N° d'ordre: 2133

THESE

présentée à

l'Université des Sciences et Technologies de Lille

pour obtenir le grade de

Docteur en Electronique

par

Ian O'CONNOR
MSc, Université d'Essex (GB)

LA CONCEPTION AUTOMATISEE DE CELLULES A COURANTS COMMUTES

Soutenue le 24 novembre 1997, devant la commission d'examen:

Président:	M.	G.	SALMER
Rapporteurs:	M.	Y.	BERTRAND
	M.	G.	GIELEN
	M.	C.	TOUMAZOU
Examinateurs:	M.	H.J.	PRANGER
	M.	J.-N.	DECARPIGNY
	M.	A.	KAISER
	M.	B.	STEFANELLI

A celle qui est à mes côtés
A Sonia

A ceux qui m'inspirent et me confondent
A mes enfants

Cette thèse a été préparée au sein de :

l'Institut d'Electronique et de Microélectronique du Nord (IEMN)
UMR CNRS 9929
Département ISEN
41 boulevard Vauban
F-59046 LILLE Cedex
FRANCE

REMERCIEMENTS

Je souhaite exprimer ma profonde reconnaissance envers mon directeur scientifique, Andreas Kaiser. Sa grande expérience, sa générosité et son attention aux détails ont été des facteurs déterminants pour l'aboutissement de ce travail et à d'autres projets pendant mon séjour à l'ISEN (Institut Supérieur d'Electronique du Nord).

Je voudrais remercier Monsieur le Professeur Georges Salmer, de l'Université des Sciences et Technologies de Lille, d'avoir accepté la présidence de ce jury.

Mes remerciements vont également à Monsieur le Professeur Yves Bertrand, de l'Université de Montpellier, à Monsieur le Professeur Georges Gielen, de la Katholieke Universiteit Leuven, et à Monsieur le Professeur Chris Toumazou, de l'Imperial College London, qui ont consacré du temps et de l'énergie pour juger ce manuscrit en tant que rapporteurs.

J'adresse ma reconnaissance à Monsieur Henk Jan Pranger, de Philips Research Eindhoven, pour avoir consenti à participer à ce jury.

Je suis également très reconnaissant envers Monsieur le Professeur Jean-Noël Decarpigny et Monsieur le Professeur Michel Lannoo, de m'avoir permis d'accomplir ce travail au sein de l'ISEN, et d'acquérir une expérience en enseignement.

Je dois énormément à Bruno Stefanelli, avec qui j'ai partagé un "espace de conception" pendant la plupart de mon séjour à l'ISEN. J'ai particulièrement apprécié sa patience et son énergie sans borne lors des projets de conception en courants commutés, où il a toujours été prêt à m'expliquer tout point technique. Il s'est également proposé de corriger cette thèse (en anglais et en français), chose pour laquelle je lui suis très reconnaissant.

Laurent Quiquerez était le deuxième correcteur. Je souhaite le remercier pour les soirées qu'il a consacré à l'aboutissement sérieux de cette tâche, pour son humour, et pour sa contribution à l'ambiance agréable dans le département.

La programmation en C++ serait restée pour moi un mystère sans les innombrables discussions que j'ai eues avec Sébastien Bozek. Son attitude libre et généreuse, ainsi que son approche méticuleuse des problèmes, ont toujours contribué à me donner plus que je ne demandais.

Je voudrais également remercier Jean-Michel Droulez, notre administrateur système, pour avoir assuré le bon fonctionnement de toute chose en silicium, et pour avoir renoncé un instant à sa fierté chauvine pour consentir à appeler mon disque "trafalgar" !

Mes remerciements vont aussi à la secrétaire du département, Valérie Vandenhende, qui a assuré le bon fonctionnement de toute chose hors silicium.

Mes quatre années à l'ISEN ont été quatre années satisfaisantes, pendant lesquelles j'ai été sans cesse incité à développer mes capacités au niveau recherche, enseignement mais également sur le plan personnel. Je souhaite remercier tous ceux qui ont participé à ce développement.

J'exprime ma gratitude à mes parents, qui m'ont donné la possibilité d'étudier, et d'atteindre le point où le travail n'est plus un fardeau, mais un plaisir. Sans leur soutien, rien n'aurait été possible.

Je voudrais également remercier ma belle-famille : Marielle et François, pour m'avoir fourni ordinateur portable, jardin, soleil et café lors d'un week-end de traduction acharnée ; Yanis et Bénédicte, pour avoir accepté de travailler un 14 juillet ; et mes beaux-parents, pour leur soutien pendant mes années en France.

Ce travail a été soutenu financièrement par l'association ISEN-Recherche et le Conseil Régional du Nord-Pas de Calais.

RÉSUMÉ

Les éléments de traitement de signal analogique ou de conversion sont de plus en plus amenés à cohabiter avec des circuits numériques sur le même substrat, afin de répondre aux besoins de secteurs tels que les télécommunications et l'imagerie médicale. La technique des courants commutés permet à ces éléments d'être fabriqués dans une technologie CMOS standard, et donc à moindre coût. Mais malgré cet avantage majeur, les circuits qui utilisent cette technique sont rares, ce qui est principalement dû à la tâche difficile de dimensionnement et au manque d'outils CAO venant en support de la technique.

Cette thèse présente un outil pour assister la conception systématique de tels circuits. Afin de permettre une certaine flexibilité de conception, l'outil reste à un niveau hiérarchique bas. Pour accélérer le temps de conception, des modèles analytiques sont utilisés, associés à une optimisation à base de règles.

Les modèles analytiques réduisent fortement le temps nécessaire à l'évaluation des performances par rapport au temps requis par la simulation numérique. Cependant, cette dernière est toujours nécessaire afin de compenser les imprécisions inhérentes aux modèles analytiques.

L'optimisation numérique, dans sa forme traditionnelle, procède de manière purement quantitative. Elle peut appliquer des variations de paramètres qu'aucun concepteur ne considérerait. L'inclusion de connaissances de conception dans un algorithme d'optimisation réduit la taille du vecteur de variables de conception, et peut augmenter de façon significative le rendement de l'algorithme.

Ces aspects ont été combinés pour créer un outil de conception automatisée dédié aux cellules à courants commutés. Plusieurs topologies ont été incluses dans la base de données, et l'extension de la bibliothèque constitue un processus simple. De nombreux exemples de circuit ont été générés avec cet outil.

ABSTRACT

Analog signal processing and data conversion elements are increasingly required to be implemented alongside digital circuitry, in order to meet the changing needs of sectors such as the telecommunications and medical imaging industries. The switched-current circuit technique allows these elements to be fabricated in a conventional CMOS process, and thus at lower cost. Despite this major advantage, circuits using this technique are uncommon, largely due to the difficult task of sizing and the lack of CAD tools supporting the technique.

This thesis presents a CAD tool supporting the systematic design of such circuits. To allow the user design flexibility, the tool remains at a low hierarchical level. Analytical models are used, coupled with rule-based optimization, to accelerate the design time.

Analytical models dramatically reduce the time necessary for performance evaluation, with respect to the time required by numerical simulation. However, the latter is still required in the outer optimization loop in order to compensate for the inherent inaccuracy of the analytical models.

Numerical optimization in its traditional form proceeds on a purely quantitative basis, and can apply parameter variations which no designer would consider. The inclusion of design knowledge into an optimization algorithm reduces the size of the design variable vector, and can significantly increase the efficiency of the algorithm.

These aspects have been combined to create a design automation tool specific to switched-current cells. A number of topologies have been included in the database, and library extension is a straightforward process. Several example designs have been generated with this tool.

TABLE DES MATIÈRES

INTRODUCTION

Les diminutions incessantes de la taille du transistor MOS permettent à présent la fabrication de systèmes complets sur un seul circuit intégré. En conjonction avec l'explosion du secteur des télécommunications, ceci a engendré un besoin pour des circuits intégrés de traitement de signal. Ce besoin est comblé en grande partie dans le domaine numérique qui peut facilement abaisser le plancher de bruit en-dessous de celui des circuits analogiques comparables. Cependant, les fonctions analogiques sont toujours nécessaires car les éléments intégrés de traitement de signal numérique doivent communiquer avec le monde réel analogique, d'où le besoin de convertisseurs de données. Selon une communication récente [EBN97], les composants semiconducteurs analogiques peuvent représenter jusqu'à un quart du coût total des composants semiconducteurs dans des produits de grand public tels que DVD (disque vidéo numérique - "digital video disc"), HDTV (télévision à haute définition - "high-definition television"), LCD (moniteurs à cristaux liquides - "liquid crystal displays"), WebTV et le "home theatre". En outre, il y a une renaissance de la demande pour des fonctions analogiques, en raison des contraintes de basse puissance dans les applications portables, ou encore de fonctionnement à haute fréquence dans les produits de télécommunications. Ces domaines peuvent être couverts par des circuits analogiques qui utilisent plus judicieusement que leurs équivalents numériques des ressources telles que la surface active ou la consommation.

Les solutions entièrement intégrées sont donc en passe de devenir de plus en plus répandues. Les performances du système sont améliorées par rapport aux circuits discrets, du fait de capacités et de résistances parasites moins importantes. D'un autre côté, l'augmentation de la diaphonie et la maîtrise réduite du concepteur par rapport aux paramètres critiques des circuits, causée par la dispersion des paramètres du procédé de fabrication, font surgir de nouvelles difficultés. Néanmoins, les arguments économiques plaident en faveur de la solution intégrée (Figure 1).

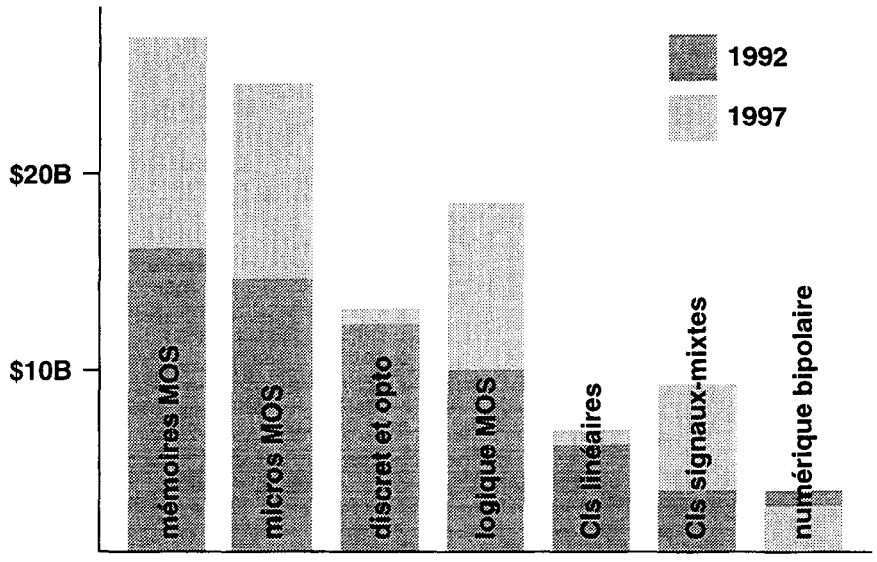


Figure 1 Evolution du marché des semiconducteurs 1992-1997

L'augmentation de l'activité de conception d'ASICs mixtes s'est manifestée par une croissance de l'intérêt pour les techniques analogiques échantillonnées pour la réalisation des convertisseurs de données, des filtres et des lignes à retard. Un souhait primordial est la réalisation de tels circuits sur le même substrat que les composants numériques. De plus, leur réalisation doit éviter de recourir à des étapes de fabrication supplémentaires et coûteuses ; c'est-à-dire en utilisant un procédé de fabrication CMOS numérique standard.

Traditionnellement, la technique la plus utilisée a été celle des capacités commutées. Cependant, ce genre de circuit nécessite des capacités flottantes (deux électrodes indépendantes des tensions d'alimentation). Si celles-ci ne sont pas réalisées par deux couches de polysilicium (une étape supplémentaire de fabrication), elles doivent l'être par des capacités métal-métal ou métal-polysilicium. Souvent, ceci implique une plus grande distance entre les armatures (optimisée pour minimiser la diaphonie et assurer une bonne isolation électrique entre les chemins des signaux ou des alimentations), et par conséquent une plus grande surface de silicium doit être occupée afin d'obtenir la valeur de capacité nécessaire. La valeur de la capacité parasite entre l'armature inférieure et le substrat est généralement importante, allant jusqu'à la moitié de celle de la capacité utile, ce qui limite l'utilisation de capacités flottantes dans les procédés CMOS standards. Ces facteurs, ainsi que le désappariement des transistors et l'augmentation de la résistance des interrupteurs, nuisent à la performance des circuits à capacités commutées.

Une nouvelle approche pour la réalisation de fonctions analogiques dans le domaine échantillonné est la technique des courants commutés [DAU88]. L'information est véhiculée sous forme de courant, ce qui en principe réduit les contraintes sur la tension d'alimentation. L'information est mémorisée sur la capacité de grille d'un transistor unique. De ce fait, aucune capacité flottante n'est requise, et le désappariement n'est

plus une préoccupation car l'acquisition et la restitution de l'information sont réalisées par ce même transistor.

Malgré ces avantages, la technique des courants commutés n'a pas pour l'instant réussi à se répandre dans le monde de conception de CIs analogiques au-delà des sphères académiques. De nombreux facteurs sont à l'origine de cet état de fait : sa nouveauté (l'idée a été proposée en 1988), le manque de techniques de conception formalisées, la tâche relativement difficile de dimensionnement, et le manque d'outils CAO venant en support de la technique. La simulation pertinente des circuits analogiques échantillonnés est notoirement difficile, ainsi que le développement de connaissances de conception formalisées. Cependant, le grand nombre d'outils de simulation et de synthèse dédiés à l'approche des capacités commutées témoigne de l'utilisation très répandue de cette technique dans les environnements industriels et académiques. Il n'en va pas de même pour la technique des courants commutés, à quelques exceptions près [GAT96] [HUG96] [NGO95].

La conception automatisée (CA) est un domaine bien établi dans la communauté de la conception numérique. En ce qui concerne la conception analogique, elle l'est beaucoup moins. Une grande partie de cette dernière est effectuée par des experts qui utilisent des connaissances intuitives de conception qui sont acquises par expérience. Ces dernières supplantent dans une grande mesure les connaissances formelles de conception apprises lors de la formation initiale. Ce déséquilibre dans les approches de conception du numérique et de l'analogique a conduit le développement des CIs signaux-mixtes à la situation illustrée par la Figure 2.

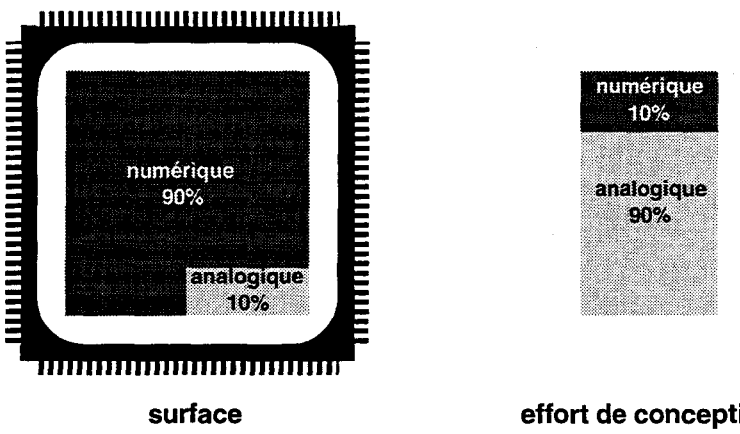


Figure 2 Distribution de l'effort de conception de CIs à signaux-mixtes

La simulation numérique est abondamment utilisée en conception analogique, du fait que des simulations, même répétées, coûtent beaucoup moins cher que la fabrication d'un seul CI prototype. Cependant, la partie analogique d'un CI est la plus susceptible aux erreurs, et peut toujours, malgré les simulations, être à l'origine d'un certain nombre d'itérations lors de la fabrication. Cet aspect, ainsi que le long processus de conception manuelle, retarde le délai de mise sur le marché, aspect de plus en plus

critique dans le monde des CIs à signaux-mixtes. Il existe donc un fort besoin en CA analogique (comme pour la CA numérique) qui fournisse un moyen d'obtenir un circuit correct par conception. Cela permettrait de réduire à la fois le temps de conception des fonctions analogiques et le nombre d'itérations lors de la fabrication. L'utilisation répétitive d'outils comportementaux de conception peut également aider le concepteur à explorer l'espace de conception et à faire de meilleurs choix pour les compromis critiques et les facteurs qui les affectent.

La raison pour laquelle la CA analogique n'est pas aussi répandue que son équivalent numérique est propre à la conception analogique. C'est un domaine intuitif et créatif, où une grande partie des connaissances est difficile à formaliser et le serait différemment par chaque concepteur. L'espace de conception est complexe, avec un nombre important de degrés de liberté et de besoins de performance. L'espace de conception numérique est réduit dans la mesure où les variables sont typiquement limitées à un choix de cellules standards et d'interconnexions, et les besoins de performance définis par un équilibre tripartite entre consommation, surface et vitesse.

Les outils existants de CA analogique sont basés sur l'optimisation, les connaissances, ou bien un mélange des deux approches [GIE91].

Les outils basés sur l'optimisation considèrent le processus de conception comme problème d'optimisation, et l'effectuent numériquement. Des algorithmes avancés de recherche d'un minimum global permettent à de tels outils de trouver la meilleure solution en termes de cahier des charges du concepteur. Cependant, ils sont lents et l'utilisateur doit souvent posséder une bonne connaissance de l'algorithme afin de l'utiliser de façon efficace.

A l'autre bout de l'échelle, des outils basés sur les connaissances utilisent des procédures explicites de conception pour dimensionner les circuits. Ces outils sont très rapides et permettent donc une utilisation plus interactive. Pourtant, cet avantage est contrebalancé par le temps nécessaire au codage de la procédure de conception, tâche laborieuse et nécessitant des connaissances approfondies. L'autre inconvénient principal de cette approche est que la solution trouvée par de tels outils n'est pas nécessairement la meilleure. En effet, la procédure de conception a normalement une cible implicite pour réduire soit la consommation, soit la surface active. Un grand nombre d'outils tentent de gérer la conception de façon hiérarchique, ce qui est très difficile à définir dans le domaine analogique, contrairement au domaine numérique.

L'objectif de cette thèse est de contribuer au franchissement de quelques-unes de ces barrières. La finalité du travail consiste à développer un outil dédié à l'automatisation de la conception de cellules mémoire de courant.

Le premier chapitre donne un résumé historique de la CA analogique. Les méthodes et stratégies importantes de la CA analogique sont extraites d'un ensemble d'outils existants. Les avantages et inconvénients de chaque élément sont soulignés. Des

conclusions en sont tirées par rapport aux besoins spécifiques de la CA des circuits à courants commutés.

Une analyse formelle des cellules à courants commutés permet une meilleure compréhension du fonctionnement de ces dernières, et par conséquent de leur conception. Cet aspect est traité dans le deuxième chapitre, consacré à la modélisation analytique de la cellule de mémoire de courant de base. Les expressions mathématiques des erreurs d'échantillonnage qui conditionnent les performances de cette cellule sont données et expliquées d'un point de vue physique et électronique. Finalement, ces équations permettent le développement de règles qualitatives de conception (ou heuristiques[1]) qui facilitent le processus d'optimisation.

Le troisième chapitre détaille les variantes structurelles et opérationnelles de la cellule de mémoire de courant de base. En utilisant des combinaisons de modèles développés, l'extension de la bibliothèque de modèles comportementaux est facilitée. Les modèles n'intègrent que leurs différences par rapport à la structure de base, et des règles qualitatives de conception sont développées de façon similaire.

La validation des modèles analytiques est effectuée dans le quatrième chapitre. Les problèmes liés à la simulation numérique de circuits analogiques à temps discret sont décrits en termes propres à la technique des courants commutés. Un jeu de simulations constituant un environnement complet de validation est ensuite décrit, et des comparaisons en termes de précision et de vitesse d'évaluation sont effectuées entre les modèles comportementaux et les simulations électriques.

Le cinquième chapitre est dédié à l'optimisation à base de règles, qui évite quelques-uns des pièges associés aux outils existants de CA analogique. Le principe du système est suivi par la justification du choix d'un algorithme d'optimisation simple utilisant les règles qualitatives de conception générées précédemment. L'algorithme complet est ensuite détaillé, et des résultats permettent d'évaluer la qualité de la solution finale, aussi bien en termes de performances du circuit synthétisé qu'en termes de rapidité d'obtention de la solution.

Les différents thèmes des chapitres précédents sont rassemblés dans le sixième chapitre, qui décrit l'utilisation de l'outil. L'interface graphique est décrite. Elle concrétise les concepts développés précédemment. Chaque topologie est testée et les résultats de ces tests sont présentés. Cette analyse met en lumière les limites de chaque topologie, ainsi que les capacités de l'outil.

1. heuristique : aide à la découverte, ou une procédure qui vise à réduire l'effort de résolution d'un problème

Références

[DAU88] S.J. Daubert, D. Vallancourt, Y.P. Tsividis, "Current Copier Cells", *Electronics Letters*, vol. 24, no. 25, pp. 1560-1562, December 1988

[EBN97] "Analog Focus: Analog benefits from digital world", Electronics Buyers' News, 10 November 1997

[GAT96] J.W. Gates, E.I. El-Masry, "Switched-Current Analysis Program", *IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 43, no. 1, pp. 24-30, January 1996

[GIE91] G. Gielen, W. Sansen, "Symbolic Analysis for Automated Design of Analog Integrated Circuits", *Kluwer Academic Publishers*, 1991

[HUG96] J.B. Hughes *et al.*, "Automated Design of Switched-Current Filters", *IEEE Journal of Solid State Circuits*, vol. 31, no. 7, pp. 898-907, July 1996

[NGO95] P. N'Goran, A. Kaiser, "A Building Block Approach to the Design and Simulation of Complex Current-Memory Circuits", *Analog Integrated Circuits and Signal Processing*, vol. 7, no. 3, pp. 189-199, *Kluwer Academic Publishers*, May 1995

1 LA CONCEPTION AUTOMATISÉE ANALOGIQUE AU NIVEAU CELLULAIRE

Il existe un déséquilibre entre les situations de conception automatisée pour les circuits numériques et les circuits analogiques. Cet état de fait est essentiellement dû au caractère intuitif de la conception analogique. Ce chapitre se consacre à une explication des principales techniques utilisées avec succès dans la conception automatisée analogique. Cependant, nombreuses sont les techniques qui ne peuvent pas s'adapter à la conception automatisée de circuits à courants commutés. Ceci donne lieu au développement d'une architecture d'outil appropriée.

Les premières tentatives de conception automatisée de circuits électroniques remontent sans doute au développement des théories de filtrage lors de la première moitié du vingtième siècle. Armés de fonctions d'approximation des filtres et de méthodologies explicites, les concepteurs pouvaient générer un filtre sans avoir recours aux principes de base ni aux longues analyses manuelles de circuit. Cependant, il a fallu attendre l'avènement de l'ordinateur pour voir apparaître la CA dans sa forme moderne. En effet, l'ordinateur a libéré le concepteur de calculs répétitifs, et les valeurs de composants du circuit pouvaient être générées en fournissant le cahier des charge et en exécutant le programme de calcul.

Mais la taille et la complexité croissantes des circuits, de plus en plus réalisés sous forme intégrée, rendaient bientôt inutilisables les méthodes explicites de CA, et l'utilisation de l'optimisation numérique a commencé. Le domaine de la conception électronique dans lequel ces techniques ont été développées et utilisées de la façon la plus efficace est sans doute le domaine numérique. Seuls quelques exemples isolés en CA analogique, où des méthodes formelles existent (par exemple les filtres), se sont répandus à grande échelle.

L'objectif de ce chapitre est de décrire les raisons d'une si grande disparité entre les situations de CA en numérique et en analogique. Un résumé de quelques unes des techniques qui ont été utilisées dans la CA analogique est présenté, et quelques conclusions seront tirées pour le développement d'un outil destiné à automatiser la conception de cellules à courants commutés.

1.1 La conception automatisée analogique et numérique

L'utilisation d'outils de conception assistée par ordinateur (CAO) pour la conception de CIs est maintenant incontournable dans la chaîne de conception générale. Pour les circuits numériques, la CAO signifie une simulation à base d'évènements (VHDL [LIP89], Verilog [THO90]), et la synthèse à partir d'une description comportementale jusqu'au layout silicium complet. Dans le domaine analogique et mixte, la CAO se restreint à la simulation numérique (SPICE [VLA94], ELDO [ANA94], Spectre [KUN95]) - la CA analogique est pratiquement inexistante dans l'arène industrielle, malgré les avancées significatives effectuées dans ce domaine par les laboratoires académiques depuis le milieu des années 1980.

Afin de comprendre ce manque d'outils de CA analogique, les différences essentielles entre la conception analogique et numérique doivent être expliqués. D'abord, la profusion d'activités dans le domaine de la conception numérique a eu pour résultat la définition de niveaux d'abstraction clairs et bien définis (comportemental, fonctionnel, transfert de registre, porte, transistor). Ces abstractions sont des approximations qui fonctionnent, car les interconnexions numériques à tous les niveaux (sauf celui du transistor) représentent des tensions qui pilotent des noeuds à haute impédance, et les blocs du circuit sont, de ce fait, pratiquement indépendants les uns des autres. Or, la conception analogique n'est pas aussi hiérarchisée, car un bloc fonctionnel peut être réalisé par de nombreux composants de différent niveau : afin de réaliser du gain, par exemple, nous pouvons utiliser un transistor ou un amplificateur opérationnel complet. De plus, l'indépendance d'un bloc analogique d'un autre n'est pas assurée : en effet, le comportement d'un bloc peut être modifié selon les impédances d'entrée et de sortie des blocs en amont et en aval.

Un exemple typique de synthèse numérique consiste à spécifier les contraintes temporelles sur les signaux d'horloge, d'entrée et de sortie, et ensuite à trouver une solution qui optimise la vitesse, la puissance ou la surface (où la vitesse est liée implicitement à la puissance). D'autre part, un exemple relativement simple de conception analogique, comme la spécification d'un amplificateur opérationnel, inclurait nécessairement la spécification du gain, de la bande passante, de la marge de phase, du "slew rate", du taux de rejection des alimentations et du mode commun, de la dynamique de sortie, du bruit, de l'impédance d'entrée et de sortie, de l'offset, de la puissance, de la surface, de la sensibilité aux parasites, de la distorsion harmonique, etc. L'espace de conception est donc bien plus complexe pour les blocs analogiques que pour les blocs numériques. Les variables de conception sont également différentes : la conception numérique utilise des bibliothèques de cellules standards afin de réaliser la fonction désirée ; la conception analogique choisit la topologie et ajuste les tailles de chaque transistor.

Les variations des paramètres technologiques et de l'environnement jouent aussi un grand rôle dans la difficulté de CA en analogique. De légères différences dans le procédé de fabrication, la polarisation, la température ou les composants parasites du layout peuvent influencer les performances du circuit de façon drastique. Cette

sensibilité est moins marquée dans les circuits numériques grâce à la dynamique importante des signaux. Une autre différence fondamentale entre les circuits analogiques et numériques concerne les applications plutôt que les circuits. Les circuits analogiques sont de plus en plus utilisés dans des applications de haute performance, avec des spécifications agressives. Ceci tend à aggraver les problèmes cités précédemment, puisque les circuits de haute performance ne peuvent remplir les spécifications que par une conception vigilante et un layout méticuleux.

1.2 Résumé des techniques de conception automatisée analogique

La plupart des outils de CA analogiques peuvent être classés par catégorie. Une distinction initiale peut d'abord être mise en évidence à travers les variables utilisées et donc inévitablement en fonction du type de circuit ciblé : cellules standards ou spécifiques. Une généralisation supplémentaire peut ensuite être faite, en classifiant les méthodes en celles qui reposent sur l'optimisation numérique, et celles qui font usage de méthodologies explicites de conception, aussi appelées outils à base de connaissances.

1.2.1 L'approche de cellules standards

Le cycle de conception peut être raccourci de façon significative par l'utilisation de bibliothèques de cellules standards, de la même manière que dans le domaine numérique. En effet, cette approche de la CA analogique provient directement des outils existants de CA numérique. Mais le succès de telles techniques dans le domaine numérique, dû à la caractérisation d'une cellule numérique par un modèle simple (bien que ceci devienne de moins en moins vrai lorsque les tailles des dispositifs diminuent et que la fréquence de l'horloge augmente) n'est pas facilement reproductible dans le domaine analogique. La modélisation comportementale de circuits et de systèmes analogiques en est encore à un stade précoce, et ne sera pas universellement utilisée, comme le sont les langages de modélisation comportementale numérique, avant quelques années. Le problème est aggravé principalement par la dépendance d'un bloc analogique à son environnement : aux variations technologiques, mais aussi à sa place dans l'architecture du système, comme décrit précédemment. Un autre problème avec cette approche concerne les applications cibles : les circuits analogiques de haute performance, tels que ceux qui sont de plus en plus requis dans un environnement principalement numérique, ne peuvent que rarement être fabriqués en utilisant une approche de cellules standards. Les cellules standards ne représentent en effet que des points discrets, alors que l'espace de conception de CIs analogiques occupe un spectre quasi-continu. De ce fait, seuls des compromis rudimentaires sont possibles. Il en résulte une approche qui permet rarement l'accomplissement de spécifications contraignantes.

Une variante de cette approche consiste en l'utilisation d'une bibliothèque de modèles à base de tableaux, qui doivent être caractérisés pour toutes les combinaisons possibles de

paramètres de conception et de conditions de fonctionnement. La caractérisation se fait par la simulation numérique extrêmement précise, mais elle utilise d'importantes ressources CPU. Une fois les spécifications de conception satisfaites par l'outil de synthèse (ou plutôt de sélection), la probabilité pour que les performances prédites dévient des performances réelles du fait des imprécisions dans les équations polynômes [OLI97] (utilisées pour réduire la taille des tableaux) est relativement faible. La caractérisation par simulation numérique est effectuée en dehors de la boucle de synthèse, c'est-à-dire en préparation d'une session de synthèse. Toutefois, l'obtention des tableaux peut être un processus de longue durée, car pour n paramètres de conception (les dimensions de la cellule et les conditions de fonctionnement) divisés en m intervalles, le temps nécessaire pour générer les tableaux est donné par :

$$t = m^n T \tag{1.1}$$

où T représente le temps de caractérisation d'une cellule. Donc, même pour des circuits simples comportant peu de variables, et en découpant l'espace de conception de façon grossière, le temps requis pour un tel processus devient facilement prohibitif.

1.2.2 L'approche spécifique

Une grande partie de la conception de circuits intégrés analogiques et mixtes se fait, à l'heure actuelle, en utilisant une approche spécifique, où les tailles des composants sont déterminées individuellement, et peuvent être ajustées finement. Ceci donne au concepteur une flexibilité maximale dans sa recherche du circuit optimal satisfaisant à toutes les spécifications. Les méthodes qui peuvent être utilisées pour le faire automatiquement sont très différentes de celles utilisées pour l'approche à base de cellules standards. Deux techniques existent :

- *basée sur l'optimisation* en utilisant des méthodes numériques,
- *basée sur les connaissances* en utilisant des procédures explicites de conception.

La première approche permet la conception sans connaissances particulières, mais nécessite un temps CPU considérable pour la mener à bien. Cet inconvénient majeur est réduit par les méthodes à base de connaissances. Cependant, la génération de procédures explicites de conception est une tâche de longue durée. Plus récemment, et en ne considérant pas les deux solutions comme mutuellement exclusives, des outils sont apparus qui rendent floue la distinction entre les approches à base d'optimisation et à base de connaissances afin de tirer les avantages des deux (Figure 1.1).

1.2.2.1 L'approche à base d'optimisation

Une approche à base d'optimisation considère le dimensionnement de transistors d'une topologie donnée comme un problème d'optimisation. Les dimensions des composants sont ajustées par des algorithmes d'optimisation itératifs afin de satisfaire les contraintes de l'utilisateur et de minimiser les coûts, en évaluant les performances du circuit à chaque itération par un programme d'analyse de circuit. Le type d'analyse dans la boucle interne d'optimisation permet également une classification en :

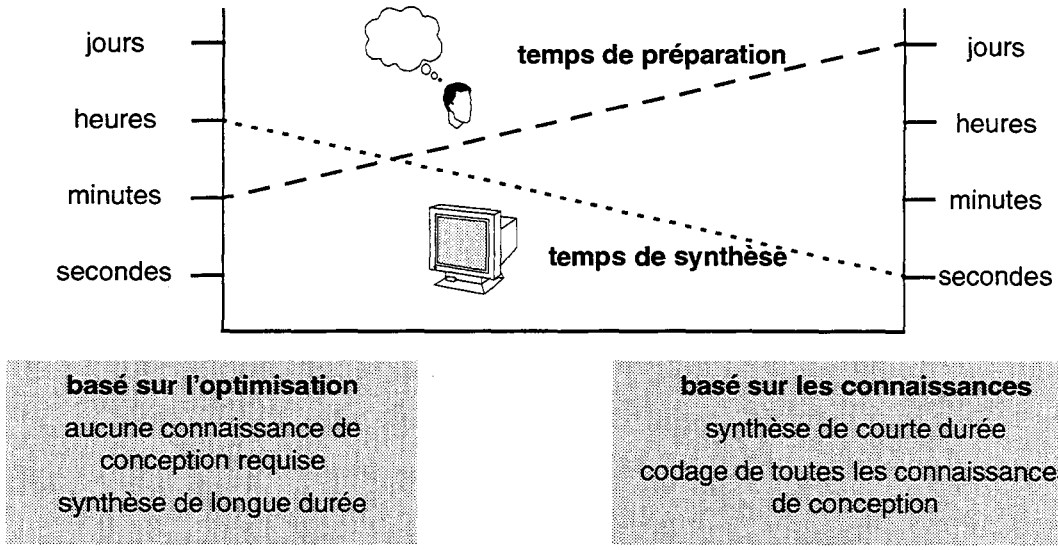


Figure 1.1 Comparaison entre les techniques basées sur l'optimisation et sur les connaissances

- *méthodes à base de simulation*, telles que DELIGHT.SPICE [NYE83], ECSTASY [SHY88] et ADOPT [LAI88], qui utilisent des algorithmes de simulation numérique afin d'évaluer les performances d'un circuit. Cette approche présente l'avantage de rendre l'outil de CA indépendant du circuit. Toutefois, la simulation numérique peut être de longue durée et son inclusion dans la boucle interne d'optimisation ralentit la convergence. Les progrès réalisés dans la puissance de calcul des ordinateurs peuvent compenser cette tendance.

- *méthodes à base d'équations*, telles que OPASYN [KOH90] et ANANAS [PRA92], qui évaluent des équations analytiques à chaque itération. De ce fait, l'évaluation des performances est accélérée par quelques ordres de grandeur, mais l'outil n'est plus indépendant du circuit car les équations analytiques appartiennent aux connaissances spécifiques au domaine de chaque circuit, et la précision peut être réduite du fait des approximations et des simplifications utilisées par les équations analytiques. L'ajout d'une boucle externe utilisant la simulation numérique améliore la précision (Figure 1.2). Les différences entre les valeurs prédites et simulées sont évaluées et utilisées dans la boucle interne d'optimisation. Cependant, des problèmes peuvent survenir lorsque l'erreur de prédiction est grande. En effet, sa valeur n'est pas simplement un décalage constant par rapport à la caractéristique réelle du circuit mais dépend du point évalué, c'est-à-dire du vecteur courant des variables de conception[1].

1. Une distance importante peut toujours exister entre le point calculé et le point optimal après la compensation des équations analytiques et l'optimisation qui s'ensuit. Dans ce cas, l'erreur de prédiction change. C'est la raison pour laquelle la simulation numérique est utilisée à la fin de la boucle interne d'optimisation, plutôt qu'à son début. En effet, le vecteur de départ n'est qu'approximatif et compenser les équations à ce point pourrait avoir un effet néfaste dès lors que l'optimisation est susceptible de changer le vecteur dans une large mesure.

   Si l'erreur de prédiction est de plus de 100%, les spécifications peuvent devenir impossibles à remplir car la valeur avant compensation devrait être nulle ou négative afin de pouvoir y incorporer le terme de compensation.

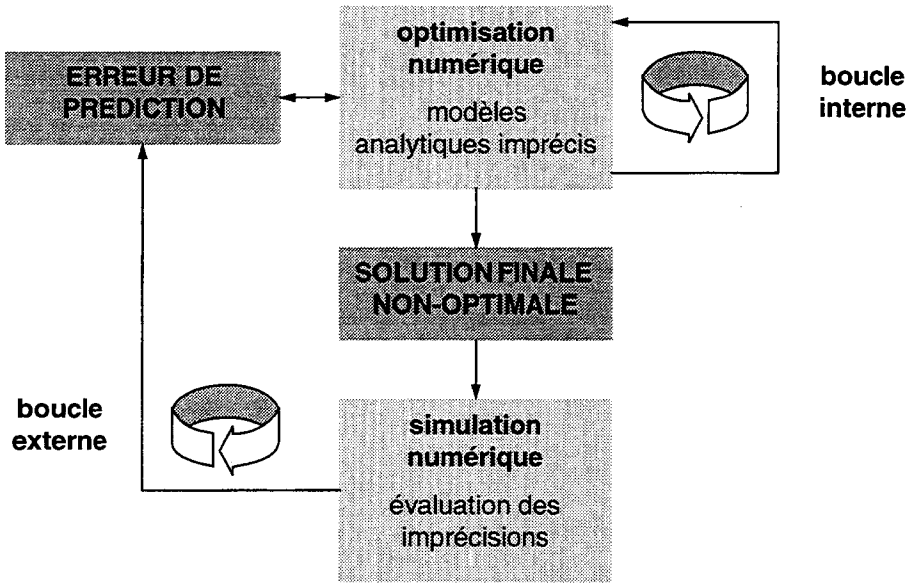


Figure 1.2 Utilisation de la simulation numérique dans la boucle externe d'optimisation afin de compenser les erreurs de prédiction des modèles analytiques

Un autre inconvénient des approches à base d'optimisation est que l'optimiseur ne converge pas nécessairement vers un minimum global, à défaut d'utiliser des techniques de recherche de minimum global telles que les algorithmes de recuit simulé ou de multiples points de départ. Si l'algorithme ne peut trouver qu'un minimum local, et si le point de départ ne constitue pas une bonne estimation initiale, d'une part la solution n'est pas optimale, et d'autre part la convergence peut être lente. Les techniques de recherche de minimum global améliorent les performances du système, mais au prix d'un effort de calcul plus important.

Bien qu'aucune connaissance approfondie de conception de circuits ne soit nécessaire, l'utilisateur doit dans une certaine mesure connaître l'algorithme d'optimisation afin de l'employer de façon efficace. Il doit également être capable de préparer la simulation afin que l'algorithme puisse en extraire les informations significatives pour l'évaluation des performances. Une double compétence est donc requise, aussi bien au niveau de la théorie de l'optimisation qu'au niveau de l'analyse du circuit.

1.2.2.2 L'approche à base de connaissances

Ces systèmes dimensionnent les circuits en utilisant les connaissances de conception explicitement formalisées, afin d'imiter l'approche de l'expert. Ces connaissances regroupent toutes les connaissances spécifiques au domaine, et peuvent inclure des connaissances analytiques, des heuristiques (règles de conception empiriques), des modèles simplifiés, et éventuellement des connaissances topologiques. L'avantage de cette approche réside en des temps de synthèse très courts, et une approche "pousse-bouton". Les connaissances de conception sont transparentes à l'utilisateur, et les spécifications du circuit peuvent être facilement énoncées sans avoir à définir une fonction objectif du circuit. Cependant, il importe de noter que les systèmes à base de

règles, au contraire des approches à base d'optimisation, ne donnent pas nécessairement la solution optimale. Celle-ci peut être atteinte en ajustant la solution donnée par les systèmes à base de connaissances par des méthodes numériques (Figure 1.3).

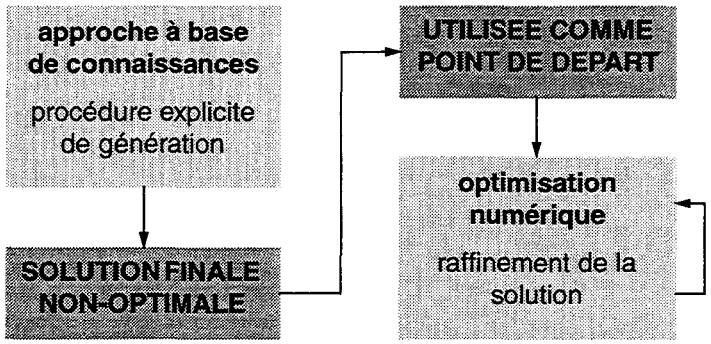


Figure 1.3 Amélioration de la solution donnée par systèmes à base de connaissances

Un classement additionnel des outils à base des connaissances peut être effectué en fonction du mécanisme de gestion des topologies :

- *les approches hiérarchiques* telles que PROSAIC [BOW85], BLADES [ELT89], OASYS [HAR89] et An_Com [BER88], décomposent le circuit ou système requis en parties ou blocs distincts. Un jeu de spécifications est attribué à chaque partie pour que, si les spécifications sont satisfaites, la combinaison des performances de ces parties permette d'obtenir les performances souhaitées du circuit. La même procédure est répétée d'une manière similaire pour les blocs aux niveaux hiérarchiques inférieurs. Un tel partitionnement du circuit implique la décomposition des spécifications, qui est en règle générale formalisée par des équations de conception et des heuristiques.

  Ces systèmes jouissent d'un degré élevé de liberté de conception, car de nouvelles topologies peuvent s'ajouter très facilement par la combinaison de primitives simples. Très peu de primitives sont requises afin d'obtenir un grand nombre d'architectures, la plupart étant, en réalité, des variantes d'une même architecture à quelques modifications mineures près. En raison de la base compacte de données des primitives, de tels systèmes sont faciles à étendre et à maintenir, et font une utilisation maximale des connaissances existantes de conception.

  Cependant, le traitement des échecs est difficile à implémenter dans ces systèmes, car il n'est pas simple de savoir comment modifier le circuit en cours de synthèse. La grande base de données des heuristiques est, contrairement à la base de données de primitives, difficile à maintenir.

- *les approches non hiérarchiques, ou de topologies fixées* telles que IDAC [DEG87] et OAC [ONO89] utilisent des topologies fixées. Celles-ci sont au niveau des dispositifs non-dimensionnés, et sont stockées à l'avance dans une base de connaissances avec les équations explicites de conception nécessaires au dimensionnement des dispositifs. Pour ces outils, les équations sont développées de façon manuelle : c'est pourquoi ces topologies sont fixées. D'autres outils comme ASAIC [GIE91] résolvent ce problème par l'utilisation d'une combinaison de méthodes d'analyse

symbolique pour générer les équations analytiques et de méthodes de programmation par contraintes. Cela permet de transformer ces équations d'analyse en équations explicites de conception.

L'approche des topologies fixées sans analyse symbolique est assez peu flexible et la réutilisation des connaissances est difficile. En effet, chaque nouvelle topologie à ajouter est une tâche majeure, qui requiert les compétences spécialisées du concepteur de l'outil et celles du concepteur expert de circuits. Cependant, certains types de circuit, notamment les circuits non-linéaires, ne peuvent se passer de cette approche.

- *les approches combinées* consistent en une suite d'opérations: génération de la topologie de façon hiérarchique, puis dimensionnement de la topologie une fois celle-ci fixée. D'autres commencent avec une topologie fixe et la dimensionnent, en modifiant par la suite des parties du circuit (ce qui rend l'approche hiérarchique), afin de satisfaire les contraintes et objectifs de l'utilisateur.

1.2.2.3 Les outils de CA analogique : état de l'art

Depuis les premières approches décrites ci-dessus, un nouveau jeu d'outils qui représente l'état de l'art dans la CA en analogique est apparu. Chaque outil repousse en partie les limites constatées précédemment, et chacun le fait différemment. Trois outils représentatifs de ces nouvelles techniques sont :

- *ASAIC* [GIE91]

  Le problème principal dans les systèmes à base de connaissances est que celles-ci doivent être codées de manière rigoureuse, ce qui est en soi un processus de longue durée, nécessitant des connaissances approfondies. ASAIC est le fondement d'un système à base de connaissances qui génère les équations de conception automatiquement, comme illustré dans la Figure 1.4. La génération des équations se fait en deux parties : l'analyse symbolique de la topologie par ISAAC [GIE89] pour déterminer les équations caractéristiques, et ensuite la manipulation des équations de conception du modèle analytique par DONALD, qui transforme le circuit en un plan de solution, convenant à l'optimisation. L'algorithme d'optimisation utilisé est une méthode d'optimisation globale, mais un algorithme local pourrait également être utilisé avec un point de départ donné par DONALD.

  Des points plus traditionnels de ce système peuvent être identifiés, qui sont l'utilisation d'équations analytiques pour l'évaluation des performances du circuit dans la boucle interne d'optimisation, et l'utilisation de la simulation numérique dans la boucle externe d'optimisation. Par la suite, la génération du layout permet l'extraction des parasites et la resimulation pour la validation du circuit sous sa forme physique. Les atouts du système résident en la génération automatique du modèle analytique et en la manipulation des équations de conception. Cependant, la génération des modèles est à présent limitée aux caractéristiques petit-signal, et les caractéristiques grand-signal et transitoire doivent toujours être générés manuellement. Pour la plupart des circuits analogiques continus, les caractéristiques petit-signal couvrent 75% des spécifications ; toutefois, les circuits à courants commutés n'appartiennent pas à cette catégorie.

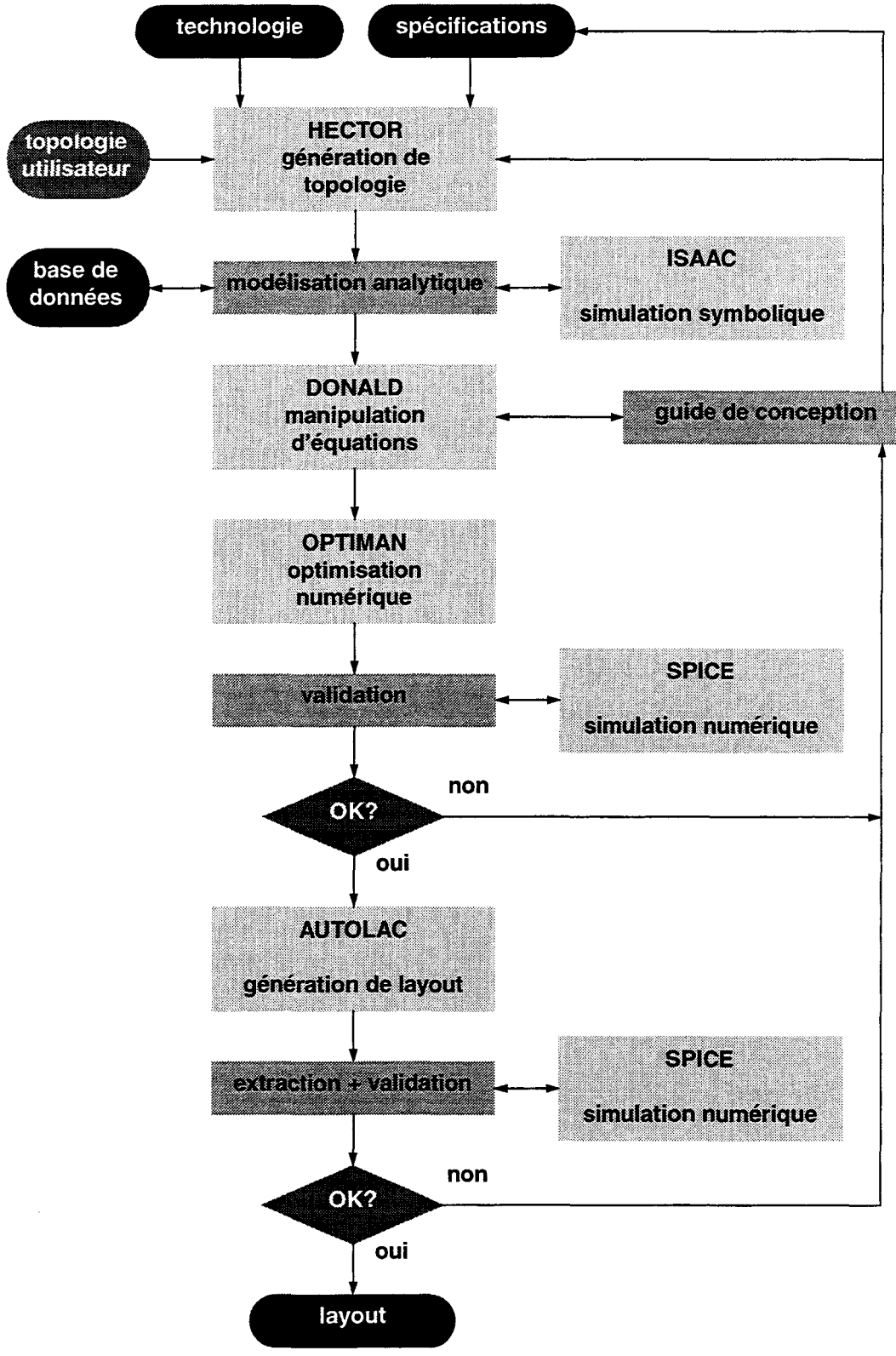


Figure 1.4 Méthodologie d'ASAIC

- *ASTRX/OBLX* [OCH96]

  Là où ASAIC résout le problème de systèmes à base de connaissances par l'utilisation de l'analyse symbolique et la manipulation des équations, ASTRX/ OBLX aborde les problèmes associés avec les systèmes à base d'optimisation, principalement limités par leur lenteur. L'objectif est d'évaluer les performances du circuit avec une vitesse qui est quelques ordres de grandeur plus élevée que par la simulation numérique, mais sans avoir recours aux équations de conception. Ceci peut être effectué en utilisant AWE (Asymptotic Waveform Evaluation), qui est une technique générale de simulation capable de prédire les performances petit-signal du circuit. Cette prédiction est possible grâce à l'utilisation d'un modèle de complexité réduite, qui est basé sur une (très) courte analyse numérique transitoire du circuit et sa transformation par la suite en un modèle équivalent petit-signal. Si le circuit est linéaire ou peut être linéarisée de façon précise, les résultats de l'analyse sont très proches de ceux d'une analyse numérique, mais sont obtenus en moins de temps et en évitant l'analyse manuelle.

  L'évaluation des performances du circuit est davantage accélérée par l'utilisation de la formulation statique approximative ("relaxed-dc formulation"), où les points de polarisation du circuit sont calculés par une solution implicite des lois de Kirchhoff. Là où les lois ne sont pas satisfaites, une erreur est générée, qui est par la suite incluse dans la fonction globale d'objectif de conception - évidemment, l'erreur d'exactitude statique doit être nulle à la fin de l'optimisation. Cette technique allège donc l'utilisation du CPU en évitant les solutions numériques des points de polarisation par l'utilisation d'algorithmes itératifs tels que Newton-Raphson.

  Un autre aspect, plus général, concerne l'évaluation individuelle des dispositifs, ce qui rend l'outil indépendant des aspects de modélisation de dispositif par des modèles autonomes ("encapsulated device models"), tout en gardant la précision de simulateurs numériques. De ce fait, les modèles implémentés peuvent être complets et non des simplifications, ce qui est souvent le cas dans l'évaluation des performances par des équations.

  Le système complet est montré en Figure 1.5.

  ASTRX génère le code décrivant la fonction objectif de conception, avec des liens aux modules d'évaluation de dispositif et à AWE. OBLX utilise ensuite le code généré afin d'évaluer les performances du circuit et une méthode d'optimisation globale (recuit simulé) pour dimensionner le circuit. Contrairement à ASAIC, une méthode d'optimisation locale ne peut pas être utilisée ici car aucune information sur le point de départ n'est disponible.

  ASTRX/OBLX est utilisé dans ACACIA, un environnement complet de synthèse qui incorpore la génération de layout. La synthèse post-layout comme dans ASAIC est donc également possible.

- *ISAID* [TOU95]

  En utilisant des équations analytiques de conception, une erreur est inévitablement introduite entre les performances prédites et celles réalisées. Le circuit synthétisé peut très bien satisfaire les spécifications, mais il n'en ira peut-être pas de même avec le circuit simulé. ISAID utilise des raisonnements qualitatifs afin d'améliorer la

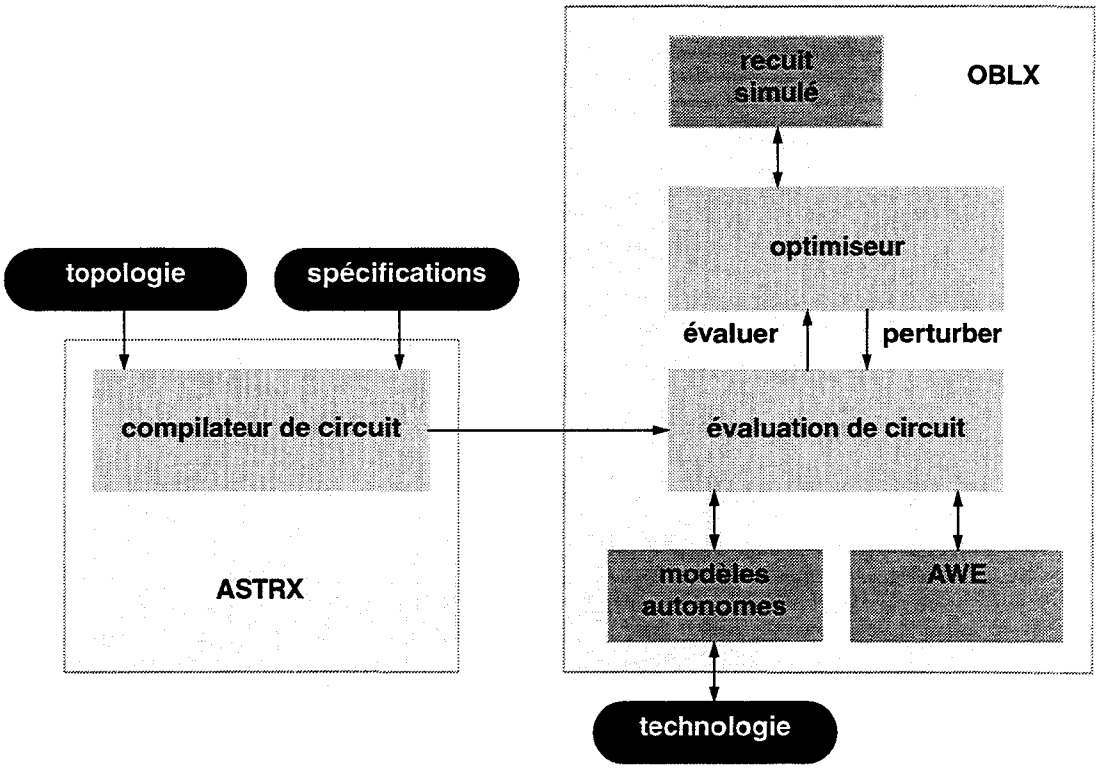


Figure 1.5 Architecture ASTRX/OBLX

précision de systèmes à base de connaissances, et n'utilise à aucun moment l'optimisation numérique; par contre, la simulation numérique est utilisée. La boucle de conception effectue très peu de cycles, et rend donc l'outil extrêmement rapide, même s'il reste plus lent que les outils purement à base de connaissances sans rebouclage.

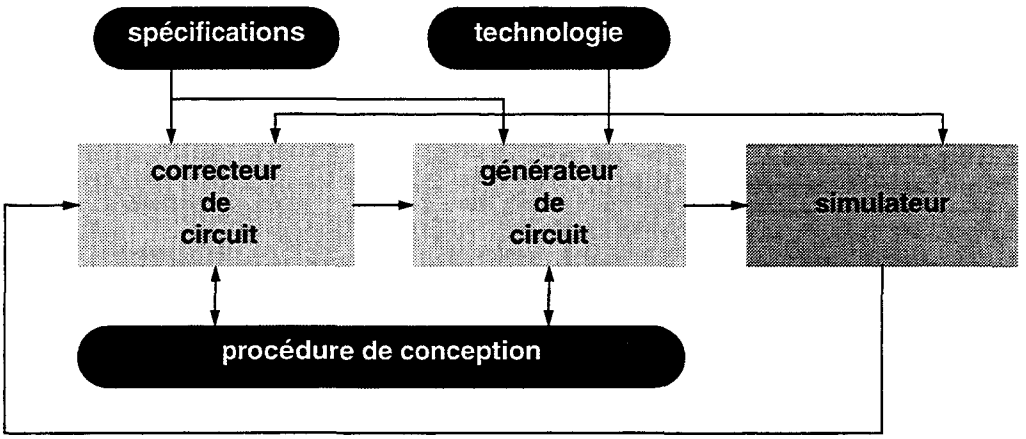


Figure 1.6 Architecture ISAID

Le système, comme illustré dans la Figure 1.6, comprend un générateur de circuit, qui utilise des connaissances d'expert afin de composer hiérarchiquement une topologie préliminaire de circuit, et un correcteur de circuit après la simulation, qui

déduit des ajustements possibles au niveau dispositif ou au niveau topologie afin d'améliorer les critères de performances non-tenues. Les deux blocs utilisent une procédure de conception, qui est constituée des connaissances d'expert requises pour la décomposition et synthèse par le générateur de circuit, et du contrôle du processus de conception requis par le correcteur de circuit.

ISAID implémente une décomposition hiérarchique rigoureuse, réduisant une topologie en "blocs de fonctions" ("task blocks") qui la constituent. Par exemple, un bloc réalisant la fonction de transistor peut être implémenté par un simple transistor, ou une configuration cascode ou cascode actif. Une fois arrivé au niveau dispositif, les spécifications décomposées correspondantes sont utilisées pour dimensionner et évaluer chaque dispositif. Cette procédure génère les dimensions des dispositif en se basant sur les conditions spécifiées de polarisation ou sur les paramètres intrinsèques aux dispositifs. Un dimensionnement initial basé sur un modèle explicite et simple est effectué, et l'ajustement fin se fait par un raffinement itératif basé sur un modèle plus précis.

Une fois de plus, cet outil est utilisé dans un cadre plus complexe (CHIPAIDE), qui génère un layout et effectue des analyses de tolérance sur les performances du circuit afin de générer un bloc de circuit intégré parfaitement fonctionnel.

1.2.3 Application aux circuits à temps-discret

Des outils de CA pour les circuits échantillonnés analogiques (capacités commutées ou courants commutés) existent, quoique l'activité dans ce domaine soit moins marquée que dans le domaine continu. En général, de tels outils implémentent une approche descendante ("top-down"), c'est-à-dire que le système est défini et les blocs le composant seront dimensionnés par la suite. La hiérarchie constitue donc une partie importante de la synthèse en temps-discret et, contrairement à la situation dans le domaine continu, est moins controversée. Ceci est dû au nombre limité de topologies disponibles pour réaliser une fonction donnée, et donc le recouvrement des niveaux hiérarchiques n'existe pas.

D'un autre côté, les approches descendantes ont une limitation sérieuse, car la liste des applications cibles ne peut pas être exhaustive, et donc les outils de CA qui utilisent l'approche descendante sont toujours orientés vers une application donnée. Dans les systèmes échantillonnés, les filtres constituent l'orientation principale. De tels outils ne peuvent donc pas être utilisés pour concevoir des convertisseurs de données, des lignes à retard ou d'autres circuits.

Telle est la philosophie implémentée par l'outil de CA de filtres à courants commutés SCADS [HUG96], qui commence avec la saisie des spécifications du système par l'utilisateur et se termine par la génération du layout d'un bloc réalisant la fonction. La chaîne de conception est illustrée dans la Figure 1.7. Une architecture est d'abord déterminée par l'utilisation de techniques explicites de synthèse de filtres. Ceci permet la génération d'un schéma bloc du système, où chaque bloc contient un modèle comportemental petit-signal. Le filtre est ensuite simulé à l'aide d'un programme d'analyse spécifique au temps discret, et les performances du circuit au niveau bruit

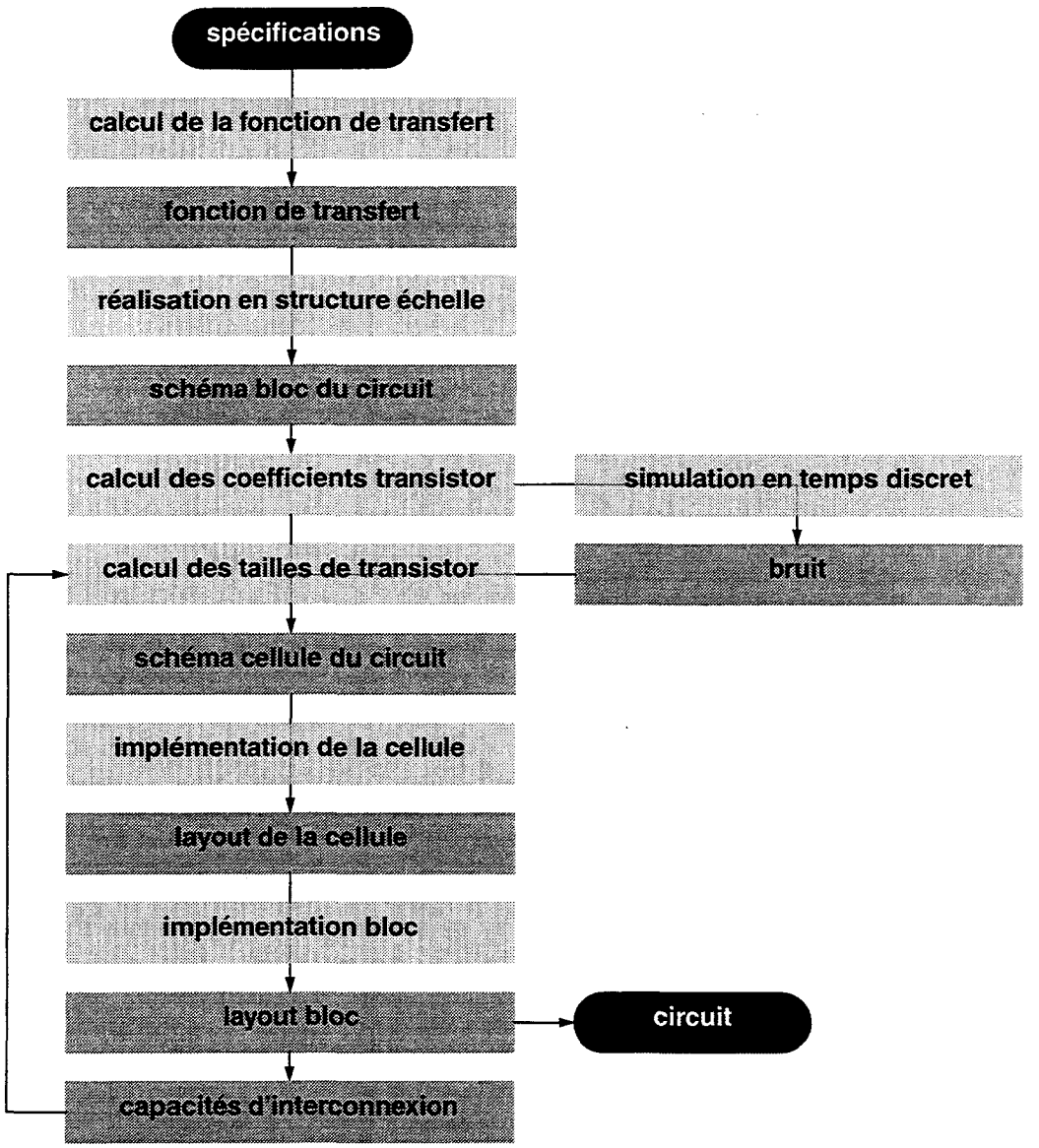


Figure 1.7 Chaîne de conception SCADS

peuvent alors être utilisées afin de décomposer la spécification finale du bruit en plusieurs spécifications pour les cellules qui constituent le système. A partir du bruit total et des modèles petit-signal, suffisamment d'information est disponible pour trouver les dimensions de chaque transistor satisfaisant les valeurs calculées de la transconductance et des capacités parasites pour un courant de polarisation donné. Afin de conserver l'indépendance par rapport aux modèles de dispositifs, les valeurs sont trouvées par convergence, en utilisant la simulation numérique à chaque itération. Puisque le simulateur n'est invoqué que sur un seul transistor en régime statique, l'évaluation est rapide.

SCADS fonctionne donc à plus d'un niveau hiérarchique, et de ce fait plusieurs degrés de liberté peuvent être mis en oeuvre par rapport à l'approche choisie. Les deux actions

de synthèse, au niveau du filtre et au niveau de la cellule, sont des procédures explicites basées sur les connaissances. Au niveau cellule, des problèmes peuvent résulter du manque de flexibilité de cette approche. Une seule cellule, de haute performance, constitue la base de données des topologies. Ceci a l'avantage de simplifier la chaîne de conception. En revanche, cette approche peut s'avérer restrictive dans le cas de certaines applications, ou mener à la conception d'un circuit trop coûteux dans le cas d'applications de basses performances. L'autre inconvénient majeur est que l'outil, par son approche descendante, est limité à la conception de filtres.

1.2.4 Les conclusions pour un outil de conception automatisée

En général, les concepteurs en analogique ne sont pas prêts à utiliser les outils automatisés à grande échelle. La conception analogique se caractérise par ses méthodes complexes et intuitives, et par sa sensibilité aux paramètres externes. Ces aspects ne sont pas aussi critiques dans la conception numérique du fait de sa nature structurée, formalisée et binaire. Il existe maintenant un consensus reconnu par la communauté de CA analogique sur le fait que, dans sa forme conçue dans les années 1980, cette dernière ne satisfait pas les besoins des concepteurs en analogique. Ces derniers doivent conserver un contrôle rigoureux sur le processus de conception, et ce besoin n'est en général pas respecté par la grande majorité des outils de synthèse. Parmi tous les outils de CA analogique introduits dans les environnements de conception, seuls les outils de bas niveau sont utilisés afin d'augmenter la productivité de conception. Il apparaît alors que c'est la direction que doivent prendre les outils de CA analogique : être des outils de bas niveau qui dimensionnent des cellules de complexité réduite, qui sont facilement configurables et qui ne nécessitent qu'un minimum de formation pour comprendre le processus de conception de l'outil. A long terme, ces outils de bas niveau peuvent être intégrés dans un outil cohérent de synthèse de haut-niveau à travers des processus conçus pour contrôler les outils de bas-niveau, mais cet objectif ne peut être réalisé qu'une fois les outils de bas niveau répandus de façon significative.

Ceci ne signifie pas que les outils de synthèse de haut niveau n'ont aucune place dans les philosophies actuelles de conception ; avec l'augmentation de la complexité des systèmes, leur rôle devient même critique pendant la phase d'exploration de conception, ou des compromis clés peuvent être faits une fois qu'une compréhension globale de l'espace de conception a été atteinte. Les langages de description comportementale analogiques et mixtes sont essentiels au succès de tels outils car aucune connaissance antérieure de l'implémentation du circuit n'est nécessaire, mais uniquement la fonction avec ses spécifications limitatives. Pour les grands systèmes, cette approche est aussi la seule utilisable pour valider le système par la simulation.

Une grande partie des travaux dans la CA analogique s'est concentrée sur le domaine continu, et nombreuses sont les techniques qui ne peuvent pas trouver d'implémentation en circuits à temps-discret. La première différence concerne l'évaluation des performances. La plupart des spécifications pour un amplificateur opérationnel, par exemple, peuvent être obtenues par une analyse fréquentielle rapide (produit gain-bande, marge de phase, taux de rejection des alimentations et du mode

commun, etc.) et celles qui nécessitent une analyse grand-signal ou transitoire (slew rate), qui est typiquement une analyse plus longue du fait de la nature non-linéaire de la matrice du circuit, sont peu nombreuses. Pour un circuit échantillonné tel que la cellule de mémoire de courant, la seule quantité significative est l'erreur d'échantillonnage et sa contribution à l'offset, à l'erreur de gain et à la distorsion harmonique. Ce spectre de l'erreur d'échantillonnage peut seulement être évalué de façon précise par une simulation transitoire de longue durée suivie d'une analyse de Fourier. Nous pouvons donc conclure que l'utilisation de la simulation numérique dans la boucle interne d'optimisation, déjà marginalisée pour les circuits continus, est inutilisable pour les circuits à temps discret. Cependant, pour des questions de précision, la possibilité de la simulation numérique dans la boucle externe peut être évoquée.

Tout en admettant que des modèles analytiques sont nécessaires afin d'accélérer la boucle de conception, des difficultés apparaissent. Premièrement, les expressions analytiques qui donnent explicitement les résultats d'une analyse de Fourier sont difficiles à générer. L'erreur d'échantillonnage est un amalgame de plusieurs composantes d'erreurs, qui peuvent être calculées individuellement. Mais leur effet cumulatif sur le spectre de puissance peut seulement être estimé [FIE90]. Donc, les erreurs individuelles peuvent être spécifiées, en supposant (d'un point de vue système) que, de ces spécifications, résultera le spectre requis pour le signal échantillonné. Le deuxième problème est que la génération automatique de ces modèles analytiques (comme dans ASAIC) n'est pas possible avec les outils existants d'analyse symbolique. Les modèles analytiques doivent donc être générés manuellement. Bien que ce soit une tâche ardue, l'utilisateur doit avoir accès à cette activité, et par conséquent la base de connaissances doit rester ouverte.

Une caractéristique attrayante de la plupart des outils modernes de CA analogique est l'indépendance par rapport aux modèles du dispositif élémentaire (transistor). Etant donné la vitesse d'évolution des technologies, il est évident qu'une restriction à un seul modèle de dispositif n'est plus viable, surtout lorsque la taille de ces derniers diminue bien en-deçà de l'échelle submicronique. Donc, pour la maintenance, les modèles analytiques doivent être écrits en termes de paramètres intrinsèques aux dispositifs, et non pas en termes de paramètres technologiques ou de modèle.

Etant donné ces considérations, nous pouvons dresser une liste de spécifications pour un outil de CA :

- conception automatisée au niveau cellulaire,
- évaluation analytique du circuit dans la boucle interne d'optimisation,
- simulation numérique du circuit dans la boucle externe d'optimisation,
- une base de connaissances ouverte,
- indépendance des modèles du dispositif et de la technologie pour assurer la pérennité de l'outil.

1.2.5 L'architecture de l'outil de conception automatisée

L'architecture haut-niveau de la plupart des outils de CA adhère à une philosophie commune, qui peut s'appliquer à une structure capable d'incorporer les points énoncés ci-dessus, et peut être formalisée comme illustré dans la Figure 1.8. L'utilisateur fournit

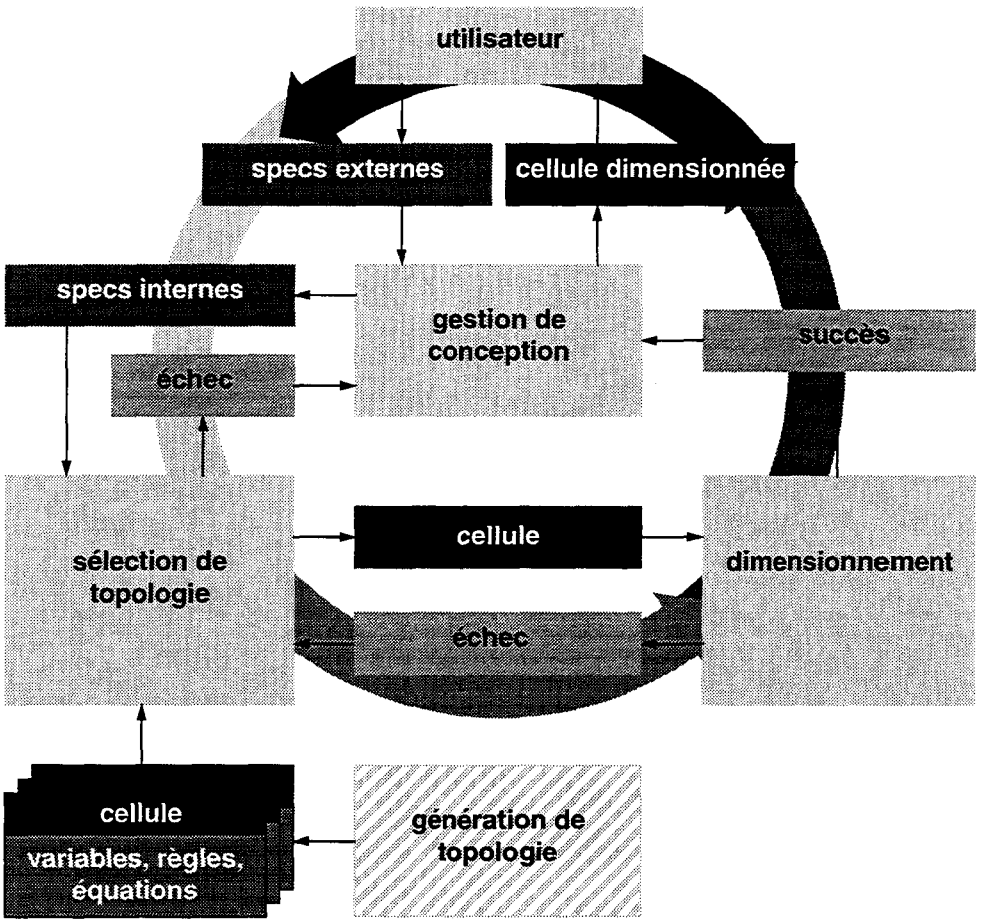


Figure 1.8 Architecture système

au bloc de gestion de conception le cahier des charges du circuit, établi en termes de contraintes (ainsi que leurs priorités relatives), de coûts et de conditions de fonctionnement. Ce cahier des charges initial est formalisé et adapté au bloc de sélection de topologie. Ce bloc consulte la bibliothèque de topologies disponibles, sélectionne et ordonne celles susceptibles de satisfaire les spécifications. Les équations analytiques, les règles heuristiques et les paramètres qui correspondent à la topologie sélectionnée sont transférés au module de dimensionnement, qui représente la partie principale du système. C'est ici que le processus d'optimisation s'effectue. Il consiste en :

- une procédure explicite analytique pour générer le circuit préliminaire,
- une optimisation à base de règles utilisant des équations analytiques afin d'ajuster de façon fine le circuit, limité par la précision des équations de conception (Figure 1.9),

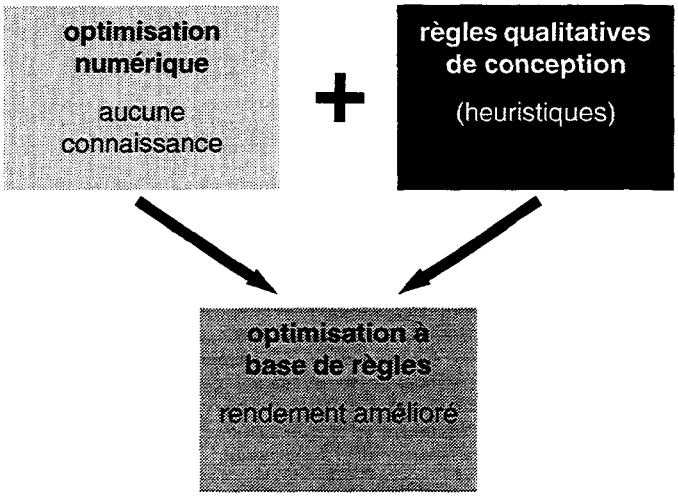


Figure 1.9 Construction de l'optimisation à base de règles

- une estimation de l'erreur de l'équation par l'utilisation de l'analyse numérique du circuit, et un rebouclage vers la deuxième étape.

Le processus est illustré dans la Figure 1.10.

Si le module de dimensionnement converge vers un résultat satisfaisant à toutes les contraintes, ce résultat est transféré vers le bloc de gestion de conception, puis vers l'utilisateur. Si, par contre, aucun résultat satisfaisant n'est trouvé, la topologie est renvoyée vers le sélectionneur de topologie, qui en choisit une autre dans la liste. Si toutes les topologies ont été essayées, et si aucune n'a pu satisfaire aux spécifications, le contrôle est retourné au bloc de gestion de conception, qui peut alors choisir de relâcher une ou plusieurs contraintes. Par exemple, à la place de contraindre un critère de performance p_1 à être plus grand qu'une valeur quelconque, p_1 est transformé en coût et doit être maximisé aussi près que possible de la valeur spécifiée.

Nous pouvons donc constater que cette architecture représente le cadre du processus de CA et qu'une grande partie de l'information nécessaire pour le dimensionnement appartient à la topologie. La majorité de cette information consiste en des équations analytiques nécessaires à l'évaluation de la topologie. Dans le chapitre suivant, nous considérerons la représentation de cette information dans une forme qui offre le maximum de bénéfice à l'utilisateur et au système de conception.

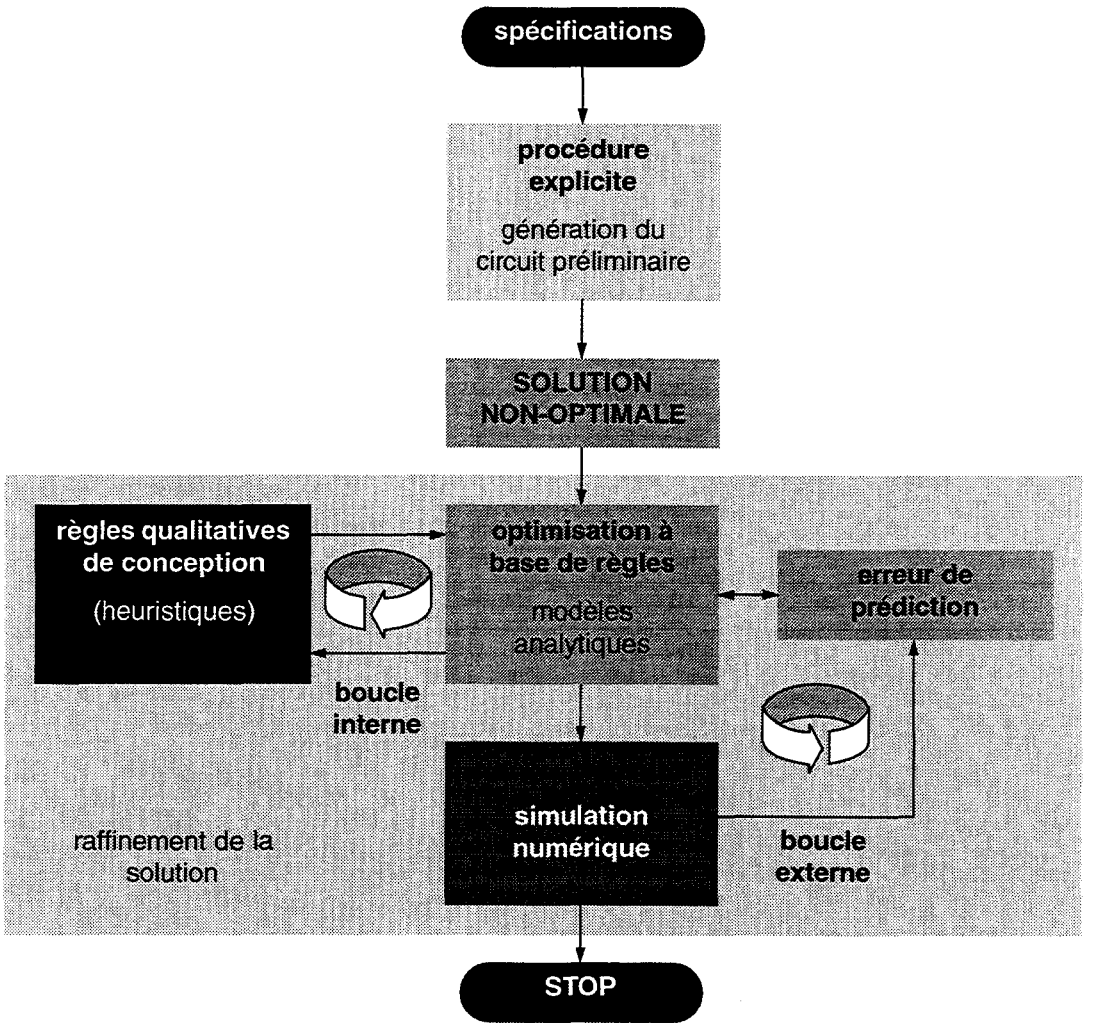


Figure 1.10 Prototype pour l'outil de CA des courants commutés

1.3 Conclusion

La CA analogique est pratiquement inexistante dans le monde industriel, alors que son équivalent numérique est fortement répandu. La nature même de la conception analogique en est responsable, et rend les techniques de CA numérique soit impossibles soit inefficaces pour une utilisation dans ce contexte. Les approches traditionnelles ont été à base d'optimisation (processus lent), ou à base des connaissances qui nécessitent des connaissances formelles de conception.

Plus récemment, des approches ont tenté de tirer avantage de ces deux techniques afin de créer les outils de CA correspondant à l'état de l'art. Cependant, leur domaine d'application reste pour la plupart confiné aux circuits continus. En identifiant et rassemblant les points clés des divers outils, une stratégie cohérente de CA a été développée et doit permettre la conception efficace et précise de cellules à courants commutés.

Références

[ANA94] Anacad Electrical Engineering Software, "ELDO User's Manual", 1994

[BER88] E. Berckan, M. Abreu, W. Laughton, "Analog compilation based on successive decompositions", *Proc. ACM/IEEE Design Automation Conference*, pp. 369-375, 1988

[BOW85] R.J. Bowman, D.J. Lane, "A Knowledge-Based System for Analog Integrated Circuit Design", *Proc. IEEE International Conference on Computer Aided Design*, pp. 210-212, 1985

[DEG87] M.G.R. Degrauwe *et al.*, "IDAC: An Interactive Design Tool for Analog CMOS Circuits", *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 6, pp. 1106-1116, December 1987

[ELT89] F. El-Turky, E.E. Perry, "BLADES: An Artificial Intelligence Approach to Analog Circuit Design", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 6, pp. 680-692, June 1989

[FIE90] T.S. Fiez, D.J. Allstot, "CMOS Switched-Current Ladder Filters", *IEEE Journal of Solid-State Circuits*, vol. 25, no. 6, pp. 1360-1367, December 1990

[GIE89] G. Gielen, H. Walscharts, W. Sansen, "ISAAC: A Symbolic Simulator for Analog Integrated Circuits", *IEEE Journal of Solid-State Circuits*, vol. 24, no. 6, pp. 1587-1597, December 1989

[GIE91] G. Gielen, W. Sansen, "Symbolic Analysis for Automated Design of Analog Integrated Circuits", *Kluwer Academic Publishers*, 1991

[HAR89] R. Harjani, R.A. Rutenbar, L.R. Carley, "OASYS: A Framework for Analog Circuit Synthesis", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 12, pp. 1247-1266, December 1989

[HUG96] J.B. Hughes *et al.*, "Automated Design of Switched-Current Filters", *IEEE Journal of Solid State Circuits*, vol. 31, no. 7, pp. 898-907, July 1996

[KOH90] H.Y. Koh, C.H. Séquin, P.R. Gray, "OPASYN: A Compiler for CMOS Operational Amplifiers", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 2, pp. 113-125, February 1990

[KUN95] K.S. Kundert, "The Designer's Guide to SPICE and Spectre", *Kluwer Academic Publishers*, 1995

[LAI88] J.C. Lai *et al.*, "ADOPT - A CAD System for Analog Circuit Design", *Proc. IEEE Custom Integrated Circuits Conference*, pp. 3.2.1-3.2.4, 1988

[LIP89] R. Lipsett, C.F. Schaefer, C. Ussery, "VHDL: Hardware Description and Design", *Kluwer Academic Publishers*, 1989

[NYE83] B. Nye *et al.*, "DELIGHT.SPICE: An Optimization-Based System for the Design of Integrated Circuits", *Proc. IEEE Custom Integrated Circuits Conference*, pp. 233-238, 1983

[OCH96] E.S. Ochotta, R.A. Rutenbar, L.R. Carley, "Synthesis of High-Performance Analog Circuits in ASTRX/OBLX", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 3, pp. 273-294, March 1996

[OLI97] O. Oliaei, H. Aboushady, P. Loumeau, "Simulation de modulateurs ΣΔ à courants commutés", *Proc. Colloque CAO de circuits intégrés et systèmes*, pp. 283-286, January 1997

[ONO89] H. Onodera, H. Kanbara, K. Tamaru, "Operational amplifier compilation with performance optimization", *IEEE Journal of Solid-State Circuits*, pp. 466-473, April 1990

[PRA92] H.J. Pranger, "ANANAS: A Program for Analog Circuit Analysis and Synthesis", Thesis, Universiteit Twente, 1992

[SHY88] J.M. Shyu, A. Sangiovanni-Vincentelli, "ECSTASY: A New Environment for IC Design Optimization", *Proc. IEEE International Conference on Computer Aided Design*, pp. 484-487, 1988

[THO90] D.E. Thomas, P.R. Moorby, "The Verilog Hardware Description Language", *Kluwer Academic Publishers*, 1990

[TOU95] C. Toumazou, C.A. Makris, "Analog IC Design Automation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 2, pp. 218-254, February 1995

[VLA94] A. Vladimerescu, "The SPICE Book", *John Wiley & Sons*, 1994

2 LA MODÉLISATION ANALYTIQUE DE LA CELLULE MÉMOIRE DE COURANT DE BASE

Les modèles analytiques permettent une évaluation rapide des performances d'un circuit, et par conséquent un cycle de conception court. Dans ce chapitre, nous présentons le cadre de l'implémentation des modèles analytiques, suivi par l'analyse physique de la cellule de base. Ceci définit le contenu du modèle. Ce dernier peut être utilisé pour développer des règles qualitatives de conception, qui constituent les connaissances de conception essentielles.

Avant de détailler l'analyse utilisée dans le développement du modèle de la cellule de mémoire de courant de base [DAU88], il est primordial de dresser un aperçu du fonctionnement du circuit (Figure 2.1).

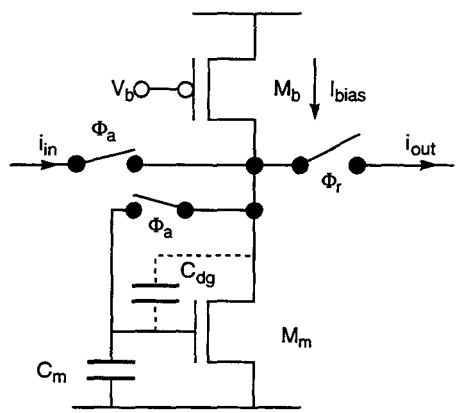


Figure 2.1 Cellule de mémoire de courant de base

En mode acquisition (Φ_a=1), le transistor mémoire M_m est en configuration diode et sa tension de grille tend vers une valeur qui correspond à celle nécessaire pour que le courant de drain soit égal à la somme du courant de polarisation I_b et le courant de signal i_{in}. En fin d'acquisition, l'interrupteur drain-grille s'ouvre (Φ_a=0) et la tension de grille acquise est mémorisée sur la capacité de grille C_m, à présent isolée. En mode restitution, l'interrupteur de sortie est fermé (Φ_r=1), et puisque le courant dans le transistor mémoire est imposé par sa tension de grille, le courant de sortie i_{out} équivaut à l'inverse du courant d'entrée acquis.

Les modèles analytiques à développer nécessitent un cadre dans lequel ils peuvent être implémentés. De nombreuses possibilités existent à cette fin. Le choix de l'approche aura de profonds effets sur la rapidité, la précision et la maintenance de l'outil final de CA, pour toutes les topologies ainsi modélisées. L'importance de ce choix signifie qu'il doit être pris en connaissant tous les avantages et limitations de chacune des approches, aussi bien en termes de langage de modélisation qu'en termes de détail des modèles analytiques, car une plus grande précision implique un temps d'évaluation plus long. Avant de considérer la modélisation des effets physiques, une stratégie cohérente de modélisation doit d'abord être élaborée. C'est l'objectif de la section suivante.

2.1 La modélisation analytique de topologies

En effectuant des simplifications raisonnables dans le développement de modèles analytiques du circuit, une analyse précise du circuit peut être effectuée en une fraction du temps nécessaire à un simulateur numérique. Ceci accélère la boucle de conception et permet au concepteur ou à l'optimiseur de faire des choix topologiques ou architecturaux plus efficaces et de faire des analyses de rendement. Dans le cadre de la CA, deux aspects de la modélisation analytique sont importants : l'implémentation des modèles, c'est-à-dire de quelle façon ils doivent être formalisés, et leur contenu.

2.1.1 L'implémentation des modèles analytiques

Les circuits analogiques ou mixtes peuvent être modélisés de multiples façons. Le choix éventuel dépend fortement des moyens disponibles pour évaluer les modèles, en termes de rapidité et de flexibilité. Les trois méthodes principales sont :

- la macromodélisation,
- la modélisation par un langage de description comportementale,
- la modélisation par un langage de programmation haut-niveau.

Les premières tentatives utilisaient des macromodèles schématiques contenant des sources idéales et des interrupteurs [BOY74], et simulaient avec l'appui de méthodes d'analyse numérique. L'avantage principal de cette approche est qu'un simulateur classique peut être utilisé, facilitant donc l'interface vers les autres blocs du système. Cependant, le nombre de noeuds dans le macromodèle, et donc dans la matrice du circuit, n'est pas réduit de façon significative. La complexité du macromodèle devient rapidement prohibitive lorsque des composants sont ajoutés afin d'augmenter la validité du modèle. De plus, les éléments idéaux utilisés dans le circuit peuvent engendrer des problèmes de convergence dans les algorithmes de simulation, qui ne sont pas toujours capable de traiter des changements brusques de valeur, ou des constantes de temps très petite.

Récemment, des innovations dans le domaine de la modélisation comportementale [IEE97] ont étendu de tels langages pour inclure des composants analogiques. Dans le cas de systèmes à temps discret, tels que les circuits à capacités commutées ou à

courants commutés, il existe une dimension supplémentaire : la simulation efficace de l'interface analogique/numérique. Jusqu'alors, les systèmes mixtes étaient simulés soit séparément avec succès incertain, ou avec un simulateur "collé" dont le point critique est la synchronisation des domaines. Avec les langages de modélisation comportementale (HDL), les blocs mixtes peuvent être décrits et simulés en tant qu'entité. De plus, un circuit modélisé par un HDL analogique obéit par construction aux lois électriques, ce qui n'est pas le cas de la modélisation avec un langage de programmation non-spécifique. Toutefois, il est plus facile d'implémenter une approche descendante en modélisant les performances par des équations purement abstraites (contenant des paramètres de haut niveau), que d'implémenter l'approche inverse, ascendante, où les performances sont fonction des dimensions des composants et des non-idéalités. Comme le jeu d'opérateurs mathématiques dans les HDLs analogiques est relativement restreint à l'heure actuelle, la seule solution est d'utiliser un langage externe de programmation tel que le C [KER88] ou le C++ [LIP91] afin de représenter la fonction du circuit avec une précision suffisante (Figure 2.2).

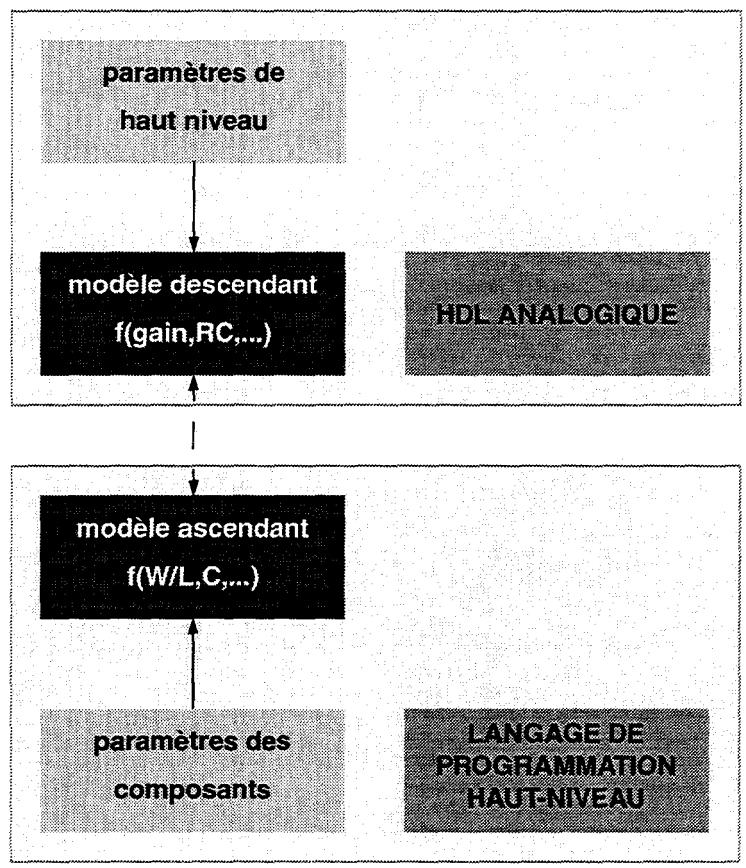


Figure 2.2 Comparaison entre approches de modélisation comportementale

Les langages de programmation de haut niveau ont déjà été utilisés dans la modélisation de grande précision de circuits. De tels langages ont en général un jeu développé d'opérateurs mathématiques, ainsi que des fonctions mathématiques de bas niveau qui sont facilement disponibles et intégrés dans les logiciels spécifiques. L'approche de la

modélisation par un langage de programmation est donc très flexible, et le degré de précision requis par le programmateur peut atteindre la précision machine. Il est également plus facile d'intégrer une telle approche dans un outil de CA. L'inconvénient principal de cette technique est qu'elle est sujette à erreur : la plupart des langages ne vérifient pas l'homogénéité des unités dans les équations, un aspect fondamental de la vérification des modèles. En effet, un modèle écrit avec un langage de programmation de haut niveau peut ne pas obéir aux lois physiques s'il n'est pas explicitement codé à cet effet : ceci peut être considéré à la fois comme une liberté et comme une contrainte. Un autre problème concerne la portabilité des outils : de nombreux compilateurs spécifiques aux différentes machines peuvent exister pour le même langage, et même là où des normes internationales existent, des différences subtiles peuvent exister dans l'interprétation de code qui peuvent engendrer des erreurs à l'exécution.

2.1.2 Le contenu des modèles analytiques

Il est également nécessaire de s'interroger sur la complexité des modèles des circuits qui sont codés. Les modèles analytiques sont de nature imprécise et l'erreur d'approximation est inversement proportionnelle au temps CPU nécessaire pour les évaluer. Un compromis doit donc être trouvé, pour qu'une erreur minimale soit atteinte pour un temps de calcul raisonnable. Un autre aspect est celui du nombre d'évaluations requis par le processus itératif de conception. La réduction du nombre d'itérations de l'algorithme d'optimisation est un problème qui sera traité ultérieurement, avec le choix entre les algorithmes de recherche locale et globale. Mais au niveau de la modélisation, le nombre d'itérations peut également être réduit en générant un point de départ de qualité pour les algorithmes d'optimisation locale. Pour ce faire, il existe un certain nombre de manières :

- *point fixe* : des valeurs fixes sont attribuées aux dimensions de la cellule, quelles que soient les spécifications. Les valeurs doivent être faisables sur une échelle importante de conditions de fonctionnement et donc définissent une cellule globalement correcte. Cette méthode n'est pas fiable car les spécifications pratiques peuvent varier de façon importante, et un mauvais point de départ peut mener à la convergence locale ou même à un échec. La détermination de valeurs fixes n'est pas non plus une tâche aisée, car il est nécessaire de prendre en compte les variations technologiques, les spécifications probables, etc.

- *aléatoire* : des valeurs aléatoires sont prises pour chaque dimension dans l'échelle attribuée. Cette méthode élimine le travail requis pour dériver les valeurs fixes, mais les problèmes associés avec la convergence restent et sont même aggravés car les variations technologiques ne sont pas prises en comptes.

- *analytique* : l'utilisation d'équations avec un modèle de premier ordre afin de générer explicitement le point de départ a de nombreux avantages. D'abord, il est plus probable que l'optimisation numérique converge rapidement vers une solution globale. De plus, il est possible de déterminer *a priori* que la topologie convient à un jeu de spécifications, plutôt qu'*a posteriori*. Cette information peut être utilisée par le sélectionneur de topologie lors de la génération d'une liste de topologies faisables.

Deux niveaux de modélisation comportementale sont donc nécessaires afin de générer le processus de conception le plus efficace : la modélisation détaillée avec des modèles industriels de dispositifs pour l'évaluation des performances, et un modèle simple qui utilise des équations explicites MOS de premier ordre pour générer un point de départ qui a une forte probabilité d'être à proximité du minimum global (Figure 2.3).

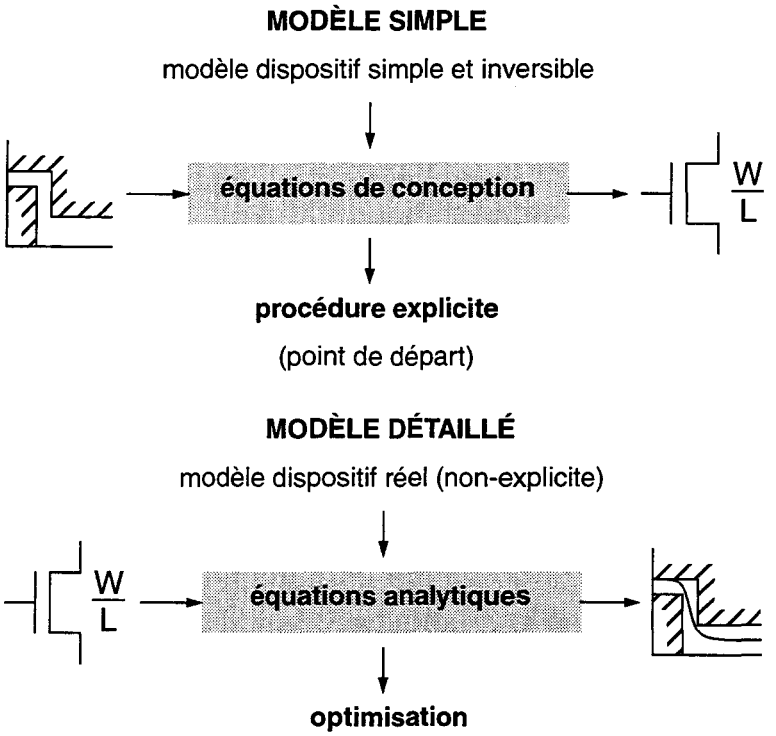


Figure 2.3 Niveaux de modélisation comportementale pour la conception automatisée

Les critères de performance pour les cellules à courants commutés peuvent être définis en termes de sources individuelles d'erreur d'échantillonnage. Ceux-ci sont d'une utilité limitée pour un concepteur travaillant au niveau du système, pour lequel les spécifications sont habituellement établies en termes d'offset, d'erreur de gain et de distorsion harmonique. Une difficulté majeure de la conception de circuits à courants commutés réside donc dans le fait qu'aucun lien explicite ne peut être établi entre les sources individuelles d'erreur et leur effet sur le spectre du signal échantillonné. Ceci est principalement dû à deux raisons, qui sont d'une part le bas niveau hiérarchique de la cellule (et donc sa forte dépendance envers les caractéristiques du dispositif), et d'autre part le fort couplage qui existe entre des cellules interconnectées (plusieurs erreurs d'échantillonnage proviennent de la perturbation causée par une cellule en acquisition sur la cellule en restitution qui lui fournit son courant d'entrée ; aucune abstraction ne peut être faite au niveau de la cellule). Par exemple, l'erreur de gain statique (section 2.5) est une erreur de gain purement linéaire uniquement si les transistors restent en zone saturée sur toute la dynamique du signal d'entrée et si la conductance de sortie est constante pour cette même dynamique (Figure 2.4). Dans le cas où le point de fonctionnement du transistor se déplace légèrement vers le "coude" de la caractéristique, la conductance de sortie n'est plus de valeur constante et des erreurs non

linéaires sont introduites, engendrant de ce fait de la distorsion harmonique. La valeur de la conductance de sortie dépend également des caractéristiques du dispositif et du modèle utilisé et n'est jamais constante en pratique.

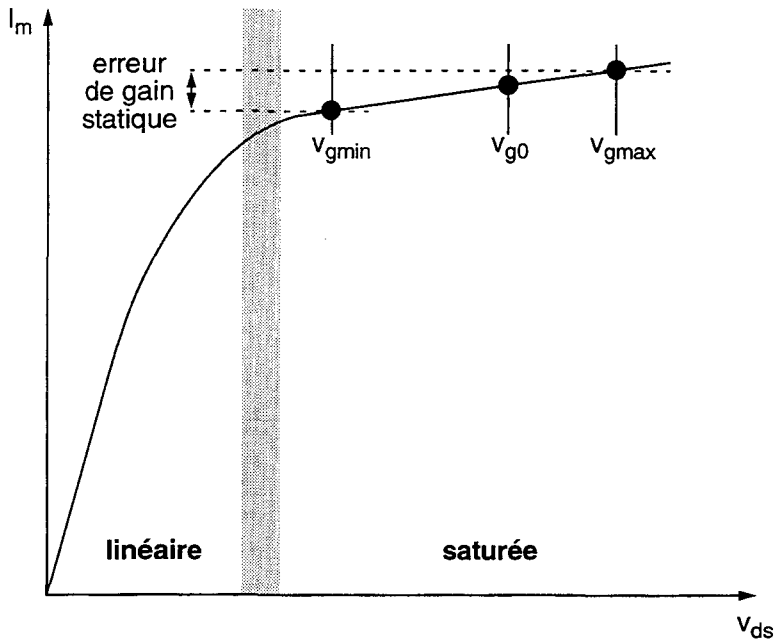


Figure 2.4 Dynamique de la tension de grille pour fonctionnement en régime saturé

L'analyse symbolique a été utilisée avec succès dans le développement de fonctions de transfert pour des circuits à capacités commutées. La fonction de transfert de l'élément à courants commutés de base est :

$$H(z) = -(1-a)z^{-n} \tag{2.1}$$

où a représente l'erreur d'échantillonnage et n le nombre de périodes d'horloge entre l'acquisition et la restitution. Cependant, les hypothèses de base d'une analyse utilisant la transformée en z sont :

- tous les interrupteurs sont idéaux ;
- chaque élément est indépendant de son environnement ;
- aucun effet grand-signal n'est présent ;
- a est constant.

Aucune de ces hypothèses n'est malheureusement valable pour l'analyse de la cellule mémoire de courant, et il en résulte que l'analyse symbolique de ce type d'élément par la transformée en z n'est pas possible.

L'ajustement numérique est utilisable lors de la modélisation numérique, afin d'augmenter la précision de façon empirique. Cependant, l'amélioration de la validité

d'un modèle pour un jeu de valeurs considérées tend à réduire l'étendue de son domaine d'application. Des modèles de dispositif fortement empiriques tels que BSIM1 et BSIM2 [MET96] nécessitent plusieurs jeux de paramètres pour la même technologie en fonction des zones d'utilisation. Ces zones sont habituellement définies par les dimensions des dispositifs considérés, mais peuvent l'être également par le régime de fonctionnement de ces derniers . Nous considérons donc que cette approche n'est pas scientifiquement valable, et limite l'utilisation du modèle. Nous préférons améliorer l'analyse physique du modèle et bien que celle-ci ne puisse pas toujours refléter parfaitement la réalité, elle reste pour le moins reproductible.

Les modèles analytiques sont fondés sur les modèles de dispositifs. La simplicité même de la cellule de courants commutés signifie également qu'elle est proche du niveau dispositif, et son comportement modélisé est par conséquent très dépendant du modèle de dispositif utilisé. Cet aspect mis à part, il y a également la question de son implémentation. Etant donné les importants systèmes d'équations associés avec les modèles industriels de dispositifs typiques, et la nécessité de l'indépendance des modèles de topologie par rapport aux modèles de dispositif, une structure de code doit être élaborée afin de satisfaire ces besoins. Ces aspects sont traités dans la section suivante.

2.2 Aspects de la modélisation de dispositifs

Les modèles comportementaux, selon leur niveau d'abstraction, peuvent à un moment donné de leurs calculs requérir une indication sur les valeurs des paramètres intrinsèques des dispositifs faisant partie du circuit ou du système qu'ils représentent. Pour la cellule de mémoire de courant, c'est particulièrement vrai car la cellule est un circuit de bas niveau et les dispositifs hiérarchiquement inférieurs sont des transistors MOS.

2.2.1 Généralités sur les besoins en modèles

Le besoin en modèles MOS de qualité devient de plus en plus marqué lorsque la taille du transistor diminue et que les effets de second ordre, jusqu'alors ignorés, deviennent de plus en plus importants, voire dominants [TSI94]. Jusqu'au milieu des années 1990, la tendance était d'introduire les effets parasites de manière empirique dans les modèles. Cela a été effectué en introduisant des équations mathématiques complexes, qui permettaient de coller aux caractéristiques mesurées (ex. BSIM2), mais qui perdaient leurs qualités de prédiction en même temps que l'aspect physique dans les équations. Nous sommes en train d'assister à un retour vers les modèles basés sur la physique (ex. BSIM3v3 [CHE96], Philips Model 9 [VEL95]) qui permettent plus de prédictions et restent de ce fait valides sur une plus grande plage de variation des dimensions du transistor.

Un problème fondamental dans les modèles MOS fournis par les fondeurs est qu'ils sont souvent optimisés pour la simulation de circuits numériques. Par exemple, un

modèle de transistor adapté à l'utilisation dans un circuit numérique prédira avec précision la réduction de mobilité des dispositifs à canaux courts, effet qui intervient pour une tension grille-source élevée. Si cette situation est typique pour les transistors des circuits numériques, elle est pourtant rare en analogique. En effet, les besoins de modélisation pour les circuits numériques sont très différents de ceux pour les circuits analogiques : un transistor dans un circuit numérique est soit bloqué ($V_{gs}<<V_t$) ou dans la zone de conduction linéaire ($V_{gs}>>V_t$; $V_{ds}\sim0$) et se simule en transitoire ; alors que le fonctionnement d'un transistor dans un circuit analogique peut être critique en inversion faible ($V_{gs}<V_t$) ou forte ($V_{gs}>V_t$), dans la zone linéaire ($V_{ds}<V_{dsat}$) ou saturée ($V_{ds}>V_{dsat}$), ou à la limite entre les deux, et se simule dans les domaines statique, fréquentiel et transitoire. Bien que la simulation transitoire soit la plus complexe, les paramètres petit-signal utilisés dans la simulation fréquentielle, calculés par une étude statique, sont essentiels à la conception analogique.

Le modèle MOS pour des applications numériques doit donc être précis pour un jeu réduit de conditions, et rapide afin de pouvoir traiter en un temps raisonnable le grand nombre de transistors typique aux circuits numériques ; alors que son équivalent pour les applications analogiques doit être précis pour de grandes variations de conditions de polarisation et plus particulièrement aux transitions entre les zones de fonctionnement[1]. Souvent, la précision globale sera sacrifiée afin de minimiser l'erreur pour les conditions de fonctionnement typiques d'un circuit numérique.

Deux modèles basés sur la physique (BSIM3v3 et Philips Model 9) sont en passe de devenir des standards pour la communication par les fondeurs des modèles de transistors submicroniques à utilisation analogique. Ce fait, ainsi que la supériorité de ces modèles par rapport à ceux moins adaptés aux canaux courts (Grove-Frohman, Lattin-Jenkins) est le critère de choix pour leur implémentation dans un outil de CA. En se fiant à la littérature et aux sources industrielles, il est difficile de savoir quel modèle (BSIM3v3 ou Model 9) est le meilleur. Cependant, le Model 9 a été utilisé dans de nombreuses situations de conception industrielle, alors qu'il n'en va pas de même pour BSIM3v3. Egalement, le BSIM3v3 utilise de nombreux multiples d'unités ($cm^2/V/s$, $\Omega\mu m$), alors que le Model 9 met tous ses paramètres en unités standards. Ce point, bien que trivial du point de vue de la physique, est très important pour la cohérence globale du système d'équations, ainsi que pour la confiance que chaque modèle inspire: il est plus facile de coder et de corriger un modèle dont les unités sont homogènes.

Il doit être précisé que l'utilisation de modèles MOS n'implique pas la simulation numérique : les modèles sont utilisés sous forme autonome, ce qui est détaillé dans la section suivante. Ceci signifie que, étant donné un jeu de dimensions et de conditions de polarisation, les paramètres intrinsèques d'un transistor tels que la transconductance, la tension de seuil, etc., peuvent être calculés. Ces paramètres sont utilisés dans les équations analytiques du circuit, qui peuvent alors rester indépendantes du modèle de

1. Il est important dans la simulation analogique d'utiliser un modèle avec une transition lisse entre régions. Souvent, ce n'est pas le cas et des discontinuités se produisent dans les fonctions dérivées (g_m, g_{ds}, g_{mb}). Mise à part la considération de la précision, une telle situation peut entraver à la convergence en simulation. Ceci est dû aux oscillations dans la valeur calculée.

dispositif. Le modèle est explicite pour W, L, V_{gs}, V_{ds} et V_{bs} connues. Les modèles avancés MOS ne permettent pas l'inversion d'équations en raison de l'interdépendance complexe des fonctions envers les paramètres des dispositifs. Donc, si un paramètre de dispositif (ex. V_{gs}) est inconnu et un paramètre intrinsèque (ex. courant de drain) est connu, le paramètre inconnu peut être trouvé par approximations successives. Cette dépendance implicite des paramètres constitue une propriété utile pour la CA.

2.2.2 La modélisation du transistor MOS dans la conception automatisée

La nature évolutive de la modélisation des transistors MOS signifie que l'utilisation d'un seul modèle pour générer les équations du circuit (ou même d'un jeu d'équations simplifiées et approximatives du modèle) limite la pérennité de l'outil. Dans ce cas, le modèle de dispositif et le modèle du circuit sont inextricablement liés. Un outil pratique de synthèse doit pouvoir autoriser l'ajout ou la modification des modèles de dispositif avec un minimum d'intervention afin de s'adapter aux évolutions des technologies. Il est donc logique d'utiliser la notion de dispositifs autonomes, déjà introduite dans la description d'ASTRX/OBLX, ou des modèles industriels complets de dispositifs sont conservés indépendamment du reste du système. Afin d'évaluer un paramètre intrinsèque de dispositif (tel que la transconductance, la conductance de sortie, la tension de seuil, etc.), les équations du circuit doivent générer une requête d'évaluation vers le modèle autonome, comme illustré dans la Figure 2.5.

Un langage orienté objet, tel que le C++ [LIP91], est idéal pour l'implémentation de ce type de séparation de code. Un dispositif possède deux parties distinctes : une partie dépendante du circuit (les dimensions et les conditions de polarisation), unique à chaque dispositif ; et une partie dépendante de la technologie (les paramètres technologiques) qui est identique pour tous les dispositifs du même type (N ou P). Pour restreindre l'utilisation de la mémoire vive à une limite raisonnable, il est souhaitable de ne charger ce modèle qu'une fois pour l'ensemble des dispositifs. Ceci signifie que le dispositif et le modèle sont deux classes distinctes, comme illustré dans la Figure 2.6.

La classe *mos* contient les dimensions du dispositif, un pointeur vers un objet de la classe *modèle*, et des fonctions de classe de haut-niveau (*méthodes*) qui représentent chaque paramètre intrinsèque devant être évalué. En fait, chaque méthode appelle la fonction correspondante dans l'objet *modèle*. La classe *modèle* est sensiblement plus complexe du fait de sa modularité inhérente. La partie commune du modèle (paramètres technologiques communs, modèle diode, etc.) est représentée par une *classe de base*. Le modèle réel hérite de cette classe de base tout en lui ajoutant une fonctionnalité spécifique. Quand une fonction est appelée à partir d'un objet *mos*, elle passe par ce qui est appelé une *fonction purement virtuelle*, qui n'existe dans la classe de base (*modèle*) que dans un but déclaratif, sa fonction réelle étant définie dans chaque classe héritée (*modèle 1*, *modèle 2*, ...).

Comme évoqué plus haut, il est souhaitable de pouvoir trouver une quantité inconnue, même quand les équations du dispositif ne sont pas écrites de façon explicite en termes de cette inconnue. Une solution par itération est relativement simple en utilisant

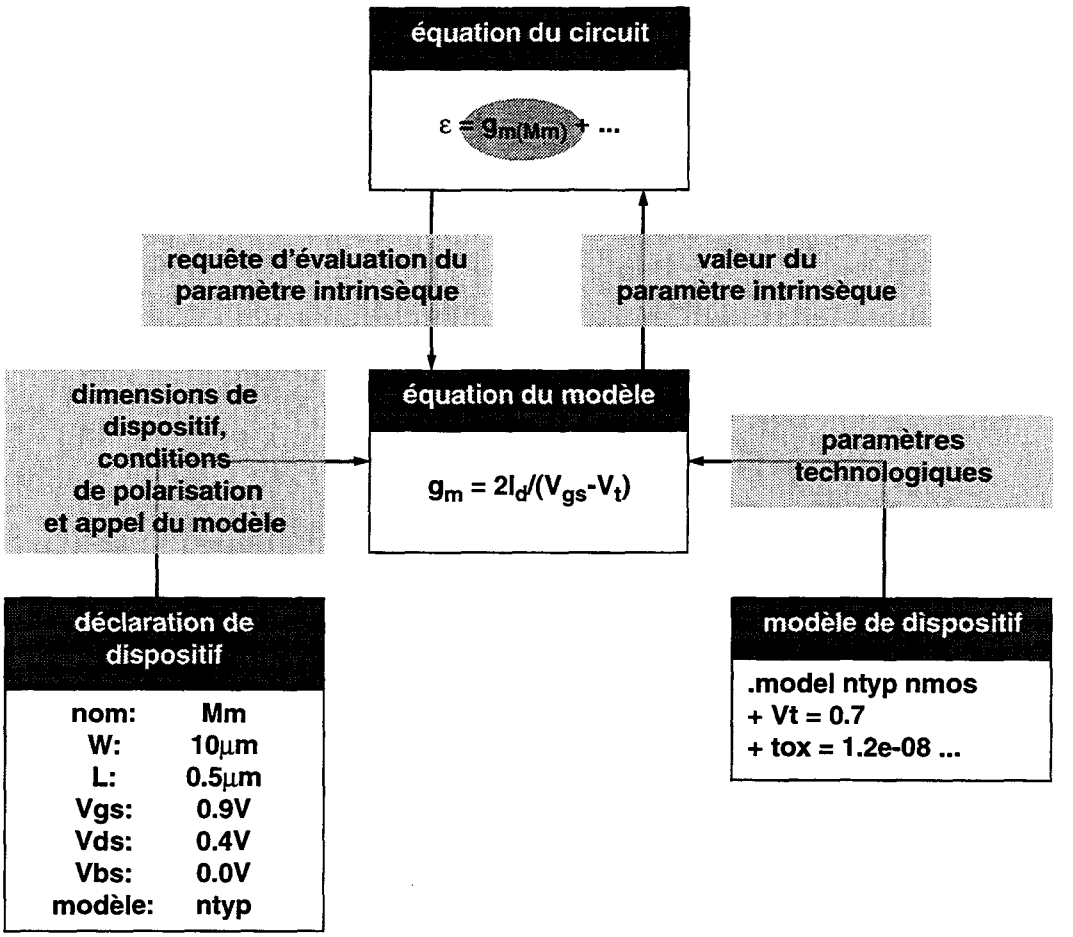


Figure 2.5 Dispositifs autonomes

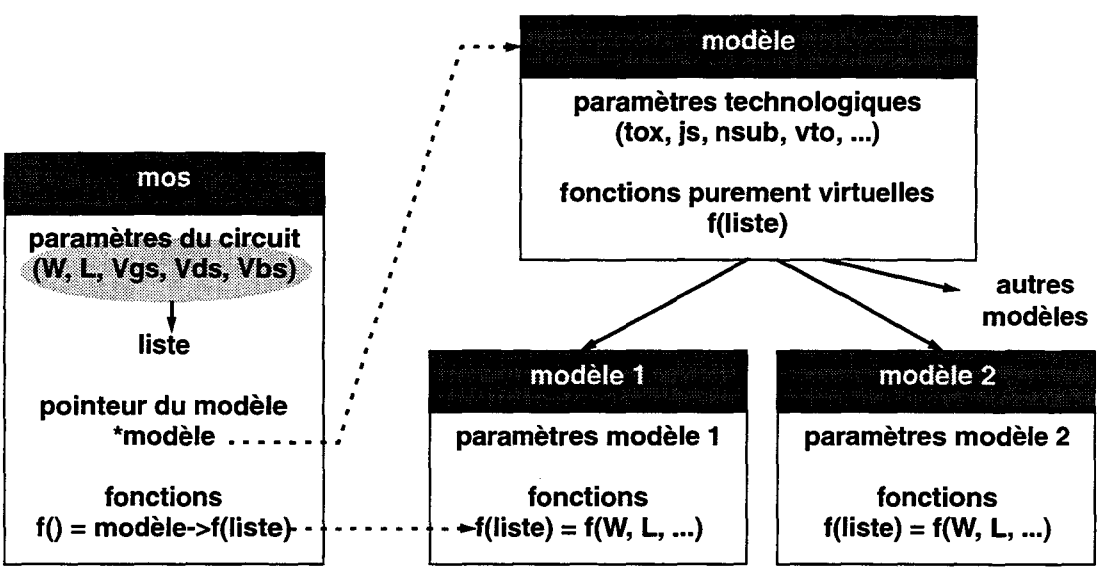


Figure 2.6 Dispositifs autonomes en langage orienté objet

l'algorithme de bissection, où une échelle de valeurs est spécifiée et divisée par bissection successive, jusqu'à ce que l'échelle soit suffisamment proche de la valeur de fonction souhaitée.

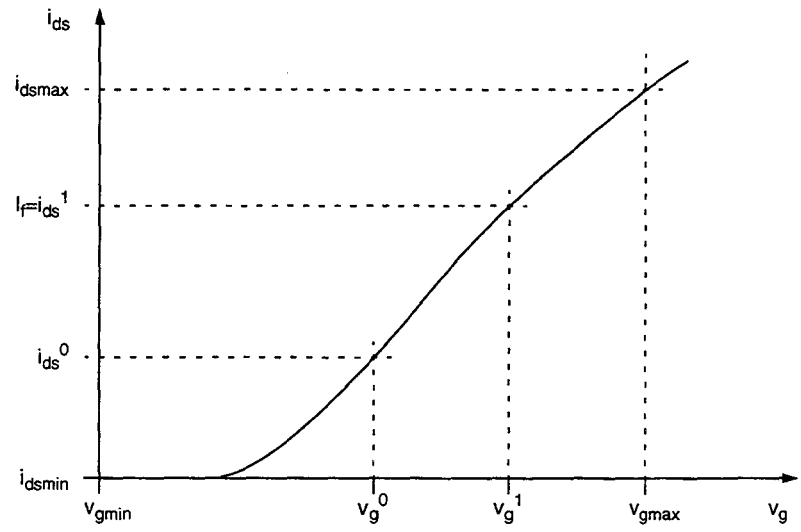


Figure 2.7 Exemple de la procédure de bissection

Dans l'exemple que montre la Figure 2.7, l'inconnue est la tension de grille, et la valeur souhaitée est le courant de drain. Tous les autres paramètres de transistor (W, L, V_{ds} et V_{bs}) sont connus. La procédure de bissection est appelée en spécifiant une échelle de valeurs pour V_{gs} entre V_{gsmin} et V_{gsmax}, et la valeur souhaitée I_f. Le courant est évalué aux points minimum et maximum ainsi qu'à la première bissection V_{gs}^0 (le point de milieu entre minimum et maximum). Puisque les valeurs évaluées du courant montrent que I_f se trouve entre i_{ds}^0 et i_{dsmax}, la nouvelle tension minimale de grille devient V_{gs}^0. Le processus se répète et révèle que le nouveau point de milieu V_{gs}^1 donne la valeur requise du courant $i_{ds}^1=I_f$.

Les sections précédentes se sont concentrées sur le développement d'un cadre dans lequel les modèles analytiques peuvent être implémentés. Deux modèles sont nécessaires au niveau topologique : un modèle détaillé et un modèle de premier ordre. Le modèle détaillé repose sur des modèles modulaires de dispositif, dans lesquels les équations sont cachées par la déclaration haut niveau du dispositif dans le modèle analytique. Il est à présent nécessaire de déterminer le contenu des modèles analytiques, c'est-à-dire la modélisation des effet physiques qui constituent l'évaluation des performances de la cellule de mémoire de courant, évaluation requise par la CA itérative.

2.3 Solution DC

Il est fondamental pour l'évaluation des performances du circuit de connaître la solution du point de polarisation pour les tensions sur tous les noeuds. Pour des valeurs données de dimensions des transistors, et de courants d'entrée et de polarisation, il est suffisant de déterminer la tension sur la grille du transistor mémoire en acquisition afin de polariser correctement les composants de la cellule pour l'évaluation des modèles autonomes. Comme expliqué dans la section 2.2.1, les modèles MOS empêchent la solution statique d'être explicite, et elle doit donc être trouvée par d'autres moyens. Deux possibilités ont été considérées :

2.3.1 Méthode de la matrice des admittances nodales

Les circuits linéaires peuvent être exprimés par une équation matricielle

$$YV = I \tag{2.2}$$

où Y est la matrice des admittances du circuit, V est le vecteur des tensions des noeuds, et I est le vecteur des courants. Sous forme matricielle, l'équation est représentée par

$$\begin{bmatrix} y_{11} & y_{12} & \cdots & y_{1n} \\ y_{21} & y_{22} & \cdots & y_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ y_{n1} & y_{n2} & \cdots & y_{nn} \end{bmatrix} \cdot \begin{bmatrix} V_1 \\ V_2 \\ \cdots \\ V_n \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ \cdots \\ I_n \end{bmatrix} \tag{2.3}$$

La matrice Y peut être extraite d'un circuit linéaire par l'observation que tous les termes diagonaux y_{ii} sont égaux à la somme des conductances branchées sur le noeud i, et que tous les termes hors-diagonale y_{ij} ($i \neq j$) sont égaux à la somme inversée de toutes les conductances qui connectent le noeud i au noeud j. Cependant, cette approche exclut les dispositifs contrôlés par un courant, les sources flottantes de tension et les inductances. Une source idéale de tension a une conductance infinie et son courant est inconnu ; l'inductance est un court-circuit en analyse statique, qui signifie que la chute de potentiel sur cet élément est nulle, et que sa conductance est donc infinie. Ce problème est surmonté par l'analyse nodale modifiée (MNA) [HO75], qui étend les équations nodales afin d'inclure les équations des sources de tension représentées par des courants dans le vecteur inconnu et par des tensions dans le vecteur des courants.

Le vecteur de solution V peut ensuite être calculé par l'inversion de la matrice Y. Cependant, cette approche nécessite un temps de calcul important, et la méthode de l'élimination Gaussienne est habituellement préférée pour les solutions numériques. L'algorithme de l'élimination Gaussienne utilise la mise à l'échelle de chaque équation, suivie par la soustraction des équations restantes afin d'éliminer les inconnues une par une jusqu'à ce que Y soit réduite à une matrice triangulaire supérieure. La solution peut ensuite être trouvée en calculant chaque élément du vecteur V en commençant par le dernier, ce qui est appelé la phase de retro-substitution.

L'approche MNA est extrêmement rapide pour des circuits linéaires car toutes les étapes peuvent être codées explicitement. C'est pour cette raison que la plupart des algorithmes de simulation numérique l'utilisent afin de formuler les équations du circuit pour les analyses statique, fréquentielle et transitoire. Toutefois, quand le circuit comporte des transistors, le circuit n'est plus linéaire du fait des caractéristiques des transistors et de la nécessité des tensions finies d'alimentation. De nombreux circuits qui comportent des transistors sont nommés linéaires car la dynamique d'entrée se limite à la partie linéaire de la fonction de transfert. La solution statique est néanmoins itérative car les conductances et les transconductances dans la matrice Y sont en fait des fonctions non-linéaires du vecteur des courants. Elles doivent de ce fait être réévaluées après chaque solution, et modifiées dans la matrice des admittances afin de converger vers une solution linéarisée. Ce processus itératif utilise l'algorithme de Newton-Raphson.

En examinant le problème, il devient évident que la formulation des équations du circuit par les méthodes matricielles n'est pas nécessaire car cette formulation fait partie des connaissances spécifiques à la topologie. Les équations peuvent donc être codées explicitement de façon séquentielle, et ensuite résolues itérativement. En principe, cette solution est plus efficace, comme le décrit la section suivante.

2.3.2 Itération d'une séquence d'équations

En utilisant les modèles autonomes de dispositif, nous pouvons dresser un système d'équations pour résolution itérative. L'algorithme illustré dans la Figure 2.8 montre la solution pour la cellule de mémoire de courant de base en acquisition, c'est-à-dire avec la tension grille-source du transistor mémoire V_{gsm} égale à la tension drain-source V_{dsm} de ce dernier. La variation de la tension sur le noeud de la grille fait également changer la tension drain-source V_{dsb} du transistor source. Le courant de polarisation est ainsi modifié, et a pour effet de changer le courant total qui passe dans le transistor mémoire. La tension de grille du transistor mémoire est donc remodifiée. En prenant la tension grille-source comme variable par laquelle toutes les autres quantités sont déterminées explicitement, il en résulte une bonne stabilité de convergence. De ce fait, la valeur initiale est arbitraire, et peut être choisie au point milieu entre la masse et la tension d'alimentation.

Cette deuxième méthode a été choisie afin de calculer le point de polarisation pour divers courants d'entrée lors des calculs analytiques du modèle. En général très peu d'itérations sont nécessaires pour obtenir la convergence, confirmant ainsi la rapidité de cette solution.

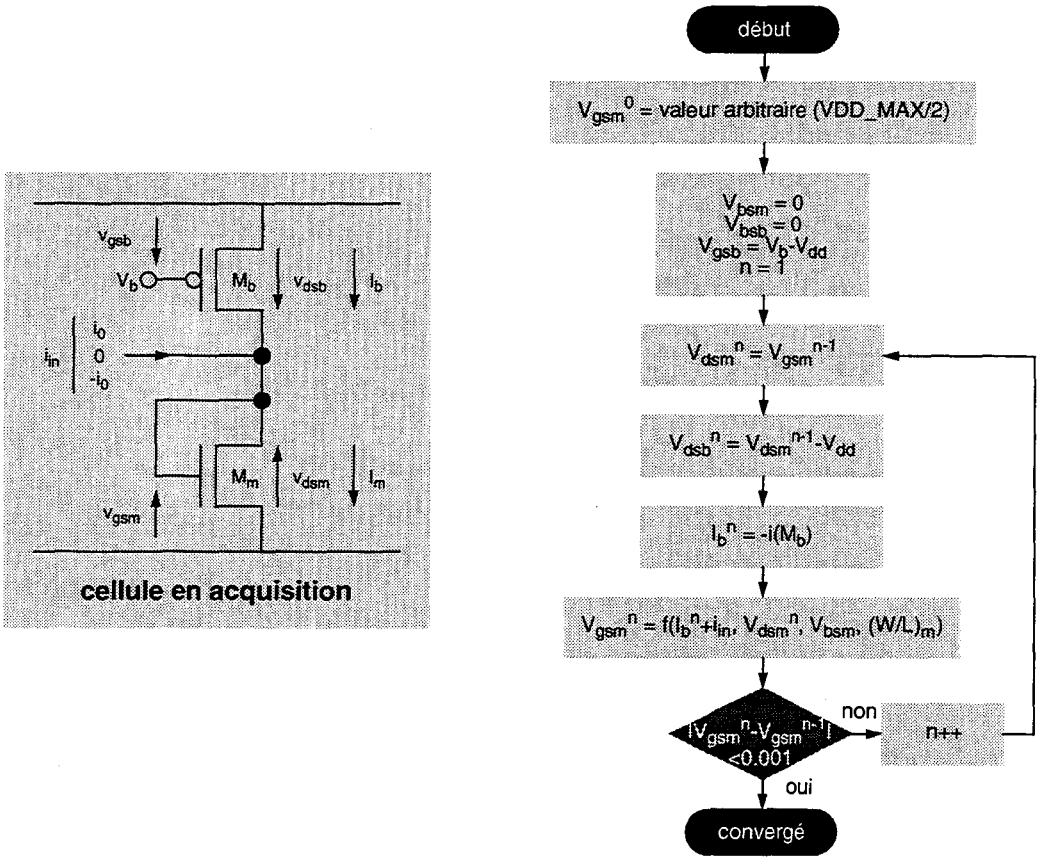


Figure 2.8 Solution DC pour la cellule de mémoire de courant de base

2.4 Considérations pour la dynamique d'entrée

Le courant d'entrée i_{in} peut s'écrire

$$i_{in} = i_0 \sin \omega t \tag{2.4}$$

Quand le courant d'entrée se trouve à sa valeur minimale ($-i_0$), le courant total dans le transistor est égal à I_b-i_0. Si la dynamique du courant d'entrée est trop élevée, ce courant total peut être trop peu important pour assurer la saturation, ce qui forcerait le transistor mémoire dans la zone linéaire de fonctionnement. Un courant total minimal dans le transistor est imposé par la limitation de la dynamique d'entrée par rapport au courant de polarisation :

$$i_0 < \eta I_b \tag{2.5}$$

où des valeurs pratiques de η varient entre 0.5 et 0.8. De fortes valeurs de η indiquent en générale des circuits de basse puissance et à bas bruit, et des valeurs faibles favorisent le fonctionnement à haute fréquence.

En supposant que nous traitons un système comportant deux cellules cascadées (Figure 2.9), nous pouvons poser que

$$i_{max} = I_b + i_0 = I_b(1+\eta) \quad \text{cellule en restitution} \tag{2.6}$$

$$i_{min} = I_b - i_0 = I_b(1-\eta) \quad \text{cellule en acquisition} \tag{2.7}$$

où i_{max} représente le courant maximal pouvant passer dans le transistor mémoire, et i_{min} représente le courant minimal dans ce même transistor.

Nous imposons le courant minimal sur la cellule en acquisition car sa tension de drain est identique à sa tension de grille, et elle impose aussi la tension de drain sur la cellule en restitution. De cette manière, nous définissons la limite de saturation pour la cellule en restitution.

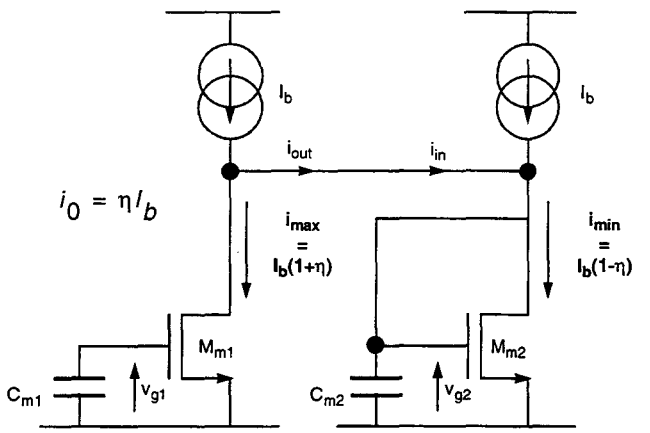


Figure 2.9 Paire de cellules communicantes pour l'analyse de la dynamique d'entrée

Par une relation explicite au premier ordre pour le courant, la tension de grille sur chaque noeud mémoire peut être trouvée, en supposant que la condition pour la saturation de M_{m1} reste valable :

$$v_{g1} = V_t + \sqrt{\frac{2I_b}{\beta}}\sqrt{1+\eta} \tag{2.8}$$

$$v_{g2} = V_t + \sqrt{\frac{2I_b}{\beta}}\sqrt{1-\eta} \tag{2.9}$$

Il est facile de démontrer que pour que la condition de saturation reste valable, il faut

$$v_{g1} - v_{g2} < V_t \tag{2.10}$$

et donc

$$\sqrt{\frac{2I_b}{\beta}}(\sqrt{1+\eta} - \sqrt{1-\eta}) < V_t \tag{2.11}$$

En développant cette expression, nous pouvons déterminer la limite de la dynamique d'entrée. Nous savons que

$$\frac{2I_b}{\beta} = (v_{g0} - V_t)^2 \tag{2.12}$$

où v_{g0} représente la tension de la grille au repos (c'est-à-dire $v_g|_{iin=0}$). Nous pouvons donc générer l'expression suivante pour η:

$$\eta > \frac{V_t}{v_{gst0}}\sqrt{1 - \frac{1}{4}\left(\frac{V_t}{v_{gst0}}\right)^2} \tag{2.13}$$

où $v_{gst0} = v_{g0}-V_t$. La limite de validité pour cette inégalité est $\eta=1$. Ceci se produit quand

$$v_{g0} = V_t\left(1 + \frac{1}{\sqrt{2}}\right) \tag{2.14}$$

La valeur maximale de v_{gst0}, qui représente la limite de la dynamique d'entrée, peut donc s'écrire :

$$v_{gst0} < \frac{V_t}{(\sqrt{1+\eta} - \sqrt{1-\eta})} \tag{2.15}$$

2.5 Erreur de gain statique

Lors de la phase d'acquisition, la tension sur le drain du transistor mémoire est égale à celle sur la grille (configuration diode). En mode de restitution, la tension sur le drain n'est pas nécessairement identique et ne le sera probablement pas. Ceci est dû au fait qu'une cellule qui restitue vers une autre aura une tension de grille correspondant à la valeur nécessaire pour que le courant dans le transistor soit égal à la somme des courants de polarisation et du signal ; et elle aura une tension de drain de valeur égale à la tension de grille de la cellule suivante. Cette dernière est définie par la somme des courants de polarisation et du signal *inversé*. De ce fait, deux composantes d'erreur apparaissent dans le courant de sortie : la première est due à la conductance de sortie non-nulle, qui contribue à apporter une composante statique à l'erreur et sera traitée dans cette section ; la deuxième est due au diviseur capacitif, qui rajoute une composante dynamique, traitée dans la section suivante.

2.5.1 Modèle au premier ordre

Un circuit équivalent petit-signal minimal pour la cellule en modes d'acquisition et de restitution est illustré par la Figure 2.10.

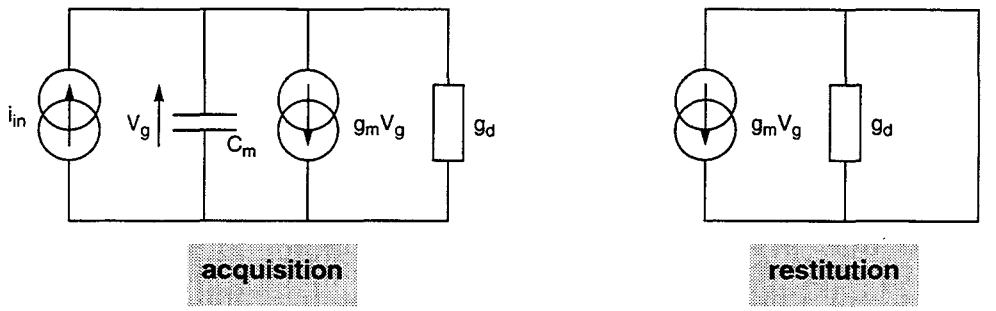


Figure 2.10 Circuit équivalent petit-signal pour l'analyse de premier ordre de l'erreur de gain statique

Pour le circuit équivalent en mode d'acquisition, la valeur statique de V_g peut s'écrire :

$$V_g = \frac{i_{in}}{g_m + g_d} \tag{2.16}$$

En mode de restitution, le courant de sortie comporte une composante due à la transconductance et une autre due à la conductance de sortie. En supposant fixe la tension du noeud de sortie (cas idéal), la composante due à la conductance de sortie peut être supposée constante, représentant ainsi un offset, qui peut être négligée dans cette analyse préliminaire.

$$i_{out} = -g_m V_g \tag{2.17}$$

Le gain du système vaut donc

$$A_{dc} = \frac{i_{out}}{i_{in}} = -\frac{1}{1 + \frac{g_d}{g_m}} \tag{2.18}$$

Idéalement le gain serait égal à -1, et l'erreur de gain statique est donc donnée par

$$\varepsilon_{dc} = \frac{1}{1 + \frac{g_m}{g_d}} \approx \frac{g_d}{g_m} \tag{2.19}$$

En forte inversion et en zone de saturation, la conductance de sortie g_d et la transconductance g_m peuvent être approximées par :

$$g_d = \frac{I_d}{V_E} \tag{2.20}$$

$$g_m = \beta(v_{gs} - V_t) \tag{2.21}$$

où V_E correspond à la tension d'Early, le paramètre de transconductance β est égal à $\mu C_{ox} W/L$, et V_t représente la tension de seuil. Une relation plus fondamentale qui

utilise la transconductance peut s'écrire [SIL96] :

$$I_d = \beta(v_{gs} - V_t)^2 \tag{2.22}$$

$$\frac{g_m}{I_d} = \frac{2}{v_{gs} - V_t} \tag{2.23}$$

et donc le rapport entre la transconductance g_m et la conductance de sortie g_d, c'est-à-dire le gain intrinsèque du transistor, est donné par

$$\frac{g_m}{g_d} = \frac{2V_E}{v_{gst}} \tag{2.24}$$

c'est-à-dire par le rapport entre la tension d'Early V_E et la tension de grille utile v_{gst} (égale à v_{gs}-V_t). Donc, pour une erreur de gain statique donnée, l'équation de cette dernière peut également imposer une limite sur v_{gst0} :

$$v_{gst0} < \frac{2V_E\varepsilon_{dc}}{1 - \varepsilon_{dc}} \tag{2.25}$$

2.5.2 Modèle détaillé

Les modèles autonomes de dispositifs permettent l'évaluation simple et précise de l'erreur de gain. Puisque les dimensions des transistors et leurs conditions de polarisation sont connues lors de l'évaluation des performances (à l'opposé de la fonction de génération du point de départ, c'est-à-dire le modèle de premier ordre, où ces valeurs sont les inconnues), il est trivial de polariser les transistors de mémoire et de source dans la cellule en restitution, comme illustré dans la Figure 2.11 afin de déterminer l'erreur de gain statique. Dans cette figure, V_{g1} représente la tension de grille de la cellule en restitution, et V_{g2} représente la tension de grille de la cellule en acquisition.

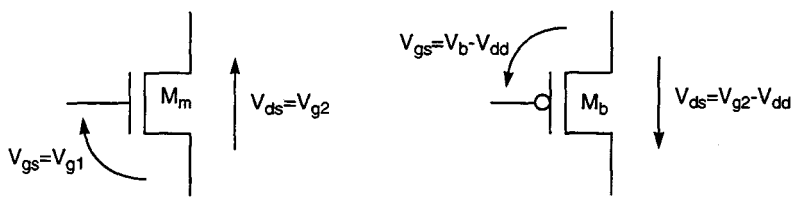


Figure 2.11 Conditions de polarisation pour les transistors de mémoire et de source

A partir de l'évaluation individuelle des courants de drain i_m (du transistor mémoire) et i_b (du transistor source),

$$i_{out} = i_m + i_b \tag{2.26}$$

et l'erreur de gain statique est donnée par

$$\varepsilon_{dc} = \left(\frac{i_{out} + i_{in}}{i_0}\right) \cdot 100\% \tag{2.27}$$

2.6 L'erreur du diviseur capacitif

La variation de tension ΔV_d sur le drain du transistor mémoire est transmise de façon dynamique sur la grille à cause d'une capacité parasite entre grille et drain, qui, avec la capacité mémoire, constitue un diviseur capacitif, comme illustré dans la Figure 2.12.

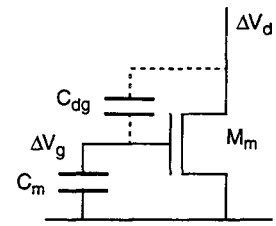


Figure 2.12 Erreur de gain dynamique due au diviseur capacitif

ΔV_d est la différence entre la tension de drain en acquisition V_{da} (égale à la tension de grille en acquisition V_{ga}) et la tension de drain en restitution V_{dr}, c'est-à-dire

$$\Delta V_d = V_{dr} - V_{da} = V_{dr} - V_{ga} \tag{2.28}$$

La variation sur la grille du transistor mémoire vaut

$$\Delta V_g = \Delta V_d \frac{C_d}{C_d + C_m} \tag{2.29}$$

où ΔV_g est la différence entre la tension de grille modifiée V_g' et la tension de grille en acquisition V_{ga} :

$$\Delta V_g = V_g' - V_{ga} \tag{2.30}$$

ΔV_g engendre une variation du courant de sortie :

$$\Delta i = g_m \Delta V_g \tag{2.31}$$

L'erreur du diviseur capacitif, normalisée à l'amplitude du courant d'entrée, est ensuite donnée par

$$\varepsilon_{cap} = \frac{\Delta i}{i_0} \cdot 100\% \tag{2.32}$$

2.7 La précision d'établissement

La cellule mémoire, étant une cellule à temps-discret, peut être caractérisée par sa réponse à un échelon en entrée. L'échelon part d'un courant d'entrée nul (mode de conservation) vers un courant d'entrée non-nul (mode d'acquisition). Puisque le temps d'établissement de la cellule n'est pas nul, la fréquence d'échantillonnage est limitée à des valeurs finies, et une relation entre la fréquence d'échantillonnage et la bande passante (pour une précision d'établissement donnée) peut être trouvée. Une première approche consiste à prendre l'hypothèse petit-signal. Plus la dynamique d'entrée de la cellule est grande, moins cette hypothèse sera valable, car des effets grand-signal, comme le slew rate, interviennent.

2.7.1 Modèle au premier ordre

En mode d'acquisition, une représentation (petit-signal) fortement simplifiée de la cellule est montrée dans la Figure 2.13.

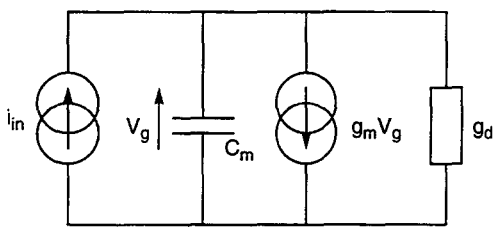


Figure 2.13 Circuit équivalent petit-signal pour l'analyse de l'erreur d'établissement au premier ordre

C'est un système du premier ordre avec un pôle unique à

$$\omega_p = \frac{g_m + g_d}{C_m} \tag{2.33}$$

Si nous supposons que $g_d << g_m$, cette équation se réduit à :

$$\omega_p \approx \frac{g_m}{C_m} \tag{2.34}$$

L'erreur d'établissement d'un système du premier ordre est donnée par :

$$\varepsilon_s = e^{-\omega_p t} \tag{2.35}$$

Il importe de noter que tant que l'erreur d'établissement est fortement inférieure à l'erreur de gain statique, les hypothèses utilisées dans le calcul de cette dernière sont valables. Dans le cas contraire, l'erreur de gain statique peut être sous-estimée (dans le cas du régime sous-amorti) puisque la dynamique réelle de la tension de grille est plus importante que celle supposée dans l'analyse statique simple.

Nous pouvons donc déterminer la capacité maximale de grille permise pour une précision d'établissement donnée :

$$C_m < \frac{t_{acq} g_m}{\ln \varepsilon_s} \tag{2.36}$$

où t_{acq} représente le temps d'acquisition, qui est nécessairement une fraction de la période de l'horloge principale.

2.7.2 Modèle petit-signal

L'analyse précédente n'a pas pris en compte la résistance des interrupteurs d'entrée, et a supposé une source de courant de polarisation idéale. En réalité, la résistance des interrupteurs est non-nulle, représentée par les conductances g_{sa} et g_{sb}, et la source de courant de polarisation est réalisée par un transistor MOS avec une conductance de sortie finie g_{db}. De plus, la source de courant d'entrée n'est pas idéale, car il peut s'agir d'une autre cellule de mémoire de courant présentant une conductance de sortie non-nulle. Dans le cas général, l'admittance de la source peut être représentée par Y_{in}. La Figure 2.14 prend en compte les effets décrits ci-dessus.

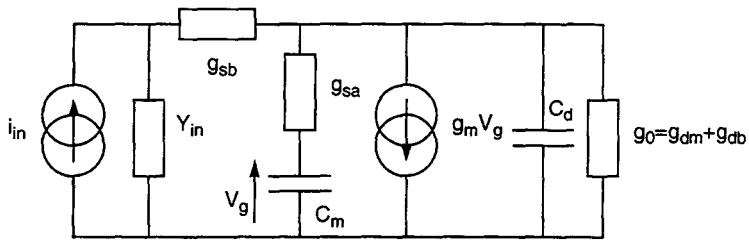


Figure 2.14 Circuit équivalent petit-signal pour l'analyse au second ordre

La fonction de transfert pour ce modèle petit-signal est :

$$H(p) = \frac{v_g}{i_{in}} = K \frac{1}{1 + \tau_1 p + \tau_2 p^2} \tag{2.37}$$

où

$$K = \frac{g_{sb}}{(g_m + g_0)(Y_{in} + g_{sb}) + Y_{in} g_{sb}} \tag{2.38}$$

$$\tau_1 = \frac{C_m((g_{sa} + g_0)(Y_{in} + g_{sb}) + Y_{in} g_{sb})}{g_{sa}((g_m + g_0)(Y_{in} + g_{sb}) + Y_{in} g_{sb})} \tag{2.39}$$

$$\tau_2 = \frac{C_m C_d (Y_{in} + g_{sb})}{C_m((g_{sa} + g_0)(Y_{in} + g_{sb}) + Y_{in} g_{sb})} \tag{2.40}$$

La fonction de transfert d'un système du second ordre dans sa forme générale est donnée par :

$$H(p) = \frac{H_0}{1 + \frac{p}{\omega_0 Q} + \frac{p^2}{\omega_0^2}} \tag{2.41}$$

et, en se référant à la fonction de transfert de la cellule, nous pouvons déduire par identification que

$$\omega_0 = \frac{1}{\sqrt{\tau_1 \tau_2}} \tag{2.42}$$

$$Q = \sqrt{\frac{\tau_1}{\tau_2}} \tag{2.43}$$

La valeur du coefficient de qualité, Q, détermine le type de réponse du circuit. Les expressions de la fonction de transfert, pour la réponse à une fonction d'échelon dans le domaine temporel, sont calculées dans les cas suivants :

- Q>0.5 (sous-amorti)

$$f(t) = \frac{2Qe^{-\frac{\omega_0 t}{2Q}}}{\sqrt{4Q^2-1}} \sin\left(\frac{\omega_0 t\sqrt{4Q^2-1}}{2Q} + \operatorname{atan}(\sqrt{4Q^2-1})\right) - 1 \tag{2.44}$$

- Q=0.5 (amortissement critique)

$$f(t) = e^{-\omega_0 t}(1 + \omega_0 t) - 1 \tag{2.45}$$

- Q<0.5 (sur-amorti)

$$f(t) = \frac{1}{2\sqrt{1-4Q^2}}\left[(1+\sqrt{1-4Q^2})e^{-\frac{\omega_0 t}{2Q}(1-\sqrt{1-4Q^2})} - (1-\sqrt{1-4Q^2})e^{-\frac{\omega_0 t}{2Q}(1+\sqrt{1-4Q^2})}\right] - 1 \tag{2.46}$$

et l'erreur d'établissement est ensuite donnée par

$$\varepsilon_s = |1 - f(t)| \tag{2.47}$$

2.7.3 Modèle linéaire par segments

Le modèle petit-signal est valable pour $i_0 << I_b$ puisque les variations des paramètres petit-signal sont négligeables. Cependant, les valeurs pratiques de i_0 peuvent atteindre $0.8I_b$, et l'amplitude du signal est telle que l'approximation petit-signal n'est pas suffisamment précise.

Afin de prendre en compte les variations grand-signal dans le circuit, nous pouvons utiliser un modèle linéaire par segments ("piecewise linear" - PWL) [MOE94]. En évaluant le comportement petit-signal pour les conditions initiales et finales, la précision du calcul est améliorée.

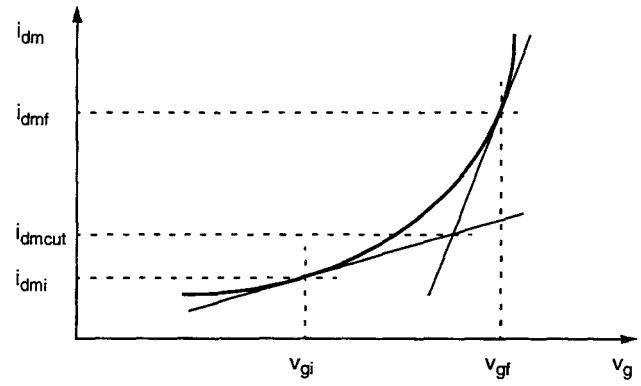


Figure 2.15 Evaluation de l'intersection pour le modèle d'établissement linéaire par segments

Le modèle prend deux points à (V_{gi}, I_{dmi}) et à (V_{gf}, I_{dmf}), comme illustré dans la Figure 2.15, et évalue ensuite le courant d'intersection i_{dmcut} par l'intersection des tangentes :

$$i_{dmcut} = \frac{g_{mi}g_{mf}(v_{gf} - v_{gi}) + (g_{mf}i_{dmi} - g_{mi}i_{dmf})}{g_{mf} - g_{mi}} \tag{2.48}$$

Le modèle petit-signal est ensuite déterminé jusqu'à l'intersection avec les valeurs initiales et, à partir de l'intersection, avec les valeurs finales (Figure 2.16). L'exemple montre que les résultats obtenus avec ce modèle sont très différents de ceux obtenus avec le modèle petit-signal classique et sont plus fidèles à la réalité.

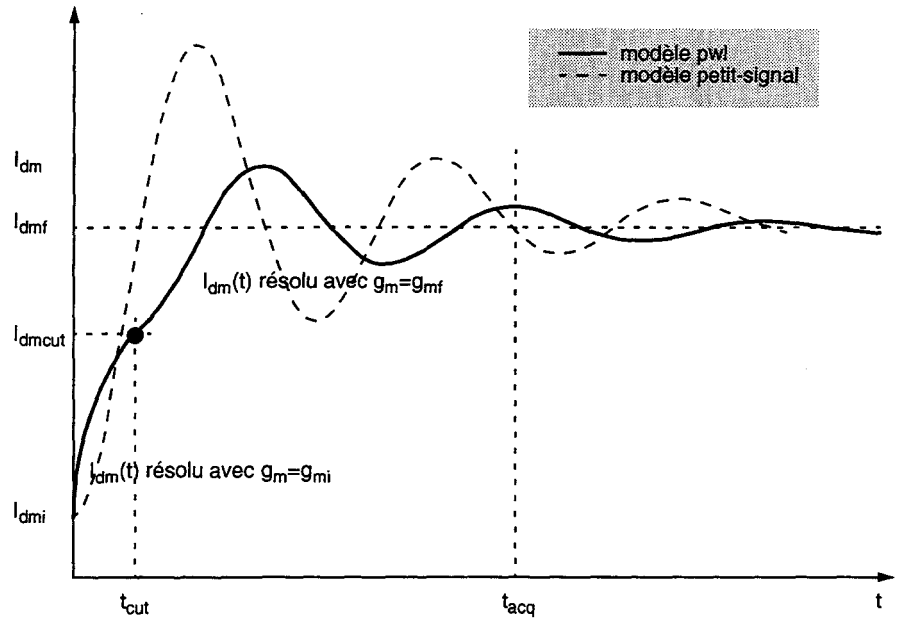


Figure 2.16 Solution "linéaire par segments" de l'erreur d'établissement

2.7.4 Enveloppe en régime sous-amorti

L'évaluation de l'erreur d'établissement à l'instant $t=t_{acq}$ pose un problème quand la réponse est sous-amortie : l'erreur peut être nulle à cet instant, alors que le circuit oscille encore. Bien que rigoureusement correcte, cette caractéristique peut tromper l'évaluation des performances dans la CA, car elle ne reflète pas la situation réelle (Figure 2.17) : la précision dépend fortement du temps d'acquisition. Une dégradation importante de la précision d'établissement peut résulter d'une légère variation dans la caractéristique causée par l'influence d'éléments parasites.

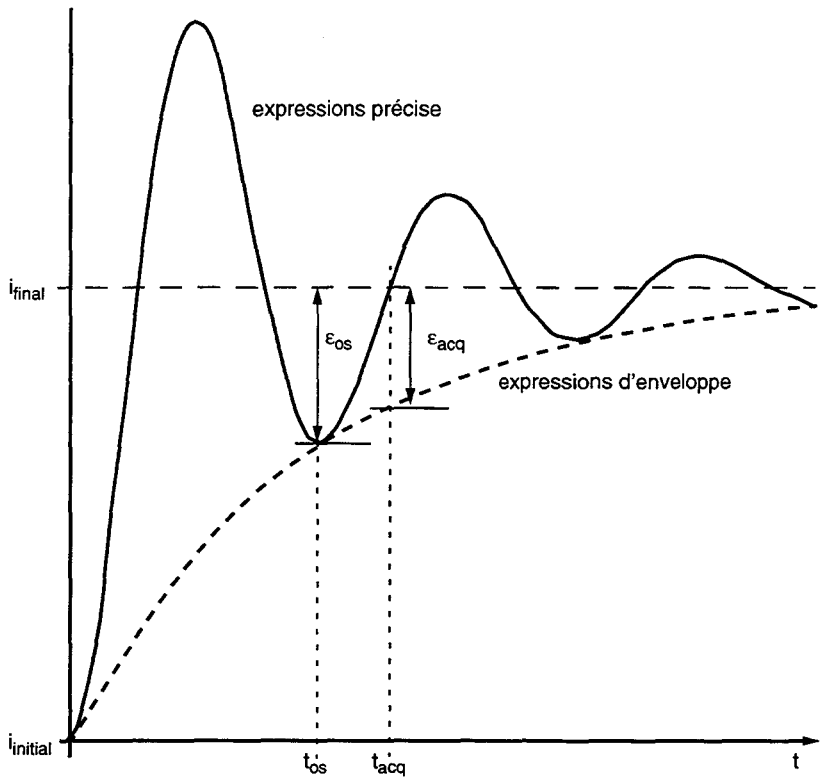


Figure 2.17 Etablissement d'un système du second ordre en régime sous-amorti

Une estimation pessimiste de l'erreur évaluerait la dernière oscillation précédant t_{acq} ($\varepsilon=\varepsilon_{os}$). Cependant, cette approche pourrait se révéler trop restrictive. De plus, l'estimation de l'erreur évoluerait par paliers (à chaque demi-période).

Nous estimons l'erreur de façon plus réaliste en évaluant l'enveloppe pour l'expression sous-amortie ($\varepsilon=\varepsilon_{acq}$). Les points permettant l'évaluation de cette enveloppe consistent en les points de contact entre l'enveloppe et les oscillations, où les gradients des deux courbes sont identiques. Ces points *tangentiels* se produisent quand le terme sinusoïdal de la fonction d'erreur est unitaire, soit pour :

$$\frac{\omega t\sqrt{4Q^2-1}}{2Q} + \operatorname{atan}(\sqrt{4Q^2-1}) = \frac{\pi}{2}(2n+1) \qquad (2.49)$$

Nous en déduisons

$$n = floor\left[\frac{1}{2}\left(\frac{2}{\pi}\left(\frac{\omega t\sqrt{4Q^2-1}}{2Q} + \text{atan}(\sqrt{4Q^2-1})\right) - 1\right)\right] \quad (2.50)$$

et nous obtenons ensuite

$$t_{os} = \frac{2Q}{\omega\sqrt{4Q^2-1}}\left[\frac{\pi}{2}(2n+1) - \text{atan}(\sqrt{4Q^2-1})\right] \quad (2.51)$$

De cette expression nous pouvons trouver la constante de temps de l'enveloppe

$$\tau_{os} = \frac{t_{os}}{|\ln\varepsilon_{os}|} \quad (2.52)$$

et l'erreur d'établissement est, de ce fait, donnée par

$$\varepsilon_{acq} = e^{-\frac{t_{acq}}{\tau_{os}}} \quad (2.53)$$

Les variations des deux expressions d'erreur (expressions précise et d'enveloppe) par rapport au facteur d'amortissement Q sont illustrées par la Figure 2.18. Nous pouvons constater que l'expression précise montre un retour à zéro autour des points entre extrêmes. L'expression d'enveloppe n'est identique à l'expression précise qu'aux points tangentiels.

Nous remarquons également un léger décrochage dans la courbe juste avant un point tangentiel. En effet, le calcul se base sur l'oscillation précédente, et les erreurs d'arrondi de machine contribuent à une erreur cumulative. Cette erreur est maximale quand le temps d'acquisition termine juste avant l'extremum suivant. Dans la pratique, elle est négligeable, et n'affecte pas le processus de synthèse de façon significative.

2.8 L'injection de charge

A la fin de la période d'acquisition, l'interrupteur relié au drain et à la grille du transistor mémoire s'ouvre. Cet interrupteur étant réalisé avec un transistor MOS, une charge est injectée sur la capacité de grille du transistor mémoire [SHI87]. Cette charge provient de deux sources (Figure 2.19) :

- pendant la conduction, une couche d'inversion est créée dans le canal du transistor de passage. Lors du processus d'extinction du courant, ces charges mobiles q_{ch} s'évacuent par le drain, la source et le substrat :

$$q_{ch} = (V_{dd} - v_g - V_{ta})C_{ch} \quad (2.54)$$

1. floor(x): nombre entier maximal, inférieur à x

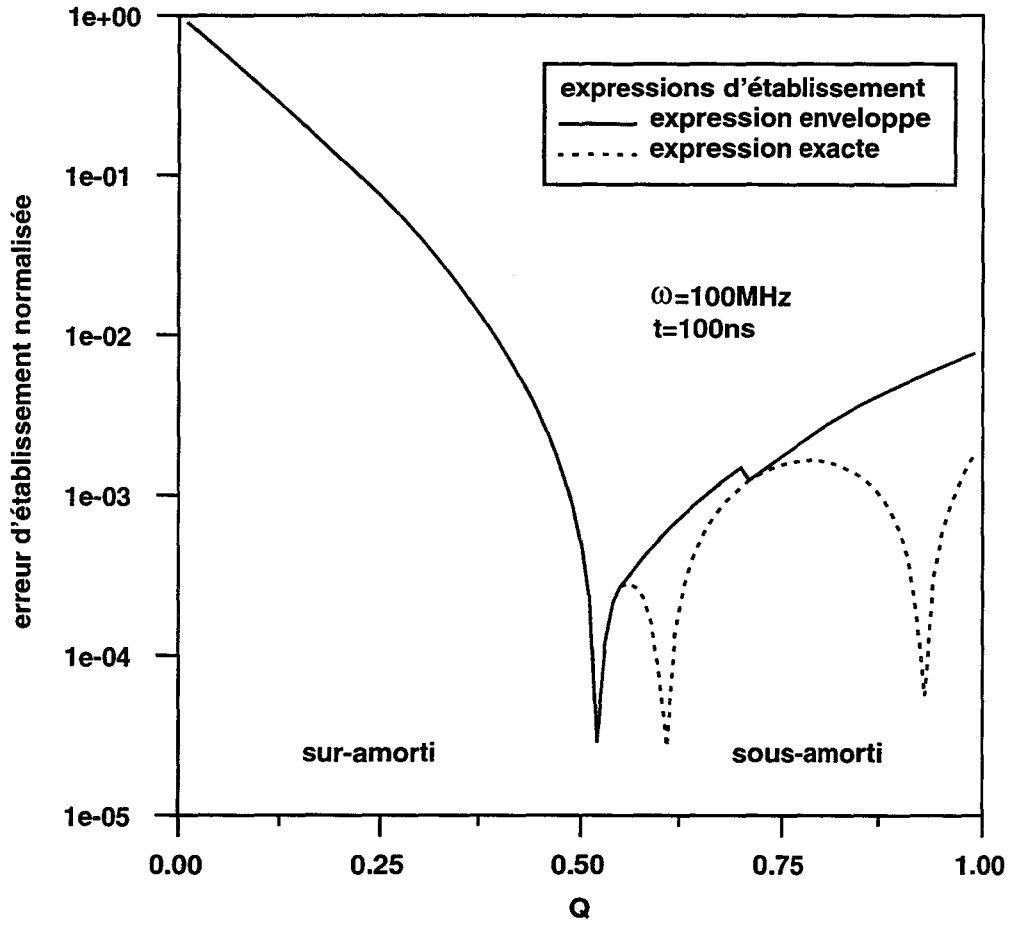


Figure 2.18 Comparaison entre l'expression exacte et l'enveloppe de l'erreur d'établissement

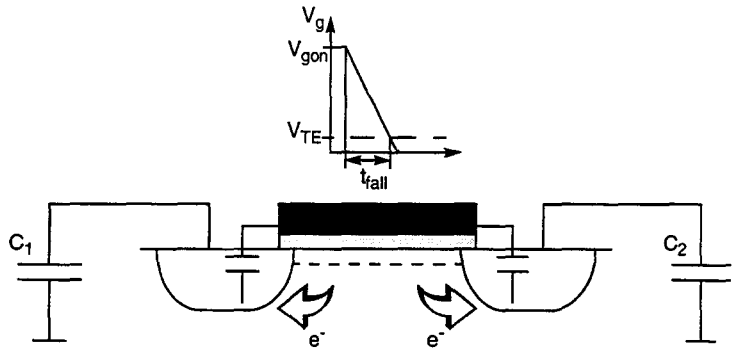


Figure 2.19 Sources de l'injection de charge

où V_{dd} est le potentiel de la grille de l'interrupteur, v_g est le potentiel du canal, et V_{ta} est la tension de seuil de l'interrupteur, prenant en compte toutes les tensions de surface et de contact entre les tensions externes et les tensions réelles internes. C_{ch} est la capacité du canal de l'interrupteur :

$$C_{ch} = C_{ox} W_{sw} L_{sw} \tag{2.55}$$

- le changement de tensions sur la grille de l'interrupteur engendre un passage de charge q_{ol} entre les capacités parasites de recouvrement grille-diffusion vers la source et le drain.

$$q_{ol} = V_{dd} C_{gsw} \tag{2.56}$$

où V_{dd} représente l'excursion de tension ΔV sur la grille de l'interrupteur, et C_{gsw} est la capacité de recouvrement entre la grille et la source.

La charge totale injectée sur la capacité de grille est donnée par

$$q = \alpha q_{ch} + q_{ol} \tag{2.57}$$

où α représente la fraction de la charge totale dans le canal qui s'évacue de l'interrupteur via la source. C'est une valeur difficile à quantifier car elle dépend de la pente du signal de commande lors de la fermeture de l'interrupteur, mais aussi des impédances relatives vues sur chaque noeud (drain et source).

L'erreur sur le courant mémorisé induite par cette injection de charge est

$$\delta I = \frac{g_m q}{C_m} \tag{2.58}$$

et constitue en général le facteur limitant dans la conception des cellules à courants commutés. Des techniques de compensation de ce problème existent, dont la plus efficace apparaît être la méthode des interrupteurs de compensation ("dummy switch"). Un interrupteur factice de dimensions αW_{sw} et L_{sw} est ajouté sur le noeud de mémorisation. Sa tension de grille est commandée par un signal inverse à celui de l'interrupteur utile. De ce fait, l'interrupteur de compensation capte $q_{ch}' = \alpha q_{ch}$ dans son canal et $q_{ol}' = q_{ol}$ sur ses capacités de diffusion-grille. Ceci a l'effet d'annuler la charge de l'interrupteur utile, en supposant que la valeur d'α est connue d'avance. En réalité, la charge totale est modifiée pour donner :

$$q = \alpha q_{ch} k_{ch} + q_{ol} k_{ol} \tag{2.59}$$

où les coefficients k_{ch} et k_{ol} représentent l'efficacité de l'interrupteur de compensation dans l'annulation de la charge provenant de l'interrupteur utile. Le paramètre critique pour la prédiction correcte de la distribution de charge est α, pour laquelle deux méthodes d'évaluation existent.

2.8.1 Modèle simple

La première technique d'évaluation de α considère que la distribution de charge est symétrique entre drain et source du transistor de passage. C'est effectivement le cas si l'interrupteur s'ouvre de façon très rapide telle que la charge doit être évacuée de façon

symétrique vers les deux côtés du canal, ou si les impédances vues sur le drain et la source sont égales. Cette approximation pose alors que

$$\alpha = 0.5 \tag{2.60}$$

Comme évoqué précédemment, les conditions pour lesquelles cette hypothèse est valable ne seront probablement pas vérifiées. Une analyse plus précise est nécessaire, qui prendra en compte les impédances des noeuds et le temps de blocage de l'interrupteur.

2.8.2 Le modèle de Vittoz

Le modèle pour l'évaluation plus précise de la distribution de charge à l'ouverture de l'interrupteur est donné dans la Figure 2.20 [WEG87].

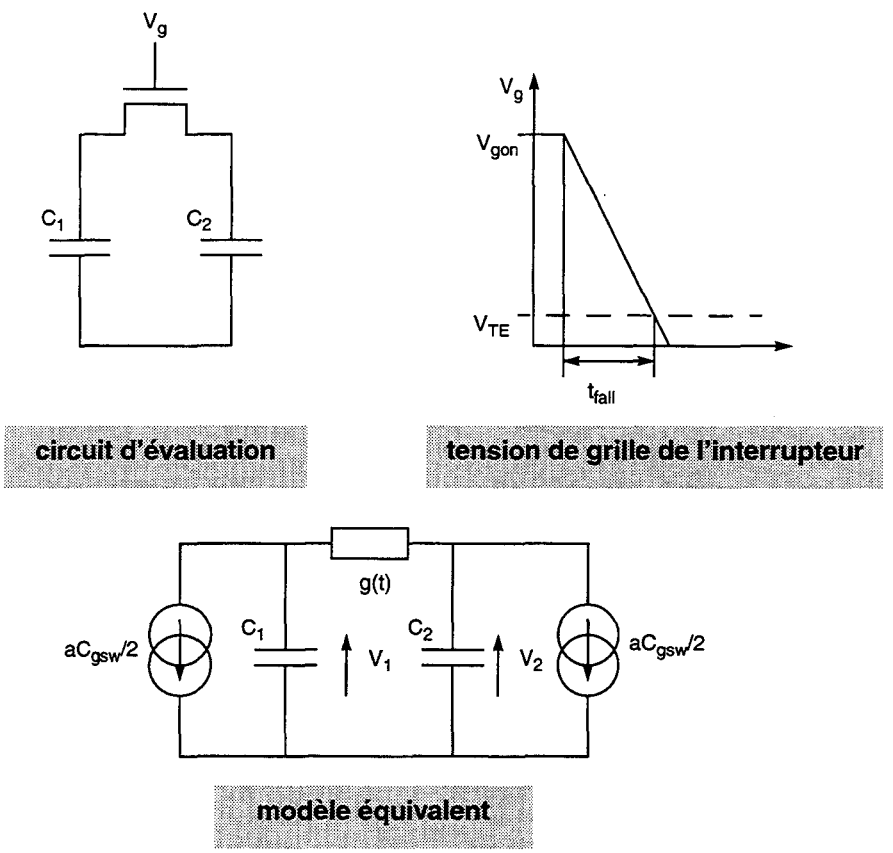


Figure 2.20 Modèle équivalent pour l'analyse détaillée de l'injection de charge

Le modèle représente l'interrupteur par une conductance variable en temps :

$$g(t) = \beta(V_g(t) - V_{TE}) \tag{2.61}$$

en prenant le modèle petit-signal du premier ordre, ce qui est suffisant pour cette analyse. Nous pouvons remplacer $V_g(t)$ dans l'équation :

$$g(t) = \beta(V_{gon} - at - V_{TE}) \tag{2.62}$$

Si le temps de descente t_{fall} est important devant la constante de temps du circuit τ (C/g(t)) et si les capacités C_1 et C_2 sont toutes les deux grandes devant la capacité de grille de l'interrupteur C_{gsw}, nous pouvons poser que le canal est uniforme et que le courant qui sort du drain est égal à celui qui sort de la source :

$$i_d = i_s = \frac{1}{2} C_{gsw} \frac{dV}{dt} = \frac{aC_{gsw}}{2} \tag{2.63}$$

Ce système d'équations peut être résolu afin d'obtenir une équation différentielle :

$$\frac{dV}{dT} = (T - B)\left[\left(1 + \frac{C_2}{C_1}\right)V + 2T\frac{C_2}{C_1}\right] - 1 \tag{2.64}$$

où

$$V = \frac{\Delta v_2}{\frac{C_{gsw}}{2}\sqrt{\frac{a}{\beta C_2}}} \tag{2.65}$$

$$T = \frac{t}{\sqrt{\frac{C_2}{a\beta}}} \tag{2.66}$$

$$B = (V_{gon} - V_{TE})\sqrt{\frac{\beta}{aC_2}} = \sqrt{\frac{t_{fall}}{\tau}} \tag{2.67}$$

$$a = \frac{V_{gon} - V_{TE}}{t_{fall}} \tag{2.68}$$

Une méthode simple d'intégration numérique (Improved Euler) est utilisée afin de résoudre l'équation par rapport à V :

$$V_{n+1} = V_n + \frac{h}{2}[f(T_n, V_n) + f(T_n + h, V_n + hf(T_n, V_n))] \tag{2.69}$$

où h est le pas d'intégration, et la fonction f est la dérivée dV/dT. En intégrant la dérivée de T=0 à T=B ($t=t_{fall}$), nous obtenons une valeur finale de V qui donne alors :

$$\Delta q_2 = C_2 \cdot V\frac{C_{gsw}}{2}\sqrt{\frac{a}{\beta C_2}} \tag{2.70}$$

Comme

$$q_{tot} = C_{gsw}(V_{gon} - V_{TE}) \tag{2.71}$$

nous pouvons en déduire la distribution de charge vers la grille.

Ce traitement théorique donne les caractéristiques représentées en Figure 2.21. Du fait de la nature petit-signal du modèle d'injection de charge, les résultats obtenus sont vérifiables par un simulateur numérique. Il est important de noter que cette analyse néglige la charge évacuée par le substrat. Elle augmente pour des temps de descente du signal de commande plus courts et pour des interrupteurs plus longs (donc de temps de transit plus long). Dans les cas réels, il est raisonnable de négliger le courant partant vers le substrat.

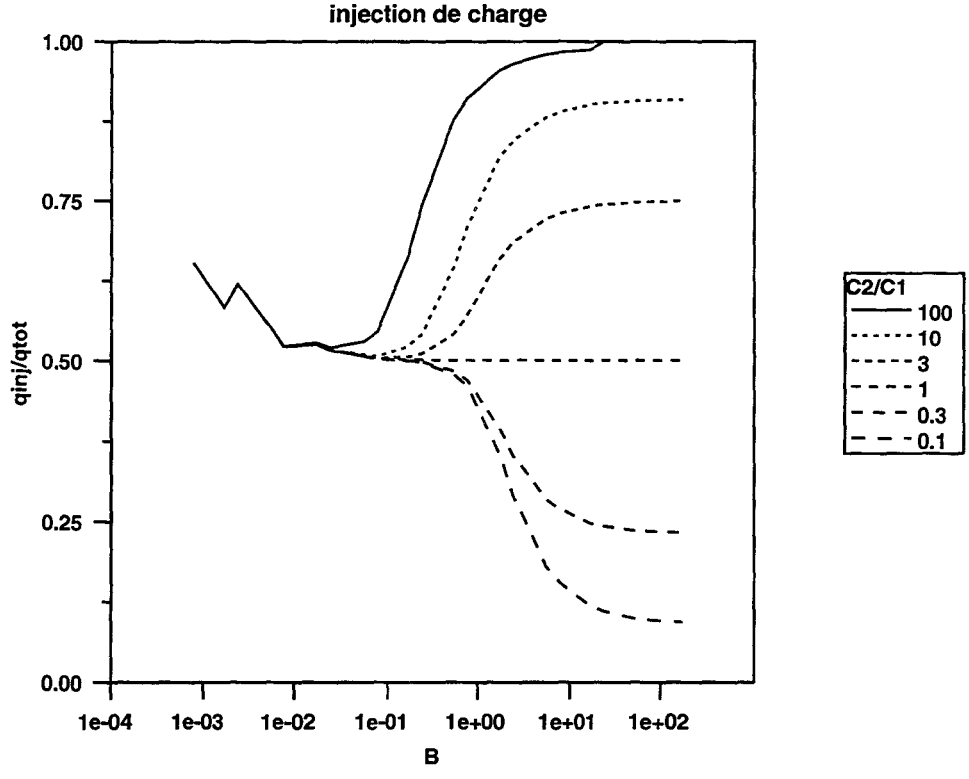


Figure 2.21 Prédiction de l'injection de charge

A l'autre extrême, c'est-à-dire avec des temps de descente très longs et des temps de transit très courts, la conductance de l'interrupteur n'est plus seulement dépendante de la tension grille-source, mais également de la tension drain-source. Une fois de plus, ce facteur aura un effet négligeable dans les cas réels sur les prédictions de l'injection de charge.

Un autre point à noter est que les paramètres qui affectent le modèle dépendent fortement des composants parasites : la pente a, par exemple, peut être de valeur inférieure à sa valeur nominale à cause de parasites supplémentaires dans le système sur le noeud de l'horloge qui ralentissent la transition ; la capacité C_1 même est constituée de parasites, et donc la longueur des pistes ou des imprécisions dans les modèles auront des effets importants sur sa valeur. En effet, l'injection de charge ne peut être simulée

de façon précise que par l'utilisation d'un modèle non-quasi-statique [ROB96], où le canal est divisé en plusieurs parties. Chaque partie est résolue séparément et l'ensemble est ensuite intégrée pour déterminer la distribution des charges à l'ouverture de l'interrupteur. Cette approche est certainement plus précise que les expressions analytiques décrites précédemment, mais elle est beaucoup trop complexe pour la CA, qui ne requiert essentiellement qu'une estimation (éventuellement pessimiste).

Dans le cas pratique, l'injection de charge est habituellement atténuée par un interrupteur de compensation, qui injecte la charge inverse de celle introduite par l'interrupteur utile. Cependant, le désappariement et les capacités asymétriques de recouvrement, avec la charge réellement injectée, contribuent à réduire son efficacité de compensation, illustré dans la Figure 2.22.

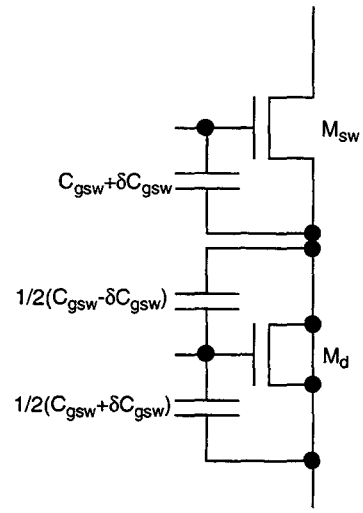


Figure 2.22 Asymétrie des capacités des interrupteurs utile et de compensation

La charge totale réellement injectée est donnée par

$$q = (\alpha - 0.5 \pm k_{ch}) q_{ch} + \Delta V C_{gsw} (\delta \pm k_{ol}) \quad \textbf{(2.72)}$$

δ trouve son origine dans le processus de fabrication. Lors de la fabrication, la diffusion est effectuée par un faisceau à 7° de la perpendiculaire à la tranche afin de réduire la diffusion profonde dans la structure cristalline du silicium. Si aucune rotation de la tranche n'est effectuée, une asymétrie peut se produire dans la diffusion latérale sous la grille (Figure 2.23) et donc également dans les valeurs des capacités de recouvrement sous la grille.

Cependant, les processus de fabrication actuels effectuent un certain nombre de rotations de la tranche pour éliminer les problèmes d'asymétrie (habituellement trois rotations de 90° assurent la diffusion symétrique, quelle que soit l'orientation des transistors par rapport au faisceau d'ions).

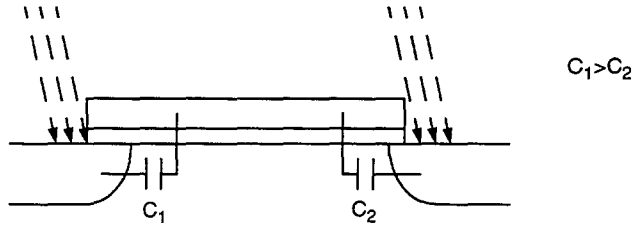


Figure 2.23 Origine de l'asymétrie des capacités

2.9 Le rapport signal à bruit

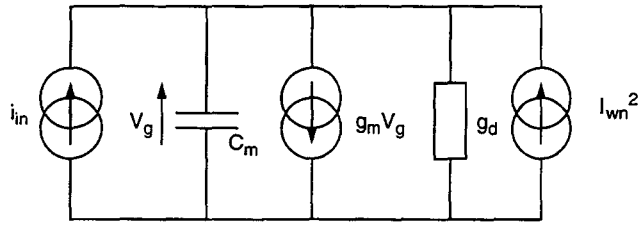


Figure 2.24 Modèle de bruit au premier ordre

En supposant que la seule source de bruit dans le circuit de premier ordre est une source de bruit blanc dans le transistor mémoire, comme l'illustre la Figure 2.24, la densité spectrale de puissance (PSD) de la source I_{wn} est donnée par

$$S_{I_{wn}} = \frac{8}{3}kTg_m \tag{2.73}$$

et la PSD de la tension de bruit ramenée sur la grille est donc donnée par

$$S_{V_{gwn}} = S_{I_{wn}} Z_d^{\,2} \tag{2.74}$$

où

$$Z_d = \frac{1}{sC_m + g_m + g_d} \tag{2.75}$$

A basse fréquence, et avec $g_d << g_m$,

$$S_{V_{gwn}} = \frac{8kT}{3g_m} \tag{2.76}$$

Pour un système passe-bas de premier ordre, la bande passante équivalente du bruit est

$$f_{NBW} = \frac{\pi}{2} f_{-3dB} = \frac{\pi \omega_p}{2 2\pi} = \frac{\omega_p}{4} \tag{2.77}$$

où

$$\omega_p = \frac{g_m}{C_m} \tag{2.78}$$

La puissance totale de la tension de bruit sur la grille du transistor mémoire est donc

$$P_{V_{gwn}} = \frac{2kT}{3C_m} \tag{2.79}$$

Nous pouvons écrire l'expression pour le rapport signal à bruit :

$$SNR = 10\log\left|\frac{P_0}{P_n}\right| \tag{2.80}$$

où

$$P_0 = \frac{i_0^2}{2} \tag{2.81}$$

$$P_n = g_m^2 P_{V_{gwn}} \tag{2.82}$$

Ce bruit continu est ramené dans le domaine échantillonné dans toute la bande de Nyquist, c'est-à-dire jusqu'à $f_s/2$, où f_s représente la fréquence d'échantillonnage. L'expression du SNR de (2.80) exprime ainsi le bruit de la cellule avec un signal d'entrée de bande passante égale à la fréquence de Nyquist.

Connaissant le SNR minimal, nous pouvons déterminer la puissance de bruit maximale, $P_{Vgwnmax}$. L'équation (2.79) impose alors une limite inférieure pour C_m :

$$C_m > \frac{2kT}{3P_{V_{gwnmax}}} \tag{2.83}$$

2.10 La dérive

Lors du mode de conservation ou de restitution, l'interrupteur reliant le drain et la grille du transistor mémoire est ouvert. Des courants de fuite des diodes de drain et de source passent du substrat vers les noeuds de source et de drain de l'interrupteur. De plus, l'interrupteur de compensation y contribue avec ses propres courants de fuite. Il en résulte que la tension sur la capacité de grille C_m dérive d'une valeur proportionnelle au temps de conservation, comme l'illustre la Figure 2.25.

Le courant total i_l qui traverse la capacité de mémorisation est donné par

$$i_l = 2i_{sbs(switch)} + i_{sbd(dummy)} + i_{sbs(dummy)} \tag{2.84}$$

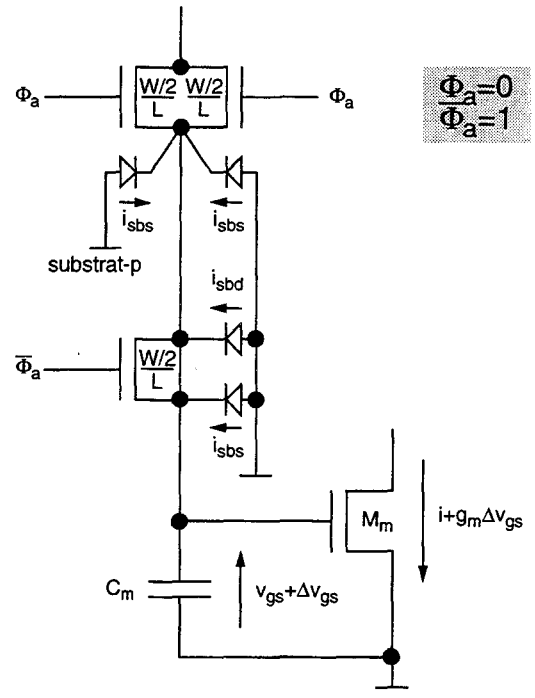


Figure 2.25 Origine de l'erreur de dérive

qui induit une variation de tension Δv sur la capacité, qui (normalisé à 1 seconde) est égale à

$$\Delta v = \frac{i_l}{C_m} \tag{2.85}$$

et l'erreur de dérive exprimée sous forme de courant mémorisé est donc

$$\Delta i = g_m \Delta v \tag{2.86}$$

Il est important de remarquer que cette quantité a pour dimension des A/s. En général le temps pendant lequel la cellule doit conserver sa tension de grille dépend de l'application. Les structures d'intégrateur utilisées dans les convertisseurs de données ou dans les filtres ont souvent un temps de conservation très court (une ou deux périodes d'horloge). Par contre, les lignes à retard peuvent avoir des temps de conservation très longs, où l'erreur due à la dérive peut être critique pour les performances du circuit.

Une autre quantité inconnue lors de la synthèse (et qui le reste même jusqu'à l'utilisation du circuit fabriqué) est la température de fonctionnement. En effet, les courants de fuite augmentent de façon exponentielle avec la température (ils doublent toutes les 10°C) : il est donc évident qu'une grande partie de l'erreur de recopie peut en provenir.

2.11 Stratégie de génération du point de départ

Les expressions et inégalités du premier ordre sont utilisées dans la procédure qui génère le point de départ (Figure 2.26). Les valeurs spécifiées pour la puissance et la tension d'alimentation maximales donnent la valeur maximale du courant de polarisation. Celle-ci est prise comme valeur effective, car ce choix donne la plus importante marge de manoeuvre pour calculer les autres paramètres.

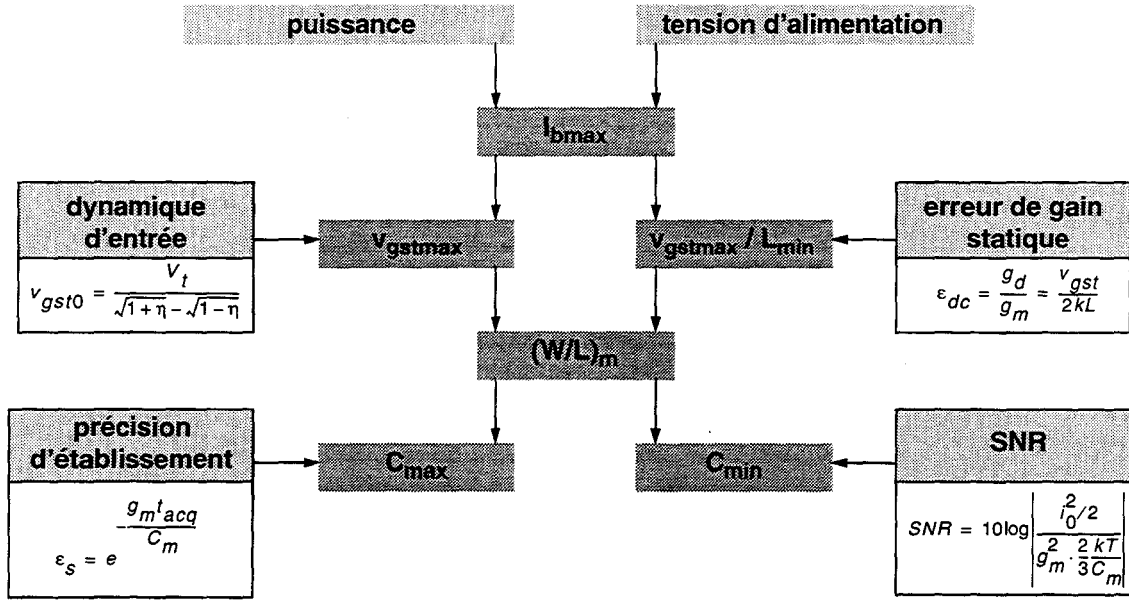


Figure 2.26 Procédure de génération du point de départ

Les expressions de la dynamique d'entrée et de l'erreur de gain statique donnent la valeur maximale de la tension de grille utile du transistor mémoire et la valeur minimale de sa longueur. Ces deux quantités permettent de calculer les dimensions minimales du transistor mémoire. De nouveau, celles-ci sont prises comme valeurs effectives, car ce choix donne la limite physique inférieure de la capacité de mémorisation. Les limites pour ce composant, définies par les performances à atteindre, sont calculées à partir des expressions de la précision d'établissement et du bruit. En cas de conflit, lorsque la valeur minimale de la capacité de mémorisation (physique ou définie par le bruit) est plus grande que la valeur maximale, la cellule est rejetée. Dans le cas contraire, elle est incluse dans la liste des topologies capables de satisfaire aux spécifications.

Les procédures explicites de dimensionnement (telles que celle-ci) ont souvent des cibles implicites d'optimisation. Dans une procédure pour la conception d'un amplificateur opérationnel, par exemple, les tailles minimales pour chaque transistor peuvent être choisies de manière systématique afin de réduire la surface. Si l'objectif principal est le fonctionnement à basse tension, cette approche est restrictive puisqu'une approche mieux adaptée minimiserait les tensions de grille utiles afin de permettre plus de dynamique - ce qui implique que les tailles des transistors sont importantes.

La procédure de génération du point de départ pour la cellule mémoire de courant possède aussi des heuristiques implicites dans le choix du courant de polarisation maximal et des dimensions minimales du transistor mémoire. Toutefois, ces choix ne satisfont pas à un objectif implicite d'optimisation, mais atteignent plutôt la marge de manoeuvre maximale dans la recherche des valeurs d'autres paramètres.

Il est intéressant de noter que, contrairement à ce que l'on pourrait supposer de manière intuitive, la diminution du courant de polarisation tend à augmenter la taille du transistor mémoire. Ceci est dû au fait que pour la même dynamique d'entrée et à un moindre courant de polarisation, la dynamique de la tension de grille augmente, aggravant ainsi le risque de désaturation. La solution est d'augmenter le rapport W/L afin de réduire la dynamique de la tension de grille et ainsi de diminuer le risque de désaturation (Figure 2.27).

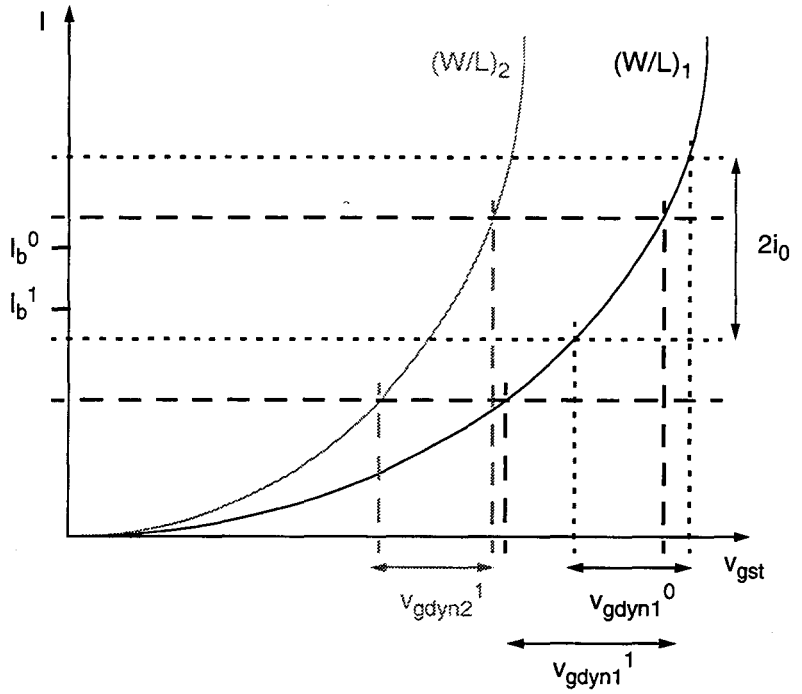


Figure 2.27 Influence de η sur la dynamique de la tension de grille

2.12 Approche qualitative de la conception

Les relations données précédemment peuvent être évaluées pour un jeu donné de variables de conception. En modifiant chaque variable individuellement, comme illustré en Annexe A, nous pouvons en tirer des déductions qualitatives concernant la conception itérative de la cellule de base.

Les variables de conception utilisées sont les tensions de grille utiles et les longueurs des transistors, la valeur de la capacité de mémorisation, le courant de polarisation et la tension d'alimentation. Les transistors ne sont pas optimisés sur leurs dimensions physiques (largeur et longueur), car ces variables ne sont pas indépendantes. Deux

paramètres intrinsèques aux dispositifs de grande importance sont la transconductance g_m et la conductance de sortie g_d. Leurs équations approximatives au premier ordre sont données par :

$$g_m = \sqrt{2\beta I_d} \tag{2.87}$$

$$g_d = \frac{I_d}{V_E} \tag{2.88}$$

En modifiant la largeur seulement, le courant I_d change, comme le paramètre de transconductance β (égal à $\mu C_{ox}W/L$). En variant la longueur seulement, I_d, β et la tension d'Early V_E changent simultanément. Il est donc difficile dans ces conditions de contrôler les performances du circuit, qui sont souvent en fonction de g_m et de g_d.

Cependant, g_m peut s'écrire

$$g_m = \frac{2I_d}{v_{gst}} \tag{2.89}$$

où v_{gst} est la tension de grille utile ("gate voltage overdrive" - GVO) v_{gs}-v_t. En jouant sur ce paramètre pour un courant de drain donné (et donc implicitement le rapport largeur sur longueur pour compenser), seule la valeur de la transconductance est modifiée. La conductance de sortie peut également être modifiée indépendamment de la transconductance en variant la longueur pour un courant donné (mettant donc implicitement à l'échelle la largeur de façon proportionnelle). Nous pouvons donc distinguer entre les *dimensions formelles*, qui représentent les variables indépendantes de conception utilisées par le processus de conception, et les *dimensions effectives*, qui représentent les dimensions physiques du circuit utilisées par les équations de conception pour évaluer les performances (Figure 2.28).

Il doit exister un lien explicite entre les dimensions formelles et effectives - toutefois, l'utilisation de la tension de seuil dans une des dimensions formelles complique la situation car cette dernière est une quantité qui dépend du modèle de dispositif. La longueur du transistor doit être attribuée de façon explicite, et la largeur peut ensuite être déterminée par itération.

Les évaluations ont été effectuées en utilisant le modèle MOS de Philips Model 9 pour la technologie ST 0.5µm. Il n'existe pas de raison de supposer que les espaces de conception changeront de manière importante avec un changement de technologie, puisque les caractéristiques sont celles du circuit. Il est néanmoins important de rappeler que l'image de l'espace de conception est loin d'être complète, car d'une complexité prohibitive. En effet, l'espace de conception ne peut montrer qu'une zone peu étendue autour du point nominal, qui a été sélectionné comme un point à partir duquel chaque critère de performance peut encore être amélioré. Le jeu complet des courbes est montré dans l'annexe A.

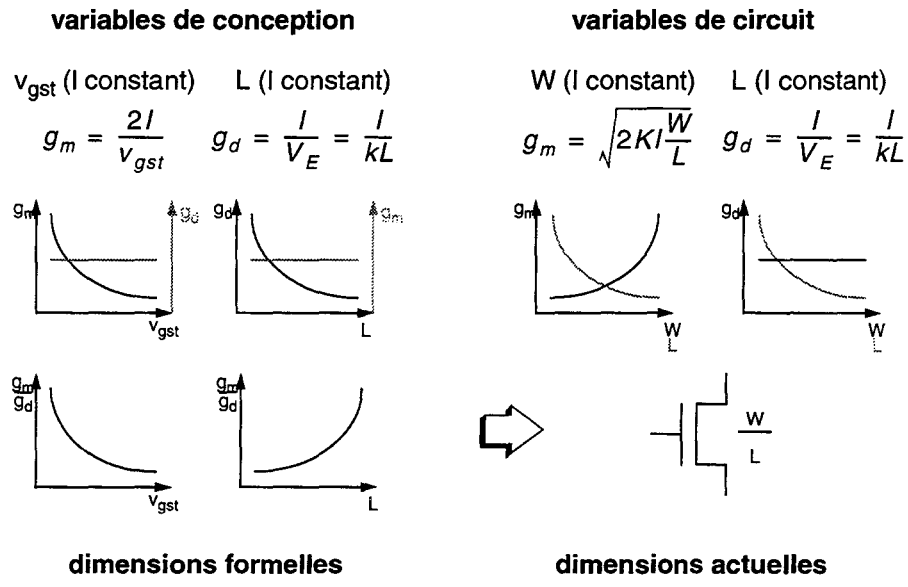


Figure 2.28 Variables d'optimisation

A partir des caractéristiques de l'espace de conception, nous pouvons déduire avec une confiance modérée des règles qualitatives d'heuristiques. Ces règles concernent donc l'amélioration de chaque performance individuelle par la variation des paramètres individuels de conception. Trois types d'heuristique peuvent être identifiés :

- *augmentation monotone*

  Lorsque la valeur de la variable augmente, la performance est améliorée, et ce pour n'importe quelle valeur de la variable. Par exemple, en augmentant la capacité de

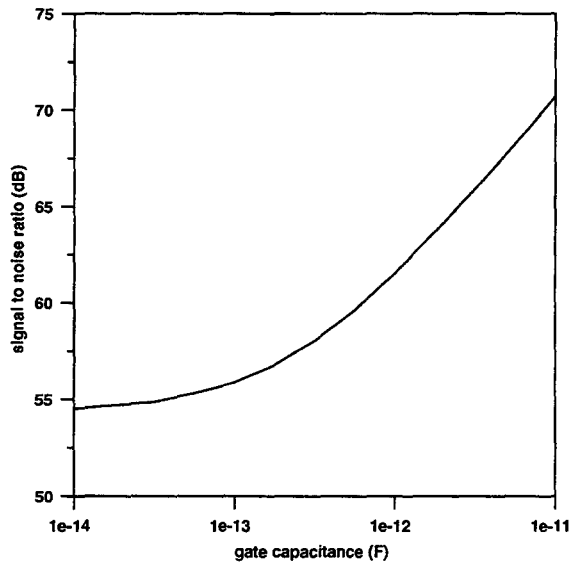


Figure 2.29 Variation du SNR avec la valeur de la capacité de mémorisation

grille, le rapport signal à bruit est également augmenté, comme l'illustre la Figure 2.29.

- *diminution monotone*

  Lorsque la valeur de la variable diminue, la performance est améliorée, et ce pour n'importe quelle valeur de la variable. Par exemple, en diminuant la tension de grille utile du transistor mémoire, l'erreur de gain statique est également diminuée, comme illustré dans la Figure 2.30.

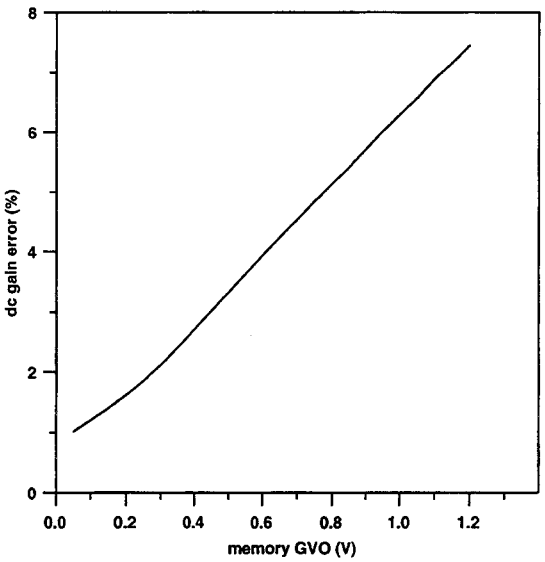


Figure 2.30 Variation de l'erreur de gain statique avec la GVO du transistor mémoire

- *non-monotone*

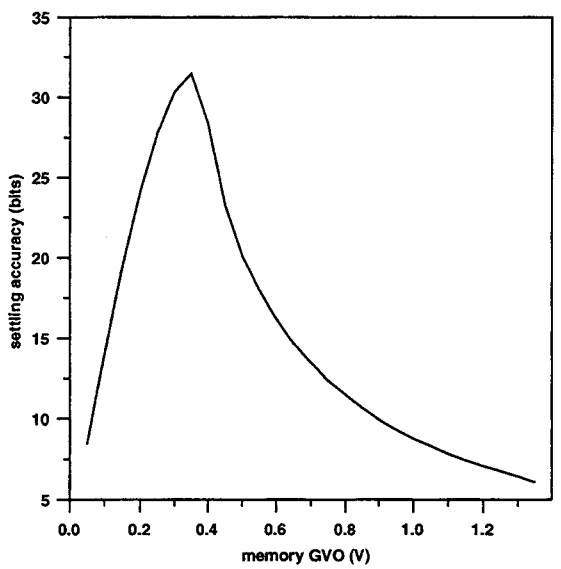


Figure 2.31 Variation de la précision d'établissement avec la GVO du transistor mémoire

L'espace de conception est non-monotone, ce qui signifie que pour certaines valeurs. de la variable, une *augmentation* améliore la performance, alors que pour d'autres valeurs de la variable, une *diminution* de sa valeur améliore la performance. Par exemple, du fait de la transition entre les comportements sur- et sous-amortis, la plupart des caractéristiques de la précision d'établissement sont non-monotones, comme illustré dans la Figure 2.31.

Un résumé de ces règles se trouve dans le Tableau 2.1.

GVO gate voltage overdrive ↗ augmentation monotone ↘ diminution monotone ✻ non-monotone effet mineur	erreur de gain statique	erreur de diviseur capacitif	précision d'établissement	injection de charge	snr	dérive	puissance	surface
GVO du transistor mémoire	↘	↗	✻	✻	↗	✻		↗
longueur du transistor mémoire	↗	↘						↘
GVO du transistor source	↗			↘				↗
longueur du transistor source	↗			↗				↘
longueur de l'interrupteur				↘		↘		↘
capacité de mémorisation		↗	✻	↗	↗	↗		↘
courant de polarisation	↗	↘	✻	✻	↘	↘	↘	↘
amplitude du courant d'entrée			↘	↗	↗			
tension d'alimentation				↘			↘	

Tableau 2.1 Règles qualitatives de conception pour la cellule mémoire de courant de base

L'*erreur de gain statique* est en grande partie réglée par la GVO du transistor mémoire, et par les longueurs des transistors mémoire et source. Ceci corrobore le modèle simplifié, qui pose que l'erreur de gain statique est définie par la transconductance du transistor mémoire (déterminée par sa GVO) ainsi que par la conductance globale de sortie (déterminée par les longueurs des deux transistors). Des effets secondaires sont également observés par rapport à la GVO du transistor source ainsi qu'au courant de polarisation. Les deux variations ont pour origine la conductance de sortie du transistor source, qui en réalité n'est pas entièrement indépendante de la GVO pour un courant donné, et vice versa. En effet, la conductance de sortie diminue légèrement pour une GVO croissante, et augmente légèrement pour un courant de drain croissant. Ces effets pourraient constituer des heuristiques de conception valables pour la base des connaissances, mais leur valeurs sont trop peu importantes pour permettre leur exploitation.

Quant à l'*erreur du diviseur capacitif*, celle-ci peut être réduite par l'augmentation du rapport capacité de mémorisation sur capacité parasite entre grille et drain. La dimension formelle de la capacité de grille C_m peut être augmentée à cette fin. Lors de la transcription de cette quantité en dimensions effectives du circuit, deux composantes doivent être prises en compte : la capacité complémentaire C_c, et la capacité provenant du transistor mémoire, qui comporte principalement sa capacité grille-substrat C_{gbm}. Cette dernière est fixée par la taille du transistor mémoire, qui est, à son tour, déterminée par les dimensions formelles de GVO et de L. La valeur de C_c est donc calculée comme

$$C_c = C_m - C_{gbm} \tag{2.90}$$

Si ce n'était pas le cas, les dimensions formelles ne représenteraient plus des variables indépendantes de conception. Si C_c était la transcription directe de C_m, nous aurions l'équation suivante pour l'erreur due au diviseur capacitif (en développant l'équation (2.32)) :

$$\varepsilon_{cap} \propto \frac{C_{gdm}}{C_{gdm} + (C_m + C_{gbm})} \tag{2.91}$$

où C_{gdm} représente la capacité grille-drain du transistor mémoire. Evidemment, C_{gdm} et C_{gbm} sont toutes les deux liées à la longueur L du transistor mémoire :

$$C_{gdm} \propto L \tag{2.92}$$

$$C_{gbm} \propto L^2 \tag{2.93}$$

Dans ces conditions, si $C_m >> C_{gbm}$, une augmentation en L augmentera la valeur de l'erreur. Si, par contre, $C_m << C_{gbm}$, l'inverse est vrai. L'heuristique reliant L à l'erreur du diviseur capacitif serait alors ambiguë. En découplant les contributions capacitives selon (2.90), cet effet est évité, et une heuristique claire peut être générée. En diminuant L, la valeur de C_{gbm} diminue également. Puisque L n'a plus d'effet sur la valeur de la capacité de grille *totale*, l'erreur due au diviseur capacitif diminue. Cependant, pour certaines conditions extrêmes, la capacité de grille réelle peut dépasser la dimension formelle C_m. Ceci se produit quand la valeur de C_m ou de la GVO du transistor mémoire est très faible, ou quand celle de L est très importante. Dans ces cas, la dépendance ambiguë ne peut pas être évitée.

Un autre moyen de réduire l'erreur est de réduire la sensibilité du courant aux modulations de la tension sur la grille, en réduisant la transconductance. Ceci peut s'effectuer en augmentant la GVO du transistor mémoire, ou en diminuant le courant de polarisation.

La *précision d'établissement* hérite son ambiguïté des régimes sous- et sur-amortis. Comme établi précédemment, le pole dominant est déterminé par g_m/C_m. Cette quantité, plutôt que les pôles de haute fréquence (qui dépendent des parasites), est

utilisée pour satisfaire aux spécifications. Il est évident, en examinant les courbes, que les trois dimensions formelles qui ont un effet majeur sur la précision d'établissement sont la capacité de grille, la GVO du transistor mémoire et le courant de polarisation. Les deux dernières quantités agissent sur la transconductance g_m du transistor mémoire.

Les paramètres principaux qui affectent l'erreur due à l'*injection de charge* sont la longueur de l'interrupteur (qui détermine la quantité de charge contenue dans le canal), la tension d'alimentation (qui détermine la variation de tension sur la grille de l'interrupteur, et par conséquent la composante de l'injection de charge provenant des capacités de recouvrement), la capacité de mémorisation (qui détermine la variation de tension sur le noeud de grille), la taille du transistor source (qui détermine la valeur de la capacité du noeud de drain) et la transconductance du transistor mémoire (qui détermine la sensibilité du courant aux variations de la tension de grille).

Nous pouvons constater que les paramètres de transconductance sont ambigus. Pour une GVO du transistor mémoire faible ou un courant de polarisation élevé (résultant en une transconductance élevée), la taille du transistor mémoire est grande. Sa contribution C_{gbm} à la capacité de grille totale dépasse, dans ces cas, la valeur de la dimension formelle de la capacité de grille. La réduction de la transconductance peut donc, sous ces conditions ($C_{gbm}>C_m$), réduire la capacité de grille totale et donc avoir l'effet opposé à celui souhaitée, à savoir que l'erreur de l'injection de charge est augmentée plutôt que diminuée. L'ambiguïté est également due à la capacité parasite du noeud de sortie. Quand la GVO du transistor mémoire est faible, cette capacité est importante, et attire plus de charge vers elle. En augmentant la GVO, la capacité du noeud de drain diminue et la charge augmente sur la capacité de grille. Encore une fois, une diminution de la GVO augmentera l'injection de charge jusqu'à un certain point, où la capacité de drain devient faible devant celle de la grille. Au-delà de ce point, l'effet principal de l'augmentation de la GVO est de diminuer la transconductance, et donc l'effet de l'injection de charge sur le courant de sortie.

Les heuristiques ont donc été choisies de façon à réduire la *cause* de l'injection de charge plutôt que son effet : c'est-à-dire que les heuristiques visent à réduire la variation de tension sur la grille, et excluent les heuristiques visant à modifier la transconductance.

Le *rapport de signal à bruit* peut être amélioré soit par l'augmentation du signal d'entrée, soit par la diminution de la puissance de bruit. Le moyen de loin le plus efficace de diminuer la puissance de bruit consiste à diminuer la bande passante de bruit par l'augmentation de la capacité de mémorisation. La bande passante peut également être réduite en diminuant la transconductance du transistor mémoire, ce qui diminue également le courant de bruit dans le transistor mémoire.

L'erreur de *dérive*, ayant la même origine que l'erreur de l'injection de charge, a des heuristiques similaires : une réduction dans la taille de l'interrupteur réduit la quantité de charge qui fuit sur la grille ; l'augmentation de la capacité de grille réduit la variation de tension sur ce noeud ; et une réduction de la transconductance réduit la sensibilité du

courant de sortie par rapport aux variations de tension sur la grille. La même ambiguïté concernant la taille du transistor mémoire réapparaît, due au rapport de transconductance et de la capacité de grille.

Les courbes de l'espace de conception ont été générées de façon automatique par l'évaluation du modèle analytique. Toutes les heuristiques ont été établies par inspection, c'est à dire de façon manuelle. La génération automatique des heuristiques nécessiterait un traitement des données afin d'évaluer les tendances dans les caractéristiques individuelles. De tels algorithmes devraient faire la distinction entre les incohérences numériques et les fonctions réellement non-monotones ; une fonction de seuil devrait être implémentée pour déterminer si une heuristique est mineure ou non ; et l'étendue des variables de conception utilisées devrait être suffisamment grande afin de s'assurer que l'image de l'espace de conception ne se limite pas à un faible volume autour d'un point nominal.

2.13 Conclusion

L'implémentation et le contenu du modèle analytique de la cellule de mémoire de courant de base ont été décrits. A partir des trois possibilités d'implémentation du modèle (macromodélisation, modélisation par HDL analogique ou par langage de programmation), la flexibilité et la disponibilité générale de l'approche du langage de programmation haut-niveau ont été préférées. Il a toutefois été noté que la prudence s'impose lors du développement des modèles afin d'assurer leur validité électrique et physique. Ensuite, nous avons déterminé que le modèle analytique doit comporter deux parties : un modèle détaillé qui utilise des modèles industriels de dispositifs afin d'obtenir une prédiction assez précise des performances de la cellule ; et un modèle simple basé sur des équations de dispositif de premier ordre afin de générer un point de départ valable. Ce dernier sert à réduire le nombre d'itérations et assiste la convergence d'un algorithme d'optimisation locale vers le minimum global.

Les modèles de dispositif de qualité sont critiques pour la précision de l'outil. Nous avons choisi d'implémenter sous forme autonome le modèle MOS Philips Model 9 qui, avec le modèle BSIM3v3, est le modèle de dispositif analogique submicronique le plus répandu. Rendre le modèle autonome signifie que pour un jeu donné de paramètres de dispositif (les dimensions et les conditions de polarisation), n'importe quel paramètre intrinsèque (courant de drain, transconductance, etc.) peut être calculé. Ce concept a été étendu pour les besoins de la CA, afin de permettre le calcul d'un paramètre de dispositif si les autres sont connus ainsi que la valeur d'un paramètre intrinsèque. Puisque le modèle MOS est complexe et ne permet pas l'inversion explicite des équations, la recherche implicite de la valeur du paramètre de dispositif est itérative, et plus précisément consiste en l'algorithme de bissection simple.

Cette introduction générale à la modélisation comportementale a été suivie par l'analyse des effet physiques qui déterminent les performances de la cellule. Chaque contribution à l'erreur d'échantillonnage a été analysée et modélisée en conséquence. Du point de vue du concepteur système, cette approche est insuffisante car aucun lien n'est fait entre les erreurs propres à la cellule et leur effet cumulatif sur les performances du système complet (c'est-à-dire l'offset, l'erreur de gain et la distorsion harmonique). D'un autre côté, il est difficile de prédire ces dernières avec précision, sans recourir à une analyse de distorsion de longue durée. Nous avons donc préféré favoriser la précision du calcul analytique des erreurs propres à chaque cellule. L'évaluation des performances globales du système permet de vérifier la pertinence des spécifications des cellules qui seront, le cas échéant, rajustées de façon itérative. Par exemple, si la distorsion harmonique après une analyse transitoire est évaluée trop importante, la spécification de l'injection de charge peut être contrainte. En effet, nous pouvons constater que la même structure à base de règles existe aussi bien au niveau système qu'au niveau cellulaire. Comme décrit dans le premier chapitre, l'outil requis pour la CA est un outil au niveau cellulaire, qui peut par la suite être piloté par un outil de CA au niveau système. Ce dernier déterminerait alors les spécifications de la cellule à partir des spécifications du système, et coordonnerait l'outil au niveau cellulaire par des exécutions itératives afin de satisfaire aux spécifications du concepteur.

Références

[BOY74] G.R. Boyle *et al.*, "Macromodeling of Integrated Circuit Operational Amplifiers", *IEEE Journal of Solid-State Circuits*, vol. SC-9, no. 6, pp. 353-363, December 1974

[CHE96] Y. Cheng *et al.*, "BSIM3v3 Manual", Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 1996

[DAU88] S.J. Daubert, D. Vallancourt, Y.P. Tsividis, "Current Copier Cells", *Electronics Letters*, vol. 24, no. 25, pp. 1560-1562, December 1988

[HO75] C.-W. Ho, A.E. Ruehli, P.A. Brennan, "The Modified Nodal Approach to Network Analysis", *IEEE Transactions on Circuits and Systems*, vol. CAS-22, pp. 502-509, June 1975

[IEE97] IEEE DASC 1076.1 Home Page, "Analog and Mixed-Signal Extensions to VHDL", *http://vhdl.vhdl.org/vi/analog*

[KER88] B.W. Kernighan, D.M. Ritchie, "The C Programming Language", 2nd Edition, Prentice-Hall, 1988

[LIP91] S.B. Lippman, "C++ Primer", 2nd Edition, Addison-Wesley, 1991

[MET96] Meta-Software, "HSPICE User's Manual", 1996

[MOE94] N. Moeneclaey, A. Kaiser, "Accurate Modelling of the Non-linear Settling Behaviour of Current Memory Circuits", *Proc. IEEE Symposium on Circuits and Systems*, CAD9.8, vol. 1, pp. 339-342, June 1994

[ROB96] E. Robilliart, "Dévéloppement de modèles non-quasi-statiques MOS et bipolaires", Thèse, Université des Sciences et Technologies de Lille, 1996

[SHI87] J.-H. Shieh, M. Patil, B.J. Sheu, "Measurement and Analysis of Charge Injection in MOS Analog Switches", *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 2, pp. 277-281, April 1987

[SIL96[F. Silveira, D. Flandre, P. Jespers, "A g_m/I_d Based Methodology for the Design of CMOS Analog Circuits", *IEEE Journal of Solid-State Circuits*, vol. 31, no. 9, pp. 1314-1319, September 1996

[TSI94] Y.P. Tsividis, K. Suyama, "MOSFET Modeling for Analog Circuit CAD: Problems and Prospects", *IEEE Journal of Solid-State Circuits*, vol. 29, no. 3, pp. 210-216, March 1994

[VEL95] R. Velghe, D. Klaassen, F. Klaassen, "MOS Model 9", Unclassified Report NL-UR 003/94, Philips Research Laboratories, Eindhoven, 1995

[WEG87] G. Wegmann, E.A. Vittoz, F. Rahali, "Charge Injection in Analog MOS Switches", *IEEE Journal of Solid-State Circuits*, vol. SC-22, no. 6, pp. 1091-1097, December 1987

3 LA MODÉLISATION DES CELLULES MÉMOIRE DE COURANT AMÉLIORÉES

La cellule mémoire de courant de base n'est pas appropriée à de nombreuses applications requierant de hautes performances. Dans la pratique, d'autres topologies sont souvent utilisées. L'outil de conception automatisée doit également pourvoir à ces besoins s'il doit être utilisé dans un environnement de conception réel. Les concepts développés dans le chapitre précédent sont de nouveau appliqués, afin de développer les modèles pour des cellules mémoire de courant améliorées. Ces modèles permettent la génération de règles qualitatives de conception, qui peuvent être utilisées dans l'optimisation à base de règles.

La cellule de mémoire de courant de base a de nombreux défauts qui limitent son utilisation dans des situations réelles de conception. La technique a été modifiée sur un certain nombre de points afin d'améliorer les performances de la cellule [NAI96]. Ces modifications comportent :

- la réduction de la conductance de sortie par l'utilisation des techniques de cascode ou de cascode actif [TOU90]. Cette approche réduit de façon importante les erreurs provenant des conductances de branches et de l'effet du diviseur capacitif. Cependant, la dynamique d'entrée est réduite et le temps d'établissement peut être plus long. Plus important, ces techniques n'ont pas d'effet sur l'injection de charge.
- la technique de la double boucle ("two-step") [HUG93]. Cette technique mémorise l'information de façon approximative dans la première boucle, et ensuite de façon précise dans la deuxième boucle, ce qui annule les erreurs d'acquisition de la première boucle. Puisque la dynamique de la deuxième mémorisation est bien moins importante que celle de la première, les erreurs deviennent presque indépendantes du signal. De ce fait, les erreurs peuvent être aisément filtrées à la sortie.
- l'opération des cellules en régime triode. Bien que la conductance de sortie soit plus élevée que dans les cellules en fonctionnement saturé, cette méthode peut néanmoins être avantageuse car la transconductance du transistor mémoire est indépendante de la tension de grille, et donc du signal. De ce fait, les erreurs d'injection de charge, qui dépendent de la transconductance, deviennent indépendantes du signal. Ces erreurs peuvent donc être enlevées par filtrage. Il est également avantageux d'utiliser le fonctionnement en zone triode lors de la diminution graduelle et généralisée des tensions d'alimentation. En effet, le fonctionnement des transistors en zone saturée dans cet environnement est de plus en plus difficile.
- les techniques de classe AB [BAT91]. Celles-ci relâchent les contraintes par rapport aux cellules de la classe A, où la dynamique maximale du signal ne peut dépasser le

niveau du courant de polarisation. Par l'utilisation d'une cellule convoyeur de courant, la consommation de courant est réduite.

Ces techniques peuvent être combinées afin d'effectuer une réduction globale de l'erreur d'échantillonnage. En effet, il existe plusieurs variantes topologiques de cellules mémoire de courant, et chacune requiert son propre modèle analytique. Il importe de noter à ce point une limite majeure à l'approche de la CA mettant en oeuvre des modèles analytiques. Une bibliothèque complète de cellules est difficile à constituer, car la génération d'un modèle pour chaque nouvelle topologie est une tâche de longue haleine qui nécessite des connaissances approfondies.

Nous avons choisi de limiter la bibliothèque aux techniques les plus communes, qui peuvent être classifiées dans les catégories suivantes :

- *les variantes structurelles*, où la structure interne, plutôt que le mode opératoire, est modifiée. Des exemples de ce type de variante sont les structures de cascode ou de cascode actif (simple boucle). Les variantes structurelles sont donc interchangeables, c'est-à-dire qu'une cellule peut remplacer une autre sans avoir à modifier le circuit externe de commande.
- *les variantes opérationnelles*, où le mode opératoire est modifié, par exemple les conditions de commutation dans la cellule de base à double boucle. En remplaçant la cellule de base à simple boucle par son équivalent à double boucle, le circuit externe de commande doit être modifié, bien que la complexité interne de la cellule soit presque identique au nombre d'interrupteurs près.

A l'aide des connaissances formelles générées pour ces variantes, l'extension de la bibliothèque afin d'inclure des combinaisons de ces variantes doit être relativement simple. Par exemple, une cellule de mémoire de courant qui utilise la technique du cascode actif et de la double boucle utilisera une combinaison d'équations générées pour les cellules qui utilisent ces mêmes techniques de façon individuelle. Les échantillonneurs qui fonctionnent en mode triode devraient également présenter peu de difficultés car la structure reste identique, seule la zone de fonctionnement étant modifiée.

3.1 Modèle de la cellule mémoire de courant à structure cascode

Afin de réduire les variations de la tension de drain du transistor mémoire (et donc les erreurs de gain statique et de diviseur capacitif) en réduisant la conductance de sortie, une branche cascode est utilisée, comme illustré dans la Figure 3.1. La conductance est effectivement réduite par un facteur équivalent au gain du transistor cascode, mais la dynamique d'entrée est réduite et le comportement d'établissement est plus complexe du fait du nombre plus important de noeuds.

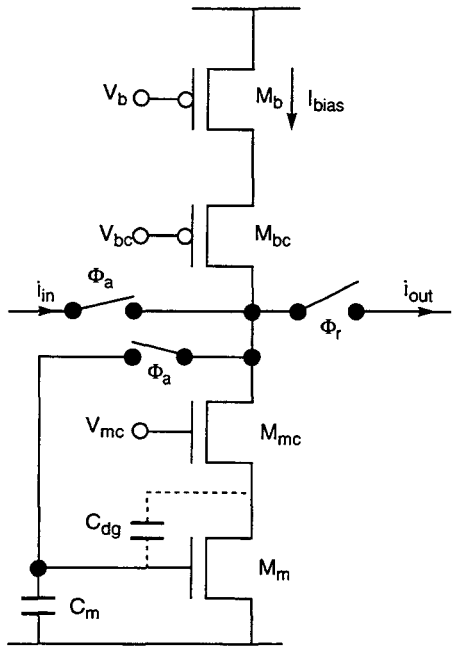


Figure 3.1 La cellule mémoire de courant cascode

Dans les analyses suivantes, seules celles qui nécessitent des modifications par rapport à la cellule de base sont décrites. La topologie cascode possède la même structure de commutation que la cellule de base, et n'a donc pas d'effet sur les erreurs qui proviennent des caractéristiques des interrupteurs, c'est-à-dire l'injection de charge et la dérive. Pour les mêmes tailles de l'interrupteur et de capacité, la même variation en tension sera observée sur la grille que dans la cellule de base. Puisque la transconductance de la branche cascode est approximativement la même que celle d'un transistor seul, aucune réduction significative n'est apparente concernant ces erreurs. De ce fait, l'injection de charge et la dérive sont évaluées de la même manière que pour la cellule de base.

3.1.1 Solution DC

Alors que le calcul du point de polarisation était simple pour la cellule mémoire de courant de base, il n'en va pas de même pour la configuration cascode. En effet, il existe de nombreuses situations où les calculs du point de polarisation sont effectués :

- *le type de dimensions spécifiées* : effectives (largeur et longueur) issues des équations analytiques de performance ; ou formelles (v_{gst} et L) issues des fonctions de transformation ou de la génération du point de départ,
- *la quantité connue* : par exemple, la valeur du courant est connue, et la tension de grille correspondante doit être trouvée ; ou à l'inverse la valeur de la tension de grille est connue et la valeur du courant doit être trouvée,
- *la configuration* : la structure cascode est en configuration diode (mode acquisition) avec la grille du transistor mémoire reliée au drain du transistor cascode ; ou bien la tension de drain est fixe (mode restitution) par une source externe.

Toutes les combinaisons ne sont pas utilisées : par exemple, des dimensions formelles ne peuvent pas être fournies si la valeur du courant n'est pas connue.

Le besoin d'une fonction spécifique est donc clair. Cette fonction doit être capable de calculer le point de polarisation pour une branche cascode N ou P (Figure 3.2) étant donné un jeu spécifique de conditions. L'algorithme utilisé pour la solution DC est illustré dans la Figure 3.3. Le principe de l'algorithme est de fixer le paramètre le plus sensible (V_{ds1}) et les tensions des autres noeuds en découlent implicitement. L'algorithme peut alors recalculer V_{ds1} de façon explicite, et reboucler ainsi tant qu'il existe une différence entre des cycles successifs.

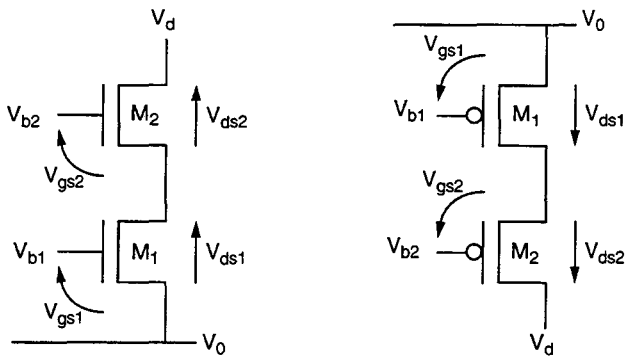


Figure 3.2 Configurations cascode des branches N et P

3.1.2 Transistor équivalent à la structure cascode

Le circuit équivalent petit-signal pour la branche cascode permet le développement limité d'un circuit équivalent identique à celui d'un transistor unique, comme illustré dans la Figure 3.4. Ceci simplifie considérablement les analyses suivantes du circuit, car la branche peut être vue comme un transistor unique équivalent.

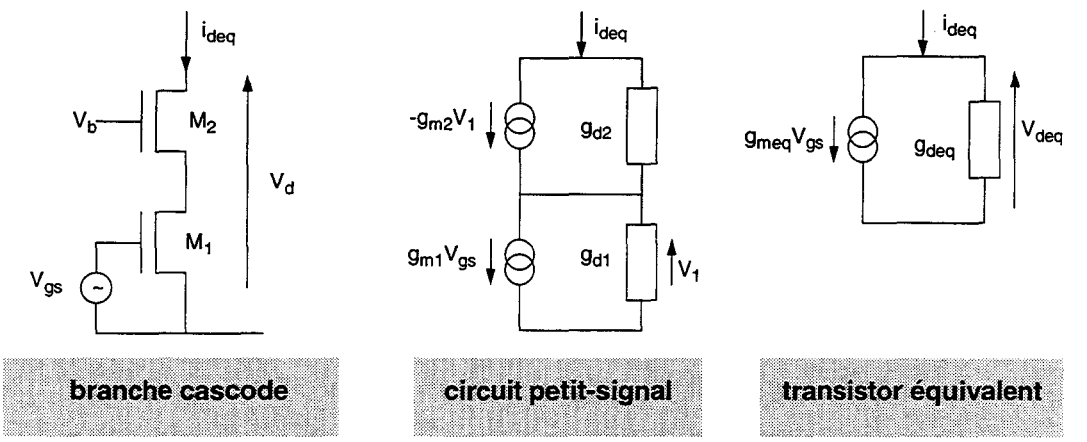


Figure 3.4 Approche du transistor équivalent pour la branche cascode

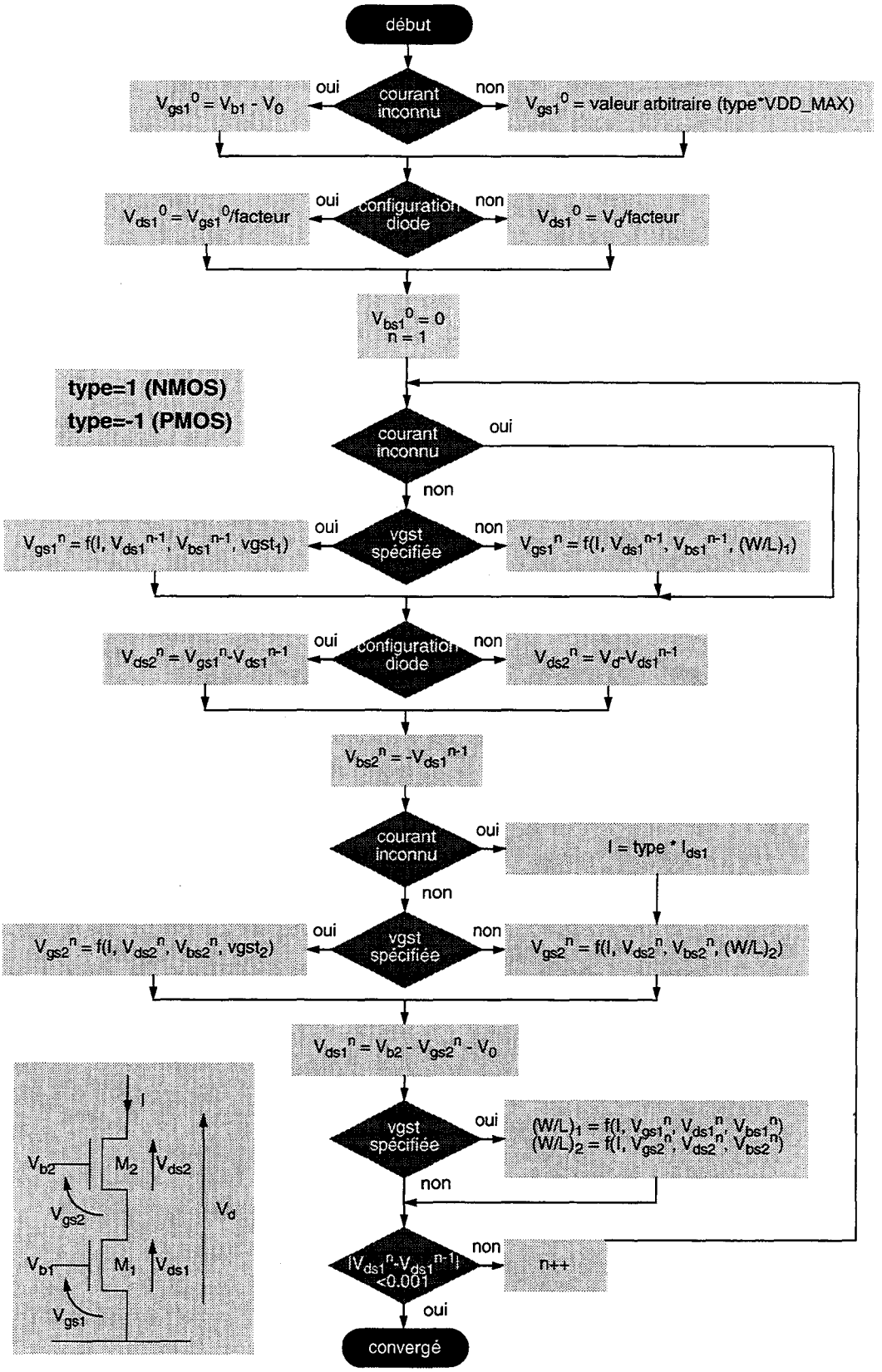


Figure 3.3 Solution DC pour la branche cascode

Les expressions de la transconductance et de la conductance équivalentes peuvent s'écrire facilement :

$$g_{meq} = \frac{g_{m1}(g_{m2} + g_{d2})}{g_{m2} + g_{d1} + g_{d2}} \approx g_{m1} \tag{3.1}$$

$$g_{deq} = \frac{g_{d1}g_{d2}}{g_{m2} + g_{d1} + g_{d2}} \approx \frac{g_{d1}g_{d2}}{g_{m2}} \tag{3.2}$$

L'équation (3.2) montre que la conductance de sortie équivalente de la branche cascode est égale à la conductance de sortie du transistor mémoire divisée par le gain du transistor cascode :

$$g_{deq} \approx \frac{g_{d1}}{A_2} \tag{3.3}$$

où

$$A_2 = \frac{g_{m2}}{g_{d2}} \tag{3.4}$$

Les erreurs associées à la conductance de sortie (erreur de gain statique, erreur due au diviseur capacitif) sont donc réduites d'un facteur équivalent au gain du transistor cascode. Les sections suivantes décrivent en détail les effets positifs et négatifs de cette modification structurelle.

3.1.3 Dynamique d'entrée

La structure cascode nécessite un certain nombre de conditions à son fonctionnement en zone saturée. Une seule condition pour la dynamique d'entrée peut être trouvée par l'analyse des conditions limitatives.

L'évaluation de la dynamique d'entrée nécessite un circuit d'analyse similaire à celui utilisé lors de l'analyse de la cellule de base (Figure 3.5).

Comme précédemment,

$$v_{g1} = V_t + \sqrt{\frac{2I_b}{\beta}}\sqrt{1+\eta} \tag{3.5}$$

$$v_{g2} = V_t + \sqrt{\frac{2I_b}{\beta}}\sqrt{1-\eta} \tag{3.6}$$

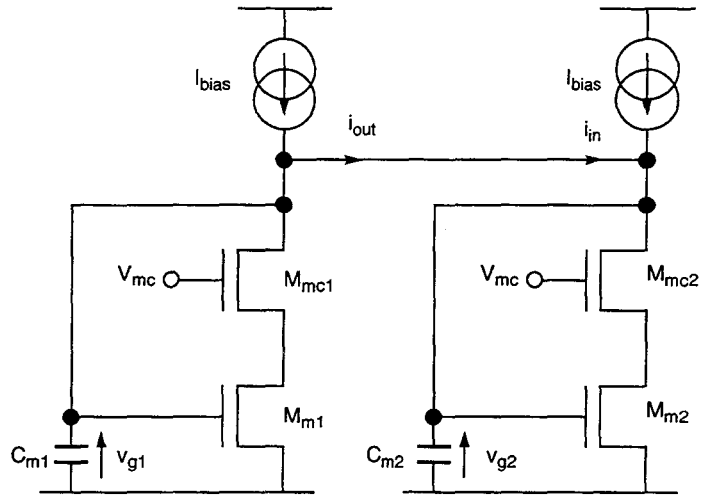


Figure 3.5 Paire de cellules communicantes pour l'analyse de la dynamique d'entrée

A la limite de saturation du transistor mémoire, nous avons

$$v_{dm1} = v_{g1} - V_t \tag{3.7}$$

$$v_{dmc1} = V_{mc} - v_{dm1} - V_{tmc1} \tag{3.8}$$

Puisque

$$v_{g2} = v_{dm1} + v_{dmc1} \tag{3.9}$$

nous pouvons écrire que

$$v_{g1} - v_{g2} = V_t - (V_{mc} - v_{dm1} - V_{tmc1}) = V_t - v_{gstmc} \tag{3.10}$$

Donc la valeur maximale de v_{gst0} est donnée par

$$v_{gst0} < \frac{V_t - v_{gstmc}}{(\sqrt{1+\eta} - \sqrt{1-\eta})} \tag{3.11}$$

En comparant ce résultat avec celui de la cellule de base (2.15), il est évident que la dynamique d'entrée est réduite par un facteur de la tension de grille utile du transistor cascode. Le compromis principal se situe entre la dynamique d'entrée (et donc la tension de grille utile minimale du transistor cascode) et la surface minimale. En effet, pour diminuer la tension de grille utile, à courant constant, il est nécessaire d'augmenter le rapport de la largeur sur la longueur (donc la surface) du transistor cascode.

3.1.4 L'erreur de gain statique

Grâce au transistor cascode, une variation de tension ΔV_d sur le drain du cascode apparaît à la grille du transistor mémoire réduit par le gain du transistor cascode. Ceci constitue l'avantage principal de la structure cascode (comme de la structure cascode actif).

3.1.4.1 Modèle au premier ordre

En utilisant le transistor équivalent à la structure cascode (cette approche est valable pour les calculs en régime statique), l'erreur de gain statique peut s'écrire :

$$\varepsilon_{dc} = \frac{1}{1+\frac{g_{meq}}{g_{deq}}} \tag{3.12}$$

En reprenant les approximations de la transconductance et de la conductance de sortie équivalentes développées dans la section 3.1.2,

$$g_{meq} \approx g_{mm} \tag{3.13}$$

$$g_{deq} \approx \frac{g_{dm}g_{dc}}{g_{mc}} \tag{3.14}$$

nous obtenons

$$\varepsilon_{dc} = \frac{1}{1+\frac{g_{mm}g_{mc}}{g_{dm}g_{dc}}} = \frac{1}{1+\frac{4V_{Em}V_{Ec}}{v_{gstm}v_{gstc}}} \tag{3.15}$$

3.1.4.2 Modèle détaillé

Comme pour la cellule de mémoire de courant de base, l'erreur de gain statique est calculée de façon simple en polarisant les transistors mémoire et source, et en évaluant ensuite la différence entre les courants générés. Les tensions nécessaires pour la polarisation de chaque transistor ont été calculées par la solution DC.

3.1.5 Diviseur capacitif

L'erreur dynamique induite par une variation sur le noeud de sortie est réduite par rapport à celle de la cellule de base. Cette réduction a les mêmes origines que pour celle de l'erreur de gain statique. En effet, la variation sur le noeud de sortie ΔV_d, telle qu'elle apparaît sur le drain du transistor mémoire, équivaut à ΔV_d réduite d'un facteur équivalent au gain du transistor cascode (Figure 3.6).

Nous recherchons la variation ΔV_g qui correspond à la variation ΔV_d au drain du transistor cascode. Cette dernière provoque une variation de tension ΔV_s au drain du transistor mémoire que nous pouvons trouver en évaluant la variation de courant statique Δi dans la branche cascode. Nous avons

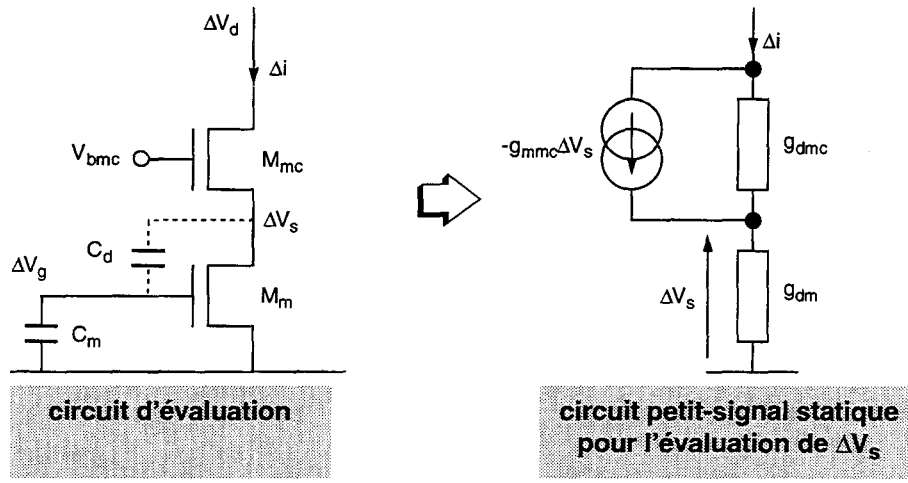


Figure 3.6 Erreur du diviseur capacitif sur la branche cascode

$$\Delta i = \Delta v_d g_{deq} \approx \Delta v_d \frac{g_{dm}}{A_{mc}} \tag{3.16}$$

Δi induit un changement de tension au drain du transistor mémoire

$$\Delta v_s = \frac{\Delta i}{g_{dm}} = \frac{\Delta v_d}{A_{mc}} \tag{3.17}$$

La tension de grille subit donc une modification dynamique

$$\Delta v_g = \frac{C_d}{C_d + C_m} \cdot \frac{\Delta v_d}{A_{mc}} \tag{3.18}$$

et la modification dynamique de la valeur de courant est donc

$$\Delta i = \left(\frac{C_d}{C_d + C_m} \cdot \frac{\Delta v_d}{A_{mc}}\right) g_{mm} \tag{3.19}$$

3.1.6 Précision d'établissement

Le développement utilisé pour générer l'équation (2.37) peut être employé de nouveau pour le modèle du transistor équivalent. Les expressions contiennent les paramètres du transistor équivalent à la place de ceux du transistor mémoire. Les équations sont rappelées ci-dessous :

$$H(s) = K \frac{1}{1 + \tau_1 s + \tau_2 s^2} \tag{3.20}$$

où

$$K = \frac{g_{sb}}{(g_{meq} + g_0)(Y_{in} + g_{sb}) + Y_{in}g_{sb}} \tag{3.21}$$

$$\tau_1 = \frac{C_m((g_{sa} + g_0)(Y_{in} + g_{sb}) + Y_{in}g_{sb})}{g_{sa}((g_{meq} + g_0)(Y_{in} + g_{sb}) + Y_{in}g_{sb})} \tag{3.22}$$

$$\tau_2 = \frac{C_m C_d (Y_{in} + g_{sb})}{C_m((g_{sa} + g_0)(Y_{in} + g_{sb}) + Y_{in}g_{sb})} \tag{3.23}$$

et

$$g_0 = g_{dmeq} + g_{dbeq} \tag{3.24}$$

Ce jeu d'équations rapproche la cellule cascode d'un système de second ordre. Cette hypothèse n'est pas rigoureusement valable car le noeud intermédiaire entre le transistor mémoire et le transistor cascode introduit un pôle supplémentaire et rend la fonction de transfert de type troisième ordre. Cependant, l'élévation de l'ordre de deux à trois augmente la complexité de l'analyse dans le domaine temporel de plusieurs ordres de grandeur. En ce qui concerne la CA, cette situation n'est pas souhaitable car la précision n'en sera que peu augmentée. Bien qu'une évaluation précise soit nécessaire pour la simulation, la synthèse ne requiert qu'une estimation. Comme évoquée dans la section 2.1.2, si les modèles analytiques étaient d'une précision extrême, ils seraient trop onéreux à évaluer. Par conséquent, un compromis entre la précision et la rapidité d'évaluation est atteint par la limitation des modèles analytiques à certains points.

Afin d'assurer la validité de l'expression de deuxième ordre, nous pouvons contraindre le pôle, due au transistor cascode, à rester à haute fréquence. L'effet de ce pôle devient alors négligeable, car trop loin des autres pôles pour modifier le comportement de type deuxième ordre.

Le comportement prédit pour l'établissement contiendra néanmoins une erreur, de valeur plus ou moins importante, dépendante de la position du pôle au noeud intermédiaire par rapport aux autres pôles. Par contre, cette erreur sera bien évaluée durant la phase de simulation numérique et corrigée par les évaluations suivantes des performances.

Comme dans la cellule de base, les variations grand signal sont prises en compte par le modèle linéaire par segments du temps d'établissement, et une expression d'enveloppe est utilisée.

3.1.7 Bruit

De la Figure 3.2, nous pouvons développer un circuit équivalent (Figure 3.7) afin d'analyser la contribution de bruit de chaque transistor.

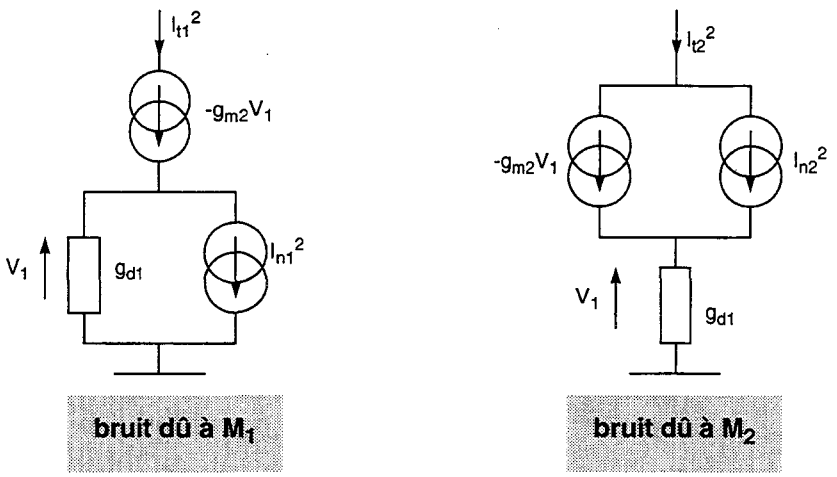


Figure 3.7 Les contributions individuelles en bruit des transistors

Le système suivant d'équations résulte d'une analyse de ces circuits :

$$I_{t1}^2 = \frac{I_{n1}^2}{\left(1+\frac{g_{d1}}{g_{m2}}\right)^2} \approx I_{n1}^2 \tag{3.25}$$

$$I_{t2}^2 = \frac{I_{n2}^2}{\left(1+\frac{g_{m2}}{g_{d1}}\right)^2} \approx 0 \tag{3.26}$$

Nous constatons que la plupart du bruit dans la configuration de cascode provient du transistor mémoire. Le bruit total dans la cellule de mémoire de courant cascode est la somme des contributions des transistors dans les branches N et P. Les expressions exactes sont utilisées dans les équations analytiques d'évaluation de performances, et les expressions approximatives dans le fonction de génération du point de départ.

3.1.8 Approche qualitative de la conception

Par une analyse quantitative similaire à celle décrite dans le chapitre précédent, un jeu de règles qualitatives de conception peut être développé pour la cellule mémoire de courant cascode, résumé dans le Tableau 3.1.

L'*erreur de gain statique* est, comme pour la cellule de base, fortement influencée par la transconductance de la branche mémoire, qui est principalement définie par la GVO du transistor mémoire. La conductance de sortie globale est également un facteur majeur, et un certain nombre de dimensions formelles interviennent dans la modification de cette quantité. Une caractérisation montre que les paramètres principaux qui affectent la

GVO	gate voltage overdrive
↗	augmentation monotone
↘	diminution monotone
✻	non-monotone
	effet mineur

	↘ erreur de gain statique	↘ erreur du diviseur capacitif	↗ précision d'établissement	↘ injection de charge	↗ snr	↘ dérive	↘ puissance	↘ surface
GVO du transistor mémoire	↘	↗	↘	↗	↗	↗		↗
longueur du transistor mémoire	↗	↘	↘	✻				↘
GVO du transistor cascode mémoire	↘	↘						↗
longueur du transistor cascode mémoire	↗	↗	↗	↗				↘
GVO du transistor source	↘				↗			↗
longueur du transistor source	↗							↘
GVO du transistor cascode source	↘		✻	↘				↗
longueur du transistor cascode source	↗		✻	↗				↘
longueur de l'interrupteur				↘		↘		↘
capacité de mémorisation		↗	↘	↗	↗	↗		↘
courant de polarisation		↘	✻	↘	↘	↘	↘	↘
amplitude du courant d'entrée			↘	↗	↗			
tension d'alimentation				↘			↘	

Tableau 3.1 Règles qualitatives de conception pour la cellule mémoire de courant cascode

conductance sont les GVOs des transistors cascodes des deux branches. Ces quantités affectent les valeur des transconductances, auxquelles les conductances de sortie des branches sont inversement proportionnelles.

Mises à part les heuristiques qui influencent le transistor cascode dans la branche mémoire, l'*erreur du diviseur capacitif* peut être modifiée de la même façon que pour la cellule de base. Les heuristiques du transistor cascode augmentent le gain de ce transistor, ce qui tend à réduire la variation de tension sur le drain du transistor mémoire.

De façon similaire, les heuristiques pour la *précision d'établissement* comportent des modifications au niveau des transistors cascodes. Ces transistors modifient la capacité parasite du noeud de sortie. Comme cette quantité ne dépend plus du transistor mémoire, comme dans la cellule de base, les heuristiques qui concernent son dimensionnement deviennent moins ambiguës. En effet, l'expérience montre qu'une heuristique monotone peut être utilisée pour la GVO et pour la capacité de grille.

Les heuristiques principales pour l'optimisation de l'*injection de charge* sont identiques à celles de la cellule de base. Parmi les heuristiques secondaires, la capacité parasite du noeud de sortie peut être modifiée sans conséquence sur la transconductance du transistor mémoire ou sur la capacité de grille. De ce fait, l'heuristique du GVO du transistor mémoire, ambiguë dans la cellule de base, peut devenir monotone. D'autres heuristiques qui modifient la valeur de la capacité du noeud de sortie se trouvent au niveau des transistors cascodes.

Les mêmes heuristiques que celles de la cellule de base peuvent être utilisées pour le *SNR*. Une heuristique supplémentaire est celle qui augmente la GVO du transistor source, ce qui réduit le bruit qui en provient.

Les heuristiques concernant l'*erreur de dérive* sont également identiques, à l'exception de la conversion des heuristiques ambiguës en heuristiques monotones.

3.2 Modèle de la cellule mémoire de courant à structure cascode actif

La structure du cascode actif (Figure 3.8) réduit la sensibilité aux variations de tension sur le noeud de sortie de façon plus importante que la structure cascode. Cependant, de la même manière que dans la structure cascode, cet avantage est acquis au prix d'une dynamique d'entrée et d'une précision d'établissement réduites.

A partir des modèles développés pour les cellules mémoire de courant de base et à structure cascode, des similarités peuvent être identifiées entre le comportement du circuit pour la cellule à cascode actif, et les cellules décrites précédemment. Une fonction similaire à celle développée dans la section 3.1.1 a été générée afin de résoudre le point de polarisation statique dans la branche du cascode actif. Cette fonction nécessite des informations supplémentaires : le courant de branche de l'amplificateur et les dimensions de ses transistors. De nouveau, une fonction de transfert du second ordre est utilisée dans le calcul de la précision d'établissement. Puisque le comportement réel est encore plus complexe que celui de la structure cascode (du fait d'un noeud supplémentaire), l'erreur induite par l'utilisation d'une expression de second ordre est d'autant plus importante. De plus, aucune modification n'est apportée à la structure de l'interrupteur, donc les erreurs dues à l'injection de charge et à la dérive restent presque identiques à celles de la cellule mémoire de courant de base, et sont évaluées en conséquence.

3.2.1 Le transistor équivalent à la structure cascode actif

Les paramètres petit-signal pour le transistor équivalent peuvent être déterminées pour la structure de cascode actif, ce qui simplifie considérablement l'analyse du circuit. La transformation du circuit au niveau transistor vers un schéma équivalent petit-signal du transistor est décrite dans la Figure 3.9. Cette transformation est en fait identique à celle du circuit équivalent de la structure cascode, avec le gain du transistor cascode M_2

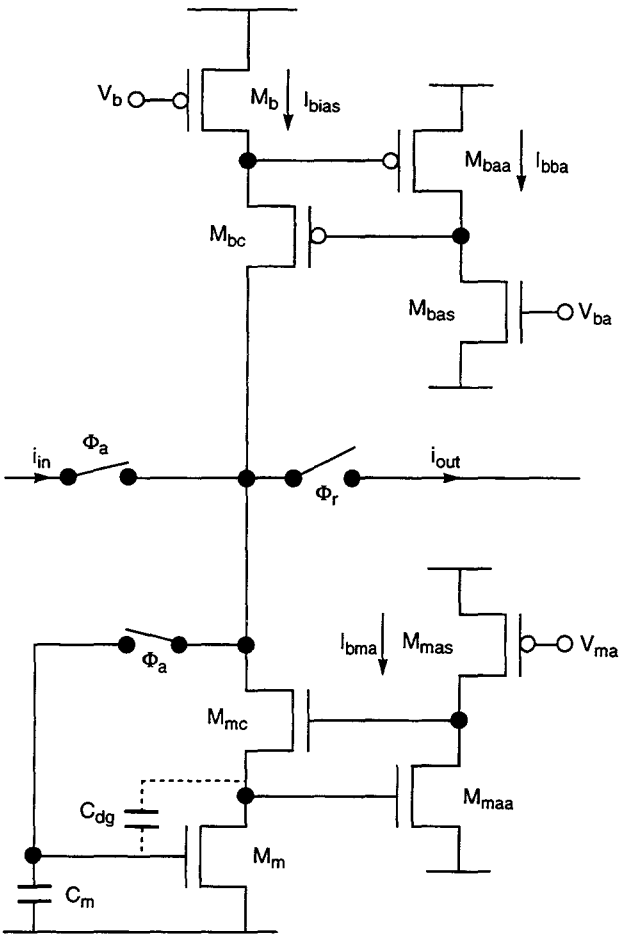


Figure 3.8 Cellule mémoire de courant à structure cascode actif

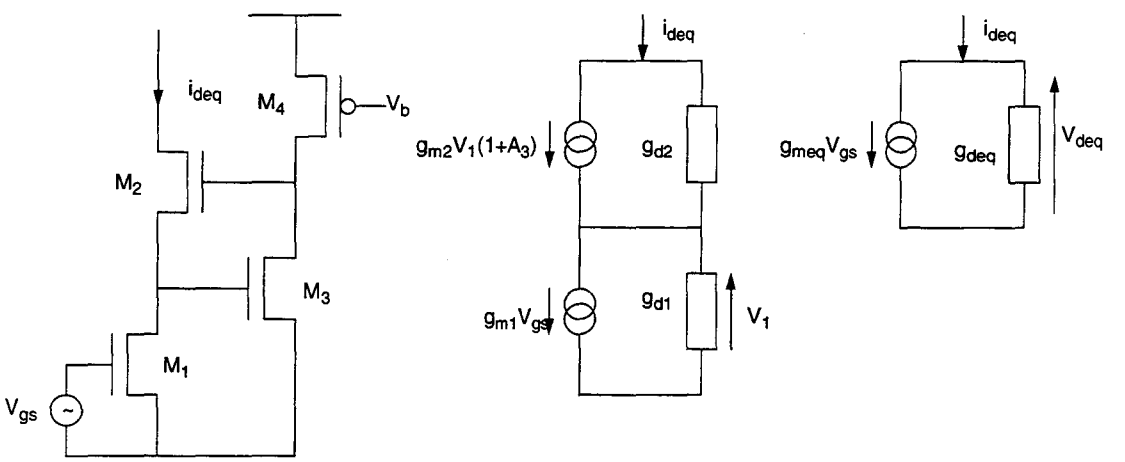


Figure 3.9 Approche du transistor équivalent pour la branche cascode actif

augmenté du gain de l'amplificateur actif inverseur, représenté par A_3. Cette quantité peut être calculée comme étant le gain du transistor M_3 avec M_4 comme charge :

$$A_3 = \frac{g_{m3}}{g_{d3} + g_{d4}} \tag{3.27}$$

Les valeurs de la transconductance et la conductance de sortie équivalentes peuvent facilement être trouvées, si A_3>>1,

$$g_{meq} \approx g_{m1} \tag{3.28}$$

$$g_{deq} \approx \frac{g_{d1}}{A_2 A_3} \tag{3.29}$$

L'avantage de la structure du cascode actif est de présenter une conductance de sortie fortement réduite. Cette propriété réduit encore les erreurs associées à cette caractéristique. Toutefois, comme le démontre la section suivante, la dynamique d'entrée subit une réduction.

3.2.2 La dynamique d'entrée

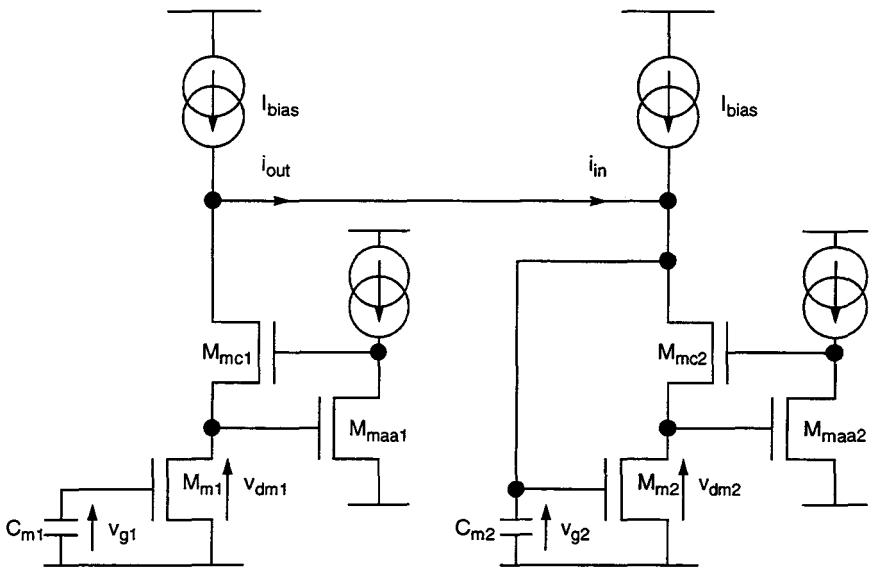


Figure 3.10 Restoring-acquiring cell pair for input dynamic range analysis

La valeur maximale de v_{gst0} développée dans la section 3.1.3 n'est pas modifiée, puisque la structure du cascode reste identique.

$$v_{gst0} < \frac{V_t - v_{gstmc}}{(\sqrt{1+\eta} - \sqrt{1-\eta})} \tag{3.30}$$

Cependant, pour que l'amplificateur fonctionne correctement, la tension de drain du transistor mémoire v_{dm} doit être supérieure à la tension de seuil du transistor de l'amplificateur M_{maa}, afin d'assurer que ce dernier fonctionne en forte inversion. A la limite de saturation du transistor mémoire, ceci impose une valeur minimale sur la valeur absolue de la tension de grille:

$$v_g > 2V_t \tag{3.31}$$

Un tel jeu de conditions est extrêmement restrictif, et illustre un problème majeur de cette structure. Lorsque la tension d'alimentation diminue (alors que la tension de seuil reste identique), la marge de manoeuvre lors de la conception devient de plus en plus limitée. Finalement, il ne sera plus possible d'utiliser la structure cascode actif.

3.2.3 L'erreur de gain statique

L'équation (3.12) peut de nouveau être utilisée avec les expressions correspondantes de la transconductance et de la conductance de sortie équivalentes. L'expression approximative est donc

$$\varepsilon_{dc} = \frac{1}{1+\dfrac{g_{mm}A_2A_3}{g_{dm}}} = \frac{1}{1+\dfrac{g_{mm}g_{mmc}}{g_{dm}g_{dmc}}\dfrac{g_{mmaa}}{g_{dmaa}+g_{dmas}}} = \frac{1}{1+\dfrac{8V_{Em}V_{Emc}V_{Emaa}V_{Emas}}{v_{gstm}v_{gstmc}v_{gstmaa}(V_{Emaa}+V_{Emas})}} \tag{3.32}$$

L'évaluation détaillée de l'erreur de gain se fait de la même manière que pour les cellules mémoire de courant de base et à structure cascode.

3.2.4 L'erreur du diviseur capacitif

Par substitution du terme équivalent à la conductance de sortie de la structure cascode actif dans l'équation (3.16), la modification du courant de sortie pour le modèle simple est donnée par

$$\Delta i = \left(\frac{C_d}{C_d+C_m}\frac{\Delta v_d}{A_{mc}A_{maa}}\right)g_{mm} \tag{3.33}$$

3.2.5 Bruit

Les approximations exposées dans les équations (3.25) et (3.26) pour la structure du cascode sont plus proche des expressions précises dans le cas de la structure cascode actif. L'erreur d'approximation est réduite d'un facteur équivalent au gain de l'amplificateur actif :

$$I_{t1}^2 = \frac{I_{n1}^2}{\left(1+\dfrac{g_{d1}}{g_{m2}A_{maa}}\right)^2} \approx I_{n1}^2 \tag{3.34}$$

$$I_{t2}^2 = \frac{I_{n2}^2}{\left(1 + \frac{g_{m2}A_{maa}}{g_{d1}}\right)^2} \approx 0 \qquad (3.35)$$

Le bruit de l'amplificateur formé par les transistors M_3 et M_4 (Figure 3.9) doit également être pris en compte. Dans la Figure 3.11, la source I_{na} représente les sources de bruit combinées de ces transistors. De même, la conductance g_{da} est la conductance de sortie globale de l'amplificateur qui résulte de la combinaison des conductances de sortie des deux transistors.

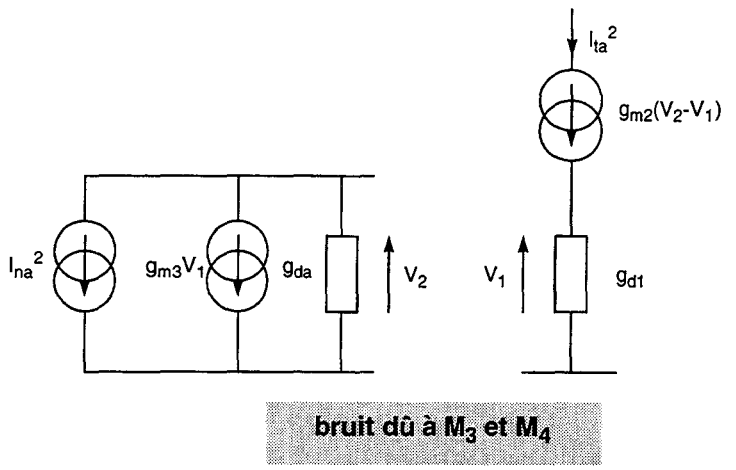


Figure 3.11 Les contributions individuelles en bruit des transistors

La contribution en bruit due à l'amplificateur est donnée par:

$$I_{ta}^2 = \frac{I_{na}^2}{\left(\left(1 + \frac{g_{m2}A_{maa}}{g_{d1}}\right)\frac{g_{da}}{g_{m2}}\right)^2} \approx 0 \qquad (3.36)$$

3.2.6 Approche qualitative de la conception

Des heuristiques pour la cellule de mémoire de courant à structure cascode actif peuvent être générées par l'évaluation de chaque équation analytique pour une variation des paramètres individuels de conception. Les résultats sont illustrés dans le Tableau 3.2.

La topologie du cascode actif utilise pratiquement les mêmes heuristiques que la topologie du cascode, car la seule différence est l'amplificateur de contre-réaction qui polarise le transistor cascode. Quelques heuristiques ont des poids différents, comparées à celles utilisées par la topologie cascode. Ceci est le résultat d'une analyse quantitative, qui montre des variations relatives plus ou moins importantes, dans les valeurs des performances, pour des variations de valeurs de dimensions similaires.

GVO	gate voltage overdrive
↗	augmentation monotone
↘	diminution monotone
✻	non-monotone
	effet mineur

	erreur de gain statique	erreur du diviseur capacitif	précision d'établissement	injection de charge	snr	dérive	puissance	surface
GVO du transistor mémoire	↘	↗	↘	↗	↗	↗		↗
longueur du transistor mémoire	↗	↘						↘
GVO du transistor cascode mémoire	↘	↘	↘					↗
longueur du transistor cascode mémoire			↗					↘
GVO du transistor de l'amplificateur mémoire								↗
longueur du transistor de l'amplificateur mémoire								↘
GVO de la source de l'amplificateur mémoire								↗
longueur de la source de l'amplificateur mémoire	↗							↘
courant de polarisation de l'amplificateur mémoire							↘	↘
GVO du transistor source					↗			↗
longueur du transistor source	↗			✻				↘
GVO du transistor cascode source	↘		✻	↘				↗
longueur du transistor cascode source	↗		✻	✻				↘
GVO du transistor de l'amplificateur source								↗
longueur du transistor de l'amplificateur source								↘
GVO de la source de l'amplificateur source								↗
longueur de la source de l'amplificateur source								↘
courant de polarisation de l'amplificateur source							↘	↘
longueur de l'interrupteur				↘		↘		↘
capacité de mémorisation		↗	↘	↗	↗	↗		↘
courant de polarisation	↘	↘	✻		↘	↘	↘	↘
amplitude du courant d'entrée			↘	↗	↗			
tension d'alimentation				↘			↘	

Tableau 3.2 Règles qualitatives de conception pour la cellule mémoire de courant à structure cascode actif

Les variantes structurelles qui ont été décrites augmentent la complexité de la cellule afin de réduire la conductance de sortie des branches. Cependant, cette modification n'a pas d'effet sur le défaut dominant de la technique des courants commutés, qui est l'injection de charge. La section suivante analyse le comportement d'une variante opérationnelle, qui modifie les contraintes de commutation, mais présente un niveau de complexité similaire à celui de la cellule de base.

3.3 Modèle de la cellule mémoire de courant S²I

L'idée principale de la technique S²I [HUG93] est de conserver et d'annuler l'erreur mémorisée. La cellule est illustrée dans la Figure 3.12.

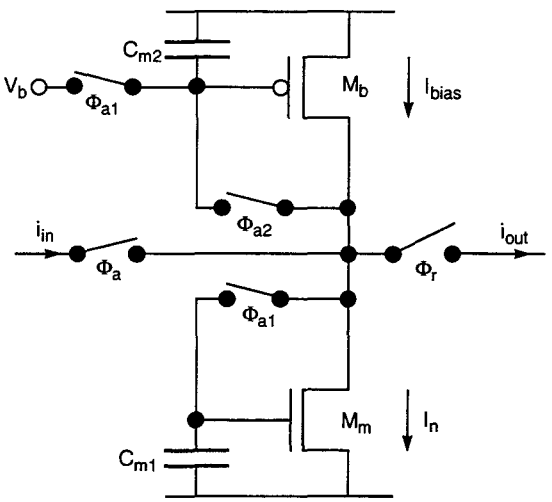


Figure 3.12 Cellule mémoire de courant de base S²I

Le processus de mémorisation est effectué en deux phases : premièrement de façon grossière, dans le transistor N, et affinée ensuite dans le transistor P, sur lequel l'erreur d'acquisition de la première phase est conservée. La chronogramme des phases est donnée par la Figure 3.13.

Lors de la phase Φ_{a1}, la cellule fonctionne comme la cellule de base. Le transistor P est relié à une tension de référence et se comporte comme une source de courant, et le transistor N est en configuration diode et est traversé par un courant de valeur égale à $I_{bias}+i_{in}$. Quand la phase Φ_{a1} est terminée, M_m mémorise un courant égal à $I_{bias}+i_{in}+\Delta i$, où Δi est l'erreur de courant (dépendant du signal) qui résulte des sources d'erreur décrites dans le chapitre précédent. Dans la phase Φ_{a2}, le transistor P est en configuration diode et sa tension de grille s'établit à une valeur telle que le courant de drain est égal à l'inverse d'un courant de $I_{bias}+\Delta i$. L'injection de charge est le principal responsable de l'erreur δi induite dans le transistor à la fin de la phase Φ_{a2}, puisque la variation de tension est peu importante sur la grille du transistor P pendant l'acquisition fine.

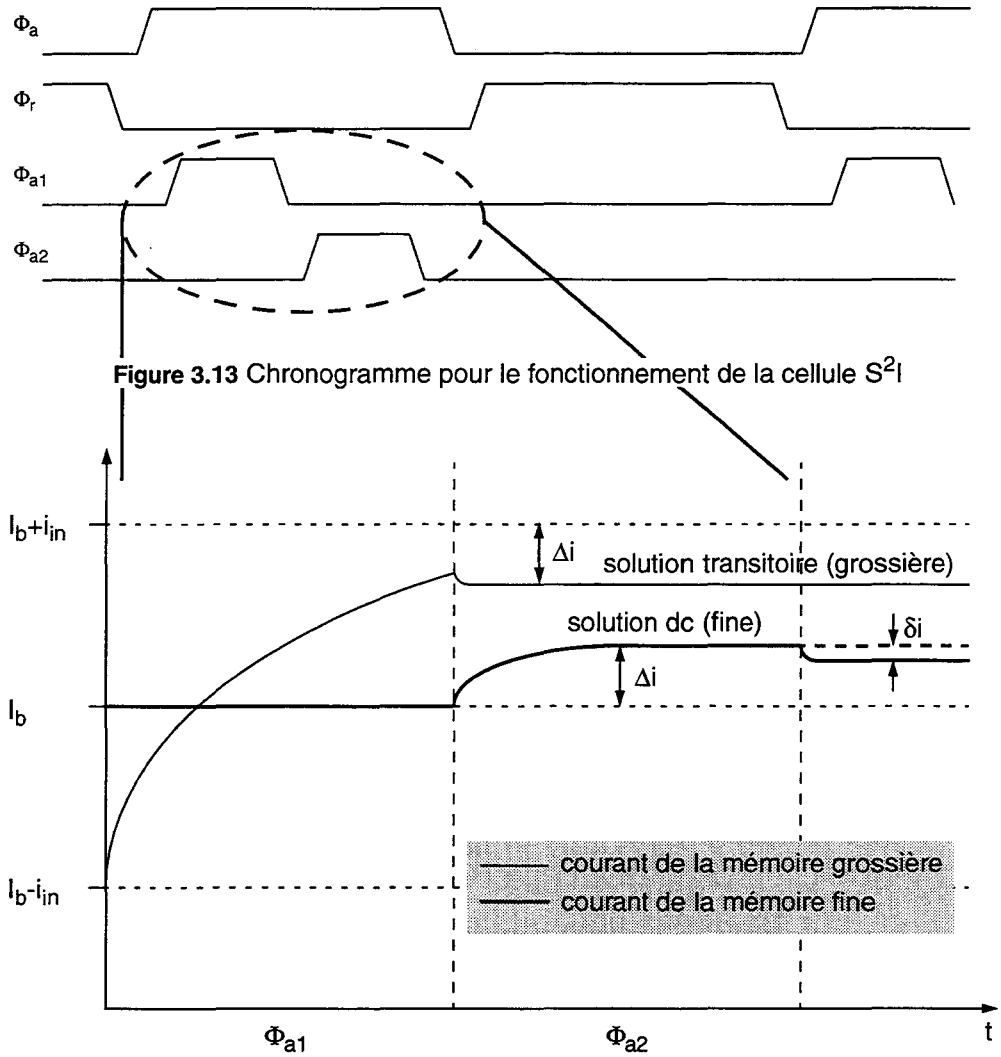


Figure 3.13 Chronogramme pour le fonctionnement de la cellule S^2I

Figure 3.13 Le besoin d'information transitoire dans la solution dc de la cellule S^2I

Le courant de sortie est alors donné par

$$i_{out} = -i_{in} + \delta i \tag{3.37}$$

Cependant, si le courant de sortie est envoyé vers une deuxième cellule qui établit une tension de drain presque égale à celle de la première (ce qu'elle fera lors de la deuxième phase d'acquisition), δi est pratiquement indépendant du signal et peut être considéré comme un offset. L'erreur peut alors être filtrée en sortie et les performances sont améliorées de quelques ordres de grandeur par rapport à la cellule mémoire de courant de base.

3.3.1 Solution DC

Contrairement aux cellules à simple boucle, la solution DC de la cellule à double boucle dépend non seulement de l'état présent de la cellule (c'est-à-dire du courant d'entrée), mais également de l'état précédent. Ceci est dû au fait que l'information essentielle à extraire de la solution DC est la situation vers laquelle la cellule tendra à la fin de la deuxième phase d'acquisition. Puisque cette dernière vise à annuler l'erreur acquise lors de la première phase, il est nécessaire d'inclure l'information provenant de celle-ci. Cette information est obtenue en appliquant l'analyse complète décrite pour la cellule mémoire de courant de base. Les formes d'onde illustrées dans la Figure 3.13 donnent forme à cet aspect, où la cellule acquiert un courant positif après avoir conservé un courant négatif de l'acquisition précédente. L'erreur grossière Δi qui doit être compensée consiste en la somme de toutes les erreurs données dans le deuxième chapitre, et particulièrement d'un point de vue temporel, de l'erreur d'établissement qui dépend non seulement du temps d'acquisition, mais également du courant précédemment mémorisé.

Le calcul de la solution DC pour la cellule à double boucle peut alors être schématisée (Figure 3.14).

3.3.2 Dynamique d'entrée et composantes d'erreur

Mis à part le fait que la mémorisation se fait sur deux capacités au lieu d'une seule (comme pour les cellules à simple boucle), la structure de la cellule S^2I est identique à celle de la cellule de mémoire de courant de base. De ce fait, le modèle est pratiquement identique, à l'exception des erreurs qui sont calculées sur le transistor P pour une dynamique fortement réduite. En effet, la variation des valeurs du courant à conserver pour ce transistor de mémorisation vaut $2\Delta i$, ce qui est quelques ordres de grandeur moins important que la variation des valeurs de courant dans le transistor N, $2i_{in}$. La dynamique du courant mémoire dans le transistor P ($2\Delta i$) ne doit pas être confondue avec la dynamique d'entrée, qui reste égale à $2i_{in}$, comme dans la cellule mémoire de courant de base.

En ce qui concerne les composantes d'erreur, chacune est évaluée comme dans les équations de la cellule de base, mais en utilisant le transistor N en tant que source pilotée par la tension grossière de grille mémorisée, et le transistor P en tant que transistor mémoire. La principale différence a pour origine l'augmentation du nombre de combinaisons possibles : nous ne pouvons plus nous limiter à la considération de l'état présent d'acquisition (c'est-à-dire pour un courant minimal, nominal ou maximal), mais nous devons également considérer l'état précédent, comme expliqué dans la section 3.3.1. Il existe donc neuf possibilités (et donc autant d'évaluations de performances) au lieu de trois pour la cellule de base.

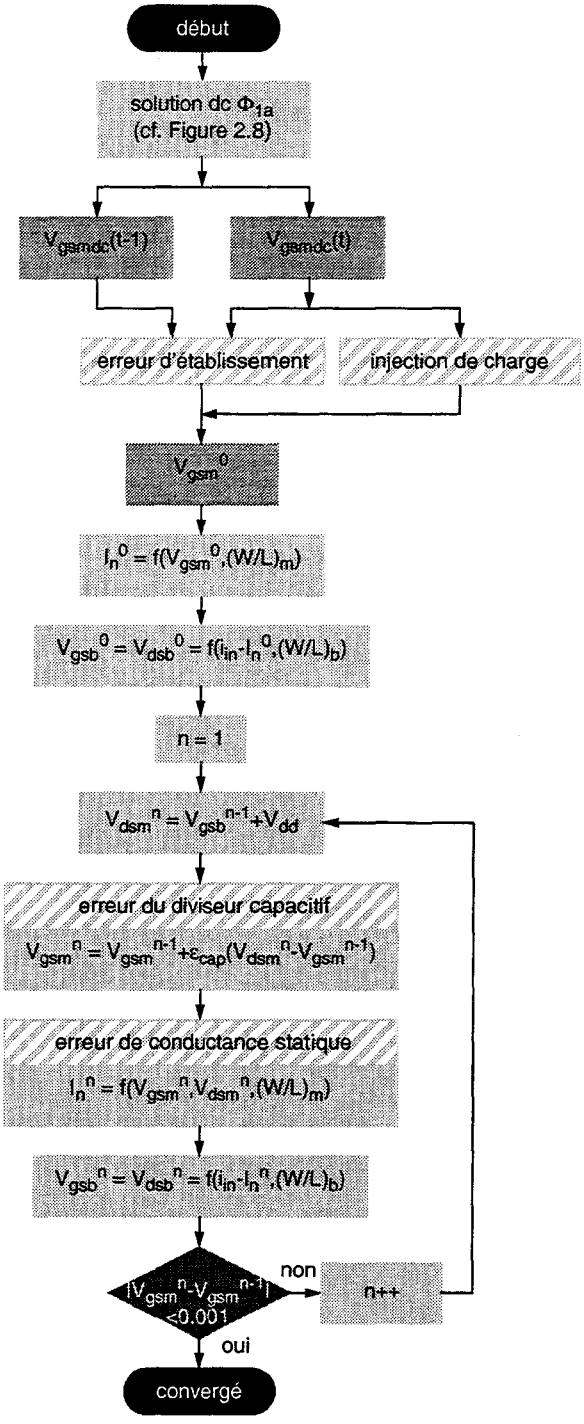


Figure 3.14 Solution DC pour la cellule de base S^2I

3.3.3 Approche qualitative de la conception

Comme précédemment, les heuristiques pouvant être utilisées dans la conception de la cellule mémoire de courant de base S^2I sont résumées dans le Tableau 3.3.

GVO	gate voltage overdrive
↗	augmentation monotone
↘	diminution monotone
✻	non-monotone
	effet mineur

	↘ erreur de gain statique	↘ erreur du diviseur capacitif	↗ précision d'établissement	↘ injection de charge	↗ snr	↘ dérive	↘ puissance	↘ surface
GVO du transistor mémoire N	↗	↗		↘	↗	✻		↗
longueur du transistor mémoire N	↗		↘					↘
GVO du transistor mémoire P			✻	↗	↗	↗		↗
longueur du transistor mémoire P	↗	↘	✻					↘
longueur de l'interrupteur	↗			↘		↘		↘
capacité de mémorisation N						↗		↘
capacité de mémorisation P		↗	↘	↗	↗	↗		↘
courant de polarisation			✻	↘	↘	↘	↘	↘
amplitude du courant d'entrée			↘	↗	↗			
tension d'alimentation				↘			↘	

Tableau 3.3 Règles qualitatives de conception pour la cellule mémoire de courant de base S^2I

Les heuristiques de l'*erreur de gain statique* visent exclusivement à diminuer la conductance de sortie. Comme la GVO du transistor mémoire P est typiquement élevée[1], les variations de cette quantité ont peu d'effet sur la transconductance. La conductance de sortie du transistor mémoire N peut être modifiée par la variable L et également par sa GVO, car la conductance de sortie n'est pas complètement indépendant de cette quantité. Puisque la GVO est typiquement faible pour ce transistor[2], l'effet des variations de cette quantité sur la conductance de sortie est accentué. Elle a également une grande influence sur la transconductance, mais comme ce transistor se comporte en source pour l'analyse de l'erreur de gain statique, ceci a peu d'importance.

1. Puisque la dynamique de courant du transistor mémoire P est faible, la transconductance est diminuée pour augmenter la variation de tension sur la grille et ainsi réduire la taille du transistor. Cependant, ceci a pour inconvénient le fait que l'erreur de sortie augmente sa dépendance envers le signal, ce qui augmente la distorsion harmonique.
2. La dynamique de courant du transistor mémoire N est élevée, et donc la transconductance doit également être élevée afin de restreindre les variations de la tension de grille à des niveaux acceptables.

Pour l'*erreur du diviseur capacitif*, la variation de la tension de grille peut être réduite en augmentant la capacité du noeud de sortie. Le moyen le plus efficace d'augmenter cette dernière est d'augmenter la GVO du transistor mémoire N qui, puisqu'elle est généralement faible, a une grande influence sur la largeur du transistor.

Les heuristiques pour la *précision d'établissement* sont similaires à celles de la cellule de base, mais sont transposées du transistor N au transistor P.

L'*erreur de l'injection de charge* peut être réduite en diminuant la GVO du transistor N (ce qui augmente la capacité du noeud de sortie), ou en augmentant la GVO du transistor P (ce qui diminue la transconductance). Un effet identique est également obtenu en diminuant le courant de polarisation. La réduction des transconductances N et P est également utile afin de diminuer le *bruit* dans le circuit.

Les heuristiques utilisées pour optimiser l'*erreur de la dérive* sont identiques à celle de la cellule de base, à l'exception de l'heuristique ambiguë de la GVO du transistor mémoire. L'erreur de la dérive est en réalité la *différence* entre les dérives individuelles N et P. En posant la diminution de l'une (P, par la réduction de la transconductance), l'autre est libre pour compenser la première. Cependant il existe de meilleures façons, plus explicites, de diminuer l'erreur de la dérive, comme le montre le Tableau 3.3.

3.4 Conclusion

La méthode analytique utilisée pour la cellule mémoire de courant de base a été appliquée à d'autres cellules afin d'étendre la bibliothèque des topologies d'une manière cohérente. Bien qu'une bibliothèque exhaustive de cellules soit difficile à constituer, la sélection des cellules implémentées peut en faciliter la tâche d'extension. Ces sélections sont effectuées en considérant que toutes les topologies proposées jusqu'ici peuvent être classées en deux catégories : des variantes structurelles (où la commutation ou le mode opératoire n'est pas modifié, mais la structure l'est) et des variantes opérationnelles (où la commutation ou le mode opératoire est modifiée, alors que la structure reste pratiquement identique).

En utilisant cette classification, deux cellules qui constituent les variantes structurelles les plus communes, c'est-à-dire le cascode et le cascode actif, ont été analysées afin de générer leur modèle analytique. L'avantage principal de ces topologies consiste en une conductance de sortie réduite, ce qui réduit les erreurs de gain. Cependant, cet avantage est obtenu au prix d'une dynamique d'entrée réduite, et aucun effet positif n'est apporté aux erreurs dues aux non-idéalités de l'interrupteur. Le calcul DC a dû être modifié par rapport à la cellule de base, et le calcul de la précision d'établissement s'éloigne de plus en plus de la réalité. En limitant l'équation à une expression du second ordre, l'approximation est abusive : cependant, un compromis est nécessaire entre le temps d'évaluation et la précision. Puisque l'erreur est corrigée par l'utilisation de la simulation numérique dans la boucle externe d'optimisation, la précision finale est cependant assurée.

L'autre classe de cellule, la variante opérationnelle, est illustrée par la cellule S^2I, qui modifie la cellule de mémoire de courant de base afin de réaliser un échantillonnage en deux temps. De cette façon, l'erreur de l'échantillonnage grossier est annulée, et l'erreur de l'échantillonnage fin est pratiquement indépendante du signal, ce qui réduit de quelques ordres de grandeur les erreurs dues au gain et à la distorsion harmonique. L'inconvénient est que la commutation est plus complexe. Dans le modèle analytique, l'analyse est identique à celle de la cellule mémoire de courant de base, sauf que l'erreur d'échantillonnage se fait sur le transistor P plutôt que sur le transistor N. La différence principale provient de la solution DC, qui est inévitablement plus complexe à calculer du fait de l'augmentation du volume d'informations requises. En effet, la tension, vers laquelle la capacité d'échantillonnage P tend, dépend de la tension échantillonnée sur la capacité d'échantillonnage N. Cette valeur ne peut être évaluée que par le calcul de toutes les erreurs pouvant être acquises dans la première phase d'échantillonnage.

Références

[BAT91] N.C. Battersby, C. Toumazou, "Class AB Switched-Current Memory for Analogue Sampled-Data Systems", *Electronics Letters*, vol. 27, no. 10, pp. 873-875, May 1991

[HUG93] J.B. Hughes, K.W. Moulding, "S^2I: A Switched-Current Technique for High Performance", *Electronics Letters*, vol. 29, no. 16, pp. 1400-1401, August 1993

[NAI96] D.G. Nairn, A. Biman, "A Comparative Analysis of Switched-Current Circuits", *IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 43, no. 11, pp. 733-743, December 1996

[TOU90] C. Toumazou, J.B. Hughes, D.M. Pattullo, "Regulated Cascode Switched-Current Memory Cell", *Electronics Letters*, vol. 26, no. 5, pp. 303-305, March 1990

4 LA SIMULATION NUMÉRIQUE DE CELLULES MÉMOIRE DE COURANT

Les modèles analytiques, utilisés pour l'évaluation des performances du circuit dans la boucle interne d'optimisation, contiennent des imprécisions inhérentes aux simplifications nécessaires à leur établissement. Afin de pallier ces imprécisions, la simulation numérique peut être utilisée dans la boucle externe d'optimisation. Des analyses permettent l'estimation de l'erreur d'évaluation des équations analytiques. Si l'erreur est trop importante, et les spécifications sont en réalité non-satisfaites, la boucle interne peut être à nouveau invoquée avec les évaluations de performances compensées. La solution finale offre donc la garantie de satisfaire à toutes les spécifications.

Comme décrit dans le premier chapitre, l'utilisation de modèles analytiques engendre une erreur dans la prédiction des performances de la cellule. Dans un outil de CA, cette erreur peut conduire la recherche du point optimal vers une solution qui, selon les modèles analytiques, constitue la meilleure solution, mais s'en éloigne dans la réalité (Figure 4.1).

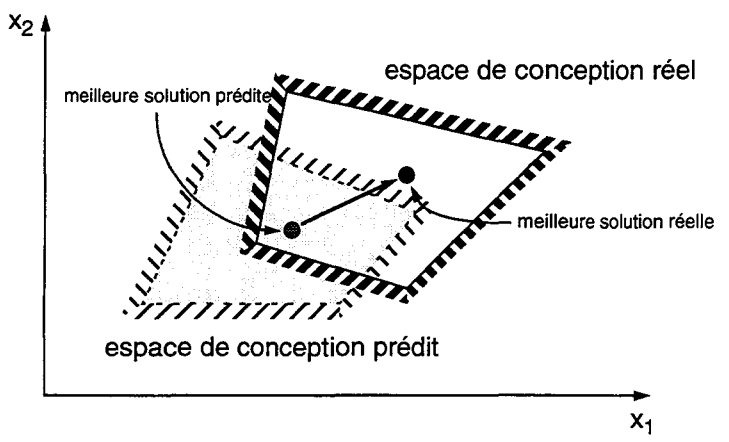


Figure 4.1 Effet des erreurs des modèles analytiques sur la conception automatisée

Il est donc souhaitable d'utiliser des techniques de simulation numérique dans la boucle externe d'optimisation où le circuit, généré par la synthèse avec des modèles analytiques, est simulé. Ainsi, des prédictions plus réalistes des performances du circuit seront obtenues, comme illustré dans le système prototype de CA décrit dans le premier chapitre. La différence entre les évaluations analytiques et numériques du circuit donnent alors lieu à des facteurs de compensation qui peuvent être ensuite incorporés dans les fonctions d'évaluation de la boucle interne (analytique) d'optimisation, si nécessaire.

Cependant, la caractérisation d'une cellule à courants commutés par simulation n'est pas une tâche aisée. Du fait de sa nature fondamentalement non-linéaire et de sa sensibilité à son environnement de fonctionnement, il est nécessaire d'effectuer un certain nombre de simulations afin de valider les spécifications. Finalement, la caractéristique fondamentale des performances est le spectre de l'erreur d'échantillonnage et les contributions de l'erreur globale à l'offset, à l'erreur de gain et à la distorsion harmonique. Cette information ne peut être obtenue que par une simulation transitoire sur un grand nombre de périodes d'horloge. Or, une telle simulation est très longue lorsqu'elle est effectuée par un simulateur numérique tel que SPICE, et l'évaluation des performances doit y recourir le plus tard possible dans le cycle de conception. Ceci signifie que le processus itératif de conception n'a pas suffisamment d'information, et qu'un moyen d'évaluer l'erreur d'échantillonnage de façon numérique doit être trouvé sans recourir à une simulation transitoire longue.

En évaluant la contribution à l'erreur d'échantillonnage de chacune des non-idéalités du circuit, nous pouvons faire une équivalence entre les simulations numériques et les analyses à base d'équations décrites dans les chapitres de modélisation des cellules. La méthode décrite par la suite réalise ces prédictions de performances par simulation numérique, en des temps de calcul acceptables. Pour que les résultats soient comparables, le circuit simulé doit fonctionner dans les mêmes conditions que pour le modèle analytique, c'est-à-dire dans le pire cas.

Bien évidemment, la simulation de la cellule seule n'est pas suffisante pour garantir un circuit de bonne qualité. Les performances de la cellule sont extrêmement dépendantes des éléments parasites, dont les valeurs ne sont pas connues avant layout. De ce fait, une boucle supplémentaire d'optimisation est nécessaire, qui consiste en la synthèse du layout, l'extraction des parasites qui lui sont associés, et la simulation numérique. Une méthode similaire a été utilisée dans SCADS [HUG96].

4.1 Analyse statique

L'objectif de cette analyse est d'évaluer l'erreur de conductance statique, ou erreur de gain statique. En effectuant une analyse DC, tout effet dynamique dû au diviseur capacitif et à la commutation est éliminé. L'erreur restante est seulement due à la conductance non-nulle de la cellule.

La fonction d'erreur recherchée relie le courant d'entrée au courant de sortie. Comme une cellule ne peut pas être en mode d'acquisition et de restitution simultanément, deux cellules sont utilisées. Une cellule acquiert le courant et génère la tension de grille correspondante, tandis qu'une autre, pilotée par cette même tension, restitue un courant vers une source de tension égale à la valeur de la tension de grille, à δv près. Le circuit correspondant est illustré dans la Figure 4.2.

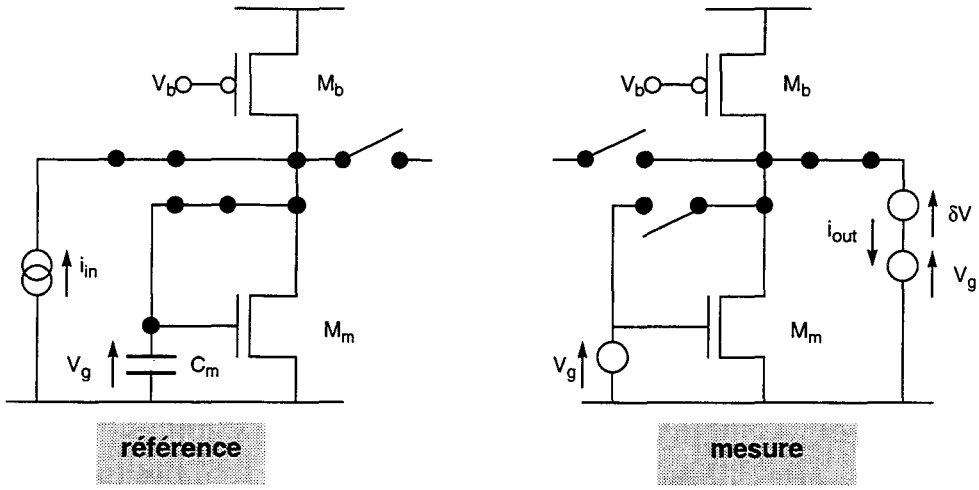


Figure 4.2 Simulation de l'erreur de gain statique

De cette façon, la conductance de sortie de la cellule est obtenue :

$$g_0 = \frac{i_{out} + i_{in}}{\delta v} \tag{4.1}$$

δv est une tension pouvant varier de zéro à (v_{gmax}-v_{gmin}), qui assure que la dynamique d'entrée de la cellule n'est pas dépassée, ce qui impliquerait la désaturation du transistor mémoire ou de la source du courant de polarisation.

L'outil de synthèse calcule l'erreur de gain statique en tant qu'erreur sur l'ensemble de la dynamique d'entrée. Pour calculer sa valeur, nous utilisons

$$\varepsilon_{dc} = \frac{i_{out} + i_{in}}{\delta v}(v_{gmax} - v_{gmin})\frac{100\%}{i_0} \tag{4.2}$$

Cette simulation a été effectuée en faisant varier δv de sa valeur minimale à sa valeur maximale pour plusieurs valeurs de i_{in} (Figure 4.3).

Pour une valeur positive de δv, le courant du transistor mémoire augmente et le courant du transistor source diminue, ce qui engendre une diminution du courant de sortie. Le raisonnement inverse est vrai pour une valeur négative de δv.

Pour un courant d'entrée négatif, l'erreur est moins importante que pour un courant d'entrée positif. En effet, le gain du transistor g_m/g_d est plus élevé pour un courant de transistor plus bas, c'est-à-dire pour un courant d'entrée négatif.

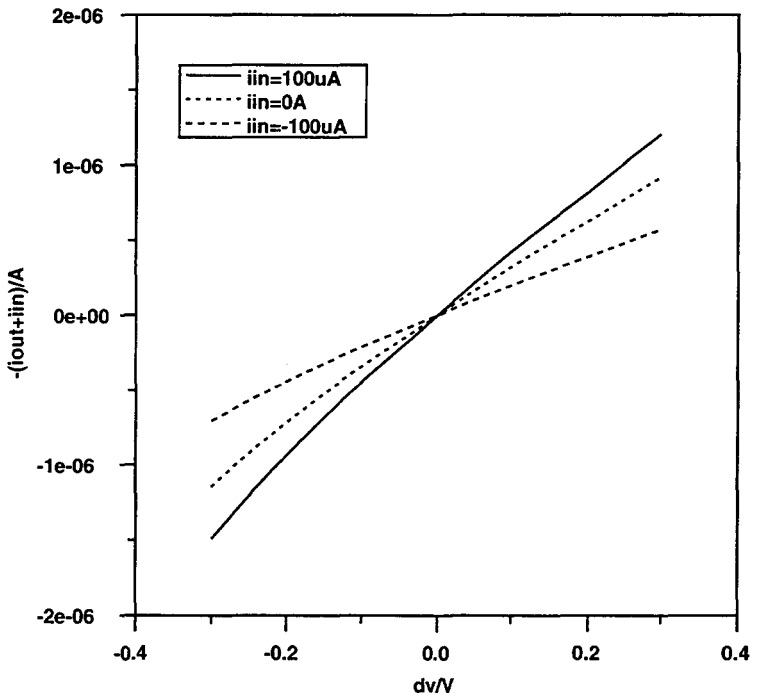


Figure 4.3 Erreur de recopie statique en fonction de l'offset sur le noeud de sortie

4.2 Analyse dynamique

Une analyse similaire à la précédente peut être effectuée de façon dynamique, où un nombre d'erreurs se cumulent et doivent donc être séparées au moyen de simulations multiples, comme illustré dans la Figure 4.4. Le courant du signal d'entrée est généré par un circuit échantillonneur-bloqueur presque idéal. Il génère une tension échantillonnée qui est ensuite convertie sous forme de courant par une source de courant contrôlée en tension (VCCS). La tension de grille idéale est générée par un circuit de référence qui est identique au circuit de mesure à excepté qu'il est toujours en mode acquisition. Sa tension de grille est donc constante. Elle est ensuite recopiée vers la sortie du circuit de mesure via une source de tension commandée en tension (VCVS). Cette source est reliée en série avec une source de tension indépendante qui génère une tension de valeur égale à δv. Cette simulation s'effectue avec plusieurs valeurs de δv et de i_{in} afin d'estimer les erreurs dynamiques.

La tension V_{in} prend les valeurs qui correspondent à l'analyse du modèle pour les cas : courant d'entrée maximal, nominal et minimal. Elle est de valeur constante pour éviter d'introduire des erreurs d'établissement. Le chronogramme des signaux est illustré dans la Figure 4.5. L'objectif est de pouvoir générer les signaux de commande des interrupteurs à partir de la fréquence d'échantillonnage et du rapport de la période d'horloge au temps de commutation σ_f.

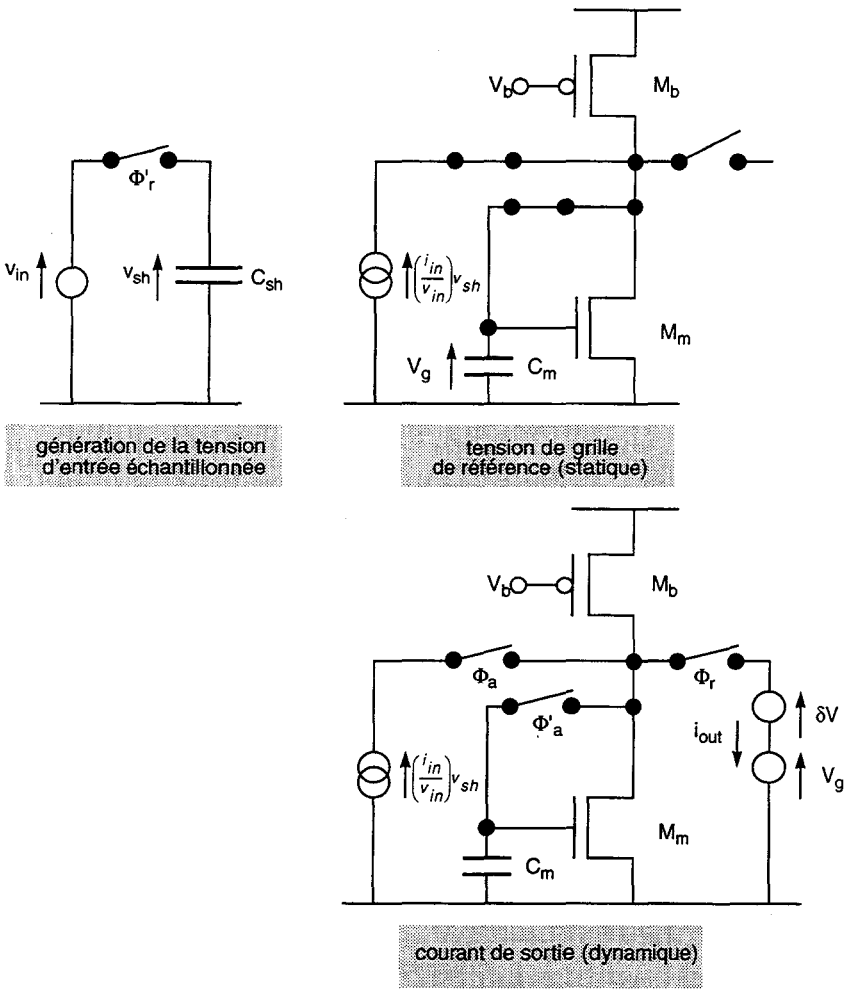


Figure 4.4 Simulation des erreurs du diviseur capacitif et de l'injection de charge

$$\sigma_f = \frac{1}{f_s t_f} \tag{4.3}$$

$$t_r = t_f \tag{4.4}$$

$$T = \frac{2}{f_s} \tag{4.5}$$

$$T_{on} = \frac{1}{f_s} \tag{4.6}$$

Comme déjà mentionné, l'erreur totale est la somme de plusieurs erreurs :

- l'erreur de conductance statique,
- l'injection de charge,
- l'erreur de conductance dynamique (diviseur capacitif).

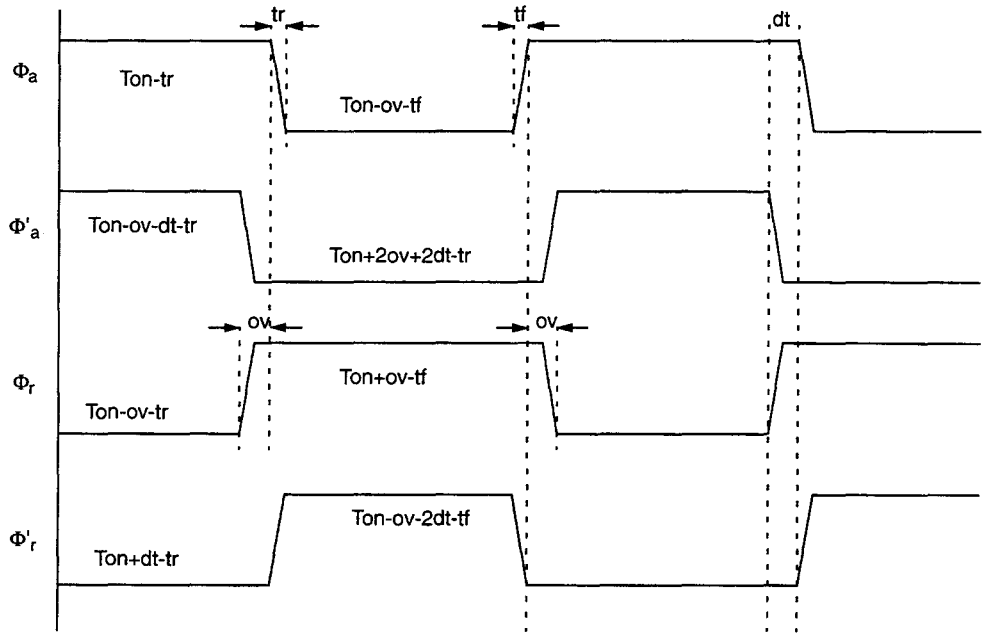


Figure 4.5 Chronogramme général des commandes des interrupteurs pour la cellule mémoire de courant de base

La première étant déjà quantifiée, nous avons besoin d'une méthode pour évaluer les deux autres de façon indépendante. Ceci peut être fait en doublant les largeurs des interrupteurs et en resimulant le circuit. De cette façon, la quantité de charge injectée à partir des interrupteurs est doublée, et la différence des courants entre les simulations donne l'erreur due à l'injection de charge (Figure 4.6).

Une erreur d'évaluation se produit pourtant, due au fait que la capacité sur le noeud du drain (ainsi que celle sur le noeud de grille) est augmentée avec l'augmentation de la largeur des interrupteurs. Ceci modifie le rapport des capacités drain/grille et donc la distribution de la charge à l'extinction des interrupteurs. Cependant, dans la plupart des cas pratiques, la capacité due à la largeur de l'interrupteur est de valeur beaucoup moins importante que les autres capacités sur le noeud, et peut donc être négligée. L'autre alternative pour l'évaluation comparative de la contribution à l'erreur dynamique provenant de l'injection de charge est de doubler la longueur de l'interrupteur. Cette approche ne modifie pas les capacités de noeud, tout en augmentant la charge du canal. Ici encore, la solution n'est pas idéale, car la charge ayant pour origine les capacités de recouvrement reste constante, alors que la charge dans le canal est plus que doublée (L_{eff}=2L-L_d, et non 2L-2L_d, où L_d représente la diffusion latérale sous la grille). Egalement, le temps de transit augmente avec la longueur de l'interrupteur, et augmente ainsi la quantité de charge sortant vers le substrat dans le cas réel. Nous supposerons donc qu'en doublant la largeur, l'approximation est suffisamment bonne pour obtenir la composante dynamique due à l'injection de charge.

$$\varepsilon_{cft} = \delta i \frac{100\%}{i_0} \qquad (4.7)$$

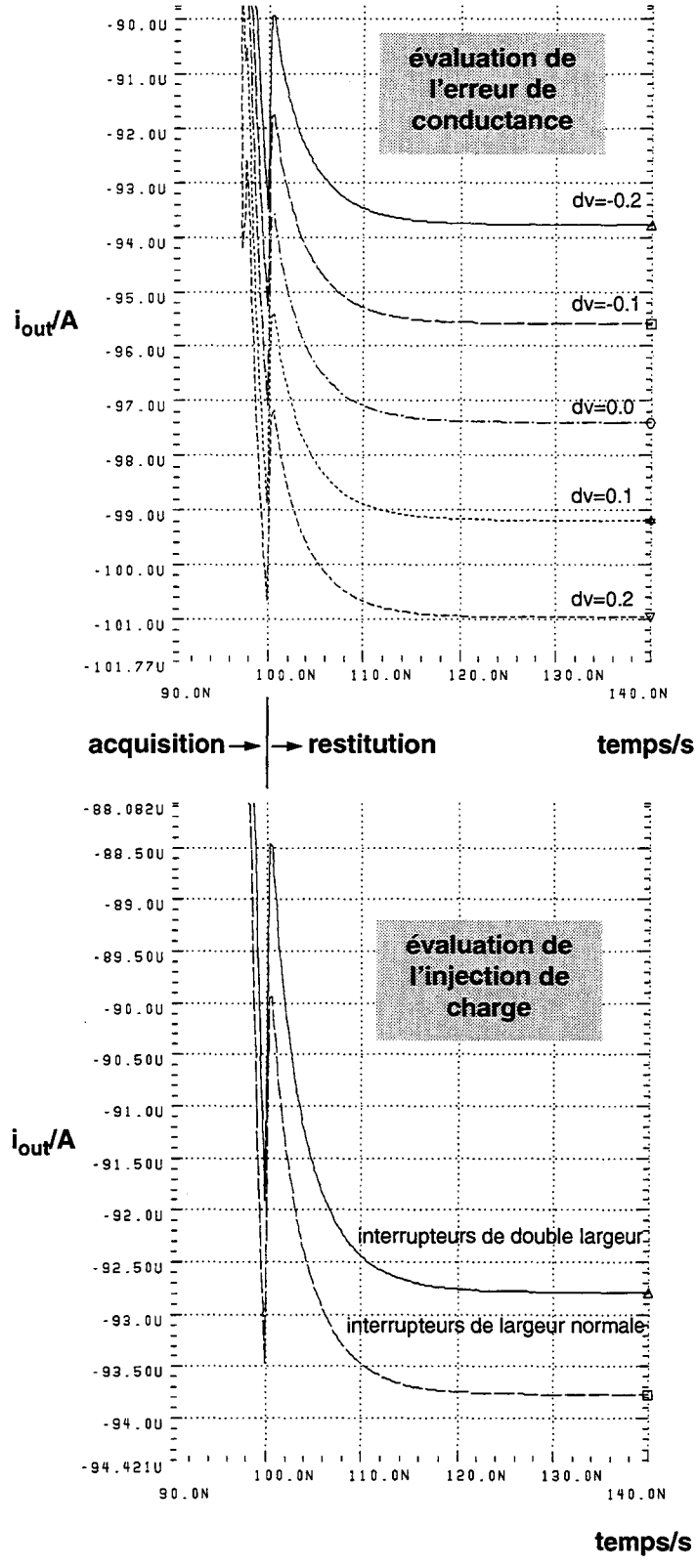


Figure 4.6 Extraction des erreurs individuelles à partir de l'analyse dynamique

La dernière composante de l'erreur dynamique peut donc être évaluée en calculant la différence entre l'erreur de conductance statique déjà quantifiée et l'erreur de conductance dynamique simulée pour la même variation de δv.

$$\varepsilon_{cap} = \left| \frac{(i_{out}|_{\delta v1} - i_{out}|_{\delta v0})}{\delta v1 - \delta v0} (v_{gmax} - v_{gmin}) \frac{100\%}{i_0} \right| - \varepsilon_{dc} \tag{4.8}$$

L'erreur de conductance globale constitue, d'un point de vue utilisateur, une quantité unique qui permet la mesure d'une partie des performances de la cellule. Cependant, l'erreur globale contient des composantes d'erreur, qui ont des sources différentes. Puisque les sources sont différentes, les heuristiques le sont aussi. C'est pourquoi l'outil de CA requiert cette décomposition de l'erreur globale.

4.3 La précision d'établissement

Une autre analyse dynamique peut être effectuée afin d'évaluer la précision d'établissement. Dans cette analyse, la cellule reste en mode d'acquisition jusqu'à sa stabilisation complète, et une comparaison est alors effectuée entre la valeur du courant stabilisé et celle du courant à la fin du temps d'acquisition. La différence est indépendante des erreurs absolues telles que celles décrites précédemment.

Le schéma est identique à celui de l'analyse dynamique précédente, à l'exception qu'aucune variation de tension δv n'est appliquée à la sortie, puisque les performances de la cellule en restitution n'ont pas d'importance pour cette analyse. Le chronogramme est légèrement différent puisque le fonctionnement n'est pas répétitif.

L'erreur d'acquisition peut être évaluée par de nombreuses méthodes. Comme indiqué précédemment, la valeur du courant à la fin de la phase d'acquisition ne constitue pas nécessairement une bonne indication de la précision, car le courant du transistor peut se trouver en milieu de course entre deux oscillations. Cependant, dans la simulation, et à l'inverse de la synthèse, cette considération a moins d'importance car en effectuant plusieurs simulations pour des situations différentes, la probabilité qu'une mesure erronée soit prise est pratiquement réduite à zero.

La précision d'établissement (en bits) peut s'écrire

$$n = \log_2\left(\frac{i_0}{\delta i}\right) \tag{4.9}$$

ou, en exprimant δi comme l'erreur maximale permise pour une valeur de n donnée

$$\delta i = \frac{1}{2} LSB = \frac{1}{2}\frac{2i_0}{2^n} \tag{4.10}$$

où i_0 est l'amplitude du signal (et $2i_0$ est sa valeur crête-crête).

Ceci nous amène à la deuxième méthode d'évaluation de la précision d'établissement, qui consiste en la mesure du dernier moment auquel le courant vaut la valeur stabilisée ± l'erreur maximale permise. Si ceci se produit avant t_{acq}, la spécification est validée (mais nous ne connaissons pas l'écart avec la valeur de précision spécifiée).

Des simulations sont donc effectuées pour six cas :

		$i_{initial}$		
		$+i_{max}$	0	$-i_{max}$
i_{final}	$+i_{max}$			
	0			
	$-i_{max}$			

qui devraient couvrir la majeure partie des possibilités pour le comportement d'établissement dans le pire cas. La précision d'établissement mesurée est donc égale à la valeur minimale des grandeurs évaluées.

4.4 Bruit

L'analyse de bruit constitue une analyse fréquentielle simple, qui détermine le bruit sur la grille du transistor mémoire par rapport au courant d'entrée. Pour une analyse fréquentielle, la cellule est configurée en mode d'acquisition (Figure 4.7).

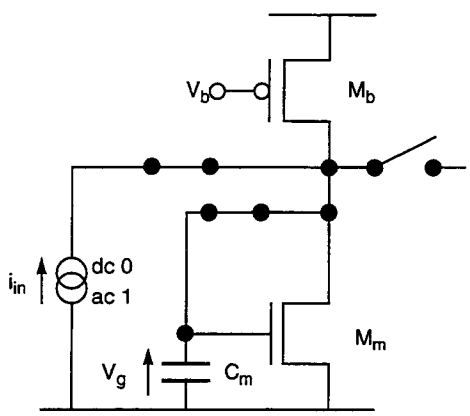


Figure 4.7 Configuration de simulation pour l'analyse de bruit

Un balayage en fréquence et une analyse de bruit au noeud de la grille permet d'obtenir la densité spectrale de puissance du bruit, illustré dans la Figure 4.8.

Le bruit total de sortie, c'est-à-dire l'intégrale du bruit sur la gamme complète de fréquences dans le domaine continu est donnée en volts dans le fichier de sortie de la simulation. Le bruit, exprimé en termes de puissance de courant est donc

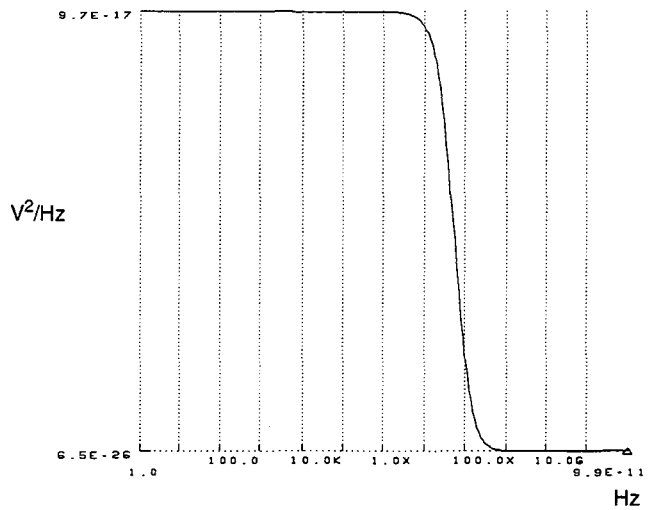


Figure 4.8 Spectre de puissance du bruit

$$i_n^2 = g_{mm}^2 v_n^2 \tag{4.11}$$

Dans le domaine à temps discret, cette valeur représente le bruit jusqu'à la fréquence de Nyquist ($f_s/2$, où f_s représente la fréquence d'échantillonnage). La puissance de bruit dans la bande du signal est

$$i_{ns}^2 = i_n^2 \frac{f_{sig}}{f_s/2} \tag{4.12}$$

et donc le rapport signal à bruit est

$$SNR = 10\log\left|\frac{i_0^2/2}{i_{ns}^2}\right| \tag{4.13}$$

où i_0 est l'amplitude du signal, et $i_0^2/2$ la puissance du signal (i_{rms}^2).

L'outil de synthèse évalue la puissance de bruit jusqu'à $f_s/2$ et ne prend pas en compte la bande passante du signal. Ceci donne de la flexibilité dans l'interprétation des résultats. A chaque fois que la bande passante du signal est divisée par rapport à la fréquence d'échantillonnage, le rapport signal à bruit augmente de 3dB. Par exemple, le SNR pour une cellule qui échantillonne à deux fois la fréquence de Nyquist ($f_s=4f_{sig}$) serait

$$SNR = 10\log\left|\frac{i_0^2/2}{i_{ns}^2}\right| + 3dB \tag{4.14}$$

4.5 Dérive

Une analyse transitoire simple avec une phase de restitution prolongée (jusqu'à 100 périodes d'échantillonnage) permet l'évaluation de la dérive du courant (Figure 4.9).

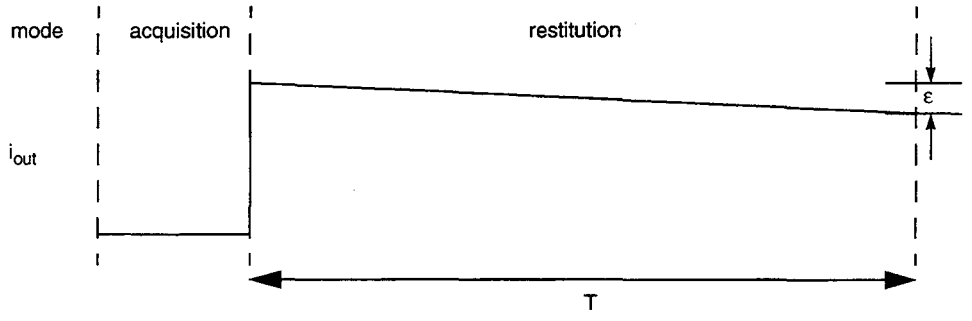


Figure 4.9 Restitution prolongée pour l'évaluation de la dérive

Comme décrit dans la section 2.10, cette dérive est due aux courants de fuite qui émanent des diodes de jonction en polarisation inverse. Le problème avec cette simulation est que ces courants de fuite sont fortement dépendants de la température ambiante (ils doublent toutes les 10°C), la valeur de celle-ci n'étant pas connue avant la mise sous tension du circuit fabriqué. Néanmoins, en simulant à la température nominale (300K), une comparaison peut être effectuée entre les modèles prédits et simulés.

A partir de la Figure 4.9, la dérive est donnée par

$$\varepsilon_{drift} = \frac{\varepsilon}{T} \tag{4.15}$$

4.6 Distorsion

Cette dernière analyse nécessite un temps de calcul beaucoup plus long que pour l'ensemble des autres. La cellule est configurée pour acquérir à partir d'une source idéale et pour restituer vers une cellule similaire sur un certain nombre de périodes du signal d'entrée, comme illustré dans la Figure 4.10 [CRA94]. Cette analyse prend en compte toutes les analyses précédentes (à l'exception du bruit), les mettant dans le contexte d'un fonctionnement réel. Le bruit n'est pas inclus dans les analyses transitoires par la plupart des simulateurs numériques. L'exception notable est ELDO [ANA94], qui simule le bruit par l'utilisation de valeurs aléatoires dans la dynamique des sources de bruit. Cependant, cette technique n'a pas pour l'instant été généralisée à d'autres simulateurs.

Cette analyse nécessite un traitement post-simulation, car l'information essentielle est constituée par la valeur effectivement échantillonnée, qui se trouve dans la source i_{meas}.

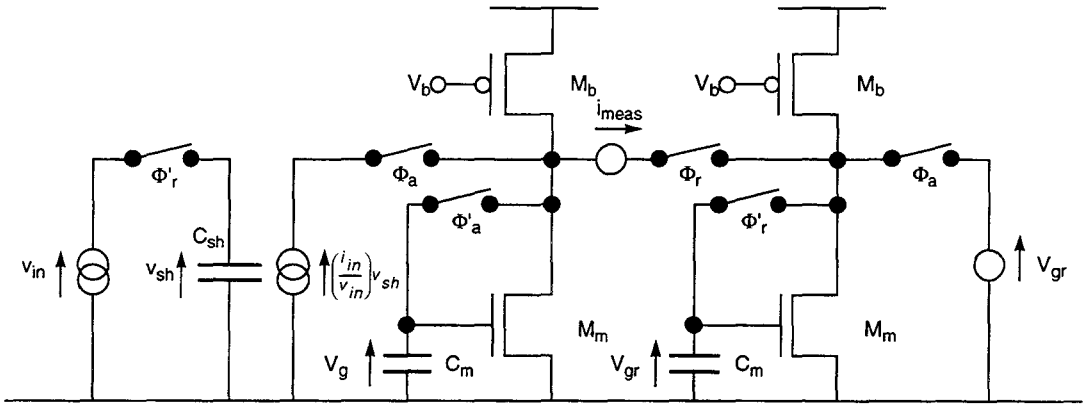


Figure 4.10 Configuration de simulation transitoire pour l'analyse de distorsion

Mais les glitches et les valeurs nulles de la sortie pendant acquisition, présentes dans le signal simulé, sont filtrés dans les applications pratiques. La sortie de la cellule lors d'un filtrage idéal est donnée par les points en fin de phase de restitution (un point par période). Ces valeurs peuvent être extraites simplement du fichier de sortie par

```
.print tran i(meas)
.tran 2T tend start=2T-dt
```

où T est la période d'horloge (la cellule restitue tous les 2T), tend est la durée de la simulation, et dt est la marge de temps avant la fin de la phase de restitution. Idéalement, cette valeur serait nulle, mais cette marge sert à éviter les effets de commutation,

Par un traitement post-simulation par *awk* (langage de traitement et d'analyse de motif de texte) [SUN95], ces résultats constituent les paramètres d'une source de courant linéaire par segments. Le courant généré est analysé par FFT afin d'obtenir la distorsion harmonique totale (THD), donnée dans la Figure 4.11.

```
.fft i(imeas) start=2T-dt stop=tend-dt freq=fsig window=harris
```

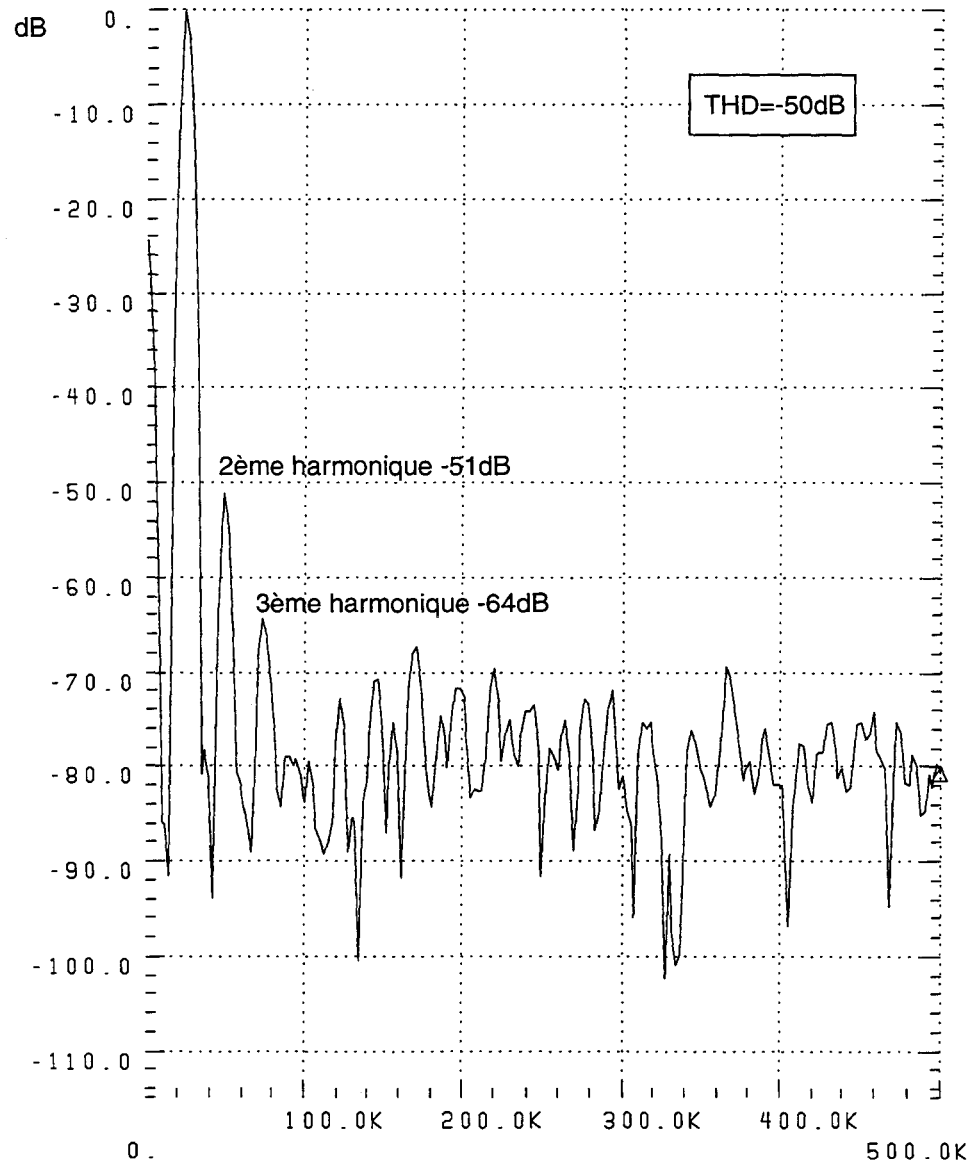


Figure 4.11 Résultats de l'analyse de distorsion

4.7 Comparaison avec le modèle analytique

L'environnement de simulation numérique qui vient d'être décrit peut être utilisé afin de valider les prédictions des modèles analytiques développés dans les chapitres précédents. Ces simulations sont nécessaires à la boucle externe d'optimisation, où le circuit final doit être validé par la simulation numérique. Le temps nécessaire à l'évaluation (environ cinq minutes sur une station de travail Ultra Sparc en utilisant HSPICE) empêche son utilisation dans la boucle interne d'optimisation.

L'environnement peut également être utilisé afin de valider les espaces de conception et donc les règles qualitatives de conception, et d'estimer ainsi l'écart entre les performances prédites et celles obtenues. Les espaces de conception sont illustrés en annexe B.

Les valeurs prédites de l'*erreur de gain statique* sont très proches des valeurs simulées. Puisque l'équation du courant de drain statique est pratiquement la seule à être utilisée dans les deux contextes, ceci sert à démontrer que le point de polarisation est calculé correctement, et également que les équations statiques ont été implémentées correctement dans le modèle de dispositif utilisé par l'outil.

L'*erreur due au diviseur capacitif* est également bien modélisée. Une légère divergence est notée pour les valeurs faibles de capacité de grille (<0.1pF). Pour de telles valeurs, la capacité de grille totale devient fortement dépendante des capacités parasites (telles que les capacités des surfaces actives de l'interrupteur). Ceci n'a pas été pris en compte dans le modèle analytique, ce qui conduit à une sous-estimation de la capacité de grille totale. L'erreur due au diviseur capacitif est donc sur-estimée pour des valeurs faibles de capacité de grille.

Le modèle est moins précis par rapport au *comportement d'établissement.* Ceci peut être attribué en partie à l'exactitude numérique élevée requise pour une valeur importante de la précision d'établissement. Par exemple, une précision de 20 bits correspond ici à la détection d'oscillations dans la gamme du pA. Les tendances prédites par les heuristiques sont, en général, respectées. Néanmoins, nous pouvons constater que la modélisation précise du comportement d'établissement de la cellule mémoire de courant reste difficile à réaliser.

Il existe également des différences entre quelques valeurs prédites et simulées pour l'*injection de charge*. Il faut rappeler que la méthode de simulation de l'injection de charge n'est pas idéale, et peut conduire à des erreurs pour des tailles importantes de l'interrupteur ou pour des capacités de grille faibles. Dans ces conditions, la capacité parasite de l'interrupteur n'est pas négligeable, et comme l'évaluation procède en doublant la largeur de l'interrupteur, ceci peut fausser les résultats de simulation. En fonction de l'effet de ce doublement sur la capacité du noeud de drain ou de grille, l'injection de charge peut être sous- ou sur-estimée dans la simulation. Nous avons donc une situation où le modèle analytique peut être plus précis que la simulation.

Cependant, ce n'est pas la seule origine des écarts. La connection drain-grille constitue une boucle de contre-réaction qui, pour des valeurs faibles de la capacité de grille, a une constante de temps faible. Par conséquent, pour des temps de commutation longs, l'interrupteur ne se comporte plus comme une conductance simple qui varie linéairement dans le temps, puisqu'elle dépend également de la tension drain-grille. L'injection de charge dans ce cas est difficile à prédire.

La variation du courant de polarisation montre une annulation complète de l'injection de charge par l'interrupteur de compensation autour de 500µA. Cet effet ne peut pas être prédit par le modèle analytique, car nous considérons que l'interrupteur de compensation est désapparié, et ne peut donc jamais compenser parfaitement la charge provenant de l'interrupteur utile. Pour des courants de polarisation plus importants, la capacité du noeud de drain augmente et attire plus de charge. L'interrupteur factice sur-compense alors, et *augmente* l'injection de charge. Ce problème est aggravé par la transconductance mémoire augmentée.

Les performances du *bruit* et de l'*erreur de la dérive* sont bien modélisées. Cependant, la capacité de grille est toujours sous-estimée, ce qui engendre une évaluation pessimiste des performances pour des valeurs faibles de cette capacité.

Dans cette section, nous avons tenté d'expliquer quelques unes des divergences entre le modèle analytique et la simulation numérique équivalente. Bien que des imprécisions existent, une grande proportion des caractéristiques des performances considérés ont montré un bon accord entre les valeurs prédites et simulées (Tableau 4.1). Là où les imprécisions se produisent, c'est souvent pour des valeurs peu probables en pratique de dimensions du circuit, telles que des faibles valeurs de la capacité de grille ou des tailles importantes de l'interrupteur. L'analyse qualitative a été validée, puisque toutes les courbes de simulation montrent les mêmes tendances que les courbes prédites.

✓ bon accord ▼ pessimiste (performance prédite inférieure à celle simulée) ▲ optimiste (performance prédite supérieure à celle simulée)		erreur de gain statique	erreur de diviseur capacitif	précision d'établissement	injection de charge	snr	dérive
GVO du transistor mémoire	min	✓	✓	✓	✓	✓	✓
	max	✓	✓	✓	✓	✓	✓
longueur du transistor mémoire	min	✓	✓	✓	✓	✓	✓
	max	✓	✓	✓	✓	✓	✓
GVO du transistor source	min	✓	✓	✓	✓	✓	✓
	max	✓	✓	✓	✓	✓	✓
longueur du transistor source	min	✓	✓	✓	✓	▲	✓
	max	✓	✓	✓	✓	✓	✓
longueur de l'interrupteur	min	✓	✓	✓	✓	✓	✓
	max	✓	✓	✓	▼	✓	▼
capacité de mémorisation	min	✓	▼	▲	▲	▼	▼
	max	✓	✓	▼	✓	▲	✓
courant de polarisation	min	✓	✓	✓	✓	✓	✓
	max	✓	✓	▼	▲	✓	✓
tension d'alimentation	min	✓	✓	✓	✓	✓	✓
	max	✓	✓	✓	✓	✓	✓

Tableau 4.1 Comparaison globale des résultats prédits et simulés

4.8 Conclusion

Les spécifications au niveau système, par exemple l'offset, l'erreur de gain et la distorsion harmonique, sont d'une part difficiles à estimer de manière analytique, et nécessitent d'autre part un temps CPU important pour leur évaluation par la simulation numérique. De plus, il est nécessaire de pouvoir confronter les performances prédites de la cellule à celles simulées et d'en extraire les corrections nécessaires à la boucle interne d'optimisation. Pour ces deux raisons, un environnement de simulation numérique a été développé, qui évite la simulation transitoire de longue durée en effectuant un jeu de simulations rapides correspondant aux analyses des modèles.

L'erreur de gain statique est évaluée par une analyse statique, qui élimine tout effet capacitif ou de commutation. Ces effets sont évalués dans une analyse équivalente dynamique, en prenant en compte l'erreur de gain statique déjà quantifiée. La précision d'établissement est également évaluée par une analyse transitoire, ainsi que la dérive. Le bruit est donné par une analyse fréquentielle de la cellule en acquisition.

Ce jeu d'analyses permet l'estimation de l'erreur de prédiction par rapport à la simulation numérique, qui est prise comme référence. L'erreur de prédiction est utilisée dans la boucle interne d'optimisation, en cas de besoin. Les simulations nous ont également permis de valider les modèles analytiques développés. L'utilisation de la simulation numérique comme référence n'est pas idéale, puisque la validation finale reste toujours la mesure des performances d'une cellule fabriquée. Contrairement à l'analyse, les mesures permettent une évaluation simple des spécifications au niveau système, alors que l'évaluation des erreurs individuelles constitue une tâche beaucoup plus complexe. De plus, la validation expérimentale de l'espace de conception à l'aide d'un grand nombre de cellules serait trop coûteuse.

Le désappariement et les variations du procédé de fabrication n'ont pas été considérés de manière complète. Cet aspect limite sérieusement l'outil de CA et la validité de l'environnement numérique. Des technologies submicroniques, utilisées pour fabriquer des circuits à courants commutés, sont sujettes à de grandes variations dans les paramètres technologiques, ce qui peut avoir un fort impact sur les performances du circuit. Alors que le désappariement est un problème limité aux cellules fonctionnant en mode différentiel, les variations des paramètres peuvent provoquer une insuffisance de performances et même une faute fonctionnelle. Il est donc souhaitable d'effectuer des analyses de rendement, avec l'appui de données suffisantes sur les variations des paramètres du procédé.

References

[ANA94] Anacad Electrical Engineering Software, "ELDO User's Manual", 1994

[CRA94] P.J. Crawley, G.W. Roberts, "Predicting Harmonic Distortion in Switched-Current Memory Circuits", *IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing*, vol. 41, no. 2, pp. 73-86, February 1994

[HUG96] J.B. Hughes *et al.*, "Automated Design of Switched-Current Filters", *IEEE Journal of Solid State Circuits*, vol. 31, no. 7, pp. 898-907, July 1996

[SUN95] Sun Microsystems, "SunOS 5.5.1 User's Manual", 1995

5 L'OPTIMISATION À BASE DE RÈGLES

L'optimisation numérique, dans sa forme traditionnelle, procède de façon purement quantitative, et peut appliquer des variations de paramètres qu'aucun concepteur ne prendrait en considération. L'inclusion des connaissances dans un algorithme d'optimisation réduit la taille du vecteur de variables de conception, de manière adaptée à la situation actuelle de conception. Cette approche peut augmenter considérablement l'efficacité de l'algorithme. Cependant, tous les algorithmes ne sont pas appropriés à l'optimisation des cellules à courants commutés, ni à l'inclusion des connaissances. Une réflexion attentive est nécessaire avant leur application dans un outil de conception automatisée.

A un moment donné dans le processus de conception, une optimisation est utile pour résoudre un problème donné par des moyens autres que les procédures analytiques simplifiées traditionnelles. Ceci est nécessaire, car les modèles complets de dispositif et de circuit sont trop complexes à traiter de façon manuelle, d'où les approximations faites dans les formules qui sont utilisées lors de la conception manuelle. La différence de complexité entre les modèles précis et les modèles approximatifs incite l'utilisation de logiciels d'analyse de circuits afin de les simuler. Basés sur ces résultats, les paramètres du circuit peuvent être ajustés par optimisation manuelle ou automatique, afin de satisfaire les spécifications pour un coût minimal.

Un processus typique d'optimisation manuelle consiste à effectuer la conception approchée d'un circuit à l'aide de formules approximatives et de modèles simplifiés de dispositif. Le circuit est ensuite simulé et les paramètres critiques du circuit sont ajustés de façon itérative afin de satisfaire aux spécifications et d'obtenir la meilleure solution en termes de coût (Figure 5.1).

Ce processus nécessite certaines connaissances afin :

a) de produire le dimensionnement initial du circuit, et

b) de déterminer quel(s) paramètre(s) ajuster et dans quelle plage de valeurs.

Cette dernière nécessite de connaître l'effet de la variation d'un paramètre sur les performances du circuit, ainsi que de connaître les contraintes qui affectent chaque paramètre. Les dimensions des composants intégrés, ainsi que des composants électroniques en général, sont contraintes de se cantonner à une certaine gamme de valeurs. Les problèmes de dimensionnement d'un CI sont donc des problèmes d'optimisation avec contraintes ("constrained optimization"), et l'optimisation automatisée des circuits doit prendre en compte cet aspect. L'ajustement automatique

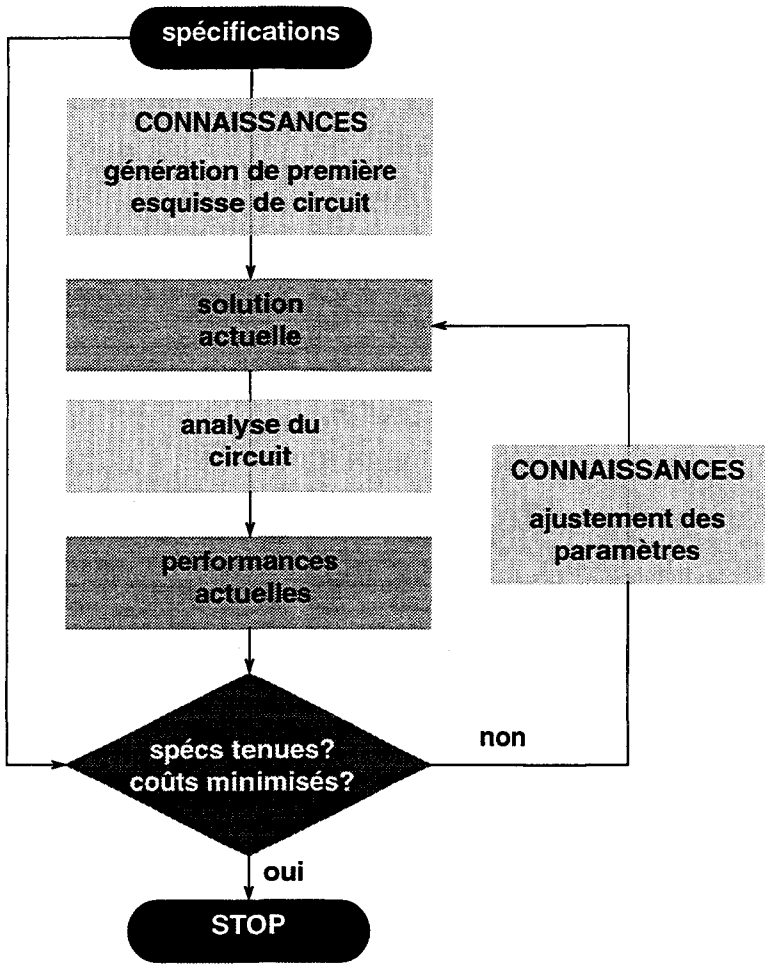


Figure 5.1 Déroulement de la conception manuelle de circuits

(et non manuelle) des paramètres dans l'enchaînement de conception de la Figure 5.1 constitue la différence principale entre les approches manuelle et par optimisation. L'optimisation peut également agir au niveau de la génération de la première ébauche du circuit, mais cette tâche constitue un problème basé sur les connaissances plutôt que basé sur l'optimisation.

Des algorithmes d'optimisation pure peuvent ajuster tous les paramètres de conception dans n'importe quelle direction. Ils comparent ensuite les performances qui résultent de l'application d'un jeu de paramètres avec celles qui résultent de l'application du jeu précédent. De cette façon, l'effet de la variation des paramètres est estimée numériquement. Les algorithmes classiques d'optimisation ne bénéficient donc pas des connaissances qualitatives de conception, du type décrit dans les deuxième et troisième chapitres. Ils procèdent au contraire sur une base purement quantitative.

L'optimisation à base de règles introduit des connaissances de conception dans le processus d'optimisation, rendant ainsi l'algorithme plus intelligent et plus efficace. L'objet de ce chapitre est de décrire le processus complet d'optimisation développé

pour la CA de cellules à courants commutés. Dans un premier temps, un résumé des techniques d'optimisation les plus courantes est présenté, suivi par une justification du choix d'un algorithme simple d'optimisation. Les méthodes qui ont été utilisées pour l'introduction de la base des connaissances dans l'algorithme global sont ensuite présentées. Enfin, des résultats d'optimisations préliminaires démontrent l'efficacité de l'inclusion des connaissances sur le processus d'optimisation.

5.1 Les méthodes d'optimisation numérique

Le problème général de l'optimisation est défini [MAS91] comme étant la minimisation d'une fonction objectif de conception ("design objective function"), Φ, d'un vecteur de variables de conception, x. $\Phi(x)$ peut s'écrire

$$\Phi(x) = \sum_{i=1}^{m} |P_{si} - P_{ri}| \tag{5.1}$$

et donne $\Phi(x)=0$ si et seulement si toutes les performances réalisées P_r sont égales aux performances spécifiées P_s. La fonction objectif (5.1) peut également s'écrire comme une fonction de type somme quadratique. Ceci donne le même résultat, mais évite l'évaluation de modules.

$$\Phi(x) = \sum_{i=1}^{m} \{P_{si} - P_{ri}\}^2 \tag{5.2}$$

5.1.1 L'optimisation avec contraintes

Les relations ci-dessus n'imposent aucune restriction sur les valeurs réalisables des variables de conception. Ces restrictions sont cependant fréquentes dans la conception, telles que

$$x_{imin} \leq x_i \leq x_{imax} \tag{5.3}$$

qui représente une contrainte sur la valeur du paramètre de conception x_i. De plus, la fonctions objectif est nulle seulement quand les spécifications sont parfaitement satisfaites. Pourtant, les spécifications peuvent aussi être décrites sous la forme

$$P_{si} \leq a \tag{5.4}$$

$$P_{sj} \geq b \tag{5.5}$$

qui constituent également des contraintes.

Ce genre de problème peut être posé formellement comme la minimisation de Φ sujette à p contraintes d'égalité

$$C_{Ej}(x) = 0, j = 1, 2, ..., p \tag{5.6}$$

et à q contraintes d'inégalité.

$$C_{Ij}(x) = 0, j = 1, 2, ..., q \tag{5.7}$$

Un exemple de situation illustré dans la Figure 5.2 montre les difficultés de l'optimisation avec contraintes. Le minimum sans contrainte se trouve dans la zone non réalisable, mais détermine aussi la direction dans laquelle les méthodes générales d'optimisation tenteront de rechercher le vecteur de solution. Le minimum avec contraintes ne représente pas nécessairement le minimum réel, et c'est la recherche du premier en présence du dernier qui pose une difficulté.

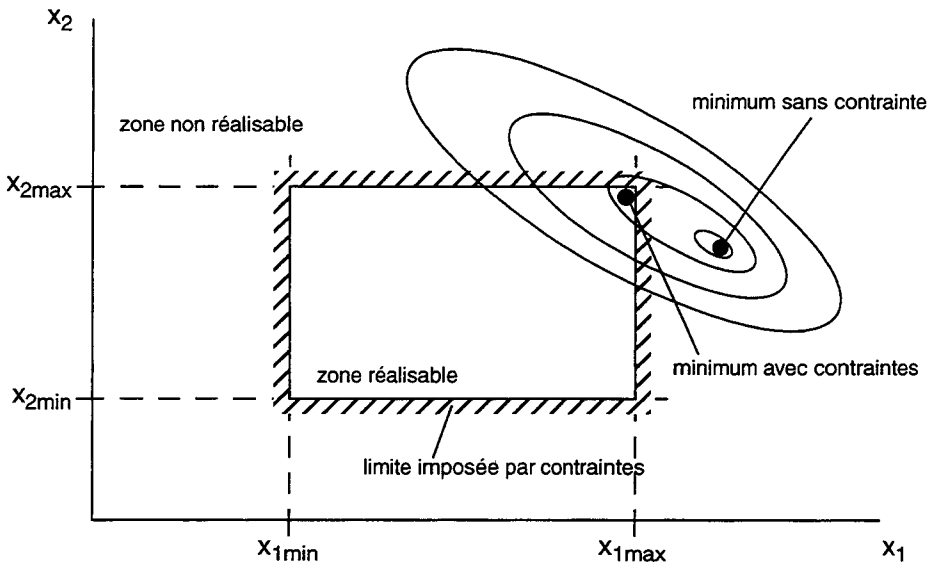


Figure 5.2 Contraintes sur une fonction objectif à deux variables

La nature de la fonction objectif et de ses contraintes déterminent le type de technique d'optimisation avec contraintes applicable à un problème donné. Il existe trois types de classification, appelés des méthodes de programmation, où le terme "programmation" signifie "optimisation avec contraintes" :

- les méthodes *linéaires* de programmation, où la fonction objectif ainsi que les fonctions de contrainte sont linéaires par rapport aux variables indépendantes de conception,
- les méthodes *quadratiques* de programmation, où la fonction objectif est de nature quadratique, et les fonctions de contrainte restent linéaires,
- les méthodes *non-linéaires* de programmation, où la fonction objectif ainsi que les fonctions de contrainte sont non-linéaires.

Ce dernier représente le cas général et constitue celui rencontré couramment dans les problèmes de conception. Il existe deux méthodes dans cette classe, qui sont résumées dans les sections suivantes.

5.1.1.1 L'approche par fonctions de pénalité

Cette méthode représente la méthode traditionnelle. La fonction d'erreur devient :

$$F(x) = \phi(x) + \sum_i W_i[C_i(x)]^2 \tag{5.8}$$

où $\Phi(x)$ est le problème de minimisation, et C_i représente les contraintes (contraintes d'égalité ou d'inégalité). Concrètement, le problème de minimisation représente les coûts (minimiser ou maximiser telle performance par rapport à telle spécification). Les contraintes d'égalité représentent les conditions (points fixes), et les contraintes d'inégalité représentent ce que nous appelons contraintes spécifiées (les spécifications de performance qui doivent être satisfaites).

Les W_i représentent des facteurs de pondération, donnés par :

$$W_i = \frac{1}{\theta_i}\left\{\frac{P_{si} - P_{ri}}{P_{si}}\right\}^2 \text{ si } P_{ri}<P_{si} \tag{5.9}$$

$$W_i = 0 \qquad \text{sinon} \tag{5.10}$$

Cette approche est valable pour une spécification similaire à (5.5). La relation d'inégalité est inversée dans le cas de (5.4).

La fonction objectif de conception devient donc effectivement une fonction pouvant être résolue par des algorithmes d'optimisation sans contrainte.

5.1.1.2 La programmation quadratique récursive

Chaque itération de cette méthode consiste à approximer la fonction réelle à une fonction qui peut être résolue par l'utilisation des techniques spécifiques à la résolution de problèmes de nature quadratique. Essentiellement, ces dernières consistent en la solution itérative d'une équation matricielle, comportant une matrice Jacobienne (les premières dérivées d'un vecteur de fonction par rapport au vecteur de variables de conception) et une matrice Hessienne (les secondes dérivées partielles d'un vecteur de fonction par rapport au vecteur de variables de conception).

L'optimisation avec contraintes par la méthode de programmation quadratique récursive est plus efficace que par l'approche des fonctions de pénalité. Cependant, l'utilisation de dérivées dans l'algorithme présente des difficultés quant à l'application de cette méthode dans une structure à base de règles spécifique aux circuits à courants commutés. Un problème concerne le traitement des erreurs. D'une itération à la suivante, plus d'une variable peut changer sa valeur. De ce fait, si le vecteur de solution

généré provoque une faute fonctionnelle du circuit par exemple, il n'est pas facile de savoir quelle variable en a la responsabilité et de quelle façon réparer la faute. L'autre problème concerne la façon dont les dérivées sont évaluées par rapport aux circuits non-linéaires. Cet aspect est considéré dans la section suivante.

5.1.2 Le calcul des dérivées

Dans les situations pratiques d'optimisation, les évaluations des dérivées sont utilisées afin de calculer des vecteurs gradients, des matrices Jacobienne et Hessienne. Il existe deux moyens d'évaluation des dérivées : l'approximation par des différences finies ; et l'évaluation précise par réseaux linéaires. Les deux approches sont maintenant considérées.

5.1.2.1 L'évaluation approximative des dérivées par les différences finies

Puisque la fonction objectif de conception $\Phi(x)$ peut être évaluée en n'importe quel point x, une approximation du gradient $g(x_0)$ au point x_0 peut être obtenue. Considérons la fonction illustrée dans la Figure 5.3.

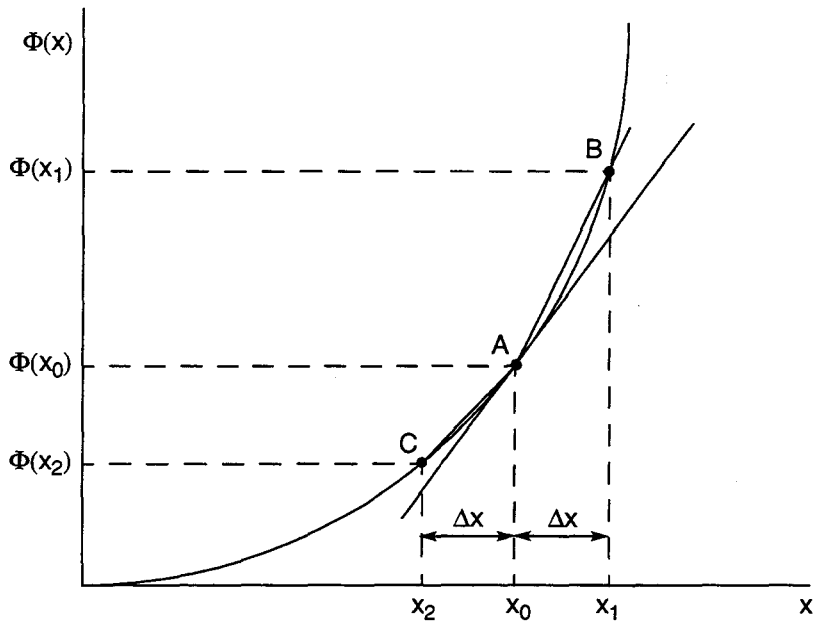


Figure 5.3 Evaluation approximative de la dérivée

Si la valeur de $\Phi(x_0)$ est connue, l'évaluation d'un point supplémentaire à $\Phi(x_0+\Delta x)$ permet l'estimation de $g(x_0)$:

$$g(x_0) = \frac{d\Phi(x)}{dx} \cong \frac{\Phi(x_1)-\Phi(x_0)}{x_1-x_0} = \frac{\Phi(x_0+\Delta x)-\Phi(x_0)}{\Delta x} \quad (5.11)$$

Cependant, cette évaluation donne la pente du segment AB plutôt que celle de la tangente à A. L'erreur peut être réduite en diminuant la distance AB, c'est-à-dire en

diminuant la valeur de Δx. Comme alternative, un troisième point peut être évalué à $\Phi(x_0-\Delta x)$, tel que $g(x_0)$ peut s'écrire :

$$g(x_0) \cong \frac{\Phi(x_1)-\Phi(x_2)}{x_1-x_2} = \frac{\Phi(x_0+\Delta x)-\Phi(x_0-\Delta x)}{2\Delta x} \quad (5.12)$$

Afin de générer un vecteur de gradient, cette méthode nécessite $2n$ analyses supplémentaires pour un problème à n variables. Si le temps d'analyse est grand, ou si n est grand, le temps d'évaluation peut être excessivement long et l'optimisation trop lente.

5.1.2.2 L'évaluation précise des dérivées par réseaux linéaires

La matrice Y des admittances nodales pour un circuit linéaire peut être facilement déterminée comme le décrit la section 2.3. La matrice Z des impédances nodales peut également se trouver facilement par l'inversion de la matrice Y. Un calcul général des dérivées peut ensuite être obtenu, ce qui permet l'évaluation explicite des dérivées à partir de la matrice des admittances nodales. Toutefois, ceci suppose qu'une matrice d'admittance nodale est disponible. Ceci est vrai pour les circuits linéaires mais, à l'inverse, les circuits non-linéaires doivent être linéarisés avant qu'une telle matrice puisse être extraite.

Dans le cas des circuits à courants commutés, cette linéarisation n'est pas possible dans le sens où ce type de circuit présente un comportement qui dépend du temps. Une cellule mémoire de courant pourrait être linéarisée afin d'évaluer les fonctions qui ne dépendent pas du temps (par exemple l'erreur de gain statique). Mais la précision d'établissement, l'injection de charge et la dérive, toutes dépendantes du temps, sont exclues de l'évaluation des dérivées par réseaux linéaires. Des méthodes qui calculent les dérivées dans les circuits non-linéaires existent bien [BRA80], mais présentent un degré de complexité supérieur à celui des méthodes présentées ici.

5.1.2.3 Conclusions

Il est donc clair que l'utilisation de techniques d'optimisation basées sur l'évaluation des gradients n'est pas applicable à la CA de cellules à courants commutés. Cette conclusion est tirée aussi bien du point de vue de l'efficacité dans le cas de la méthode des différences finies, que du point de vue de la faisabilité dans le cas de la méthode de réseaux linéaires. En conséquence, l'optimisation avec contraintes ne peut pas être effectuée par la programmation quadratique récursive, malgré l'efficacité de cet algorithme. La seule solution restante est celle des fonctions de pénalité qui convertissent un problème d'optimisation avec contraintes en un problème d'optimisation sans contrainte. Les sections suivantes analysent l'aptitude des algorithmes d'optimisation les plus communs à la solution du problème sans contrainte.

5.1.3 L'optimisation sans contrainte

Les classes principales de méthodes d'optimisation sans contrainte sont :

- *les méthodes de recherche directe*, qui n'utilisent pas de dérivée de la fonction objectif,
- *les méthodes à gradient*, qui utilisent les dérivées premières de la fonction objectif, et parfois même les dérivées d'ordre supérieur.

Les raisons citées précédemment nous contraignent à utiliser les méthodes de recherche directe, qui peuvent être classées en algorithmes d'optimisation locale ou globale. La différence réside dans le type de minimum que recherche chaque algorithme. Un algorithme d'optimisation locale, une fois commencé, ne peut que diminuer la fonction objectif, et ne peut pas quitter la vallée dans laquelle il se trouvait au début. Si le minimum global ne se trouve pas dans cette vallée, il ne sera pas trouvé. D'où la nécessité d'un point de départ de bonne qualité pour ces algorithmes (Figure 5.4). Cette considération est importante pour les outils généralisés de CA, puisque les points de départ sont difficiles à formuler de façon générale.

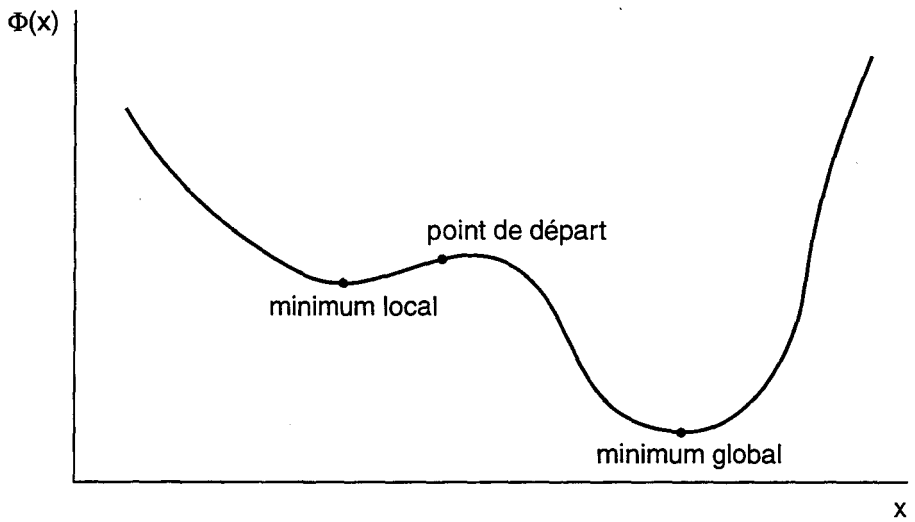


Figure 5.4 Fonction monovariable avec minima local et global

A l'inverse, un algorithme d'optimisation globale n'a pas besoin d'un point de départ, procédant en général de façon probabiliste. Ces algorithmes peuvent augmenter la fonction objectif. Ce genre d'algorithme constitue *a priori* une meilleure base d'optimisation.

Les principes de chaque approche, ainsi que leurs avantages et inconvénients respectifs, sont maintenant détaillés.

5.1.3.1 Les méthodes d'optimisation globale

Afin de surmonter le problème de l'optimisation vers un minimum local plutôt que global, un certain nombre de méthodes existent. Nous pouvons les classer dans deux catégories :

- une recherche *exhaustive*;
- une recherche *statistique*.

Une recherche exhaustive peut procéder selon de nombreuses manières. Si l'espace de conception est de taille peu importante par rapport au nombre de variables, une évaluation d'un nombre de points finis permet de trouver la meilleure solution. Ceci suppose que la différence entre chaque point engendre une variation suffisamment petite dans la fonction objectif afin d'obtenir une image détaillée de la surface de la fonction. Toutefois, si l'espace de conception est grand avec un nombre plus important de variables (comme souvent), cette solution devient rapidement irréalisable. Une approche similaire génère une estimation approximative de la surface et la raffine de façon successive jusqu'à ce que la surface devienne suffisamment détaillée. Cependant, plus le maillage de l'espace de conception est grossier, plus la probabilité que cette méthode converge vers un minimum local est grande. Des méthodes plus pratiques utilisent des techniques à points de départ multiples, ces derniers étant déterminés soit de façon aléatoire, soit à partir d'une grille régulière. Ensuite, un algorithme d'optimisation locale permet de détailler la surface en-dessous de chaque point. La convergence globale n'est toujours pas assurée.

Dans la catégorie des méthodes statistiques, une seule méthode est réellement disponible. C'est la méthode de recuit simulé [KIR83], qui se base sur une génération aléatoire des déplacements et sur une acceptation statistique de ces derniers. L'algorithme génère un déplacement vers un nouveau point dans l'espace de conception, sélectionnant de manière statistique de nouvelles valeurs de variables de conception dans une certaine limite autour de leurs valeurs actuelles. La fonction objectif se calcule ensuite pour le nouveau vecteur des variables de conception. Si la fonction objectif est évaluée à un niveau inférieur au niveau précédent, le déplacement est accepté. A l'inverse, si elle a augmentée, le déplacement n'est pas nécessairement rejeté, comme c'est le cas dans les algorithmes d'optimisation locale. En se basant sur la "température" et la probabilité d'acceptation de déplacements "montants", une décision est prise concernant l'acceptation ou le rejet du déplacement de façon statistique. La température (nommée ainsi à cause de la similarité de l'algorithme avec le recuit physique des matériaux), la probabilité d'acceptation et les limites de déplacement de chaque variable de conception sont grandes au début de l'optimisation, et diminuent de façon graduelle au cours de la convergence vers un minimum global. Le fait que l'algorithme permette des augmentations dans la valeur de la fonction objectif est la clé de son succès. Cependant, il nécessite en général un grand nombre d'évaluations de la fonction objectif. L'utilisation des règles permet de réduire le nombre d'évaluations, en inhibant le déplacement de certaines variables selon les performances du circuit en cours de synthèse. Mais une telle intervention serait difficile à coder, et toucherait aux propriétés de convergence globale de cet algorithme, qui constituent son principal intérêt.

Il en va de même pour les autres algorithmes de ce type, les algorithmes "génétiques" [REN95]. Ceux-ci génèrent une population d'individus (vecteurs de variables de conception), chacun possédant sa caractéristique de survie (évaluation des performances). La variation aléatoire ou statistique du vecteur de variables de conception de chaque individu produit une nouvelle génération. Après un certain nombre de générations (itérations), la population tend vers une majorité d'individus ayant le facteur de survie le plus élevé, c'est-à-dire ayant les meilleures performances. De même qu'avec le recuit simulé, cette technique requiert un grand nombre d'itérations. Bien que l'introduction des connaissances pour le guidage du processus d'évolution paraisse être simple, la convergence vers un minimum global n'est plus garantie.

5.1.3.2 Les méthodes d'optimisation locale par recherche directe du minimum

L'algorithme le plus commun appartenant à cette catégorie est probablement celui de Hooke et Jeeves. La méthode se base sur une approche à deux phases :

1. Phase d'exploration

Le point de base courant x^i est modifié en ajoutant séquentiellement des incréments fixes Δx à chaque variable. De ce fait,

$$x_j^{i+1} = x_j^i + \Delta x, \quad \text{si } \Phi(x_j^{i+1}) < \Phi(x_j^i), \tag{5.13}$$

sinon

$$x_j^{i+1} = x_j^i - \Delta x, \quad \text{si } \Phi(x_j^{i+1}) < \Phi(x_j^i), \tag{5.14}$$

sinon

$$x_j^{i+1} = x_j^i, \quad \text{pour } j=1, 2, ..., n \tag{5.15}$$

Si au moins une variable est modifiée de façon à provoquer une réduction de la fonction objectif, $x^{i+1} \neq x^i$. A l'inverse, le pas Δx est réduit. Une phase d'exploration réussie pour un système à deux variables est illustrée dans la Figure 5.5.

2. Phase de déplacement de motif

La phase de déplacement de motif étend la phase d'exploration en supposant que la fonction objectif peut être davantage réduite en continuant dans la direction déterminée par la phase d'exploration. Un nouveau vecteur de solution x^{i+2} est déterminé de telle sorte que

$$x^{i+2} = 2x^{i+1} - x^i \tag{5.16}$$

Une phase supplémentaire d'exploration est ensuite effectuée à partir de x^{i+2} afin de générer un nouveau point x^{i+3}. La phase de déplacement de motif a réussi si

$$\Phi(x^{i+3}) < \Phi(x^{i+1}) \tag{5.17}$$

Si tel est le cas, une phase supplémentaire de déplacement de motif s'effectue en

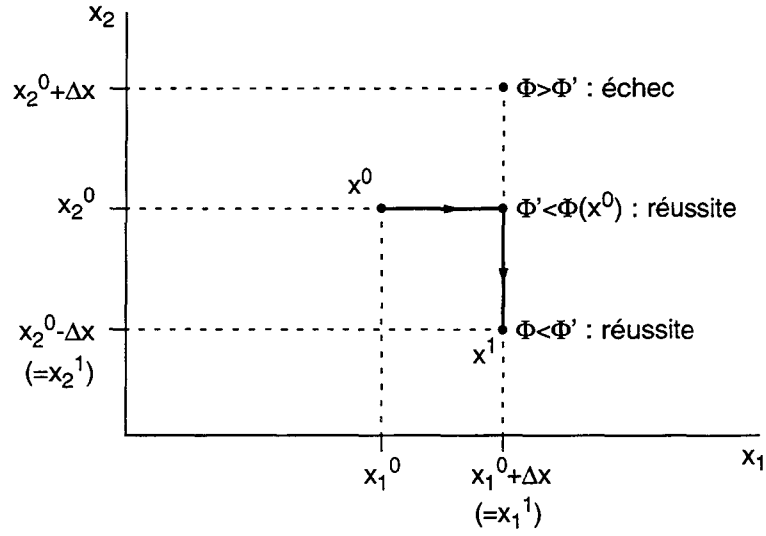


Figure 5.5 Hooke et Jeeves : phase d'exploration

remplaçant x^i et x^{i+1} par x^{i+1} et x^{i+3} respectivement. La distance vectorielle est donc augmentée, ce qui a pour objectif d'accélérer la convergence. Si la phase de déplacement de motif échoue, c'est-à-dire que (5.17) n'est pas vraie, alors x^i est remplace par x^{i+1}, et une nouvelle phase d'exploration commence (Figure 5.6)

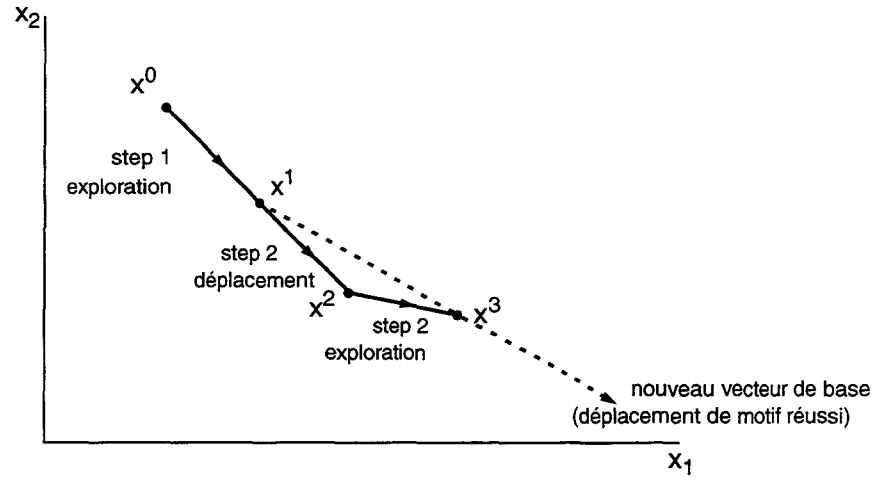


Figure 5.6 Hooke et Jeeves : déplacement de motif

L'algorithme de Hooke et Jeeves est généralement considéré comme un algorithme robuste, qui réussit dans de nombreux cas. Il est également simple à coder et à tester, comme la plupart des méthodes d'optimisation à recherche directe. De ce fait, son extension afin d'incorporer des règles de connaissances ne présente aucun problème.

Il reste le problème de la nécessité d'un point de départ. Ceci constitue le plus grand inconvénient de cet algorithme, car le point de départ n'est pas facile à formuler de façon générale. Néanmoins, quand le domaine d'application est réduit, comme pour la classe limitée des circuits à courants commutés, une solution faisable peut être trouvée.

5.1.3.3 Choix de l'algorithme

Le choix de l'algorithme d'optimisation est donc pratiquement limité à deux algorithmes : Hooke et Jeeves et la catégorie des algorithmes statistiques. Les points clés de chaque algorithme sont résumés ci-dessous :

Hooke et Jeeves	Algorithmes Statistiques
simple à coder	difficile à coder
implémentation facile des règles	difficile d'implémenter les règles - possibilité d'une dégradation des performances dans ce cas
convergence locale - besoin d'un point de départ	convergence globale - aucun besoin du point de départ
nombre peu important d'évaluation des performances	nombre important d'évaluation des performances

Apres réflexion, il apparaît clair que l'algorithme simple de Hooke et Jeeves soit le plus approprié à un outil de CA à base de règles spécifique aux circuits de courants commutés. Le problème de la nécessité d'un point de départ de bonne qualité pour chaque optimisation est compensé par le fait que les connaissances dont cette génération a besoin ont déjà été développées sous la forme d'équations analytiques décrites dans les deuxième et troisième chapitres. La façon dont cette information est implémenté dans l'outil est décrite dans la section suivante.

5.2 L'architecture de l'algorithme à base de règles

La section précédente a décrit les techniques principales de l'optimisation, qui existent et qui sont d'utilisation courante. Par un processus d'élimination, il a été démontré que l'algorithme le plus adapté au "dopage de connaissances" est l'algorithme Hooke et Jeeves, comme schématisé par la Figure 5.7. En tant qu'algorithme d'optimisation purement numérique, ce dernier est relativement inefficace, nécessitant en moyenne 3n/2 évaluations par cycle, où n est le nombre de variables de conception indépendantes (la taille du vecteur de variables). Il tend aussi à ralentir en se rapprochant du minimum.

Afin d'améliorer l'efficacité de l'algorithme de Hooke et Jeeves, les connaissances sont incluses sous la forme d'heuristiques, ou règles qualitatives de conception, développées dans les deuxième et troisième chapitres. Etant donnée une situation intermédiaire de conception, le principe de notre approche est que l'optimiseur appliquera la variation du paramètre de conception la plus susceptible d'améliorer les performances. Le choix du paramètre de conception qui sera modifié et dans quelle direction cela sera fait, est déterminé par la simple inspection de toutes les heuristiques disponibles afin de réduire l'erreur de contrainte maximale. Autrement dit, cette variation doit améliorer la performance la plus hors-spécification. Il est donc nécessaire d'implémenter la structure de règles d'une façon cohérente afin de tirer le bénéfice maximal de l'inclusion des connaissances dans le processus d'optimisation.

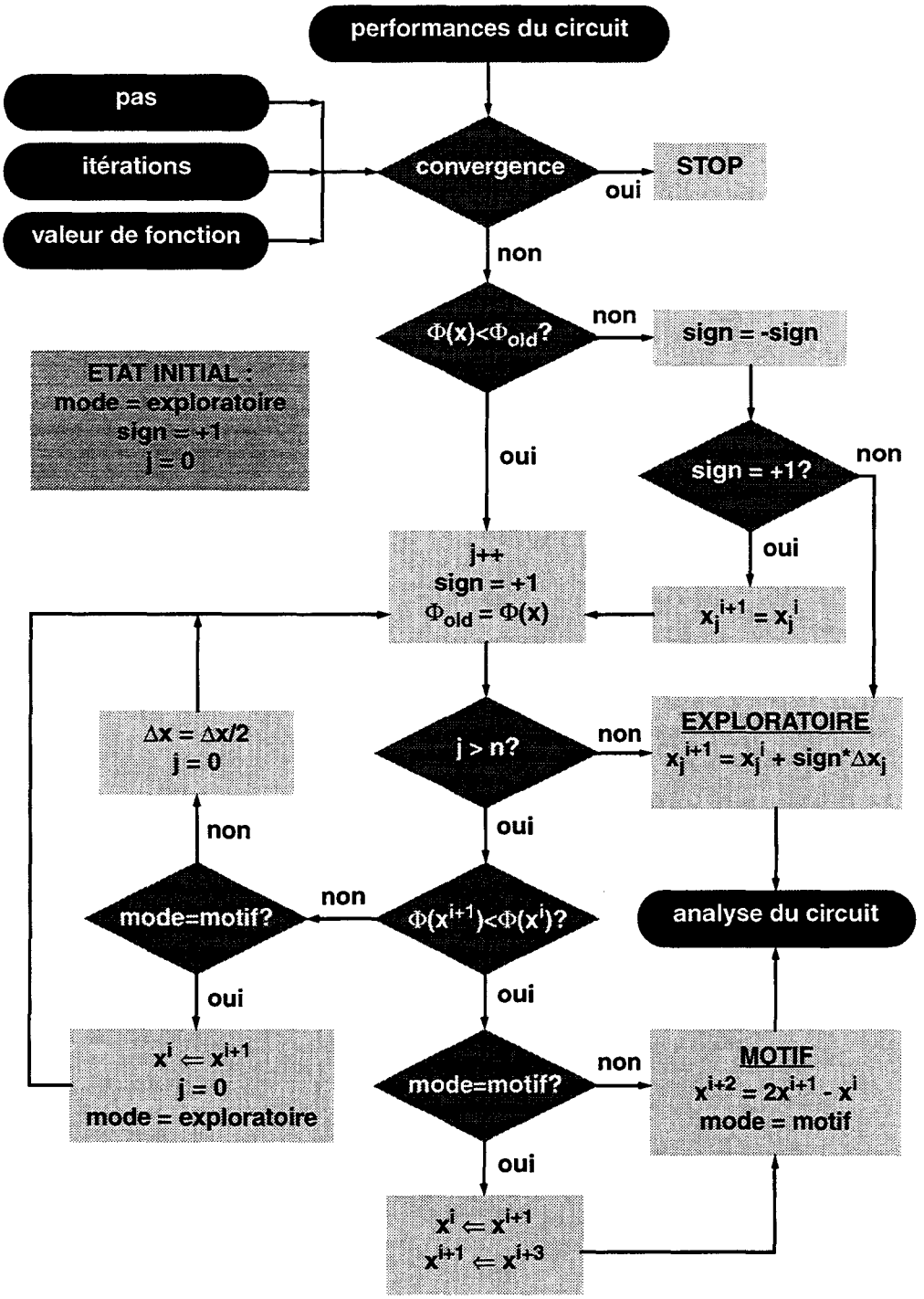


Figure 5.7 Hooke et Jeeves : organigramme

Cependant, le dopage par connaissances n'est pas la seule modification devant être effectuée afin de construire la procédure finale d'optimisation. La première étape est d'accommoder les contraintes, c'est-à-dire de transformer la méthode sans contrainte en une méthode avec contraintes. Essentiellement, nous considérons cette tache comme

une tache d'interface, par laquelle des tests de contraintes dévieront l'enchaînement vers différents points de l'algorithme sans contrainte. Ce concept est considéré plus en détail dans la section suivante.

5.2.1 Transformation en un algorithme avec contraintes

Les contraintes avec lesquelles l'outil de conception doit procéder ne sont pas homogènes. Trois types de contraintes peuvent être identifiées :

- les contraintes de *dimension*;
- les contraintes *fonctionnelles*;
- les contraintes *spécifiées*.

Les deux premiers types de contraintes sont implicites, et constituent des considérations de faisabilité. Les *contraintes de dimension* décrivent les limites physiques des tailles des composants. Par exemple, un transistor a une longueur minimale et une largeur minimale. En général nous pouvons également spécifier des dimensions maximales afin de restreindre l'espace de conception à des solutions réalistes en terme de surface active ainsi qu'en terme de rendement. Ce type de contrainte ne nécessite pas une évaluation des performances pour déterminer la violation d'une contrainte.

Une *contrainte fonctionnelle*, implicite à la topologie et indépendante des spécifications, est déterminée par la zone de fonctionnement du circuit. Un exemple de contrainte fonctionnelle veut que tous les transistors non-interrupteurs restent dans la zone de saturation pour tous les courants d'entrée. Une violation de ce type de contrainte peut également être détectée sans avoir à effectuer une évaluation des performances, bien qu'une analyse statique limitée doive être accomplie dans ce cas. Si la violation d'une contrainte est détectée, l'évaluation des performances n'a pas d'utilité car le circuit ne fonctionnera pas correctement et doit être rejeté dans tous les cas.

Les *contraintes spécifiées* sont définies par l'utilisateur et déterminent les performances du circuit requis. Les violations de ces contraintes ne peuvent être détectées que par une évaluation des performances. Une telle contrainte ne peut jamais être négative : quand elle est satisfaite, sa contribution à la fonction objectif est nulle.

Il est donc clair que le traitement des contraintes dans le processus d'optimisation est scindé en deux parties : une analyse de faisabilité pour les contraintes de dimensions et les contraintes fonctionnelles ; et une analyse des performances pour les contraintes spécifiées. L'optimiseur doit donc pouvoir distinguer chaque type de contrainte, car l'action qu'il entreprend pour chacune d'elles est très différente.

Cette philosophie donne l'architecture du module de dimensionnement, illustrée dans la Figure 5.8.

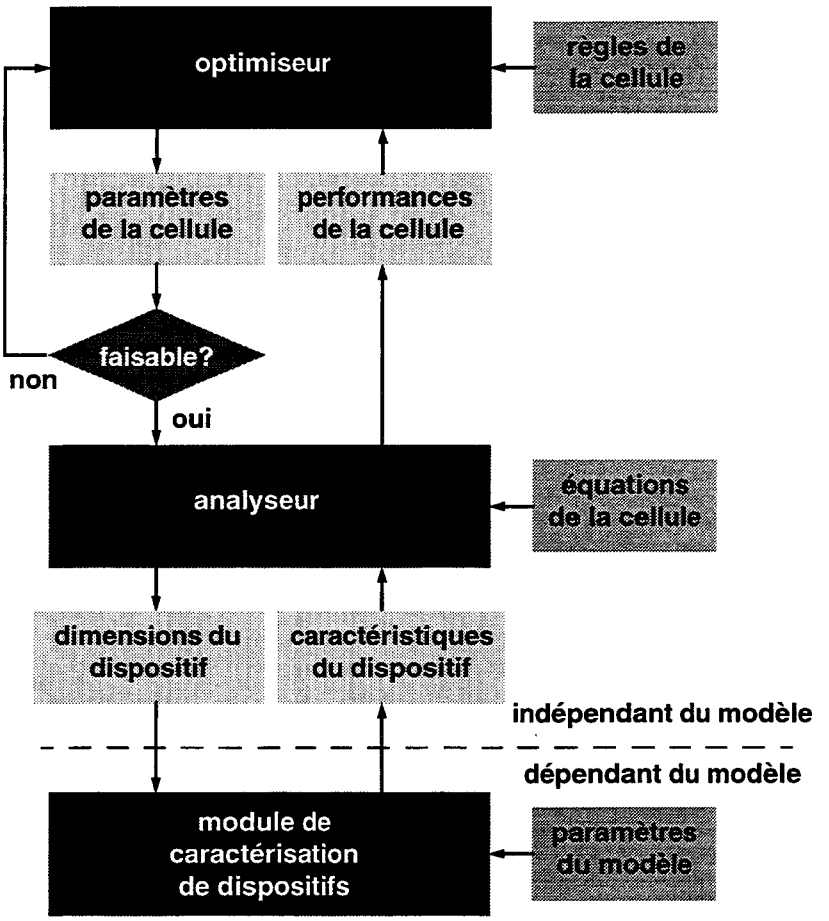


Figure 5.8 Architecture du module de dimensionnement

L'algorithme de l'optimiseur a été représenté schématiquement comme une boite noire qui utilise les performances de la cellule courante et/ou les résultats de l'étude de faisabilité, ainsi que les règles qui guident l'optimisation de la cellule. En se basant sur ces facteurs, l'optimiseur génère un nouveau jeu de paramètres de conception, qui est ensuite passé au bloc d'analyse de faisabilité.

C'est ici que les contraintes implicites sont vérifiées : les restrictions de taille physique, ainsi que l'étude statique. L'analyse réelle dépend de la topologie à dimensionner, mais en général l'analyse détermine la zone de fonctionnement de certains transistors en certains points représentatifs de la dynamique d'entrée :

$$i_{in} = \{-i_0, 0, i_0\} \tag{5.18}$$

Si une violation de contrainte fonctionnelle se produit, l'optimiseur doit être capable d'appliquer les règles conçues pour résoudre de tels problèmes.

Une fois que la faisabilité d'un circuit a été validée, les paramètres sont envoyés vers l'analyseur afin d'évaluer les performances du circuit. L'analyseur utilise les équations

détenues par la topologie afin de générer les résultats. Les équations sont en fait celles développés dans les chapitres traitant de la modélisation comportementale des cellules. Ces équations sont écrites en termes de paramètres intrinsèques aux dispositifs. Afin d'évaluer ces paramètres et ainsi de compléter la boucle de conception, l'analyseur doit invoquer le bloc de caractérisation des dispositifs, qui implémente les modèles autonomes du dispositif considéré. Les paramètres intrinsèques du dispositif sont calculés par rapport à ses dimensions, à ses conditions de polarisation, à ses paramètres technologiques et au modèle de dispositif utilisé.

A partir de ces différents points d'entrée du bloc de l'optimiseur, une interface est requise afin de diriger les données provenant de l'analyse du circuit vers les chemins appropriés de l'algorithme (Figure 5.9).

Dans le cas d'une faute physique ou d'un déplacement non réussi, l'action est la même. Le paramètre actuel de conception reprend son ancienne valeur et l'optimisation continue, avec la connaissance du fait que le dernier déplacement a constitué un échec. Si une faute fonctionnelle est détectée dans le circuit, la première utilisation des règles apparaît. L'interface peut détecter si la faute provient d'une analyse statique avec un courant d'entrée de valeur nominale, maximale ou minimale. Les règles propres à chaque cas sont différentes. Par exemple, si le circuit est fautif pour un courant d'entrée nominal, il serait souhaitable de rehausser les tensions de grille du transistor mémoire tout en conservant la même dynamique. Si par contre le circuit est fautif pour un courant d'amplitude maximale, il serait peut-être plus avantageux de réduire la dynamique sur la tension de grille, tout en retenant une tension nominale de grille identique. Le traitement de ces règles et de celles qui se focalisent sur des contraintes spécifiques est détaillé dans la section suivante.

5.2.2 Incorporation de la base de règles

5.2.2.1 Structure des règles

Les heuristiques qui constituent les règles définissent les relations qualitatives entre variable de conception et critère de performance. Les règles peuvent être classées selon le critère de performance (et le paramètre de conception qui peut être varié afin de l'améliorer) ou bien la variable de conception (et l'effet qu'a une variation positive sur les divers critères de performance). Chaque approche a différentes implications, qui sont maintenant illustrées (Figure 5.10).

Dans le premier cas, il est nécessaire pour l'optimiseur de sélectionner la règle à appliquer, en fonction des performances actuelles de la cellule. Le mécanisme de sélection le plus évident est de sélectionner la règle qui améliore le critère de performance le plus hors-spécification, c'est-à-dire celui qui possède la valeur individuelle de fonction objectif de conception la plus élevée[1]. Mais de cette façon,

1. D'autres mécanismes de sélection peuvent être envisagées, permettant de retarder le changement de règle. Ceci favoriserait la convergence individuelle et réduirait les effets de commutation entre deux règles : cette approche n'a cependant pas été retenue pour l'outil prototype.

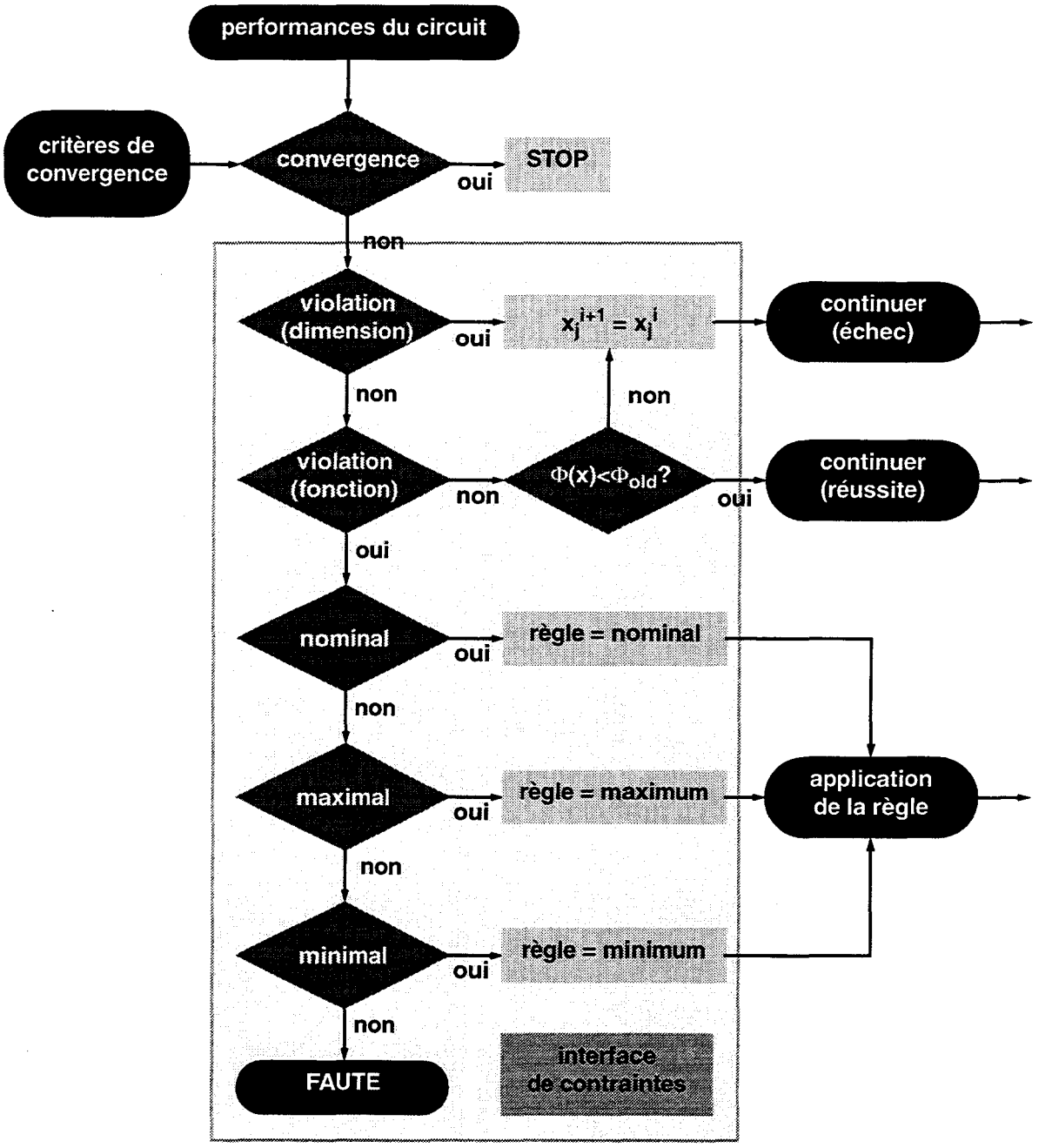


Figure 5.9 Interface de contraintes vers algorithme sans contrainte

l'effet de l'application du jeu d'heuristiques de conception choisi sur d'autres critères de performance n'est pas pris en compte, et le processus d'optimisation peut se bloquer entre deux règles opposées d'heuristiques. Cependant, ce n'est vrai que si l'algorithme est autorisé à sauter entre les règles (voir la section 5.2.2.3). Car il est peu vraisemblable que deux règles existent et possèdent des heuristiques exactement opposées.

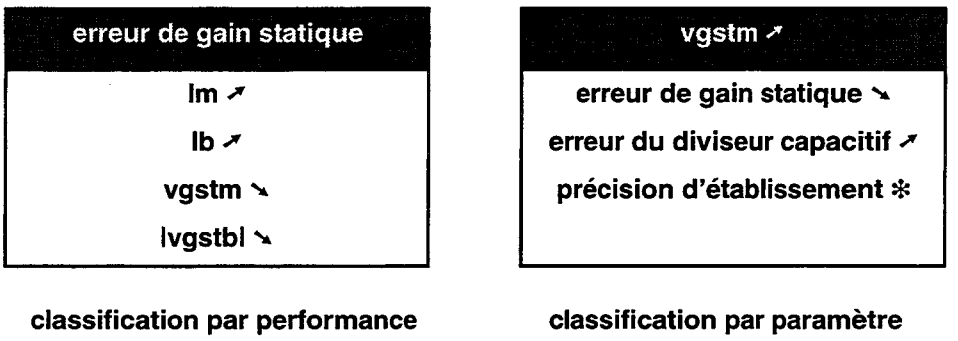


Figure 5.10 Classification des règles

Dans le deuxième cas, en construisant les règles à partir des heuristiques rattachées à un paramètre de conception, l'optimiseur trouve le meilleur paramètre à faire varier pour une situation de conception particulière. Dans l'exemple montré dans la Figure 5.10, une situation dans laquelle il serait avantageux d'appliquer une variation positive à v_{gstm} comporte la sur-réalisation de la spécification de l'erreur de gain statique et la non-réalisation de celle du diviseur capacitif. L'état de la spécification de la précision d'établissement, étant ambigu, n'apporte rien à la décision concernant l'application de la règle. Les limitations deviennent rapidement évidentes : les chances sont faibles de se trouver dans une situation de conception identique à celle décrite par les règles ; et les heuristiques bidirectionnelles, ou les heuristiques "ambiguës" (voir la section 5.2.2.2), sont difficiles à traiter.

Finalement, le premier cas constitue la meilleure solution, puisqu'il donne une plus grande flexibilité à l'optimiseur, et les heuristiques ambiguës sont mieux contrôlées. Le problème concernant les conflits entre heuristiques peut se résoudre en introduisant une épuration de la règle afin d'éliminer les heuristiques qui sont en conflit directe avec d'autres critères non-satisfaits.

5.2.2.2 Les heuristiques ambiguës

La plupart des heuristiques ne sont pas ambiguës, puisqu'elles donnent une directive claire sur la façon dont un paramètre doit varier afin d'améliorer une performance donnée. Ces heuristiques correspondent aux fonctions monotones de l'espace de conception. Par contre, là où les espaces de conception sont non-monotones, les heuristiques sont nécessairement ambiguës car, selon la valeur du paramètre de conception, la fonction peut augmenter ou diminuer pour un sens de variation donné du paramètre. Cette information n'est pas connue *a priori*, ce qui est illustré avec l'exemple de la précision d'établissement.

Le pôle dominant de la cellule s'écrit

$$p_1 = \frac{g_m}{C_m} \tag{5.19}$$

Supposons que le circuit ait une réponse sur-amortie (c'est-à-dire que le pôle soit très dominant et que le système puisse être approximé à un circuit RC à pôle unique). Une réduction du pôle par l'augmentation de C_m ou v_{gstm} ralentira davantage la réponse et la précision d'établissement diminuera. Considérons maintenant l'autre cas, la réponse sous-amortie (c'est-à-dire que le pôle est proche d'un autre, défini par la capacité parasite du drain ; de sorte que la réponse oscille avant de se stabiliser). Une réduction du pôle principal dans cette configuration augmentera l'écart de fréquence entre les deux pôles, donc la stabilité du circuit. De ce fait, la diminution du pôle augmentera la précision d'établissement.

Les solutions à cette ambiguïté sont :

- de décomposer la fonction de précision d'établissement en ses composantes sur-amortie et sous-amortie, et développer des heuristiques pour chacun des cas, ou
- de déterminer numériquement le gradient de la fonction donnée pour le vecteur de variables de conception, ou
- d'essayer successivement les deux directions en évaluant les effets.

La première solution est certainement la plus efficace, puisqu'elle ne requiert aucune évaluation de fonction supplémentaire et donne les heuristiques correctes immédiatement. Cependant, la précision d'établissement n'est pas la seule fonction non-monotone, et cette solution ne peut pas être étendue facilement à d'autres fonctions.

La deuxième solution s'applique facilement dans le cas général, mais nécessite deux évaluations de fonction supplémentaires afin d'estimer le gradient. Aussi, si la fonction se trouve à un point stationnaire, le gradient est nul, ce qui réduit la fonctionnalité de cette approche. Le cas d'un point stationnaire maximal est plus probable, car un point stationnaire minimal signifie qu'un minimum (local ou global) a été trouvé. En évaluant le gradient de la fonction à zéro, l'algorithme ne peut plus continuer.

La dernière solution est la plus facile à implémenter et la plus proche de la philosophie originale de Hooke et Jeeves, mais elle peut nécessiter une analyse supplémentaire de toutes les performances si la première tentative échoue. Néanmoins, cette solution a été choisie, car elle peut difficilement échouer, et cela constitue un avantage déterminant sur la deuxième solution. En effet, pour le cas d'un point stationnaire maximal, la première tentative résulte en une diminution de la fonction objectif, et sera acceptée (Figure 5.11). Par contre, le cas d'un point stationnaire minimal constitue toujours un cas d'indécision, mais est rarement rencontré.

5.2.2.3 Le saut entre règles

Un concepteur de circuits ne continue pas à améliorer une performance individuelle si sa valeur spécifiée est atteinte, et que d'autres critères de performance ne sont pas réalisés ou seraient même dégradés par une amélioration continuelle de cette première performance. L'amélioration d'une performance individuelle devrait même être interrompue si un autre critère de performance s'éloignait trop de l'objectif.

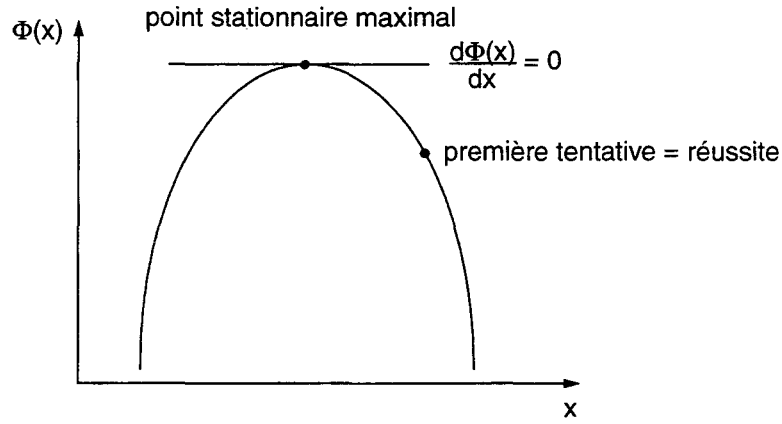


Figure 5.11 Espace non-monotone : résolution par gradient et par double tentative

L'optimiseur doit donc pouvoir "sauter" de liste d'heuristiques (c'est-à-dire entre règles[1]) quand l'erreur maximale des performances ne correspond plus à la règle utilisée, et nécessite l'utilisation d'une nouvelle règle.

Cet aspect est traité par l'optimiseur dans le mode "lemming" : quand il est activé, la règle sera implémentée jusqu'à l'heuristique finale, quel que soit le changement dans la situation de conception. Quand ce mode est désactivé, le saut entre règles est autorisé.

Le saut entre règles affecte l'algorithme d'optimisation en rendant difficile à implémenter les phases de déplacement de motif. Comme la liste des heuristiques n'est pas complètement implémentée, la direction qui serait utilisée pendant la phase de déplacement de motif peut être mauvaise. Cela peut même augmenter la fonction objectif global de conception si la liste des heuristiques n'est pas appropriée à l'erreur maximale (qui peut avoir changé). En éliminant la phase de déplacement de motif de l'algorithme de Hooke et Jeeves, ce dernier est transformé en une méthode simple de recherche aux monovariations ("univariate search"). Le coeur de l'algorithme devient moins efficace, mais l'inclusion des connaissances compense largement cet aspect.

L'autre effet du saut entre règles est que l'ordre des heuristiques dans la règle devient important. Plus l'heuristique se trouve à un niveau élevé dans la liste, plus il est probable que cette heuristique soit utilisée avant que l'optimiseur ne change de règle. Ceci peut être bénéfique si les heuristiques peuvent satisfaire dans leur ensemble une spécification implicite. Par exemple, si toute heuristique appelant à augmenter la capacité de grille se trouve systématiquement au bas de toutes les listes, une solution efficace en termes de surface pourrait apparaître plus rapidement. A l'inverse, cette spécification implicite est fixe, et peut rendre l'algorithme moins flexible vis à vis d'autres spécifications.

Le saut entre règles peut également rendre l'optimisation moins efficace en introduisant des conflits circulaires d'heuristiques, qui ne peuvent pas être nécessairement détectés par les mécanismes d'épuration (Figure 5.12).

1. Une règle est une liste d'heuristiques.

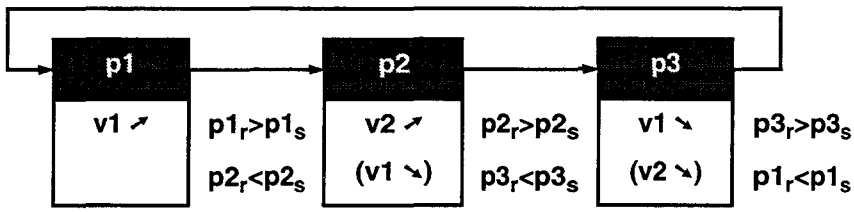


Figure 5.12 Conflits circulaires de heuristiques

Au début, la performance réalisée p_1 est hors spécification. La règle appliquée augmente la variable v_1. La performance p_2, qui satisfaisait sa spécification avant l'application de la règle, ne la satisfait plus maintenant. La variable v_2 doit alors être augmentée pour que la spécification soit à nouveau satisfaite. La performance p_3 devient alors la spécification à satisfaire, pour laquelle v_1 est diminuée, retournant à la situation initiale. Les listes de règles doivent donc être assemblées en prenant cet éventualité en compte.

5.2.2.4 La gestion des coûts

Le dernier problème rencontré pour l'implémentation d'un algorithme cohérent est la gestion des coûts de performance par rapport aux contraintes de performance. Les contraintes représentent les spécifications qui doivent être satisfaites, et les coûts représentent les spécifications qui doivent être minimisées ou maximisées. Deux approches sont possibles, et aucune ne présente d'avantage clair sur l'autre :

- la gestion *séquentielle* de coûts, où les coûts sont inclus dans la fonction objectif de conception dès que les contraintes sont satisfaites,
- la gestion *combinée* de coûts, où les coûts sont incorporés avec les contraintes dans la fonction objectif de conception dès le début du processus d'optimisation.

La première méthode a l'avantage de maximiser la probabilité qu'un circuit faisable soit trouvé ; cependant, le fait que les coûts ne soient pas pris en compte avant une étape relativement tardive du processus de conception pourrait nuire à la qualité du minimum trouvé à la fin de la procédure de synthèse. La deuxième méthode peut générer une solution plus efficace en termes de coût. Mais, comme les coûts peuvent être minimisés avant satisfaction de toutes les contraintes, cela peut empêcher l'algorithme de converger. Les deux solutions ont été implémentées, le choix de leur utilisation étant laissé à l'utilisateur.

L'algorithme final est montré dans la Figure 5.13.

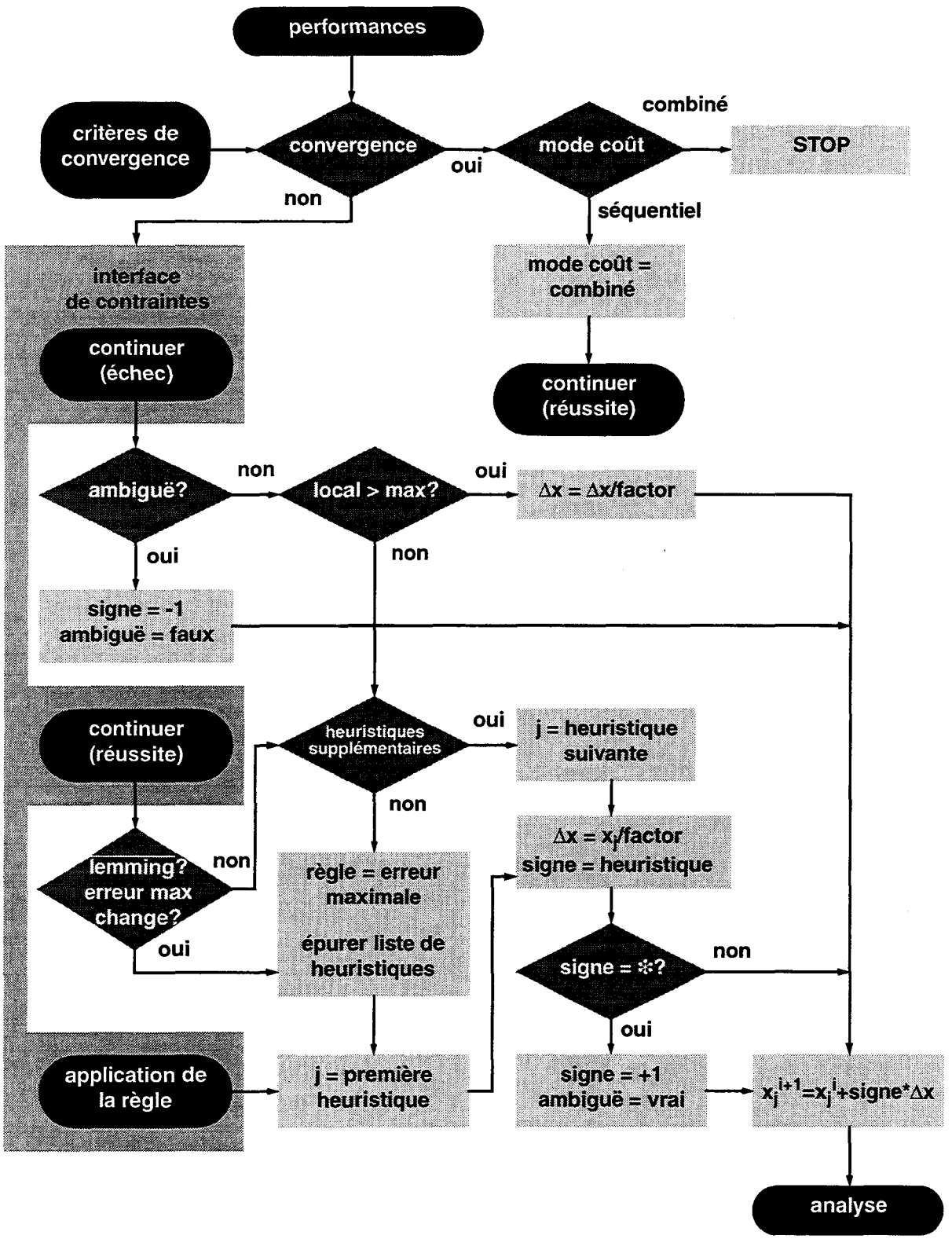


Figure 5.13 Algorithme complet d'optimisation à base de règles

5.3 Validation de l'algorithme

Afin de valider les hypothèses utilisées précédemment, une analyse comparative est présentée, qui démontre l'avantage de l'optimisation à base de règles sur la recherche aux monovariations. Les choix suivants sont proposés au concepteur :

- l'optimisation à base de règles ou par la recherche aux monovariations,
- le saut autorisé entre règles, ou à l'inverse le mode lemming,
- la gestion des coûts séquentielle ou combinée.

Trois choix binaires donnent huit combinaisons possibles :

cas	optimisation	mode lemming	gestion de coûts
A	règles	désactivé	séquentielle
B	règles	désactivé	combinée
C	règles	activé	séquentielle
D	règles	activé	combinée
E	monovariations	désactivé	séquentielle
F	monovariations	désactivé	combinée
G	monovariations	activé	séquentielle
H	monovariations	activé	combinée

Les modes concernant le saut entre règles et la gestion de coûts sont configurables à partir de l'algorithme. Il a été supposé que le saut entre règles nécessiterait l'épuration des heuristiques en conflit, alors que si le saut entre règles n'est pas activé, l'épuration n'est pas nécessaire.

Nous forçons la recherche aux monovariations en déclarant toutes les heuristiques ambiguës. Le jeu de variables de conception présenté à l'optimiseur constitue alors tous les degrés de liberté qui existent.

L'optimiseur peut être configuré afin de relâcher ou de resserrer ses critères de convergence par les paramètres suivants :

- le nombre maximal d'itérations locales, l_{max}, qui correspond à un nombre maximal de divisions de l'incrément initial, pour une variable donnée. Une plus grande valeur donne une optimisation plus fine.
- le nombre maximal d'itérations sans progrès, p_{max}. C'est le nombre maximal d'itérations pendant lequel aucune réduction d'erreur n'est effectuée. Le facteur du pas ("step factor") est augmentée quand cette situation est détectée.
- facteur du pas maximal, f_{max} : la valeur la plus élevée du facteur du pas. Quand ce dernier dépasse f_{max}, l'optimisation est supposée avoir convergé.

Le problème d'optimisation suivant a été utilisé pour chaque cas sur la topologie de base :

contraintes	coûts	conditions
erreur statique < 5%	minimiser puissance (1mW)	tension d'alimentation = 5V
erreur capacitive < 5%	minimiser surface (100µm^2)	échantillonnage = 20MHz
précision > 10 bits		courant d'entrée = 100µA
injection de charge < 5%		cellules connectées = 1
SNR > 60dB		
dérive < 1mA/s		

La configuration commune de l'optimiseur est la suivante pour ce problème :

$$l_{max} = 3 \tag{5.20}$$

$$p_{max} = 4 \tag{5.21}$$

$$f_{max} = 4 \tag{5.22}$$

La Figure 5.14 montre la convention utilisée pour l'évaluation des résultats de convergence. Une convergence réussie consiste en la satisfaction de toutes les contraintes avant terminaison du processus d'optimisation. Deux points de mesure sont alors disponibles : la somme finale des coûts, s ; et le nombre total d'itérations, i. Quand les contraintes ne sont pas toutes satisfaites (convergence non-réussie), un troisième point de mesure peut être relevé. Ceci est le taux de non-satisfaction des contraintes, t, nous indiquant l'écart restant entre les contraintes non-satisfaites et leur satisfaction.

Ces résultats peuvent être résumés dans un tableau selon ces critères, qui mesurent l'efficacité de l'algorithme et la qualité du point final :

cas	i	t	s
A	22	0.0 (satisfaites)	4.18
B	20	+0.34	0.57
C	33	+0.002	3.12
D	15	+0.57	0.98
E	31	+0.32	1.09
F	17	+3.92	3.52
G	23	+2.09	5.12
H	12	+5.62	6.04

Le tableau montre le net avantage de l'optimisation à base de règles (cas A-D) sur la recherche aux monovariations (cas E-H). Les itérations sont moins nombreuses, le point final est plus proche de la satisfaction des contraintes, et son coût est moins élevé.

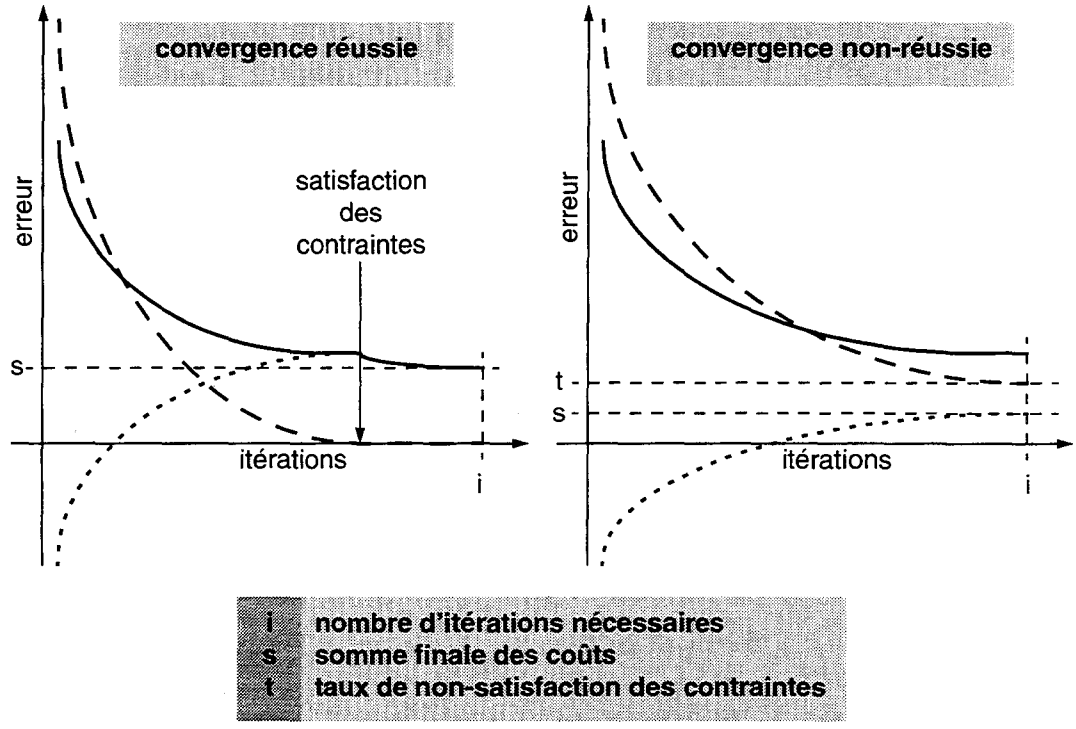


Figure 5.14 Convention utilisée pour l'évaluation des résultats de convergence

Entre les deux stratégies de gestion de coût, la méthode combinée nécessite moins d'itérations, mais le point final, bien qu'à coût moins élevé, ne satisfait pas à toutes les contraintes. En effet, la définition des fonctions de coût et leur valeurs de référence associées est bien plus critique que dans la méthode séquentielle. Par exemple, le coût de la surface peut présenter une erreur plus élevée que n'importe quelle contrainte, et l'heuristique visant à la minimiser sera choisie. Or, ce choix pourrait éloigner le vecteur solution d'un minimum faisable. La probabilité de convergence finale vers un point qui satisfait à toutes les contraintes et qui minimise tous les coûts est donc réduite. La configuration de l'optimiseur peut, bien sûr, être assouplie afin de permettre à ces algorithmes de converger.

La stratégie séquentielle, plus robuste, a nécessité un peu plus de dix itérations pour satisfaire à toutes les contraintes, et la minimisation des coûts a nécessité dix itérations de plus.

Les optimisations utilisant les sauts de règles ont convergé plus rapidement. Dans le mode lemming, l'algorithme a nécessité des critères de convergence moins contraignants, mais peut converger vers une solution ayant un coût moins élevé. Ce concept de saut entre règles pourrait donc être utilisé pour les recherches préliminaires, puis inhibé pour générer des solutions à bas coût.

5.4 Conclusion

Les algorithmes traditionnels d'optimisation numérique souffrent d'une inefficacité par rapport au nombre de déplacements réussis. Ceci est dû à l'approche purement quantitative prise par ces méthodes, ainsi qu'à la variation du jeu complet de paramètres dans toutes les directions. De nombreuses méthodes d'optimisation numérique ont été considérées, dans l'objectif de rajouter des heuristiques. Ceci réduit le nombre de variations possibles dans une situation donnée de conception, et rend ainsi l'algorithme plus efficace.

Les algorithmes qui utilisent les dérivées sont les plus efficaces, comme l'optimisation avec contraintes par la programmation quadratique récursive, et l'optimisation sans contrainte en utilisant les gradients. Par contre, l'évaluation des dérivées par réseaux linéaires nécessite une matrice linéarisée du circuit, ce qui n'est pas faisable pour les systèmes à temps discret tels que les circuits à courants commutés. L'autre méthode d'évaluation (par différences finies) est inefficace.

De ce fait, le choix se limite à l'optimisation avec contraintes par des fonctions de pénalité, utilisant un algorithme de recherche directe ou statistique. Parmi ces méthodes, celle de Hooke et Jeeves a été choisie pour sa simplicité, ainsi que la facilité avec laquelle un guidage par les connaissances peut s'y greffer.

Les modifications apportées à l'algorithme ont été décrites. Premièrement, une interface de contraintes a été introduite afin de diriger les données vers les parties concernées dans l'algorithme, dépendant du type et de l'état de toutes les contraintes. L'incorporation de la base de règles a ensuite été décrite. L'algorithme sélectionne la règle qui correspond à l'erreur de contrainte individuelle la plus élevée, et les paramètres de conception associés à cette règle sont modifiés dans la direction spécifiée. Ceci réduit effectivement le vecteur de variables de conception et augmente l'efficacité de l'algorithme. Le cas spécifique des heuristiques ambiguës a été décrit, où l'évaluation des deux possibilités de variation du paramètre de conception résout l'ambiguïté. Le concept du saut entre règles a été introduit, dans lequel une partie seulement d'une liste d'heuristiques est appliquée, s'il n'est pas plus avantageux d'appliquer la totalité. Finalement, l'aspect de gestion des coûts a été évoqué, car le moment où les coûts sont inclus dans la fonction objectif de conception a une influence majeure aur la direction de l'optimisation.

De multiples essais d'optimisation ont été effectués afin de valider les hypothèses présentées. Il a été démontré que l'optimisation à base de règles, qui utilise une gestion séquentielle des coûts et l'autorisation du saut entre règles, possède un net avantage sur ses concurrentes.

Référence

[BRA80] R.K. Brayton, R. Spence, "Sensitivity and Optimization", Elsevier Scientific, 1980

[KIR83] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, "Optimization by Simulated Annealing", *Science*, vol. 220, no. 4598, pp. 671-680, 13 May 1983

[MAS91] R.E. Massara, "Optimization Methods in Electronic Circuit Design", Longman Scientific and Technical, 1991

[REN95] J.-M. Renders, "Algorithmes génétiques et réseaux de neurones", Editions Hermès, 1995

6 ASIMOV : APPLICATIONS

Un outil de conception automatisée est validé lors de son utilisation. A cette fin, l'utilisateur requiert une interface conviviale, aussi bien pendant l'exécution de l'outil que pour la génération des topologies. Bien que cette dernière soit une tâche difficile, l'effort maximal doit être consacré à rendre l'outil le plus ouvert possible. L'exécution de l'outil peut être envisagée de deux façons : en détail sur une session de synthèse, permettant la compréhension des algorithmes utilisés ; et comparativement pour de nombreuses sessions de synthèse faisant appel à de nombreuses topologies différentes. Ces mesures de performances sont utiles, car elles éclairent l'utilisateur sur les capacités de l'outil à générer des cellules et sur les performances des cellules à courants commutés ainsi conçues.

Les principes abordés dans les chapitres précédents ont été utilisés pour créer un outil de CA des cellules à courants commutés [OCO97]. L'outil a été baptisé Asimov, qui signifie "A switched-current (SI) MOdule generation enVironment". Pour améliorer l'accessibilité de l'outil pour les utilisateurs, nous y avons incorporé une interface graphique et un module de génération de topologies. Asimov a ensuite été testé pour chaque topologie et pour un large éventail de spécifications.

En plus d'une utilisation dans une chaîne complète de conception, Asimov peut être utilisé pour explorer l'espace de conception. Autrement dit, l'utilisateur peut, grâce à Asimov, déterminer les limites d'une topologie par rapport à une condition de fonctionnement.

6.1 Interface graphique utilisateur

Un aspect important des outils de CA, et même d'une grande partie des logiciels scientifiques, est l'interface utilisateur. Nous les classons dans deux types :

- *l'interface alphanumérique*, qui est plus facile à coder, nécessite peu de mémoire lors de son utilisation et se porte aisément d'une plateforme à une autre. Cependant, de tels systèmes ne sont pas très appréciés par l'utilisateur, car il devient vite rébarbatif de communiquer les spécifications à travers des menus imbriqués. Il est également plus difficile de repérer les informations essentielles.
- *l'interface graphique*, qui est plus intuitive, est plus rapide à l'utilisation. L'utilisateur peut naviguer à travers les menus avec un nombre réduit de commandes de souris. Le problème principal de cette approche est que de nombreux systèmes incompatibles existent (Windows, Macintosh, Motif, Openwindows), bien que des normes internationales soient en passe de les faire converger vers une structure

commune. Ces interfaces ne sont donc pas nécessairement portables entre plateformes. Les autres problèmes associés aux interfaces graphiques sont les difficultés de codage (compensées par les outils d'aide à la conception), et la taille de la mémoire requise lors de l'utilisation.

Comme les outils de CA analogique sont difficiles à comprendre, nous avons choisi de faciliter l'utilisation d'Asimov en implémentant une interface utilisateur graphique. Nous avons décidé que les informations essentielles devaient être visibles dans la fenêtre principale, alors que les options de configuration seraient disponibles dans des fenêtres spécifiques, à la demande de l'utilisateur.

6.1.1 Fenêtre principale

Les informations essentielles à l'utilisateur sont présentées dans la fenêtre principale, comme illustré dans la Figure 6.1.

L'espace est constitué de trois zones :

- *la zone de configuration* dans la partie supérieure de la fenêtre,
- *la zone de conception* qui occupe la majeure partie de la fenêtre,
- *la zone de contrôle* dans la partie inférieure de la fenêtre.

La zone de configuration permet l'appel aux fenêtres de configuration de l'optimiseur, des modèles analytiques et de la sélection des topologies. La zone de contrôle permet d'exécuter le processus de CA (Run) ou d'arrêter l'application (Quit).

La zone de conception consiste en trois champs : les contraintes (supérieur à ou inférieur à), les conditions (égal) et les coûts (minimiser ou maximiser). Cependant, cette classification n'est pas rigide, et chaque spécification de performance peut se transformer en un autre type de spécification par l'utilisation du bouton de relation. Ces transformations ne sont possible que si elles ont un sens (par exemple, les spécifications des performances évaluées ne peuvent pas être des conditions ; la puissance ne peut pas être maximisée, etc.). Chaque spécification a une valeur cible, qui est fixe pour les conditions et définit une valeur raisonnable pour les coûts. La fonction objectif des coûts est évaluée comme :

$$W_i = \frac{1}{\theta_i}\left|\frac{P_{si} - P_{ri}}{P_{si}}\right| \tag{6.1}$$

Il est évident que la contribution d'un coût à la fonction objectif est déterminé par la valeur de priorité θ_i ainsi que par la valeur spécifiée P_{si}. Cette remarque est aussi valable pour les contraintes, alors que pour les conditions, qui ne sont pas évaluées, la valeur de la priorité n'a pas de signification. La dernière colonne présente les valeurs réalisées, qui sont mises à jour durant le processus de conception. Ceci permet à l'utilisateur de suivre la progression de l'optimisation.

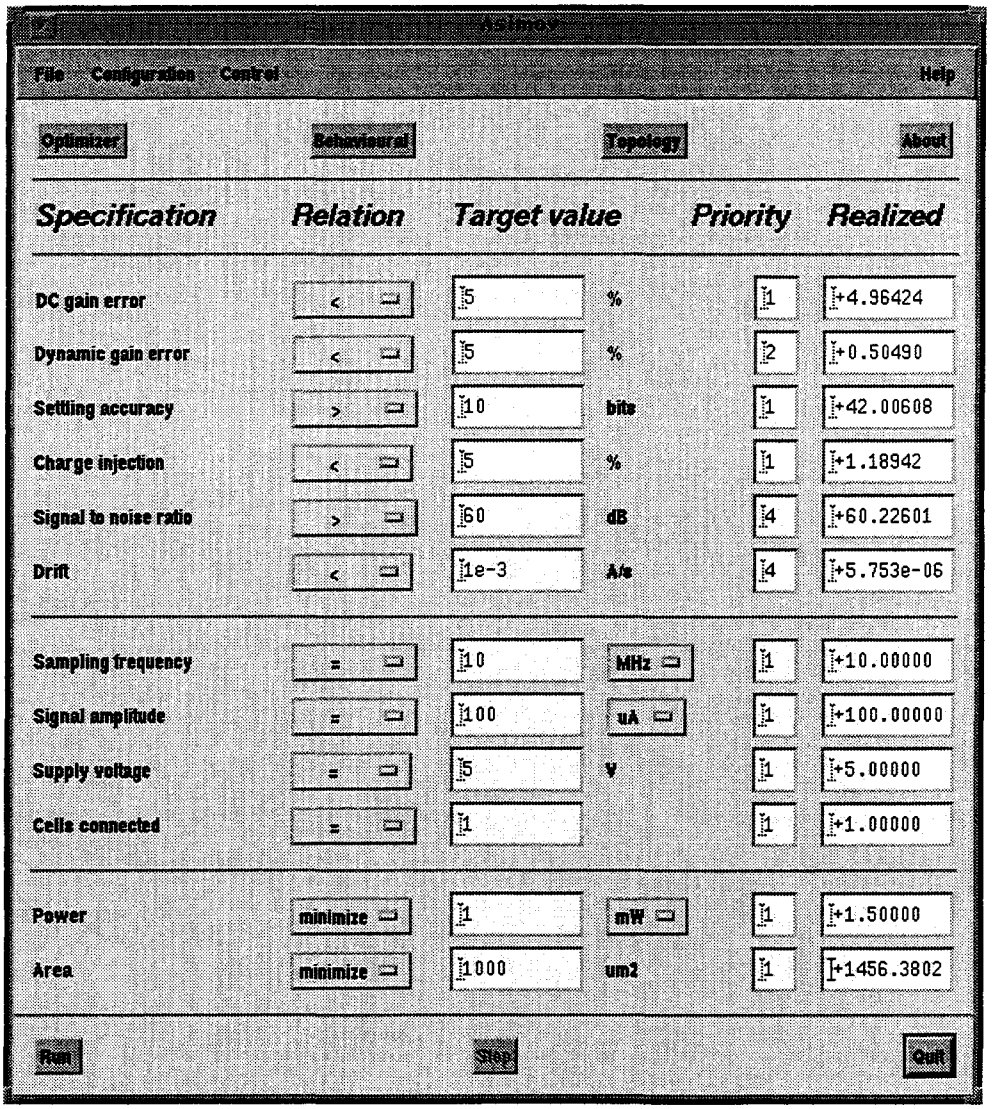


Figure 6.1 Asimov : fenêtre principale

6.1.2 Fenêtre de configuration de l'optimiseur

L'utilisateur expérimenté peut configurer l'algorithme d'optimisation afin de relâcher ou de resserrer les critères de convergence, ou de choisir entre les modes disponibles, comme cela a été expliqué dans le chapitre 5. La disposition de la fenêtre est illustrée par la Figure 6.2, et se divise en une zone de critères de convergence (où les valeurs sont entrées), et une zone de mode, où des interrupteurs activent ou inhibent les divers modes de calcul.

6.1.3 Fenêtre de configuration des modèles analytiques

Dans la fenêtre de configuration des modèles analytiques (Figure 6.3), l'utilisateur peut entrer des informations qui ne sont pas disponibles dans le modèle de technologie. Ce fichier est défini dans la partie inférieure de la fenêtre, mais ne comporte néanmoins pas

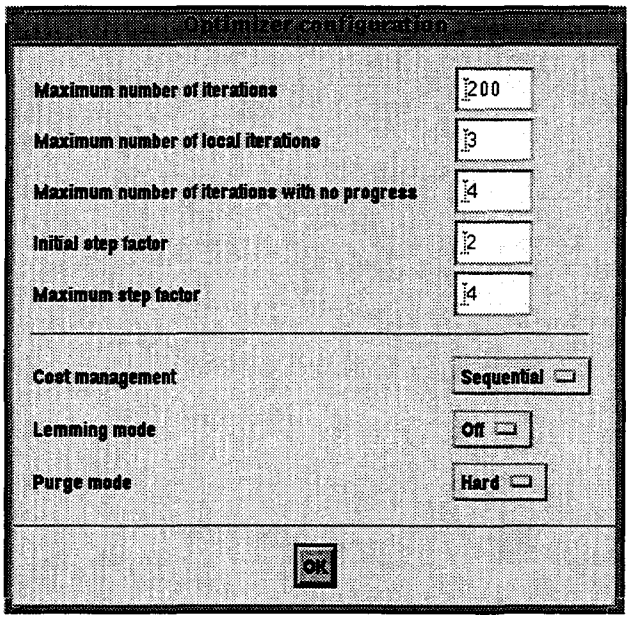


Figure 6.2 Asimov : fenêtre de configuration de l'optimiseur

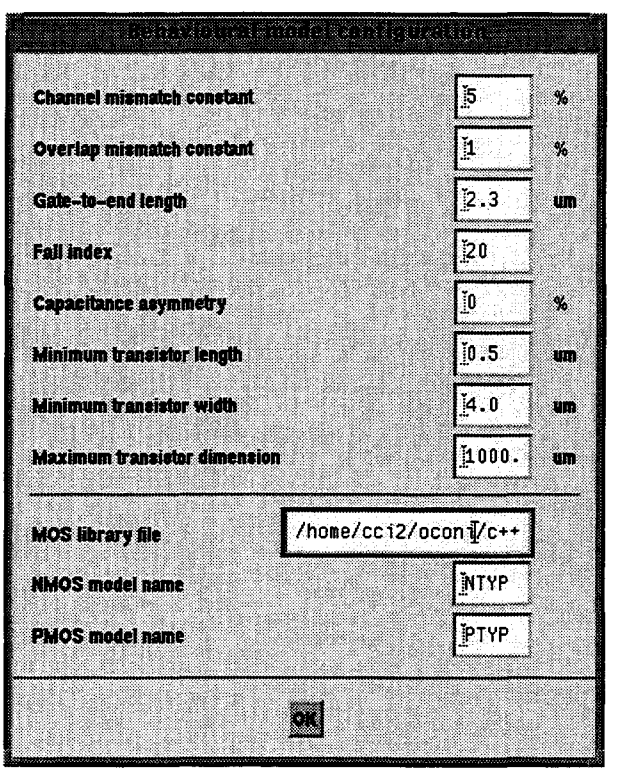


Figure 6.3 Asimov : fenêtre de configuration des modèles analytiques

toutes les informations nécessaires à l'évaluation précise des modèles analytiques. Par exemple, les informations technologiques telles que les facteurs de désappariement ou de l'asymétrie des capacités, utilisés dans le modèle de l'injection de charge, dépend du procédé technologique. Ces paramètres ne se trouvent pas dans le fichier du modèle SPICE, qui comporte les caractéristiques nominales du procédé utilisé. L'autre paramètre, l'indice de descente ("fall index") est dépendant du système. Il a été défini dans la section 4.2 comme étant la période d'horloge divisée par le temps de commutation.

6.1.4 Fenêtre de sélection de topologies

Enfin, l'utilisateur peut choisir entre une synthèse utilisant toutes les topologies, et une synthèse comportant une liste réduite de topologies. Dans le premier cas, c'est le sélectionneur qui prend les décisions topologiques, et dans les deuxième, l'utilisateur expérimenté élimine les topologies qui sont peu adaptées aux spécifications. Par exemple, une architecture de système qui met la priorité sur le fonctionnement à basse tension peut obliger le concepteur à enlever certaines variantes structurelles de la liste (Figure 6.4).

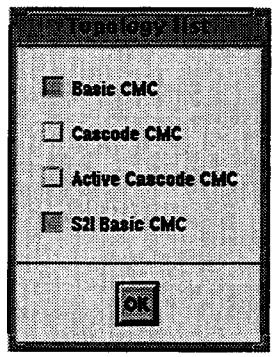


Figure 6.4 Asimov : fenêtre de sélection des topologies

6.2 Génération des topologies

Parmi les spécifications de l'outil de conception énoncées dans la section 1.2.4, l'une d'entre elles est relative à l'ouverture de l'outil, c'est-à-dire que l'utilisateur doit avoir la possibilité de rajouter de nouvelles topologies ou de modifier des topologies existantes. La structure d'Asimov requiert la compilation de toutes les topologies en un seul fichier exécutable. Les topologies doivent donc être codées dans le langage approprié, qui est le C++. Cependant, la structure interne d'Asimov, décrite dans l'annexe C, repose fortement sur les listes [GNU95] pour l'ordonnement des paramètres et pour leur passage entre les fonctions. L'inconvénient des listes est qu'elles rendent le code répétitif et lourd. Il n'est donc pas réaliste de s'attendre à ce que l'utilisateur, qui ne possède pas nécessairement l'expérience de programmation, puisse générer le fichier final de topologie.

Pour cette raison, nous avons mis en place un interpréteur de topologie, qui transcrit un pseudo-code en langage C++, comme le montre la Figure 6.5. La volonté d'aider les utilisateurs non-programmeurs a pour limite le fait qu'un langage trop simplifié serait trop restrictif pour la définition correcte du comportement de la cellule. De ce fait, une grande partie du code de calcul reste en C++, mais les parties répétitives de programmation sont générées automatiquement. Cette solution de compromis n'est probablement pas la meilleure, et nous ne pouvons que constater les difficultés dans le domaine de l'interface entre l'utilisateur et formalisation des connaissances.

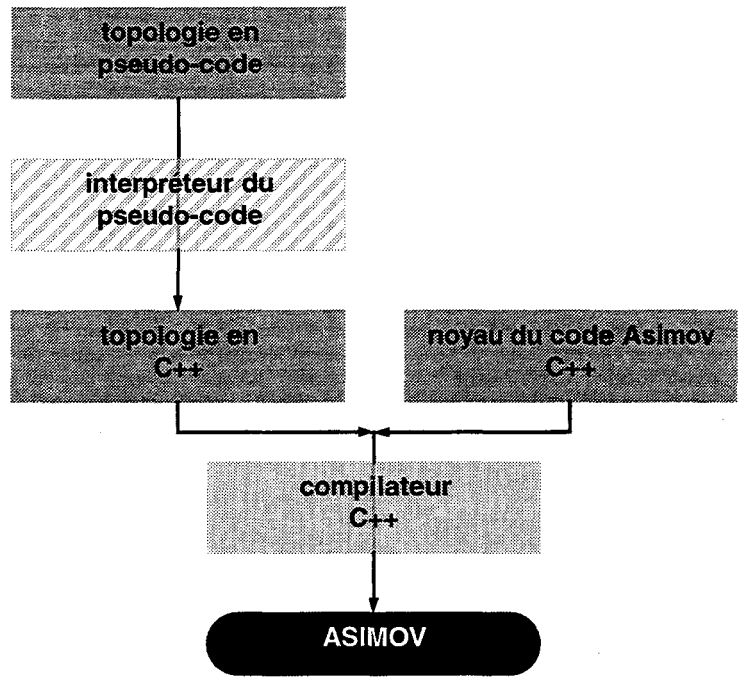


Figure 6.5 Génération d'une topologie par l'utilisation de sa description en pseudo-code

6.2.1 Préliminaires

Les préliminaires consistent en l'attribution d'un identificateur unique à la topologie, puis d'un nom pour la cellule et sa catégorie, utilisé lors de la sortie des informations vers l'utilisateur (Listing 6.1).

```
/*****************************************************************/
.prefix BCMC          <-- identificateur unique
.name Basic
.category Current_Memory_Cell   <-- information de sortie pour l'utilisateur
/*****************************************************************/
```

Listing 6.1 Préliminaires

6.2.2 Déclaration de dimension

Les dimensions sont déclarées dans deux listes qui correspondent aux dimensions formelles et effectives (Listing 6.2) (voir section 2.12).

```
/*****************************************************************/
.formal     (unités)   (min)     (max)      (type de variation)
vgstm       V          0         4          2
lm          M          L_MIN     L_MIN*5    1
.end
/*****************************************************************/

/*****************************************************************/
.actual     (unités)   (min)     (max)
mmw         M          W_MIN     REAL_MAX
mml         M          L_MIN     L_MIN*5
.end
/*****************************************************************/
```

Listing 6.2 Déclarations de dimensions

A chaque dimension, un nom unique est attribué, suivi par ses unités, par les valeurs minimale et maximale et, dans le cas des dimensions formelles, par son type de variation, où :

- 0 indique une variation linéaire des entiers par incrément ou décrément

$$x \rightarrow x \pm 1 \tag{6.2}$$

  qui est utilisée, par exemple, dans la dimension du nombre de cellules connectées.

- 1 indique une variation logarithmique simple

$$x \rightarrow x\left(1 \pm \frac{1}{2^n}\right) \tag{6.3}$$

  qui est utilisée pour les longueurs L des transistors, la capacité de grille C_g, la tension d'alimentation V_{dd}, etc.

- 2 indique une variation de type racine de logarithme

$$x \rightarrow \sqrt{1 \pm \frac{1}{2^n}} \tag{6.4}$$

  qui est utilisée pour les tensions de grille utiles (GVO) v_{gst}.

Le fait que les heuristiques utilisent des dimensions formelles comme v_{gst} (tension de grille utile d'un transistor) et L (longueur d'un transistor) implique que la façon dont la variable est modifiée est importante. Une variation logarithmique simple est bien adaptée aux dimensions physiques telles que L, mais ne l'est pas aux variables de nature quadratique telles que v_{gst}. En doublant v_{gst}, la valeur du courant est modifiée de façon plus importante que la division de L par deux. Il en résulte qu'il est souhaitable

d'égaliser de façon approximative les variations sur les paramètres intrinsèques aux dispositifs par rapport aux dimensions formelles, pour un facteur de pas donné. C'est le cas quand une variation de type racine de logarithme est utilisée pour les variables de nature quadratique.

Par exemple, l'expression du premier ordre pour le courant de drain d'un transistor en saturation donne :

$$I = \frac{K}{2}\frac{W}{L}v_{gst}^2 \tag{6.5}$$

Si nous faisons varier la longueur L de façon logarithmique simple, nous avons que :

$$L \rightarrow L + \frac{L}{2} = \frac{3L}{2} \tag{6.6}$$

et à courant égal,

$$W \rightarrow \frac{3W}{2} \tag{6.7}$$

Pour atteindre une variation de la largeur (à courant égal) du même ordre de grandeur, nous faisons varier la tension de grille utile v_{gst} de façon racine de logarithme :

$$v_{gst} \rightarrow v_{gst}\sqrt{1 - \frac{1}{2}} = \frac{v_{gst}}{\sqrt{2}} \tag{6.8}$$

$$W \rightarrow 2W \tag{6.9}$$

Si nous appliquons une variation de façon logarithmique simple à v_{gst}, la valeur de W est très différente :

$$W \rightarrow 4W \tag{6.10}$$

En résumé, la différence entre les types de variation des dimensions formelles sert à égaliser de façon approximative leurs effets sur les paramètres intrinsèques du transistor.

6.2.3 Relations entre dimensions formelles et effectives

La relation entre une dimension formelle et une dimension effective peut être définie explicitement par une commande *.relation* (Listing 6.3), où "lm" est transcrit directement en "mml" au δL (FINELINE) près. Si la transcription est plus complexe, comme dans le cas de la convergence itérative vers une valeur de W pour des valeurs données de v_{gst}, L et I, elle peut être codée dans la fonction *.update*, qui est invoquée au début de chaque itération.

```
/*****************************************************************/
.relation mmi
roundoff ( lm, FINELINE )
.end
/*****************************************************************/
```

Listing 6.3 Relation entre dimensions formelles et effectives

6.2.4 Code du modèle

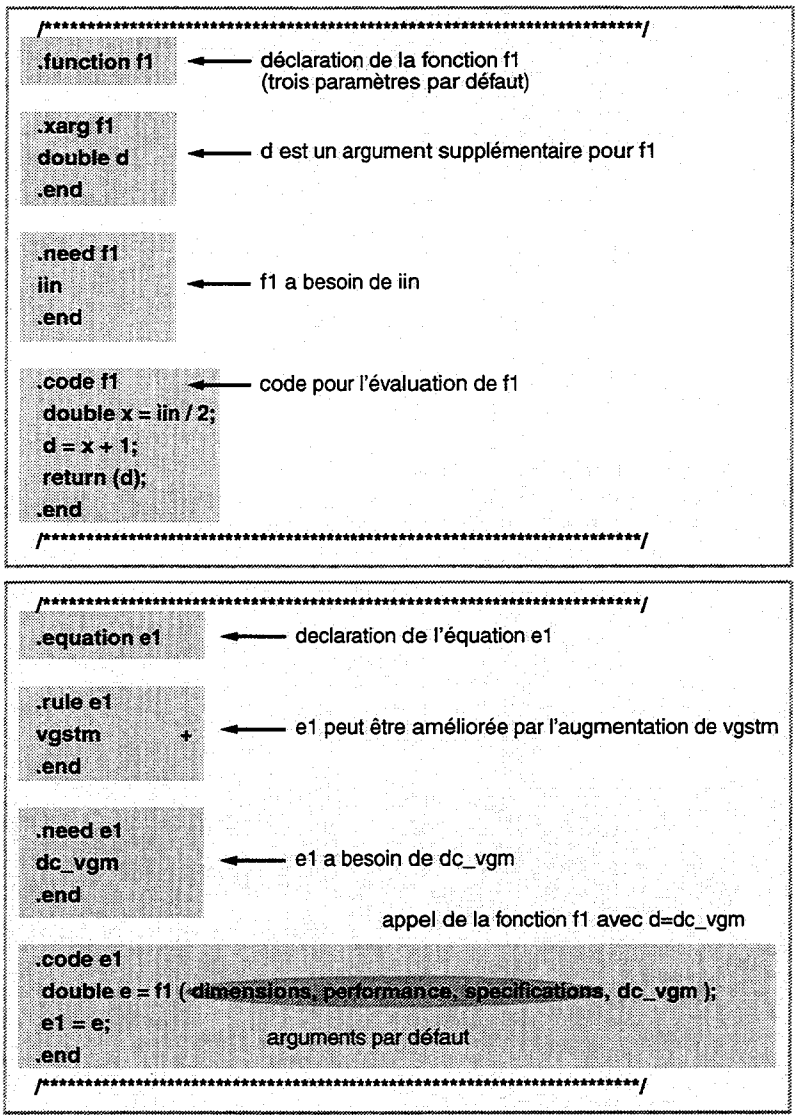
Trois types de fonction constituent le codage du modèle. La première est la fonction *.setup*, qui est unique et constitue la génération du point de départ pour la topologie (c'est-à-dire les modèles du premier ordre développés dans les deuxième et troisième chapitres). Le deuxième type, défini par le mot clé *.function*, constitue une fonction d'aide qui est utilisée par le troisième type, les fonctions *.equation*, qui représentent l'évaluation des performances de la cellule.

La différence entre les fonctions et les équations est que les fonctions sont libres d'inclure des arguments supplémentaires et peuvent retourner leurs résultats par des pointeurs. Les équations, par contre, retournent leur résultat directement et doivent être conformes à un format défini par le pointeur vers les performances de la topologie. Ce format comporte trois arguments par défaut :

- la liste des dimensions,
- la liste des performances,
- la liste des spécifications.

De plus, les règles sont attachées aux équations, et non pas aux fonctions.

Les équations et les fonctions déclarent toutes deux les variables dont elles ont besoin, telles que le courant d'entrée, la valeur de la tension de grille au repos, etc., dans une boite *.need*. Quand le code est traduit en C++, le type de chaque variable (contrainte, condition, coût ou performance), est déterminé automatiquement.



Listing 6.4 Exemple de fonctions et équations

6.3 Exemple d'une session Asimov

L'enchaînement global de conception utilisé par Asimov est illustré par la description d'un exemple pratique. Une quantité considérable d'informations concernant la progression du processus d'optimisation est conservée dans le fichier d'historique ("log file"). Celui-ci peut être consulté par un utilisateur expérimenté souhaitant déterminer comment s'est déroulée la conception.

6.3.1 Formalisation du problème d'optimisation

Cet exemple utilise le problème d'optimisation qui a été décrit dans la section 5.3, c'est-à-dire pour la topologie de la cellule de base,

contraintes	coûts	conditions
erreur statique < 5%	minimiser puissance (1mW)	tension d'alimentation = 5V
erreur capacitive < 5%	minimiser surface (100μm^2)	échantillonnage = 20MHz
précision > 10 bits		courant d'entrée = 100μA
injection de charge < 5%		cellules connectées = 1
SNR > 60dB		
dérive < 1mA/s		

Les modes utilisés étaient identiques à ceux utilisés dans le cas du problème d'optimisation mentionné ci-dessus: l'optimisation à base de règles, le mode lemming désactivé (les sauts entre règles autorisés), gestion de coûts séquentielle. La configuration de l'optimiseur a été légèrement modifiée afin de générer un exemple plus démonstratif (l_{max}=3, p_{max}=5, f_{max}=4).

Le fichier d'historique commence toujours avec un résumé de l'environnement dans lequel fonctionne Asimov (Listing 6.5), comprenant la date et l'heure du commencement de la session, ainsi que les modèles dispositifs employés.

```
******************************************
          A S I M O V
          ───────
     A switched current (SI)
  MOdule generation enVironment
          Version 0.20
******************************************
Session started Mon Aug 4 17:36:13 1997
Loading models in file /home/cci2/oconi/c++/asimov/st05.lib
```

Listing 6.5 Début du fichier d'historique

Les spécifications externes sont ensuite résumés (Listing 6.6), en indiquant à quelle liste (de contraintes, de coûts ou de conditions) chacune appartient. Les contraintes sont résumées sous la forme

nom (+ offset) relation (< / >) valeur spécifiée : valeur de priorité

Les coûts indiquent leurs directions et valeurs de "référence", tandis que les conditions indiquent leurs valeurs fixes.

```
constraint_list:
DC gain error (+ 0) < 5 : priority = 1
Capacitive error (+ 0) < 5 : priority = 2
Settling accuracy (+ 0) > 10 : priority = 1
Charge injection (+ 0) < 5 : priority = 1
Signal to noise ratio (+ 0) > 60 : priority = 4
Drift (+ 0) < 0.001 : priority = 4

cost_list:
minimize Power ( good value is 0.001 )
minimize Area ( good value is 1e-10 )

condition_list:
Sampling frequency = 2e+07 Unit_none
Input current = 0.0001 Unit_none
Supply voltage = 5 Unit_none
Number of cells connected = 1 Unit_none
```

Listing 6.6 Résumé des spécifications

Chaque topologie dans la bibliothèque des cellules est ensuite dimensionnée de façon préliminaire, et selon les spécifications. Les topologies qui réussissent à générer un point de départ sont incluses dans la liste des topologies actives, c'est-à-dire dans la liste des topologies qui seront transférées au bloc de dimensionnement. Dans cet exemple, nous avons désactivé toutes les topologies à l'exception de la cellule de base.

Les spécifications externes sont alors transformées en spécifications internes sans modifications (car a priori, toutes les spécifications peuvent être satisfaites). Le résumé des spécifications internes, présent dans le fichier d'historique, est identique à celui des spécifications externes.

Un résumé de la topologie choisie pour optimisation suit. Les dimensions formelles sont énoncées en premier (Listing 6.7), sous la forme

nom = valeur unités (< inférieur, supérieur>) état de contrainte = 1 / 0

Les dimensions effectives les suivent (Listing 6.8), dans le même format.

Les règles pour le dimensionnement de la cellule sont ensuite détaillées (Listing 6.9), rangées par critère de performance.

```
Selected Basic Current Memory Cell for optimization
design = Basic Current Memory Cell

formal dimension list:
Memory transistor GVO = 0.589035 Unit_V ( < 0, 4 > ) constraint state = 1
Memory transistor L = 5e-07 Unit_M ( < 5e-07, 2.5e-06 > ) constraint state = 1
Source transistor GVO = -0.25 Unit_V ( < -4, 0 > ) constraint state = 1
Source transistor L = 5e-07 Unit_M ( < 5e-07, 2.5e-06 > ) constraint state = 1
Switch L = 5e-07 Unit_M ( < 5e-07, 2.5e-06 > ) constraint state = 1
Bias current = 0.0002 Unit_A ( < 0, 0.001 > ) constraint state = 1
Gate capacitance = 0 Unit_F ( < 0, 1e-11 > ) constraint state = 1
Input current = 0.0001 Unit_A ( < 0, 0.001 > ) constraint state = 1
Supply voltage = 5 Unit_V ( < 2.5, 5 > ) constraint state = 1
Sampling frequency = 2e+07 Unit_Hz ( < 1000, 1e+12 > ) constraint state = 1
Number of cells connected = 1 Unit_none ( < 1, 10 > ) constraint state = 1
```

Listing 6.7 Résumé des dimensions formelles

```
actual dimension list:
Memory transistor width = 6.1e-06 Unit_M ( < 4e-06, 0.001 > ) constraint state = 1
Memory transistor length = 5e-07 Unit_M ( < 5e-07, 0.001 > ) constraint state = 1
Source transistor width = 8.64e-05 Unit_M ( < 4e-06, 0.001 > ) constraint state = 1
Source transistor length = 5e-07 Unit_M ( < 5e-07, 0.001 > ) constraint state = 1
Switch width = 4e-06 Unit_M ( < 4e-06, 0.001 > ) constraint state = 1
Switch length = 5e-07 Unit_M ( < 5e-07, 0.001 > ) constraint state = 1
Gate capacitor = 0 Unit_F ( < 0, 1e-11 > ) constraint state = 1
Actual_ib = 0.0002 Unit_A ( < 0, 0.001 > ) constraint state = 1
Actual_bvb = 4.22693 Unit_V ( < 0, 5 > ) constraint state = 1
Input current = 0.0001 Unit_A ( < 0, 0.001 > ) constraint state = 1
Supply voltage = 5 Unit_V ( < 2.5, 5 > ) constraint state = 1
Power = 0.001 Unit_W ( < 0, 1 > ) constraint state = 1
Sampling frequency = 2e+07 Unit_Hz ( < 1000, 1e+12 > ) constraint state = 1
Number of cells connected = 1 Unit_none ( < 1, 10 > ) constraint state = 1
```

Listing 6.8 Résumé des dimensions effectives

6.3.2 Satisfaction des contraintes

Une fois ce résumé général terminé, le processus de conception commence. Chaque itération suit la même procédure, qui est maintenant expliquée. L'analyseur de faisabilité vérifie les tailles de chaque dimension, et le fait qu'aucune contrainte n'est violée. Il applique les fonctions d'analyse statique afin de s'assurer de la fonctionnalité du circuit pour tout courant d'entrée inclus dans la dynamique spécifiée. Si ces analyses ne révèlent pas d'erreur, les équations analytiques sont invoquées. Un résumé des performances de la cellule est présenté. Chaque critère est écrit sous la forme

nom	(+ offset)	<	valeur spécifiée :	valeur de priorité
valeur réalisée		erreur	état de contrainte (0/1)	

Comme le mode de gestion des coûts est séquentiel, seules les contraintes apparaissent pendant la première partie du processus de conception (Listing 6.10). Les critères de performance sont alors triés par ordre décroissant de valeur individuelle des fonctions objectif, c'est-à-dire des fonctions d'erreur. Le listing montre que l'erreur la plus élevé est pour l'injection de charge. La règle associée est extraite afin de créer une liste d'heuristiques qui visent à réduire cette erreur (Listing 6.11). La liste des heuristiques

```
rule list:
rules for Nominal dc conditions
increase Bias current
decrease Memory transistor GVO
decrease Source transistor GVO

rules for Maximum dc conditions
decrease Memory transistor GVO

rules for Minimum dc conditions
decrease Memory transistor GVO

rules for DC gain error
increase Memory transistor L
increase Source transistor L
decrease Memory transistor GVO
increase Source transistor GVO

rules for Capacitive error
increase Gate capacitance
increase Memory transistor GVO

rules for Settling accuracy
unknown direction Memory transistor GVO
unknown direction Bias current
decrease Gate capacitance

rules for Charge injection
decrease Switch L
increase Gate capacitance

rules for Signal to noise ratio
increase Input current
increase Gate capacitance
increase Memory transistor L

rules for Drift
increase Gate capacitance
decrease Switch L

rules for Power
decrease Supply voltage
decrease Bias current

rules for Area
decrease Bias current
decrease Gate capacitance
increase Memory transistor GVO
increase Source transistor GVO
decrease Source transistor L
decrease Switch L
```

Listing 6.9 Résumé des règles d'heuristiques

```
DC gain error (+ 0) < 5 : priority = 1
value: 16.3442 : error: 2.26883 : met: 0
Capacitive error (+ 0) < 5 : priority = 2
value: 17.3259 : error: 1.23259 : met: 0
Settling accuracy (+ 0) > 10 : priority = 1
value: 94.0984 : error: 0 : met: 1
Charge injection (+ 0) < 5 : priority = 1
value: 31.3474 : error: 5.26948 : met: 0
Signal to noise ratio (+ 0) > 60 : priority = 4
value: 43.1369 : error: 0.070263 : met: 0
Drift (+ 0) < 0.001 : priority = 4
value: 0.0182707 : error: 4.31768 : met: 0
```

Listing 6.10 Performances de la cellule

est épurée afin d'enlever les heuristiques qui sont en conflit avec celles d'autres contraintes non-satisfaites, ainsi que celles qui affectent certaines dimensions. Ces dimensions sont celles qui ne peuvent être modifiées dans la direction indiquée sans induire une faute physique. Dans cet exemple, la règle extraite est constituée de deux heuristiques. La première, "diminuer la L de l'interrupteur", ne s'applique pas car cette dimension est égale à sa limite inférieure, 0.5µm. La deuxième, "augmenter la capacité de mémorisation", peut l'être, car cette dimension est loin de sa limite supérieure, et le seul conflit de contrainte qui peut exister se trouve dans la règle appartenant à la précision d'établissement. Puisque la contrainte relative à cette performance est satisfaite, elle n'empêche pas l'application de l'heuristique choisie. Dans la situation où aucune heuristique ne peut être appliquée pour réduire l'erreur maximale, l'erreur suivante est choisie et la règle correspondante extraite. Cette opération se répète jusqu'à ce qu'une variable de conception sans restriction soit trouvée.

```
Sorted list:
Charge injection (+ 0) < 5 : priority = 1
Drift (+ 0) < 0.001 : priority = 4
DC gain error (+ 0) < 5 : priority = 1
Capacitive error (+ 0) < 5 : priority = 2
Signal to noise ratio (+ 0) > 60 : priority = 4
Settling accuracy (+ 0) > 10 : priority = 1
Sorted list:
minimize Power ( good value is 0.001 )
minimize Area ( good value is 1e-09 )
Removing at limit heuristic decrease Switch L
Purged heuristic list:
increase Gate capacitance
```

Listing 6.11 Génération d'une liste d'heuristiques applicables

Ayant généré la liste des heuristiques applicables, la première heuristique est appliquée. Comme la dimension formelle de la capacité de mémorisation a une valeur nulle, sa variation se base sur le pas minimal (calculé à partir des valeurs des limites supérieures et inférieures), plutôt que sur sa valeur courante, comme est habituellement le cas. Juste avant que l'optimiseur rende le contrôle à la boucle d'analyse, un court résumé des statistiques d'optimisation est donné (Listing 6.12).

```
heuristic: increase Gate capacitance
dimension being altered: Gate capacitance = 0 Unit_F ( < 0, 1e-11 > ) constraint state = 1
dim_value: 5e-14 : prev_dim_value: 0

iterations: 1 : local_iterations: 1 : last_success: 0
largest_error: 5.26948 : sum_of_errors: 13.1588 : lowest_sum_of_errors: 13.1588 : step_factor: 2
```

Listing 6.12 Application de l'heuristique choisie et fin de l'intervention de l'optimiseur

L'évolution des fonctions objectif de conception individuelles pendant la première phase (contraintes seulement) du processus d'optimisation est illustrée dans la Figure 6.6. Les zones grises indiquent les itérations pendant lesquelles l'erreur concernée est la plus importante de toutes les erreurs. Dans ce cas, la règle qui s'applique vise à améliorer cette erreur individuelle.

Le déroulement du processus de conception peut être extrait du fichier d'historique. Il est résumé dans le Tableau 6.1, ou chaque ligne montre :

- le nombre d'itérations (it),
- l'erreur de performance individuelle maximale à cet instant,
- si la règle vient d'être extraite, et dans l'affirmative, si ce changement de règle constitue un saut entre règles,
- pour une nouvelle règle, de quelle façon la liste des heuristiques a été épurée,
- l'heuristique choisie,
- la variation du paramètre de conception réellement appliquée.

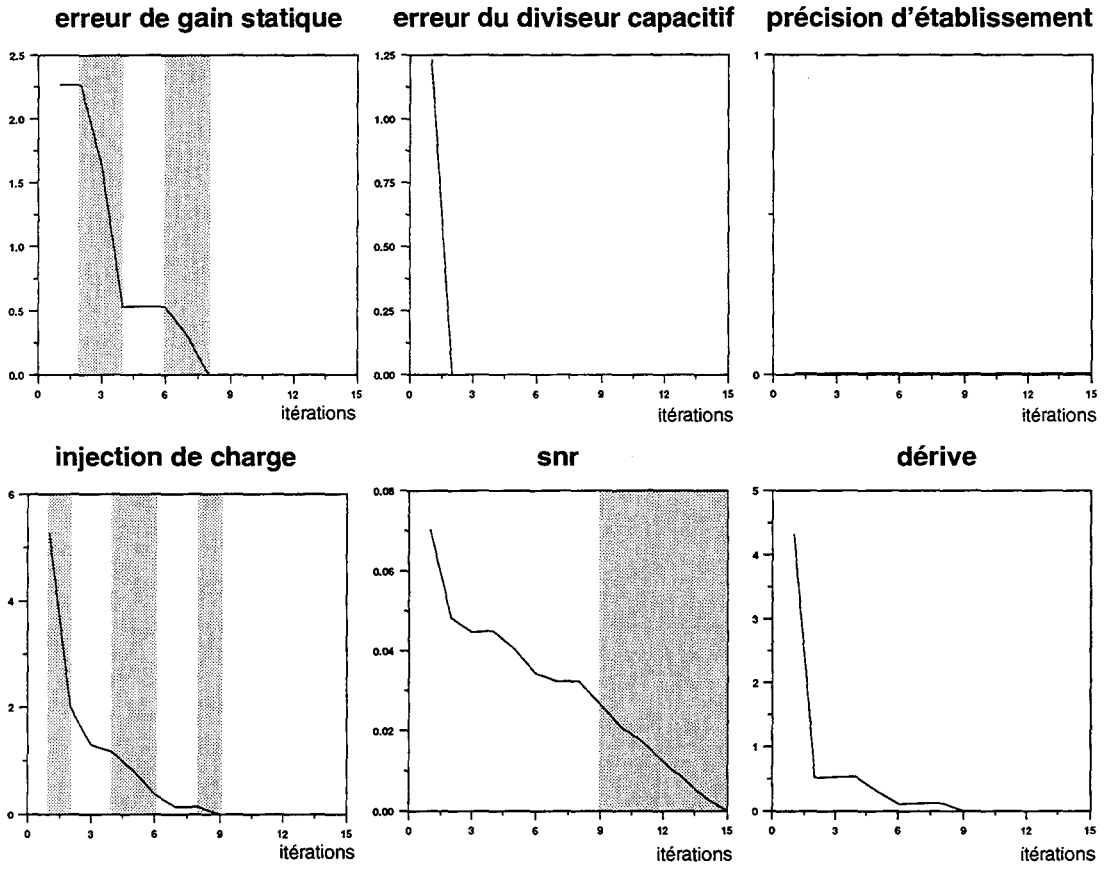


Figure 6.6 Evolution des fonctions objectif de contraintes individuelles pendant optimisation

Le processus peut se passer d'explication, mis à part les deux évènements suivants :

A la neuvième itération, l'heuristique "augmenter l'amplitude du courant d'entrée" est enlevée de la liste des heuristiques. Ceci est dû au fait que l'amplitude du courant d'entrée est une condition, imposée par l'environnement externe. Lors d'une optimisation où cette quantité constitue un degré de liberté, cette heuristique serait en effet utilisée.

A l'itération 13, la heuristique "augmenter la longueur du transistor mémoire" est enlevée de la liste des heuristiques, bien qu'elle ait été utilisé avec succès jusqu'à ce point. La raison pour cette action est que cette dimension a atteint sa limite supérieure (2.5µm), imposée par le modèle analytique.

it	erreur maximale	nouvelle règle	action d'épuration	heuristique choisie	variation de paramètre
1	injection de charge	oui	M_{sw} L ↘: limite	C_m ↗	0→50f
2	erreur de gain statique	oui	aucune	M_m L ↗	0.5µ→0.8µ
3		non	-	M_b L ↗	0.5µ→0.8µ
4	injection de charge	oui/saut	M_{sw} L ↘: limite	C_m ↗	50f→75f
5		non	M_{sw} L ↘: limite	C_m ↗	75f→113f
6	erreur de gain statique	oui	aucune	M_m L ↗	0.8µ→1.1µ
7		non	-	M_b L ↗	0.8µ→1.1µ
8	injection de charge	oui/saut	M_{sw} L ↘: limite	C_m ↗	113f→169f
9	snr	oui	i_0 ↗: condition	C_m ↗	169f→253f
10		non	-	M_m L ↗	1.1µ→1.7µ
11		oui	i_0 ↗: condition	C_m ↗	253f→380f
12		non	-	M_m L ↗	1.7µ→2.5µ
13		oui	i_0 ↗: condition M_m L ↗: limite	C_m ↗	380f→570f
14		oui	i_0 ↗: condition M_m L ↗: limite	C_m ↗	570f→854f

Tableau 6.1 Résumé du processus de conception lors de la phase de satisfaction des contraintes

6.3.3 Minimisation des coûts

Une fois les contraintes satisfaites, les coûts sont inclus dans la fonction objectif de conception. Du fait que les contraintes ont une contribution nulle pour la fonction objectif de conception (du moins tant qu'elles ne sont pas violées par la minimisation des coûts), cette dernière comprend uniquement les coûts.

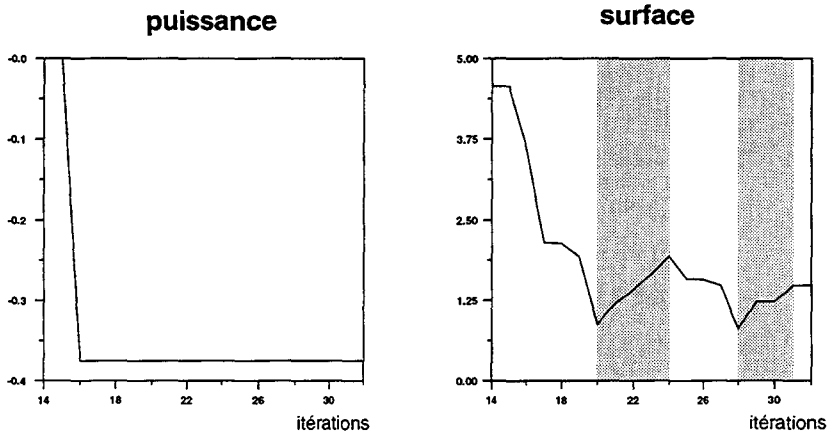


Figure 6.7 Evolution des fonctions objectifs de coûts individuelles pendant l'optimisation

L'évolution des coûts individuels (puissance et surface) est montrée dans la Figure 6.7. The zones grises correspondent ici à une augmentation de la fonction de coût, suite à l'application d'une variation de paramètre qui engendre la violation d'une contrainte. Dans cet exemple, la longueur du transistor source est diminuée pour réduire la surface. Cependant, la performance de l'erreur de gain statique ne satisfait plus aux spécifications. L'augmentation du coût constitue donc la recherche menée par l'algorithme en vue de définir la longueur minimale pouvant satisfaire la contrainte.

Le processus complet de conception pendant la phase de minimisation des coûts a été résumé dans le Tableau 6.2. Il est important de noter que la numérotation des itérations continue à partir de celle du tableau précédent.

it	erreur maximale	nouvelle règle	action d'épuration	heuristique choisie	variation de paramètre
15	surface	oui	M_{sw} L ↘: limite	I_b ↘	200µ→125µ*
16		non	-	C_m ↘	854f→427f
17		non	-	M_m v_{gst} ↗	0.59→0.61
18		non	-	M_b v_{gst} ↘	-0.25→-0.27
19		non	-	M_b L ↘	1.1µ→0.5µ
20		non	-	M_b L ↘	1.1µ→0.8µ
21		non	-	M_b L ↘	1.1µ→0.9µ
22		non	-	M_b L ↘	1.1µ→1.0µ
23		oui	M_{sw} L ↘: limite	I_b ↘	125µ→125µ*
24		non	-	C_m ↘	427f→320f
25		non	-	M_m v_{gst} ↗	0.61→0.62
26		non	-	M_b v_{gst} ↘	-0.27→-0.28
27		non	-	M_b L ↘	1.1µ→0.85µ
28		non	-	M_b L ↘	1.1µ→0.98µ
29		non	-	M_b L ↘	1.1µ→1.03µ
30		non	-	M_b L ↘	1.1µ→1.05µ
31		oui	M_{sw} L ↘: limite	I_b ↘	125µ→125µ*

Tableau 6.2 Résumé du processus de conception lors de la phase de minimisation des coûts

La réduction du courant de polarisation est quelque peu singulière, représentée par une "*" dans le tableau. A l'itération 15, le courant devrait se voir attribuer une réduction de la moitié de sa valeur, c'est-à-dire 100µA. Toutefois, ce n'est pas le cas car le courant de polarisation violerait alors l'inégalité (6.11)

$$i_0 < \eta I_b \tag{6.11}$$

qui a été donné dans le deuxième chapitre. La quantité i_0 représente l'amplitude du

courant d'entrée, et η le facteur de la dynamique d'entrée. Nous posons une valeur maximale de 0.8 pour η, ce qui impose une limite inférieure à I_b. Cependant, nous ne pouvons pas l'implémenter comme une limite inférieure de la contrainte dans le pseudo-code pour le modèle analytique, car la valeur de i_0 n'est pas connue avant le début de la conception. Par conséquent, un piège à erreur doit être mis en place afin d'empêcher l'inégalité d'être violée.

Convergence est obtenue quand le facteur de pas dépasse la valeur maximale définie, f_{max} (égale à 4 dans cet exemple). Ici, le facteur de pas commence avec une valeur de 2 et double quand un nombre d'itérations égal à p_{max} (5 dans ce cas) est atteint sans réduction de la fonction objectif de conception. Ceci peut se produire quand la réduction des coûts est obtenue au prix de la violation des contraintes.

C'est le cas pour la réduction de la longueur du transistor source. Elle est initialement réduite d'une quantité normale. Puisque ceci engendre la violation d'une contrainte, elle reprend son ancienne valeur, réduite d'une quantité moindre. Ce processus peut se répéter jusqu'à l_{max} fois (où l_{max} est le nombre maximal d'itérations locales, c'est-à-dire 4). Si la contrainte reste non-satisfaite, l'optimiseur remet la dimension à sa valeur initiale, et continue avec l'heuristique suivante, en remettant le compteur d'itérations locales à zero.

Le problème avec ce système est que les heuristiques prometteuses qui sont utilisées à la suite d'heuristiques problématiques sont pénalisées, car elles peuvent être appliquées avec un facteur de pas augmenté, ou pas du tout si la convergence est obtenue. Avec du recul, un meilleur système aurait dû être de nature plus cyclique, appliquant toutes les heuristiques dans une règle et ne déterminant les caractéristiques de convergence que par la suite.

6.3.4 La validation des performances par la simulation numérique

Après la convergence vers un dimensionnement de cellule qui satisfait les contraintes et minimise les coûts, cette cellule doit être validée par la simulation numérique pour évaluer les imprécisions dues aux équations analytiques. Cette phase finale applique les dimensions de la cellule à l'environnement de simulation numérique développé pour la caractérisation des cellules.

Les résultats sont ensuite rassemblés et intégrés dans un résumé global des performances de la cellule. Ici, ce sont les contraintes qui sont concernées (Listing 6.13).

```
edc = 3.96601
ecap = 1.16798
setacc = 23.7229
cft = 3.1
snr = 63.7469
drift = 8.68e-05
DC gain error (-0.857881) < 5 : priority = 1
value: 4.8239 : error: 0 : met: 1
Capacitive error (-0.554546) < 5 : priority = 2
value: 1.72253 : error: 0 : met: 1
Settling accuracy (-4.96693) > 10 : priority = 1
value: 28.6898 : error: 0 : met: 1
Charge injection (+2.01823) < 5 : priority = 1
value: 1.08177 : error: 0 : met: 1
Signal to noise ratio (+2.90523) > 60 : priority = 4
value: 60.8416 : error: 0 : met: 1
Drift (-0.000276105) < 0.001 : priority = 4
value: 0.000362905 : error: 0 : met: 1
```

Listing 6.13 Intégration des résultats de la simulation numérique aux contraintes

Dans le cas illustré, toutes les contraintes sont toujours satisfaites. Dans le cas où ce n'aurait pas été vrai, l'optimisation à base de règles aurait de nouveau été invoquée en appliquant les offsets afin de générer un dimensionnement qui compense ces écarts. Ceci suppose que les dimensions ne seront que très peu modifiées, de telle sorte que les offsets ne sont presque pas altérés. Là où de grandes variations de paramètres sont nécessaires afin de compenser les offsets, une nouvelle simulation peut montrer que les offsets ont changés, et que les contraintes sont de nouveau non satisfaites.

6.4 Test du système

Afin de tester le système, Asimov a été utilisé pour dimensionner chaque topologie individuelle. Ces dimensionnements ont été effectués pour trois jeux de spécifications, d'agressivité variable.

Le premier jeu est le même que celui qui a déjà été utilisé lors de la description du processus de conception, et est réalisable par toutes les topologies. Il est résumé dans le Tableau 6.3.

contraintes	coûts	conditions
erreur de gain statique < 5%	minimiser puissance (1mW)	tension d'alimentation = 5V
erreur capacitive < 5%	minimiser surface (100μm^2)	fréquence d'éch. = 20MHz
précision d'ét. > 10 bits		amplitude d'entrée = 100μA
injection de charge < 5%		cellules connectées = 1
SNR > 60dB		
dérive < 1mA/s		

Tableau 6.3 Jeu de spécifications 1

Le deuxième jeu resserre les spécifications sur les erreurs associées avec la conductance de sortie, résumé dans le Tableau 6.4.

contraintes	coûts	conditions
erreur de gain statique < 1%	minimiser puissance (1mW)	tension d'alimentation = 5V
erreur capacitive < 1%	minimiser surface (100μm^2)	fréquence d'éch. = 20MHz
précision d'ét. > 10 bits		amplitude d'entrée = 100μA
injection de charge < 5%		cellules connectées = 1
SNR > 60dB		
dérive < 1mA/s		

Tableau 6.4 Jeu de spécifications 2

Le troisième jeu requiert une injection de charge faible, et constitue le jeu le plus tendu des trois (Tableau 6.5).

contraintes	coûts	conditions
erreur de gain statique < 1%	minimiser puissance (1mW)	tension d'alimentation = 5V
erreur capacitive < 1%	minimiser surface (100μm^2)	fréquence d'éch. = 20MHz
précision d'ét. > 10 bits		amplitude d'entrée = 100μA
injection de charge < 1%		cellules connectées = 1
SNR > 60dB		
dérive < 1mA/s		

Tableau 6.5 Jeu de spécifications 3

Les résultats d'Asimov, en appliquant ces spécifications à chaque topologie, sont maintenant résumés.

Dans chaque cas les valeurs de performances spécifiées, prédites et simulées sont données, permettant la comparaison de chaque quantité. De la différence entre les valeurs des performances prédites et simulées, nous pouvons identifier deux types d'erreur de prédiction, qui ont des effets différents sur le processus d'optimisation.

La première erreur provient d'une prédiction *optimiste*, où la valeur prédite de la performance est meilleure que la valeur simulée. Ceci peut signifier que la spécification correspondante n'est en réalité pas satisfaite, et peut donc conduire à un retour dans la boucle interne d'optimisation. Le temps global de conception est ainsi augmenté.

La deuxième erreur provient d'une prédiction *pessimiste*, où la valeur prédite de la performance est pire que la valeur simulée. Ceci peut sur-contraindre une valeur de dimension par rapport à d'autres spécifications non-satisfaites. Une solution possible pourrait alors ne pas être considérée.

Les résultats du processus d'optimisation sont également importants, en termes de nombre d'itérations et de temps de conception. A partir de ces résultats, nous pouvons déterminer le temps moyen d'évaluation analytique, t_a, et le comparer au temps moyen de simulation, t_s, afin d'obtenir le gain en temps d'évaluation A_T:

$$A_T = \frac{t_s}{t_a} \tag{6.12}$$

Toutes les sessions ont été exécutées sur une machine Ultra Sparc.

6.4.1 Résultats pour la cellule mémoire de courant de base

La cellule mémoire de courant de base a un temps moyen d'évaluation analytique court (t_a=7.7s), devant le temps moyen de simulation (t_s=281.3s), c'est-à-dire un gain en temps d'évaluation A_T de 36.5. Comme le montre le Tableau 6.6, la précision des prédictions est relativement bonne, à l'exception de l'injection de charge. Cette quantité apparaît être assez arbitraire, car dans un cas (jeu 1), l'erreur de prédiction est optimiste, et dans les autres cas (jeux 2 et 3), elle est pessimiste. La première configuration présente une capacité de grille faible, ce qui, comme le montre l'annexe B et la section 4.7, peut conduire à des valeurs d'injection de charge fortement sous-estimées. Les deuxième et troisième configurations présentent une GVO de source et une capacité de grille plus importantes, ce qui conduit à une sur-estimation de l'injection de charge.

La surface utilisée augmente avec l'agressivité des spécifications. Les sources principales de cette augmentation se trouvent dans la capacité de grille, pour satisfaire aux spécifications du diviseur capacitif et de l'injection de charge, et dans les longueurs des transistors, pour satisfaire à la spécification d'erreur de gain statique.

	PERFORMANCES								
	jeu 1			jeu 2			jeu 3		
	spécifiée	prédite	simulée	spécifiée	prédite	simulée	spécifiée	prédite	simulée
erreur de gain statique	<5	4.61	3.85	min	1.59	1.45	min	3.74	3.65
erreur du diviseur capacitif	<5	1.41	0.94	min	1.29	1.07	min	0.63	0.55
précision d'établissement	>10	23.0	20.4	>10	29.8	31.9	>10	47.2	32.8
erreur d'injection de charge	<5	0.98	1.75	<5	3.21	1.00	min	1.50	0.4
snr	>60	61.4	63.5	>60	62.7	81.6	>60	60.2	64.5
erreur de la dérive	<1m	0.3m	0.1m	<1m	0.2m	0.1m	<1m	0.2m	0.1m
fréquence d'échantillonnage	=20MHz			=20MHz			=20MHz		
amplitude d'entrée	=100µA			=100µA			=100µA		
tension d'alimentation	=5V			=5V			=5V		
cellules connectées	=1			=1			=1		
puissance	min	0.625mW		min	1mW		min	1mW	
surface	min	316µm2		min	347µm2		min	488µm2	
nombre d'itérations	24			79			36		
temps de conception	3m 06s			10m 12s			4m 36s		
temps de simulation	4m 30s			4m 59s			4m 35s		

Tableau 6.6 Résultats d'Asimov pour la cellule mémoire de courant de base

6.4.2 Résultats de la cellule mémoire de courant à structure cascode

Asimov a bien fonctionné sur la cellule mémoire de courant à structure cascode, où les valeurs prédites des performances étaient très proches de celles simulées (Tableau 6.7). La précision de prédiction, en ce qui concerne la performance de la précision d'établissement, montre que l'utilisation d'un modèle analytique du deuxième ordre est possible, si le pôle du cascode est maintenu à une fréquence suffisamment haute pour empêcher son intervention dans la réponse du circuit.

Le temps moyen d'évaluation analytique est plus long que celui de la cellule de base (t_a=40.6s). Le temps moyen de simulation est un peu plus long (t_s=345s), à cause du nombre plus important de transistors dans le circuit. Le gain en temps d'évaluation A_T est de 8.5 dans ce cas. Une grande partie du temps d'évaluation est consacré aux solutions DC. Cet aspect pourrait être amélioré afin d'accélérer le temps d'évaluation.

Le deuxième jeu de spécifications a nécessité deux optimisations "internes", car la simulation après la première a révélé une valeur de SNR réelle de 59.9dB, comparée à la valeur prédite de 60.2dB. Ceci a été corrigé par la deuxième optimisation. En fait, l'erreur de prédiction est systématiquement optimiste pour cette quantité. Il importe de

noter que l'erreur prédite de gain statique dans ce cas est hors spécification, car l'offset évalué par la simulation numérique après la première optimisation est négatif (-0.07). La valeur compensée (et réelle) est donc dans les spécifications.

Le dernier jeu de spécifications n'a pas été complètement satisfait : la performance de l'injection de charge est 0.05% hors spécification. Cependant, une telle valeur n'est pas très significative, car cette quantité est fortement dépendante d'autres facteurs, notamment le mismatch. Ceci ne nous permet pas d'atteindre une telle précision dans la prédiction de l'injection de charge.

Comme attendu, la surface a augmenté pour des spécifications plus agressives, ce qui est principalement dû à une valeur de capacité de mémorisation plus importante. Pour le deuxième jeu de spécifications, la spécification plus serrée de l'erreur de gain statique a nécessité une augmentation dans la transconductance du transistor mémoire (obtenue en diminuant la GVO de ce transistor). Afin d'atteindre la même performance de SNR, la capacité de grille a dû être augmentée. Pour le troisième jeu de spécifications, la valeur de la capacité de grille a été augmentée afin de minimiser l'injection de charge.

	PERFORMANCES								
	jeu 1			jeu 2			jeu 3		
	spécifiée	prédite	simulée	spécifiée	prédite	simulée	spécifiée	prédite	simulée
erreur de gain statique	<5	1.17	1.13	<1	1.02	0.95	<1	0.99	0.92
erreur du diviseur capacitif	<5	0.06	0.09	<1	0.05	0.06	<1	0.03	0.05
précision d'établissement	>10	24.5	25.1	>10	20.7	20.5	>10	11.2	12.0
erreur d'injection de charge	<5	1.55	1.10	<5	1.51	1.19	min	1.05	1.05
snr	>60	61.1	60.8	>60	61.4	61.0	>60	64.2	63.7
erreur de la dérive	<1m	131µ	45µ	<1m	99µ	37µ	<1m	64µ	21µ
fréquence d'échantillonnage	=20MHz			=20MHz			=20MHz		
amplitude d'entrée	=100µA			=100µA			=100µA		
tension d'alimentation	=5V			=5V			=5V		
cellules connectées	=1			=1			=1		
puissance	min	0.875mW		min	1mW		min	0.875mW	
surface	min	430µm2		min	618µm2		min	779µm2	
nombre d'itérations	24			23 + 9			54		
temps de conception	15m 32s			15m 42s + 6m 22s			36m 19s		
temps de simulation	5m 41s			5m 48s + 5m 49s			5m 42s		

Tableau 6.7 Résultats d'Asimov pour la cellule mémoire de courant à structure cascode

6.4.3 Résultats de la cellule mémoire de courant à structure cascode actif

Le temps moyen d'évaluation analytique pour la cellule mémoire de courant à structure cascode actif (Tableau 6.8) est le plus important de toutes les topologies (t_a=61.3s), ainsi que le temps moyen de simulation (t_s=443.8s), ce qui donne un gain en temps d'évaluation A_T de 7.2. Comme pour la topologie cascode, une grande partie du temps d'évaluation est consacrée aux solutions DC.

	PERFORMANCES								
	jeu 1			jeu 2			jeu 3		
	spécifiée	prédite	simulée	spécifiée	prédite	simulée	spécifiée	prédite	simulée
erreur de gain statique	<5	0.99	0.05	<1	0.19	0.05	<1	0.11	0.04
erreur du diviseur capacitif	<5	0.18	0.15	<1	0.06	0.15	<1	0.02	0.05
précision d'établissement	>10	33.9	31.4	>10	33.9	31.2	>10	14.6	14.6
erreur d'injection de charge	<5	2.72	1.1	<5	2.72	1.1	<1	0.83	0.96
snr	>60	63.3	60.9	>60	63.3	60.9	>60	65.6	64.1
erreur de la dérive	<1m	189µ	59µ	<1m	189µ	59µ	<1m	79µ	30µ
fréquence d'échantillonnage	=20MHz			=20MHz			=20MHz		
amplitude d'entrée	=100µA			=100µA			=100µA		
tension d'alimentation	=5V			=5V			=5V		
cellules connectées	=1			=1			=1		
puissance	min	1.04mW		min	1.04mW		min	1.4mW	
surface	min	255µm2		min	255µm2		min	647µm2	
nombre d'itérations	57 + 14			58 + 14			64		
temps de conception	49m 05s + 18m 10s			48m 13s + 14m 55s			1h 07m 34s		
temps de simulation	7m 29s + 7m 14s			7m 27s + 7m 26s			7m 23s		

Tableau 6.8 Résultats d'Asimov pour la cellule mémoire de courant à structure cascode actif

Les dimensionnements générés par l'application des premier et deuxième jeux de spécifications se ressemblent. En fait, la seule différence est la GVO du transistor source dans l'amplificateur de la branche mémoire. La différence provient de légères différences dans la phase de minimisation des coûts, pendant laquelle la performance de l'erreur de gain statique a dépassé 1.0%, violant ainsi une contrainte dans le deuxième jeu de spécifications. Ceci a conduit à une réduction du facteur de pas. La minimisation ultérieure de la surface, par une augmentation de la GVO du transistor source de l'amplificateur, applique une variation moins importante à ce paramètre.

La spécification de l'injection de charge a été satisfaite dans le troisième jeu de spécifications, contrairement au cas de la cellule cascode. Une augmentation importante dans la valeur de la capacité de mémorisation a été appliquée afin de réduire cette

quantité en-dessous de sa valeur spécifiée. Ceci constitue la source principale de l'augmentation de la surface.

6.4.4 Résultats de la cellule mémoire de courant de base S^2I

Puisque la complexité de la cellule mémoire de courant de base S^2I est similaire à celle de la cellule de base (simple boucle), le temps moyen d'évaluation analytique est du même ordre de grandeur (t_a=10.2s). Le temps moyen de simulation, cependant, ne l'est pas (t_s=492.3s), puisque la commutation des interrupteurs est plus compliquée[1]. Ceci donne un gain en temps d'évaluation A_T de 48.3, valeur la plus importante obtenue pour les cellules considérées. La conception automatisée de ce type de cellule a donc le plus à gagner (en termes de temps d'évaluation) de l'utilisation de modèles analytiques dans la boucle interne d'optimisation.

Les divergences principales entre les performances prédites et simulées (Tableau 6.9) se trouvent dans les erreurs du diviseur capacitif et de la dérive, optimistes toutes les deux. Ceci n'intervient cependant pas dans la validité globale de la solution finale, car tous les jeux de spécifications ont été satisfaits.

A cause des deux capacités de mémorisation, la taille des cellules a été plus importante que pour les cellules à simple boucle, pour une performance comparable. Toutefois, comme mentionné précédemment, l'avantage principal de la technique S^2I est de rendre l'injection de charge pratiquement indépendante du signal, réduisant ainsi la distorsion harmonique.

6.4.5 Comparaison globale

Les résultats donnés dans les sections précédentes ont été moyennés et comparés (Figure 6.8). Nous constatons une erreur moyenne de prédiction inférieure à 20%, bien en-dessous de la marge critique de 100% mentionnée en section 1.2.2. Les erreurs de prédiction pour les structures cascode et cascode actif sont particulièrement faibles, impliquant que le découplage entre le noeud de sortie et le noeud de drain du transistor mémoire tend à limiter quelques erreurs. Les coûts finaux augmentent avec la complexité pour les structures cascodes ; la capacité de mémorisation supplémentaire est la cause de l'augmentation du coût (principalement en surface) dans la structure S^2I. Le nombre moyen d'itérations est assez faible, en accord avec les propriétés des algorithmes de recherche directe. Cependant, le temps total de conception pour les structures cascodes est important, ce qui réduit l'interactivité de l'outil avec le concepteur. La même tendance est visible pour le temps moyen d'évaluation analytique: ceci est facilement compréhensible puisque la plupart du temps de conception se passe à l'intérieur de la boucle interne d'optimisation. Le fait que le noeud de sortie ne soit pas

1. Lors de l'évaluation de l'effet d'une transition d'interrupteur, le pas de temps doit être souvent réduit dans la simulation transitoire, afin d'atteindre une erreur de troncation locale (LTE - "local truncation error") acceptable. De cette façon, la convergence de la méthode d'intégration numérique est obtenue, mais au prix d'un temps de simulation plus long. Nous utilisons la méthode d'intégration numérique de Gear, qui est généralement considérée comme la méthode la plus stable pour la simulation transitoire de circuits commutés.

	PERFORMANCES								
	jeu 1			jeu 2			jeu 3		
	spécifiée	prédite	simulée	spécifiée	prédite	simulée	spécifiée	prédite	simulée
erreur de gain statique	<5	0.53	0.33	<1	0.72	0.65	<1	0.56	0.30
erreur du diviseur capacitif	<5	4.4m	0.04	<1	0.04	0.06	<1	6.8m	0.03
précision d'établissement	>10	14.8	13.0	>10	31.7	13.3	>10	12.3	10.2
erreur d'injection de charge	<5	1.34	1.34	<5	2.03	2.20	<1	0.79	0.97
snr	>60	66.1	63.0	>60	63.2	60.4	>60	68.4	64.5
erreur de la dérive	<1m	71µ	0.4m	<1m	0.1m	0.2m	<1m	17µ	0.2m
fréquence d'échantillonnage	=20MHz			=20MHz			=20MHz		
amplitude d'entrée	=100µA			=100µA			=100µA		
tension d'alimentation	=5V			=5V			=5V		
cellules connectées	=1			=1			=1		
puissance	min	0.863mW		min	0.82mW		min	0.991mW	
surface	min	635µm2		min	1012µm2		min	1652µm2	
nombre d'itérations	15			23 + 10			34		
temps de conception	2m 42s			3m 41s + 1m 46s			5m 29s		
temps de simulation	8m 13s			8m 08s + 8m 07s			8m 21s		

Tableau 6.9 Résultats d'Asimov pour la cellule mémoire de courant de base S^2I

le noeud de drain du transistor mémoire dans les structures cascodes est la cause principale de cette caractéristique, car l'affectation explicite d'une tension ne peut pas être effectuée, contairement aux structures simples. La solution DC doit être appelée à

partir d'équations analytiques - le fait que la solution DC soit plus complexe aggrave le problème. Une solution possible est de déléguer la tâche de la solution DC à un simulateur externe - le temps perdu au traitement des fichiers et à l'initialisation du simulateur devrait compenser les inefficacités liées à la recherche d'une solution DC explicite. Le temps de simulation augmente avec le nombre de transistors pour les structures cascodes, et avec l'activité de commutation pour la structure S^2I. Enfin, le gain en temps, qui représente l'intérêt principal des modèles analytiques, est important pour les structures simples, et faible pour les structures cascodes.

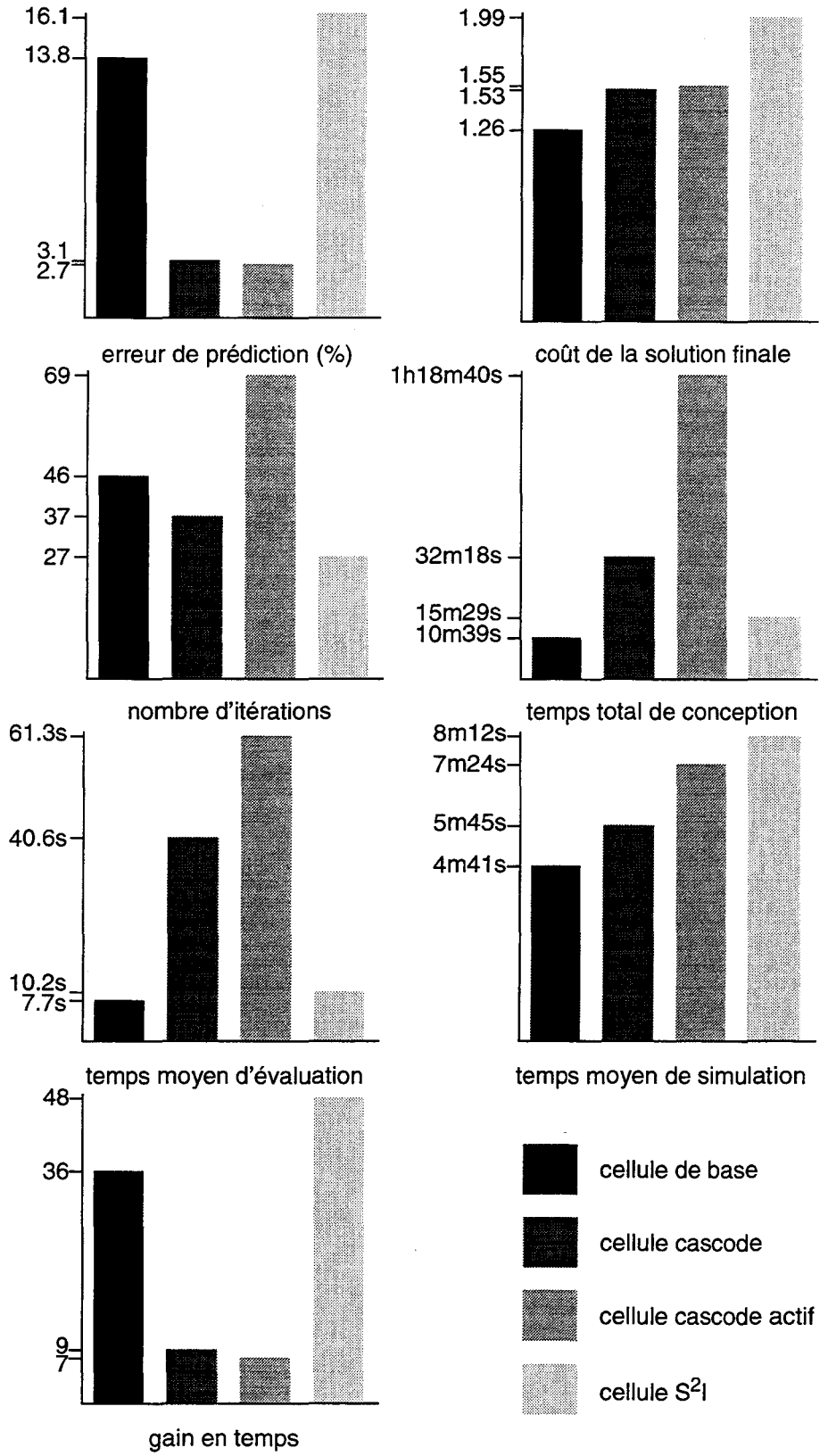


Figure 6.8 Comparaison globale des résultats de conception automatisée

6.5 L'utilisation d'Asimov dans une architecture de haut-niveau

Cette thèse a été consacrée au développement d'un outil pour automatiser la conception de cellules à courants commutés. Cependant, l'utilité d'un tel outil est accrue lorsqu'il est intégré à un environnement de plus haut niveau .

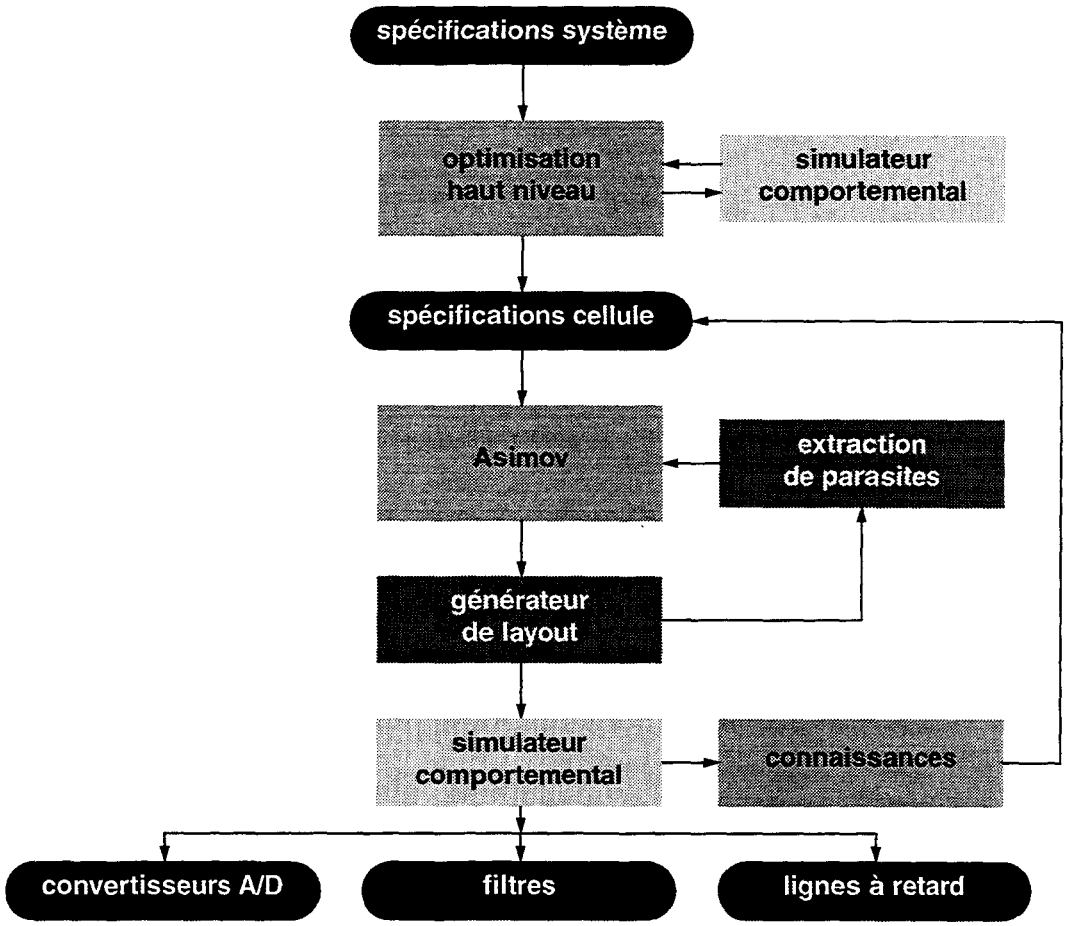


Figure 6.9 Outil de conception automatisée pour systèmes à courants commutés

Dans l'application envisagée, les spécifications pour un système à courants commutés sont fournies à un bloc de haut-niveau qui combine la simulation comportementale avec l'optimisation (soit automatique, soit manuelle) des paramètres associés aux cellules individuelles. Il est nécessaire d'effectuer ces opérations sur le système complet afin d'évaluer l'interaction de toutes les cellules, puisque les performances d'une cellule individuelle sont indissociables de son environnement.

Une fois que les spécifications au niveau système ont été décomposées en un jeu de spécifications propres à la cellule, Asimov est invoqué sur chaque cellule. Initialement, aucune information précise sur les composants parasites des noeuds externes n'est disponible, et le circuit initial utilise des valeurs nominales. Les dimensions du circuit sont ensuite transférées à un outil de génération de layout [DES97], qui permet l'extraction des valeurs effectives des composants parasites. Des itérations successives

dans cette boucle permettent la convergence vers un point qui satisfait à toutes les contraintes des cellules en prenant en compte les composants parasites.

Une autre simulation comportementale est effectuée sur le système dimensionné. Cette fois, les modèles comportementaux sont plus orientés vers le modèle du dispositif [NGO94], et peuvent prédire de façon précise le comportement d'un système à courants commutés, sans avoir cependant besoin d'utiliser la simulation numérique qui devient, à ce niveau de complexité, extrêmement coûteuse en temps de calcul. Si les performances du système ne sont pas suffisantes, des heuristiques peuvent agir sur les spécifications des cellules avant une nouvelle invocation d'Asimov. A ce niveau, il est préférable d'agir simultanément sur plusieurs spécifications de cellules, en utilisant un raisonnement qualitatif [TOU95] pour déterminer le meilleur jeu de spécifications à modifier.

Afin de générer un circuit robuste, il est nécessaire d'inclure des analyses de rendement ou de sensibilité afin que le circuit soit conçu en vue d'un rendement maximal (souvent appelé "design for manufacturability", ou DfM). Ceci ne constitue pas un point trivial, car les analyses de rendement procèdent habituellement d'une analyse Monte-Carlo qui nécessite un grand nombre de simulations identiques avec des variations statistiques sur des paramètres sensibles. Le problème, clairement énoncé, est qu'il s'agit d'une analyse post-conception, coûteuse en temps de calcul, dont les résultats sont nécessaires à une étape pré-conception.

Un domaine intéressant de recherche est celui de la conception en vue du test ("design for test", ou DfT) pour les circuits à courants commutés. Puisque l'insertion, dans le chemin du signal, de dispositifs propres aux opérations de test n'est pas souhaitable, les méthodes actuelles agissent sur les signaux d'horloge afin de transformer n'importe quelle structure en une ligne à retard de type "bucket-brigade". Le poids des modifications nécessaires à la DfT est ainsi déplacé du circuit même vers le générateur d'horloge. Ce n'est toutefois pas anodin car le nombre de lignes d'horloge doit augmenter, ce qui pourrait augmenter la diaphonie. La possibilité de générer des circuits qui incluent la fonctionnalité de test serait cependant un ajout utile à l'outil au niveau système.

Les applications ciblées sont les convertisseurs A/D (sigma-delta [MOE96] ou pipeline [BRA97]), les filtres [HUG94] et les lignes à retard [STE98]. De tels blocs fonctionnels sont des éléments importants dans la conception d'ASICs mixtes, et représentent souvent les point critiques dans la chaîne de conception. L'application décrite d'Asimov serait un moyen flexible de réduire de façon importante le temps de conception.

6.6 Comparaison avec SCADS

Pendant ce travail, un outil de conception automatisée spécifique aux filtres à courants commutés [HUG96] a été développé par une autre équipe. Ceci démontre le besoin industriel pour de tels outils et fournit de plus un point de comparaison. Le principe de son fonctionnement a été considéré en section 1.2.3, antérieur au développement

d'Asimov. Il est maintenant intéressant d'y retourner afin de comparer les deux approches.

Premièrement, la portée d'Asimov est plus restreinte que celle de SCADS, qui fonctionne à un niveau hiérarchique plus élevé. SCADS est consacré à la synthèse de filtres à courants commutés, et des travaux sont en cours afin d'étendre le concept à des convertisseurs A/D. Ces travaux résulteront en un outil capable de générer une interface complète à des circuits numériques, réalisée entièrement en courants commutés. Cependant, c'est une architecture figée qui, bien que capable de générer des circuits optimaux dans une bande de fréquences ciblée (vidéo), peut rencontrer des limitations aux limites de son domaine d'utilisation. La modification de l'architecture du circuit (qui est implémentée à plusieurs niveaux de l'outil) afin de l'adapter à l'application devient alors un travail considérable. Ceci n'est pas le cas de l'outil au niveau système décrit dans la section 6.5, car il utilise Asimov comme moteur au niveau hiérarchique le plus bas, et génère son circuit à partir de ce point. Dans ce sens, l'approche globale est plus flexible, car il est facile de changer l'architecture du système pour trouver une solution adaptée à l'application. Par contre, elle demande au concepteur de posséder la connaissance de l'implémentation du circuit à construire.

Au niveau de la cellule, SCADS est basé sur les connaissances. Cette approche est sujette à erreurs, difficile à maintenir, longue à développer et possède des cibles implicites, inconvénients qui ont déjà été cités en section 1.2.2.2. Puisque quelques connaissances sont incluses dans Asimov, ces inconvénients sont également valables pour notre approche, bien que fortement atténués. Les connaissances sont incluses dans Asimov sous la forme d'équations analytiques, qui diffèrent des équations de conception puisqu'elles peuvent facilement être validées par la simulation numérique. Les connaissances qualitatives supplémentaires dans Asimov sont également faciles à valider, mais l'ordonnancement des heuristiques dans les règles peut être difficile à établir.

SCADS a été intégré dans la chaîne de conception industrielle Cadence [CDS97]. Ceci est un avantage sur Asimov car les outils de saisie du schéma et de génération du layout existent, ce qui offre au concepteur une base de données complète pour le système. Asimov est un outil indépendant, qui s'interface seulement avec des outils de simulation. Cependant, l'intégration de SCADS dans Cadence a été poussée à un point tel que l'outil est entièrement écrit dans le langage propriétaire de Cadence, le SKILL [CDS97.1]. Ceci n'est pas une base de langage stable, à l'opposé des langages de haut niveau normalisés tel que C++ utilisé pour Asimov. Une interface à Cadence n'est pas difficile à développer, et minimise aussi l'utilisation de SKILL, facilitant ainsi des maintenances ultérieurs nécessitées par les mises à jour de Cadence.

Globalement, Asimov offre une approche plus flexible que SCADS. Cependant, plus de travail est nécessaire afin d'intégrer Asimov dans un système comparable à SCADS. Pour de nombreuses applications à courants commutés, SCADS reste une bonne solution.

6.7 Conclusion

Ce chapitre s'est consacré à la description d'Asimov, un outil de CA de cellules mémoire de courant, basé sur les techniques décrites dans les chapitres précédents. Deux aspects ont été détaillés : l'interface utilisateur, aussi bien au niveau utilisation qu'au niveau génération de topologies ; et l'application de l'outil dans un certain nombre de situations de conception.

Nous avons choisi d'utiliser une interface graphique avec Asimov, car elle est plus intuitive et plus rapide d'utilisation. Les spécifications sont communiquées par la fenêtre principale, avec des fenêtres périphériques permettant la configuration de l'algorithme d'optimisation, des modèles analytiques et de la sélection des topologies. Cependant, une fois l'outil démarré, toute l'information est dirigée vers un fichier d'historique, et l'utilisateur n'a pas d'autre information que les mises à jour des performances dans la fenêtre principale. Un processus plus interactif permettrait à l'utilisateur de voir d'un coup d'oeil l'état de progression du processus d'optimisation : quel paramètre est varié et dans quelle direction, et l'état des dimensions de la cellule. Avec ces informations, l'utilisateur pourrait interrompre et modifier le processus, si un problème de convergence était détecté.

Un autre aspect de l'interface utilisateur est le module de génération de topologies. Cette tâche comporte deux parties : la formalisation de connaissances, et la communication de ces connaissances à l'outil. La première tâche requiert un concepteur expert, et la deuxième dépend de l'outil. Nous nous sommes efforcés de rendre le processus de génération de topologies aussi simple que possible, par l'introduction d'un interpréteur de pseudo-code. Nous sommes, néanmoins, toujours conscient des nombreux défauts dans l'interface homme-machine.

La description de l'application de l'outil a été divisée en deux sections. La première s'est consacré à une analyse détaillée d'une session de conception automatisée, afin d'expliquer l'application des points majeurs de l'algorithme. La deuxième, qui est plus une analyse comparative, a appliqué plusieurs jeux de spécifications à toutes les topologies dans la bibliothèque.

Une analyse détaillée de la procédure de conception permet de démontrer comment les règles accélèrent l'optimisation d'une cellule donnée, à la fois dans la phase de satisfaction des contraintes, ainsi que dans le phase de minimisation des coûts. Par le fichier d'historique, nous pouvons facilement extraire la documentation de la conception, détaillant les phases dans la génération du dimensionnement final.

Dans le test comparatif, trois jeux de spécifications ont été appliqués à chaque topologie dans la bibliothèque. La cellule de base était rapidement limitée par ses performances ; les autres cellules mémoires, de performances améliorées, étaient capables de satisfaire la plupart des jeux des spécifications. Les seules cellules capable de satisfaire toutes les spécifications étaient les cellules cascode actif et S^2I. Cette dernière, bien qu'elle nécessite une surface de silicium plus importante, a de meilleures performances d'un

point de vue système, car pour une performance équivalente d'injection de charge, la distorsion harmonique est moins importante.

Il importe de noter que, puisque l'algorithme d'optimisation recherche un minimum local, il dépend du point de départ. La solution finale est également très dépendante des heuristiques qui constituent les règles. Finalement, la technique d'optimisation à base de règles requiert une extension de la phase de préparation dans les outils à base de connaissances. Non seulement les modèles analytiques et les points de départ doivent être générés, mais de plus les heuristiques doivent également être déterminées, codées et testées. La génération automatique des connaissances, concevable pour les modèles analytiques et les points de départ en utilisant l'analyse symbolique et la manipulation des équations, semble moins certaine en ce qui concerne la génération d'heuristiques.

Le test comparatif constitue, dans une certaine mesure, l'exploration de l'espace de conception. Cette exploration, avec Asimov, nous permet de tester les limites de chaque cellule, une des applications principales de l'outil. L'autre application principale est son utilisation dans un enchaînement complet de conception. Bien que le dimensionnement des cellules soit une tâche importante et critique, d'autres sont aussi importantes, et aussi critiques. Comme exemple extrême, la conception d'un système qui utilise un grand nombre de cellules à courants commutés identiques (par exemple une ligne à retard [STE98]) nécessite très peu de conception de la cellule. Par contre, elle nécessite énormément de macromodélisation et simulation du système, de conception de circuits périphériques, de layout méticuleux, d'optimisation des pistes de signal, etc. C'est ici que d'autres outils de CA sont utiles, comme la synthèse haut-niveau avec la modélisation comportementale, la CA bas-niveau pour circuits continus, la synthèse de layout bas-niveau, et un processus pour coordonner la génération du CI complet. Mais l'ingéniosité humaine génère souvent des idées de circuit qui ne sont pas supportées dans un système fixe. Tel est le problème qui se présente aux chercheurs dans le domaine de la conception automatisée analogique haut-niveau.

Références

[BRA97] Mark Bracey, "Current Domain Analogue-to-Digital Conversion Techniques for CMOS VLSI", PhD Thesis, University of Southampton, 1997

[CDS97] Cadence Design Systems, "Design Framework II", 1997

[CDS97.1] Cadence Design Systems, "Skill Language User Guide", 1997

[DES97] M.A. Dessouky, "CAIRO: Symbolic Analog Layout Language User Manual", Laboratoire LIP6, 1997

[GNU95] GNU C++ version 2.7.2, *Free Software Foundation*, 1995

[HUG94] J.B. Hughes, K.W. Moulding, "An 8MHz, 80Ms/s Switched-Current Filter", Proc. IEEE International Solid-State Conference, pp. 60-61, 1994

[HUG96] J.B. Hughes *et al.*, "Automated Design of Switched-Current Filters", *IEEE Journal of Solid State Circuits*, vol. 31, no. 7, pp. 898-907, July 1996

[MOE96] N. Moeneclaey, A. Kaiser, "A High-Resolution Current-Mode Sigma-Delta Converter", Proc. European Solid-State Circuits Conference, pp. 260-263, 1996

[NGO94] P. N'Goran, "Simulation à temps discret de circuits à mémoire de courant", PhD Thesis, Université des Sciences et Techniques de Lille, 1994

[OCO97] I. O'Connor, A. Kaiser, "Synthèse de circuits à courants commutés", *Proc. Colloque CAO des circuits intégrés et systèmes*, pp 2-5, January 1997

[STE98] B. Stefanelli, I. O'Connor, L. Quiquerez, A. Kaiser, D. Billet, "An Analog Beam-Forming ASIC Using Switched-Current Delay Lines", *to be published*

[TOU95] C. Toumazou, C.A. Makris, "Analog IC Design Automation", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 2, pp. 218-254, February 1995

CONCLUSION

Le travail présenté dans cette thèse a conduit au développement d'un outil de CA spécifique à la conception de cellules à courants commutés. Les préceptes capitaux qui sont à la base de cet outil sont :

- l'utilisation de modèles analytiques dans la boucle interne d'optimisation afin de réduire le temps d'évaluation des performances à des niveaux acceptables,
- une boucle externe d'optimisation qui consiste en une simulation numérique. Celle-ci évalue la différence entre les critères de performances modélisés et simulés, c'est-a-dire l'erreur d'évaluation,
- l'optimisation à base de règles, ou guidée, qui améliore le rendement d'un algorithme numérique simple en incorporant des connaissances heuristiques. L'optimisation tente ainsi d'imiter le processus utilisé par un concepteur,
- la limitation de l'étendue de l'outil à des cellules de bas niveau, permettant ainsi plus de flexibilité au concepteur par rapport à l'utilisation des résultats de la synthèse dans un système.

L'objectif de cette conclusion est de résumer chaque chapitre et d'en extraire les points importants. Nous commentons également les limitations actuelles de l'outil et les perspectives pour les travaux futurs.

Une vue d'ensemble des méthodes existantes disponibles dans la CA analogique a été donnée dans le premier chapitre. Les limites des approches extrêmes, à base d'optimisation et à base de connaissances, a conduit à la conclusion que les deux approches peuvent être utilisées conjointement, afin d'améliorer la performance globale de la CA. A travers trois outils de l'état de l'art, l'efficacité de cette approche composite a été démontrée. Cependant, chaque outil est plus ou moins restreint aux circuits continus. Le seul outil utilisé dans la CA de circuits à courants commutés [HUG96] est

un outil entièrement à base de connaissances, et utilise l'approche descendante. La considération de tous ces points a permis le développement de l'architecture de l'outil de CA, conforme aux points exposés ci-dessus.

Le premier point à résoudre était la modélisation analytique des différentes topologies de cellules mémoire de courant. Premièrement, dans le deuxième chapitre, nous avons réfléchi à l'implémentation des modèles, et une structure cohérente a été développée. Les modèles du circuit sont décrits à deux niveaux. Le premier niveau consiste en un modèle simple basé sur des équations de dispositif explicites (et donc inversibles) du premier ordre, afin de générer un point de départ, nécessaire aux algorithmes d'optimisation locale. Le deuxième niveau consiste en un modèle détaillé qui utilise des modèles industriels de dispositif autonomes, appelés à partir des équations analytiques. Ces dernières sont écrites en termes de paramètres intrinsèques aux dispositifs, et sont donc indépendantes du modèle de dispositif utilisé (actuellement le modèle MOS Philips Model 9 [VEL95]). Un tel système est alors plus facile a maintenir et peut suivre les avancés technologiques.

Le modèle analytique de la cellule mémoire de courant de base a ensuite été développé. Dans l'analyse de circuits à courants commutés, les effets physiques (tels que l'erreur de gain statique, l'injection de charge, etc.) sont plus évidents à modéliser précisément que leurs effets cumulatifs sur le spectre de l'erreur d'échantillonnage (tels que l'offset, la distorsion harmonique, etc.). Cependant, c'est la situation inverse à celle des mesures de circuits, ou de la spécification haut-niveau de circuits. Comme l'outil de CA requis n'est pas un outil de CA haut-niveau, mais plutôt un outil de niveau cellulaire, la formulation de spécifications en termes d'effets physiques individuels peut être justifiée. Mais, c'est toujours dans le contexte du contrôle par un autre processus (qui peut également être à base de règles), qui relie les spécifications de haut-niveau à celles de niveau cellulaire de façon heuristique.

Les caractéristiques de performances ont été donc générées pour :

- *l'erreur de gain statique* : la composante statique de l'erreur de conductance, qui est due à la conductance drain-source finie du transistor mémoire (modulation du courant de sortie par la tension de sortie),
- *l'erreur due au diviseur capacitif* : la composante dynamique de l'erreur de conductance, qui est due à la capacité drain-grille qui forme un diviseur capacitif avec la capacité de mémorisation,
- *la précision d'établissement* : l'évaluation de l'erreur d'acquisition qui existe à la fin de la période d'échantillonnage due au temps d'établissement de la cellule,
- *l'erreur de l'injection de charge* : à la fin de la période d'acquisition, l'interrupteur drain-grille s'ouvre et libère dans la capacité mémoire une partie de la charge contenue dans son canal, modifiant ainsi la valeur mémorisée. Cette erreur est critique pour les performances de la cellule et un système de compensation de l'injection de charge est habituellement implanté dans la cellule,

- *le bruit* : le bruit sur la grille de la cellule mémoire a deux composantes : le bruit échantillonné pendant l'acquisition, et le bruit de la cellule pendant la phase de restitution.
- *l'erreur de la dérive* : pendant la phase de mémorisation, où la cellule est inactive, un courant de fuite en provenance de l'interrupteur d'acquisition modifie la charge sur la capacité de mémorisation. Il en résulte que la tension stockée sur cette capacité subit une dérive proportionnelle au temps de mémorisation.

Le développement d'un modèle analytique nécessite des connaissances approfondies et est sujet à erreurs. Ceci constitue une limite majeure de l'outil, car le temps de préparation (le développement de modèles et leur validation) peut nécessiter des jours, voire des semaines, de travail. Pourtant, il existe à présent très peu d'autres solutions : une approche purement à base d'optimisation réduit de façon radicale l'interactivité du concepteur, à cause des temps de synthèse très longs ; et la génération automatique d'équations n'est pas, pour l'instant, possible pour des quantités grand-signal ou qui dépendent du temps.

En modifiant la valeur de chaque variable de conception autour d'un point nominal, un aperçu de l'espace de conception pour chaque performance peut être obtenu. Ceci permet la validation ultérieure des modèles analytiques, et la génération de règles qualitatives de conception, ou heuristiques, qui peuvent être utilisées dans l'optimisation à base de règles. Chaque heuristique décrit la relation qualitative entre une performance individuelle et un paramètre de conception individuel, constituant ainsi des connaissances de conception vitales. Les dépendances de toutes les performances en fonction de tous les paramètres peuvent être décrite en utilisant les classifications d'heuristiques suivantes : augmentation monotone, diminution monotone, non-monotone, effet mineur, pas d'effet. Cependant, le développement de telles heuristiques représente, comme le développement de modèles analytiques, une tâche de longue durée nécessitant des connaissances approfondies, et truffée d'erreurs potentielles. L'analyse quantitative limitée autour du point nominal permet la génération d'heuristiques initiales, mais des connaissances de conception sont toujours requises afin de juger de la validité des heuristiques ainsi générées.

Des analyses similaires ont été effectuées pour trois autres topologies de cellule à courants commutés (cascode, cascode actif et S^2I de base) dans le troisième chapitre. Bien d'autres topologies de cellule existent, mais la construction et la maintenance d'une bibliothèque exhaustive, dans un cadre à base de connaissances, constitue une tâche difficile, sinon impossible. Pour l'outil prototype, nous avons décidé de limiter les modèles analytiques à ceux décrivant des techniques couramment utilisées. Ces modèles ont été divisés en deux catégories : les variantes structurelles et opérationnelles. Les variantes structurelles (cascode et cascode actif) modifient la structure de la cellule sans changer le mode opératoire. Les variantes opérationnelles (S^2I) changent les procédures de commutation ou le mode opératoire des transistors (et peuvent donc nécessiter des modifications dans les circuits périphériques) mais retiennent un niveau similaire de complexité. L'idée de ce développement est que les extensions ultérieures de la bibliothèque sont ainsi facilitées, puisque d'autres cellules

sont souvent des combinaisons de techniques individuelles. Un résumé des modèles développés, ainsi que ceux qui peuvent facilement être développé dans les versions futures de l'outil, est donné dans le Tableau 1.

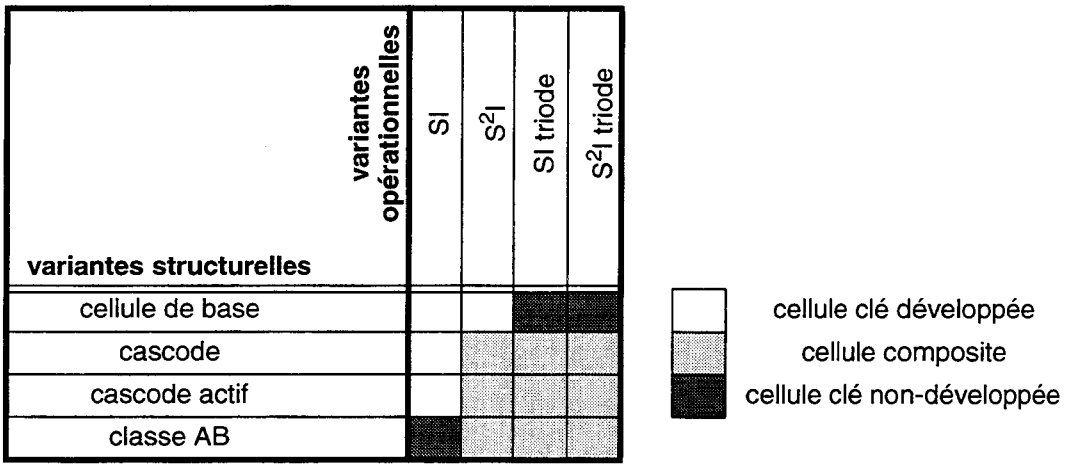


Tableau 1 Classification des topologies de cellules mémoire de courant

Le quatrième chapitre a traité de la simulation numérique de cellules à courants commutés. Un environnement de simulation a été développé, dans lequel les performances de la cellule, pour lesquelles des équations analytiques correspondantes existent, peuvent être évaluées. Cet environnement est utilisé à la fin de la boucle interne d'optimisation, où la simulation numérique peut révéler que des contraintes satisfaites soient en réalité non-satisfaites. Dans ce cas, l'erreur d'évaluation est utilisée comme offset afin de compenser les imprécisions dans les équations analytiques lors d'une optimisation supplémentaire en boucle interne.

L'environnement de simulation a également été utilisé pour valider le modèle analytique de façon générale. Les mêmes conditions, que celles utilisées pour générer les heuristiques, ont été prises, et les courbes ont été comparées. Les résultats étaient en général favorable, mettant en évidence une bonne concordance entre modèle analytique et simulation numérique. Les erreurs principales qui existent sont dans les équations de la précision d'établissement. Bien que le plus grand soin ait été pris pour la modélisation correcte de cette quantité, le comportement non-linéaire et dépendant du temps reste relativement difficile à prédire. Cependant, les résultats montrent que le modèle analytique est souvent pessimiste, ce qui, si ces erreurs doivent subsister, est préférable à un modèle optimiste.

Le développement d'un algorithme d'optimisation à base de règles a été décrit dans le cinquième chapitre. L'objectif de l'optimisation à base de règles est d'augmenter le rendement de méthodes purement quantitatives par l'introduction d'informations qualitatives, c'est-a-dire des heuristiques. A cause de la nature, non-linéaire et dépendant du temps, du circuit devant être optimisé, l'utilisation de méthodes qui requièrent des gradients est infaisable. Le choix a donc été limité à des méthodes de recherche directes ou statistiques. L'algorithme de Hooke et Jeeves (recherche directe) a été choisi à cause de sa simplicité, et aussi car des connaissances peuvent facilement y

être incorporées avec un profit immédiat. Il nécessite, néanmoins, un point de départ, généré dans les deuxième et troisième chapitres.

L'algorithme a ensuite été modifié pour accepter des contraintes. Celles-ci peuvent être classées en trois catégories : physique, fonctionnelle et spécifiée. Seule la dernière requiert une évaluation complète des performances : la violation des autres peut être détectée par simple inspection des dimensions ou par une analyse statique limitée.

Les modifications suivantes de l'algorithme se sont focalisées sur les règles. L'algorithme choisit la règle qui correspond à l'erreur individuelle de contrainte la plus importante. Les paramètres de conception associés à cette règle sont ensuite modifiés dans la direction spécifiée. Ceci réduit effectivement le vecteur de variables de conception, et augmente le rendement de l'algorithme.

Dans le sixième chapitre, l'outil de CA de cellules à courants commutés Asimov a été décrit. Nous avons utilisé une interface graphique pour communiquer des informations entre l'utilisateur et le système. Bien que les fenêtres disponibles suffisent pour un outil prototype, plus d'effort doit être consacré à cet aspect dans les versions futures. La GUI est, pour le concepteur, très importante. Il doit être possible de voir ce que fait l'optimiseur (c'est-à-dire quelle heuristique est appliquée, l'état courant des dimensions de la cellule), et également d'interrompre et de modifier manuellement le processus d'optimisation.

La génération de topologies par l'utilisation de descriptions en pseudo-code a également été décrite. Un exemple d'une session Asimov a ensuite été détaillé, montrant les phases principales dans le processus de conception. Comme montre l'exemple, la génération de documents par le fichier log est relativement simple, permettant ainsi la génération d'archives systématiques de conception.

Asimov a ensuite été utilisé pour dimensionner chaque topologie pour trois jeux de spécifications. Les résultats ont montré un avantage de performances des cellules S^2I sur les cellules à simple boucle, mais au prix d'une surface plus importante. Pour une tension d'alimentation suffisamment élevée, les cellules cascode ou cascode actif peuvent réaliser des performances relatives à l'injection de charge du même ordre de grandeur que celles de la cellule S^2I. Mais cette erreur d'échantillonnage n'est pas indépendante du signal dans les cellules à simple boucle, alors qu'elle l'est dans la cellule à double boucle. Aussi, comme les tensions d'alimentation continue de baisser, les performances de cellules qui utilisent des techniques d'empilement de transistors sont affectées de façon défavorable.

Finalement, les performances d'un outil de CA sont jugées par les performances d'un circuit synthétisé et fabriqué. Cependant, nous avons pris comme référence un circuit synthétisé et simulé. Nous avons préféré nous consacrer aux aspects de la modélisation analytique et de l'optimisation à base de règles de l'outil. Pour compléter le travail, un outil de synthèse de layout doit être intégré dans Asimov afin de générer un circuit en

silicium, de pouvoir extraire les parasites et réévaluer le circuit. Ceci permettrait alors la validation formelle de l'outil.

L'architecture d'Asimov n'est pas nécessairement limitée aux cellules à courants commutés. Sa structure à base de règles pourrait également être utilisée pour dimensionner d'autres types de circuit[1]. Dans le domaine à temps-continu, les modèles analytiques et les règles heuristiques pourraient en principe être générés de manière automatique, en utilisant des outils d'analyse symbolique, un concept similaire à celui d'ASAIC [GIE91] (Figure 1). Si, à l'avenir, les techniques d'analyse symbolique deviennent suffisamment puissantes pour pouvoir traiter les circuits à courants commutés, une des limitations majeures de l'outil, c'est-à-dire la génération de modèles et d'heuristiques, aura disparu.

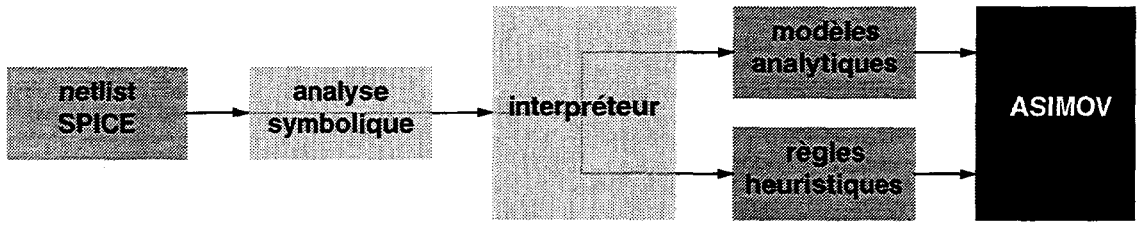


Figure 1 Interface d'analyse symbolique pour la génération de modèles analytiques et d'heuristiques

Asimov, utilisé comme moteur de conception au niveau de la cellule dans un outil de plus haut-niveau, représente une solution flexible et relativement ouverte. Il existe un bon équilibre entre la vitesse (obtenue par l'utilisation d'équations analytiques, qui réduisent le temps d'évaluation ; et d'heuristiques, qui réduisent le nombre total d'évaluations) et la précision (obtenue par l'utilisation de la simulation numérique dans la boucle externe d'optimisation).

1. Cependant, le fait qu'une méthode d'optimisation par recherche directe est utilisée le rend moins efficace quand il est appliqué à des circuits qui peuvent être linéarisés, les méthodes de gradient étant dans ce cas mieux adaptées. Une extension possible serait la capacité d'attacher une méthode de conception à un type donné de circuit.

Références

[GIE91] G. Gielen, W. Sansen, "Symbolic Analysis for Automated Design of Analog Integrated Circuits", *Kluwer Academic Publishers*, 1991

[HUG96] J.B. Hughes *et al.*, "Automated Design of Switched-Current Filters", *IEEE Journal of Solid State Circuits*, vol. 31, no. 7, pp. 898-907, July 1996

[VEL95] R. Velghe, D. Klaassen, F. Klaassen, "MOS Model 9", Unclassified Report NL-UR 003/94, Philips Research Laboratories, Eindhoven, 1995

LEXIQUE DES TERMES UTILISÉS

ASIC	circuit intégré spécifique ("application-specific integrated circuit")
Asimov	"A switched-current (SI) MOdule generation enVironment"
AWE	asymptotic waveform evaluation
β	paramètre de transconductance d'un transistor, $\mu C_{ox}W/L$ (A/V^2)
BCMC	cellule mémoire de courant de base ("basic current memory cell")
CAO	conception assistée par ordinateur
CMC	cellule mémoire de courant
CMOS	complementary metal oxide semiconductor (process)
C_{ox}	capacité de l'oxyde de grille en $F/\mu m^2$
CPU	unité centrale de traitement ("central processing unit")
CA	conception automatisée
CI	circuit intégré
DfM	conception en vue de maximiser le rendement ("design for manufacturability")
DfT	conception en vue de test ("design for test")
f_s	fréquence d'échantillonnage ("sampling frequency")
g_d	conductance de sortie d'un transistor
g_m	transconductance d'un transistor
GUI	interface graphique utilisateur ("graphical user interface")
GVO	tension de grille utile d'un transistor, $V_{gs}-V_t$ ("gate voltage overdrive")

HDL	langage de modélisation comportementale ("hardware description language")
I_{ds}	courant drain-source d'un transistor
L	longueur d'un transistor
μ	mobilité des charges dans le canal d'un transistor N ou P (cm^2/Vs)
MNA	analyse nodale modifiée ("modified nodal analysis")
MOS	metal oxide semiconductor (transistor)
PSD	densité spectrale de puissance ("power spectral density")
PWL	linéaire par segments ("piecewise linear")
S^2I	double boucle
SNR	rapport signal à bruit ("signal to noise ratio")
THD	distorsion harmonique totale ("total harmonic distortion")
V_{bs}	tension substrat-source d'un transistor
V_{ds}	tension drain-source d'un transistor
V_E	tension d'Early d'un transistor
V_{gs}	tension grille-source d'un transistor
v_{gst}	tension de grille utile d'un transistor, $V_{gs}-V_t$
V_t	tension de seuil d'un transistor
W	largeur d'un transistor

A L'ESPACE DE CONCEPTION DE LA CELLULE DE BASE : MODÈLE ANALYTIQUE

Tous les paramètres formels de conception de la cellule mémoire de courant de base ont été variés afin de générer un aperçu de la façon dont chaque performance individuelle varie en fonction de chaque paramètre de conception individuel. L'objectif de cette analyse quantitative est double : d'abord, de déterminer visuellement les règles qualitative de conception, ou heuristiques, à appliquer lors de l'optimisation à base de règles ; et ensuite, de valider la précision analytique. Ce second point est considéré dans l'annexe B.

Les paramètres ont été variés dans une zone limitée autour d'un point nominal. Ce dernier a été choisi comme le résultat d'une optimisation antérieure pour un ensemble de spécifications simples, laissant suffisamment de marge pour l'amélioration de chaque performance individuelle. Les variations ont été choisies dans les limites imposées par la faisabilité : aucun point dans les courbes ne représente une violation de contraintes physiques ou fonctionnelles.

Idéalement, les espaces de conception seraient entièrement caractérisés en utilisant une matrice comportant tous les paramètres de conception. Cependant, le temps de caractérisation serait excessivement long, et il y aura peu de différence entre cette solution et la solution à base de tableaux décrite dans la section 1.2.1. Les différences reposent sur le fait que la première solution est (en principe) indépendante de la technologie et ne requiert pas une discrétisation fine pour obtenir les résultats qualitatifs (heuristiques). A l'inverse, la deuxième solution est quantitative, et donc dépend de la technologie. Elle nécessite également une discrétisation fine.

Le courbes suivantes sont disposées en fonction de performance :

- Figure A.2, "L'espace de conception de l'erreur de gain statique," à la page 191
- Figure A.3, "L'espace de conception de l'erreur due au diviseur capacitif," à la

A chaque page, le format suivant est adopté en fonction du paramètre de conception varié afin d'obtenir la courbe :

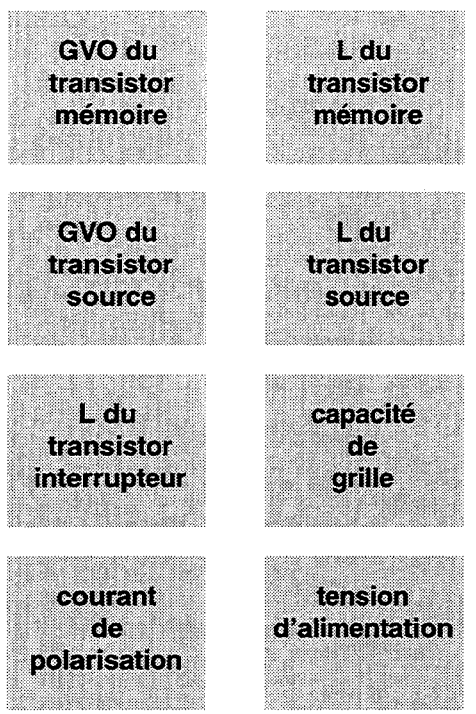


Figure A.1 Légende pour les courbes d'espace de conception

Une discussion de ces résultats a été donnée dans la section 2.12.

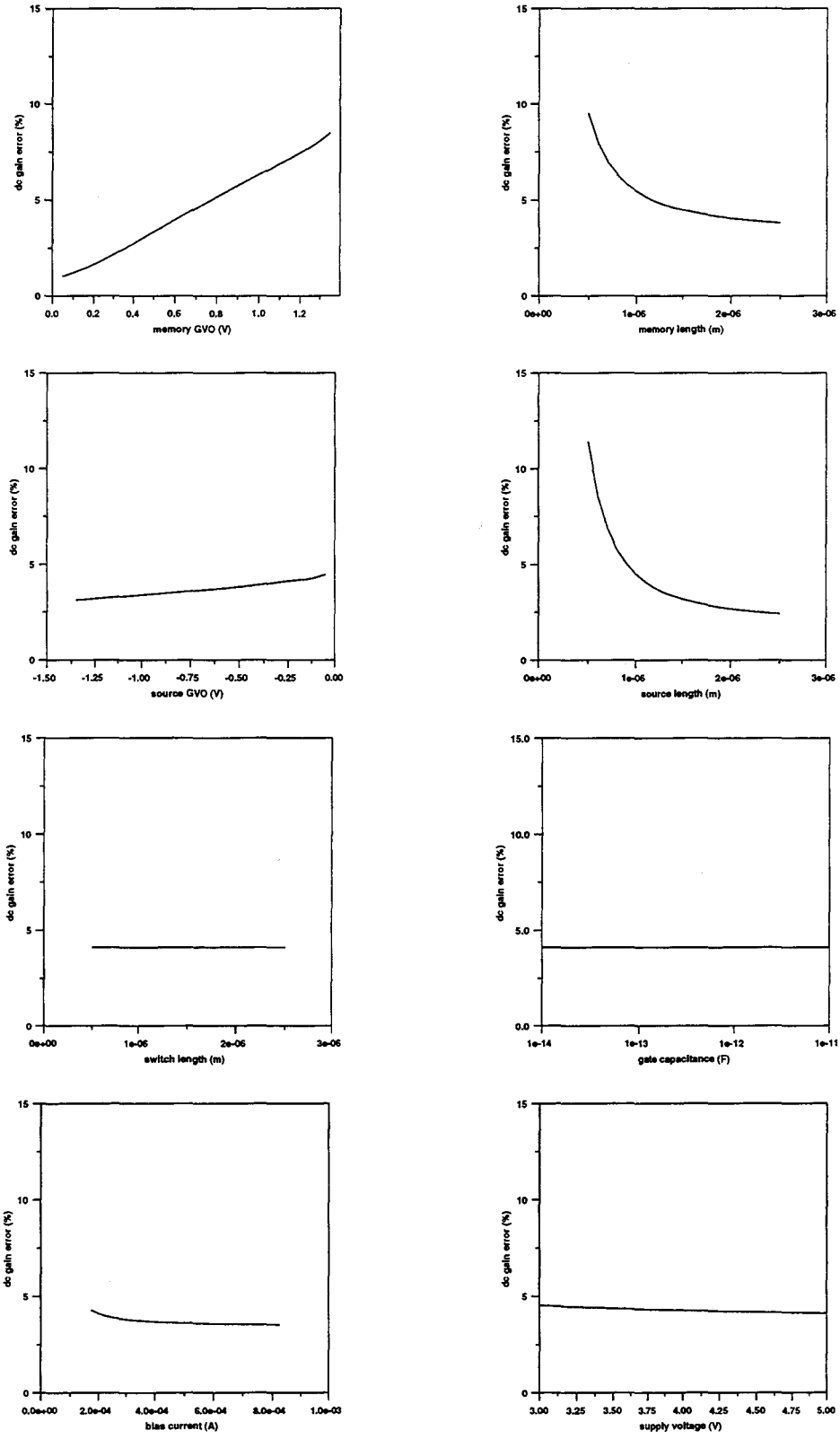


Figure A.2 L'espace de conception de l'erreur de gain statique

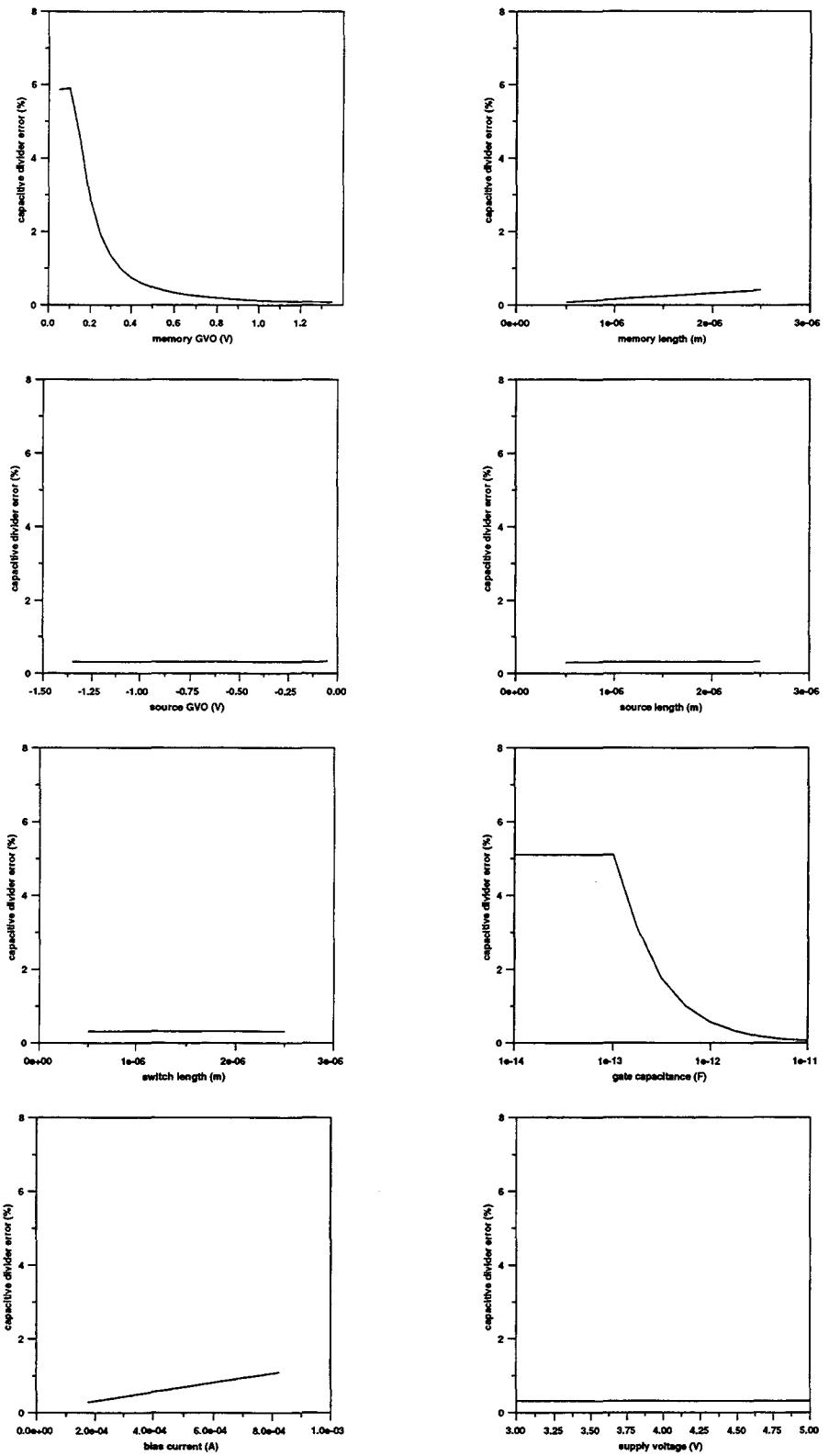


Figure A.3 L'espace de conception de l'erreur due au diviseur capacitif

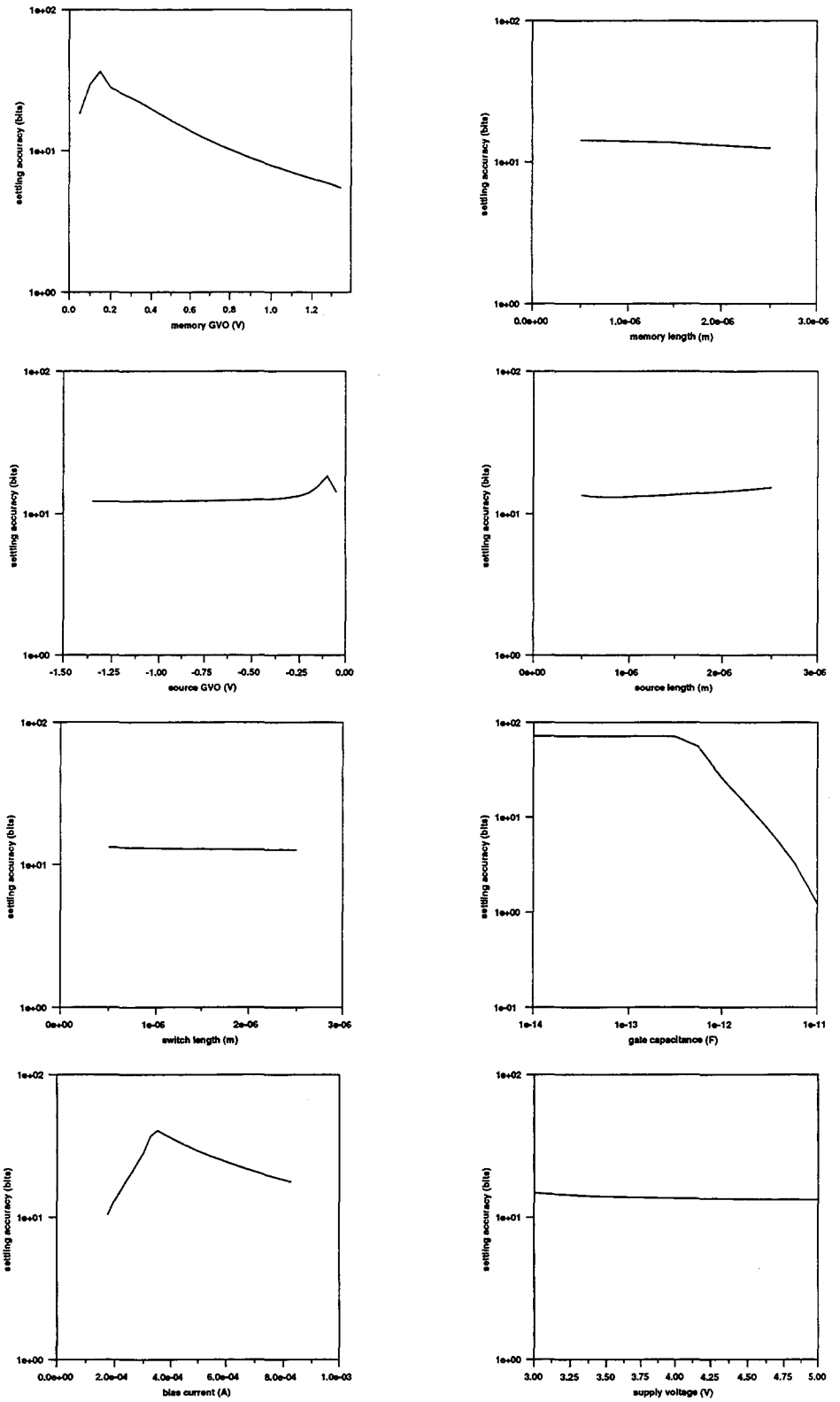


Figure A.4 L'espace de conception de la précision d'établissement

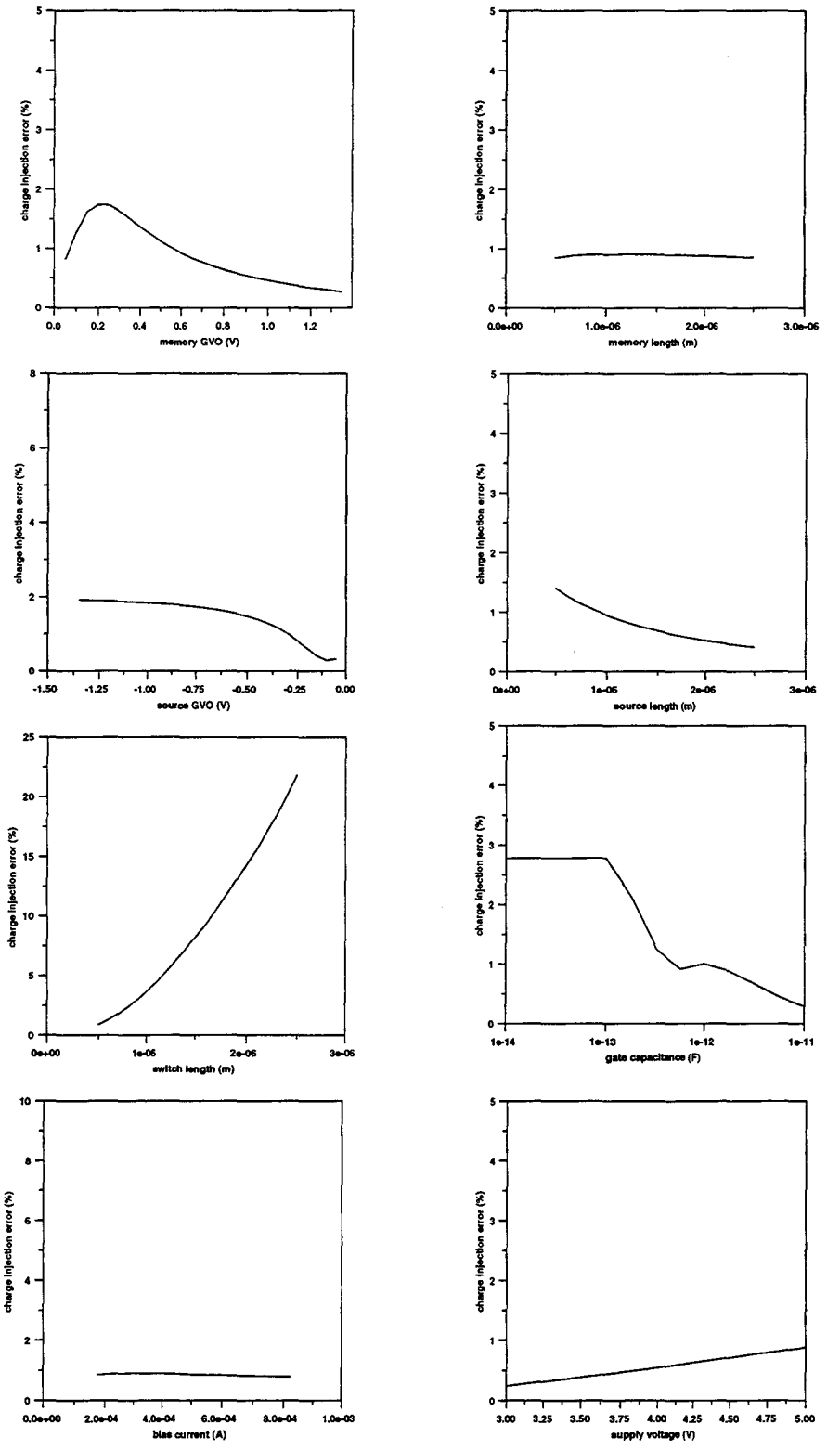


Figure A.5 L'espace de conception de l'injection de charge

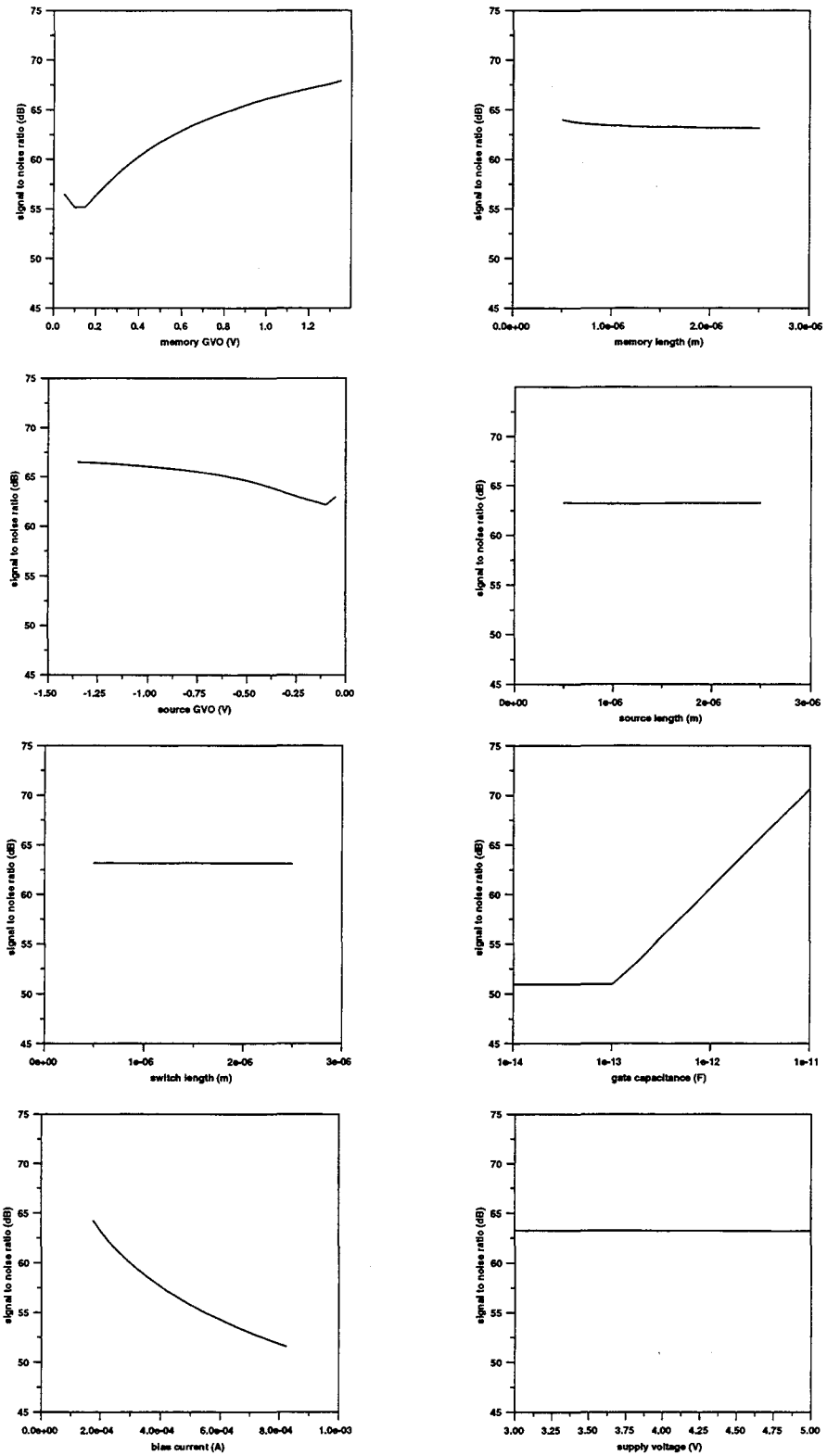


Figure A.6 L'espace de conception du rapport signal à bruit

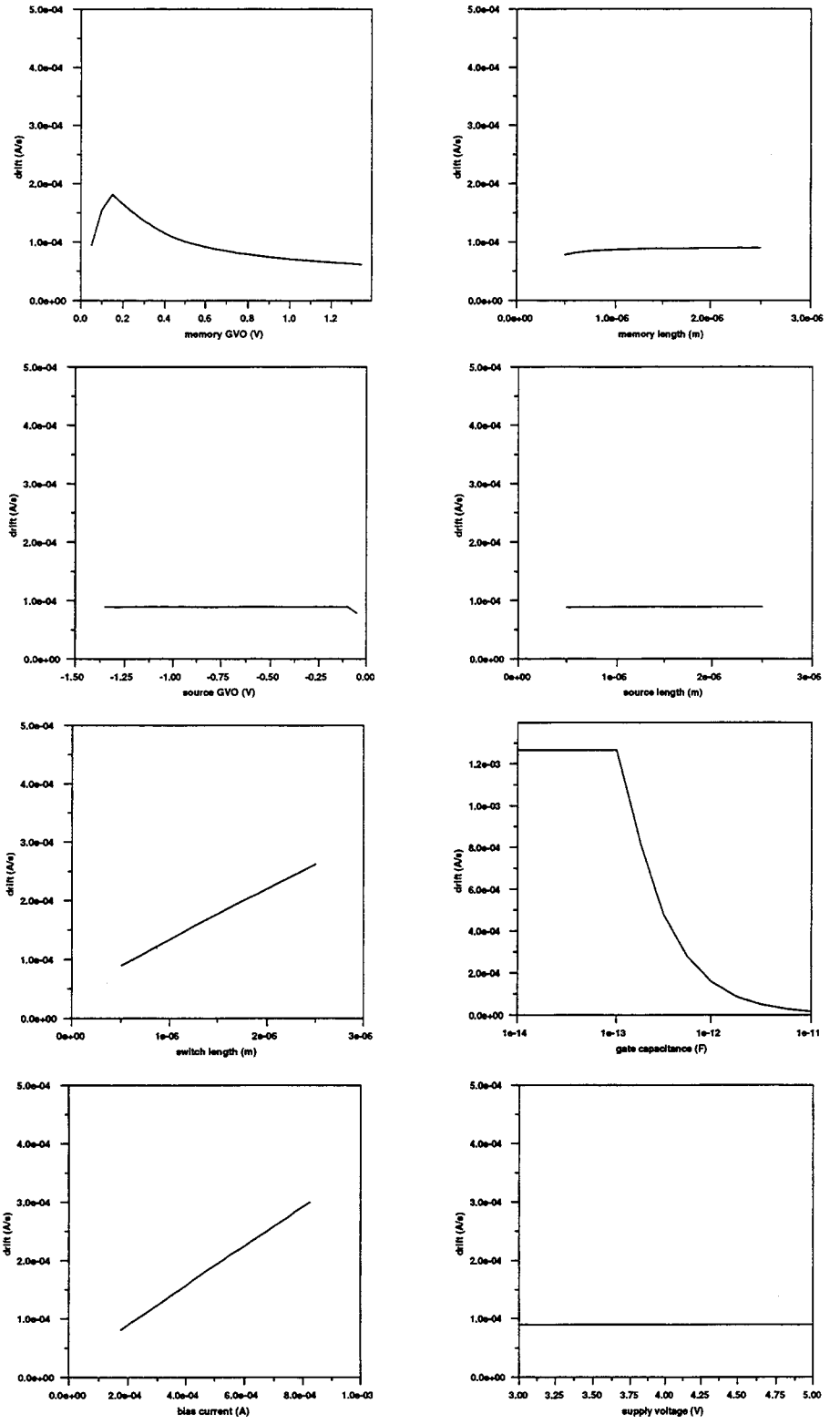


Figure A.7 L'espace de conception de l'erreur due à la dérive

B L'ESPACE DE CONCEPTION DE LA CELLULE DE BASE : CONFRONTATION ENTRE LE MODÈLE ANALYTIQUE ET LA SIMULATION NUMÉRIQUE

En utilisant les dimensions effectives générées pour chaque vecteur de dimensions formelles, l'environnement de simulation numérique peut être appliqué afin de comparer les résultats des équations analytiques avec les valeurs simulées des performances. De cette manière, la précision de chaque équation analytique peut être évaluée.

Ceci suppose que les valeurs simulées des performances peuvent être considérées comme références. Ceci est vrai, mais les références finales restent celles les mesures effectuées sur un lot de circuits fabriqués. Cependant, les équations analytiques modélisent des effets physique, et ont un effet cumulatif sur le signal de sortie. L'évaluation de la contribution individuelle de chaque effet dans un environnement de test de circuit représente une tâche complexe. De plus, le nombre total de points considérés dans cette analyse est d'environ 200 ; ce chiffre augmenterait pour un nombre plus important de dimensions formelles. La fabrication et la mesure de plusieurs lots de 200 circuits de caractérisation différents représente une expérience coûteuse. Si tous les circuits étaient dans le même boîtier, un système complexe de multiplexage analogique devrait être implanté afin de restreindre le nombre de broches. Ce système devrait être conçu afin de ne pas modifier les caractéristiques individuelles des cellules.

Nous utilisons donc la simulation numérique pour comparer les performances prédites par les équations analytiques avec les performances "réelles". Comme dans l'annexe A, les courbes suivantes sont disposées en fonction de performance :

- Figure B.2, "L'espace de conception de l'erreur de gain : prédit et simulé," à la page 199
- Figure B.3, "L'espace de conception de l'erreur due au diviseur capacitif : prédit et simulé," à la page 200

- Figure B.4, "L'espace de conception de la précision d'établissement : prédit et simulé," à la page 201
- Figure B.5, "L'espace de conception de l'injection de charge : prédit et simulé," à la page 202
- Figure B.6, "L'espace de conception du rapport signal à bruit : prédit et simulé," à la page 203
- Figure B.7, "L'espace de conception de l'erreur due à la dérive : prédit et simulé," à la page 204

De nouveau, le format suivant est adopté par rapport aux paramètres de conception variés afin d'obtenir les courbes .

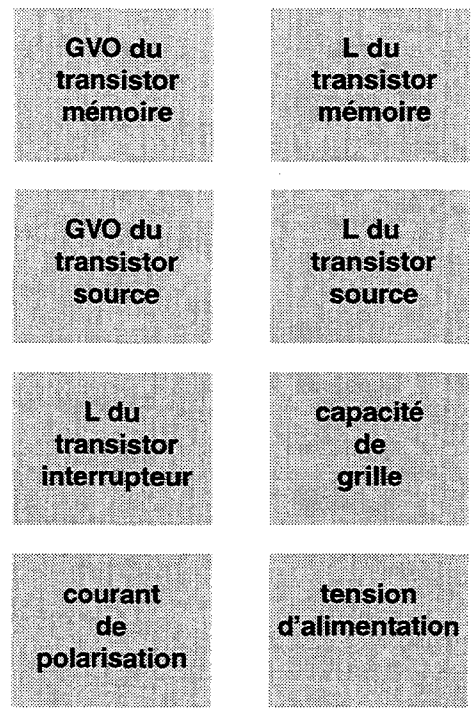
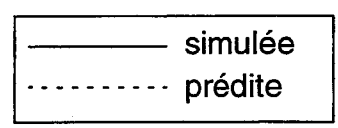


Figure B.1 Légende pour les courbes d'espace de conception

Pour chaque courbe, la ligne pleine représente les performances simulées, et la ligne en pointillés représente les performances prédites.



Une discussion des résultats se trouve dans la section 4.7.

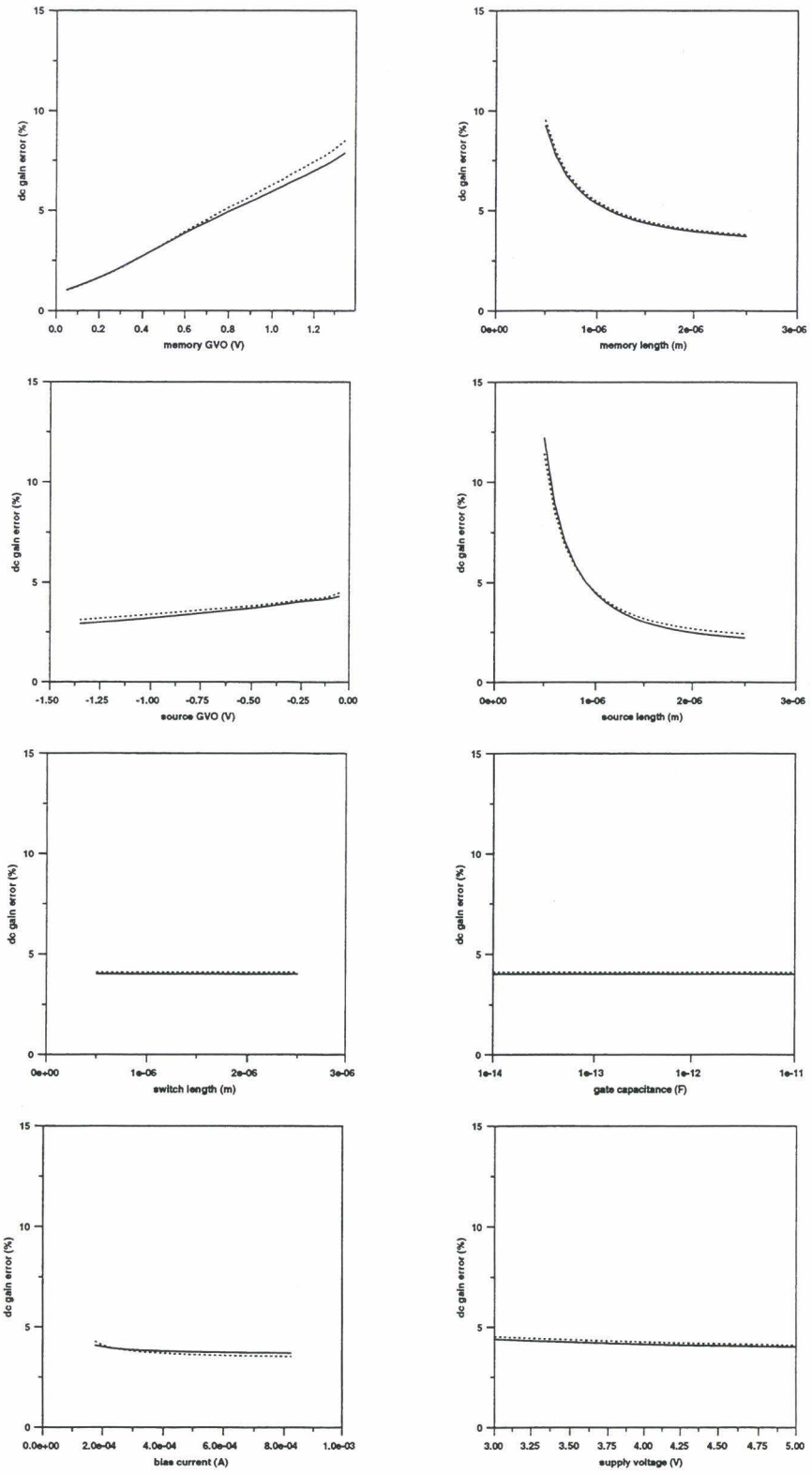


Figure B.2 L'espace de conception de l'erreur de gain : prédit et simulé

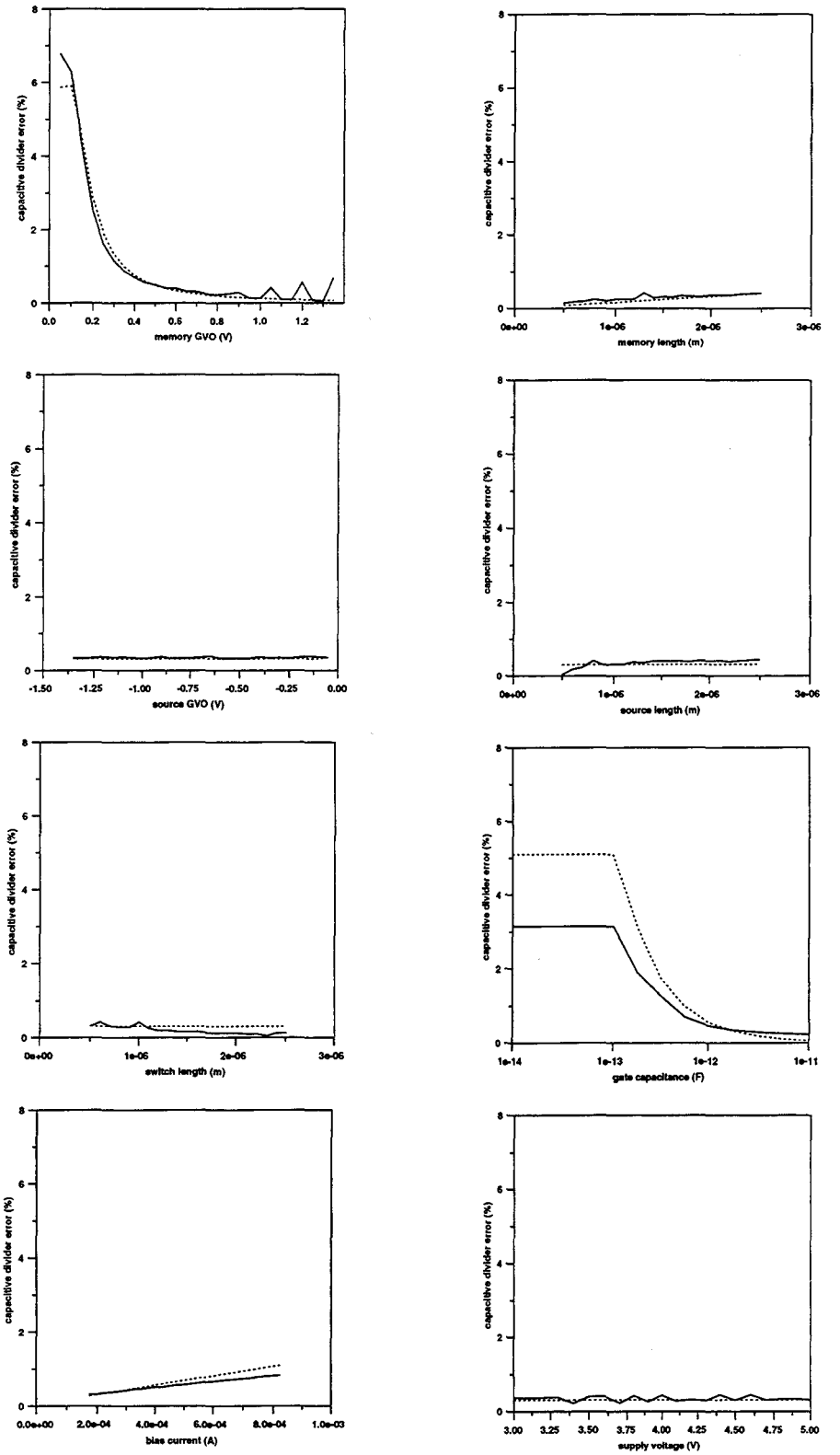


Figure B.3 L'espace de conception de l'erreur due au diviseur capacitif : prédit et simulé

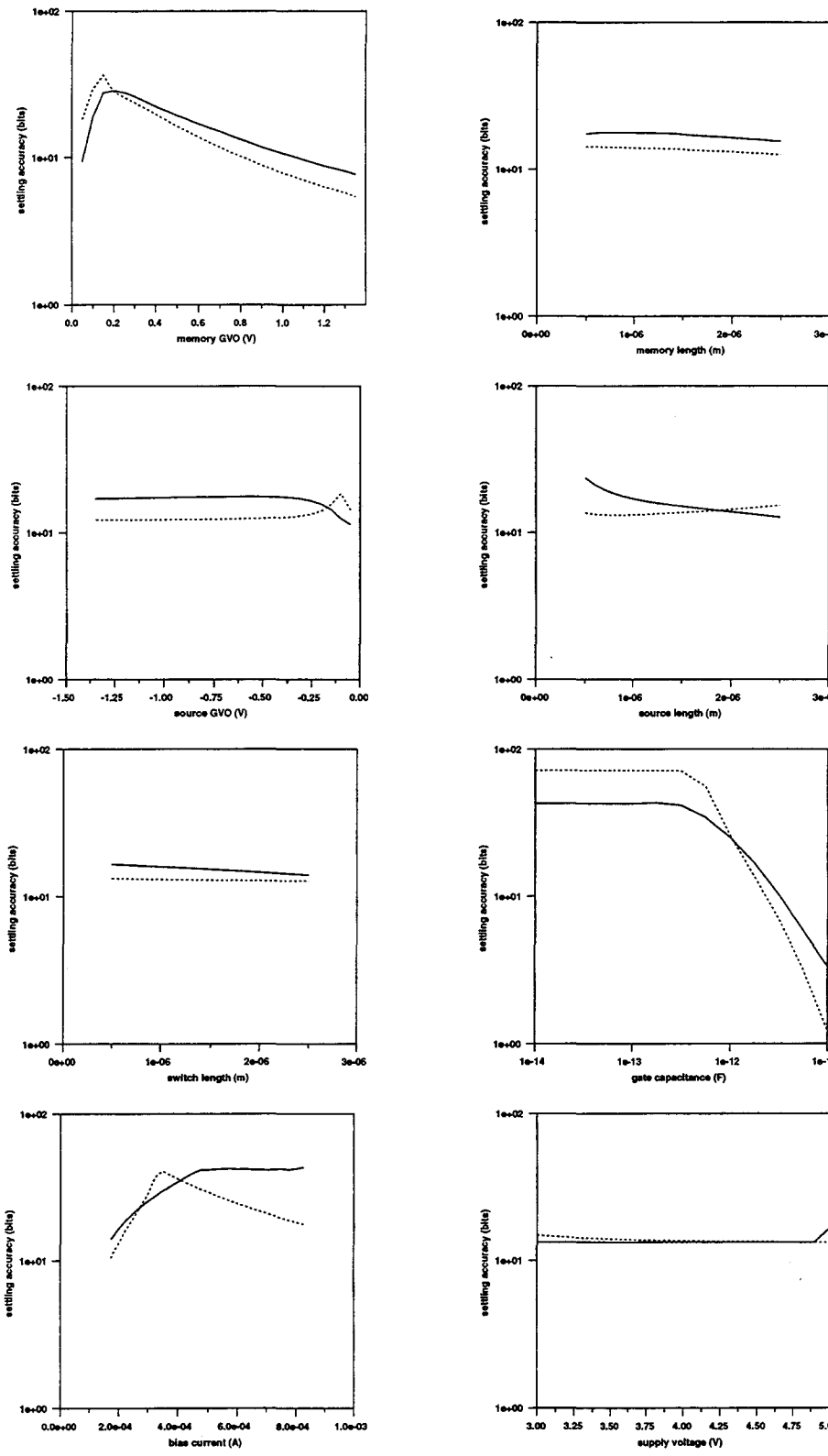


Figure B.4 L'espace de conception de la précision d'établissement : prédit et simulé

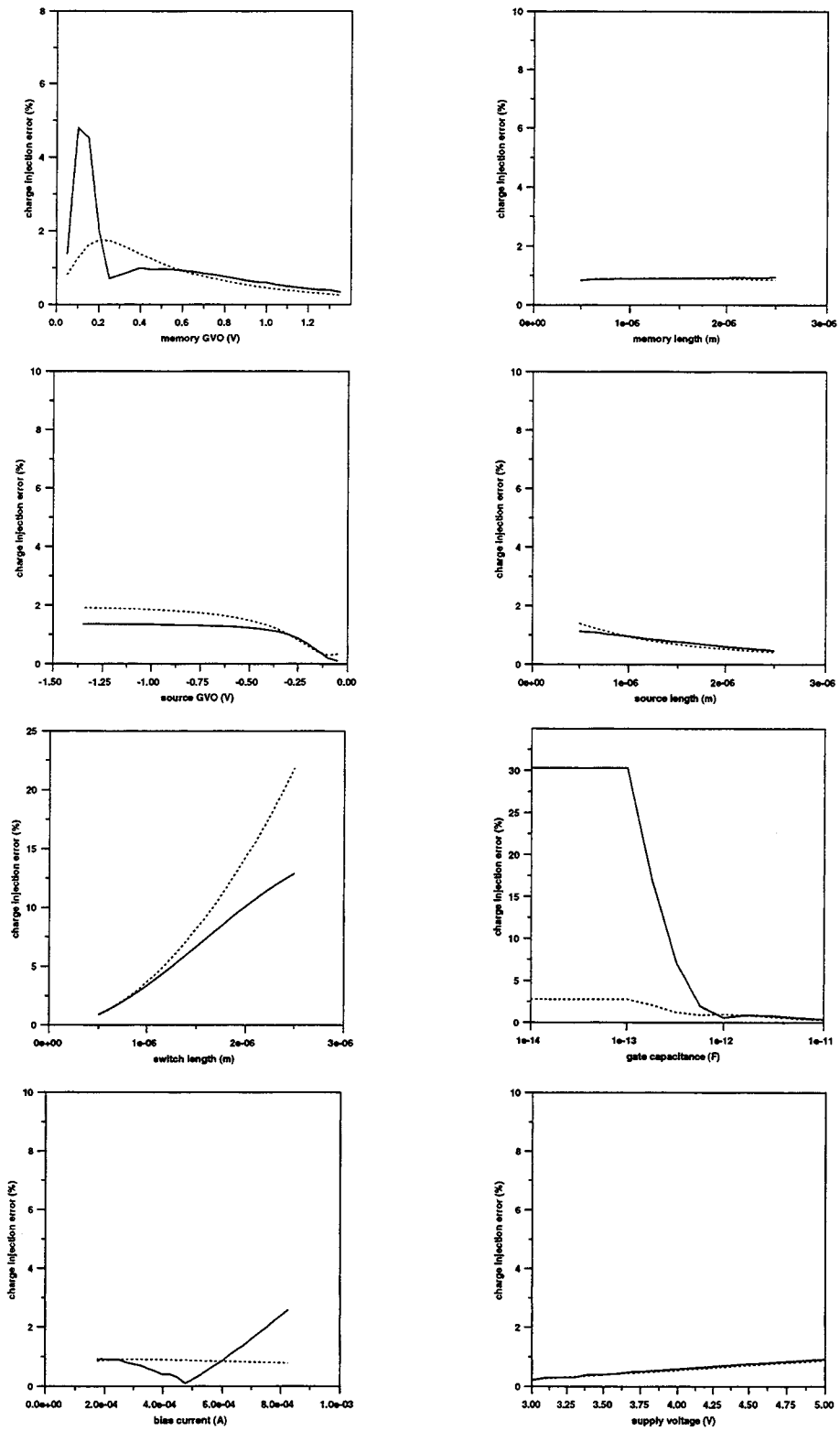


Figure B.5 L'espace de conception de l'injection de charge : prédit et simulé

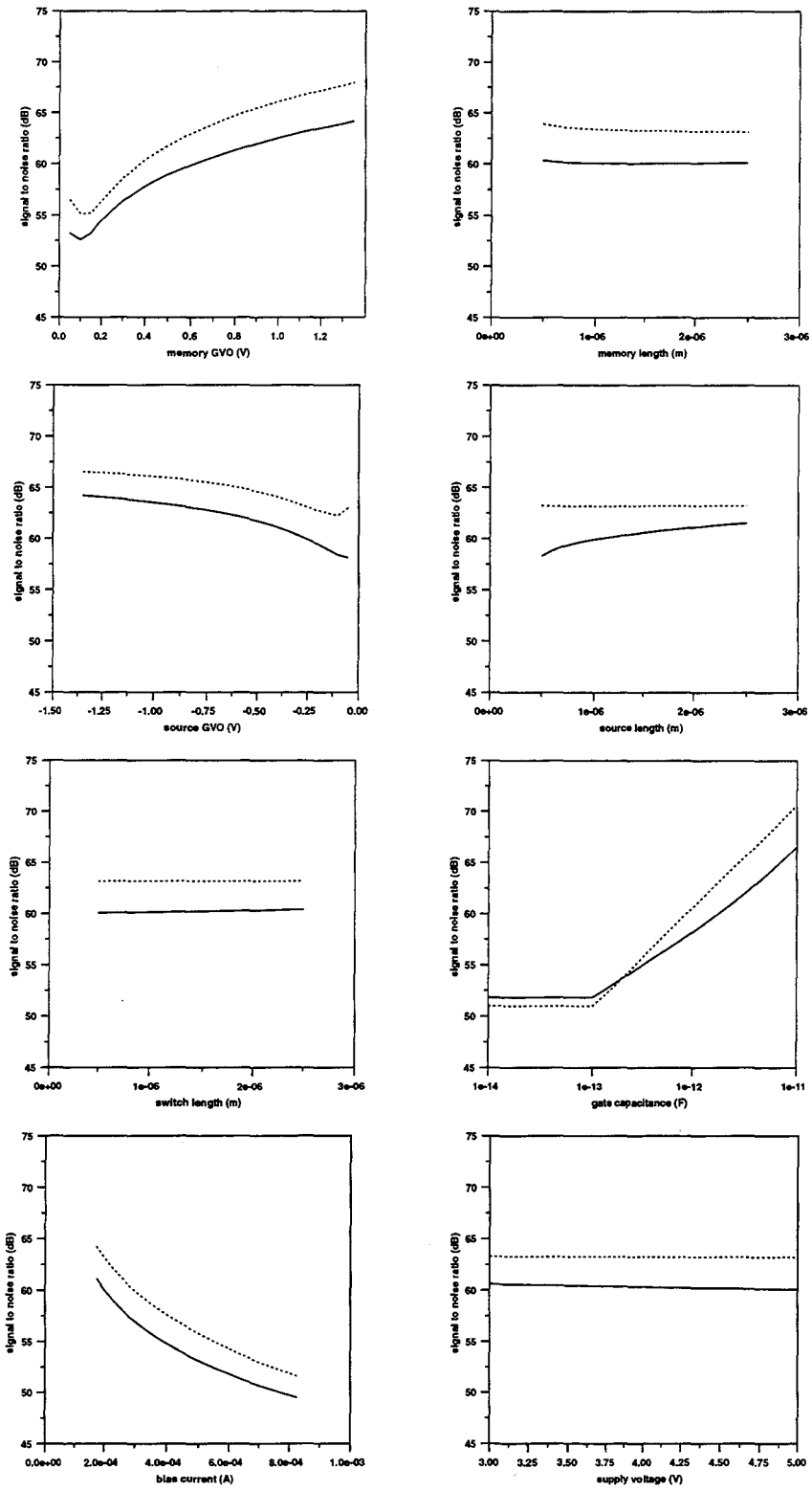


Figure B.6 L'espace de conception du rapport signal à bruit : prédit et simulé

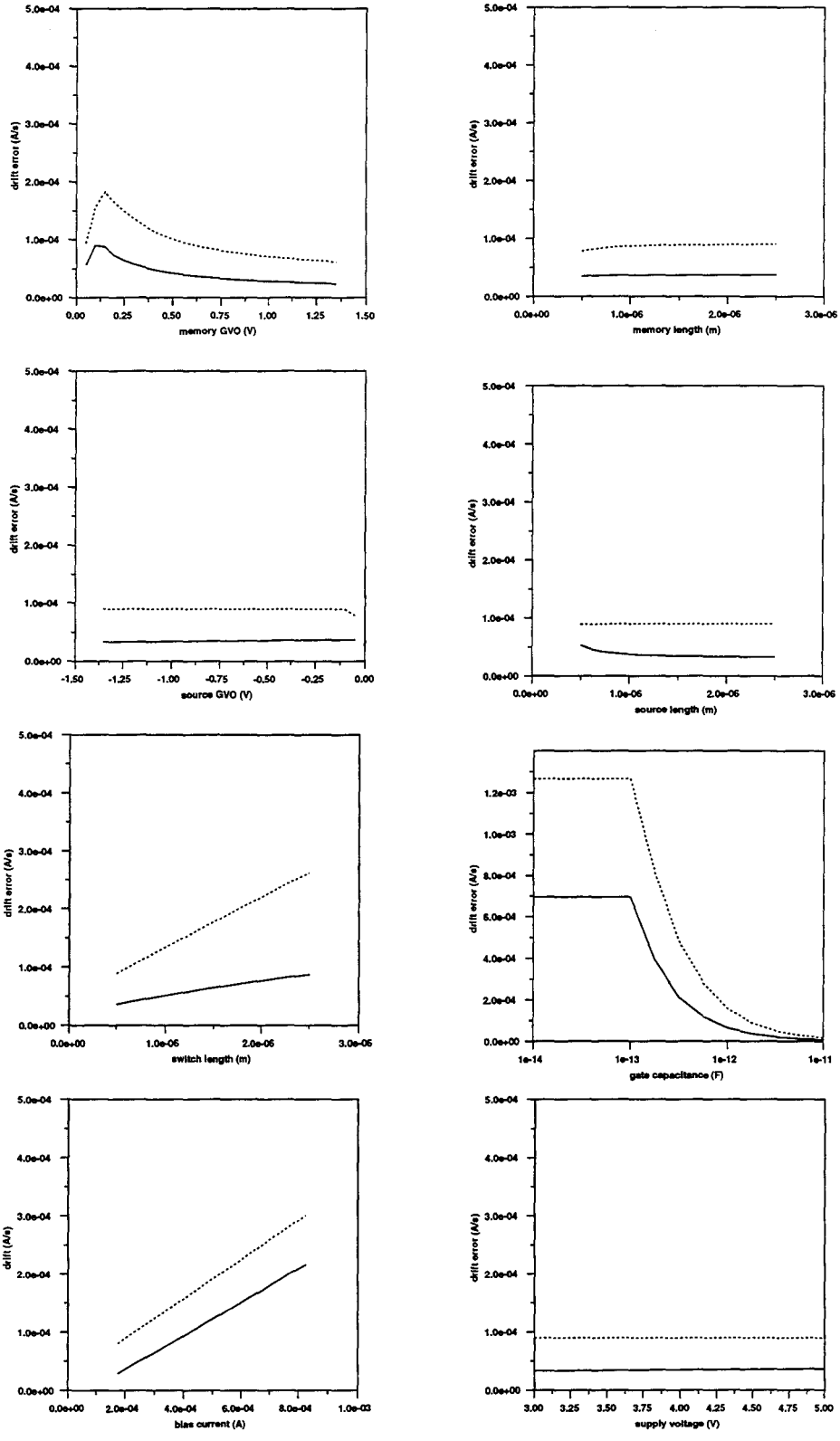


Figure B.7 L'espace de conception de l'erreur due à la dérive : prédit et simulé

C IMPLÉMENTATION D'ASIMOV EN C++

L'objectif de cette annexe est de documenter l'implémentation C++ de la structure d'Asimov, qui a été schématisée dans la section 1.2.5 (reproduit dans la Figure C.1). Dans le domaine informatique, tous les blocs dans ce schéma du système global sont représentés par des objets de classe ou par des drapeaux booléens.

Il importe de noter que, tout au long de cette annexe, les exemples de code sont dénués de constructeurs, de destructeurs, de fonctions d'interface aux classes, d'opérateurs, de pièges à erreurs et de sorties alphanumériques. Ceci permet une meilleure lisibilité du code.

C.1 L'architecture du système

Le niveau le plus élevé de code C++ qui décrit cette architecture est donné dans le Listing C.1. Tous les objets sont déclarés comme pointeurs vers des objets initialisés. Le drapeau *run* sert à deux choses :

- il constitue un signal de sélection pour le bloc de gestion de coûts (*design manager*).

  Si *run* = *faux*, c'est la première fois que ce bloc est invoqué. Sa tâche dans ce cas est de transcrire les spécifications externes en spécifications internes.

  If *run* = *vrai*, ce bloc a été invoqué par un message d'échec provenant du bloc de sélection de topologie. Dans ce cas, il doit modifier les spécifications internes par la conversion de contraintes en coûts.

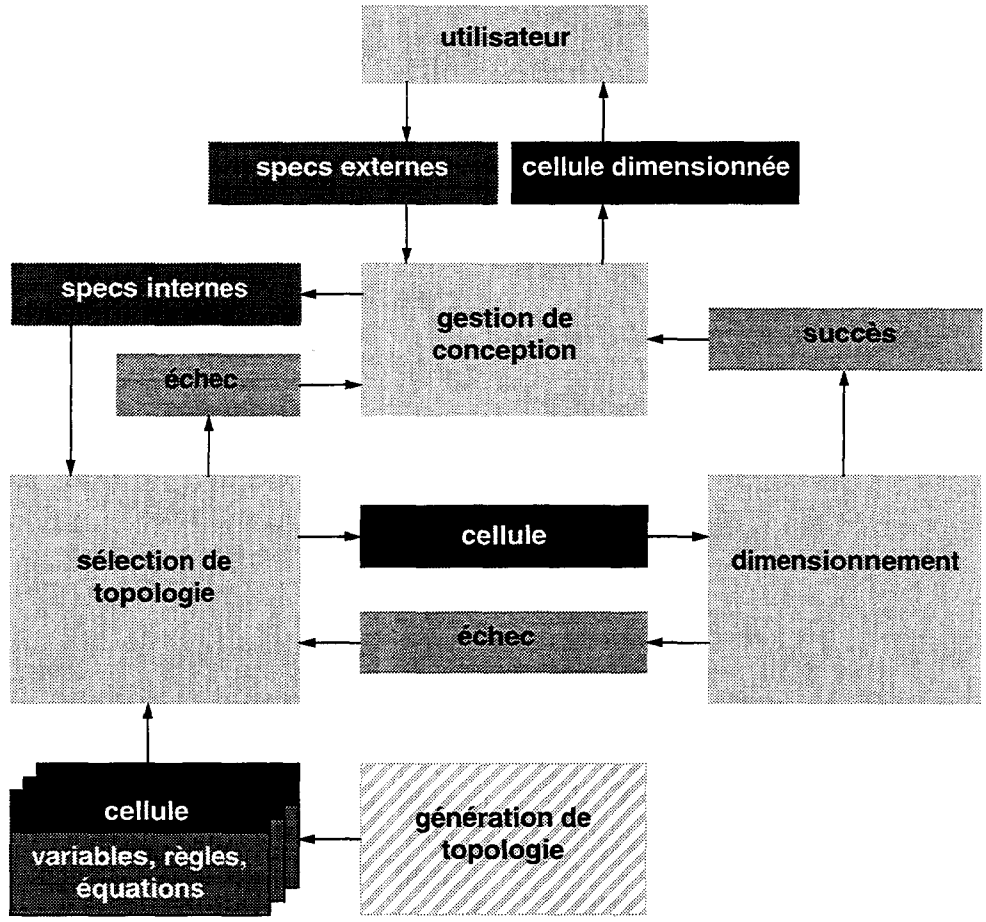


Figure C.1 Architecture système d'Asimov

- il constitue également une condition pour la boucle *while* globale.

  Tant que *run* = *vrai*, le processus de synthèse continue. Seul le bloc de gestion de conception peut sortir sans erreur par l'utilisation de la méthode output_to_user. Le système a été construit de telle sorte que, finalement, cette méthode doit être appelée.

La tâche du *bloc de sélection de topologie* est de sélectionner une topologie, et d'y faire pointer la cellule actuelle. Cette sélection s'effectue par un dimensionnement initial de la cellule candidate (une procédure explicite), qui génère également le point de départ. Si cette opération se termine par un dimensionnement réussi, la cellule est considérée comme une solution possible aux spécifications données. A l'inverse, si un problème se produit à ce niveau, la cellule est rejetée, car insuffisamment performante pour satisfaire aux spécifications.

```
specs* external = new specs;
specs* internal = new specs;
manager* design_manager = new manager;
topology* topology_selector = new topology;
design* current_design = new design;
sizer* sizing_module = new sizer;

bool run_flag = FALSE;

do
{
 design_manager->manage_loop ( external, internal, &run_flag );
 while ( topology_selector->select ( internal, sizer_flag, current_design ) )
 {
  current_design->short_print ();
  if ( sizing_module->size ( current_design, internal ) )
  {
   design_manager->output_to_user ( current_design );
  }
  sizing_module->reset ();
 }
 topology_selector->reset ();
} while ( run_flag );
```

Listing C.1 Code C++ *Asimov*

Les spécifications internes et le pointeur de cellule actuelle sont transférés au *bloc de dimensionnement*, qui constitue la majeure partie du système. La cellule est déjà dimensionnée (initialement) via le point de départ, généré par le *bloc de sélection de topologie*. Les règles de conception sont alors utilisées dans la variation des paramètres de conception, et la cellule converge (en principe) vers un dimensionnement satisfaisant à toutes les spécifications.

Avant de détailler les objets principaux de la boucle, c'est-à-dire les blocs de gestion de conception, de sélection de topologie et de dimensionnement, il est essentiel de comprendre la structure des classes d'objet sur lesquelles ils agissent, c'est-à-dire *specs* et *design*.

C.2 Formalisation des spécifications utilisateur (classe *specs*)

Les spécifications représentent le moyen par lequel l'utilisateur communique ses besoins au système. Elles peuvent être classées en trois catégories, qui sont représentées par des listes dans Asimov.

C.2.1 Classe *constraint*

Les contraintes, comme codées dans le Listing C.2, caractérisent les limites des performances acceptables, et consistent en un moyen d'évaluer la performance concernée (*performance_ptr*), un *identificateur de relation* (supérieur à, ou inférieur à), une valeur de limite (*value*), une valeur de *priorité* et un *offset*. La valeur de l'offset représente la différence entre les performances prédites et simulées de la cellule.

```
class constraint
{
public:
 bool met ();
 double error ();

private:
 performance* performance_ptr;
 token relation_token;
 double value;
 unsigned int priority;
 double offset;
};
```

Listing C.2 Code C++ *constraint*

Il existe également deux méthodes : *met*, et *error*.

C.2.1.1 Méthode *met*

Cette méthode retourne *vrai* si la contrainte est respectée, c'est-à-dire si la valeur de la performance (avec offset) satisfait aux spécifications (Listing C.3).

```
bool constraint::met ()
{
 if ( relation_token == CT_GREATER_THAN )
 {
  // performance value > constraint -> TRUE
  return ( value < performance_ptr->read_value () + offset );
 }
 if ( relation_token == CT_LESSER_THAN )
 {
  // performance value < constraint -> TRUE
  return ( value > performance_ptr->read_value () + offset );
 }
}
```

Listing C.3 Méthode C++ *constraint::met*

C.2.1.2 Méthode *error*

Si la contrainte n'est pas respectée,

$$error = \left| \frac{value - performance + offset}{priority \cdot value} \right| \quad \text{(C.1)}$$

Il est important de noter que la priorité la plus élevée est *priority=1*.

Si la contrainte est respectée, l'erreur évaluée est nulle (Listing C.4).

```
double constraint::error ()
{
 if ( ! met () )
 {
  return ( fabs ( value - ( performance_ptr->read_value () + offset ) ) / ( value * priority ) );
 }
 else
 {
  return ( 0.0 );
 }
}
```

Listing C.4 Méthode C++ *constraint::error*

C.2.2 Classe *cost*

Les coûts (Listing C.5) n'ont pas de limite, par opposition aux contraintes, qui doivent être satisfaites relativement à une valeur fixée à l'avance. Ils ont néanmoins un *identificateur de direction* d'optimisation (minimiser, maximiser). Les autres aspects (*value*, *offset*), sont similaires à ceux des contraintes.

```
class cost
{
public:

private:
 performance* performance_ptr;
 token direction_token;
 double good_value;
 double offset;
};
```

Listing C.5 Code C++ *cost*

C.2.3 Classe *parameter*

Les conditions de fonctionnement sont des points fixes (Listing C.6). Dans Asimov, elles sont décrites par une classe plus générale, *parameter*, qui consiste en une *liste de synonymes* pour les besoins de reconnaissance, une valeur fixe (*value*) et un *identificateur d'unité*. *Parameter* est une classe de base pour plusieurs autres classes.

```
class parameter
{
public:

protected:
 list<String> synonym_list;
 double value;
 token unit_token;
};
```

Listing C.6 Code C++ *parameter* (*condition*)

C.2.4 Classe *specs*

Ayant maintenant défini ses composantes, nous pouvons examiner la classe *specs* elle-même (Listing C.7). Cette classe consiste en trois listes qui correspondent aux catégories de spécification détaillées ci-dessus (contraintes, coûts et conditions). Deux méthodes agissent sur ces données.

```
class specs
{
public:
 bool constraint_to_cost ();
 bool constraints_met ();

private:
 list<constraint*> constraint_list;
 list<cost*> cost_list;
 list<parameter*> condition_list;
};
```

Listing C.7 Code C++ *specs*

C.2.4.1 Méthode *constraints_met*

C'est une méthode qui itère à travers la liste des contraintes et ne retourne *vrai* que si toutes les contraintes sont respectées (Listing C.8).

```
bool specs::constraints_met ()
{
 list<constraint*>::iterator ctl_it;
 constraint* cttmp;
 bool met_flag = TRUE;

 for ( ctl_it = constraint_list.begin (); ctl_it != constraint_list.end (); ++ctl_it )
 {
  cttmp = *ctl_it;
  if ( !cttmp->met () )
  {
   met_flag = FALSE;
  }
 }
 return ( met_flag );
}
```

Listing C.8 Méthode C++ *specs::constraints_met*

C.2.4.2 Méthode *constraint_to_cost*

Cette méthode (Listing C.9) est utilisée par le bloc de gestion de conception quand le bloc de sélection de topologie envoie un message d'erreur. Elle enlève de sa liste la contrainte de moindre priorité présentant la fonction d'erreur la plus élevée. Elle la place ensuite dans la liste des coûts avec la direction appropriée. Ceci relâche la contrainte. Cette méthode ne doit jamais retourner *faux*, car la dernière application valable est celle où toutes les contraintes sont relâchées, et le bloc de dimensionnement doit réussir.

```
constraint* specs::constraint_to_cost ()
{
 list<constraint*>::iterator ctl_it;
 constraint* cttmp;
 constraint* ctrem;
 unsigned int ctp = 0;
 bool successful_transformation = TRUE;
 token direction_token;
 cost* cstmp = new cost;
 double lowest_value = DBL_MAX;

 if ( !constraint_list.empty () )
 {
  for ( ctl_it = constraint_list.begin (); ctl_it != constraint_list.end (); ++ctl_it )
  {
   cttmp = *ctl_it;
   if ( cttmp->read_priority () > ctp ||
        ( cttmp->read_priority () == ctp &&
          cttmp->read_performance_ptr()->read_value () < lowest_value ) )
   {
    // convert the least constraint, both in terms of priority and
    // in terms of value
    ctp = cttmp->read_priority ();
    lowest_value = cttmp->read_performance_ptr()->read_value();
    ctrem = cttmp;
   }
  }
  constraint_list.remove ( ctrem );

  if ( ctrem->read_relation_token () == CT_GREATER_THAN )
  {
   direction_token = CS_MAXIMIZE;
  }
  else
  {
   direction_token = CS_MINIMIZE;
  }
  cstmp->write_cost ( ctrem->read_performance_ptr(), ctrem->read_equation_token (),
                      direction_token, ctrem->read_value () );
  cost_list.push_back ( cstmp );
 }
 else
 {
  successful_transformation = FALSE;
 }

 return ( ctrem );
}
```

Listing C.9 Méthode C++ *specs::constraint_to_cost*

C.3 Classe *design*

Cette classe est un cadre par lequel le concepteur saisit ses connaissances afin de décrire une topologie donnée (Listing C.10). Il est important de noter que des modèles analytiques, c'est-à-dire les topologies, peuvent être intégrés sans grande difficulté. Cette classe comporte deux identificateurs de nom et des listes de dimensions, d'équations, de performances et de règles.

```
class design
{
public:

private:
 token design_token;
 token category_token;
 list<dimension*>* dimension_list;
 list<equation*>* equation_list;
 list<performance*>* performance_list;
 list<rule*>* rule_list;
};
```

Listing C.10 Code C++ *design*

C.3.1 Identificateurs de nom

Ceux-ci constituent un identificateur composite qui dénote la catégorie du circuit (par exemple, "cellule mémoire de courant") et la topologie unique (par exemple, "S^2I").

C.3.2 Liste de dimensions

Le jeu (ou liste) de dimensions décrit la valeur individuelle des composants de la topologie (Listing C.11). Les dimensions sont indépendantes les unes des autres, ce qui signifie qu'un appariement des paramètres (par exemple, tous les longueurs des transistors doivent être égales), doit être codé comme une topologie différente.

```
class dimension : public parameter
{
public:
 bool check_constraint ();

private:
 double lower_limit, upper_limit;
};
```

Listing C.11 Code C++ *dimension*

Une dimension hérite de la classe *parameter* (voir section C.2.3), tout en ajoutant des limites *supérieure* et *inférieure* aux valeurs qu'elle peut prendre. Ainsi, l'utilisateur peut restreindre l'espace de conception à des valeurs réalistes. La vérification du respect de ces limites s'effectue par la méthode *check_constraint* (Listing C.12).

```
bool dimension::check_constraint ()
{
 if ( value >= lower_limit && value <= upper_limit )
 {
  return ( TRUE );
 }
 else
 {
  return ( FALSE );
 }
}
```

Listing C.12 Méthode C++ *dimension::check_constraint*

C.3.3 Liste d'équations

C'est un jeu d'équations analytiques pour l'évaluation des performances (Listing C.13). L'évaluation par la simulation numérique dans un outil de synthèse est coûteuse en termes de temps de calcul. Ceci est d'autant plus vrai dans le cas des circuits à courants commutés, d'où le choix de l'évaluation par des expressions analytiques.

```
class equation
{
public:
 double execute ( list<dimension*>, list<performance*>, specs* );

private:
 token equation_token;

public:
 double (*equation_ptr) ( list<dimension*>, list<performance*>, specs* );
};
```

Listing C.13 Code C++ *equation*

En termes d'informatique, la classe *equation* consiste en un identificateur unique (*equation_token*) et un pointeur (*equation_ptr*) vers une fonction globale qui accepte comme arguments les listes des dimensions et des performances, ainsi que les spécifications. Quand la méthode *execute* est invoquée, la valeur de l'équation est retournée. La liste des performances est modifiée implicitement dans la fonction appelée (Listing C.14).

```
double equation::execute ( list<dimension*> dl, list<performance*> pl, specs* s )
{
 return ( equation_ptr ( dl, pl, s ) );
}
```

Listing C.14 Méthode C++ *equation::execute*

C.3.4 Liste des performances

Le jeu des performances représente simplement une zone où les résultats des évaluations analytiques peuvent être conservés (Listing C.15). Un objet de cette classe est un *parameter* avec un identificateur supplémentaire (*equation_token*) pour rechercher l'équation pertinente.

```
class performance : public parameter
{
public:

private:
 token equation_token;
};
```

Listing C.15 Code C++ *performance*

C.3.5 Liste des règles d'heuristiques

Une heuristique est définie comme une aide à la découverte, ou une procédure qui vise à réduire l'effort de résolution d'un problème. Plutôt que laisser l'optimiseur procéder à une recherche exhaustive de minimum, les règles heuristiques présentent un jeu réduit de directions de recherche prometteuses et guident ainsi l'algorithme dans l'espace de conception. Les heuristiques sont basées sur l'expérience et reflètent l'approche du concepteur dans le dimensionnement de la cellule.

Une règle consiste donc en un identificateur pour reconnaître l'équation concernée (*equation_token*), et une liste d'heuristiques (*heuristic_list*) (Listing C.16).

```
class rule
{
public:

private:
 token equation_token;
 list<heuristic*> heuristic_list;
};
```

Listing C.16 Code C++ *rule*

L'heuristique (Listing C.17) contient un pointeur vers une dimension donnée (*dimension_ptr*) et un *identificateur de direction*, qui indique si la valeur de la dimension doit être augmentée ou diminuée .

```
class heuristic
{
public:

private:
 dimension* dimension_ptr;
 token direction_token;
};
```

Listing C.17 Code C++ *heuristic*

C.4 Classe *design manager*

Le bloc de gestion de conception (Listing C.18) a deux tâches, formalisées par deux méthodes.

```
class manager
{
public:
 void manage_loop ( specs*, specs*, bool* );
 void output_to_user ( design* );

private:
};
```

Listing C.18 C++ code *design manager*

C.4.1 Méthode *manage_loop*

Les arguments sont des pointeurs vers les spécifications interne et externe (*espp*, *ispp*), et un pointeur vers le drapeau global *run*. Ce dernier, comme indiqué précédemment, constitue un signal de sélection qui indique si l'invocation actuelle est une première ou non. L'action de la méthode *manage_loop* (Listing C.19) est décidée par l'état du drapeau : soit une copie simple des spécifications externes vers les spécifications internes est effectuée, soit une modification des spécifications internes par une conversion *constraint_to_cost*.

```
void manager::manage_loop ( specs* espp, specs* ispp, bool* run )
{
 if ( !(*run) )
 {
  // start
  (*ispp).write_specs ( espp->read_constraint_list (), espp->read_cost_list (),
                        espp->read_condition_list () );
  *run = TRUE;
 }
 else
 {
  // modify specs
  (*ispp).constraint_to_cost ();
 }
}
```

Listing C.19 Méthode C++ *design manager::manage_loop*

C.4.2 Méthode *output_to_user*

Cette méthode est invoquée quand le bloc de dimensionnement réussit. La cellule dimensionnée (*dp*) est sortie, et Asimov termine (Listing C.20).

```
void manager::output_to_user ( design* dp )
{
 dp->short_print ();
 cout << "Session ended" << endl;
 exit ( 0 );
}
```

Listing C.20 Méthode C++ *design manager::output_to_user*

C.5 Classe *topology selector*

Cette classe contient une liste de cellules réalisables et un itérateur qui suit et sélectionne la cellule actuelle (Listing C.21). Deux méthodes appartiennent à cette classe.

```
class topology
{
public:
 void reset ();
 bool select ( specs*, design* );

private:
 list<design*> design_list;
 int iterator;
};
```

Listing C.21 Code C++ *topology selector*

C.5.1 Méthode *reset*

Afin de remettre le bloc de sélection de topologie à zero (quand la liste des topologies est épuisée, ce qui engendre l'envoi d'un message d'échec au bloc de gestion de conception), l'itérateur est remis à zero (Listing C.22).

```
void topology::reset ()
{
 iterator = 0;
}
```

Listing C.22 Méthode C++ *topology selector::reset*

C.5.2 Méthode *select*

La méthode *select* (Listing C.23) requiert les spécifications internes (*spp*) et un pointeur vers la cellule courante (*dp*) afin de sélectionner une topologie. Premièrement, elle vérifie la position de l'itérateur dans la liste. S'il se trouve à la fin, la méthode retourne une booléenne *fausse* et le contrôle retourne au bloc de gestion de conception. Si ce n'est pas le cas, la cellule suivante dans la liste est attribuée à la cellule courante. Le dimensionnement initial est invoqué à partir du bloc de dimensionnement. Si ce dimensionnement échoue, le bloc retourne immédiatement un message d'échec au bloc de sélection de topologie, ce qui provoque le choix de la cellule suivante dans la liste.

```
bool topology::select ( specs* spp, design* dp )
{
list<design*>::iterator design_list_iterator;
unsigned int i;
bool topos_left = TRUE;

if ( iterator >= design_list.size () )
{
 topos_left = FALSE;
}

if ( topos_left )
{
 for ( i = 0; i < iterator; i++ )
 {
  design_list_iterator++;
 }
 *dp = **design_list_iterator;
 iterator++;
}

return ( topos_left );
}
```

Listing C.23 Méthode C++ *topology selector::select*

C.6 Classe *sizing module*

Le bloc de dimensionnement est un sous-système, décrit dans la section 5.2 et de nouveau schématisé dans la Figure C.2. Le code implémentant ce bloc est donné dans le Listing C.24.

```
class sizer
{
public:
 bool size ( design*, specs* );

private:
 optimizer* optimizer_ptr;
 feasibility* feasibility_ptr;
 analyzer* analyzer_ptr;
 bool success;
 bool run;
};
```

Listing C.24 Code C++ bloc de dimensionnement

En fait, le processus de dimensionnement (la méthode *size*, Listing C.25) commence par le test de faisabilité (*feasibility*), puisque la cellule est déjà dimensionnée à ses valeurs de point de départ dans sa fonction d'initialisation. Si aucun problème physique ou de polarisation n'est décelé, l'évaluation des performances par *analyser_ptr* peut être effectuée. Si un problème est détectée, par contre, cette évaluation n'est pas effectuée, car les problèmes fonctionnels doivent être éliminés avant. L'*optimiseur*, comme contrôleur du processus, génère un nouveau vecteur de paramètres de conception, en se basant sur l'état de la faisabilité et sur la valeur de la fonction objectif.

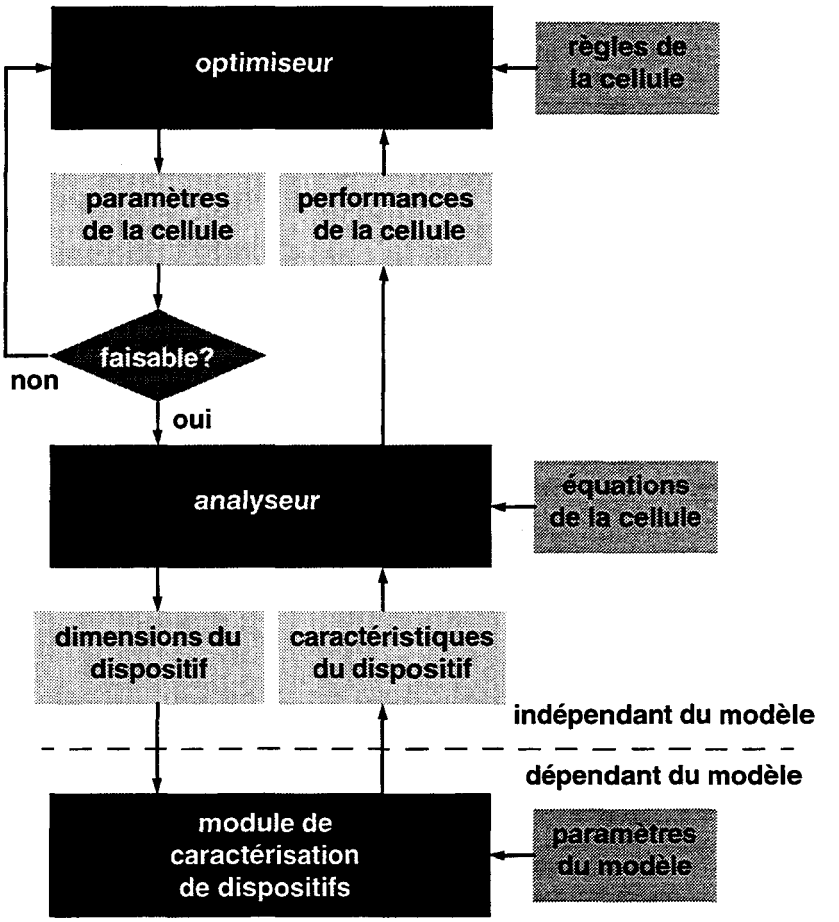


Figure C.2 Architecture du module de dimensionnement

Cette dernière est calculée par l'évaluation des performances et des contraintes spécifiées. L'optimiseur a le droit d'interrompre le processus de dimensionnement en désactivant le drapeau *run*, et aussi de modifier l'état du drapeau de réussite (*success*), qui est retourné à la boucle principale.

```
bool sizer::size ( design* dp, specs* sp )
{
 optimizer_ptr->write_optimizer ( dp, sp, feasibility_ptr, &success, &run );
 analyzer_ptr->write_analyzer ( dp, sp );

 do
 {
  if ( feasibility_ptr->ok ( dp, sp ) )
  {
   analyzer_ptr->evaluate ();
  }
  optimizer_ptr->next_move ();
  optimizer_ptr->print_stats ();
 } while ( run );

 return ( success );
}
```

Listing C.25 Méthode C++ *sizer::size*

C.7 Classe *feasibility*

La classe de faisabilité (Listing C.26) contient des drapeaux booléens, qui conservent les résultats des méthodes qui représentent les échecs dûs aux tailles des composants (*physical*), au comportement électrique pour un courant d'entrée nominal (*dc_nom*) et au comportement électrique pour des courants d'entrée aux extrêmes (*dc_max* et *dc_min*). Il existe également un certain nombre de méthodes pour manipuler ces données.

```
class feasibility
{
public:
 void reset ();
 bool ok ( design*, specs* );
 void physical_analysis ( design* );
 void dc_analysis ( design*, specs* );

private:
 bool physical;
 bool dc_nom;
 bool dc_max;
 bool dc_min;
};
```

Listing C.26 Code C++ *feasibility*

C.7.1 Méthode *OK*

La méthode ok (Listing C.27) est le regroupement de tous les drapeaux. Un seul échec suffit pour signifier un échec de faisabilité.

```
bool feasibility::ok ( design* d, specs* s )
{
 physical_analysis ( d );
 dc_analysis ( d, s );

 return ( read_physical () && read_dc_nom () && read_dc_max () && read_dc_min () );
}
```

Listing C.27 Méthode C++ *feasibility::ok*

C.7.2 Méthode *physical_analysis*

La méthode d'analyse physique (Listing C.28) vérifie l'état des contraintes de toute dimension. Ceci s'effectue par utilisation de la méthode *check_constraint*, appartenant à la classe *dimension*.

C.7.3 Méthode *dc_analysis*

La méthode d'analyse DC (Listing C.29) invoque la cellule afin de déterminer si elle peut fonctionner sous les conditions données. Pour cela elle nécessite une fonction d'analyse DC, qui s'identifie par E_DC_ANALYSIS dans le modèle. Elle nécessite

```
void feasibility::physical_analysis ( design* d )
{
 list<dimension*> dl = *(d->read_dimension_list ());
 list<dimension*>::iterator dit;
 dimension* dtmp;

 physical = TRUE;

 for ( dit = dl.begin (); ( dit != dl.end () ) && ( physical != FALSE ); ++dit )
 {
  dtmp = *dit;
  if ( ! ( dtmp->check_constraint () ) )
  {
   physical = FALSE;
  }
 }
}
```

Listing C.28 Méthode C++ *feasibility::physical_analysis*

également des fonctions d'analyse statique individuelles pour des courants d'entrée de valeur nominale, maximale ou minimale, identifiées par E_DC_NOM, E_DC_MAX et E_DC_MIN, respectivement.

```
void feasibility::dc_analysis ( design* d, specs* s )
{
 list<equation*> el = *(d->read_equation_list ());
 list<equation*>::iterator eit;
 equation* etmp;

 for ( eit = el.begin (); ( eit != el.end () ) && ( etmp->read_equation_token () != E_DC_ANALYSIS ); ++eit )
 {
  etmp = *eit;
  if ( etmp->read_equation_token () == E_DC_ANALYSIS )
  {
   etmp->execute ( *(d->read_dimension_list ()), *(d->read_performance_list ()), s );
  }
 }
 for ( eit = el.begin (); eit != el.end (); ++eit )
 {
  etmp = *eit;
  if ( etmp->read_equation_token () == E_DC_NOM )
  {
   dc_nom = bool ( etmp->execute ( *(d->read_dimension_list ()), *(d->read_performance_list ()), s ) );
  }
  if ( etmp->read_equation_token () == E_DC_MAX )
  {
   dc_max = bool ( etmp->execute ( *(d->read_dimension_list ()), *(d->read_performance_list ()), s ) );
  }
  if ( etmp->read_equation_token () == E_DC_MIN )
  {
   dc_min = bool ( etmp->execute ( *(d->read_dimension_list ()), *(d->read_performance_list ()), s ) );
  }
 }
}
```

Listing C.29 Méthode C++ *feasibility analyser::dc_analysis*

C.8 Classe *analyser*

La classe analyser nécessite également des informations provenant en grande partie de la cellule concernée (Listing C.30). Dans la méthode *evaluate* (Listing C.31), les listes des dimensions et des performances contenues par la classe *design* sont transférées, avec les *spécifications* (seule la liste des conditions est nécessaire), aux méthodes *execute* de tous les pointeurs d'équation de la cellule. La liste des performances est mise à jour implicitement dans la fonction réalisant l'équation.

```
class analyser
{
public:
 void evaluate ();

private:
 design* design_ptr;
 specs* specs_ptr;
};
```

Listing C.30 Code C++ *analyser*

```
void analyser::evaluate ()
{
 list<equation*> el = *(design_ptr->read_equation_list ());
 list<equation*>::iterator eit;
 equation* etmp;

 list<dimension*> dl = *(design_ptr->read_dimension_list ());
 list<dimension*>::iterator dit;

 list<performance*> pl = *(design_ptr->read_performance_list ());
 list<performance*>::iterator pit;

 for ( eit = el.begin (); eit != el.end (); ++eit )
 // go through all equations
 {
  etmp = *eit;
  // evaluate equation
  double v = etmp->execute ( dl, pl, specs_ptr );
 }
}
```

Listing C.31 Méthode C++ *analyser::evaluate*

C.9 Classe *optimizer*

L'optimiseur a de nombreuses variables composantes dans sa structure (Listing C.32). L'enchaînement des décisions est assez complexe et se représente par un organigramme (Figure C.3), qui fait appel aux concepts développés dans la section 5.2.

```
class optimizer
{
public:
 void reset ();

 void next_move ();

private:
 design* design_ptr;                    // pointer to current design object
 specs* specs_ptr;                      // pointer to current specs object
 feasibility* feasibility_ptr;          // pointer to feasibility object
 bool* success_flag;                    // pointer to external flag
 bool* run_flag;                        // pointer to external flag

 rule* rp;                              // pointer to current rule object
 dimension* dmp;                        // pointer to changing dimension

 double prev_dim_value;                 // previous dimension value

 unsigned int iterations;               // number of iterations
 unsigned int local_iterations;         // number of iterations in local loop
 unsigned int last_success;             // record of last successful move

 list<heuristic*> hl;                   // current heuristic list
 list<heuristic*>::iterator hit;        // current heuristic list iterator
 heuristic* hptr;                       // pointer to current heuristic

 double largest_error;                  // largest error used to evaluate
                                        // which rule to apply
 double sum_of_errors;                  // used to determine progress
 double prev_sum_of_errors;
 double lowest_sum_of_errors;

 bool iteration_flag;                   // true when iterations >= MAX
 bool no_progress_flag;                 // true when iterations >= last_success
                                        //+ IT_NO_PROGRESS

 token equation_token;                  // compare with previous to see if
 token prev_equation_token;             // largest error still the same

 int sign;                              // +1 when heuristic increase
                                        // -1 when heuristic decrease

 double step;                           // step size for dimension variation
 double step_factor;                    // convergence

 void initial_vector ();                // apply pointer to design setup
 void next_vector ();                   // invoked when no feasibility problems
 void previous_failed ();               // go back and try again
 void fix_dc ();                        // invoke dc heuristics
 bool converged ();                     // test for convergence
 void get_rule ( token );               // get rule corresponding to equation
 void write_vector ();                  // change the dimension and do it
 void divide_step ();                   // reduce step size
 void move_hptr ();                     // change heuristic
 void extract_largest_error ();         // determine largest error
 void change_rule ();                   // start at beginning of rule heur list
};
```

Listing C.32 Code C++ *optimizer*

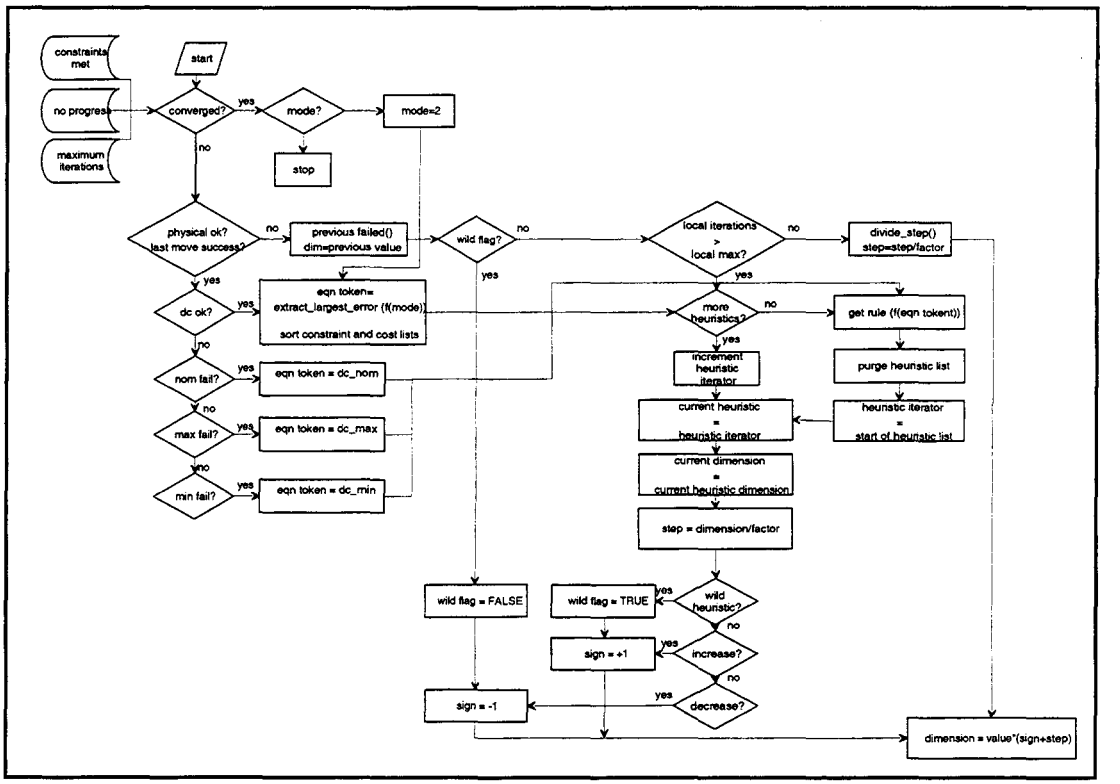


Figure C.3 Organigramme *optimizer*