THÈSE DE DOCTORAT DE L'UNIVERSITÉ PARIS VI

Spécialité Informatique

Présentée par M. Pascal REMY

Pour obtenir le titre de

DOCTEUR DE L'UNIVERSITÉ PARIS VI

OUTILS DE VÉRIFICATION POUR CIRCUITS VLSI ASGA MESFET PAR DES MÉTHODES D'ABSTRACTION FONCTIONNELLE

Soutenue le 15 décembre 1997, devant le jury composé de

M. Alain GUYOT	Rapporteur
M. Jean-Louis NOULLET	Rapporteur
M. Francis DEVOS	Président du jury
M. Alain GREINER	Directeur de thèse
M. Yves NISSIM	Examinateur
M. Serge PRAVOSSOUDOVITCH	Examinateur

À mon père, ma mère et mon frère,
avec toute mon affection.

Remerciements

Je tiens à remercier les membres de l'équipe MASI, récemment devenu ASIM pour leur sympathie et pour la bonne ambiance de travail qu'ils font régner au laboratoire. Nombreux sont ceux qui m'ont témoigné leur amitié et leur soutien et je leur en suis reconnaissant.

Je remercie en particulier Alain GREINER, professeur au *LIP6*, qui m'a permis d'effectuer ce travail au sein de son équipe ainsi que pour son soutien en tant que directeur de thèse.

J'adresse également ma gratitude pour l'intérêt qu'ils ont accordé à mon travail et pour leurs remarques constructives aux membres du jury : Alain GUYOT du *TIMA*, Jean-Louis NOULLET de l'*INSA Toulouse*, Francis DEVOS professeur à l'*IEF*, Yves NISSIM du *CNET* et Serge PRAVOSSOUDOVITCH professeur au *LIRMM*.

J'adresse enfin des remerciements à mes proches et à mes amis auprès desquels, bien souvent, j'ai pu me ressourcer et trouver l'énergie qui m'a permis de poursuivre et de faire aboutir ce travail.

Résumé

Dans le cadre de cette thèse, un environnement de vérification pour la circuiterie AsGa MESFET a été développé. Il s'appuie sur une représentation en étages orientés du circuit obtenue par abstraction fonctionnelle à partir de sa représentation en transistors. La construction de cette représentation fait appel à des méthodes purement algorithmiques ainsi qu'à des méthodes basées sur la reconnaissance de formes.

Cette représentation constitue le point de départ pour les outils de vérification développés dans le cadre de la thèse. Les vérifications sont d'ordre électrique, fonctionnelle et temporelle. À la différence de la circuiterie CMOS, la circuiterie AsGa MESFET est très sensible aux erreurs de dimensionnement des transistors qui peuvent compromettre la fonctionnalité. L'outil pour la vérification électrique signale automatiquement les violations de règles électriques parmi un ensemble préétabli. Il est paramétrable et évolutif. L'outil pour la vérification fonctionnelle fournit le modèle VHDL à partir de la *net-list* en transistors extraite du dessin des masques. Enfin, l'outil pour la vérification temporelle définit des modèles temporels pour chaque étage orienté. Le chemin critique et les temps de propagation du circuit sont obtenus à partir de ces modèles.

Cet environnement a été à la base de la validation des circuits AsGa MESFET conçus au laboratoire MASI au cours de ces dernières années. Les expériences réalisées sur des circuits de taille variable démontrent la complexité linéaire des algorithmes mis en œuvre. Le bilan montre que la méthode proposée permet de traiter des circuits variés et de complexité significative.

Mots clés

Circuits intégrés sur Arséniure de Gallium
Circuits E/D MESFET DCFL
Abstraction fonctionnelle
Vérification électrique
Vérification fonctionnelle
Vérification temporelle
Reconnaissance de formes

Abstract

As part of this PhD, a verification environment for GaAs MESFET circuits has been developed. It is based on an oriented gates representation of the circuit obtained through functional abstraction of its transistor net-list representation. To obtain this representation, purely algorithmic methods but also pattern recognition methods have been used.

This representation is the starting point for the verification tools also developed as part of this PhD. The verifications involved are of electrical, functional and timing nature. Unlike CMOS circuits, GaAs MESFET circuits are very sensitive to transistor sizing errors that can cause malfunctions. The electrical verification tool automatically indicates any electrical rule violation amongst a pre-established set of rules. It uses parameters that can be modified and is upgradable. The functional verification tool supplies a VHDL model from the mask extracted transistor net-list. Finally the timing verification tool defines timing models for each oriented gate. The circuit's critical path as well as it's propagation delays are then obtained from these models.

This environment has been used to validate the GaAs MESFET circuits that have been devised at MASI laboratory these past years. Experimentation accomplished on circuits of various sizes demonstrates the linear complexity of the algorithms used. The overall results show that the adopted method enables us to deal with circuits of various sizes and of significant complexity.

Key words

Gallium Arsenide integrated circuits
DCFL E/D MESFET circuits
Functional abstraction
Electrical verification
Functional verification
Timing verification
Pattern recognition

Table des matières

Table des illustrations

1. Introduction

Nous présentons dans cette thèse un environnement logiciel pour la vérification de la circuiterie VLSI AsGa MESFET. L'environnement s'articule autour d'une technique d'abstraction permettant de reconstruire une *net-list* en portes orientées obtenue à partir de la *net-list* en transistors du circuit à traiter. Il propose un ensemble d'outils avec lesquels il est possible d'effectuer des vérifications électrique, fonctionnelle et temporelle.

Le premier chapitre présente la circuiterie et la famille logique AsGa que nous ciblons dans notre travail et explique ce choix. Il donne ensuite les outils d'aide à la conception qui sont nécessaires ou souhaitables afin de garantir le bon fonctionnement de tels circuits.

Dans l'état de l'art, nous présentons les bases sur lesquelles l'environnement de vérification que nous présentons s'appuie.

Le troisième chapitre expose les principes sur lesquels les divers éléments de l'environnement de vérification sont fondés. D'une part pour les éléments de la phase d'abstraction, d'autre part pour les éléments des phases de vérifications.

Le chapitre 5 explique le fonctionnement des modules intervenant dans la phase d'abstraction. On verra à cette occasion que quatre étapes sont nécessaires.

De même, le chapitre 6 vise à expliquer les modules de vérification électrique, fonctionnelle et temporelle.

Le chapitre 7 est consacré aux résultats obtenus au moyen de cet environnement de vérification ainsi qu'à ces performances.

Enfin le chapitre 8 donne les évolutions souhaitables en vue d'optimiser certains aspects de l'environnement.

2. Problématique

2.1 Introduction

Le succès des technologies CMOS pour la réalisation de circuits VLSI a généré une profusion d'outils CAO, tant pour la conception que pour la vérification. Cela n'est pas le cas en ce qui concerne les circuits Arséniure de Gallium. Cette technologie comporte pourtant de nombreux avantages tels que la rapidité et la résistance aux radiations [Har93]. Dans cette thèse, nous avons focalisé notre attention sur les problèmes de vérification : outre les habituelles vérifications fonctionnelle et temporelle, la technologie AsGa impose des vérifications d'ordre électrique.

2.2 Choix technologiques

Bien que de nombreuses technologies Arséniure de Gallium soient disponibles, seul un nombre limité d'entre elles permet d'atteindre des densités de l'ordre de la centaine de milliers de transistors, c'est-à-dire des densités VLSI.

Dans son travail de thèse P. Royannez [Roy96] analyse la technologie proposée par Vitesse [Vit93]. Celle-ci est de type E/D MESFET (Enriched/Depleted MEtal Semiconductor Field Effect Transistor). Elle propose le transistor enrichi et le transistor déplété ainsi que la diode Schottky. Cette technologie permet la conception de circuits numériques au moyen de diverses familles logiques : DCFL (Direct Coupled FET Logic), BFL (Buffered FET Logic), SDFL (Schottky Diode FET Logic), SCFL (Source Coupled FET Logic). Chacune de ces familles possède ses avantages et ses inconvénients et les critères principaux par rapport auxquels un choix a été établi sont : la consommation statique par étage, le nombre de transistors nécessaires pour la réalisation d'une fonction logique donnée, et l'immunité au bruit. À partir de ces éléments, ce choix s'est porté sur la logique DCFL, qui offre le meilleur compromis puissance/surface/délais et qui se trouve être une des plus simples. Cette famille se caractérise cependant par ses faibles marges aux bruits, sa sensibilité élevée à la charge en sortie et sa faible tolérance aux erreurs de procédé. Elle n'emploie qu'exceptionnellement la diode Schottky et utilise deux tensions d'alimentation. Nous appellerons Vdd la tension d'alimentation fournissant le niveau logique haut ou "1" et Gnd celle fournissant le niveau logique bas ou "0". Cette famille logique possède une forte ressemblance avec la logique NMOS : la porte de base est l'étage NOR.

2.3 Vérification de circuits VLSI à base d'Arséniure de Gallium

Nous rappelons ci-dessous quelles sont les différentes vérifications qui doivent être effectuées sur un circuit AsGa DCFL afin d'en garantir le fonctionnement. Nous limitons notre étude aux circuits numériques.

2.3.1 Vérification fonctionnelle

Son rôle est de vérifier que la fonctionnalité du circuit correspond à celle des spécifications. En d'autres termes, il s'agit de s'assurer que la fonction logique réalisée par le circuit correspond à celle spécifiée. Pour cela, il faut abstraire une description comportementale de niveau RTL, à partir de la *net-list* en transistors extraite du dessin des masques.

De tels outils d'abstraction fonctionnelle existent pour les technologies CMOS, citons REX ([Neb86], [Neb87]), DESA et FAON ([Mur88], [Dev92] et [Mu88b]), VERA ([Kos89], [Kos91]), LOGEX ([Boe88]), CCC ([Ste92]), et enfin DESB ([Lau94]). Nous avons étudié leur extension à la circuiterie AsGa MESFET.

2.3.2 Vérification temporelle

La vérification temporelle est nécessaire afin de s'assurer que les performances temporelles d'un circuit sont conformes aux spécifications. Cela revient à analyser les temps de propagation des signaux entre les points caractéristiques du circuit. Les différentes paires de points concernées peuvent être :

- une entrée et une sortie du circuit,
- une entrée et un point mémorisant du circuit,
- deux points mémorisants du circuit,
- un point mémorisant et une sortie du circuit.

Ici encore, le point de départ est la *net-list* de transistors et capacités extraite du dessin des masques. La difficulté ici est de transformer un schéma en transistors non-orientés, en un schéma en portes orientées : l'analyse temporelle statique (c'est-à-dire indépendante des stimuli) suppose en effet que l'on connaisse le sens de propagation des signaux.

2.3.3 Vérification électrique

La circuiterie AsGa MESFET est particulièrement sensible aux erreurs de dimensionnement des transistors. Contrairement au CMOS ou une erreur de dimensionnement se traduit généralement par une dégradation des caractéristiques temporelles, en AsGa ce type d'erreur a pour conséquence le disfonctionnement des portes. En effet il est à l'origine de sur ou sous-polarisation, ou de la non-réalisation d'états logiques corrects.

Les portes de puissance permettent de régénérer un signal et/ou d'attaquer plusieurs portes avec un même signal. Les marges d'erreur sur les niveaux électriques des signaux étant très faibles, il faut s'assurer que le contexte dans lequel se trouve chaque porte de puissance est correct. Ici aussi, le point de départ de l'outil de vérification des règles électriques (ERC) est la *net-list* en transistors extraite du dessin des masques. Des règles de dimensionnement différentes d'un type de porte à un autre sont à vérifier. Il est donc indispensable de reconstruire un schéma en portes et d'identifier le type de chaque porte pour effectuer les vérifications.

2.4 Conclusion

Les différentes vérifications visées (fonctionnelle, temporelle, électrique) supposent la reconstruction d'une *net-list* de portes orientées et typées à partir d'une *net-list* de transistors non-orientée. Cette phase d'abstraction constitue donc la difficulté centrale à résoudre.

3. État de l'art

3.1 Introduction

Des environnements de conception et de vérification permettant de réaliser des circuits AsGa DCFL sont utilisés dans l'industrie et dans certaines universités. Citons notamment le kit AsGa de la chaîne de CAO Compass, la paramétrisation de la chaîne de CAO OPUS de Cadence pour l'AsGa ainsi que la chaîne de CAO Magic de Berkeley [Lon90]. Cependant, il n'existe pas, à notre connaissance, d'outils d'abstraction fonctionnelle pour la circuiterie AsGa MESFET. Dans tous les cas, ces chaînes, initialement prévues pour des technologies à base de Silicium, ont effectué une migration vers l'AsGa par paramétrisation et/ou ajout de composants logiciels. En ce qui nous concerne, nous nous appuierons sur la chaîne de CAO CMOS Alliance [Gre92].

Un des outils logiciels de la chaîne de CAO Alliance permet la transformation à des fins de vérification d'un réseau de transistors MOS en un réseau de portes orientées. C'est la transposition à la circuiterie AsGa DCFL qui va retenir notre attention dans ce qui suit. Nous commencerons donc par décrire l'outil DESB permettant l'abstraction des circuits CMOS. Puis, nous verrons dans quelle mesure, une telle approche peut être utilisée pour les circuits AsGa DCFL et comment y parvenir.

La deuxième partie de ce chapitre est consacrée aux aspects électriques de la circuiterie AsGa DCFL. Sont exposés, plus particulièrement, les problèmes électriques que la conception de tels circuits peut engendrer. De cette analyse, une liste de règles électriques à vérifier a été établie afin de prévenir autant que possible ces problèmes.

3.2 DESB

Le désassembleur DESB a été développé dans le cadre de la thèse de M. Laurentin [Lau94]. Son rôle est de fournir une description en portes orientées à partir de la description en transistors. La caractéristique essentielle de cet outil est qu'il parvient à ce résultat de manière algorithmique, sans faire appel à la reconnaissance de forme. Cet outil est basé, d'une part, sur la définition d'un objet nommé cône qui représente une porte orientée. Il est basé, d'autre part, sur la définition d'une méthode d'abstraction. Les deux sections qui suivent présentent respectivement chacun de ces aspects.

3.2.1 Modèle des cônes

Soit un circuit représenté sous forme d'une *net-list* de transistors. Y figurent des connecteurs, des transistors et les interconnexions de ces éléments entre eux. Chaque interconnexion correspond à une équipotentielle ou signal. La construction de la représentation en "cônes" CMOS est fondée sur les définitions suivantes :

1. tout signal G sur lequel se trouve la grille d'au moins un transistor forme la sortie d'un cône dit cône G,
2. tous les transistors au travers desquels peuvent s'établir des chemins électriques entre G et tout signal d'alimentation appartiennent au cône G,
3. tout ensemble de transistors par lesquels un tel chemin électrique peut s'établir forme une branche,
4. les signaux attaquant les grilles des transistors appartenant au cône G forment ses entrées.

Soulignons que les définitions 1 et 4 ne sont appliquées qu'à des transistors *commutants*, à savoir ceux dont les trois équipotentielles de connecteur sont distinctes. Remarquons par ailleurs qu'un chemin électrique ne peut s'établir au travers d'un transistor bloqué. Par conséquent les transistors bloqués n'apparaissent dans aucune branche et sont signalés.

La différence essentielle entre des *net-lists* de transistors CMOS et AsGa DCFL est que des transistors CMOS et AsGa considérés comme bloqués n'ont pas la même configuration au niveau de leur connectique. Ainsi, par exemple, un transistor CMOS en configuration "diode" (grille et drain court-circuités) correspond à la configuration "pull-up" ou "pull-down" d'un transistor AsGa. Ce type de configuration est fréquent en AsGa DCFL. En effet, les "pull-up" sont présents dans toutes les portes NOR.

3.2.2 Méthode d'abstraction de DESB

Les cônes sont construits un à un à partir des signaux de grille des transistors du circuit. À partir du signal G qui marque la sortie d'un cône G, les chemins électriques vers les signaux d'alimentation sont recherchés. Cette recherche se fait par signaux et transistors interposés, de proche en proche, en omettant les signaux de grille. Seuls les chemins conformes aux principes suivants sont retenus :

1. un chemin ne doit pas passer plus d'une fois par un même signal,
2. un chemin ne doit pas passer par deux transistors de type différents dont les signaux de grille sont identiques,
3. un chemin ne doit pas passer par un transistor dont le signal de grille se trouve en amont ou en aval de ce même chemin.

Ainsi, une branche du cône est construite à partir de chaque chemin retenu. Enfin, les dépendances entre les cônes sont mémorisées sous la forme de deux listes, à savoir :

- la liste des cônes qu'un cône G a en entrée, c'est-à-dire ceux dont le signal de sortie attaque la grille d'au moins un des transistors contenus dans le cône G,
- la liste des cônes qui ont le cône G en entrée, c'est-à-dire ceux contenant au moins un transistor dont le signal de grille est G.

Pour la circuiterie AsGa, la première différence concerne la propagation qui ne doit plus être limitée à une propagation de source à drain et inversement. Elle doit pouvoir se faire en tout sens. Prenons un exemple : une diode AsGa DCFL (transistor dont les signaux de source et de drain sont connectés) peut avoir sa grille reliée au signal de sortie de cône.

Par ailleurs, la circuiterie AsGa DCFL ne peut avoir de chemins électriques passant par plus de deux transistors en série.

Enfin, les deuxième et troisième principes sont issus d'une caractéristique propre à la circuiterie CMOS et ne sont pas généralisables à l'AsGa.

En conséquence, le modèle des "cônes" utilisé par l'outil DESB peut être conservé. En revanche, la méthode de construction des cônes devra être modifiée pour la circuiterie AsGa DCFL.

3.3 Circuiterie AsGa DCFL

Dans son travail de thèse, P. Royannez fait une analyse de la circuiterie AsGa DCFL. À partir de celle-ci, il définit un ensemble de règles de dimensionnement qui doivent être respectées pour un fonctionnement électrique correct. Dans ce qui suit, nous commençons par quelques rappels et généralités concernant la circuiterie AsGa DCFL. Puis, nous exposons les principaux résultats de l'analyse mentionnée, suivant le type de porte rencontré. Les portes logiques DCFL, respectivement d'un point de vue statique puis dynamique sont d'abord présentées. Puis sont abordées les portes de puissance, également d'un point de vue statique puis dynamique. Enfin, nous résumons les règles qui en sont déduites.

3.3.1 Rappels et Généralités

La circuiterie DCFL emploie deux alimentations que nous avons appelé Vdd et Gnd. Elle utilise des MESFETs enrichis et déplétés. Les transistors enrichis ont une tension de seuil positive alors que celle des transistors déplétés est négative. On rappelle que la tension de seuil appliquée entre la grille et la source d'un transistor est la tension à partir de laquelle celui-ci devient passant. Le transistor enrichi est donc bloqué lorsque sa tension grille-source V_{GS} est nulle. Par contre, dans une même situation, le transistor déplété est passant. Les transistors enrichis sont principalement employés en tant qu'éléments commutants possédant trois équipotentielles distinctes au niveau de leurs connecteurs. Par contre, les transistors déplétés sont utilisés principalement comme charge active ("pull-up" ou "pull-down") avec leurs connecteurs de grille et de source reliés.

Les jonctions grille-source des transistors ne sont pas assimilables à des capacités comme pour le CMOS mais à des diodes Schottky. Il s'ensuit que la tension V_{GS} des transistors enrichis est écrêtée à la tension de seuil de la diode. Il s'agit là d'une caractéristique de la jonction métal/semi-conducteur. Il en est de même pour la jonction grille-drain et il est donc possible de réaliser une diode Schottky en reliant la source et le drain d'un transistor enrichi. Ces derniers forment alors la cathode de la diode dont la grille est l'anode.

Le dimensionnement des transistors s'effectue en intervenant sur la longueur et la largeur de leur grille. En général, la longueur de grille est minimale, et seule la largeur varie. Notons enfin qu'il n'y a pas de différence structurelle entre la source et le drain des transistors enrichis. Nous considérerons que le connecteur de source est celui qui a une tension plus basse que celle se trouvant sur le connecteur de drain.

Afin de faire apparaître les difficultés qu'engendre la conception des circuits DCFL, il est nécessaire d'en comprendre le fonctionnement. C'est ici qu'intervient la modélisation du comportement électrique des différents éléments de cette circuiterie. Les formules et les résultats qui suivent ont pour origine un modèle de Statz et al [Sta87] modifié, proposé par Vitesse [Vit93] et donné dans le tableau 3.1. Notons que nous négligerons et considérerons comme nuls les courants I_{sub}. Comme nous l'avons fait jusqu'ici, nous éviterons autant que possible toute valeur numérique. Voici une liste de définitions de termes non encore rencontrés et utiles pour la suite [Sed90] :

- les transistors enrichi et déplété sont respectivement identifiés par T_E et T_D.
- V_{TE} et V_{TD} correspondent respectivement aux tensions seuils des transistors enrichi et déplété. V_d désigne la tension seuil de la diode Schottky. Les jonctions grille-source et grille-drain des transistors forment de telles diodes.
- W_E et L_E (W_D et L_D) correspondent respectivement à la largeur et à la longueur de grille d'un transistor enrichi (déplété).
- V_{IH} est la valeur minimale de l'état haut en entrée d'étage.
- V_{IL} est la valeur maximale de l'état bas en entrée d'étage.
- V_{OH} est la valeur minimale de l'état haut en sortie d'étage.
- V_{OL} est la valeur maximale de l'état bas en sortie d'étage.
- V_{GS} et V_{DS} désigne respectivement la tension grille-source et la tension drain-source.
- V_T désigne la tension seuil de la jonction considérée.
- I_{DS} est le courant drain-source.
- I_{sub} désigne un courant de fuite sous le seuil.

Conditions	**État**	**I_{DS} en fonction de W et L**
$V_{GS} \leq V_T$	*bloqué*	$I_{DS} = I_{sub}$
$V_{GS} > V_T$ et $V_{DS} < 3/\alpha$	*linéaire*	$I_{DS} = K\cdot(W/L)\cdot(V_{GS}-V_T)^2\cdot(1+\lambda\cdot V_{DS})\cdot\{1-[1-(\alpha\cdot V_{DS})/3]^{\varepsilon}\}+I_{sub}$
$V_{GS} > V_T$ et $V_{DS} \geq 3/\alpha$	*saturé*	$I_{DS} = K\cdot(W/L)\cdot(V_{GS}-V_T)^2\cdot(1+\lambda\cdot V_{DS})+I_{sub}$

Tableau 3.1 : modèle de Statz et al modifié

3.3.2 Portes DCFL

La porte logique AsGa DCFL de base est l'inverseur. Elle est constituée d'une charge active et d'un transistor enrichi (voir figure ci-contre). La charge active correspond à un transistor déplété dont le drain est relié à Vdd et dont la grille et la source sont reliées à la sortie de la porte O. Il s'agit d'un "pull-up". Le transistor enrichi a son drain relié à O et sa source à Gnd. Il se comporte en interrupteur "fermé" laissant passer le courant lorsque sa grille est à l'état logique haut. La sortie O de la porte tend alors vers l'état logique bas. Par contre, lorsque sa grille est à l'état logique bas, il se comporte en interrupteur "ouvert" et ne laisse pas passer le courant. La sortie O de la porte tend alors vers l'état haut.

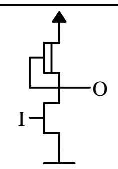


Figure 3.1 : inverseur DCFL

La porte NOR est réalisée par ajout de transistors enrichis entre O et Gnd. Leur drain est relié à O et leur source est reliée à Gnd. Le nombre de transistors enrichis correspond au nombre d'entrées du NOR. Si la grille d'au moins un de ces transistors est à l'état haut, alors la sortie O tend vers l'état logique bas. L'état logique haut est obtenu sur la sortie O lorsque toutes les entrées de la porte sont à l'état bas. C'est-à-dire lorsque tous les transistors enrichis sont bloqués.

3.3.2.1 Étude statique de l'inverseur DCFL

Afin de comprendre le fonctionnement statique des portes DCFL, plaçons-nous dans le cas le plus simple. Celui-ci consiste en un inverseur DCFL attaquant un autre inverseur DCFL (voir figure ci-contre). Commençons par donner les schémas équivalents de cette configuration lorsque l'entrée du premier inverseur est à l'état haut ($I = V_{IH}$; figure 3.3) puis à l'état bas ($I = V_{IL}$; figure 3.4) :

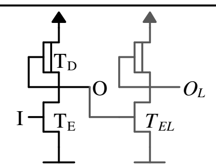


Figure 3.2 : deux inverseurs DCFL

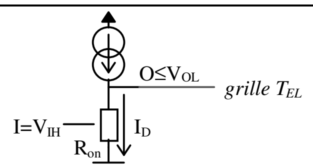


Figure 3.3 : DCFL, entrée I à l'état haut

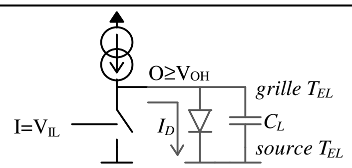


Figure 3.4 : DCFL, entrée I à l'état bas

Première constatation : afin d'obtenir un niveau bas correct en O ($O \leq V_{OL}$), il faut que la résistance équivalente du transistor déplété soit sensiblement plus forte que la résistance équivalente du transistor enrichi. En utilisant les expressions du tableau 3.1, nous trouvons la relation :

$$\beta = \frac{W_E / L_E}{W_D / L_D} \geq f(V_{IH}, V_{OL}, ...).$$

Par ailleurs, une des caractéristiques de la circuiterie DCFL est la faible différence entre V_{OH} et V_{OL} (cette différence est appelée "output logic swing") et entre V_{IH} et V_{IL} ("input logic swing"). Les marges au bruit haute et basse, respectivement définies par $NM_H = V_{OH} - V_{IH}$ et $NM_L = V_{IL} - V_{OL}$ dépendent, entre autres, de ces différences et conditionnent la stabilité d'un circuit. Plus elles sont faibles plus la tolérance aux erreurs de procédé d'un circuit est faible et plus le risque de disfonctionnement est élevé. Des marges aussi grandes que possible mais aussi équilibrées sont d'important critères qu'il faut prendre en compte lors de la conception de circuits AsGa DCFL. Les marges dépendent également du rapport $\beta = \frac{W_E / L_E}{W_D / L_D}$. C'est en observant le comportement des marges en fonction de β que l'on déduit $\beta_{optimal}$, le β offrant le meilleur compromis au niveau des marges. On obtient la relation suivante où tol est un facteur de tolérance que l'on peut prendre égal à 10% par exemple :

<u>règle 1</u> : $\beta_{optimal} - tol \leq \beta \leq \beta_{optimal} + tol$

Deuxième constatation : afin d'obtenir un niveau haut correct en O ($O \geq V_{OH}$), il faut que le courant traversant T_D, qui est en mode saturé, soit suffisant pour polariser les jonctions grille-source des transistors attaqués. D'après le modèle de Statz et al, le courant traversant T_D en mode saturé est une fonction linéaire de sa largeur W_D : $I_{pol} = a_D \cdot W_D + b_D$. Soit W_L la somme des largeurs des grilles des transistors attaquées. La charge à polariser est proportionnelle à W_L. Le courant minimal à fournir pour polariser cette charge peut lui aussi s'écrire sous la forme $I_{Lmin} = a_{Lmin} \cdot W_L + b_{Lmin}$. Comme nous cherchons à nous assurer que $I_{pol} \geq I_{Lmin}$ on obtient :

<u>règle 2</u> : $W_D \geq \frac{a_{Lmin}}{a_D} \cdot W_L + \frac{b_{Lmin} - b_D}{a_D}$

Troisième constatation : un niveau haut trop élevé en O entraîne un mauvais niveau bas en O_L. En effet, si le potentiel entre la grille et la source de T_{EL} se rapproche de V_d, alors la jonction grille-drain de T_{EL} devient passante. Cela a pour conséquence de laisser passer du courant vers O_L, conduisant à une dégradation du niveau bas en ce point. Le courant traversant T_D ne doit donc pas dépasser le courant nécessaire à la polarisation des jonctions grille-source des transistors attaqués, soit que $I_{pol} \leq I_{Lmax}$. De même que précédemment, I_{pol} et I_{Lmax} peuvent s'exprimer respectivement sous la forme d'une fonction linéaire de W_D et W_L. Nous obtenons :

$$\textbf{règle 3} : W_D \leq \frac{a_{L\max}}{a_D} \cdot W_L + \frac{b_{L\max} - b_D}{a_D}$$

3.3.2.2 Nombre d'entrées du NOR

Chaque transistor enrichi d'un NOR présente un courant de fuite I_{sub} vers Gnd. Plus le nombre d'entrées d'une telle porte augmente, plus le seuil de commutation d'une transition niveau bas vers niveau haut sur les entrées se rapproche de V_{IL}. Le nombre d'entrées d'un NOR ne devrait donc pas excéder 3 pour une utilisation courante. **règle 4**. Ce nombre pourra atteindre 5 à condition que toutes les entrées du NOR soient attaquées par des portes générant un excellent niveau bas. Les portes de puissance présentées plus loin répondent à cette condition. **règle 5**.

3.3.2.3 Le NAND

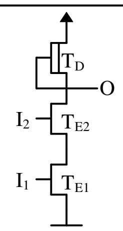


Figure 3.5 : NAND DCFL

Il est constitué d'un "pull-up" et de deux transistors enrichis en série comme le montre la figure ci-contre. Nous avons mentionné la nécessité d'optimiser les marges au bruit du fait des erreurs de procédé et du faible écart entre V_{OH} et V_{OL} et entre V_{IH} et V_{IL}. Par ailleurs la tension drain-source aux bornes du transistor enrichi T_{E1} n'est jamais nulle lorsque la grille est à l'état haut. Cela réduit la marge au bruit et rend la mise au point d'une telle porte très délicate. Nous verrons cependant un exemple de porte comprenant deux transistors enrichis en série (le bus à précharge). Ainsi aucune porte ne doit contenir plus de deux transistors enrichis en série. **règle 6**.

3.3.2.4 De l'utilisation des transistors de passage

Les tensions drain-source engendrent une réduction de marge au bruit pour la même raison que celle que nous venons de voir et pose donc un problème. La nature de la jonction MESFET dont la grille n'est pas isolée créerait une consommation excessive si l'utilisation de transistors de passage était possible. Pour ces deux raisons, les transistors de passage sont à proscrire. **règle 7**.

3.3.2.5 Étude dynamique de l'inverseur

L'objet est ici de contrôler les temps de commutation. Ces temps sont respectivement désignés par t_r : le temps de montée et par t_f : le temps de descente. On cherche à exprimer ces temps en fonction de C_{load}, V_{IH} - V_{IL}, et en fonction des largeurs des transistors concernés. C_{load} est la capacité de charge visible sur la sortie de l'inverseur, elle est due au routage et à la largeur de la grille attaquée. Elle peut donc s'écrire $C_{interco} + C_0 \cdot W_L$, où C_0 est la capacité de grille par unité de longueur.

t_r est proportionnel à $\dfrac{C_{load} \cdot (V_{IH} - V_{IL})}{a_D \cdot W_D + b_D}$

t_f est proportionnel à $\dfrac{C_{load} \cdot (V_{IH} - V_{IL})}{a_E \cdot W_E + b_E - (a_D \cdot W_D + b_D)}$

Nous pouvons exploiter ces formules afin de vérifier qu'aucune porte d'un circuit n'a de temps de montée ou de descente dépassant des valeurs maximales préétablies : **règle 8**.

3.3.3 Éléments de puissance

D'après ce que nous venons de voir dans le paragraphe précédent et du fait de la grande sensibilité à la charge en sortie des portes DCFL, l'utilisation d'éléments de puissance dans certaines situations est nécessaire. Ces situations varient suivant deux critères : la surface totale de grille attaquée par un signal et l'importance de la capacité de routage. Afin de faire face à l'ensemble de ces situations, trois éléments de puissance ont été retenus : l'élément de puissance BDCFL, l'élément de puissance SBDCFL et enfin l'élément de puissance SqBDCFL. Le choix de l'un de ces éléments est dicté par le meilleur compromis *puissance/délai/surface/immunité au bruit* obtenu. Les domaines d'utilisation en fonction de la capacité d'interconnexion et de la largeur totale de grille sont illustrés ci-dessous.

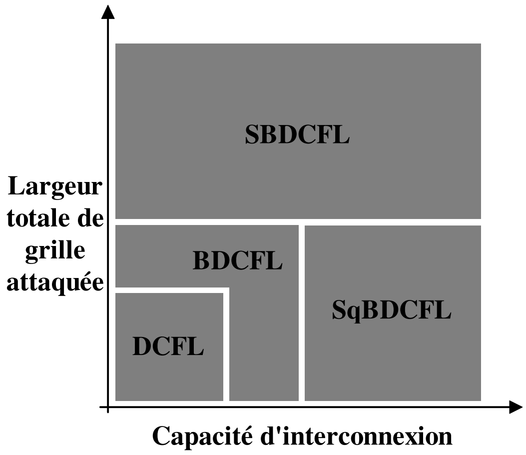


Figure 3.6 : domaine d'utilisation des éléments de puissance

L'emploi d'éléments de puissance en sortie des portes DCFL n'est pas nécessaire lorsque la capacité d'interconnexion et la sortance sont peu élevées. Dans la pratique, cela correspond à ce que les interconnexions à la porte DCFL soient locales ainsi qu'à une sortance maximale de trois portes de même puissance qu'elle. Nous reprenons dans ce qui suit l'étude de chacun des trois éléments de puissance.

3.3.3.1 Portes BDCFL

On fait appel à ce type de porte lorsque la capacité d'interconnexion reste faible (interconnexions locales) mais que la sortance augmente (augmentation de la surface de grille. Ces portes peuvent fournir plus de courant et permettent ainsi d'éviter le phénomène de "sous-polarisation" décrit au paragraphe précédent. Une illustration d'une telle porte est donnée ci-contre. Elle est composée d'une porte DCFL et d'un étage suiveur. Ce dernier est composé d'un transistor enrichi et d'un "pull-down". Le transistor enrichi a son drain relié à Vdd et sa source reliée à O. Sa grille est attaquée par la sortie de la porte DCFL. L'étage suiveur est également composé d'un transistor déplété fonctionnant en "pull-down". Son drain est relié à O et sa grille et sa source sont reliées à Gnd.

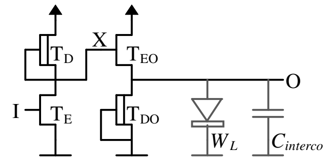


Figure 3.7 : porte BDCFL

La porte DCFL et l'étage suiveur, sont découplés et la sortie du premier attaque l'entrée du second. Lorsque l'entrée I est au niveau logique bas, le signal X se retrouve à l'état logique haut et vice versa.

Le fonctionnement statique du suiveur est le symétrique de celui de l'inverseur DCFL. Lorsque X est à l'état logique haut, T_{EO} est passant et le courant qu'il fournit, dont une partie est absorbée par T_{DO}, doit être suffisant pour polariser la charge en sortie (voir figure 3.8). Il nous faut vérifier qu'il n'y a ni "sous-polarisation" ni "sur-polarisation". On retrouve les mêmes règles 2 et 3, et leurs formules respectives mais avec $I_{pol} = a_{EO} \cdot W_{EO} + b_{EO} - (a_{DO} \cdot W_{DO} + b_{DO})$ (**règle 9** et **règle 10**). Par contre lorsque X est à l'état logique bas, T_{EO} est bloqué et comme T_{DO} est passant, il garanti un bon niveau bas en sortie (voir figure 3.9).

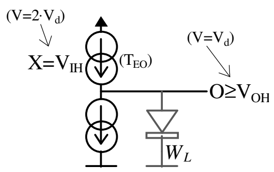


Figure 3.8 : BDCFL, entrée I à l'état bas

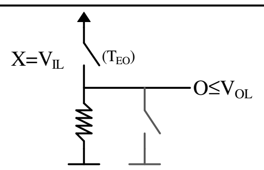


Figure 3.9 : BDCFL, entrée I à l'état haut

Il est important de noter que le niveau logique haut fournit à la grille du transistor enrichi T_{EO} n'est pas V_d mais $2 \cdot V_d$. Rappelons en effet que les jonctions grille-source correspondent à des diodes Schottky ayant une tension de seuil de V_d. Il faut vérifier qu'aucune porte DCFL attaquant un étage suiveur n'attaque également une autre porte DCFL : **règle 11**.

On s'assurera également que l'étage suiveur est bien attaqué par une porte DCFL et qu'elle-même attaque bien une porte DCFL en sortie : **règle 12** afin de garantir que les interactions entre différentes portes utilisant différents niveaux logiques haut soient cohérentes.

En ce qui concerne le comportement dynamique, les résultats obtenus sont les suivants : t_r est proportionnel à $\frac{C_{load} \cdot (V_{IH} - V_{IL})}{a_{EO} \cdot W_{EO} + b_{EO} - (a_{DO} \cdot W_{DO} + b_{DO})}$ et t_f est proportionnel à $\frac{C_{load} \cdot (V_{IH} - V_{IL})}{a_{DO} \cdot W_{DO} + b_{DO}}$. En calculant ces temps et en les confrontant à des valeurs pré-définies, il est possible de signaler un problème éventuel en cas de dépassement. **règle 13**.

3.3.3.2 Porte OR

Si nous ajoutons un ou plusieurs transistors enrichis en parallèle au suiveur décrit ci-dessus, nous obtenons une porte OR. Un OR à deux entrées est présenté ci-contre. Mais l'on se trouve confronté à deux problèmes. Le premier problème est lié à l'écrêtage à $2 \cdot V_d$ des niveaux logiques hauts appliqués en entrée du OR. Ceux-ci ne sont en effet pas compatibles avec le niveau logique haut DCFL. Il faut donc s'assurer que les signaux attaquants les entrées du OR n'attaquent que des étages suiveurs. Le deuxième est lié au fait qu'une telle porte peut avoir un ou plusieurs transistors enrichis passant simultanément. Or cela fait varier le courant fournit en sortie O entre un et k fois le courant de saturation d'un transistor enrichi, si k est le nombre de transistors passants. Ces variations étant difficile à gérer, l'on vérifiera qu'aucune porte OR n'a plus de deux entrées afin d'éviter toute "sous-polarisation" mais surtout toute "sur-polarisation". **règle 14**.

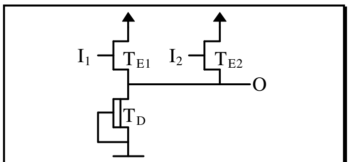


Figure 3.10 : porte OR

3.3.3.3 Portes SBDCFL

Les portes BDCFL supportant mal des sortances élevées, si la sortance devient trop grande, il est nécessaire de passer aux portes SBDCFL. Comme le montre la figure ci-contre, l'étage d'amplification est constitué d'un transistor enrichi relié à Vdd et attaqué par la sortie de l'inverseur. Il est également composé d'un deuxième transistor enrichi relié à Gnd et attaqué par le signal d'entrée de l'inverseur. Ces deux transistors ont des signaux de grille complémentaires et fonctionnent en "push-pull-ee" ("push-pull" enrichi/enrichi). Cette approche structurelle est empruntée à la logique NMOS qui utilise des portes dites "Super Buffer" [Mea80]. Leur unique différence se situe au niveau du

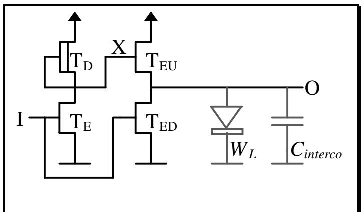


Figure 3.11 : porte SBDCFL

transistor T_{EU} qui est de nature déplété pour le NMOS. Leur principe de fonctionnement restant le même.

La sortie de la porte DCFL attaque un étage "push-pull-ee" et leurs entrées sont communes. Lorsque celle-ci est à l'état logique bas, X est à l'état logique haut. T_{EU} est alors passant et T_{ED} est bloqué. La sortie O se retrouve donc à l'état logique haut. Ici aussi, il faut s'assurer qu'il n'y a ni "sous-polarisation", ni "sur-polarisation" (voir figure 3.12). Il s'agit de vérifier les mêmes règles 2 et 3, et leurs formules respectives mais avec $I_{pol} = a_{EU} \cdot W_{EU} + b_{EU}$ (**règle 15** et **règle 16**). Lorsque l'entrée I est à l'état haut, X est à l'état logique bas. T_{EU} est bloqué et T_{ED} est passant ce qui garanti un bon niveau bas en sortie (voir figure 3.13).

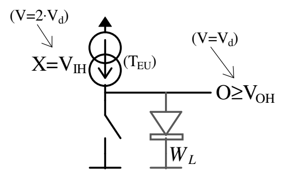


Figure 3.12 : SBDCFL, entrée I à l'état bas

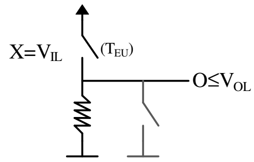


Figure 3.13 : SBDCFL, entrée I à l'état haut

Comme pour le BDCFL et pour les mêmes raisons, le niveau logique haut fournit à la grille du transistor enrichi T_{EU} n'est pas V_d mais $2 \cdot V_d$. Il faudra vérifier qu'aucune porte DCFL attaquant un étage "push-pull-ee" n'attaque également une autre porte DCFL : **règle 17**.

On s'assurera également que l'étage "push-pull-ee" est bien attaqué par une porte DCFL et qu'elle-même attaque bien une porte DCFL en sortie : **règle 18** afin de garantir que les interactions entre différentes portes utilisant différents niveaux logiques haut soient cohérentes.

Du point de vue dynamique, les temps de montée et de descente sont respectivement proportionnels à $\frac{C_{load} \cdot (V_{IH} - V_{IL})}{a_{EU} \cdot W_{EU} + b_{EU}}$ et à $\frac{C_{load} \cdot (V_{IH} - V_{IL})}{a_{ED} \cdot W_{ED} + b_{ED}}$. Si l'on désire vérifier qu'ils ne dépassent pas certaines limites fixées à l'avance, alors il suffit de calculer ces valeurs et de les comparer à ces dernières. **règle 19**.

Par ailleurs, il faudra vérifier que les transistors de l'étage "push-pull-ee" ne sont pas sur-dimensionnés. Car lors des transitions, il existe un cours instant pendant lequel un chemin électrique s'établit entre Vdd et Gnd ce qui provoque un pic de courant. **règle 20**.

3.3.3.4 Portes SqBDCFL

La porte SqBDCFL présentée ci-contre est idéale lorsque la capacité d'interconnexion est élevée et que la sortance est faible. Cette porte, connue sous le nom de "Squirt Buffer" a été introduite pour la circuiterie AsGa MESFET. Elle emploie *un étage DCFL muni d'une branche de contre réaction*, que l'on appellera étage Squirt, ainsi qu'un étage "push-pull-de" ("push-pull" déplété/enrichi). L'étage "push-pull-de" est constitué d'un transistor déplété dont le drain est relié à Vdd et la source à la sortie O de la porte. La grille quant à elle, est attaquée par la sortie de l'étage Squirt. Le transistor enrichi a son drain relié à la sortie O de la porte et sa source reliée à Gnd. Sa grille est attaquée par le signal d'entrée de la porte. La branche de contre réaction est constituée d'une diode et d'un transistor enrichi en série. L'anode de cette diode Schottky est reliée à la sortie de l'étage inverseur DCFL et sa cathode est reliée au drain du transistor enrichi. Ce dernier a sa source reliée à Gnd et sa grille reliée à la sortie de la porte SqBDCFL.

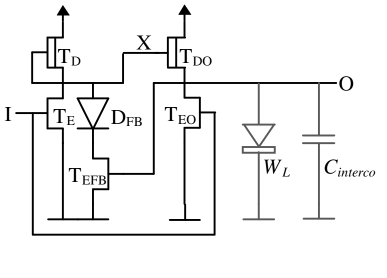


Figure 3.14 : porte SqBDCFL

Voyons maintenant le comportement statique de la porte. Lorsque l'entrée I est à l'état logique bas, les transistors T_E et T_{EO} sont bloqués. Le signal X est à l'état logique haut et T_{DO} est passant. L'utilisation de la branche de contre réaction permet d'éviter toute "sur-polarisation" sur la sortie O. En effet, alors que le niveau logique en O augmente vers son niveau haut, le transistor T_{EFB} devient passant. D'une part, cela permet à la branche de contre réaction d'empêcher que le signal X ne devienne trop élevé en écoulant une partie du courant du nœud X. D'autre part, cela permet d'écouler l'excédent de courant de la sortie O au travers la jonction du transistor T_{EFB}. Le rôle de la diode D_{FB} est d'éviter que le niveau logique haut en X ne s'abaisse trop à cause de la branche de contre réaction, ce qui risquerait de dégrader la quantité de courant fournit par T_{DO} (voir figure 3.15). Comme précédemment, il est nécessaire de vérifier qu'il n'y a ni "sous-polarisation" ni "sur-polarisation". Il s'agit toujours de vérifier les règles 2 et 3 mais pour un courant de polarisation valant $I_{pol} = a_{DO} \cdot W_{DO} + b_{DO}$ (**règle 21** et **règle 22**). Lorsque l'entrée I est à l'état logique haut, les transistors T_E et T_{EO} sont passants et le nœud X est à l'état bas. T_{EO} ayant été dimensionné en conséquence, il assure l'écoulement de tout le courant fournit par le transistor déplété T_{DO} qui reste passant (voir figure 3.16).

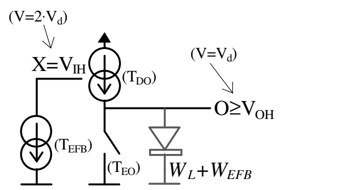


Figure 3.15 : SqBDCFL, entrée I à l'état bas

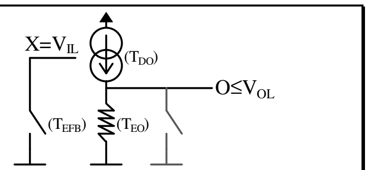


Figure 3.16 : SqBDCFL, entrée I à l'état haut

De même que pour le SBDCFL et pour les mêmes raisons, le niveau logique haut fournit à la grille du transistor déplété T_{DO} n'est pas V_d mais $2 \cdot V_d$. Il faudra vérifier qu'aucune porte DCFL attaquant un étage "push-pull-de" n'attaque également une autre porte DCFL : **règle 23**.

Comme pour le SBDCFL, on s'assurera que l'étage "push-pull-de" est bien attaqué par une porte DCFL et qu'elle-même attaque bien une porte DCFL en sortie : **règle 24** afin de garantir que les interactions entre différentes portes utilisant différents niveaux logiques haut soient cohérentes.

3.4 Conclusion

Dans ce chapitre, nous avons présenté l'outil d'abstraction fonctionnelle DESB, dédié à la vérification des circuits CMOS, dont nous nous sommes fortement inspirés. Nous avons particulièrement insisté sur l'algorithme de reconstruction des portes qui devra être profondément modifié pour la technologie AsGa DCFL.

Par ailleurs, nous avons présenté les principaux résultats de l'étude sur la circuiterie DCFL réalisée par P. Royannez, ainsi qu'une liste explicite de règles électriques, déduite de cette analyse, et qui devront être vérifiées. Nous avons montré en particulier que ces règles étaient spécifiques à chaque type de porte.

4. Principes généraux de la méthode proposée

4.1 Introduction

Comme nous l'avons mentionné précédemment, nous cherchons à effectuer des vérifications fonctionnelle, électrique et temporelle pour des circuits prêts à être envoyés en fabrication. L'information disponible en entrée est la *net-list* au niveau transistor, telle qu'elle peut être extraite du dessin des masques de fabrication. Cette description initiale a le mérite de refléter fidèlement les propriétés électriques du circuit physique, mais elle est peu adaptée à nos besoins. En effet, dans un circuit numérique, il existe un sens de propagation de l'information des entrées vers les sorties à travers des portes logiques orientées, niveau d'abstraction qui convient particulièrement bien aux vérifications que nous désirons entreprendre. Nous cherchons donc à atteindre ce niveau d'abstraction.

Il faut donc reconstruire une *net-list* d'étages orientés à partir de la *net-list* de transistors. Pour cela, nous reprenons le modèle des "cônes" proposé dans la thèse de M. Laurentin [Lau94]. Un étage orienté peut être assimilé en première approximation à une porte logique ayant plusieurs entrées et une seule sortie. De façon plus formelle, nous définissons un étage orienté comme un ensemble de transistors ayant les propriétés suivantes :

- tout signal X attaquant la grille d'un transistor commutant définit un étage orienté, dont il constitue le signal de sortie. Un transistor commutant est un transistor dont la grille, la source et le drain sont connectés à trois équipotentielles distinctes, et dont la grille n'est pas connectée à une alimentation.
- L'ensemble des transistors appartenant à un chemin électrique entre le signal de sortie X et un des deux signaux d'alimentation Vdd et Gnd appartiennent à l'étage orienté. Chaque chemin électrique constitue une branche de l'étage orienté. Une branche connectée à l'alimentation Vdd sera dite de type "UP". Une branche connectée à l'alimentation Gnd sera dite de type "DOWN".
- Un étage orienté possède un ou plusieurs signaux d'entrée correspondant aux signaux qui sont connectés aux grilles des transistors commutant des branches.

L'étage orienté figurant ci-contre possède :

- une sortie O
- une branche de type UP
- trois branches de type DOWN
- trois entrées I_1, I_2 et I_3

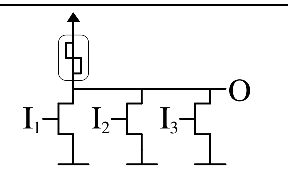


Figure 4.1 : représentation en étage d'un NOR 3

Le modèle initialement proposé dans la thèse de M. Laurentin [Lau94] pour la circuiterie CMOS, et implémenté dans l'outil DESB, a été étendu pour intégrer le concept de cellule. Une cellule est constituée par un assemblage d'étages orientés réalisant une fonction complexe, telle qu'un registre, un émetteur de bus trois-états, ou un amplificateur.

Pour ce qui concerne le mécanisme d'abstraction, la principale différence entre l'approche mise en œuvre dans DESB pour la circuiterie CMOS, et l'approche proposée ici pour la circuiterie AsGa est l'utilisation systématique de techniques de reconnaissance de formes dans la phase de reconstruction des étages orientés. En effet, au moins deux des vérifications que nous souhaitons effectuer (vérification électrique et analyse temporelle) nécessitent un typage fort des portes : une porte logique DCFL et une porte logique BDCFL n'ont pas les mêmes contraintes de dimensionnement électriques, ni les mêmes modèles de calcul des temps de propagation. Un mécanisme de reconnaissance de formes est donc inévitable pour effectuer ce typage.

L'inconvénient majeur des techniques de reconnaissance de formes est évidemment l'obligation de définir a priori un ensemble de formes à reconnaître. Compte tenu des caractéristiques particulières de la circuiterie AsGa MESFET, ce choix fondamental semble incontournable. Toutes les vérifications ultérieures reposent en effet sur le typage des étages ou, en d'autres termes, sur la classification adoptée.

Signalons dès à présent que la reconnaissance d'un étage accepte évidemment un certain degré de généricité. La taille des transistors est variable (ce qui peut se traduire par la mise en parallèle de transistors) ainsi que le nombre d'entrées de certains étages comme le NOR.

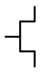
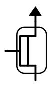
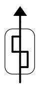
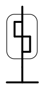
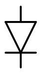
À la fin de l'abstraction, on obtient une *net-list* d'étages orientés interconnectés, qui constitue une information plus riche, car plus structurée que la *net-list* de transistors initiale. Il s'agit en effet d'un graphe orienté (on a résolu le problème du sens de propagation des signaux) et typé (chaque étage orienté possède un type auquel on peut attacher des caractéristiques particulières du point de vue de la vérification électrique ou du comportement temporel). De plus, les structures complexes, résultant de l'assemblage de plusieurs étages orientés qu'il est nécessaire de connaître pour les vérifications électrique et temporelle (registres, amplificateurs, etc.), sont clairement identifiées en tant que cellules. La structure de données correspondante, appelée "CNS" est décrite dans la section 4.4 de ce chapitre. Cette structure de données, définie pour les besoins de la circuiterie AsGa, est maintenant également utilisée par les outils de vérification CMOS de la chaîne de CAO ALLIANCE.

4.2 Présentation de la classification

Nous l'avons dit, le typage des étages orientés est un point central de la méthode de vérification proposée. Deux sortes d'étages peuvent être présents dans un circuit AsGa MESFET utilisant la logique DCFL. Les étages "simples" réalisent des portes logiques élémentaires, indépendamment de la présence d'un autre étage en amont ou en aval. Les étages "spécifiques" ne correspondent pas à des portes logiques indépendantes :

ils sont nécessairement utilisés en complément avec d'autres étages pour réaliser une fonction logique particulière telle qu'un amplificateur ou un registre. Ils apparaissent toujours dans la composition d'une "cellule" (Rappelons qu'une cellule est un assemblage particulier d'étages orientés).

Une *bonne* typologie doit permettre d'analyser des circuits contenant aussi bien de la logique combinatoire que de la logique séquentielle. Pour la logique combinatoire, la bibliothèque de formes à reconnaître doit contenir un minimum de portes élémentaires à partir desquelles n'importe quel opérateur logique peut être construit. Il doit contenir en outre des éléments de puissance permettant d'amplifier un signal. En ce qui concerne la logique séquentielle, nous nous sommes limités dans cette première approche à un seul schéma de *latch*, qui a été caractérisé par le concepteur de la bibliothèque [Roy96]. L'ensemble des types reconnus est constitué de onze étages et de cinq cellules. Nous donnons les caractéristiques de chaque type d'étage puis de chaque type de cellule accompagné d'un schéma utilisant les symboles suivants :

- 🡅 et ⊥ représentent respectivement les tensions d'alimentation Vdd et Gnd,
-  représente le transistor MESFET enrichi,
-  représente le "driven pull-up",
-  et  représentent respectivement le "pull-up" et le "pull-down",
-  représente la diode Schottky.

Le "driven pull-up" correspond à un transistor déplété dont le drain est relié à Vdd. Les charges actives "pull-up" et "pull-down" sont réalisées au moyen d'un transistor déplété sur lequel la grille et la source ont été court-circuitées. Cet élément est relié à Vdd dans le cas du "pull-up" et à Gnd dans le cas du "pull-down". La diode Schottky est réalisée au moyen d'un transistor enrichi sur lequel la source et le drain ont été court-circuités. La grille du transistor correspond à l'anode de la diode.

4.2.1 Les étages simples

L'opérateur logique élémentaire est le NOR. Représenté ci-contre, cet étage est constitué d'un "pull-up" et d'un certain nombre de transistors enrichis reliés à Gnd. Les signaux de grilles de ces transistors correspondent aux entrées de cet étage. Celui-ci est utilisé comme inverseur lorsqu'il n'a qu'une entrée, sinon il peut en comporter jusqu'à 5. S'il y a plus de 3 entrées, il est nécessaire que chaque étage en entrée du NOR soit un étage de puissance assurant ainsi un niveau bas correct.

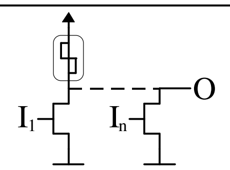


Figure 4.2 : étage NOR

L'étage OR est constitué d'un "pull-down" et de un ou deux transistors enrichis reliés à Vdd. Les signaux de grille correspondent à ses entrées. Il peut être utilisé en tant qu'étage de puissance lorsqu'il n'a qu'une seule entrée. Il se comporte alors en étage suiveur. Par ailleurs, il permet de réaliser un NAND à deux entrées : chaque entrée du NAND attaque un inverseur, chacun attaquant à son tour une des entrées d'un OR à deux entrées.

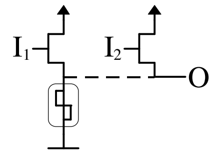


Figure 4.3 : étage OR

Le NOR à Précharge et le BUS à Précharge permettent la réalisation de logique distribuée. Le BUS à Précharge notamment, est utilisé pour la lecture des bancs de registres que nous présentons plus loin. Chacun de ces étages possède sur son signal de sortie un transistor enrichi dont la grille doit être reliée à un signal d'horloge et dont le rôle sera de précharger au niveau haut le signal de sortie.

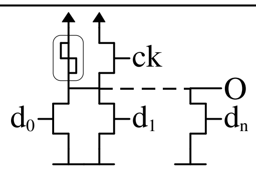


Figure 4.4 : étage NOR à Précharge

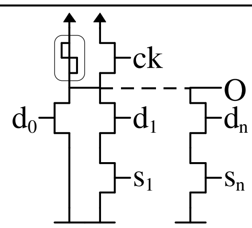


Figure 4.5 : étage BUS à Précharge

Il est nécessaire d'avoir des étages fournissant les niveaux logiques "0" et "1". Ces derniers sont les plus simples possibles puisque n'étant composés respectivement que d'un "pull-down" et d'un "pull-up".

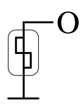


Figure 4.6 : étage Mise à Zéro

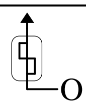


Figure 4.7 : étage Mise à Un

Le comportement VHDL de chacun des étages présentés ci-dessus est fourni dans l'annexe 2.

4.2.2 Les étages spécifiques

Le Super Buffer E représenté ci-contre supporte de fortes charges capacitives en sortie et permet d'attaquer un nombre conséquent d'étages. En revanche, il n'est pas découplé des étages qui l'attaquent en entrée. Cet étage fait partie de la porte SBDCFL présentée à la section 3.3.3.3. La cellule de puissance Super Buffer présentée dans la prochaine section est basée sur cet étage de puissance.

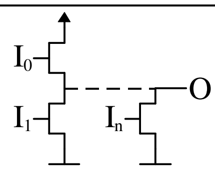


Figure 4.8 : étage SBE

L'étage de puissance Super Buffer D et l'étage Squirt Buffer figurant ci-dessous sont utilisés conjointement afin de réaliser la cellule de puissance Squirt Buffer décrite à la section suivante. Ces deux étages ont été présentés au chapitre précédent, ils font partie de la porte SqBDCFL (voir 3.3.3.4).

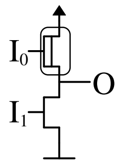


Figure 4.9 : étage SBD

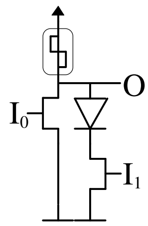


Figure 4.10 : étage Squirt

L'étage Demi-Latch représenté à droite est nécessaire pour la construction de la cellule Latch. Cet étage est composé d'un "pull-up", d'une branche constituée d'un seul transistor enrichi qui servira pour le rebouclage, et d'une branche constituée de deux transistors enrichis correspondant à la donnée et à l'autorisation d'écriture respectivement.

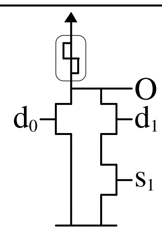


Figure 4.11 : étage Demi-Latch

Ci-contre figure l'étage Tristate qui est employé comme étage de sortie des cellules portant le même nom. La sortie de cet étage est un bus sur lequel sont reliés autant de couples de MESFETs enrichis, l'un connecté à Vdd et l'autre connecté à Gnd, qu'il y a d'émetteurs sur le bus.

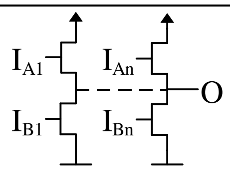
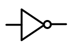
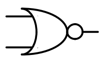


Figure 4.12 : étage Tristate

Il n'y a pas de comportement VHDL associé à chacun des étages spécifiques présentés ci-dessus puisque ceux-ci doivent toujours être utilisés à l'intérieur d'une cellule.

4.2.3 Les cellules

Rappelons qu'une cellule est un assemblage particulier d'étages orientés. Nous introduisons deux nouveaux symboles, afin de ne pas alourdir les schémas :

-  représente l'inverseur qui correspond au NOR à une entrée,
-  représente le NOR à deux entrées.

La cellule de puissance Super Buffer regroupe un étage NOR en entrée suivi d'un étage SBE qui a donc autant de MESFETs enrichis vers Gnd que l'étage NOR en entrée n'a lui-même d'entrées.

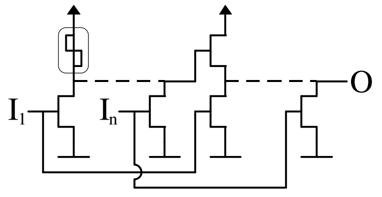


Figure 4.13 : cellule Super Buffer

La cellule de puissance Squirt Buffer est constituée de l'étage Squirt Buffer en entrée et de l'étage SBD en sortie.

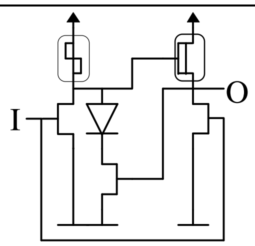


Figure 4.14 : cellule Squirt Buffer

La cellule Tristate est composée de quatre étages à savoir d'un inverseur (étage NOR à une entrée), de deux étages NORs à deux entrées et d'un étage de puissance SBE. Chaque étage NOR à deux entrées attaque un des deux MESFETs enrichis de l'étage SBE. Ces étages NORs ont en entrée un signal "noe" (no output enable) qui met la sortie de l'étage SBE en haute impédance lorsqu'il est à l'état logique haut. La donnée ainsi que son inverse sont respectivement fournis sur la deuxième entrée de chaque étage NOR.

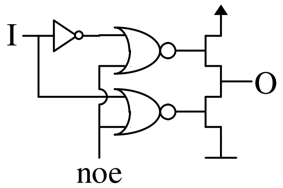


Figure 4.15 : cellule Émetteur Tristate

La cellule Latch est composée de deux étages Demi-Latch rebouclés. L'écriture est effectuée au moyen de deux branches "pull-down" différentielles. Chacune de ces branches est composée de deux transistors enrichis en série reliés à Gnd. Le signal de sélection en écriture attaque la grille du transistor enrichi de chaque branche relié à Gnd. La donnée et son inverse attaquent respectivement la grille du deuxième transistor enrichi de chaque branche.

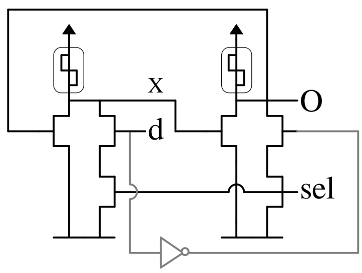


Figure 4.16 : cellule Latch

En vue de réaliser un circuit de lecture pour bancs de registres, une cellule de Lecture a été ajoutée. Celle-ci est construite à l'aide d'un étage BUS à Précharge qui attaque un étage NOR à deux entrées. Le transistor enrichi allant à Vdd du BUS à Précharge ainsi qu'une des entrées du NOR sont attaqués par l'horloge. Un inverseur de rebouclage est utilisé afin d'accélérer la précharge et de compenser les courants de fuite des branches de sélection.

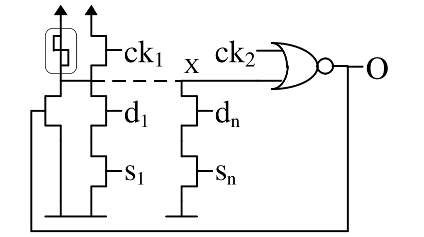


Figure 4.17 : cellule Bus de Lecture

Le comportement VHDL de chacune des cellules présentées dans cette section se trouve dans l'annexe 3.

En résumé, la classification proposée consiste en un ensemble de onze *types* d'étages et de cinq *types* de cellules.

4.3 Architecture logicielle

4.3.1 Vue d'ensemble

L'architecture logicielle que nous présentons dans cette section est illustrée par la figure 4.18 :

- les rectangles représentent les différents modules ou outils logiciels de l'environnement de vérification VEGA ;
- la forme ovale correspond à la structure de données CNS qui est la structure représentant la *net-list* des étages et des cellules orientées ;
- les cylindres désignent les différents fichiers en entrée et en sortie de l'environnement.

Il existe trois types de modules logiciels : les modules TNA, FOX, OSR, ESR correspondent aux quatre phases d'abstraction. Les modules EVM, BXM, TAGA sont les outils de vérification.

Différentes personnes ont contribué à développer les modules de vérification. Les règles électriques vérifiées par EVM ont été définies dans la thèse de P. Royannez [Roy96]. L'analyseur de performances temporelles TAGA a été développé par J. Pistorius dans le cadre de son stage de DEA. J'ai personnellement défini l'architecture générale de cet environnement de vérification, j'ai développé la structure de donnée CNS, le module BXM, ainsi que tous les modules d'abstraction.

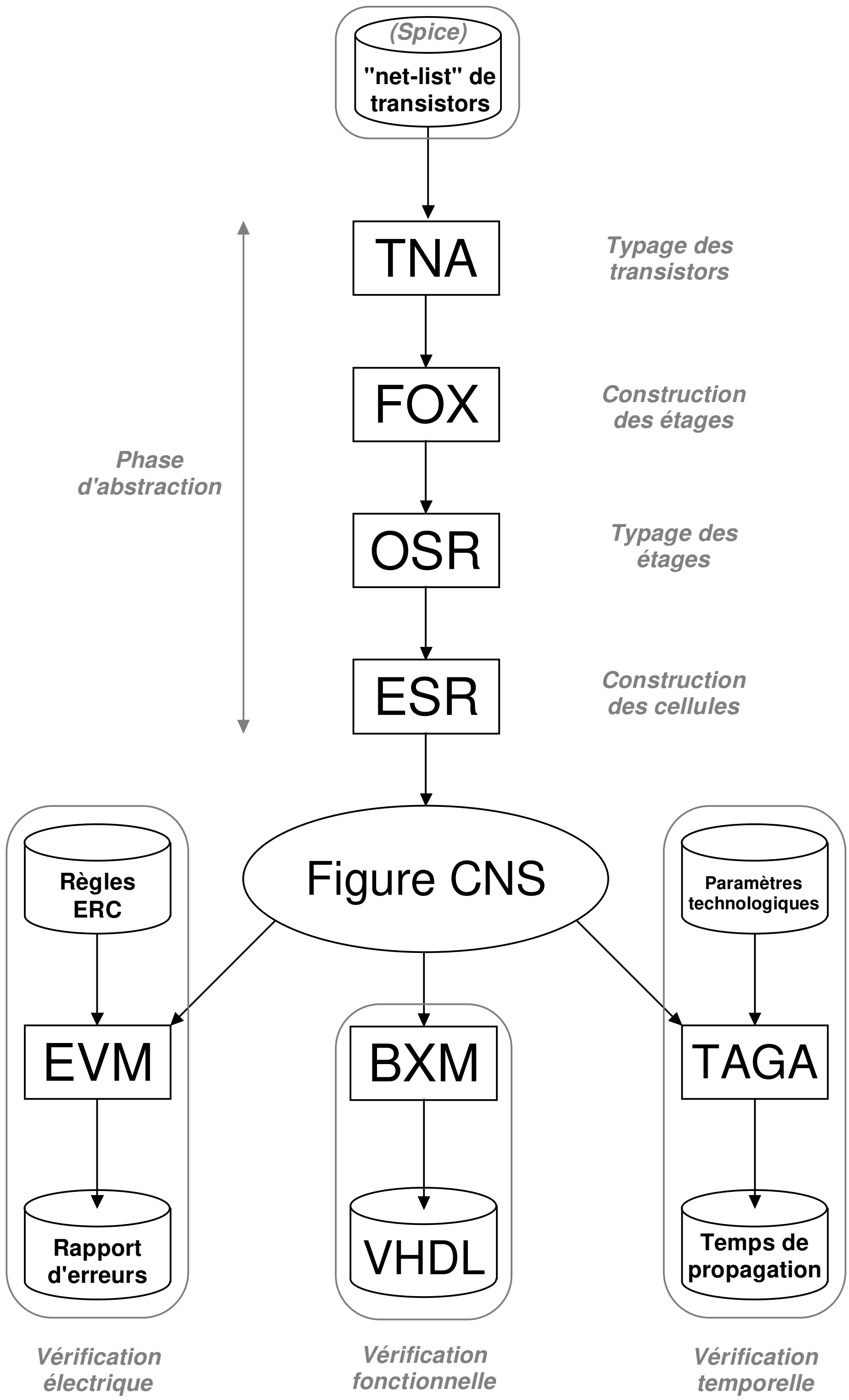


Figure 4.18 : architecture logicielle de l'environnement de vérification VEGA

4.3.2 Typage des transistors : module TNA

Comme nous l'avons dit, l'environnement de vérification prend en entrée la *net-list* de transistors préalablement extraite du dessin des masques du circuit devant être vérifié. La principale fonction de ce module est de classer les transistors en cinq types, en analysant localement la connectique et la nature (enrichi ou déplété) de chaque transistor.

La complexité algorithmique de cette étape est linéaire avec le nombre de transistors.

4.3.3 Construction des étages : module FOX

Cette deuxième étape est entièrement algorithmique et s'inspire fortement de la méthode mise en œuvre dans DESB pour la circuiterie CMOS.

Rappelons que la sortie d'un étage correspond à un signal sur lequel doit se trouver au moins une grille d'un transistor commutant ou un connecteur externe. Un étage est constitué de l'ensemble des transistors par lesquels peuvent s'établir des chemins électriques reliant une alimentation à la sortie de l'étage. Une branche est l'ensemble des transistors appartenant à un chemin électrique entre une alimentation et une sortie d'étage. Un étage a donc autant de branches qu'il y a de chemins électriques reliant une alimentation à sa sortie. La circuiterie DCFL ne permet que de manière exceptionnelle la mise en série de deux transistors. La grande majorité des étages rencontrés seront donc composés de branches n'ayant qu'un transistor. À partir d'une alimentation, la profondeur d'exploration du graphe des signaux interconnectant les transistors est donc limitée à 2. Un chemin électrique de longueur supérieure à 2 est signalé comme une erreur.

À la fin de ce module, nous sommes en présence d'une *net-list* d'étages orientés, mais non typés. Ces étages peuvent avoir n'importe quelle structure en termes de nombre et de type de branches, tant que la longueur des branches est inférieure ou égale à 2.

La complexité algorithmique de cette étape est essentiellement linéaire avec le nombre de transistors.

4.3.4 Typage des étages : module OSR

Cette troisième étape a pour fonction de valider et de typer chaque étage construit lors de l'étape précédente. On utilise une méthode de reconnaissance de forme. La *net-list* d'étages va donc être parcourue une fois et chaque étage va être confronté aux formes présentées dans les sections 4.2.1 et 4.2.2 afin d'être reconnu puis typé. La comparaison entre un étage et une forme donnée s'effectue au niveau structurel et elle possède deux degrés de liberté à savoir la taille des transistors (qui peut également s'exprimer par la présence de branches parallèles), et le nombre d'entrées. La présence d'un étage ne correspondant à aucun type connu est signalée comme une erreur.

Cette étape achevée, nous nous retrouvons en présence d'une *net-list* d'étages orientés typés.

La complexité algorithmique de cette étape est linéaire avec le nombre d'étages, qui est lui-même proportionnel au nombre de transistors.

4.3.5 Construction des cellules : module ESR

Il s'agit à présent de déceler dans cette *net-list* d'étages orientés les structures d'un niveau sémantique plus élevé que sont les cellules. Cela, afin de parvenir à la "figure cns" complète et correcte à partir de laquelle il sera possible d'effectuer les vérifications électrique, fonctionnelle et temporelle. Précisons que cette identification des cellules ne modifie pas la *net-list* des étages orientés, mais se contente de l'enrichir. Le regroupement des étages en cellules ne constitue pas une partition de l'ensemble des étages, puisqu'un même étage peut être partagé par plusieurs cellules, et que tous les étages n'appartiennent pas nécessairement à une cellule. La *net-list* d'étages est parcourue en une passe.

À partir de la figure cns à laquelle nous sommes parvenus à ce stade, il est possible d'effectuer indépendamment les différentes vérifications électrique, fonctionnelle et temporelle.

4.3.6 Vérification électrique : module EVM

Ce module a pour tâche de signaler toute violation des règles de conception électriques. Ces règles permettent de vérifier le comportement électrique statique et, dans une moindre mesure, dynamique, du circuit. Elles peuvent être de nature structurelle comme par exemple : "*un étage NOR ne doit pas avoir plus de 5 entrées*". Elles peuvent porter sur le dimensionnement des transistors, comme par exemple *"le coefficient β d'un étage NOR doit être compris entre 9 et 10"*. Les règles sont attachées soit à un type d'étage, soit à un type de cellule. La règle doit donc être vérifiée pour tous les étages ou toutes les cellules du type considéré.

Dans une première approche, le nombre de règles effectivement vérifiées dans le prototype logiciel a été limité à 12. Il est clair que ces 12 règles ne couvrent pas de manière exhaustive tous les problèmes électriques que peut engendrer la circuiterie AsGa MESFET DCFL. En revanche, le jeu de règles choisi est représentatif de tous les types de contraintes que nous souhaitons vérifier, et permet de valider le principe de la méthode.

Les paramètres utilisés par les règles sont issus de simulations SPICE et sont placés dans un fichier texte qui est lu par le module EVM. Paramètres et règles sont issus des travaux de thèse de P. Royannez [Roy96] et sont propres à la technologie utilisée, qui est en l'occurrence celle de Vitesse pour le procédé H-GaAsII [Vit93].

Si une règle est violée, une erreur est signalée au concepteur.

4.3.7 Vérification fonctionnelle : module BXM

Deux méthodes existent pour vérifier la fonctionnalité d'un circuit, il s'agit de la preuve formelle et de la co-simulation. Elles consistent à vérifier que deux représentations comportementales d'un même circuit, l'une issue des spécifications, l'autre extraite de la réalisation, sont fonctionnellement identiques. Dans un cas comme dans l'autre, la première étape consiste à générer un modèle VHDL de niveau RTL à partir du schéma en transistors correspondant à la réalisation. C'est l'objet du module BXM.

La description VHDL générée est un modèle VHDL de type "data-flow", qui ne contient donc que des affectations concurrentes. Par ailleurs, le modèle VHDL généré est un modèle "zéro délais", sans information temporelle détaillée (c'est-à-dire sans utilisation de la construction "AFTER"). Le principe de base est simple : on définit pour chaque type d'étage simple, et pour chaque type de cellule, le comportement VHDL correspondant à la fonction réalisée par l'étage ou par la cellule. On parcourt alors la structure de données CNS, et on génère pour chaque étage et/ou chaque cellule présente dans la structure l'affectation concurrente qui convient, en remplaçant les noms de paramètres formels par les noms de signaux présents dans la structure CNS. Les affectations étant concurrentes, il n'y a pas de problème d'ordonnancement, dès lors que les registres sont clairement identifiés.

4.3.8 Vérification temporelle

On vise ici l'analyse temporelle statique. L'intérêt de cette méthode de vérification par rapport à celles basées sur la simulation est de pouvoir vérifier un circuit de manière exhaustive indépendamment de tout stimuli externe.

De manière générale l'analyse statique consiste à rechercher les temps de propagation entre certains signaux particulièrement significatifs d'un circuit. Ces signaux peuvent être des entrées, des sorties ou encore des registres internes. Sa finalité principale est la détermination, pour un circuit donné, du chemin critique. Connaître ce chemin permet d'identifier les éléments qui devraient être optimisés afin d'améliorer les performances du circuit.

La première étape consiste à calculer pour chaque étage présent dans la structure CNS tous les temps de propagation entre la sortie et chacune des entrées de l'étage en cours de traitement. La partie la plus délicate de ce module consiste dans la définition et la validation des modèles temporels associés à chaque type d'étage. Ces modèles sont paramétrés par la taille des transistors, la capacité de charge du signal de sortie, et les durées des fronts des signaux d'entrée. À chaque type d'étage correspond donc un jeu d'équations et un ensemble de paramètres propre à la technologie, ces derniers ayant été obtenus par simulation SPICE. La *net-list* des étages est parcourue une seule fois et les temps de propagation de chaque étage sont calculés suivant le type de l'étage.

La seconde étape a pour fonction de déduire, à partir de la structure CNS enrichie de ces temps de propagation locaux, les temps de propagation "globaux" entre les points caractéristiques du circuit (entrées, sorties, et registres internes). Cette étape de l'analyse n'est pas spécifique à la circuiterie AsGa, et on a pu réutiliser sans modification cette partie de l'analyseur temporel TAS développé par K. Dioury au laboratoire MASI pour la circuiterie CMOS [Dio97].

4.4 Structure de données CNS

La nature des nombreuses vérifications devant être effectuées sur les circuits CMOS aussi bien que de celles que l'on se propose d'effectuer pour les circuits AsGa et présentées dans cette thèse nous a poussé, pour les raisons explicitées au cours des sections précédentes, à développer une structure de données spécifiquement adaptée à ces tâches. L'aspect central de cette structure est la représentation d'un circuit sous forme de *net-list* d'étages orientés, ou "cônes". CNS est l'acronyme de "Cône Net-list Structure". Dans la suite, on utilisera le terme "cône" pour "étage orienté".

Un certain nombre d'éléments, parmi les plus significatifs de la structure, ont déjà été présentés au cours de ce chapitre. Il s'agit des cellules, des cônes, des branches et des maillons. On utilise la technique des listes chaînées pour représenter des ensembles dont le cardinal est quelconque (par exemple la liste des branches d'un cône).

Tous les éléments de la structure sont constitués de différents champs et chacun de ces champs correspond soit à une liste d'autres éléments soit à des valeurs. Seuls les champs caractéristiques des éléments choisis sont donnés, le lecteur pourra se référer à l'annexe 1 qui contient une description complète et détaillée de la structure et des fonctions.

4.4.1 La figure cns

La figure cns est la représentation d'un circuit ou d'un sous-circuit. Elle correspond à la racine de la structure. Les algorithmes de construction de la structure ou de vérification étant de complexité linéaire, une figure cns peut contenir plusieurs dizaines de milliers de cônes. Cette figure cns comporte sous forme de listes chaînées les éléments suivants :

- la liste des connecteurs externes du circuit ; il s'agit de son interface,
- la liste des transistors contenus dans le circuit,
- la liste des cônes composant le circuit,
- la liste des cellules présentes dans le circuit,
- un champ permettant d'accueillir sa description comportementale.

4.4.2 Le cône

Le cône est un étage orienté. Les champs les plus significatifs sont les suivants :

- un nom qui correspond à celui du signal de sortie du cône,
- un type le caractérisant (par exemple : NOR),

- la liste des cônes que le cône en question a en entrée,
- la liste des cônes qui ont le cône en question en entrée,
- la liste des branches de type UP contenues dans le cône,
- la liste des branches de type DOWN contenues dans le cône,
- la liste des cellules contenant le cône en question.

4.4.3 La branche

Comme nous l'avons vu dans les sections précédentes, une branche représente un chemin électrique entre une sortie de cône et un signal d'alimentation. Elle contient tous les transistors qui se trouvent sur un chemin donné. Ces champs sont :

- un type la caractérisant (par exemple : UP),
- la liste des maillons contenus dans la branche.

4.4.4 Le maillon

Élément de plus bas niveau de la structure, il entre dans la composition des branches.

- un type le caractérisant (par exemple : "pull-up"),
- l'élément en question (un transistor ou un connecteur),
- la capacité visible sur le connecteur de transistor se trouvant "le plus proche" du signal de sortie du cône ou visible sur le connecteur.

4.4.5 La cellule

La cellule est composée d'au moins deux cônes. L'ensemble que forment les cônes qu'elle contient correspond à une fonction logique. Les champs significatifs sont les suivants :

- un type la caractérisant (par exemple : Super Buffer),
- la liste des cônes internes à la cellule,
- un champ permettant d'accueillir sa description comportementale.

4.4.6 Interface procédurale

Une interface procédurale permet de manipuler cette base de données. Le détail de chacune des fonctions disponibles est décrit dans l'annexe 1.

Le fait que la plupart des éléments de la structure soient des listes chaînées rend possible l'utilisation de fonctions génériques. La gestion de la mémoire s'appuie sur l'allocation par page des divers éléments de la structure.

Les fonctions génériques s'appliquent à un élément donné ou à une liste chaînée d'éléments :

- initialiser_élément (dans une page mémoire),
- libérer_élément (de sa page mémoire),
- libérer_liste_d'éléments (de ou des pages mémoire concernées),
- ajouter_élément,
- supprimer_élément,
- concaténer_listes_d'éléments,
- afficher_élément.

4.5 Conclusion

Nous avons justifié dans ce chapitre le choix d'une méthode mixte pour la phase d'abstraction fonctionnelle : cette approche est purement algorithmique dans la phase de construction des portes, et s'appuie sur la reconnaissance de formes pour la phase de typage des portes. Nous avons introduit le concept de cellule pour représenter les montages complexes utilisés en circuiterie AsGa DCFL. Nous avons proposé une classification complète des portes et des cellules utilisées pour les circuits AsGa DCFL. Enfin, nous avons présenté l'architecture générale de la chaîne de vérification.

Les chapitres suivants vont détailler les différents modules présents dans cette chaîne.

5. Algorithmes mis en œuvre dans la phase d’abstraction

5.1 Introduction

VEGA, l'environnement de vérification que nous avons développé comprend les quatre étapes d'abstraction et les trois modules de vérification présentés au chapitre précédent. Nous détaillons dans ce chapitre les étapes d'abstraction dans l'ordre dans lequel elles sont effectuées au sein de VEGA. Aucune de ces étapes n'est facultative. Rappelons que ces différentes étapes ont pour but de faire passer la description en *net-list* de transistors du circuit à vérifier en sa représentation en *net-list* CNS comportant cônes et cellules. Le niveau d'abstraction croît à chaque étape. Les vérifications se prêtant bien à un niveau d'abstraction donnée sont effectuées au cours de l'étape associée.

5.2 TNA : typage des transistors

Nous sommes en présence de la *net-list* de transistors du circuit devant être vérifié. Cette représentation permet d'accéder aux caractéristiques de chaque transistor ainsi qu'aux équipotentielles reliant leurs connecteurs. Il est possible de connaître la fonctionnalité de chacun d'eux par une analyse locale.

Nous commençons par donner les configurations des transistors qui sont reconnus puis nous décrivons le fonctionnement du module.

5.2.1 Configurations reconnues

Cinq configurations de transistors sont reconnues, elles correspondent chacune à une sorte de transistor et à une connectique donnée.

- Le "transistor commutant" correspond à un transistor enrichi pour lequel son drain, sa source et sa grille se trouvent sur des équipotentielles distinctes.
- Le "driven pull-up" correspond à un transistor déplété ayant son drain connecté à Vdd, et pour lequel sa source et sa grille se trouvent sur des équipotentielles distinctes.
- Le "pull-up" correspond à un transistor déplété ayant son drain connecté à Vdd, sa source et sa grille étant connectées en sortie de porte.
- Le "pull-down" correspond à un transistor déplété ayant sa source et sa grille connectée à Gnd, son drain étant connecté en sortie de porte.
- La "diode Schottky" correspond à un transistor enrichi ayant sa source et son drain connectés ensemble.

5.2.2 Description de l'algorithme

Cet algorithme, qui a pour fonction de s'assurer que chaque transistor du circuit est bien dans une configuration connue, est linéaire avec le nombre de transistors. Il consiste en un parcours de la *net-list* de transistors au cours duquel la configuration dans laquelle se trouve chaque transistor est comparée successivement à chacune des cinq configurations décrites précédemment. Dès que celle-ci est identifiée, son type associé est rattaché au transistor en question. Si la configuration n'est pas identifiée, alors un message d'avertissement ou d'erreur, suivant le cas rencontré, informe l'utilisateur. Un exemple de message d'avertissement est "transistor bloqué" lorsque la grille d'un transistor enrichi est attaquée par Gnd. Un exemple de message d'erreur est "court circuit" lorsque le drain d'un transistor est attaqué par Vdd alors que sa source l'est par Gnd. Lorsqu'un tel message est généré, il est accompagné d'informations permettant à l'utilisateur d'identifier précisément le transistor au sein de la *net-list* de transistors. En effet, les noms et index des signaux de drain, source et grille sont alors donnés.

Les configurations reconnues sont celles utilisées dans les familles logiques AsGa DCFL, BDCFL, SBDCFL et SqBDCFL.

5.2.3 Conclusion

À la fin de cette première étape, tous les transistors ont été localement vérifiés. Ceux dont la configuration a été reconnue ont été "étiquetés" en vue du typage ultérieur des maillons. Ceux qui ne l'ont pas été ont engendré des messages d'erreur.

5.3 FOX : construction des cônes

Cette deuxième étape, entièrement algorithmique, a pour objectif la construction d'une *net-list* d'étages orientés appelés cônes à partir de la *net-list* de transistors.

5.3.1 Description de l'algorithme

Le principe de la construction des cônes diffère peu de celui employé par DESB (voir 3.2) ; la *net-list* de transistors est parcourue afin d'y déceler les signaux marquant les sorties de cônes. Chaque fois qu'un signal de cette nature est rencontré, un cône est construit. Rappelons qu'un tel signal se caractérise par la présence d'au moins une grille de transistor commutant. Le nom affecté au cône correspond à celui de ce signal permettant ainsi d'assurer une correspondance entre les deux *net-list*.

C'est à partir de ce signal de sortie de cône que sont recherchés tous les chemins électriques menant aux alimentations. Cette recherche s'effectue par propagation à travers les transistors, du drain vers la source, ou de l'anode vers la cathode d'une diode. Notons que l'on ne doit pas repasser par un signal déjà rencontré. Par ailleurs, comme nous l'avons vu en 4.3.3, la profondeur maximale autorisée est de deux transistors. Dans ces conditions, si l'on parvient à atteindre un signal d'alimentation, alors une branche est créée. Par contre, si l'on ne détecte pas de signal d'alimentation après un deuxième transistor, un message d'erreur en informe l'utilisateur. La fonctionnalité des branches d'un point de vue structurel est vérifiée. Ainsi les branches non fonctionnelles (typiquement les branches contenant une diode montée à l'envers) ne sont pas construites.

Une branche est donc constituée d'un ou de deux maillons. Un maillon contient un transistor ainsi que la valeur de la capacité d'interconnexion associée au signal se trouvant "le plus proche" de la sortie de cône. La branche reçoit le type de l'alimentation à laquelle elle est connectée.

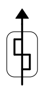
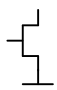
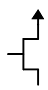
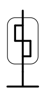
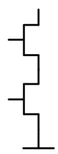
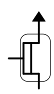
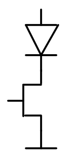
Chaque cône contient la liste des cônes qu'il a en entrée ainsi que celle des cônes qu'il a en sortie. Ces deux listes sont créées au fur et à mesure de la construction.

C'est au cours de cette étape qu'est traité le problème des ensembles de branches structurellement identiques. De telles branches doivent réunir les conditions suivantes :

- elles ont toutes le même nombre de transistors,
- les transistors commutants de même niveau de chaque branche sont de même nature et ont la même équipotentielle de grille,
- les transistors non commutants de même niveau de chaque branche sont de même nature et ont leurs équipotentielles dans une configuration identique.

Nous qualifions ces branches de branches parallèles. Elles sont marquées en conséquence et seront considérées comme une seule branche dans la suite de l’algorithme.

Pour les familles logiques qui nous intéressent, on constate que le nombre de branches contenant deux maillons est peu nombreux, la majorité des branches n'en contenant qu'un seul. Le nombre de structures de branche que l'on peut rencontrer est donc limité. En ce qui concerne la classification que nous avons présenté au chapitre 3, on en dénombre sept. Précisons que ce sur-typage est effectué lors de la construction des cônes, dès qu'une branche est ajouté au cône en cours de construction. Voici les sept types de branches :

-  : branche de type 1 (UP).
-  : branche de type 2 (DOWN).
-  : branche de type 3 (UP).
-  : branche de type 4 (DOWN).
-  : branche de type 5 (DOWN).
-  : branche de type 6 (UP).
-  : branche de type 7 (DOWN).

5.3.2 Conclusion

À l'issu de cette étape tous les cônes pouvant être construits l'ont été et ceux-ci peuvent être considérés comme structurellement corrects. Les problèmes d'ordre structurels rencontrés ont été signalés.

On a transformé la *net-list* de transistors en une *net-list* de cônes orientés.

5.4 OSR : typage des cônes

Une *net-list* de cônes composés de branches ayant au plus deux maillons et issue de la *net-list* de transistors a été créée à l'étape précédente. À ce stade, seule une analyse fine de la structure des cônes permet de les typer.

5.4.1 Présentation du module

Il s'agit, comme nous l'avons vu au chapitre précédent, de reconnaître en chaque cône une des onze formes de la classification afin de leur attribuer le type correspondant. La reconnaissance est faite par comparaison structurelle entre la forme du cône analysé et la forme d'un des étages de la classification.

Lorsqu'un cône a été reconnu, l'algorithme passe au cône suivant. En effet, la reconnaissance est stricte en ce sens qu'un cône ne peut à la fois être isomorphe à deux formes de la classification. L'ordre dans lequel les formes de la classification sont comparées à la forme d'un cône n'a donc pas d'importance. On notera cependant, que placer les formes de la classification dans un ordre correspondant aux formes les plus fréquemment rencontrées a pour effet d'améliorer les temps de traitement.

Rappelons qu'en ce qui nous concerne deux degrés de liberté sont autorisés lors des comparaisons. Le premier correspond au nombre de branches parallèles que peuvent contenir les cônes et qui peut être quelconque. Le second correspond au nombre d'entrées des cônes qui peut être variable pour certains d'entre eux (cf. 4.3.4). La taille des transistors comme degré de liberté n'a pas été mentionnée ci-dessus car les dimensions des transistors n'interviennent pas dans la reconnaissance.

5.4.2 Principe de la reconnaissance

Un type de cône peut être caractérisé par :

- les types de branches qu'il contient,
- le nombre de branches dans chaque type.

Afin d'optimiser le premier contrôle, les sept types de branches ont été codés sur des entiers mutuellement exclusifs bit à bit. Ainsi, en effectuant un ou inclusif bit à bit sur l'ensemble des types de branches d'un cône, on obtient "signature" caractéristique du type du cône. Il ne reste plus alors qu'à faire une comparaison avec la signature de chaque forme de la classification afin d'effectuer le premier contrôle.

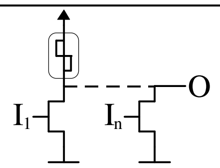


Figure 5.1 : étage NOR

Signature : (1,2).
Relation : néant.

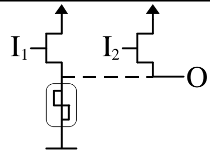


Figure 5.2 : étage OR

Signature : (3,4).
Relation : néant.

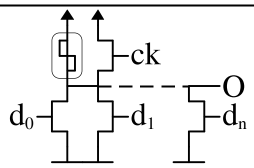


Figure 5.3 : étage NOR à Précharge

Signature : (1,3,2).
Relation : une seule branche de type 3.

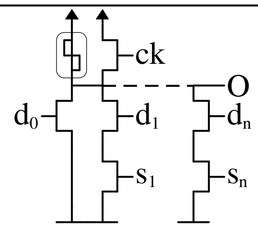


Figure 5.4 : étage BUS à Précharge

Signature : (1,3,2,5).
Relations : une seule branche de type 3 et une seule branche de type 2.

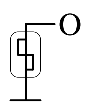


Figure 5.5 : étage Mise à Zéro

Signature : (4).
Relation : néant.

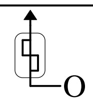


Figure 5.6 : étage Mise à Un

Signature : (1).
Relation : néant.

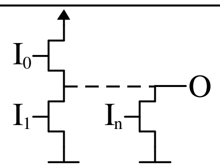


Figure 5.7 : étage SBE

Signature : (3,2).
Relation : une seule branche de type 2.

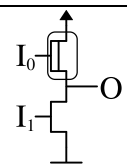


Figure 5.8 : étage SBD

Signature : (6,2).
Relation : une seule branche de type 6 et une seule branche de type 2.

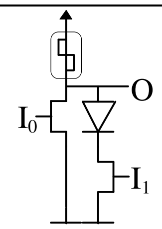


Figure 5.9 : étage Squirt

Signature : (1,2,7).
Relations : une seule branche de type 2 et une seule branche de type 7.

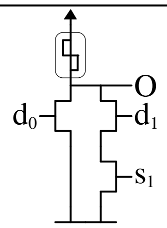


Figure 5.10 : étage Demi-Latch

Signature : (1,2,5).
Relations : une seule branche de type 2 et une seule branche de type 5.

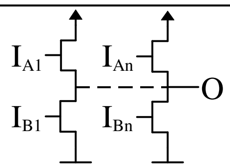


Figure 5.11 : étage Tristate

Signature : (3,2).
Relation : le même nombre de branches de type 3 et de type 2

On constate que dans la plupart des cas, la signature est suffisante pour déterminer le type du cône.

En cas d'ambiguïté (comme dans le cas des étages SBE et Tristate, qui ont la même signature) il faut compter les branches de chaque type.

Dans l'exemple que nous venons de donner, il s'agira du nombre de branches de type 4 qui doit valoir 1 dans le premier cas et qui doit être égal au nombre de branches de type 2 dans le second.

Nous présentons ci-dessous chaque cône de la classification accompagné de sa signature et des relations à vérifier. Cette dernière est représentée à la manière d'un ensemble de types ; par exemple (1,2,5) signifie que le cône considéré contient au moins une branche de type 1, au moins une branche de type 2 et au moins une branche de type 5.

5.4.3 Description de l'algorithme

La *net-list* de cônes est parcourue. Pour chaque cône rencontré le traitement suivant est effectué :

1 - Calcul de la signature du cône : un ou inclusif bit à bit sur les types des branches est réalisé pendant le parcours des branches du cône.

2 - Le cône est successivement traité par chaque routine de comparaison de forme tant qu'il n'a pas été reconnu. On rappelle qu'il y a 11 formes et donc 11 routines, chacune ayant pour tâche de reconnaître une forme particulière. Ces routines sont toutes basées sur le même modèle que voici :

- comparaison bit à bit entre la signature du cône et celle de la forme. S'il n'y a pas identité, on sort de la routine.
- Les signatures sont identiques, alors on effectue le deuxième contrôle concernant les nombres de branches pour chaque type. Les vérifications faites ici sont propres à la routine (voir 5.4.2). Si le cône a été reconnu, on le type en conséquence et on passe au prochain cône.

3 - Si le cône n'a pas été reconnu alors un message d'erreur est émis.

Notons que l'ordre dans lequel sont effectuées les routines de comparaisons n'a pas d'importance. Cependant en ordonnant celles-ci de la forme la plus fréquente à celle qui l'est le moins, on accélère le traitement.

Si le cône reconnu est relié à un connecteur externe en sortie alors celui-ci est typé ; son orientation est ainsi déterminée.

5.4.4 Conclusion

À l'issue de cette étape, tous les cônes du circuit se voient attribuer un des onze types définis au chapitre 4.2.

5.5 ESR : reconnaissance de forme au niveau cellule

À ce stade de la phase d'abstraction, les cônes ont été validés et typés. Les entités fonctionnelles englobant plusieurs cônes sont les cellules. Le traitement consiste à les reconnaître dans la *net-list* de cônes. Nous rappelons que ce traitement ne modifie en aucun cas la structure CNS mais l'enrichit de la liste des cellules reconnues dans la *net-list* de cônes.

5.5.1 Description de l'algorithme

Une cellule correspond à un ensemble précis de cônes qui sont reliés entre eux d'une manière spécifique. Un type de cône est choisi parmi ceux que contient la cellule. À chaque fois que ce cône "déclencheur" est rencontré lors du parcours de la *net-list* de cônes, la procédure de reconnaissance spécifique à cette cellule est déclenchée.

1 - Lorsque le cône "déclencheur" est un cône partagé par plusieurs cellules, la construction de toutes les cellules en question doit être entreprise.

2 - Lorsqu'une cellule contient un étage spécifique, il est choisi comme "déclencheur". Il n'y a, dans ce genre de situation, aucune ambiguïté ; lorsque l'on rencontre un étage spécifique, on doit construire la cellule qui le contient. On rappelle que les étages spécifiques n'ont pas de comportement propre et qu'ils ne peuvent pas être présents "seuls" dans le circuit.

3 - Lorsqu'une cellule ne contient pas d'étages spécifiques, n'importe quel cône peut être choisi comme "déclencheur". Chaque fois qu'un cône de ce type va être rencontré lors du parcours de la *net-list* de cônes, une procédure de reconnaissance va être déclenchée. Souvent "pour rien". Afin de limiter d'inutiles tentatives de reconnaissance, on choisit comme "déclencheur" le cône que l'on rencontre le moins souvent.

À partir du cône "déclencheur" on explore la *net-list* de cônes par les entrées et sorties de cônes afin de vérifier les types de tous les cônes composant la forme, le nombre d'entrées de chacun d'entre eux, et surtout la manière dont ils sont interconnectés. Au niveau de ces éléments, il ne peut y avoir aucune variabilité.

5.5.2 Conclusion

La méthode employée pour la reconnaissance des cellules est suffisante tant que le nombre d'interconnexions entre les cônes qu'elles contiennent reste peu élevé. Elle n'a posé aucun problème pour les formes de cellules présentées dans cette thèse (voir la classification en 4.2.3).

5.6 Conclusion

Nous avons présenté dans ce chapitre la méthode d'abstraction au cours de laquelle la structure CNS contenant cônes et cellules est créée. Cette structure appelée "figure cns" est le point de départ des vérifications que l'on désire entreprendre dans l'environnement de vérification VEGA. Il faut noter que tous les algorithmes mis en œuvre ont une complexité linéaire avec le nombre de transistors. La force de la méthode réside dans le mariage judicieux de techniques purement algorithmiques, et de techniques de reconnaissances de forme, s'appuyant sur les caractéristiques de la circuiterie DCFL.

6. Vérification fonctionnelle, électrique et temporelle

6.1 Introduction

La structure CNS, dont la construction a été présentée au chapitre précédent, va servir de base aux différentes vérifications. Nous abordons en premier lieu la vérification électrique, vient ensuite la vérification fonctionnelle et enfin la vérification temporelle. Nous verrons, pour chacune d'entre elles, les bénéfices apportés par les choix méthodologiques exposés dans cette thèse.

6.2 EVM : module de vérification électrique

Parmi les règles électriques présentées au chapitre 3, on constate que certaines s'appliquent à des cônes particuliers et que d'autres font intervenir des relations entre cônes. Nous avons donc émis l'hypothèse selon laquelle toute règle nécessitant d'être vérifiée appartient à l'une de ces deux catégories [Rem97]. Rappelons qu'il s'agit de règles permettant la vérification de certains aspects du comportement électrique statique et dynamique d'un circuit. Elles font appel aux dimensions des transistors, aux capacités d'interconnexions ainsi qu'à des paramètres caractéristiques de la technologie cible obtenus une fois pour toute par simulation SPICE. Nous présentons dans ce qui suit ces deux catégories de règles.

6.2.1 Règles locales à un cône

Pour vérifier ces règles, seules les caractéristiques internes de certains cônes sont requises. Elles ne concernent que le comportement statique du cône qui, rappelons-le est essentiel pour ce type de circuiterie. Un exemple d'une telle règle est le suivant :

règle 1 : le rapport de dimensionnement β des cônes NOR doit valoir 10 à 10 % près.

Cette règle fait intervenir les largeurs et longueurs des transistors enrichi et déplété constituant chaque cône NOR.

Nous avons choisi de créer une procédure de vérification électrique par type de cône à vérifier. Chacune regroupe l'ensemble des règles locales à vérifier pour un type de cône donné.

6.2.2 Règles faisant intervenir plusieurs cônes

Ces règles, qui sont majoritaires, font notamment appel à la charge vue de la sortie du cône en cours de vérification et font donc intervenir plusieurs cônes. Autrement dit, la vérification de ces règles nécessite les caractéristiques de dimensionnement d'un cône "source" et de plusieurs cônes "destination". La règle suivante illustrera notre propos :

règle 11 : aucun cône NOR attaquant un cône OR ne doit attaquer simultanément un autre cône NOR.

Comme précédemment nous avons créé une procédure de vérification électrique par type de cône à vérifier. Dans ces conditions notons que chaque procédure regroupe, pour un type de cône donné, à la fois l'ensemble des règles locales à vérifier (s'il y en a) et l'ensemble des règles faisant intervenir plusieurs cônes pour lesquelles le cône considéré est le cône source.

6.2.3 Remarques

1 - Une autre catégorie de règles n'a pas été mentionnée jusqu'ici ; les règles 6 et 7 en sont un exemple (cf. chapitre 3). Les règles de cette catégorie sont globales, la 6 et la 7 par exemple font référence aux transistors de passage ainsi qu'aux transistors en série. Elles sont en fait vérifiées par construction puisque la validation structurelle de la *net-list* découle de l'emploi d'une méthode de reconnaissance de forme. Par conséquent, nous ne tenons pas compte de telles règles au niveau de ce module.

2 - La *net-list* de transistors, à partir de laquelle est abstrait la structure CNS en vue de la vérification électrique, a subi un traitement spécifique avant la phase d'abstraction. En effet, les calculs effectués durant la phase de vérification électrique s'appuient sur les dimensions efficaces des transistors ainsi que sur les capacités totales (capacités d'interconnexion, de diffusion et de grille). Ce traitement fait appel à une librairie de fonctions nommée ELP également développée au laboratoire [Kia94]. Ces fonctions se basent sur un fichier de paramètres technologiques afin d'effectuer les calculs nécessaires sur la *net-list* de transistors.

6.2.4 Récapitulatif

Voici pour résumer, la liste des règles vérifiées à ce jour et présentées au chapitre 3 que nous classons par catégorie et par type de cône concerné :

- ♦ règles locales à un type de cône :
  - cônes de type NOR
    - ∗ statiques : 1 et 5
  - cônes de type OR
    - ∗ statique : 14

- ♦ règles faisant intervenir plusieurs types de cônes :
  - à partir d'un cône de type NOR
    - * statiques : 2, 3, 4, 11, 17, 23
    - * dynamique : 8
  - à partir d'un cône de type OR
    - * statiques : 9, 10, 12
    - * dynamique : 13
  - à partir d'un cône de type SBE
    - * statiques : 15, 16, 18
    - * dynamiques : 19, 20
  - à partir d'un cône de type SBD
    - * statiques : 21, 22, 24

6.2.5 Conclusion

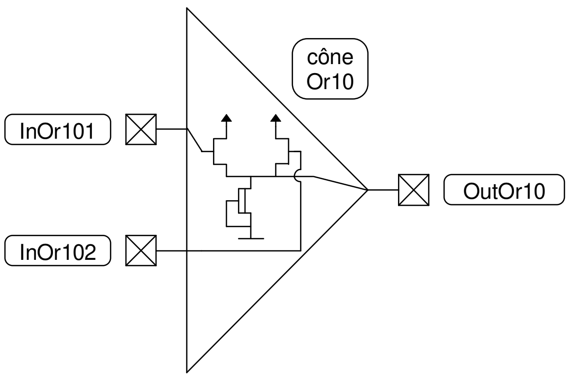
Les vérifications électriques devant être effectuées s’adressent soit à un cône seul, soit à un cône et ceux avec lesquels il interagit. La *net-list* de cônes de la structure CNS se prête bien à ces vérifications puisque que l'on peut associer à un type de cône une procédure de vérification. Il est alors possible de rajouter de nouvelles procédures accompagnant la vérification de nouveaux types de cônes. Mais aussi d'enrichir les procédures existantes avec de nouvelles règles à vérifier. Enfin rappelons que ces règles peuvent être indifféremment de nature statique ou dynamique.

6.3 BXM : module d’abstraction comportementale

On se propose de générer le modèle VHDL de niveau RTL du circuit à partir de sa structure CNS. Celle-ci comporte des cônes simples, des cônes spécifiques ainsi que des cellules. Les cônes spécifiques appartiennent nécessairement à une cellule et certaines d'entre elles peuvent partager un même cône. Nous rappelons que le modèle VHDL généré est de type "data flow" et "zéro délai" et que nous le générons en effectuant un parcours sur la structure CNS. Voyons comment va se dérouler cette abstraction dans le cas des cônes simples, des cônes spécifiques, des cellules ne partageant aucun cône et enfin de celles partageant au moins un cône. Nous ne considérons que les signaux internes dans un premier temps.

6.3.1 Les cônes simples

Nous avons vu que les cônes simples correspondent à des ensembles de transistors ayant une fonctionnalité déterminée. À chacun d'eux peut être associée une description comportementale. Le comportement associé à un cône entrant dans la composition d'une cellule n'est pas pris en compte puisqu'on ne s'intéresse qu'au comportement global de la cellule. Ainsi, à chaque fois qu'un cône simple n'appartenant à aucune cellule est rencontré, on génère l'affectation concurrente qui est associée à son type en remplaçant les noms des paramètres formels par ceux des signaux de la *net-list*. On rappelle que le nom d'un cône correspond à celui du signal initialement présent dans la *net-list* et associé à sa sortie. La figure précédente montre ce qui se passe lorsqu'un cône de type OR est rencontré.



Modèle comportemental VHDL d'un cône de type OR :

```
ENTITY OR IS
        PORT (
        I1 : IN BIT;
        ...
        In : IN BIT;
        O : OUT BIT
        );
END OR;

ARCHITECTURE behaviour_data_flow OF OR IS
BEGIN
O <= (I1 OR ... OR In)
END;
```

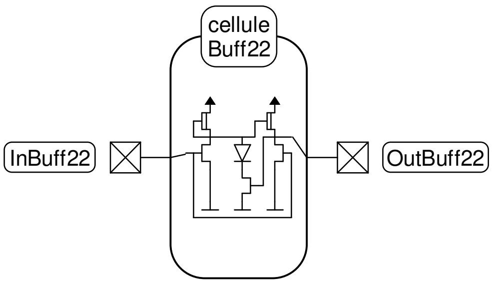
Modèle comportemental VHDL du cône Or10 :

```
ENTITY Or10 IS
        PORT (
        InOr101 : IN BIT;
        InOr102 : IN BIT;
        OutOr10 : OUT BIT
        );
END Or10;

ARCHITECTURE behaviour_data_flow OF Or10 IS
BEGIN
OutOr10 <= (InOr101 OR InOr102)
END;
```

6.3.2 Les cellules ne partageant pas de cônes

Nous considérons ici les cellules ne partageant aucun cône avec d'autres cellules. Nous avons vu que les cellules correspondent à des ensembles d'au moins deux cônes ayant une fonctionnalité déterminée. Une description comportementale va donc être associée à chaque cellule. Ainsi, lorsqu'une cellule est rencontrée, on génère l'affectation concurrente qui est associée à son type en remplaçant les noms des paramètres formels par ceux des signaux connectés aux entrées et sorties de la cellule. La figure suivante montre ce qui se passe lorsque qu'une cellule de type Squirt Buffer est rencontrée :



Modèle comportemental VHDL d'une cellule de type Squirt Buffer :

```
ENTITY Squirt Buffer IS
          PORT (
          I : IN BIT;
          O : OUT BIT
          );
END Squirt Buffer;

ARCHITECTURE behaviour_data_flow OF Squirt Buffer IS
BEGIN
O <= NOT (I)
END;
```

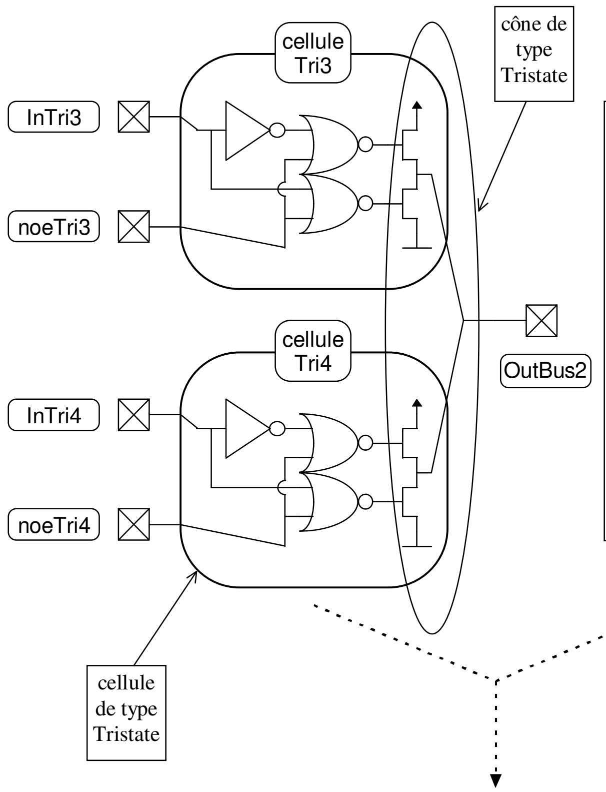
Modèle comportemental VHDL de la cellule Buff22 :

```
ENTITY Buff22 IS
          PORT (
          InBuff22 : IN BIT;
          OutBuff22 : OUT BIT
          );
END Buff22;

ARCHITECTURE behaviour_data_flow OF Buff22 IS
BEGIN
OutBuff22 <= NOT (InBuff22)
END;
```

6.3.3 Les cellules partageant un cône

Le cas des cellules partageant un même cône est traité de la même manière que précédemment lorsqu'elles n'ont aucun signal de sortie en commun. Par contre, lorsqu'elles ont un signal de sortie en commun, un traitement supplémentaire est effectué lors de la récupération des modèles comportementaux des cellules. Comme dans l'exemple ci-dessous, les expressions régissant chaque émetteur tristate sont fusionnées lors de l'incorporation des modèles des cellules dans celui du circuit :



Modèle comportemental VHDL d'un signal de bus attaqué par une cellule de type Tristate :

```
ENTITY BusSig IS
        PORT (
        I : IN BIT;
        noe : IN BIT;
        O : OUT MUX_BIT BUS
        );
END BusSig;

ARCHITECTURE behaviour_data_flow OF BusSig IS
BEGIN
        lbl1 : BLOCK (noe = '0')
        BEGIN
            O <= GUARDED NOT (I);
        END BLOCK lbl1;
END;
```

Modèle comportemental VHDL du signal OutBus2 :

```
ENTITY OutBus2 IS
        PORT (
        InTri3 : IN BIT;
        noeTri3 : IN BIT;
        InTri4 : IN BIT;
        noeTri4 : IN BIT;
        OutBus2 : OUT MUX_BIT BUS
END OutBus2;

ARCHITECTURE behaviour_data_flow OF OutBus2 IS
BEGIN
        lbl1 : BLOCK (noeTri3 = '0')
        BEGIN
            OutBus2 <= GUARDED NOT (InTri3);
        END BLOCK lbl1;
        lbl2 : BLOCK (noeTri4 = '0')
        BEGIN
            OutBus2 <= GUARDED NOT (InTri4);
        END BLOCK lbl2;
END;
```

6.3.4 Le cas des signaux externes

Deux cas sont à considérer ; les signaux d'entrée et les signaux de sortie et d'entrée/sortie. Un connecteur est présent sur chacun de ces signaux et chaque connecteur est représenté par un cône au niveau de la structure CNS. Ce cône est appelé cône d'entrée ou cône connecteur dans le premier cas. Dans le deuxième cas, il peut s'agir de n'importe quel cône de la classification.

Un cône d'entrée correspond donc à un connecteur d'entrée. Sa description comportementale se résumant à un port d'entrée, elle est écrite de manière ad hoc dans le modèle du circuit.

Lorsqu'il s'agit d'un cône ou d'une cellule attaquant un connecteur externe, d'une part le port correspondant est ajouté dans le modèle du circuit. D'autre part, l'expression régissant le cône ou la cellule est inséré dans le modèle comportemental du circuit comme cela a été décrit aux sections précédentes.

6.3.5 Conclusion

Ce second module montre la bonne adéquation entre la structure CNS faite de blocs fonctionnels précaractérisés et la génération d'un modèle comportemental du circuit associé. L'ajout de nouveaux types de cônes ou cellules se fait par ajout d'une procédure correspondante contenant un modèle comportemental ainsi qu'une routine de recherche et remplacement des paramètres.

6.4 TAGA : préparation à la vérification temporelle

L'analyse temporelle statique de niveau transistor effectuée par l'outil TTV du laboratoire [Dio97] se base sur une représentation CNS enrichie des temps de propagation et de transition de chaque cône. Il nous incombe donc de modéliser et de calculer ces temps afin de fournir une représentation CNS conforme aux besoins de TTV. C'est l'objectif de ce troisième module qui est décrit ci-dessous.

6.4.1 Considérations préliminaires

La caractérisation temporelle doit être effectuée sur la *net-list* de cônes. Chaque cône est caractérisé dans son contexte d'utilisation, en prenant en compte la pente du signal d'entrée et la capacité totale du signal de sortie. Puisque les délais élémentaires sont attachés aux cônes, le concept de cellule n'a pas, ici, de signification et est donc ignoré.

6.4.2 Modèle temporel des cônes

Chaque cône possède des caractéristiques électriques qui lui sont propres, liées à sa structure et à son dimensionnement. Ce qui différencie deux cônes d'un même type, provient des degrés de liberté autorisés, c'est-à-dire des dimensions des transistors. Il a donc été envisagé de modéliser le comportement temporel suivant le type de cône, en fonction de ces caractéristiques. Il ressort du travail de J. Pistorius, dans le cadre de son DEA, qu'il est possible d'attribuer à chaque type de cône de la classification un modèle temporel à partir duquel les temps de propagation et le temps de front de tout cône peuvent être calculés [Pis96].

La forme générale des temps de propagation est la suivante :

$tp = r(W,W').C_L + s(W,W').F + constante$ où

- r(W,W') est un coefficient qui dépend des largeurs et longueurs des transistors des branches UP ou DOWN. Il correspond à une résistance équivalente et caractérise l'influence de la capacité de sortie.
- C_L correspond à la capacité de charge, somme de la capacité d'interconnexion et des capacités de grilles des transistors attaqués vus sur la sortie du cône en question.
- s(W,W') est un coefficient qui dépend des largeurs et longueurs des transistors des branches UP ou DOWN. Il caractérise l'influence du front d'entrée.
- F correspond à la durée du front montant ou descendant en entrée de branche. Il dépend de la capacité d'interconnexion sur cette entrée.
- La constante permet d'apporter une correction éventuelle. Elle correspond aux termes non exprimés (simplifications et termes du deuxième ordre).

Cela a donc été réalisé pour chacun des cônes de la classification. Ces résultats ainsi que le détail de la méthode employée pour les obtenir sont donnés dans l'annexe 5. On trouvera également dans cette annexe les résultats obtenus lors de la phase de validation de TAGA. À titre d'illustration, nous proposons dans ce qui suit l'étude des deux cônes entrant dans la composition de la cellule Super Buffer :

6.4.3 Modèle du cône NOR

Le cône NOR illustré ci-contre possède une entrée E et une sortie S. Les temps de propagation qui le concerne sont ceux de niveau logique haut vers niveau logique bas et de niveau logique bas vers niveau logique haut. Soit tp_{hl} et tp_{lh} ces temps de propagation, donnés en pico seconde. Les temps de transition de montée et de descente sont représentés par t_r et t_f et sont donnés aussi en pico seconde.

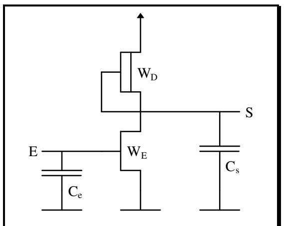


Figure 6.1 : cône NOR

$tp_{hl} = r.C_s + s.F + c$ avec

$r = 1/(0,0385.W_E + 0,9433)$ kΩ
$s = 0,5816$
$c = 100$ ps

$tp_{lh} = r'.C_s + c'$ avec

$r' = 1/(0,0603.W_E - 0,1637)$ kΩ

c' = 20 ps

$t_f = r''.C_s + c''$ avec

$r'' = 1/(0{,}3149.W_E + 0{,}5030)$ kΩ
c'' = 80 ps

$t_r = r'''.C_s + c'''$ avec

$r''' = 1/(0{,}0976.W_E - 0{,}2710)$ kΩ
c''' = 35 ps

C_s est en fF, W_E est en µm.

Remarque : la dépendance en W_D n'apparaît pas, car elle est exprimée dans le rapport de dimensionnement $W_D = 4{,}5.W_E$.

6.4.4 Modèle du cône SBE

Le cône SBE illustré ci-contre possède deux entrées E_1 et E_2 ainsi qu'une sortie S. Les temps de propagation qui le concerne sont ceux de niveau logique haut vers niveau logique haut et de niveau logique haut vers niveau logique bas. Soit tp_{hh} et tp_{hl} ces temps de propagation, donnés en pico seconde. Les temps de transition de montée et de descente t_r et t_f sont donnés aussi en pico seconde.

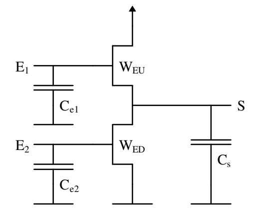


Figure 6.2 : cône SBE

$tp_{hh} = r.C_s + s.F + c$ avec

$r = 1/(0{,}1169.W_E + 1{,}4800)$ kΩ
$s = 1/(0{,}0110.W_E + 0{,}3640)$
c = 100 ps

$tp_{hl} = r'.C_s + s'.F + c'$ avec

$r' = 1/(0,\ 1157.W_E + 1,\ 425)$ kΩ
$s' = - 0{,}0366.W_E + 0{,}8213)$
c' = 130 ps

$t_f = r''.C_s + c''$ avec

$r'' = 1/(0{,}4883.W_E - 0{,}3373)$ kΩ
c'' = 180 ps

$t_r = r'''.C_s + c'''$ avec

$r''' = 1/(1{,}2600.W_E - 2{,}5470)$ kΩ
$c''' = 250$ ps

C_s est en fF, W_E est en µm.

Remarques : la dépendance en W_D n'apparaît pas, car elle est exprimée dans le rapport de dimensionnement $W_D = 4{,}5.W_E$.

6.4.5 Conclusion

Comme le montre cette section, l'analyse temporelle statique peut donc être employée pour la circuiterie AsGa que nous utilisons. Elle est basée sur une modélisation temporelle de chaque type de cône permettant le calcul des délais et des temps de transition nécessaire aux outils en aval.

6.5 Conclusion

Ce chapitre a présenté l'aspect vérification de l'environnement VEGA consacré aux circuits AsGa. Nous avons vu qu'il est possible d'automatiser les phases les plus laborieuses de ces vérifications d'ordre électrique, fonctionnelle et temporelle. Enfin, on notera que l'environnement permet une évolution de ces vérifications à d'autres règles en ce qui concerne les vérifications électriques, à d'autres cônes et cellules en ce qui concerne la vérification fonctionnelle et temporelle.

Le typage réalisé dans la phase d'abstraction et de construction des cônes et cellules est fortement utilisé par les différents modules de vérification.

Cette utilisation démontre a posteriori la pertinence de la classification proposée.

7. Résultats et performances

7.1 Introduction

Afin de valider l'environnement VEGA, nous avons travaillé, dans un premier temps avec des circuits se réduisant à un seul cône ou une seule cellule. Pour évaluer les performances, nous avons utilisé des circuits réels de différentes complexités. Pour les circuits peu complexes, la validation des modules **fox**, **osr** et **esr** construisant la structure CNS peut être faite "à la main", en analysant le fichier image de la structure CNS. La validation du module de vérification électrique **evm** a été faite en "jouant" sur les largeurs des transistors et en faisant varier certaines valeurs du fichier de paramètres électriques. Le module de génération de modèles VHDL *data-flow*, **bxm** a été validé par preuve formelle entre les modèles générés et leurs modèles de référence.

Par la suite, l'environnement de vérification a été utilisé lors de la conception et de la mise au point de générateurs paramétrables (additionneurs, multiplieurs, etc.). Ces divers circuits ont été conçus dans le cadre de la thèse de P. Royannez [Roy96]. Si les premiers bénéficiaires de cet environnement étaient les concepteurs, nous avons bénéficier en retour d'informations qui nous ont permis de corriger et d'améliorer VEGA.

7.2 Complexité algorithmique

Nous nous proposons d'aborder à présent la question de la complexité algorithmique des différents modules de l'environnement. Nous avons en effet mis en avant, tout au long de la thèse, la complexité linéaire des algorithmes mis en œuvre dans les différents modules. Pour analyser les temps de calcul en fonction du nombre de transistors, nous avons choisi un certain nombre de circuits disponibles au laboratoire. Les temps qui sont donnés correspondent au temps "utilisateur" que nous avons obtenu au moyen de la fonction C getrusage() disponible sur la plate-forme Sun. Enfin, ces tests ont été effectués sur une machine Sun sparc 10/51.

Ces circuits sont issus des quatre générateurs paramétrables du laboratoire : le générateur d'additionneurs **clara**, le générateur de décaleurs arithmétiques **sara**, le générateur de multiplieurs **bat**, et enfin le générateur de bancs de registres **raga**. Voyons leurs caractéristiques.

7.2.1 Les additionneurs

- Le nombre de transistors varie entre 1596 pour le 28 bits et 2996 pour le 48 bits.
- Le contenu en type de cônes et de cellules est le suivant :
  - cônes simples : NOR et OR.
  - cellules : Super Buffer.

7.2.2 Les décaleurs

- Le nombre de transistors varie entre 817 pour le 12 bits et 2333 pour le 32 bits.
- Le contenu en type de cônes et de cellules est le suivant :
  - cônes simples : NOR et OR.
  - cellules : Super Buffer.

7.2.3 Les multiplieurs

- Le nombre de transistors varie entre 2622 pour le 8x8 et 34924 pour le 32x32.
- Le contenu en type de cônes et de cellules est le suivant :
  - cônes simples : NOR et OR.
  - cellules : Super Buffer.

7.2.4 Les bancs de registres

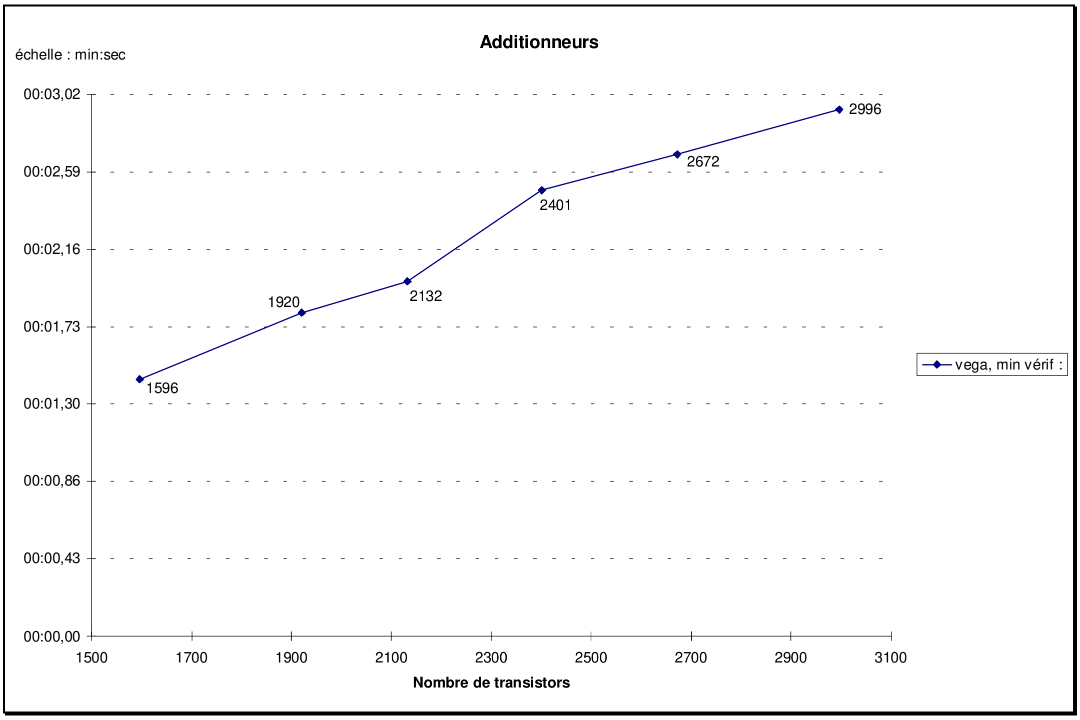
- Le nombre de transistors varie entre 4142 pour le 32 registres de 8 bits et 26130 pour le 64 registres de 32 bits.
- Le contenu en type de cônes et de cellules est le suivant :
  - cônes simples : NOR, OR et Bus à Précharge.
  - cellules : Super Buffer, Tristate, Latch et le Bus de Lecture.

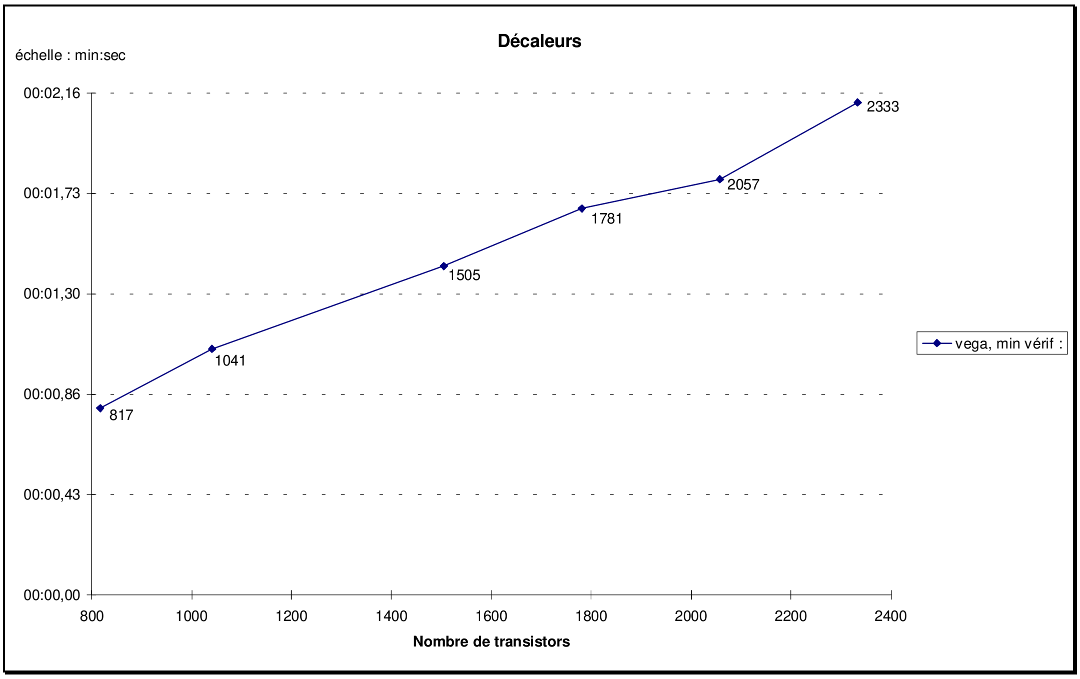
7.2.5 Démarche expérimentale

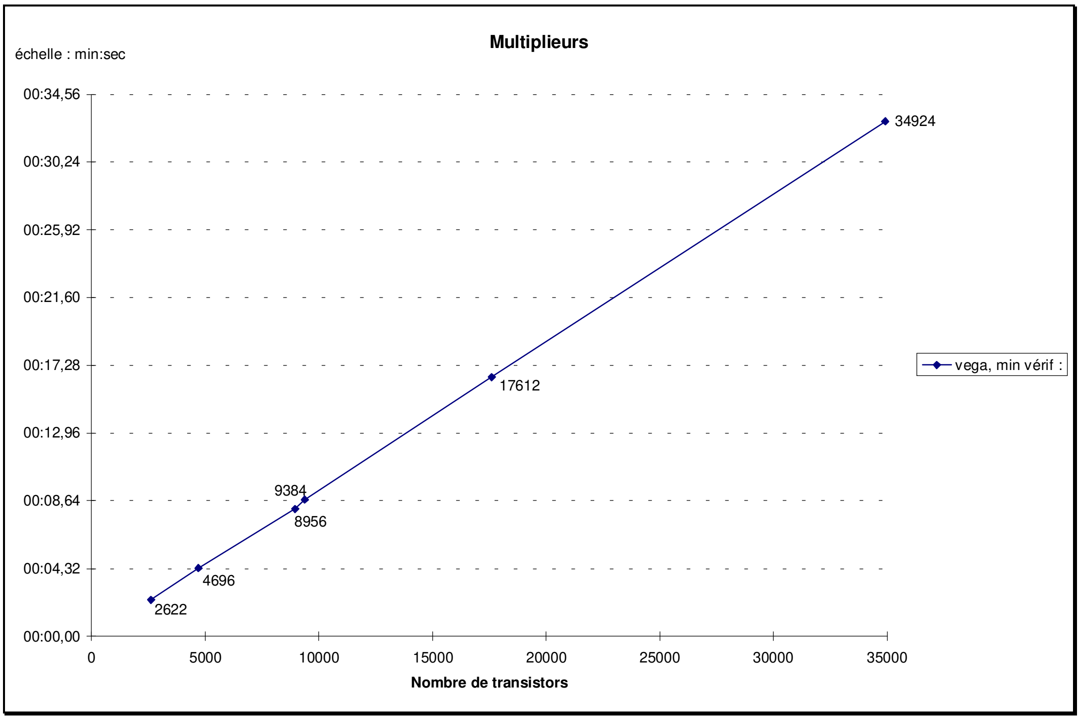
Nous avons généré 6 circuits de chaque type et avons exécuté **vega** à trois reprises sur chacun d'entre eux avec toutes les options. Trois fois, d'une part afin de vérifier la reproductibilité de l'expérience, d'autre part afin de pouvoir se prémunir d'éventuelles erreurs expérimentales ou de retranscription des résultats. Ainsi, à partir des trois jeux de résultats, nous avons utilisé la moyenne arithmétique des deux valeurs les plus proches, en rejetant donc une des trois valeurs. Nous obtenons le tableau de l'annexe 4, ordonné suivant le nombre croissant de transistors. C'est à partir de ce tableau que l'on va tirer un certain nombre de graphiques nous permettant d'interpréter plus facilement ces résultats.

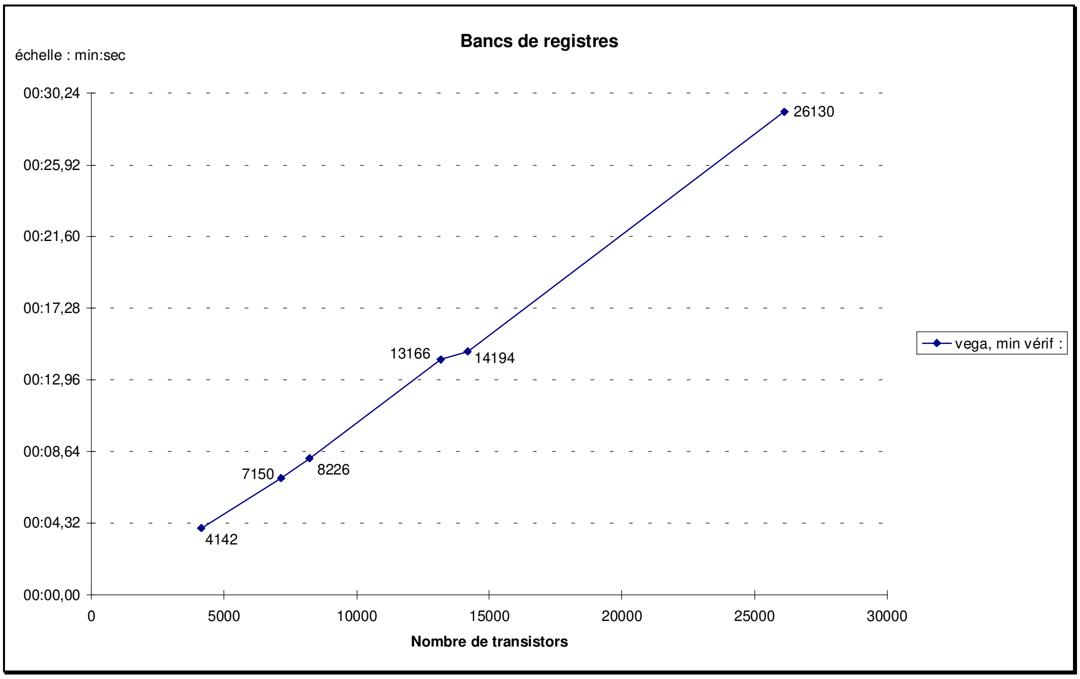
7.2.6 Interprétation des résultats

Un graphique par type de circuit a été réalisé, chacun représentant le temps d'exécution de **vega** en fonction du nombre de transistors. L'axe des abscisses donne le nombre de transistors, et l'axe des ordonnées donne le temps d'exécution de **vega** en minutes, secondes et centièmes de seconde :


Additionneurs
échelle : min:sec
00:03,02
00:02,59
00:02,16
00:01,73
00:01,30
00:00,86
00:00,43
00:00,00
1500
1700
1900
2100
2300
2500
2700
2900
3100
Nombre de transistors
1596
1920
2132
2401
2672
2996
vega, min vérif :

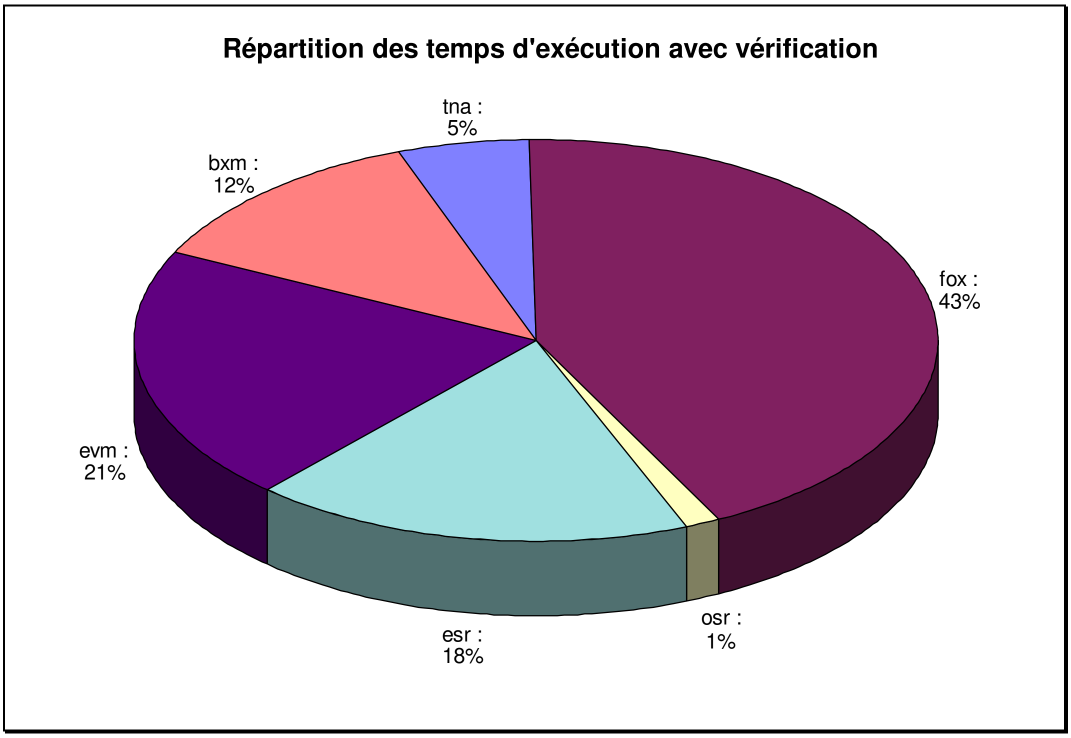

Décaleurs
échelle : min:sec
00:02,16
00:01,73
00:01,30
00:00,86
00:00,43
00:00,00
800
1000
1200
1400
1600
1800
2000
2200
2400
Nombre de transistors
817
1041
1505
1781
2057
2333
vega, min vérif :





Comme nous pouvons le constater, ces graphiques rendent bien compte de la complexité linéaire de **vega**.

L'analyse des temps passés dans les différents modules de la chaîne de vérification montre que la phase d'abstraction représente à peu près les deux tiers du temps total.



Ce graphique révèle que le module de la phase d'abstraction **fox** est prépondérant par rapport aux autres modules. Il reste néanmoins peu élevé par rapport aux appels externes dont la fonction la plus coûteuse est le chargement du circuit.

7.3 Conclusion

Les mesures de performances présentées dans ce chapitre démontrent clairement que la méthode proposée permet de traiter des circuits de complexité significative : un bloc de 35000 transistors est traité en 33 secondes.

Nous avons en particulier vérifié expérimentalement que tous les algorithmes mis en œuvre tant dans la phase d'abstraction que dans les étapes de vérification avaient une complexité linéaire.

8. Conclusion et perspectives

Nous avons présenté au cours de cette thèse un environnement d'aide à la conception des circuits AsGa MESFET. Il permet de vérifier ces circuits de manière électrique, fonctionnelle et temporelle.

Cet environnement vient s'insérer dans la chaîne de CAO Alliance développée au LIP6 de l'Université Pierre et Marie Curie. Grâce à ces nouveaux composants, Alliance offre maintenant la possibilité de concevoir des circuits numériques AsGa. Des générateurs paramétrables d'additionneurs, de décaleurs, de multiplieurs ainsi que de bancs de registres ont été développés avec succès. Une bibliothèque dédiée à la synthèse logique a également été développée, elle contient une trentaine de cellules précaractérisées. La famille logique et la technologie ciblées à l'heure actuelle sont respectivement la DCFL, et la technologie de Vitesse 0.8 μ. Cependant, l'environnement peut être configuré pour traiter d'autres familles logiques et d'autres technologies.

Cet environnement possède deux parties distinctes : une partie concerne la phase d'abstraction, elle se compose de 4 modules distincts. La deuxième partie concerne les phases de vérifications électrique, fonctionnelle et temporelle.

L'efficacité de la méthode d'abstraction proposée tient au mariage judicieux de méthodes purement algorithmiques (pour la construction des cônes) et de méthodes de reconnaissance de formes (pour le typage des cônes et des cellules). Ces techniques, héritées en partie de l'outil de désassemblage DESB, dédier au CMOS, ont influencé en retour la conception d'un nouvel outil dédié à l'abstraction fonctionnelle des circuits CMOS : en particulier, la structure de données CNS est actuellement utilisée pour le CMOS.

Au niveau de la création de la structure CNS, il serait souhaitable de se doter à terme d'un langage de description de cellules. Cela rendrait l'introduction de nouvelles formes plus simples.

Annexe 1 : la structure de données CNS

1. Présentation

La structure de données CNS (Cône Netlist Stucture) permet de représenter tout circuit sous la forme d'une *net-list* d'étages orientés. Cette représentation en *net-list* d'étages orientés est indépendante de toute technologie. Cette structure de données est écrite en langage C et comprend un certain nombre d'éléments. L'élément central est le cône qui correspond à un étage orienté. Chaque élément est représenté au moyen d'une *structure C*. On trouve le **maillon**, la **branche**, le **cône**, les **cellules** et pour finir la **figure cns**. Ce qui suit résume les caractéristiques essentielles de la structure CNS et définit la terminologie employée. Une documentation anglaise détaillée, écrite par A. Lester, et s'adressant aux programmeurs, est disponible au laboratoire [Les97].

2. Initialisation de l'environnement

Avant toute utilisation de la structure de données un appel de la fonction **cnsenv()** est indispensable. Cette fonction permet d'une part, l'initialisation d'un certain nombre de variables et d'autre part, la mise en place de la gestion mémoire des éléments de la structure. La gestion se fait par allocation par page au moyen d'un ensemble de fonctions propres à chaque élément. Elle est transparente pour l'utilisateur.

3. Le maillon

Il est l'élément le plus petit de la structure. Il correspond à l'association d'un transistor ou d'un connecteur avec une capacité. Dans le cas du transistor, la capacité en question correspond à celle vue de son connecteur le plus "éloigné" de toute alimentation. Dans le cas du connecteur, cette capacité est celle vue du signal sur lequel est monté le connecteur. Un *type* est associé à chaque maillon. Ce *type* le caractérise de manière unique. S'il est "typé" *pull-up* par exemple, cela signifie que ce maillon contient un transistor AsGa MESFET déplété ayant son drain relié à Vdd et ayant sa grille et sa source reliées entre elles.

4. La branche

Elle comprend l'ensemble des maillons par lesquels un chemin électrique peut s'établir entre un signal donné et une alimentation ou un connecteur externe. Le signal en question correspond obligatoirement à une sortie d'étage. Un *type* est associé à chaque branche. Ce *type* donne des informations quant à la nature de la branche (par exemple *Vdd* qui signifie que celle-ci est reliée à Vdd).

5. Le cône

L'élément central de la structure est le cône, il correspond à un étage orienté. Tout signal sur lequel se trouve au moins une grille de transistor commutant correspond à une sortie de cône ou "tête de cône". Un cône porte le nom du signal sur lequel il est construit (son signal de sortie). Toutes les branches partant de ce signal appartiennent au cône. Celui-ci contient la liste des cônes l'attaquant ainsi que la liste de ceux qu'il attaque. Il contient également la liste des cellules auxquelles il appartient. Deux *types* sont associés à chaque cône. Le premier, facultatif, renseigne sur des caractéristiques qui sont indépendantes de toute technologie. Le second renseigne sur des caractéristiques qui sont propres à une technologie donnée.

6. La cellule

Une cellule correspond à l'association d'au moins deux cônes. L'ensemble des cônes composant une cellule forme un bloc fonctionnel complexe ne pouvant s'exprimer au moyen d'un cône unique.

7. La figure cns

Elle représente le circuit dans sa globalité sous sa forme désassemblée et porte son nom. La figure cns contient la liste des transistors du circuit ainsi que la liste de ses connecteurs externes. De même que la liste des cônes et des cellules qui composent le circuit.

8. La bibliothèque de fonctions

Les fonctions permettant la manipulation des éléments de la structure sont au nombre de 80. Elles sont regroupées en une seule bibliothèque. Ces fonctions permettent d'accéder à tous les niveaux de la structure. Ainsi, au niveau le plus bas, on trouve des fonctions permettant la gestion mémoire par page des éléments de la structure. Au niveau le plus haut on trouve des fonctions de mise au point, d'écriture (driver) et de lecture (parser) de la structure dans un fichier et enfin de génération de statistiques (**cnsstat()**). Un exemple d'informations statistiques est fourni en section 12.

9. Le driver

Cette fonction permet de générer dans un fichier lisible une image de la structure CNS se trouvant en mémoire. Un exemple est donné à la section 11.

10. Le parser

Cette fonction permet de charger en mémoire une structure CNS préalablement sauvegardée dans un fichier par le driver.

11. Exemple de fichier CNS

Il a été généré au moyen du driver. L'exemple donné correspond à la cellule AsGa MESFET Bus de Lecture.

```
(CNS V2.01 edmesfet al read vega Tue Oct 28 18:30:58 1997

(SCALE
(X 100))

(CONNECTORS
(1 output O)
(2 ck2 I)
(3 ck1 I)
(4 sel1 I)
(5 d1 I)
(6 sel0 I)
(7 d0 I)
(8 vdd Vdd)
(9 vss Vss))

(TRANSISTORS
(1 8 3600 3450 300 81 1000 1000 200 200 E)
(2 puller 1670 2900 47 221 800 1100 65513 65496 D)
(3 10 3600 3450 300 81 1000 1000 200 200 E)
(4 7 3600 3450 300 81 1000 1000 200 200 E)
(5 puller 1670 2900 47 221 800 1100 65513 65496 D)
(6 6 3600 1650 700 81 1800 1800 200 200 E)
(7 5 3600 1650 700 81 1800 1800 200 200 E)
(8 4 3600 1650 700 81 1800 1800 200 200 E)
(9 3 3600 1650 700 81 1800 1800 200 200 E)
(10 2 3600 1650 700 81 1800 1800 200 200 E))

(CELLS

(1 Read
(8 10)))

(CONES

(2 d0 Ext -  0 0 0 0
(INPUT
(Ext 7))
(OUTPUT
(Cone 10))
(BRANCHES
(1 Ext
(In 7 0.01))))

(3 sel0 Ext -  0 0 0 0
(INPUT
(Ext 6))
(OUTPUT
(Cone 10))
(BRANCHES
(1 Ext
(In 6 0.01))))

(4 d1 Ext -  0 0 0 0
(INPUT
(Ext 5))
(OUTPUT
(Cone 10))
(BRANCHES
(1 Ext
(In 5 0.01))))
```

```
(5 sel1 Ext -  0 0 0 0
(INPUT
(Ext 4))
(OUTPUT
(Cone 10))
(BRANCHES
(1 Ext
(In 4 0.01))))

(6 ck1 Ext -  0 0 0 0
(INPUT
(Ext 3))
(OUTPUT
(Cone 10))
(BRANCHES
(1 Ext
(In 3 0.01))))

(7 ck2 Ext -  0 0 0 0
(INPUT
(Ext 2))
(OUTPUT
(Cone 8))
(BRANCHES
(1 Ext
(In 2 0.00))))

(8 output  - Nor 0 0 0 0
(INPUT
(Cone 7)
(Cone 10))
(OUTPUT
(Cone 10)
(Ext 1))
(BRANCHES
(1 Vdd
(P D 5 0.01))
(2 Vss
(S E 3 0.01))
(3 Vss
(S E 4 0.01)))
(CELLS
(1)))

(10 x  - Pr_Bus 0 0 0 0
(INPUT
(Cone 4)
(Cone 5)
(Cone 2)
(Cone 3)
(Cone 6)
(Cone 8))
(OUTPUT
(Cone 8))
(BRANCHES
(1 Vdd
(P D 2 0.02))
(2 Vdd
(S E 6 0.02))
(3 Vss
(S E 8 0.02)
(S_c E 7 0.01))
(4 Vss
(S E 10 0.02)
(S_c E 9 0.01))
(5 Vss
(S E 1 0.02)))
(CELLS
(1)))))
```

12. Exemple d'informations statistiques

Ces informations ont été générées au moyen de la fonction **cnsstat()** et insérées dans un fichier généré par VEGA. L'exemple donné correspond à un bancs de registres AsGa MESFET 32 mots de 32 bits.

```
VEGA V3.02 edmesfet al rgf_32_32 vega Wed Nov  5 18:58:32 1997

 number of vdd connnectors....................... 1

 number of gnd connnectors....................... 0

 number of vss connnectors....................... 1

 number of external connnectors.................. 75

 number of transistors........................... 13166

 number of dynamic transistors................... 10169

 number of passive transistors................... 2997

 technology...................................... edmesfet

 power number.................................... 2

 maximum series link number...................... 2

 vdd power connector name........................ vdd

 gnd power connector name........................ gnd

 vss power connector name........................ vss

 transistor grid connector name.................. grid

 transistor source connector name................ source

 transistor drain connector name................. drain

 vdd cones....................................... 0

 gnd cones....................................... 0

 vss cones....................................... 0

 conflicting cones............................... 0

 tristate cones.................................. 0

 memsym cones.................................... 2048

 latch cones..................................... 1024

 rs cones........................................ 0

 flip-flop cones................................. 0

 cmos cones...................................... 0
```

```
dual cmos cones................................. 0
vdd degraded cones.............................. 0
gnd degraded cones.............................. 0
vss degraded cones.............................. 0
nor cones....................................... 817
or cones........................................ 4
precharged nor cones............................ 0
sbe cones....................................... 193
sbd cones....................................... 0
tristate_o cones................................ 32
squirt_o cones.................................. 0
latch_o cones................................... 2048
precharged bus cones............................ 64
zero cones...................................... 0
one cones....................................... 0
total number of cones --------------------------> 3158

super buffer cells.............................. 193
tristate cells.................................. 64
squirt buffer cells............................. 0
latch cells..................................... 1024
read cells...................................... 64
total number of cells --------------------------> 1345
```

Annexe 2 : les modèles de cônes

Remarques préliminaires :

- ↑ et ⊥ représentent respectivement les tensions d'alimentation Vdd et Gnd.
- représente le transistor MESFET enrichi.
- représente le "driven pull-up".
- et représentent respectivement un "pull-up" un "pull-down".
- représente la diode Schottky.
- Les descriptions comportementales suivantes sont simplifiées de façon à ne présenter que les informations pertinentes.

1. Le cône NOR

Modèle comportemental VHDL du cône O :

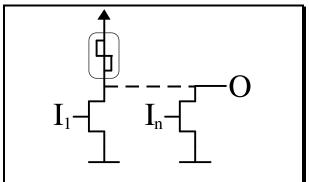


Figure A2.1 : cône NOR

```
ENTITY NOR IS
    PORT (
    I1 : IN BIT;
    ...
    In : IN BIT;
    O : OUT BIT
    );
END NOR;

ARCHITECTURE behaviour_data_flow OF NOR IS

BEGIN

    O <= NOT (I1 OR ... OR In)

END;
```

2. Le cône OR

Modèle comportemental VHDL du cône O :

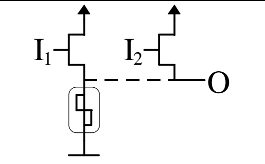


Figure A2.2 : cône OR

```
ENTITY OR IS
    PORT (
    I1 : IN BIT;
    ...
    In : IN BIT;
    O : OUT BIT
    );
END OR;

ARCHITECTURE behaviour_data_flow OF OR IS
BEGIN

    O <= (I1 OR ... OR In)

END;
```

3. Le cône NOR à Précharge

Modèle comportemental VHDL du cône O :

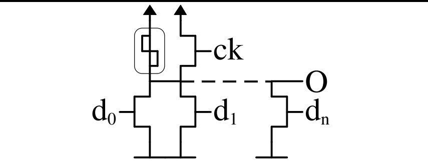


Figure A2.3 : cône NOR à Précharge

```
ENTITY NOR_P IS
    PORT (
    ck : IN BIT;
    d0 : IN BIT;
    ...
    dn : IN BIT;
    O : OUT BIT
    );
END NOR_P;

ARCHITECTURE behaviour_data_flow OF NOR_P IS

BEGIN

    lbl00 : BLOCK (d0 = '1')
    BEGIN
        O <= GUARDED '0';
    END BLOCK lbl00;

    lbl01 : BLOCK (d1 = '1')
    BEGIN
        O <= GUARDED '0';
    END BLOCK lbl01;
    ...
    lbl0n : BLOCK (dn = '1')
    BEGIN
        O <= GUARDED '0';
    END BLOCK lbl0n;

    lblpr : BLOCK (ck = '1')
    BEGIN
```

```
        O <= GUARDED '1';
    END BLOCK lblpr;

END;
```

4. Le cône BUS à Précharge

Modèle comportemental VHDL du cône O :

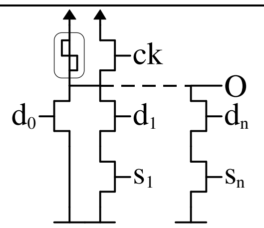


Figure A2.4 : cône BUS à Précharge

```
ENTITY BUS_P IS
    PORT (
    ck : IN BIT;
    d0 : IN BIT;
    d : IN BIT_VECTOR (1 to n);
    s : IN BIT_VECTOR (1 to n);
    O : WOR_BIT BUS
    );
END BUS _P;

ARCHITECTURE behaviour_data_flow OF BUS _P IS

BEGIN

    lbl00 : BLOCK (d0 = '1')
    BEGIN
        O <= GUARDED '0';
    END BLOCK lb100;

    lbl01 : BLOCK ((d1 AND s1) = '1')
    BEGIN
        O <= GUARDED '0';
    END BLOCK lbl01;
    ...
    lbl0n : BLOCK ((dn AND sn) = '1')
    BEGIN
        O <= GUARDED '0';
    END BLOCK lbl0n;

    lblpr : BLOCK (ck = '1')
    BEGIN
        O <= GUARDED '1';
    END BLOCK lblpr;

END;
```

5. Le cône Mise à Zéro

Modèle comportemental VHDL du cône O :

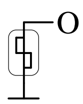


Figure A2.5 : cône Mise à Zéro

```
ENTITY ZERO IS
    PORT (
    O : OUT BIT
    );
END ZERO;
```

```
ARCHITECTURE behaviour_data_flow OF ZERO IS

BEGIN

    O <= '0'

END;
```

6. Le cône Mise à Un

Modèle comportemental VHDL du cône O :

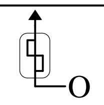


Figure A2.6 : cône Mise à Un

```
ENTITY UN IS
    PORT (
    O : OUT BIT
    );
END UN;

ARCHITECTURE behaviour_data_flow OF UN IS

BEGIN

    O <= '1'

END;
```

7. Le cône SBE

Schéma :

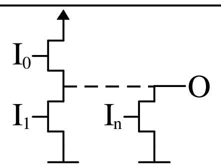


Figure A2.7 : cône SBE

Le cône O n'a pas de modèle comportemental VHDL propre.

8. Le cône SBD

Schéma :

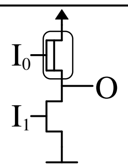


Figure A2.8 : cône SBD

Le cône O n'a pas de modèle comportemental VHDL propre.

9. Le cône Squirt Buffer

Schéma :

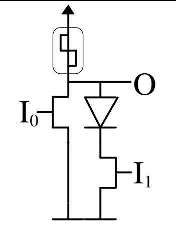


Figure A2.9 : cône Squirt Buffer

Le cône O n'a pas de modèle comportemental VHDL propre.

10. Le cône Demi-Latch

Schéma :

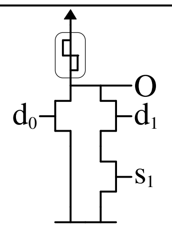


Figure A2.10 : cône Demi-Latch

Le cône O n'a pas de modèle comportemental VHDL propre.

11. Le cône Tristate

Schéma :

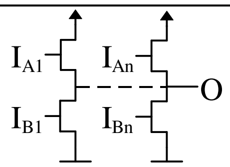


Figure A2.11 : cône Tristate

Le cône O n'a pas de modèle comportemental VHDL propre.

Annexe 3 : les modèles de cellules

Remarques préliminaires :

- ↑ et ⊥ représentent respectivement les tensions d'alimentation Vdd et Gnd.
- représente le transistor MESFET enrichi.
- représente le "driven pull-up".
- et représentent respectivement un "pull-up" un "pull-down".
- représente la diode Schottky.
- Le symbole représente l'inverseur qui correspond au NOR à une entrée présenté dans l'annexe 2.
- Le symbole représente le NOR à deux entrées présenté dans l'annexe 2.
- Les descriptions comportementales suivantes sont simplifiées de façon à ne présenter que les informations pertinentes.

1. La cellule Super Buffer

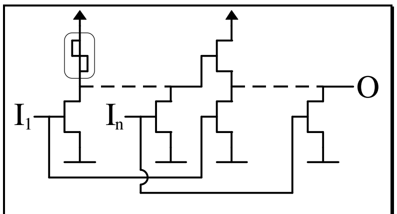


Figure A3.1 : cellule Super Buffer

Modèle comportemental VHDL du cône O :

```
ENTITY SUPER_B IS
    PORT (
    I1 : IN BIT;
    ...
    In : IN BIT;
    O : OUT BIT
    );
END SUPER_B;

ARCHITECTURE behaviour_data_flow OF SUPER_B IS

BEGIN

    O <= NOT (I1 OR ... OR In)
```

```
END;
```

2. La cellule Squirt Buffer

Modèle comportemental VHDL du cône O :

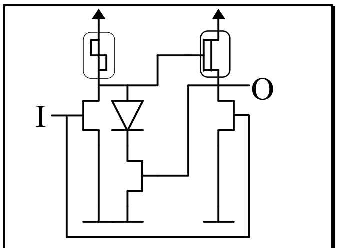


Figure A3.2 : cellule Squirt Buffer

```
ENTITY SQUIRT_B IS
    PORT (
    I : IN BIT;
    O : OUT BIT
    );
END SQUIRT_B;

ARCHITECTURE behaviour_data_flow OF SQUIRT_B IS

BEGIN

    O <= NOT (I)

END;
```

3. La cellule Latch

Modèle comportemental VHDL du cône O :

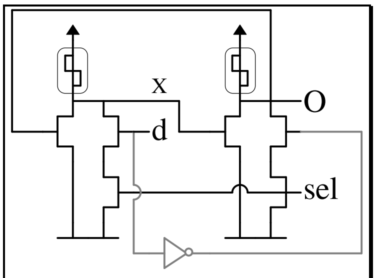


Figure A3.3 : cellule Latch

```
ENTITY LATCH_D IS
    PORT (
    sel : IN BIT;
    d : IN BIT;
    O : OUT BIT
    );
END LATCH_D;

ARCHITECTURE behaviour_data_flow OF LATCH_D IS

SIGNAL X : REG_BIT REGISTER;

BEGIN

    lbl00 : BLOCK (sel = '1')
    BEGIN
        X <= GUARDED NOT (d);
    END BLOCK lbl00;

    O <= NOT (X)

END;
```

4. La cellule Émetteur Tristate

Modèle comportemental VHDL du cône O :

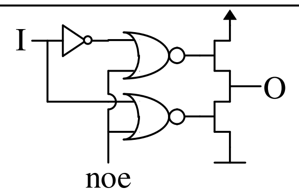


Figure A3.4 : cellule Émetteur Tristate

```
ENTITY E_TRISTATE IS
    PORT (
    noe : IN BIT;
    I : IN BIT;
    O : OUT MUX_BIT BUS
    );
END E_TRISTATE;

ARCHITECTURE behaviour_data_flow OF E_TRISTATE IS

BEGIN

    lbl00 : BLOCK (noe = '0')
    BEGIN
        O <= GUARDED NOT (I);
    END BLOCK lbl00;

END;
```

5. La cellule Bus de Lecture

Modèle comportemental VHDL du cône O :

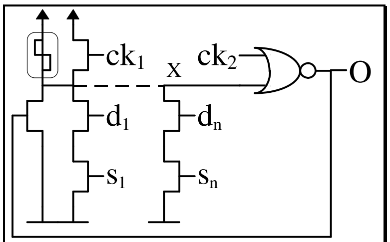


Figure A3.5 : cellule Bus de Lecture

```
ENTITY B_LECTURE IS
    PORT (
    ck1 : IN BIT;
    ck2 : IN BIT;
    d : IN BIT_VECTOR (1 to n);
    s : IN BIT_VECTOR (1 to n);
    O : OUT BIT
    );
END B_LECTURE;

ARCHITECTURE behaviour_data_flow OF B_LECTURE IS

SIGNAL X : WOR_BIT REGISTER;

BEGIN

    ASSERT ((ck1 XOR ck2) = '0')
        REPORT "ck1 and ck2 are not identical"
        SEVERITY ERROR;

    lbl01 : BLOCK ((d1 AND s1) = '1')
    BEGIN
        X <= GUARDED '0';
    END BLOCK lbl01;
    ...
    lbl0n : BLOCK ((dn AND sn) = '1')
    BEGIN
        X <= GUARDED '0';
```

```
    END BLOCK lbl0n;

    lblpr : BLOCK (ck1 = '1')
    BEGIN
        X <= GUARDED '1';
    END BLOCK lblpr;

    O <= (ck2 NOR X)

END;
```

Annexe 4 : temps d'exécution

Circuit :	**sh12**	**sh16**	**sh20**	**add28**	**sh24**	**add32**
Nombre de transistors :	*817*	*1041*	*1505*	*1596*	*1781*	*1920*
Nombre de cônes :	*291*	*363*	*519*	*684*	*607*	*739*
Nombre de cellules :	*15*	*15*	*19*	*9*	*19*	*11*
fonctions :						
initialisations :	00:00,01	00:00,00	00:00,01	00:00,00	00:00,01	00:00,01
getlofig :	00:00,57	00:00,73	00:01,06	00:01,14	00:01,27	00:01,42
lofigchain :	00:00,01	00:00,01	00:00,01	00:00,01	00:00,01	00:00,01
elpenv :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00
elpLoadElp :	00:00,00	00:00,00	00:00,01	00:00,01	00:00,01	00:00,00
elpLofigShrink :	00:00,00	00:00,01	00:00,00	00:00,00	00:00,00	00:00,00
elpLofigCapaGrid :	00:00,00	00:00,00	00:00,01	00:00,01	00:00,01	00:00,01
elpLofigCapaDiff :	00:00,00	00:00,01	00:00,00	00:00,01	00:00,01	00:00,01
tna :	**00:00,00**	**00:00,01**	**00:00,01**	**00:00,02**	**00:00,02**	**00:00,03**
fox :	**00:00,05**	**00:00,05**	**00:00,06**	**00:00,09**	**00:00,11**	**00:00,12**
lotrscount :	00:00,01	00:00,00	00:00,00	00:00,00	00:00,01	00:00,01
osr :	**00:00,00**	**00:00,00**	**00:00,01**	**00:00,00**	**00:00,01**	**00:00,01**
esr :	**00:00,06**	**00:00,07**	**00:00,07**	**00:00,05**	**00:00,08**	**00:00,05**
cnsstats :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,01	00:00,00
savecnsfig :	00:00,20	00:00,24	00:00,34	00:00,42	00:00,45	00:00,49
evm :	**00:00,02**	**00:00,04**	**00:00,04**	**00:00,05**	**00:00,08**	**00:00,06**
dellofig :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00
bxm :	**00:00,02**	**00:00,02**	**00:00,04**	**00:00,05**	**00:00,04**	**00:00,07**
vhdlsavebefig :	00:00,05	00:00,06	00:00,09	00:00,09	00:00,10	00:00,12
vega :	**00:01,05**	**00:01,35**	**00:01,85**	**00:01,94**	**00:02,22**	**00:02,41**

vega, min :	00:00,76	00:00,97	00:01,31	00:01,30	00:01,51	00:01,65
appels externes :	00:00,59	00:00,74	00:01,08	00:01,14	00:01,28	00:01,44
vega, min vérif :	00:00,81	00:01,06	00:01,42	00:01,44	00:01,67	00:01,81
appels externes vérif :	00:00,59	00:00,76	00:01,10	00:01,17	00:01,31	00:01,46
Δtrs :	224	464	91	185	139	137
Δco :	72	156	165	-77	132	-44
Δce :	0	4	-10	10	-8	8
Δco/Δtrs :	32,143%	33,621%	181,319%	-41,622%	94,964%	-32,117%
Δce/Δtrs :	0,000%	0,862%	-10,989%	5,405%	-5,755%	5,839%

Circuit :	**sh28**	**add36**	**sh32**	**add40**	**m8x8**	**add44**
Nombre de transistors :	*2057*	*2132*	*2333*	*2401*	*2622*	*2672*
Nombre de cônes :	*695*	*821*	*783*	*925*	*1010*	*1029*
Nombre de cellules :	*19*	*12*	*19*	*14*	*13*	*16*
fonctions :						
initialisations :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00
getlofig :	00:01,49	00:01,54	00:01,74	00:01,74	00:01,90	00:01,96
lofigchain :	00:00,01	00:00,01	00:00,02	00:00,02	00:00,02	00:00,02
elpenv :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00
elpLoadElp :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,01	00:00,00
elpLofigShrink :	00:00,02	00:00,01	00:00,02	00:00,01	00:00,01	00:00,01
elpLofigCapaGrid :	00:00,01	00:00,01	00:00,01	00:00,01	00:00,01	00:00,01
elpLofigCapaDiff :	00:00,01	00:00,01	00:00,01	00:00,01	00:00,02	00:00,01
tna :	**00:00,03**	**00:00,03**	**00:00,03**	**00:00,02**	**00:00,04**	**00:00,03**
fox :	**00:00,09**	**00:00,14**	**00:00,13**	**00:00,15**	**00:00,14**	**00:00,19**
lotrscount :	00:00,02	00:00,01	00:00,01	00:00,00	00:00,01	00:00,01
osr :	**00:00,00**	**00:00,01**	**00:00,02**	**00:00,01**	**00:00,01**	**00:00,02**
esr :	**00:00,07**	**00:00,06**	**00:00,08**	**00:00,09**	**00:00,06**	**00:00,07**
cnsstats :	00:00,01	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00
savecnsfig :	00:00,54	00:00,60	00:00,65	00:00,63	00:00,71	00:00,69
evm :	**00:00,10**	**00:00,07**	**00:00,09**	**00:00,11**	**00:00,09**	**00:00,08**
dellofig :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00
bxm :	**00:00,06**	**00:00,07**	**00:00,06**	**00:00,08**	**00:00,07**	**00:00,11**
vhdlsavebefig :	00:00,09	00:00,13	00:00,14	00:00,14	00:00,16	00:00,18
vega :	**00:02,43**	**00:02,68**	**00:02,91**	**00:03,26**	**00:03,21**	**00:03,56**

vega, min :	00:01,59	00:01,78	00:01,94	00:02,26	00:02,12	00:02,47
appels externes :	00:01,50	00:01,55	00:01,76	00:01,76	00:01,92	00:01,98
vega, min vérif :	00:01,79	00:01,96	00:02,12	00:02,49	00:02,34	00:02,69
appels externes vérif :	00:01,53	00:01,58	00:01,79	00:01,79	00:01,97	00:02,01
Δtrs :	75	201	68	221	50	324
Δco :	126	-38	142	85	19	124
Δce :	-7	7	-5	-1	3	2
Δco/Δtrs :	168,000%	-18,905%	208,824%	38,462%	38,000%	38,272%
Δce/Δtrs :	-9,333%	3,483%	-7,353%	-0,452%	6,000%	0,617%

Circuit :	add48	rgf_32_8	m16x8	rgf_32_16	rgf_64_8	m32x8
Nombre de transistors :	*2996*	*4142*	*4696*	*7150*	*8226*	*8956*
Nombre de cônes :	*1153*	*1110*	*1812*	*1798*	*2196*	*3458*
Nombre de cellules :	*18*	*409*	*16*	*721*	*809*	*23*
fonctions :						
initialisations :	00:00,00	00:00,01	00:00,00	00:00,00	00:00,00	00:00,00
getlofig :	00:02,27	00:02,75	00:03,38	00:05,01	00:05,83	00:06,74
lofigchain :	00:00,02	00:00,03	00:00,03	00:00,06	00:00,06	00:00,06
elpenv :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00
elpLoadElp :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00
elpLofigShrink :	00:00,01	00:00,01	00:00,02	00:00,03	00:00,03	00:00,03
elpLofigCapaGrid :	00:00,01	00:00,02	00:00,03	00:00,03	00:00,04	00:00,05
elpLofigCapaDiff :	00:00,02	00:00,03	00:00,02	00:00,06	00:00,05	00:00,05
tna :	**00:00,03**	**00:00,05**	**00:00,06**	**00:00,08**	**00:00,09**	**00:00,11**
fox :	**00:00,17**	**00:00,29**	**00:00,25**	**00:00,67**	**00:00,74**	**00:00,54**
lotrscount :	00:00,01	00:00,01	00:00,01	00:00,03	00:00,03	00:00,03
osr :	**00:00,01**	**00:00,01**	**00:00,01**	**00:00,02**	**00:00,02**	**00:00,03**
esr :	**00:00,09**	**00:00,43**	**00:00,08**	**00:00,64**	**00:00,87**	**00:00,12**
cnsstats :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,01	00:00,01
savecnsfig :	00:00,78	00:01,11	00:01,20	00:01,81	00:02,43	00:02,52
evm :	**00:00,12**	**00:00,17**	**00:00,21**	**00:00,46**	**00:00,46**	**00:00,33**
dellofig :	00:00,00	00:00,00	00:00,00	00:00,01	00:00,00	00:00,00
bxm :	**00:00,10**	**00:00,08**	**00:00,13**	**00:00,14**	**00:00,19**	**00:00,26**
vhdlsavebefig :	00:00,13	00:00,21	00:00,29	00:00,36	00:00,42	00:00,74
vega :	**00:03,84**	**00:05,35**	**00:05,84**	**00:09,22**	**00:11,08**	**00:11,40**

vega, min :	00:02,69	00:03,70	00:03,92	00:06,29	00:07,42	00:07,38
appels externes :	00:02,28	00:02,79	00:03,41	00:05,07	00:05,89	00:06,80
vega, min vérif :	00:02,94	00:04,03	00:04,36	00:07,04	00:08,22	00:08,14
appels externes vérif :	00:02,32	00:02,85	00:03,48	00:05,19	00:06,01	00:06,93
Δtrs :	1146	554	2454	1076	730	428
Δco :	-43	702	-14	398	1262	162
Δce :	391	-393	705	88	-786	5
Δco/Δtrs :	-3,752%	126,715%	-0,570%	36,989%	172,877%	37,850%
Δce/Δtrs :	34,119%	-70,939%	28,729%	8,178%	-107,671%	1,168%

Circuit :	**m16x16**	**rgf_32_32**	**rgf_64_16**	**m32x16**	**rgf_64_32**	**m32x32**
Nombre de transistors :	*9384*	*13166*	*14194*	*17612*	*26130*	*34924*
Nombre de cônes :	*3620*	*3158*	*3524*	*6802*	*6180*	*13490*
Nombre de cellules :	*28*	*1345*	*1425*	*35*	*2657*	*59*
fonctions :						
initialisations :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00
getlofig :	00:06,73	00:09,61	00:09,64	00:13,29	00:17,67	00:25,21
lofigchain :	00:00,07	00:00,09	00:00,09	00:00,13	00:00,19	00:00,25
elpenv :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00
elpLoadElp :	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00	00:00,00
elpLofigShrink :	00:00,03	00:00,05	00:00,07	00:00,07	00:00,10	00:00,14
elpLofigCapaGrid :	00:00,05	00:00,07	00:00,08	00:00,10	00:00,14	00:00,19
elpLofigCapaDiff :	00:00,05	00:00,08	00:00,08	00:00,11	00:00,16	00:00,21
tna :	**00:00,12**	**00:00,18**	**00:00,17**	**00:00,23**	**00:00,34**	**00:00,44**
fox :	**00:00,58**	**00:01,86**	**00:01,91**	**00:01,05**	**00:05,83**	**00:02,02**
lotrscount :	00:00,03	00:00,05	00:00,05	00:00,06	00:00,09	00:00,13
osr :	**00:00,03**	**00:00,04**	**00:00,05**	**00:00,06**	**00:00,08**	**00:00,13**
esr :	**00:00,13**	**00:00,93**	**00:01,07**	**00:00,18**	**00:01,49**	**00:00,36**
cnsstats :	00:00,01	00:00,01	00:00,01	00:00,02	00:00,02	00:00,03
savecnsfig :	00:02,47	00:03,44	00:03,90	00:04,68	00:07,43	00:10,38
evm :	**00:00,34**	**00:00,74**	**00:00,74**	**00:00,78**	**00:01,70**	**00:01,42**
dellofig :	00:00,00	00:00,01	00:00,01	00:00,01	00:00,02	00:00,02
bxm :	**00:00,26**	**00:00,31**	**00:00,39**	**00:00,53**	**00:00,78**	**00:01,06**
vhdlsavebefig :	00:00,69	00:00,80	00:00,93	00:02,00	00:02,17	00:05,82
vega :	**00:11,88**	**00:18,44**	**00:19,52**	**00:23,24**	**00:38,76**	**00:49,07**

vega, min :	00:07,95	00:12,89	00:13,27	00:14,86	00:26,13	00:29,63
appels externes :	00:06,80	00:09,70	00:09,73	00:13,42	00:17,86	00:25,46
vega, min vérif :	00:08,72	00:14,20	00:14,68	00:16,52	00:29,12	00:32,82
appels externes vérif :	00:06,92	00:09,90	00:09,96	00:13,70	00:18,25	00:26,00
Δtrs :	3782	1028	3418	8518	8794	
Δco :	-462	366	3278	-622	7310	
Δce :	1317	80	-1390	2622	-2598	
Δco/Δtrs :	-12,216%	35,603%	95,904%	-7,302%	83,125%	
Δce/Δtrs :	34,823%	7,782%	-40,667%	30,782%	-29,543%	

Circuit :	moyennes :
Nombre de transistors :	7 269
Nombre de cônes :	2 365
Nombre de cellules :	322
fonctions :	
initialisations :	00:00,00
getlofig :	00:05,19
lofigchain :	00:00,05
elpenv :	00:00,00
elpLoadElp :	00:00,00
elpLofigShrink :	00:00,03
elpLofigCapaGrid :	00:00,04
elpLofigCapaDiff :	00:00,04
tna :	00:00,09
fox :	00:00,72
lotrscount :	00:00,03
osr :	00:00,02
esr :	00:00,30
cnsstats :	00:00,01
savecnsfig :	00:02,00
evm :	00:00,34
dellofig :	00:00,00
bxm :	00:00,20
vhdlsavebefig :	00:00,66
vega :	00:09,85

vega, min :	00:06,49
appels externes :	00:05,25
vega, min vérif :	00:07,18
appels externes vérif :	00:05,36
Δtrs :	1482,91
Δco :	573,87
Δce :	1,91
Δco/Δtrs :	56,790%
Δce/Δtrs :	-4,996%

Annexe 5 : TAGA

Nous reproduisons dans cette annexe certaines parties du rapport de J. Pistorius [Pis96].

Annexe 6 : glossaire

Acronymes	Français	English
AsGa	**Ars**éniure de **Ga**llium	Gallium Arsenide
β	rapport de dimensionnement	sizing ratio
BDCFL	élément de puissance basée sur la DCFL dit "DCFL Bufferisé"	**B**uffered **DCFL**
BFL	logique de puissance à base de transistors à effet de champ à deux étages	**B**uffered **F**ield effect transistor **L**ogic
BXM	module de génération du modèle VHDL	**B**ehavioural e**X**traction **M**odule
C_0	capacité de grille par unité de longueur	unit length gate capacity
$C_{interco}$	capacité d'interconnexion	wiring capacitance
C_{load}	capacité de charge	load capacity
CMOS	logique basée sur les transistors à effet de champ à barrière métal oxyde semiconducteur	**C**omplementary **M**etal **O**xide **S**emiconductor logic
CNS	structure écrite en langage C pour la représentation de *net-lists* de cônes	**C**one **N**et-list **S**tructure
DCFL	logique à base de MESFET s'apparentant à la logique NMOS	**D**irect **C**oupled **F**ield effect transistor **L**ogic
DOWN	signifiant vers Gnd	meaning towards Gnd
E/D	**E**nrichissement/**D**éplétion	**E**nhancement/**D**epletion
ERC	vérificateur de règles électriques	**E**lectrical **R**ules **C**hecker
ESR	module de reconnaissance des cellules au niveau structurel	c**E**ll **S**tructural **R**ecognition module
EVM	module de vérification électrique	**E**lectrical **V**erification **M**odule
FOX	module de construction des cônes à partir de la *net-list* de transistors	**F**lattened net-list c**O**ne e**X**traction module
GaAs	Arséniure de Gallium	**Ga**llium **A**r**s**enide
Gnd	tension d'alimentation fournissant le niveau logique bas ou "0"	Power supply, logic low level
I_{DS}	courant drain-source	drain-source current
I_{pol}	courant de polarisation	biasing current
I_{sub}	courant sous le seuil	sub-threshold current
L	longueur de grille de transistor	transistor gate length

Acronymes	Français	English
MASI	Laboratoire de Paris 6 : **M**éthodes, **A**rchitectures et **S**ystèmes **I**nformatiques. Renommé ASIM en 1997 : **A**rchitecture des **S**ystèmes Intégrés & **M**icro-électronique	
MESFET	transistor à effet de champ à barrière métal semiconducteur	**ME**tal **S**emiconductor **F**ield **E**ffect **T**ransistor
NM_H	marge au bruit haute	high noise margin
NM_L	marge au bruit basse	low noise margin
NMOS	logique basée sur les transistors à effet de champ à barrière métal oxyde semiconducteur et à canal N	**N**-channel **M**etal **O**xide **S**emiconductor logic
OSR	module de reconnaissance des cônes au niveau structurel	c**O**ne **S**tructural **R**ecognition module
RTL	transfert entre registres	**R**egister **T**ransfer **L**evel
SBDCFL	élément de puissance basée sur la DCFL dit "DCFL Super Buffer"	**S**uper **B**uffer **DCFL**
SCFL	logique à base de transistors à effet de champ et à source de courant commandé	**S**ource **C**oupled **F**ield effect transistor **L**ogic
SDFL	logique à base de transistors à effet de champ et de diodes Schottky	**S**chottky **D**iode **F**ield effect transistor **L**ogic
SqBDCFL	élément de puissance basée sur la DCFL dit "DCFL Squirt Buffer"	**Sq**uirt **B**uffer **DCFL**
TAGA	outil pour le calcul des temps de propagations et de transition des cônes pour la circuiterie AsGa MESFET	a **T**iming **A**nalyser for **G**allium **A**rsenide
T_D	transistor déplété	depletion transistor
T_E	transistor enrichi	enhancement transistor
t_f	temps de descente	fall time
t_{hh}	temps de propagation niveau logique haut vers niveau logique haut	high to high transition propagation delay
t_{hl}	temps de propagation niveau logique haut vers niveau logique bas	high to low transition propagation delay
t_{lh}	temps de propagation de niveau logique bas vers niveau logique haut	low to high transition propagation delay

Acronymes	Français	English
t_{ll}	temps de propagation de niveau logique bas vers niveau logique bas	low to low transition propagation delay
TNA	module d'analyse de la *net-list* de transistors	**T**ransistor **N**etwork **A**nalysis module
t_r	temps de monté	rise time
UP	signifiant vers Vdd	meaning towards Vdd
V_d	tension seuil de la diode Schottky	Schottky diode threshold voltage
Vdd	tension d'alimentation fournissant le niveau logique haut ou "1"	Power supply, logic high level
V_{DS}	tension drain-source	drain-source voltage
VEGA	environnement de vérification pour portes	**V**erification **E**nvironment for **GA**tes
V_{GS}	tension grille-source	gate-source voltage
VHDL	langage de description de matériel du projet VHSIC	VHSIC **H**ardware **D**escription **L**anguage
VHSIC	circuits intégrés à très haute vitesse	**V**ery **H**igh **S**peed **I**ntegrated **C**ircuits
V_{IH}	tension minimale correspondant a un niveau logique haut en entrée d'étage	minimum gate input voltage identified as logic high
V_{IL}	tension maximale correspondant a un niveau logique bas en entrée d'étage	maximum gate input voltage identified as logic low
VLSI	intégration à très haute échelle	**V**ery **L**arge **S**cale **I**ntegration
V_{OH}	tension minimale correspondant a un niveau logique haut en sortie d'étage	minimum gate output voltage identified as logic high
V_{OL}	tension maximale correspondant a un niveau logique bas en sortie d'étage	maximum gate output voltage identified as logic low
V_T	tension seuil	threshold voltage
V_{TD}	tension seuil du transistor déplété	depletion transistor threshold voltage
V_{TE}	tension seuil du transistor enrichi	enhancement transistor threshold voltage
W	largeur de grille de transistor	transistor gate width
W_D	largeur de grille de transistor déplété	depletion transistor gate width
W_E	largeur de grille de transistor enrichi	enhancement transistor gate width

Acronymes	Français	English
$\mathbf{W_L}$	somme des largeurs de grille de transistors vu d'une équipotentielle	sum of transistor gate widths as seen from a given signal

Bibliographie

[Boe88] Michael Boehner, "LOGEX - An Automatic Logic Extractor from Transistor to Gate Level for CMOS Technology". 25th ACM/IEEE Design Automation Conference, IEEE 1988, paper 34.2, pp. 517-522.

[Dev92] Philippe Déverchère, Jean Christophe Madre, Jean Bruce Guignet, Michel Currat, "Functional Abstraction and Formal Proof of Digital Circuits". Proceedings of the European Conference on Design Automation EDAC'92, IEEE 1992, pp. 458-462.

[Dio97] Karim Dioury, Alain Greiner, Marie-Minerve Rosset-Louërat, "Accurate static timing analysis for deep submicronic CMOS circuits". Proceedings of the International Conference on Very Large Scale Integration (VLSI'97), Gramado, Brasil, August 1997, pp. 439-450.

[Gre92] Alain Greiner, François Pêcheux, "Alliance: A complete Set of CAD Tools for teaching VLSI Design". Proceedings of the 3rd Eurochip Workshop on VLSI Design Training, September 1992, pp. 230-237.

[Har93] S. J. Harrold "An introduction to GaAs IC design". Prentice Hall International (UK) Ltd., 1993.

[Kia94] Payam Kiani, "Librairie de fonctions de caractérisation électrique de *net-list* de transistors". LIP6, Université Paris VI, juin 1996.

[Kos89] Ton Kostelijk, "VERA, a Rule-Based Verification Assistant for VLSI Circuit Design". Proceedings of the international Conference on Very Large Scale Integration, IEEE 1989.

[Kos91] Ton Kostelijk, Bart De Loore "Automatic Verification of Library-Based IC Designs". IEEE Journal of Solid-State Circuits, Vol. 26, N°3, March 1991.

[Lau94] M. Laurentin, "Abstraction fonctionnelle de circuits intégrés CMOS". Thèse de doctorat de l'Université Paris VI, décembre 1994.

[Les97] A. Lester, "CNS. Cone Netlist Structure V2.00. Programmers guide". CNS, guide du programmeur. Laboratoire ASIM, LIP6, Université Paris VI, septembre 1997.

[Lon90] S. I. Long, S. E. Butner "Gallium Arsenide Digital Integrated Circuit Design". McGraw Hill Publishing Company, 1990.

[Mea80] C. Mead, L. Conway "Introduction to VLSI systems". Addison-Wesley Series in Computer Science, Addison-Wesley Publishing Company, Inc., October 1980.

[Mur88] J. Y. Murcin, "FAON, a Functional Abstractor of Netlist". Bull internal Report, Les Clayes Sous Bois 1988.

[Mu88b] J. Y. Murcin, "DESA, a Netlist Desassembler". Bull internal Report, Les Clayes Sous Bois 1988.

[Neb86] W. Nebel, "Automatic Extraction Of RT-Level Description from Integrated Circuit Layout Data". Doctoral Thesis of Kaiserlauten University, 1986.

[Neb87] W. Nebel, R. W. Hartenstein "Functional Design Verification by Register Transfer Net Extraction". Proceeding of COMPEURO 1987.

[Pis96] Joachim Pistorius, "TAGA, un analyseur temporel pour les circuits intégrés AsGa". Mémoire de DEA du laboratoire ASIM du LIP6, Université Paris VI, septembre 1996.

[Rem97] Pascal Remy, Philippe Royannez, Amara Amara, "A GaAs electrical rule checker", X Simpósio Brasileiro de Concepção de Circuitos Integrados (SBCCI '97), Gramado, Brasil, August 1997, pp. 1-10.

[Roy96] Philippe Royannez, "Réalisation de bibliothèques et d'outils logiciels pour la conception de circuits intégrés sur Arséniure de Gallium". Thèse de doctorat de l'Université Paris VI, juin 1996.

[Sed90] A. S. Sedra, K. C. Smith, "Microelectronic circuits - 3rd edition". The Holt, Rinehart and Winston Series in Electrical Engineering. Saunders College Publishing, 1990.

[Sta87] Hermann Statz, Paul Newman, Irl W. Smith, Robert A. Pucel, Hermann A. Haus, "GaAs FET Device and Circuit Simulation in SPICE". IEEE Transactions on Electron Devices, Vol. ED-34, N°2, February 1987, pp. 160-169.

[Ste92] Ronald Stewart, Véronique Anjubault, Philippe Garcin, Jacques Benkoski, "Automatic Import of Custom Designs into Cell-Based Environment using Switch-Level and Circuit Simulation". Proceedings of the European Design Automation Conference EURO-DAC'92, IEEE 1992, pp. 28-31.

[Vit93] Vitesse Foundry Design Manual 5.0, Vitesse Semiconductor Corporation, March 1993.