Projet ANR VALMEM

Délivrable : D2.1

Titre : *Etat de l'art des méthodes de validation des mémoires*

Partie 1 : Abstraction fonctionnelle et abstraction temporelle

Auteurs : P. Bazargan Sabet, P. Renault

Version : 2

Date : 13 juillet 2007

VALMEM : Validation fonctionnelle et temporelle des mémoires embarquées décrites au niveau transistor par des méthodes formelles

Etat de l'art des méthodes de validation des mémoires

Pirouz Bazargan Sabet, Patricia Renault

1 Description du document

Version préliminaire du déliverable D2.1, fournie par le LIP6.

2 Introduction

Classiquement, on distingue deux phases dans le processus de conception d'un circuit ou d'un système intégré. Dans la phase descendante, le concepteur s'attache à créer une description plus détaillée du circuit en partant d'une description abstraite. La phase ascendante consiste à vérifier que la description détaillée ainsi obtenue est conforme aux spécifications initiales. La conception des circuits intégrés nécessite donc l'utilisation de deux types d'outils logiciels. Les outils de conception tels que les outils de placement, de routage, de synthèse, etc. permettent de franchir une étape dans la phase descendante. Les outils de vérification sont utilisés dans la phase ascendante.

L'étape de vérification post-layout est une étape importante dans la phase ascendante. Elle constitue l'ultime étape de vérification et consiste à s'assurer, avant la fabrication du circuit, que la réalisation obtenue respecte effectivement les spécifications. Les outils qui interviennent dans cette étape prennent en entrée le fichier de description des masques de fabrication du circuit. Les abstractions fonctionnelle et temporelle constituent la base sur laquelle sont bâtis la plupart des outils de ce niveau.

L'abstraction fonctionnelle s'attache à retrouver la fonction réalisée par un circuit en partant du dessin des masques de fabrication.

L'abstraction temporelle, utilise le résultat de l'abstraction fonctionnelle. Elle permet d'ajouter des informations temporelles aux fonctions reconnues par l'abstraction fonctionnelle.

Ce document passe en revue les différentes techniques utilisées pour effectuer ces traitements.

3 Abstraction fonctionnelle

Depuis le début des années 80, l'abstraction fonctionnelle intervient dans le processus de vérification des circuits intégrés. Elle consiste à obtenir une description plus abstraite du circuit en partant de la description des masques de fabrication. On peut distinguer trois contextes dans l'utilisation de l'abstraction fonctionnelle.

A l'origine, l'apparition de l'abstraction fonctionnelle est liée à la nécessité d'accélération des outils de vérification. En effet, avec l'augmentation du nombre de transistors intégrés sur un circuit, les concepteurs ont été amenés à faire face à la lenteur des outils de vérification tels que la simulation ou l'analyse temporelle. Dans ce cadre, l'abstraction fonctionnelle consiste à substituer à la description du circuit au niveau transistors, une description au niveau portes. Cette substitution peut couvrir la totalité ou seulement une partie du circuit. A chaque porte est associé un modèle abstrait permettant d'accélérer la simulation ou l'analyse.

Une seconde utilisation est liée à l'apparition des outils de preuve formelle. Au début des années 80, Brayant publie une technique de représentation des expressions booléennes [Bry86] basée sur la décomposition de Shannon : les ROBDD (Reduced Ordered Binary Decision Diagram). Cette représentation présente une propriété fondamentale : la canonicité. Quelle que soit la manière dont une expression booléenne peut être écrite, étant donnée une relation d'ordre strict entre les variables, la représentation de cette expression sous forme d'un ROBDD est unique. Cette technique a ouvert la voie à l'essor des outils qui manipulent des expressions booléennes tels que la synthèse logique ou la preuve formelle booléenne. Les outils de preuve formelle proposent une alternative à la simulation logique. Ils consistent à prouver que deux ensembles d'expressions booléennes réalisent la même fonction. Dans ce contexte, l'utilisation de l'abstraction fonctionnelle permet d'obtenir une description au niveau RTL (Register Transfer Level) du circuit en partant du dessin des masques. Dans cette description, le circuit est modélisé comme un ensemble de signaux combinatoires et d'éléments mémoire (latch ou registre) reliés entre eux par des expressions booléennes. Cette abstraction doit couvrir la totalité du circuit. Alors, le résultat de l'abstraction peut être utilisé par un outil de preuve formelle booléenne pour être comparé aux spécifications du circuit au niveau RTL (description du circuit avant l'étape de projection structurelle).

Enfin, l'outil d'abstraction fonctionnelle peut intervenir dans une troisième situation : la migration technologique. Il arrive souvent chez les concepteurs que l'on dispose des masques de fabrication d'un circuit dont on a perdu les spécifications. Or, ces masques sont attachés à une technologie de fabrication particulière. La migration de ces circuits d'une technologie ancienne vers une nouvelle technologie passe par l'abstraction fonctionnelle. Dans ce cadre, l'abstraction fonctionnelle permet d'obtenir une description abstraite du circuit par exemple au niveau RTL. Cette description est ensuite utilisée comme spécification pour une nouvelle conception du même circuit dans une technologie récente. Cette même technique peut également servir à la *reverse engineering*. La *reverse engineering* consiste à récupérer un circuit conçu par un concurrent et à en faire un clone. Pour ce faire, les ingénieurs photographient à l'aide d'un microscope, couche par couche, le circuit et reconstitue le dessin des masques. Puis, ils utilisent l'abstraction fonctionnelle pour obtenir les spécifications. Celles-ci sont utilisées pour concevoir un clone du circuit original.

Comme on le voit, suivant le contexte d'utilisation, les contraintes imposées aux outils d'abstraction fonctionnelle peuvent être très variables.

D'un côté, on peut exiger de l'abstraction fonctionnelle de couvrir la totalité ou seulement une partie du circuit. D'un autre côté, elle doit fournir pour la partie abstraite un ensemble de portes associées à un modèle ou, des expressions booléennes. Enfin, l'outil d'abstraction fonctionnelle peut avoir connaissance ou pas des éléments à partir des quels le circuit a été conçu. Pour les circuits construits à partir d'une bibliothèque de cellules standard, la connaissance de cette bibliothèque peut faciliter la mise en oeuvre de l'abstraction. Dans ce cas, l'abstraction peut faire appel à des techniques de pattern matching. A l'inverse, le traitement des circuits custom et l'absence de connaissance des éléments du circuit poussent à l'utilisation de technique de reconnaissance plus formelle. Ces techniques sont basées sur l'analyse de la structure d'interconnexion des transistors ou sur l'analyse des chemins électriques.

Dans sa thèse [Lau94] Marc Laurentin fournit une bibliographie très détaillée des techniques d'abstraction fonctionnelle. Les lignes qui suivent sont en partie extraites du chapitre état de l'art de son manuscrit.

3.1 Abstraction fonctionnelle par partitionnement et reconnaissance de formes

La vérification logique chez Toshiba

Une méthode de vérification logique des circuits intégrés utilisée à Toshiba est décrite dans [Kaw82]. Elle consiste à substituer à certaines parties du circuit, une description comportementale en vue d'accélérer la simulation.

Ce système extrait une net-list de transistors depuis le dessin des masques, puis classe les noeuds du circuit. Par exemple, les noeuds pull-up, pour les circuits Nmos sont distingués, ainsi que les noeuds reliés à une grille de transistor. Le système s'appuie sur cette classification pour partitionner le circuit en blocs distincts. Ces sous circuits sont ensuite comparés aux éléments d'une bibliothèque qui contient des portes élémentaires : Nand, Nor, Not, Or, And.

Bien entendu, certains transistors restent célibataires après le partitionnement et n'appartiennent à aucune des portes prédéfinies de la bibliothèque. Ceux la font l'objet d'une description comportementale élémentaire décrivant le transistor comme un interrupteur.

Conçu au départ pour accélérer la simulation, cet outil remplace également certaines configurations de portes en une description comportementale de plus haut niveau. Cette fonctionnalité est typiquement mise en oeuvre pour la description des éléments mémorisants. En revanche, ce module ne sait traiter que des circuits de petite taille

(quelques centaines de transistors) et impose donc que le circuit soit découpé en plusieurs blocs.

Ce système est certes imparfait puisqu'une partie du circuit reste quasiment décrite au niveau transistors et que, comme dans tous les outils décrits dans cette partie, seule une simulation mixte (transistors et modèles comportementaux) est envisageable à partir de la description obtenue. Cependant, on y trouve les interrogations élémentaires concernant l'abstraction fonctionnelle.

En revanche, en hésitant entre le partitionnement automatique, et la reconnaissance de forme, ce système s'interdit d'avoir une véritable bibliothèque ouverte, puisque le partitionnement mis en oeuvre dicte le type d'éléments reconnaissables.

Notons que l'idée d'un module capable de passer d'une description de niveau portes à une description plus abstraite est intéressante, et certains outils en sont capables aujourd'hui.

BLEX

L'objectif de BLEX [Lue84] développé à l'université de Hannovre, est de reconnaître dans un circuit, un bloc fonctionnel pré défini.

BLEX prend en entrée deux descriptions, le circuit et le bloc à reconnaître qui peuvent tous les deux être décrits au niveau transistors (dans un format similaire à SPICE). Les deux descriptions sont représentées par un graphe biparti que l'on peut voir de la façon suivante.

Un premier type de noeuds sont les éléments actifs (transistors, résistances en analogique etc., et les alimentations). Un deuxième type de noeuds (les "spiders") sont les équipotentielles qui relient les "pattes" des éléments actifs. Les arêtes entre les noeuds et les spiders, représentent les "pattes" des éléments actifs. Ces arêtes sont valuées par un nombre premier, en fonction du type d'élément actif auxquels elles appartiennent. Dans l'exemple illustré sur la figure 3.1, on a choisi une valeur de 2 pour les grilles, 3 pour les drains et les sources, 5 pour Vss etc. Enfin, les spiders, qui représentent finalement les équipotentielles, sont pondérés avec le produit des valeurs des arêtes auxquelles ils sont reliés. Ce produit est le reflet non ambigu de l'ensemble des composants qui sont reliés à une équipotentielle donnée. Cette représentation permet d'identifier rapidement quelles équipotentielles du circuit, sont susceptibles de correspondre, d'un point de vue topologique, à une équipotentielle du bloc prédéfini. Ce sont les équipotentielles qui sont *au moins* connectées aux mêmes éléments actifs que celle du bloc prédéfini. Pour le vérifier, il suffit de diviser le poids de l'équipotentielle du circuit par le poids de l'équipotentielle du bloc prédéfini : Un résultat *entier* indique une bonne correspondance entre les deux équipotentielles.

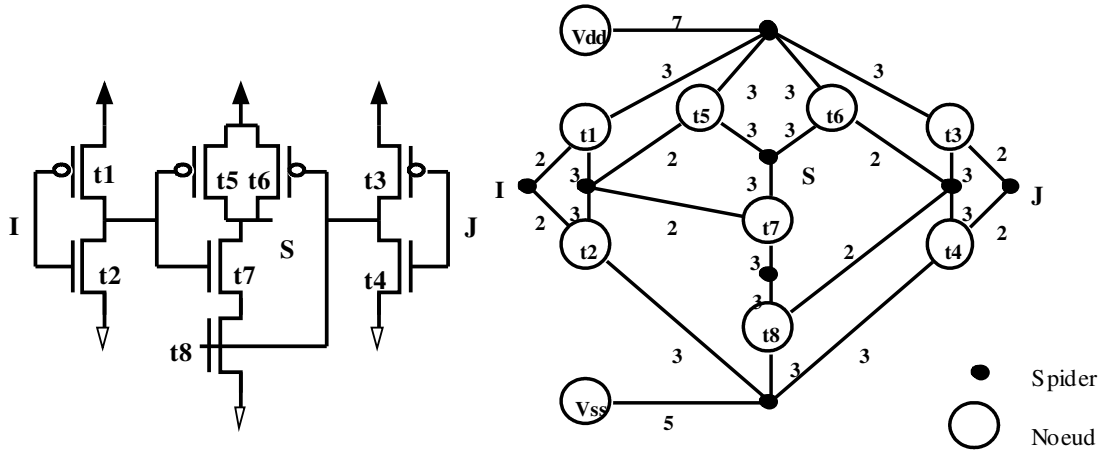

Fig 3.1 : Représentation d'un circuit dans BLEX

Le point fort de BLEX est qu'il n'effectue aucun partitionnement préalable sur les descriptions d'entrées, en conséquence de quoi, aucune contrainte structurelle ne limite la complexité du bloc fonctionnel à extraire. D'autre part, l'aspect général de l'algorithme, permet l'extraction de blocs sur des descriptions de n'importe quel niveau hiérarchique.

En revanche, BLEX ne peut prendre en entrée, hormis la description du circuit, qu'une seule description de bloc fonctionnel à la fois, ce qui ne permet pas de comparer un circuit et une bibliothèque complète d'une part, et empêche tout traitement des ambiguïtés (sous circuits pouvant appartenir à plusieurs blocs fonctionnels).

REX

REX [Neb86] [Neb87], est un système développé à l'université de Kaiserslauten, qui se propose de générer à partir du dessin des masques, une description en blocs fonctionnels.

Ce système incorpore d'une part, un extracteur qui produit la net-list de transistors à partir du dessin des masques, et d'autre part, un programme capable d'abstraire cette net-list au niveau bloc en se basant sur une bibliothèque d'éléments prédéfinis. Cette bibliothèque ne contient pas que des portes élémentaires telles des Nand, Nor etc., mais peut contenir des blocs plus complexes comme des additionneurs ou des multiplieurs. Il s'agit donc d'une approche "pattern matching". REX ne génère pas de vue comportementale du circuit, mais la net-list générée est au format EDIF, et peut être utilisée, dans un environnement logiciel (KarlIII) développé dans cette Université.

Le circuit et les cellules de références sont représentés comme dans BLEX par un graphe biparti à la différence qu'ici les spiders représentent les transistors et les noeuds du graphe représentent les équipotentielles.

Une signature est calculée pour chaque transistor du circuit et chaque transistor des cellules de référence, en fonction du type du transistor et de son environnement immédiat. Les équipotentielles sont typées : interne, externe etc. A chaque transistor du circuit est

associé une liste de candidats, qui est composée des transistors des cellules de la bibliothèque qui possèdent la même signature que lui et dont les connecteurs sont reliés à des équipotentielles de même type. Les équipotentielles ont, elles aussi, une liste de candidats.

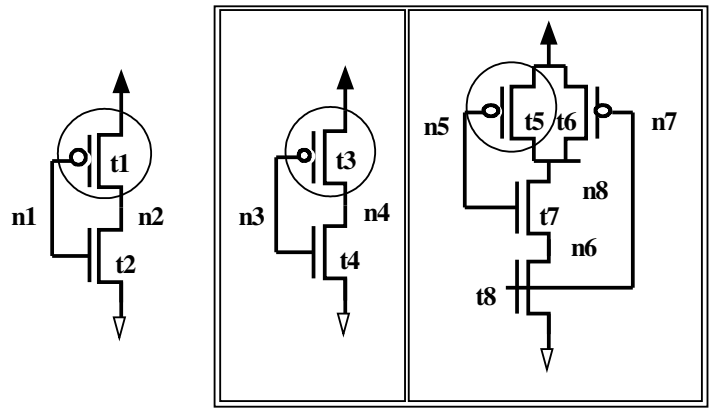

Fig 3.2-a

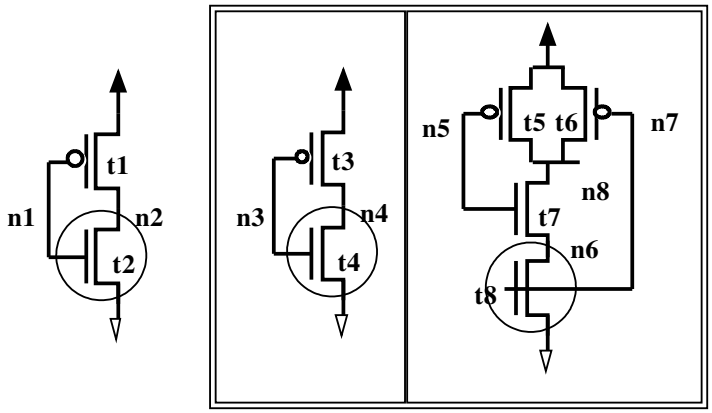

Fig 3.2-b

Fig 3.2 : Détection itérative des équipotentielles isomorphes

Pour que l'algorithme soit indépendant de l'ordre des cellules dans la bibliothèque, une correspondance entre un transistor du circuit et un transistor d'une cellule de la bibliothèque, n'empêche pas la recherche d'une autre correspondance avec des transistors des autres cellules de la bibliothèque.

Notons d'ailleurs, qu'un transistor du circuit peut avoir deux transistors candidats appartenant à la même cellule de la bibliothèque : les deux transistors N d'un Nor CMOS ont la même connectique et étant de même type, ils ne sont pas discernables.

Sur la figure 3.2.a le transistor t1 de l'inverseur du circuit, a deux candidats : t3 de l'inverseur, et t5 du Nand de la bibliothèque. Les candidats induits pour l'équipotentielle n2 sont donc respectivement n4 et n8. Par ailleurs, (figure 3.2.b) les candidats pour t2, sont d'une part, t4 de l'inverseur, et d'autre part, t8 du Nand de la bibliothèque. Ces transistors induisent à leur tour les candidats respectifs n4, et n6, pour l'équipotentielle n2 de l'inverseur du circuit. Dans ce cas l'examen des candidats des signaux, permet de rejeter la

solution "Nand", et de ne garder pour les transistors de l'inverseur du circuit, que ses candidats dans l'inverseur de la bibliothèque.

A l'issue de ces étapes, on ne sait pas obligatoirement à quelle cellule de la bibliothèque on peut associer le transistor du circuit si celui-ci possède encore plusieurs candidats.

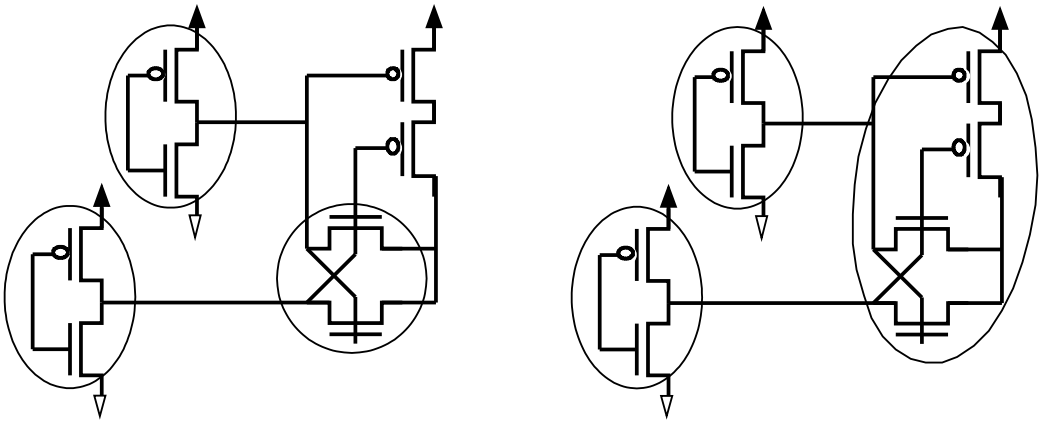

Fig 3.3-a *Fig 3.3-b*

Fig 3.3 : Partitionnement et couverture

Pour lever les ambiguïtés, la notion de couverture est introduite : La couverture est le nombre de transistors reconnus comme appartenant à une cellule, sur le nombre total de transistors. Le circuit est alors totalement, et correctement partitionné si la couverture vaut 1.

Dans le cas du circuit de la figure 3.3, et avec une bibliothèque qui contient un inverseur, un multiplexeur 2/1 à transistors de passages, et le Nxor, on peut obtenir soit une couverture de 0,75 (figure 3.3.a) et deux transistors célibataires ou une couverture de 1 (figure 3.4.b) qui correspond à un partitionnement correct.

Toutes les solutions sont envisagées par REX et c'est le partitionnement qui obtient la meilleure couverture qui est gardé comme solution. Lorsque deux partitionnements ont la même "meilleure" couverture, c'est celui qui présente le moins de composants qui est gardé.

On retrouve dans REX des caractéristiques intéressantes de BLEX, en particulier, l'indépendance technologique : N'importe quel type de circuit peut être traité par REX qui n'effectue que de la manipulation de graphes. Pour la même raison, la taille des blocs de la bibliothèque ainsi que leur nombre n'est pas limité, ce qui permet à REX d'être un outil ouvert, et évolutif.

Notons cependant, que le CMOS n'est pas la technologie la plus facile pour REX compte tenu de la symétrie drain/source, qui augmente significativement la complexité des algorithmes mis en oeuvre. En effet la recherche de candidats, ne doit pas chercher que des correspondances drain-drain et source-source, mais les quatre combinaisons possibles.

Le fait que REX reste au niveau transistor, pour effectuer la reconnaissance de formes est une source de complexité : En effet, on ne dénombre pas moins de 8 partitionnements différents (dont un seul est correct) à l'issu de REX sur un incrémenteur/décrémenteur Nmos, constitué de 15 transistors.

L'utilisateur aura donc intérêt à minimiser la bibliothèque en fonction de la composition attendue du circuit. C'est d'ailleurs là l'inconvénient récurent des outils basés sur la reconnaissance de formes : Il faut souvent bien connaître le circuit à traiter pour pouvoir les utiliser efficacement.

DESA

DESA [Dev92] [Mur88] est un désassembleur développé à BULL qui fourni une net-list de portes à partir d'une net-list de transistors. La net-list de portes fournie par DESA peut ensuite être utilisée par l'abstracteur fonctionnel FAON [Mur88b]. La tâche de FAON est principalement de substituer dans la net-list structurelle, les portes par leur comportement ré exprimé en fonction de leurs entrées, d'ordonner les équations de sorte que les variables soient affectées avant d'être référencées, et de vectoriser quand c'est possible, les bus.

C'est la tâche "abstraction au niveau portes" de DESA qui nous intéresse ici. DESA utilise également une approche de type "pattern matching" et donc une bibliothèque de portes prédéfinies. DESA ne peut donc pas tout reconnaître, mais comme on va le voir, sa bibliothèque peut contenir des cellules "exotiques". On y distingue 3 classes de portes ou *formes.*

1. Les portes CMOS duales et les réseaux CMOS duaux.
2. Les cellules non duales telles que les Nxor à transistors de passage, les portes trois états etc.
3. Les structures non duales paramétrables. On y trouve les multiplexeurs à transistors de passage dont le nombre d'entrées peut varier ou les Nor préchargés, là encore à nombre d'entrées variable.

La bibliothèque étant définie, les règles principales de fonctionnement de DESA sont les suivantes :

- L'ordre dans lequel apparaissent les cellules dans la bibliothèque, détermine l'ordre dans lequel DESA va chercher à les reconnaître dans le circuit. Les structures complexes doivent donc y apparaître en premier, si l'on ne veut pas que certains de leurs transistors soient interprétés comme des constituants de portes plus petites.
- Une cellule de la bibliothèque est recherchée dans tout le circuit, une fois pour toutes. Une fois passé à la cellule suivante, la précédente ne sera plus recherchée.
- Un transistor ne peut appartenir qu'à une seule cellule. C'est donc une partition stricte du circuit qui est visée.

- Une porte reconnue peut être réutilisée à un niveau d'abstraction plus élevé. C'est principalement le cas des inverseurs, qui entrent dans la composition des latchs et des registres.

Ces règles générales étant précisées, DESA part de noeuds particuliers pour chercher les formes. En particulier, les noeuds constituant des points de rencontre entre les drains/source de transistors de types différents, sont des points de départ pour la recherche de réseaux CMOS duaux.

D'autre part, certains noeuds doivent être désignés à l'aide d'une convention de nom comme des sorties de réseaux à transistors de passage. C'est grâce à ces conventions, que DESA peut reconnaître, et orienter, une chaîne carry-manchester.

La bibliothèque n'est pas accessible par les utilisateurs du programme. En effet, le moteur de reconnaissance de formes est écrit en PROLOG, ce qui a permis de décrire les blocs fonctionnels de la bibliothèque sous formes de prédicats. C'est la une contrainte très forte, qui complique fortement l'introduction de nouvelles portes dans la bibliothèque. Enfin, les conventions de nommage de certains noeuds, nécessaires pour la logique à transistors de passage, sont une contrainte très forte, et peuvent de surcroît, constituer une source d'erreurs.

VÉRA

Développé à PHILIPS, VERA [Kos89], [Kos91] est un extracteur "RT level" hiérarchique.

VÉRA prend en entrée d'une part, la description du circuit, en format similaire à SPICE, d'autre part la "description de types" qui constitue une bibliothèque de blocs prédéfinis. La bibliothèque peut contenir les descriptions au niveau transistors de cellules élémentaires comme les Nand, etc. Elle peut aussi contenir des descriptions de plus haut niveau.

Enfin, et c'est une originalité du système, une troisième entrée constitue la "base de règles". La base de règles contient des règles de type "SI *situation* ALORS *action"*. Un ensemble de primitives de situation et d'action permettent de définir les règles.

Citons des exemples que l'on peut trouver dans la partie situation

- *Recognize* qui s'appuie sur la "description de type" pour reconnaître un sous circuit dans le circuit.
- *Chain* qui reconnaît des montages en "série" du même élément.
- *Fork* qui reconnaît les montages en parallèle du même élément. et dans la partie action,
- *Abstract* qui remplace un sous-circuit reconnu, composé de transistors par exemple, par sa description de niveau supérieur.

- *Expand* est l'action duale de *Abstract*, elle substitue un élément par sa vue de niveau hiérarchique inférieur.

La règle SI *Recognize* ALORS *Abstract* est bien sur, tout indiquée pour l'extraction "RT level".

Dans VÉRA, les blocs fonctionnels sont distingués suivant trois classes :

1. Les modules non paramétrables tels que les cellules pré caractérisées
2. Les modules paramétrables de fonctionnalité constante. Cette appellation regroupe les motifs répétitifs ou chaque motif présente la même fonctionnalité. L'aspect répétitif se caractérise soit par un montage en "série" de plusieurs éléments identiques, soit par leur mise en parallèle, soit par les deux simultanément. C'est le cas par exemple d'un banc de registres. Les registres appartenant à la même ligne de mot sont abstraits en un mot, puis les mots sont abstraits par un banc de registres.
3. Enfin, les modules paramétrables à fonctionnalité variable. Ce sont les motifs répétitifs qui n'expriment pas tous la même fonctionnalité : C'est le cas des ROM ou des PLA, pour lesquels on a bien une structure régulière, avec cependant des différences fonctionnelles suivant les paramètres (mots, bits etc.). La reconnaissance de ces modules est sujette à l'écriture de fonctions particulières, l'ensemble de primitives de base étant insuffisant. Par exemple, une fonction LISP existe spécifiquement pour la reconnaissance des ROM

Dans tous les cas, il faut fournir dans la base de règles, toutes les règles nécessaires à la reconnaissance et à l'abstraction des éléments dont la présence est attendue dans le circuit. Pour reconnaître et abstraire le point mémoire de la figure 3.4, il faut que la base de règles contienne autant d'appels à une primitive *Recognize* (et *Abstract*), accompagnées des bons paramètres (en particulier le nom du bloc fonctionnel de la bibliothèque : "inverseur", puis "mémoire") qu'il y a de niveaux de hiérarchie à abstraire.

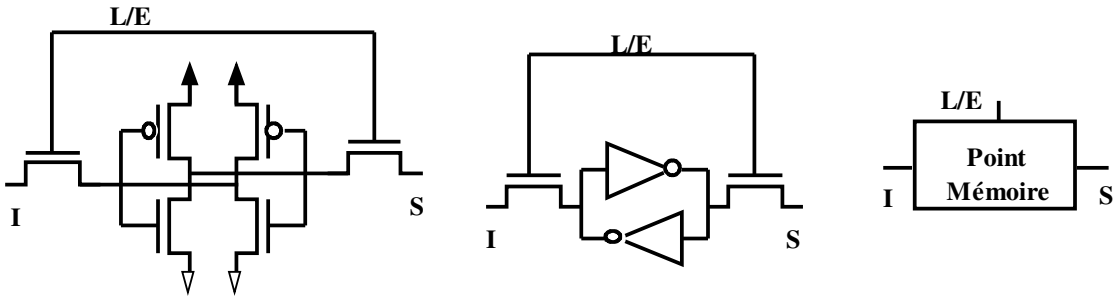

Fig 3.4-a *Fig 3.4-b* *Fig 3.4-c*

Fig 3.4 : Les étapes de l'abstraction d'un point mémoire

VERA nécessite donc d'avoir dans sa base de règles une description très précise de ce que le concepteur a voulu implémenter. Pour cette raison, VERA est utilisé principalement pour la vérification des circuits réalisés par des compilateurs de silicium tels que

CATHEDRAL [VGi86] ou PIRAMID [Hui88] qui génèrent des circuits aux structures régulières.

En revanche, VERA n'est pas adapté pour abstraire des circuits très peu réguliers comme les circuits réalisés en full custom car il est alors impossible de lui fournir une base de règles conforme.

En fait, on peut dire que l'approche choisie dans VERA n'est pas de *générer* une description structurelle simulable du circuit, mais est de *comparer* la net-list extraite au niveau transistor avec une description hiérarchique prédéfinie de celui-ci. VERA n'est donc pas adapté non plus pour abstraire des circuits dont on ne connaît pas la structure interne.

3.2 La réduction en séries de parallèles

CTL

CTL [Tak82] a été développé par une autre équipe de Toshiba. L'objectif de CTL est de déduire un réseau de portes avec leur comportement à partir d'un réseau de transistors.

Les sous réseaux de transistors N et P sont identifiés pour certains noeuds du circuit (intersection drain/source N et P, signaux de grille). Pour chaque sous réseau est ensuite construit un arbre qui le représente comme une composition de séries et de parallèles et ou les variables de grilles de transistors apparaissent sur les feuilles terminales. Cette représentation est utilisée pour tester la complémentarité des deux réseaux. En effet, il suffit de remplacer les *séries* par des *parallèles* dans l'un des deux graphes et de tester son équivalence avec l'autre pour conclure ou non à la complémentarité des deux réseaux. Un exemple est illustré sur la figure 3.5.

Lorsque cette règle est vérifiée, l'ensemble des deux réseaux est remplacé par une description en portes AND (séries) et OR (parallèles) correspondant au graphe du réseau N (on inverse la sortie de la dernière porte).

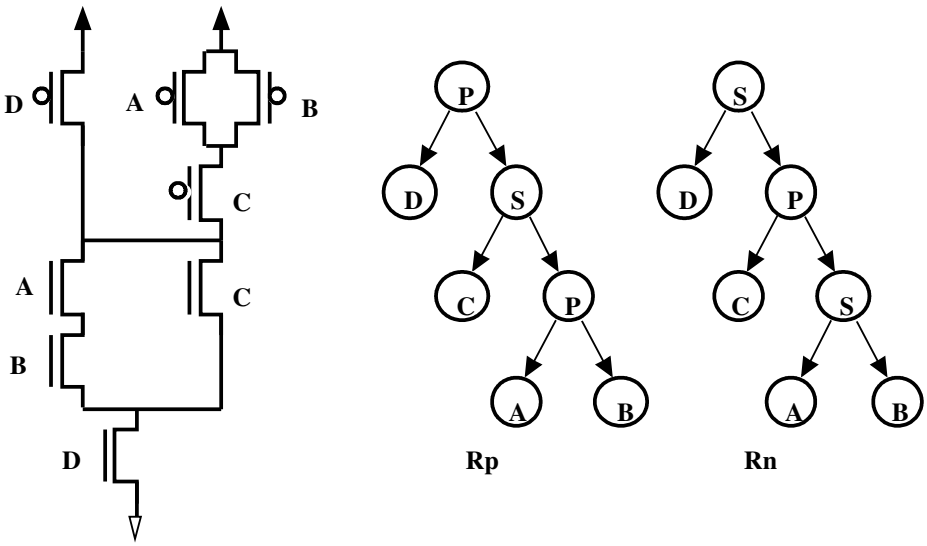

Fig 3.5 : La représentation en séries/parallèles

Une exception à cette règle est tolérée : Dans le cas ou un transistor apparaîtrait en série dans les deux réseaux, une porte CMOS "clockée" est reconnue (sous réserve que le signal qui attaque ce transistor soit inversé selon le réseau auquel il appartient).

Cette approche utilise une méthode formelle pour conclure à la dualité des deux réseaux N et P d'une porte CMOS par les techniques classiques de détection d'isomorphisme entre deux graphes. Cela dit, la simple connaissance de la non complémentarité des deux réseaux N et P, n'est pas suffisante pour conclure à une erreur de conception et cette approche, bien qu'enrichie pour répondre à un type de circuiterie particulier (les portes "clockées"), n'est réellement utilisable que pour les portes CMOS duales : Faire l'hypothèse que tout le circuit peut être décrit en termes de séries et de parallèle est limitatif. Enfin, rien n'est prévu dans CTL concernant la détection ou la description des registres.

On trouve dans [Kis83] une approche tout à fait similaire à CTL, et la description d'un algorithme permettant de détecter l'isomorphisme entre deux sous graphes des réseaux N et P. Ceci permet de détecter les transistors qui sont en séries dans les deux réseaux.

REDUCE

Développé à Hewlett-Packard à l'origine pour le Nmos [Apt82], puis étendu au CMOS [Bur83], REDUCE est un programme de réduction topologique destiné à réduire l'occupation en mémoire de la description du circuit extraite du dessin des masques. REDUCE se propose de générer des équations booléennes afin d'accélérer la simulation.

REDUCE partitionne la net-list de transistors en blocs dont l'interface est constituée soit par un signal de grille de transistor, soit la jonction drain/source de deux transistors de types différents. Les entrées du bloc sont les signaux qui attaquent les grilles des transistors des réseaux N et P (qui constituent alors les sous bloc) du bloc. En réduisant les sous blocs en séries de parallèles, et en substituant les séries par des AND, les parallèles

par des OR, on obtient comme pour CTL des équations du circuit. Ici, la complémentarité des deux réseaux est testée sur la base de l'équation trouvée pour chacun d'entre eux, et non à partir de la structure des deux graphes, ce qui est plus général.

A l'inverse de CTL, REDUCE accepte que le circuit ne soit pas entièrement constitué de séries de parallèles. En représentant le circuit à l'aide d'un graphe, dans lequel les noeuds représentent les équipotentielles, et les arêtes représentent les transistors, et en dédoublant les parties du graphe correspondant aux transistors bidirectionnels, les portes contenant des transistors bidirectionnels dans leurs réseaux N et P peuvent être traitées.

On retrouve donc dans REDUCE, l'approche séries/parallèles. Là encore, le modèle est enrichi pour prendre en compte un cas particulier que posent les transistors bidirectionnels. Néanmoins, REDUCE comme CTL, n'est pas adapté pour la logique à transistors de passages. Par exemple, le Xor à transistors de passages ne peut être décrit par ce modèle. Par conséquent, certaines parties du circuit restent décrites au niveau transistor. Enfin, il n'y a pas dans REDUCE, de module de reconnaissance des registres.

3.3 Approches formelles

LOGEX

LOGEX [Boe88] développé à Siemens, est un extracteur au niveau portes qui prend en entrée une net-list de transistors dans un format proche de SPICE.

LOGEX construit des *branches :* C'est une série de transistors entre deux noeuds. Au départ chaque transistor définit une branche. Les branches sont mises en séries si leur extrémité commune n'est pas reliée à un troisième connecteur.

LOGEX effectue donc de la réduction séries/parallèles sur toute partie du circuit qui respecte cette contrainte, comme l'illustre le figure 3.6.

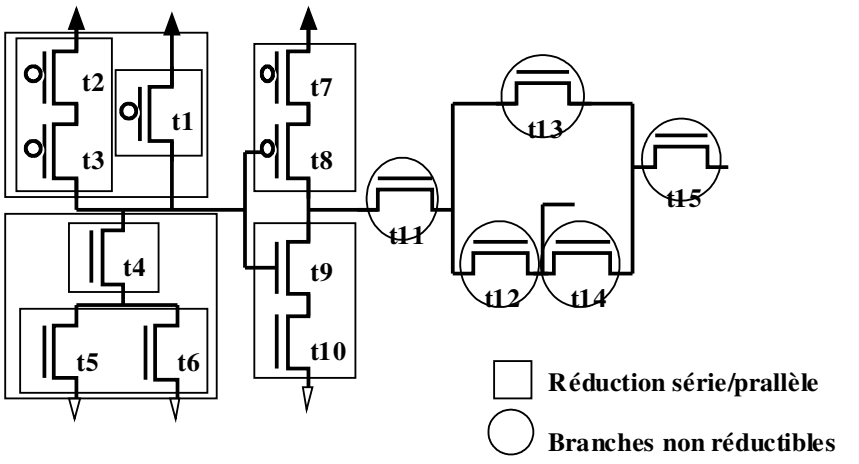

Fig 3.6 : Partitionnement et réduction série/parallèle dans LOGEX

Au terme de cette réduction, LOGEX obtient des branches unidirectionnelles vers VSS (type 1), vers VDD (type 2), et des branches bidirectionnelles (type 3) qui sont toutes les autres branches.

A chaque branche réduite, est associée une fonction logique préfixée en terme d'opérateur AND/OR, dans laquelle apparaît ou non l'inversion des entrées. Par exemple, la branche réduite constituée des transistors {t4, t5, t6} aura pour fonction A2(E,O2(E,E)) ou 'A' définit le AND, 'O' le OR, '2' est le nombre d'entrée, et le `E' dénote une entrée non inversée. Chaque nouvelle fonction enrichit un catalogue de fonction, au départ vide, et la fonction complémentaire est générée : Par exemple, la complémentaire de A2(E,O2(E,E)) donne O2(I,A2(I,I)).

Pour chaque noeud extrémité à la fois d'une branche de type 1 et 2, LOGEX teste si les deux fonctions sont complémentaires, compte tenu de la place de chacune des entrées des branches. Quand c'est le cas, LOGEX abstrait l'ensemble des deux branches par une cellule dont la fonction est donnée par l'inverse de la fonction du réseau N. La branche vers VDD est supprimée.

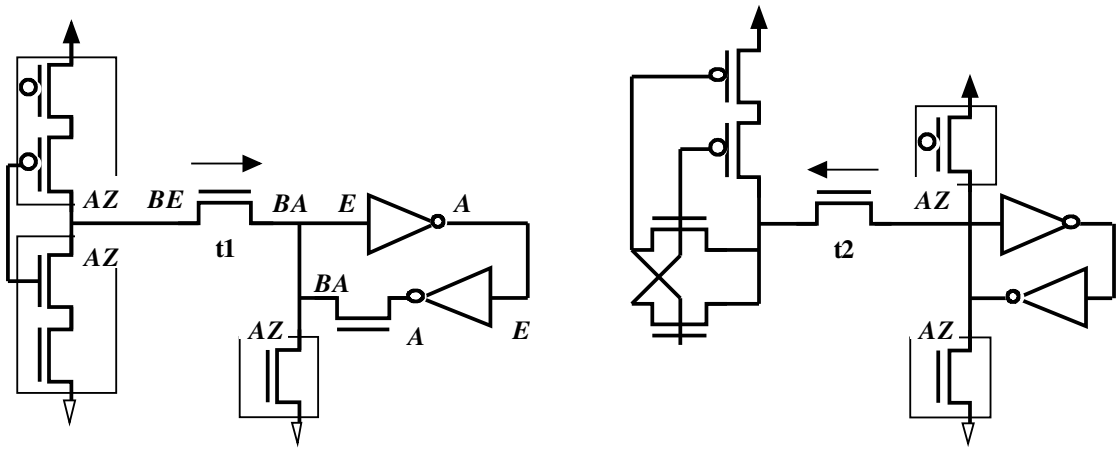

Fig 3.7-a *Fig 3.7-b*

Fig 3.7 : Orientation des transistors par LOGEX

LOGEX tente ensuite de diriger les branches bidirectionnelles restantes, en s'appuyant sur un ensemble d'heuristiques.

- L'extrémité d'une branche bidirectionnelle reliée à la sortie d'une cellule (au sens LOGEX) est l'entrée de la branche.
- L'extrémité d'une branche bidirectionnelle reliée à l'entrée d'une cellule (au sens LOGEX) est la sortie de la branche, si aucune autre extrémité de branche bidirectionnelle n'y est connectée.

Pour diriger les branches bidirectionnelles restantes, une *force* est affectée aux extrémités des branches. Voici leurs valeurs en ordre décroissant :

1. *A*, sortie toujours active (sortie de cellule)

2. *B*, sortie d'un inverseur, entrée d'un autre inverseur
3. *AZ,* extrémité de type sortie trois états d'une branche VDD ou VSS
4. *BZ,* extrémité d'une branche bidirectionnelle.
5. *BA, BZ* changée en extrémité de type sortie trois états par l'application d'une règle d'orientation
6. *BE, BZ* changée en extrémité de type entrée par l'application d'une règle d'orientation
7. *E*, entrée d'une cellule, grille d'un transistor d'une branche.

En considérant, en nombre et en valeur, les forces des extrémités des branches reliées respectivement au drain et à la source d'un transistor jusqu'alors bidirectionnel, il est possible de l'orienter.

Le transistor t1 de la figure 3.7.a (inspiré de [Boe88]) est orienté parce qu'il est relié à deux branches *AZ* d'un côté, et à une seule branche *AZ* de l'autre.

Ceci fait, LOGEX génère une net-list de portes, accompagnée du comportement de chacune des portes référencées dans la net-list. Toutes les branches qui n'ont pas fait l'objet d'une transformation en cellule duale, sont décrites comme des portes trois états.

LOGEX, par son approche générale, tente de supprimer toute référence à un catalogue de cellules prédéfinies ce qui est un avantage décisif Bien qu'il fasse de la réduction série/parallèle, le problème de l'orientation des transistors, est abordé et les règles mises en oeuvre sont satisfaisantes dans bon nombre de cas.

Cependant elles deviennent rapidement inapplicables dans les circuits contenant de la logique à transistors de passage. En remplaçant l'inverseur trois états par un Nxor à transistors de passage, et en ajoutant une commande de mise à un au latch, on obtient le circuit de la figure 3.7.b. Dans ce cas, le transistor t2 sera orienté dans le mauvais sens. Citons aussi pour exemple, le cas de la chaîne carry-manchester qui tient ces règles en échec.

La raison principale de leur inefficacité dans certains cas, est que LOGEX a une approche trop locale pour traiter ce problème. Si LOGEX savait mettre en évidence que le Nxor a sa sortie toujours active, il pourrait orienter le transistor dans le bon sens mais le partitionnement qui est choisi l'empêche d'avoir une vue globale de cette porte. Dans le cas de la chaîne d'anticipation de retenue, là aussi, l'information fonctionnelle qui permet d'orienter les transistors est ailleurs, en amont, et ne peut être déduite de la structure de la chaîne carry-manchester elle-même.

D'autres travaux proposent des solutions à l'orientation des transistors [Viv86] [Jou87] en s'appuyant sur des critères structurels ou électriques locaux. Ces règles se heurtent

cependant à des situations ambiguës qu'elles ne peuvent résoudre ou qu'elles interprètent mal. Dans tous les cas, ces règles traduisent des hypothèses sur la validité du circuit, alors que cela n'est pas acquis au moment ou elles sont appliquées (un signal est toujours généré, et reçu par exemple).

CCC

On trouve dans [Ste92] un système qui à partir d'une net-list de transistors, génère une description au niveau portes. Les portes sont caractérisées logiquement et temporellement.

Ce système utilise l'outil ANAMOS, qui est le pré processeur du simulateur switch-level COSMOS [Bry87], pour partitionner le circuit en CCC (Connected Channel Components) et pour les caractériser fonctionnellement. Un CCC est un ensemble de noeuds susceptibles d'être reliés électriquement à travers les canaux des transistors. La description en portes est directement issue de la décomposition en CCC. Les sorties d'un CCC sont les signaux qui attaquent les grilles de transistors appartenant à d'autres CCC.

A chaque sortie d'un CCC est associé un graphe représentant son comportement logique en fonctions des valeurs des entrées (X:H correspond à X=1, X:L correspond à X=0).

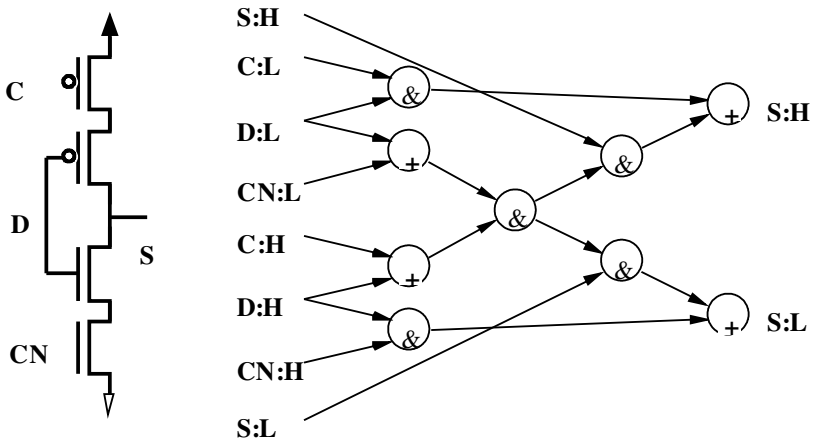

Fig 3.8 : CCC et le graphe d'évaluation généré par ANAMOS

Lorsque la sortie du CCC peut présenter un état Z, on retrouve celle-ci en tant qu'entrée dans le graphe. De cette façon, les conditions pour lesquelles on peut observer un effet mémorisant sur ce signal sont décrites par le graphe.

Le point fort de ce système est sa grande généralité que lui permet la technique de la simulation switch-level pour caractériser logiquement les CCC, et son aptitude à détecter les éléments mémorisants dynamiques.

En raison de sa généralité, cette modélisation du circuit en CCC interconnectés était déjà utilisée au sein d'autres outils. C'est le cas notamment d'un simulateur de fautes et d'un

générateur automatique de vecteurs de test [Pra87] qui partitionnent le circuit de cette façon.

En revanche, dans le cas d'ANAMOS, cette approche présente des difficultés lorsque le nombre d'entrées des CCC devient grand (colonne de micro ROM préchargée) du fait de l'explosion combinatoire des configurations d'entrées à analyser pour établir la table de transitions.

En effet, une table de transition est générée en évaluant le graphe pour toutes les configurations d'entrées possibles. Chaque fois que celle-ci montre un état haute impédance ou un conflit sur la sortie, une simulation électrique est effectuée, et la table est corrigée. Les sorties sont alors décrites par les états 0,1, et X ou X décrit généralement un conflit (statique ou dynamique)

D'autre part, l'analyse des CCC se faisant hors contexte, des simulations électriques coûteuses en performance, sont fréquemment invoquées pour caractériser des états haute impédances ou des conflits qui n'en sont pas (dans l'exemple de la figure 3.8, il n'y a pas de conflit sur S si CN = not (C)). Ces simulations ne peuvent pas mettre en évidence qu'un chemin électrique n'est jamais passant compte tenu de la logique en amont.

Par ailleurs, il existe des configurations pour lesquelles il faut indiquer la direction des transistors de passage à ANAMOS pour que son analyse soit correcte (Xor à transistors de passage, par exemple, dont la sortie n'est pas décrite par une fonction complète). Enfin, les portes trois états sont le plus souvent interprétées comme des éléments mémorisants dynamiques.

Ce système produit donc une description hiérarchique simulable au niveau switch-level. Cependant, le partitionnement obtenu n'est pas toujours satisfaisant car les CCC peuvent comporter des branches qui ne jamais passantes. La présence de ces fausses branches altère la description comportementale en y introduisant des faux conflits, et crée de fausses dépendances entre des signaux du circuit. Enfin, cette approche ne fournit pas de description comportementale du circuit.

DESB

DESB [Lau92] [Lau94] a été développé au laboratoire LIP6. Il fait partie des premières versions de la chaîne ALLIANCE [Gre92]. Cet un outil d'abstraction fonctionnelle produit à la fois une net-list de portes et une description RTL du circuit abstrait. Il prend en entrée une description au niveau transistors.

DESB est basé sur une méthode formelle de reconnaissance des portes. Le point de départ de l'algorithme est un signal relié à une grille de transistor. En partent de ce signal, DESB construit tous les chemins qui relient ce signal à VDD ou à VSS à travers le canal des transistors. Les chemins qui mènent à VDD permettent de construire l'expression de

mise à un du signal (F_{up}). Les chemins qui mènent à VSS constituent l'expression de mise à zéro (F_{down}). DESB utilise les BDD pour représenter les expressions booléennes.

Pour les portes combinatoires, F_{up} est la fonction inverse de F_{down}. Par contre, si ces deux fonctions ne sont pas complémentaires, il peut y avoir un conflit (F_{up} *and* F_{down} = 1) ou une condition de non conduction ou de haute impédance (F_{up} *or* F_{down} = 0). Dans ce cas DESB effectue une analyse plus poussée pour tenter de définir s'il s'agit d'un vrai conflit (ou d'une vraie condition de haute impédance). Cette analyse consiste à vérifier le contexte d'utilisation de la porte. En effet, les entrées de la porte peuvent être corrélées de telle manière que la condition de conflit (ou de haute impédance) ne puisse jamais se produire. Cette analyse consiste donc à remplacer dans la fonction de conflit (F_{up} *and* F_{down}) ou de haute impédance (F_{up} *or* F_{down}) les signaux par leur expression. Si après expansion, le conflit persiste, DESB remonte recommence le processus d'expansion. Les points d'arrêt de cette expansion sont, bien entendu, les entrées du circuit ou les points mémorisant. Il est donc indispensable d'identifier correctement les points mémorisants.

Une boucle combinatoire est symptomatique d'un point mémorisant Dans DESB, les points mémorisants sont reconnus en utilisant une technique de pattern matching. De plus, le pattern à reconnaître est câblé dans le programme et ne permet de reconnaître qu'un seul type de point mémoire.

Un autre point faible de DESB réside dans le processus d'expansion. En effet, si la corrélation entre les signaux se trouve très en amont, l'expansion des BDD aboutit à une explosion combinatoire.

Enfin, DESB construit entièrement les chemins de mise à 1 et de mise à zéro d'un signal, avant de commencer l'analyse booléenne. Par conséquent, une porte peut comporter un très grand nombre de chemins au départ même si l'analyse booléenne va révéler par la suite que ces chemins ne sont pas fonctionnels. De ce fait, il peut également y avoir une explosion du nombre des chemins.

YAGLE

YAGLE [Les98] [Les99] a également été développé au laboratoire LIP6 et distribué avec la chaîne ALLIANCE. Actuellement, cet outil continue à être développé et est commercialisé pour la société AVERTEC [Ave98].

YAGLE est le successeur de DESB. Le principe de fonctionnement de YAGLE est similaire à DESB. Cependant, YAGLE met en oeuvre certains mécanisme pour répondre aux faiblesses de DESB. YAGLE est composé de 5 modules.

Le premier reconnaît les portes CMOS duales (les portes pour lesquelles F_{up} *et* F_{down} sont complémentaires). Ce module permet d'abstraire la majeure partie du circuit.

Le second module traite les portes non duales. Mais, contrairement à DESB, l'étape d'expansion des fonctions F_{up} *et* F_{down} s'effectue en même temps que la construction des chemins. Ceci permet d'éviter de construire les chemins qui ne sont pas statiquement fonctionnelles.

Comme dans DESB, la reconnaissance des éléments mémoire passe par le pattern matching. C'est le rôle du module FCL de YAGLE. Dans YAGLE, contrairement à DESB, les schémas à reconnaître ne sont pas câblés dans le code mais décrits par des fichiers. De plus, YAGLE comporte un module mettant en oeuvre une technique formelle pour reconnaître les éléments mémoire. Cette technique est basée sur la dérivation booléenne et permet de reconnaître les éléments mémoires où la fonction d'écriture peut être analysée logiquement (sans analyse électrique).

Le dernier module de YAGLE est GENIUS. Il s'agit d'un module de reconnaissance de blocs. Ce module est placé après FCL et permet d'identifier des assemblages de portes reconnues par FCL. GENIUS associe un modèle abstrait à un bloc reconnu. La description des blocs est faite à l'aide d'un fichier de règles.

4 Abstraction temporelle

Quelle que soit l'approche utilisée pour l'abstraction fonctionnelle, le résultat est une interconnexion de portes. L'analyse temporelle prend en entrée ce résultat. Du point de vue de cette analyse, il est extrêmement important que les caractéristiques temporelles d'une porte puissent être déterminées indépendamment de son environnement, c'est à dire des portes auxquelles elle est reliée. Autrement dit, pour que l'analyse temporelle soit réalisable de manière efficace, il faut que les portes reconnues par l'abstraction fonctionnelle soit électriquement isolées les unes des autres.

L'abstraction temporelle consiste alors à enrichir le réseau de portes de caractéristiques temporelles. Elle comporte deux aspects totalement distincts : le modèle d'abstraction et le modèle temporel.

Le modèle d'abstraction définit le modèle attaché à chaque porte à l'issue de l'abstraction temporelle. Autrement dit, le modèle d'abstraction définit la représentation d'une porte du point de vue de l'abstraction temporelle. Ce modèle comporte des caractéristiques temporelles telles que des délais de transition ou la durée des transitions.

Le modèle temporel définit comment les caractéristiques temporelles d'une porte sont-elles calculées.

4.1 Modèles d'abstraction

D'une manière générale, du point de vue de l'abstraction temporelle, on peut utiliser le paradigme des automates d'états pour représenter une porte. Ces automates sont habituellement appelés les STG (State Transition Graph).

Dans ces graphes, les états représentent l'état des signaux d'entrée-sortie d'une porte. A chaque transition sont attachées des informations temporelles telles que le délai de transition minimal, le délai maximal et éventuellement, le temps de la transition (modélisé par exemple par la pente de la transition). Ce triplet (D_{min} , D_{max} , *Slope*) caractérise la transition du point de vue temporelle.

On peut distinguer trois types de STG pour modéliser une porte.

Le modèle dit de l'inverseur généralisé

Il s'agit du STG le plus simple que l'on puisse imaginer. Les états de l'automate se différencient par l'état du signal de sortie. Chaque porte est donc modélisée par un automate d'états à 2 états (sans tenir compte de la fonctionnalité de la porte).

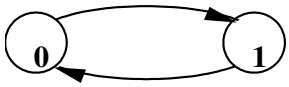

Fig 4.1 : Modèle STG de l'inverseur généralisé

La transition de 0 vers 1 représente le front montant et la transition de 1 vers 0 le front descendant. Bien évidemment, dans ce modèle ne figure pas l'origine de la transition de la sortie. Autrement dit, on ne peut pas distinguer les caractères temporels suivant l'entrée qui a provoqué le changement de l'état de la sortie. Lorsque l'on utilise ce modèle dans l'analyse temporelle d'un circuit, la méconnaissance de l'origine de la transition de la sortie peut représenter un inconvénient majeur. Ainsi, si l'on considère l'exemple de la figure 4.2 et si l'on fait l'hypothèse que le délai de propagation maximal sur la porte NAND provient de la transition de C, il apparaît clairement que le chemin critique le plus long du circuit va être toujours grossièrement surestimé. De même, le chemin critique le plus court du circuit sera toujours sous-estimé.

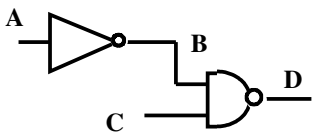

Fig 4.2 : Exemple d'interconnexion de portes pour l'analyse temporelle

Le modèle STG Entrée-Sortie

Ce modèle permet de pallier le défaut du modèle de l'inverseur généralisé. Dans ce modèle, chaque porte est représentée par plusieurs STG. On associe un STG à chaque

couple entrée-sortie. Une porte à N entrées sera donc modélisée par N STG. Un état de l'automate est spécifié par l'état de l'entrée et de la sortie. Potentiellement chaque STG comporte donc 4 états. Naturellement, pour une porte particulière, compte tenu de sa fonction booléenne, tous les états ne sont pas accessibles. Par exemple, pour une porte NAND, il est impossible d'avoir à la fois une entrée et la sortie à zéro.

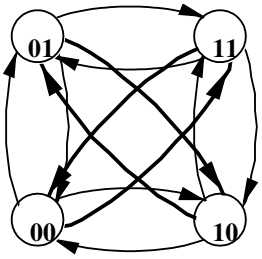

Fig 4.3 : Modèle STG entrée-sortie. Les transitions en gras sont caractérisées

Dans ces STG, seul les transitions qui incluent une transition de la sortie sont temporellement caractérisées. Pour obtenir ces caractéristiques il faut imposer une certaine configuration aux autres entrées de la porte de telle manière que la transition de l'entrée provoque une transition de la sortie. On dit qu'il faut rendre la porte *sensible* à la transition de l'entrée. Par exemple, dans le cas d'un NAND à deux entrées (figure 4.4), dans le STG AS, pour obtenir une transition de la sortie, il faut que B soit égale à 1. Bien entendu, les caractéristiques temporelles dépendent de la configuration choisie. Ainsi, chaque composante de la caractéristique temporelle doit-elle être calculée pour la configuration la plus défavorable.

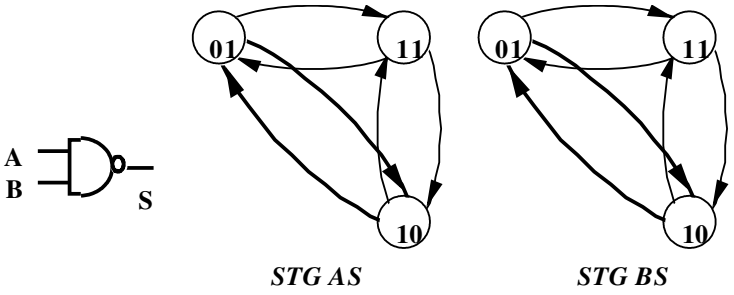

Fig 4.4 : Modèle STG d'une porte Nand à 2 entrées

L'utilisation de ce type de modèle permet d'éviter de faire la somme de délai les plus longs pour calculer le chemin critique d'un circuit. En reprenant l'exemple de la figure 4.2, on voit bien que le délai de transition de A vers D sera bien la somme des délais de A vers B et de B vers D.

Cependant, même en utilisant ce modèle, l'analyse temporelle d'un circuit peut être faussée. En effet, la transition d'une entrée vers la sortie est caractérisée pour la configuration la plus défavorable. Or, dans un circuit, il arrive fréquemment que les entrées d'une série de portes soient corrélées. Dans ce cas, la configuration la plus

défavorable pour une porte peut rendre la porte suivante insensible à la transition de l'entrée.

La figure 4.5 illustre ce phénomène. On s'intéresse à la transition de A vers F. Si l'on suppose que la configuration la plus défavorable pour la transition de B vers D impose C=1 et D=0, alors la corrélation entre D et E interdit la transition de D vers F.

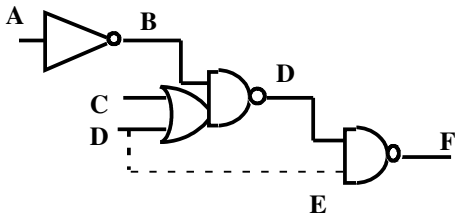

Fig 4.5 : Exemple d'interconnexion de portes pour l'analyse temporelle

Le modèle STG complet

Ce modèle permet de prendre en compte la corrélation entre les signaux dans l'analyse temporelle d'un circuit. Dans ce modèle, à une porte à N entrées est associé un STG comportant 2^N états. Chaque état correspond à une configuration des entrées (l'état indique également l'état de la sortie). Le STG correspondant à une porte NAND à deux entrées contient donc 4 états. Comme toujours, seul les transitions qui comportent une transition de la sortie sont temporellement caractérisées.

Ce modèle prend en compte non seulement, la transition de la sortie provoquée par la transition d'une entrée, mais peut également représenter la transition simultanée de plusieurs entrées. L'inconvénient majeur de ce modèle réside dans sa complexité exponentielle. De plus, la caractérisation temporelle des transitions nécessite une quantité de calcul importante.

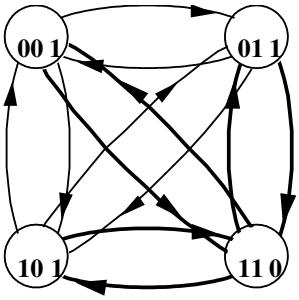

Fig 4.6 : STG complet d'un Nand à 2 entrées. Les transitions en gras sont caractérisées

4.2 Modèles temporels

Le modèle d'abstraction d'une porte comporte des caractéristiques temporelles. Quel que soit le type de STG choisi, le modèle temporel définit comment ces caractéristiques peuvent-elles être obtenues.

Le point de départ pour parvenir à ces caractéristiques est le schéma en transistors des portes. L'abstraction fonctionnelle doit donc préserver ce schéma et les informations qu'il contient (taille de transistors). Mais, ce schéma à lui seul ne permet pas de calculer les informations temporelles de manière précise.

Les fils d'interconnexion qui relient la sortie d'une porte à l'entrée d'une autre porte ont une importance fondamentale pour l'analyse temporelle. Dans les technologies de fabrication récentes, les fils ne peuvent pas être considérés comme des équipotentiels. La finesse de gravure dans ces technologies fortement submicroniques est telle que la résistance des fils d'interconnexion ne peut pas être négligée. Un fil est donc vu comme un réseau de résistances et de capacités. De ce fait, les caractéristiques temporelles des portes dépendent, bien sûr, de la structure en transistors de la porte mais aussi et surtout du réseau RC des fils d'interconnexion.

Bien entendu, le réseau RC à la sortie d'une porte varie suivant la porte réceptrice. Dans l'exemple de la figure 4.7, le délai de transition de A vers B est différent du délai de transition de A vers C à cause de la différence entre les réseaux RC qui relient la sortie de l'inverseur de A à B et à C. Il convient donc d'associer un modèle d'abstraction non pas à chaque porte mais à chaque récepteur de la sortie d'une porte. Dans, l'exemple de la figure 4.7, deux STG seront associés à l'inverseur : un pour caractériser les transitions de B et un pour les transitions de C.

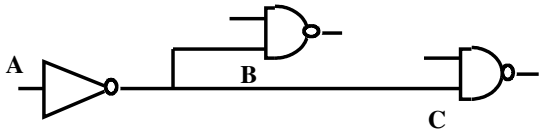

Fig 4.7 : Transition d'un signal relié à plusieurs portes

Le modèle temporel prend donc en compte le schéma en transistors d'une porte et le réseau RC du fil d'interconnexion. En fonction de la transition montante ou descendante d'une ou de plusieurs entrées et d'une certaine configuration des autres entrées, le modèle temporel fournit les caractéristiques temporelles de la transition de la sortie.

Deux solutions sont envisageables.

Modèles temporels analytiques

Les caractéristiques temporelles d'une transition d'une porte, par exemple les délais de propagation, dépendent du courant qui traverse les transistors lorsqu'un chemin de conduction se crée vers VDD ou vers VSS.

L'approche analytique consiste à établir une équation qui définit le courant drain-source (I_{DS}) du transistor en fonction de la tension de la grille (V_{GS}) et du drain (V_{DS}) par rapport à la source. Mais, un vrai transistor comporte également des éléments parasites tels que des capacités de couplage, des courants de fuites, etc. Ainsi, dans ces modèles analytiques, un transistor est remplacé par un schéma équivalent comportant une source de courant

contrôlé par la tension de la grille et du drain, des capacités et des résistances. Le problème revient alors à établir le système d'équations différentielles qui représente la configuration de la porte lors de la transition de la sortie. Si l'on parvient à résoudre analytiquement ce système, on obtient une équation qui définit la forme de la transition de la sortie en fonction de la transition d'entrée.

La figure 4.8 montre l'exemple d'un inverseur et un schéma équivalent associé. Dans ce cas, la transition montante de l'entrée A provoque une transition descendante de la sortie. Cette configuration est caractérisée par un système de deux équations différentielles.

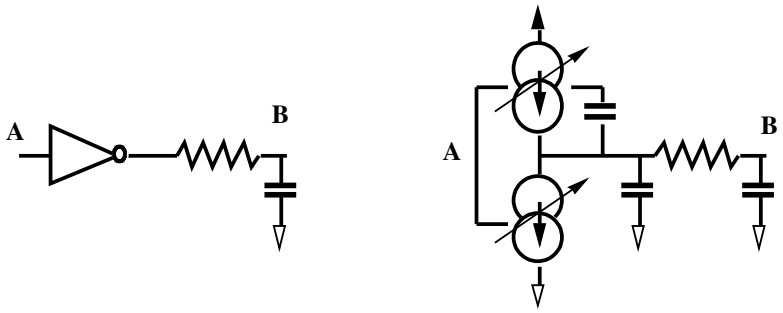

Fig 4.8 : Une porte et un schéma équivalent associé

La méthode d'analyse temporelle proposée par TAS [Haj91] [Haj92] [Dio98] suit cette approche. Cet outil a été développé laboratoire LIP6.

La difficulté de cette technique réside dans la définition du schéma équivalent à un transistor et dans la définition de l'équation du courant I_{DS}. En effet, pour établir ce modèle temporel, on est confronté à deux contraintes contradictoires. D'un côté, il faut que le schéma et l'équation reproduisent aussi fidèlement que possible le comportement du transistor. De cette précision dépend la précision de la caractérisation temporelle. La recherche de cette précision augmente la complexité du modèle. D'un autre côté, le modèle temporel doit être suffisamment simple pour permettre de résoudre le système d'équations différentielles.

Une autre difficulté provient de la paramétrisation du modèle temporel en fonction de la technologie. En effet, le modèle temporel contient un certain nombre de paramètres constants qu'il faut fixer pour se rapprocher au mieux du comportement d'un transistor pour une technologie donnée. L'obtention du modèle temporel passe donc par la définition d'une méthode permettant de calculer le meilleur un jeu de paramètres.

Simulation électrique

Une alternative à la définition d'un modèle temporel analytique est la simulation électrique. Cette approche consiste à effectuer une simulation électrique de la porte et du réseau d'interconnexion pour la configuration désirée. Les caractéristiques temporelles sont ensuite extraites du résultat de la simulation.

La version de TAS commercialisée par AVERTEC met en oeuvre cette technique. Cette approche a l'avantage d'utiliser directement les modèles électriques des transistors tels qu'ils sont définis par le fondeur. De plus elle se base sur les outils de simulation électrique existants et reconnus tels que ELDO [Eld00] ou SPICE. Enfin, la simulation s'effectue sur un petit nombre d'éléments (les transistors qui composent une porte et le réseau RC de l'interconnexion). De ce fait, le temps nécessaire pour effectuer ces simulations reste raisonnable.

5 Références

[Apt82] R. Apte, N. S. Chang, J. A. Abraham - "Logic Function Extraction for Nmos Circuits" - Proceedings of the International Conference on Circuits and Computer - IEEE 1982

[Ave98] www.avertec.com

[Boe88] M. Boehner - "LOGEX An Automatic Logic Extractor from Transistor Level for CMOS Technology" - Proceedings of the 25th Design Automation Conference - IEEE 1988

[Bry86] R. Bryant - "Graph-Based Algorithms for Boolean Function Manipulation" - IEEE Transaction on computers - IEEE August 1986

[Bry87] R. E. Bryant - "Boolean Analysis of MOS Circuits" - IEEE Transactions on Computer Aided Design - IEEE 1987

[Bur83] G. Burrow, R. Apte - "Topology Reduction for Cmos and Dynamic Nmos VLSI Circuits" - Proceedings of the International Conference on Computer Aided Design - IEEE 1983

[Dev92] P. Deverchere, J. C. Madre, J. B. Guignet, M. Currat - "Functional Abstraction and Formal Proof of Digital Circuits" - Proceedings of the European Conference on Design Automation EDAC92 - IEEE 1992

[Dio98] K. Dioury - "Analyse temporelle hiérarchique des circuits VLSI à très haute densité d'intégration" - Thèse de l'Université Paris 6 - Septembre 1998

[Eld00] http://www.mentor.com/

[Gre92] A. Greiner, F. Pecheux - "Alliance, A Complete Set of CAD Tools for Teaching VLSI Design" - Proceedings of the 3rd Eurochip Workshop on VLSI Design Training - 1992

[Haj91] A. Hajjar, A. Greiner, R. Marbot, P. Kiani - "TAS, an Accurate Timing Analyser for CMOS VLSI" - Proceedings of the European Conference on Design Automation EDAC'91 - IEEE 1991

[Haj92] A. Hajjar - "Modélisation des temps de propagation et analyse temporelle statique des circuits intégrés CMOS" - Thèse de Université Paris 6 - 1992

[Hui88] J. A. Huisken et al. - "Design of DSP Systems using the PIRAMID Library and Design Tools" - VLSI Signal Processing III - IEEE Press 1988

[Jou87] N. P. Jouppi - "Derivation of Signal Flow Direction in MOS VLSI" - IEEE Transactions on Computer Aided Design - IEEE May 1987

[Kaw82] M. Kawamura, K. Hirabayashi - "Logical Verification of LSI Mask Artwork by Mixed Level Simulation" - Proceeding of the International Symposium on Circuits and Systems - IEEE 1982

[Kis83] A. Kishimoto, K. Mori, S. Takashi, H. Kawanishi - "A Logic Function Extraction Algorithm for MOS VLSI" - Proceedings of the International Conference on Computer Aided Design - IEEE 1983

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

[Kos91] T. Kostelijk, B. De Loore - "Automatic Verification of Library-Based IC Designs" IEEE Journal of Solid-State Circuits - 1991

[Lau92] M. Laurentin, A. Greiner, R. Marbot - "DESB, a Functional Abstractor for CMOS VLSI" - Proceedings of the European Design Automation Conference EURO-DAC'92 - IEEE 1992

[Lau94] M. Laurentin - "Abstraction fonctionnelle des circuits intégrés CMOS" - Thèse de l'Université Paris 6 - Décembre 1994

[Les98] A. Lester Anthony, P. Bazargan-Sabet, A. Greiner - "YAGLE, a Second generation Functional Abstractor for CMOS VLSI Circuits" - Proceedings of the International Conference on Microelectronics -1998

[Les99] A. Lester - "Abstraction Fonctionnelle des Circuits Numériques VLSI avec une méthode formelle basée sur une extraction de réseau de portes" - Thèse de l'Université Paris 6 - Décembre 1999

[Lue84] F. Luellau, T. Hoepken, E. Barke - "A Technolgy Independent Block Extraction Algorithm" - Proceedings of the 21t h Design Automation Conference - IEEE 1984

[Mur88] J. Y. Murzin - "FAON, a Functional Abstractor of Netlist" - Bull Internal Report. Les Clayes sous Bois 1988

[Mu88b] J. Y. Murzin, J. B. Guignet - "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" - Proceedings of COMPEURO - 1987

[Pra87] S. Pravossoudovitch - "Contribution au test des circuit intégrés MOS : Génération Automatique de Vecteurs de Test au niveau Transistors Interrupteurs" - Thèse d'état de l'université des sciences et techniques du Languedoc - 1987

[Ste92] R. Stewart, V. Anjubault, P. Garcin, J. 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

[Tak82] M. Takashima, T. Misthuhashi, T. Chiba, K. Yoshida - "Programs for Verifying Cicuits Connectivity of MOS/LSI Mask Artwork" - Proceedings of the 19th Design Automation Conference - IEEE 1982

[VGi86] J. Van Ginderdeuren et al. - "A High Quality Digital Audio Filter Set Designed by Silicon Compiler CATHEDRAL-1" - IEEE Journal of Solid State Circuits, December 1986

[Viv86] C. Vivier, G. Fournie - "Automatic Modelling of CMOS Transistor Networks for Test Pattern Generation" - Proceedings of the International Test Conference - IEEE 1986