Titre: Title:	Conception d'une bibliothèque et d'un convolueur 3*3TSPC
Auteur: Author:	Bertrand Le Chapelain
Date:	1999
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Le Chapelain, B. (1999). Conception d'une bibliothèque et d'un convolueur 3*3TSPC [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8856/

UNIVERSITÉ DE MONTRÉAL

CONCEPTION D'UNE BIBLIOTHÈQUE

ET

D'UN CONVOLUEUR 3*3 TSPC

BERTRAND LE CHAPELAIN
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GENIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
JUIN 1999

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

CONCEPTION D'UNE BIBLIOTHÈQUE TSPC

ET

D'UN CONVOLUEUR 3*3 TSPC

présenté par: Le Chapelain Bertrand

en vue de l'obtention du diplôme de: Maîtrise ès Sciences Appliquées

a été dûment accepté par le jury d'examen constitué de:

M. SAWAN Mohamed, Ph. D., président

M. BOIS Guy, Ph. D., membre et directeur de recherche

M. SAVARIA Yvon, Ph. D., membre et directeur de recherche

M. BLAQUIÈRE Yves, Ph. D., membre

REMERCIEMENTS

Je désire remercier mon directeur de recherche M. Guy Bois pour son attention, ses conseils et sa disponibilité en tout temps. Son appui et sa compréhension, tout au long de ma maîtrise, m'ont permis de me sentir à l'aise pour discuter ouvertement de tout problème. J'aimerais également exprimer ma gratitude envers mon codirecteur de recherche, M. Yvon Savaria pour m'avoir accepté dans son groupe de recherche en microélectronique. Je le remercie d'avoir trouvé le temps de suivre le déroulement de mes travaux et de les avoir orientés dans de nouvelles directions. Le partage de sa vision des choses, en particulier pour les applications de mes travaux, a été une source de réflexion constante. Je les remercie pour leur soutien matériel, ainsi que Design Workshop et Hyperchip pour m'avoir financé lors de mes études.

Je tiens également à remercier les nombreuses personnes qui ont contribué aux différents projets de ma maîtrise lors d'application de cours et de stages. Kamel Ridouh pour m'avoir formé sur la plupart des outils de la microélectronique, Youcef Fouzar, Alexandre Méchain, Robert Chehi, Jean-Christophe Petit et Ginette Monté pour leur participation au convolueur. Jérome Laporte et Bernard Antaki pour m'avoir aidé à développer des bibliothèques de cellules TSPC.

Enfin, je profite de l'occasion pour dire merci à tous les étudiants du laboratoire GRM94 qui ont pu m'offrir leurs conseils lors des deux dernières années, notamment Mathieu Gagnon pour ses nombreux conseils sur Cadence et Réjean Lepage pour le maintien d'un réseau stable.

RÉSUMÉ

Dans ce mémoire, nous présentons une heuristique des bascules bâtie selon le style de conception des circuits à horloge monophasée (True Single Phase Clock, TSPC). Des études précédentes ont montré que les circuits TSPC peuvent opérer à de très grandes fréquences. Notre but est d'implémenter de gros blocs logiques qui peuvent opérer à 1 GHz ou plus (en technologie CMOS 0.35 um). Notre stratégie a débuté avec le développement d'une bibliothèque de base selon une méthode ad-hoc. Une méthode rapide semi-automatique a alors été développée pour réduire le nombre de simulations. Basé sur l'expérience acquise lors des optimisations TSPC, nous proposons un algorithme d'optimisation automatique des cellules TSPC et une méthode pour réaliser des bibliothèques de cellules logiques standards. Ce mémoire décrit les différentes étapes à suivre pour élaborer une bibliothèque TSPC. La méthode et la bibliothèque de cellules sont validées par la caractérisation de petits blocs logiques.

Une version TSPC (True Single Phase Clock) très haute fréquence d'un convolueur 3*3 est également développée. La régularité de l'opération de convolution en fait une candidate idéale pour une réalisation basée sur la technique de conception TSPC. Ce convolueur TSPC permet d'entrevoir les limites de performance d'un coprocesseur conçu à l'aide d'une technologie CMOS à 0.35 microns.

ABSTRACT

In this memory, we present a heuristic study of split-output latches built according to the true single-phase clocked circuits (TSPC) design style. Previous studies have shown that TSPC circuits can operate at very high frequencies. Our goal is to implement large digital building blocks operating at 1 GHz or more (in 0.35 µm CMOS technology). Our methodology started with the development of a first set of cells based on ad-hoc design. A fast semi automatic method was then developed to reduce the number of simulations. Based on the experience acquired with TSPC optimization, we proposed an automatic algorithm to optimize TSPC cells and a method to realize libraries of logical standard cells. This memory describes the different steps followed to elaborate the TSPC library. The method and the cell library are validated by characterizing small digital building blocks.

A very high speed TSPC (True Single Phase Clock) version of the 3*3 convolver's datapath is also shown. The regularity of the 2-D convolution makes it an ideal candidate for the fine-grained pipelining associated with the TSPC scheme. The design of the TSPC convolver shows the performance limits which can be achieved by a coprocessor based on a CMOS 0.35 micron technology.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION

Les concepteurs de microprocesseurs essaient continuellement d'augmenter les performances des machines qu'ils réalisent. Actuellement, une façon d'améliorer les micro-architectures est d'augmenter la granularité du pipeline afin de diminuer le nombre de cycles d'horloge des instructions. Les processeurs fonctionnent à des fréquences très élevées, maintenant à 500 MHz et bientôt à 1 GHz. Cette croissance des fréquences d'opération est principalement dûe à une recherche très efficace dans le développement de nouvelles technologies CMOS offrant maintenant des résolutions de 0.25 μm et 0.18 μm, et de 0.15 μm dans un an.

À la conférence IEEE International Solid-State Circuits en février 1999, le laboratoire d'IBM d'Austin [1] a présenté un processeur expérimental à 64 bits appelé guTS (gigahertz unit Test Site). Le but de ce processeur était de démontrer que des techniques de conception de circuit couplées à un centrage de ces circuits pouvaient augmenter de façon significative la performance des microprocesseurs, donc d'ouvrir une nouvelle voie de recherche pour contribuer à augmenter la performance des microprocesseurs et de la technologie CMOS. Pour clairement distinguer les améliorations découlant des méthodes de conception de celles découlant des nouvelles technologies de fabrication, une technologie de 1997 avait été choisie (CMOS 0.25 μm 1.8 V). Maintenant encore, une telle perfor-

mance n'est pas accessible même avec les dernières technologies 0.18 μm, en utilisant des techniques de conception conventionnelles.

Le processeur "guTS" est un circuit dédié, composé presque à 100% de circuits dynamiques. Il est capable d'exécuter 96 instructions entières du jeu d'instruction de l'architecture Power PC. Toutes les instructions, dont les chargements et les rangements, sont exécutées en un seul cycle. Ce processeur montre qu'il est possible de concevoir des circuits de très hautes performances sans être dépendant exclusivement de nouvelles technologies pour progresser.

Les cellules dynamiques utilisées par IBM étaient des cellules Domino. Nelson F. Goncalves et Hugo J. De Man [2] ont élaboré des techniques de conception de cellules dynamiques NORA, pour des structures logiques pipelinées. Dans les circuits conventionnels, il existe une redondance d'information. À chaque dispositif de type N correspond un dispositif de type P. En fait, une fonction logique complète est fabriquée avec des dispositifs de type N, doublés de dispositifs de type P. Ainsi une grande quantité de silicium est gaspillée, spécialement pour les réseaux logiques complexes. Un autre problème des réseaux de portes statiques CMOS provient des courses dans les circuits pipelinés. Pour propager l'information entre deux circuits pipelinés, des portes de transmission sont habituellement utilisées.

En logique CMOS, ces portes de transmission sont généralement implémentées avec deux transistors de type N et P en parallèles et contrôlées par des horloges ϕ et $\bar{\phi}$. L'utilisation

de simples portes de type P ou N est à éviter, à cause de la dissipation de puissance et de la faible marge de bruit. Ces portes de transmission de type N et P souffrent également de courses intenses. En effet, pendant la phase d'évaluation, toutes les portes de transmission sont activées, ce qui peut corrompre le flot d'information, dépendant du rapport entre le délai de la porte et le biais de synchronisation de l'horloge.

Dans la technique Domino, Krambeck et al. [3] ont résolu le problème en plaçant des inverseurs statiques après chaque bloc dynamique. Une limitation de la technique Domino est le manque de signaux inversés. La combinaison d'un bloc dynamique avec des inverseurs statiques donne un signal non inversé. Cette solution diminue la flexibilité logique et nécessite par conséquent plus de transistors pour une fonction logique. Le problème de course d'horloge n'est toujours pas résolu dans cette technique. La technique NORA permet de palier à ces problèmes. Les fonctions logiques sont alors implémentées à partir de cellules dynamiques CMOS de type N et P et de fonctions C^2MOS. Pour résoudre les problèmes de courses entre signaux dans les circuits pipelinés, l'information doit toujours être enregistrée dans une fonction C^2MOS. De plus, les signaux inversés et non inversés sont préservés. Quand des couplages entre des blocs dynamiques sont désirés, les fonctions logiques sont implémentées par l'alternance de blocs N et P. Les blocs de même type peuvent communiquer à l'aide d'inverseurs comme dans la technique Domino.

Un circuit pipeliné Nora est insensible aux problèmes de course si le nombre total d'inversions (statiques et dynamiques) entre deux bascules C^2MOS est semblable.

Les cellules NORA sont limitées par des problèmes de partage de charge et de marge de bruit. Le signal de sortie des cellules dynamiques est relié aux noeuds de stockage. Lors d'une commutation d'un transistor d'un état inactif à un état actif, un effet de redistribution de charge peut apparaître entre la capacité de sortie et les capacités parasites du réseau logique. Ces limites sont communes à toutes les techniques dynamiques.

Cependant, avec la NORA, comme il n'existe pas de temps mort et de problème de biais de synchronisation, nous pouvons nous attendre à atteindre de plus hautes fréquences d'horloge qu'avec la technique C^2MOS. Un développement plus profond de la stratégie de distribution d'horloges devrait utiliser encore moins de signaux d'horloge. La technique dynamique CMOS à vraie horloge monophasée (TSPC) [4] utilise seulement un signal d'horloge qui n'est jamais inversé. Donc, il n'existe aucun biais entre une horloge et son complément, excepté pour les problèmes de délai de l'horloge, ainsi une plus grande fréquence d'horloge peut être atteinte. Comme nous l'expliquons ensuite, la conception à vraie horloge monophasée peut remplacer non seulement les cellules dynamiques mais également les cellules statiques et dans la plupart des cas la technique CMOS NORA.

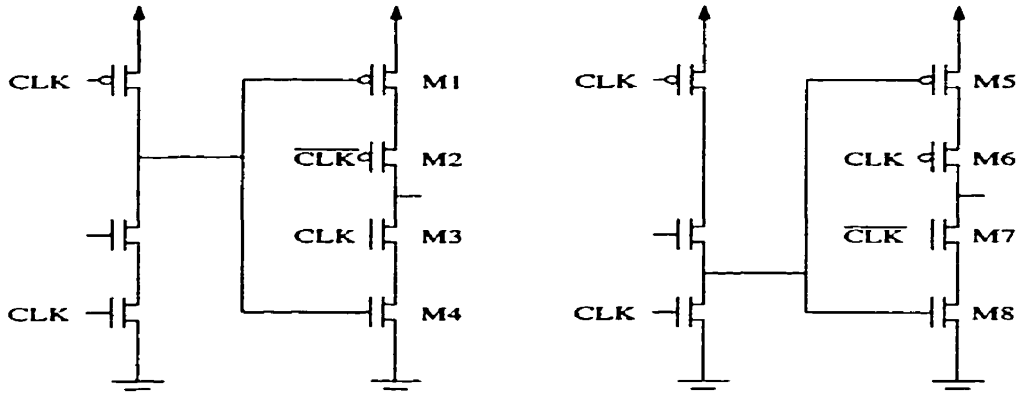


Figure 0.1: Technique dynamique NORA

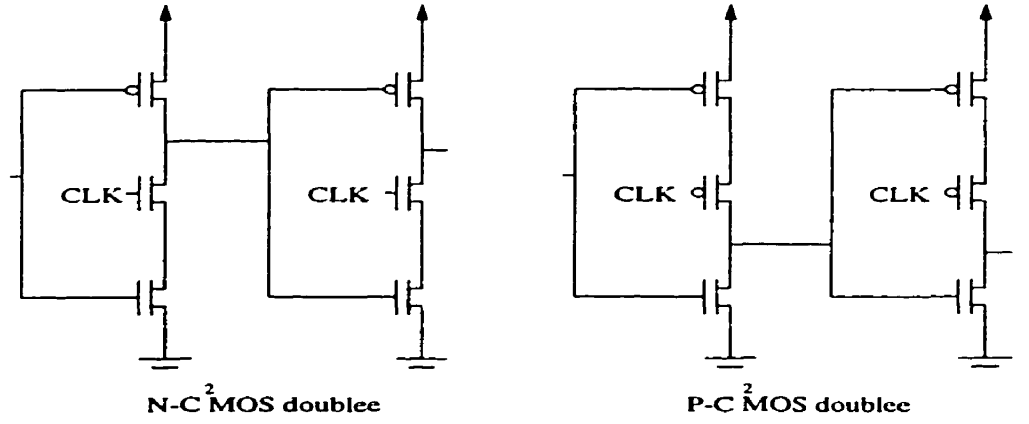


Figure 0.2: Bascules TSPC

Dans la Figure 0.1, nous avons deux bascules C^2MOS contrôlées par deux signaux d'horloge CLK et $\overline{CLK}$. La nécessité d'utiliser $\overline{CLK}$ réside dans le contrôle des transistors M2 et M7. Cependant, ceci peut être donné par un signal d'horloge CLK connecté aux deux bascules par des inverseurs comme nous le voyons à la figure 0.2. Nous appelons les deux différentes unités étage N-C^2MOS et étage P-C^2MOS, respectivement, et si nous utilisons

ces étages en série, ils deviennent des bascules TSPC. Les blocs logiques de type N et P sont utilisés alternativement avec le même signal d'horloge. Tant que le délai du signal d'horloge est inférieur au délai de la porte, le système est stable. Apparamment, c'est mieux que la stratégie de pseudo horloge à deux phases sans recouvrement, et aussi du point de vue des contraintes logiques, cela peut se comparer avec la stratégie d'horloge à deux phases NORA. Il est également possible de n'utiliser que des portes TSPC avec de la logique NMOS si les blocs sont séparés par des inverseurs une fois sur deux. Les blocs logiques TSPC sont expliqués davantages dans le chapitre 1. Le principal problème des cellules dynamiques est de pouvoir les utiliser dans des outils de synthèse afin de concevoir rapidement des circuits de taille respectable. Il est possible de les utiliser dans des circuits dédiés ou dans les points critiques des gros circuits. Edefors [5] a proposé un outil afin de réaliser la synthèse et le placement et routage des cellules TSPC. Une bibliothèque de base a été conçue de façon dédiée en respectant les sortances. Ces cellules sont compatibles avec une horloge rapide qui impose de nombreuses contraintes sur les règles de conception. Chaque cellule est balancée, avec des temps de montée et de descente semblables. De plus toutes les cellules ont des délais similaires. D'autre part, entre chaque cellule, il faut mettre des longueurs de fil comparables. En effet, les fils engendrent, à haute fréquence, une capacité non négligeable qui peut empêcher les signaux d'arriver en même temps si les distances entre les interconnexions sont trop importantes.

La dissipation de puissance est devenue un important domaine de recherche dans les circuits numériques, surtout dans les équipements portables. Ainsi, de nombreuses techniques de circuits à faible consommation apparaissent pour allonger la durée de vie des

batteries. Dans les circuits CMOS, la majorité de la consommation de puissance est dûe aux commutations. Dans les circuits synchrones, une bonne part de la consommation de puissance est liée à l'activité de l'horloge, indépendamment de l'activité logique. La logique dynamique peut ainsi offrir un potentiel pour réduire l'énergie dissipée car les portes dynamiques sont plus petites que les portes statiques. Farnsworth, Edwards et Sikand [6] expliquent l'intérêt d'utiliser des portes dynamiques pour réduire la dissipation de puissance dans les circuits asynchrones. Un exemple intéressant réalisé avec des cellules TSPC est le processeur Alpha de DEC où l'horloge est responsable de 40% de la dissipation de puissance. On voit donc l'importance de diminuer le nombre de noeuds commandés par l'horloge. Pour atteindre cet objectif, la méthode TSPC a été utilisée dans la conception de tous les registres, à partir des cellules proposées par Svensson and Yuan [4]. Les deux principaux avantages de cette technique sont qu'un seul noeud d'horloge est nécessaire, et que seulement quatre transistors de taille minimale par registre sont connectés au noeud d'horloge avec la configuration retenue. Ceci permet une diminution de près de 50% de la puissance d'horloge par rapport aux horloges à deux phases sans recouvrement. Un autre avantage est que le nombre total de transistors est de 11, comparé aux bascules sans recouvrement, ce qui permet un gain d'intégration en surface. Le fait d'utiliser un seul noeud d'horloge permet également d'utiliser un seul canal de routage, et donc de réduire encore la surface. Il existe deux inconvénients à la méthode TSPC. D'abord, il s'agit d'un registre dynamique, donc l'horloge a besoin d'une fréquence minimale d'opération. Les mesures ont montré que ce minimum était de l'ordre de 500 Hz à 25^0C. Un autre inconvénient est une course interne qui apparaît à l'entrée de l'inverseur sur le front

montant de l'horloge. Ce phénomène a été observé en simulation sans charge, mais il n'a jamais été vu dans un contexte d'utilisation réaliste où chaque porte a au moins une sortance de une porte.

Le premier chapitre de ce mémoire est consacré au développement de bibliothèques de cellules TSPC. Nous présentons tout d'abord les différentes cellules TSPC, leur avantages, puis différents travaux qui leur sont dédiés. Une fois que ces travaux ont été présentés et analysés, nous présentons notre propre méthode de conception de bibliothèque qui est valable pour toutes les cellules dynamiques. L'heuristique présentée est dédiée aux cellules TSPC. Enfin, nous mettons en valeur certains points faibles des cellules TSPC à sortie partagée.

Le second chapitre présente l'architecture d'un convolueur 3*3 TSPC et la solution que nous avons choisie. Une brève description de chaque module est présentée. Ensuite, sont présentés les cellules de base du convolueur 3*3 TSPC, puis le chemin de données qui réalise l'opération de convolution.

Le troisième chapitre explique davantage la méthode que nous avons adopté pour développer un circuit de haute performance avec la présentation de circuits de test efficaces à plus d'un GHz. Ce chapitre présente aussi la stratégie de distribution d'horloge et les méthodes de placement que nous avons utilisées.

CHAPITRE 1

METHODE DE CONCEPTION D'UNE BIBLIOTHEQUE TSPC

1.1 Bascules dynamiques synchronisées par des horloges monophasées

1.1.1 Introduction

La technologie CMOS conventionnelle utilise des formes de logiques statiques et dynamiques. Une technique de synchronisation très en vogue pour les circuits à logique dynamique est celle de la synchronisation pseudo double-phase (Suzuki, Odagawa et Abe, 1973; Weste et Eshraghian, 1985). Celle-ci requiert quatre signaux d'horloge. Les cellules dynamiques permettent d'atteindre des fréquences très élevées tout en étant synchronisées. Les cellules dynamiques les plus courantes sont les cellules C^2MOS, Domino et Nora. Toutes ces cellules ont en commun un système de précharge et des transistors commandés par une horloge qui libèrent les charges à la fréquence voulue. Évidemment, plus la fréquence est élevée, moins le réseau logique de ces portes est complexe. Les avantages de la synchronisation multiphasée sont la sécurité et l'efficacité mesurée en terme du nombre de transistors. Cependant, la distribution des horloges requiert une superficie de silicium relativement élevée. Des circuits récents basés sur l'horloge monophasée permettent d'atteindre des vitesses supérieures à 1 GHz avec une technologie comme la CMOS 0.8 μ m. La synchronisation à horloge monophasée s'avère aussi sécuritaire et efficace en terme du nombre de transistors que la technique à quatre phases (Yuan, Karlson et Sven-

son, 1987). Elle permet également des fréquences d'horloge plus élevées que les techniques multiphasées (Yuan et Svenson. 1989).

La première partie de ce chapitre est consacrée à la définition des cellules TSPC. Dans la deuxième partie, nous présentons et analysons les différentes méthodes d'optimisation qui ont été explorées. Notre méthode d'optimisation, ainsi que ses résultats, sont proposées dans la troisième partie.

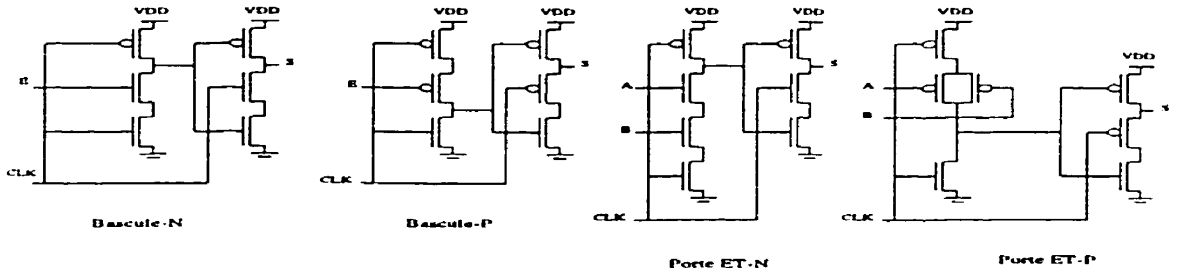
1.1.2 Configurations de bascules et de portes TSPC

Les cellules TSPC sont dynamiques et se caractérisent par l'utilisation d'une horloge à phase unique. Ces cellules font appel à l'utilisation de transistors de précharge et à des noeuds dynamiques, afin d'accroître la vitesse de commutation et de réduire la complexité des éléments de mémoire. Il est ainsi possible d'associer une bascule logique à chaque porte logique. Nous pouvons donc former un pipeline à granularité très fine dont chaque étage correspond à une porte logique. Il faut seulement alterner des étages de polarité N avec des étages de polarité P, comme pour les cellules Nora. Pour une cellule de type N, le niveau haut de l'horloge active un transistor d'évaluation, alors que la précharge se réalise sur son niveau bas. Pour une cellule de type P, le niveau bas de l'horloge active le transistor d'évaluation, alors que la précharge se réalise sur le niveau opposé.

La figure 1.1 présente différentes configurations de bascules et de portes TSPC, telles que définies par Yuan et Svensson [4]. Pour chacune des configurations (TSPC-1, TSPC-2,

C^2MOS double et sortie partagée), des bascules N et P sont illustrées ainsi que des portes ET de type N et P.

Configuration TSPC-1



Configuration TSPC-2

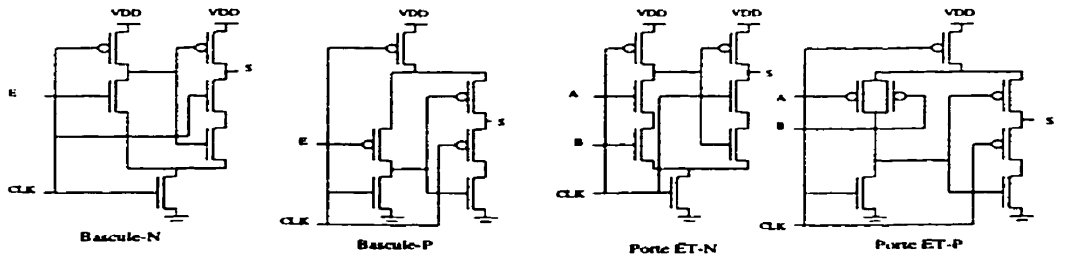
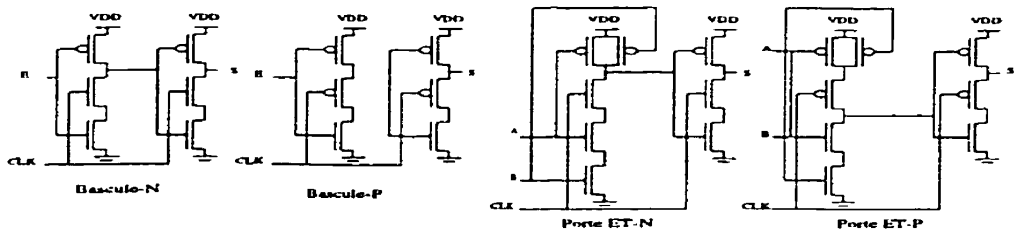


Figure 1.1: Différentes configurations des cellules TSPC

Configuration C^2MOS double



Configuration à sortie partagée

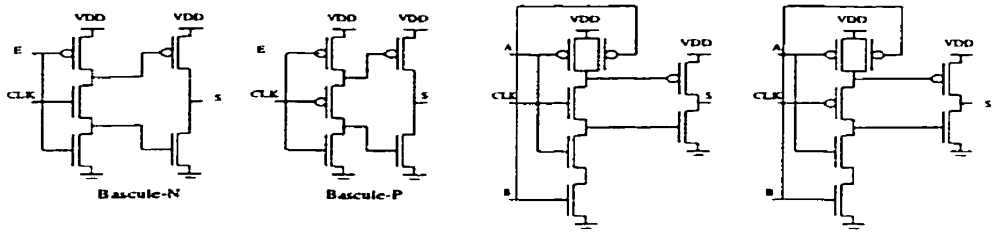


Figure 1.1: Différentes configurations des cellules TSPC

Bosi [8] et Chtchvyrkov [9] ont présenté une bonne description du fonctionnement des différentes classes de cellules TSPC. Chtchvyrkov a montré que la configuration TSPC à sortie partagée donnait les meilleurs résultats en terme de vitesse maximale. Le fonctionnement d'une bascule TSPC à sortie partagée de type N est assez simple. Lorsque l'horloge est à 0, si l'entrée est à 0, nous avons une précharge à 1 sur le drain du transistor d'horloge. Lorsque l'horloge passe à 1, elle agit comme une porte de transmission et propage le 1 à sa source, ce qui active le transistor nMOS de sortie. Le transistor Nmos de

sortie envoie alors un 0 sur le noeud de sortie. Le fonctionnement est similaire si on veut propager un 1. En fait, il faut analyser cette cellule comme deux inverseurs en série. Le premier inverseur produit une précharge à l'opposé logique de l'entrée. Le second inverseur inverse alors la précharge propagée par l'horloge. Pour des cellules efficaces et capables de supporter les charges typiquement rencontrées dans un circuit logique, nous pouvons nous attendre à ce que les transistors de l'inverseur de sortie soient plus grands que les transistors de l'inverseur d'entrée. Nous nous sommes donc concentrés sur ce type de cellule afin de réaliser une bibliothèque de cellules TSPC. Les études précédentes [8] [9] s'étaient surtout concentrées sur la réalisation de bascules et nous les avons prolongées pour couvrir la réalisation d'une bibliothèque TSPC portable d'une technologie à l'autre.

1.2 Méthode d'optimisation d'une bibliothèque TSPC.

1.2.1 Introduction

Plusieurs personnes ont essayé de développer des bibliothèques de cellules TSPC en exploitant la puissance des outils de conception et de simulation qui sont à notre disposition. En fait, lorsqu'une nouvelle technologie arrive il faut développer une nouvelle bibliothèque de cellules TSPC. Chtchvyrkov a tout d'abord testé les diverses familles de cellules TSPC. Ses nombreuses simulations ont permis de confirmer que la technique TSPC à sortie partagée donne les meilleurs résultats. Il travaillait à l'époque en technologie CMOS 3 µm. Il a également mis au point un algorithme d'optimisation de cellules dynamiques que nous allons exposer puis analyser. Bosi a également développé une bibliothèque de

cellules TSPC en schématique pour la technologie CMOS 1.2 µ m en vue d'implanter un convolueur 3*3 TSPC. Il a, pour ce faire, exploité les algorithmes d'optimisation disponibles dans HSPICE. Antaki et Patenaude [10] se sont également intéressés à l'importance du transistor d'horloge dans les cellules TSPC à sortie partagée. Ils ont travaillé sur des chaînes de bascules en proposant un algorithme d'optimisation séduisant par sa simplicité. Les sections 2.1, 2.2 et 2.3 sont relatives à la description et à l'évaluation des différentes méthodes de développement de bibliothèques de cellules TSPC, dont découle la section 3 consacrée à une nouvelle méthode d'optimisation des cellules TSPC.

1.2.2 Une méthode d'optimisation systématique

Chtchvyrkov a étudié les cellules TSPC1, TSPC2, C^2MOS et SP (Sortie partagée). Il a réalisé une étude de chaque cellule. Deux versions de chaque bascule furent considérées: l'une étant en configuration positive (sensible à une valeur positive de l'horloge, type N), l'autre négative (type P). Il a ainsi pu évaluer les avantages et inconvénients de chaque technique selon les vitesses d'évaluation, de précharge, de temps de montée et de descente. Pour évaluer ces techniques de conception, il a d'abord conçu des bascules selon chaque technique en les optimisant à l'aide d'un algorithme d'optimisation automatique. Il a établi un certain nombre de critères qui sont partiellement repris dans la section 3. Son programme d'optimisation a été réalisé en langage ANSI C. Son logiciel fournit une netlist à HSPICE qui lui renvoie des paramètres de temps de montée, de descente, de tension maximale... Ces données sont analysées par son logiciel qui renvoie une nouvelle Netlist à Hspice, qui renvoie à son tour des données jusqu'à l'obtention d'une convergence.

Son algorithme d'optimisation se résume ainsi:

1) à chaque étape, le logiciel choisit une des configurations possibles et exécute une simulation HSPICE pour en établir les performances;

2) les résultats sont comparés aux résultats antérieurs;

3) la meilleure configuration est retenue et le processus continue, jusqu'à la satisfaction d'un critère d'arrêt.

Pour réduire le nombre de simulations, un algorithme introduisant des perfectionnements graduels a été proposé afin d'accélérer la recherche. La recherche s'effectue sur une dimension à la fois, c'est-à-dire que toutes les largeurs de transistors sont fixées, sauf une. Une fois que la meilleure solution est déterminée pour une dimension, elle sert de point de départ dans la recherche de la dimension suivante. La complexité de ce genre de recherche est linéaire relativement au nombre de dimensions à explorer.

En fait, dans cet algorithme, la fréquence d'opération à laquelle les cellules sont simulées est constante. Il faut donc déterminer la fréquence de travail et être certain que la bibliothèque pourra entièrement être conçue à cette fréquence. Le but de cette opération est de réaliser une bibliothèque de cellules balancées. Chaque cellule doit avoir des temps de montée et de descente proches et des délais semblables. Il faut donc mettre des critères sur chaque paramètre afin que le signal de sortie de chaque porte se conforme à un gabarit imposé.

Donc, dans cet algorithme, une largeur de transistor est proposée et sa valeur évolue tant que les performances s'améliorent. La méthode optimise tous les transistors de la cellule en suivant un gradient de performance. La rapidité d'exécution pourrait être accrue en utilisant une recherche dichotomique pour le meilleur rendement de la taille d'un transistor. Cet algorithme consomme un temps de calcul considérable, car il prend en compte un nombre important de paramètres et peut demander plusieurs passes avant de converger. Il semble donc nécessaire de diminuer l'effort de recherche, cependant on peut remarquer que cet algorithme donne de bons résultats malgré un nombre d'itérations important. Notons que dans l'algorithme implanté par Chtchvyrkov, la charge peut varier et les inverseurs d'entrée produisant un signal de format typique sont fixes. Il aurait été bon de laisser varier les inverseurs d'entrée afin d'avoir des signaux correspondant à la taille variable de la nouvelle bibliothèque. En effet, si les inverseurs d'entrée ont une taille importante, les transistors d'entrée de la cellule à optimiser ont un grand degré de liberté et ils peuvent ainsi atteindre une taille importante. Dans la réalité, lorsque les étages N et P sont alternés, nous pouvons rencontrer des erreurs. L'étage de sortie des bascules est configuré pour commander deux inverseurs en espérant n'en rencontrer qu'un ou deux. Si les transistors d'entrée sont trop importants, l'étage de sortie sera insuffisant pour contrôler la bascule suivante. Il faut donc que les transistors qui commandent l'entrée soient ajustées avec ceux de la cellule en voie d'optimisation. Il faut également tenir compte des interconnexions

1.2.3 Méthode d'optimisation des bascules selon trois dimensions.

Patenaude et Antaki [10] ont étudié l'importance du transistor d'horloge sur des registres à décalage TSPC réalisés avec des cellules à sortie partagée. Ce transistor est très important dans ces cellules, car il joue le rôle d'une demi-porte de transmission. Dans leur recherche, ils évaluent l'importance de la taille de ce transistor ainsi que l'effet de la sortance et des longues interconnexions en sortie. Pour ce faire, ils ont développé un outil d'optimisation automatique. Cet outil réalisé en langage C invoque également Hspice, de façon itérative, en vue de converger vers une solution qui n'est pas forcement la meilleure, mais qui correspond quand même à un résultat valable.

Dans cette chaîne de bascules, seules trois dimensions peuvent évoluer en fonction de la fréquence. On ne travaille que sur trois variables: les transistors Nmos, Pmos et le transistor d'horloge à partir d'une Netlist extraite de Cadence. Il est ainsi possible de connaître la fréquence maximale que peut atteindre une configuration de tailles de transistors et de faire varier cette configuration pour accroître les performances. En modifiant l'algorithme, il est également possible d'obtenir la meilleure configuration de transistors à la fréquence de travail souhaitée. L'algorithme proposé est plus complexe que le précédent, mais il tient compte de moins de variables.

Les résultats ont démontré l'importance du transistor d'horloge et dévoilé également les limites des simulateurs à hautes fréquences. Ainsi, lorsque seul le transistor d'horloge évolue linéairement, on s'attend à des résultats qui progressent linéairement, hors ce n'est

pas le cas. Des zones d'ombre, où se produisent des chutes de performance inexpliquées et des pics de performance pour une taille de transistor montrent les limites des modèles des transistors à haute fréquence. Ceci s'explique par des changements de modèles de transistors lorsque les tailles changent. Les modèles de transistors d'une technologie varient selon la longueur de la grille et la largeur du transistor ainsi que le procédé de fabrication. Donc, lorsque la taille des transistors évolue, il peut apparaître des pics dans les résultats qui sont encore plus importants à haute fréquence. Évidemment, le fait de ne faire varier que les transistors Nmos et Pmos diminue le degré d'optimisation des résultats, car nous n'avons que 2 paramètres, mais cette méthode peut donner rapidement des résultats. Nous jugeons que la qualité des solutions obtenue est insuffisante lorsqu'il s'agit de développer une bibliothèque de cellules.

1.2.4 Méthode d'optimisation à partir d'un outil d'optimisation d'un simulateur

Bosi [8] a également développé une bibliothèque de cellules TSPC en technologie CMOS 1.2 µm. Ces cellules ont été réalisées au niveau schématique. Pour cette optimisation, seul l'outil d'optimisation disponible dans HSPICE est utilisé. Cet outil permet aux transistors de varier selon une table prédéfinie, dans le but de rencontrer un certain nombre de critères. En fait, Hspice réalise des mesures et ne conserve que les configurations qui atteignent un certain nombre d'objectifs en terme de performance. Cette méthode rejoint celle de Chtchvyrkov et permet d'obtenir rapidement une bibliothèque de cellules balancées . Il est par exemple possible de demander que le temps de montée soit à 90% de VDD et inférieur à 0.2 ns. En fait, il faut dessiner un gabarit et Hspice fait varier les tailles de tran-

sistors tant que les objectifs n'ont pas été atteints. Évidemment, Hspice ne tient pas compte de résultats aberrants sur les dimensions des transistors et peut offrir des solutions peu élégantes. Il faut vraiment se méfier de telles optimisations. De plus, il faut travailler à fréquence constante et il est difficile de connaître la meilleure configuration dans une technologie donnée. Cette méthode est tout de même intéressante pour obtenir une première idée des tailles de transistors pour une bibliothèque.

1.3 Méthode d'optimisation de la taille des transistors

Cette section présente une heuristique pour optimiser des bascules TSPC à sortie partagée. Les études précédentes ont montré que les circuits TSPC peuvent opérer à de très hautes fréquences. Notre but est d'implémenter de larges blocs numériques pouvant opérer à 1 GHz ou davantage (dans une technologie CMOS 0.35 µm). Notre méthode a débuté avec le développement d'un premier ensemble de cellules basées sur un design ad-hoc.

Une méthode semi-automatique fut alors développée pour réduire le nombre de simulations. Basé sur l'expérience acquise dans l'optimisation de cellules TSPC, nous proposons un algorithme pour optimiser automatiquement les cellules TSPC et une méthode pour réaliser des bibliothèques de cellules logiques normalisées. Cette section décrit les différentes étapes à suivre pour élaborer une bibliothèque TSPC. La méthode et la bibliothèque de cellules sont validées par la caractérisation de petits blocs numériques.

Les concepteurs de systèmes électroniques luttent pour augmenter les performances qu'ils obtiennent. Chaque fois qu'une nouvelle technologie apparaît, les concepteurs rencontrent la nécessité de développer de nouvelles bibliothèques. Pour éviter cette tâche, une méthode qui optimise les performances des cellules TSPC est proposée. Le but de la méthode est de réduire, autant que possible, le nombre de simulations requises pour atteindre un design quasi optimal. L'optimisation dépend des critères adoptés. Nous avons pu observer dans des travaux précédents que les cellules pouvaient être balancées pour une fréquence cible spécifique, et, dans ce cas, le but général est de minimiser la surface pour atteindre cette fréquence. La section 1.3.1 décrit des critères d'optimisation. La section 1.3.2 propose une méthode qui utilise ces critères, suivie d'un ensemble d'étapes. La section 1.3.3 présente quelques résultats comprenant une bibliothèque de base avec ses tailles de transistors et son utilisation pour réaliser de petits blocs logiques.

1.3.1 Règles d'optimisation

1.3.1.1 Banc d'essai

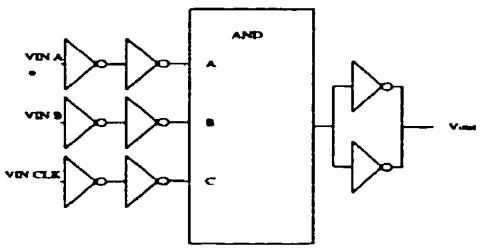


Figure 1.2: Banc d'essai typique (AND à 2 entrées)

Des bancs d'essai similaires à ceux montrés à la figure 1.2 furent utilisés lors de l'optimisation de la bibliothèque de cellules de base. Tous les signaux d'entrée d'un circuit testé passent à travers des paires d'inverseurs pour produire des transitions réalistes. Les inverseurs utilisés sont ceux de notre bibliothèque standard. La vitesse maximale d'un bloc TSPC tend à être contrôlée par la vitesse de la cellule la plus lente. Il est donc nécessaire d'optimiser au maximum toutes les cellules lentes, en l'occurrence celles qui ont un transistor Pmos pour l'horloge. Une charge réaliste doit être introduite pour obtenir des résultats valides. Nous avons pu remarquer que les sortances de nos blocs logiques, numériques et arithmétiques étaient d'un ou de deux avec une grande fréquence. Donc deux inverseurs ont été utilisés pour représenter cette charge. Notons que pendant l'optimisation, les tailles des transistors de tous les inverseurs (entrée et sortie) varient selon celles des transistors dans les cellules (tous liés à un même paramètre de variation). Il faut également ajouter une capacité d'interconnexion pour la sortance.

1.3.1.2 Critères d'optimisation

Une bibliothèque convenablement optimisée doit être en mesure de propager tous les signaux d'entrée. Cependant, un petit nombre de vecteurs d'entrée sont suffisants pendant la phase d'optimisation pour atteindre cet objectif. Par exemple, les vecteurs 000100010... et 11101110... testent la faculté d'un chemin sensibilisé à répondre à des valeurs 0 et 1 isolées, tandis que le vecteur 010101010 force la cellule à réagir rapidement à des signaux qui varient constamment. Les cellules à deux entrées sont testées avec les quatre combinaisons logiques possibles. À chaque fois, nous conservons le résultat du pire vecteur.

L'optimisation permet de trouver la fréquence maximale atteignable par une cellule et la plus petite taille de transistor nécessaire pour une cellule à une fréquence et à une sortance spécifique. Un ensemble de critères sont utilisés pour établir la faculté du circuit à propager le signal à une vitesse donnée. Ces critères sont:

* Le niveau haut logique de la tension de sortie doit atteindre 90% de VDD et le niveau bas logique doit être inférieur à 10% de VDD;

*Si l'on considère le signal d'entrée 010101 dont la période est de deux cycles d'horloge, quand ce signal est propagé à travers une cellule, la durée des signaux valides 0 et 1 en sortie devront être de plus de 5% de la durée d'un cycles d'horloge.

*les transitions de montée et de descente des sorties doivent être de moins de 40% du cycle d'horloge.

1.3.1.3 Modèle utilisé pour la simulation.

Afin d'utiliser un bon modèle pour l'optimisation, une première version de chaque cellule a été conçue au niveau schéma des masques. Pour le schéma des masques, tous les transistors ont la même taille (W=7 µm dans notre exemple). Les éléments parasites typiques sont extraits du schéma des masques afin de générer une *netlist* plus complète.

1.3.2 Étapes d'optimisation pour élaborer une bibliothèque TSPC.

Cette section commence par la description des différentes étapes d'optimisation qui peuvent être combinées lors du développement d'une bibliothèque TSPC. Un algorithme utilisant ces étapes sera proposé ensuite. Deux technologies ont été supportées dans nos exemples [11,12]

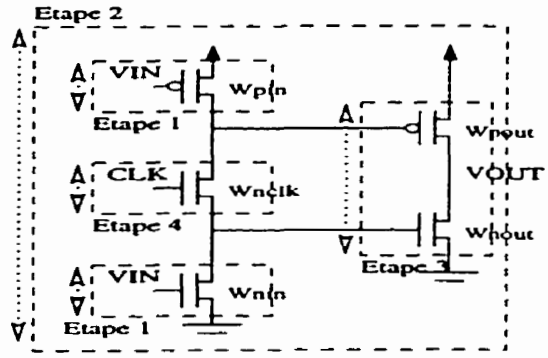


Figure 1.3: Différentes étapes d'optimisation pour une bascule de type N

1.3.2.1 Modèle utilisé pour la simulation.

Les quatre étapes d'optimisation sont liées à la structure des cellules TSPC à sorties partagées, et elles sont donc applicables à tous les types de portes, comme montré aux figures 1.3 et 1.4, pour une bascule et une porte ET respectivement.

Étape 1:

Cette étape est utilisée pour trouver le meilleur rapport x entre les largeurs des transistors Nmos et Pmos, c'est à dire: W_p/W_n où:

$$W_p = W_{pin} = W_{pclk} = 0.5 * W_{pout} \quad (1)$$

$$W_n = W_{nin} = W_{nclk} = 0.5 * W_{nout} \quad (2)$$

Tout d'abord, W_n est fixé et nous déterminons W_p en considérant l'espace $[W_n, \ldots, 5W_n]$. Différents incréments peuvent être utilisés à l'intérieur de l'espace à différentes phases du processus d'optimisation. Pour simplifier, nous avons utilisé un incrément de 5 µm à différentes phases de l'algorithme. Donc $W_p \in \{W_n, W_{n+5}, W_{n+10}\ldots\}$. Évidemment, d'autres incréments peuvent être utilisés pour obtenir une meilleure résolution. Toutefois, de petits incréments augmenteront les ressources matérielles de calcul. Il est également possible de débuter avec des incréments plus importants pour effectuer une première évaluation, puis de diminuer la valeur des incréments lorsque l'algorithme progresse. Quand l'algorithme est appliqué à une technologie CMOS 0.35 µm, la solution converge à $W_p=4*W_n$ pour une porte ET de type N et à $W_p=2*W_n$ pour une porte ET de type P.

Étape 2:

Dans cette étape, nous déterminons la meilleure hauteur des cellules pour une technologie. Le rapport $W_p = x* W_n$ a été obtenu de l'étape précédente. Ainsi la valeur des transistors p est liée à celle des transistors n. La recherche d'une hauteur idéale amène à la remarquable conclusion que la performance continue de croître lorsque la hauteur croît. Le circuit tend vers une performance asymptotique maximale. Dans une technologie 1.5 µm, cette asymptote était plus ou moins atteinte avec une taille de 50 µm et 95% de la

performance maximale était obtenu avec W_n=12 µm. Par contre, dans une technologie 0.35 µm, cette asymptote était plus ou moins atteinte avec une taille de 30 µm et 95% de la performance maximale était déjà atteinte avec une largeur W_n=5 µm. Dans cette étape, des équations similaires à (1) et (2) sont utilisées mais W_p est remplacé par $x*W_n$.

Étape 3:

Dans cette étape, nous optimisons la taille des transistors de sortie. Leur taille peut changer jusqu'à l'obtention de la meilleure performance pour une sortance donnée. Après avoir obtenu le meilleur rapport y entre W_{pout} et W_{nout}, W_{pout} est remplacé par $y*W_{nout}$. La taille des transistors varie jusqu'à l'obtention des meilleurs résultats. À cette étape seuls deux transistors varient. Dans notre exemple (technologie CMOS 0.35 µm), nous avons utilisé un incrément de 5 µm et trouvé un rapport de 2 pour chaque cellule de la bibliothèque (W_{pout}=20 µm et W_{nout}=10 µm). W_{pout} et W_{nout} dépendent de la sortance. Les transistors de sortie jouent un rôle très important car ils restaurent le signal de sortie.

Étape 4:

Cette étape optimise la taille du transistor d'horloge. Sa largeur varie jusqu'à l'obtention des meilleurs résultats. Dépendant de la polarité de la grille, n ou p, W_{nclk} ou W_{pclk} changent de 1 µm à 5 fois la valeur de W_{nin} selon la valeur de l'incrément d'optimisation.

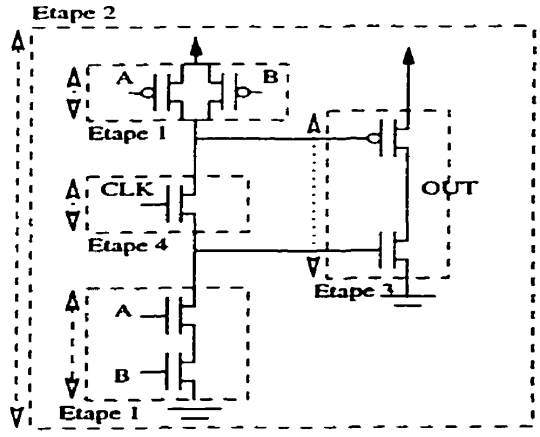


Figure 1.4: Différentes étapes d'optimisation pour AND TSPC de type NMOS

1.3.2.2 Algorithme d'optimisation automatique d'une bibliothèque de cellules TSPC

L'algorithme débute avec une phase d'initialisation qui est suivie par la boucle principale. La phase d'initialisation commence avec une chaîne de bascules TSPC dont la polarité des transistors d'horloge est alternée afin d'obtenir $W_p=x*W_n$. L'étape 2 est alors utilisée pour caractériser la performance selon le rapport taille/asymptote. L'utilisateur peut choisir une fréquence cible ou sélectionner un point d'opération sur l'asymptote, qui sera par exemple la largeur des transistors qui correspond à la hauteur de sa bibliothèque. Il faut noter que dans la recherche du paramètre de taille optimale, la charge et les transistors de commande doivent varier en même temps que les bascules. Dans une technologie CMOS 0.35 µm, nous avons obtenu W_{nin}=5 µm et W_{pin}=10 µm, comme l'indiquent les figures 1.3 et 1.5 pour des bascules de type N et P.

La principale boucle d'optimisation optimise chaque cellule de la bibliothèque. Les étapes 1,3 et 4 sont utilisées dans cette tâche. Ces étapes peuvent être répétées dans un ordre différent. Elles peuvent également être renouvelées plus d'une fois afin d'améliorer le résultat. Pour obtenir la bibliothèque présentée à la figure 4, la séquence d'étape 1,3,4,1,4 et 3 a été utilisée sur chaque cellule. Il faut noter que l'étape 1 change après sa première utilisation. Seuls W_{pin} et W_{nin} changent, tandis que W_{clk} et W_{out} conservent leur valeur obtenue lors de la précédente optimisation.

Les inverseurs prennent les valeurs $W_{pinv} = W_{pin}$ et $W_{ninv}=W_{nin}$, obtenues lors de l'initialisation, pendant la phase d'optimisation. Quand la bibliothèque est finie, les inverseurs de la nouvelle bibliothèque adoptent les tailles des transistors W_{pout} and W_{nout} obtenues pour la bascule TSPC Nmos. Ainsi, pour obtenir un NON-ET, nous avons juste à ajouter un inverseur au ET. Quand des tailles stables ont été obtenues ou après un nombre spécifié d'itérations, nous pouvons optimiser davantage les cellules en faisant varier seulement un transistor à la fois, tandis que les autres restent statiques comme dans l'algorithme proposé par Chtchvyrkov.

Les quatre étapes d'optimisation ont produit les six cellules présentées à la figure 1.5.

1.3.3 Résultats de la bibliothèque de cellules TSPC

1.3.3.1 Tailles des différentes cellules:

La figure 1.5 présente les résultats obtenus par notre algorithme dans une technologie CMOS 0.35 µm.

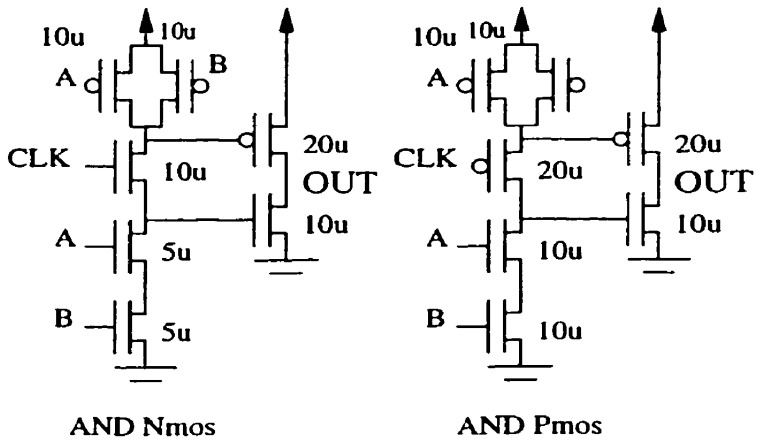


Figure 1.5: Résultats de la bibliothèque de cellules TSPC

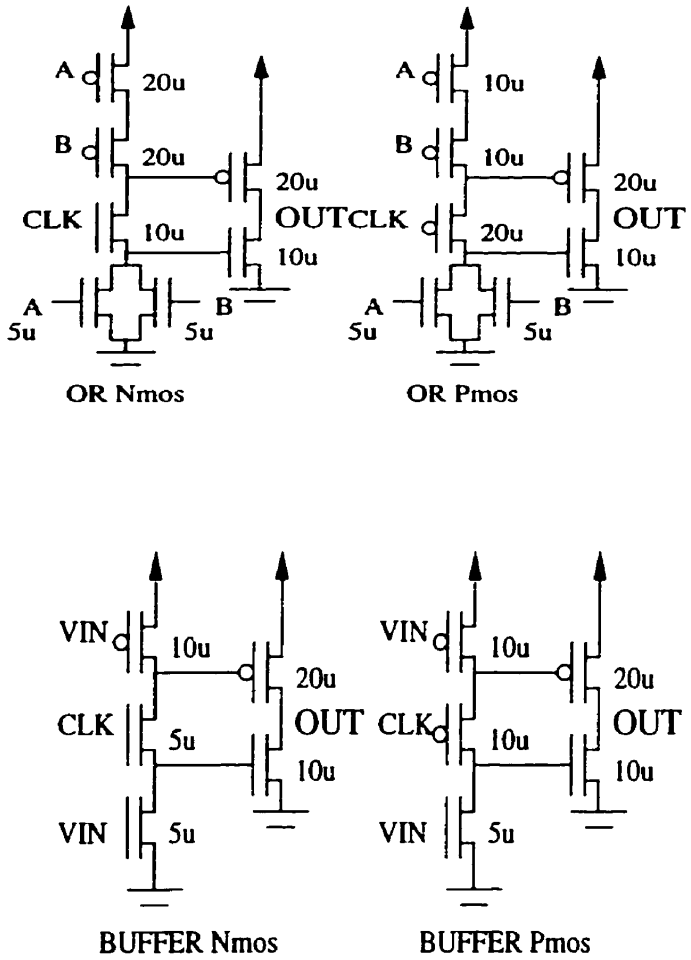


Figure 1.5: (suite) Résultats de la bibliothèque de cellules TSPC

1.3.3.2 Performance obtenue dans la technologie 0.35 µm

Cette section valide les performances de la bibliothèque. Nous avons réalisé des circuits typiques utilisant la plupart des cellules comme l'additionneur complet et nous démontrons les performances des circuits TSPC avec un registre à décalage qui fonctionne à une fréquence proche de la vitesse limite de la technologie.

=> Chaîne de six bascules TSPC avec une sortance de deux inverseurs.

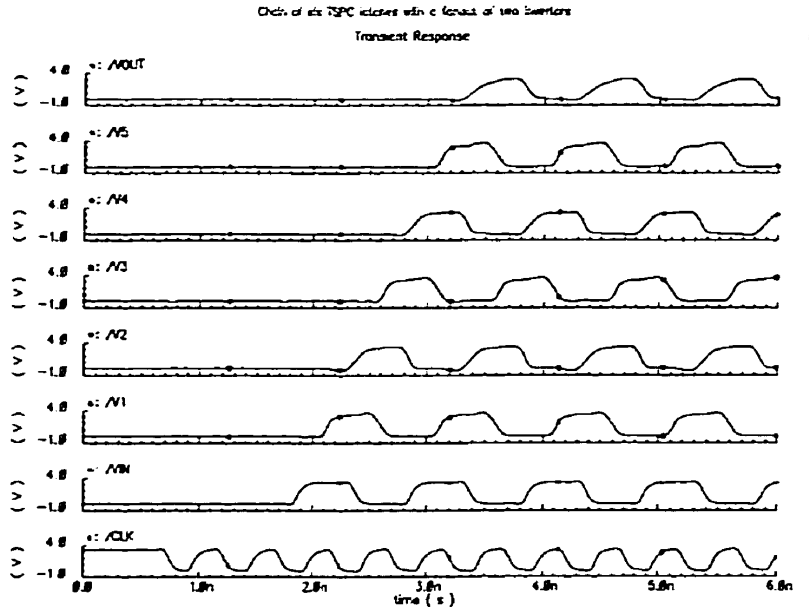


Figure 1.6: Simulation d'une chaîne de 6 bascules à 2 GHz

La figure 1.6 présente les résultats d'une simulation d'une chaîne de six bascules opérant à 2 GHz. Cette chaîne est composée de 6 bascules à polarités Nmos et Pmos alternées. La sortance est toujours de deux inverseurs en parallèle comme dans le banc d'essai.

=> Additionneur Complet

Un additionneur complet dont la structure est illustrée à la figure 1.7 a été caractérisé par simulation. Le chemin critique de cette cellule est situé au niveau de la porte NAND Nmos du XOR. Ce type de circuit doit éviter la logique négative autant que possible, car

les inverseurs ajoutent du délai. La fréquence d'horloge maximale du NAND Nmos est 1.6 GHz. La fréquence d'horloge de l'additionneur complet, tel que montré à la figure 1.8, est également de 1.6 GHz lorsqu'il est chargé par deux inverseurs.

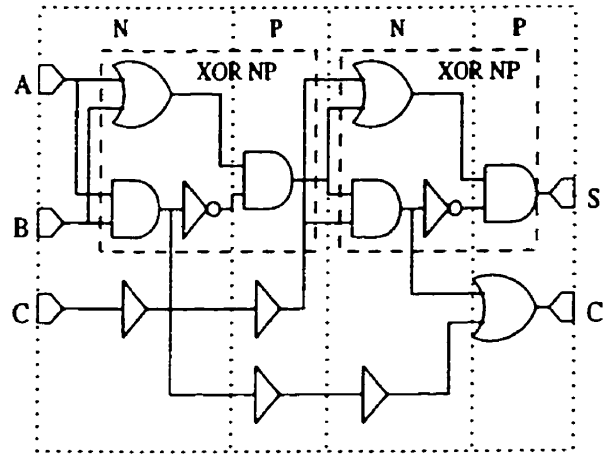


Figure 1.7: Schéma en logique statique d'un additionneur complet

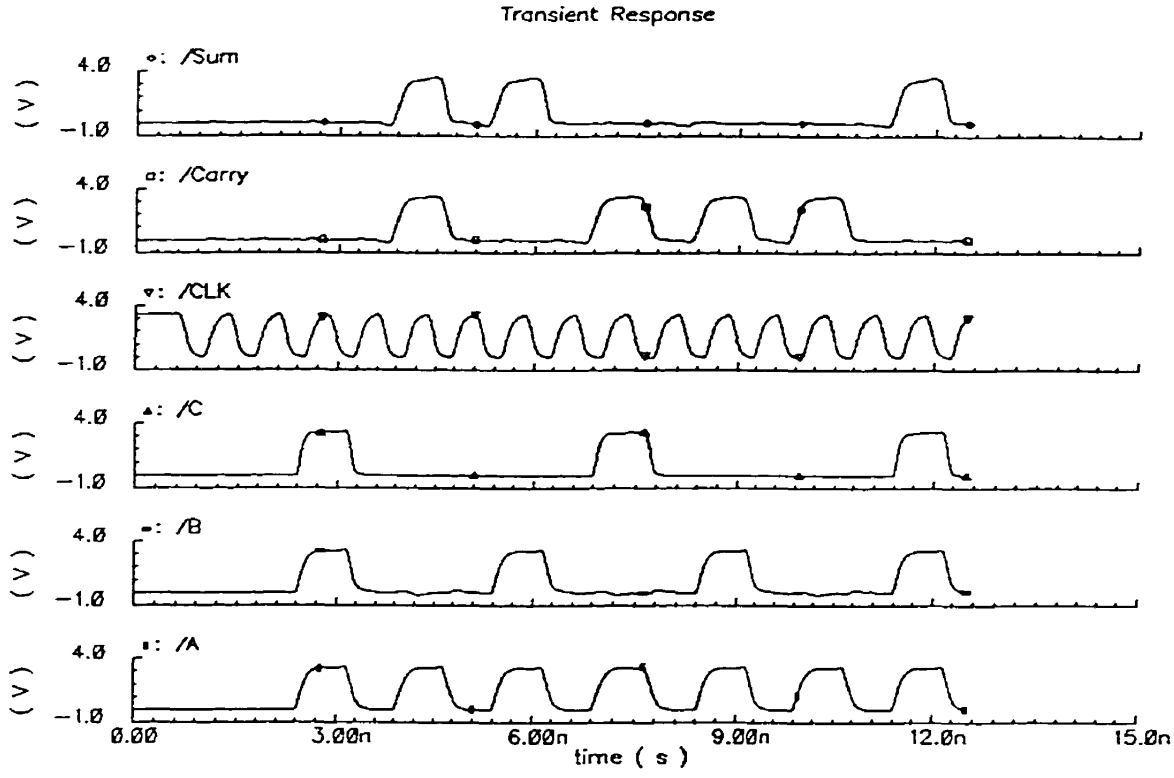


Figure 1.8: Simulation d'un additionneur complet à 1.6 GHz

Il est pertinent de mentioner que les simulations réalisées avec Hspice donnent une fréquence maximale de 1.6 GHz, tandis que celles réalisées avec Spectre de Cadence donnent une fréquence de 1.9 GHz et ce pour un modèle de transistor très proche, le modèle de transistor est le BSIM3v3, développé par l'université de Berkeley.

1.3.3.3 Étude d'une cellule xor TSPC.

Nous avons vu que la technique TSPC repose sur un pipeline d'une granularité très fine. Il est également possible de concevoir des blocs plus importants au détriment de la vites-

se, mais en diminuant la latence. Un exemple concret est la porte XOR TSPC de polarité N ou P. Nous avons vu à travers le sommateur qu'il est possible de réaliser des XOR à l'aide de trois portes élémentaires. La fréquence atteinte était alors de 1.6 GHz pour cette cellule. Il est également possible de réaliser le XOR avec une seule cellule et des performances moindre. La fréquence atteinte pour le XOR N est alors de 1 GHz.

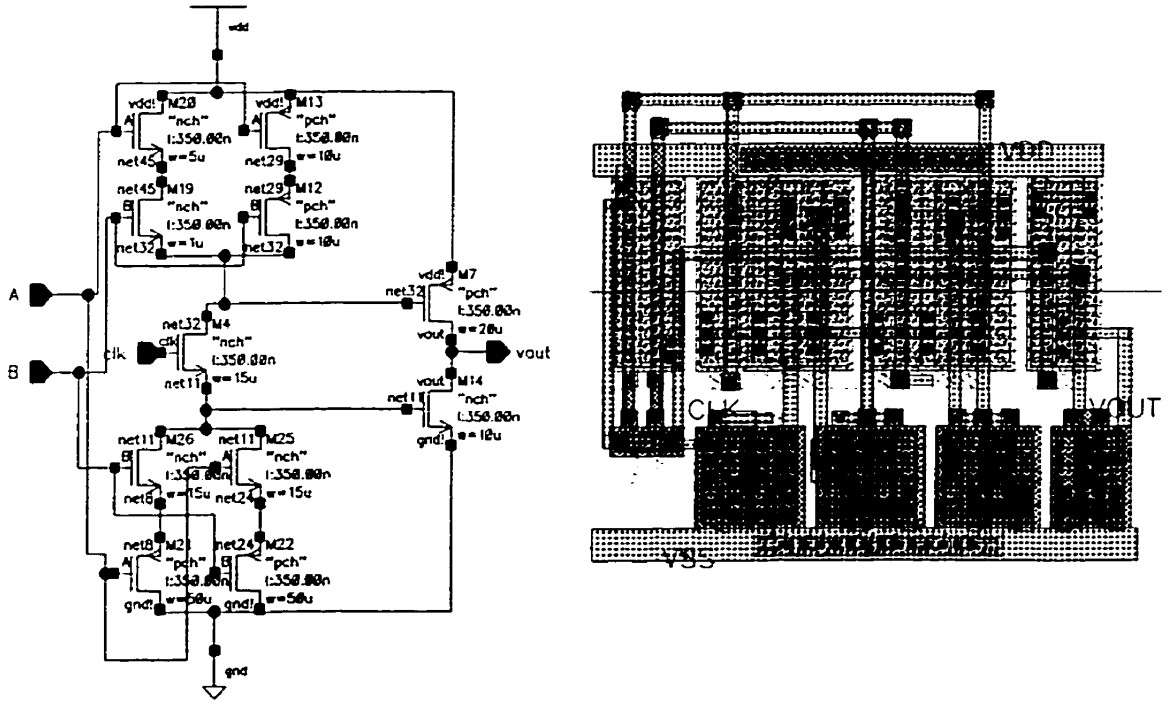


Figure 1.9: Schématique et schéma des masques du Xor TSPC

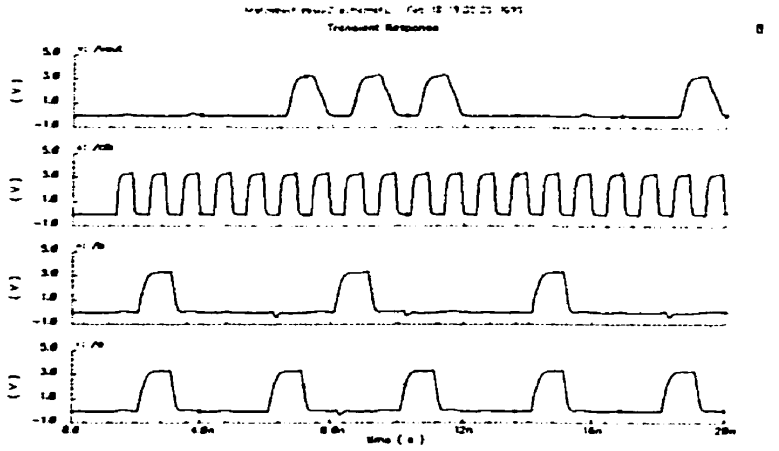


Figure 1.10: Simulation du Xor TSPC à 1 GHz

Cette optimisation a demandé un effort plus important que pour les autres cellules. Nous avons toutefois utilisé la même méthode. Les résultats sont présentés dans les figures 1.9 et 1.10. Un résultat surprenant concerne les transistors Nmos au dessus de l'horloge. En fait, seul le transistor Nmos au-dessus de l'horloge est vraiment critique. Sa taille doit être proche de la taille minimale. Quand sa taille augmente, la performance de la cellule chute. Si A=1 et B=0 dans une séquence, puis que A=0 et B=1 dans la séquence suivante, lorsque l'horloge fait passer un 0, le transistor Nmos M19 conduit et se décharge. Le transistor de sortie Pmos n'a plus alors un vrai 0 à sa grille et conduit moins vite pouvant entraîner une faute logique à haute fréquence. Pour palier à ce problème, le transistor Nmos qui est plus rapide que les transistors Pmos doit être "ralenti". Pour ce faire, il suffit de diminuer sa largeur. En effet, les meilleurs résultats ont été obtenu, lorsque le transistor Nmos 19 avait une taille minimale. La technique TSPC offre également de bonnes performances pour des cellules un peu plus importantes. L'avantage d'utiliser des cellules plus complexes est

également de diminuer la charge de l'horloge qui est considérable dans un pipeline aussi profond que celui que nous utilisons.

1.3.3.4 Problèmes des cellules TSPC à sortie partagée

Lors de la conception de blocs logiques assez importants, nous avons pu remarquer que ces derniers dissipaient beaucoup de puissance et nécessitaient de grands fils d'alimentation. Une étude approfondie des cellules a permis de relever deux problèmes. Le premier inconvénient des cellules TSPC à sortie partagée réside dans leur transistor d'horloge qui agit comme une demi-porte de transmission. Les tensions sur les drains et sources de ce transistor sont donc sujettes à des pertes de tensions de seuil. Nous n'aurons donc pas toujours un vrai 0 ou 1 à l'issue de la transmission de notre précharge. Les inverseurs de sortie recevant des 0 ou des 1 altérés vont donc commuter moins vite et ils propagent des signaux imparfaits. De plus, comme les commutations des transistors de sortie sont moins rapides, ils dissipent davantage de puissance par le mécanisme de courant de court-circuit.

Le XOR TSPC a permis de mettre davantage en évidence le problème de partage de charges associé aux courses internes. Ce problème se répercute sur la plupart des cellules TSPC à sortie partagée, sauf sur les bascules. Nous retrouvons ce problème à chaque fois que deux transistors PMOS ou NMOS sont en série. Dans une porte OU, par exemple, nous avons les deux transistors PMOS d'entrée en série. Pour l'analyse qui suit, nous supposons que dans un premier temps les entrées sont à 0. Lorsqu'une entrée passe à 1 et que l'autre ne change pas, nous pouvons avoir une faute logique. En effet, lors de l'évaluation,

la valeur découlant de la précharge essaie d'activer le transistor Pmos de sortie alors que le transistor PMOS d'entrée qui est toujours à 0 libère ses charges. Le problème est évidemment amplifié lorsque nous utilisons une porte OR TSPC de type PMOS, car la demi-porte de transmission du transistor de l'horloge amène une perte de seuil. Nous n'avons donc plus tout à fait un 0 logique. Dans la porte XOR que nous avons présenté dans la section 3.3.3, le problème est doublement accru, car nous avons à chaque extrémité du transistor d'horloge une paire de transistors en parallèle avec une autre paire de transistors, le problème de charge partagée est encore plus important. Dans la configuration présentée, nous avons éliminé ce problème en nous assurant que les transistors aux extrémités du transistor d'horloge aient des entrées différentes. Le problème majeur de la porte XOR TSPC et de toutes les grosses cellules TSPC est que la précharge doit commander une grosse capacité; deux transistors aux extrémités du transistor d'horloge ont été ajoutés pour ralentir la propagation de la précharge. Ce dernier problème est vraiment fonction de la fréquence d'utilisation. Il faut bien remarquer que lorsque le transistor d'horloge conduit, il propage le VDD ou le VSS sélectionnés par les entrées. Si la fréquence est peu élevée, la charge emmagasinée par le OR TSPC sur le transistor PMOS d'entrée à 0 a le temps de passer à 0 et nous aurons donc seulement un problème de perte de seuil au pire des cas. Lorsque le transistor d'horloge est désactivé, nous avons une vraie précharge qui continue d'activer le transistor de sortie jusqu'à la prochaine évaluation ou jusqu'au changement des signaux d'entrée. Nous avons pu remarquer qu'à des fréquences proches de 1.5 GHz, nos blocs logiques donnaient des réponses cohérentes.

Notre bibliothèque n'est pas sensible au phénomène de courses intenses dans les fréquen-

ces d'utilisation de l'ordre de 800 MHz à 1 GHz. À des fréquences supérieures nous pouvons avoir des courses intenses et donc des fautes logiques.

Ces deux problèmes nous montrent la difficulté de développer une bibliothèque de cellules TSPC à sortie partagée ainsi que ceux qui surviennent quand on veut concevoir des cellules complexes.

1.4 Conclusion

Dans ce chapitre, nous avons étudié en détail la conception des cellules TSPC, ainsi que diverses méthodes d'optimisation. La méthode que nous proposons pourrait être davantage optimisée en la programmant dans un outil en langage C. Pour avoir une optimisation optimale, il faudrait coupler notre outil avec un outil de génération de cellules automatique qui dessinerait les cellules au niveau schéma des masques à partir des paramètres que lui propose notre outil. En effet, le fait d'avoir un fichier extrait tenant compte des capacités réelles des interconexions serait très utile pour obtenir rapidement des résultats précis. La simplicité de notre algorithme permet de développer très rapidement une bibliothèque de cellules sans la nécessité d'un outil automatique qui est difficile à mettre en place à cause du nombre de paramètres à contrôler. Grâce à la méthode proposée, une bibliothèque de cellules de base TSPC efficaces peut-être développée en un mois, quelque soit la technologie employée.

Après avoir développé une bibliothèque de cellules dans la technologie CMOS 0.35 µm,

nous avions besoin d'un circuit assez important pour valider notre bibliothèque à une fréquence élevée. Nous avons ainsi développé un convolueur 3*3 TSPC que nous présentons au chapitre 2.

CHAPITRE 2

Architecture du convolueur 3*3 TSPC

2.1 Introduction

Après avoir présenté les performances et la méthode de conception de notre bibliothèque de cellules TSPC, nous souhaitons évaluer ses performances dans un circuit de grande complexité pour valider nos cellules et connaître davantage le potentiel des cellules dynamiques. Les circuits, construits à partir de cellules TSPC, entraînent une architecture fortement pipelinée qui induit une latence importante mais qui procure une très grande vitesse d'exécution. Il est donc nécessaire d'utiliser une application nécessitant un flot de données continues suffisamment long pour justifier une grande latence. Les applications liées au traitement de l'image nécessitent un effort prolongé et spécifique des microprocesseurs variant selon la taille de l'image à traiter. Il est donc intéressant de se doter de co-processeur spécialisés dans ce genre d'application.

Bosi [8] a proposé une première architecture d'un convolueur 3*3 TSPC que nous avons analysé, puis optimisé, afin de pouvoir le concevoir au niveau schéma des masques. Nous nous proposons donc de concevoir un circuit intégré avec une architecture complètement dédiée pouvant fonctionner à de très hautes fréquences.

La section 2.1 définit la convolution 2D, tandis que la section 2.2 présente la fonction au niveau système. La section 2.3 est consacrée à la conception des cellules de base du con-

volueur 3*3 TSPC et la section 2.4 à leur assemblage. Par la suite, le chapitre III présentera les blocs spécifiques de notre convolueur, la distribution d'horloge, la méthode de conception et les résultats obtenus.

2.1.1 La convolution 2D

L'opération de convolution 2-D permet d'effectuer différents types de filtrages sur des images. La figure 2.1 illustre comment est effectuée une convolution 3*3 sur une image.

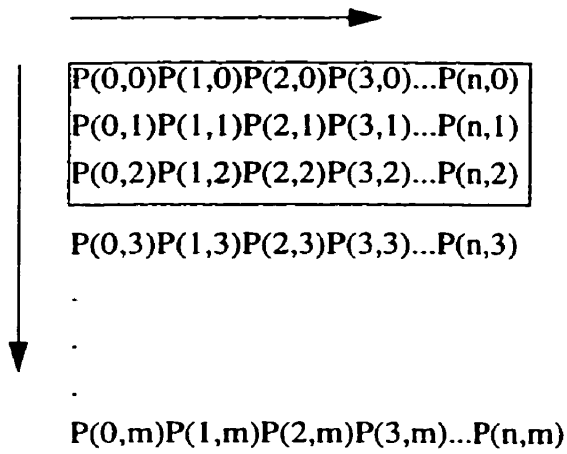


Figure 2.1: Fenêtre de convolution

L'opération consiste à déplacer un masque couvrant 3*3 pixels sur la totalité de l'image; à chaque déplacement du masque (un pixel à la fois, une direction à la fois) le calcul de la convolution est exécuté. La convolution 3*3 revient à faire une somme de produits de pixels (P) et de poids (W), comme l'indique l'équation 2.1:

$$\hat{P}(x, y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} P(x+i, y+j) \times W(i, j) \qquad (2.1)$$

Somme des produits des poids et des pixels

Chacun des poids du masque de convolution est multiplié à un pixel de la fenêtre de convolution selon la règle illustrée à la figure 2.1. La somme des 9 produits pixels*poids représente alors la convolution 3*3 du pixel central de la fenêtre considérée. Ainsi, en déplaçant une fenêtre de convolution semblable à celles proposées dans la figure 2.2, un produit de convolution peut être calculé pour chacun des pixels de l'image.

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

1	2	1
-2	4	-2
1	-2	1

Figure 2.2: Exemples de masque de convolution: (masques Laplaciens)

Parmi les applications de la convolution 2-D, on retrouve toute la gamme des filtres spatiaux FIR (Finite Impulse Response). Par exemple, le filtre moyen est un passe-bas qui permet de réduire le niveau de bruit dans une image et d'interpoler les valeurs de pixels bruités à partir des pixels avoisinants. Le filtre Laplacien permet de détecter les contours dans une image et il met en évidence les maxima et les minima locaux rencontrés.

2.2 Fonction au niveau système du convolueur 3*3

2.2.1 Oscillateur à tension commandé

L'oscillateur représenté dans le schéma-bloc de la figure 2.5 sert à générer l'horloge interne du système. Il devra pouvoir fonctionner entre 100 MHz et 2 GHz. Afin de couvrir toute notre plage de fréquence, nous avons choisi d'utiliser cinq oscillateurs ayant chacun des plages de fonctionnement différentes: à 100 MHz, de 100 MHz à 300 Mhz, de 400 Mhz à 1.2 GHz, de 1 GHz à 1.5 GHz et de 1.5 à 2 GHz . Il s'agit des différents oscillateurs en anneaux présentés dans le chapitre 3. Pour commander ces oscillateurs, nous avons besoin d'une entrée sur laquelle la tension de commande est amenée, et une entrée de sélection de l'oscillateur et d'initialisation.

2.2.2 Générateur de vecteurs pseudo-aléatoires

Le générateur de vecteurs pseudo-aléatoires transmet au convolueur des vecteurs correspondant aux pixels d'une image. Nous effectuons la convolution de 9 pixels par 9 poids. Chaque poids est constitué de 4 bits et un pixel a une largeur de 8bits(non signé). Donc nous avons besoin d'un circuit BIST (Built In Self Test), détaillé dans le chapitre 3, à 8 entrées au minimum (on peut concevoir d'avoir un générateur pour chaque pixel). Si nous voulons générer une séquence de vecteurs de dimension 8*8 pixels (2^{12} bits), nous devons prévoir que le nombre minimal de bits du générateur sera de 13 de façon à couvrir un nombre suffisant de vecteurs. Ce BIST a également été conçu en VHDL afin de vérifier toute

la logique du convolueur et de générer les vecteurs de sortie. Grâce au BIST réalisé en VHDL, nous pourrons également valider le BIST conçu au niveau schématique et schéma des masques. Les contraintes découlant du placement font que l'utilisation de plusieurs générateurs de vecteurs à des positions différentes évite des problèmes de synchronisation entre des entrées très éloignées. Une approche définitive n'a pas été arrêtée: les options retenues sont d'avoir 3 générateurs de 24 bits ou encore 9 générateurs de 13 bits; la décision se prendra par rapport aux contraintes de placement. Le générateur de vecteurs pseudo-aléatoires est développé selon une architecture TSPC afin d'atteindre les performances souhaitées ainsi que des entrées toujours synchronisées sur l'horloge.

2.2.3 Filtres de convolution (ROM)

Les filtres de convolution qui sont disponibles sont conservés dans une mémoire statique. Un simple multiplexeur nous permet de choisir celui qui nous intéresse. Pour l'instant, nous envisageons la possibilité de choisir un filtre parmi 8; donc nous avons une entrée sur 3 bits, ***poids_sel***, qui permet de sélectionner le filtre qui devra être appliqué aux pixels. Ce bloc est dessiné en pseudo N-mos (par opposition aux cellules TSPC); le choix du filtre étant fixe pour toute la durée de la convolution, nous n'avons pas de contraintes de temps pour l'établissement des poids sur leurs lignes de distribution. Pour des raison de placement, il a été convenu d'avoir un minimum de 3 blocs de ROM qui sont disposés près des multiplieurs qui les utilisent. Chaque poids du filtre est représenté par un mot de 4 bits de large. Pour huit filtres, nous avons donc besoin de 32 bits par pixel.

2.2.4 Multiplieurs restreints

Chacun des 9 multiplieurs inclus dans le design du convolueur 3*3 a pour tâche la multiplication de la valeur d'un pixel de la fenêtre de convolution par le poids du masque de convolution qui lui est associé d'après sa position. Pour limiter la complexité requise par 9 multiplieurs complets, les poids du masque de convolution ont été limités aux valeurs {-4, -2, -1, 0, 1, 2, 4}. De nombreux opérateurs utiles en traitement d'images peuvent être élaborés à partir de cet ensemble restreint de poids [17]. Des multiplieurs complets auraient été trop coûteux. De tels multiplieurs ne sont plus nécessaires pour supporter l'ensemble des poids mentionnés auparavant. Des multiplieurs restreints ont alors été conçus pour ne supporter que les poids {-4, -2, -1, 0, 1, 2 et 4}. L'ensemble des actions devant être supportées par ces multiplieurs restreints pouvaient alors être réduites aux opérations élémentaires suivantes:

- décalage d'une ou de deux positions vers la gauche,

- transformation en complément à deux pour représenter les nombres négatifs et

- mise à zéro.

Le tableau 2.1 dresse la liste des opérations élémentaires appliquées par les multiplieurs restreints sur chaque pixel selon la valeur du poids qui lui est associé.

Tableau 2.1 : Opérations du multiplieur restreint

Valeur du poids	Opération effectuée sur la valeur du pixel
-4	décalage de 2 positions vers la gauche suivi d'une transformation en complément à 2
-2	décalage de 1 position vers la gauche suivie d'une transformation en complément à 2
-1	transformation en complément à 2
0	mise à zéro
1	laisse passer le mot pixel
2	décalage de 1 position vers la gauche
4	décalage de 2 positions vers la gauche

La sortie de chaque multiplieur restreint est représentée sur 11 bits afin de préserver la précision complète du résultat. En effet, avec des pixels représentés en format de 8 bits sans signe, pour un décalage maximal de 2 positions et la possibilité d'obtenir des résultats positifs ou négatifs (donc 1 bit de signe), 11 bits suffisent pour transmettre sans ambiguïté le résultat de chaque multiplieur restreint à la prochaine étape de calcul. Cette dernière étant la sommation des 9 produits poids*pixels.

2.2.5 Module de sommation

Le module de sommation s'occupe de cumuler les 9 mots de 11 bits générés par la multiplication des pixels de la fenêtre de convolution par les poids du masque de convolution. Pour effectuer cette sommation de 9 termes, un arbre de modules à stockage de retenue

ou "Carry Save Adders" (CSA) a été choisi pour sa rapidité et sa complexité acceptable. Il est évidemment conçu avec des cellules de type TSPC. L'exemple de la figure 2.4 illustre le principe de fonctionnement du CSA.

Comme nous avons 9 résultats à additionner, il nous faudra plusieurs CSA pour produire le résultat total. L'architecture qui a été retenue est de forme arborescente (figure 2.3), ce qui permet d'obtenir plus rapidement le résultat final par rapport à une forme linéaire.

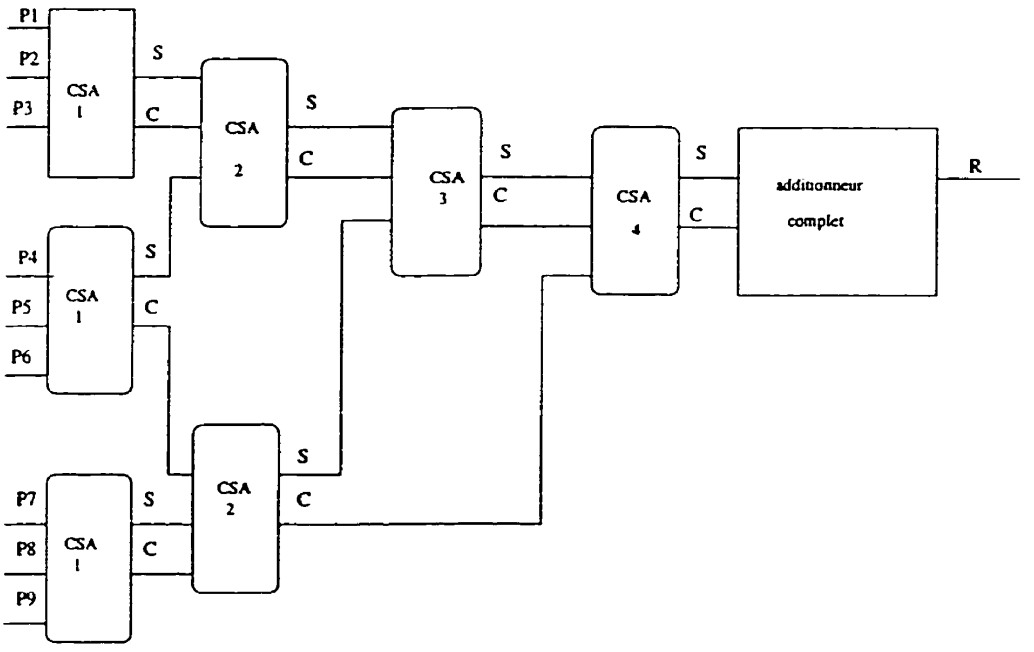


Figure 2.3: Arbre de sommation

X+Y+Z

x = 0 1 0 1 0 1 0 1

y = 1 0 1 0 1 1 1 0

z = 0 1 0 0 0 1 0 1

s = 1 0 1 1 1 1 1 0

c = 0 1 0 0 0 1 0 1 0

r = 1 0 1 0 0 1 0 0 0

Figure 2.4: Principe de fonctionnement du CSA

La figure 2.4 illustre le principe du CSA (addition avec sauvegarde de la retenue). La somme de x,y et z par un additionneur complet donne la somme s et la retenue c. La somme de s et c donne le résultat r de l'addition de x, y et z.

2.2.6 Compacteur (analyse de signatures)

À la sortie du convolueur, un nombre de 15 bits est généré à la fréquence de 1 GHz. Ce nombre ne peut être transmis à la sortie d'un plot car ces derniers ont une bande passante limitée à 300 MHz. Donc dans une première étape, nous réduisons une suite de nombres de 15 bits en une suite de nombres de 1 bit en utilisant des registres d'analyse de signature

à plusieurs entrées. Ces registres sont également réalisés en cellules TSPC. Ce module est également conçu en VHDL toujours dans le souci de valider au niveau fonctionnel tout notre prototype.

2.2.7 Démultiplexeur

Un démultiplexeur sert à distribuer le bit issu du compacteur sur des convertisseurs série parallèle reliées directement aux plots. Ainsi, si le coprocesseur fonctionne à 1 GHz et que 10 sorties parallèles sont reliées au plot, chaque plot verra un bit tous les 10 ns. Donc chaque plot aura une fréquence de 100 MHz, ce qui est acceptable. Le démultiplexeur est conçu à partir de cellules TSPC.

2.2.8 Schéma bloc (figure 2.5) et signaux

Comme notre prototype n'utilise pas un contrôleur, nous avons tous nos signaux de commande qui viennent de l'extérieur. Ainsi pour chaque bloc nécessitant des signaux de contrôle, voici la définition de ses ports d'entrée.

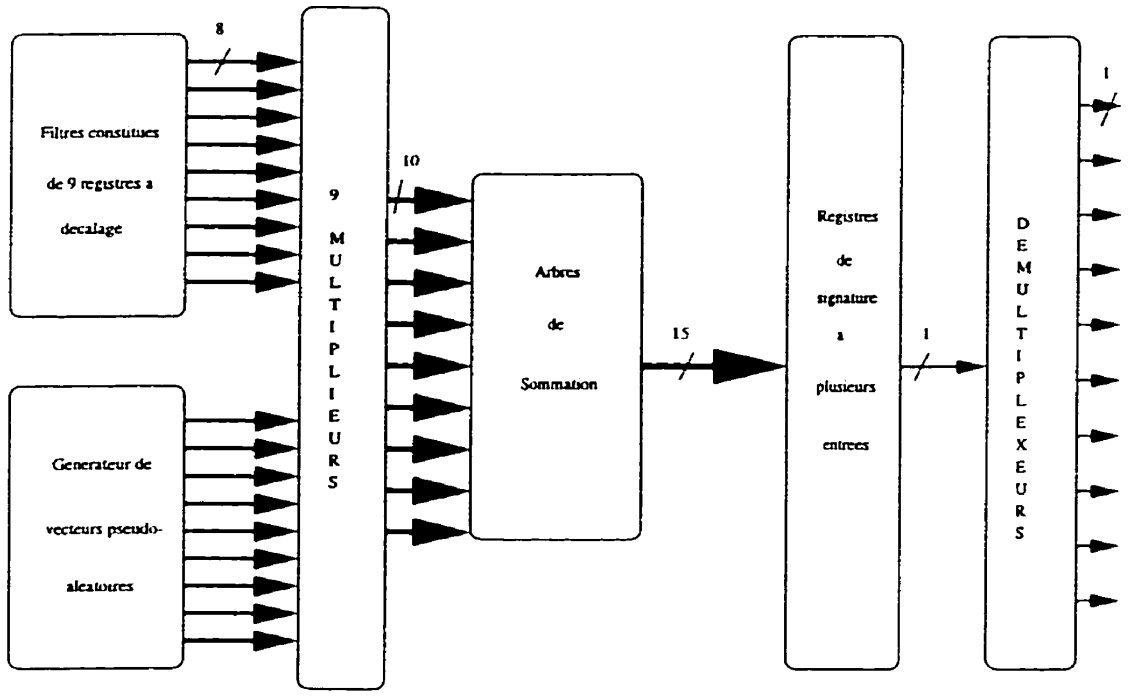


Figure 2.5: Schéma Bloc du prototype convolueur 3*3

- **Oscillateur**: nous avons cinq oscillateurs, trois fonctionnent de 100 MHz à 1 GHz et deux autres prennent le relai de 1 Ghz à 2 GHz. La fréquence d'utilisation est environ de 1 GHz. Le signal ***osc_1*** lance le premier oscillateur tandis que le signal ***osc_2*** lance le second oscillateur... L'impulsion de départ vient du signal ***impul*** envoyé aux différents oscillateurs par ***osc_1*** ou ***osc_2...***

- **Générateur de vecteurs pseudo-aléatoires**: quelque soit le nombre de générateurs pseudo-aléatoires utilisés, ils sont tous initialisés au même moment par le signal ***gen***.

- **Mémoire statique**: la mémoire statique contient les différents poids qui nous sont nécessaires. Nous utilisons 8 filtres parmi les plus courants afin de limiter l'utilisation des plots. Ainsi avec 3 plots, nous pouvons commander la mémoire voulue. Le signal de sélection s'appelle ***poids_sel***.

Multiplieurs: les multiplieurs peuvent recevoir un signal d'initialisation qui s'appelle ***reset***.

Arbre de sommation: l'arbre de sommation génére un nombre de 15 bits qui peut être directement envoyé sur 15 plots de sortie si ***sortie_15*** est activé. Si ***sortie_1*** est choisi alors le mot de 15 bits ira vers le compacteur.

Compacteur et Démultiplexeur: Ces 2 blocs sont synchronisés avec l'horloge et le signal ***demult*** les active. Le démultiplexeur envoit les sorties sur dix plots de sortie qui peuvent être les mêmes que les plots de sortie du sommateur en basse fréquence.

Un certain nombre de signaux sont connectés à la sortie de chaque bloc afin de pouvoir les tester à basse fréquence. Lorsque des signaux de sortie ne sont pas utilisés, il faut prendre soin de les désactiver afin d'éviter de rajouter les capacités des ports de sortie. Un simple transistor commandé par les signaux de commande sera suffisant pour isoler les ports du circuit en jouant le rôle d'interrupteur.

2.3 Conception des cellules de base du convolueur 3*3 TSPC

Dans cette section, nous présentons les cellules qui permettent de réaliser le chemin de donnée (figure 2.6) du convolueur 3*3 TSPC que nous avons développé. Ces cellules de base sont au nombre de trois: un additionneur complet, un registre à décalage d'un bit et un multiplieur restreint. À partir de ces cellules, il est possible de bâtir toute la logique du convolueur 3*3. En effet, le générateur de vecteurs pseudo-aléatoires alimente neuf multiplieurs, qui eux-même alimentent l'arbre de sommation constitué de blocs additionneurs complet et de registres à décalage pour retarder l'entrée des données à additionner. L'additionneur complet de 15 bits est lui-même constitué de registres à décalage et d'additionneurs complets.

Ces cellules sont développées à partir des cellules normalisées TSPC à sortie partagée du chapitre 1. Elles ont été développées de façon complètement dédiée afin de minimiser leur surface et d'obtenir le maximum de performance.

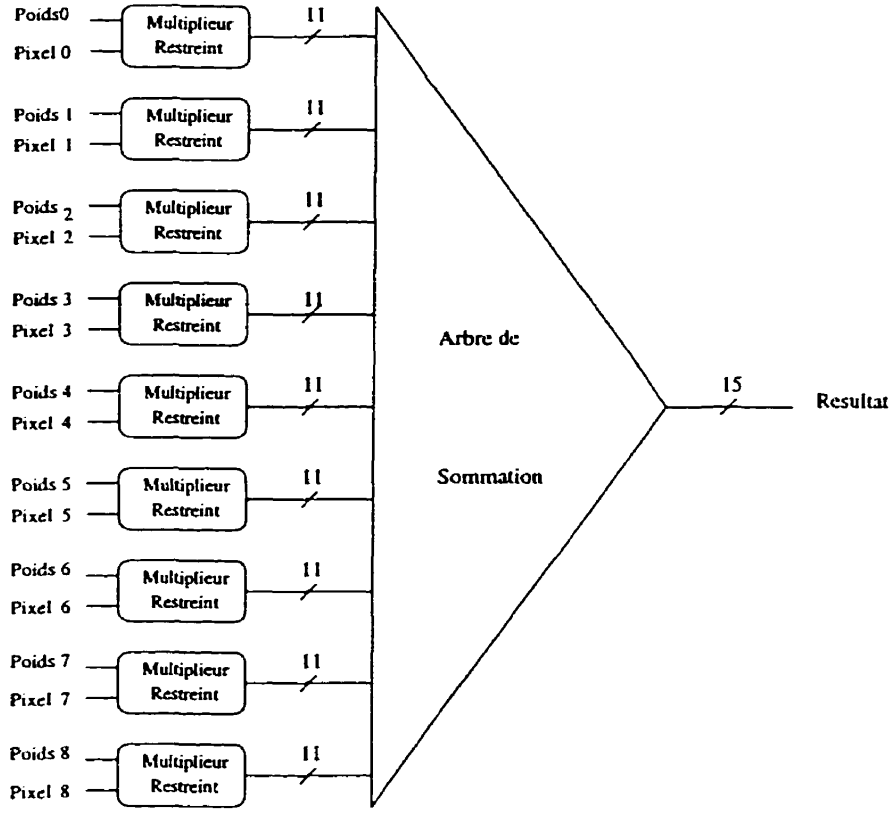


Figure 2.6: Chemin de données

2.3.1 La cellule registre à décalage

Dans notre circuit, nous avons souvent utilisé des registres à décalage afin de propager les signaux qui ne peuvent être utilisés tout de suite comme c'est le cas dans l'additionneur complet pour le signal de retenue (section 2.3.2) et dans l'arbre de sommation où tous les bits ne peuvent être additionnés en même temps.

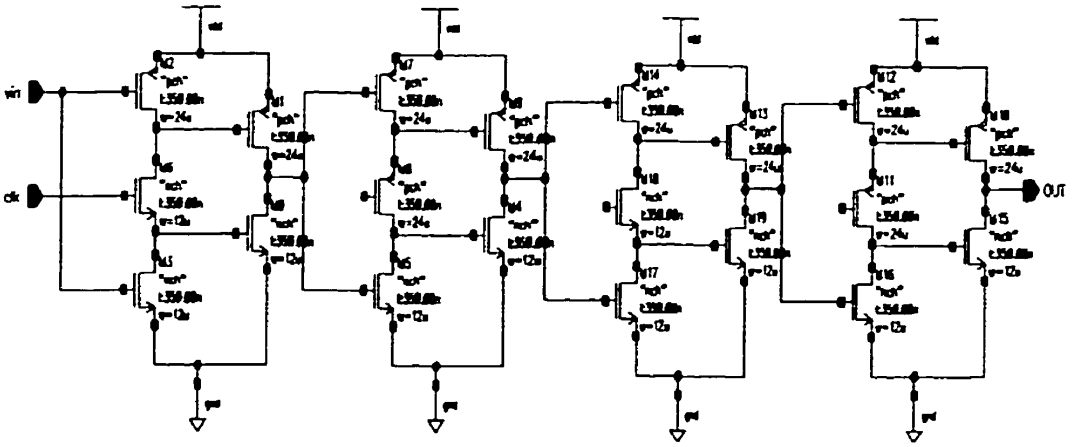


Figure 2.7: Registre à décalage 2 bits

La figure 2.7 propose un registre à décalage de deux bits. Chacun des bits de ce registre est formé de l'alternance d'une bascule TSPC de type N suivie d'une bascule TSPC de type P. La figure 2.8 contient les résultats de simulation d'une chaîne de bascules N et P. On peut voir dans cette simulation le déplacement graduel du signal EN jusqu'aux sortie SN1, SN2, SN3, puis à la sortie SP d'une bascule P. Notre simulation a été effectuée à deux GHz. Le chemin critique se trouve dans la bascule de type P, nettement moins rapide que la bascule de type N.

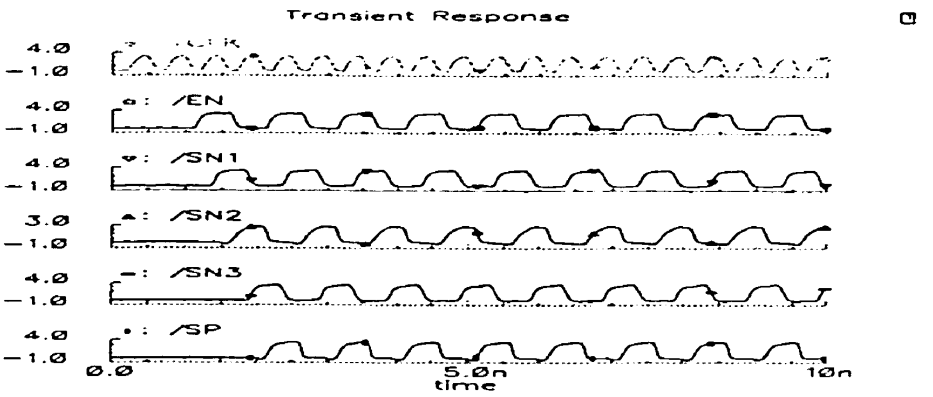


Figure 2.8: Simulation à 2 GHz de la cellule registre à décalage 2 bits

2.3.2 Additionneur complet

Plusieurs cellules additionneur complet, réalisées en technique TSPC, ont été proposées par divers auteurs dont Yuan et Svensson [4], ainsi que Lu, Samueli, Yuan et Svensson [13]. Ces cellules utilisent cependant un mélange des différentes configurations TSPC présentées précédemment, sans aucune justification de la part des auteurs. La figure 2.9 illustre une cellule additionneur complet réalisée exclusivement à l'aide de portes TSPC à sortie partagée. Notre additionneur complet est constitué de 13 cellules de notre bibliothèque de base. La figure 2.9 représente la vue schématique de la cellule.

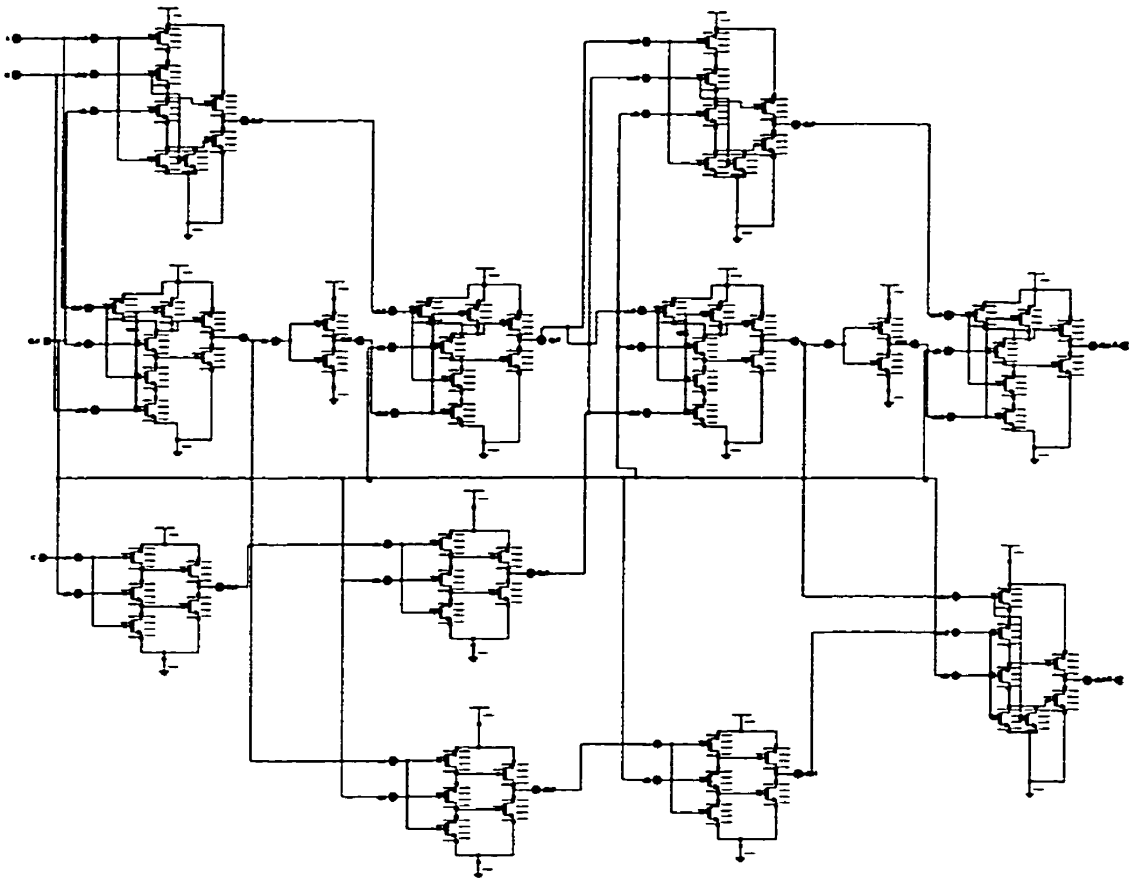


Figure 2.9: Sommateur Complet du convolueur 3*3

L'architecture de cette cellule est plus compréhensible lorsque nous regardons la figure 2.10 qui montre en logique statique la représentation précédente. Nous pouvons remarquer que cette cellule est constituée d'un pipeline de 4 étages. Il faut donc 4 demi-cycles de latence pour produire le résultat. De plus, il y a alternance d'étages N et d'étages P et chaque étage a une latence d'un demi-cycle.

On remarque également dans cette cellule la présence d'inverseurs entre les étages N et P. En effet, nous ne disposons pas de cellules négatives dans notre bibliothèque. Donc

pour obtenir une NAND ou une NOR, il faut ajouter un inverseur à une porte ET ou à une porte OU. Évidement le fait de rajouter un inverseur ajoute un délai non négligeable. Également, il est très difficile de développer des cellules de base balancées à cause de ce problème.

Par simulation, nous avons observé, que le chemin critique de notre additionneur complet se situe au niveau de la porte Nand (formée d'une porte ET suivie d'un inverseur). En effet, les simulations ont montré que toutes les cellules développées avaient une vitesse supérieure à 2 GHz. Par contre, la simulation d'une Nand a donné une fréquence de 1.6 GHz. La simulation de l'additionneur complet donne également une vitesse maximale de 1.6 GHz. Nous sommes donc sûr que notre chemin critique se trouve sur notre Nand N. La porte And N a beau être plus rapide que la porte And P, elle ne peut compenser le fait de lui rajouter un inverseur en sortie. Il est donc préférable d'éviter au maximum la logique négative dans notre architecture. Si elle est nécessaire, il est préférable de rajouter les inverseurs sur des Nmos plus rapides que les étages de type P. Cependant une fois qu'un inverseur a été ajouté dans le flot de donnée comme un NAND, nous pouvons avoir autant de NAND que nous souhaitons car le chemin critique sera toujours le NAND.

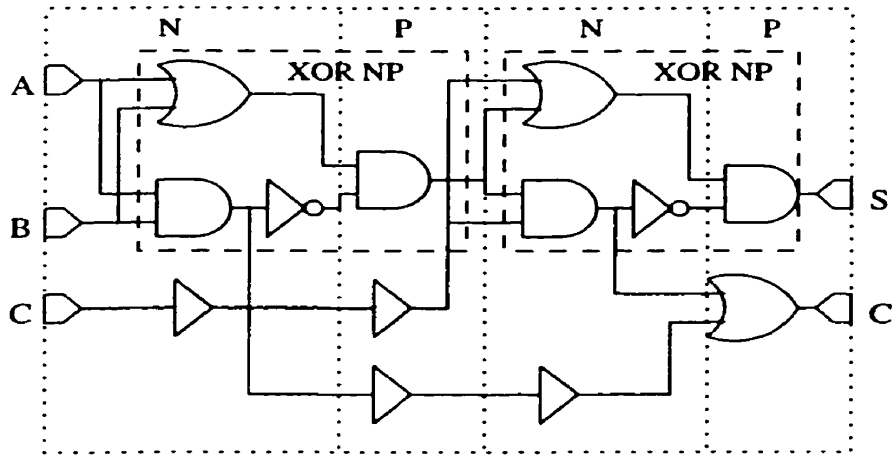


Figure 2.10: Schématique du additionneur complet en logique statique

Il est intéressant d'observer la présence de bascules dans le circuit de la figure 2.10. Le signal C n'intervient que dans le second Xor. Cependant notre circuit est pipeliné et nous voulons que nos signaux arrivent au même moment. Un registre à décalage d'un bit a la même latence qu'un Xor. Nous avons donc intérêt à utiliser cette cellule pour propager le signal C afin qu'il soit disponible au même moment que le signal de sortie de A Xor B.

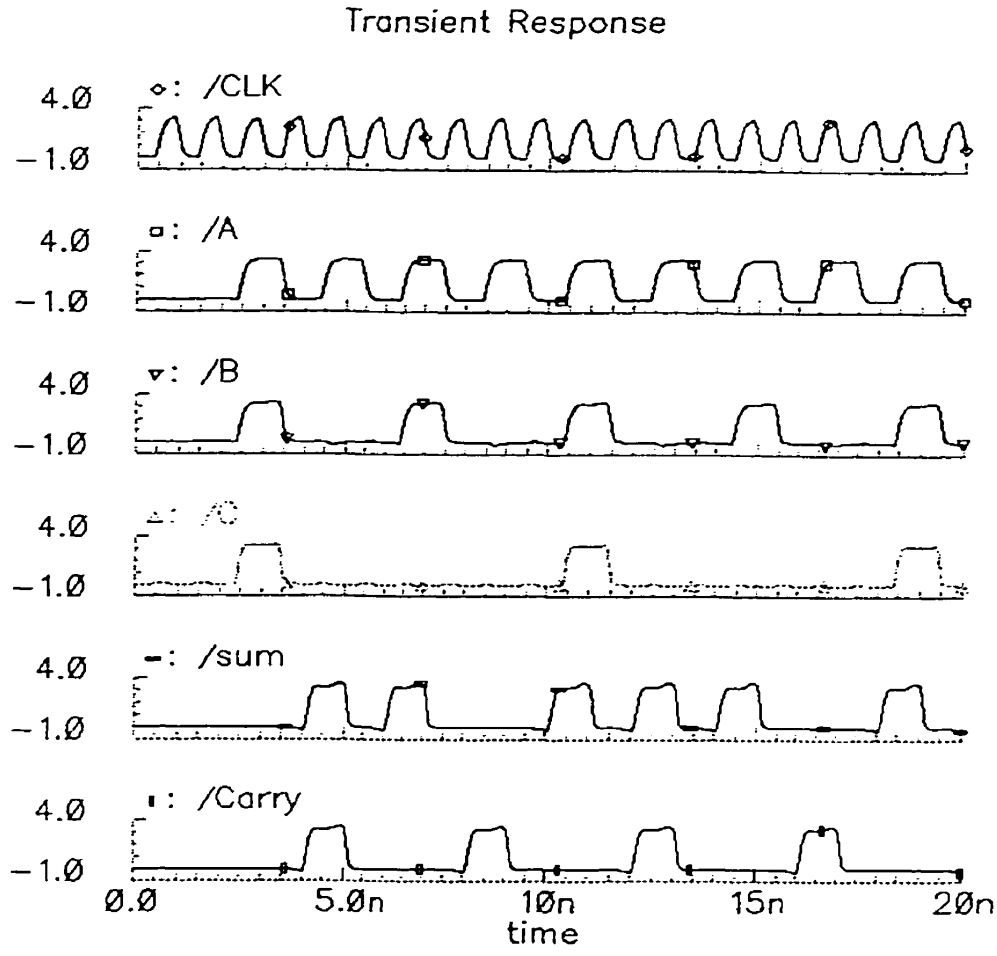


Figure 2.11: Simulation de l'additionneur complet à 1 GHz

Nous remarquons également que la porte Xor montrée à la figure 1.9 et qui fonctionne à 1.1 GHz n'est pas utilisée. En effet, cette porte permettrait de diminuer la latence de l'additionneur complet à un cycle d'horloge. Cependant la fréquence maximale de notre circuit ne serait plus que de 1.1 GHz. Il n'est donc pas indiqué d'utiliser cette cellule dans notre architecture, car elle serait le chemin critique. La Xor TSPC pourrait toutefois être intéressante dans des circuits moins rapides. Son avantage est également une surface de

silicium moindre, car peu de transistors sont nécessaires. De plus, un seul transistor d'horloge est nécessaire ce qui diminue la charge sur l'arbre d'horloge. La dissipation de ce circuit serait également moins élevée que celle d'un Xor constitué de trois cellules de base. Cette cellule serait donc une solution de choix pour des circuits TSPC à plus basse consommation ou pour des circuits nécessitant une fréquence d'opération moins élevée.

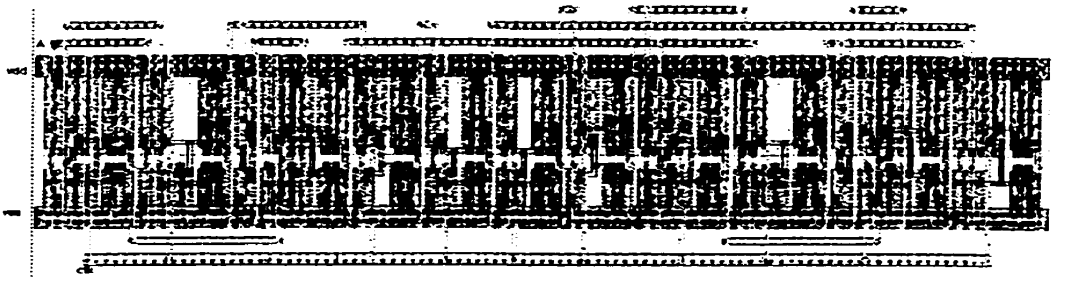


Figure 2.12: Schéma des masques de l'additionneur complet

La figure 2.12 montre le schéma des masques de l'additionneur complet réalisé à partir de notre bibliothèque de cellules normalisées. L'outil de placement et routage donne le même résultat qu'un placement manuel. La figure 2.11 montre la simulation de cette cellule.

2.3.3 Multiplieur-restreint

Le multipleur restreint permet de multiplier un nombre de 8 bits qui correspond à un pixel par des poids limités à l'ensemble {-4, -2, -1, 0, 1, 2 et 4} seulement. Finalement il s'agit d'un décaleur. Bosi [8] avait proposé un circuit possédant 3 cycles de latence que nous

avons réduit à 2 cycles de latence. Nous proposons deux circuits différents permettant d'atteindre cet objectif dans les figures 2.13 et 2.14.

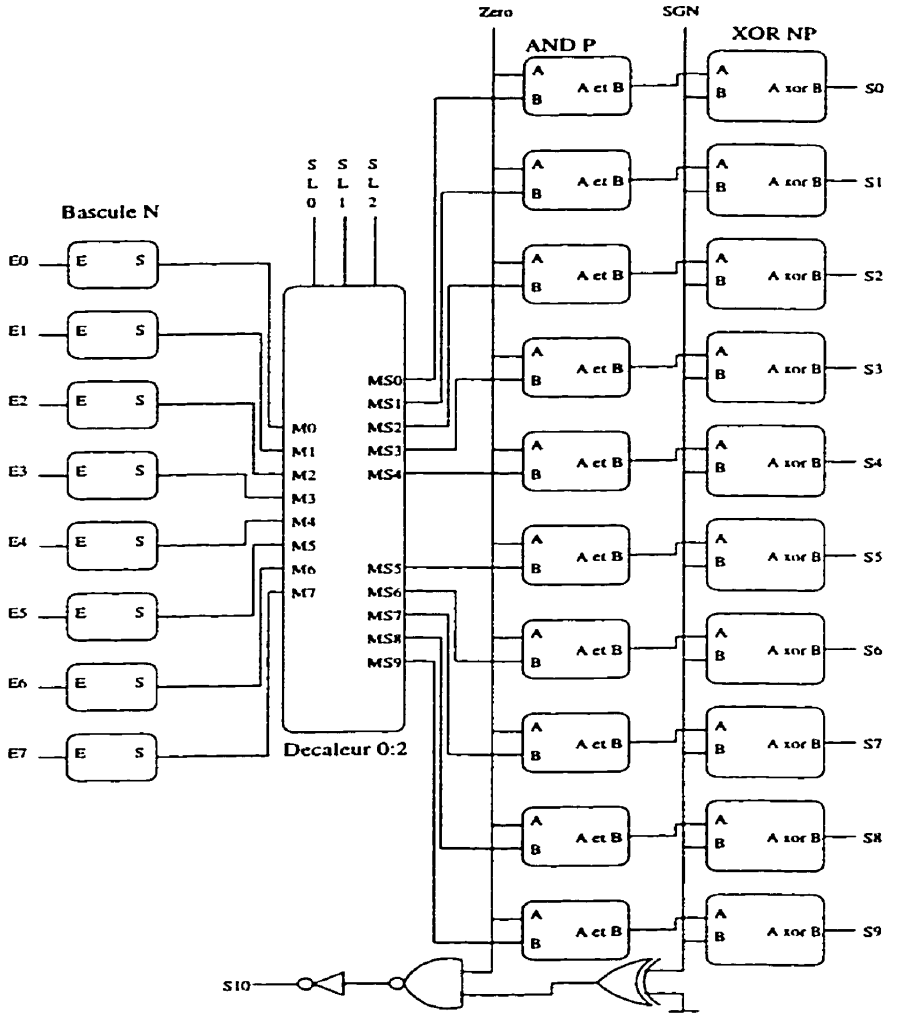


Figure 2.13: Cellule multiplieur restreint

Dans ce premier circuit (figure 2.12), nous avons une architecture proche de celle proposée par Bosi. Nous avons simplement enlevé les deux tampons Pmos. Le premier tampon de son circuit sert à régénérer le signal issu du décaleur tandis que le second est nécessaire

pour finir notre bloc multiplieur sur un étage Pmos afin d'attaquer un nouveau bloc sur un étage Nmos. Nous avons donc remplacé l'étage de réamplification Pmos par une porte AND Pmos, associée au signal zéro. La porte And de type P est aussi efficace qu'une bascule de type P car elle propose deux transistors Pmos en parallèle. Le chemin critique ne se trouve pas sur les deux transistor Nmos en série mais sur les transistors Pmos. Notre XOR N,P finit sur un étage Pmos comme le multiplieur précédent afin de relier cette cellule à des additionneurs complets (XOR N,P car il est constitué d'un pipeline avec un étage N suivi d'un étage P, confert figure 2.10).

Au premier étage N, la valeur du pixel (E0-E7) est décalée vers la gauche de 0, 1 ou 2 positions selon la valeur du poids qui multiplie ce pixel. Les signaux SL0, SL1 et SL2 proviennent de la ROM contenant le poids du masque de convolution associée à ce multiplieur restreint. Ce circuit a fonctionné à 1.6 GHz comme l'additionneur complet. Le chemin critique se trouve toujours dans le NAND du XOR NP.

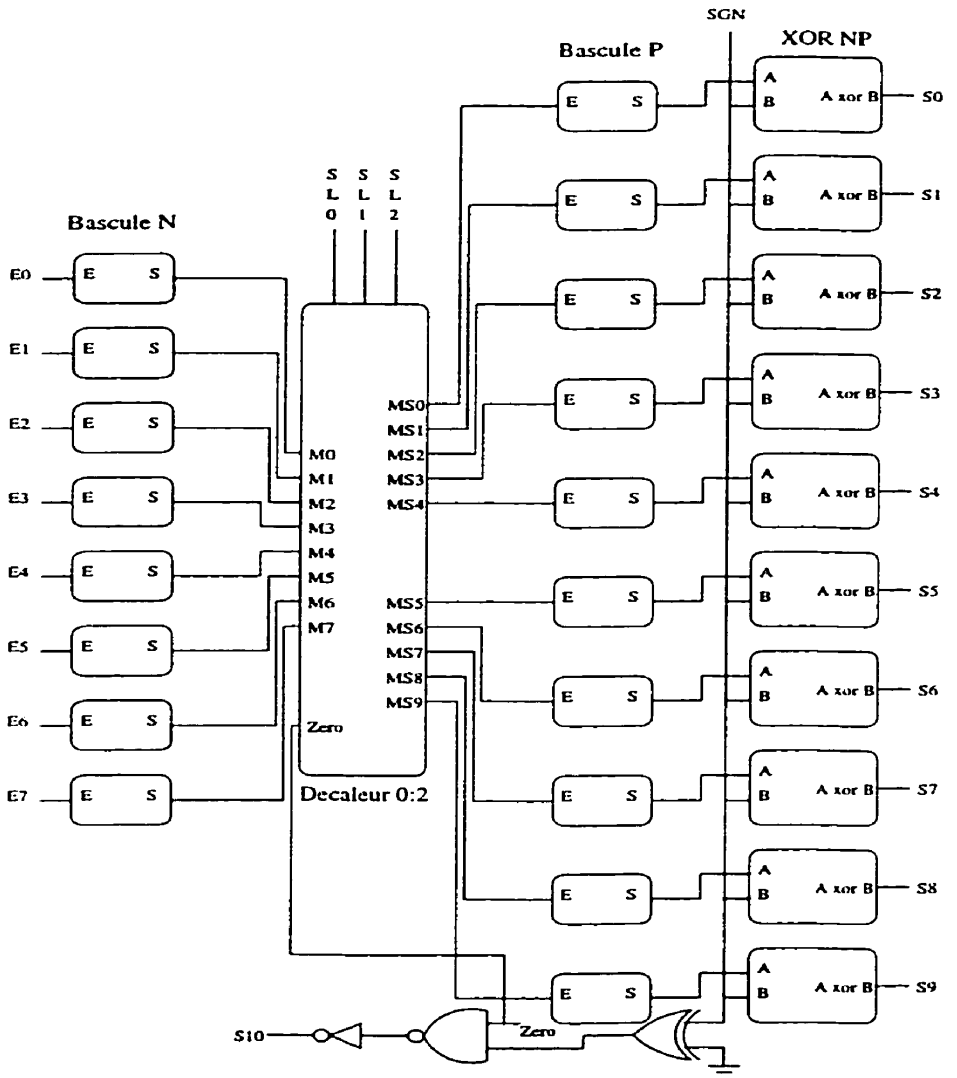


Figure 2.14: Multiplieur restreint avec 4 sélection dans le décaleur

Le circuit de la figure 2.14 poursuit l'objectif précédent. Cette fois nous transformons le bloc décaleur en ajoutant le signal zéro. Nous pouvons donc à nouveau utiliser les bascules Pmos de réamplification. Ce circuit n'a pas été testé mais devrait donner des résultats semblables au circuit précédent de la figure 2.13.

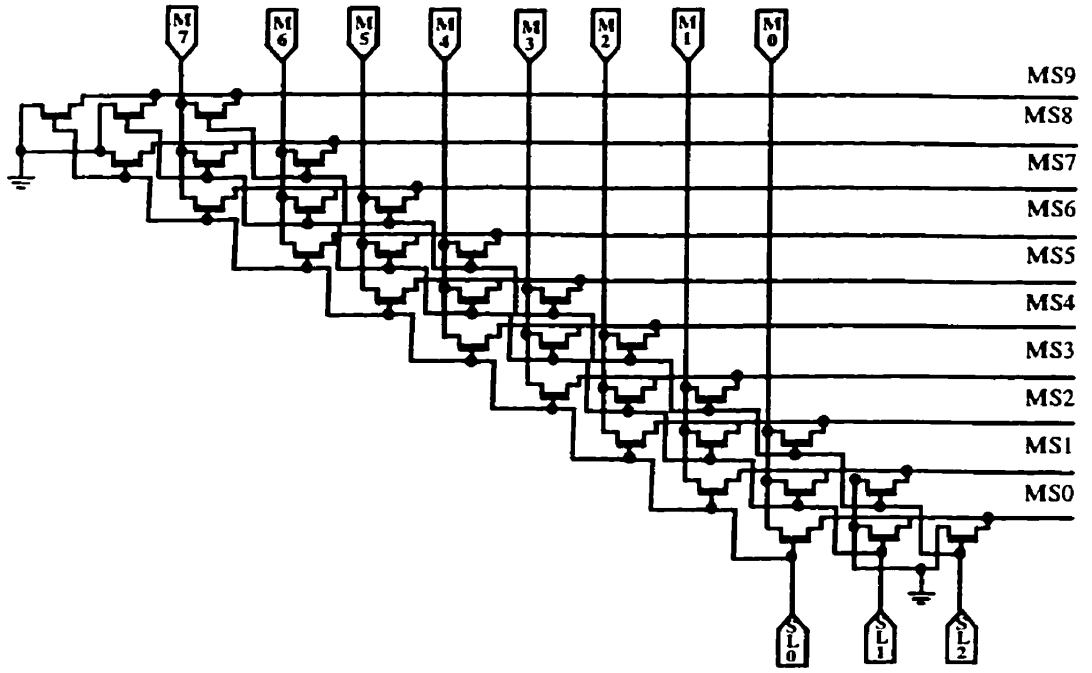


Figure 2.15: Decaleur (0:2)

Comme le montre la figure 2.15, le décaleur (0:2) a été conçu à l'aide de demi-portes de transmission Nmos, ce qui implique que les niveaux logiques hauts sont atténués par la perte d'un seuil de conduction des transistors Nmos. Il est donc préférable de rétablir à leur pleine tension ces signaux, avec un tampon Pmos, avant de les utiliser comme entrées pour d'autres portes logiques. Nous pouvons ainsi utiliser les mêmes cellules XOR-NP que celles dans la cellule additionneur complet.

Les figures 2.16 et 2.17 donnent une simulation du multiplieur restreint de la figure 2.13 au niveau schéma des masques.

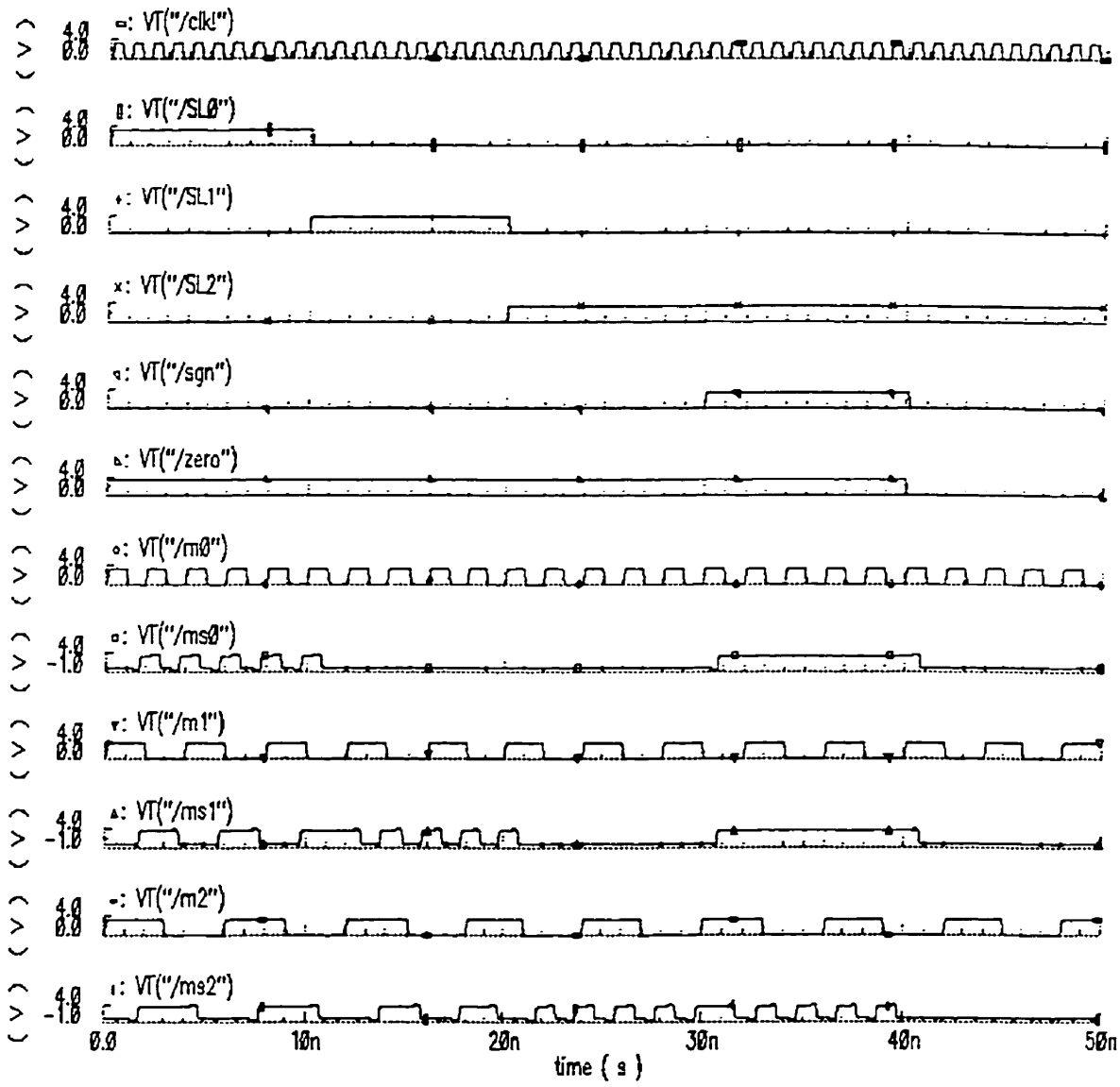


Figure 2.16: Simulation du multiplieur restreint (a)

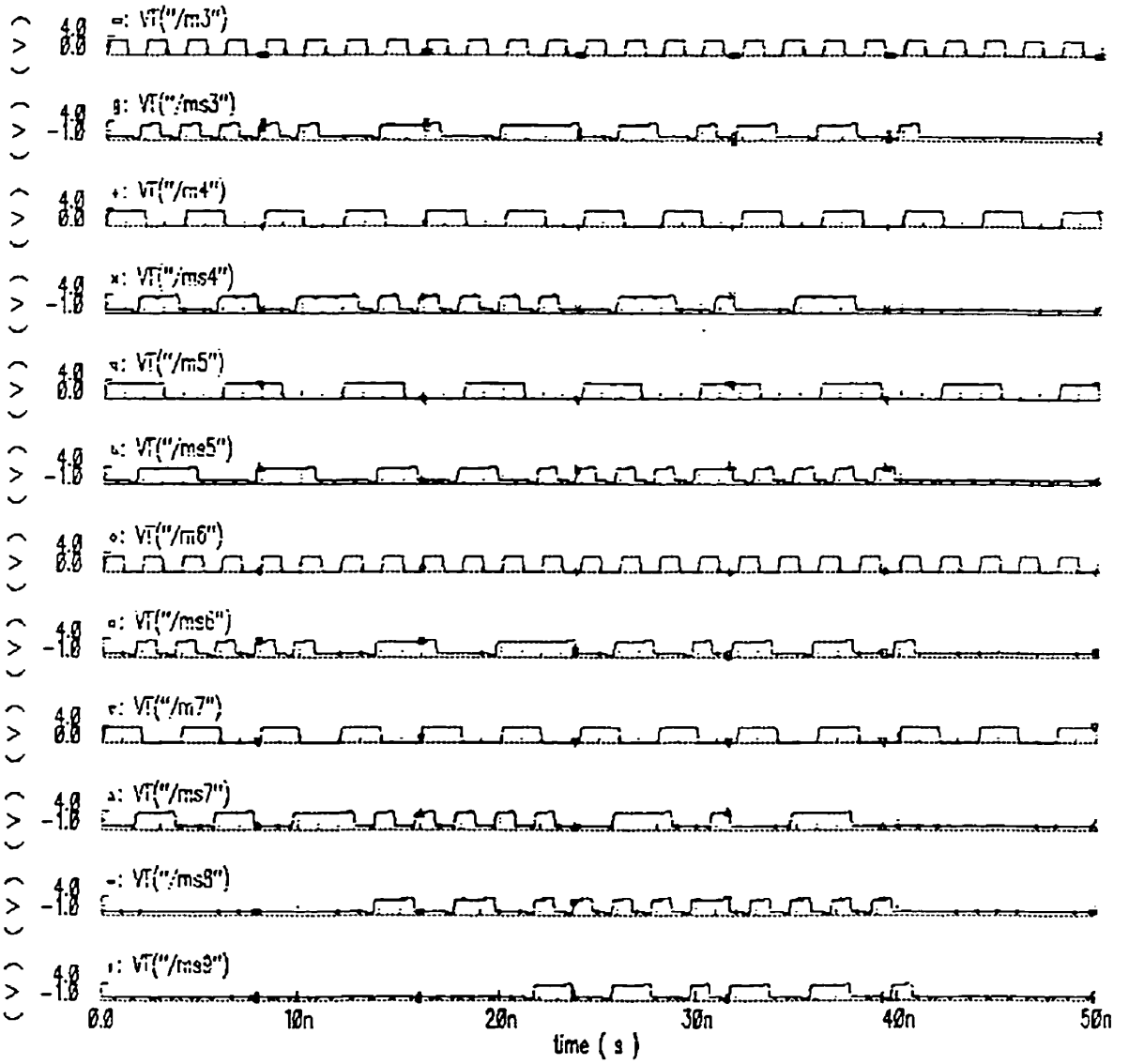


Figure 2.17: Simulation du multiplieur restreint (b)

2.4 Architecture du chemin de données du convolueur 3*3

Les cellules de base définies précédemment permettent de réaliser la totalité du chemin de données du convolueur 3*3. Dans notre chemin de données, nous pouvons remarquer que les registres à décalage propageant les pixels de la fenêtre de convolution ont disparu. En effet, afin de diminuer la complexité du circuit, nous avons seulement conservé l'opération de convolution. Notre bloc est constitué de 9 multiplieurs restreints et d'un arbre de sommation. Les autres blocs formant l'entrée et la sortie du circuit sont présentés au chapitre 3.

2.4.1 Les multiplieurs restreints

Chaque cellule multiplieur restreint du convolueur 3*3 reçoit, comme entrée, la valeur d'un pixel de la fenêtre de convolution, ainsi que les signaux provenant du masque de convolution. Le tableau 2.2 présente le nombre de transistors requis pour concevoir le multiplieur et chacun de ses composants.

Tableau 2.2 : Complexité du multiplieur restreint

Composant	Nombre de composants	Nombre de transistors par composants	Nombre de transistors au total
Bascule-N	8	5	40
Décaleur (0:2)	1	30	30
Xor-NP	10	23	230

Tableau 2.2 : Complexité du multiplieur restreint

Composant	Nombre de composants	Nombre de transistors par composants	Nombre de transistors au total
And-P	10	7	70
Xor CMOS	1	12	12
Nand CMOS	1	4	4
Inverseur CMOS	1	2	2
Total	32		388

Nous remarquons que notre multiplieur permet une économie de près de 100 transistors par rapport au multiplieur proposé par Bosi avec des performances équivalentes, soit 1.6 GHz au maximum à cause du XOR-NP. Notre configuration est composée de 388 transistors. Donc pour les 9 multiplieurs restreints faisant partie du convolueur 3*3, nous avons 9*388 = 3492 transistors.

2.4.2 L'Arbre de sommation

L'arbre de sommation de la figure 2.18 permet d'effectuer la somme des 9 produits poids * pixels. Il est constitué de seulement deux types de cellules. Il s'agit de cellules registre à décalage et de cellules additionneur complet. Le schéma bloc de l'arbre de sommation est représenté à la figure 2.18. Les blocs portant le titre REG_DEC2_XX sont des registres à décalage de taille égale à la valeur XX ayant 2 cycles de latence, alors que les blocs additionneurs_complet_YY sont des colonnes de YY cellules sommateur-complet. Ces derniers blocs agissent donc comme des modules CSA (Carry-Save Adders) puisqu'il n'y a aucune propagation de retenue entre chaque cellule. On notera que chaque cellule

additionneurs_complet_YY accepte comme entrées des signaux de taille égale à (YY-1). Ceux-ci doivent cependant être étendus à YY bits à l'entrée de chaque bloc additionneur complet-YY et Rip-Add-ZZ (Le Rip-Add est un additionneur en cascade, ZZ est son nombre d'additionneur complet) selon les deux règles suivantes:

1)Pour tous les signaux d'entrée E0 à E8 et Cs, ainsi que tous les signaux provenant de la sortie S d'une cellule additionneur complet, une extension de signe doit être effectuée à l'entrée du bloc additionneur complet-YY où ils aboutissent;

2) Pour tous les signaux provenant de la sortie R d'une cellule additionneur complet, un décalage arithmétique d'une position vers la gauche doit être effectuée à l'entrée du prochain bloc (un "0" doit être inséré à la position du bit le moins significatif du signal).

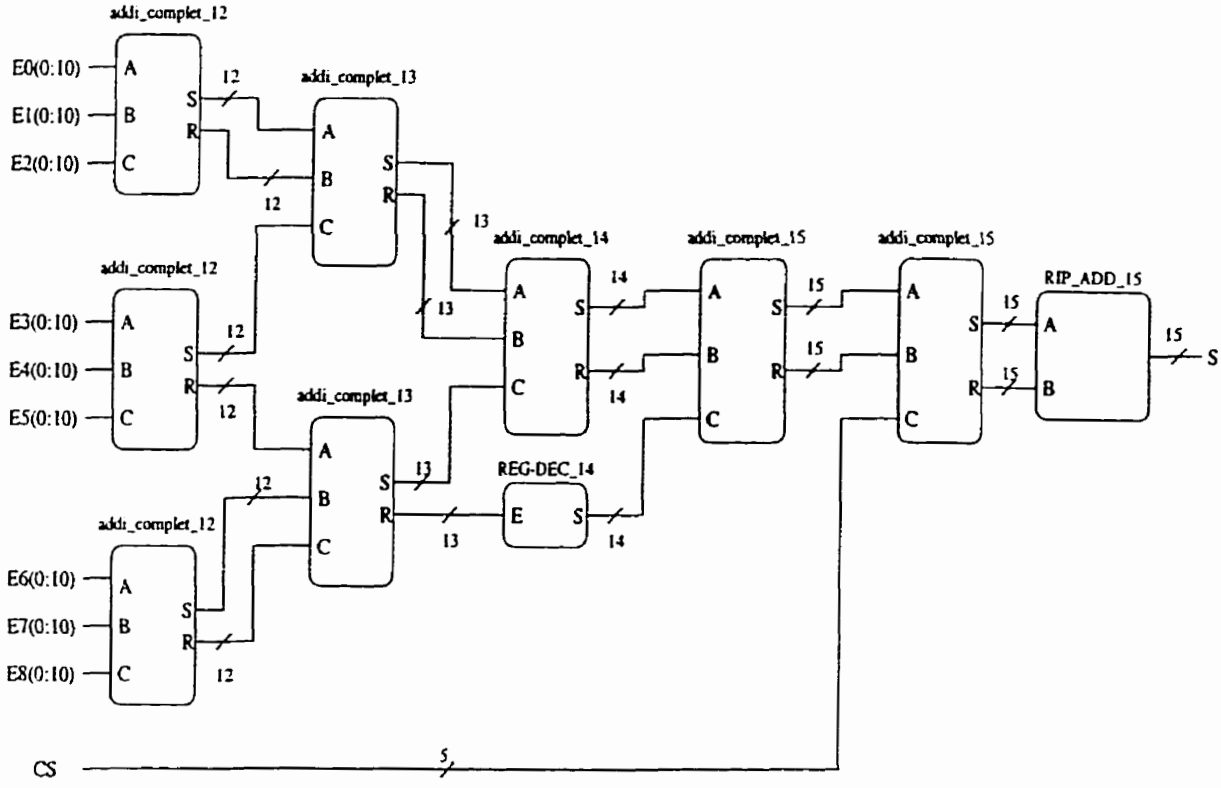


Figure 2.18: Arbre de sommation

L'entrée CS contient les bits de poids négatifs prédéfinis par la ROM. L'arbre de sommation permet de rassembler, en un seul signal, tous les bits qui devaient être ajoutés aux pixels dont les bits avaient été inversés, à cause d'un poids du masque de convolution négatif dans les multiplieurs restreints. Pour changer le signe d'un nombre en complément à deux, ses bits doivent d'abord être inversés, puis le résultat de cette inversion doit être incrémenté. Les inversions ayant déjà été réalisées par les multiplieur restreints, il ne reste plus qu'à effectuer les incrémentations pour compléter le changement de signe des pixels multipliés par des poids négatifs. Au lieu d'inclure un incrémenteur dans chaque multiplieur restreint, on a préféré retarder cette opération pour l'effectuer, à l'aide d'un seul

bloc additionneur complet supplémentaire, dans l'arbre de sommation.

Le dernier bloc de l'arbre de sommation TSPC de la figure 2.19 est un additionneur à propgation de la retenue de 15 bits portant le nom de RIP_ADD_15. Cet additionneur permet de faire la somme des deux derniers termes S et R de l'arbre de sommation. En logique statique, différentes techniques (anticipation de la retenue, sélection de la retenue) permettent de réduire le chemin de propagation du signal de retenue à travers toute la largeur des deux termes à additionner. La technique TSPC rend ces stratégies inutiles, car le chemin de propagation de la retenue d'un simple additonneur en cascade TSPC fait partie intégrante du pipeline formé par les portes TSPC. Le chemin de propagation de la retenue est donc réduit au minimum, soit le délai de propagation à travers une porte TSPC. Le schéma bloc interne de l'additionneur RIP-ADD-15 est illustré à la figure 2.19. Cependant d'autres techniques permettent de réduire la latence de notre RIP-ADD-15 de 2 ou 3 cycles avec un coût supplémentaire en matériel est donc en distribution d'horloge. Un arbre de sommation avec anticipation de la retenue utilisera d'avantage de transistors par exemple.

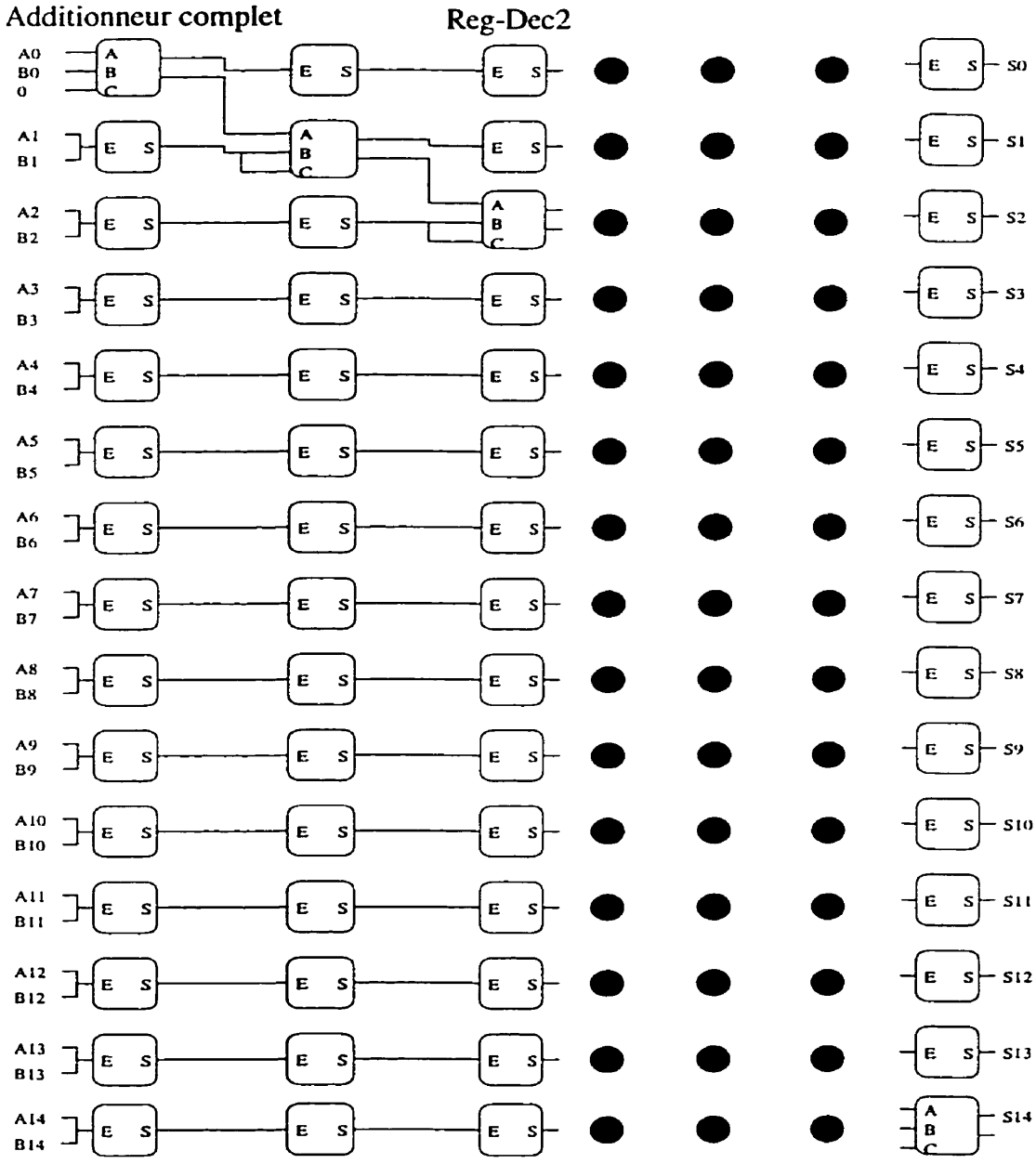


Figure 2.19: Rip-Add_15

Nous remarquons que la latence de l'additionneur en cascade de 15 bits est égale à la latence de 15 cellules additionneur complet. En effet, un bit du résultat final est produit à la

sortie de chaque cellule additionneur complet, alors que la retenue produite par cette cellule sert à calculer le prochain bit du résultat. On notera que les bits des deux termes à additionner doivent être retardés jusqu'à ce qu'ils arrivent à l'entrée d'une cellule additionneur complet. De même, les bits du résultat final doivent être acheminés dans des registres à décalage pour qu'ils parviennent tous en même temps aux sorties S(0) à S(14). Le Tableau 2.3 dresse le bilan du nombre de transistors de l'arbre de sommation.

Tableau 2.3 : Complexité de l'arbre de sommation

Composant	Nombre de composants	Nombre de transistors par composant	Nombre de transistor au total
REG-DEC2 14	1	280	280
additionneur complet_12	3	876	2628
additionneur complet_13	2	949	1898
additionneur complet_14	1	1022	1022
additionneur complet_15	2	1095	2190
RIP-ADD_15			
REG-DEC2	105	20	2100
additionneur complet	15	73	1095
			11213

Conclusion

Ce chapitre a présenté l'architecture logique d'un convolueur 3*3 TSPC. L'architecture est basée sur un nombre d'opérations important et répétitif. Nous faisons appel au parallélisme et au pipelinage. Notre architecture montre les avantages et les inconvénients des circuits à granularité très fine. Ils permettent évidemment des fréquences d'horloge très élevée, de l'ordre de 1 GHz. Cependant, ils procurent une latence également importante. Nous allons maintenant nous intéresser à la réalisation de cet ASIC en présentant des méthodes de conception adaptées pour la haute fréquence.

CHAPITRE 3

BLOCS DE TEST ET MÉTHODE DE CONCEPTION DU CONVOLUEUR 3*3

3.1 Introduction

Maintenant que nous avons défini l'architecture du convolueur 3*3, nous présentons les différents circuits de test, les horloges, la distribution d'horloge, le placement et routage, les problèmes de dissipation de puissance et les solutions que nous avons adoptées devant chaque problème. Tous les circuits doivent être opérationnels à 1 GHz et nous devons également être capables de lire les données de sortie sur des plots limités à une fréquence maximale de 300 MHz.

Nous nous proposons donc de présenter les différentes techniques que nous avons adoptés devant la difficulté de concevoir un ASIC fonctionnant à 1 GHz dans une technologie sub-micronique courante.

3.2 Bloc "oscillateurs"

3.2.1 Oscillateurs utlisés

L'objectif de notre ASIC est de fonctionner à une fréquence proche du GHz. Cependant,

devant la difficulté de le tester à haute fréquence et devant l'objectif de la performance à atteindre, nous avons conçu ce circuit afin de pouvoir le tester sur une large plage de fréquence tout en augmentant graduellement cette fréquence. Nous souhaitons fonctionner à 100 MHz afin de valider notre circuit et nous voulons augmenter progressivement sa fréquence d'utilisation afin de connaître ses limites thermiques et de synchronisation. La fréquence maximale de notre bloc "oscillateurs " est de 1.9 GHz, car les variations de procédés et la précision des simulateurs à haute fréquence peuvent donner des résultats très inférieurs à ceux obtenus lors des simulations. Nous avons développé pour atteindre cet objectif cinq oscillateurs en anneau à tension commandée. Les trois premiers oscillateurs de la figure 3.1 sont constitués par une chaîne d'inverseurs C^2MOS, il suffit de varier le nombre d'étages et la taille des transistors pour atteindre la plage de fréquence souhaitée.

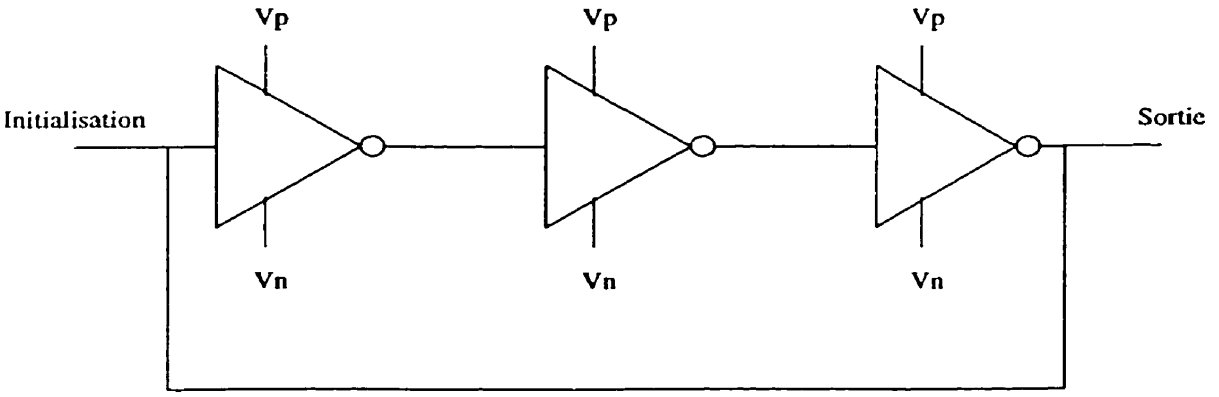


Figure 3.1: Oscillateurs en Anneau à Tension commandée

Tableau 3.1 : Fréquence d'oscillation (GHz) pour Osci1 en fonction de la polarisation vp et vn.

	vp		
vn	0v	0.66v	1.32v
3.3v	0.30	0.28	0.24
3v	0.297	0.291	0.24
2.6v	0.293	0.278	0.24
2.2v	0.28	0.27	0.23
1.8v	0.26	0.25	0.23
1.4v	0.197	0.192	0.178
1.2v	0.146	0.142	0.135

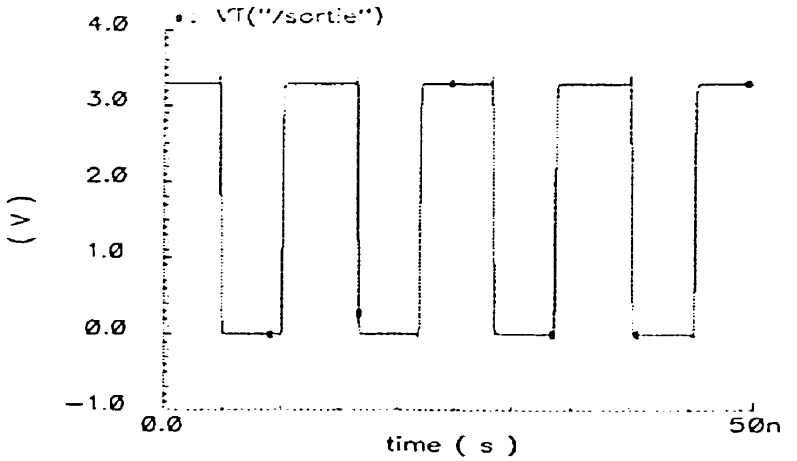


Figure 3.2: Signal de sortie de l'oscillateur 1 à 185 MHz

Tableau 3.2 : Fréquence d'oscillation (GHz) pour Osci2 en fonction de la polarisation vp et vn.

	vp		
vn	0v	0.66v	1.32v
3.3v	0.66	0.63	0.55
3v	0.65	0.62	0.552
2.6v	0.64	0.62	0.54
2.2v	0.62	0.60	0.53
1.8v	0.57	0.55	0.50
1.4v	0.42	0.40	0.38
1.3	0.36	0.35	0.335

Tableau 3.3 : Fréquence d'oscillation (GHz) pour Osci3 en fonction de la polarisation vp et vn.

	vp		
vn	0v	0.66v	1.32v
3.3v	1.15	1.10	1.10
3v	1.14	1.09	1
2.6v	1.11	1.07	0.98
2.2v	1.07	1.03	0.94
1.8v	0.98	0.95	0.88
1.4v	0.73	0.70	0.67
1.3v	0.63	0.61	0.58

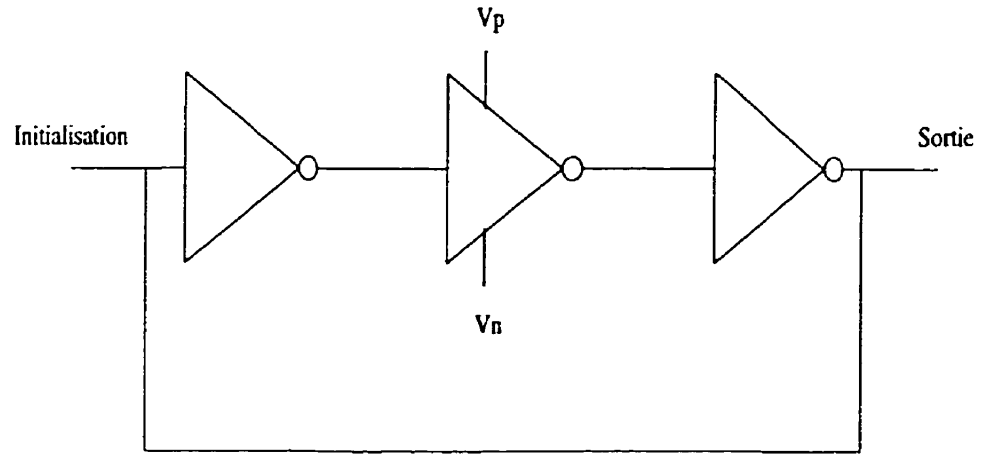


Figure 3.3: Oscillateurs en Anneau proposé par Nikili

Par contre des oscillateurs comme celui proposé à la figure 3.3 permettent d'atteindre des fréquences nettement plus élevées, car ils n'utilisent qu'un seul inverseur à tension commandée. Les autres inverseurs n'ont pas de transistors de régulation en série et ils peuvent aller plus vite. Un signal plus rapide est donc obtenu à la sortie de l'inverseur à tension commandée. Les tableaux 3.4 et 3.5 montrent la plage de fonctionnement des deux oscillateurs les plus rapides.

Tableau 3.4 : Fréquence d'oscillation (GHz) pour Osci4 en fonction de la polarisation Vp et Vn.

	vp		
vn	0v	0.66v	1.32v
3.3v	1.5	1.47	1.42
3v	1.5	1.47	1.41
2.6v	1.48	1.46	1.40
2.2v	1.46	1.44	1.38
1.8v	1.41	1.39	1.34

Tableau 3.4 : Fréquence d'oscillation (GHz) pour Osci4 en fonction de la polarisation Vp et Vn.

	vp		
1.4v	1.24	1.23	1.21
1.3v	1.17	1.16	1.15

Tableau 3.5 : Fréquence d'oscillation (GHz) pour Osci5 en fonction de la polarisation Vp et Vn.

	vp		
vn	0v	0.66v	1.32v
3.3v	1.9	1.88	1.82
3v	1.9	1.87	1.81
2.6v	1.88	1.86	1.8
2.2v	1.85	1.84	1.77
1.8v	1.78	1.77	1.72
1.6v	1.7	1.69	1.66
1.5	1.65	1.64	1.61
1.4v	1.57	1.57	1.56

Le tableau 3.6 donne la dimension des transistors des inverseurs dans les oscillateurs ainsi que leur nombre d'étages. Nous pouvons visualiser les schémas des masques et les performances dans l'annexe 2.

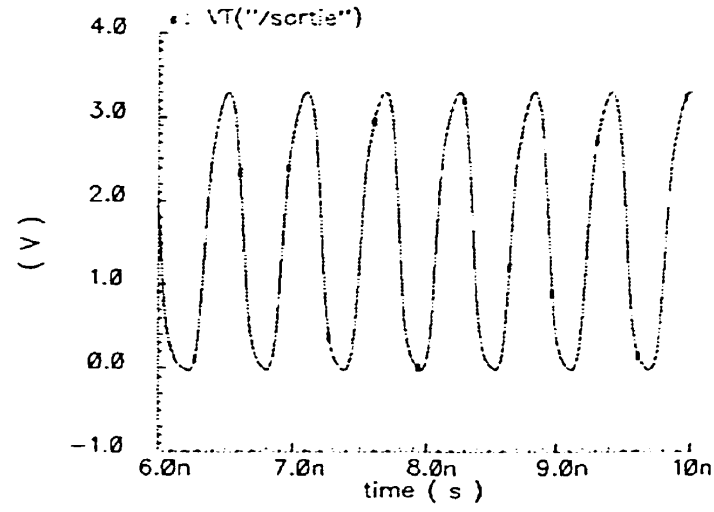


Figure 3.4: Oscillateur5 à 1.7 GHz

Tableau 3.6 : Paramètres des oscillateurs utilisés

	Oscillat-eur1	Oscillat-eur2	Oscillat-eur3	Oscillat-eur4	Oscillat-eur5
largeur du transistor Pmos	4	4	10	8	20
largeur du transistor Nmos	2	2	5	4	10
Nombre d'étages	7	3	3	3	3

3.2.2 Assemblage des oscillateurs

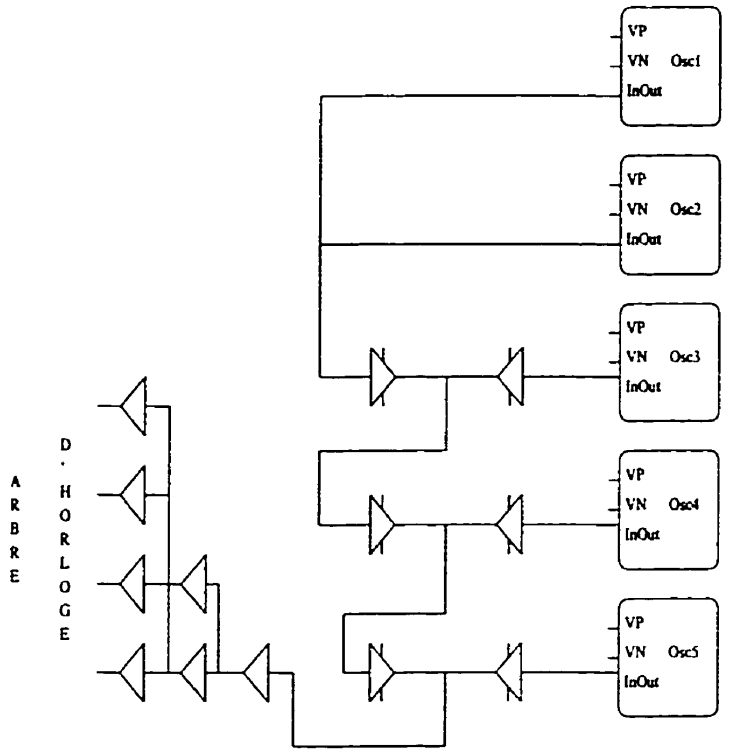


Figure 3.5: Bloc d'oscillateurs

Le Bloc d'oscillateurs de la Figure 3.5 illustre la méthode que nous avons utilisée pour commander et propager les signaux de tous les oscillateurs. Chaque oscillateur est à tension commandée. Afin d'éviter la charge découlant de la mise en parallèle de cinq oscillateurs, nous avons utilisé une propagation sérielle semblable à la propagation d'une chaine de retenue à travers des inverseurs à tension commandée. Les inverseurs à tension commandée utilisés sont les même que ceux utilisés dans les oscillateurs. Nous avons seulement ajusté les tailles des transistors pour gérer les problèmes de charge et de propagation

de signaux carrés. Nous pouvons remarquer qu'à 1.6 GHz, notre signal de sortie est davantage sinusoïdal. Nous avons également placé les oscillateurs par fréquence croissante, en effet, les inverseurs à tension commandée ont un effet néfaste sur les signaux rapides. Notamment, l'oscillateur5 qui avait une fréquence maximale de 1.9 GHz, lorsque simulé en isolation, n'a plus qu'une fréquence maximale de 1.7 GHz, ce qui est très correct tout de même.

3.2.3 Diviseur de fréquence

Pour connaître exactement la fréquence des oscillateurs fournis par le bloc générateur de fréquence, nous avons connecté un diviseur de fréquence par 8 directement à un plot de sortie. Ainsi, si le signal d'horloge oscille à 1 GHz, nous pouvons lire 125 MHz sur notre plot de sortie. Ce diviseur de fréquence est conçu à l'aide de 3 bascules T réalisées en logique TSPC. D'après Chtchvyrkov [9], les diviseurs de ce type peuvent fonctionner à des fréquences élevées, proche de la limite caractéristique de la technologie utilisée. En simulation, ces bascules donnent des résultats satisfaisants jusqu'à 2 GHz. La figure 3.6 présente le schématique d'un des étages (une bascule T) du diviseur de fréquence que nous avons utilisé.

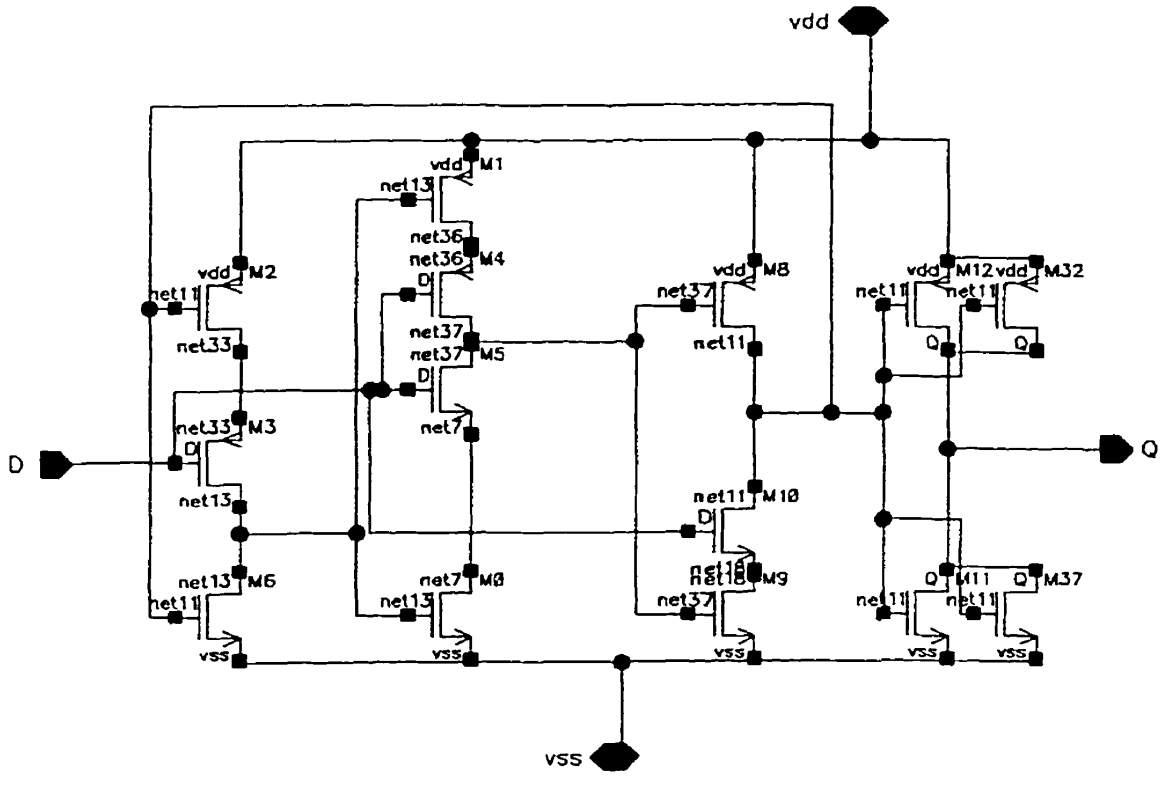


Figure 3.6: Diviseur dynamique de fréquence

3.3 Bloc Générateur de vecteurs pseudo-aléatoires

Le bloc générateur de vecteurs pseudo-aléatoires a été conçu afin de remplacer les pixels d'une image venant de l'extérieur. Nous avons inclu trois générateurs de vecteurs de 13 bits. En effet, pour des problèmes d'interconnexions trop importants, chaque groupe de 3 pixels est modélisé par un générateur de vecteur de 13 bits. Nous tirons 8 bits pour obtenir un pixel, chaque pixel utilise des bits différents ou dans un ordre différent. Nous pouvions nous contenter d'un générateur de 8 bits, mais pour avoir un nombre de vecteurs consé-

quent et proche d'une image 64*64 pixels, nous avons opté pour 13 bits. Le fait d'utiliser un seul générateur de vecteurs serait une contrainte trop importante sur la longueur des fils, car les multiplieurs recevant en entrée les pixels sont souvent très éloignés. Ces générateurs de vecteurs ont été également utilisés afin d'éviter de compliquer notre circuit par la réception de pixels venant d'un bus à l'extérieur.

Évidemment, ces générateurs doivent également fonctionner à plus d'un GHz. La seule nécessité est donc d'utiliser des techniques semblables à celles utilisées dans le chemin de données du convolueur. La référence [14] propose une méthode de conception de générateur de vecteurs pseudo-alétaoires en TSPC. Cet article propose une méthode de test basée sur le circuit "Built in Self Test" adapté à la méthode de conception TSPC. Le schéma proposé est basé sur les automates cellulaires que nous allons présenter maintenant. La figure 3.7 présente la structure de notre BIST.

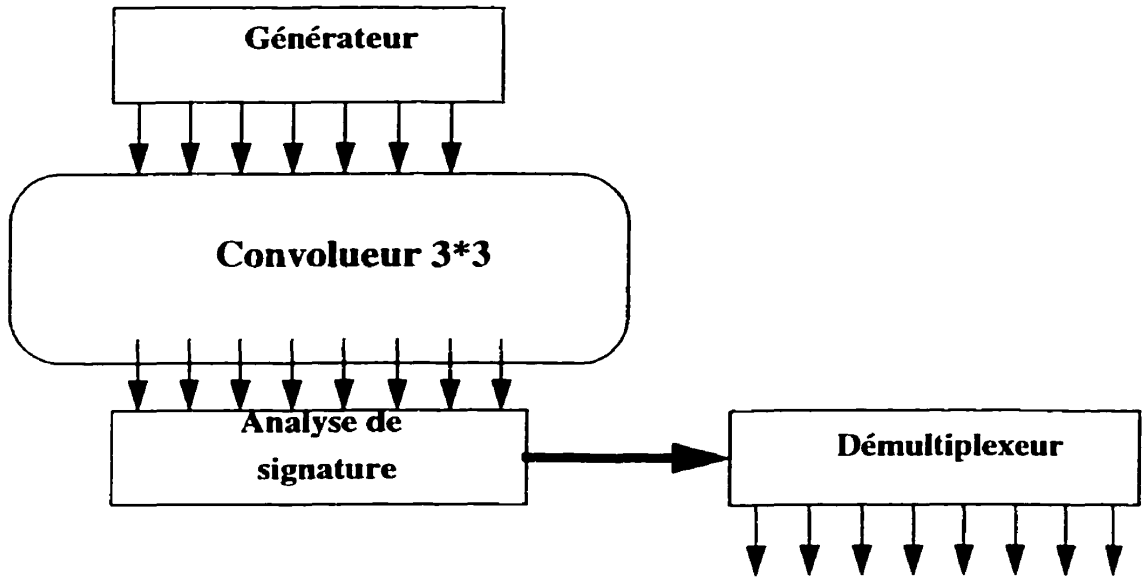


Figure 3.7: Structure du BIST

Les avantages d'un circuit avec auto-test intégré sont:

- la réduction du coût et de la complexité de génération des vecteurs de test,

-la réduction de la complexité et du coût des équipements de tests externes,

-l'exécution du test à la vitesse de fonctionnement du circuit d'où une réduction du temps de test et une meilleure couverture de pannes.

3.3.1 Principe des Automates Cellulaires

Les Automates Cellulaires "AC" sont obtenus par la connexion uniforme de machines à états finis similaires dans un espace de *n* dimension. L'état suivant de chaque cellule est

déterminé d'une manière synchrone, selon une fonction locale de transition appelée ***la règle de l'AC***. Cette dernière dépend de l'état présent de la cellule considérée et de l'état d'un ensemble de cellules appelé ***voisinage de la cellule***.

D'une manière générale, les ACs sont caractérisés par quatre propriétés principales: la dimension de l'AC, la spécification du voisinage, le nombre d'états par cellule et la règle associée à chaque cellule.

Nous considérons un AC unidimensionnel où chaque cellule peut avoir deux états possibles, soit l'état "0" ou l'état "1". La spécification du voisinage d'une cellule est définie par le nombre de cellules connectées à la cellule considérée. Dans notre cas, nous considérons un AC dont les interconnexions entre les cellules sont restreintes aux deux cellules voisines les plus proches: la cellule de gauche et celle de droite. Enfin, la règle de l'AC est définie comme étant une fonction locale associée à chaque cellule qui spécifie la façon dont chaque cellule détermine son état suivant. La figure 3.8 et le tableau 3.6 montrent la structure d'un AC (composée de n cellules) telle qu'expliquée plus loin. Pour les cellules situées aux extrémités de la chaîne et qui n'ont qu'une seule cellule au voisinage, deux conditions aux extrémités sont possibles: une condition constante et une condition périodique aux extrémités. La condition constante aux extrémités est représentée par quatre nombres binaires de deux bits chacun: 00, 01, 10 ou 11. Le premier (respectivement deuxième) bit représente la valeur constante du voisin fictif de gauche (repectivement droite) de la chaîne. En ce qui concerne la condition périodique aux extrémités, les deux cellules situées aux extrémités sont reliées entre elles de sorte que la chaîne forme une

boucle.

3.3.2 Représentation d'une règle d'Automate Cellulaire

La règle de l'AC est exprimée sous forme numérique en prenant l'équivalent décimal de la suite binaire obtenue en sa sortie pour toutes les combinaisons possibles des entrées. La figure 3.8 donne un exemple d'une règle d'AC. La première rangée représente toutes les combinaisons qui peuvent être appliquées en entrée de la fonction locale de transition d'états de la cellule considérée, c'est-à-dire toutes les combinaisons possibles des états de la cellules considérée et des états de ses deux voisins.

	111	110	101	100	011	010	001	000
Règle 90	0	1	0	1	1	0	1	0
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Figure 3.8: Exemple d'une règle 90 d'un AC

Tableau 3.7 : Table de vérité de la règle 90

Entrées de la fonction	Sorties de la fonction
0 0 0	0
0 0 1	1
0 1 0	0
0 1 1	1
1 0 0	1

Entrées de la fonction	Sorties de la fonction
1 0 1	0
1 1 0	1
1 1 1	0

Il y a au total $2^3 = 8$ combinaisons possibles qui sont données par ordre décroissant de la gauche vers la droite. En dessous de chaque combinaison, on retrouve la valeur de la fonction locale de transitions d'états lorsque cette combinaison est appliquée en entrée. Les valeurs en sortie de la fonction constituent un nombre binaire de 8 bits compris entre 0 et 255 dont le bit de poids faible est à droite et en dessous de la combinaison 000. L'équivalent décimal de ce nombre correspond à la représentation numérique de la fonction ou de la règle d'AC. Le nombre binaire 01011 1010 (la deuxième rangée de la figure 3.6) a pour équivalent décimal le nombre 90. Celui-ci est utilisé pour donner le nom de la règle.

Une règle d'AC peut être aussi exprimée par une fonction booléenne, puisqu'elle peut être décrite sous forme d'une table de vérité à 8 entrées. Le tableau 1 donne la table de vérité de la règle 90. La fonction booléenne correspondante de la règle 90 est donnée ci-dessous:

Règle 90: $$x\begin{pmatrix} t+1 \\ i \end{pmatrix} = x\begin{pmatrix} t \\ i-1 \end{pmatrix} \otimes x\begin{pmatrix} t \\ i+1 \end{pmatrix}.$$

La règle 150 est basée sur le même principe:

$$x\begin{pmatrix} t+1 \\ i \end{pmatrix} = x_{i-1} \oplus x_i \oplus x_{i+1}$$

Des règles de construction pour générer des cycles de longueur maximale avec un AC sont présentées dans [15]:

Le tableau 3.8 résume quelque-unes de ces règles de construction. Dans ce tableau, les 1 correspondent à des XOR3 et les 0 à des XOR2.

Tableau 3.8 : Règles de construction

longueur	Construction	longueur du cycle
4	0101	15
5	11001	31
6	010101	63
7	1101010	127
8	11010101	255
9	110010101	511
10	0101010101	1023
11	11010101010	2047
12	010101010101	4095
13	1100101010100	8191

3.3.3 Représentation TSPC d'une règle d'Automate Cellulaire

Pour concevoir un générateur de vecteur de pseudo aléatoire, nous nous sommes inspirés des travaux de Soufi, Rochon, Savaria et Kaminska [14], dans lesquels sont présentés un automate cellulaire TSPC.

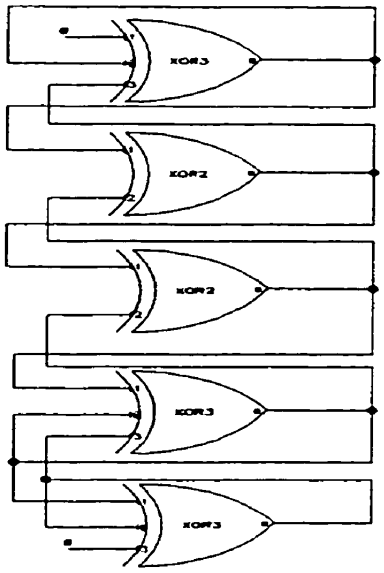


Figure 3.9: Schéma du générateur de vecteurs par automate cellulaire

Dans la figure 3.9, nous pouvons voir des XOR2 et des XOR3. Le XOR2 est semblable à celui utilisé dans l'additionneur complet, sauf que nous y avons ajouté un registre à décalage de deux bits en sortie, afin d'avoir la même latence qu'avec le XOR3. Le XOR3 est formé de deux XOR2 (voir figure 3.11), nous utilisons à nouveau des bascules pour que le signal C arrive au bon moment. Le XOR2 correspond à la règle 90 et le XOR3 correspond à la règle 150.

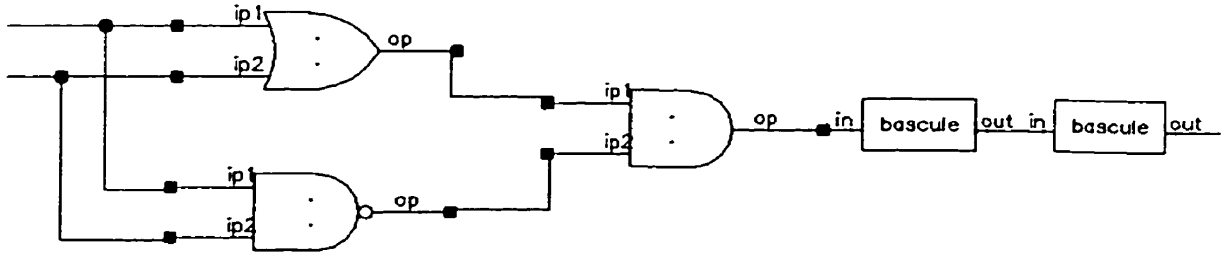


Figure 3.10: Schéma de la règle 90

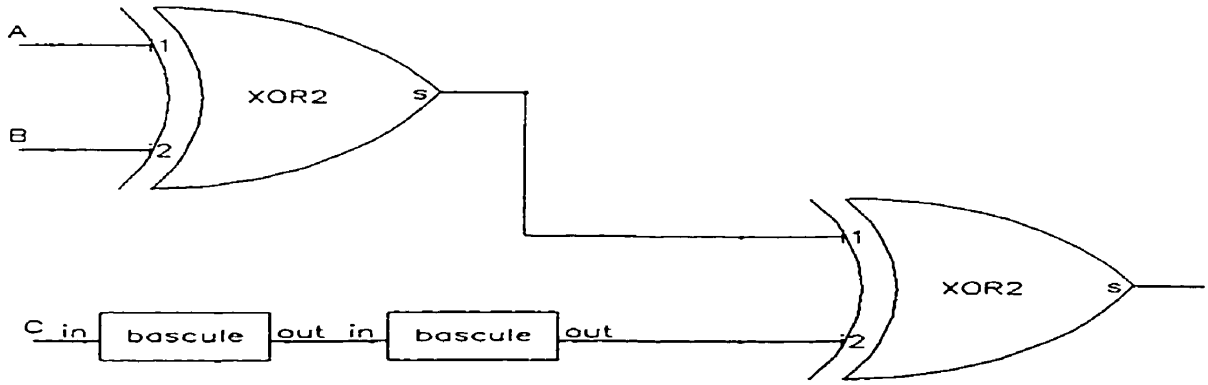


Figure 3.11: Schéma de la règle 150

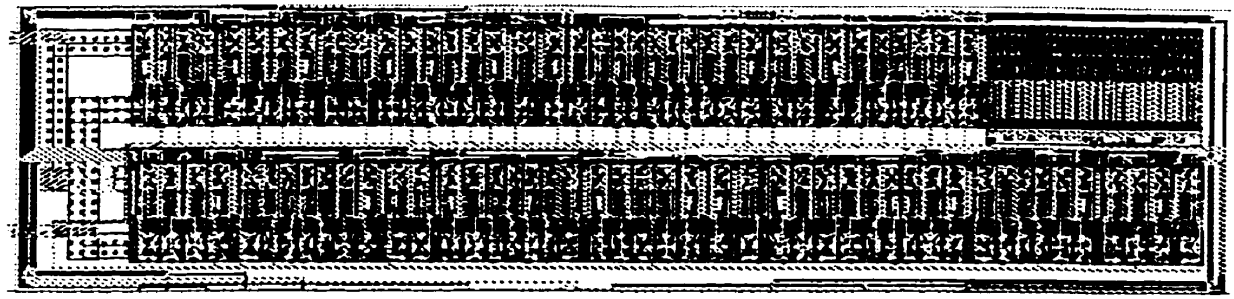


Figure 3.12: Masques d'un compteur pseudo-aléatoire de 10 bits

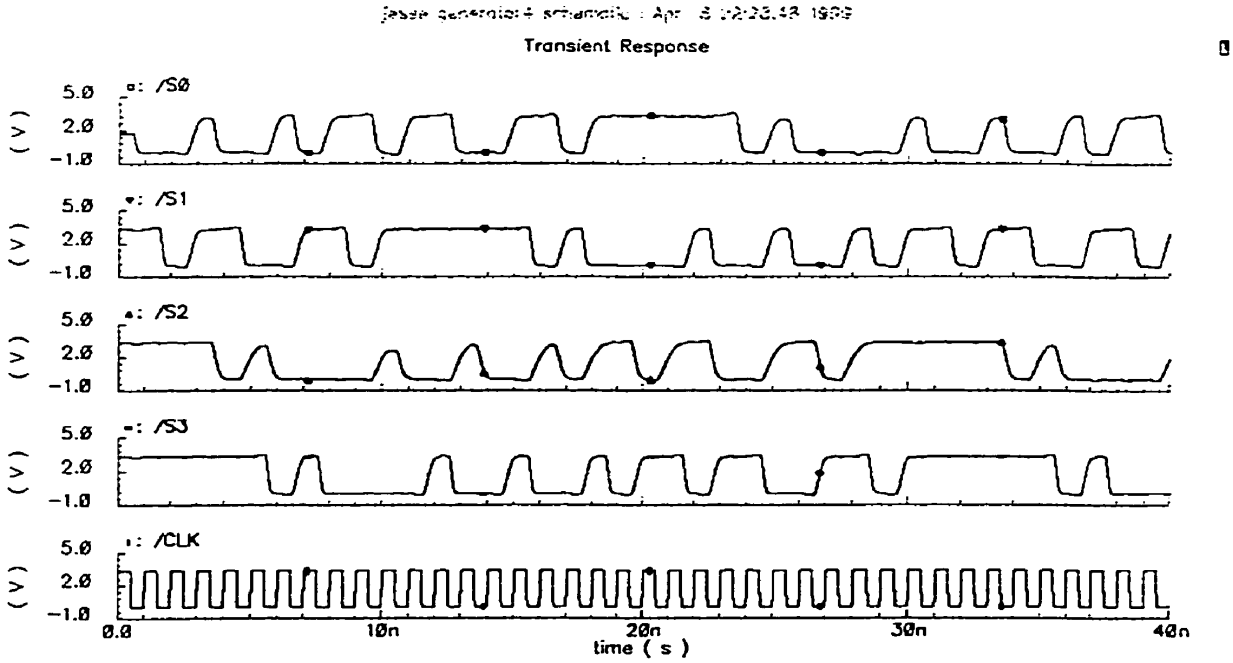


Figure 3.13: Simulation avec HSPICE du circuit extrait de la figure 3.12

3.4 Bloc "Analyse de signature"

3.4.1 Présentation du circuit Analyse de Signature

La compression de l'information est un aspect essentiel d'une machine autotestable. En effet, on ne peut pas envisager que la machine garde en mémoire l'ensemble des réponses normales; le résultat du processus de vérification devrait être binaire (bon, pas bon), avec une probabilité d'erreur raisonnablement faible.

L'analyse de signatures est une technique efficace qui permet de comprimer l'information avec une faible probabilité de masquage. Il serait passablement long de reproduire un grand nombre de fois la séquence pseudo-aléatoire en multiplexant tour à tour chacune des sorties à tester. Bien qu'on puisse insérer plusieurs registres d'analyse de signatures, il vaut mieux utiliser un registre d'analyse de signatures à entrées parallèles. Il semble cependant que le registre parallèle d'analyse de signatures soit davantage sujet au phénomène de masquage que la version sérielle (un bit à la fois). Avec une telle technique de compression de données, la machine autotestable se contente de mémoriser et de comparer un petit nombre de signatures pour que le processus de vérification soit complet. Nous pouvons remarquer une certaine similarité avec le compteur pseudo-aléatoire.

Dans notre circuit, le flot de donnée se déplace à une vitesse d'un GHz. Les plots d'entrée/sortie sont incapables de transmettre l'information à plus de 300 MHz, à moins d'utiliser des plots LVDS (Low Voltage Differential Signaling) [19] qui ne sont pas

disponibles à la CMC. Pour lire nos données, nous effectuons tout d'abord une compression de notre sortie venant d'un additionneur de 15 bits. Nous comprimons avec un registre d'analyse de signatures à entrées parallèles un mot de 15 bits en un mot d'un bit. Notre bit de sortie se charge toujours à 1 GHz, nous le démultiplexons sur 10 plots de sortie afin d'obtenir une information de sortie de 100 MHz. Nous pouvons retrouver la théorie dans [16]. La figure 3.15 montre les masques de notre circuit "Analyse de signature" et la figure 3.14, sa simulation.

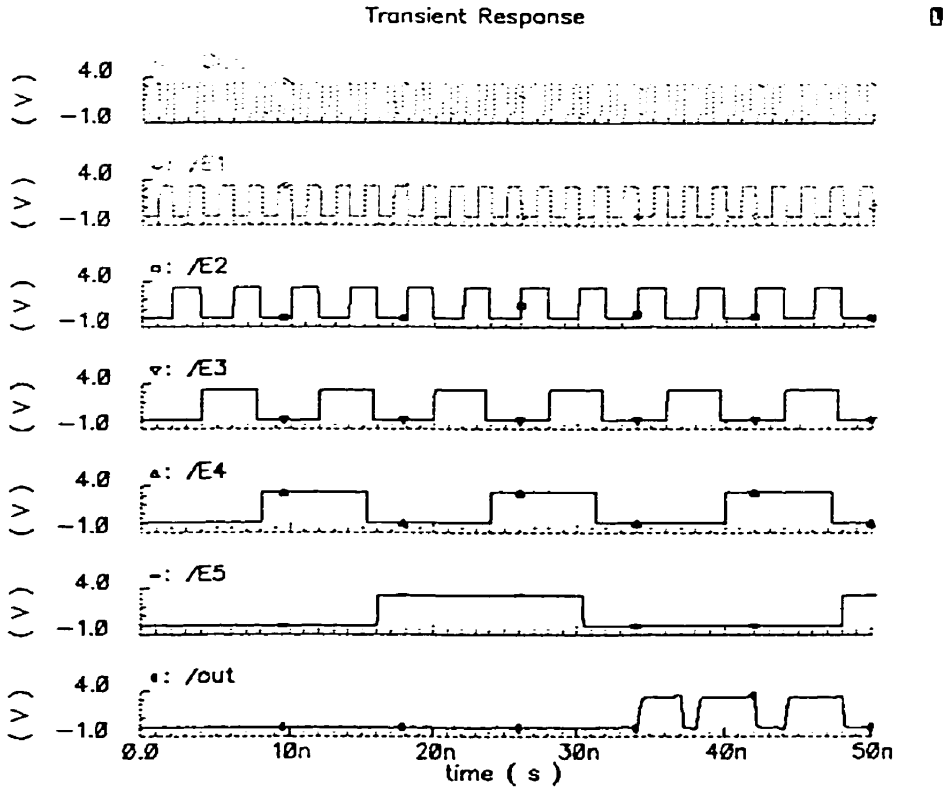


Figure 3.14: Simulataion du bloc analyse de signature

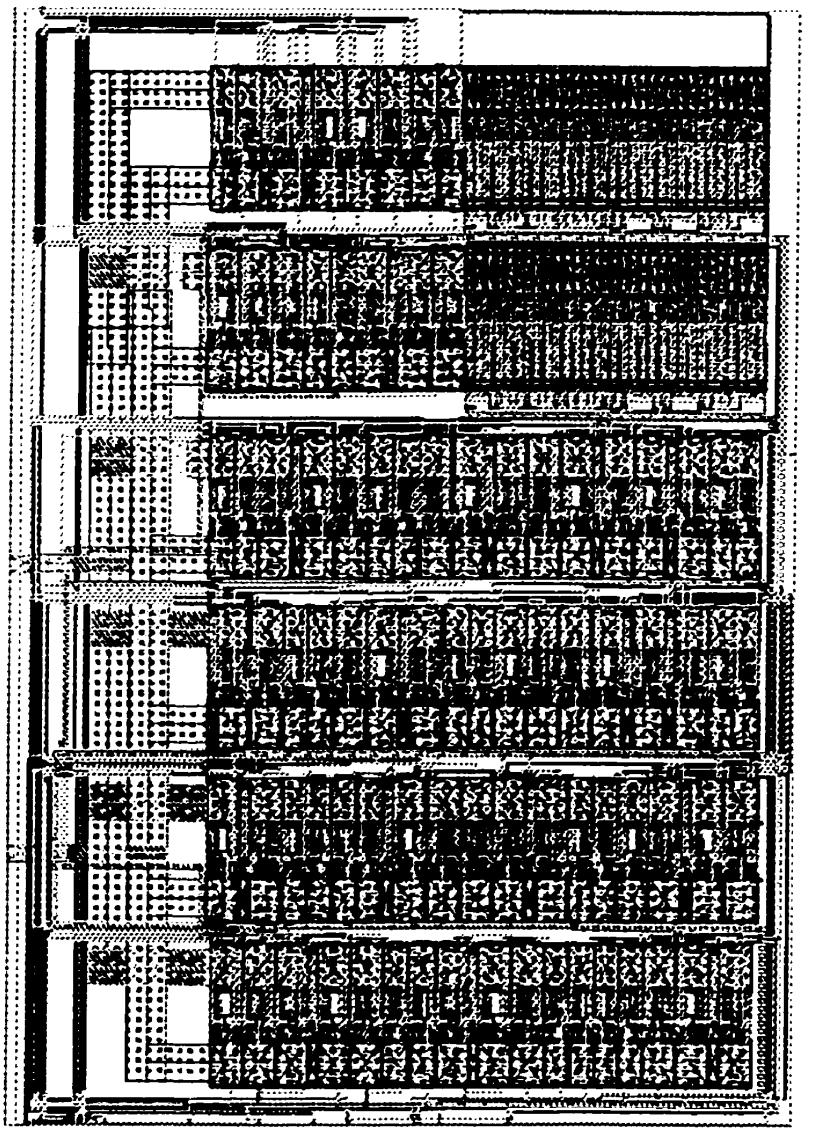


Figure 3.15: Schéma des masques du bloc analyse de signature

3.5 Démultiplexeur de haute performance

Le convolueur 3*3 est conçu pour propager des signaux à près d'un GHz. Nous souhaitons également récupérer les signaux de sortie afin de tester notre circuit. Les plots sont limités à une fréquence de 300 MHz. Pour ralentir nos signaux de sortie, nous utilisons une sortie parallèle à l'aide d'un démultiplexeur. Le démultiplexeur utilisé est un compteur en anneau simple (par exemple un compteur par 10) qui est préchargé avec la séquence 1000000000. Lorsque le signal de contrôle est enlevé, le bit à 1 se propage d'une porte à l'autre et alors un seul étage passe un signal à 1 à la fois. Ce signal peut donc être utilisé pour contrôler un signal à haute fréquence. Donc dans cet exemple, chaque sortie reçoit un signal à toutes les 10 nano-secondes. Les figures 3.16, 3.18, 3.17 présentent respectivement le démultiplexeur au niveau schématique, le dessin des masques et les simulations basées sur un circuit extrait correspondant.

La complexité du démultiplexeur par 10 est montrée dans le tableau 3.8:

Tableau 3.9 : Complexité du compteur 10

Composant	Nombre de composants	Nombre de transistors par composant	Nombre de transistor au total
And-n	9	7	63
buffer-p	10	5	50
inverseur	1	2	2
total	20		115

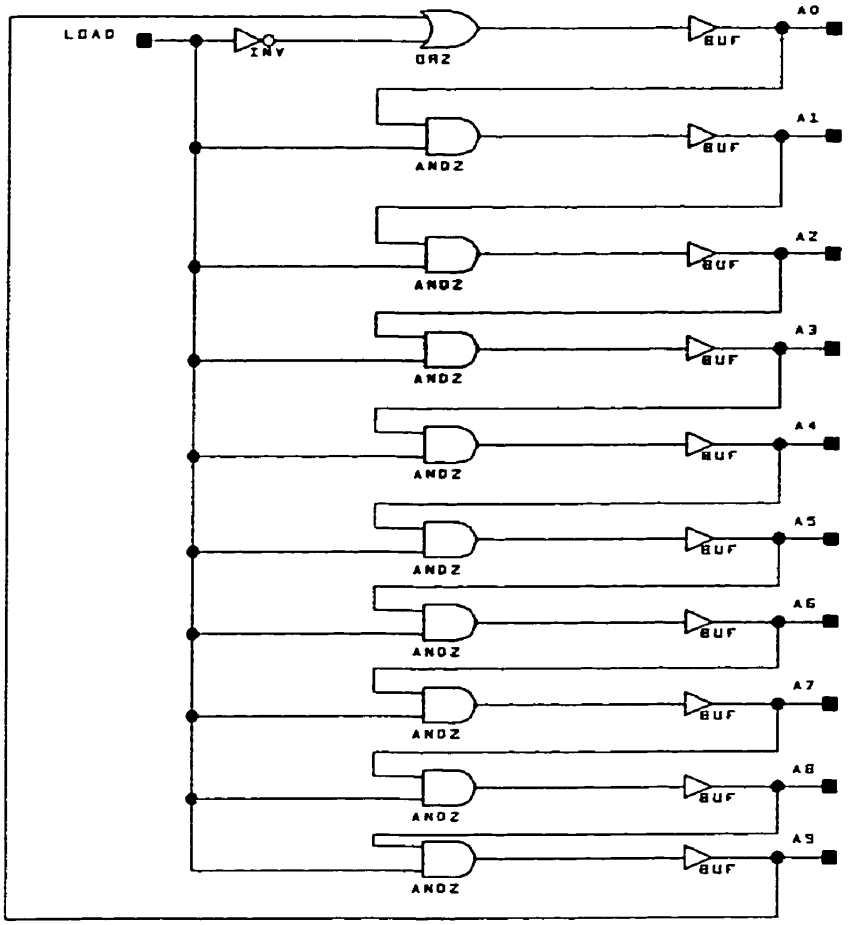


Figure 3.16: Exemple d'un compteur par 10 en anneau (réalisé en TSPC)

Les cellules de la figure 3.16 sont des cellules TSPC. Ce circuit n'est donc pas combinatoire. Une simulation des résultats produits par ce circuit est présentée à la figure 3.17.

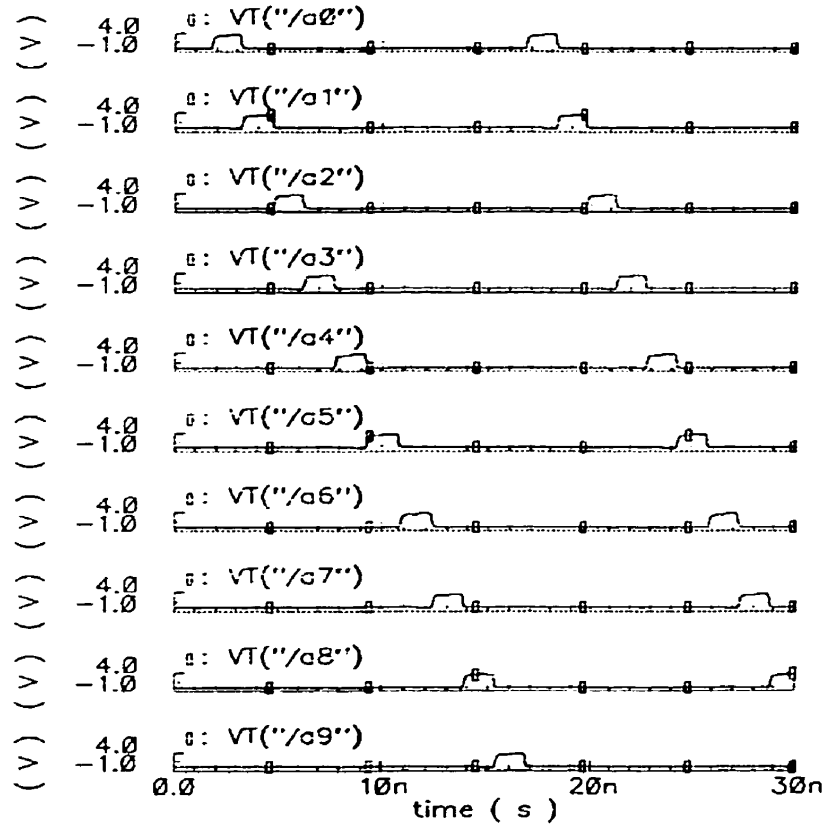


Figure 3.17: Simulation du compteur par 10 en anneau à partir d'un modèle extrait des masques

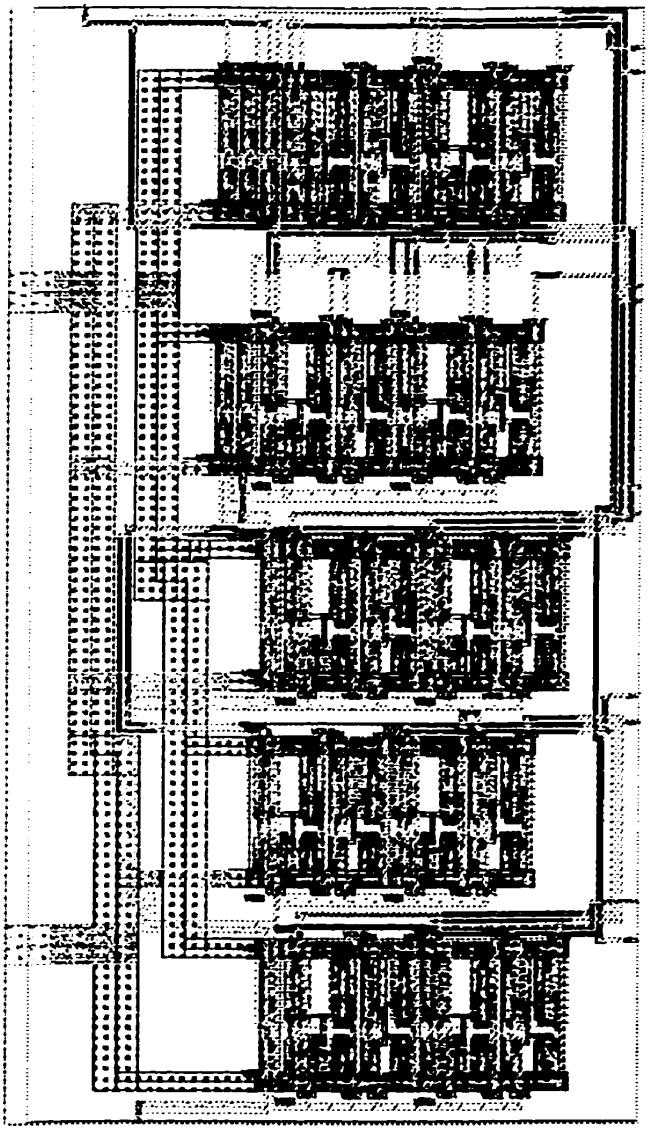


Figure 3.18: Schéma du compteur par 10 en anneau au niveau des masques

3.6 ROM

Le convolueur peut utiliser un filtre de convolution parmi 7 qui sont déjà en mémoire dans le circuit. Les 3 lignes de sélection permettent jusqu'à 8 filtres. La mémoire utilisée à cette fin est de type pseudo-nMOS. Le choix du filtre étant fixe pour toute la durée de la convolution, nous n'avons pas de contrainte de temps pour l'établissement des poids sur leur ligne de distribution. Donc, une configuration pseudo Nmos est suffisante, comme dans la figure 3.19.

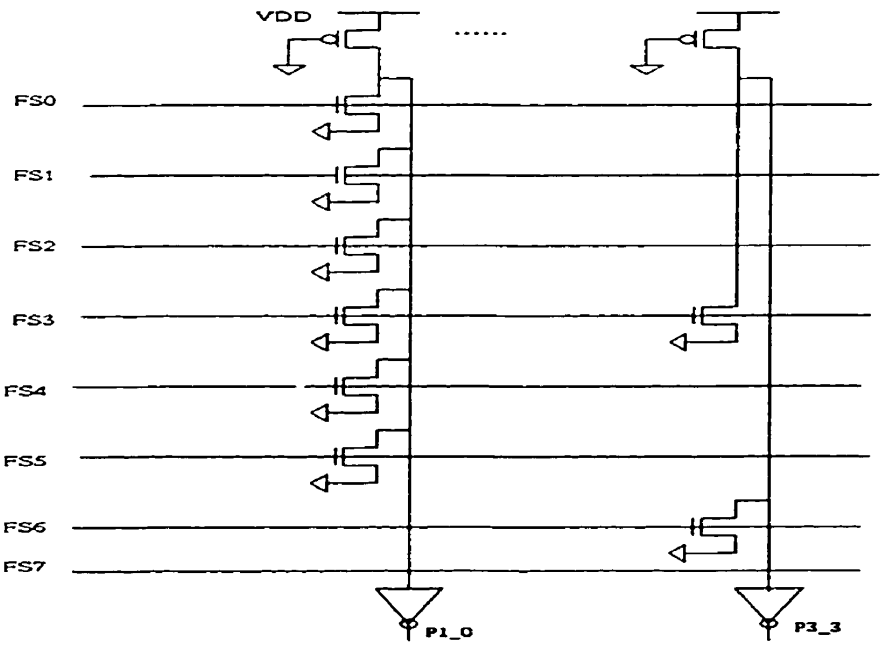


Figure 3.19: Un filtre de type pseudo NMOS

Pour chaque pixel de la fenêtre de convolution, 4 bits de mémoire sont nécessaires (3 bits pour le poids et 1 bit de signe). Comme 9 points sont définis dans la fenêtre de convolu-

tion, 36 bits sont utilisés par filtre. Comme 8 filtres sont possibles, le tout est multiplié par 8. Nous avons prévu le partitionnement de cette mémoire de façon à pouvoir placer au plus près de chacun des multiplieurs les signaux des poids de convolution. Nous avons finalement 3 blocs de mémoire; chacun de ces blocs alimente 3 multiplieurs. Les blocs de mémoire reçoivent 3 signaux de sélection qui sont démultiplexés par un décodeur 3_8 et à la sortie du pseudo nMOS, 36 bits (9*4 bits) sont nécessaires à la multiplication de 3 pixels. La figure 3.20 illustre plus en détail le design de cette mémoire.

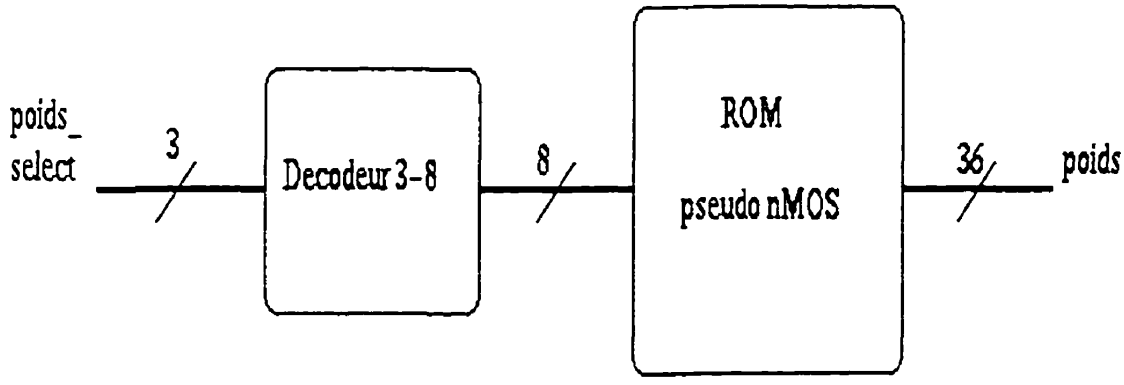


Figure 3.20: Décodeur 3-8

À la figure 3.21, nous retrouvons le dessin des masques du décodeur 3-8. Évidemment, cette façon de procéder crée de la redondance et augmente la taille du circuit. Cependant, avec la réduction de la taille minimale des technologies, cet aspect est de moins en moins important; dans notre cas, il s'agit de blocs plutôts petits, donc ils représentent un très faible pourcentage de la surface totale. L'avantage de cette façon de faire se situe surtout au niveau du routage des signaux. Les blocs de mémoire ne sont pas constitués de cellules

de type TSPC car, la valeur des poids du masque de convolution est fixe pour la durée de la convolution sur toute une image. Afin de caractériser la complexité de la ROM, on veut évaluer le nombre de transistors qui constituent ces trois blocs. Chaque bloc contient un décodeur 3-8 (8portes NOR à 6 transistors chacunes et 3 inverseurs à 2 transistors), soit 54 transistors. La partie ROM contient un nombre de transistors qui dépend de la valeur des filtres. Dans le tableau 3.10, nous avons la valeur des bits pour chaque filtre, pour chacun des 4 bits de chaque pixel de la fenêtre de convolution.À partir de ce tableau, on détermine que la partie pseudo nMOS contient un total de 59 transistors. Donc l'ensemble des filtres de convolution utilise 3*54 + 59 transistors, soit un total de 221 transistors.

Tableau 3.10 : Valeur des poids de convolution

	Fs0	Fs1	Fs2	Fs3	Fs4	Fs5	Fs6	Fs7
P1_3	0	0	1	0	1	0	0	0
P1_2	0	0	0	0	0	0	0	0
P1_1	0	0	0	0	0	0	0	0
P1_0	1	1	1	1	1	1	0	0
P2_3	0	0	0	0	0	0	0	0
P2_2	0	0	0	0	0	0	0	0
P2_1	0	1	0	1	0	0	0	0
P2_0	1	0	0	0	0	1	1	0
P3_3	0	0	0	0	0	0	0	0
P3_2	0	0	0	0	0	0	0	0
P3_1	0	0	0	0	0	0	0	0
P3_0	1	1	1	1	1	1	0	0
P4_3	0	0	1	0	1	0	0	0
P4_2	0	0	0	0	0	0	0	0

P4_1	0	1	1	0	0	0	0	0
P4_0	1	0	0	0	1	0	1	0
P5_3	0	0	0	0	0	0	1	0
P5_2	0	1	0	0	0	0	1	0
P5_1	0	0	0	0	0	0	0	0
P5_0	1	0	0	0	0	0	0	0
P6_3	0	0	0	0	0	0	0	0
P6_2	0	0	0	0	0	0	0	0
P6_1	0	1	1	0	0	0	0	0
P6_0	1	0	0	0	1	0	1	0
P7_3	0	0	1	1	1	1	0	0
P7_2	0	0	0	0	0	0	0	0
P7_1	0	0	0	0	0	0	0	0
P7_0	1	1	1	1	1	1	0	0
P8_3	0	0	0	1	0	1	0	0
P8_2	0	0	0	0	0	0	0	0
P8_1	0	1	0	1	0	0	0	0
P8_0	1	0	0	0	0	1	0	0
P9_3	0	0	0	1	0	1	0	0
P9_2	0	0	0	0	0	0	0	0
P9_1	0	0	0	0	0	0	0	0
P9_0	1	1	1	1	1	1	0	0

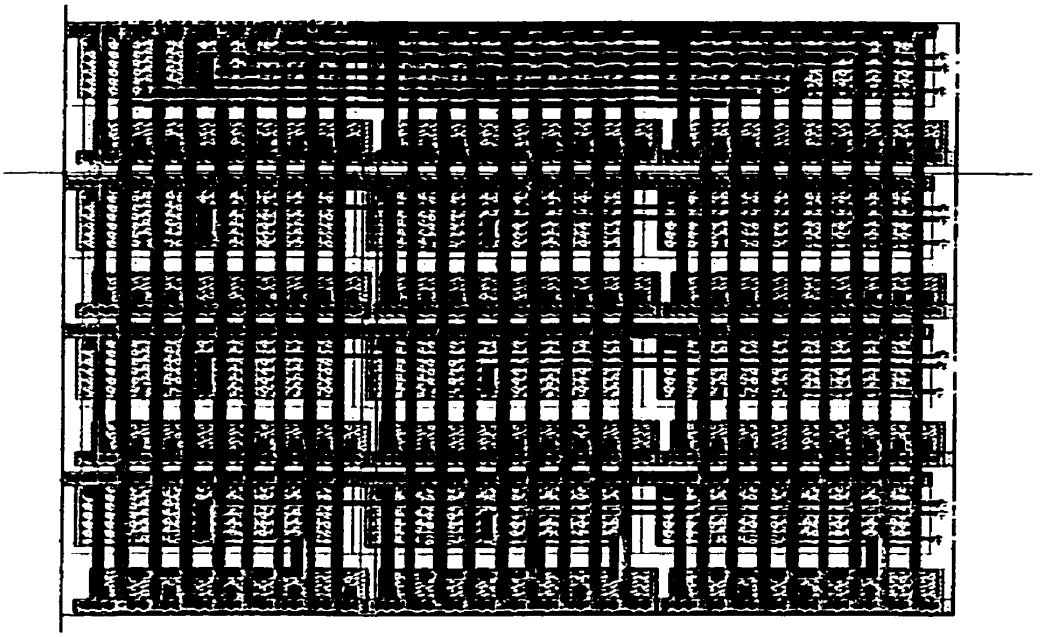


Figure 3.21: Schéma des masques du décodeur 3-8

3.7 Méthode de Distribution de l'Horloge

La méthode la plus utilisée pour distribuer uniformément l'horloge dans les circuits VLSI est la distribution par arbre d'horloge. Cependant cette méthode n'est pas optimale dans les circuits de haute performance. En effet, elle demande un très bon contrôle de la répartition des charges au niveau des feuilles de l'arbre. Il faut que chaque feuille ait une charge semblable et que le chemin physique et électrique entre le centre de l'arbre et toutes les feuilles soit semblable. Évidemment, dans les hautes fréquences, tous ces paramètres deviennent critiques, ils sont très sensibles à de mauvaises répartitions. Il est donc difficile de mettre en oeuvre un arbre d'horloge pour les hautes fréquences.

Pour palier à ces problèmes deux stratégies sont couramment utilisées. La première concerne les DSP. L'architecture des DSP est fortement parallèle. Nous avons par exemple plusieurs lignes horizontales de même dimension. Il faut s'assurer que ces lignes soient régulières donc de même dimension et pas trop longues, selon le biais de synchronisation qui est permis. Sur la ligne verticale, d'où partent toutes les lignes horizontales, nous appliquons l'horloge. Ainsi chaque ligne horizontale a le même biais. Cette solution est intéressante pour de petits circuits fortement parallèles.

La deuxième stratégie couramment utilisée fait appel à une grille d'horloge, comme présentée à la figure 3.22. Cette stratégie a été adoptée dans de gros processeurs comme le processeur ALPHA et elle permet un très petit biais et des fréquences très élevées: près de 800 MHz. La grille d'horloge est répartie sur une seule couche de métal. Ensuite, l'horloge est répartie de facon régulière sur la grille, de façon centrique ou de gauche à droite avec de gros amplificateurs. Cette grille correspond à un gros condensateur. Il faut beaucoup de puissance pour commander une telle charge et donc un très gros amplificateur de commande de grosses charges capacitives. Les amplificateurs de distribution doivent fonctionner à de très hautes fréquences et ils sont très nombreux. Cette solution est efficace mais elle engendre un gros problème de dissipation de puissance.

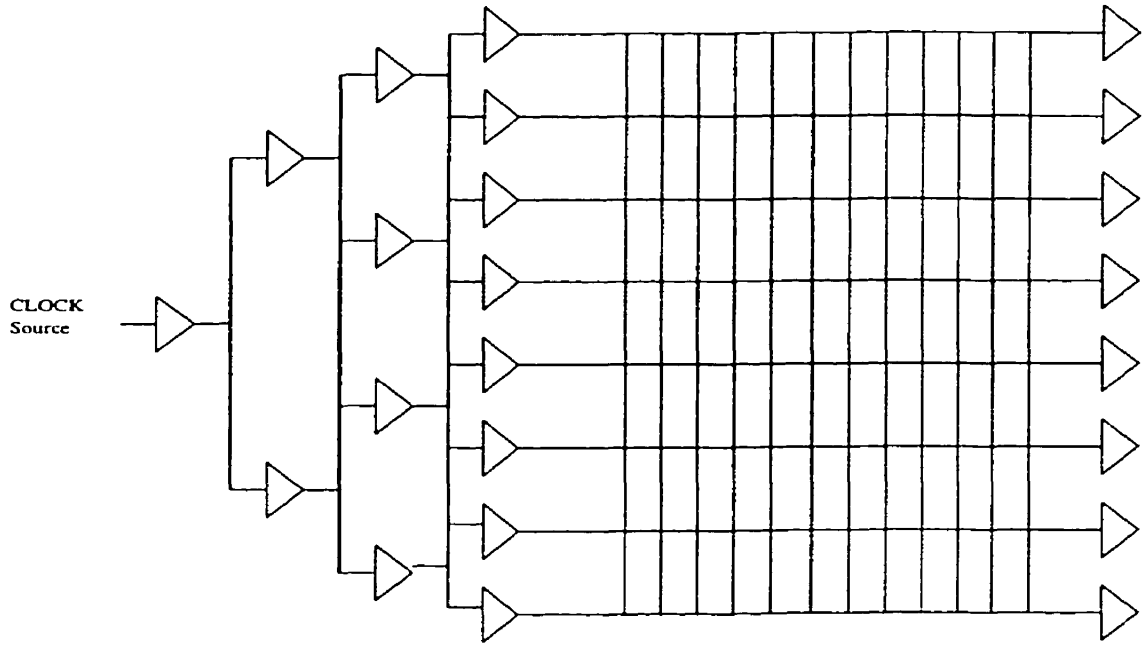


Figure 3.22: Réseau de distribution d'horloge du DEC Alpha

Nous avons adopté la solution de la grille d'horloge pour le convolueur car cette solution est facile à mettre en oeuvre et nous offre une plus grande flexibilté au niveau du placement bien que notre architecture soit fortement parallèle et permette également la solution adoptée par les DSP.

Nous n'avons pas réalisé l'amplificateur final. Chaque bloc contient son propre amplificateur selon un inverseur pour 3 cellules. Nous avons réparti régulièrement dans nos blocs logiques, des blocs amplificateurs de 15 inverseurs. Dans ce bloc amplificateur, un inverseur commande deux inverseurs qui en commande quatre inverseurs puis huit. Nous avons utilisé ces blocs pour faciliter le placement et la distribution de l'horloge. Tous les blocs amplificateurs sont reliés entre eux de façon à répartir la charge de façon globale.

La grille est connectée sur le premier inverseur de chaque bloc. L'amplificateur final commande la grille.

3.8 Méthode de Placement et Routage

Plusieurs méthodes ont été adoptées pour réaliser le convolueur 3*3 TSPC. Ces méthodes peuvent être également utilisées dans tous les circuits de haute performance. Il faut notamment analyser deux problèmes importants: le placement puis le routage. La première étape a été de bâtir une architecture réalisable à haute fréquence valide pour la TSPC. Il faut une architecture de type flot de donnée telle que chaque module communique avec le module suivant ou un module très proche. Le mieux est évidement d'avoir des modules en cascade. Ensuite, il faut limiter la complexité des circuits de commande car ils peuvent ralentir le flot de données. Les registres pipelinés peuvent induire des fréquences moins élevées à cause du contrôle nécessaire pour les gérer, même si le pipeline fait en principe gagner du temps. Dans la haute fréquence, nous utilisons des architectures fortement pipelinées et il est judicieux d'éviter d'utiliser des blocs de contrôle comme les machines à états. Notre première étape a donc été de concevoir cette architecure de haute performance, puis de la placer quitte à modifier notre architecure si elle s'avérait difficile à placer. La figure 3.23 illustre le placement que nous avons adopté.

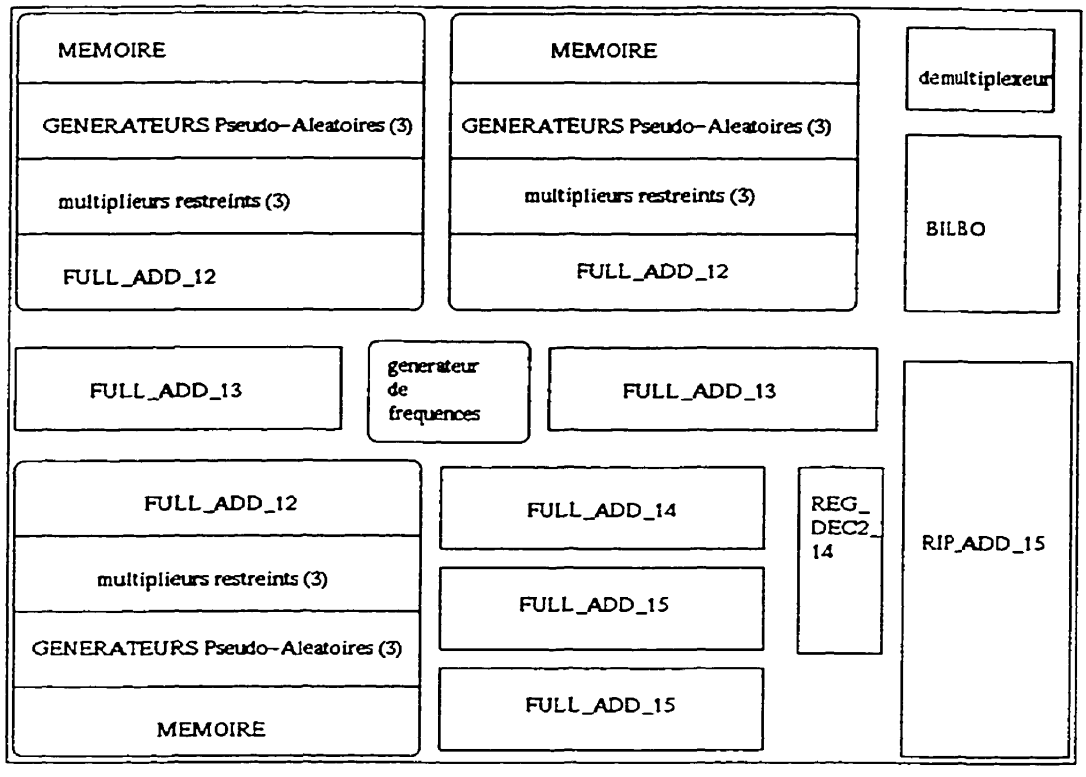


Figure 3.23: Plan de masse des différents blocs du convolueur 3*3 TSPC

Nous pouvons remarquer une certaine régularité dans notre placement. Nous avons tout d'abord 3 gros modules semblables au niveau de l'entrée, un exemple du dessin des masques d'un de ces modules est proposé à la figure 3.24. Chacun de ces modules comporte tout d'abord un générateur de vecteurs pseudo-aléatoires et une ROM contenant les coefficients des filtres avec leurs bit de signe, il s'agit des signaux d'entrée de notre convolueur. Les signaux d'entrée sont alors envoyés aux multiplieurs restreints. La mémoire est statique ainsi que ses signaux, donc il est possible de la disposer partout dans le circuit. Cependant, pour éviter des problèmes de routage et de couplage électro-magnétique, nous

avons préféré disposer la mémoire dans chacun des modules. Les signaux d'entrée sont alors envoyés aux multiplieurs, chaque module a trois multiplieurs. Les signaux de sortie sont ensuite envoyés aux additionneurs complets-12 qui commencent l'arbre de sommation. Les additionneurs complets-12 qui sont assez proches communiquent alors avec les additionneurs complets-13, puis de façon sérielle vers le Rip-Add-15, qui communique avec le bloc "Analyse de signature". Nos signaux sortent par le biais du module Démultiplexeur. Les 3 modules font converger l'information de façon parallèle vers les additionneurs complets-13 qui communiquent ensuite de façon sérielle avec les autre modules. Nous pouvons remarquer que le coeur du circuit est occupé par notre bloc oscillateur. Étant donné les fréquences utilisées, nous sommes obligés d'avoir une horloge interne. La meilleure façon de la distribuer est alors de la mettre au centre du circuit. L'inconvénient de cette technique est que nous ne pouvons pas effectuer de routage à travers le bloc de distribution d'horloge. En effet, nous ne disposons dans notre technologie que de 3 couches de métal et pour éviter des interférences électromagnétiques, nous avons choisi d'éviter de passer au travers de ce bloc.

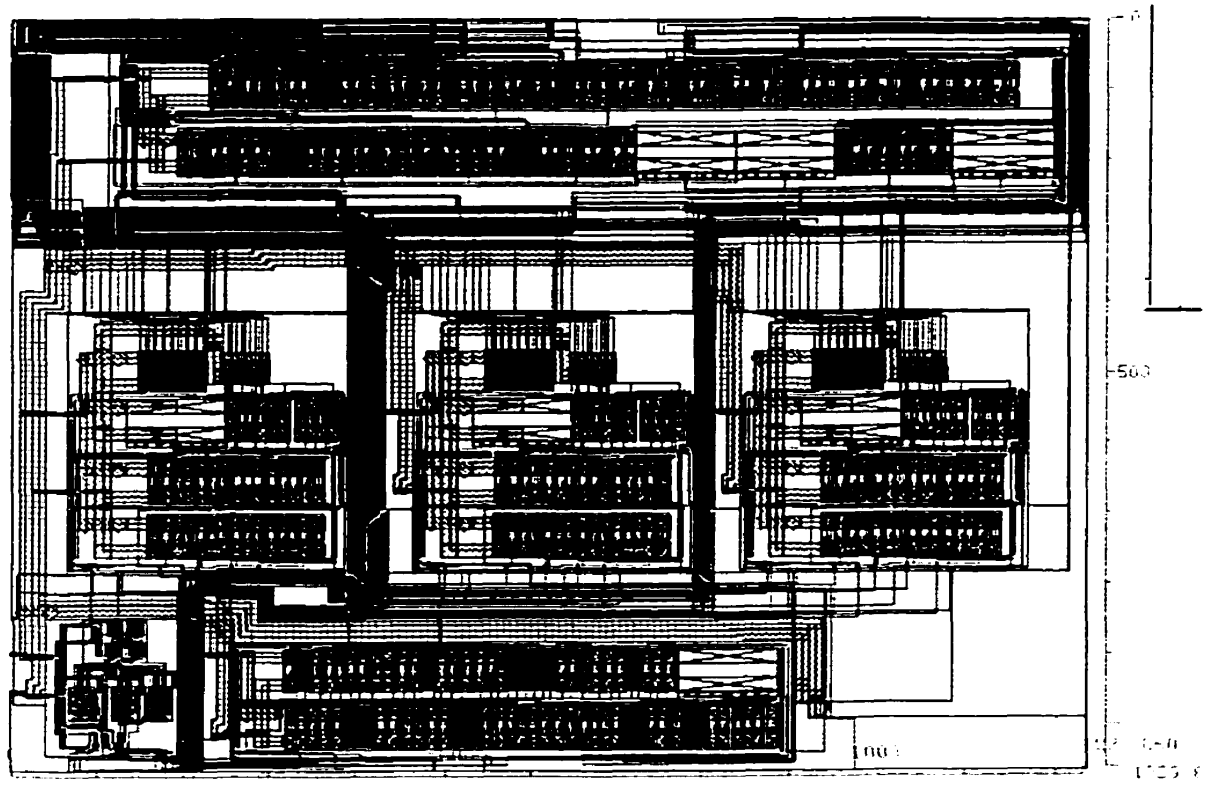


Figure 3.24: Placement et routage d'un des 3 modules du convolueur

Nous pouvons remarquer que le placement et le routage sont fortement liés. En effet, les simulations ont montré qu'il était interdit d'utiliser des interconnexions de plus de 200 µm, si nous voulions éviter les courses entre les signaux. Cette contrainte nous a évidemment obligé à avoir des modules très proches. Nos trois gros modules occupent un demi-cercle dont le centre est l'additionneur complet-13. Notre placement est donc centrique puis sériel. Pour atteindre notre objectif, nous avons utilisé plusieurs passes comportant des étapes de: placement, routage puis distribution d'horloge, jusqu'à ce que nos objectifs soient rencontrés.

3.8.1 Outils Utilisés

La meilleure façon d'obtenir des circuits de haute performance est évidemment de faire un circuit dédié. Cependant, la réalisation de gros circuits rappelle vite les limites de tels circuits. Nous avons donc adopté une approche mixte. La première étape a été de réaliser une bibliothèque de cellules normalisées. Ces cellules sont dédiées à l'architecture du convolueur 3*3 TSPC. Ensuite, nous avons étudié la puissance des outils de placement et routage. Si ces outils sont correctement paramétrisés et guidés, ils peuvent réaliser de très beaux placements et routages. Notre stratégie a été de réaliser de petits blocs simulables au niveau schéma des masques puis de les assembler. La plupart des blocs peuvent être générés automatiquement à partir des cellules normalisées. En fait, seul le bloc oscillateur et la ROM ont été dédiés. Les fonctions de *mapping* permettent ensuite de disposer ces blocs où nous le souhaitons et de forcer le routage selon nos contraintes. Nous avons également utilisé deux couches de métal pour notre routage, car la troisième couche est réservée à l'horloge.

3.8.2 Dissipation de puissance

Les simulations des premiers blocs ont donné des résultats inquiétants au niveau de la dissipation de puissance. Une cellule additionneur complet utilise un courant de près de 5 mA à 1 GHz et nous avons un grand nombre de ces cellules. Notre technologie recommande en moyenne selon les couches de métal utilisées d'allouer au moins une largeur de 1 μm pour 1 mA. Donc pour un additionneur complet, il faut un fil d'alimentation de 5

µm. Évidemment notre placement nous impose d'avoir des lignes d'au plus 200 µm. Il faut quatre additionneurs complets pour obtenir une ligne de 200 µm, donc notre fil d'alimentation doit faire au moins 20 µm de large. La bibliothèque de cellules normalisées a alors été modifiée afin de faciliter le travail de placement et routage. Les gros fils d'alimentation sont par contre dessinés manuellement. Le fil d'horloge a une largeur de 1.5 µm également suivant les mêmes règles.

Nous utilisons un seul canal par cellule. Ce canal est le même pour toutes les cellules de la même ligne, afin de répartir les charges capacitives de façon globale et d'éviter les biais de synchronisation. Nous avons utilisé le canal en bas des cellules pour propager l'horloge. Les canaux du haut sont dédiés au routage entre les cellules. Il aurait été possible de forcer un canal entre les transistors NMOS et PMOS afin de faciliter le routage. Dans notre cas, ce n'était pas une contrainte. À titre d'exemple, nous avons disposé cinq additionneurs complets en série et simulé leur dissipation de puissance. Les cinq additionneurs avec leur arbre d'horloge utilisent 26 mA, soit une dissipation de puissance de 85 mW. L'arbre d'horloge formé de 31 inverseurs utilise 9.4 mA, soit une dissipation de 31 mW. Les résultats ont été donné par HSPICE après simulation. Un additionneur complet représente 73 transistors et nous avons plus de 18000 transistors. Une première approximation donnerait 4.19 Watt. Il faut également ajouter l'amplificateur d'horloge ainsi que la grille d'horloge. Cette dissipation est trop importante pour un circuit de cette taille. Le processeur "GuTs" d'IBM dissipe 8.5 W, mais il est de bien plus grande complexité.

3.9 Conclusion

Dans ce chapitre nous avons présenté notre stratégie de conception d'un circuit de haute performance incluant un certain nombre de circuits de test adaptés. Notre bloc oscillateur est facile à mettre en oeuvre quelque soit la technologie et il est très robuste. Évidemment, nous aurions pu utiliser un VCO à partir d'amplificateurs opérationnels mais ce genre de circuit est plus complexe et il ne peut atteindre des fréquences aussi élevées. Le générateur de vecteurs pseudo-aléatoires et le bloc d'analyse de signature sont très intéressants car ils sont faciles à mettre en place et permettent des fréquences très élevées. Le démultiplexeur est une solution naturelle pour communiquer avec l'extérieur à des fréquences moindre. La stratégie adoptée lors de notre conception de circuit de haute performance peut-être adoptée avec n'importe quel circuit et elle est rapidement appliquable.

CONCLUSION

Dans ce mémoire, nous avons proposé une approche permettant de réaliser des circuits de haute performance. Nous avons analysé toutes les étapes menant à la conception d'un circuit rapide, de l'architecture au niveau schéma des masques. Les circuits ultra-pipelinés sont vraisemblablement le meilleur axe de recherche après le développement de nouvelles technologies pour améliorer les performances des microprocesseurs. Ils constituent la suite logique des architectures RISC. La technique TSPC est la plus récente des techniques de conception de circuits dynamiques et elle constitue une solution prometteuse pour les années à venir.

Le chapitre 1 a permis de développer une bibliothèque de cellules TSPC de haute performance. À partir des travaux de recherche précédents sur les bascules, nous avons extrapolé par analogie un certain nombre de règles de conception propres aux cellules TSPC. Il nous est ainsi possible de disposer d'une bibliothèque de cellules TSPC de qualité en moins d'un mois. Ces recherches ont également mis en évidence la difficulté de modéliser de telles cellules dont chaque transistor a une influence sur l'autre. Nous nous sommes surtout concentrés sur les cellules TSPC à sortie partagée, fruit des travaux de recherche précédents. Ces travaux s'étaient surtout concentrés sur les bascules. Une bibliothèque

complète et la simulation de gros blocs logiques au niveau schéma des masques ont mis en évidence la sensibilité de ces cellules au problème du partage de charges propres aux cellules dynamiques. Nous savons dorénavant que les cellules TSPC ne sont pas forcément les plus adéquates pour réaliser des circuits de hautes performances à cause de leur forte dissipation de puissance. Cependant ces cellules sont très intéressantes pour gérer la distribution d'horloge, car elles n'utilisent qu'un seul transistor d'horloge contrairement aux autres techniques et permettent des fréquences très élevées. Ces cellules restent très intéressantes au niveau des bascules pour la haute performance. Pour éviter des problèmes de partage de charges, il suffirait de rajouter un transistor de part et d'autre du transistor d'horloge qui serait en série avec les réseaux logiques. Cette solution est intéressante mais ralentirait évidemment nos cellules. Notre bibliothèque de cellule TSPC à sortie partagée pourrait également être un choix judicieux pour réduire la consommation dans un processeur comme le DEC ALPHA.

Dans le chapitre 2, nous présentons une architecture TSPC fortement pipelinée d'un convolueur 3*3. Le convolueur 3*3 est une application très utilisée dans les traitements d'image. Il serait donc intéressant de posséder un tel co-processeur de haute performance. Notre architecture montre les avantages et les inconvénients des circuits à granularité très fine. Ils permettent évidemment des fréquences d'horloge très élevée. Cependant, ils procurent une latence également importante. Ces circuits sont donc très intéressants pour des applications demandant un effort de calcul prolongé. Le traitement des images est donc une application parfaite pour ce type de circuit, de plus leur demande est très grande et ne cesse de croître.

Le chapitre 3 montre des blocs de test de haute performance et des solutions de conception pour la haute performance. Nous avons développé des circuits de test très importants, comme le diviseur de fréquence, le générateur de vecteurs pseudo-aléatoires et l'analyse de signature. Le démultiplexeur de haute performance est une solution élégante pour propager les informations de circuits rapides sur des plots standards. Des plots LVDS permettraient de faciliter la tâche et de propager des données vers l'extérieur sans avoir à utiliser un multiplexeur. Notre méthode de distribution peut être utilisée par tous les circuits rapides avec un biais de synchronisation raisonnable. Dans le même ordre d'idée, la stratégie de conception adoptée dans le bloc oscillateur est très facile à transposer. Les outils de placement et routage ont démontré leur potentiel pour développer des circuits de haute performance pourvu qu'ils soient correctement manipulés.

BIBLIOGRAPHIE

[1] H.Peter Hofstee, Sang H.Dhong, David Meltzer, Kevin J. Nowka, Joel A. Silberman, Jeffrey L. Burns, Stephen D. Posluszny, Osamu Takahashi, IBM Austin Research Laboratory (1998), Designing for a Gigahertz, IEEE International Solid-State Circuits Conference, 66-73.

[2] Nelson F. Goncalves, Hugo J. De Man (1983), NORA: A racefree Dynamic CMOS technique for pipelined logic structures,IEEE Journal of solid-state circuits, 18, 261-266.

[3] R. H. Krambeck, C. M. Lee, and H. S. Law (1982), High-speed compact circuits with CMOS, IEEE J. Solide-State Circuits, 17, 614-619.

[4] Jiren Yuan and Christer Svensson (1989), High-Speed CMOS circuit technique, IEEE J. Solide-State Circuits, 24, 62-69.

[5] Per Larsson-Edefors (1996), Technology mapping onto very high speed standard CMOS hardware, IEEE Transactions on computer-aided design of integrated circuits and systems, 15, NO.9, 1137-1144.

[6] C. Farnsworth, D.A. Edwards ans S.S. Sikand (1997), Utilising dynamic logic for low power consumption in asynchronous circuits, Université d'Oxford en Angleterre (Department of Computer Science, The University, Oxford Road, Manchester, M13 9PL, UK).

[7] Tom Burd (1994), Low-Power CMOS Library Design Methodology, mémoire de maîtrise de l'université de Berkeley.

[8] B. Bosi (1995), Méthodes de conception de convolueur dédiés et reconfigurables, mémoire de maîtrise de l'École Polytechnique de Montréal.

[9] S. Ghannoum, D. Chtchvyrkov et Y. Savaria (1993), A comparative study of single-phase clocked latches using estimation criteria, IEEE International Symposium on Circuits and Systems, 4, 347-350.

[10] B. Antaki, S. Patenaude, L. Trognon et Y. Savaria (1997), A study on split-output TSPC CMOS circuits, ISCAS 97, 4, 1600-1604.

[11] TSMC 0.35-micron CMOS design and fabrication service that CMC is now offering through PMC-Sierra.

[12] Canadian Microelectronics Corporation Mitel15 Design Kit V3.1 for Cadence Analog Artist.

[13] F. LU, H. Samueli, J. Yuan et C. Svensson (1993), A 700 MHz 24-b pipelined Accumulator in 1.2 μm CMOS for application as a Numerically Controlled Oscillator, IEEE Jounal State Circuits, 28, 878-886.

[14] M. Soufi, S. Rochon, Y. Savaria et B. Kaminska (1996), Design and Performance of CMOS TSPC Cells for High Speed Pseudo Random Testing, 14^{th} VLSI Test symposim, 14, 368-373.

[15] P. D. Hortensius, R. D. McLeod et H. C. Card (1989), Parallel Random Number for VLSI Systems Using Cellular Automata, IEEE Transactions on computers, 38, 1466-1471.

[16] P. Bardell, W. McAnney et J. Savir (1996), Built in Test for VLSI, 334 pages.

[17] D. Ballard, C. Brown, (1987), Computer Vision, 517 pages.

[18] B. Konemann, J. Mucha and G. Zwiehoff, Built-in logic block observation techniques (1979), The Proceedings of the International Test Conference, 37-41.

[19] LVDS Owner's Manual (1997), National Semiconductor, Design Guide.

ANNEXE 1

Librairie TSPC

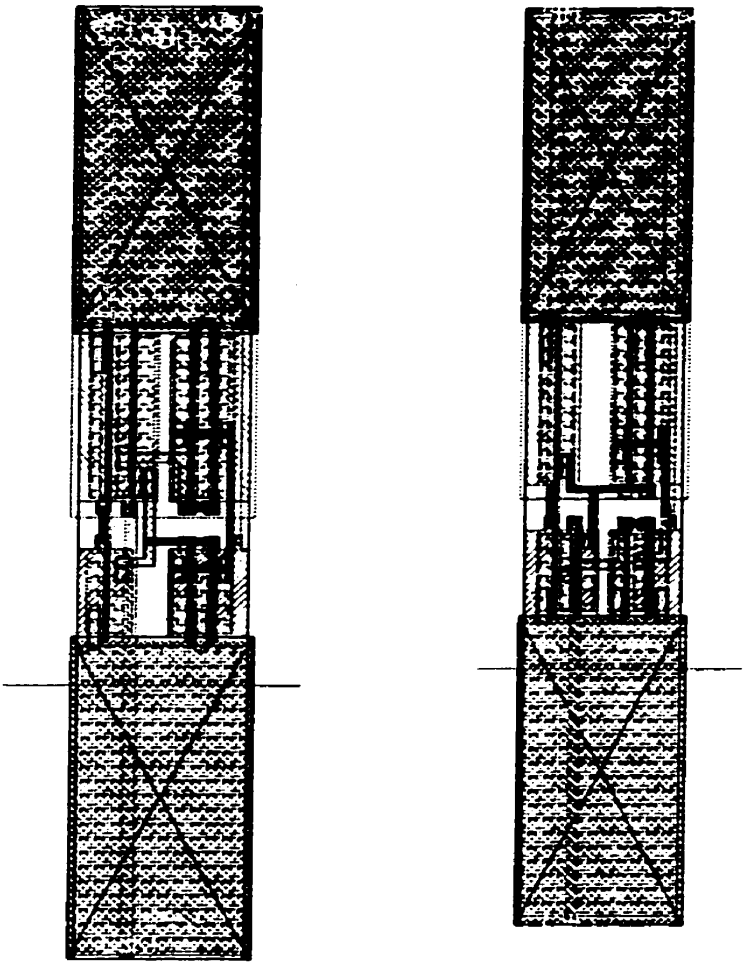
Librairie TSPC

Les cellules de base de type TSPC sont au nombre de 6. A celles-ci se rajoute un inverseur de taille semblable aux cellules de base TSPC. Nous allons retrouver ci après les vues layout de toutes ces cellules ainsi que les résultats de simulation sur ces cellules. Ces simulations on t été effectuées à une fréquence de 1,5 Ghz.

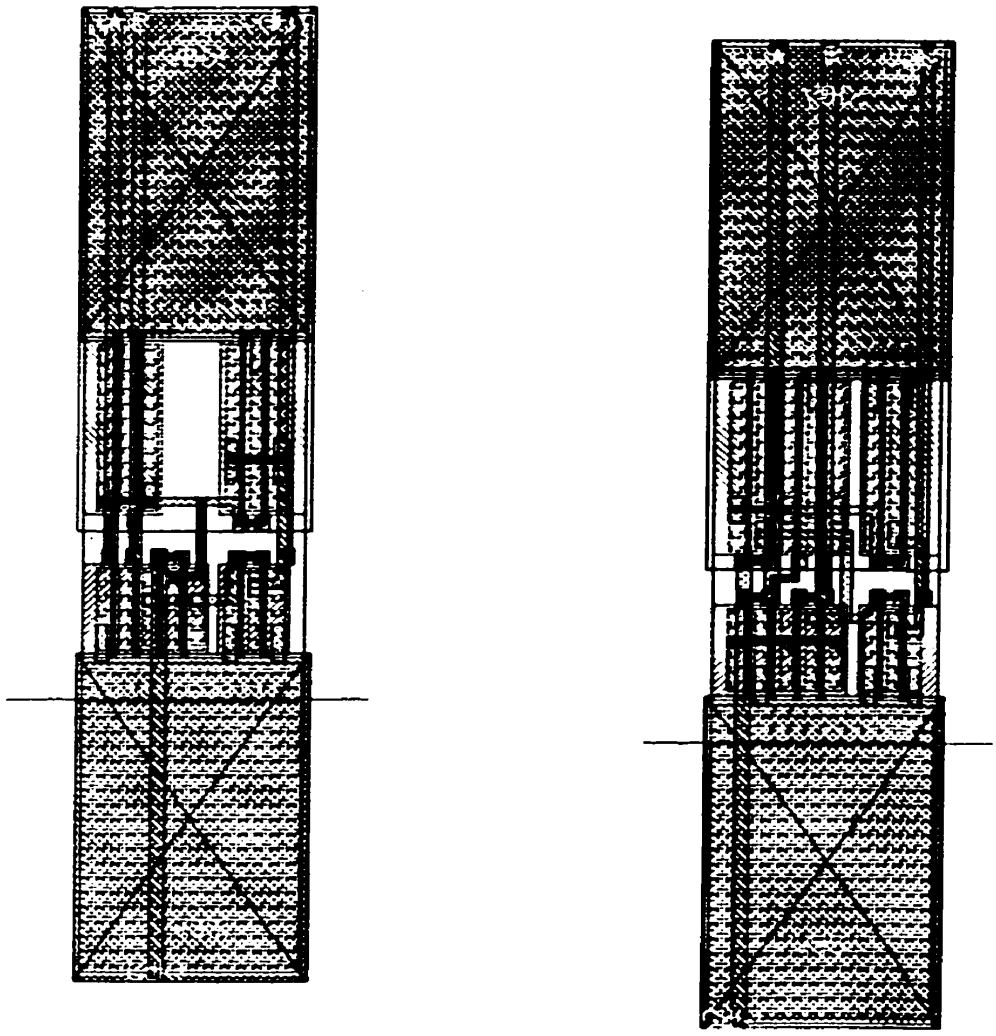
Pour illustrer également les autres vues constituant la librairie de même que les résultats obtenus pour le LVS, nous utiliserons la cellule buffern.

Les vues schématiques onnt été présentées à la section 2 du présent rapport.

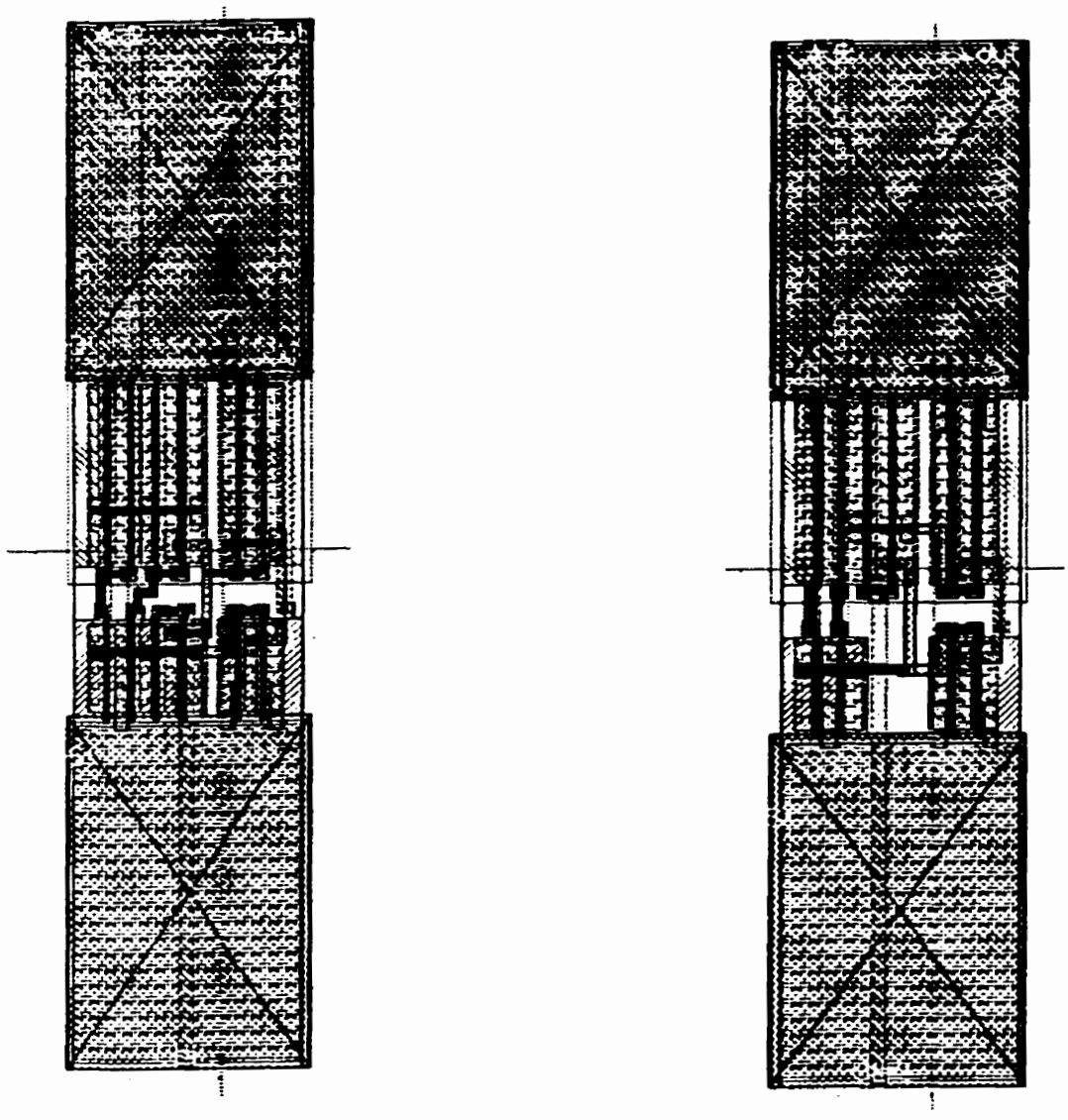
VUE LAYOUT



Buffer N et P



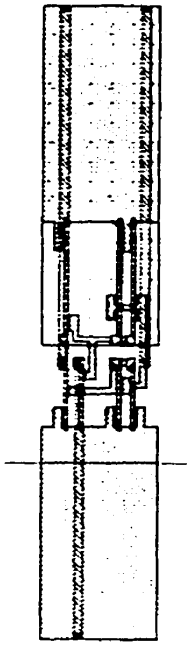
And N et P



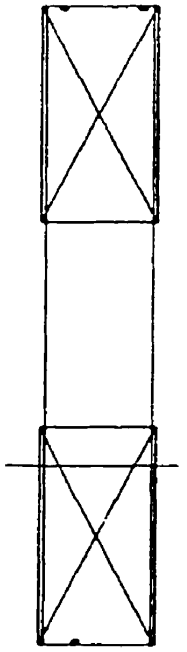
Or N et P

VUE EXTRACTED

(buffern)

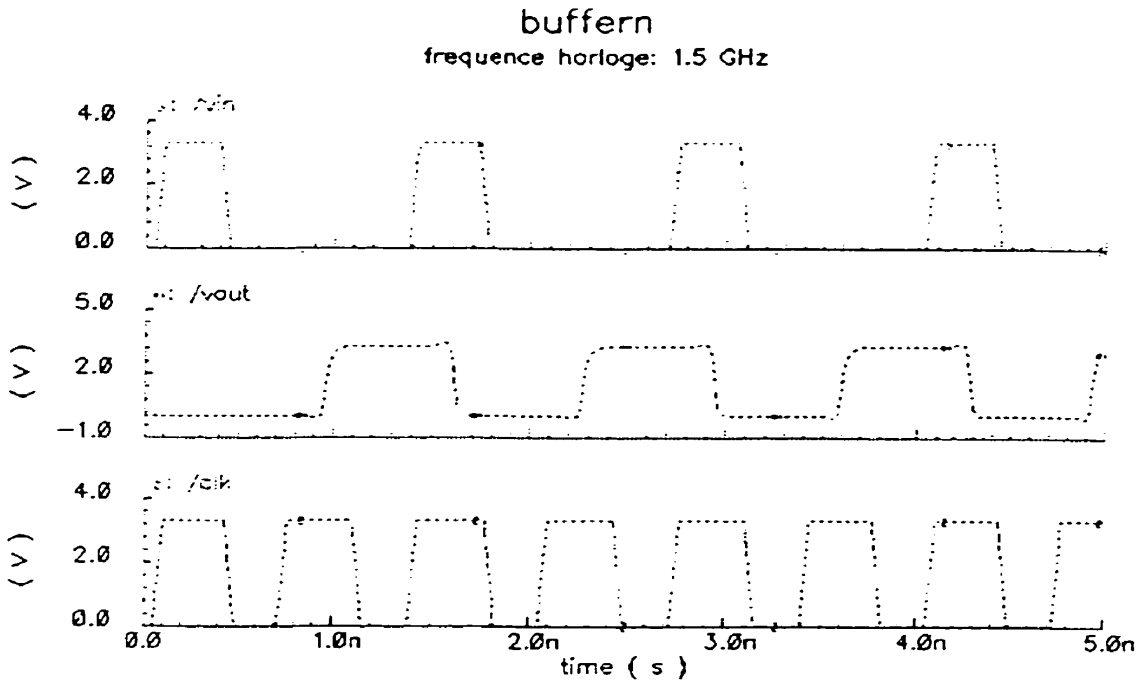


VUE ABSTRACT

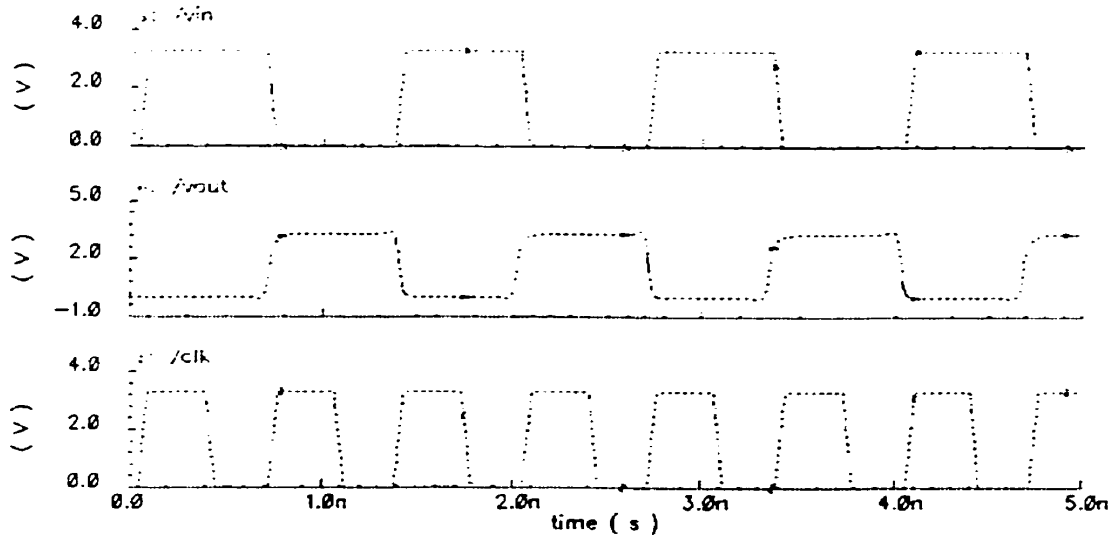


SIMULATIONS (Hspice)

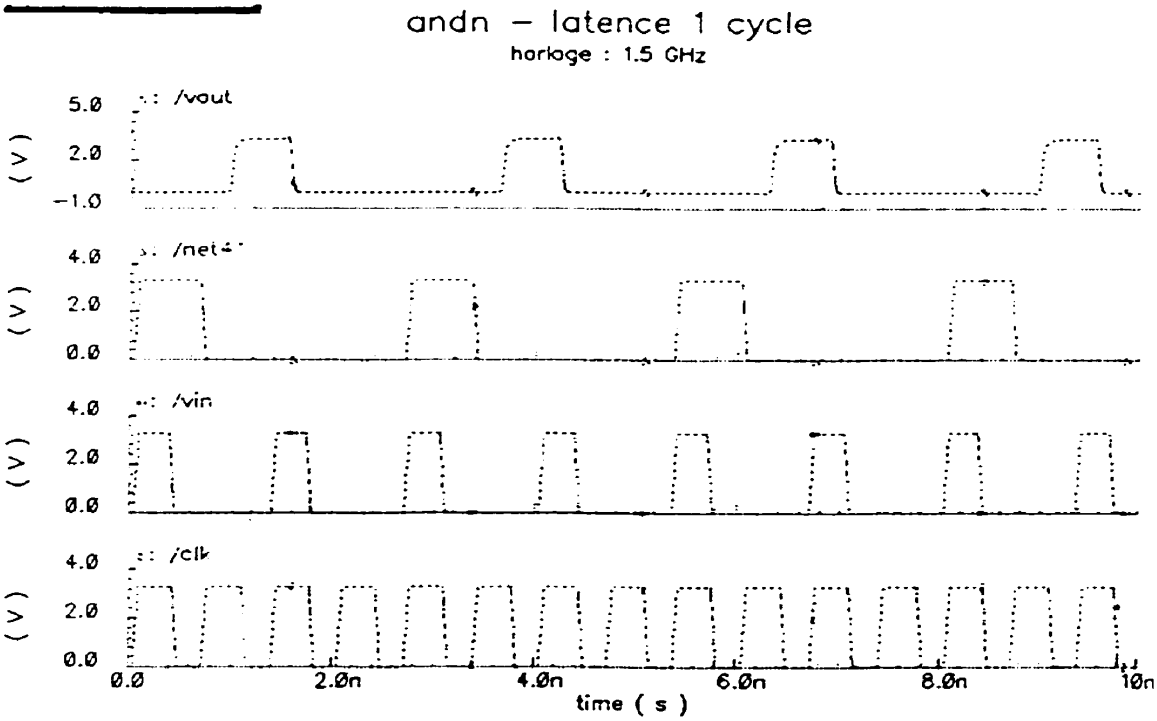
Buffern



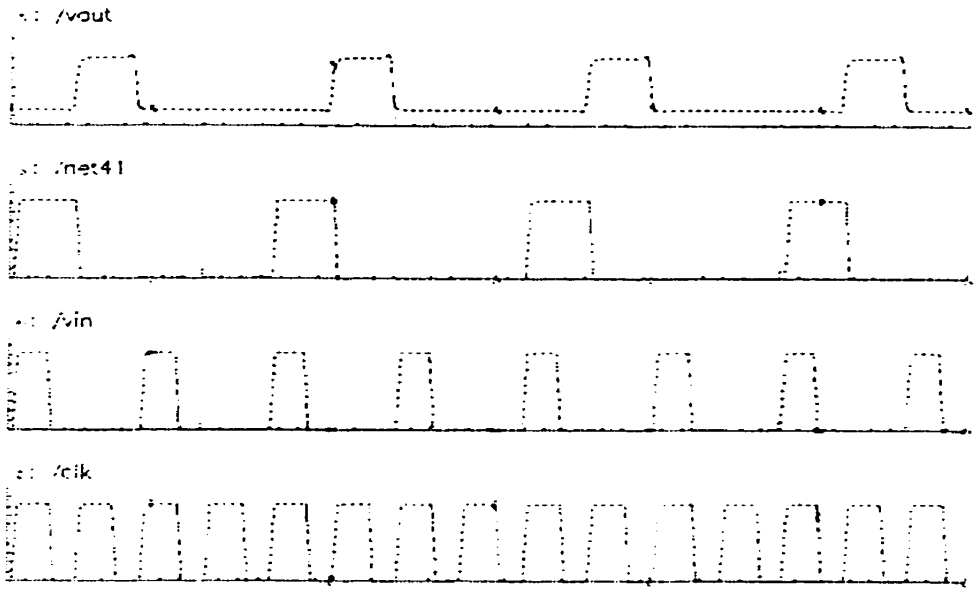
bufferp


bufferp
frequence horloge: 1.5 GHz
/vin
/vout
/clk
4.0
2.0
0.0
5.0
-1.0
(V)
0.0
1.0n
2.0n
3.0n
4.0n
5.0n
time (s)

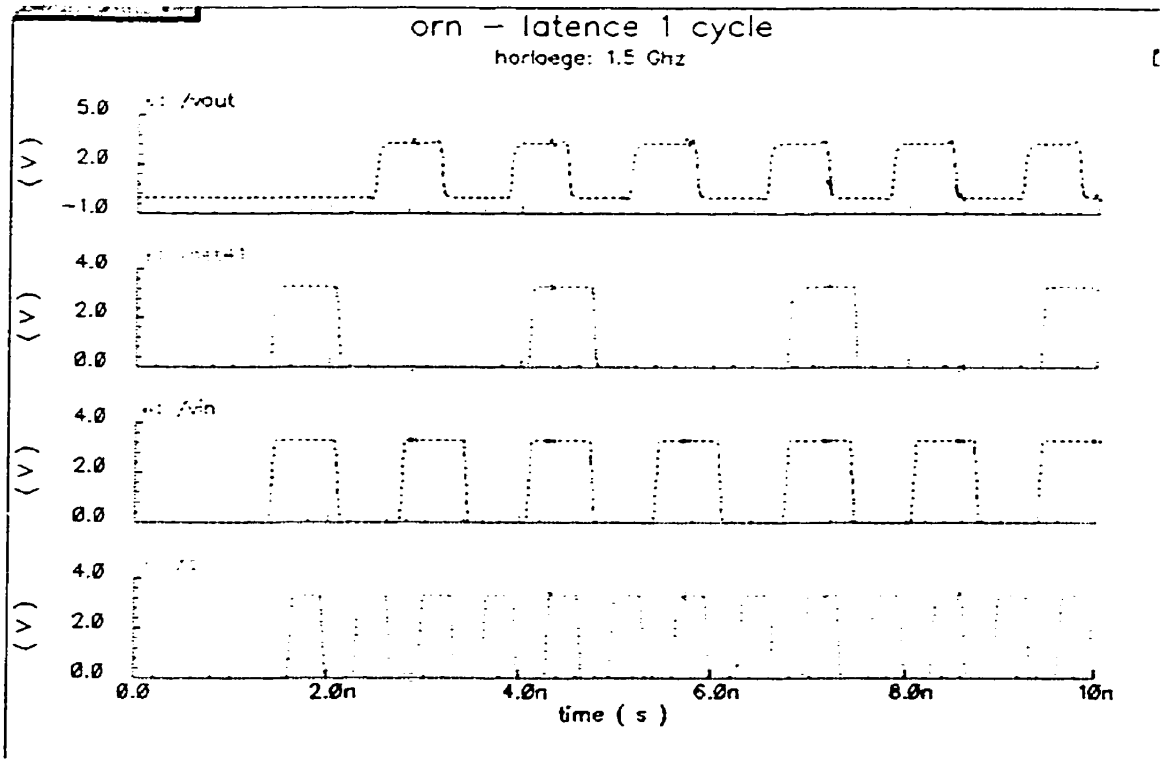
andn


andn – latence 1 cycle
horloge : 1.5 GHz
/vout
/net4
/vin
/clk
5.0
2.0
-1.0
4.0
0.0
(V)
0.0
2.0n
4.0n
6.0n
8.0n
10n
time (s)

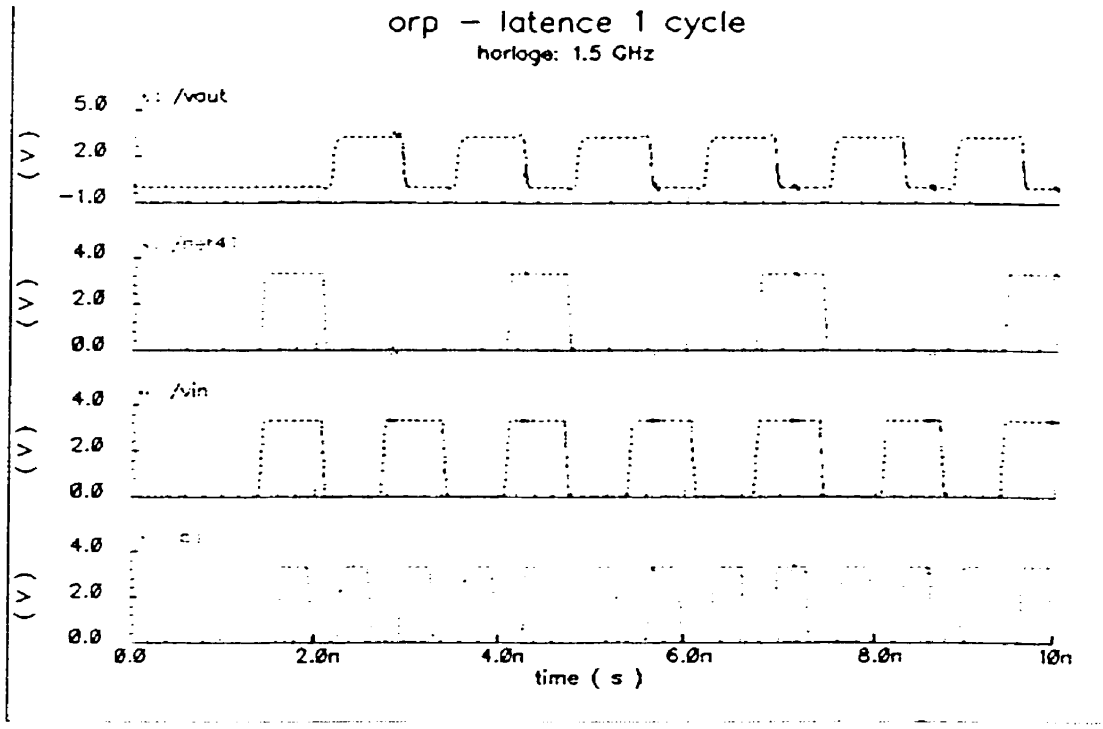
andp


/vout
/net41
/vin
/clk

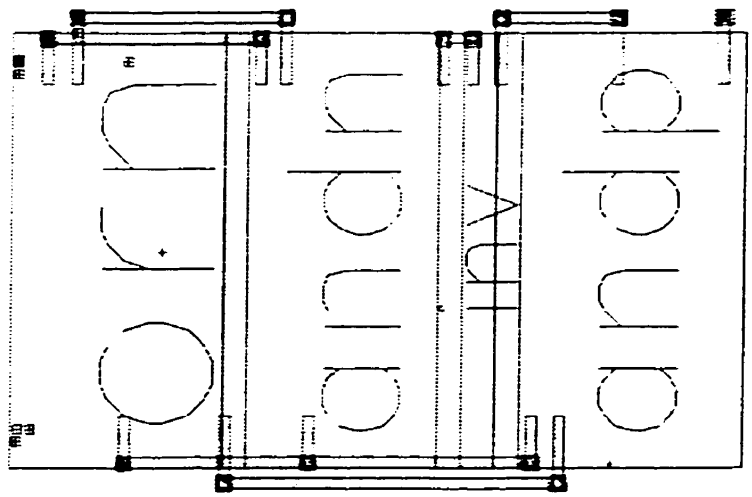
orn

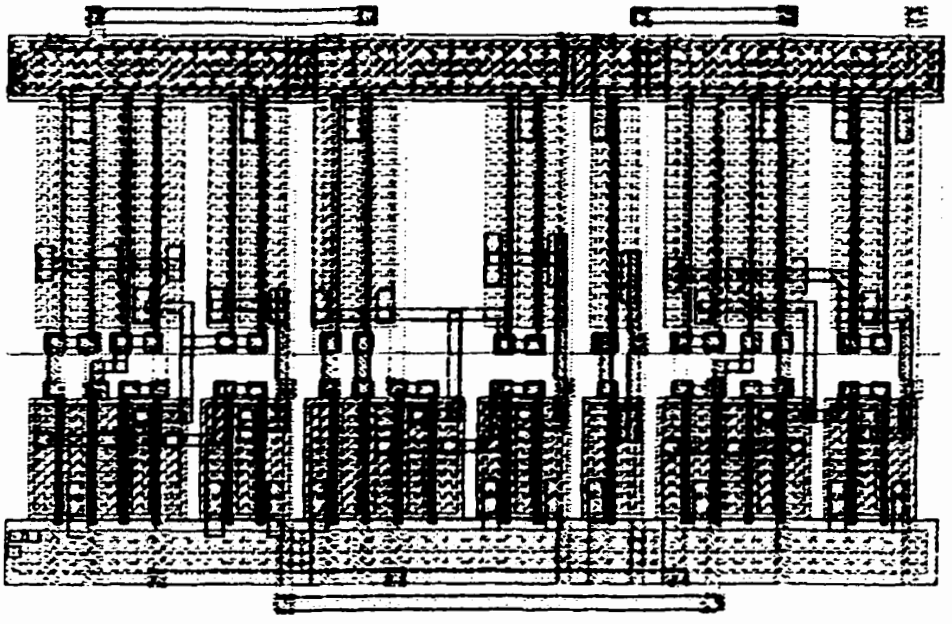

orn – latence 1 cycle
horloege: 1.5 Ghz
/vout
5.0
2.0
-1.0
4.0
2.0
0.0
/vin
4.0
2.0
0.0
4.0
2.0
0.0
(V)
0.0
2.0n
4.0n
6.0n
8.0n
10n
time (s)

orp



Parmi les cellules de base TSPC qui peuvent être formées à partir de celles-ci, on retrouve les XOR2. Voici une vue mixte layout-abstract du xor-NP. La figure qui suit est la vue layout du xor-NP.

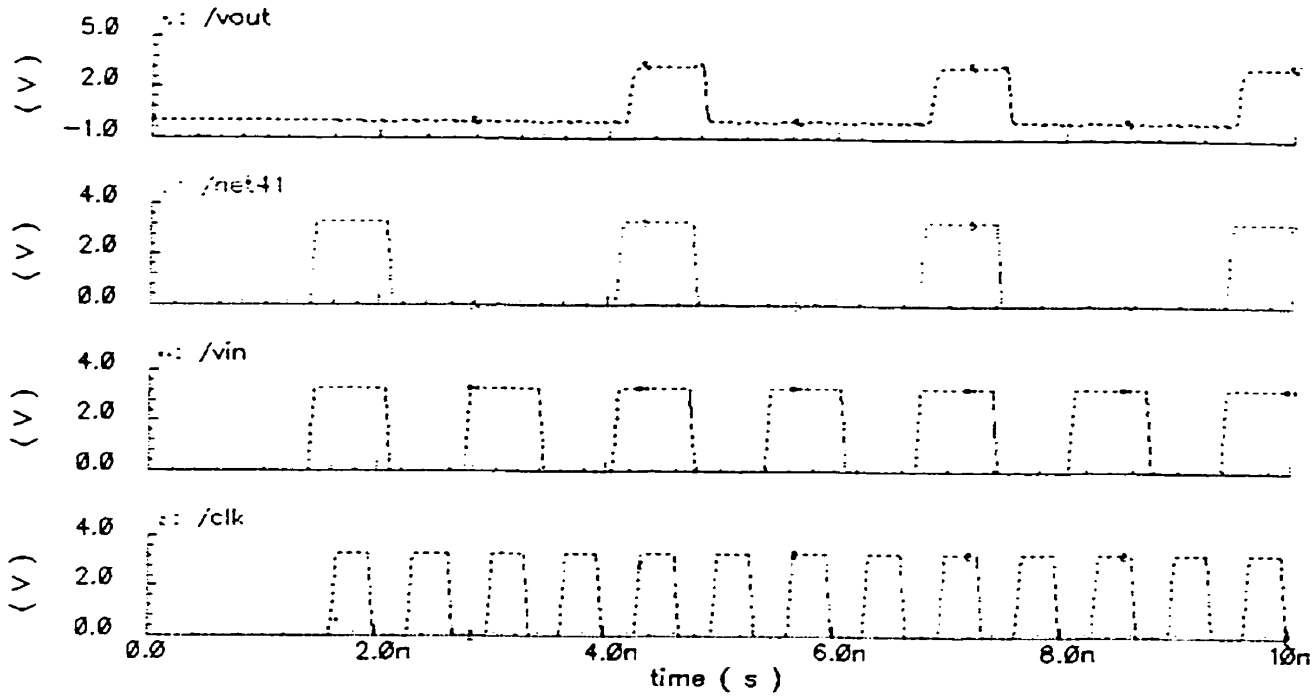




Et le résultat de la simulation de cette cellule est la suivante:

xor_NP – latence 1 cycle

horloge: 1.5GHz



Résultat LVS pour le bufferp

now running "/CMC/tools/cadence.97a/tools.sun4v/dfII/local/.simrc"
Appel direct de CMOSP35simrc.il
Loading CMOSP35simrc.il ...
Verilog Library Directory =
/CMC/tools/cadence.97a/tools.sun4v/dfII/local/lib/cmosp35/models/verilog/nwb
/CMC/tools/cadence.97a/tools.sun4v/dfII/local/lib/cmosp35/models/verilog/udp
Veritime Library Directory =
/CMC/tools/cadence.97a/tools.sun4v/dfII/local/lib/cmosp35/models/verilog/nwb
/CMC/tools/cadence.97a/tools.sun4v/dfII/local/lib/cmosp35/models/verilog/udp
Completed CMOSP35simrc.il
finished running "/CMC/tools/cadence.97a/tools.sun4v/dfII/local/.simrc"

Running simulation in directory: "/share/henri/4/convol/ginette/LVS".

Begin netlist: Feb 19 14:56:45 1999
view name list= ("auLvs" "extracted" "schematic")
stop name list = ("auLvs")
library name = "ginette"
cell name = "buffern"
view name = "extracted"
globals lib = "cmosp35"
Running Artist Flat Netlisting ...
End netlist: Feb 19 14:56:51 1999

Moving original netlist to extNetlist
Removing parasitic components from netlist
presistors removed: 0
pcapacitors removed: 0
pinductors removed: 0
pdiodes removed: 0
trans lines removed: 0
8 nodes merged into 8 nodes

Begin netlist: Feb 19 14:56:52 1999
view name list= ("auLvs" "schematic")
stop name list = ("auLvs")
library name = "ginette"
cell name = "buffern"
view name = "schematic"
globals lib = "cmosp35"

Running Artist Flat Netlisting ...
End netlist: Feb 19 14:56:53 1999

Moving original netlist to extNetlist
Removing parasitic components from netlist
presistors removed: 0
pcapacitors removed: 0
pinductors removed: 0
pdiodes removed: 0
trans lines removed: 0
8 nodes merged into 8 nodes

Running netlist comparison program: LVS
Begin comparison: Feb 19 14:56:54 1999
@(#)$CDS: LVS version 4.4.1 06/17/98 23:22 (cds10067) $
Warning: Unknown device "resistor" on a compareDeviceProperty command.
Warning: Unknown device "capacitor" on a compareDeviceProperty command.
Warning: Unknown device "resistor" on a permuteDevice command.
Warning: Unknown device "resistor" on a permuteDevice command.
Warning: Unknown device "capacitor" on a permuteDevice command.
Warning: Unknown device "capacitor" on a permuteDevice command.

The net-lists match.

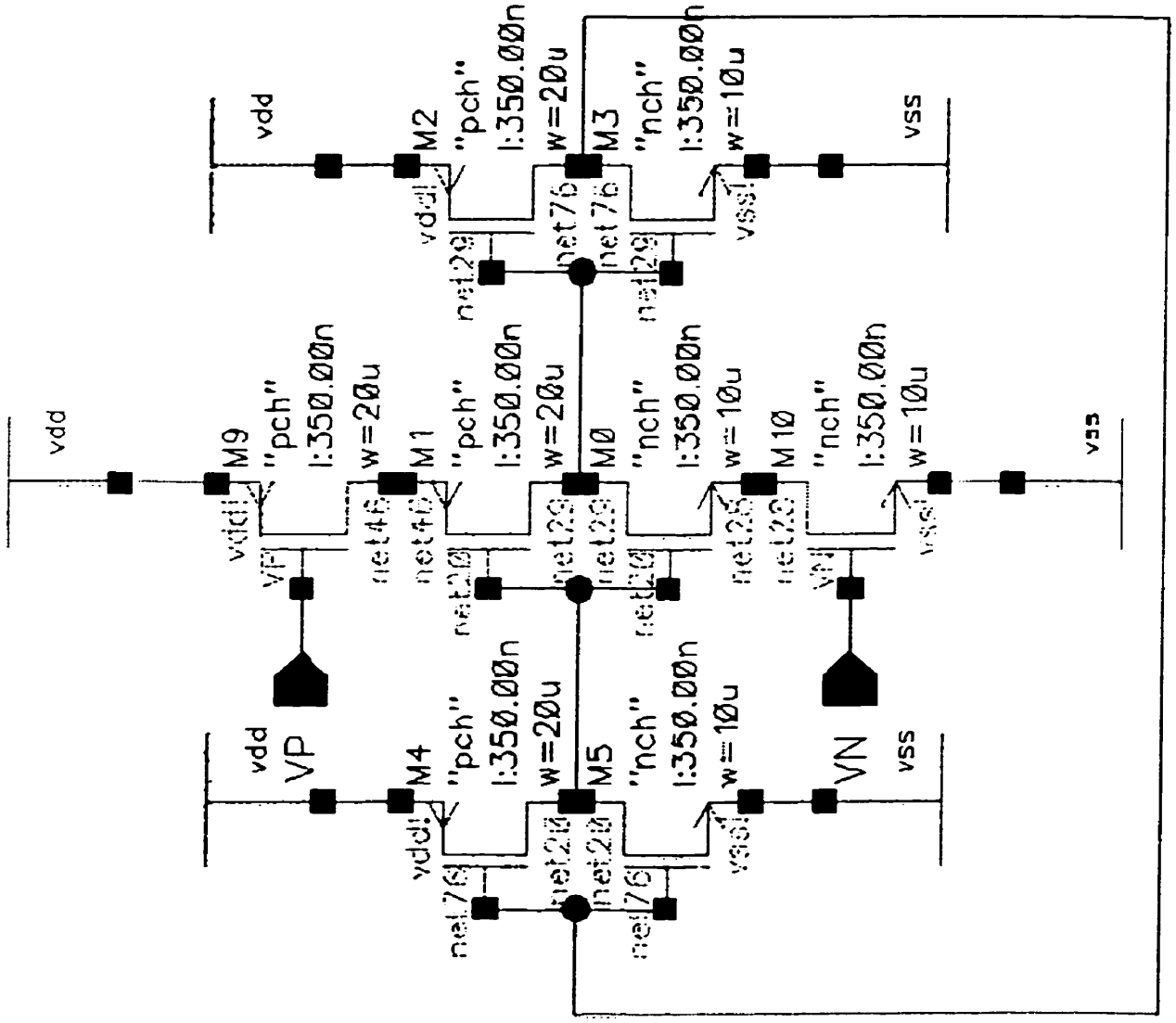
	layout	schematic	
	instances		
un-matched	0	0	
rewired		0	0
size errors	0	0	
pruned	0	0	
active	7	7	
total	7	7	
	nets		
un-matched	0	0	
merged		0	0
pruned	0	0	
active	7	7	
total	7	7	
	terminals		
un-matched	0	0	
total	6	6	

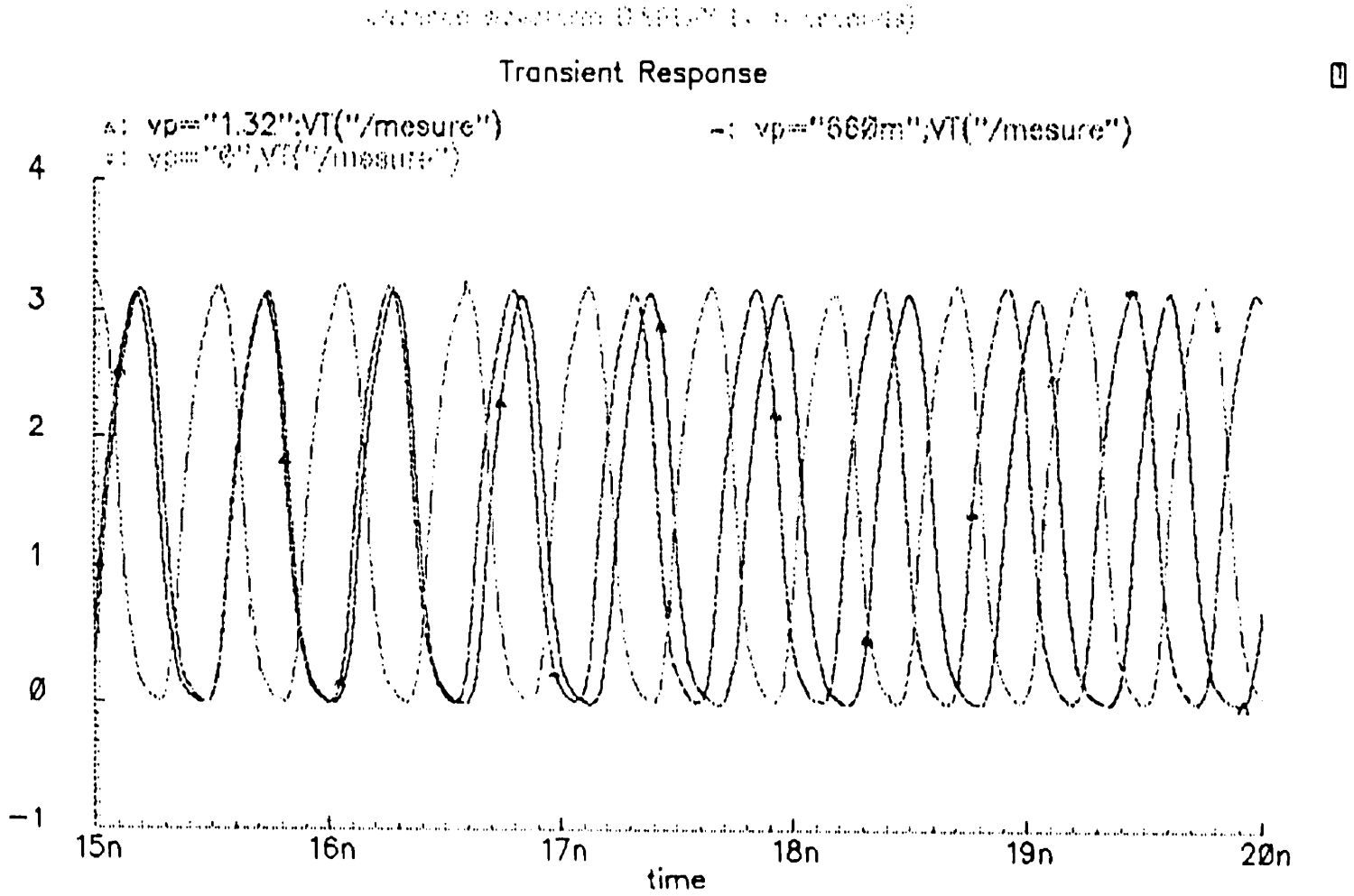
End comparison: Feb 19 14:56:56 1999

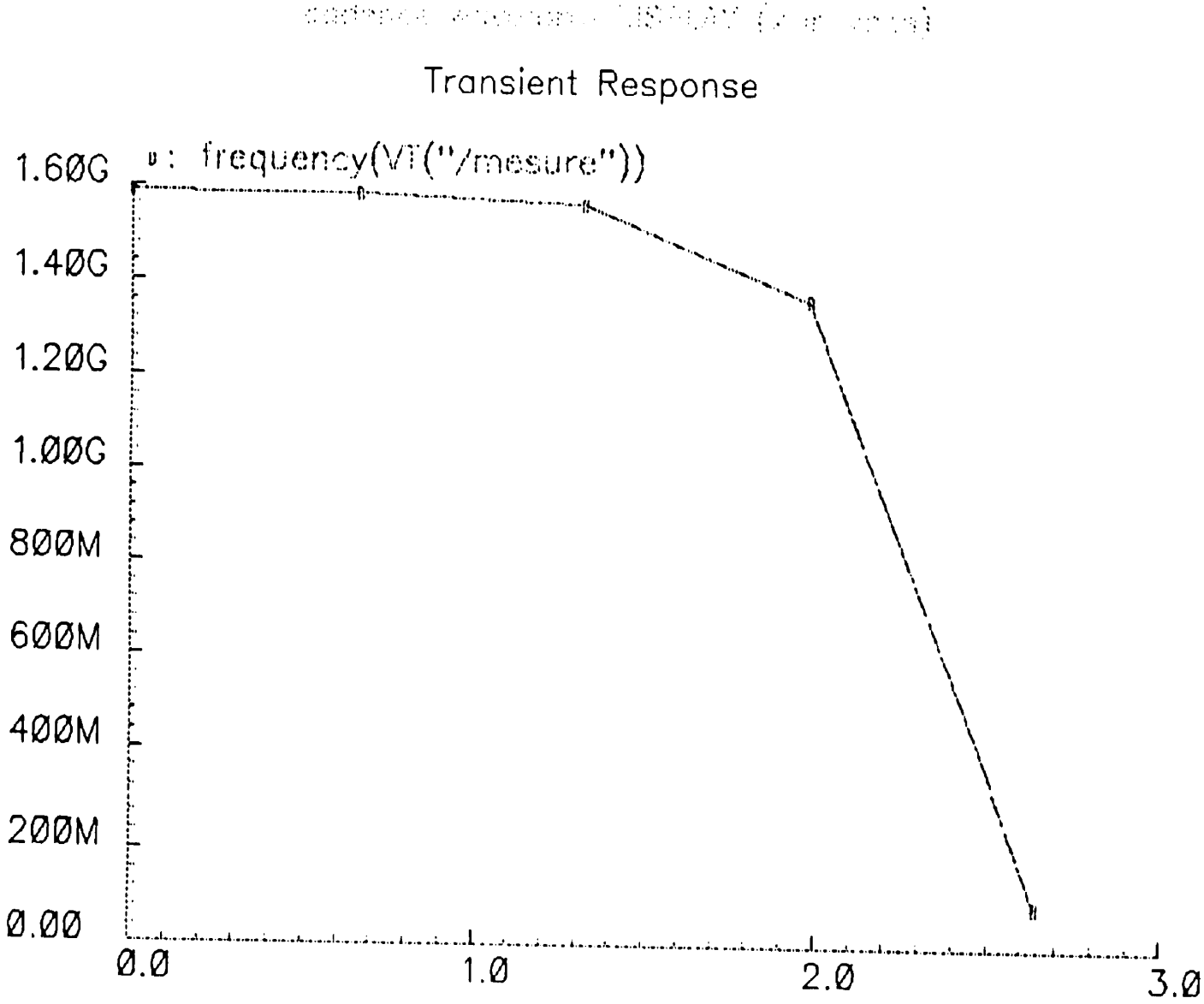
ANNEXE 2

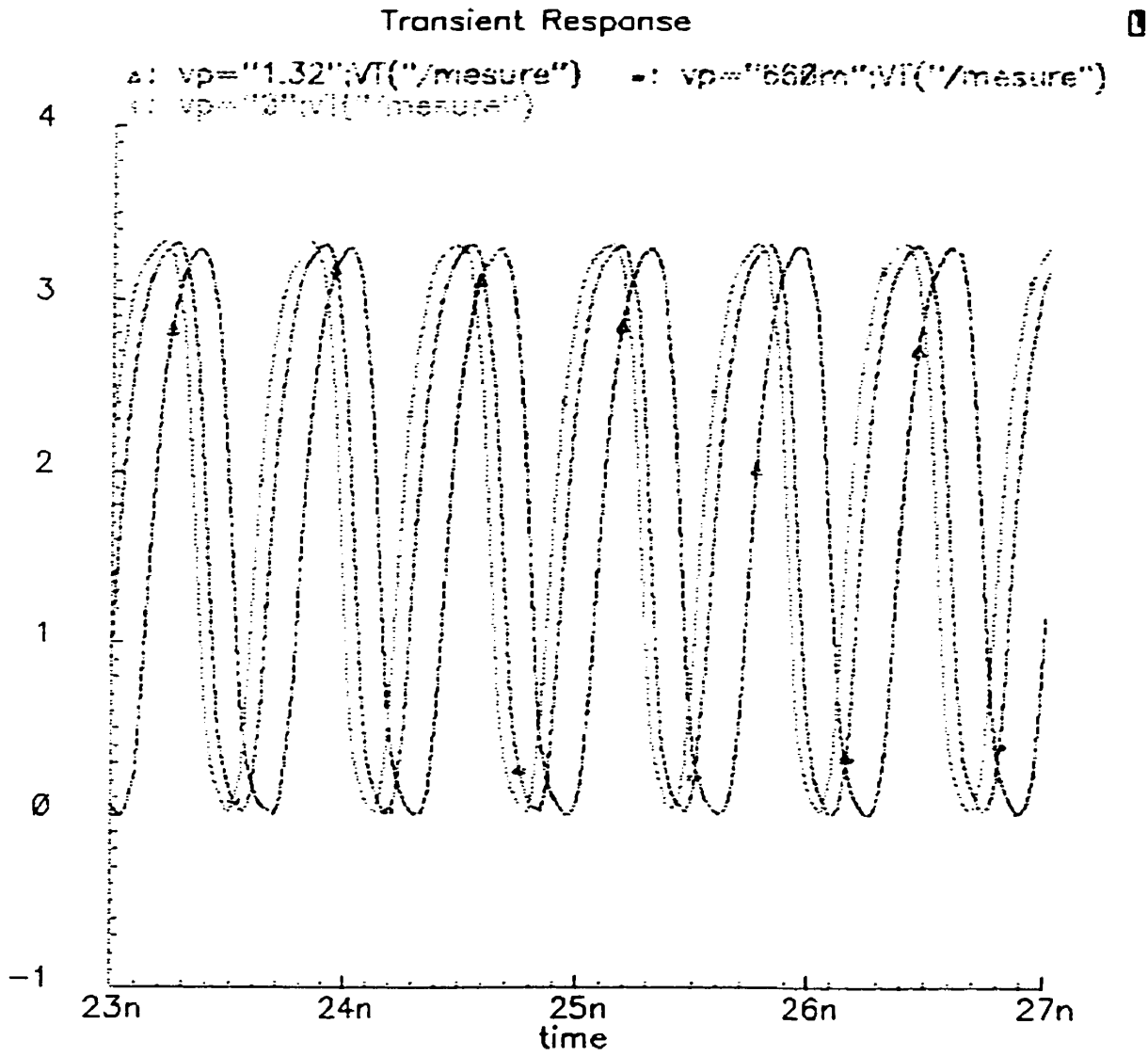
Oscillateurs à tension commandée

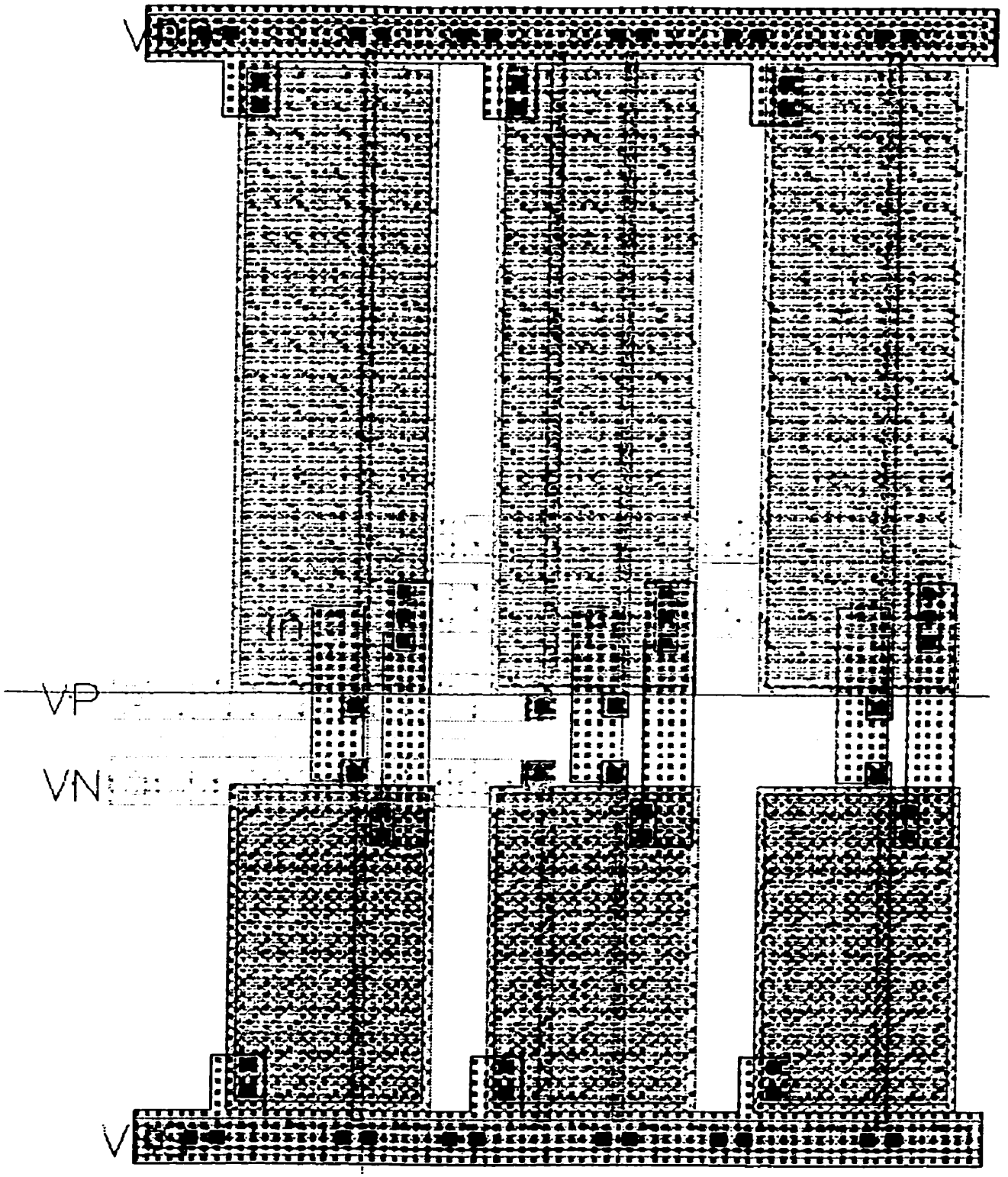
oscillateur Nekili - oscil1

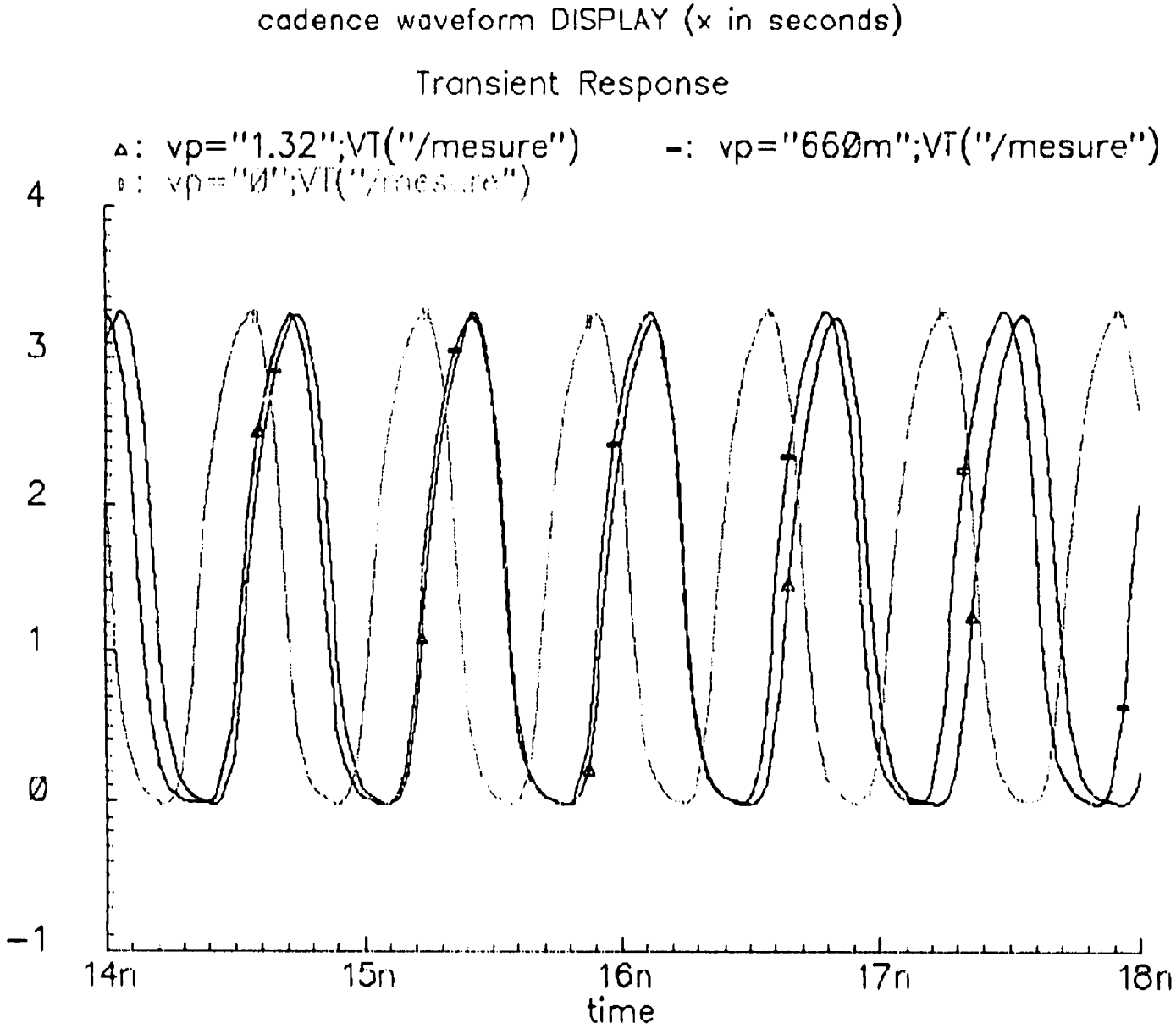


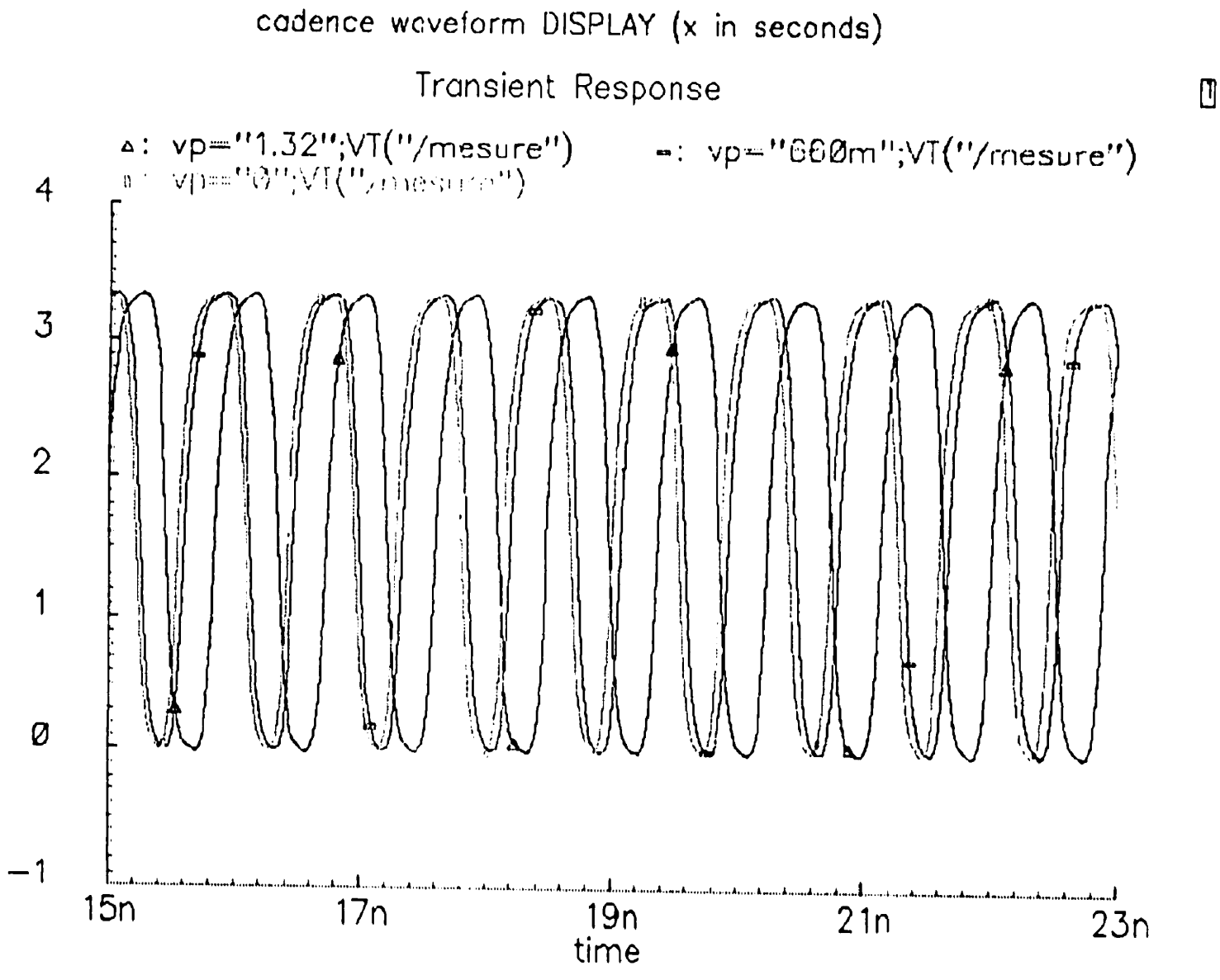

Transient Response
a: vp="1.32":VT("/mesure")
-: vp="660m":VT("/mesure")
4
3
2
1
Ø
−1
15n
16n
17n
18n
19n
20n
time

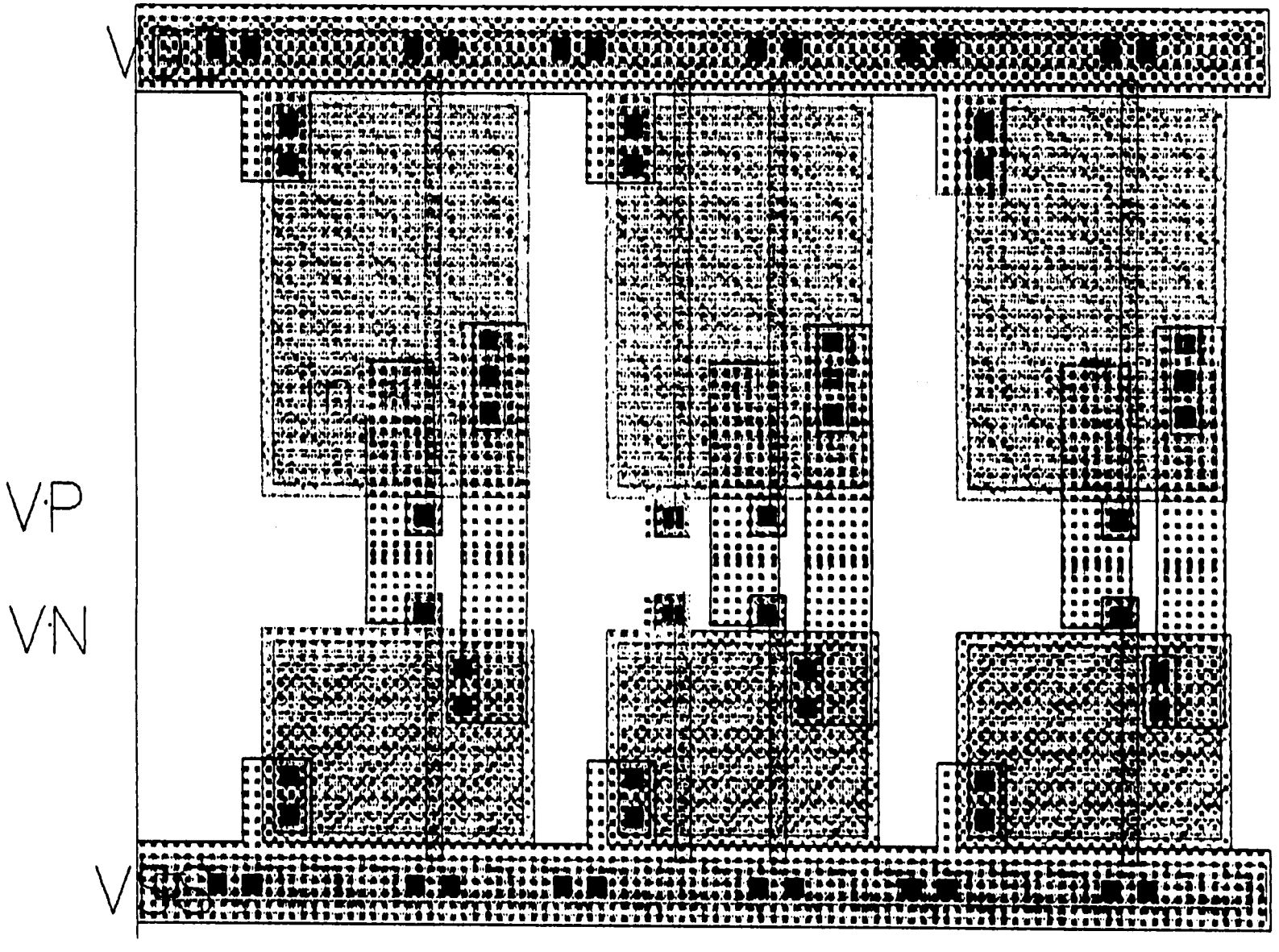

Transient Response
v : frequency(VT("/mesure"))
1.60G
1.40G
1.20G
1.00G
800M
600M
400M
200M
0.00
0.0
1.0
2.0
3.0

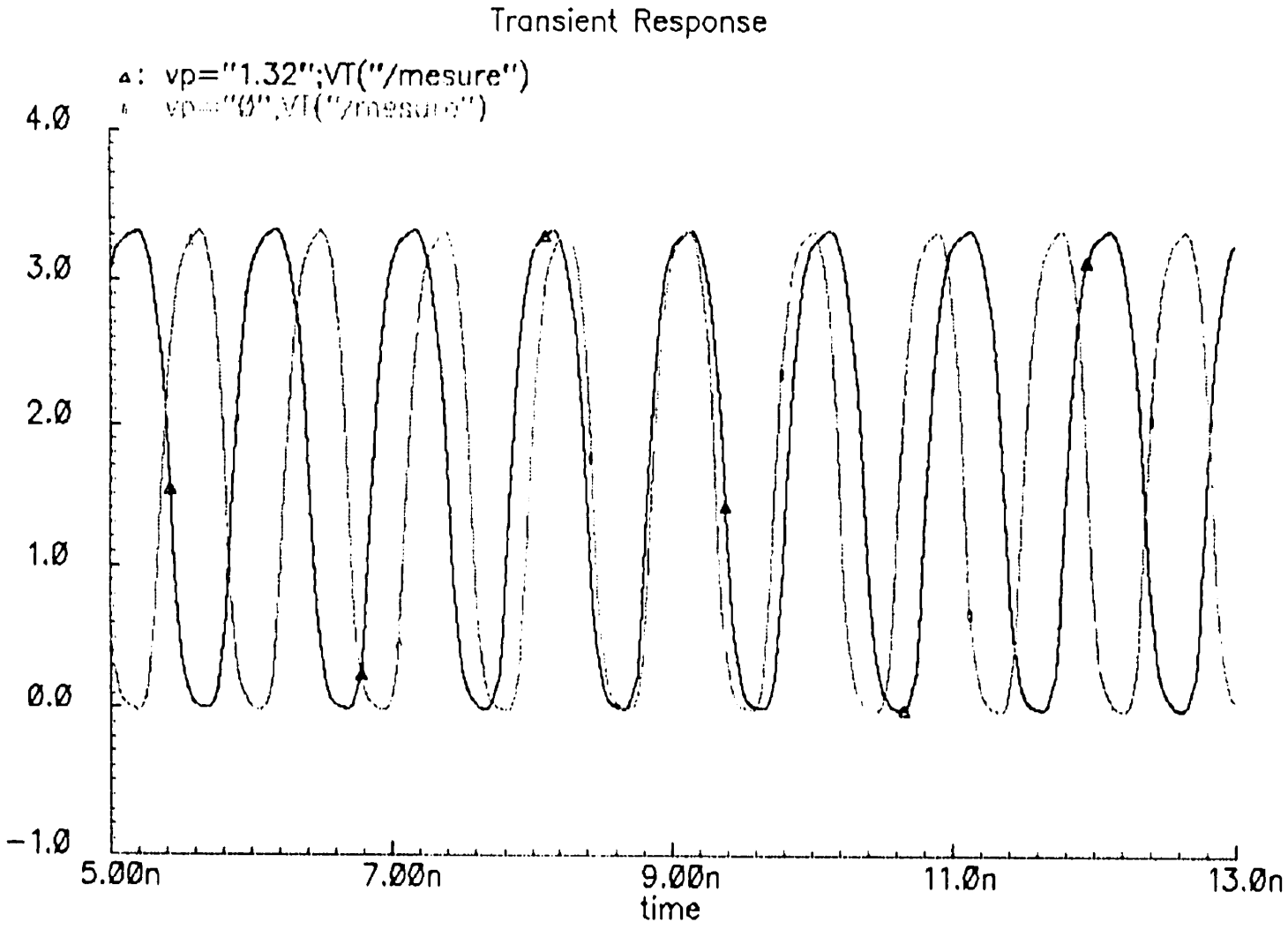

Transient Response
a: vp="1.32";VT("/mesure")
4
3
2
1
Ø
−1
23n
24n
25n
26n
27n
time

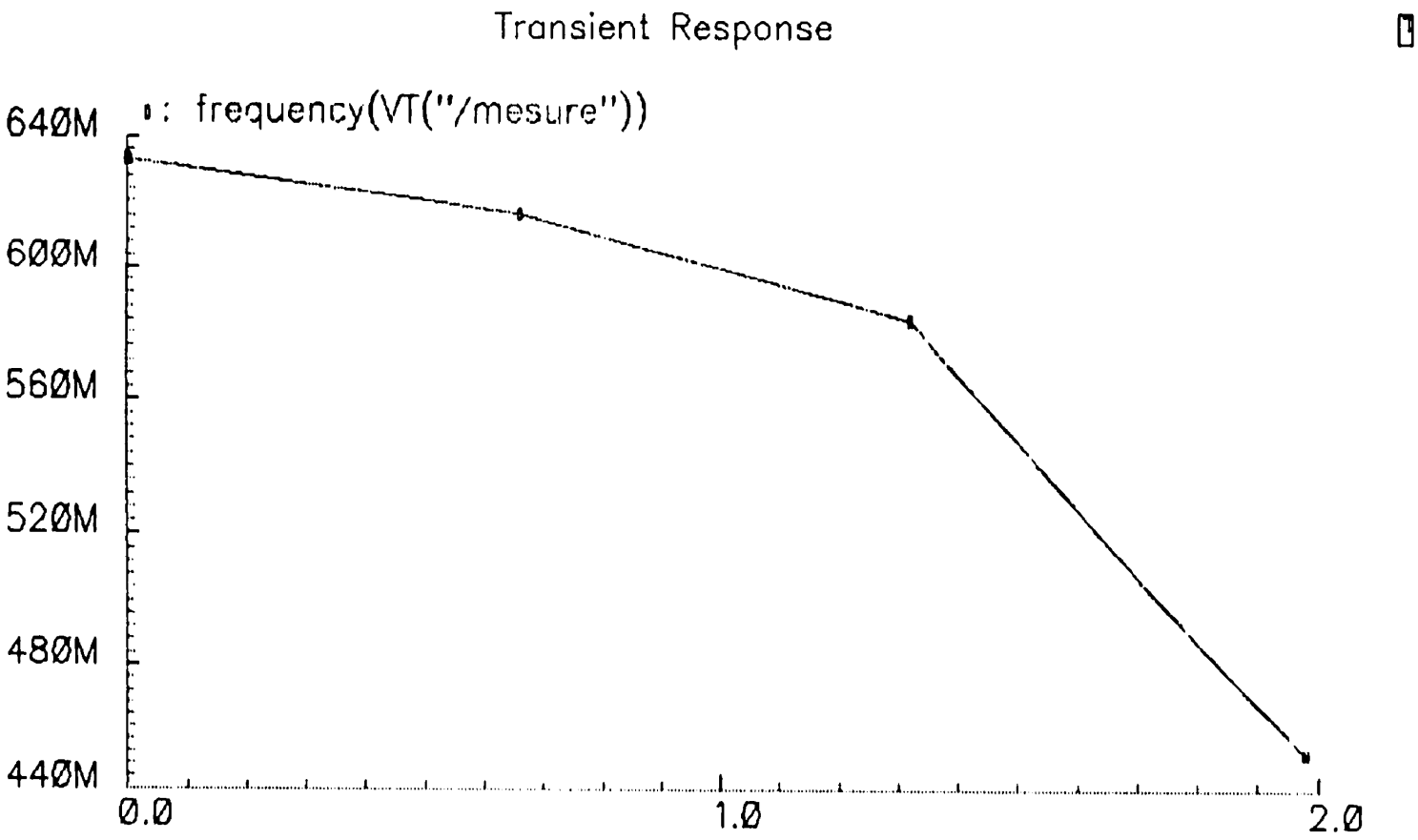

VP
VN

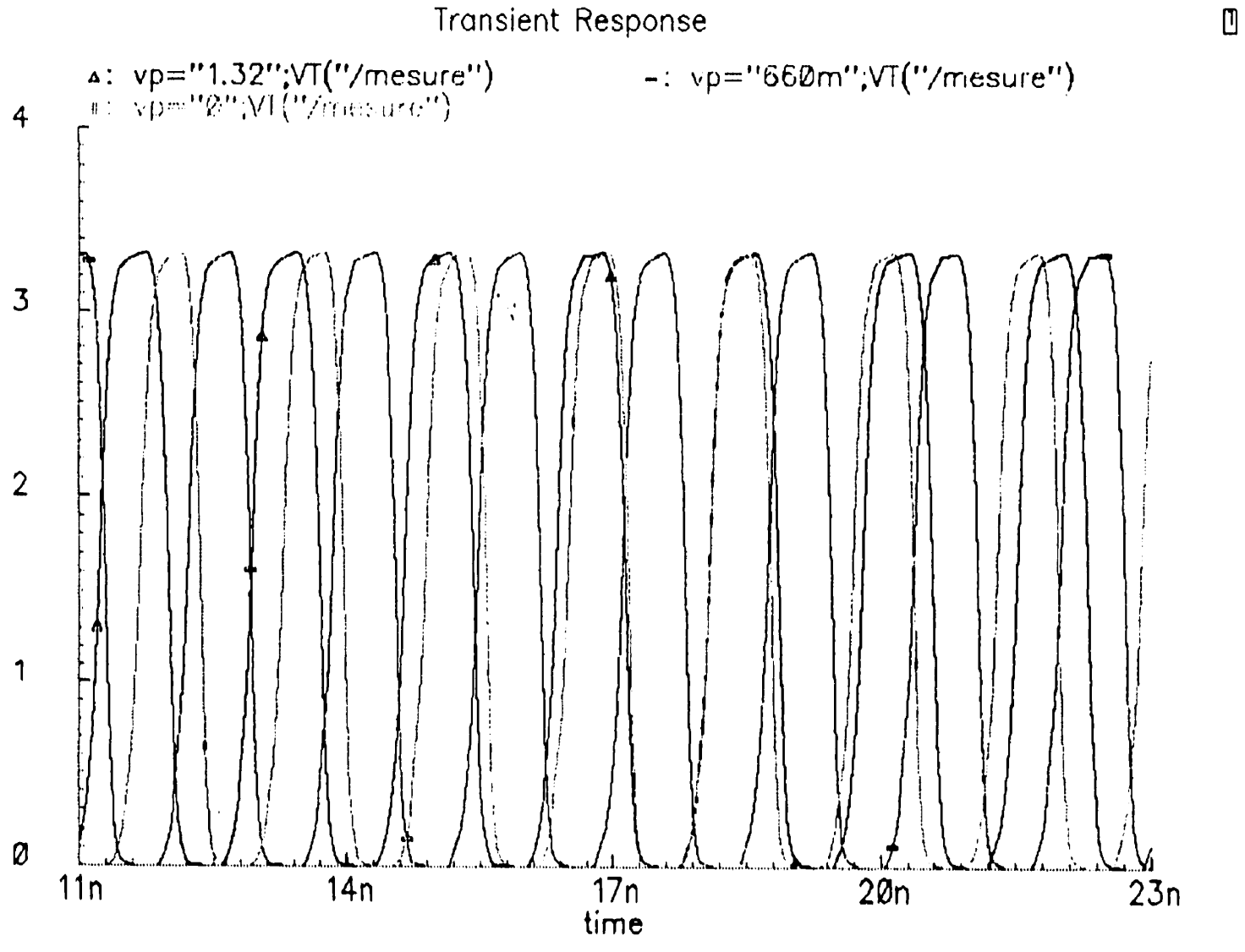

cadence waveform DISPLAY (x in seconds)
Transient Response
▵: vp="1.32";VT("/mesure")
-: vp="660m";VT("/mesure")
4
3
2
1
Ø
−1
14n
15n
16n
17n
18n
time

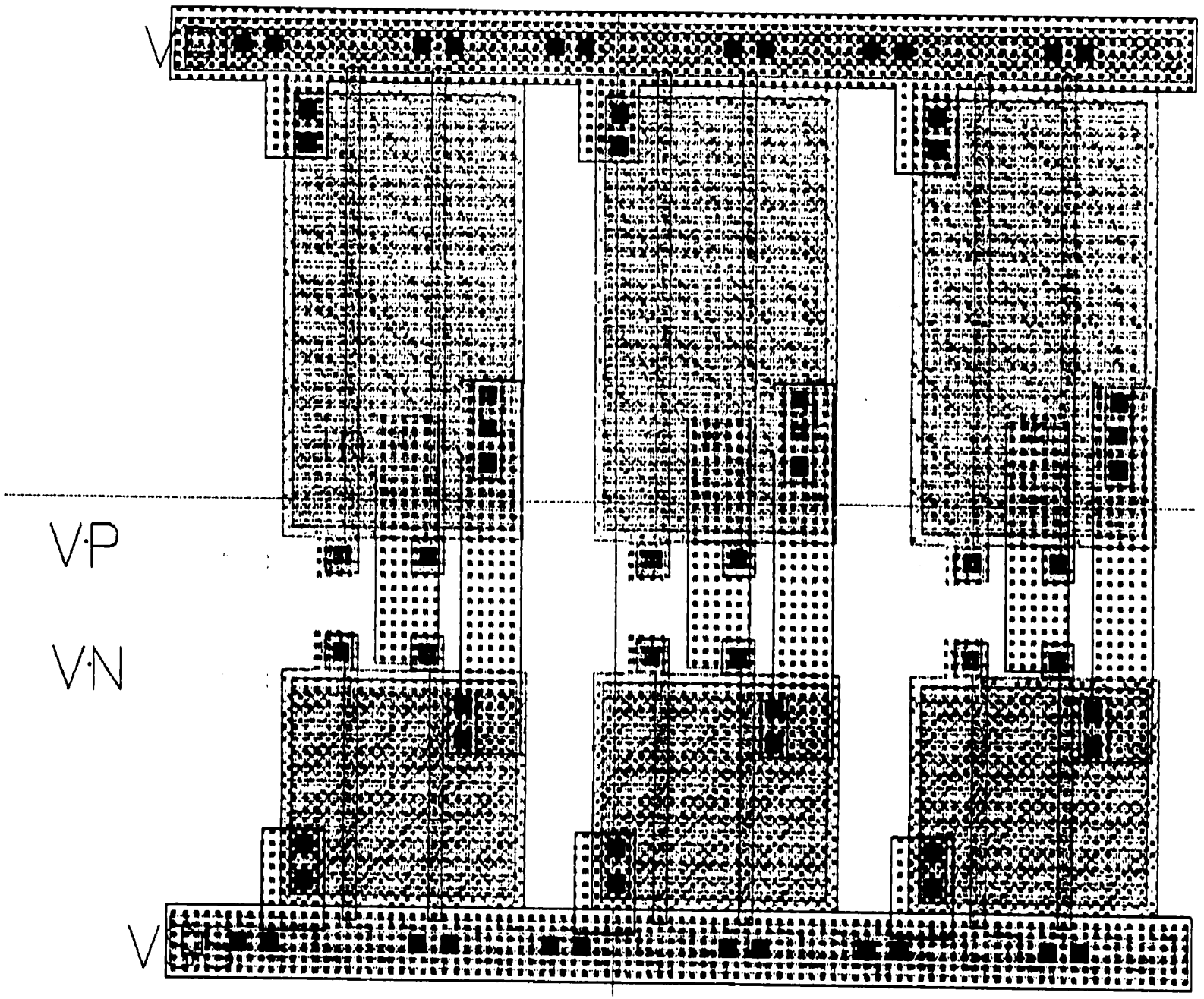

cadence waveform DISPLAY (x in seconds)
Transient Response
▵: vp="1.32";VT("/mesure")
▫: vp="600m";VT("/mesure")
4
3
2
1
0
−1
15n
17n
19n
21n
23n
time

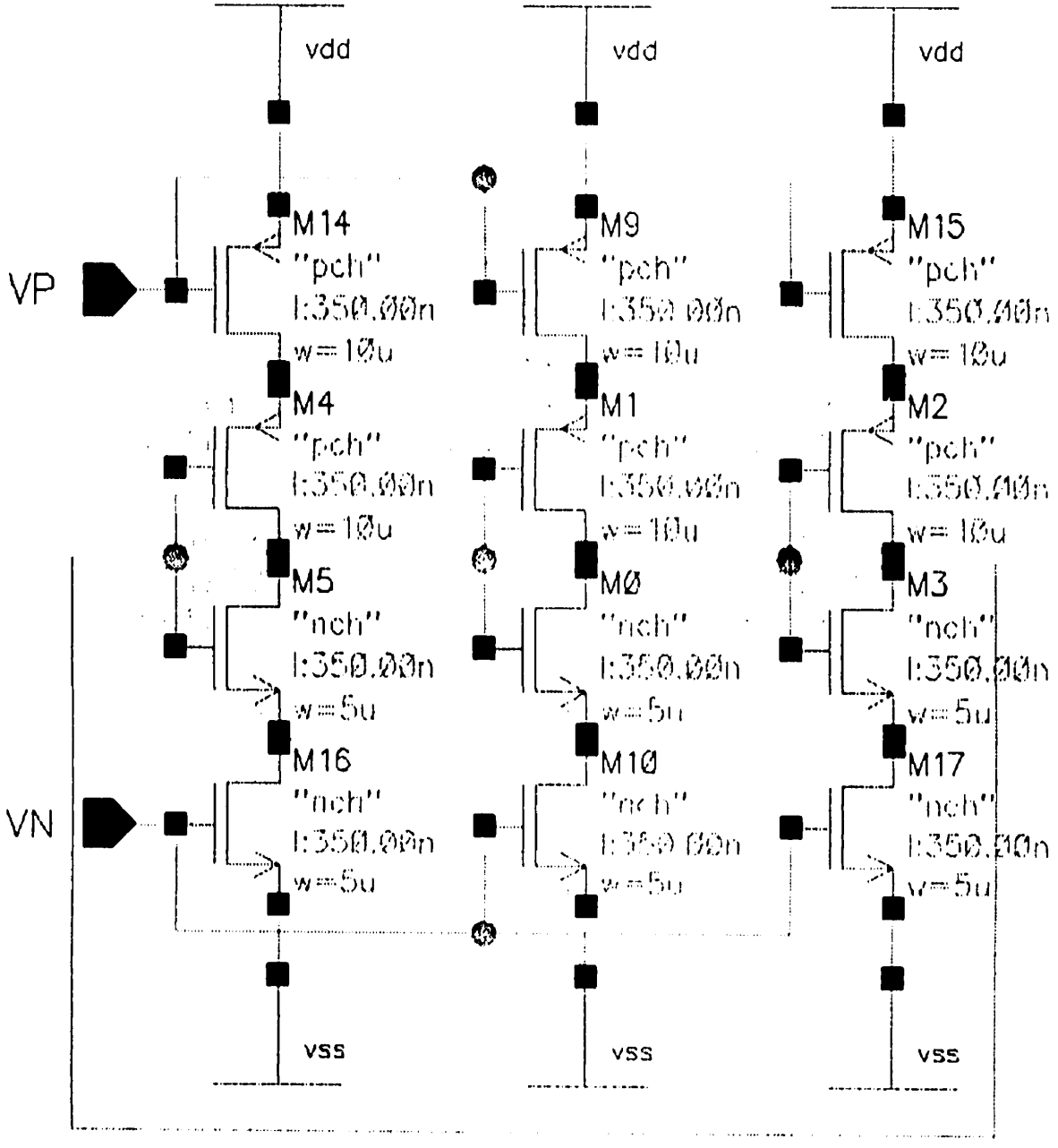

V
V·P
V·N
V

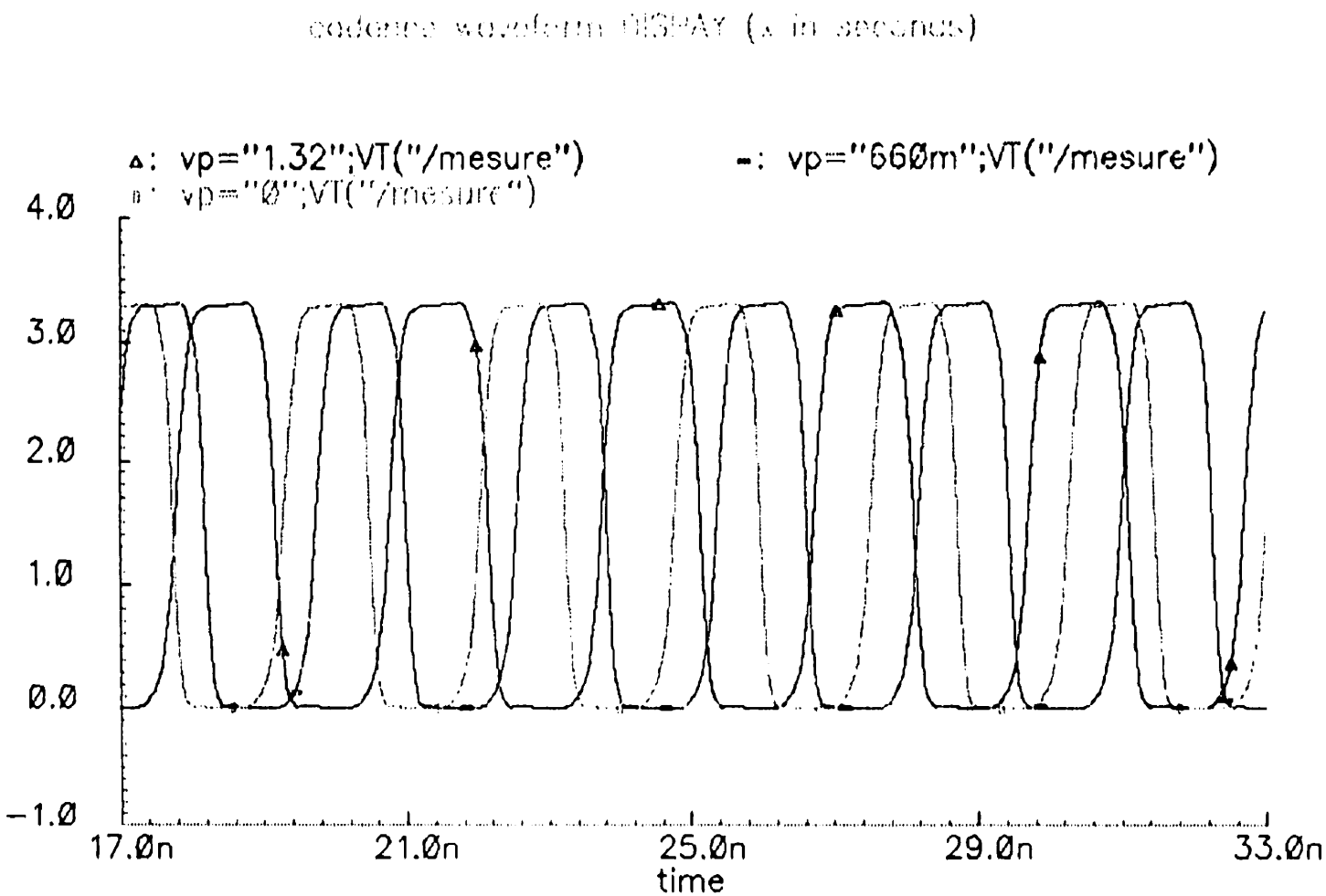

Transient Response
a: vp="1.32";VT("/mesure")
4.Ø
3.Ø
2.Ø
1.Ø
Ø.Ø
−1.Ø
5.ØØn
7.ØØn
9.ØØn
11.Øn
13.Øn
time

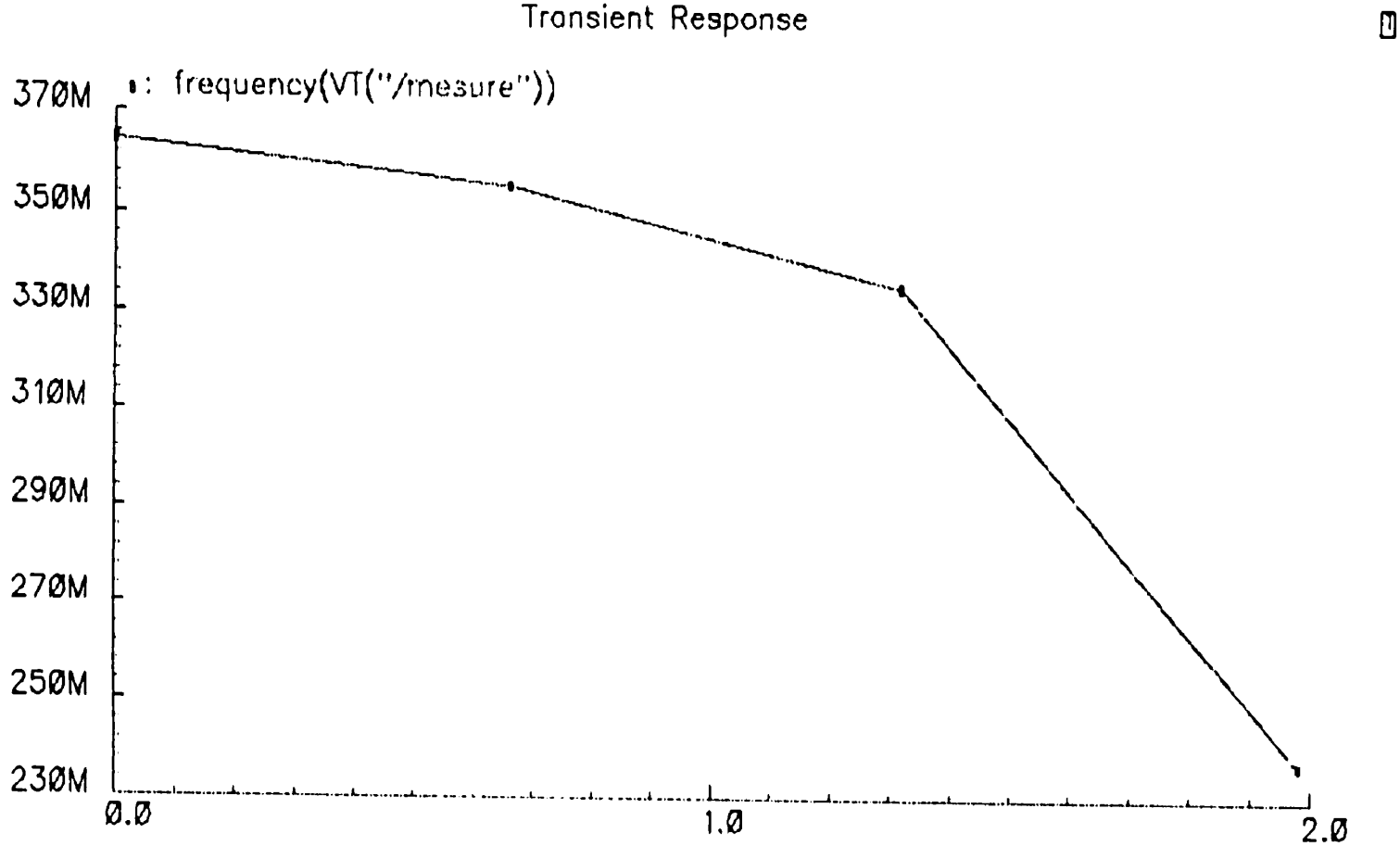

Transient Response
: frequency(VT("/mesure"))
640M
600M
560M
520M
480M
440M
0.0
1.0
2.0

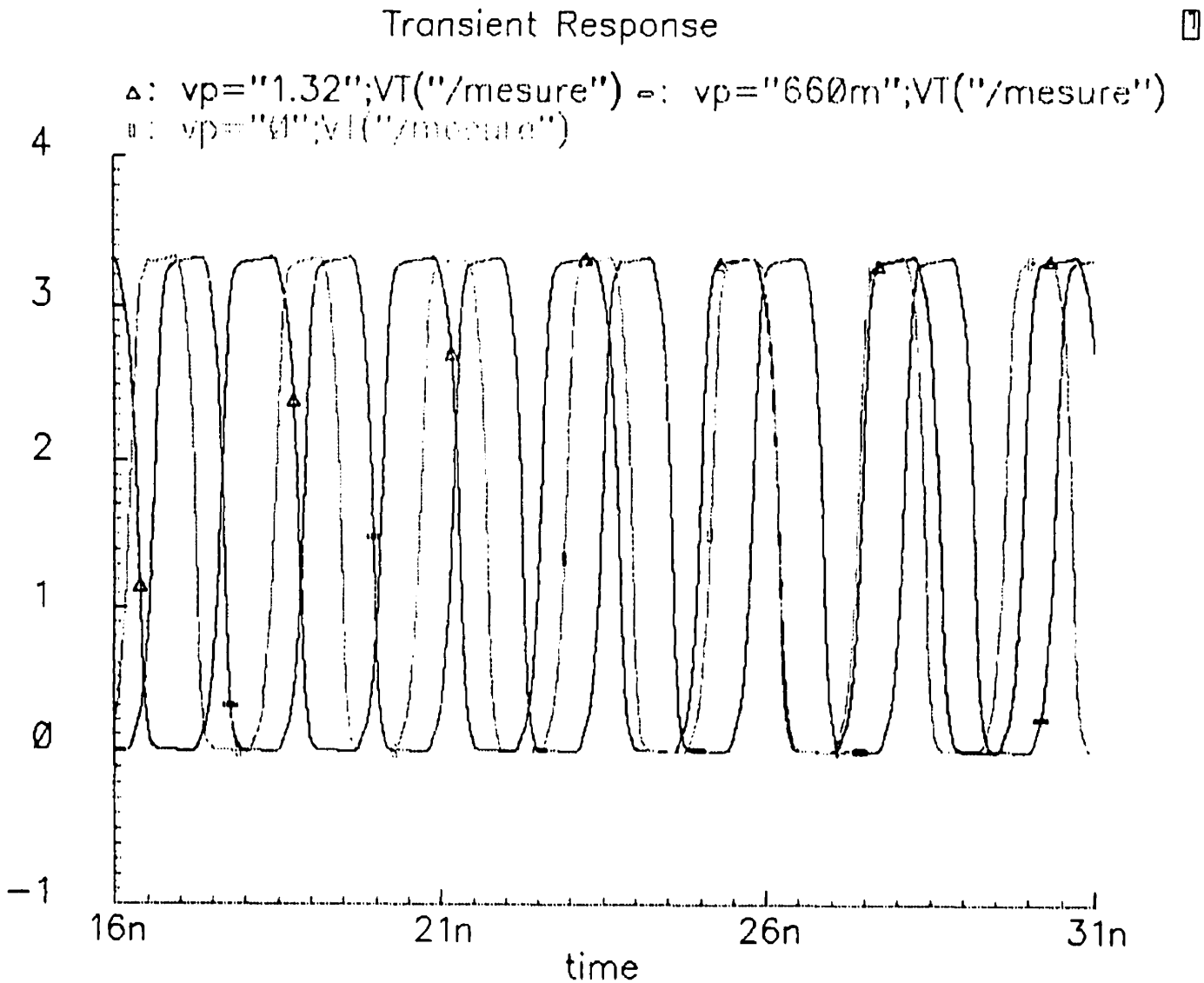

Transient Response
▵: vp="1.32";VT("/mesure")
-: vp="660m";VT("/mesure")
4
3
2
1
Ø
11n
14n
17n
20n
23n
time

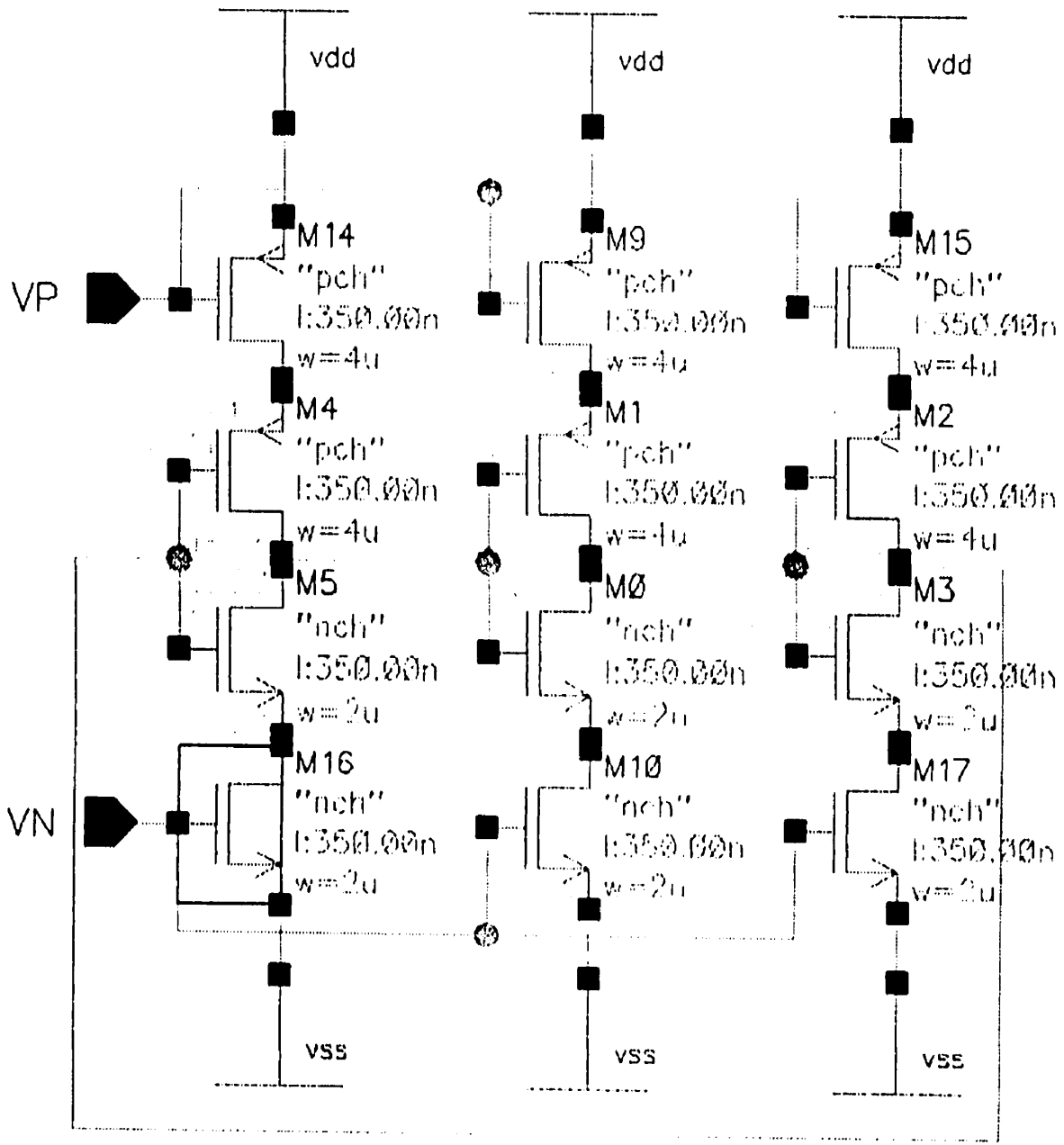

V
V·P
V·N
V

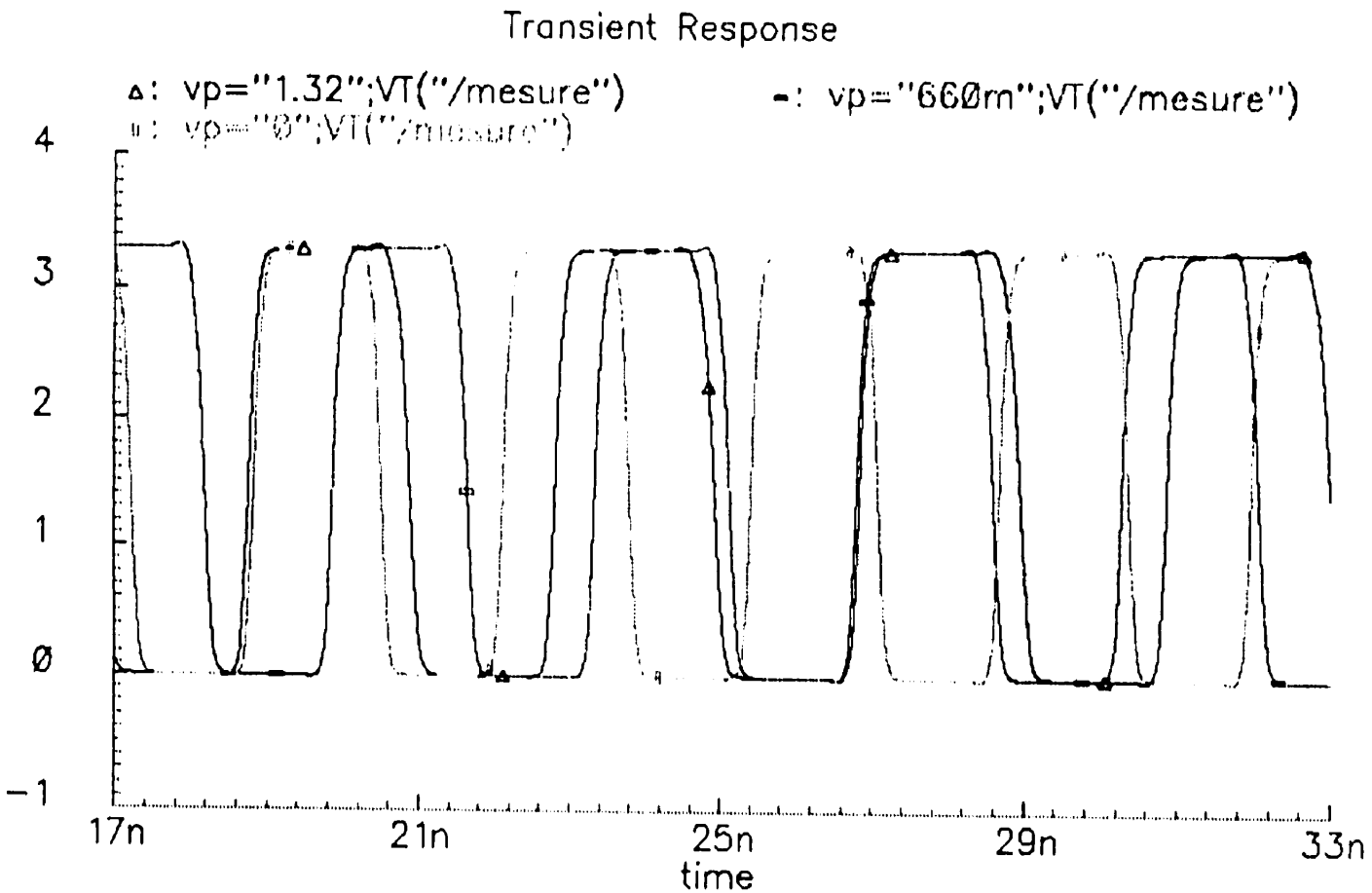

vdd
vdd
vdd
VP
VN
M14
"pch"
l:350.00n
w=10u
M4
"pch"
l:350.00n
w=10u
M5
"nch"
l:350.00n
w=5u
M16
"nch"
l:350.00n
w=5u
M9
"pch"
w=10u
M1
"pch"
w=10u
M0
"nch"
w=5u
M10
"nch"
w=5u
M15
"pch"
w=10u
M2
"pch"
w=10u
M3
"nch"
w=5u
M17
"nch"
w=5u
vss
vss
vss

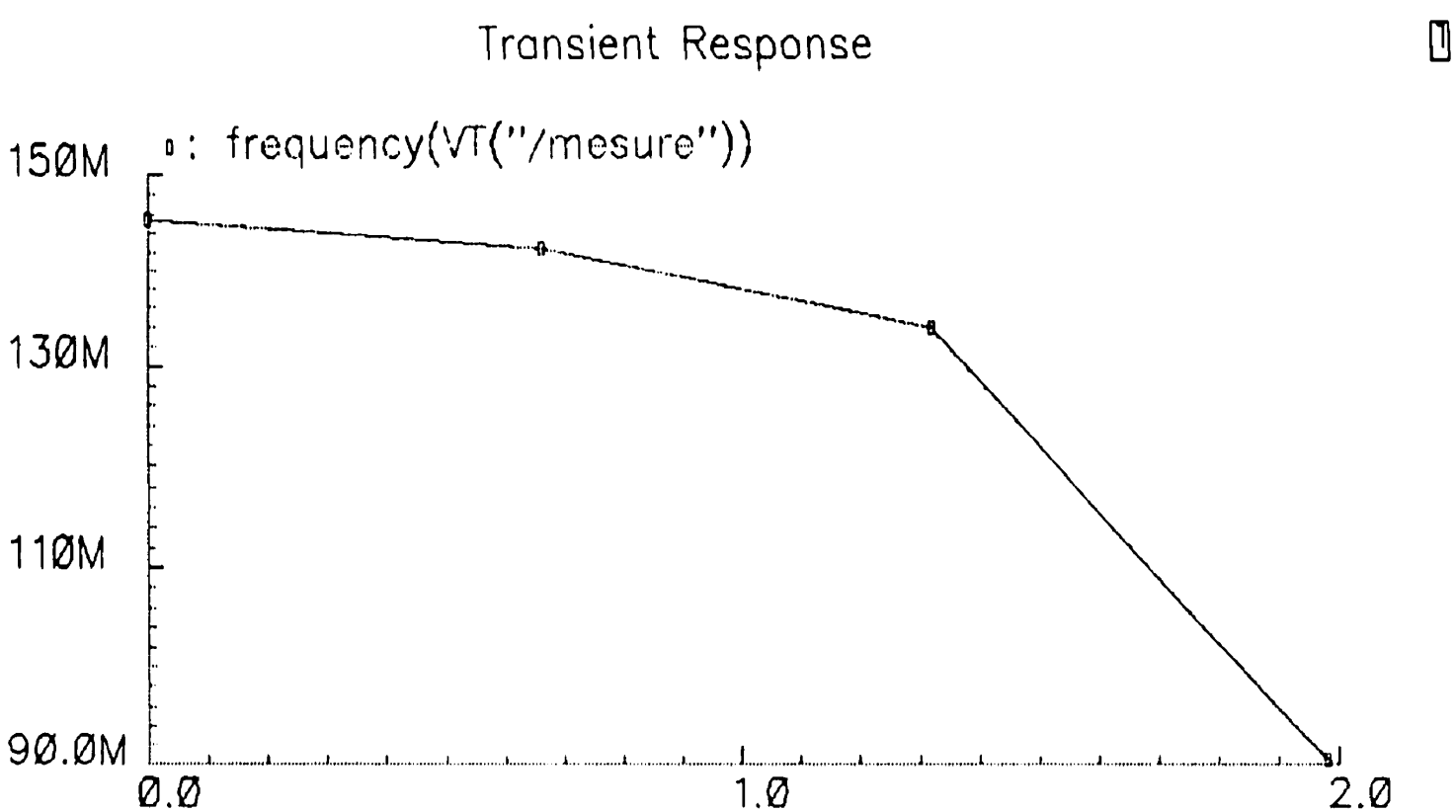

cadence waveform DISPLAY (x in seconds)
▵: vp="1.32";VT("/mesure")
▫: vp="0";VT("/mesure")
-: vp="660m";VT("/mesure")
4.0
3.0
2.0
1.0
0.0
-1.0
17.0n
21.0n
25.0n
29.0n
33.0n
time

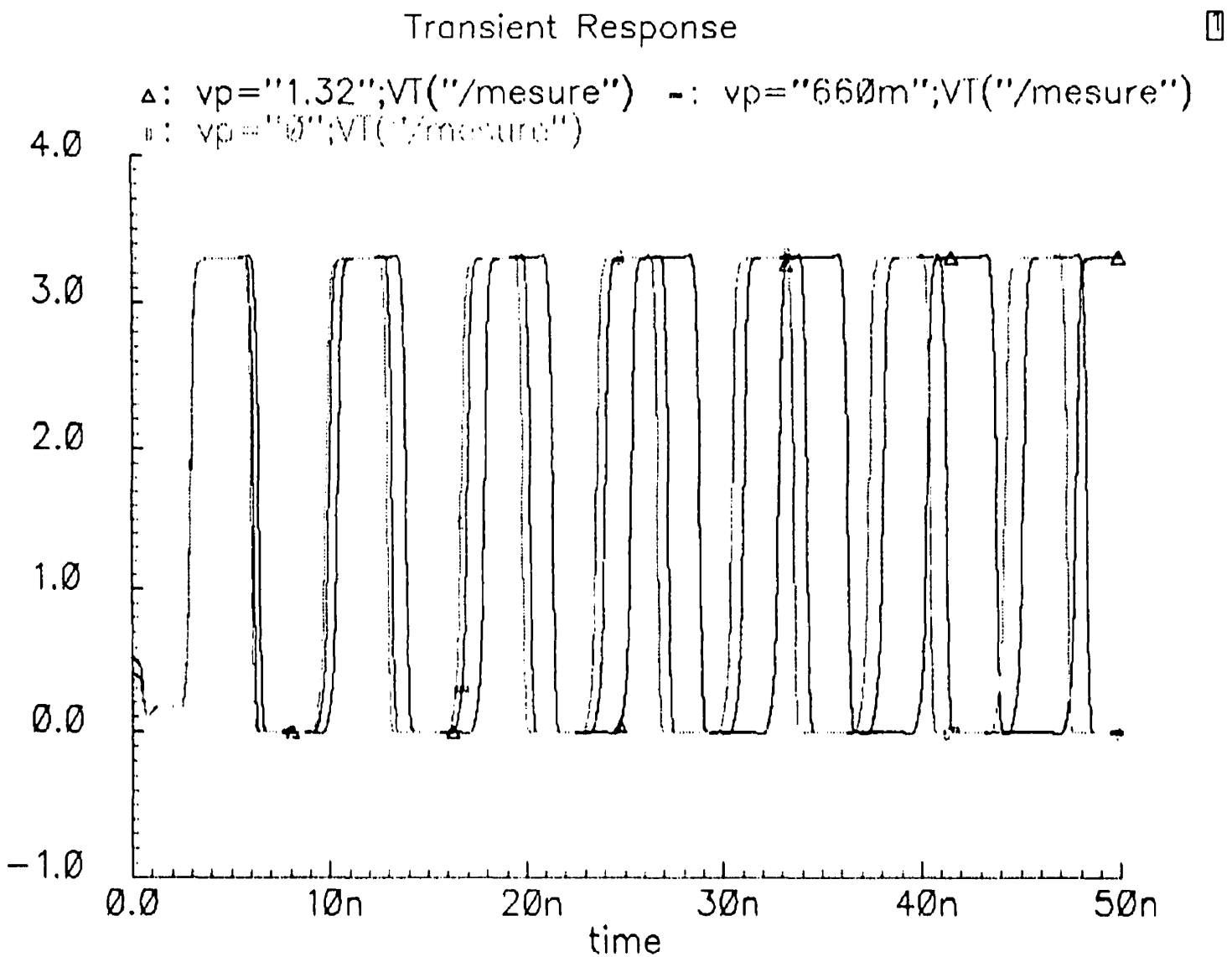

Transient Response
: frequency(VT("/mesure"))
370M
350M
330M
310M
290M
270M
250M
230M
0.0
1.0
2.0

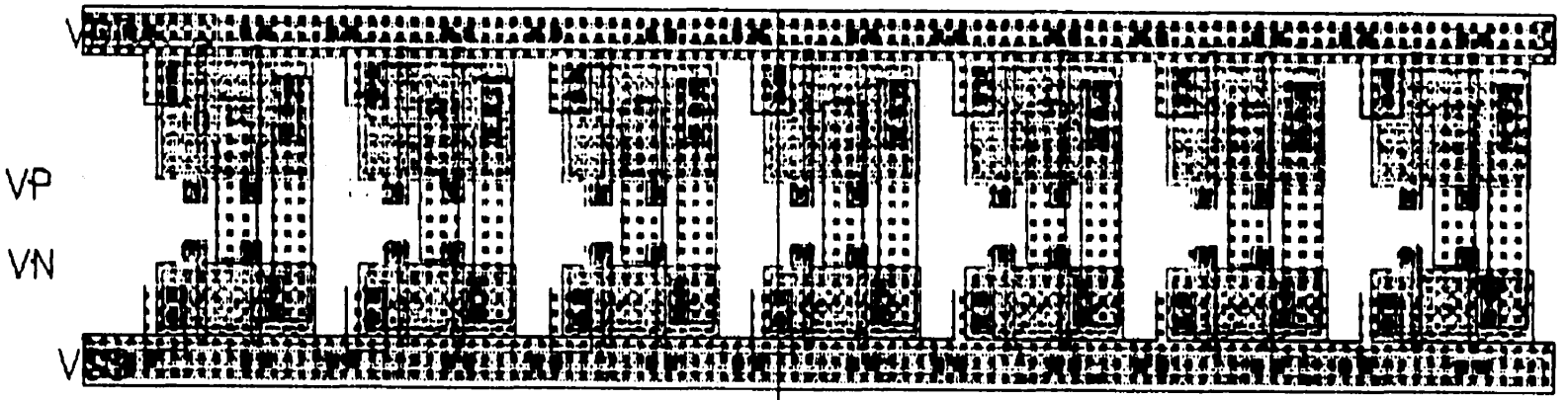

Transient Response
▵: vp="1.32";VT("/mesure")
▫: vp="660m";VT("/mesure")
4
3
2
1
Ø
−1
16n
21n
26n
31n
time

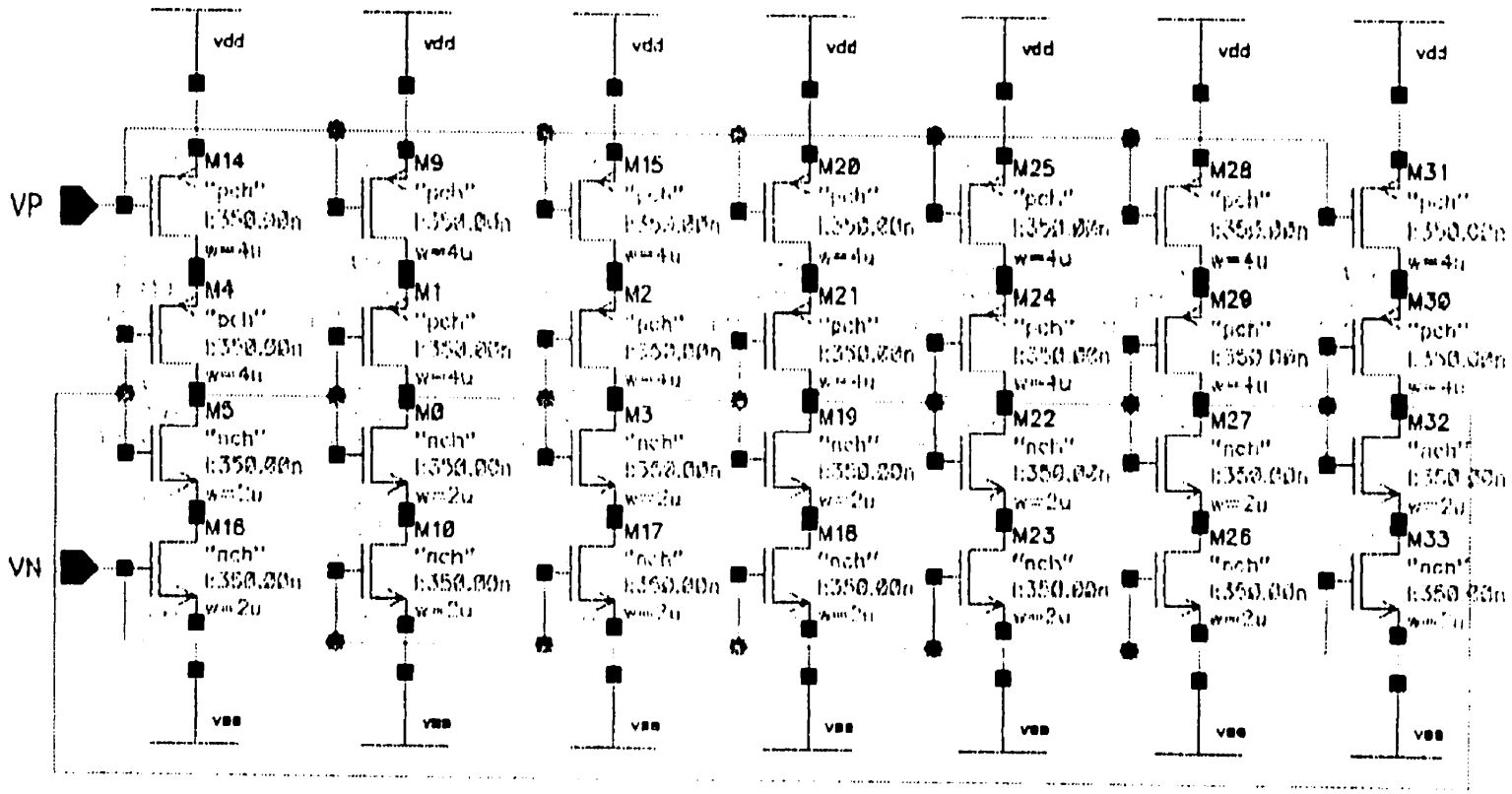

vdd
vdd
vdd
VP
VN
M14
"pch"
l:350.00n
w=4u
M4
"pch"
l:350.00n
w=4u
M5
"nch"
l:350.00n
w=2u
M16
"nch"
l:350.00n
w=2u
M9
"pch"
l:350.00n
w=4u
M1
"pch"
l:350.00n
w=4u
MØ
"nch"
l:350.00n
w=2u
M1Ø
"nch"
l:350.00n
w=2u
M15
"pch"
l:350.00n
w=4u
M2
"pch"
l:350.00n
w=4u
M3
"nch"
l:350.00n
w=2u
M17
"nch"
l:350.00n
w=2u
vss
vss
vss


Transient Response
Δ: vp="1.32";VT("/mesure")
-: vp="660m";VT("/mesure")
4
3
2
1
Ø
-1
17n
21n
25n
29n
33n
time


Transient Response
: frequency(VT("/mesure"))
150M
130M
110M
90.0M
0.0
1.0
2.0


Transient Response
△: vp="1.32";VT("/mesure")
-: vp="660m";VT("/mesure")
▫: vp="0";VT("/mesure")
4.0
3.0
2.0
1.0
0.0
-1.0
0.0
10n
20n
30n
40n
50n
time

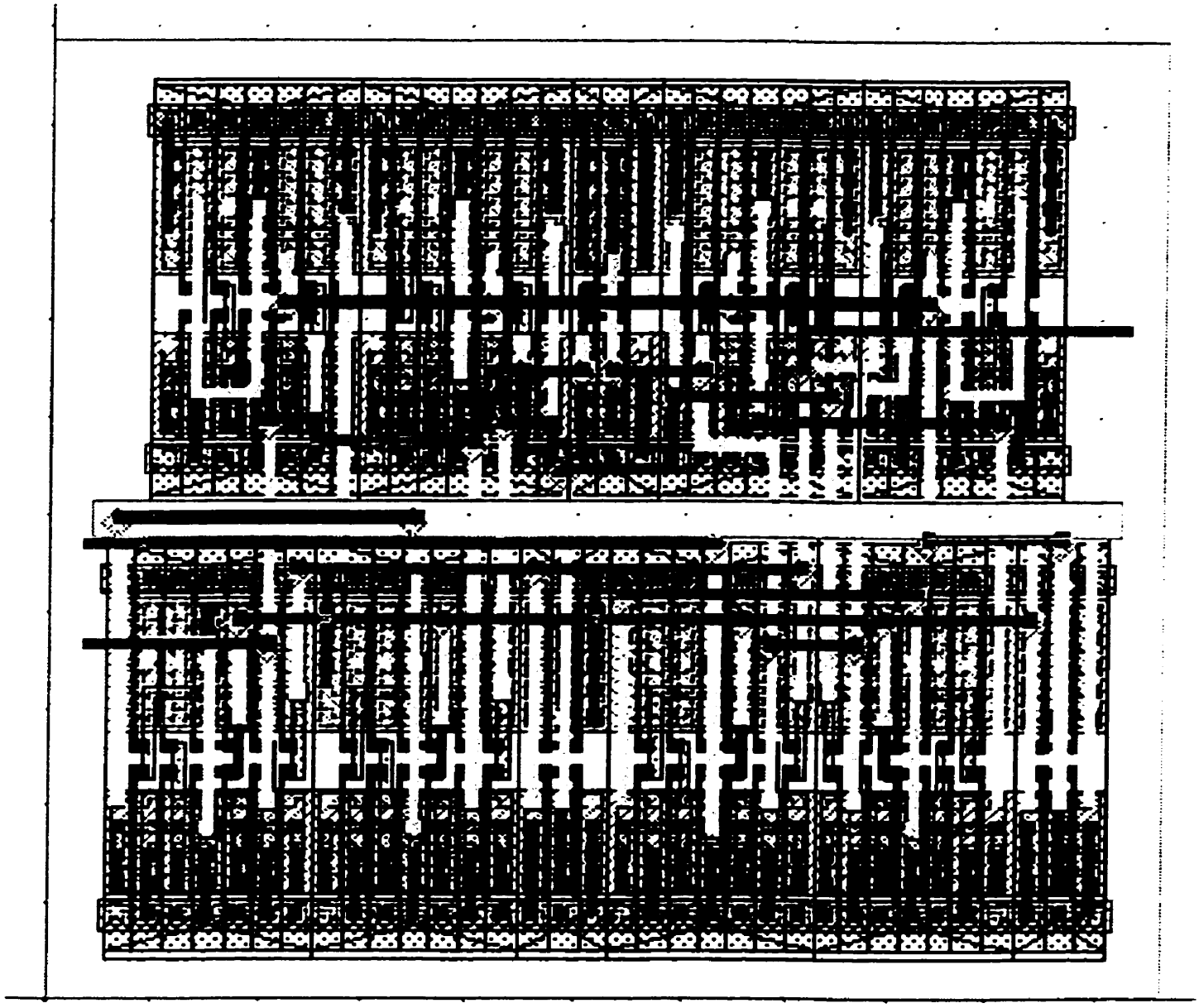

VP
VN

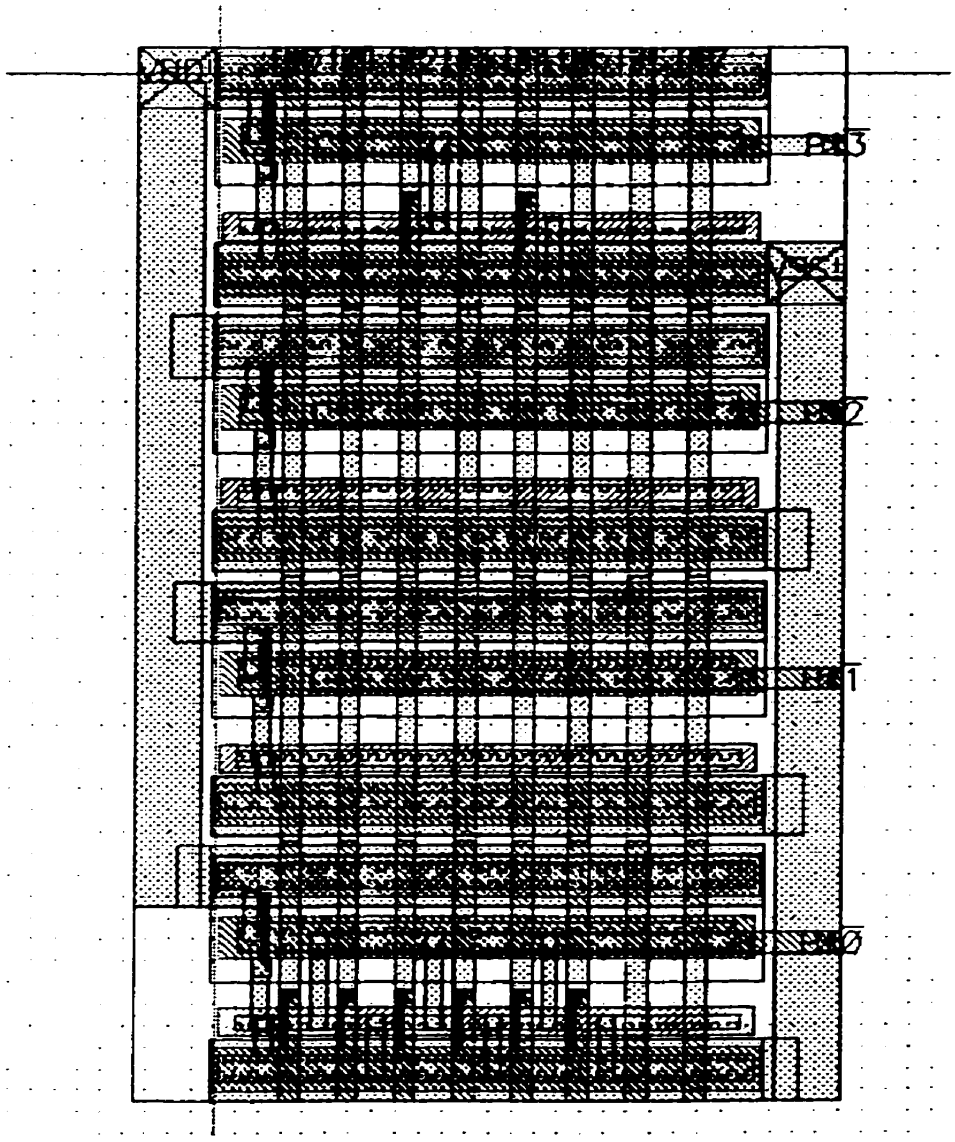

vdd
VP
VN
M14
M4
M5
M16
M9
M1
M0
M10
M15
M2
M3
M17
M20
M21
M19
M18
M25
M24
M22
M23
M28
M29
M27
M26
M31
M30
M32
M33
"pch"
"nch"
w=4u
w=2u
vss

ANNEXE 3

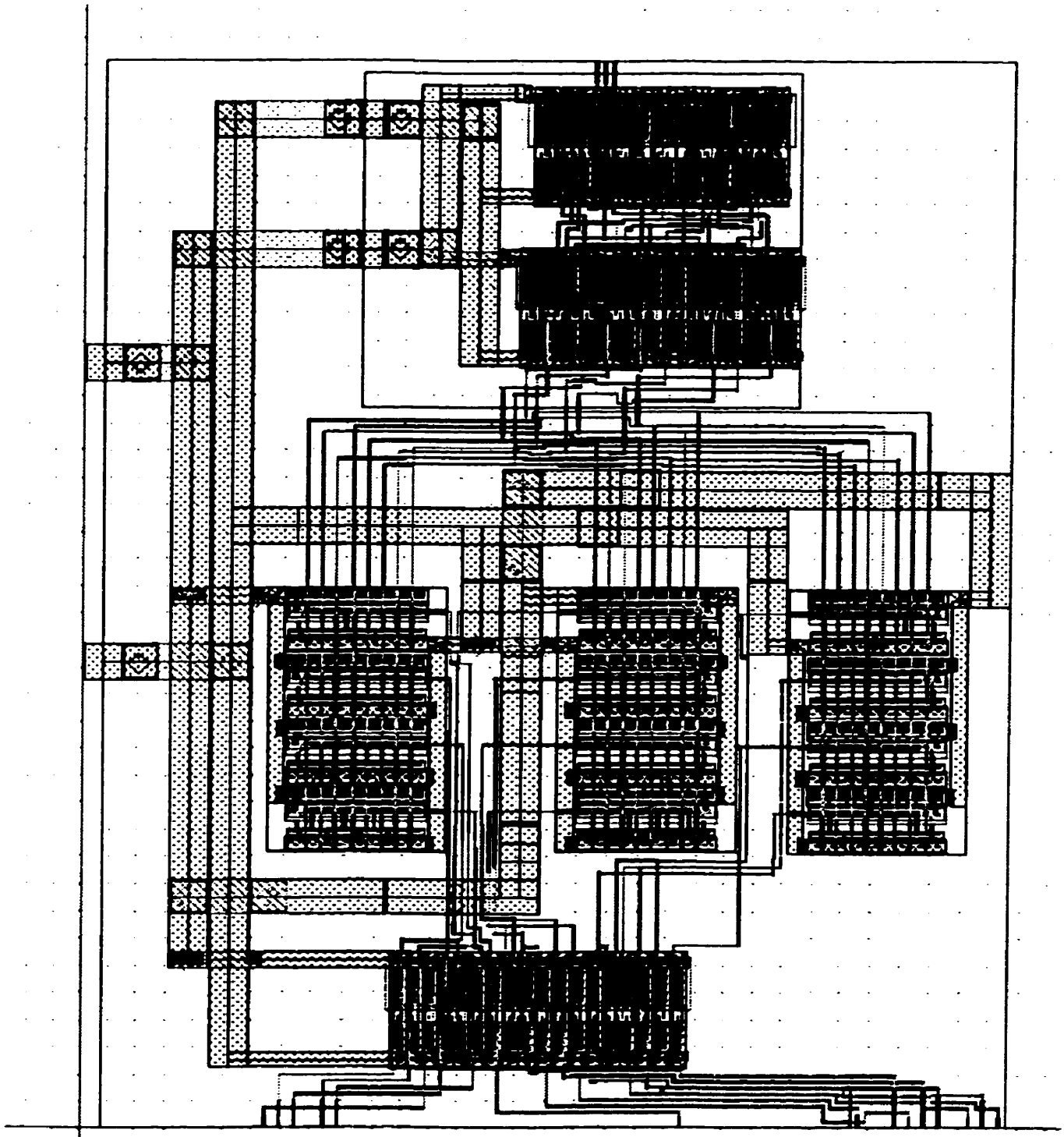
Dessins de masques

layout du décodeur 3_8





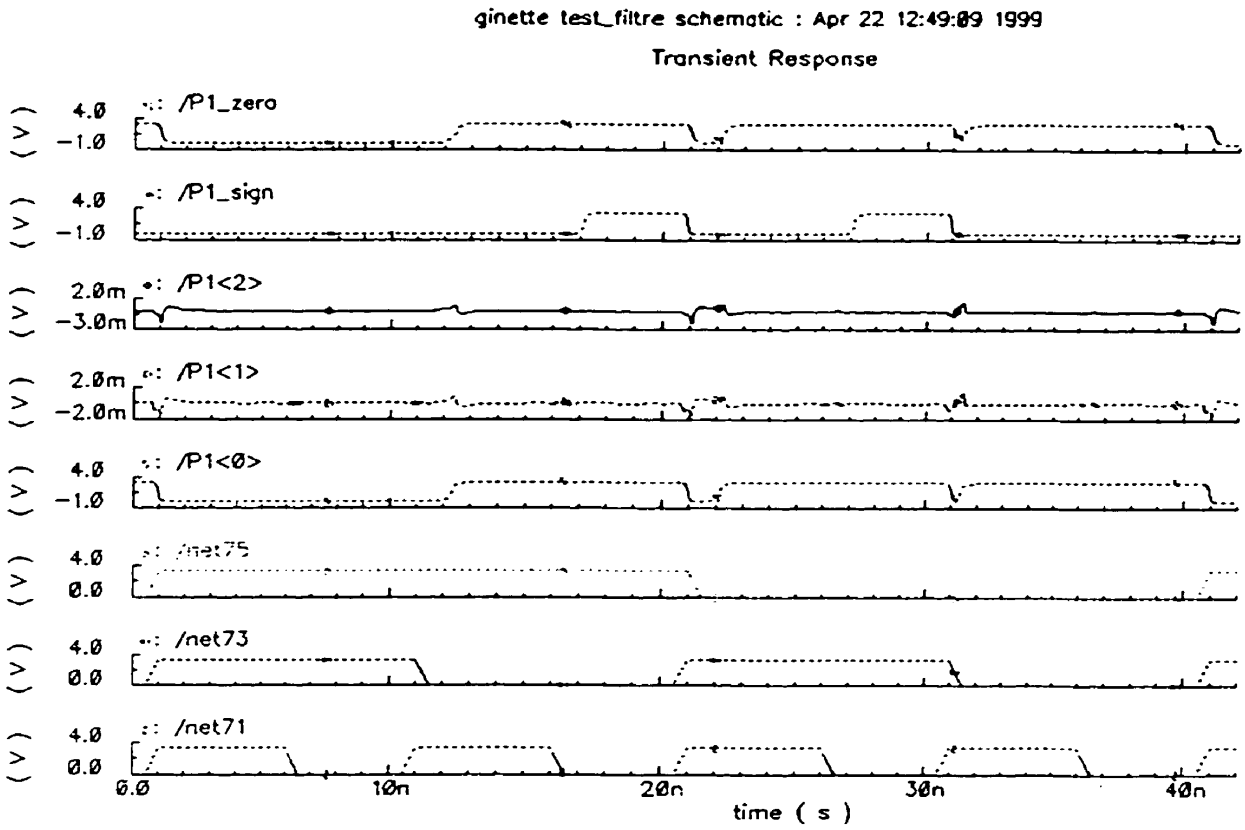
mémoire pour un pixel

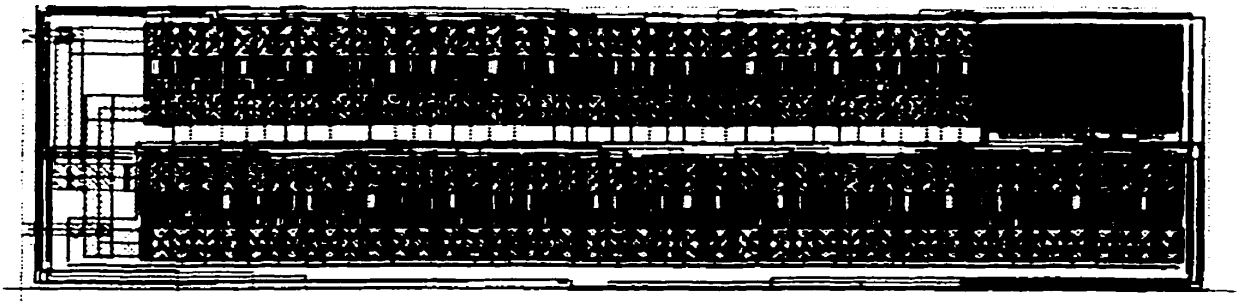


filtre pour 3 pixels, incluant le décodeur3-8 (en haut) et les inverseurs et logique statique de sortie (en bas).

Simulation du filtre de la page précédente.

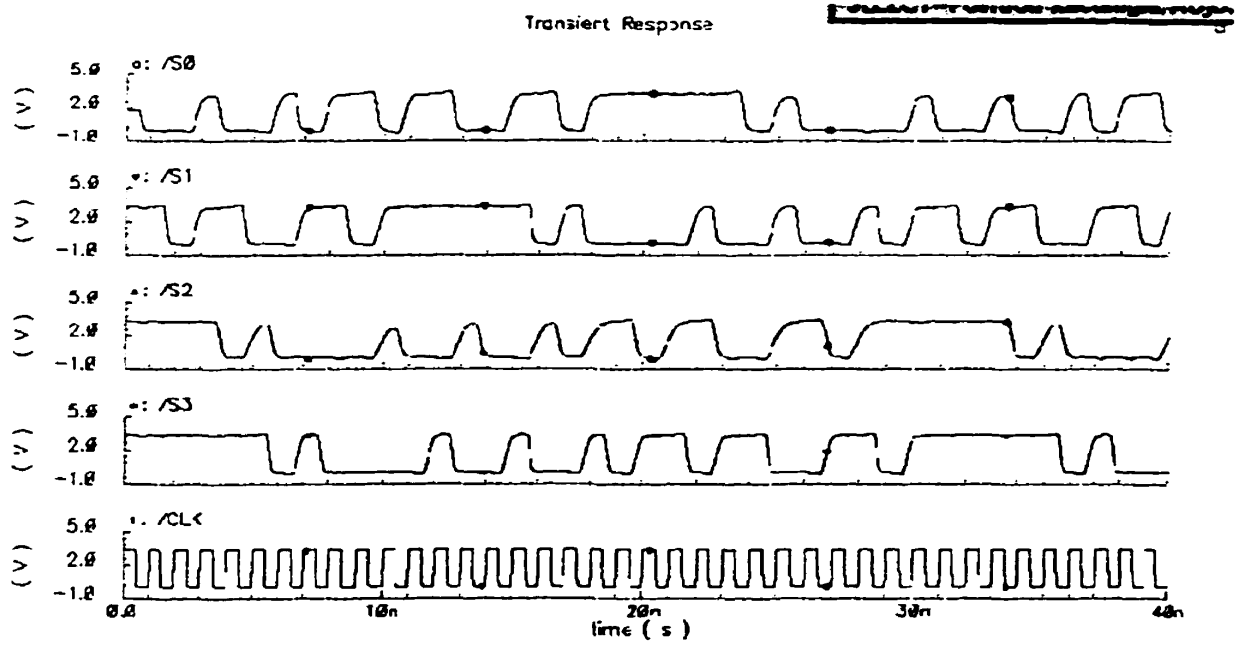
On montre les résultats pour les 4 bits du pixel 1.

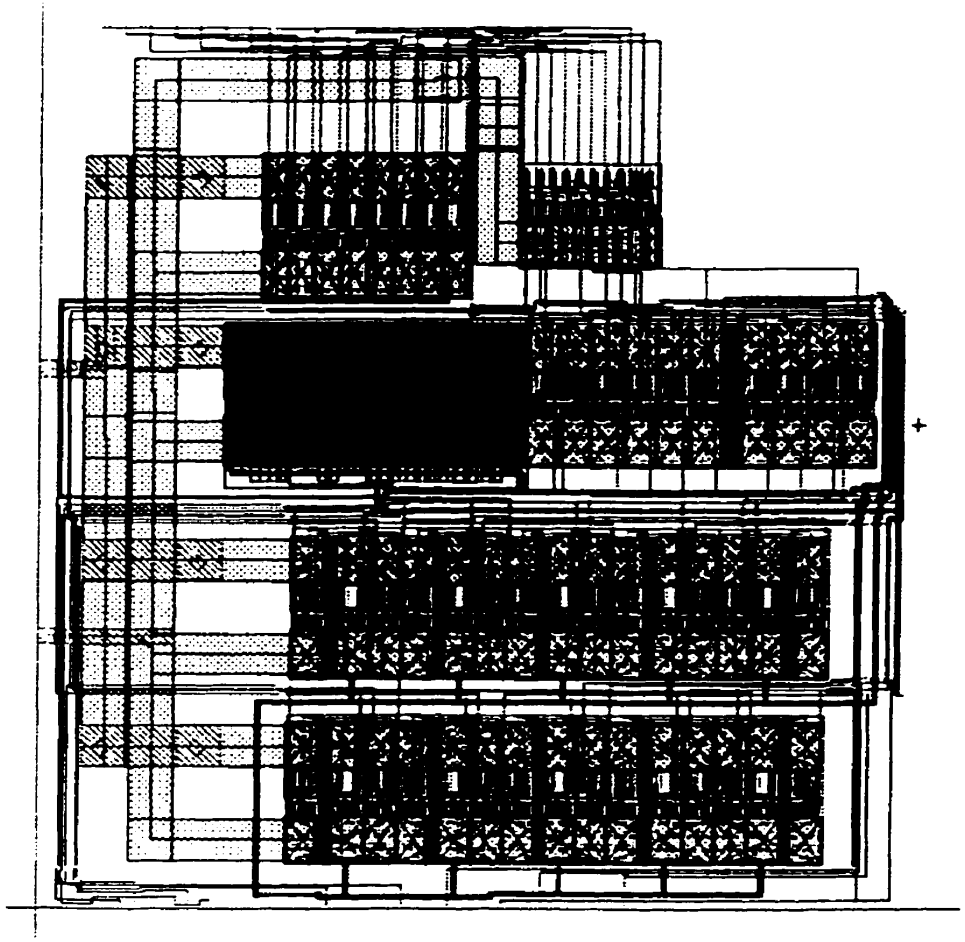




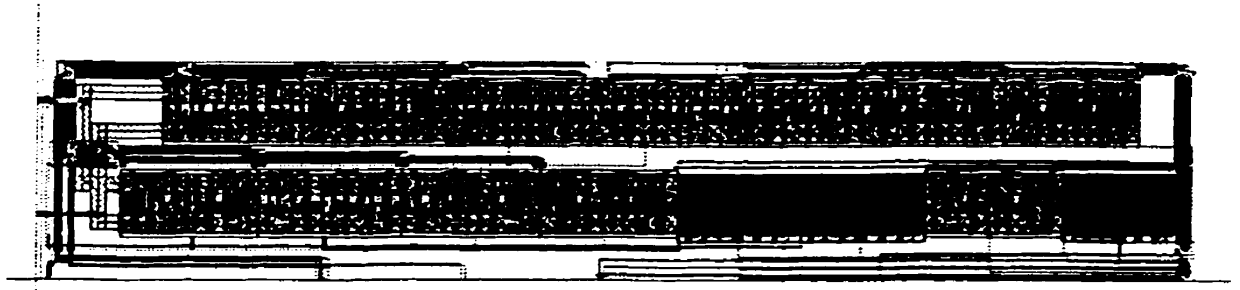
Générateur de vecteur pseudo-aléatoires de 13 bits

Simulation d'un générateur 4 bits (à partir de la vue extracted)

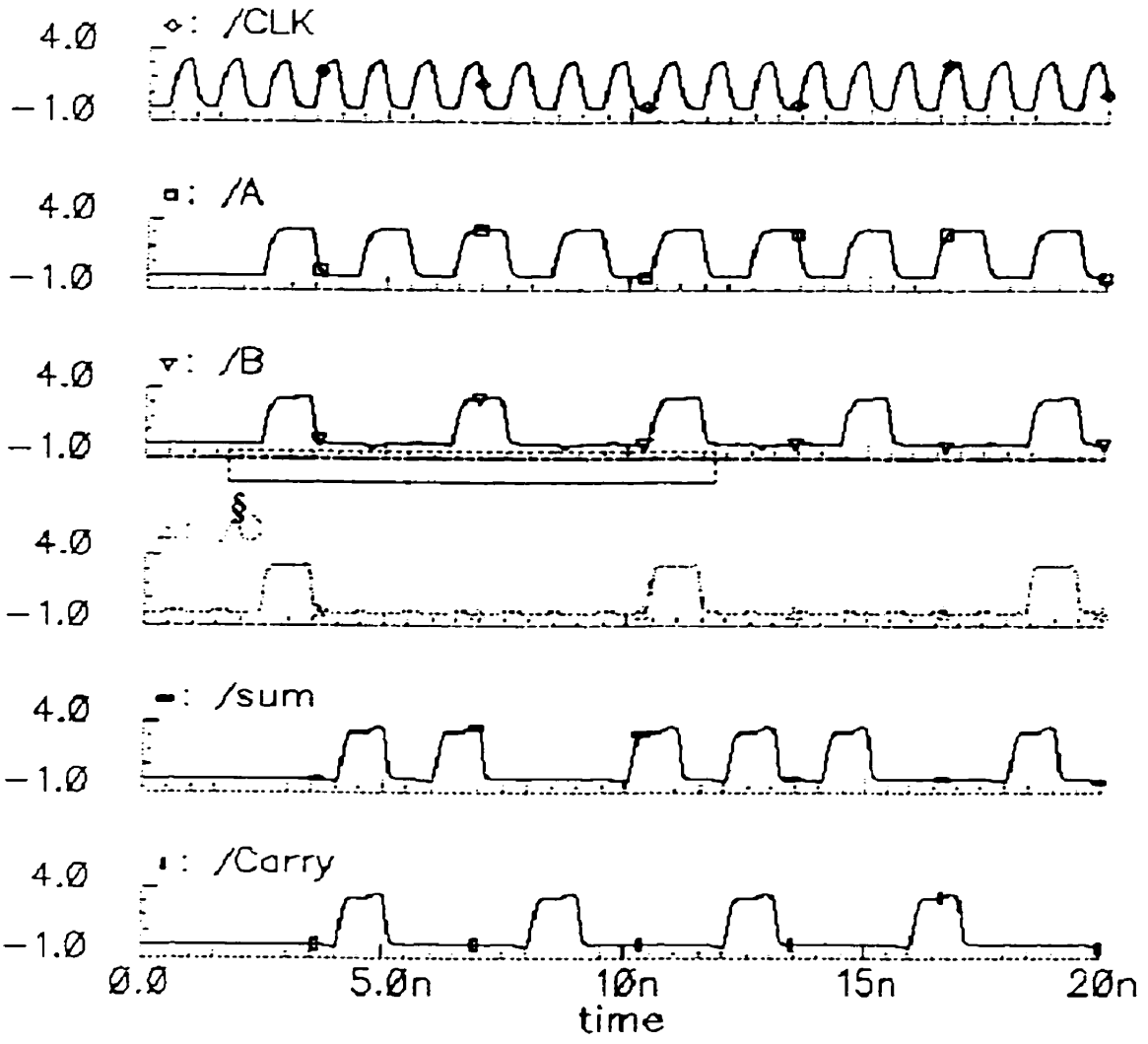




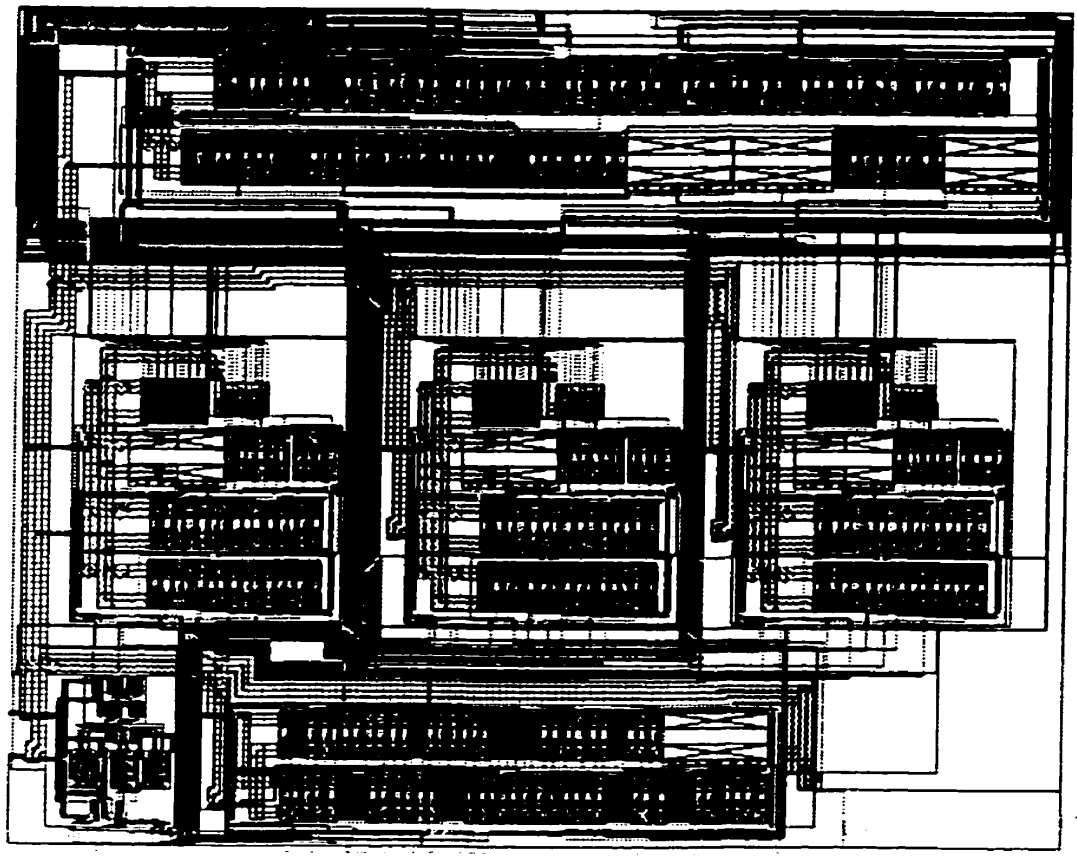
Multiplieur restreint



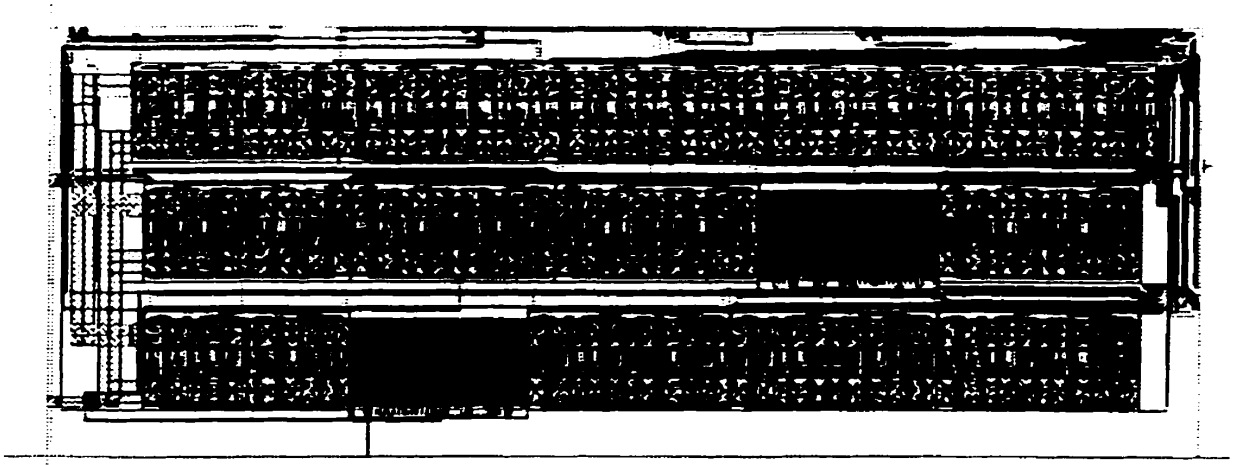
Additionneur 12 bits (FullAdd12)



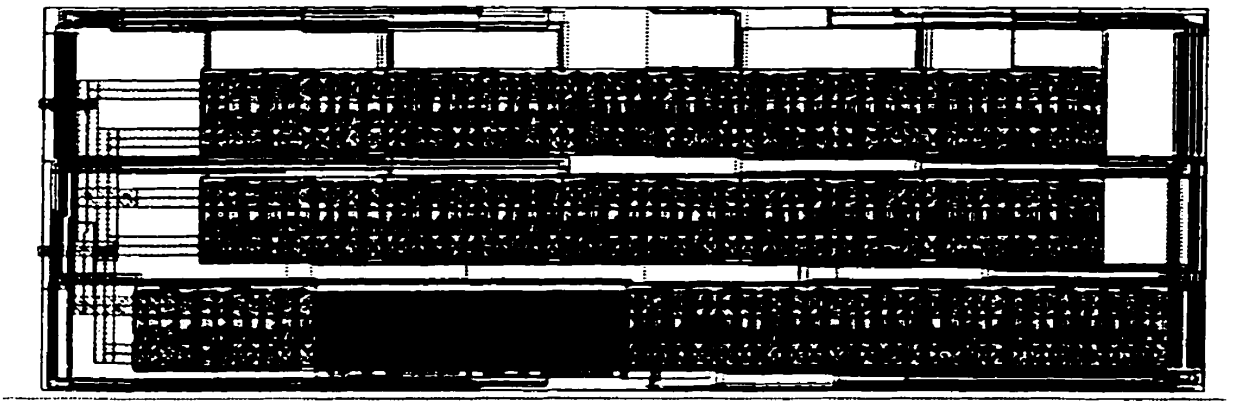
Simulation de l'additionneur CSA 1 bit.



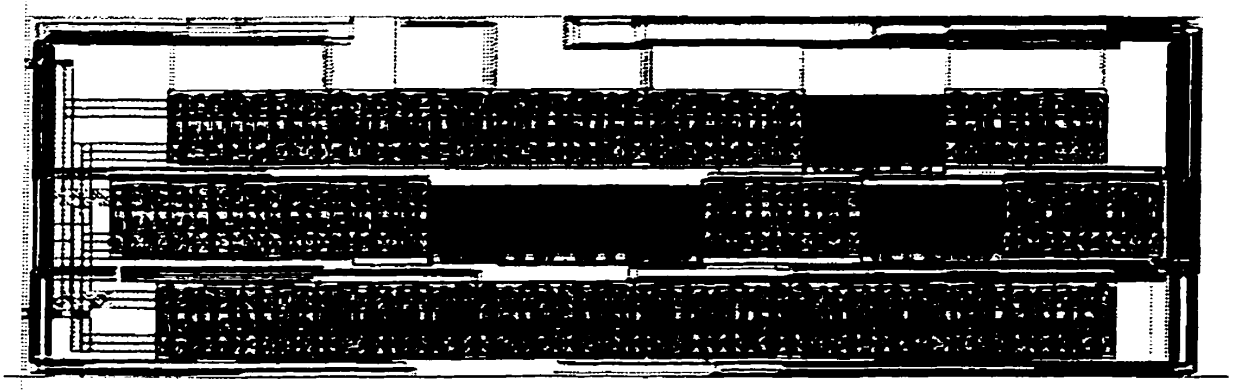
Coin Incluant filtre pour 3 pixels, un générateur 13 bits, 3 multiplieurs et un additionneur 12 bits.



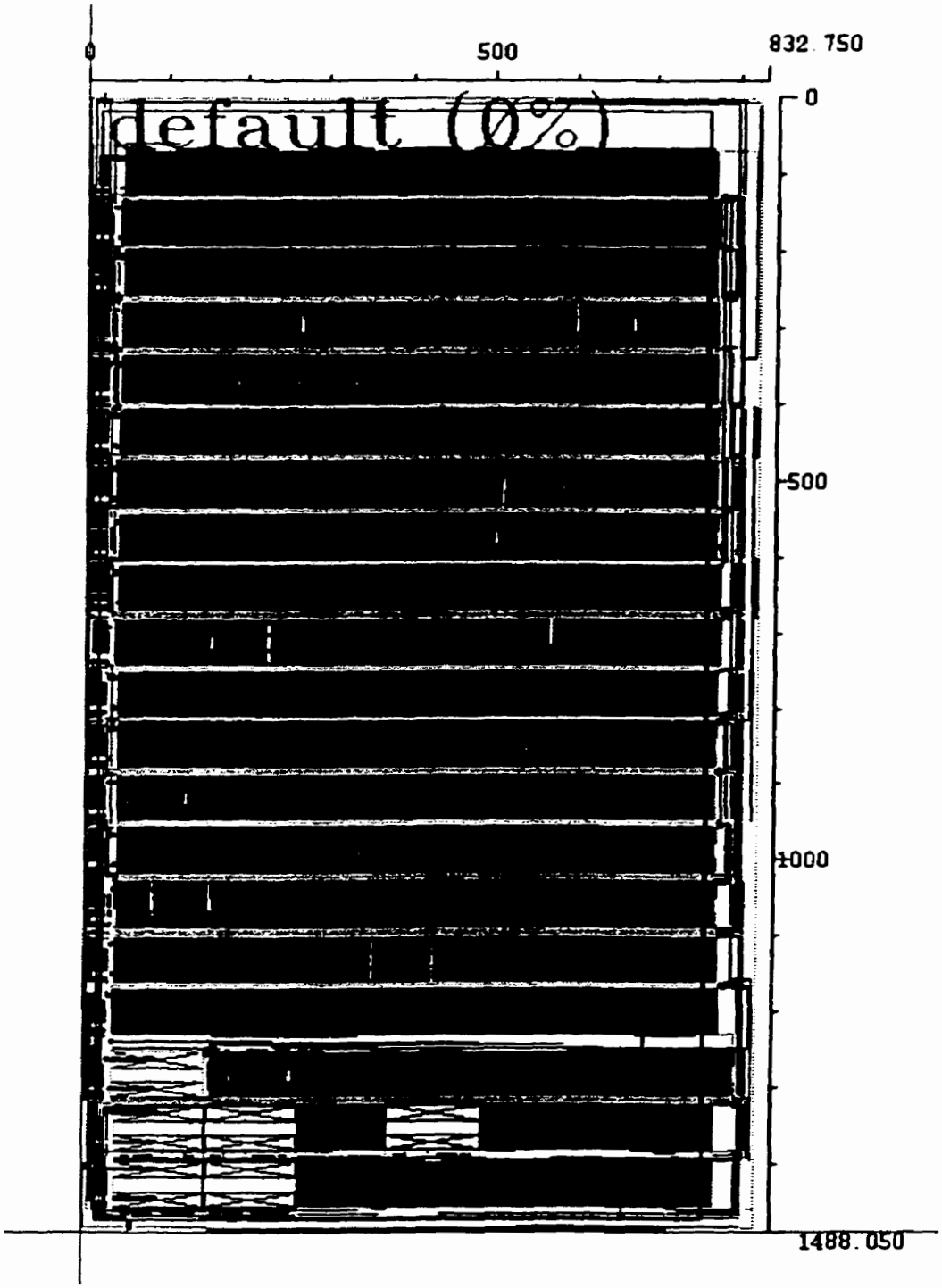
Additionneur 13 bits (FullAdd13)



Additionneur 14 bits (FullAdd13)

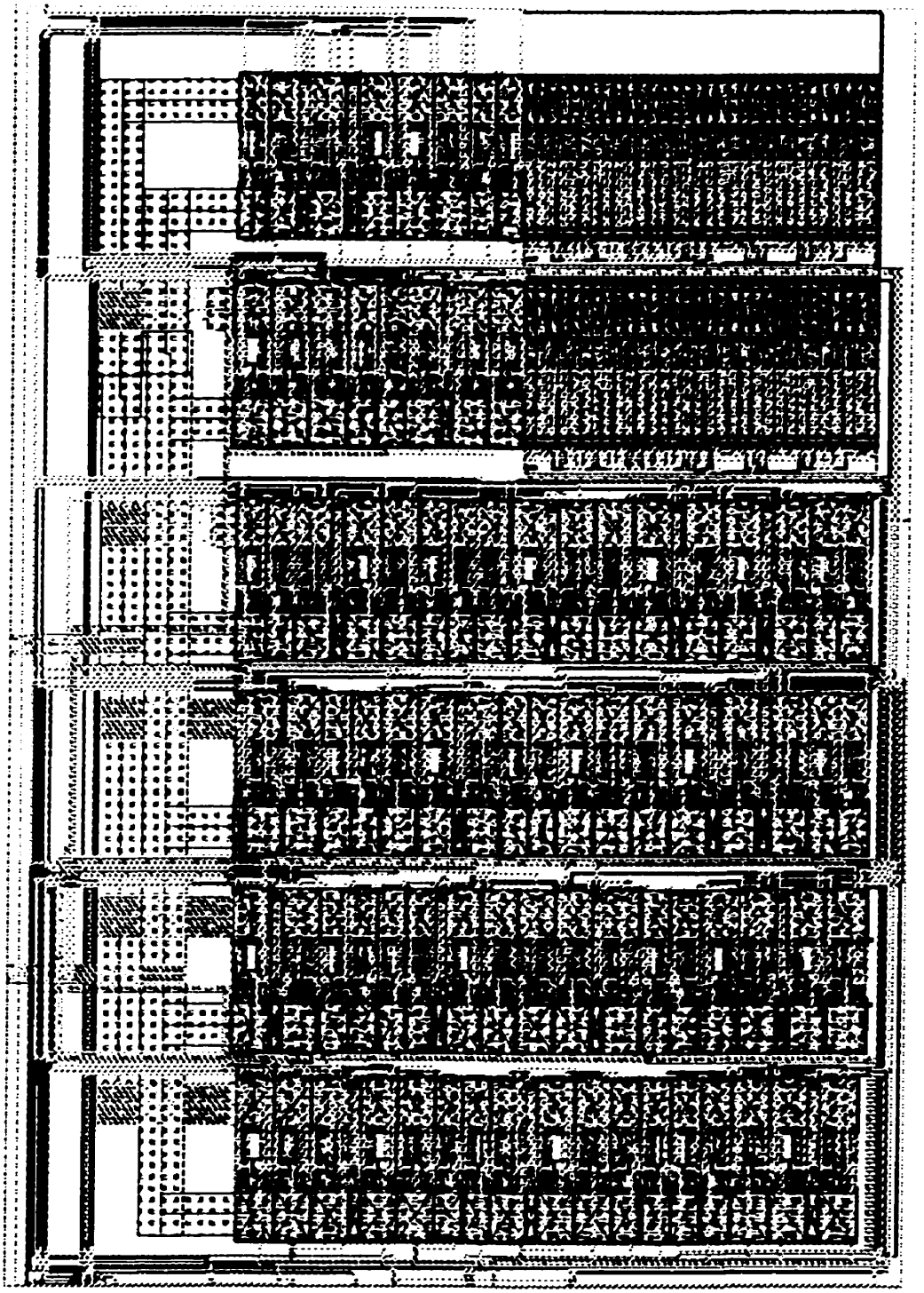


Additionneur 15 bits (FullAdd15)

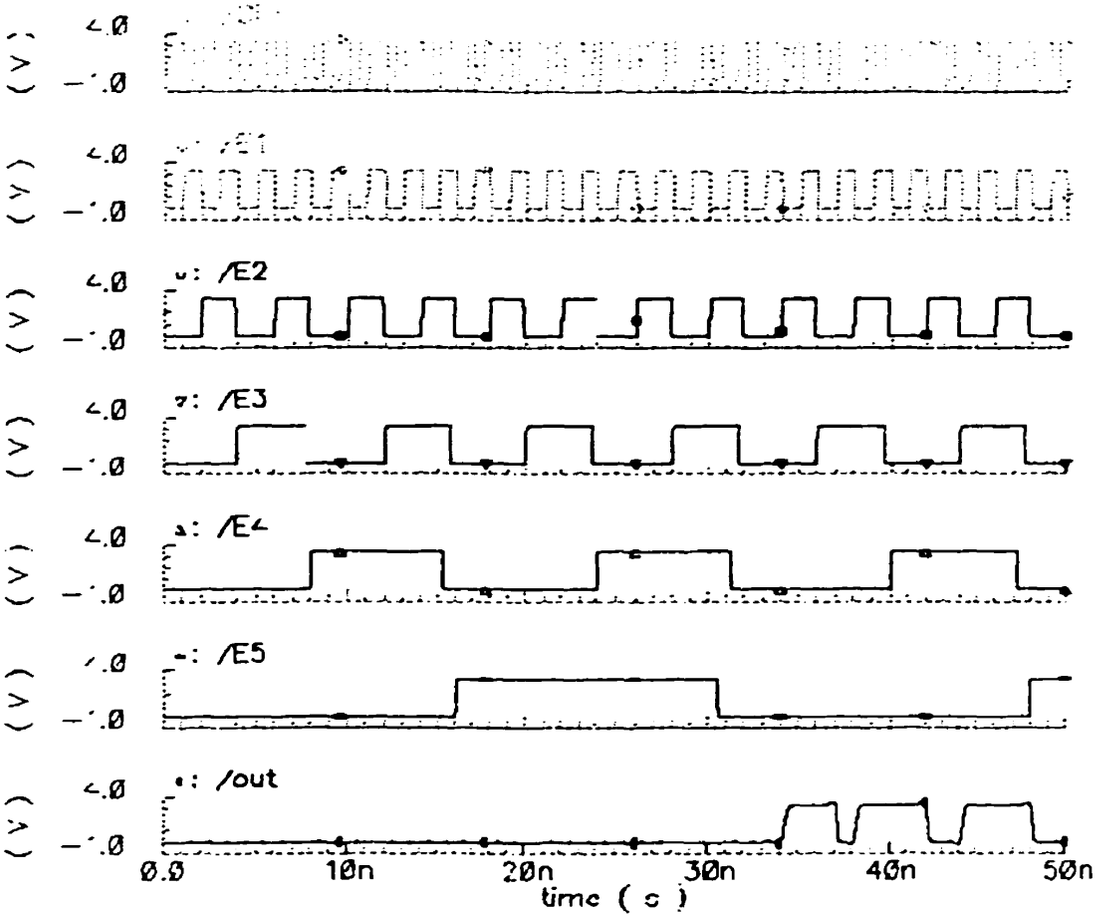


Additionneur en cascades 15 bits.

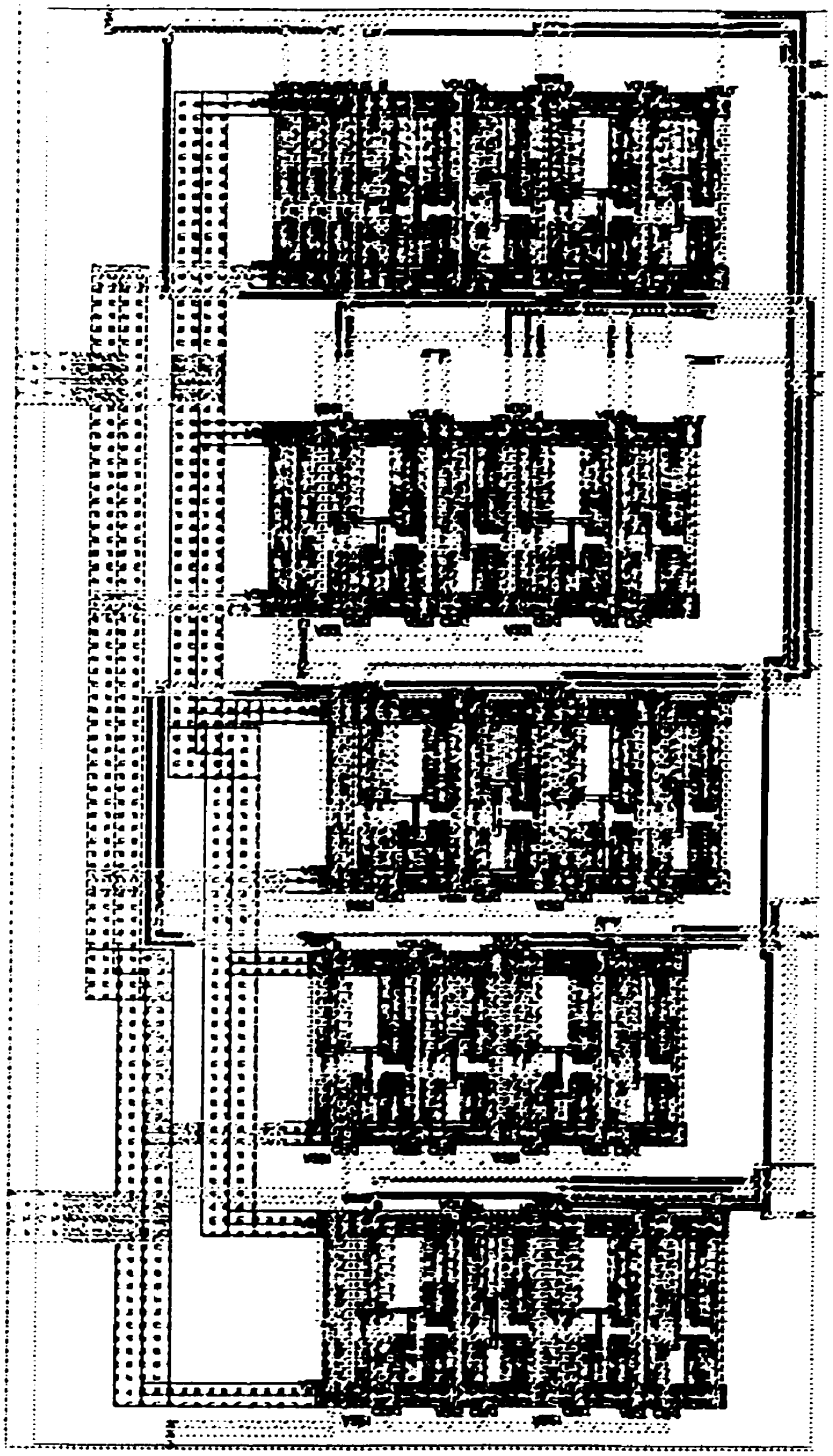
On note les dimensions imposantes de ce bloc.



Analyseur de signatures

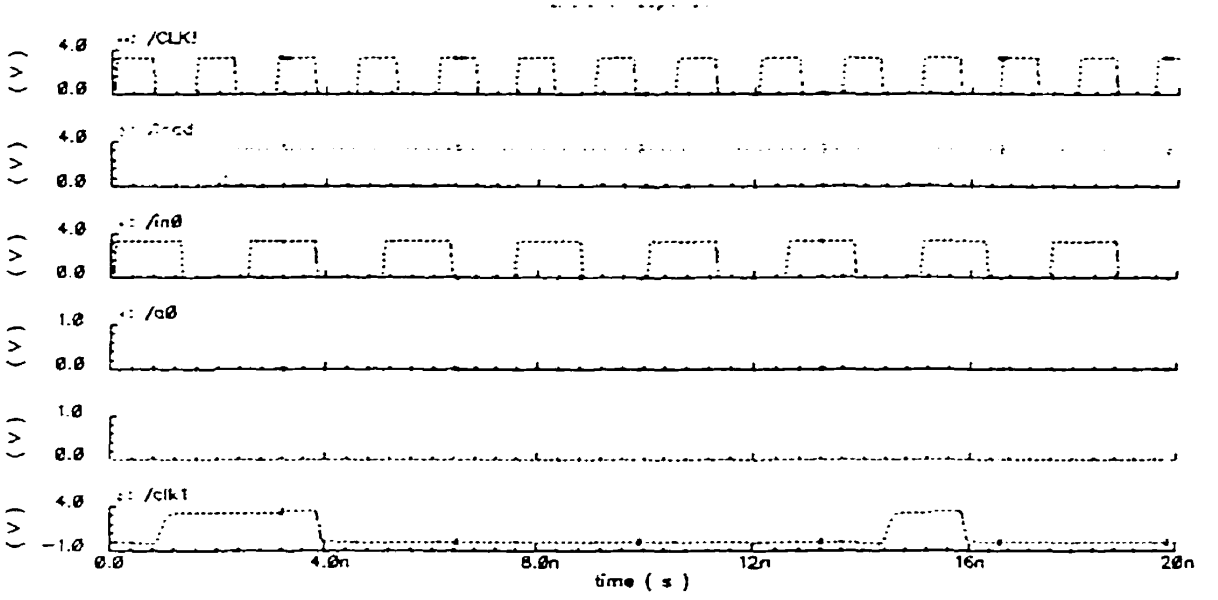


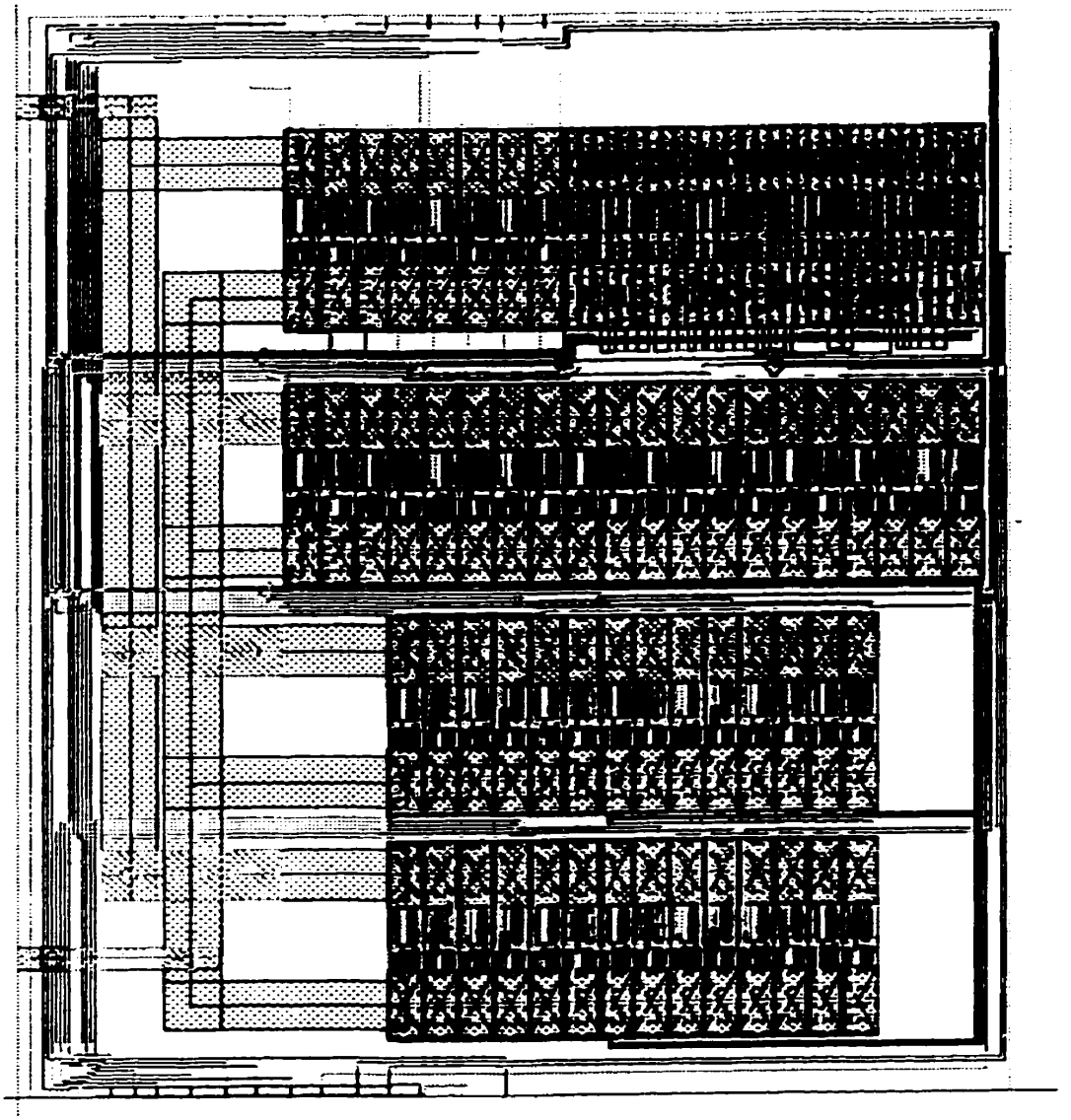
Simulation de l'analyseur de signature



Démultiplexeur

Simulation du démultiplexeur





Registre a décalage de 14 bits

Simulation du registre a décalage d'un bit