réalité imposante durant toute la production. Regardons les matériaux sous un angle capitalistique, les coûts sont relevés par [EniG03].

«

Matériaux interconnexion in chip :

Le prix de l'aluminium, pur à 99.9 %, sous forme de granules, est de 46 €/kg.

Le prix du cuivre, pur à 99.9 %, en granules, est de 26 € pour 500 g. soit 52 €/kg.

Matériaux pouvant devenir semi-conducteur :

Le prix du silicium (Si), pur à 99.999 %, sous formes de granules, est de 67 € / kg.

Le prix du germanium (Ge), pur à 99.9999 %, en morceaux, est de 1088 € pour 120 g. soit 9067 €/kg.

Le prix du gallium (Ga) pur à 99.99 %, en morceaux, est de 930 € pour 500 g. soit 1860 €/kg.

Le prix de l'arsenic (As) pur à 99.5 %, sous forme de mousse est de 323 € pour 100 g. soit 3230 €/kg.

Le prix de l'indium (In) pur à 99.9 %, en granules, est de 400 € pour 250 g. soit 1600 €/kg.

Le prix de l'antimoine (Sb), pur à 99.9 %, en morceaux, est de 154 € pour 450 g. soit 343 €/kg.

Le graphite en poudre, pur à 99.9 %, coûte 47 € / kg.

»

Ce constat succinct montre que le Silicium est de loin un matériau semiconducteur bon marché, en tout cas meilleur marché que le Germanium (environ 135 fois inférieur), et les autres élements des colonnes III-V de la table de Mendeleïev. Dans l'avenir, il me paraît très probable que la transition vers d'autres matériaux semiconducteurs que le silicium et le carbone soit restreinte aux marchés de niche. Maintenant, il est intéressant de noter que le prix du graphite est du même ordre que celui du silicium. Par ailleurs, en 2007, on trouve d'ors et déjà des nanotubes de carbone au prix de $250 / kg, ce chiffre baisse au fur des années en même temps que la production mondiale augmente. Il est vrai que les prix de la matière première subit des fluctuations constantes, elles dépendent des enjeux géopolitiques, du coût de l'extraction, enfin de la disponibilité. La transformation chimique a également un coût. Il existe des modèles économiques plus justes qui pourraient montrer le coût final selon la composition de matériaux. D'ailleurs, l'évolution du coût des masques, de R&D, et de production est un facteur constant et pertinent pour l'industrie. Bien sûr, à leurs tours, ils devront être pris en compte dans l'évaluation globale.

La densité d'intégration dépend de la dimension minimale des règles de dessin imposée par la méthode de fabrication. En partant de l'analyse du tableau 2, on voit, selon les projections, en comparant la densité d'intégration des nanocomposants que le transistor traditionnel n'est pas le plus performant (device-density.cm^{-2} : 10^{10} pour le transistor traditionnel, 6.10^{10} pour les SET, 10^{12} pour la matrice moléculaire). La même analyse sur les différentes technologies de mémoire montre d'une part que la densité d'intégration des technologies DRAM (en Baseline) est proche de l'optimum, et d'autre part que la densité d'intégration des autres technologies pourrait la dépasser si les projections les plus optimistes se réalisent [ITRS, ERD 2007, Tab.5b].

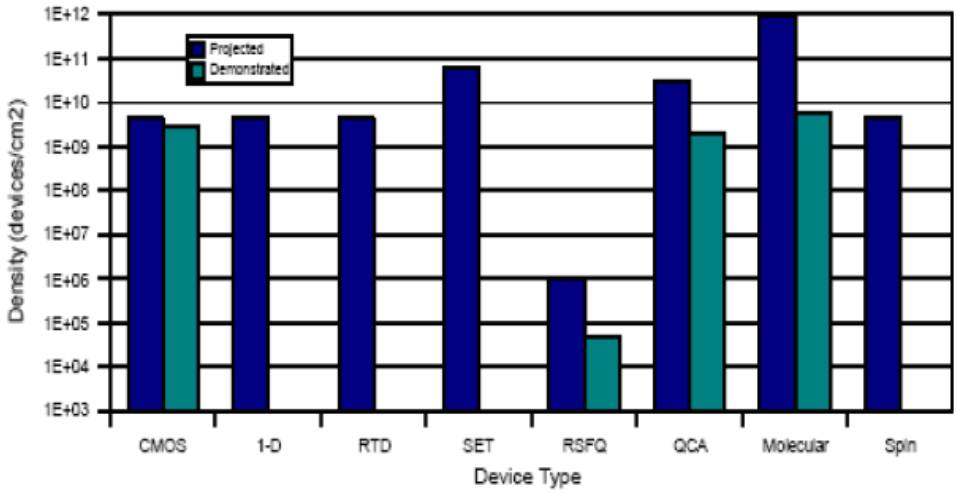


Figure 4. [Teix06], [Zhir05] La densité d'intégration des nanocomposants.

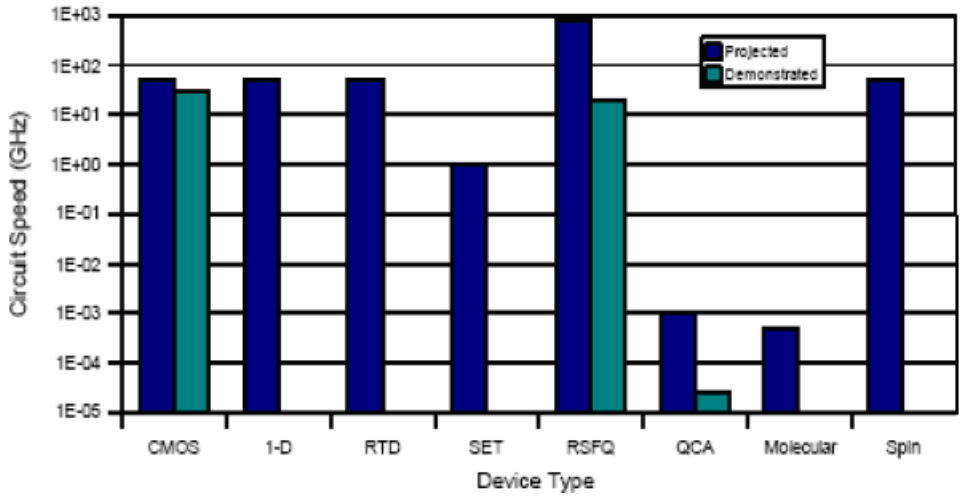


Figure 5. [Teix06], [Zhir05] La vitesse du circuit des nanocomposants.

En règle générale, pour les systèmes de calcul sur puce la dissipation

thermique va également limiter la densité d'intégration. Un système de calcul doit alors minimiser sa consommation. Pour un système proche de l'équilibre thermodynamique, la hauteur de la barrière permettant de discriminer les états logiques doit être beaucoup plus grande (on compte 1000 fois plus) que l'énergie de fluctuation thermique (von Neumann-Landauer, 3.10^{-21} J). La comparaison des énergies de commutation dans le tableau 2 montre que ce critère peut favoriser le transistor à silicium face aux nouvelles technologies. Tenant compte de tous ces facteurs (dimensions, fréquence, énergie), la puissance de commutation surfacique par unité de fréquence pour différentes nanotechnologies projetée est montrée par le tableau 4. Elle équivaut à la multiplication de la densité d'intégration, par l'énergie de commutation par cycle. On remarque que la plupart des nanocomposants convergent vers une puissance de commutation de l'ordre de 10^{-8} W par Hertz par centimètre carré, sauf pour la technologie moléculaire qui semble incapable d'offrir un meilleur compromis. En ce qui concerne la technologie actuelle, à cause de la consommation électrique, et de dissipation de chaleur, un maximum de la fréquence de fonctionnement pour les microprocesseurs a été atteint au milieu des années 2000, elle stagne à 3.3 GHz, à refroidissement par flux d'air.

nanocomposant	Si FET	Structure 1D	RTD	SET	Molécule.	Ferromag.	Spin FET
Max. Puissance dissipée dans les démontrateurs ($W.Hz^{-1}.cm^{-2}$)	2,80E-08	4,00E-04	1,00E-06	1,60E-08	6,00E+00	9,60E-09	-
Puissance dissipée théorique ($W.Hz^{-1}.cm^{-2}$)	3,00E-08	1,35E-08	1,35E-08	6,00E-08	5,00E-05	5,00E-08	1,35E-08

Tableau 4. Comparaison de la puissance dissipée pour différents nanocomposants, à partir de [ITRS, 2005–Emerging Research Devices, Tab 59].

Une faible dissipation permet une fréquence de fonctionnement élevée, et une plus grande puissance de calcul, qui est en réalité proportionnelle au produit de la fréquence de fonctionnement et du nombre d'opérations effectué par cycle. Les figures 4 et 5 présentent la projection [Teix06], [Zhir05], [ITRS] en termes de densité d'intégration et de vitesse de fonctionnement des circuits. On constate que, selon

toute vraisemblance, le CMOS dispose d'un bon profil. En principe la mobilité des porteurs contribue à améliorer la conductivité et un meilleur comportement dynamique. En augmentant la fréquence de fonctionnement du système, le délai de propagation doit être diminué.

nanocomposants	mobilité
μ(InSb)	$7{,}7.10^4$ $cm^2/V.s$
μ(Graphene) [Berg06]	$2{,}7.10^4$ $cm^2/V.s$
μ(AsGa)	$9{,}2.10^3$ $cm^2/V.s$
μ(CNTFET) [Fuhr02]	9.10^3 $cm^2/V.s$
μ(Ge)	$3{,}6.10^3$ $cm^2/V.s$
μ(Si monocristallin)	$1{,}3.10^3$ $cm^2/V.s$
μ(Ge/Si-Nanofil) [Xian06]	730 $cm^2/V.s$
μ(Si-Nanofil) [Cui03]	30 - 560 $cm^2/V.s$
μ(Pentacene) [Jurc04]	35 $cm^2/V.s$

Tableau 5. La mobilité électronique de différents matériaux semi-conducteur à T=300 K

En réalité, la résistance d'accès (contact), introduite lors de l'intégration limite le courant I_{ds}, ce qui peut conduire finalement à un gain médiocre. On constate que la tension utile aux bornes du nanocomposant intrinsèque est d'autant plus faible que les résistances d'accès sont importantes. En conséquence, le courant I_{ds} va chuter plus que prévu, ce qui conduit à une transconductance (g_m=dI/dV) relative plus faible. Ce phénomène est souvent observé. Le problème de contact ressort régulièrement au niveau des interconnexions des nanocomposants. Le comportement I(V) est sensible aux paramètres environnant extrinsèques, et non seulement à son transport électronique intrinsèque. Les propriétés des nanocomposants observées ne seront

pas toujours reproductibles dans un environnement quelconque.

nanocomposants	Transconductance g_m
Si FET [Nave00]	4000 mS / mm
CNTFET [Nihe03]	8.7 µS pour un diamètre 1,5 nm
Si-Nanofil [Cui03]	45 - 800 nS pour un diamètre 5 nm
Ge/Si Nanofil [Xian06]	26 µS (diamètre 18nm)
SpinFET [Suga06]	2700 mS / mm

Tableau 6. La transconductance de différentes nanotechnologies

Aujourd'hui, on constate que les contraintes sur l'usage des nanocomposants dans un système sur puce sont nombreuses pour réellement offrir une alternative. Pour l'instant, aucune technologie alternative n'a encore convaincu de sa réussite complète. En conclusion, l'analyse précédente fait ressortir que les nanocomposants ne peuvent prétendre remplacer le transistor à silicium. Si on prend le critère de l'énergie, rien ne montre qu'il existe un choix alternatif au CMOS. En termes de densité d'intégration, seule la matrice moléculaire fait gagner deux ordres de grandeur par rapport au CMOS, mais reste à surmonter son taux de défaut important, et l'accessibilité des nanocomposants, et leur consommation élevée. L'utilisation de méthodes d'auto-assemblage permettrait d'envisager l'exploitation de tels circuits. Dans ce cas il resterait à fonctionnaliser le système par une méthode de programmation. Enfin, cette course semble plutôt indiquer une nanotechnologie complémentaire au CMOS.

4. Réalisations expérimentales en nanoélectronique

Des circuits ont été construits à partir des nanocomposants connus. Parmi ceux-là, je vais maintenant présenter les portes logiques à base de switchs

moléculaires, de nanofils, de nanotubes, puis la construction de matrice à partir de ces éléments.

Dans [Coll99], on peut constater que les éléments logiques peuvent être construits à partir d'une matrice de switchs moléculaires configurables. Cette matrice est constituée de monocouche de molécules de rotaxanes imbriquée dans deux électrodes métalliques opposées. Un effet bistable est obtenu dans cette structure. Les électrodes envisagées sont une composition de Ti-Al, ou Ti-Au, ou encore Cr-Al. Les switchs sont irréversibles dans [Coll99]. La tension de programmation est de 0,7 V. Elle est appelée la tension d'oxydation (oxidizing voltage). Elle est utilisée pour la configuration de la molécule. On estime qu'elle devrait être deux fois supérieure (voire plus) par rapport à la tension de lecture. Sur la figure suivante, une vue en coupe montre l'entassement des couches.

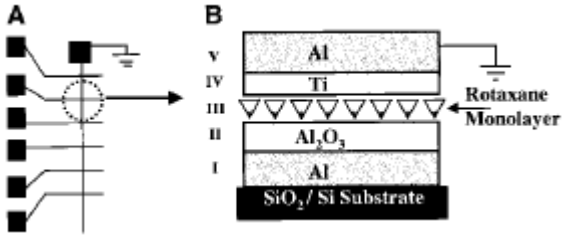


Figure 6. [Coll99] A) 6 nanocomposants sont connectés à la colonne, relié à une résistance de pull-down. B) Vue en coupe de la strucure impliquant le nanocomposant. L'épaisseur de barrière de la couche de passivation Al2O3 est de 1 à 1,5 nm. La couche interface de Ti-rotaxane est de 0,5 nm.

L'auteur a donné le nom de CAEN (chemically assembled electronic nanocomputers) à l'architecture.

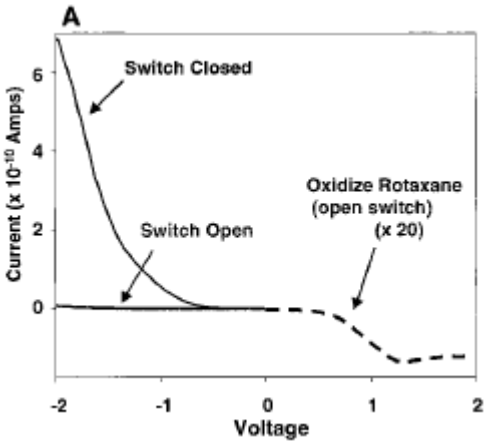


Figure 7. [Coll99] Caractéristiques I(V) du nanocomposant

Cette molécule chimique « rotaxane » est supposée adopter un comportement différent selon sa conformation. Elle change selon la tension électrique qui lui est appliquée. De ce fait, sa résistance électrique passe d'un état faible à un état fort. Les états différents de résistance permettent de concevoir des éléments basés sur la logique à diode.

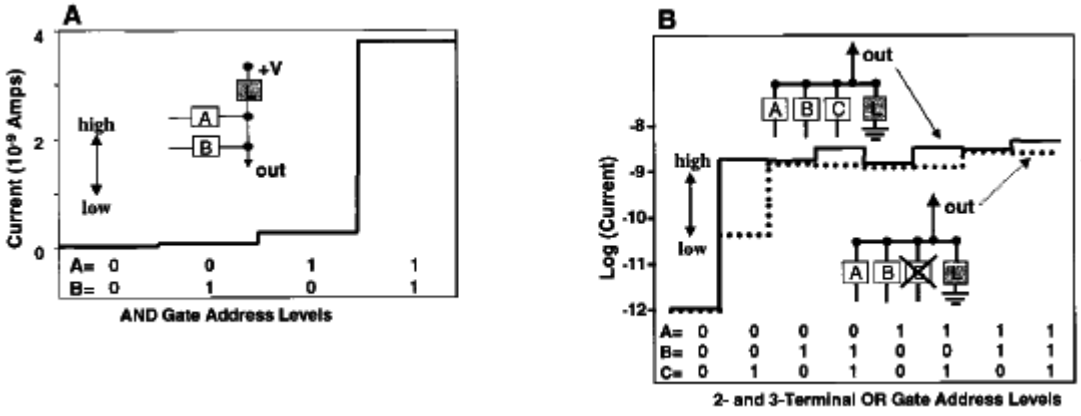


Figure 8. [Coll99] A) Porte AND avec une résistance de pull-up. B) Porte OR avec une résistance de pull-down.

Les différentes molécules sont connectées sur la ligne, et selon le type d'architecture une résistance de pull-up (connectée à +V) ou une résistance de pull-down (connectée à 0) peut être employée. D'après les tests, le rapport Ion et Ioff est alors de 15 pour une porte AND, et 30 pour une porte OR. Cette approche constitue une première expérience prouvée de l'architecture nano-électronique. Cependant, elle manque d'un élément actif non linéaire pour la restauration du signal. Dans [Huan01-2], une réalisation expérimentale a été portée sur des nanofils croisés. La disposition des nanofils est orthogonale. Ils forment alors une matrice de jonctions. Les nanofils peuvent être assemblés de façon efficace. Il existe des méthodes (alignement fluidique) [Huan01] d'alignement parallèle pour créer une portion de la matrice dans une surface désignée. Les propriétés électroniques et la taille des nanofils sont contrôlables avec précision au cours de la synthèse. Les nanofils de p-Si, n-GaN ont été utilisés dans cette expérience. Les nanofils catalysés ont des diamètres de 10 à 25 nm. Ces jonctions peuvent aussi être utilisées pour l'effet de transistor à effet de champs. Le nanofil est à la fois un fil conducteur et l'électrode d'entrée. Les caractéristiques mesurées I(V) montrent que ces nanofils ont une tension de seuil variant entre 0,6 et 1,3 V. Outre cette dispersion, 5 % des nanofils

sont défectueux. Comme précédemment, la logique à diode permet de construire les portes logiques OR, AND, NOR (cf. figure 9, 10, 11), mais réalisables à condition d'avoir testé les nanofils au préalable. Un additionneur est alors synthétisable en interconnectant ces portes logiques entre elles.

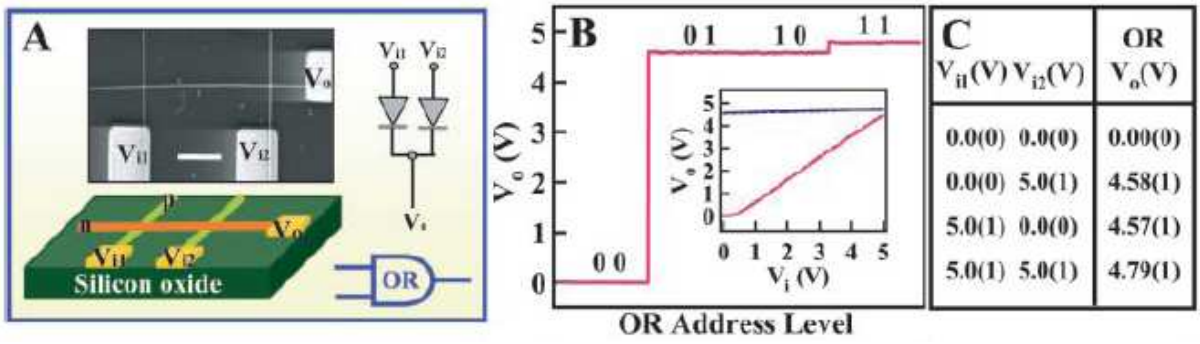


Figure 9. [Huan01-2] La mise en œuvre de la porte OR est basée sur la logique à diode. Les différentes tensions, la sortie Vo et les entrées Vi1, Vi2 sont montrées sur la figure.

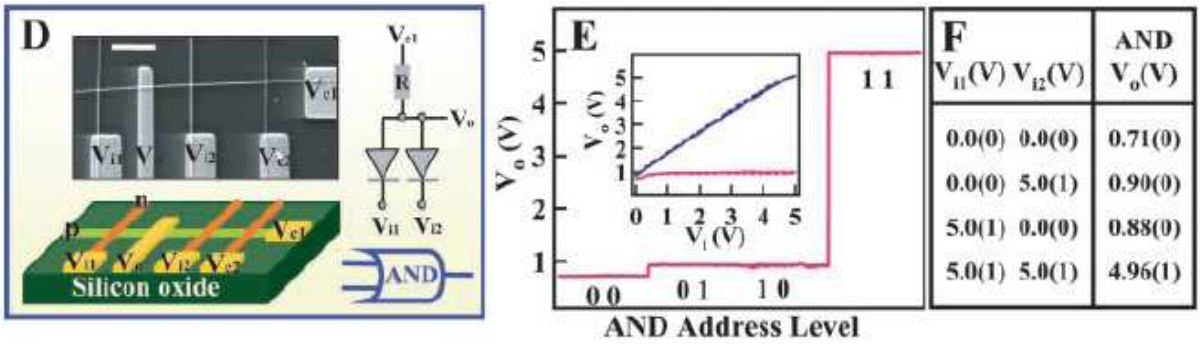


Figure 10. [Huan01-2] La mise en œuvre de la porte AND est basée sur la logique à diode. Les différentes tensions, la sortie Vo et les entrées Vi1, Vi2 sont montrées sur la figure. Ici une résistance de pull-up est utilisée.

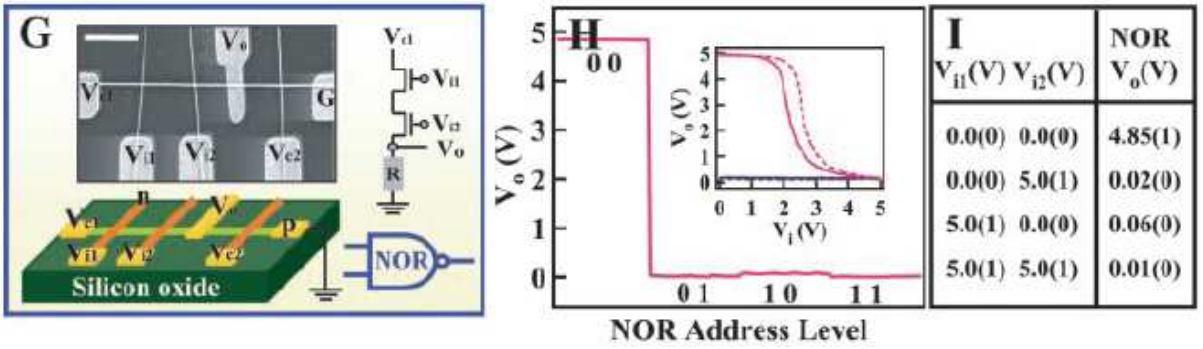
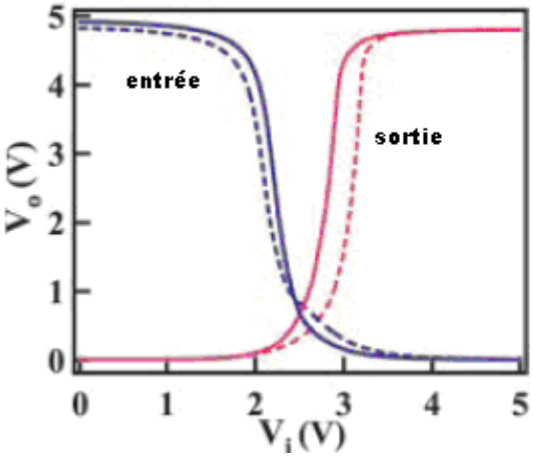


Figure 11. [Huan01-2] La mise en œuvre de la porte NOR est basée sur la logique à diode. Les différentes tensions, la sortie Vo et les entrées Vi1, Vi2 sont montrées sur la figure. Ici une résistance de pull-down est utilisée.

L'apport essentiel de ce travail est la démonstration d'une non linéarité de la sortie, avec un gain plus grand que 5. Elle permet de constater l'effet d'amplification crée par les nanofils, ce qui est loin d'être négligeable. En effet, pour tout calcul, le signal doit être relayé avec un niveau restauré pour la porte suivante.



V_{i1}(V)	V_{i2}(V)	CARRY "AND" V_o	SUM "XOR" V_o
0.0(0)	0.0(0)	0.82(0)	0.02(0)
0.0(0)	5.0(1)	1.08(0)	4.81(1)
5.0(1)	0.0(0)	0.95(0)	4.78(1)
5.0(1)	5.0(1)	4.98(1)	0.01(0)

Figure 12. [Huan01-2] A gauche, la porte XOR avec une entrée fixée à 0V (logique bas), l'autre varie entre 0V et à 5V (logique haut), en pointillée la configuration des entrées est alternée. A droite, les tenions logiques du full-adder.

Une approche avec les nanotubes de carbones au lieu de nanofils est réalisée par [Bach01]. La méthode de fabrication est devenue courante. La couche d'Aluminium est déposée sur un wafer de Silicium oxydé et le motif de la grille (backgate) est lithographié par l'e-beam. Des nanotubes de carbone monoparoi sont suspendus dans une solution de dichloroéthane, pour obtenir une concentration moyenne adéquate. Puis ils sont enduits sur le wafer. Avec un microscope à force atomique, les nanotubes qui sont situés au-dessus des grilles d'Aluminium sont sélectionnés. Les nanofils localisés sont ensuite soudés par laser sur les électrodes de contact en Or à proximité. La largeur de l'électrode est de 250 nm, et celle de la grille est de 400 nm. Cependant il existe un caractère aléatoire dans cette expérience. Le type des nanotubes de carbones et leur position sur la grille de contrôle sont souvent inconnus d'avance. Les nanofils ne se trouvent pas forcément dans la bonne position entre les électrodes en Or. Cependant en utilisant un cantilever nanométrique pour la manipulation individuelle des nanofils sur la surface de puce, les portes logiques peuvent être faites de nanotubes de carbone semi-conducteur. La fonction inverseuse, NOR, un point mémoire SRAM, et un oscillateur à 3 étages ont été réalisés dans [Bach01].

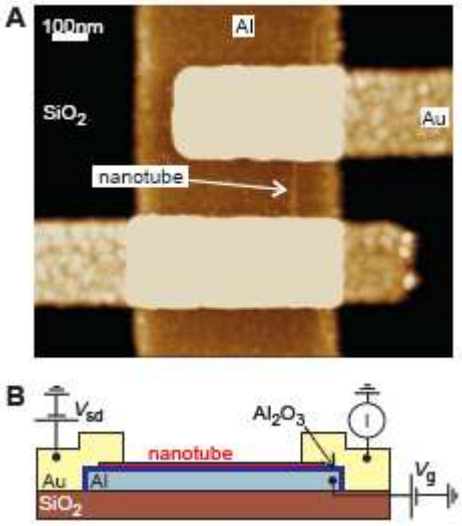


Figure 13. [Bach01] A) le nanofil se situe entre deux électrodes de contact en Or, la backgate est faite en Aluminium. B) La représentation schématique du test.

La référence [Melo03] montre que la densité d'intégration peut être très importante sans recourir à la photolithographie. Les auteurs sont parvenus à la production de matrices de nanofils semi-conducteurs et métalliques, d'une densité de jonctions supérieure à $10^{11}.cm^{-2}$. Les nanofils de diamètre 8 nm sont séparés de 16 nm (de centre à centre), ils sont disposés en parallèles. Les nanofils sont transférés par tampon sur le wafer, par paquet, ce procédé (cf. figure 14) peut être effectué plusieurs fois.

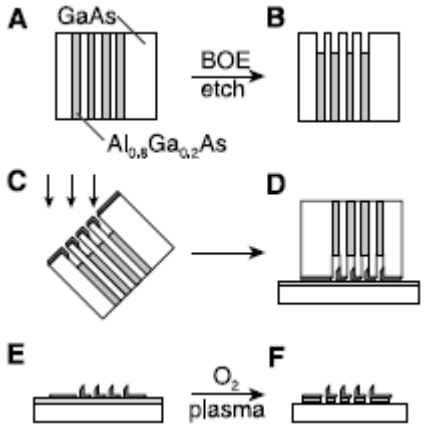


Figure 14. [Melo03] A) Treillis en GaAs/AlGaAs, B) après gravure sélective d'AlGaAs C) dépôts de nanofils métalliques sur le tampon, incliné à 36°, D) contact sur une couche adhésive sur silicium, E) libération de nanofils métalliques par gravure d'oxyde de GaAs, F) supprimer la couche adhésive par plasma

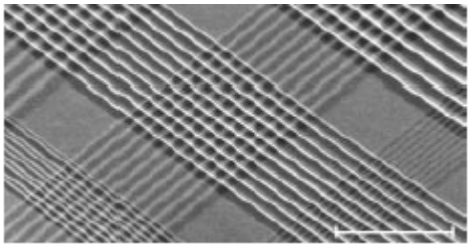


Figure 15. [Melo03] Les crossbar transversals sont fabriqués à partir des nanofils de Pt. Le crossbar central a une densité de jonctions 5.10^{10} cm^2.

Dans [Gree07], une extension de circuit [Coll99] à base de mémoire moléculaire (rotaxane) de 160 000 bits a été fabriquée. La matrice est composée de nanofils de Silicium / rotaxane / nanofils de Titanium. La taille de la molécule n'a pas d'influence sur le mécanisme de mémorisation.

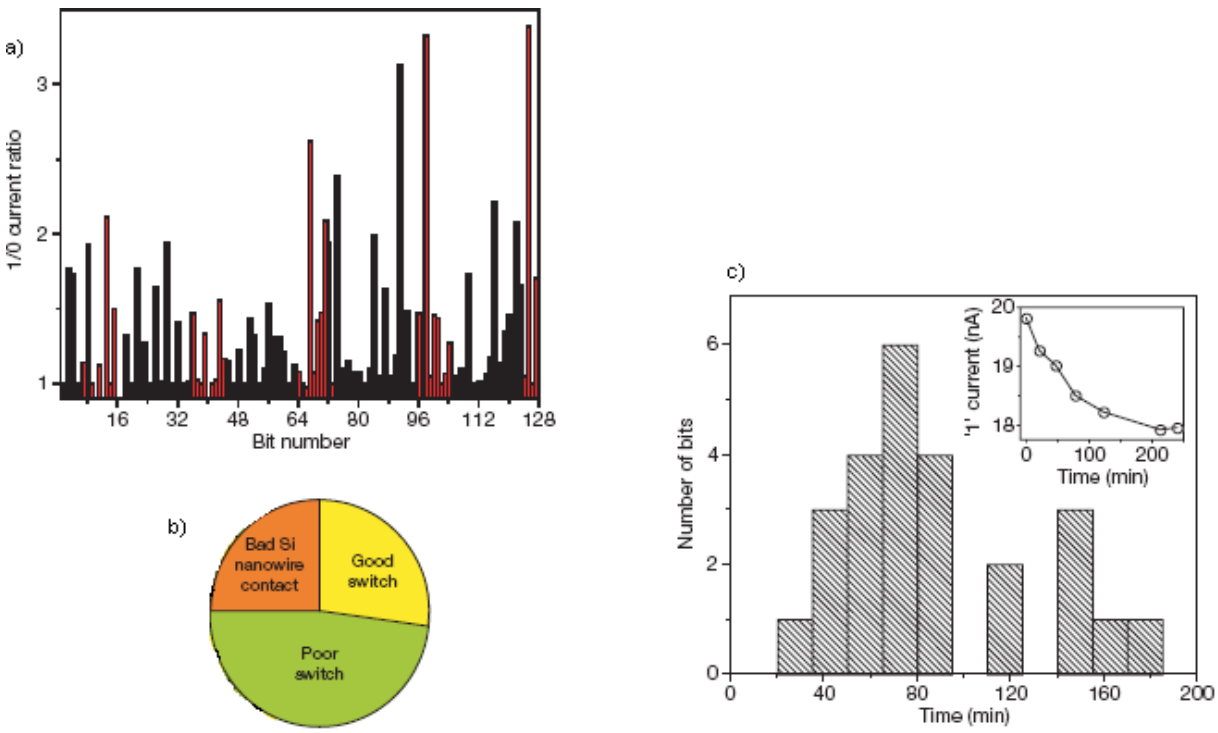


Figure 16. [Gree07] A) le rapport de courant de lecture entre l'état 1 et 0. b) la proportion mesurée de la configurabilité des switchs (bits). c) le temps de rétenion; à partir de 190 minutes, tous les bits basculent à leur état initial.

La durée de vie est d'environ 5 cycles d'écriture/effacement. Le temps de rétention est très aléatoire, dépendant de la molécule, la durée maximale est moins de 200 minutes. Certains switchs sont défectueux, appelé « poor switch ». 48 % de tous les bits testés sont de nature non progammables, dont 26% sont en circuit ouvert, et 22 % en court circuit. Le rapport (Ion/Ioff) de courant de lecture témoigne

une forte dispersion entre les molécules. Malgré cela, la densité du circuit pourrait atteindre 10^{11} bits.cm^{-2} pour le stockage de données. Selon les projections de l'ITRS, la DRAM n'attendrait la même densité qu'en 2020. La publication rapporte que les défauts affectant ce circuit sont d'une part dus aux mauvais contacts entre la molécule et les nanofils, et d'autre part aux courts-circuits entre les nanofils. Les bits fonctionnels sont détectés et analysés. Entre les défauts des nanofils, les défauts des molécules, seulement un quart des bits conçus sont utilisables.

5. Architectures nanoélectroniques

A partir des éléments des réalisations expérimentales, des architectures ont été proposées pour tenter d'exploiter la haute densité d'intégration des nanocomposants. Les auteurs ont d'abord du faire face à la complexité du circuit résultant du grand nombre de nanocomposants intégrables. Ensuite, des solutions permettant la robustesse ont été étudiées pour faire face à la dispersion de caractéristiques, et à la quantité importante de défauts. Enfin, les auteurs ont cherché à répondre aux difficultés apparaissant dans l'utilisation des nanocomposants, en particulier concernant l'adressage et la restauration de niveaux logiques. Les concepteurs s'intéressent notamment au domaine des circuits reconfigurables. Je vais donner une description des principales architectures de système utilisant des nanocomposants.

Une première approche [Deho05] consiste à réaliser des matrices de nanofils. Les nanofils alignés sont croisés de façon othogonale, par bloc séparé. La photolithographie est utilisée pour la découpe de nanofils. Le nombre de nanofils par bloc est un paramètre variable. Un balayage fonctionnel par bloc est nécessaire pour déterminer la validité des nanofils. Pour cela, il est nécessaire de pouvoir adresser chaque nanofil. Pour adresser sélectivement les nanofils, [Deho05] propose de les rendre sensible chacun à un code différent grâce à un dopage longitudinal sur les nanofils désalignés. Le procédé découle de la croissance des structures 1D hétérogènes longitudinales. Les adresses sont fournies sur des lignes lithographiées. Des nanofils d'un diamètre de l'ordre du nanomètre peuvent être sélectionnés par effet de champ appliqué par les lignes d'adresses. L'intersection des nanofils peut agir comme des diodes programmables ou des transistors à effet de champ. Un code d'adressage permet de sélectionner éventuellement un ou plusieurs nanofils

mélangés au sein de la matrice. L'idée de la matrice programmable est d'abord basée sur la logique à précharge [Wang04]. Un code correcteur [Snid07] peut être ajouté en sus du code sélecteur pour la tolérance aux défauts. Dans [Deho05], l'auteur a insisté sur la nécessité d'interconnecter des blocs de petite taille, et il a aussi évoqué le problème de la limite des longueurs des nanofils. Ailleurs, un crossbar plus ou moins similaire a été fabriqué au laboratoire HP [Chen03].

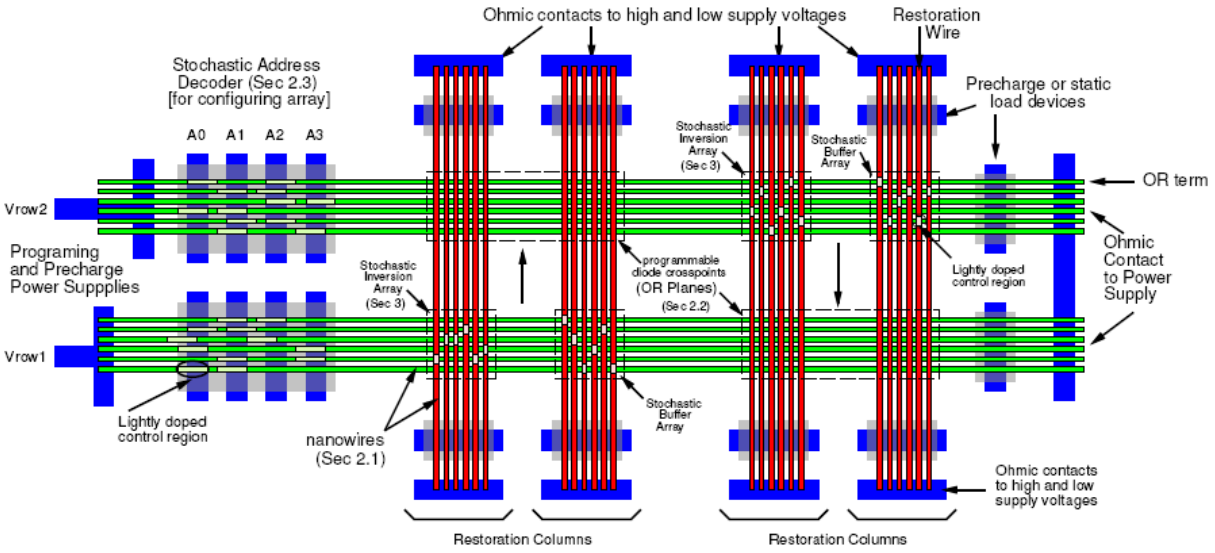


Figure 17. La matrice PLA à base de nanofils [Deho05], sur ce schéma on peut constater la présence de la matrice programmable à logique de diode, l'unité de restauration de signaux, et l'adressage stochastiques pour la sélection des nanofils.

La question qui se pose ensuite est, comment organiser les blocs (ou appelés nanoBlock dans [Gold01]). Dans le cadre de l'architecture CAEN, avec une densité de défauts élevée, [Gold01] s'intéresse aux techniques de l'organisation des nanoBlocks. Une nanoFabric est une matrice de nanoBlock connectés entre eux. Elle doit être reconfigurable et muni de mécanismes d'auto-test. Un nanoBlocks peut intégrer une fonction de porte logique. Il est composé de trois sections (cf. figure 18):

(1) la logique à diodes de matrice moléculaire

(2) les latchs, utilisés pour la restauration de signal, et la mise en oeuvre séquentielle circuit

(3) entrées/sorties, utilisées pour connecter le nanoBlock à ses voisins.

Il peut également servir de switch pour router les signaux dans le circuit. Un nanoBlock est directement connecté à ses quatre plus proches voisins. Comme dans les FPGA, les fils longs servent à interconnecter deux nanoBlocks lointains.

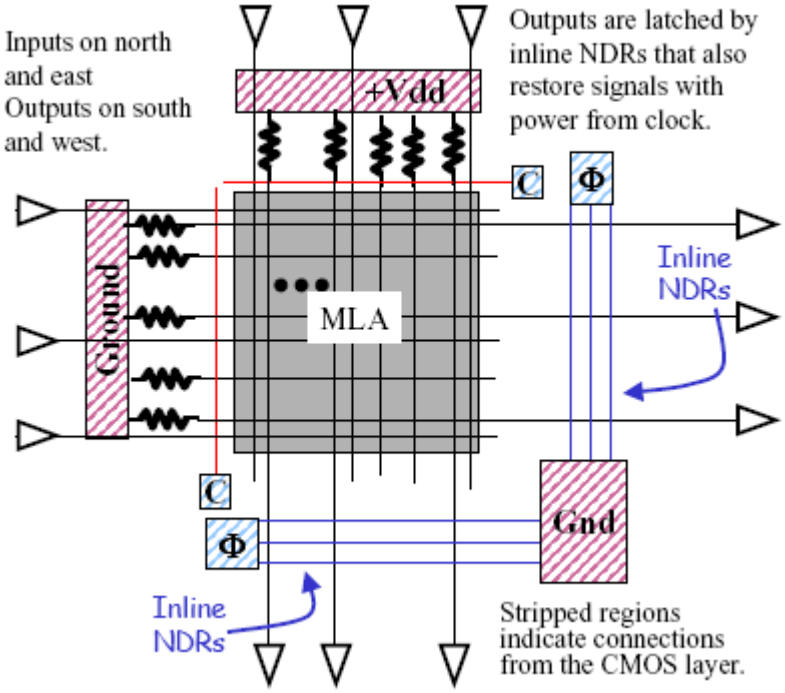


Figure 18. [Gold01] La description du nanoBloc. Les entrées sont situées à gauche et en haut du nanoBloc. Les sorties sont situées à droite et en bas de celui-ci.

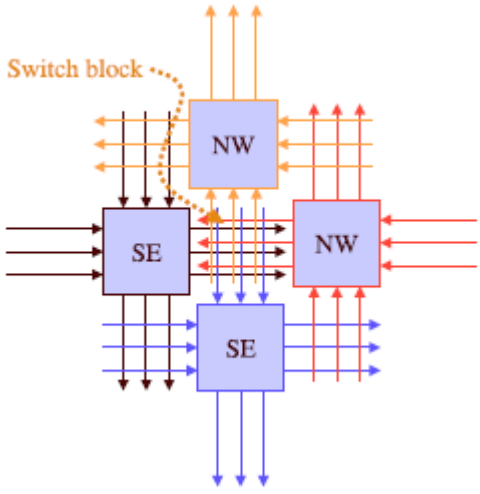


Figure 19. [Gold01] 4 nanoBlock connectés à travers d'un Switch block. Les nanoBlocks sont notes NW, SE, possède chacun 6 entrées et 6 sorties.

Il est alors indispensable de cartographier les défauts. Les défauts diminuent la densité de nanoBlocks utilisables. Pour adresser ce problème, les composants CMOS sont utilisés comme testeur, qui occupe une région sans faute (cela peut rester une problématique). Le testeur peut être dirigé par un circuit externe. Après

avoir testé un nombre suffisant de ressources, à leur tour, les nanoBlocks fonctionnels peuvent ainsi tester leurs voisins par une série de préconfigurations, et ensuite rapportent leur état au testeur. La deuxième phase commence par le téléchargement de la configuration dans la puce. Dans ce travail, l'architecture atteint déjà une complexité avancée.

Dans [Gold02], l'étude du latch est présentée plus en détail. En effet, tous les calculs effectués par la logiqe à diode devraient utiliser un latch pour restaurer le signal afin de le relayer à l'étage suivant. Le latch, ici, utilise le principe de la bistabilité de deux RTD moléculaires opposés montés en tête brèche. Cette structure peut alors se trouver dans un nanofil. Une simulation a été réalisée.

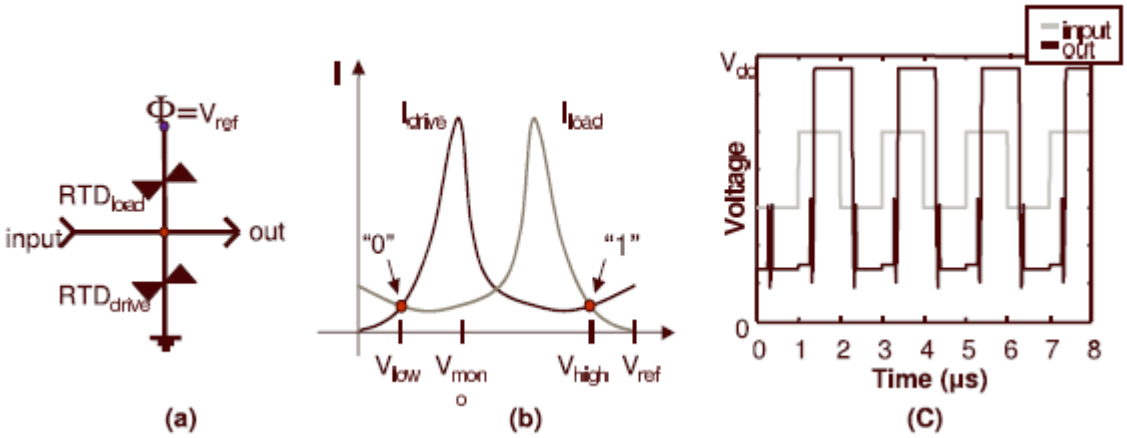


Figure 20. [Gold02], [Math99] a) circuit de latch RTD, b) chargment, bistabilité d'un RTD, c) simulations montrant la tension de latch et restauration

Un autre type de latch est étudié par [Kuek05], le concept serait particulièrement adapté à l'application au nanocomposant moléculaire dipôlaire. La molécule peut être utilisée comme un switch, ouvert ou fermé, suivant la tension au dessus du seuil appliqué. Selon la tension à ses bornes, si elle est positive ou négative, elle induirait une fermeture ou ouverture du switch. Si deux switch moléculaires sont montés en parallèlement inverse, pour la même tension d'entrée appliquée à la borne commune, la différence de potentiels à leurs bornes sera de signe contraire, à condition que les tensions des bornes opposées respectives aient la même polarité. Il en résulte à ce moment qu'un switch soit ouvert alors que l'autre soit fermé. Les fils (cf. figure 21) C_A et C_B serait l'horloge du latch traditionnel.

Cependant, ce type de restauration rencontre pour l'instant deux contraintes, la durée de vie limitée (100 s de cycles) et la fréquence de commutation à 0,1 kHz.

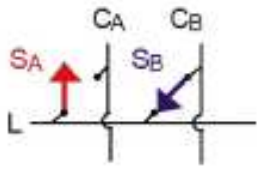


Figure 21. [Kuek05] Deux molécules sont situées en parallèle, dont le sens est inversé.

Quand on regarde précisément les caractéristiques de la molécule, il existe deux types d'ouverture et fermeture, l'un est inconditionnel, l'autre est conditionnel. Le premier nécessite une plus grande différence de potentiel aux bornes du nanocomposant par rapport au second.

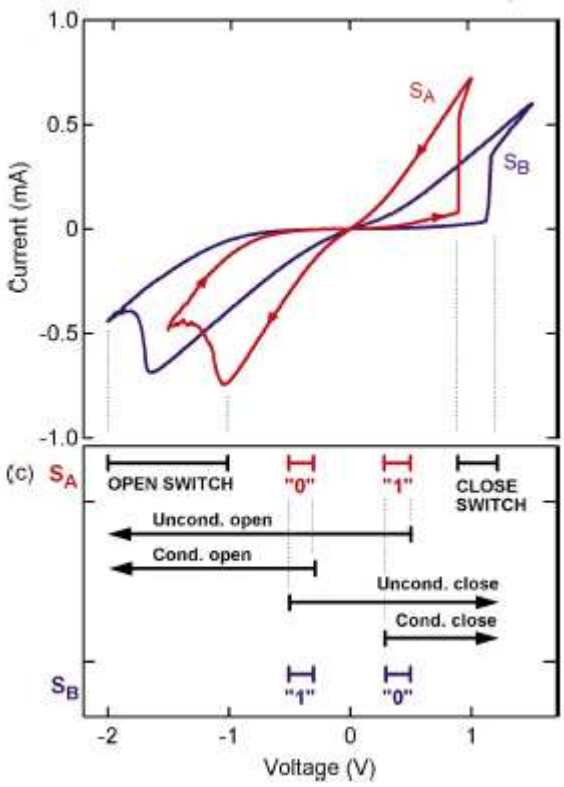


Figure 22. [Kuek05] Caractéristiques des switchs, ouverture et fermture, en mode inconditionnelle ou conditionnelle

L'utilisation du latch se déroule comme suit :

(1) l'ouverture inconditionnelle des switchs S_A et S_B se fait en appliquant une forte tension au dessus du seuil de configuration, avec une tension positive sur C_A et négative sur C_B. A ce stade, les deux switchs sont ouverts.

(2) appliquer une tension de fermeture conditionnelle de switch, négative sur C_A et positive sur C_B. A l'issue de cette opération, l'un des switchs est fermé en fonction du niveau logique de L (milieu).

(3) appliquer le niveau logique :

1 à ligne de C_A

0 à ligne de C_B

Il y a donc restauration de niveau logique correspondant au switch fermé.

En inversant le niveau logique sur C_A et C_B, la sortie est complémentée.

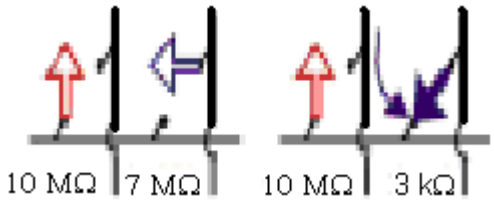


Figure 23. [Kuek05] A gauche, ouverture inconditionnelle. A droite, fermeture conditionnelle du switch bleu S_B. Le passage du signal va automatiquement fermer l'un des switch et ouvrir l'autre, parce que la différence de potentiel est de signe contraire aux bornes de S_A (en rouge) et aux bornes de S_B.

On peut revenir à l'architecture de [Deho05] pour la partie restauration du signal. Elle s'appuie sur la restauration par précharge à deux étages. Pour chaque étage, la précharge est désactivée lors de l'évaluation. La sortie Ainv (cf. figure 24) sera préchargée à 0 initialement (/prechargeA, prechargeB), et sera rechargé (pull-up) à Vcc si l'entrée Ainput est 0 faible. Si l'entrée bascule à 1 faible, la sortie sera laissée dans l'état précédent, à savoir préchargée à 0. L'inversion est réalisée de cette manière. Ce mode de fonctionnement nécessite l'alternance de deux phases (evaluation, préchargement). Le signal /eval est activé seulement après la précharge, et pendant l'évaluation. La bufferisation provient de la double inversion obtenue en interconnectant deux portes NOR en boucle. La précharge consiste à décharger la capacité du nanofil à travers la résistance de contact. La variante NAND existe également, réalisable suivant le principe de pull-down.

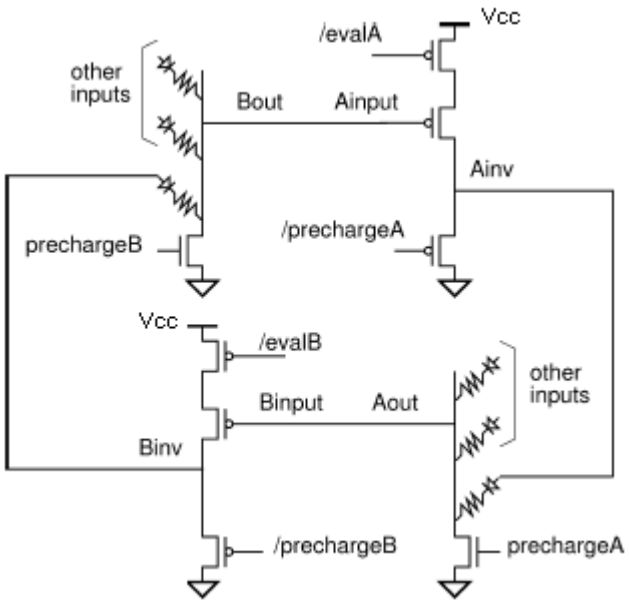


Figure 24. [Deho05], schéma de cycle de précharge, logique NOR-NOR

La fabrication Bottom-Up, en particulier dans le cadre de l'auto-assemblage, fournit un moyen pour construire un circuit. Une quantité importante de nanocomposants peut être sélectionnée à travers un décodeur d'adresse, puisque le nombre de lignes d'adresse (du décodeur) évolue linéairement (voire en $N.\log_2 N$) alors que le nombre d'éléments adressables croît en 2^N. Compte tenu de sa structure implémentée, la complexité du décodeur doit être recalculée. La sélection par nanofils a été évoquée dans [Deho05] [Will01]. À l'échelle moléculaire, il sera difficile d'envisager un adressage déterministe, du fait de l'absence de contrôle de la position des éléments fabriqués par la méthode Bottom-Up. On peut avoir recours à l'adressage stochastique. En effet, il est possible de faire croître au hasard un ensemble de particules d'Or entre les entrées et les nanofils (nanotubes) à l'échelle moléculaire. Les particules d'Or servent de contact d'interconnexion. L'ensemble des connexions à chaque nanofil agit comme un code de sélection. Dans ces conditions, l'espace d'adressage n'est ni continu, ni déterministe. Tous les nanofils adressables ont une adresse unique, mais une adresse peut désigner plusieurs nanofils ou aucun. Ce codage est aléatoire. Le test des codes sera nécessaire pour découvrir la correspondance entre chaque nanofil et son adresse. Une fois cette information découverte, la fonctionnalité peut alors être testée.

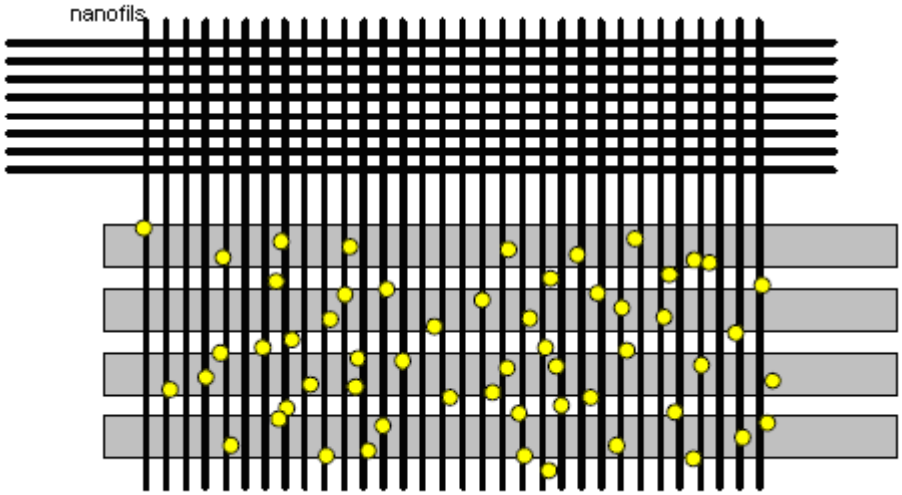


Figure 25. [Deho05] [Will01] Les nanofils (en haut du dessin) sont contrôlés par l'adressage stochastique.

L'interconnexion des nanofils peut également être réalisée par une méthode d'écriture directe [Stew07] pour former un point d'interconnexion vertical métal-métal, entre les nanofils métalliques (60 nm de diamètre). L'irradiation de faisceau d'électrons a été utilisée pour modifier la conductivité électrique d'un film polymère isolant de 23 nm d'épaisseur qui sépare les nanofils métalliques et les microfils du décodeur. L'isolant utilisé est le polymère poly-methyl-methacrylate (PMMA). La conductivité G de la jonction passe de $G < 10^{-11}$ S à $G > 10^{-6}$ S, ce qui correspond à une variation de 5 ordres de grandeur. Cependant, cette méthode requiert de forts courants de faisceau électroniques (des dizaines de picoampères) et un temps de programmation relativement long (des dizaines de secondes), ce qui ne permet pas d'envisager ce processus d'interconnexion dans la fabrication de circuits intégrés en grand volume. L'augmentation de la vitesse de programmation est réalisable avec une couche mince réduite de polymère au détriment d'une réduction de rapport R_{on}/R_{off}. L'emploi de nouveaux polymères sera à l'étude.

Toutefois, une technique d'optimisation d'adressage est proposée par [Stru05]. Chaque adresse est désormais munie d'un code correcteur d'erreur (Error Correction Code). L'auteur estime que la recherche exhaustive (en essayant toutes les combinaisons possibles) des nanofils est une approche « lourde » à cause d'un nombre important de nanofil, ce qui nécessiterait beaucoup de temps. La solution de balayage devrait non pas détecter les défauts nanofil par nanofil, mais par bloc. La matrice est divisée en sous blocs. Après avoir balayé les blocs, les blocs non

fonctionnels (dans lesquels, un nanofil ne répond pas) sont connus. Ainsi, à partir de leur emplacement physique, on peut dresser une liste de correspondance, entre l'adresse physique réelle et l'adresse virtuelle. Les adresses virtuelles sont contigües et la conversion d'adresses se fait sur un circuit CMOS. Le nombre de bits de l'adresse virtuelle est inférieur au nombre de bits de l'adresse réelle. Selon l'estimation, le module de code correcteur d'erreur (ECC) occupe une surface totale de 10% de la puce. Les calculs détaillés sont dans [Stru05].

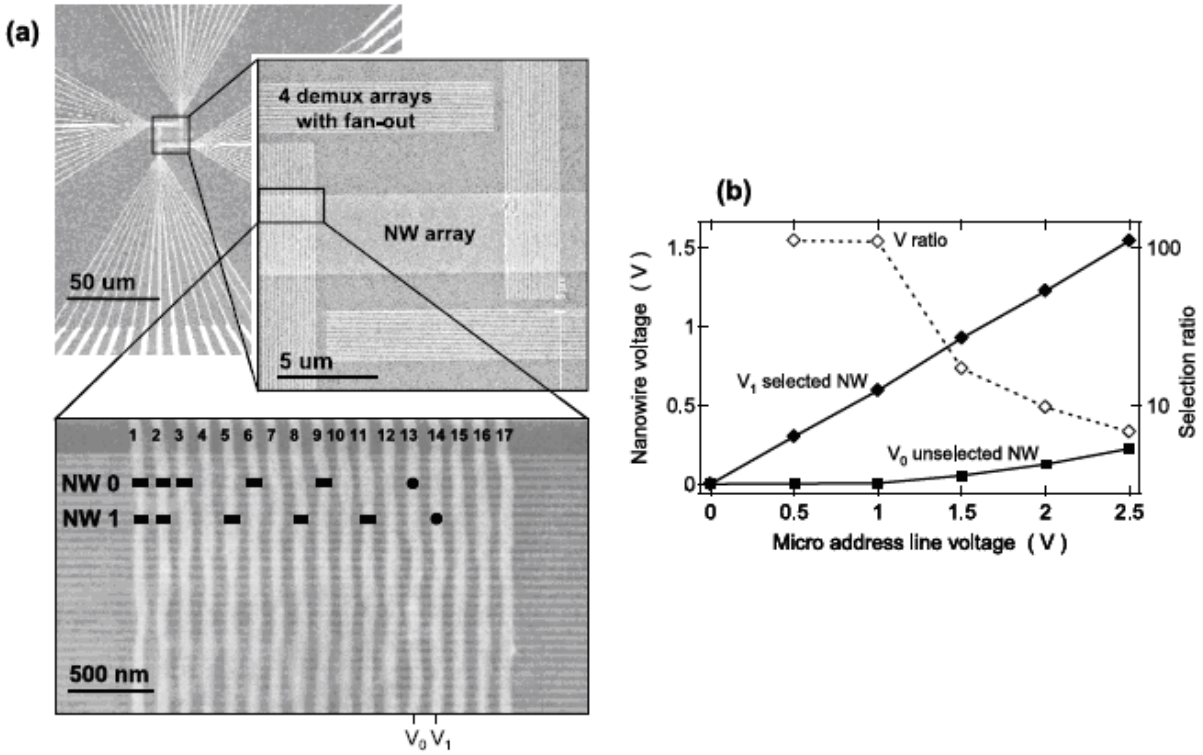


Figure 26. [Stew07] a) Photo par microscopie électronique, des nanofils (en horizontale) croisent des lignes de décodeur (en vertical). Les barres noires représentent les connexions. Pour sélectionner le nanofil NW0, les colonnes 1,2,3,6,9 doîvent être activées (à VDD). De même pour NW1, ce sont les colonnes 1,2,5,8,11. Les colonnes 13,14 servent de témoins. b) Les comportements de nanofils soumis à une tension externe imposée par le décodeur d'adresse. Le rapport de tensions entre le nanofil séléctionné et celui non séléctionné est de 100 pour une tension de commande inférieure à 1 V. Ce rapport a tendance à chuter pour une plus grande tension.

Le concept de l'architecture CMOL [Likh03] repose sur une couche de nanocomposants interfacés sur une matrice de nanofils connectés par des vias à un circuit CMOS classique. L'accès aux nanocomposants est assuré par l'intermédiaire d'interconnexions CMOS. En d'autres termes, les nanocomposants réalisent des

interconnexions entre les éléments logiques en CMOS. Cette partie permet le contrôle des nanocomposants qui se trouvent à l'intersection des nanofils de la couche supérieure. Les contacts sont de dimension large par rapport aux nanofils, ne pouvant pas être gravés côte à côte d'un pas égal ou inférieur à la séparation de deux nanofils, chaque contact de la partie CMOS doit se positionner sur un nanofil unique et décalé. L'arrangement repose sur une superposition de deux couches biaisées d'un angle α (cf. figure 27). Cependant, selon les auteurs, la disposition amène un meilleur rendement. (Les coupures de nanofils ont un impact moins sévère.) Le pas entre deux nanofils consécutif est appelé $2.F_{nano}$, et le pas entre deux points de contact de la couche CMOS est $2.\beta.F_{CMOS}$. L'auteur annonce que pour assurer le point de connexion (cf. figure 27) l'angle α doit respecter la relation arcsin $(F_{nano}/\beta F_{CMOS})$. La restauration entre les points de contacts se trouve au niveau CMOS. Le principe de l'activation de nanofil s'assimile à une porte AND, comme dans [Deho05], [Will01], [Stew07].

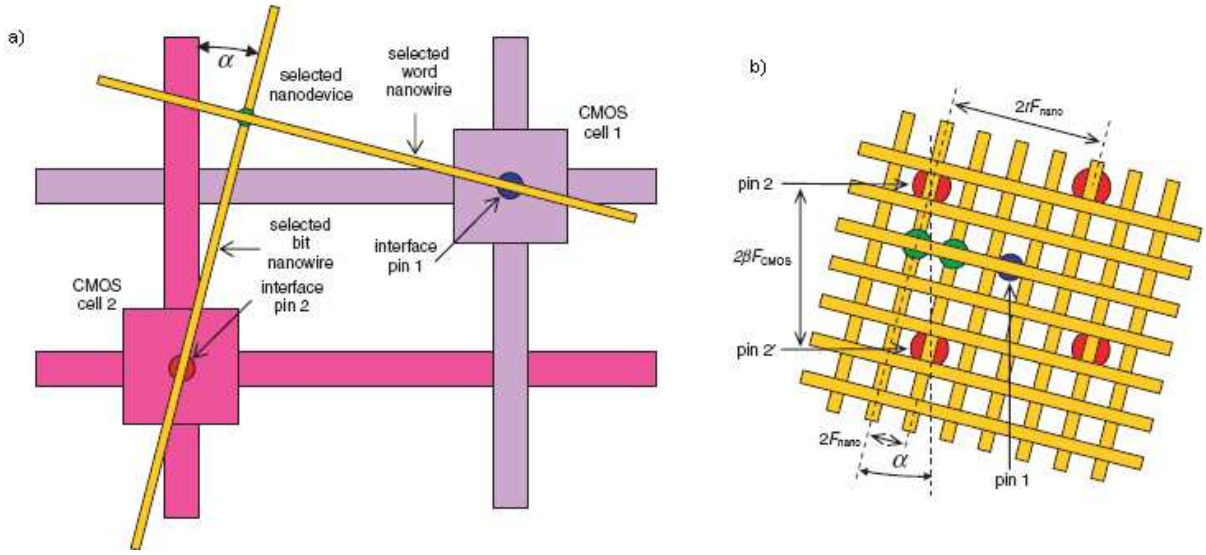


Figure 27. [Stru05-2] a) Vue de dessus, la couche des nanofils (en jaunes) est inclinée d'un angle α par rapport à la matrice de sélection (en violet). b) Les points de mémoires (en vert) sont dans le croisement des nanofils. Les colonnes (les points rouges) et les lignes (le point bleu) de sélection se trouvent dans la couche CMOS.

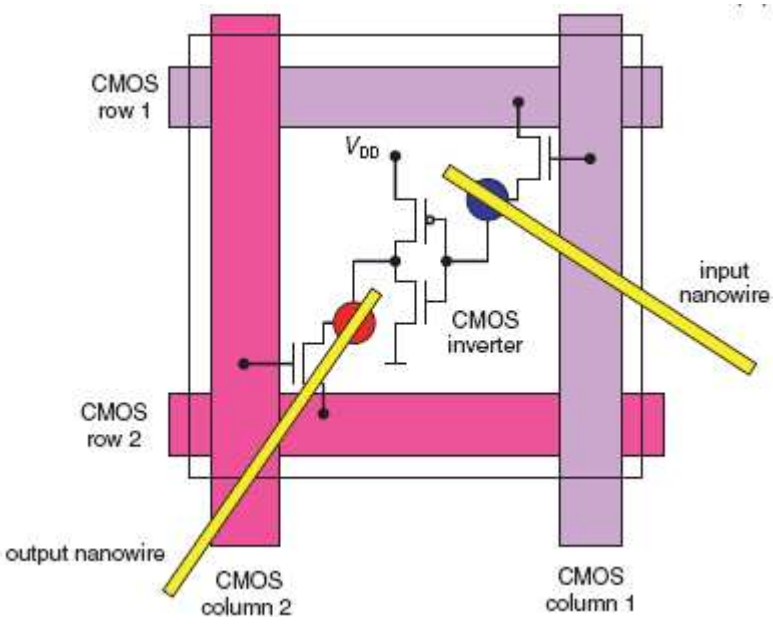


Figure 28. [Stru05-2]. La restauration de signal peut être fait au niveau CMOS

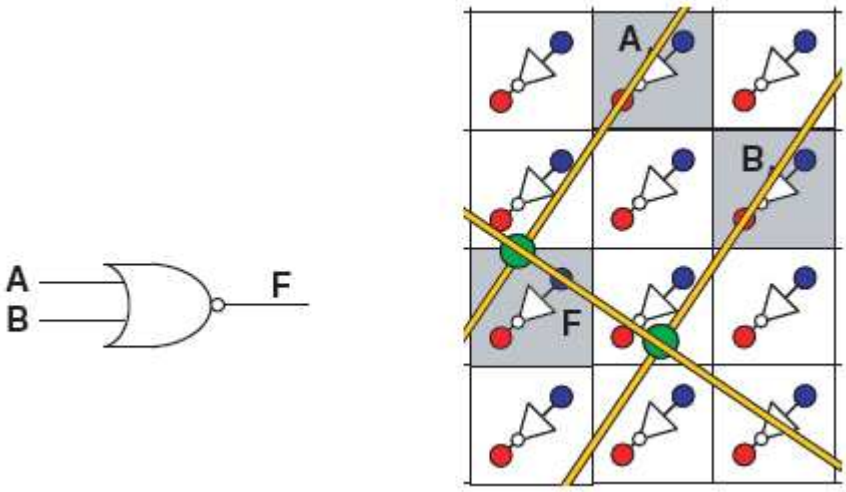


Figure 29. [Stru05-2]. La réalisation de la porte NOR, F=/A./B=/(A+B), la restauration est l'inverse de F au point rouge (3^{e} case de la première colonne)

La conception [Mori04], sous le nom de NASIC, est une structure similaire au PLA [Deho05]. La multiplication des signaux est utilisée dans cette structure. Elle permet une redondance interne aux portes logiques. Des copies redondantes de signaux sont envoyées aux nanofils (cf. figure 30).

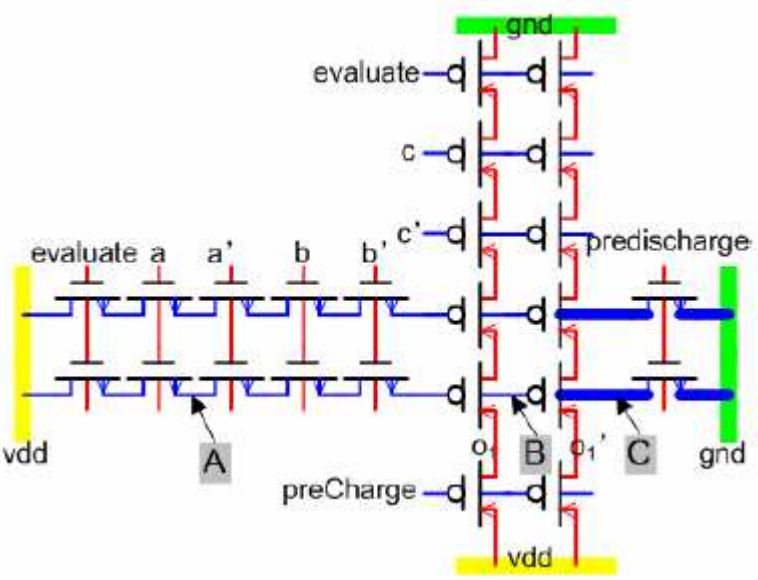


Figure 30. Le circuit implémente la porte logique ab+c, a', b', c' sont des copies respectives de a,b,c. Les points de défaut (A,B,C) du nanofil n'affecte pas le résultat final, l'évaluation est par pair ici, assurée par le nanofil adjacent.

Jusqu'ici les architectures sont de type matriciel. Cette organisation de mémoire n'est en réalité pas la solution unique pour le stockage d'information. Elle a le mérite de proposer une meilleure contrôlabilité des éléments répartis uniformément. Une autre méthode a été élaborée pour la configuration d'un réseau de molécules « NanoCell » [Tour02] dont la topologie interne est stochastique. Les portes logiques sont basées sur un circuit utilisant des éléments bistables de résistance différentielle négative (NDR). Les auteurs montrent des simulations d'algorithmes génétiques permettant de configurer les switchs moléculaires de façon à effectuer certaines fonctions logiques. Le problème de cette approche se trouve surtout dans l'hypothèse de l'omnipotence. En effet, les auteurs supposent que l'état de tous les switchs est accessible en lecture comme en écriture. L'intérêt du circuit est réduit parce que le nombre des éléments contrôlables est restreint par l'adressage qui pose de nombreux problèmes.

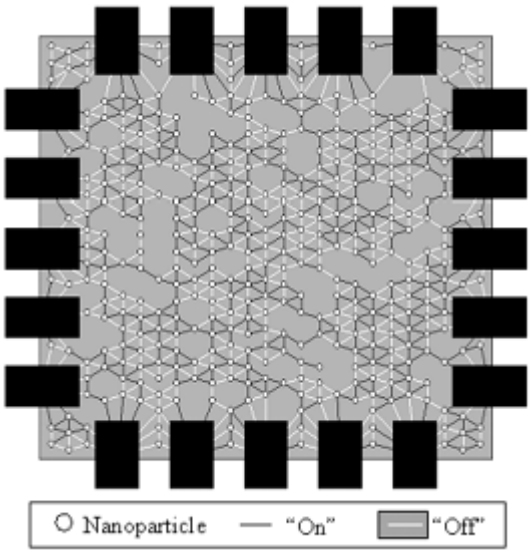


Figure 31. Puce constituée de nanocells interconnectés aléatoirement [Tour02]

Dans [Tour03] un prototype a été fabriqué. Il est constitué d'îlots métalliques désordonnés, et interconnectés par des molécules. Il peut fonctionner comme une mémoire ou un élément logique programmable. La mémoire de type moléculaire « switch » est caractérisée par une lecture destructrice, contrairement à la mémoire de type conductive. Les deux types de mémoire sont stables pour une durée de test d'une semaine et plus à la température ambiante. Un rapport de résistance de 10^4 à 10^6 a été observé après traitement par l'ozone, ce qui semble diminuer le courant de fuite. Le temps de rétention est de 11 jours. Cependant, les tentatives de réalisations expérimentales sont restées décevantes, au plus un effet latch ou bistable a été observé.

Nous avons constaté que les recherches sont d'abord parties de la gestion de la complexité des systèmes du fait du grand nombre de nanocomposants pour se focaliser sur les difficultés d'utilisation des nanocomposants. Nous avons évoqué l'utilisation de l'adressage, et la restauration de niveaux logiques dans un circuit. La poursuite de la recherche dans ce domaine passe d'abord par la conception électronique à base de nanocomposants. Avant d'aborder l'architecture et la conception de porte logique neuronale, qui constitue l'objectif de ma thèse, je vais introduire les réseaux de neurones.

6. Architecture Bio-Inspirée : les réseaux de neurones

- **A. Historique**

Tout au long du 20e siècle, suite à de nombreuses recherches pour comprendre le fonctionnement des réseaux de neurone biologiques à partir des observations neuropsychologiques, les chercheurs ont jeté les bases de ce qu'on appelle couramment l'apprentissage en intelligence artificielle en utilisant le modèle de calcul des réseaux de neurones formels. Ils ont pour but de trouver la relation synaptique entre neurones, et ce par une description mathématique plausible. En réalité, le fonctionnement du système nerveux, notamment la partie biochimique, est bien plus complexe que ce qui avait été imaginé par les chercheurs. Les premières publications expliquèrent les mécanismes d'association entre neurones par la stimulation conjointe répétée. La modification synaptique est sensible aux potentiels d'action mutuels (différence temporelle, force, type de cellule) des neurones connectés. Pour chaque stimulus présenté, en entrée d'un réseau, une sortie attendue peut être associée de façon automatique. La dynamique de l'apprentissage est vue comme une évolution cyclique et itérative.

Le principe de base de différentes architectures s'inspire du fonctionnement des neurones biologiques. Les algorithmes d'apprentissage y étaient associés. Historiquement, l'exemple le plus connu est le Perceptron [Ande98]. Ensuite il y a eu le réseau de neurones formels « Adaline » [Widr60] élaboré à l'Université de Stanford dans les années 1960. Il est basé sur le neurone de McCulloch et Pitts et sur l'apprentissage de la règle delta. La difficulté d'implémentation architecturale est liée à la réalisation des poids synaptiques. L'élément constituant de la synapse doit être une résistance non seulement réglable sur plusieurs niveaux, mais encore de taille suffisamment petite pour une meilleure scalabilité. Quoique non insurmontable, ces spécifications techniques ont néanmoins constitué une difficulté pour l'implémentation d'architecture électronique. Deux idées pertinentes voyaient le jour dans les années précédentes. La première propose de détourner l'implémentation analogique de la synapse multi-niveau. Cela revient à recoder les poids synaptiques en binaire. Par exemple, de nombreuses implémentations ont eu lieu sur les FPGA [Zhu03]. La deuxième consiste à utiliser un transistor à grille flottante [Bilo66], pour la

synapse analogique [Hasl95], or cette solution n'est pas totalement satisfaisante à cause de la surface utilisée.

La ressemblance entre le comportement des réseaux de neurones et les fonctions pseudo-logiques (activation, inhibition) a été évoquée dès les années 1940 [MC&P43]. Théoriquement le neurone de McCulloch et Pitts pouvait réaliser des calculs logiques élémentaires. Parmi les premières tentatives, le Perceptron est devenu le classifieur linéaire le plus simple [Rose58]. Il a été capable de reconnaître les chiffres à partir de leurs images numérisées. Inspiré par la théorie cognitive [Hebb49], l'auteur du Perceptron avait démontré les possibilités des réseaux de neurones formels. Plus tard sa limite potentielle fut mise en exergue. Cette machine a seulement une couche de sorties connectée synaptiquement à toutes les entrées. Dans cette configuration, les classifications réalisables sont limitées à celles qui séparent deux classes par un hyperplan dans l'espace vectoriel défini par les entrées. On parle alors des fonctions linéairement séparables. Les plus petites fonctions booléennes non linéairement séparables sont la fonction Ou-Exclusif (XOR) et son complément.

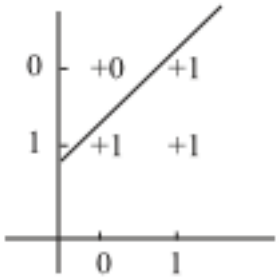
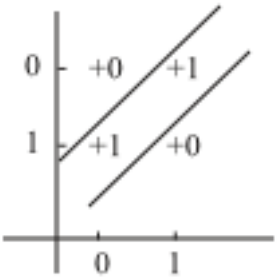
	
Fonction linéairement séparable, une droite suffit à séparer les deux classes	**Fonction non linéairement séparable, car il faut deux droites pour séparer les deux classes**

Figure 32. Fonctions linéairement séparable et non linéairement séparable

Les synapses furent réalisées avec des potentiomètres, réglables manuellement, puis plus tard par des moteurs asservis. Le réajustement de toutes les synapses pour développer une application est relativement peu pratique. Le concept devint désuet. Des années après, est apparue la vague des circuits logiques reprogrammables, des circuits PLD, puis les FPGA. Dans [Elia93], l'auteur montre qu'une similitude entre l'associativité des neurones et la logique programmable (PLA)

existe. L'analogie indique que les propriétés de cette classe de réseaux de neurones peuvent devenir une extension des propriétés de la PLA.

- **B. Algorithmes**

Les neurones sont reliés entre eux par des connexions pondérées, formant ainsi un graphe dont les neurones sont les sommets et dont les connexions synaptiques sont les arcs. Les états des neurones sont calculés selon les états de leurs voisins dans le graphe. L'influence des voisins du neurone sur celui-ci est appelée le potentiel post-synaptique Vi. Il est égal à la somme de leur état multiplié par les poids synaptiques respectifs.

$$V_i \in \Re$$
$$V_i = \sum W_{ij} X_j \qquad \textbf{(Eq. 1)}$$

Ensuite, le nouvel état du neurone est calculé par une fonction Ψ non linéaire ou d'activation à partir de cette somme pondérée, ou seuil. Les fonctions d'activation les plus fréquemment rencontrées dans la littérature actuelle sont :

La fonction sigmoïde (appelé logistique):

$$\Psi(V_i) = \frac{1}{1+\exp(-V_i)}$$

La tangente hyperbolique :

$$\Psi(V_i) = \frac{1-\exp(-V_i)}{1+\exp(-V_i)}$$

La Gaussienne :

$$\Psi(V_i) = \exp\left(-\frac{V_i^2}{2}\right)$$

Linéaire par moceaux :

$$\Psi(V_i) = \frac{1}{2}\left(|V_i + 1| - |V_i - 1|\right)$$

La fonction l'échelon de Heaviside :

$$\forall V_i < 0, \Psi(V_i) = 0$$
$$\forall V_i \geq 0, \Psi(V_i) = 1$$

La fonction signe :

$$\forall V_i < 0, \Psi(V_i) = -1$$
$$\forall V_i \geq 0, \Psi(V_i) = 1$$

Les études des réseaux de neurones artificiels ont amené à introduire un autre modèle de fonction d'activation. En effet, on veut se servir pleinement des propriétés des constructions mathématiques (transformation d'espace vectoriel) pour les réseaux de neurones, en particulier dans le domaine de la reconnaissance :

La fonction à base radiale (RBF) :

$\Psi(V_i) = \exp\left(-\beta(\sum(W_{ij}X_j - Ci))^2\right)$, Ci est une valeur de référence (centre)

Pour un problème, différents algorithmes d'apprentissage permettent de trouver les poids synaptiques. J'ai choisi d'évoquer quelques algorithmes d'apprentissage. Je commence ici par citer la règle de Hebb. Elle stipule que deux neurones excités conjointement voient le lien qui les unit renforcé. L'apprentissage du Perceptron exploitant cette règle fait converger les poids synaptiques vers une assignation correcte pour toute classification dite linéairement séparable. Cette règle ne tient pas compte de la réponse propre du neurone. En ce sens, la règle de Hebb fonctionne en boucle ouverte. Elle tend à maximiser la corrélation entre les potentiels post-synaptiques obtenus et les sorties attendues.

Soit la règle de Hebb :

$$W_{ij}(t) = W_{ij}(t-1) + X_i X_j \quad \textbf{(Eq. 2)}$$

X_i est la valeur de l'entrée du neurone i
X_j est la valeur du neurone j connexe à la sortie du neurone i
W_{ij} est le poids synaptique entre le neurone i et le neurone j

Dans la règle delta, on définit l'erreur comme la différence entre la sortie fournie par le réseau et la sortie désirée. Cette quantité est calculée par le réseau de neurone, pour ajuster les poids synaptiques. Ce fonctionnement en boucle fermée permet d'avoir une programmation synaptique rapide qui peut s'arrêter d'elle même. La règle delta ajuste les poids en fonction de la différence de réponse, en la multipliant par l'entrée X_i, et par un pas Δ. Ce produit est ensuite ajouté à la valeur précédente du poids synaptique.

Soit F la valeur de sortie d'un neurone et S la valeur désirée de la sortie. Pour chaque poids Wij, la nouvelle valeur est calculée par :

$$W_{ij}(t) = W_{ij}(t-1) + d(S_j - F_j)X_i \quad \textbf{(Eq. 3)}$$

d détermine le pas dans une étape d'apprentissage, cette quantification peut être variable en fonction de l'avancement de l'apprentissage.

Pour le Perceptron multi-couche, on peut utiliser l'algorithme de la rétro-propagation. C'est un modèle de réseau de neurone formel, dont le principe repose sur la propagation de l'erreur dans le sens inverse du sens normal. L'algorithme de la règle de rétropropagation peut être écrit sous cette forme :

Propagation dans le sens direct :

$$F_j = \Psi(\sum W_{ij}X_i)$$

Calcul de l'erreur depuis la couche de sortie :

$$\Delta_j = S_j - F_j$$

Propagation dans le sens inverse de réseau, par couches n successives :

$$\Delta_{j,n} = (\sum W^{T}_{j,n+1}.\Delta_{j,n+1}).f'(X_{j,n})$$

Mettre à jours les poids synaptiques :

$$\Delta W_{ij,n} = X_{j,n-1}.\Delta_{j,n}$$

Nous allons maintenant introduire la machine de Boltzmann [Ackl85]. D'abord la dynamique de la machine de Boltzmann est stochastique. La probabilité d'activation du neurone est définie par la relation suivante :

$$P(X_j = 1) = \frac{1}{1+\exp\left(\frac{-V_i}{T}\right)} \qquad \textbf{(Eq. 4)}$$

T est un paramètre réel strictement positif qui agit comme la température dans un recuit simulé. On peut alors observer une dynamique dans laquelle on voit décroître l'énergie du réseau, $E = \sum X_i^T.W_{ij.}X_j$.

La machine de Boltzmann fait intervenir un apprentissage reposant sur deux phases. La première phase dite libre (relaxation stochastique) fait évoluer l'état des neurones lorsque les sorties sont libres. La deuxième est la phase forcée, où les états de neurones de sorties sont imposés. Les deux phases s'alternent. Durant les deux phases, on mesure la co-activité des neurones connectés à chaque synapse.

$$C_{ij}^{+,-} = \frac{1}{N}\sum_{k=1}^{N} X_i X_j \qquad \textbf{(Eq. 5)}$$

L'apprentissage tend à rapprocher le fonctionnement du réseau en phase libre de celui en phase forcée. Pendant l'apprentissage la mise à jour de valeur de poids se fait par la soustraction des activités de co-occurrences entre les deux phases.

Soit Cij^+ la co-occurrence estimée durant la phase forcée et Cij^- la co-occurrence

estimée durant la phase libre, η le pas d'apprentissage :

$$W(t) = W(t-1) + \eta(C_{ij}^{+} - C_{ij}^{-}) \quad \textbf{(Eq. 6)}$$

- **C. Implémentation**

Par opposition à l'émulation numérique de réseaux de neurones avec calculs numériques, les véritables circuits électroniques analogiques de neurones artificiels tendent à imiter les comportements électro-physiologiques des neurones. Et ce à partir de différents niveaux, le premier modèle proposé par McCulloch-Pitt [Mc&P40] a situé le fondement des niveaux logiques en fonction du seuil d'excitation. Les réalisations électroniques s'en suivent, on peut commencer par mentionner SNARK [Ande98] [Beiu03], une première conception du genre, composée de 40 neurones. Suivi de l'architecture du Perceptron MARK I, 512 poids synaptiques ont été intégrés dans un réseau avec une couche cachée, totalisant 24 neurones. L'architecture ADALINE proposée par [Widr60] a intégré la règle delta pour l'apprentissage. Elle est connue pour la réussite de l'apprentissage, et la reconnaissance des chiffres. Fort de ce succès, ses auteurs ont fondé la société « Memistor Corporation ». Depuis 20 ans, les conceptions de neurones se font en VLSI. Le modèle de Hodgkin-Huxley a été exploité pour le comportement dynamique de la conductance ionique de la membrane de neurones. Ce type de circuits simule la dépendance de la tension de la membrane. Le modèle impulsionnel, appelé Integrate & Fire, est souvent utilisé. Il consiste à intégrer les courants d'entrée pré-synaptique, puis génère une impulsion de tension analogue à un potentiel d'action lorsque la tension atteint le seuil d'impulsion. On peut constater le principe de la conversion de tensions en courant selon son poids synaptique. La structure basée sur le modèle d'Axon-Hillock est proposée by Carver Mead [Mead89]. Elle constitue une référence dans le domaine, où le circuit LANN21 [Fusi00] est composé de 21 neurones dérivés de la structure a été fabriqué. Ce circuit a utilisé la technologie CMOS 1.2 µm, le die fait 1.5mm x 1.38mm. Ensuite, dans [Hasl06] en utilisant la technologie CMOS 0,35µm, 1000 neurones ont pu être intégrés. Dans [Wije08], la technologie CMOS 0,35µm est employée, et la cellule de base occupe un espace de 2800 μm^2 contenant un

neurone et ses buffers. Le neurone est adapté du modèle d'Izhikevich. Enfin, un circuit [Hynn07] de 7200 neurones a été fabriqué sur une surface de 11.5mm^2 avec la technologie CMOS 0,25µm. Une surface de 774 µm^2 est utilisée par le neurone. Par ailleurs, les circuits ont évolué vers d'autres applications ciblées. S'ils étaient conçus d'emblée pour la reconnaissance de forme, ils ont également pu montrer la première percée technologique en bioniques en se basant sur l'intégration VLSI (bien que le silicium soit toxique pour le corps biologique).

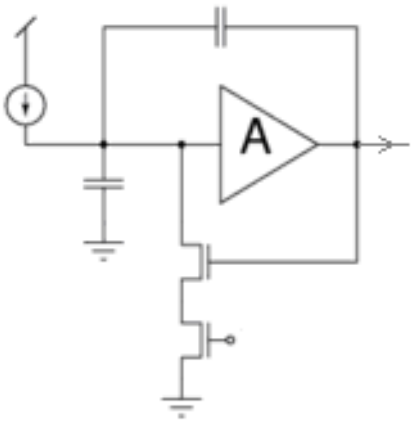


Figure 33. La structure du neurone simplifié peut être réalisée à l'aide des dispositifs analogues à ceux des implémentations VLSI [Mead89].

En ce qui concerne la nanoarchitecture, je vais reparler de l'architecture CMOL CrossNet. Dans [Lee07], le circuit est conçu pour les tâches de classification, ou de traitement d'image. Dans chaque Crossnet (cf. figure 34), les cellules de neurones, appelées "soma", sont implémentées dans le sous-système CMOS, tandis que les synapses sont en fait des matrices de switchs en nanocomposant. La densité importante des synapses par rapport aux neurones explique cette répartition. Les nanofils qui se croisent peuvent être assimilés aux connexions synaptiques entre axones et dendrites. Les signaux transitent entre les cellules. Par ce biais, la cellule est directement connectée à un nombre M d'autres cellules. Du fait de leur nature binaire, il faut intégrer plusieurs switchs en guise de poids fractionnaires pour réaliser une synapse réelle. Malgré ces limitations, les auteurs de CrossNets proposent de réaliser un apprentissage par un ajustement de poids synaptiques permettant d'effectuer l'étage de classification. Le réseau de neurone formé serait alors un Perceptron multi-couche en mode propagation directe. Selon [Gao07], le circuit peut alors avoir une densité de puissance inférieure à 0.64 W.mm^{-2}.

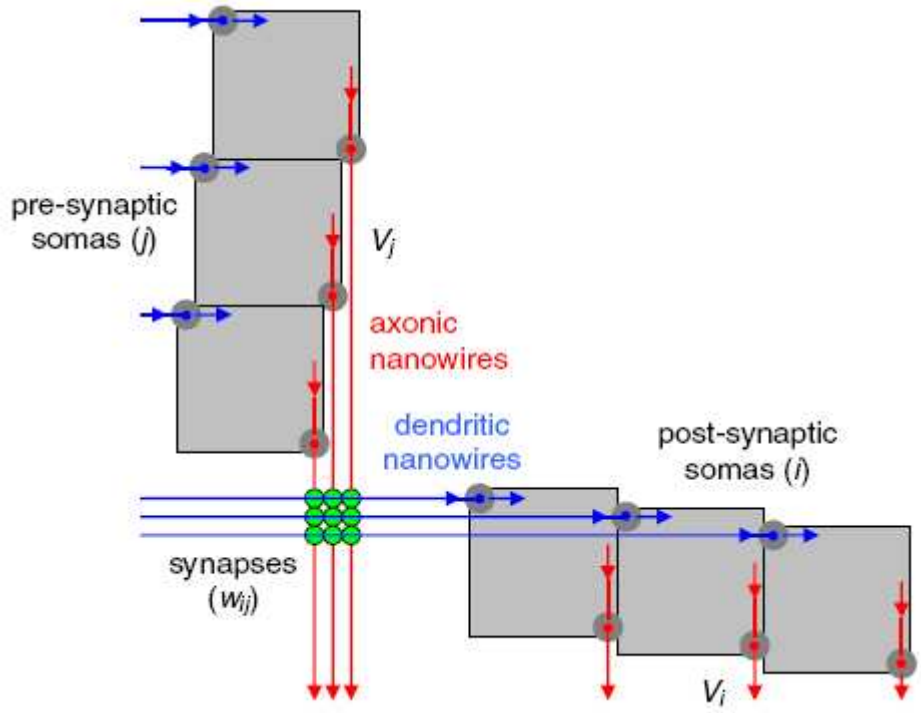


Figure 34. Structure de CrossNet. Les points en vert sont des synapses. Les nanofils bleus et rouges véhiculent les signaux en provenance des neurones. En bleu, la fonction post-synaptique, ou soma.

L'utilisation de la mémoire multi-niveau est justifiée pour une simple raison. Une plus grande plage de conductance variable permet l'apprentissage des fonctions différentes à entrées multiples. Quant au nanocomposant, la mémoire ionique s'insère dans la catégorie de mémoires à résistances variables. Ce point sera traité dans le chapitre 3.

7. Architectures « hypothétiques »

Sous l'hypothèse de la faisabilité, les architectures suivantes atteignent un niveau de conception plus élevé, à granularité variable.

Nous pouvons analyser l'architecture issue du contexte d'auto-assemblage guidé par l'ADN (l'Acide DésoxyriboNucléique) [Winf98], [Seem99], [Yan03]. L'ADN est connu pour être l'élément codant de la vie. Les auteurs [Patw06] proposent l'architecture « NANA », il s'agit d'un réseau actif constitué des noeuds

nanostructurés individuels. La taille des nœuds est limitée par la technologie de fabrication. Le taux de défauts est très variable, allant de quelque pourcents à plus de trois quarts. Dans cette architecture, les nœuds peuvent fonctionner indépendamment les uns des autres. Ils sont reliés par des canaux de communication. Le protocole de communication entre noeuds est spécifié par l'architecte. Les noeuds peuvent recevoir et émettre des paquets de données après les avoir traités ou non. Ils doivent se munir d'un module de communication. L'interconnexion aléatoire des noeuds peut induire à un chemin de donnés multiple, le protocole de communication doit la prendre en compte. Chaque nœud a une fonction particulière, calcul (nœud P) ou mémoire (nœud M). Le calcul est une opération propre à un type de nœud. Les modèles nanostructurés d'ADN sont l'échafaudage pour le placement et l'interconnexion des éléments actifs. Par le procédé ascendant, les noeuds sont reliés afin de créer un circuit complet. Le désordre et la désorientation des nœuds sont inhérents à ce genre de circuit. Ici, comme le noeud est de taille limitée, les opérations effectuées par celui-ci sont simplifiées. Différents types d'opérations sont répartis dans les nœuds P.

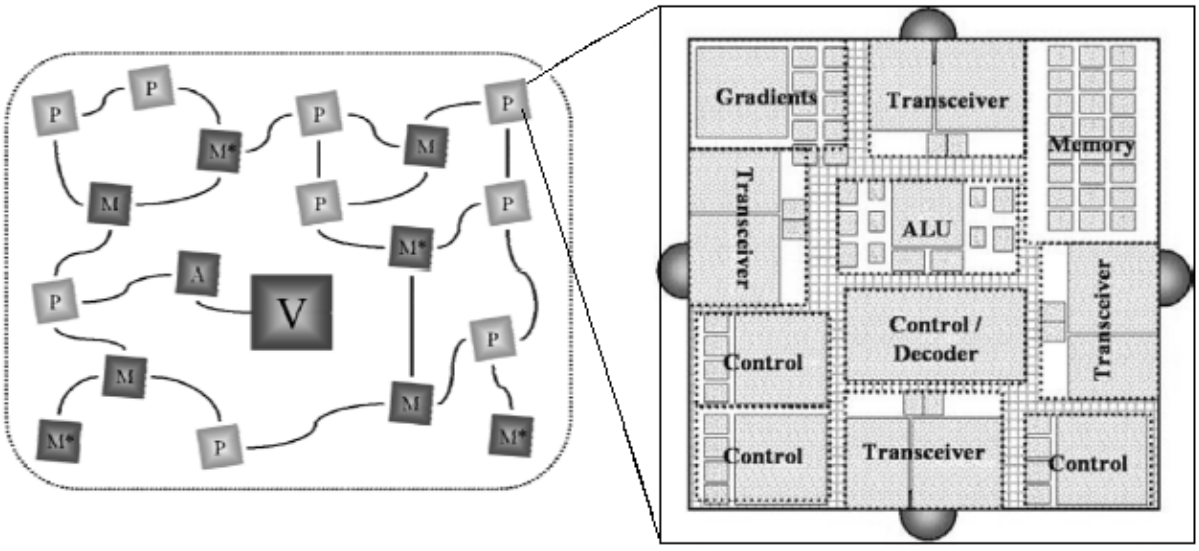


Figure 35. A gauche, un réseau de nœuds constitués de type P pour traitement de calcul, de type M pour stockage de mémoire, et de type V pour I/O. A droite, la composition du nœud P [Patw06]

Au niveau de la synthèse du système, une reconfiguration fondée sur la méthodologie de contournement de défauts est proposé [He07]. L'architecture des nanoFabrics utilise des critères de probabilité. Pour chaque fonction logique, un très grand nombre d'alternatives d'implémentation est réalisable. La configuration des nanoFabrics nécessite au préalable le test de chaque élément. La programmation s'effectue de manière flexible, elle se fait sur un ensemble de modules apte à effectuer une combinaison spéciale. Une nanoFabric peut implémenter un ensemble de fonctions spécifiques sans qu'elles soient faisables par un autre. L'ensemble des possibilités de configuration de portes logiques est enregistré. L'édification s'établit en graphes de chemin de données reconfigurable (Data Flow Graph), qui représente les dépendances de données entre un certain nombre d'opérations. Il s'agit de la synthèse de haut niveau. Dans la plupart des cas, la carte de composition finale n'est pas unique, ni connue d'avance. De manière empirique les auteurs montrent une proposition algorithmique permettant d'explorer efficacement la répartition des clusters suivant la complexité des fonctions.

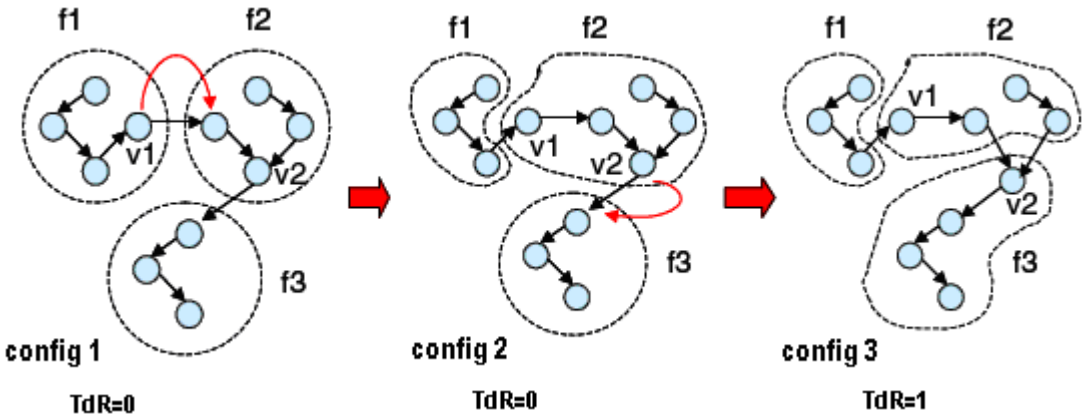


Figure 36. Repartitionnement de cluster DFG jusqu'à trouver le taux de réussite (TdR) égale à un [He07]

Quelque soit le niveau de conception, les nanocomposants peu fiables ont pesé sur l'importance de l'architecture tolérante aux défauts. Elle doit être fonctionnelle et robuste.

8. Conclusion

Dans cette étude bibliographique, de nombreuses solutions ont été présentées, elles ont donné des pistes pour résoudre le problème d'adressage des nanocomposants. Elles ont montré les possibilités offertes par des architectures basées sur l'interconnexion d'éléments désordonnés. Elles ont proposé des solutions qui s'adaptent à la technologie. Cependant, le plus souvent les détails de la conception ne descendent pas jusqu'au niveau de la simulation électrique. En outre, on remarque que les architectures publiées nécessitent souvent un long processus de test, ce qui est difficilement acceptable connaissant le nombre très important de nanocomposants à intégrer. Cette difficulté conduit à envisager des architectures neuro-inspirées.

En effet, dans le contexte d'une architecture de calcul logique, les réseaux de neurones formels apportent l'apprentissage de fonctions logiques, qui permet à priori de fusionner les processus de calibrage et test. Ce point sera traité dans le chapitre suivant.

CHAPITRE 3 : ARCHITECTURE & CONCEPTION

1. Introduction

Les systèmes de calcul issus des nanotechnologies devront surmonter les difficultés liées aux nanocomposants. On doit faire face d'un côté un taux de défauts élevé et des caractéristiques dispersées des nanocomposants, et d'un autre côté une importante résistance de contact, conduisant à un gain limité. Envisager le recours à des architectures alternatives tolérant ces désavantages est alors indispensable. Dans ce contexte, plusieurs propositions ont été étudiées dans le chapitre précédent. Parmi celles-ci, un objectif architectural neuro-inspiré, composé d'unité élémentaire de traitement de type neuronal me semble pertinent. J'ai porté toute mon attention sur l'exploitation de petits réseaux neuronaux, réalisés à partir des nanocomposants. En effet, ils peuvent simuler efficacement une Look-Up Table (LUT), et peuvent servir à effectuer des calculs. La synthèse de fonctions logiques est alors possible en utilisant le caractère de mémoire hétéro-associative des réseaux neuronaux, programmables par apprentissage.

La connaissance des caractéristiques des composants est primordiale. À partir de celles-ci, j'ai dressé un tableau comparatif des différents composants susceptibles de servir de base à la constitution d'une synapse. Puis j'ai modélisé le comportement au niveau électrique de la synapse, en tant que résistance programmable multi-niveau réversible. La partie active utilise le modèle compact de nanotube de carbone semiconducteur. La mise en oeuvre de la cellule s'appuie sur la conception détaillée des éléments clés de l'architecture. S'agissant du processus d'apprentissage de fonction par blocs élémentaires, l'algorithme supervisé de la règle delta est utilisé pour la configuration synaptique. Mes travaux ont tenté de mettre en évidence l'intérêt d'utiliser un réseau de neurones minimal pour simuler les fonctions logiques de base. Je vais traiter l'architecture dans ce qui suit. Puis je vais décrire les modèles compacts de simulation. Enfin, je vais expliquer la simulation globale de l'apprentissage d'une fonction logique.

2. Architecture en reconnaissance

La réalisation analogique de RNF nécessite l'intégration de synapses, et de neurones. Les synapses sont des composants linéaires. Pour autoriser l'apprentissage, nous avons choisi d'exploiter des conductances ajustables, multi niveaux, et non volatiles si on veut une rétention à long terme. Les neurones, éléments de décision non linéaire sont réalisables grâce aux éléments actifs de type transistor à effet de champs. Visant l'implémentation d'opérateurs Booléens il s'agit d'un simple seuil.

Un neurone de seuil dont l'état X_s est constant, toujours à 1, est ajouté en entrée de la matrice. Cette entrée invariable permet de remplacer le seuil -t par un poids synaptique W_s, afin de déplacer le seuil de décision -t de la fonction d'activation. La somme pondérée est obtenue en prenant en compte le seuil t. Le potentiel post-synaptique V_j de l'unité *j* est calculé par :

$$V_j = \sum_{i=1}^{m} X_i W_{ij} + X_s W_s \quad \textbf{(Eq. 7)}$$

Avec :

$t = W_S = W_S.X_S$

W_{ij} : poids synaptiques entre le neurone i et le neurone j

Le neurone est à proprement parlé une structure de seuil. C'est cette réaction fondamentale qui nous intéresse ici. La comparaison du potentiel post-synaptique à un seuil d'activation donne l'état du neurone. Les réalisations des portes TLG (Threshold Logic Gates) en VLSI sont nombreuses [Beiu03]. Les fonctions de seuils sont définies simplement comme ci-dessous :

$$\Psi(V_i) = \begin{cases} 1 \text{ si } V_i \geq 0 \\ 0 \text{ si } V_i < 0 \end{cases}$$

L'amplification du signal est nécessaire pour la propagation et la restauration du signal. Celui-ci vise à interconnecter de façon à ce que la sortie d'un neurone puisse être aussi l'entrée d'un ou plusieurs autres neurones.

Le nouvel état du neurone est donné par :

$$X_i = \Psi(V_i)$$

X_i : état du neurone i

Ψ : fonction d'activation (non linéaire)

L'architecture de réseaux de neurones que nous avons envisagée vise à tirer parti des hautes densités d'intégration projetées pour les réseaux de nanocomposants. L'architecture la plus immédiate qui en découle est organisée autour d'une matrice synaptique de nanocomposants dans laquelle les synapses se trouvent à l'interconnexion des lignes et colonnes. Comme nous visons la réalisation de LUT, les entrées et les sorties de la matrice sont Booléennes. La réponse du réseau est le produit de la matrice synaptique et du vecteur d'entrée.

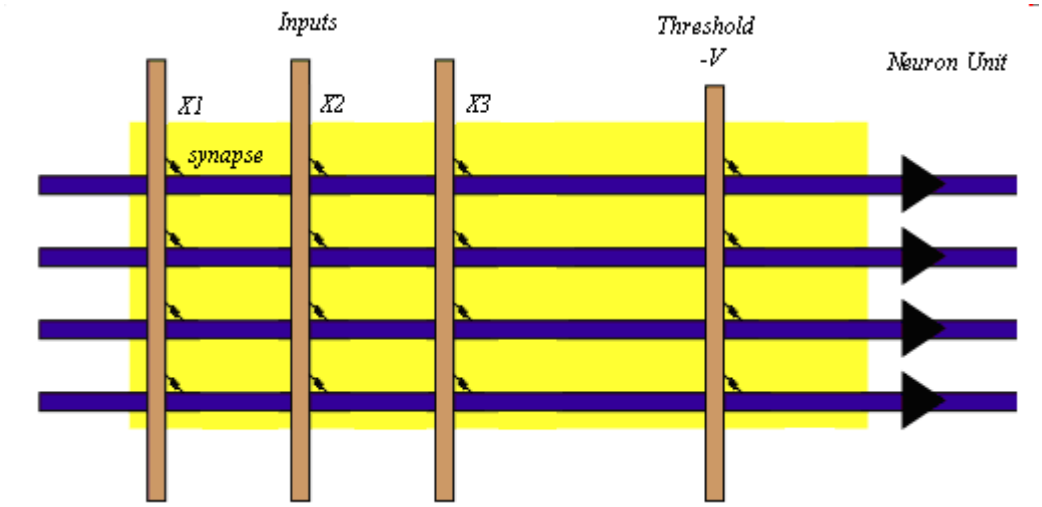


Figure 37. [Widr60] Matrice de synapses, où l'état de neurone est codé en unique rail. Les entrées (colonnes) sont connectées aux neurones (à droite) à travers les synapses se trouvant à l'intersection des lignes. Une tension de seuil –V est ajoutée pour réaliser la somme post-synaptique.

Les poids synaptiques W_{ij} sont signés alors qu'un élément passif ne peut pas simuler une résistance négative. Un élément actif peut simuler une synapse de valeur négative. Une conductance peut être de valeur négative par l'amplification NIC (Negative Impedance Converter). Mais cette solution conduit à l'utilisation d'un amplificateur opérationnel. Sa surface est imposante, et ne convient pas à réaliser

une synapse de dimension nettement plus petite, soit je table sur 10 à 1000 fois inférieure au neurone. Le réseau de neurones réagit aux stimuli d'entrée. Selon le stimulus, sa réponse est différente. Son comportement peut être assimilé à une porte logique, et il peut changer selon l'apprentissage imposé. Quelles sont les fonctions réalisables avec seulement des résistances passives ? Pour répondre à cette question, parcourons l'ensemble des fonctions logiques, il existe en effet une famille de fonctions logiques qui n'ont pas recours à l'utilisation d'une vraie somme pondérée impliquant des poids synaptiques négatifs. Il suffit pour cela que le seuil implémenté soit négatif. L'ensemble des fonctions de cette catégorie pouvant disposer uniquement de poids positifs est appelé l'ensemble unaire dans [Zhan04]. Pour cette catégorie de fonction, le réseau de neurones ne réagit qu'aux entrées actives (non nulles). La structure de matrice de synapses a été décrite dans l'architecture Adaline [Widr60].

Récapitulons par le tableau ci-dessous, les fonctions à deux entrées, implantables en ne recourrant qu'à des poids synaptiques de signe positif. On peut compter 2^4=16 fonctions binaires pour l'ensemble à deux entrées. Seulement 4 parmis elles entre dans la catégorie des poids synaptiques uniquement positifs. Les 12 autres fonctions ne peuvent pas être mises en œuvre par ce qu'elles nécessitent l'inversion quelconque d'une entrée.

	W_1	W_2	W_{s-}	
Y=X1	1	0	0	Fonction de désignation
Y=X2	0	1	0	"
Y=X1+X2	1	1	0,5	Fonction associative OU
Y=X1.X2	1	1	1,5	Fonction de condition ET

Tableau 7. Fonctions unaires (X1, X2) à 2 entrées

Avec :

W_{1o}, poids synaptique connecté au neurone de l'entrée X1 et la sortie

W_{2o}, poids synaptique connecté au neurone de l'entrée X2 et la sortie

W_{s-}, poids synaptique connecté au neurone de seuil, rail en tension négative

X1, l'entrée 1 de la fonction

X2, l'entrée 2 de la fonction

O, la sortie

Y, la sortie de la fonction, résultant de l'interaction de X1 et X2.

En règle générale pour N entrées il existe un ensemble de $2^{(2^N)}$ fonctions logiques. Nous nous penchons maintenant sur les fonctions à trois entrées, implantables en ne recourrant qu'à des poids synaptiques de signe positif, résumé par le tableau ci-dessous.

	W_1	W_2	W_3	W_{s-}	
Y=X1	1	0	0	0	fonction de désignation
Y=X2	0	1	0	0	"
Y=X3	0	0	1	0	"
Y=X1+X2	1	1	0	0,5	fonction associative OU
Y=X2+X3	0	1	1	0,5	"
Y=X1+X3	1	0	1	0,5	"
Y=Maj()	1	1	1	1,5	fonction de majoritaire
Y=X1.X2.X3	1	1	1	2,5	fonction ET à trois entrées
Y=X1.X2	1	1	0	1,5	fonction de condition ET à deux entrées
Y=X1.X3	1	0	1	1,5	"
Y=X2.X3	0	1	1	1,5	"

Tableau 8. Fonctions unaires (X1, X2, X3) à 3 entrées

Avec :

W_{3o}, poids synaptique connecté au neurone de l'entrée X3 et la sortie

X3, l'entrée 3 de la fonction

Ici, 2^8=256 fonctions binaires constitue l'ensemble à trois entrées, seulement

onze entre elles sont à poids synaptiques uniquement positifs. Certes, ces fonctions logiques ont l'avantage de n'utiliser que des poids synaptiques de valeurs positives. Néanmoins les poids uniquement positifs ne suffisent pas à la réalisation de l'ensemble des fonctions logiques, notamment celles nécessitant le complément logique d'une donnée. Dès lors, on peut envisager l'utilisation d'une représentation différentielle (positive et négative) du signal d'entrée et d'une paire de résistances pour chaque synapse, l'une représentant les poids positifs et l'autre les poids négatifs. Cela permet de simuler un vrai produit synaptique signé, et ainsi réaliser l'ensemble des fonctions logiques.

J'ai pris l'exemple de l'implémentation Alu4 de LGSynth'91 pour se rendre compte de l'efficacité de l'approche. Ce module comporte 14 entrées et 8 sorties. L'architecture est composée de blocs identiques. La méthode de la décomposition des fonctions quelconques en fonctions de seuils (TLG) est réalisée suivant [Zhan04]. Le nombre de neurones regroupés est identique pour tous les blocs, en fait chaque bloc est un ganglion. Pour cela, j'ai envisagé 5 cas concernant les matrices de synapses.

Type de structure	PLA (AND-OR) 2x2	PLA (AND-OR) 3x3	PLA (AND-OR) 4x4	PLA (AND-OR) 6x6	PLA (AND-OR) 8x8
Niveaux intermédiaires	36	29	29	24	24
Nœuds	1166	924	759	488	491
Type de structure	Matr. Synap. 2x2	Matr. Synap. 3x3	Matr. Synap. 4x4	Matr. Synap. 6x6	Matr. Synap. 8x8
Niveaux (couches) intermédiaires	36	27	27	22	21
Nœuds (Neurones)	1166	819	578	388	405
Réduction Matr.Synap./PLA	0 %	11,4 %	23,8 %	20,5 %	17,5 %

Tableau 9. Comparaisons entre PLA (AND-OR) et matrices de synapses

En comparant les résultats dans le tableau 9, on constate que le nombre de nœuds et le nombre de niveaux intermédiaires sont moins importants avec l'implémentation de la matrice synaptique multivalué par rapport aux PLA en bits binaires. Le nombre de nœuds réduit occupe un espace moins important. Le nombre de niveaux intermédiaires réduit permet d'avoir un temps de propagation plus courte. Elle démontre ainsi l'avantage dans l'utilisation des poids synaptiques multivalués par rapport aux connexions bistables des PLA. Considérons le routage des blocs,

plus exactement la taille de l'interconnexion interblocs, nous pouvons voir les résultats issus des simulations, au tableau 10 (en réalité, indépendamment de la technologie de l'implémentation de la LUT). J'ai pris l'exemple de s1423.bench d'ISCAS89. Dans les architectures modélisées les LUTs possèdent chacun 4 entrées. En variant le nombre de LUT par bloc (cluster) et le nombre d'entrées, la taille de l'interconnexion a changé globalement. On peut alors constater que le délai de propagation critique est diminué de 46 % lorsque ce nombre passe de 4 LUTs / bloc à 32 LUTs / bloc, en effet le temps de propagation d'intrabloc est généralement plus court que celui d'interbloc qui dépend de la longueur variable des fils. En même temps, la surface de réseaux d'interconnexion est diminuée de 15%, car les fils d'interconnexion intrabloc plus courts ont compensé ceux d'interbloc. Les paramètres de simulations pour le routage selon [Betz97] sont les suivants :

Rmetal.Cmetal: 337e-15 s

Switch Tdel: 456e-12 s

T_subblock T_comb: 546e-12 s, T_seq_in: 845e-12 s, T_seq_out: 478e-12 s

Structure de bloc	32 LUT / 32 entrées	16 LUT / 16 entrées	8 LUT / 16 entrées	8 LUT / 8 entrées	4 LUT / 8 entrées
Nombre de blocs / topologie	7 / 3x3	16 / 4x4	28 / 6x6	38 / 7x7	57 / 8x8
Nombre de canaux de routage	8	10	14	16	18
Largeur de canaux de routage (Channel Width)	19	17	13	11	10
Surface de réseaux d'interconnexion (unité L^2)	152 L^2	170 L^2	182 L^2	176 L^2	180 L^2
Délai de propagation critique (e-08 s)	2.104	2.868	3.011	3.433	3.915

Tableau 10. Réseaux d'interconnexion

La figure 38 décrit l'approche de la conception de crossbar pour la réalisation de la matrice de synapses. Elle est constituée des entrées en fils verticaux, chaque entrée est représentée par deux signaux, elle et son inverse. Les poids synaptiques se trouvent alors à l'intersection des fils horizontaux et verticaux. La première ligne horizontale VA est issue de la sommation post synaptique des entrées X1, X2, X3 et le seuil V avec leurs poids respectifs RA1+/RA1-, RA2+/RA2-, RA3+/RA3- et RA+/RA-. La deuxième ligne VB est une autre sommation post-synaptique, l'équivalent d'une 2^e LUT, des mêmes entrées avec leurs poids respectifs différents

RB1, jusqu'à RB3. Ainsi de suite, nous obtenons un ensemble de réponses neuronales. Elles sont restaurées par chaque neurone connecté (à droite). Les signaux seront ainsi relayés vers la prochaine matrice synaptique. Dans mon architecture, j'ai intégré un module d'apprentissage suivant la logique de programmation. Se référant à [Gold01] et [Deho05], l'apport essentiel de mon travail est la mise en application de la reconfiguration intelligente des points mémoires dans le crossbar, ici les synapses.

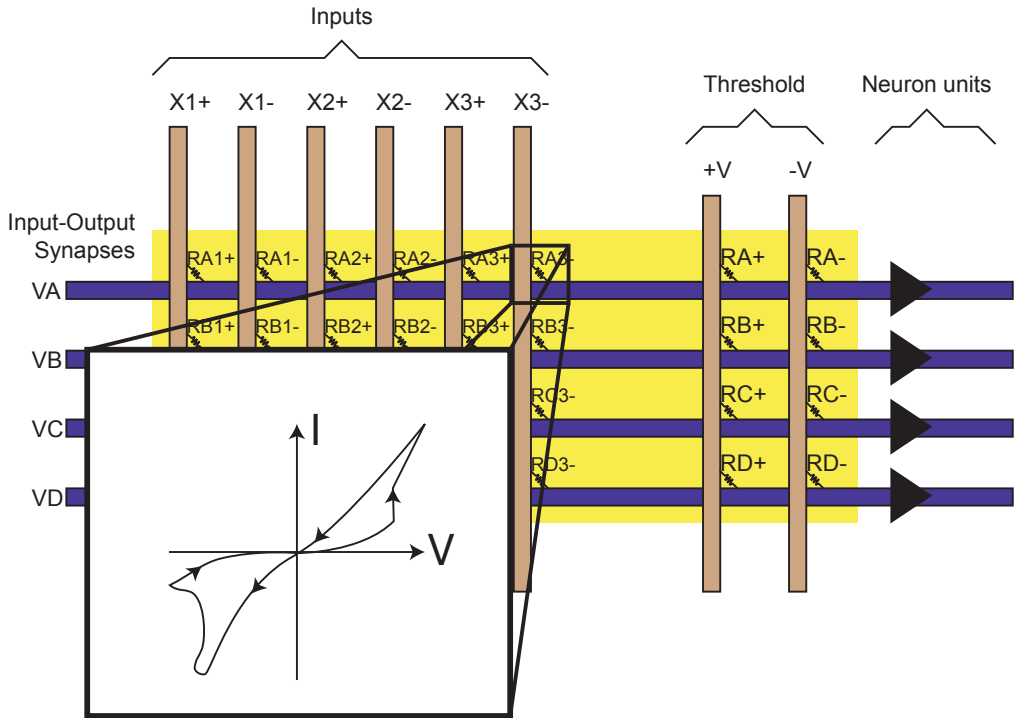


Figure 38. Matrice de synapses, les signaux en mode dual rail

3. Choix des composants

Avant d'aller plus loin, il est nécessaire d'investiguer les nanocomposants présentant des caractères compatibles avec la spécification de l'architecture. On peut se référer à l'état de l'art en étudiant les publications actuelles. Nous avons choisi pour composant actif d'amplification le transistor à nanotube de carbone, du fait de ses propriétés semi-conductrices intéressantes d'une part, et de la disponibilité [Preg06], [Mane06] du modèle compacte de simulation d'autre part. Le deuxième choix concernant le nanocomposant présentant des niveaux multiples de résistivité programmable reste délicat. De plus, il est indispensable d'étudier la compatibilité entre les nombreux éléments susceptibles d'implanter cette fonction et

les CNTFET.

Les nanocomposants actifs fonctionnent aujourd'hui dans la plage des microampères. Il est alors préférable d'y associer des résistances ajustables nécessitant un courant de programmation que peut débiter un élément actif. En effet, multiplier le nombre de nanotubes pour obtenir plus de courant revient à augmenter drastiquement la surface d'implantation réduisant ainsi la densité globale d'intégration. Ce n'est pas une solution idéale. Il faut donc rechercher des résistances multi-niveaux dont le seuil de programmation est compatible avec l'utilisation d'un CNTFET. On veut trouver un nanocomposant qui possède une caractéristique I(V), présentant plusieurs hystérésis programmables, similaire à la figure 39. Elle se caractérise par une plage de tension pour laquelle il n'y a pas de variation de résistance. Dans la plage de configuration, la variation de résistance est contrôlable par sauts.

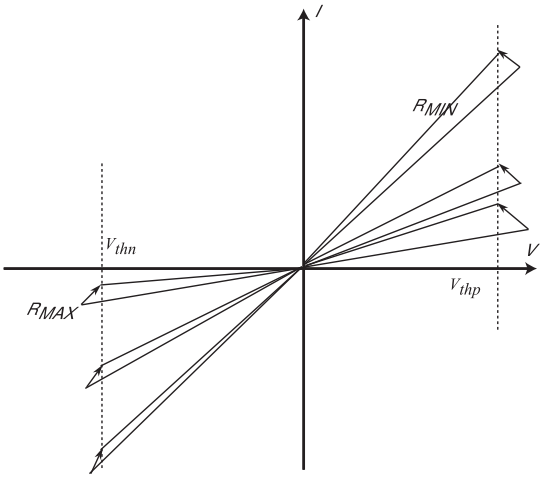


Figure 39. L'hystéresis du nanocomposant à résistance variable multi-niveau

Comme les phénomènes physiques à l'origine de la variation de conductance dépendent des technologies utilisées, les densités de courant et les tensions de programmation sont différentes. Le courant de programmation doit être contrôlable à des échelles plus fines pour parvenir à une modification par pas des niveaux de résistance électrique de nanocomposant. Une grande attention devrait être accordée aux possibilités de la mise à l'échelle, à la non-volatilité, à la fiabilité à long terme, et à la température de fonctionnement. Tous ces facteurs contribueront à faire le meilleur choix parmi ces nanocomposants pour la mise en oeuvre des poids

synaptiques implémentés dans l'architecture.

Les principales caractéristiques des nanocomposants permettant la réalisation de conductance multi-niveau sont résumées dans le tableau suivant :

Technologies	Seuil de prog. Current/Tension	Plage de la résistance	Plage de la transconductance
CNTFET with ALD gate dielectric [Zavo07]	+2.5 V / - 2.5 V		0.1nS / 2 µS
NW Hybrid Field Configurable Transistor [Lai08]	+5 V / -5V		0.25µS / 4.5 µS
CBRAM(Ag) [Kund05]	-100µA / 100µA	1 kΩ / 1 MΩ	
PMC(Ag/Cu-GeS) [Kozi05]	-250mV / 450mV	50 Ω / 1 kΩ	
PMC(Ag-As2S3) [Stra06]	- 2V / +120mV	10 Ω / 1 kΩ	
Multiwall nanotube [Coll01]	0,2mA/1,9 mA	630 Ω / 8,5 kΩ	
Organic switch Monolayer(C20) [Stew04]	-4 mA / 4 mA	250 Ω / 2,1 kΩ	
PCM (chalcogenide) [Rao07]	-600µA / 600µA	1 kΩ / 1 MΩ	
SrTiO3, SrZrO3 oxide metal de transition [Karg07]	100 µA / -100µA	10 kΩ / 500 kΩ	

Tableau 11. Huit familles de technologies de synapses

Pour choisir la technologie adéquate pour la synthèse de synapse, nous devons d'abord regarder ses caractéristiques plus en détails. Nous pouvons exclure les nanocomposants non réversibles quant à la variation de conductance, comme les nanotubes multiparois à effet de claquage. Dans ce type de dispositif, les parois peuvent être progressivement détruites, de façon irréversible, par des impulsions de courant. En première analyse, le switch moléculaire en monocouche paraît adapté du fait de ses possibilités de fonctionnement incrémentale. Malheureusement, son courant de programmation démontré à ce jour est trop important (0,8mA par [Kuek05], 4mA dans [Stew04]). On devrait également abandonner les conductances multi-niveaux dont la plage de configuration est restreinte. On peut enfin s'intéresser aux mémoires ioniques, ou plus particulièrement aux CBRAM (Conductive Bridge

RAM). Elles offrent de multiples avantages. Deux décades de résistivité programmable ont été démontrées par l'expérience [Kund05], tout en utilisant un courant de programmation faible. En outre, elles permettent d'atteindre un temps d'écriture particulièrement rapide, de l'ordre de 10 ns. En comparaison, le temps d'écriture des PCRAM (Phase Change) est de l'ordre de 100 ns. Celui des OxRAM (mémoire oxide) atteint d'ors et déjà 5 ns. Pour toutes ces raisons j'ai choisi d'étudier plus précisément cette technologie dans ce travail de thèse. Nous allons maintenant montrer comment les CBRAMs peuvent s'insérer dans une architecture pour effectuer un apprentissage.

4. Conductive Bridge Random Access Memory (CBRAM)

Les expériences de l'électrolyse en chimie ont mis en évidence les réactions de différents couples oxydo-réducteurs en impliquant des électrolytes. La valeur de la conductivité électrique dépend de la nature des ions présents dans la solution et de leurs concentrations.

L'étude des mémoires ioniques dans le semi-conducteur se concentre dans les électrolytes purement solides. Une couche d'électrolyte est englobée entre 2 électrodes métalliques. Le principe de la réaction chimique interne est similaire à une électrodéposition.

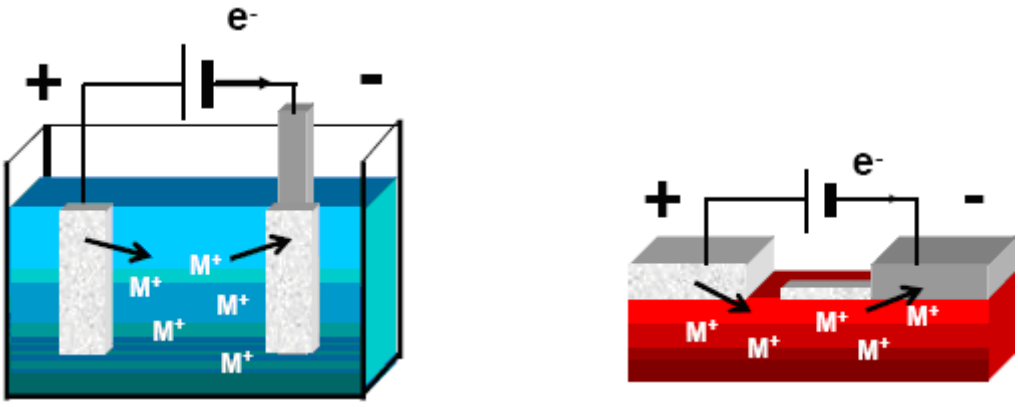


Figure 40. [Kozi05] A gauche, dans une solution, soumi à un champ électrique, les électrolytes sont des ions conducteurs chargés négativement qui viennent se déposer sur l'électrode négative, c'est le processus de l'électrodéposition. A droite, les électrolytes peuvent également être dans un état solide, typiquement dans des films minces de matériau amorphe. Des chemins conducteurs peuvent être crées entre les 2 électrodes.

A l'intérieur du nanocomposant, un film mince de dimension inférieur à 100 nm d'épaisseur contient une grande quantité d'ions mobiles. Dans [Kozi05], les ions sont les électrolytes solides de type Argent, où 33 % de celui-ci peut être imbriqué dans le matériau $Ge_{30}Se_{70}$. Dans d'autres cas, l'électrolyte se trouve dans un matériau chalcogenide ou dans l'oxyde métallique à l'état amorphe. La réduction de l'Argent dans la couche d'électrolyte au cours de l'écriture par tension électrique entraîne la réduction des ions Ag+ dans la couche d'électrolyte, en formant un filament conducteur, tandis que l'oxydation des ions Ag+ constitue une libération d'électrons, et par conséquent le filament est annihilé.

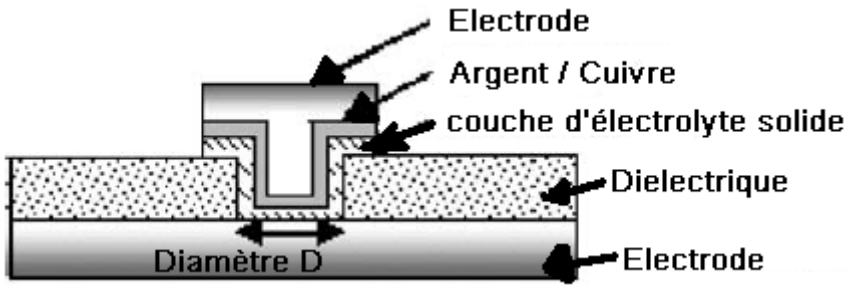


Figure 41. [Kozi05] Le nanocomposant est constitué d'une électrode métallique, puis une interface d'Argent (ou parfois Cuivre), ensuite une autre couche mince d'électrolyte solide, enfin une autre électrode inerte.

Une anode en argent oxydable et une cathode inerte (par exemple, en tungstène W) sont positionnées de part et d'autres. Les ions Ag+ peuvent être réduits à la cathode, et un flux d'ions entre dans la couche d'électrolytes par oxydation de l'anode. Avec une tension de quelques centaines de mV à ses bornes, la réaction chimique est amorcée. La tension nécessaire pour maintenir l'électrodéposition est de la moitié de la valeur initiale. La neutralité de charge totale est maintenue par l'équilibre engendré entre l'oxydation et la réduction. L'électro-déposition cesse quand la réaction est consumée. Les chemins conducteurs crées restent dès lors inchangés à moins qu'on inverse le champ électrique.

La non-uniformité de la distribution des ions favorise l'électrodéposition de l'argent et un filament conducteur peut s'étendre à partir de l'électrode d'origine. Ce phénomène se propage alors sur l'ensemble des électrolytes parce que le champ

électrique entraîne ensuite les charges électriques à se diriger vers l'électrode positive à travers le chemin conducteur produit. Comme la couche mince d'électrolyte est d'une épaisseur faible (moins de 100 nm), le processus de croissance est d'une rapidité extrême (estimé à environ 1nm/ns). Durant la phase de l'électrodéposition, la résistance électrique de la structure décroît de plusieurs ordres de grandeur. La diminution de la résistance à cause de cet effet augmente le courant électrique jusqu'à la limite de claquage.

Théoriquement, la réduction et l'oxydation de l'Ag dans la couche d'électrolyte sont des réactions réversibles. La résistance peut changer dans les deux sens au cours de la programmation en alternant la polarité du champ appliqué. Pour le nanocomposant de type CBRAM, la tension positive appliquée est au voisinage de 200 mV. Cette opération est appelée l'écriture. En l'absence de limiteur de courant, on constate que la résistance baisse rapidement pour atteindre une valeur minimale. En présence de limiteur de courant, des états intermédiaires de résistance peuvent être atteints. La cinétique de la réaction dépend de la tension appliquée, mais cette relation n'est pas bien connue. En appliquant une tension négative, la résistance électrique du nanocomposant devient de plus en plus forte, cela s'appelle l'effacement. En général, la durée d'effacement prendra moins de temps que l'opération d'écriture [Gilb05]. Selon les observations, les temps de réaction sont dissymétriques.

Les états intermédiaires peuvent être obtenus si on maîtrise la réaction de l'électrodéposition. Les explications les plausibles se basent sur des chemins multiples de propriétés conducteurs différentes. Les étapes de la programmation sont basées sur l'effet de l'interruption de la transformation chimique totale. Ce processus est très rapide car la résistance passe de $10^{10}\ \Omega$ à 3 k Ω en moins d'une centaine de nano secondes. La programmation en multi-niveaux peut être effectuée en mode impulsionnel. Plus d'un million de cycles de programmation-effacement sont possibles selon l'estimation de l'endurance [Kund05]. Le temps de rétentions avec 4 niveaux distincts a été apporté à une durée de 27 heures [Kund05]. En fait, à l'heure actuelle la limite ultime est encore inconnue.

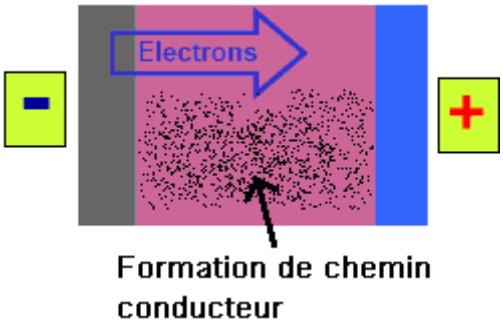


Figure 42. Création de chemins multiples conducteurs entre 2 électrodes

Les mémoires disposant d'un fonctionnement similaire font partie de la catégorie de mémoire ionique. Cette programmation s'appuie sur le principe de la redistribution électrochimique à l'échelle nanométrique de quantités de métaux dans les structures contenant des solides électrolytes. A l'origine, cette propriété est exploitée pour créer des interrupteurs de contact [Kaer05]. Il a été démontré que les états absolus peuvent être atteints d'une résistance minimale de 10^1 Ω à une valeur maximale de 10^{12} Ω.

La mémoire résistive variable par pas constitue une étape importante dans la conception de mémoire synaptique.

5. Modélisation des CBRAM

Je vais présenter dans cette section la modélisation des CBRAMs que j'ai menée pour la réalisation des synapses. J'ai écrit la version Verilog-A du modèle compact comportemental de CBRAM. Il ressort de la bibliographie deux informations certaines. Selon les observations, d'une part il existe une plage de tension, une zone insensible pour laquelle la conductance ne varie pas. Quand la tension aux bornes du composant sort de cette plage, la résistance va tendre vers une des valeurs extrêmes, soit minimale, soit maximale. La variation de résistance dépend du signe et l'amplitude de la tension de programmation. D'autre part, pour obtenir plusieurs niveaux de résistance on peut limiter le courant lors de la programmation. La valeur du courant limite va fixer la valeur finale de la résistance. Enfin la variation de la

résistance reste comprise entre deux bornes extrêmes. On peut maintenant s'intéresser à l'explication la plus vraisemblable des phénomènes physiques. La variation de la conductance est produite par l'électrodéposition qui forme des nanofilaments conducteurs. La vitesse à la quelle ce processus va s'effectuer est donc lié à la vitesse de déplacement des ions. L'existence d'une tension de seuil nous oblige à supposer qu'il existe un champ électrique minimum pour entraîner le mouvement des ions.

La modélisation de ce type de dispositif présente des difficultés et ne fait pas consensus à ce jour. On trouve principalement deux principes de modélisation sans que les résultats publiés permettent de trancher définitivement en faveur de l'une ou de l'autre. La première modélisation correspond à une commande en tension de la variation de la conductance, en prenant compte de l'existence de la tension de seuil V_s. Dans la deuxième modélisation [Snid07-2], l'intégration peut se faire sur une fonction hyperbolique de l'état lié à la tension aux bornes du nanocomposant. La variation de résistance est alors considérée comme "infinitésimal" dans la zone insensible. Dans mon travail, on peut supposer que la variation temporelle de la conductance est proportionnelle à la variation de la tension excédant la tension de seuil.

Si ($V > V_{s+}$ et $G < G_{max}$)

$$\frac{dG}{dt} = K_{+}.(V - V_s) \qquad \textbf{(Eq. 8)}$$

Si ($V < V_{s-}$ et $G > G_{min}$)

$$\frac{dG}{dt} = K_{-}.(V - V_s) \qquad \textbf{(Eq. 9)}$$

Où :

V_{s+} : la tension de seuil dans le sens positif

V_{s-} : la tension de seuil dans le sens négatif

G : la conductance de nanocomposant

G_{min} : la conductance minimale admise

G_{max} : la conductance maximale admise

K_+ : le facteur d'amplication dans le sens positif

K_- : le facteur d'amplication dans le sens négatif

Sinon il n'y a pas de variation de conductance.

La seconde approche de modélisation correspond au memristor [Chua71], repris récemment par l'équipe de recherche HP qui en a réalisé la synthèse. C'est une résistance multi-niveau dont l'état dépend soit de la charge qui l'a traversé, soit d'une grandeur homogène à un flux Φ correspondant à l'intégrale de la tension appliquée aux bornes du composant.

$$\Phi = \int V dt$$

Plus généralement, le memristor est décrit par une relation non linéaire entre le flux total et la charge. A chaque instant on retrouve la résistance en dérivant le flux par rapport à la charge, selon la fonction d'état M(q).

$$M(q) = \frac{d\Phi}{dq}$$

Cette approche [Snid07-2] décrit la relation suivante la variation de conductance dans un composant de type [Karg07] :

$$\frac{dw}{dt} = K.w.\sinh M(q).v \qquad \textbf{(Eq. 10)}$$

Où :

K : le facteur d'amplification

w : le poids synaptique

v : la tension aux bornes du nanocomposant

M(q) : l'état du nanocomposant

Nous avons considéré que la variation de déplacement des ions, donc la vitesse de variation de conductance est proportionnelle au champ électrique, soit la tension. Il s'agit bien d'une commande en tension, et non en charge. Cependant

nous verrons plus loin que nous avons également pris en compte l'hypothèse d'une commande en charge dans la conception de notre circuit.

Dans notre modélisation de la mémoire ionique, si la tension V appliquée aux bornes du dispositif est supérieure à une tension de seuil Vs, la configuration se produit et la variation de la conductivité s'exprime comme suit :

$$\Delta G = \int_{t}^{t+\Delta T} K.V.dt \qquad \textbf{(Eq. 11)}$$

Nous allons maintenant montrer comment ces modélisations permettent d'expliquer les variations de conductance observées sur ce genre de nanocomposant. Le dispositif étant typiquement asymétrique, la conductance peut être modifiée dans les deux sens, à vitesse différente. Nous pouvons ainsi définir deux cas : soit la tension appliquée est positive, soit elle est négative.

<u>Concernant le cas V > 0 :</u>

Lorsqu'on applique une tension positive aux bornes du nanocomposant CBRAM deux cas peuvent se produire :

1er sous cas : Le nanocomposant CBRAM voit une tension V imposée à ses bornes inférieure à la tension de seuil V_s, il n'y a aucune modification de résistance électrique intrinsèque. Il reste en état d'équilibre stable.

2e sous cas : La tension V aux bornes du nanocomposant CBRAM est supérieure à la tension de seuil V_s, il y a diminution de sa résistance électrique intrinsèque qui tend vers sa valeur minimale jusqu'à ce que la tension à ses bornes soit inférieure ou égale à la tension de seuil. Il s'ensuit la modification de la valeur R vers une autre valeur limite telle que la tension finale à ses bornes soit égale ou inférieur à la tension de seuil. Autrement dit tant que la tension aux bornes du CBRAM est supérieure à V_s, il y a modification de résistance en fonction du temps. On peut imaginer trois situations possibles.

- situation 1 : une source de tension V limitée en courant à une valeur maximale Is fixe la tension qui traverse le nanocomposant. L'expérience montre qu'on trouve une résistance finale $R = \frac{V_s}{I_s}$. Cela peut s'expliquer par une analyse du rôle de la limitation de courant. La diminution de la résistance se traduit par une augmentation du courant qui atteint ainsi sa valeur limite I_s. La tension change continuellement ou par saut tant que le courant est supérieur à I_s. Le point de fonctionnement passe donc sous le seuil V_s avec le courant I_s. A cet instant la résistance vaut $R = \frac{V_s}{I_s}$ et cesse de diminuer. C'est sa valeur finale.

- situation 2 : un générateur de tension V_g associé à une résistance interne R_g. Comme précédemment, la diminution de la résistance induit une augmentation du courant. Celle-ci se poursuit tant que la tension aux bornes de la résistance est supérieure au seuil. Le point de fonctionnement suit donc la caractéristique du générateur. Lorsque cela se produit, on a la valeur R qui tend rapidement vers R_x. Soit :

$$\left\{ \begin{array}{l} V + R_g.I = U_g \\ V_s = R_x.I \end{array} \right\}$$

$$R_x = \frac{R_g V_g}{V_g - V_s}$$

- situation 3 : pour un générateur parfait de tension V_g, tant que V_g est supérieur à V_s la résistance du nanocomposant CBRAM va chercher à tendre vers sa valeur minimale intrinsèque R_{min}.

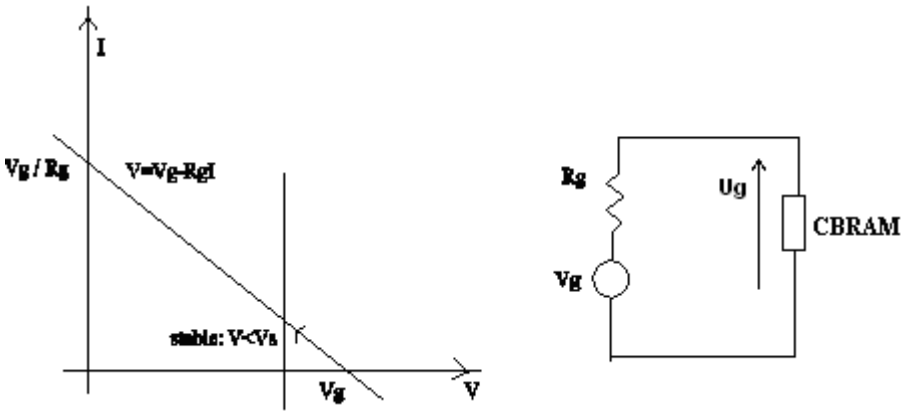


Figure 43. La loi de variation de résistance

Concernant le cas V < 0 :

Lorsqu'on applique une tension fixe V négative aux bornes du composant CBRAM et donc un courant I négatif, le courant traversant celui-ci suit la loi d'Ohm classique. De ce fait, deux cas peuvent se produire:

1er sous cas : La tension aux bornes du nanocomposant CBRAM est inférieure à la tension de seuil V_s. Il n'y a aucune modification de l'état de résistance intrinsèque. Il est stable.

2e sous cas : La tension aux bornes du nanocomposant CBRAM est supérieure en valeur absolue à la tension de seuil V_s. Il y a augmentation de résistance électrique intrinsèque. Cette modification induit une diminution de courant. En conséquence, contrairement au cas $V>V_s>0$, sans la limitation de courant, le processus ne peut s'arrêter. La tension doit être ramenée sous le seuil pour cela. Dans le cas contraire, l'état de résistance maximale sera rapidement atteint. La transformation cesse après que la limite de résistance soit atteinte.

Avant de concevoir un circuit dédié à la variation de conductance, nous allons décrire en langage comportemental le modèle correspondant au composant choisi.

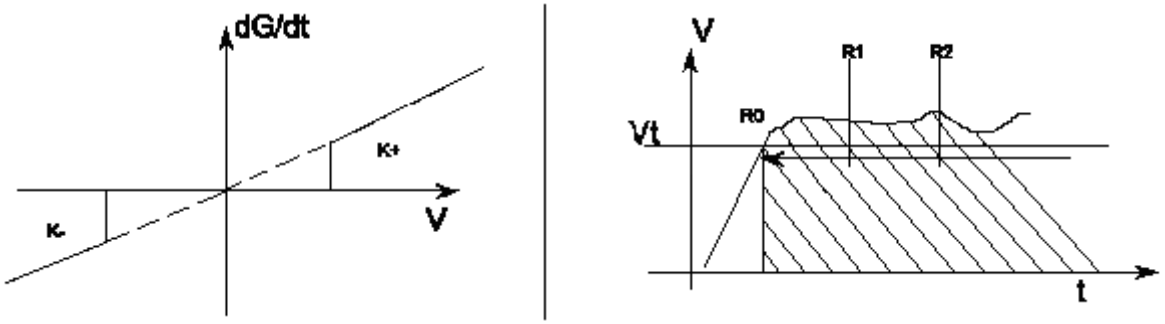


Figure 44. On peut extraire de la figure de gauche la caractéristique des tensions de seuil, la largeur de «zone insensible» et les valeurs de pente K de programmation. A droite, pendant la phase de programmation le nanocomposant subit la variation de sa résistance, quand la tension dépasse le seuil. La variation de la conductance est proportionnelle à l'intégrale de la tension représentée sur la figure

Le code du modèle de simulation est écrit en Verilog-A (cf. Annexe). Le principe de la simulation est d'appliquer une variation +/- ΔG à la conductance lorsque, les seuils étant dépassés, l'intégrale défini par l'équation (2) atteint une quantité ΔG.

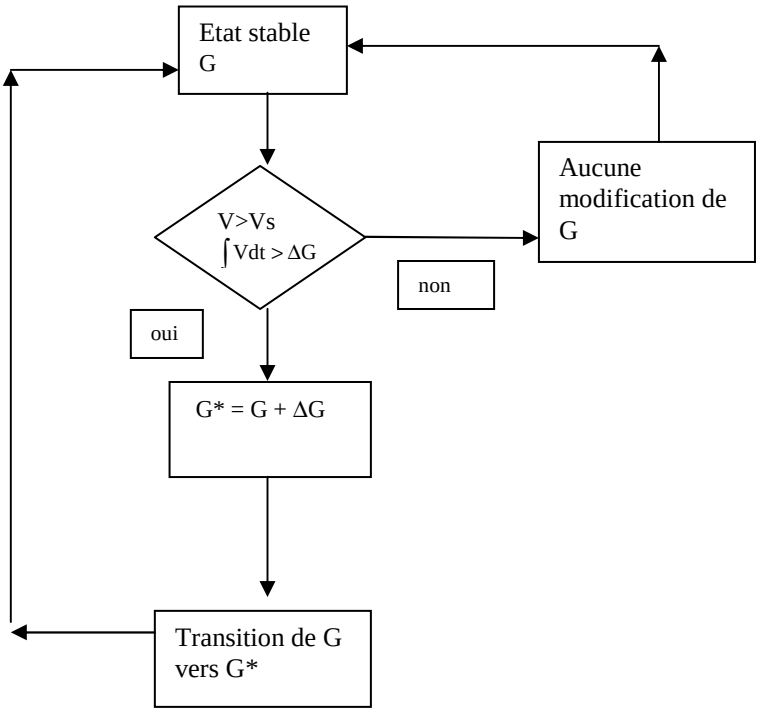


Figure 45. Le diagramme de la modélisation pour V>0

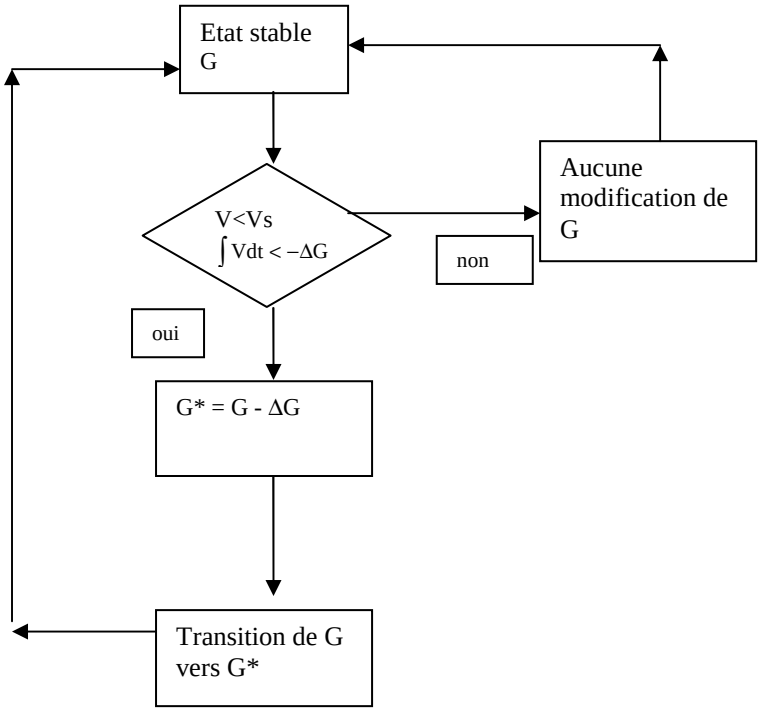


Figure 46. Le diagramme de la modélisation pour V<0

Trois blocs de fonctionnement sont inclus, et décrits ci-dessous :

//Le franchissement de seuil fixe la direction de variation

//la variable Vthp est le seuil à partir duquel se déclenche la variation de résistance

@(cross(V(control,out)-Vthp, -1)) begin

if (direction != 0) begin

direction = 0;

//Puis, l'intégration démarre lorsque la direction est initialisée

if (direction == 0) int_time=1;

swenergy1=idt(V(control,out),0,int_time);

//Suivant la direction, la variation de conductance se voit modifiée suivant une quantification.

if ((direction == 1) && (G==nextG) && (abs(swenergy1)>quantum_threshold_pos) && (G<Gmax)) begin

*delta_G = ohm_step_p*swenergy1;*

if ((G + delta_G)>0) begin

```
            nextG = G + delta_G;
        end
      int_time=0;
      swenergy1=0;
    end

    if ((direction == -1) && (G==nextG) && (abs(swenergy2)>quantum_threshold_neg)
&& (G>Gmin)) begin
      delta_G = ohm_step_n*swenergy2;
      if ( (G + delta_G)>0 ) begin
            nextG = G + delta_G;
        end
      int_time=0;
      swenergy2=0;
    end

//le temps de transition est gouverné par tconv
    G = transition(nextG, tconv, trise, tfall);
```

6. Modèles de CNTFET

L'élément actif de l'architecture choisie est le transistor à nanotube de carbone (CNTFET). Son modèle compact est conçu pour les nanotubes de carbones semi-conducteurs, disposant d'un profil similaire au MOSFET [Preg06] [Mane06]. Le but est d'expérimenter la simulation des circuits logiques ou analogiques avec un modèle compact.

Différents modèles de simulation élaborés sont disponibles en provenance des universités suivantes, de Strasbourg [Preg06], de Bordeaux [Mane06], d'Arizona [Bali07], de Stanford [Wong06] et de Purdue [Ren03], [Rayc04].

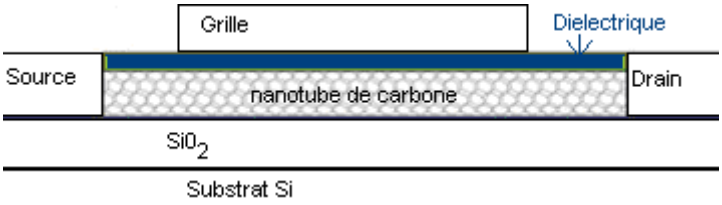


Figure 47. Le transistor à nanotube de carbone de diélectrique ZrO_2

L'évaluation de réseaux de caractéristiques de CNTFET est donc particulièrement importante. J'ai testé le modèle compact fourni par le Laboratoire de l'Intégration du Matériau au Système (IMS) de l'université Bordeaux-1 [Mane06], issue de notre collaboration au sein de l'ACI (Action Concertée Incitative) Nanosys, puis de l'ANR (Agence Nationale pour la Recherche) Panini (Programme Architecture Nanoélectronique Intégrées Neuro-Inspirées). Le jeu de paramètres a été choisi pour le modèle compact employé dans les simulations :

Vecteur de chiralité (n,m)	**(19,10)**
Résistance de grille	**10 Ω**
Résistance de contact S,D	**20 kΩ**
Capacité de grille	**100 pF**
Capcité électrostatique de drain, et de source	**0.1 aF**
Capacité substrat-source	**0.4 aF**
Distance inter-carbone	**0.142 nm**
Tension Flat-Band Vfb	**-40 mV**
Longueur	**100 nm**
Diamètre	**1.4878 nm**
Covering factor	**-3.033 eV**
Température	**300 K**
Nombre de sous bandes	**7**

Tableau 12. Paramètres de simulations par défaut

Pour connaître les réseaux de caractéristiques de transistor, les résultats de

simulations du CNTFET sont montrés ci-dessous. Les simulations électriques ont été réalisées par le simulateur Spectre dans l'environnement Cadence ®.

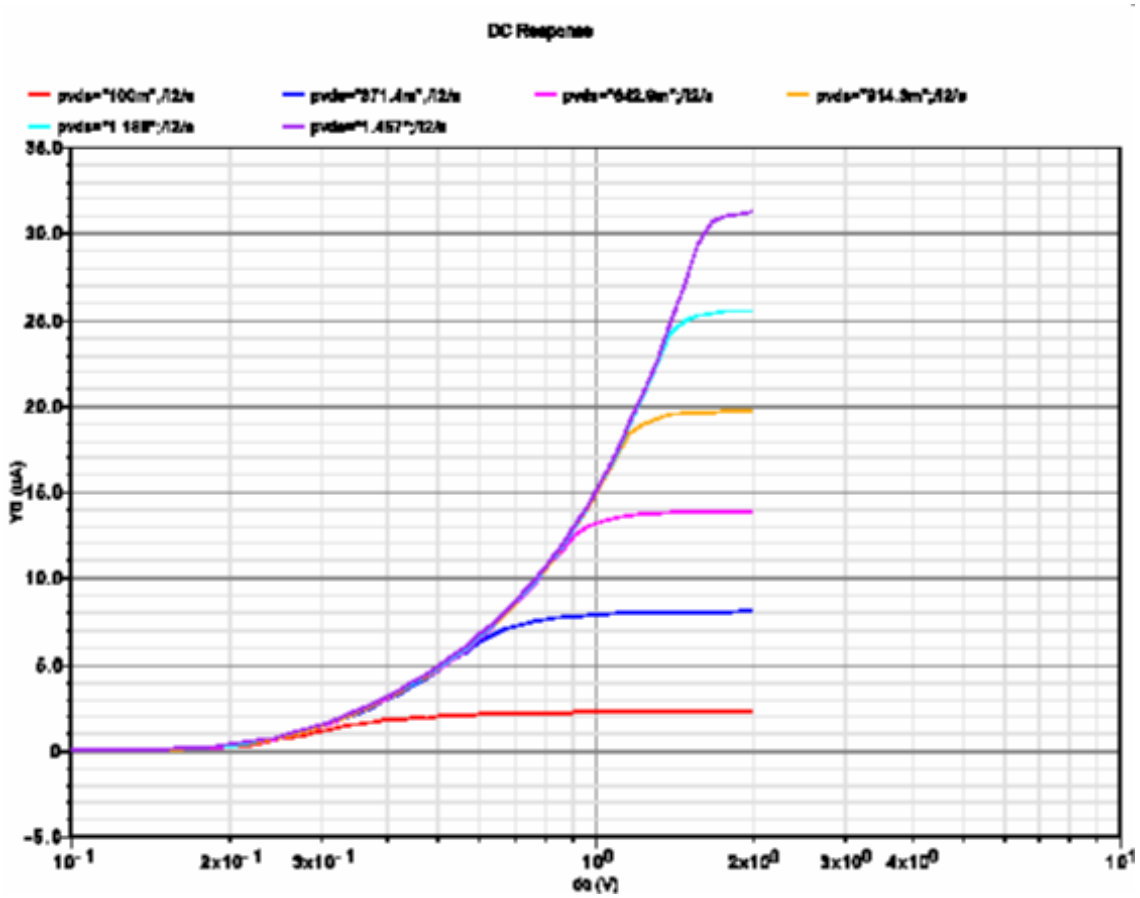


Figure 48. Les réseaux de caractéristiques Id=f(Vgs) de nanotube de carbone de type N. En axe des abscisses, la tension Vgs [0.1V ; 2V]. En axe des ordonnées, le courant Ids. Simulé à partir du modèle [Mane06]. Les tensions pvds sont la tension drain source du nanotube de carbone. Dans la simulation paramétrique, les valeurs assignées aux pvds sont, 100mV, 371mV, 643mV, 914mV, 1.186V, 1.457V.

Pour déterminer les paramètres de simulation les plus adaptés, différents jeux de paramètres ont été évalués, notamment la tension source drain V_{ds} et la tension de bande plate (flat-band) V_{fb}. On observe un bon fonctionnement pour une faible tension V_{ds} réelle. Par la suite, la simulation d'un inverseur peut se révéler intéressante. Elle nous informe sur le délai et sur le couple tension-courant. L'inverseur en question est constitué d'un CNTFET de type P et de type N, associés de manière CMOS.

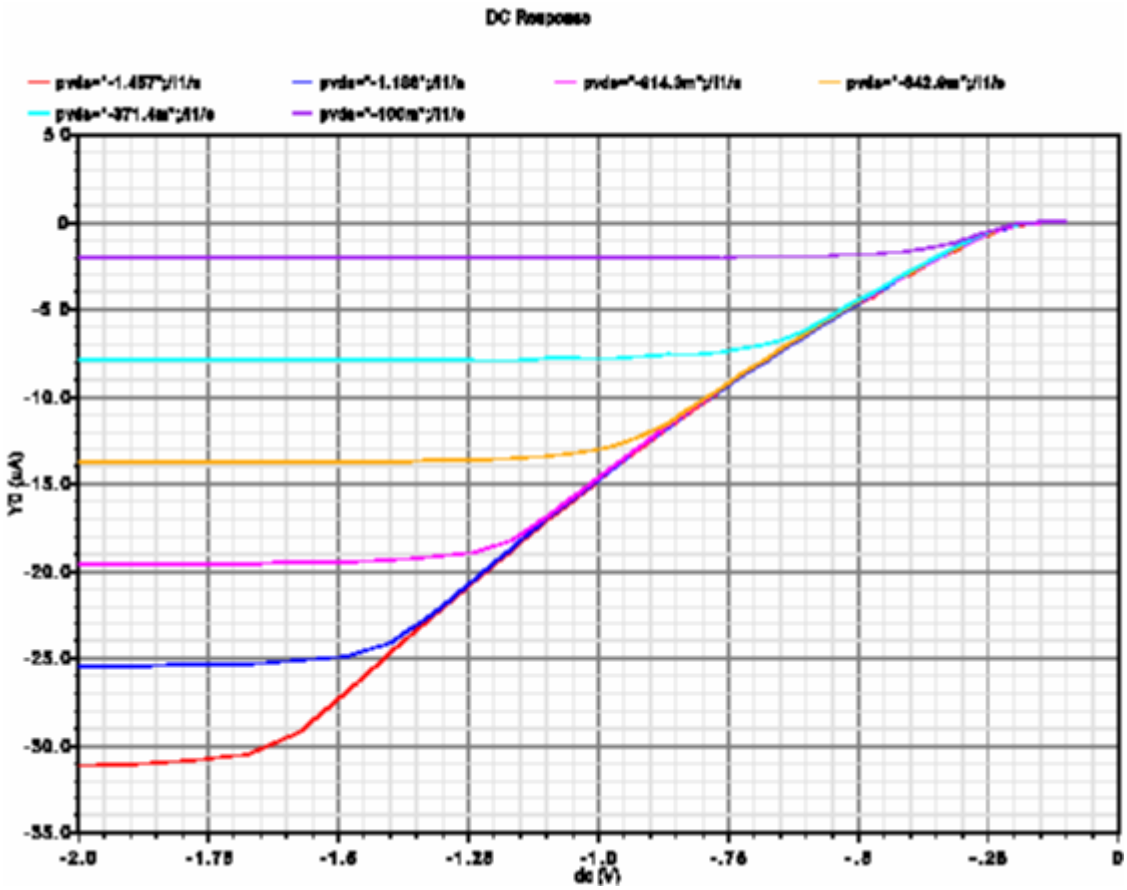


Figure 49. Les réseaux de caractéristiques Id=f(Vgs) de nanotube de carbone de type P. En axe des abscisses, la tension Vgs [0.1V ; 2V]. En axe des ordonnées, le courant Ids. Simulé à partir du modèle [Mane06]. Dans la simulation paramétrique, les valeurs assignées aux tensions pvds sont, 100mV, 371mV, 643mV, 914mV, 1.186V, 1.457V.

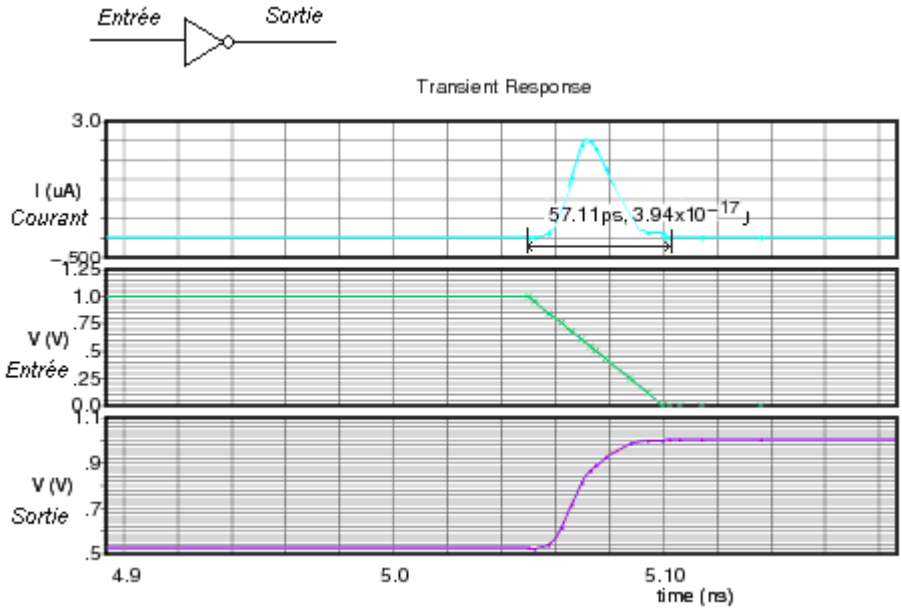


Figure 50. Simulation d'un inverseur à CNTFET

En comparant cette figure avec les deux précédentes, on peut remarquer que le courant de saturation (20µA) n'est jamais atteint. Le temps de commutation est de l'ordre de 57 ps, voire légèrement inférieur. La tension d'alimentation utilisée est de 1V. En fait, les caractéristiques principales du CNTFET permettent d'envisager de construire n'importe quel circuit fait de transistor traditionnel, ainsi un exemple ci-dessous, pour réaliser un montage en différentiel.

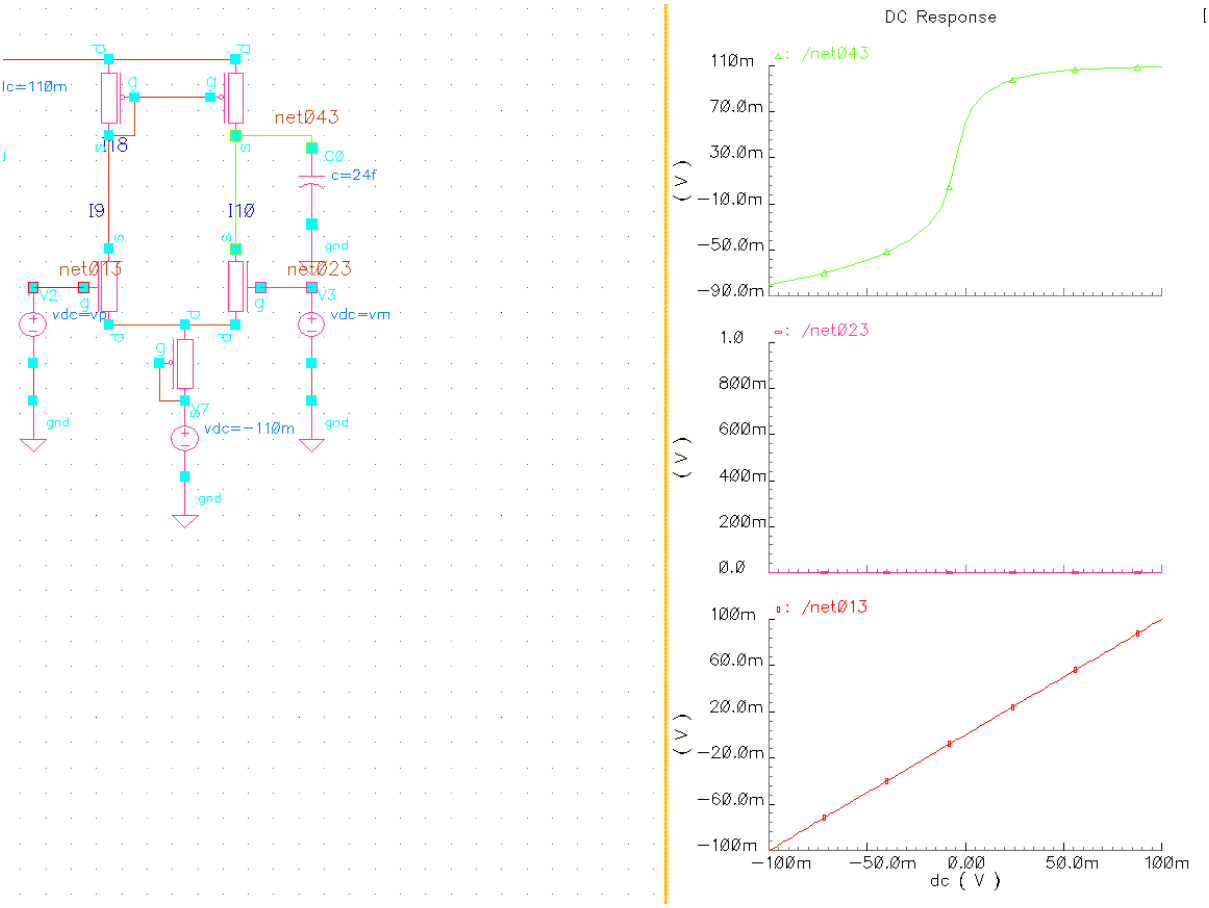


Figure 51. Simulation électrique d'un amplificateur différentiel réalisé à partir des nanotubes de carbone

7. Circuit de programmation des mémoires multi-niveaux

Le but de la programmation multi-niveau est d'arrêter le processus de variation de la conductance avant qu'il n'ait atteint un état extrême. L'analyse montrée par la publication [Kozi05] montre qu'il existe certainement une relation entre la vitesse de variation de la conductance en fonction du temps et la tension appliquée, mais pour l'instant aucune équation n'a été établie. La durée de cette variation, relativement

courte (de l'ordre de quelques dizaines de ns), n'empêche pas de supposer que le processus de modification de la conductance puisse être interrompu avant qu'un des états de résistance extrême ne soit atteint. Malheureusement, à ce jour, les caractéristiques temporelles sont manquantes pour pouvoir décrire la vitesse de modification de la conductance de façon précise et avec certitude. Aucune référence sur la modélisation du nanocomposant n'est connue avant d'entamer la nôtre. Nous avons cependant fait l'hypothèse de la programmation du nanocomposant par impulsions.

Il existe plusieurs façons d'implémenter l'unité de programmation suivant que la commande se fasse en tension ou en charge.

La commande en tension est la plus simple. Elle consiste à appliquer des impulsions de tension suffisamment courtes. Nous avons décrit cette partie en verilog-A. Deux signaux numériques scu_I, scu_D commandent respectivement l'incrémentation ou la décrémentation de la conductance:

```
//la partie centrale de la commande numérique en tension de sortie vout
  analog begin
    @(initial_step("ac","dc","tran","xf" )) begin
       vout = 0;
    end
    if  (V(scu_I) > vth)  vout=vpos;
    else if (V(scu_D) > vth)  vout=vneg;
    else begin
              I(scu_Out) <+ 0.0;
              vout = 0.0;
       end
  V(scu_Out) <+ transition (vout, tconv, trise, tfall);
  end
```

La commande en charge nécessite un circuit de programmation différent. Pour cela, on peut réaliser un transfert de charge accumulée. Le fonctionnement d'injection de charge [Zhu95] étudié ici, s'appuie sur la commande du miroir de

courant (cf. le diagramme ci-dessous) :

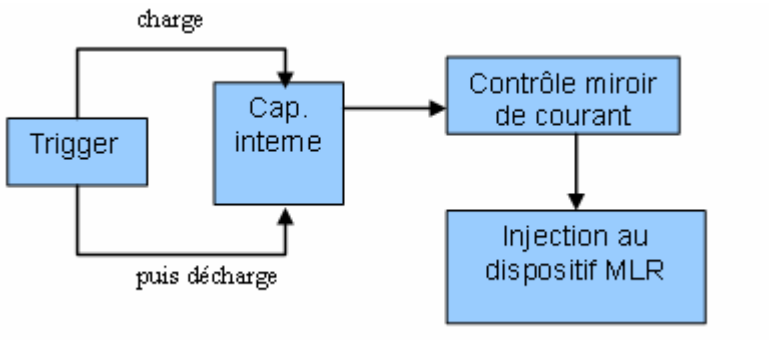


Figure 52. Injecteur de charge commandé par les signaux trigger issus de l'unité logiques

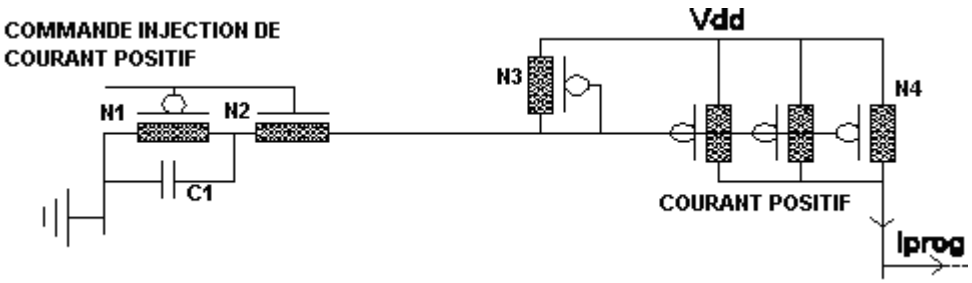


Figure 53. Le miroir de courant de l'injecteur de charge

L'injecteur de charge introduit l'utilisation d'un miroir de courant. Il y a un courant lorsqu'un niveau haut est appliqué à la grille de N2, qui devient ainsi passant alors que N1 reste bloqué. Un courant charge alors la capacité C1 jusqu'à ce que sa tension atteigne la valeur Vdd moins la tension de seuil de N2 et N3. Durant ce laps de temps, un courant est recopié par le miroir de courant à travers l'ensemble des nanotubes N4. On peut compter sur un effet d'amplification de courant pour autant que le nombre de nanotubes dans cette branche soit augmenté de manière significative. En temps normal, ce courant est injecté dans la résistance multi-niveau, il franchit le seuil de la variation résistive, sa résistance électrique passe à l'état de transition. Alors la résistance du nanocomposant commence à changer, elle cesse lorsque le courant de programmation diminue et repasse sous seuil. Le courant est relié à C1 chargée en même temps. Puis, le signal de commande d'injection est réinitialisé à 0, C1 se décharge à travers N1.

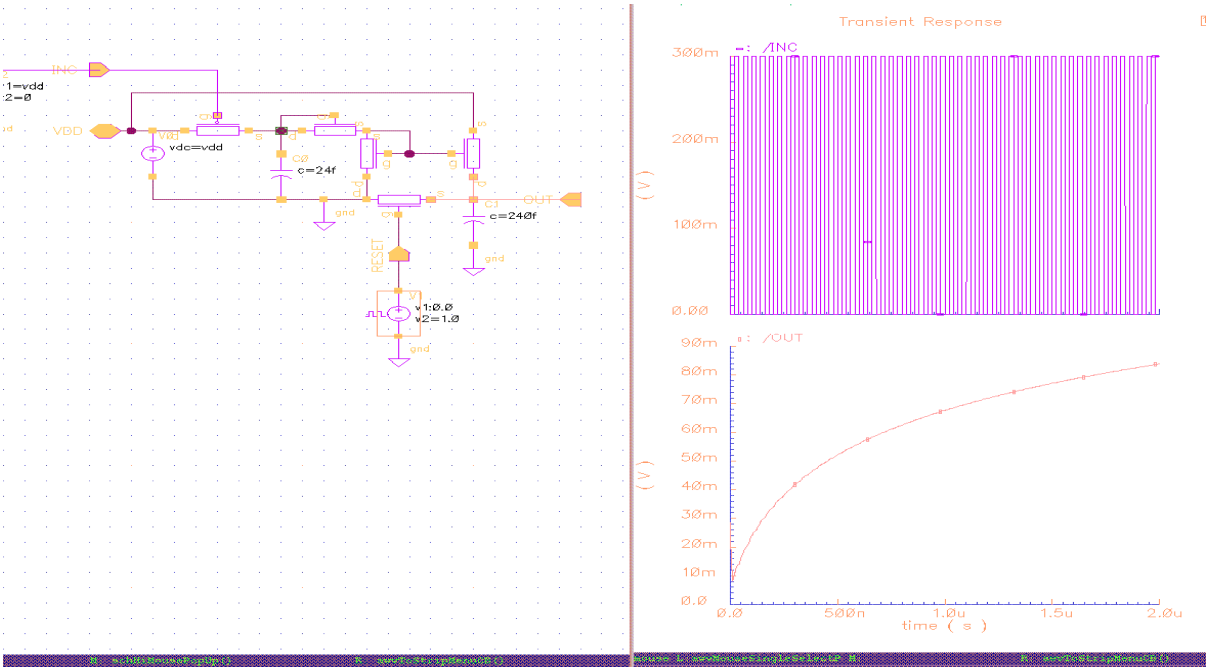


Figure 54. Simulation électrique de l'injecteur de charge réalisé à partir des nanotubes de carbone

Le miroir de courant peut être utilisé avec un coefficient multiplicateur qui dépend alors du courant à injecter dans le nanocomposant.

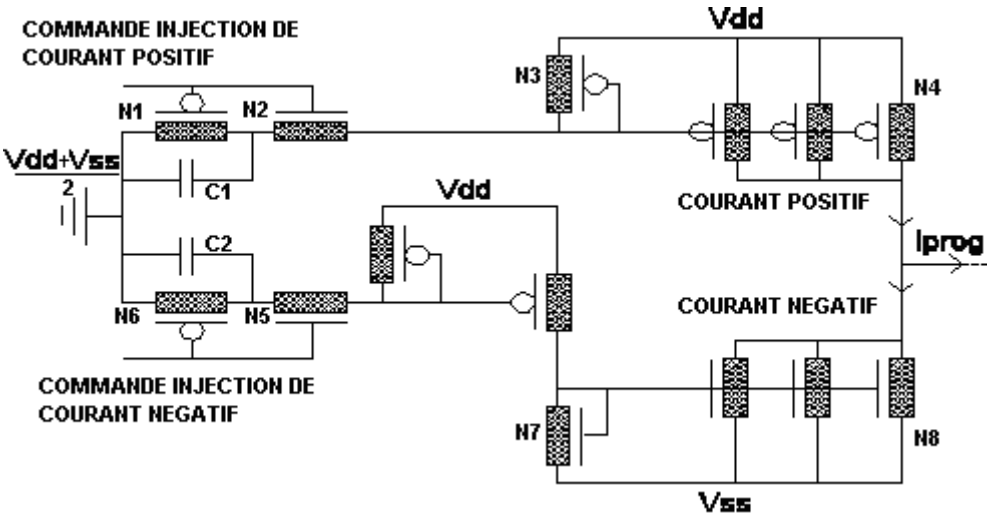


Figure 55. Injection de courant en deux branches distinctes

L'injection de charge consiste en impulsions de courant positif à travers le nanocomposant pour augmenter sa conductance, et en impulsion de courant négatif à travers le nanocomposant pour diminuer sa conductance. Pour l'injection de courant négatif, le raisonnement est analogue avec la commande de N5, et C2. Il est à noter qu'un autre miroir de courant intermédiaire est inséré dans le circuit pour

inverser le sens du courant. Pour effectuer ces opérations, une quantité de charge suffisante doit être injectée à travers ce nanocomposant.

Pendant l'impulsion de programmation, en considérant une simple résistance équivalente pour modéliser les CNTFET, le courant de charge décroît exponentiellement.

$$I(t) = \frac{V_p}{R_{cntfet}} * \exp(\frac{-t}{R_{CNTFET} C})$$

Grâce au gain du miroir de courant, la tension aux bornes de la résistance multi-niveau est donnée par l'équation suivante, avec N_{mirror} le nombre de nanotube de carbones sur la branche du mirroir de courant.

$$V(t) = \frac{N_{mirror} I(t)}{G(t)}$$

En considérant $\Delta G << G(t)$, la valeur de la conductance utilisée pour calculer la tension aux bornes de la résistance est constante et égale à sa valeur initiale.
On peut donc calculer la variation de la conductance en intégrant cette tension.

$$V(t) = N_{mirror} \frac{I(t)}{G(t_0)} = V_0 \exp^{\frac{-t}{R_{CNTFET} C}} \qquad \textbf{(Eq. 12)}$$

Avec :

$$V_0 = \frac{N_{mirror} I(t_0)}{G(t_0)} . \frac{Vp}{R_{CNTFET}}$$

La variation de conductance cesse à l'instant ΔT lorsqu'elle repasse sous le seuil, et sera donc égale à, cf. Eq.8, Eq.9.

$$\Delta G = \int_{t=t_0}^{t_0+\Delta t} K.V(t).dt$$

Où ΔT est :

$$\Delta T = R_{CNFET} C.\mathrm{Log}(\frac{V_0}{V_T}) \qquad \textbf{(Eq. 13)}$$

Ce qui donne la relation de variation de conductance comme celle-ci :

$$\Delta G = K.R_{CNTFET}C.V_0 = K.R_{CNTFET}C.\frac{N_{mirror}}{G(t_0)}.\frac{Vp}{R_{CNTFET}} \qquad \textbf{(Eq. 14)}$$

Nous avons simulé électriquement la variation de conductance obtenue avec ce type d'injecteur de charge. Le comportement est illustré ci-dessous.

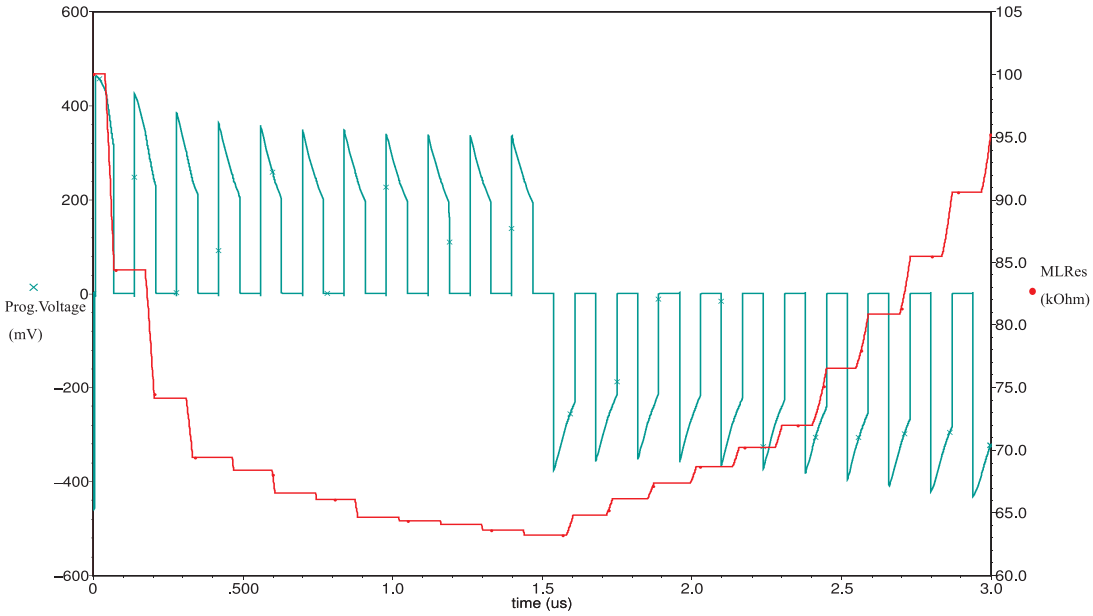


Figure 56. Variation synaptique par étapes, en rouge la résistance modifiée, $K=50e^{3}$, $R_{cntfet}.C=150e^{-12}$

Le test de programmation synaptique est effectué au moyen de 18 impulsions, allant de 100kΩ à 63 kΩ dans la première phase, puis en deuxième phase de 65kΩ à 98kΩ. Les valeurs de résistances correspondantes sont présentées dans le tableau suivant :

PHASE 1 (décroissance de résistance synaptique)

Résistance(kΩ)	100	85	74	70	68	66
ΔG simulé		1,76E-06	1,75E-06	7,72E-07	4,20E-07	4,46E-07
Résistance(kΩ)	65,5	64,5	64	63,5	63	
ΔG simulé	1,16E-07	2,37E-07	1,21E-07	1,23E-07	1,25E-07	

PHASE 2 (croissance de résistance synaptique)

Résistance(kΩ)	65	66	67	68,5	70	72
ΔG simulé		-2,33E-07	-2,26E-07	-3,27E-07	-3,13E-07	-3,97E-07
Résistance(kΩ)	76,5	81	85,5	91,5	98	
ΔG simulé	-8,17E-07	-7,26E-07	-6,50E-07	-7,67E-07	-7,25E-07	

Tableau 13. Cycles de test de modification de valeurs de résistance

Nous avons tenté d'exploiter un circuit utilisant les transferts de charges pour contrôler la variation de conductance alors qu'elle est commandée en tension dans

notre modèle. Le principe de ce circuit de programmation présente l'avantage de limiter intrinsèquement la quantité de charges injectée ou extraite. Cette limitation garantit que la variation de conductance sera contrôlée par le nombre d'ions participant à la réaction. A contrario, la dynamique de variation est limitée par la relation entre la conductance et sa variation relative. En effet, lorsque la valeur de la résistance est trop grande, la variation relative de la conductance est trop importante pour un contrôle fin nécessaire à l'apprentissage. Inversement, lorsque la valeur de la résistance est trop petite, la variation de conductance est minime et conduirait à une durée d'apprentissage excessive. En outre, dans le cas où l'injecteur de charges est utilisé pour programmer les résistances multi-niveaux, cette conception peut induire une surface de condensateur relativement importante pour un pas de configuration spécifié, on peut estimer la valeur de la capacité selon la relation de la variation de la conductance décrite précédemment. Soit pour les paramètres suivants: ΔG=1e-07, K=50e3, Nmirror=10, Vp=0.2, on déduit C=14,2 fF.

Par conséquent l'apprentissage devrait être multiplexé pour un ensemble de synapses si bien que l'utilisation de la cellule soit partagée. Le but est de réduire la surface de son implémentation.

Les simulations qui suivent ont donc été réalisées avec le modèle fonctionnel écrit en Verilog-A pour une commande en tension. La seule contrainte dans ce cas là sera de ne pas tomber sous une valeur minimale de résistance.

La programmation des résistances multi-niveaux n'est possible que lorsque la tension appliquée à ses bornes dépasse le seuil en tension de configuration. Lorsque la résistance du composant devient trop faible, le courant requis pour y développer une tension suffisante devient trop important pour traverser un CNTFET sans le détruire. Dans notre circuit, les nanotubes de carbones sont utilisés pour la programmation. On peut donc calculer une valeur de résistance minimale R_{min} tolérée.

$$V_g \cdot \frac{R_{min}}{R_{CNTFET} + R_{min}} > V_s$$

Or R_{CNTFET} est essentiellement dominée par les résistances de contact.

Prenons l'hypothèse suivante :

V_s = 0.24V

V_g = 0.6 V

R_{CNTFET} = 75 kΩ (pour V_{ds}=0.6, I_{ds}=8µA)

$$R_{min} > \frac{R_{CNTFET} * V_s}{V_g - V_s}$$

R_{min} = 50 kΩ

Cette limite nous contraint à conserver des niveaux de résistance supérieure à cette valeur.

8. Architecture pour l'apprentissage

Ici le sens de variation des conductances associées aux synapses va être donné par les algorithmes d'apprentissage décrits au Chapitre 2. Lorsqu'on considère les applications de la logique les états de neurones seront binaires. Dans ce cas les règles d'apprentissage peuvent être décrites par des équations booléennes dépendantes de l'état de l'entrée Xi, de la sortie obtenue F et attendue S.

On peut ainsi « binariser » la règle delta :

Le poids synaptique augmente quand : $\left\{\begin{matrix} X_i = S \\ S \neq F \end{matrix}\right\}$ **(Eq. 15)**

Le poids synaptique diminue quand : $\left\{\begin{matrix} X_i \neq S \\ S \neq F \end{matrix}\right\}$ **(Eq. 16)**

On peut dresser le tableau de variation du poids synaptique :

	$Xi=E_0$	$Xi=E_1$
$S=E_0,F=E_0$	0	0
$S=E_1,F=E_0$	$-\Delta$	$+\Delta$
$S=E_0,F=E_1$	$+\Delta$	$-\Delta$
$S=E_1,F=E_1$	0	0

Tableau 14. Variation de poids synaptique (règle delta)

Soit : E0, l'état de neurone en repos

E1, l'état de neurone en activité

Δ, le pas de variation du poids synaptique

Cette règle peut être généralisée. En effet, considérons maintenant la règle d'apprentissage dédiée à la machine de Boltzmann. Comme nous avons déjà vu précédemment, la mise à jour des poids synaptiques est proportionnelle à la différence des co-activités durant les deux phases libre et forcée, $C_{ij} = (C_{ij}^{+} - C_{ij}^{-})$ estimée par la relation suivante

$$C_{ij} = \frac{1}{N}\sum_{k=1}^{N} X_i . X_j$$

Si la fonction d'activation était déterministe (à t=0), cette moyenne sur N itérations reviendrait toujours à la même valeur. Si on laissait de côté l'aspect stochastique, on aurait:

$$C_{ij} = X_i . X_j$$

D'où la variation synaptique :

$$W(t) = W(t-1) + \eta(X_j^{+} - X_j^{-})Xi$$

Concernant la règle d'apprentissage de la machine de Boltzmann, la variation synaptique est nulle si des comportements identiques de neurones sont observés dans les deux phases, libre et forcée. En suivant la même approche on peut dresser maintenant le tableau de variation de poids synaptique. Si on considère que la

température T est nulle, et que $X_j^+ = S$ et $X_j^- = F$, on constate une parfaite correspondance avec le tableau de variation de poids de la règle delta.

	$X_i=E_0$	$X_i=E_1$
$X_j^+=E_0$, $X_j^-=E_0$	0	0
$X_j^+=E1$, $X_j^-=E_0$	$-\Delta$	$+\Delta$
$X_j^+=E_0$, $X_j^-=E_1$	$+\Delta$	$-\Delta$
$X_j^+=E_1$, $X_j^-=E_1$	0	0

Tableau 15. Variation de poids synaptique (machine de Boltzmann)

C'est le cas en absence de neurones cachés. A chaque itération on calcule C_{ij} à l'équilibre pour chaque synapse des deux neurones adjacents X_i et X_j, pour les deux phases de l'apprentissage. Enfin quand il y a autant de co-occurrence dans les deux phases, C_{ij} sera nul. La variation du poids synaptique est également nulle.

L'aspect stochastique, qui constitue le principe de la machine de Boltzmann, nécessite au préalable un générateur d'aléatoire pour le tirage au sort de l'état du neurone. Sans cela il est équivalent à l'apprentissage par la règle de delta au sens de la variation synaptique principale. La réalisation du générateur d'aléatoire peut faire augmenter de façon drastique la complexité et la surface. Par la suite, j'ai décidé d'implémenter la règle delta dans l'architecture qui peut se ramener à une base de conception auto-correctrice. Cette règle d'apprentissage est le dénominateur commun de toutes les règles d'apprentissage en mode supervisé que j'ai étudiées.

La variation de poids synaptique suit la règle delta :

$$\Delta G_{ij} = \Delta W_{ij} = \alpha \sum_{i=0}^{N} x_i \times (S_j - F_j)$$

La version binaire de la règle delta correspondant à la variation binaire au sens stricte du terme. Les conditions de la croissance et la décroissance des conductances sont décrites par les tables de vérité suivantes :

	$X_i=E_0$	$X_i=E_1$
$S=E_0, F=E_0$	0	0
$S=E_1, F=E_0$	0	1
$S=E_0, F=E_1$	1	0
$S=E_1, F=E_1$	0	0

Tableau 16. Logiques de la croissance de poids synaptique

	$X_i=E_0$	$X_i=E_1$
$S=E_0, F=E_0$	0	0
S=E1, $F=E_0$	1	0
$S=E_0, F=E_1$	0	1
$S=E_1, F=E_1$	0	0

Tableau 17. Logiques de la décroissance de poids synaptique

Nous pouvons en déduire une expression de ces deux signaux :

le signal INC = $S.\overline{F}.X_i + \overline{S}.F.\overline{X_i}$

le signal DEC = $S.\overline{F}.\overline{X_i} + \overline{S}.F.X_i$

En simplifiant on retrouve la relation logique suivante :

INC = NOR (INV(A),B)

DEC= NOR (A,B)

Avec :

A = XNOR(X_i,S)

B = XNOR(S,F)

Ensuite, selon la commande issue de la logique d'apprentissage, un circuit va imposer un courant à travers la résistance multi-niveau, dans un sens ou dans l'autre. Il s'ensuivra une modification de la conductance.

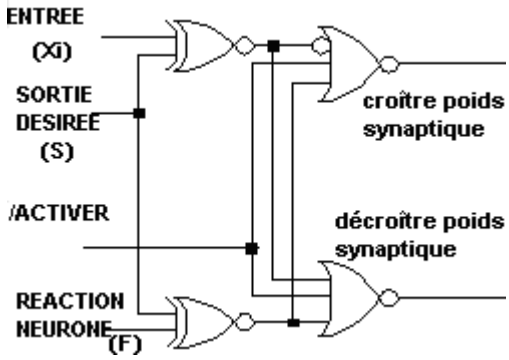


Figure 57. L'unité de configuration logique

La programmation est individuelle. Les nanocomposants voisins qui ne sont pas concernés par la modification synaptique doivent être déconnectés de la ligne. Nous constatons que l'unité de configuration logique prend en entrée les états X_i, S et F associés à une synapse. L'unité de configuration logique ne pouvant être utilisée que pour une synapse à la fois, nous proposons de la multiplexer par ligne. La programmation est donc intrinsèquement séquentielle, s'effectuant colonne par colonne, gérée par un sélecteur de synapse. La configuration des synapses se fait de manière synchrone, commandée par une horloge /ACTIVER. L'apprentissage est effectué dans la puce. Bien entendu, la modification synaptique est basée sur l'unité de configuration logique.

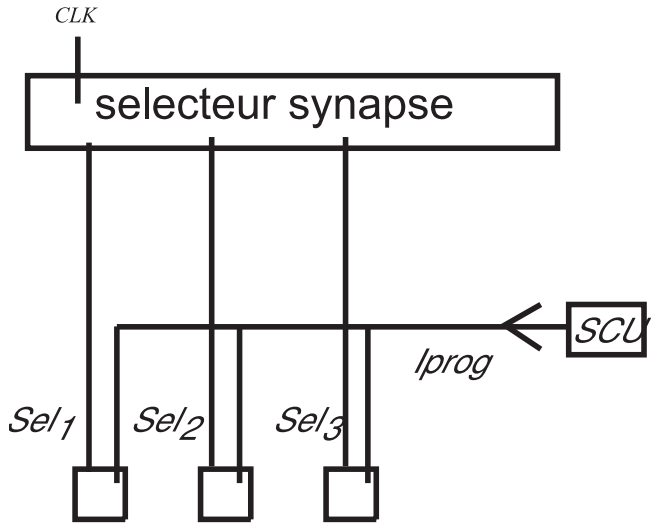


Figure 58. Adressage de synapse par sélecteur

La procédure d'apprentissage consiste à modifier les poids synaptiques jusqu'à obtenir la réponse désirée pour tous les patterns d'entrée. On appelle pattern, le stimulus d'entrée et les états désirés de neurones en sortie du réseau. On appelle epoch la présentation de l'ensemble des patterns de la base d'apprentissage. Pour chaque pattern, la phase d'apprentissage est décomposable en 2 étapes. La première est la présentation de pattern, qui impose les stimuli sur les entrées des neurones. Elle sert juste à obtenir la réponse naturelle F du neurone associé, puis dans la deuxième étape, les entrées sont injectées successivement à l'unité de configuration. La réponse désirée S est également présentée à l'unité de

configuration logique. Il s'ensuit le calcul du sens de variation qui définit la tension ou le courant de configuration synaptique. Dans l'architecture considérée, c'est-à-dire dans la matrice de synapses, la programmation se fait en sélectionnant la colonne séquentiellement. A partir de la nécessité d'une programmation séquentielle par colonne, nous avons déduit l'architecture représentée sur la figure suivante.

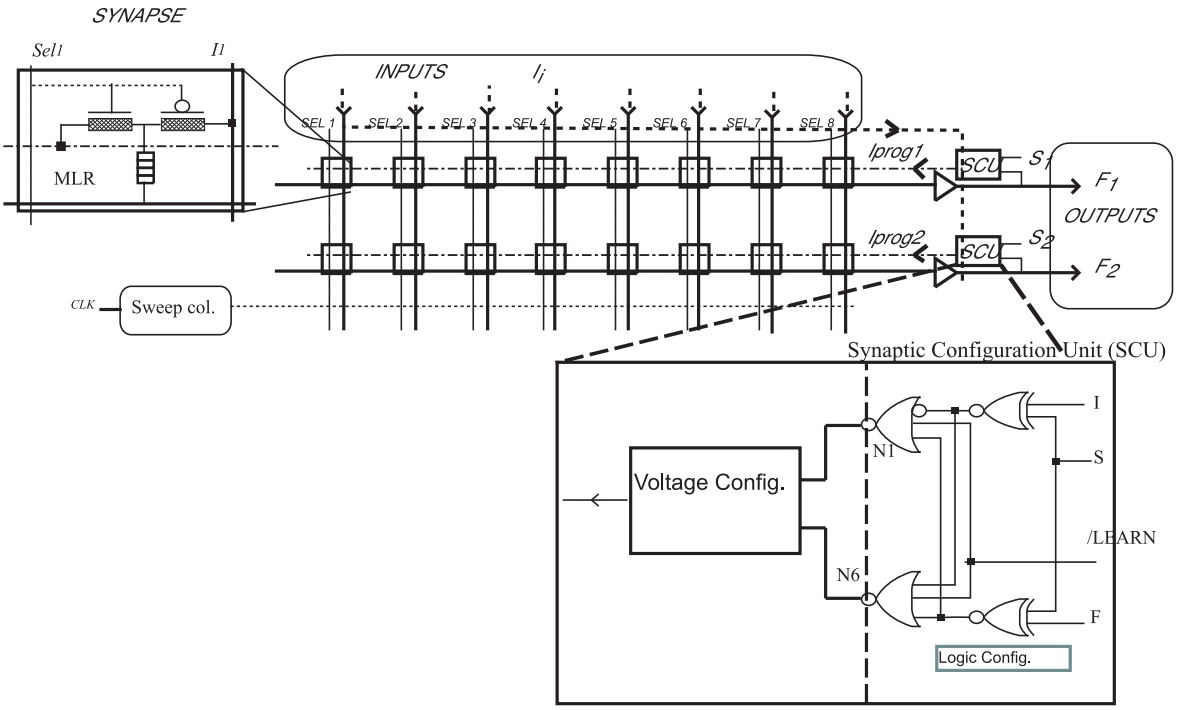


Figure 59. Le circuit avec l'unité d'apprentissage SCU

Pendant la phase normale d'exploitation, aucune tension et courant ne servent de programmation à travers la synapse. Ceci est assuré par un commutateur. Les stimuli ne servent pas à la programmation synaptique.

Pour la phase d'apprentissage, une unité d'apprentissage (cf. figure 57) est utilisée pour la configuration synaptique. Dans l'architecture en question, elle est incluse dans le neurone. Elle gère un ensemble de synapses connectées aux entrées du neurone. Ainsi, l'entrée sélectionnée de la colonne est acheminée à la cellule SCU au moyen de sélecteur de colonne. D'abord se trouve en amont l'unité de comparaisons des patterns. Elle commande le bloc analogique « Voltage Config » qui contrôle la variation de la conductance. Ce bloc est dépendant de la technologie

utilisée pour les synapses. Le bloc « Logic Config » est, quant à lui composé de portes logiques Booléennes réalisant la règle delta dans notre architecture. Dans les simulations qui suivront, les portes logiques sont conçues à partir des modèles de CNTFET. Deux signaux existent pour commander le sens de la variation synaptique. D'une part, la commande INC est un signal pour incrémenter la conductance de la synapse. D'autre part, la commande DEC permet de décrémenter la conductance synaptique.

9. Apprentissage de fonctions logiques

Nous allons maintenant présenter une simulation réalisant un apprentissage de fonctions logiques. Cet apprentissage est constitué de plusieurs épochs de façon à obtenir la convergence du réseau. Chaque époch est lui-même constitué de plusieurs itérations correspondant à la présentation de chaque pattern. Pour chaque pattern, les différentes colonnes seront sélectionnées séquentiellement. Dans la simulation suivante, on peut voir au total 7 épochs pour un apprentissage d'une fonction à 3 entrées binaires. Chaque époch est composé par 8 (2^3) patterns différents correspondant à la séquence codée en binaire : 000, 001, 010 ... 111 pour chaque étape. La simulation dure 70 µs correspondant à 7 époche de 10 µs. Chaque présentation de pattern dure 1 µs. Chaque modification synaptique dure 70ns.

On peut observer dans la simulation suivante, la 1ère ligne correspondante à la tension de programmation de la SCU. La 2ème ligne montre les états obtenus en sorties du neurone appliqué pour les différentes entrées. La 3ème ligne est la sortie désirée imposée au neurone. La 4ème ligne correspond au signal du décodeur qui balaye les colonnes. La 5ème ligne est le signal qui déclenche l'évaluation de l'état du neurone. Enfin à la 5ème époque, le comportement du neurone réagit comme celui du modèle. L'unité d'apprentissage ne modifie plus les conductances synaptiques. A ce moment précis le neurone est configuré. Ces simulations montrent que l'apprentissage de fonctions logiques linéairement séparable est extrêmement efficace. D'une part la convergence est particulièrement rapide. D'autre part au stade final de la programmation, les poids synaptiques ne subissent plus aucun

changement. L'apprentissage s'arrête de lui-même quand la réponse naturelle du neurone est identique à la réponse désirée pour tous les stimuli d'entrée.

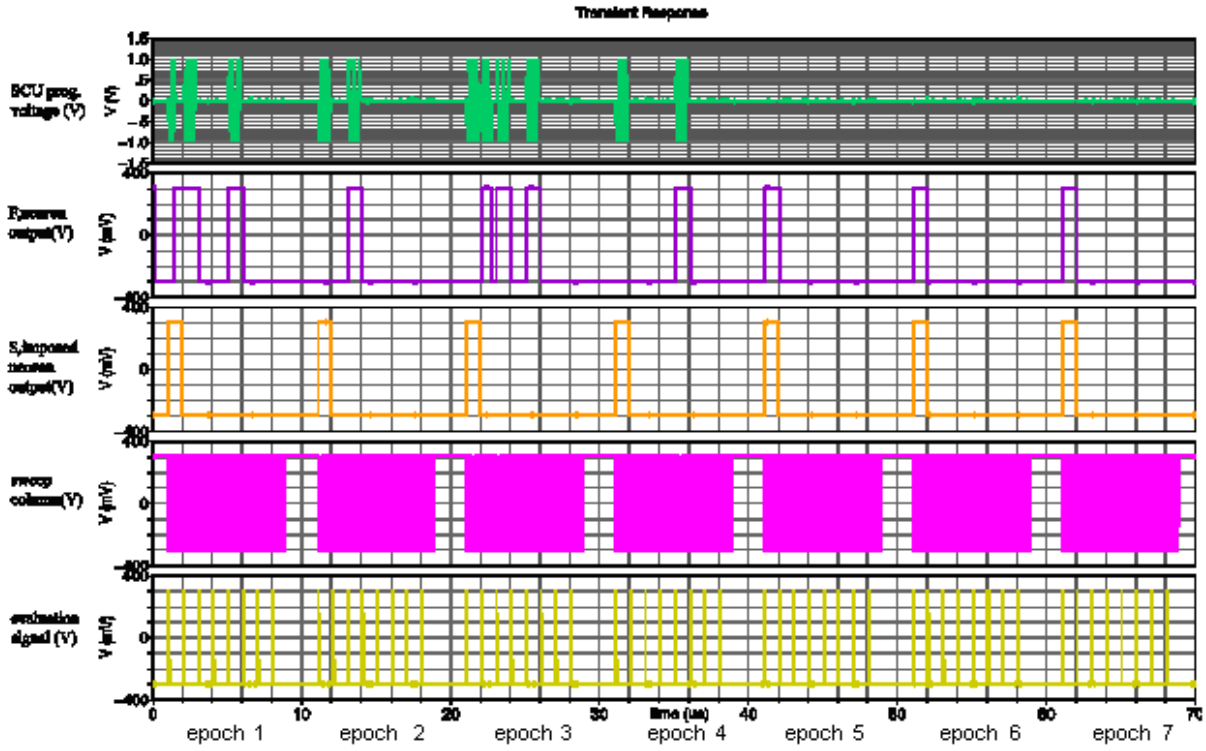


Figure 60. Le chronogramme du circuit d'apprentissage, pour la fonction NOR3

Durant la simulation, certains blocs analogiques du circuit sont écrits directement en verilog-A, pour pallier aux difficultés de convergence de simulations. La résistance de contact du nanocomposant CBRAM a été ramenée à 10 kΩ. D'autres fonctions logiques ont été simulées par apprentissage. Dans notre modèle, elles doivent être linéairement séparables. L'apprentissage du réseau dure typiquement de 2 à 5 époch pour les fonctions à 3 entrées. Les poids synaptiques s'étendent sur une plage autour de 100 kΩ.

	W0	W1	W2	Wth
OR3	2,23	2,29	2,25	2,20
MIN3	-2,47	-2,11	-2,11	4,15
AND3	2,83	2,29	2,25	6,75
MAJ3	2,61	2,67	2,62	4,15

Tableau 18. Les valeurs de résistances converties aux poids synaptiques relatifs

10. Conclusion

Les nanocomposants, comme les nanotubes de carbone, et les mémoires à conductances multi-niveaux pourront faire partie de la technologie de semi-conducteur du futur. Dans ce chapitre, la programmation de fonctions logiques a été simulée dans un circuit électrique incluant des modèles idéaux de nanocomposants CBRAM. Leurs résistances variables représentant les synapses, sont de nature multi-niveaux, s'étalant sur une zone programmable 1kΩ – 1MΩ. La modélisation du nanocomposant tire son inspiration dans les phénomènes physiques observables, certaines suppositions seront encore à vérifier. Pour démontrer la faisabilité de la conception dans ce contexte idéal, nous avons fait une simulation électrique de l'apprentissage d'une porte logique. Le réseau de neurones se comporte comme la porte logique désignée. On peut prétendre que dans ce contexte le réseau de neurones incarne une solution d'alternative efficace. Par la suite, nous allons nous intéresser à la robustesse. En effet, l'auto-correction des dispersions de caractéristiques du circuit est alors possible à travers le processus d'apprentissage. C'est ce que j'appelle la méthode d'inclusion de défauts. Elle propose une façon d'allier la tolérance aux défauts et l'auto-correction. J'ai montré que cette approche est réalisable avec les réseaux de neurones. De plus, leur redondance intrinsèque pour l'auto-compensation d'erreurs me paraît intéressante. Il s'agit d'étudier la robustesse en post-process, traité dans le chapitre suivant.

CHAPITRE 4 : ROBUSTESSE

1. Introduction

Pour augmenter la fiabilité, un système basé sur des composants peu fiables peut être construit à partir de la redondance des modules. Dans ce cas, on peut chercher à substituer un ensemble de modules défaillants par des modules supplémentaires dépourvus d'erreur. Dans une architecture reconfigurable tolérante aux défauts [Deho05], [Mori07], la reconfiguration consiste à détecter et localiser les modules défaillants, rerouter la structure afin de reconstruire un système fiable. Cependant, la surface allouée aux modules redondants est nécessairement limitée. Une autre façon de faire face à cette difficulté consiste à utiliser des codes correcteurs d'erreur, mais là encore, des portes logiques supplémentaires doivent être ajoutées. Dans ces architectures [Deho05], [Mori07] la densité d'intégration est réduite à cause de la redondance.

Compte tenu de ces limitations, je suis amené à étudier une autre alternative. On peut considérer les réseaux de neurone intrinsèquement redondants et tolérant aux défauts [Belf89], [Dama89], [Phat92], [Chey97], [Schm03]. Cette redondance est principalement due à un ensemble de neurones en surnombre. La question est alors la suivante, peut-on envisager de faire fonctionner un système de traitement d'information basé sur une approche neuronale en présence de défauts ? Dans ce cas, on peut envisager l'auto-correction d'erreur grâce aux remaniements des poids synaptiques par un algorithme d'apprentissage.

Dans ce chapitre, divisé en cinq sections, je me propose d'étudier la tolérance aux fautes des RNF (Réseau de Neurones Formels) sur une architecture de topologie particulière. La première section introduit les réseaux de neurones et la redondance. J'y explique la notation utilisée par la suite. La deuxième section contient la description des modèles de défauts implémentés dans la simulation fonctionnelle. Puis, la troisième section décrit le type d'algorithme utilisé pour la simulation. La quatrième section présente les résultats de simulations. A partir de ces

modèles de défauts, dans la cinquième section je modélise l'effet d'un apprentissage sur un réseau contenant des défauts. Enfin, je conclurai à partir de ces observations.

2. Réseaux de neurones et redondance

Les études précédentes [Beiu04] ont montré que le calcul logique peut être effectué de façon radicalement différente du modèle de von Neumann. En intégrant l'utilisation de la conductance multi-niveau, des synapses programmables pourront être incorporées dans un réseau de neurones avec une grande densité. En effet, il est possible de réaliser les neurones avec des composants actifs classiques, et les synapses en nanocomposants. Cela présente un double avantage. Premièrement la technologie des transistors traditionnels est bien maîtrisée. Deuxièmement les synapses faites de nanocomposants auront une taille suffisamment petite pour être massivement intégrées autour du neurone. Ceci doit permettre d'atteindre un haut degré d'intégration, tout en conservant une forte fiabilité pour la partie la plus importante du circuit.

Dans ce contexte, nous allons considérer l'impact des défauts présents dans la partie de l'architecture utilisant des nanocomposants. Pour cela, nous évaluons les possibilités d'auto-correction par apprentissage et préciserons les architectures de réseaux permettant d'atteindre un niveau de redondance suffisant. Intrinsèquement, la redondance existe dans un réseau de neurones sous la forme des neurones cachés supérieurs au nombre requis pour la convergence de l'apprentissage. Nous avons choisi de quantifier cette redondance par le rapport entre le nombre de neurones cachés implantés et le nombre de neurones minimum nécessaire à l'apprentissage. Cependant, cela ne protège pas des erreurs provenant des entrées et des sorties. Pour dépasser cette limitation, il est nécessaire de répliquer les entrées et les sorties. Nous verrons qu'il se constitue alors des sous réseaux neuronaux qui agissent comme un système global capable de répondre correctement à chaque stimulus de la base d'apprentissage. Une fonction logique peut être programmée par apprentissage dans un ensemble de sous réseaux. Les signaux d'apprentissage peuvent être envoyés simultanément à tous les sous réseaux. Au final le comportement chaotique des sous réseaux défaillants est compensé par

l'effet de redondance. En effet, la réponse étant fonction de la somme pondérée des entrées reçues, les sous-réseaux fautifs devraient, si l'algorithme d'apprentissage est adapté, se voir attribuer des poids synaptiques plus faibles que les réseaux fonctionnels. D'abord, je vais expliquer le schéma conçu de réseaux redondants tolérant aux défauts :

Premièrement, les signaux d'entrée sont recopiés sur plusieurs voies par la première couche de neurones. Les mêmes informations copiées se propagent vers les différents sous réseaux de neurones.

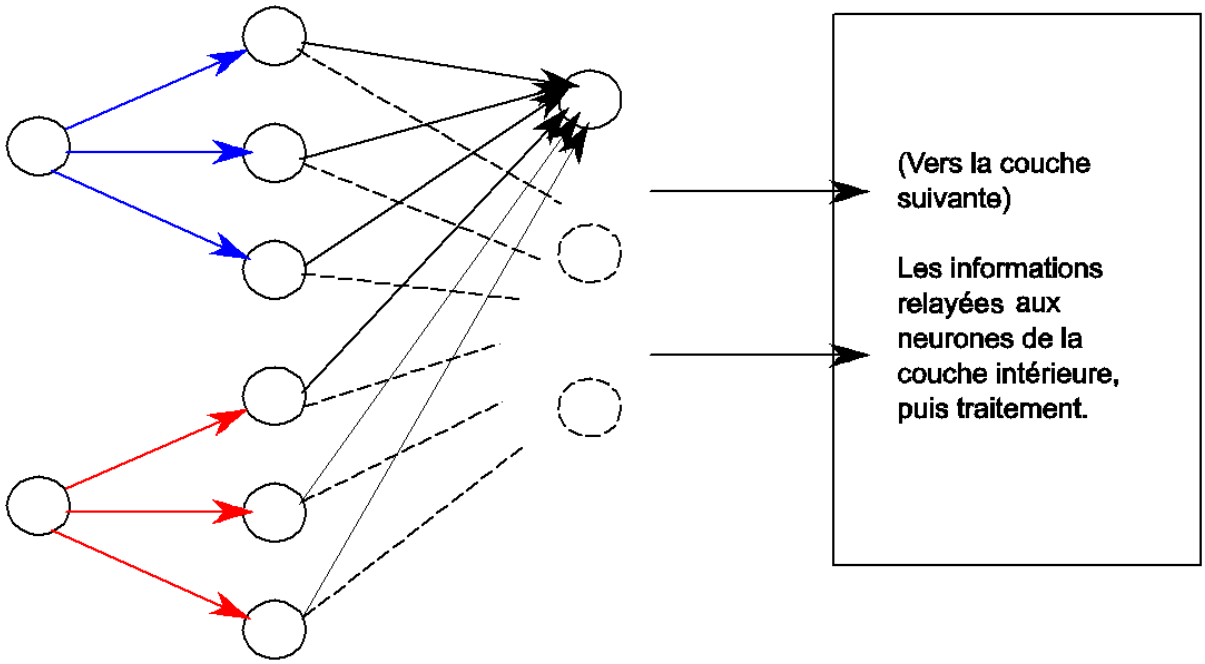


Figure 61. Schéma de la distribution d'informations

Dans l'exemple de la figure 61, la même information est représentée en trois fois. Il appartient aux neurones recevant les signaux de les relayer vers les couches suivantes. Cette topologie permet de véhiculer les entrées dans les différents sous réseaux afin de conserver l'intégralité de l'information.

Ensuite, la multiplication du nombre de neurones cachés conduit à une forme de redondance intrinsèque. Le nombre total de neurones cachés est fonction du nombre de neurones de sous réseau et du nombre de sous réseaux. Pour une fonction donnée, il existe un réseau de taille minimale capable d'effectuer

l'apprentissage. Le taux de redondance est donc le rapport entre le nombre total de neurones des sous réseaux et le nombre de neurones cachés dans le réseau minimal.

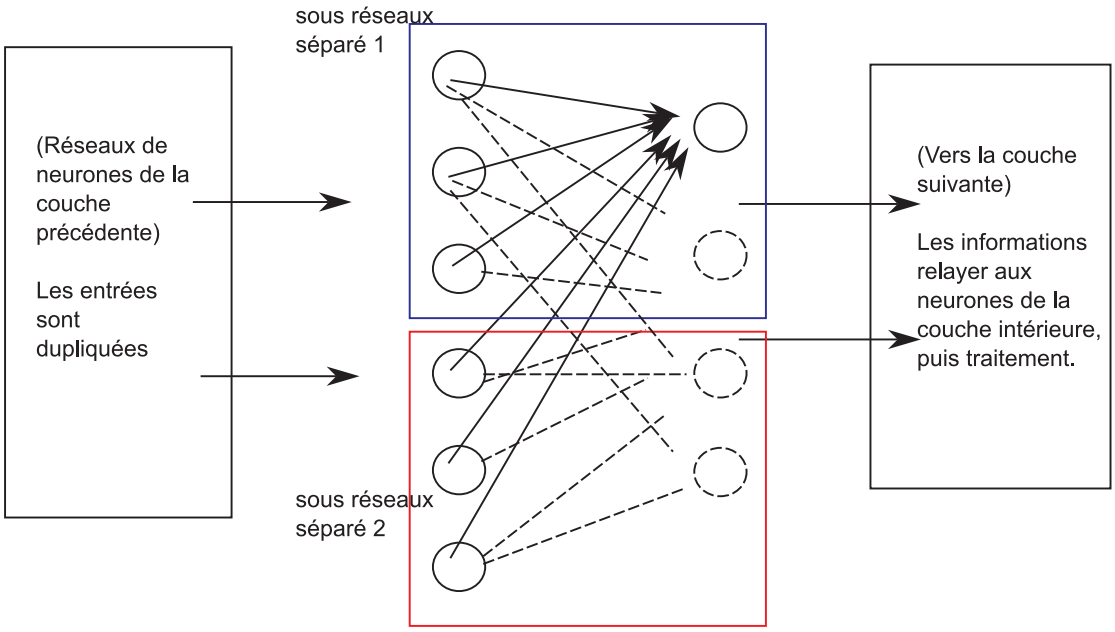


Figure 62. Schéma du traitement d'informations en parallèle

Enfin, la sortie est naturellement dotée d'une fonction de vote majoritaire (cf. figure 63). Il s'agit d'une fonction à poids synaptiques positifs. Elle prend en compte l'état majoritaire parmi les sorties de l'ensemble des sous réseaux. Selon l'étude [Beiu05], l'usage de la porte majoritaire peut devenir très efficace pour une probabilité de défauts donnée, et à un facteur de redondance définie par la relation R=N+N/k dans [Beiu05], N étant le nombre de sortie, et k le nombre d'étage de multiplexage de von Neumann.

La phase d'apprentissage doit déterminer automatiquement les réseaux pertinents à prendre en compte, et atténue les réponses des réseaux fautifs.

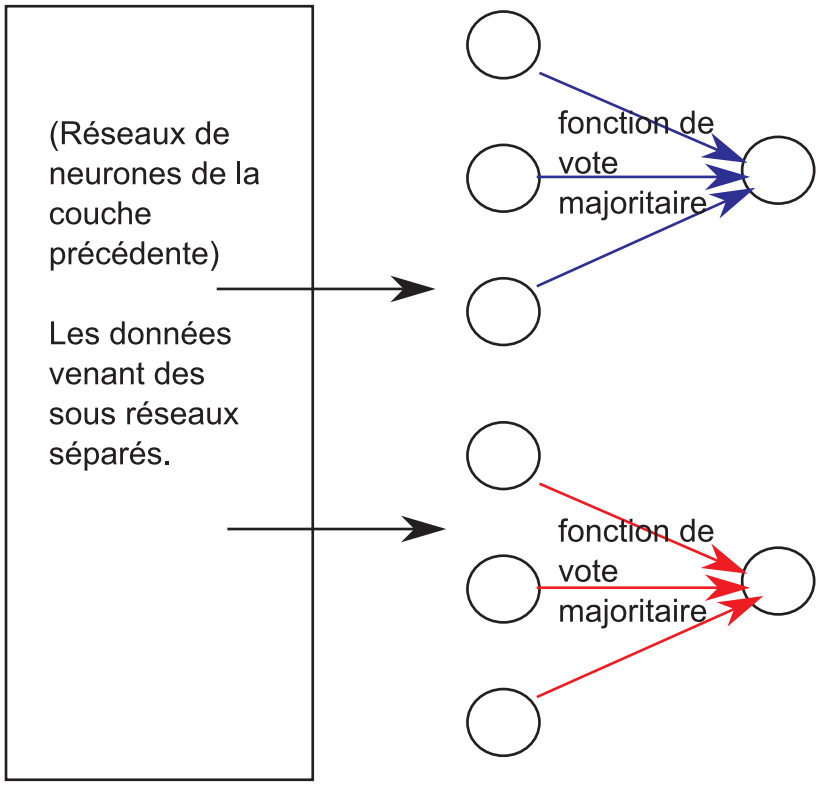


Figure 63. Schéma de la collecte d'informations

Dans la suite, la topologie des réseaux, notée K*i*-L*j*-M*k*, sera caractérisée par les trois paramètres entiers positifs *i*, *j* et *k*. Le premier paramètre *i* correspond au nombre de multiplications des entrées. Le deuxième paramètre *j* donne le nombre de neurones dans la couche cachée. Le troisième paramètre *k* est associé à la fonction de vote majoritaire permettant d'effectuer un filtrage des sorties des différents sous-réseaux, ainsi le paramètre *k* représente le nombre de sous réseaux en parallèle.

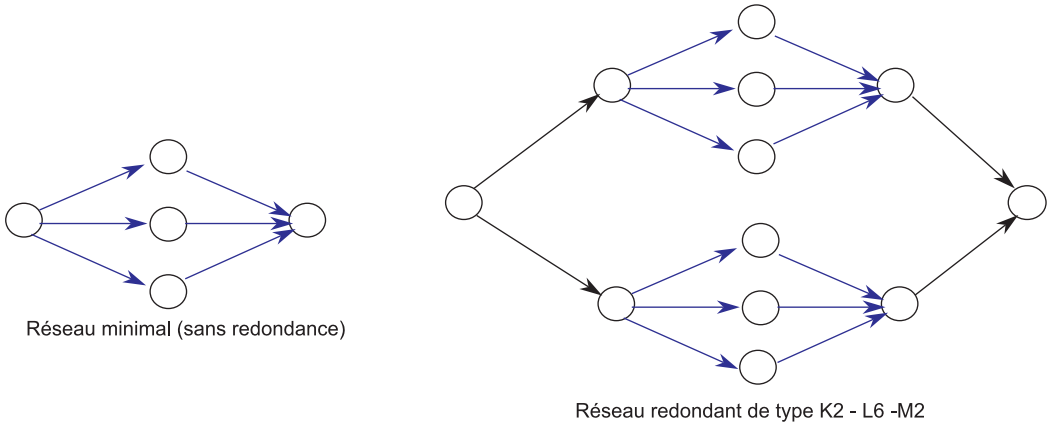


Figure 64. Un réseau sans redondance en comparaison avec une topologie de redondance de type K2-L6-M2

Par exemple, un réseau de topologie « K2 – L6 – M2 » indique que les signaux d'entrée sont dupliqués deux fois, et passe par une couche intermédiaire de 6 neurones cachés au total. Dans une configuration générale, le même neurone peut servir dans différents sous réseaux. Finalement, la décision est issue d'un vote majoritaire des sous réseaux. Ici je prends l'exemple de deux sous réseaux composés d'un nombre total de six neurones, c'est-à-dire trois neurones cachés par sous réseaux. Enfin, la sortie filtre deux sous réseaux. L'interaction des modules doit prémunir le système contre un mauvais fonctionnement. Avant d'exposer la série de simulations dans le but de tester le comportement de l'apprentissage d'une fonction, les modèles de fautes sont expliqués ci-après.

3. Modèle de faute considéré

Le modèle de défauts doit représenter les défaillances possibles des nanocomposants. Il s'agit essentiellement de modéliser les fautes dues aux synapses et celles dues aux neurones. Typiquement le collage à un état invariable serait dû à un défaut du composant. C'est le cas traité dans notre étude.

Pour un neurone, il s'agit du collage de l'état (Stuck-At) de sa sortie. Il présente donc un état indépendant des stimuli. Pour une synapse, cela se traduit par un état de résistance constant, sur lequel l'apprentissage n'a pas d'effet.

Dans nos simulations, les états des neurones et des synapses défectueux seront tirés au sort au temps initial. A chaque couche de neurones et à chaque matrice synaptique nous associons une probabilité de défauts. Les neurones sont déclarés défectueux par un tirage au sort initial à partir de la probabilité de défaut associée à leur couche de neurone. Lorsqu'ils sont déclarés défectueux, l'état de collage est tiré au sort pour toute la durée de la simulation, chaque état étant équiprobable. Les neurones défectueux et leur état de collage sont mémorisés dans deux tables. Lors du calcul des états des neurones de la couche considérée, les états des neurones défectueux sont remplacés par les états mémorisés dans la table.

Pour la génération de fautes concernant les synapses défectueuses, les synapses sont également touchées selon une probabilité prédéfinie. Leur poids synaptique est alors collé est à une valeur quasi infinie (court circuit), ou à contrario à une valeur nulle (circuit ouvert). Tous les poids synaptiques intermédiaires peuvent être corrigés par compensation.

4. Algorithmes d'apprentissage

Dans cette section je suis amené à décrire les deux algorithmes utilisés dans le cadre des simulations. Il s'agit de la machine de Boltzmann d'une part et la règle de la rétro-propagation d'autre part. J'ai déjà montré au chapitre précédent comment les deux algorithmes se rejoignaient lorsque l'on considère un réseau monocouche à température nulle. J'exploite cette similitude pour combiner les avantages des deux types d'apprentissage sur les mêmes réseaux. Dans un réseau multi-couche, l'apprentissage utilisant la rétro-propagation du gradient converge très rapidement comparé à l'apprentissage de la machine de Boltzmann. Nous pouvons donc l'utiliser pour configurer initialement un réseau off-line correspondant à une fonction logique.

Cependant, l'apprentissage par comparaison des cooccurrences d'activation utilisé dans la machine de Boltzmann est bien plus aisé à implémenter matériellement sur un circuit pour réaliser l'apprentissage on-chip, puisqu'il est purement local et ne nécessite pas le calcul de la dérivée de la fonction d'activation. Il est donc intéressant de l'utiliser comme algorithme de réapprentissage on-chip de façon à compenser les défauts. Pour que les deux algorithmes d'apprentissage soient compatibles, nous avons adapté la topologie de la machine de Boltzmann de façon à considérer des connexions synaptiques asymétriques.

Le code Matlab ® (Markworks) ci-dessous implémente la simulation de l'apprentissage pour une machine de Boltzmann. L'opérateur Kron est le produit tensoriel de Kronecker :

```
%%phase -
%%force l'entrée, sortie libre
for stabiliz=1:maxsteps
   for iLevel=1:nLevel-1
      XP{iLevel+1}=bm_active_f(W{iLevel}'*Act_State{iLevel},temp(stabiliz));

       Pij_relax{iLevel}=Pij_relax{iLevel}+kron(Act_State{iLevel},XP{iLevel+1}')+k
       ron(Act_State{iLevel+1}(1:nCells(iLevel+1))',Act_State{iLevel});
   end;
end ;

%%phase +
%%force l'entrée et la sortie
for stabiliz=1:maxsteps
   for iLevel=1:nLevel-1
      XP{iLevel+1}=bm_active_f(W{iLevel}'*Act_State{iLevel},temp(stabiliz));

       Pij_force{iLevel}=Pij_force{iLevel}+kron(Act_State{iLevel},XP{iLevel+1}')+k
       ron(Act_State{iLevel+1}(1:nCells(iLevel+1))',Act_State{iLevel});
   end;
```

```
        Pij_force{nLevel}=Pij_force{nLevel}+kron(Act_State{nLevel},XP{nLevel+1}'
        )+kron(Act_State{nLevel+1}',Act_State{nLevel});
    end;

    for iLevel=1:nLevel
      W{iLevel}=W{iLevel}+eta*(Pij_force{iLevel}-0.01)/(2*maxsteps);
    end;
```

Passons maintenant au deuxième algorithme implémenté: la rétro-propagation. L'estimation du signal d'erreur diffère suivant les neurones considérés. Pour les cellules de la couche de sortie, l'erreur est évaluée en comparant la réponse donnée par la cellule avec la réponse attendue.

Soient :

e, vecteur d'erreur
S, réponse désirée
F, réponse du réseau

L'erreur pour le k-ième stimulus vaut donc :

$$e_k = (S_k - F_k)$$

Le signal d'erreur prend en compte l'erreur commise par la cellule et l'état d'activation de la cellule. Pour la N^{ième} cellule de la couche de sortie, ceci est défini comme :

$$\delta_N = \eta . e_k . f'(W_N . N)$$

f ' représente la dérivée de la fonction d'activation. Si elle est de type sigmoïde, cette dérivée équivaut au produit de la fonction f et (1-f). Le gain multiplicatif η est appelé le pas d'apprentissage, qui devrait être pris en compte pour modifier les poids. Il peut éventuellement diminuer au cours de l'apprentissage. Dans mes simulations η

restera constant, aucune variation n'interviendra.

A partir de cette erreur, je calcule la variation synaptique. Elle correspond au produit du signal d'erreur et de l'état du neurone présynaptique.

$$\Delta W = X_{N-1}.\delta_N$$

code Matlab :

```
Error{nLayer}= target(np,:) - Act_State{nLayer};
Derr{nLevel}=Error{nLayer}.*bp_active_fp(N_WSum{nLayer});
W{nLevel}=W{nLevel}+(eta*(Act_State{nLevel}'*Derr{nLevel}));
```

Pour les cellules de la couche cachée, l'erreur de la couche suivante est attribuée aux neurones de la couche précédente proportionnellement au poids synaptique qui les relie :

$$\delta_N = \eta.(W_{N+1}.e_{N+1}).f'(W_N.X_N)$$

code Matlab :

```
for NumLayer=nLevel-1:-1:1
    if NumLayer>1
        Error{NumLayer}=Derr{NumLayer}(1:NeurLayer(NumLayer))*W{NumLayer}';
    end;
    Derr{NumLayer}=Error{NumLayer+1}.*([bp_active_fp(N_WSum{NumLayer+1}),
    seuil_0]);

    W{NumLayer}=W{NumLayer}+(eta*(Act_State{NumLayer}'*Derr{NumLayer}(1:N
    eurLayer(NumLayer))));
end;
```

5. Résultats des simulations

- ### A. Comparaison des algorithmes d'apprentissage

Pour valider la robustesse de l'approche calculatoire par RNF, j'ai effectué la simulation de l'apprentissage d'un additionneur 1-bit (Full-Adder). L'architecture du réseau correspond à un Perceptron multi-couche, les informations s'y propageant uniquement de l'entrée vers la sortie, l'algorithme d'apprentissage utilisé est celui de la machine de Boltzmann. L'additionneur est pourvu de 2 entrées et 2 sorties (la retenue C et la somme partielle S). En logique CMOS, cette fonction peut être réalisée avec une porte majoritaire pour la retenue et la porte OU-Exclusif pour la somme. Basé sur le paradigme du RNF on peut implémenter l'additionneur en utilisant des neurones interconnectés. Je vais d'abord décrire les protocoles d'expérimentation.

Pour les simulations numériques les algorithmes nécessaires ont été implémentés à l'aide du logiciel Matlab. Le bon comportement de l'apprentissage du réseau normal sans injection de faute est vérifié avant les simulations fonctionnelles comportant les fautes.

L'objectif de cette simulation permet dans un premier temps de vérifier la convergence attendue de l'apprentissage selon les différents algorithmes. Le comportement final du réseau après l'apprentissage doit converger vers le modèle d'objectif. Puis elle offre la possibilité de comparer la vitesse de l'exécution de l'apprentissage, mise en avant par le lancement de deux tests algorithmiques qui sont la rétro-propagation, et la machine de Boltzmann.

Algorithmes	Rétropropagation	Machine de Boltzmann
Topologie	1x-3N-1x	1x-4N-1x
Résultat	3 s	45 s

Tableau 19. Comparaison de convergence d'un Full-Adder 1-bit. Simulations effectuées sur un Pentium IV 640 Prescott (90nm, socket 775 LGA) 3.2 GHz

Pour l'apprentissage de la fonction Full-Adder, on constate que la durée de l'apprentissage est 15 fois plus courte avec l'algorithme de la rétro-propagation. En effet, la machine de Boltzmann utilise des calculs intensifs pour le calcul de la statistique interne, ce qui explique le processus d'apprentissage soit plus long. Néanmoins les poids synaptiques finaux issus de ces différents algorithmes sont de valeurs relatives proches. La compatibilité entre modèles permet de constater le caractère universel de l'apprentissage [Hopf82].

On note cependant que, pour l'apprentissage des fonctions non triviales, la convergence peut échouer, ne fait état d'aucune convergence après un grand nombre d'itérations, fixé au maximum à 3000 dans mes simulations. Ces fonctions se décomposent en générales en différentes étapes de calcul combinant de nombreuses entrées et des résultats de calculs intermédiaires.

- **B. Détermination des tailles de réseau minimal**

Que ce soit dans la perspective d'optimiser la vitesse d'apprentissage logiciel (off-line) ou pour minimiser le nombre de modifications de conductance à opérer sur les nanocomposants servant de synapses, l'impact de l'architecture du réseau sur la vitesse de convergence de l'apprentissage doit être étudié. Une topologie de réseau de neurone est soit variable, soit fixe. Dans le premier cas, on peut citer l'algorithme constructif dit « Constructive Learning Algorithme » [Meza89]. Il débute avec un petit réseau de neurones (généralement un seul neurone suffit pour démarrer) qui se construit progressivement pour atteindre le nombre « presque minimal » (near minimal). De manière dynamique le développement du réseau se fait par l'ajout de neurones et de leurs connexions synaptiques, jusqu'à la convergence. A l'inverse, on peut évoquer l'algorithme destructif, dit « Destructive Learning Algorithme » [Sank91], dont le principe est l'inverse du précédent. Les neurones sont retirés du réseau au fur et à mesure. Du fait de leur complexité, ce type d'architecture n'est pas envisagé ici.

Il existe, pour toute fonction, une taille de réseau minimale, pour que l'apprentissage puisse converger en temps fini. Des approches théoriques

permettent d'encadrer le nombre de cellules cachées et la taille du poids synaptique [Beiu03-2] pour l'addition. L'objectif est de quantifier la complexité du réseau de neurones. Dans nos tests, le nombre de neurones caché maximum est fixé à 25. L'entropie de la fonction est une information utile pour déterminer le nombre total de bits que doivent contenir tous les poids synaptiques, mais son estimation reste un problème délicat [Beiu03-2]. Dans les simulations suivantes, on suppose que le dynamique range est suffisamment large, où la précision de poids ne pose pas de limite qui fera échouer la convergence. Il serait alors intéressant de s'appuyer sur les tests de convergence des algorithmes d'apprentissage pour trouver le nombre de neurones requis pour former un réseau minimum de calcul. Cette taille de réseau dépend de la complexité de la fonction à apprendre. En particulier, lorsque le nombre des entrées de la fonction augmente, la quantité d'information augmente. Pour donner un élément de comparaison, la capacité mémoire nécessaire pour implanter une fonction logique avec une table de mémorisation (LUT) croît en 2^N. De même, le nombre de stimuli exhaustif à présenter au réseau pour apprendre une fonction logique en couvrant toutes les configurations d'entrée possible croît également en 2^N. Pour quantifier cette complexité, nous avons cherché à déterminer la taille minimale du réseau capable d'apprendre une fonction logique.

Additionneur N-Bits	Nombre minimum neurones (+1 : neurone de seuil)
1	1+1
2	4+1
3	5+1
4	8+1
5	12+1
6	16+1
7	22+1
8	24+1

Tableau 20. Test de convergence de réseaux minimums de l'additionneur (full-adder, avec somme et retenue), la complexité du RNF est bornée par $\Omega(3N+\varepsilon)$.

Ainsi, en comparant l'apprentissage de deux fonctions élémentaires, d'une part un additionneur (cf. tableau 20) et d'autre part un multiplieur (cf. tableau 21), on peut remarquer que pour le même nombre d'entrée, le multiplieur nécessite un nombre de neurones plus important par rapport à l'additionneur pour constituer un réseau minimum. Le nombre de neurones nécessaire pour effectuer une opération arithmétique et logique dépend grandement de sa complexité.

Multiplieur N-Bits	Nombre minimum neurones (+1 : neurone de seuil)
1	1+1
2	4+1
3	22+1

Tableau 21. Test de convergence de réseaux minimums du multiplieur.

Le nombre d'itérations et la durée de l'apprentissage deviennent rédhibitoires lorsque le nombre d'entrée augmente. Il y a donc nécessité de décomposer les fonctions à N entrées en un ensemble de fonctions à M entrées (avec M<N), pourvu qu'elle puisse être scindées en plusieurs fonctions minimales [Zhan04]. Les simulations expérimentales faites ici permettent de vérifier la convergence d'apprentissage sur l'Additionneur et le Multiplieur, ceci en fonction des nombres de bits d'entrée, le nombre de neurone, et le nombre de stimuli par époch.

Il est intéressant de constater (cf. tableau 22) les cas où en fixant le nombre de neurones au nombre d'entrées, et en considérant un neurone caché supplémentaire, le nombre de fonctions pouvant être appris. Les fonctions dites linéairement séparable peuvent être apprises en nécessitant seulement une couche, sans neurones intermédiaires. L'algorithme d'apprentissage utilisé est la règle delta. Pour certaines fonctions non linéairement séparables, nous pouvons constater que pour les fonctions à 2 entrées binaires, toutes les fonctions peuvent être apprises avec un neurone caché supplémentaire. Pour 3 entrées binaires, 60 % de fonctions pouvant être apprise avec un neurone supplémentaire, et dans la même configuration pour 4 entrées binaires, seulement 6 % de l'ensemble des fonctions pouvant être appris

avec une convergence certaine. Ce chiffre peut être revu à la baisse à cause de la dynamique de poids plus limitée en réalité qu'en simulation.

N entrées	Le nombre total de fonctions 2^{2^N}	Le nombre de fonctions pouvant être apprises en une couche	Le nombre de fonctions pouvant être apprises avec un neurone caché suppl.
2	16	14	16
3	256	104	153
4	65536	1882	3911

Tableau 22. Test de convergence de réseaux ayant le nombre de neurones égal au nombre d'entrées, et plus un neurone caché.

On constate un gain en stabilité et performance de convergence en segmentant la complexité de la fonction à apprendre. Ici, il serait avisé de fragmenter l'apprentissage d'une fonction à N entrées.

- **C. La robustesse de redondance des réseaux de neurones**

J'ai cherché à évaluer l'apport des différentes formes de redondance en termes de robustesse face aux différents types de défauts dans un réseau de neurone. L'algorithme d'apprentissage de réseaux de neurones choisi est la machine de Boltzmann. Le nombre d'itération maximum est plafonné à 3000 itérations, ce qui est très supérieur au nombre attendu pour la convergence normale. La fonction à apprendre par le réseau est le Full-Adder.

En testant différents nombres de neurones dans la couche cachée, la convergence est testée avec des réseaux redondants. Par exemple en choisissant arbitrairement 6 neurones en entrées, 8 neurones dans la couche cachée, et 4 neurones en sortie, on peut étudier la robustesse d'un réseau redondant composé de sous réseaux. En respectant la nomenclature définie au paragraphe précédent, le réseau déterminé est de type K2-L8-M2.

La conséquence de l'apprentissage en présence de neurones redondants est perceptible directement au niveau de la matrice de poids synaptiques. Il est relativement facile d'identifier des neurones remplissant les mêmes rôles et formant ainsi les différents sous réseaux. Le tableau ci-dessous illustre la similitude des relations synaptiques dans les trois groupes de neurones cachés identifiés, les neurones de différents sous réseaux accomplissant les mêmes rôles ont les poids synaptiques identiques ou presque avec d'autres neurones :

La relation synaptique entre neurones					
0,0207	-0,0005	0,0206	-0,0005	0,0207	-0,0005
-0,0207	-0,0779	-0,0208	-0,0779	-0,0207	-0,0779
0,0201	-0,0803	0,02	-0,0803	0,0201	-0,0803
-0,0594	-0,0404	-0,0595	-0,0404	-0,0594	-0,0404
-0,0604	0,0378	-0,0603	0,0378	-0,0604	0,0378
0,0583	0,0403	0,0584	0,0403	0,0583	0,0403
0,1378	-0,08	0,1379	-0,08	0,1378	-0,08
0,0194	0,0396	0,0195	0,0396	0,0194	0,0396
GROUPE 1		GROUPE 2		GROUPE 3	

Tableau 23. Les poids synaptiques redondants

Ensuite, les erreurs sont réparties sur les trois couches. Pour simplifier l'interprétation, les résultats présentés correspondent à l'introduction d'erreur de collage uniquement dans une couche à la fois. Chaque ligne du tableau porte sur un ensemble de 10 simulations correspondant à 10 tentatives d'apprentissage. Pour chacune, on relève le nombre de fois où l'apprentissage a réussi à converger vers l'ensemble des bonnes réponses.

Entrée		Cachée		Sortie		Expériences	Succès
Neurone	Nb. collage	Neurone	Nb. collage	Neurone	Nb. collage		
6	1	8	0	4	0	10	10
6	2	8	0	4	0	10	7
6	3	8	0	4	0	10	5

Tableau 24. Simulation de fautes injectées aux neurones d'entrée

Entrée		Cachée		Sortie		expériences	succès
Neurone	Nb. collage	Neurone	Nb. collage	Neurone	Nb. collage		
6	0	8	1	4	0	10	9
6	0	8	2	4	0	10	7
6	0	8	5	4	0	10	3

Tableau 25. Simulation de fautes injectées aux neurones de la couche cachée

Entrée		Cachée		Sortie		expériences	succès
Neurone	Nb. collage	Neurone	Nb. collage	Neurone	Nb. collage		
6	0	8	0	4	1	10	10
6	0	8	0	4	2	10	8
6	0	8	0	4	3	10	3

Tableau 26. Simulation de fautes injectées aux neurones de sortie

Ces simulations montrent d'abord que la correction d'erreurs par le processus d'apprentissage peut être efficace et caractérisable. Un neurone défectueux, ayant son état collé à 1, se comporte en réalité comme une unité de seuil supplémentaire. Cependant, l'information qu'il doit relayer est perdue. Le réseau n'est donc réparable par apprentissage qu'à la condition que les neurones fonctionnels constituent encore un sous-réseau suffisant pour réaliser la fonction à apprendre. À partir de cette hypothèse le taux de succès de l'apprentissage peut être déterminé analytiquement. Je vais le décrire dans la section suivante.

6. Prévision de la probabilité du succès de l'apprentissage

Les simulations précédentes mettent en évidence certaines caractéristiques de robustesse dans les réseaux de neurones formels. On peut observer l'effet de l'interaction des sous réseaux de neurones. Quelque soit l'environnement, tant qu'il y a un parmi N sous réseaux constitutifs dépourvu de défauts, et que les sous-réseaux dotés de comportements erronés peuvent être inhibés, le système global peut continuer à fonctionner normalement. Cela est supposé possible grâce à l'ajustement des poids synaptiques lors de l'apprentissage.

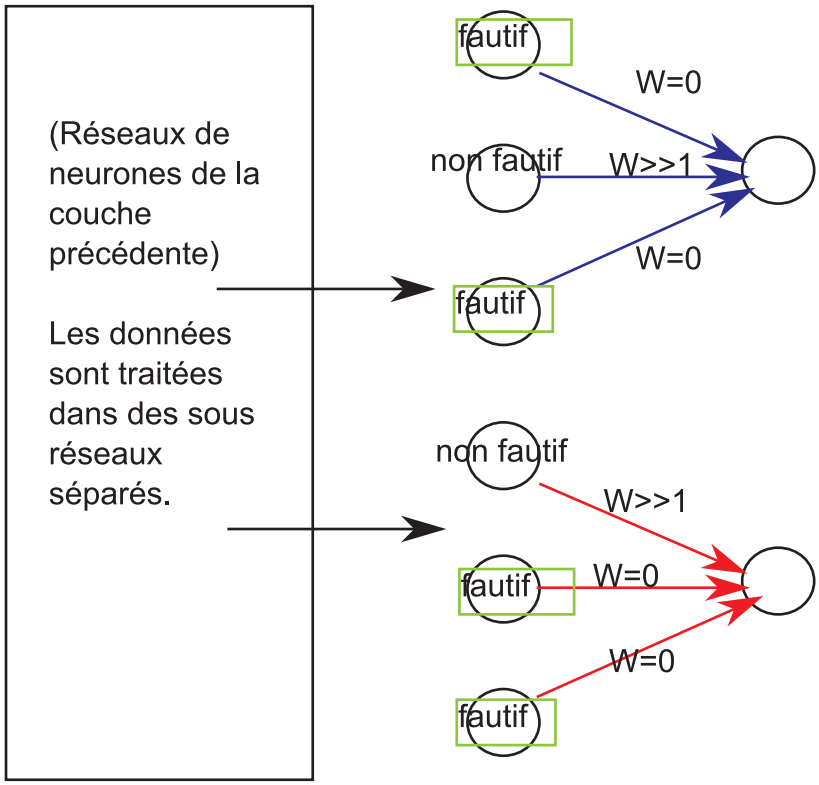


Figure 65. Schéma de pré-sélection d'informations

Soit *y* la probabilité de défaut pour chaque neurone. Pour que l'information associée à ce neurone soit perdue il faut que tous les neurones redondants soient défectueux. Avec une redondance *r*, cet évènement a une probabilité y^r. On peut en déduire la probabilité de $(1 - y^r)$ d'avoir au moins une entrée correcte relayée vers la couche de duplication. Soit n le nombre d'information d'entrées primaires (avant duplication), la probabilité que toute information utile au traitement soit relayée est la suivante :

$$Ci = (1 - y^r)^n \qquad \textbf{(Eq. 17)}$$

Où :

Ci : la probabilité que toute information utile au traitement soit relayée

y : la probabilité de défaut pour chaque neurone

r : le nombre fois la même source d'information est répliquée

n : le nombre d'information de source unique

Cette relation permet de prédire la convergence d'un réseau de neurones à partir de la fiabilité de ses composants et de son degré de redondance. A l'inverse également, elle permet de calculer le degré de redondance nécessaire en fonction de la fiabilité connue (Eq. 18). Nous pouvons vérifier que cette estimation correspond aux résultats que nous avons obtenus dans les premières simulations fonctionnelles.

$$r = \frac{\log(1 - Ci^{\frac{1}{n}})}{\log(y)} \quad \textbf{(Eq. 18)}$$

Avec 33% de neurones défectueux :

On obtient 79% de taux de convergence prédit tandis que 7 / 10 de l'apprentissage réussi a été constaté.

Avec 50% de neurones défectueux :

On obtient 56% de taux de convergence prédit tandis que 5 / 10 de l'apprentissage réussi a été constaté.

Les résultats présentés ici sont proches du modèle simplifié.

L'apprentissage est une méthode de correction d'erreur. On peut affirmer que si chaque information irréductible a au moins une représentation correcte au sein du réseau de neurones, la convergence serait assurée. En d'autres termes, un ensemble d'informations intactes doivent être injecté aux réseaux pour que l'apprentissage soit réussi en un nombre d'époch fini.

Les références dont nous disposons n'ont pas de points en commun permettant de faire une comparaison par rapport ce qui a été fait dans cette partie. Dans les travaux de l'évaluation de la robustesse, le type de réseaux [Kerl94] avec redondance de neurones cachés n'a pas été étudié. Dans [Phat95], le problème de

l'encoder 3-2-3 a été évoqué, l'auteur n'a pas communiqué le taux de défauts utilisé dans son article pour établir une comparaison plausible. Dans [Vela00], les fautes transitoires ont été prises en compte et non les défauts du réseau.

7. Comparaison de LUT-SRAM et LUT-Réseau-de-neurones

Dans les circuits reconfigurables, les LUT permettent sont des portes logiques reprogrammables, l'information de la table de vérité est stockée dans sa mémoire binaire. Les réseaux de neurones peuvent également simuler les portes logiques. Au moyen d'un apprentissage, les réponses des neurones artificiels à un stimulus peuvent être imposées. Chaque neurone ne peut calculer qu'une fonction logique linéairement séparable. Cependant par des compositions un réseau à deux couches peut implémenter n'importe quelle fonction logique.

La Look-Up Table (LUT) est une structure de mémorisation de données, dans le cas du FPGA il est fait d'une matrice de points mémoires. Comme on sait qu'un réseau de neurones est une mémoire hétéro-associative, il est aussi capable d'effectuer des tabulations logiques. Dans l'implémentation des nanocomposants, les poids synaptiques sont de nature analogique multi-niveau. Maintenant nous prétendons qu'un réseau de neurones pourrait remplacer la matrice LUT traditionnelle, la question est qu'en est-il de la surface utilisée dans la puce en comparant les deux méthodes ? Nos estimations sont basées sur la comparaison du nombre de neurones et de leur surface avec la surface d'une SRAM capable d'implémenter les mêmes fonctions sous forme de LUT. Les estimations réalisées sont basées sur la projection d'une technologie dans le futur, le nœud 11 nm. A ce moment là, la surface d'une capacité minimale serait de 1100 nm^2 [ITRS - Table FEP6b], la longueur de grille du transistor est de 4 nm [ITRS - PIDS2b]. A partir de la densité de transistors (4 938 $Mtransistors/cm^2$), on en déduit que la surface moyenne de celui-ci est de 0,02 μm^2 (bien qu'il y ait une différence de taille entre le PMOS et le NMOS).

Connaissant les dimensions des cellules, on peut alors évaluer le rapport entre la surface utilisée par une LUT-Réseau-de-neurones et une LUT-SRAM.

$$R = \frac{\beta(S_{Neur}. + \alpha.S_{Syn})}{\kappa.S_{SRAM}} \quad \textbf{(Eq. 19)}$$

Où :

S_{Neur} : Surface occupée par la cellule de neurone

β : nombre de neurones

S_{Syn} : Surface occupée par la synapse

α : nombre de synapses par neurones

S_{SRAM} : Surface d'un point mémoire SRAM

κ : nombre total de points mémoires SRAM

R : rapport entre la surface de la LUT-réseau-de-neurones et la LUT-SRAM

Selon mes estimations concernant le nœud 11 nm, en se basant sur la structure de neurone [Mead89], soit 7 transistors et 2 capas, avec les capacités minimales, la surface du neurone serait autour de 0,4 μm^2, la synapse à 0,0004 μm^2 (20 nm x 20 nm) [Kund05]. Dans un premier temps on peut négliger la surface des synapses par rapport aux neurones, pour une connectivité relativement moyenne voire faible.

Un point mémoire SRAM (w/overhead) aurait une surface estimée de 0,02 μm^2 [ITRS 2007 - Executive Summary - Table 1h]. Pour N entrées, le nombre de points mémoires SRAM est de 2^N. La topologie de réseau de neurones est variable, elle dépend de la fonction à implémenter.

Maintenant, on peut calculer le nombre minimum de bit de redondance pour corriger une ou deux erreurs avec le codage de Hamming. Soit N le nombre de bits du code message, et m le nombre de bits du code de contrôle, pour corriger une erreur, l'équation à résoudre est $2^m \geq N + m + 1$. Pour corriger deux erreurs, l'équation est $2^m \geq \frac{(N + m).(N + m - 1)}{2} + N + m + 1$.

Cela donne en fonction du nombre d'entrée, la taille de la LUT en SRAM suivant :

Additionneur	N. Entrées	Taille κ de la LUT	Taille κ de la LUT avec 1 erreur corrigeable	Taille κ de la LUT avec 2 erreurs corrigeables
1 bit	2	4	7	10
	3	8	12	15
2 bit	4	16	21	25
	5	32	38	42
3 bit	6	64	71	76
	7	128	136	142

Tableau 27. En fonction de N entrées, la taille de la LUT représentant le nombre de points mémoires

Afin de limiter la taille des éléments combinatoire, la taille de la LUT doit conserver un nombre réduit d'entrées. La surface occupée, pour une application neuronale va dépendre de la complexité de la fonction et donc du nombre de neurones cachés, et aussi le nombre de valeurs que peut représenter un poids synaptique [Beiu03-2]. Dans une architecture neuronale, nous avons supposé qu'en ajoutant un neurone supplémentaire, une erreur peut être corrigée. Si on reprend l'exemple de l'additionneur (Full-Adder), à deux entrées (A, B) et deux sorties (somme et retenue), le nombre d'entrées serait le nombre total de bits de A et de B.

	Additionneur 1 bit			Additionneur 2 bit			Additionneur 3 bit		
Correction	0 erreur	1 erreur	2 erreurs	0 erreur	1 erreur	2 erreurs	0 erreur	1 erreur	2 erreurs
neurones cachés β	1	2	3	4	5	6	5	6	7
Surface µm2	0,4	0,8	1,2	1,6	2	2,4	2	2,4	2,8

Correction	0 erreur	1 erreur	2 erreurs	0 erreur	1 erreur	2 erreurs	0 erreur	1 erreur	2 erreurs
Taille κ LUT SRAM	8	12	15	32	38	42	128	136	142
Surface µm2	0,16	0,24	0,3	0,64	0,76	0,84	2,56	2,72	2,84

rapport R	2,50	3,33	4,00	2,50	2,63	2,86	0,78	0,88	0,99

Tableau 28. Rapport surfacique R entre deux méthodes d'implémentation (avec redondance) de la LUT pour un additionneur à N entrées

	Mult. 1 bit			Mult. 2 bit			Mult. 3 bit		
Correction	0 erreur	1 erreur	2 erreurs	0 erreur	1 erreur	2 erreurs	0 erreur	1 erreur	2 erreurs
neurones cachés β	1	2	3	4	5	6	22	23	24
Surface µm2	0,4	0,8	1,2	1,6	2	2,4	8,8	9,2	9,6

Correction	0 erreur	1 erreur	2 erreurs	0 erreur	1 erreur	2 erreurs	0 erreur	1 erreur	2 erreurs
Taille κ LUT SRAM	8	12	15	32	38	42	128	136	142
Surface µm2	0,16	0,24	0,3	0,64	0,76	0,84	2,56	2,72	2,84

rapport R	2,50	3,33	4,00	2,50	2,63	2,86	3,44	3,38	3,38

Tableau 29. Rapport surfacique R entre deux méthodes d'implémentation (avec redondance) de la LUT pour un multiplieur à N entrées

On peut calculer une estimation de la surface pour un additionneur tolérant aux défauts (correction d'une ou de deux erreurs) dans le cas de l'architecture neuronale, comme dans l'architecture en SRAM (cf. tableau 28, et 29). Selon [Ross03], avec la méthode du code correcteur d'erreur (11,8), pour un défaut de 10^{-4}, on peut tabler sur une moyenne de la redondance r=37,5%. Ce taux de composants en surplus permet d'assurer un fonctionnement correct. Dans les mêmes conditions, pour une architecture neuronale, la redondance requise peut être estimé à partir l'équation 18. On peut comparer les deux méthodes pour l'implémentation d'un additionneur en se référant au tableau 28, et puis 29, on constate qu'un rapport inférieur à 1 est atteignable. Cependant, il faut compter sur une fonction muni d'un nombre d'entrées et/ou sortie relativement importantes. En d'autres termes, le nombre de synapses multi-niveaux nécessaire à la convergence évolue moins vite que le nombre de points mémoire SRAM de la LUT. Ceci est valable pour une précision de poids synaptique suffisante pour effectuer la fonction. Pour effectuer une fonction à N entrées et S sorties le nombre de points mémoires SRAM évolue en $S.2^N$. Alors que pour réaliser la même fonction, le nombre de poids synaptiques évolue en $\alpha.\beta$. Il est ainsi remarquable que dans certaines conditions, la solution de réseau de neurones serait meilleure. L'amélioration postérieure concernera essentiellement la réduction de taille du neurone pour gagner plus d'efficacité.

8. Conclusion

Les architectures tolérantes aux défauts font partie des grands défis pour l'intégration des nanocomposants. La reconfiguration semble de prime abord l'une des méthodes les plus efficaces pour supporter un nombre élevé de défauts tout en limitant la surface ajoutée par les éléments redondants [Fors04]. Ceci peut être attribué à sa faculté d'éviter les défauts. On lui reproche cependant la grande difficulté de mise en œuvre qui requiert l'identification de tous les défauts puis le routage adapté à chaque composant fabriqué. Cette étape ne paraît pas réaliste. Je pense qu'une efficacité comparable peut être atteinte sans passer par cette étape en utilisant l'apprentissage neuronal comme méthode d'évitement des défauts.

J'ai donc traité dans ce chapitre l'étude de la robustesse à partir des simulations fonctionnelles dans le domaine des réseaux de neurones. La redondance des réseaux est une solution pour arriver à obtenir un système fiable en dépit des défauts de collages. Il faut alors multipliquer les entrées et les sorties. Les sous réseaux soutiennent un fonctionnement global correct. En effet, nous avons alors supposé que lorsqu'un sous réseau minimum fonctionnel existe dans le réseau global, la correction automatique est assurée. Nos simulations sont en accord avec cette hypothèse. Les résultats ont donc montré que l'implémentation d'une LUT dans un FPGA par un réseau de neurones apporte une robustesse face aux défauts considérés. Cette architecture devrait avoir tendance à favoriser une granularité fine pour apprendre des fonctions simples, ce qui favoriserait à une convergence rapide. Pour un nombre d'entrée et/ou sortie suffisamment important, l'implémentation de la LUT en réseau de neurones est plus efficace que la table en SRAM. En outre, l'apprentissage corrige aussi bien les défauts de collage à 1 que les collages à 0. Il serait maintenant intéressant d'évaluer la robustesse à des fautes dynamiques notamment en les introduisant lors de l'apprentissage.

De cette comparaison, nous pouvons tirer, quelques enseignements. D'abord, la surface allouée aux synapses est négligeable au regard de la surface qu'occuperaient les cellules neurones incluant un mécanisme d'apprentissage réalisées en CMOS ultime. Ainsi, la surface occupée par les neurones cachés ferait perdre une grande partie des avantages en termes de densité comparée à une

solution à base de LUT-SRAM. Ce point est aggravé par la différence de densité projetée entre mémoire et logique au point que l'avantage d'une solution à base de réseau de neurone n'est absolument pas évident. Pour y échapper, nous pouvons envisager deux pistes. D'abord, il est souhaitable d'envisager l'implémentation des neurones et les fonctions d'apprentissage en utilisant entièrement des nanocomposants permettant d'atteindre la plus grande densité possible. D'autre part, nous devons envisager d'intégrer des réseaux de tailles plus importantes, mais pour l'une ou l'autre de ces deux voies, il reste des problèmes ouverts liés par exemple aux difficultés de convergence de l'apprentissage sur des réseaux plus vastes et à la résolution limitée des poids synaptiques.

CONCLUSION FINALE

Nous sommes arrivés à la conclusion de ce travail, et il m'a semblé intéressant d'avoir traité un sujet de thèse qui a tenté une prospective d'un enjeu des nanotechnologies, au sujet d'une architecture alternative. Il a été préparé en sondant les divers champs interdisciplinaires, de l'étude des alternatives, puis l'investigation des nanocomposants, enfin la conception et l'analyse de la tolérance de défauts. Après avoir exploré les différents horizons, le sujet est récapitulé en quelques points essentiels.

Voici ma vision. De plus en plus de technologies somme toutes différentes continueront à apparaître. Le choix de déploiement d'une technologie particulière répondant à une demande sectorielle sera réel. Il dépendra, du coût additionnel, de la gamme de température, de la consommation, et de la puissance de calcul exigée. Les critères de choix de fabrication de puces seront constitués d'emblée par de nombreux facteurs, et leur qualité de mise en oeuvre devra être suffisante. A court terme, les conditions ne seront pas réunies pour voir apparaître une technologie capable de remplacer intégralement la technologie de semi-conducteur employée actuellement. Dès lors, on doit s'attendre à l'intégration hétérogène, et en même temps l'organisation de l'ensemble doit être cohérente. Pour répondre aux nouvelles exigences dans un tournant rapide de la technologie, on doit déterminer les rôles des nanocomposants dans un projet de conception, tel l'architecture des réseaux de neurones. Il s'agit au préalable de vérifier la compatibilité entre eux, en analysant les critères de choix. Au premier abord, l'impact des nanocomposants sur les systèmes sur puce semble marginal, et ce d'autant plus si on se limite à l'implémentation de fonction logique Booléenne, mais en se saisissant des propriétés intrinsèques aux nanocomposants on peut aussi s'attendre à un impact d'un autre genre sur de nouvelles architectures. On peut également l'anticiper en prenant dès maintenant des initiatives d'adaptation dans un nouveau cadre de conception de calcul, sans rester passif devant les conséquences de changement.

S’il y a forte à parier pour une imposante dispersion de caractéristique, pour un taux de défauts élevé, l’impossibilité de voir le CMOS comme une solution potentielle, alors on pourra chercher l’inspiration du côté des architectures reconfigurables, redondantes, et tolérantes aux défauts. Une tendance peut se dégager en rapport avec cette dernière. Dans les circuits reprogrammables, il existe des blocs qui servent aujourd'hui d'éléments logiques universels (LUT). J’ai choisi de développer de petits réseaux neuronaux, réalisés à partir des nanocomposants pour simuler des fonctions logiques. Grâce au caractère programmable des mémoires présentant plusieurs niveaux de conductances et faites de nanocomposants, les synapses peuvent être implémentées plus efficacement. Toutefois, le circuit nécessite l’implémentation du module d’apprentissage pour la plasticité synaptique. Mes travaux ont mis en évidence l'intérêt d'une telle approche au niveau du circuit tout en s’appuyant sur une conception détaillée des éléments clés de l'architecture. J’ai d’abord comparé les alternatives proposées. Ensuite, j’ai modélisé un nanocomposant pour l’implémentation de la synapse électronique. J’ai réuni les modèles compacts des nanocomposants utilisés dans mon architecture, puis j’ai élaboré un algorithme d'apprentissage et l’ai modélisé au niveau du circuit. J’ai fait la simulation électrique des portes logiques avec l’apprentissage. Les études démontrent que ce type d’architecture pourra laisser profiler une solution sérieuse. Enfin une procédure générale consiste à évaluer la tolérance aux défauts. J'ai effectué les simulations fonctionnelles de ces réseaux de neurones. J’ai exposé les résultats de simulation. J’ai calculé le taux de succès de l’apprentissage en fonction du niveau de redondance. J’ai comparé les surfaces consacrées au niveau de la puce pour les deux méthodes d’implémentation de la LUT.

En conclusion, selon moi dans un futur proche les systèmes pour traitement de l'information auront inclus les nanocomposants. Une nouvelle architecture de type neuronal serait une alternative concevable. Les traits connus des circuits intégrés pourront changer substantiellement, peu importe si les causes seront endogènes à la technologie ou exogènes. Une solution basée sur les réseaux de neurones sera non seulement une vraie alternative en architecture de calcul, mais aussi extensible aux traitements d’image et de son, à la reconnaissance, à l’intelligence augmentée et d’autres applications, donc elle s’incorporera facilement dans la tendance "More than

Moore". Il restera à vérifier par les comparaisons pertinentes des architectures alternatives en prenant compte des exigences d'intégration.

Même si la technologie CMOS semble conserver sa place de leader dans les technologies du semi-conducteur, les nouvelles technologies s'appuyant sur les nouvelles architectures devront faire leurs apparitions prochainement, peut être dans le processus de l'hybridisation avec CMOS. Dans cette thèse, la direction est principalement donnée sur l'implémentation des réseaux de neurones pour circuit reconfigurable.

Références

[Ackl85] D.H. Ackley and G.E. Hinton, "A Learning Algorithm for Boltzmann Machines," Cognitive Science, 1985, vol. 9, pp. 147-169

[Ago05] H. Ago, S. Imamura, T. Okazaki, T. Saito, M. Yumura, and M. Tsuji, “CVD Growth of Single-Walled Carbon Nanotubes with Narrow Diameter Distribution over Fe/MgO Catalyst and Their Fluorescence Spectroscopy,” J. Phys. Chem. B, 2005, vol. 109, pp. 10035-10041

[Ande98] James A. Anderson and Edward Rosenfeld, "Talking Nets: An Oral History of Neural Networks," Cambridge, MA, and London: MIT Press, 1998

[Aust04] M.D. Austin, H. Ge, W. Wu, M. Li, Z. Yu, D. Wasserman, S.A. Lyon, and Stephen Y. Chou, "Fabrication of 5 nm linewidth and 14 nm pitch features by nanoimprint lithography", Appl. Phys. Lett., juin 2004, vol. 84, no. 26, pp. 5299

[Bach01] A. Bachtold, P. Hadley, T. Nakanishi, and C. Dekker, “Logic circuits with carbon nanotube transistors,” Science 294, Nov 2001, pp. 1317-1320

[Bali07] A. Balijepalli, S. Sinha, and Y. Cao, "Compact modeling of carbon nanotube transistor for early stage process-design exploration," Proceedings of the 2007 International Symposium On Low Power Electronics and Design, August 2007, Portland, Oregon, USA, pp. 502-507

[Beiu03] V. Beiu, “VLSI implementation of threshold gates - A comprehensive survey,” IEEE Transactions on Neural Networks, May 2003, vol 14, no. 5, pp. 1217-1243

[Beiu03-2] V. Beiu, "A Survey of Perceptron Circuit Complexity Results," Proceedings of the International Joint Conference on Neural Networks, 20-24 July 2003, vol. 2, pp. 989- 994

[Beiu04] V. Beiu, "A novel highly reliable low-power nano architecture when von Neumann augments Kolmogorov," Proceedings 15th IEEE International Conference on Application-Specific Systems, Architectures and Processors, Galveston, TX, USA, 27-29 Sept. 2004, pp. 167-177

[Beiu05] V. Beiu, S. Aunet, J. Nyathi, R. R. III Rydberg, and A. Djupdal, “On the advantage of serial architectures for low power reliable computations,” IEEE International Conference on Application-specific Systems,

Architectures and Processors, Samos, Greece, 23-25 July 2005, pp. 276-281

[Beiu07] V. Beiu, "Grand Challenges of Nanoelectronics and Possible Architectural Solutions," IEEE International Symposium on Multiple Valued Logic ISMVL'07, Oslo, Norway, May 14-16, 2007

[Belf89] L.A. Belfore II, and B.W. Johnson, "The fault tolerance characteristics of Neural Networks" International Journal of Neural Networks Research and Applications, Jan 1989, vol. 1, no. 1, pp. 24-41

[Berg06] C. Berger, Z. Song, X. Li, X. Wu, N. Brown, C. Naud, D. Mayou, T. Li, J. Hass, A.N. Marchenkov, E.H. Conrad, P.N. First, and W.A. de Heer, "Electronic Confinement and Coherence in Patterned Epitaxial Graphene Originally," Science Express, 2006, vol. 312. no. 5777, pp. 1191-1196

[Betz97] V. Betz, and J. Rose, "VPR: A New Packing, Placement and Routing Tool for FPGA Research," Seventh International Workshop on Field-Programmable Logic and Applications, London, UK, 1997, pp. 213-222

[Bilo66] A. Bilotti, "Operation of a MOS transistor as a variable resistor," Proceedings of the IEEE, Aug. 1966, vol. 54, no. 8, pp.1093-1094

[Cavi05] R.K. Cavin and V.V. Zhirnov, "Future Devices for Information Processing," Proceedings of ESSCIRC, 2005, Grenoble, France, pp. 7-12

[Chau05] R. Chau, S. Datta, M. Doczy, B. Doyle, B. Jin, J. Kavalieros, A. Majumdar, M. Metz, and M. Radosavljevic, "Benchmarking Nanotechnology for High-Performance and Low-Power Logic Transistor Applications," IEEE Transactions on Nanotechnology, vol. 4, no. 2, Mar. 2005, pp. 153-158

[Chey97] Ph. Cheynet, J.-C. Rubio, R. Velazco, and J.-D. Muller, "Evaluating neural network robustness with an architecture built around L-Neuro 2.3," 6th Int. Conference on Microelectronics for Neural Networks, Evolutionary and Fuzzy Systems (MICRONEURO'97), Dresden, Allemagne, 24-26 Sep. 1997, Poster Session I, pp. 244-250

[Chen03] Y. Chen, G.Y. Jung, D.A.A. Ohlberg, X. Li, D.R. Stewart, J.O. Jeppesen, K.A. Nielsen, J.F. Stoddart and R.S. Williams, "Nanoscale molecular-switch crossbar circuits," Nanotechnology, 2003, vol. 14, pp. 462-468

[Chen06] T.C. Chen, "Where CMOS is Going: Trendy Hype vs. Real Technology," IEEE International Solid-State Circuits Conference 2006, Digest of Technical Papers. Feb. 6-9, 2006, pp. 1-18

[Chua71] L.O. Chua, "Memristor - the missing circuit element," IEEE Trans. Circuit Theory vol. 18, 1971, pp. 507-519

[Coll01] P.G. Collins, M. Hersam, M. Arnold, R. Martel, and Ph. Avouris, “Current Saturation and Electrical Breakdown in Multiwalled Carbon Nanotubes,” Physical Review Letters, 2 April 2001, vol. 86, number 14, pp. 3128-3131

[Coll99] C.P. Collier, E.W. Wong, M. Belohradsky, F.M. Raymo, J.F. Stoddart, P.J. Kuekes, R.S. Williams, and J.R. Heath "Electronically ConÞgurable Molecular Based Logic Gates," Science, vol. 285, 16 Jul. 1999, pp. 391-393

[Cowb00] R.P. Cowburn and M. E. Welland, "Room Temperature Magnetic Quantum Cellular Automata," in Science, vol. 25 February 2000, vol. 287, no. 5457, pp. 1466-1468

[Cui03] Y. Cui, Z. Zhong, D. Wang, W.U. Wang, and C.M. Lieber, "High Performance Silicon Nanowire Field Effect Transistors," Nano Letters, 2003, vol. 3, no. 2, pp. 149-152

[Dai02] H.-J. Dai, "Carbon Nanotubes: Synthesis, Integration, and Properties," American Chemical Society, 2002, vol. 35, no. 12, pp. 1035-1044

[Dama89] T.R. Damarla, and P.K. Bhagat, “Fault Tolerance of Neural Networks”. In Proceedings of Southeastcon, IEEE Computer Society Press, 1989, pp. 328–331.

[Datt90] S. Datta and B. Das, “Electronic analog of the electrooptic modulator”, AppriedPhysics Leners, 1990, vol. 56, no. 7, pp. 665-667

[Deho05] A. Dehon, "Nanowire Based Programmable Architectures," in ACM Journal on Emerging Technologies in Computing Systems, vol. 1, no. 2, July 2005, pp. 109-162

[Denn74] R.H. Dennard, F.H Gaensslen, V.I. Rideout, E. Bassous, and A.R. Le Blanc, "Design of Ion Implanted MOSFET's with Very Small Physical Dimensions," IEEE Journal of Solid-State Circuits, Oct. 1974, vol. 9, no. 5, pp. 256- 268, pp. 668-678

[Denn03] C.L. Dennis, C. Sirisathitkul, G.J. Ensell, J.F. Gregg and S.M. Thompson, "High current gain silicon-based spin transistor," J. Phys. D: Appl. Phys. 2003, vol. 36, pp. 81-87

[Elia93] V. Eliashberg, "A Relationship between neural networks and programmable logic arrays," International Conf. on Neural Networks, San Francisco, USA, March 28 -Apr.1 1993, vol. 3, pp. 1333-1337

[EniG03] Recensement de matériaux sur http://www.periodni.com/, la dernière modification du site est en Avril 2003, Eni Generalic, Faculty of Chemistry and Technology, KEMIJSKO - TEHNOLOŠKI FAKULTET, Teslina 10/V, 21000 Split , Croatia, HRVATSKA

[Fauc04] N. Faucheux, R. Schweiss, K. Lützow, C. Werner, and T. Groth, "Self-assembled monolayers with different terminating groups as model substrates for cell adhesion studies," Biomaterials, 2004 vol. 25, pp. 2721-2730

[Fors04] M. Forshaw, R. Stadler, D. Crawley, and K. Nikolic, "A short review of nanoelectronic architectures," Nanotechnology, 2004, vol. 15 pp. 220-223

[Fuhr02] M.S. Fuhrer, B.M. Kim, T. Dürkop, and T. Brintlinger, "High-Mobility Nanotube Transistor Memory," Nano Letters, 2002, vol. 7, pp. 755-759

[Fusi00] S. Fusi, P. Del Giudice, and D.J. Amit, "Neurophysiology of a VLSI spiking neural network: LANN21," IEEE-INNS-ENNS international joint conference on neural networks, Como , Italie, 24-27 Jul. 2000, Vol. 3, pp. 121-126

[Gao07] C. Gao and D. Hammerstrom, "Cortical Models Onto CMOL and CMOS - Architectures and Performance/Price," IEEE Trans. on Circuits and Systems I: regular papers, vol. 54, no. 11, Nov. 2007 pp. 2502-2515

[Gilb05] N.E. Gilbert, C. Gopalan, and M.N. Kozicki, "A macro model of programmable metallization cell devices," Solid-State Electronics, vol. 49, 2005, pp. 1813-1819

[Gold01] S.C. Goldstein and M. Budiu, "NanoFabrics: Spatial Computing Using Molecular Electronics," Proceedings. 28th Annual International Symposium on Computer Architecture, 2001, pp.178-189

[Gold02] S.C. Goldstein and D. Rosewater, "Digital Logic Using Molecular Electronics," Digest of Technical Papers ISSCC. IEEE International Solid-State Circuits Conference, 2002, vol.1, pp. 204

[Gree07] J.E. Green, J.W. Choi, A. Boukai, Y. Bunimovich, E. Johnston-Halperin, E. DeIonno, Y. Luo, B.A. Sheriff, K. Xu, Y.S. Shin, H.-R. Tseng2, J.F. Stoddart, and J.R. Heath1, "A 160-kilobit molecular electronic memory patterned at 10^{11} bits per square centimetre", Nature, vol. 445, 25 January 2007, pp. 414-417

[Hasl95] P. Hasler, C. Diorio, B. A. Minch, and C.A. Mead, "Single transistor learning synapses," in Gerald Tesauro, David S. Touretzky, and Todd K. Leen (eds.), Advances in Neural Information Processing Systems 7, MIT Press, Cambridge, MA, 1995, pp. 817-824

[Hasl06] P. Hasler, E. Farquhar and C. Gordon, "Building Large Networks of Biological Neurons," 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS '06, Aug. 30 2006-Sept. 3 2006, pp. 6548-6551

[He07] C. He, and M.F. Jacome, "Defect-Aware High-Level Synthesis Targeted at Reconfigurable Nanofabrics," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, May 2007, vol. 26, no. 5, pp. 817-833

[Hebb49] D.O. Hebb, "The organization of behavior," New York, Wiley, 1949

[Hopf82] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," Proc. Natl. Acad. Sci. USA, April 1982, Biophysics, vol. 79, pp. 2554-2558

[Huan01] Y. Huang, X. Duan, Q. Wei, and C.M. Lieber, “Directed Assembly of One Dimensional Nanostructures into Functional Networks”, Science, 2001, vol. 291, pp. 630

[Huan01-2] Y. Huang, X. Duan, Y. Cui, L.J. Lauhon, K.-H. Kim, and C.M. Lieber "Logic Gates and Computation from Assembled Nanowire Building Blocks," Science, vol. 294, 9 Nov. 2001, pp. 1313-1316

[Hutc06] D. Hutcheson, "450mm: What Will It Cost to Get There ?", Future Fab Intl. no. 20, July 2006

[Hynn07] K.M Hynna and K. Boahen "Silicon neurons that burst when primed," IEEE International Symposium on Circuits and Systems, ISCAS 2007, 27-30 May 2007, pp. 3363-3366

[Imre06] A. Imre, G. Csaba, L. Ji, A. Orlov, G. H. Bernstein, and W. Porod, "Majority Logic Gate for Magnetic Quantum-Dot Cellular Automata," Science, vol. 311, no.13, Jan. 2006, pp. 205-208

[Ishi04] M. Ishida, H. Hongo, F. Nihey, and Y. Ochiai, “Diameter-Controlled Carbon Nanotubes Grown from Lithographically Defined Nanoparticles”, Japan. J. Appl. Phys., vol. 43, 2004, pp. 1356-1358

[ITRS] International Technology Roadmap for Semiconductors, http://www.itrs.net

[Jurc04] O.D. Jurchescu, J. Baas, and T.M. Palstra, "Effect of impurities on the mobility of single crystal pentacene," Appl. Phys. Lett., 2004, vol. 84, no. 16, pp. 3061

[Karg07] S. Karg, G.I. Meijer, D. Widmer, R. Stutz, J.G. Bednorz, and C. Rettner, "Nanoscale Resistive Memory Device Using SrTiO3 Films," The 22nd IEEE Non-Volatile Semiconductor Memory Workshop, Monterey, CA, USA, 26-30 Aug. 2007, pp. 68-70

[Kerl94] Kerlirzin Philippe, "Etude de la robustesse des réseaux multicouches," Thèse doctorale, Univ. Paris Sud XI, Orsay, 1994

[Kond06] D. Kondo, S. Sato, and Y. Awano, “Low-temperature synthesis of single-walled carbon nanotubes with a narrow diameter distribution using size-classified catalyst nanoparticles,” Chem. Phys. Lett., 2006, vol. 422, pp. 481-487

[Kong98] J. Kong, H.T. Soh, A. Cassell, C.F. Quate, and H. Dai, “Synthesis of Individual Single-Walled Carbon Nanotubes on Patterned Silicon Wafers,” Nature, vol. 395, 1998, pp. 878

[Koca06] C. Kocabas, M. Shim, and J.A. Rogers,” Spatially Selective Guided Growth of High-Coverage Arrays and Random Networks of Single-Walled Carbon Nanotubes and Their Integration into Electronic Devices,” J. Am. Chem. Soc., vol. 128, 2006, pp. 4540

[Kozi05] M.N. Kozicki, M. Balakrishnan, C. Gopalan, C. Ratnakumar, and M. Mitkova, “Programmable Metallization Cell Memory Based on Ag-Ge-S and Cu-Ge-S Solid Electrolytes,” in Non-Volatile Memory Technology Symposium (NVMTS 2005), D5, 7-10 Nov. 2005, pp. 83-89

[Kund05] M. Kund, G. Beitel, C-U Pinnow, T. Röhr, J. Schumann, R. Symanczyk, K-D. Ufert, and G. Müller, "Conductive bridging RAM (CBRAM): An emerging non-volatile memory technology scalable to sub 20nm," IEEE International Electron Devices Meeting, 5-7 Dec. 2005, pp. 754-757

[Kuek05] P.J. Kuekes, D.R. Stewart, and R.S. Williams, "The crossbar latch: Logic value storage, restoration, and inversion in crossbar circuits," Journal of Appl. Phys., 2005, vol. 97, pp. 034301

[Lai08] Q. Lai, Z. Li, L. Zhang, X. Li, W.F. Stickle, Z. Zhu, Z. Gu, T.I. Kamins, R. Stanley Williams, and Y. Chen, "An Organic/Si Nanowire Hybrid Field Configurable Transistor," Nano Lett. 2008, vol. 8, no. 3, pp. 876-880

[Lee00] Jeong-O Lee, C Park, Ju-Jin Kim, Jinhee Kim, Jong Wan Park and Kyung-Hwa Yoo, "Formation of low-resistance ohmic contacts between carbon nanotube and metal electrodes by a rapid thermal annealing method," in J. Phys. D: Appl. Phys. 33 (2000), pp. 1953-1956

[Lee07] J.H. Lee and K.K. Likharev, "Defect-tolerant nanoelectronic pattern classifiers," Intl. J. of Circcuit Theory and Application, 2007, vol. 35, pp. 239-264

[Lent93] C.S. Lent, P.D. Tougaw, W. Porod, and G.H. Bernstein, "Quantum cellular automata," Nanotechnology, 1993, vol. 4, pp. 49-57

[Levi06] M. Levitt, "The Role of Design in Enhancing Nanometer Process Yield," International Engineering Consortium, IEC Newsletter, 18 Jan. 2006

[Li96] W.Z. Li, Z.W. Pan, S.S. Xie, L.X. Qian, B.H. Chang, B.S. Zou, W.Y. Zhou, R.A. Zhao and G. Wang, “Large-Scale Synthesis of Aligned Carbon Nanotubes," Science, vol. 274, pp. 1701-1703, Dec. 6, 1996

[Li08] X. Li, X. Wang, L. Zhang, S. Lee, and H. Dai, "Chemically Derived, Ultrasmooth Graphene Nanoribbon Semiconductors," Science, vol. 319, 29 FEBRUARY 2008

[Likh03] K.K. Likharev, "Neuromorphic CMOL circuits," Intl. Proc. IEEE Nanotechnology, 2003, IEEE-NANO 2003, vol. 1, pp. 339- 342, 12-14 Aug. 2003

[Lloy00] S. Lloyd, "Ultimate physical limits to computation," Nature, August 2000, vol. 406, pp. 1047-1054

[Mane06] C. Maneux, J. Goguet, S. Frégonèse, T. Zimmer, H. Cazin d'Honincthun, and S. Galdin-Retailleau, "Analysis of CNTFET physical compact model," Proceedings IEEE International Conference Design and Test of Integrated Systems in Nanoscale Technology 2006, vol. 1, pp. 40-45

[Math99] R.H. Mathews, J.P. Sage, T. Sollner, S.D. Calawa, C. Chen, L.J. Mahoney, P.A. Maki, and K.M. Molvar "A new rtd-fet logic family," Proc. IEEE, 1999, vol. 87, no. 4, pp. 596

[Maye01] T.S. Mayer, S. Goldstein, T. Mallouk, C. Keating, and T. Jackson, "Molecular electronic building blocks: Functional metal nanowires," American Chemical Society National Meeting, Aug. 2001, Chicago, IL

[MC&P43] W.S. McCulloch and W. Pitts, "A logic calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biophysics, 1943, vol. 5, pp. 115-133

[Mead89] C. Mead, "Analog VLSI and neural systems," Addison Wesley Publishing Company, 1989

[Melo04] N.A. Melosh, A. Boukai, F. Diana, B. Gerardot, A. Badolato, P.M. Petroff, and J.R. Heath "Ultrahigh-Density Nanowire Lattices and Circuits," Science, vol. 300, 4 Apr. 2003, pp. 112-115

[Meza89] M Mezard and Jean-P Nadal, "Learning in feedforward layered networks: the tiling algorithm," Journal of Physics A: Mathematical and General, 1989, vol. 22, pp. 2191-2203

[Mizu07] Y. Mizuno, S. Ohya, P. N. Hai and M. Tanaka, “Spin-dependent transport properties in GaMnAs-based spin hot-carrier transistors,” Applied Physics Letters, 2007, vol. 90, pp. 162505

[Moor65] Gordon E. Moore, "Cramming more components onto integrated circuits," Electronics, April 19 1965, vol. 38, no. 8, pp. 114-117

[Mori07] C.A. Moritz, T. Wang, P. Narayanan, M. Leuchtenburg, Y. Guo, C. Dezan, and M. Bennaser "Fault-Tolerant Nanoscale Processors on Semiconductor Nanowire Grids," IEEE Transactions on Circuits and Systems I, Nov. 2007, vol. 54, no. 11, pp. 2422-2437

[Nave00] Y. Naveh and K.K. Likharev, "Modeling of 10-nm-scale ballistic MOSFET" IEEE Electron Device Letters, May 2000, Volume 21, Issue 5, pp. 242-244

[Nihe03] F. Nihey, H. Hongo, Y. Ochiai, M. Yudasaka and S. Iijima, "Carbon-Nanotube Field-Effect Transistors with Very High Intrinsic Transconductance," Japan. J. Appl. Phys., vol. 42, 2003, pp. L1288-L1291

[Orlo00] A.O. Orlov, G. Toth, I. Amlani, R. Kummamuru, R. Ramasubramaniam, C.S. Lent, G.H. Bernstein, and G.L. Snider, "Experimental studies of quantum dot cellular automata devices," the 58th Device Research Conference, 2000, Digest, pp. 157-158

[Oya05] T. Oya, A. Schmid, T. Asai, Y. Leblebici and Y. Amemiya, "On the fault tolerance of a clustered single-electron neural network for differential enhancement," IEICE Electronics Express, 2005, vol. 2, no. 3, pp. 76-80

[Pace07] G. Pace, "Self-Assembly of functional molecules at surfaces," Thèse doctorale à l'Institut de Science et d'Ingénierie Supramoléculaire, Université Louis Pasteur, UMR 7006 Nanochemistry Laboratory (ISIS-ULP), 2007

[Pari03] M.C.B. Parish, "Modelling of Physical Constraints on Bistable Magnetic Quantum Cellular Automata," PhD thesis, in Dept. of Physics and Astronomy, University College London, 2003

[Patw06] J.P. Patwardhan, C. Dwyer, A.R. Lebeck, and D.J. Sorin, "A Nano-Scale Active Network Architecture," ACM Journal on Emerging Technologies in Computing Systems, January 2006, vol. 2, no. 1, pp. 1-30

[Phat92] D.S. Phatak and I. Koren, "Fault Tolerance of Feedforward Neural Nets for Classification Tasks," Proceedings of International Joint Conference on Neural Nets (IJCNN), Baltimore, MD, Jun. 1992, vol. II, pp. 386-391

[Phat95] D.S. Phatak and I. Koren, "Complete and partial fault tolerance of feedforward neural nets," IEEE Transactions on Neural Networks, Mar 1995, vol. 6, no. 2, pp. 446-456

[Pram04] S. Pramanik, S. Bandyopadhyay, and M. Cahay, "Why is the spin field effect transistor elusive?" 4th IEEE Conference on Nanotechnology, 16-19 Aug. 2004, pp. 101-103

[Preg06] F. Prégaldiny, C. Lallement, and J.-B. Kammerer, "Design-oriented compact models for CNTFETs," Proc. IEEE International Conference on Design & Test of Integrated Systems in Nanoscale Technology (DTIS' 06), pp. 34-39, Tunis (Tunisie), 5-7 Septembre 2006

[Rao07] F. Rao, Z. Song, M. Zhong, L. Wu, G. Feng, B. Liu, S. Feng, and B. Chen, "Multilevel Data Storage Characteristics of Phase Change Memory Cell with Double layer Chalcogenide Films ($Ge_2Sb_2Te_5$ and Sb_2Te_3)," Japanese Journal of Applied Physics, 2007, vol. 46, no. 2, pp. L25-L27

[Rayc04] A. Raychowdhury, S. Mukhopadhyay, and K. Roy, "A circuit-compatible model of ballistic carbon nanotube field-effect transistors," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Oct. 2004, vol. 23, no. 10, pp. 1411-1420

[Ren03] Z. Ren, R. Venugopal, S. Goasguen, S. Datta, and M.S. Lundstrom, "nanoMOS 2.5: A Two -Dimensional Simulator for Quantum Transport in Double-Gate MOSFETs," IEEE Trans. Electron. Dev. 2003, Special issue on Nanoelectronics, vol. 50, pp. 1914-1925.

[Rose58] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," Psychological Review, 1958, vol. 65, pp, 386-408

[Ross03] D. Rossi AND C. Metra, "Error Correcting Strategy for High Speed and High Density Reliable Flash Memories," Journal of Electronic Testing: Theory and Applications, 2003, vol. 19, pp. 511–521

[Rubi86] I. Rubinstein, E. Sabatani, R. Maoz and J. Sagiv, "Organized Monolayers on Gold Electrodes, Electrochemical Sensors for Biomedical Applications," C.K.N. Li (Ed.), Electrochemical Society 1986, pp. 175

[Sait92] R. Saito, M. Fujita, G. Dresselhaus, and M.S. Dresselhaus, "Electronic structure of chiral graphene tubules," Appl. Phys. Lett., 1992, vol. 60, pp. 2204-2206

[Sank91] A. Sankar and R.J. Mammone, "Optimal pruning of neural tree networks for improved generalization," International Joint Conference on Neural Networks, Seattle, USA, 8-14 Jul. 1991, vol. 2, pp. 219 -224

[Saun08] C. Saunier, "Rapport sur l'évolution du secteur de la micro/nanoélectronique", Rapport Parlementaire du Sénat, no. 407, Session Ordinaire 2007-2008, Office Parlementaire D'évaluation Des Choix Scientifique Et Technologiques, Jun. 2008

[Schm03] A. Schmid and Y. Leblebici, "A modular approach for reliable nanoelectronic and very-deep submicron circuit design based on analog neural network principles," IEEE Nanotechnology, 2003. vol. 2, pp. 647-650, Aug. 2003

[Seem99] N.C. Seeman, “DNA engineering and its application to nanotechnology Trends," in Biotech 17, pp. 437-443, 1999

[Snid07] G.S. Snider and R.S. Williams, "Nano/CMOS architectures using a field-programmable nanowire interconnect," Nanotechnology, 2007, vol. 18 no. 3, 2007, Notes: 035204.1-035204.11

[Snid07-2] G.S. Snider, "Self-organized computation with unreliable, memristive nanodevices,” Nanotechnology, 2007, vol. 18, no. 36, pp. 365202

[Stew04] D. R. Stewart, D. A. A. Ohlberg, P. A. Beck, Y. Chen, and R.S. Williams, J.O. Jeppesen, K. A. Nielsen, and J. Fraser Stoddart, “Molecule-Independent Electrical Switching in Pt/Organic Monolayer/Ti Devices,” Nano Letters 2004, vol. 4, no. 1, pp. 133-136

[Stew07] D.R. Stewart, G. Gibson, G.Y. Jung, W. Wu, J. Straznicky, W. Tong, Z. Li and R.S. Williams, "Direct-write programming of nanoscale demultiplexer arrays," Nanotechnology, 2007, vol. 18, pp. 415201

[Stra06] I. Stratan, D. Tsiulyanu, and I. Eisele, "A programmable metallization cell based on Ag-As2S3," Journal of optoelectronics and advanced materials, vol. 8, no. 6, December 2006, pp. 2117-2119

[Stru05] D.B. Strukov and K.K. Likharev, "Prospects for terabit-scale nanoelectronic memories," Nanotechnology, 2005, vol. 16, pp. 137–148

[Stru05-2] D.B. Strukov and K.K. Likharev, "CMOL FPGA: a reconfigurable architecture for hybrid digital circuits with two-terminal nanodevices," Nanotechnology, 2005, vol. 16, pp. 888-900

[Suga06] S. Sugahara and M. Tanaka, "Spin MOSFETs as a basis for spintronics," ACM Transactions on Storage, vol. 2, no. 2, May 2006, pp. 197-219

[Tans97] S.J. Tans, M.H. Devoret, H. Dai, A. Thess, R.E. Smalley, L.J. Geerligs and C. Dekker, "Individual single-wall carbon nanotubes as quantum wires," Nature, 1997, vol. 386, pp. 474-477

[Teix06] D. Teixera, Franco, J.-F. Naviner, and L. Naviner, "Yield and reliability issues in nanoelectronic technologies," Annales des télécommunications, Nov.-Dec. 2006, vol. 61, no. 11-12

[Tour02] J.M. Tour, W.L. Van Zandt, C.P. Husband, S.M. Husband, L.S. Wilson, P.D. Franzon, and D.P. Nackashi, "Nanocell logic gates for molecular

computing," IEEE Transactions on Nanotechnology, vol. 1, no. 2, pp. 100-109, June 2002

[Tour03] James M. Tour, Long Cheng, David P. Nackashi, Yuxing Yao, Austen K. Flatt, Sarah K. St. Angelo, Thomas E. Mallouk, and Paul D. Franzon, "NanoCell Electronic Memories," J. Am. Chem. Soc. 2003, vol. 125, pp. 13279-13283

[Ural02] A. Ural, Y. Li, and H. Dai, “Electric-field-aligned growth of single-walled carbon nanotubes on surfaces,” Appl. Phys. Lett., 2002, vol. 81, pp. 3464

[Vela00] R. Velazco, Ch. Godin, Ph. Cheynet, S. Torres-Alegre, D. Andina and M. B. Gordon, "Study of two ANN digital implementations of a radar detector candidate to an on-board satellite experiment," Engineering Applications of Bio-Inspired Artificial Neural Networks, 1999, vol. 1607, pp. 615-624

[Wang04] T. Wang, Z. Qi and C.A. Moritz, "Exploring Nanoscale ICs and Architectures", Proc. of the First Conference on Computing Frontiers, Italy, April 2004, pp. 503-511

[Wass89] S.R. Wasserman, Y.T. Tao, and G.M. Whitesides, “Structure and Reactivity of Alkylsiloxane Monolayers Formed by Reaction of Alkyltrichlorosilanes on Silicon Substrates,” Langmuir, 1989, vol. 5, no. 4, pp. 1074-1087

[Widr60] B. Widrow and M. E. Hoff. “Adaptive switching circuits,” IRE WESCON Convention Record, New York, 1960, pp. 96-104

[Wije08] J.H.B. Wijekoon and P. Dudek, "Compact silicon neuron circuit with spiking and bursting behaviour" Neural Networks, Special Issue, vol. 21, 2008, pp. 524-34

[Will01] S. Williams and P. Kuekes, “Demultiplexer for a Molecular Wire Crossbar Network,” United States Patent Number: 6,256,767, July 3 2001

[Winf98] E. Winfree, F. Liu, L.A. Wenzler, and N.C. Seeman, “Design and self-assembly of two dimensional DNA crystals,” Nature, 1998, vol. 394, no. 6693, pp. 539-544

[Wong05] H.-S. P. Wong, "Nanoelectronics: Nanotubes, Nanowires, Molecules, and Novel Concepts," Proceedings of ESSCIRC, 2005, Grenoble, France, pp. 55-61

[Wong06] H.S.P Wong, J. Deng, A. Hazeghi, T. Krishnamohan, and G.C. Wan, "Carbon nanotube transistor circuits: models and tools for design and performance optimization," International Conference on Computer

Aided Design IEEE-ICCAD 2006, San Jose, CA, USA, Nov. 2006, pp. 651-654

[Xian06] J. Xiang, W. Lu, Y. Hu, Y. Wu, H. Yan and C.M. Lieber, "Ge/Si nanowire heterostructures as highperformance field-effect transistors," Nature, 25 May 2006, vol. 441, pp. 489-493

[Yan03] H. Yan, T.H. Labean, L. Feng, and J.H. Reif, “Directed nucleation assembly of DNA tile complexes for barcode-patterned lattices,” Proceedings of the National Academy of Sciences, July 2003, vol. 100, no.14, pp. 8103-8108

[Zavo07] M.Y. Zavodchikova, A. Johansson, M. Rinkio, J.J. Toppari, A.G. Nasibulin, E.I. Kauppinen, and P. Torma, " Fabrication of carbon nanotube-based field-effect transistors for studies of their memory effects," Physica Status Solidi (b), 2007, vol. 244, no. 11, pp. 4188-4192

[Zhan04] R. Zhang, P. Gupta, L. Zhong, and N.K. Jha, "Synthesis and optimization of threshold logic networks with application to nanotechnologies" Proceedings of Design, Automation and Test in Europe 2004, vol. 2, pp. 904-909, 16-20 Feb. 2004

[Zhir05] V.V. Zhirnov, J.A. Hutchby, G.I. Bourianoff, and J.E. Brewer, "Emerging research logic devices," IEEE Circuits & Devices Magazine, vol. 21, no. 3, pp. 37-46, May-Jun. 2005

[Zhu95] Y. Zhu, "Contribution à la réalisation électronique de réseaux de neurones formels: Intégrations analogique de l'apprentissage des machines de Boltzmann, " Thèse doctorale, Univ. Paris Sud XI, Orsay, 1995

[Zhu03] J. Zhu and P. Sutton, "FPGA Implementation of Neural Networks - A Survey of a Decade of Progress" 13th International Conference on Field-Programmable Logic and Applications (FPL 2003), Lisbon, Portugal, September 2003, pp.1062-1066

Annexes

I. Code source de CBRAM, résistance multi niveau programmable par impulsion (Chapitre 3)

```
// VerilogA : la conductance multi niveau programmable par pas temporel
// Michel HE - 2007, IEF

`include "constants.vams"
`include "disciplines.vams"

module cbram4 (control,out);
  input control;
  output out;

  electrical control, out;

  parameter real tau = 0.0001;
  parameter real Rinit = 50e3;
  parameter real K = 50e3;
  parameter real Gmin = 1e-6;
  parameter real Gmax = 1e-3;
  parameter real show_variation = 1e-5;
//caracteristiques temporels
  parameter real  tconv = 1n  from (0:inf);
  parameter real  trise = 1p  from (0:inf);
  parameter real  tfall = 1p  from (0:inf);

  parameter real ohm_step_p = 3;
  parameter real ohm_step_n = 1;

  parameter real quantum_threshold_pos = 1e-8;
  parameter real quantum_threshold_neg = 1e-8;

  real G,delta_G,nextG;
  real Vthn,Vthp;
  real swenergy1,swenergy2;
  integer direction,int_time;


    analog begin
        @(initial_step("ac","dc","tran")) begin
                G=1/Rinit;
                nextG=G;
                Vthn=-0.24;
                Vthp=0.24;
              $display("Init: %M %f ohm",Rinit);
        end

//evenements
      @(cross(V(control,out)-Vthp, -1)) begin
        if (direction != 0) begin
              direction = 0;
```

```
            //$display("%M phase 1 %d",direction);
            //$display("%M swenergy1 to %E",swenergy1);
            //if (abs(delta_G)>show_variation)
                  $display("etat stable (pos), %M time %e, rfinal %e, delta_G %e, nextG %e, swenergy1
%e",$abstime,1/G,delta_G,nextG,swenergy1);
       end
     end

     @(cross(V(control,out)-Vthp,+1)) begin
       if (G<Gmax) begin
             direction=1;
       end
             //$display("%M phase 2 %d",direction);
             int_time=0;
       end

     @(cross(V(control,out)-Vthn, +1)) begin
       if (direction != 0) begin
            direction = 0;
            //$display("%M phase 3 %d, delta_G %e",direction,delta_G);
            //$display("%M swenergy2 to %E",swenergy2);
            //if (abs(delta_G)>show_variation)
                  $display("etat stable (neg), %M time %e, rfinal %e, delta_G %e, nextG %e, swenergy2
%e",$abstime,1/G,delta_G,nextG,swenergy2);
       end
     end

     @(cross(V(control,out)-Vthn,-1)) begin
       if (G>Gmin) begin
             direction=-1;
       end
             //$display("%M phase 4 %d",direction);
             int_time=0;
     end



//integrale
        swenergy1=idt(V(control,out),0,int_time);

        swenergy2=idt(V(control,out),0,int_time);

   if (direction == 0) int_time=1;

/*par pas de simulation*/
   if ((direction == 1) && (G==nextG) && (abs(swenergy1)>quantum_threshold_pos) && (G<Gmax))
begin
     delta_G = ohm_step_p*swenergy1;  //appoximation lineaire
     if ( (G + delta_G)>0 ) begin
            nextG = G + delta_G;
        end
     int_time=0;
     $display("pos, %M time %e, swenergy1 %e, delta_G %e",$abstime,swenergy1,delta_G);
     swenergy1=0;
   end

   if ((direction == -1) && (G==nextG) && (abs(swenergy2)>quantum_threshold_neg) && (G>Gmin))
begin
     delta_G = ohm_step_n*swenergy2;  //appoximation lineaire
     if ( (G + delta_G)>0 ) begin
```

```
            nextG = G + delta_G;
        end
      int_time=0;
      $display("neg, %M time %e, swenergy2 %e, delta_G %e",$abstime,swenergy2,delta_G);
      swenergy2=0;
    end

    G = transition(nextG, tconv, trise, tfall);

    I(control,out) <+ V(control,out) * G;

    @(final_step("ac","dc","tran","xf" )) begin
       $display("%M %f ohm",1/G);
    end


    end


endmodule
```

II. Codes sources matlab (Chapitre 4)

1. Rétropropagation

```
function
[W,DefectCells_Order,DefectCells_Value,DefectSynapses_Order,DefectSynapses_Value,E,cvsteps]=
backprop(target,input,Nri,NeurLayer,Nro,eta,maxiter,nthreshold,defect_proba_neur,defect_proba_syn
ap,weight_noise,W_in,init_dispersion)
% USAGE

seuil_1(1:nthreshold)=1;
seuil_0(1:nthreshold)=0;

origine_target=target;
[origine_nresponses,origine_ncellsO]=size(origine_target);
output_elements=origine_nresponses*origine_ncellsO;
%duplication
input=repmat(input,1,Nri);
target=repmat(target,1,Nro);

[nstim,ncellsO]=size(target);
[nrepons,ncellsI]=size(input);
if nstim~=nrepons & ncellsI==ncellsO;
  target=target';input=input';
  [nstim,ncellsO]=size(target);
  [nrepons,ncellsI]=size(input);
end;
if nstim~=nrepons;
%Target et Input doivent etre conformables
error('Le nombre de résultats DOIT ETRE egale au nombre de stimulus');
error('ou Le nombre de dimensions entre résultats et stimulus DOIT ETRE egale');
end;

size_NeurLayer=size(NeurLayer);
nLevel=size_NeurLayer(2)+1;

%initialisation
%%matrices de poids :
   %%distribution G. ou U.
```

```
   %%converge pas si les W possèdent des valeurs
   %%excessivement grand
   W{1}=randn(ncellsI+nthreshold,NeurLayer(1))/10000;
   nSynapses(1)=(ncellsI+nthreshold)*NeurLayer(1);
   for NumLayer=2:nLevel-1
      nSynapses(NumLayer)=(NeurLayer(NumLayer-1)+nthreshold)*(NeurLayer(NumLayer));
      W{NumLayer}=randn(NeurLayer(NumLayer-1)+nthreshold,NeurLayer(NumLayer));
   end;
   nSynapses(nLevel)=(NeurLayer(nLevel-1)+nthreshold)*ncellsO;
   W{nLevel}=randn(NeurLayer(nLevel-1)+nthreshold,ncellsO)/10000;

if nargin==11;
   %
else
   %%dispersion
   %%duplication de poids
   for iLevel=1:nLevel
      size_W_in=size(W_in{iLevel});
      W{iLevel}=normrnd(W_in{iLevel}(1:size_W_in(1),1:size_W_in(2)),init_dispersion);
   end;
end;


nCells=[ncellsI NeurLayer ncellsO];
size_nCells=size(nCells);
nLayer=size_nCells(2);
sprintf('nombre de couche de neurones %d',nLayer)
%%generation de fautes d'états dans un désordre
for NumLayer=1:nLayer;
   Err_Value=[0];
   Err_Count=0;
   for idef=1:nCells(NumLayer)
      if rand<defect_proba_neur(NumLayer);
         Err_Count=Err_Count+1;
         if rand<0.5
            if rand<0.5
               Err_Value(Err_Count)=-inf;
            else
               Err_Value(Err_Count)=inf;
            end;
         else
            if rand<0.5
               Err_Value(Err_Count)=randn/2.7;
            else
               Err_Value(Err_Count)=0;   %0.001 ou eps
            end;
         end;
      end;
   end;
   NumberDefectLayer{NumLayer}=Err_Count;
   DefectCells_Value{NumLayer}=Err_Value;
   DefectCells_Order{NumLayer}=randperm(nCells(NumLayer));
   DefectCells_Order{NumLayer}=DefectCells_Order{NumLayer}(1:Err_Count);
end;

NumberDefectLayer
nCells
%%generation de fautes de poids synaptiques dans un désordre
for NumLayer=1:nLevel;
   Err_Value=[0];
```

```
   Err_Count=0;
   for idef=1:nSynapses(NumLayer)
      if rand<defect_proba_synap(NumLayer);
         Err_Count=Err_Count+1;
         %remarque la valeur 'alea' de conductance est absente ici
         Err_Value(Err_Count)=inf;
      end;
   end;
   NumberDefectSynapsesLayer{NumLayer}=Err_Count;
   DefectSynapses_Value{NumLayer}=Err_Value;
   DefectSynapses_Order{NumLayer}=randperm(nSynapses(NumLayer));
   DefectSynapses_Order{NumLayer}=DefectSynapses_Order{NumLayer}(1:Err_Count);

   %teste des cas avec defauts uniquement
   if (Err_Count<1) && (defect_proba_synap(NumLayer)>0);
      E=0;
      cvsteps=0;
      return;
   end;

end;

NumberDefectSynapsesLayer
nSynapses
% precision
niter=0;
e=ones(nstim,ncellsO);
err=zeros(1,maxiter);

O{1}=input;

while (niter < maxiter);
  niter=niter+1;
%go
  ordre=randperm(nstim);
  for k=1:nstim;
    %commence une iteration
    np=ordre(k);
    %Forward
    Act_State{1}=input(np,:);
    for NumLayer=1:nLevel
       Act_State{NumLayer}=[Act_State{NumLayer},seuil_1];
       %%injection de faute(collage) d'états
       if NumberDefectLayer{NumLayer}>0
            Act_State{NumLayer}(DefectCells_Order{NumLayer})=DefectCells_Value{NumLayer};
       end;
       N_WSum{NumLayer+1}=Act_State{NumLayer}*W{NumLayer};
       Act_State{NumLayer+1}=bp_active_f(N_WSum{NumLayer+1});
    end;
    %%simulation : injection erreur à la sortie
    if NumberDefectLayer{nLayer}>0
         Act_State{nLayer}(DefectCells_Order{nLayer})=DefectCells_Value{nLayer};
    end;

    %calcul de l'erreur
    %Backward
    Error{nLayer}=target(np,:)-Act_State{nLayer};
    Derr{nLevel}=Error{nLayer}.*bp_active_fp(N_WSum{nLayer});
    W{nLevel}=W{nLevel}+(eta*(Act_State{nLevel}'*Derr{nLevel}));
    %%bruits
```

```
    if weight_noise>0
       W{nLevel}=normrnd(W{nLevel},weight_noise);
    end;
    if NumberDefectSynapsesLayer{nLevel}>0
      W{nLevel}(DefectSynapses_Order{nLevel})=DefectSynapses_Value{nLevel};
    end;
    Error{nLevel}=Derr{nLevel}*W{nLevel}';
    fprintf('epoch %d',niter);
    for NumLayer=nLevel-1:-1:1
       Derr{NumLayer}=Error{NumLayer+1}.*([bp_active_fp(N_WSum{NumLayer+1}),seuil_0]);
       NumLayer
       W{NumLayer}
       (eta*(Act_State{NumLayer}'*Derr{NumLayer}(1:NeurLayer(NumLayer))))

W{NumLayer}=W{NumLayer}+(eta*(Act_State{NumLayer}'*Derr{NumLayer}(1:NeurLayer(NumLayer)))
);
       %%bruits
       if weight_noise>0
         W{NumLayer}=normrnd(W{NumLayer},weight_noise);
       end;
       if NumberDefectSynapsesLayer{NumLayer}>0
         W{NumLayer}(DefectSynapses_Order{NumLayer})=DefectSynapses_Value{NumLayer};
       end;
       if NumLayer>1
          Error{NumLayer}=Derr{NumLayer}(1:NeurLayer(NumLayer))*W{NumLayer}';
       end;
    end;

   %%fin d'une iteration
   end;
 %%fin d'une epoque

  %%calcul global, feed-forward
  for iLevel=1:nLevel
       WSum{iLevel+1}=[O{iLevel},ones(nstim,nthreshold)]*W{iLevel};
       O{iLevel+1}=bp_active_f(WSum{iLevel+1});
  end;
  Sortie=O{nLevel+1};

 if Nro>1;
   %%moyenne des sorties redondantes puis binarisation
   Sortie=mean(reshape(Sortie,output_elements,Nro)');
   Sortie=reshape(Sortie,origine_nresponses,origine_ncellsO);
 end;

 %%logic binaire -1/1
 %%le seuillage peut etre remplacé par la fonction d'activation(echelon...)
 Sortie=double(Sortie>0);Sortie(Sortie==0)=-1;
 err_total=norm(Sortie-origine_target);

 err(niter)=err_total;

  if err_total<1
    %%mission accompli
    disp(niter);
    E=err(1:niter);
    cvsteps=niter;
    return;
 end;
end;
```

```
%%donne la reponse finale
E=err(1:niter);
disp('inf');
cvsteps=inf;
```

2. Machine de Boltzmann

```
%%Michel HE - 2006, IEF
function
[W,DefectCells_Order,DefectCells_Value,DefectSynapses_Order,DefectSynapses_Value,eoutput,cvst
eps]=boltzmann_machine(target,input,Nri,NeurLayer,Nro,eta,maxiter,nthreshold,defect_proba_neur,d
efect_proba_synap,weight_noise,W_in,init_dispersion)
% USAGE : Machine de Boltzmann
seuil_1(1:nthreshold)=1;
seuil_1=seuil_1';
%%initialisation
origine_target=target;
[nrespons,origine_ncellsO]=size(origine_target);
%%duplication d'entrée et sortie
%%la redondance Nr est entré en paramètre
input=repmat(input,1,Nri);
target=repmat(target,1,Nro);

%%nouvelles matrices :
[nstim,ncellsI]=size(input);
[nrepons,ncellsO]=size(target);

%%sert à tracer le graphe
eoutput=(1:maxiter);
eoutput=zeros(maxiter,1);

%%iteration pour la somme Pij
maxsteps=100;

%%taux de décroissance de la température
%%(plus le taux est grand,
%%moins la température baissera rapidement)
%%peut conduire à non-convergence
%%si 0 est atteint trop rapidement
txtemp=20;

%%generation de valeurs de températures
temp=exp(-[1:maxsteps]/txtemp);

%%generation de fautes d'états dans un désordre

size_NeurLayer=size(NeurLayer);
nLevel=size_NeurLayer(2)+1;

nCells=[ncellsI NeurLayer ncellsO];
size_nCells=size(nCells);
nLayer=size_nCells(2);

%initialisation
%%matrices de poids :
   %%distribution G. ou U.
   %%converge pas si les W possèdent des valeurs
```

```
   %%excessivement grand
   %W{1}=rand((ncellsI+nthreshold),NeurLayer(1))-0.5;
   W{1}=zeros((ncellsI+nthreshold),NeurLayer(1));
   nSynapses(1)=(ncellsI+nthreshold)*NeurLayer(1);
   %W{1}=rand(ncellsI,NeurLayer(1))-0.5;
   %nSynapses(1)=ncellsI*NeurLayer(1);
   for NumLayer=2:nLevel-1
      nSynapses(NumLayer)=(NeurLayer(NumLayer-1)+nthreshold)*NeurLayer(NumLayer);
      %W{NumLayer}=rand((NeurLayer(NumLayer-1)+nthreshold),NeurLayer(NumLayer))-0.5;
      W{NumLayer}=zeros((NeurLayer(NumLayer-1)+nthreshold),NeurLayer(NumLayer));
   end;
   nSynapses(nLevel)=(NeurLayer(nLevel-1)+nthreshold)*ncellsO;
   %W{nLevel}=rand((NeurLayer(nLevel-1)+nthreshold),ncellsO)-0.5;
   W{nLevel}=zeros((NeurLayer(nLevel-1)+nthreshold),ncellsO);

if nargin==11;
   %
else
   %%dispersion
   %%duplication de poids
   for iLevel=1:nLevel
      size_W_in=size(W_in{iLevel});
      W{iLevel}=normrnd(W_in{iLevel}(1:size_W_in(1),1:size_W_in(2)),init_dispersion);
   end;
end;


sprintf('nombre de couche de neurones %d',nLayer)
%%generation de fautes d'états dans un désordre
for NumLayer=1:nLayer;
   Err_Value=[0];
   Err_Count=0;
   for idef=1:nCells(NumLayer)
      if rand<defect_proba_neur(NumLayer);
         Err_Count=Err_Count+1;
         if rand<0.5
            if rand<0.5
               Err_Value(Err_Count)=-1; %inf;
            else
               Err_Value(Err_Count)=1; %inf;
            end;
         else
            if rand<0.5
               Err_Value(Err_Count)=randn/2.7;
            else
               Err_Value(Err_Count)=0;   %0.001 ou eps
            end;
         end;
      end;
   end;
   NumberDefectLayer{NumLayer}=Err_Count;
   DefectCells_Value{NumLayer}=Err_Value;
   DefectCells_Order{NumLayer}=randperm(nCells(NumLayer));
   DefectCells_Order{NumLayer}=DefectCells_Order{NumLayer}(1:Err_Count);
end;
NumberDefectLayer
nCells
%%generation de fautes de poids synaptiques dans un désordre
for NumLayer=1:nLevel;
   Err_Value=[0];
```

```
    Err_Count=0;
    for idef=1:nSynapses(NumLayer)
        if rand<defect_proba_synap(NumLayer);
            Err_Count=Err_Count+1;
            Err_Value(Err_Count)=inf;
        end;
    end;
    NumberDefectSynapsesLayer{NumLayer}=Err_Count;
    DefectSynapses_Value{NumLayer}=Err_Value;
    DefectSynapses_Order{NumLayer}=randperm(nSynapses(NumLayer));
    DefectSynapses_Order{NumLayer}=DefectSynapses_Order{NumLayer}(1:Err_Count);

    %teste des cas avec defauts uniquement
    if (Err_Count<1) && (defect_proba_synap(NumLayer)>0);
        E=0;
        cvsteps=0;
        return;
    end;


end;

NumberDefectSynapsesLayer
nSynapses
   %%attribuer les etats aleatoires dans les couches cachées + seuil
    for iLayer=2:nLayer-1
        Act_State{iLayer}=2*(randn(nCells(iLayer),1)>0)-1;
        Act_State{iLayer}=[Act_State{iLayer};seuil_1];
        %%erreur de collage
        if NumberDefectLayer{iLayer}>0
           Act_State{iLayer}(DefectCells_Order{iLayer})=DefectCells_Value{iLayer};
        end;
    end;
    Act_State{nLayer}=2*(randn(nCells(nLayer),1)>0)-1;
    %%erreur de collage
    if NumberDefectLayer{nLayer}>0
       Act_State{nLayer}(DefectCells_Order{nLayer})=DefectCells_Value{nLayer};
    end;

        disp(0);
        W{1}
        W{2}

for epoch=1:maxiter
     %pstim=randperm(nstim);
     pstim=1:nstim;
     err=0;
     for k=1:nstim;

        Act_State{1}=[input(pstim(k),:)';seuil_1];
        %%erreur de collage
        if NumberDefectLayer{1}>0
          Act_State{1}(DefectCells_Order{1})=DefectCells_Value{1};
        end;

            %%
            XP{1}=Act_State{1};
            %%

        %%initialise les co-activités
```

```
    for iLevel=1:nLevel
       Pij_relax{iLevel}=zeros(nCells(iLevel)+nthreshold,nCells(iLevel+1));
       Pij_force{iLevel}=zeros(nCells(iLevel)+nthreshold,nCells(iLevel+1));
    end;



    %%phase -
    %%force l'entrée, sortie libre
    for stabiliz=1:maxsteps

       for iLevel=1:nLevel-1

          %mode feedforward
          XP{iLevel+1}=bm_active_f(W{iLevel}'*Act_State{iLevel},temp(stabiliz));

%aspect temportel
%Act_State(T+1) -> XP
%aspect de couche
%X -> Y


Pij_relax{iLevel}=Pij_relax{iLevel}+kron(Act_State{iLevel},XP{iLevel+1}')+kron(Act_State{iLevel+1}(1:n
Cells(iLevel+1))',Act_State{iLevel});
          %  X(T).Y(T+1)+Y(T).X(T+1)

       end;

          iLevel=nLevel;
          XP{iLevel+1}=bm_active_f(W{iLevel}'*Act_State{iLevel},temp(stabiliz));


Pij_relax{iLevel}=Pij_relax{iLevel}+kron(Act_State{iLevel},XP{iLevel+1}')+kron(Act_State{iLevel+1}(1:n
Cells(iLevel+1))',Act_State{iLevel});

       %%%%%%%%%%%%%%%%%%%%%%%%%%
       %%mise a jour des états %%
       %%%%%%%%%%%%%%%%%%%%%%%%%%
       for iLayer=2:nLayer-1
          Act_State{iLayer}=[XP{iLayer};seuil_1];
          %%erreur de collage
          if NumberDefectLayer{iLayer}>0
             Act_State{iLayer}(DefectCells_Order{iLayer})=DefectCells_Value{iLayer};
          end;
       end;
          Act_State{nLayer}=XP{nLayer};
          %%erreur de collage
          if NumberDefectLayer{nLayer}>0
             Act_State{nLayer}(DefectCells_Order{nLayer})=DefectCells_Value{nLayer};
          end;
    end

    if Nro>1;
       %%moyenne des sorties redondantes puis binarisation
       Xfo=mean(reshape(Act_State{nLayer},origine_ncellsO,Nro)')>0;

       %%binaire 0/1 -> -1/+1
       Xfo=2*Xfo-1;
    else
       Xfo=Act_State{nLayer}'
```

```
      end;
      err=err+norm(Xfo-origine_target(pstim(k),:));

     %%phase +
      %%force l'entrée et la sortie
      Act_State{nLayer}=target(pstim(k),:)';
      for iLayer=2:nLayer-1
         Act_State{iLayer}=2*(randn(nCells(iLayer),1)>0)-1;
         Act_State{iLayer}=[Act_State{iLayer};seuil_1];
      end;

      for stabiliz=1:maxsteps
         for iLevel=1:nLevel-1
            XP{iLevel+1}=bm_active_f(W{iLevel}'*Act_State{iLevel},temp(stabiliz));

Pij_force{iLevel}=Pij_force{iLevel}+kron(Act_State{iLevel},XP{iLevel+1}')+kron(Act_State{iLevel+1}(1:n
Cells(iLevel+1))',Act_State{iLevel});


         end;


Pij_force{nLevel}=Pij_force{nLevel}+kron(Act_State{nLevel},XP{nLevel+1}')+kron(Act_State{nLevel+1}
',Act_State{nLevel});
         %  X(T).Y(T+1)+Y(T).X(T+1)  -- non convergence

%Pij_force{nLevel}=Pij_force{nLevel}+kron(Act_State{nLevel},XP{nLevel+1}')+kron(Act_State{nLevel+
1}',[XP{nLevel};seuil_1]);

         %%%%%%%%%%%%%%%%%%%%%%%%%
         %%mise a jour des états %%
         %%%%%%%%%%%%%%%%%%%%%%%%%
         for iLayer=2:nLayer-1
            Act_State{iLayer}=[XP{iLayer};seuil_1];
            %%erreur de collage
            if NumberDefectLayer{iLayer}>0
               Act_State{iLayer}(DefectCells_Order{iLayer})=DefectCells_Value{iLayer};
            end;
         end;

      end;

      %%modification de poids
      for iLevel=1:nLevel
        %W{iLevel}=W{iLevel}+eta*(Pij_force{iLevel}-Pij_relax{iLevel})/(maxsteps);
        W{iLevel}=W{iLevel}+eta*(Pij_force{iLevel}-0.01)/(2*maxsteps);
        %%ajout de bruits (normal) dans les poids synaptiques :
        if weight_noise>0
           W{iLevel}=normrnd(W{iLevel},weight_noise);
        end;
        %%erreur de collage
        if NumberDefectSynapsesLayer{iLevel}>0
           W{iLevel}(DefectSynapses_Order{iLevel})=DefectSynapses_Value{iLevel};
        end;
      end;
   end;
      disp(epoch);
      W{1}
      W{2}
```

```
        %err
        eoutput(epoch)=err;
        if err<1;
            disp(epoch);
            cvsteps=epoch
            eoutput=eoutput(1:epoch);
            return
        end;
end;
%inf ou >maxiter
cvsteps=inf
```

Liste des travaux publiés

Article avec comité de lecture :
"Mixed analog-digital design of a learning nano-circuit for neuronal architectures"
3rd International Conference on Design and Technology of Integrated Systems in Nanoscale Era, IEEE-DTIS 2008, Tozeur, Tunisie, 25-27 March 2008, ISBN: 978-1-4244-1576-2

Article avec comité de lecture :
"Design and electrical simulation of on-chip neural learning based on nanocomponents"
Electronics Letters, vol. 44, num. 9, pp. 575-576 pages, 2008

Communication orale avec Actes :
"Architecture of Neural Synaptic Array - Design and Simulation"
7th IEEE International Conference on Nanotechnology, IEEE Nano 2007, Hong Kong, Chine, du 02 août 2007 au 05 août 2007; Proceedings of the IEEE, vol. 1, pp. 601-603, 2007, ISBN 1-4244-0608-0

Poster :
"Nano-LUT inspirée de réseaux de neurones"
Journées Nationales du Réseau Doctoral en Microélectronique JNRDM 2007, Lille, France, du 14 mai 2007 au 16 mai 2007

Poster :
"Une alternative pour implémenter la LUT : le crossbar synaptique"
Colloque National du GDR SOC-SIP, Paris, France, du 13 juin 2007 au 15 juin 2007

Poster avec Actes :
"Neural Network for nanoscale architecture"
IEEE Conference on Nanotechnology, Cincinnati, USA, du 16 juillet 2006 au 20 juillet 2006; Proc. 6th IEEE conf. on Nanotechnology, vol. 1, p.p 367-370,