Ecole Nationale Polytechnique
Département d'Electronique

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

◆◆◆

Projet de Fin d'Etudes

INGENIORAT D'ETAT EN ELECTRONIQUE

THEME

CAO d'ASICs
Application à la Conception d'un Multiplieur et d'un Diviseur

Proposé et suivi par :

Mr A. FARAH

Etudié par :

Mr M. OUMEDDOUR
Mr M. EL - ALLIA

Juin 1994

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

ملخص :

هذه المذكرة تشرح وتبين طرق إنجاز دارتين متكاملتين CMOS من نوع ASIC دارة متكاملة ذات تطبيق خاص). حيث تعتمد بنية دارتي الضرب والقسمة على خوارزميتي بوغ-وولي وكابا-هاماشر المحسن على التوالي بإدماج شبكة المحافظة على الباقي.

الهدف الرئيسي من هذا التصميم هو إنجاز دارة ضرب ودارة قسمة متكاملتين يمكن إدماجهما في حاسوب خاص بمعالجة الإشارة.

نتائج الإنجاز هي دارة ضرب متكاملة 16x16 بايت (BITS) ودارة قسمة متكاملة 8/16 بايت للأعداد الموجبة، السالبة أو هما معا (مختلطة).

Abstract:

This memoir describes the design and analysis of an ASIC CMOS multiplier and divider. The design is based on the well known Baugh-Wooley algorithm and modified Cappa-Hammacher algorithm, combined to a carry-save array. The main objective of the design was to provide a multiplier and a divider that could be used efficiently in the design of digital signals processors, using CAD tools.

The results were a 16x16 bits multiplier and a 16/8 bits divider which can manipulate a signed, unsigned, two's complement and mixed operands.

Résumé:

Cet mémoire décrit la conception et la réalisation d'un multiplieur et d'un diviseur CMOS de type ASIC. L'architecture du multiplieur et du diviseur est basée respectivement sur l'algorithme de Baugh-Wooley et Cappa-Hammacher modifié, combinés à un réseau carry-save. L'objectif principal de la conception etait de réaliser un multiplieur pouvant être efficacement integre dans un processeur de traitement de signal.

Les résultats de la réalisation sont un multiplieur 16x16 bits et un diviseur 16/8 bits pour des opérandes non signés, complément à deux ou mixte.

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

DEDICACES

Je dédie ce modeste travail à:

- Mon père
- Ma mère
- Mes frères
- Ma soeur
- Tous les amis

Mourad EL-ALLIA

Je dédie ce modeste travail à:

- Ma mère
- Mon père
- Mes frères et soeurs
- Et surtout mes neveux, la petite Amina, et le petit Abderrahmane
- Enfin à tous mes amis

OUMEDDOUR Messaoud

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

REMERCIEMENTS

Nos remerciements vont spécialement à Monsieur A.FARAH, Maître de conférence à l'Ecole Nationale Polytechnique et promoteur de notre thèse, nous lui exprimons notre profonde reconnaissance pour son aide chaleureuse, ses conseils et sa compréhension qui nous ont orientés et soutenu durant toute la période d'élaboration de cette thèse.

Nous tenons aussi à remercier:
Melle GUENDOUZ pour l'aide précieuse et constante qui nous a prodigué pour la réalisation de ce travail.

Mr A.NASRI, pour sa disponibilité à nous aider pendant notre présence au laboratoire "Techniques Digitales et Systèmes".

Enfin, nous remercions tous ceux qui nous aidés de près ou de loin à l'accomplissement de ce travail.

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

SOMMAIRE

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

المدرسة الوطنية المتعددة التقنيات
BIBLIOTHEQUE — المكتبة
Ecole Nationale Polytechnique

INTRODUCTION

Le but de ce projet est la conception d'un multiplieur et d'un diviseur CMOS. Ceux-ci doivent être conçus à l'aide d'outils de CAO, sous forme d'ASIC pour un traitement numérique des signaux.La CAO (Conception Assistée par Ordinateur) est un ensemble d'outils logiciels et matériels se présentant sous forme de stations de travail et permettant de faciliter la conception de circuits. Ces stations sont maintenant remplacées par des micro ordinateurs de type PC ou MAC. Elles sont généralement, même les moins onéreuses, suffisantes pour faire la conception logique la saisie de schémas jusqu'à la simulation logique. Les outils plus récents, permettent de traiter en local toute la conception jusqu'à la bande de fabrication, celle se fait chez un fondeur de silicium.Ce travail de conception se scinde en huit chapitres.Le chapitre 1 rappelle les notions théoriques de la technologie CMOS, en traitant l'inverseur de base de cette famille, son temps de commutation, le compromis espace-temps ainsi que les avantages de la famille CMOS. Par la suite on présente les réseaux logiques programmables (PLA) et leurs évolutions vers les ASICs. Une étude détaillée sur les ASICs conclut le chapitre.Les chapitres 2 et 3 abordent les différents types de multiplieurs séries et parallèles ainsi que leurs arbres de sommations, génération des produits partiels et les performances de chaque algorithme présenté. Deux types de diviseurs pour un environnement VLSI, celui de Guild et de Cappa-Hamacher modifié sont traités au chapitre 3.Le chapitre 4 présente une réflexion sur les contraintes de conception et leurs impacts sur le choix de l'algorithme de multiplication ou de division, influencé directement par trois facteurs indépendants: régularité, vitesse et surface. On décrit également le choix de l'algorithme de multiplication et de division. En effet, notre choix s'étant porté sur l'algorithme de (Baugh-Wooley) combiné à un réseau "Carry-Save". Dans ce chapitre on va déterminer les différents éléments qui constituent le multiplieur et le diviseur. On va essayer également d'apporter une réponse à deux problèmes importants: l'extension du signe et la propagation de la retenue dans l'accumulateur.
Le chapitre 5 est consacré à la présentation de l'outil informatique proposé pour la conception du multiplieur et du diviseur, cet outil étant le MCE (Micro Circuit Engineering). On présentera également la méthodologie à suivre selon le logiciel MCE pour élaborer un circuit intégré (ASIC) ainsi que les tests qui s'en suivent.
Le chapitre 6 décrit les étapes d'implémentation des circuits à l'aide du logiciel MCE et les performances de chacun d'eux.Enfin, on terminera notre projet de fin d'étude par une conclusion.

Spécifications du travail réalisé

- Le multiplieur est conçu dans l'optique d'une intégration de type ASIC.
- Afin de rendre l'unité de traitement du processeur la plus performante possible, l'architecture du multiplieur et du diviseur choisie est une architecture parallèle.
- Le nombre de bits sur lequel doit se faire la multiplication est de 16 fois 16 bits, celui de la division est de 16 par 8 bits.
- La nature des opérandes manipulée n'étant pas spécifiée au départ, le multiplieur ainsi que le diviseur doivent être capable de traiter aussi bien des nombres non-signés que des nombres complément à 2, ainsi que la combinaison de ces deux notations. Cette particularité rend l'utilisation du multiplieur et du diviseur plus souple.
- Parce que le multiplieur est déstiné à être inséré dans un circuit plus grand, on s'efforcera de trouver en plus de la régularité, un compromis vitesse/surface. Il ne sert à rien de concevoir un multiplieur ou un diviseur ultra rapide si celui-ci occupe trop de place sur le silicium.

CHAPITRE I

CIRCUITS INTEGRES ET ASICs

1.1. Introduction

Nous commençons ce chapitre par un rappel des propriétés de base du transistor à effet de champ (FET), à canal N, en métal-oxyde-semiconducteur (MOS). Nous décrirons ensuite les principaux circuits résultants de l'interconnexion de transistors a effet de champ MOS.Les systemes intégrés (ASICs) en technologie CMOS comportent trois niveaux de matériaux conducteurs, séparés par des couches de matériaux isolants.En partant du haut vers le bas, ces niveaux sont appelés:La métallisation, le silicium poly-cristalin et la diffusion. Le dessin des liaisons à ces trois niveaux, ainsi que la localisation des points de contact dans le matériau isolant qui permettent de relier des liaisons de niveaux différents, sont matérialisés pendant le processus de fabrication, à l'aide de masques semblables aux négatifs photographiques.En l'absence de point de contact à travers le matériau isolant, les liaisons du niveau métallisation peuvent croiser des liaisons en poly-cristalin ou des liaisons du niveau de diffusion sans interférence significative. Par contre, chaque fois qu'une liaison en poly-cristalin croise une liaison de diffusion, un transistor est créé. Ce transistor a les caractéristique d'un interrupteur simple, pour lequel la tension de la liaison en poly-cristalin régule l'intensité du courant de la liaison de diffusion. Les circuits constitués de ces transistors, reliés par des liaisons dessinées aux trois niveaux, constituent les éléments de base de nos assemblages. Les systèmes intégrés sont concus à l'aide de ces circuits de base. Ces systèmes seront fabriqués à la surface des cristaux de silicium de structure monocristaline.

1.2. L'Inverseur de base

Le premier circuit logique que nous décrivons est l'inverseur digital de base. L'étude de ce circuit est ensuite étendue à celle des portes logiques NAND, AND et NOR. La fonction logique de l'inverseur est de générer une sortie qui soit le complément de son entrée. Le schéma de l'inverseur de base est donné en (Figure 2.1):

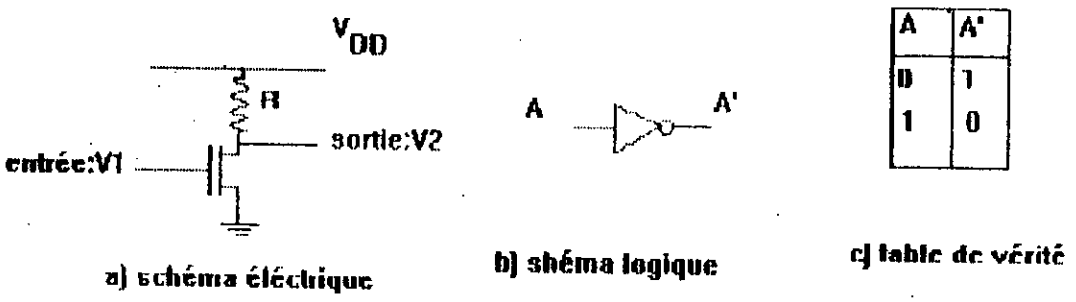


figure 2.1:Inverseur de base

Le transistor est bloqué lorsque la tension d'entrée V1 de l'inverseur est inférieure au seuil Vseuil du transistor. La tension de sortie est alors le complément de l'entrée. Si V1 est supérieur à Vseuil, le transistor est passant et le courant s'écoule de l'alimentation à la masse, à travers la résistance R. Si R est suffisamment importante, V2 peut être rappelée vers la masse bien en dessous de Vseuil, afin de complémenter à nouveau l'entrée.II-1 Temps de commutation d'un inverseur:Une contrainte obligatoire pour un inverseur est de pouvoir attaquer un élément identique à lui-même. Des inverseurs connectés en chaînes sont représentés sur la (Figure 2.2):

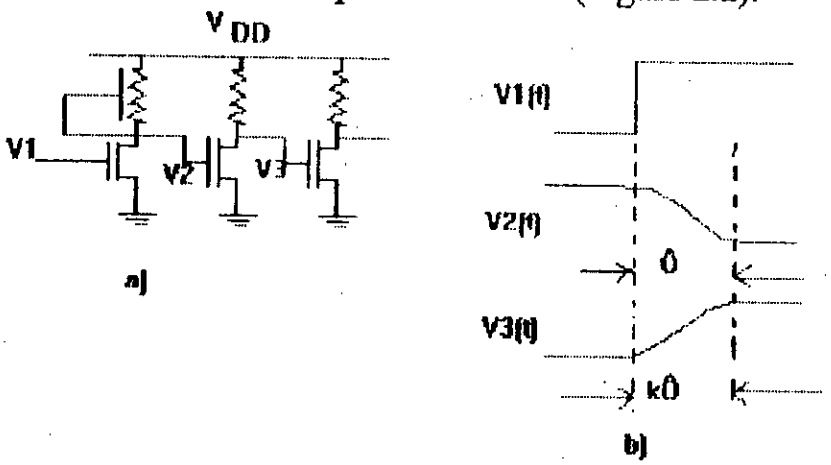


figure 2.2: temps de commutation de l'inverseur

Temps de commutation de l'inverseur

Nous appellerons rapport (k) de l'inverseur le rapport Zcharge/Zsignal, où Zcharge est l'impédance de la charge du transistor.Zsignal est l'impédance du transistor qui travail en commutation.Supposons alors, qu'avant le temps t=0, la tension à l'entrée du premier inverseur soit nulle, de telle sorte que la tension de sortie du second inverseur soit nulle aussi. Au temps t=0, appliquons une tension VDD à l'entrée du premier inverseur et examinons la suite des événements. La sortie du premier inverseur, qui est reliée à l'entrée du second, est initialement à VDD. Puisque la charge ne peut fournir tout au

plus qu'un courant égal à 1/k fois celui du transistor, le temps de propagation à travers le second inverseur sera environ k fois plus grand que pour le premier inverseur. Il est alors plus simple de parler du temps de commutation à travers une paire d'inverseurs. Ce temps comporte un "temps de montée" et un "temps de descente"; il est approximativement égal à (1+k) temps de commutation (figure 2.2.b). La figure 2.3 montre un inverseur qui attaque l'entrée de plusieurs autres inverseurs. Dans ce cas, pour un facteur de sortance f, il est clair que l'élément actif doit fournir ou extraire f fois plus de charges que dans le cas d'une seule entrée. Le temps de propagation, à la montée et à la descente, est alors augmenté dans un rapport approximativement égal à f. Pour une transition descendante, le temps de commutation est environ f fois le temps de commutation du transistor. Pour une transition montante, le temps de commutation est environ kf fois le temps de commutation du transistor.

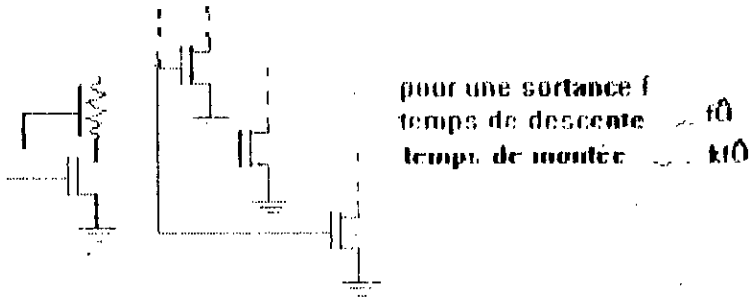


figure 2.3: Sortance

1.2.1 Compromis espace/temps

Lorsqu'un inverseur en commande un autre à quelque distance, il doit charger non seulement la grille de cet inverseur, mais aussi la capacité repartie de la liaison qui relie les deux inverseurs. Toute augmentation de la distance entre les deux inverseurs accroît le temps de commutation de la paire d'inverseurs. Il n'y a aucun moyen d'empêcher le fait que le transport d'un signal, d'un point à un autre situé à une certaine distance, implique soit la charge soit la décharge d'une capacité et requiert par là un certain temps. Afin de minimiser et le temps et l'espace requis pour implanter les fonctions d'un système, il faut utiliser les circuits les plus petits et nous les placerons de façon à réduire la distance des interconnexions.Si au fil des ans, les techniques de fabrication s'améliorent, en remplaçant alors le circuit par un autre pour lequel toutes les dimensions sont réduites dans un rapport α, y compris les épaisseurs, les tensions d'alimentation VDD et de seuil Vseuil sont aussi réduites dans le même rapport α. La raison de cette réduction d'échelle est évidente: Le nouveau circuit pourra comporter α^2 fois plus de

circuits [1] par unité de surface. La puissance dissipée par unité de surface resterait constante. Toutes les tensions seraient α fois plus faible. Le courant à fournir par unité de surface serait α fois plus élevé, le temps de commutation serait réduit α fois, le facteur de monté serait alors divisé par α^3.

1.3 Les avantages de l'utilisation de la famille CMOS et les differentes technologies utilisées

En microéléctronique toutes dimensions sont données en fonction d'une distance élémentaire appellée "l'unité de longueur" qui permet de définir la technologie de fabrication (5µ,3µ,2µ,1µ,0.8µ).Par exemple en 1978 la valeur de cette longueur notée "λ" a été approximativement de 3µ pour les procédés technologiques courant et à des fins commerciales, et de 0.8µ pour l'arseniur de galium.

Les avantages de la famille CMOS par rapport à la famille TTL sont:

1- La dissipation de puissance des circuits CMOS est normalement située 2 à 3 ordres de grandeur en dessous de celle des circuits bipolaires. Ceci permet d'augmenter dans de larges proportions la densité des composants sur des chips de grandes dimensions sans rencontrer de limitations du caractère thermique. Cette porte ouverte sur intégration à très grande échelle est un avantage prépondérant de la CMOS.

2- Le fonctionnement des circuits CMOS est intrinsèquement simple et peu délicat, ce qui n'est pas sans intérêt lorsque la complexité des fonctions réalisées devient importante.

3- Une seule source d'alimentation nécessaire.

4- Fonctionnement assuré pour la majorité des circuits entre 3 volts et 18 volts d'alimentation.

5- Excellente immunité au bruit (30% de la tension d'alimentation).

6- Sortance extrêmement élevée (plus de 50 entrées CMOS par sortie)

1.4.Evolution des réseaux Logiques Programmables (PLA et EPLD) aux ASICs

Souvent, il est commode d'implanter la logique combinatoire avec des structures régulières de transistors interrupteurs (les modules de logique combinatoire ne contiennent pas d'élément memoire, les sorties d'un tel module sont fonctions uniquement de ces entrées, compte tenu du temps nécessaire à la propagation des signaux d'entrée à travers les circuits). Cependant, nous rencontrerons souvent d'importantes fonctions combinatoires qui ne s'implantent pas facilement avec de telles structures

régulières (logique combinatoire utilisée dans les boucles de contre-réaction des automates d'etats finis). Si la logique combinatoire etait implantée avec une structure irrégulière, les changements résultants de l'examen tardif des fonctions logiques détaillées pourraient nécessiter une remise en oeuvre significative de tout le travail de conception Heureusement, il existe un moyen d'implanter des fonctions combinatoires irregulieres, en utilisant les réseaux logiques programmables (PLA,EPLD) cette technique d'implantation de fonctions logiques a un grand avantage:Les fonctions peuvent subir des modifications significatives sans nécessiter un changement majeur de la conception, ou du dessin de la structure du PLA.La figure 2.4 illustre la structure complète d'un PLA Le schéma comprend

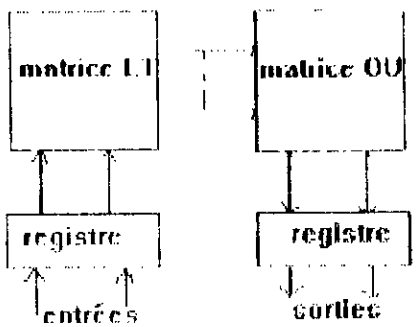


fig2.4: structure d'un PLA

Les registres d'entrées et de sorties pour montrer comment ceux-ci peuvent facilement être intégrés à la conception du PLA. Au début les entrées sont mémorisées dans le registre d'entree, puis sont distribuées verticalement dans une matrice d'elements physiques constituant la matrice ET. La matrice ET génère les combinaisons logiques particulières des variables d'entrées complementées. Les sorties de la matrice ET ont une direction perpendiculaire à celle des entrées. Elles sont distribuées horizontalement dans une autre matrice OU. Les sorties de la matrice OU sont verticales et sont mémorisées dans le registre de sortie.Vue les développements architecturals du VLSI qui permettent de concevoir des dispositifs de faibles coûts, haute densité et haute vitesse, combinés avec une diminution du temps de conception, ceux ci sont employés pour la réalisation des systèmes spécifiques (ASICs) d'où une forte concurrence de ces derniers aux circuits intégrés existant auparavant, car il permettent de remplacer des cartes à circuits imprimés contenant de nombreux circuits du commerce Enfin ils permettent aussi une plus grande fiabilité pour toute la gamme des systèmes électroniques, informatiques, contrôles et télécommunications. Pour ces

raisons ils constituent le domaine du marché des semi-conducteurs dont la croissance est la plus forte.

1.5. Les ASICs

1.5.1 Définition

ASIC: littéralement traduit signifie (circuit intégré spécifique à une application). Un tel circuit est capable d'exécuter les fonctions contenues dans un cahier des charges défini par un client pour une application donnée. Ces circuits intégrés sont conçus pour un objectif spécifique, et regroupent sur une même puce l'ensemble des éléments fonctionnels nécessaire à cette tâche.

1.5.2 Structure des réseaux prédiffusés et caractérisés

Un circuit intégré peut être classés selon la figure 2.5 en:

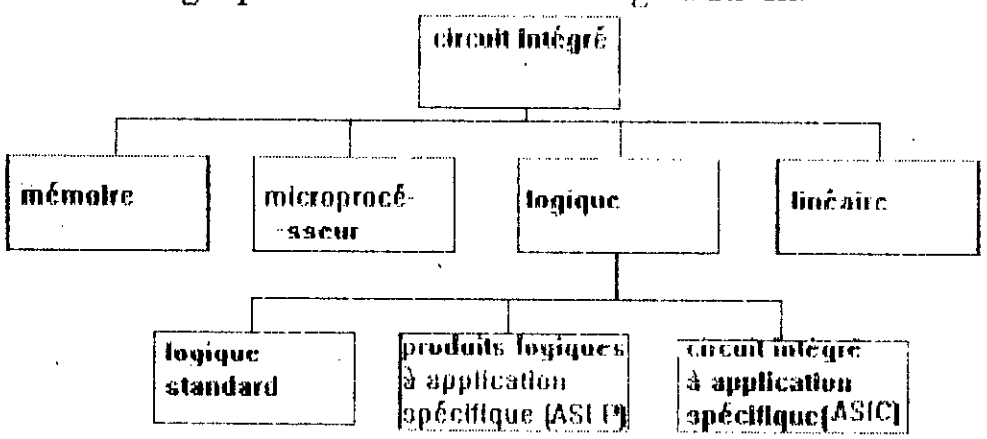


figure 2.5: arbre de ramification d'un circuit intégré

mémoires, microprocesseurs, circuits logiques et circuits linéaires, cependant un circuit logique lui même peut être ramifié en: circuit logique standard, circuit ou produit logiques à application spécifique "Application Specific Logic Products (ASLPs)" et circuit intégré à application spécifique "Application Specific Integrated Circuits (ASICs)", lui même peut être subdivisé selon la figure 2.6 en:

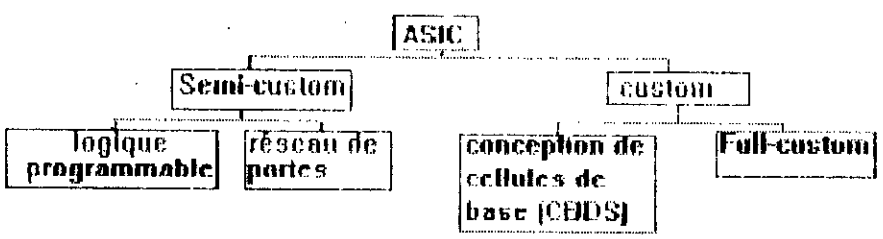
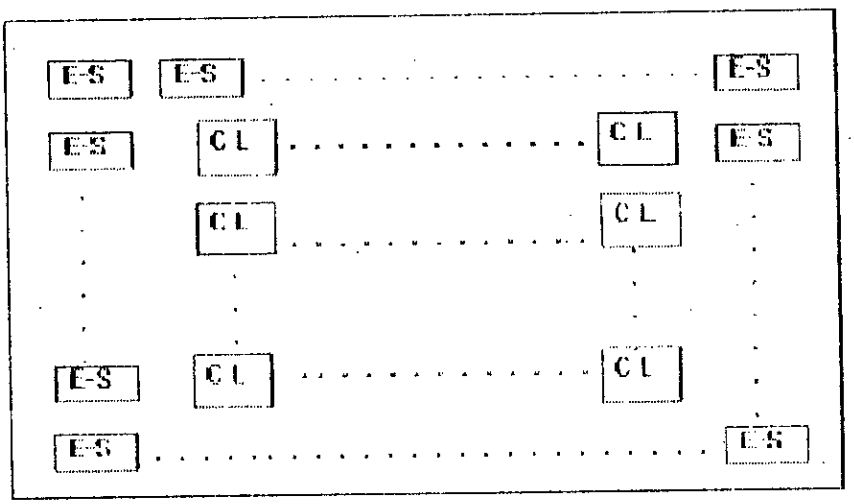


figure 2.6: arbre de ramification d'un ASIC

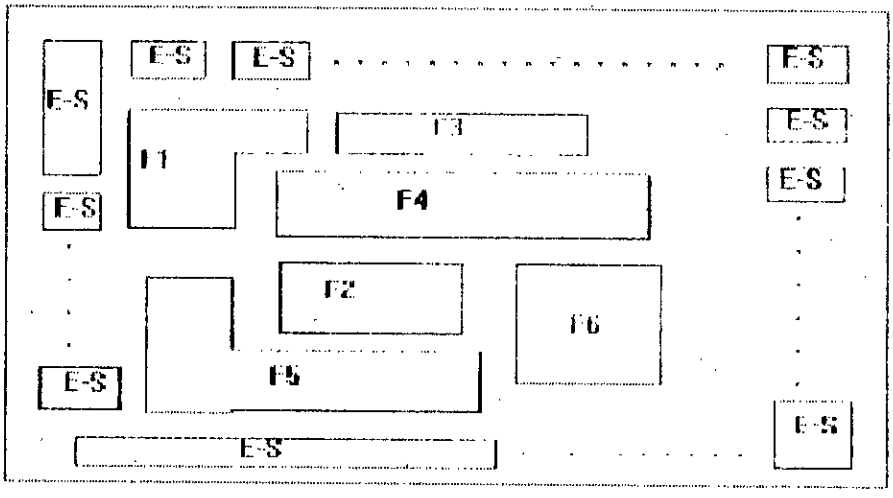
semi custom qui contient la logique programmable et le réseau de porte (Gate Arrays) et custom qui contient les réseaux à conception de cellule de base (Cell Based Design (CBD)) et les Full custom, donc un réseau prédiffusé vierge (Gate Array) comporte des cellules d'entrée et de sortie en la périphérie et des cellules logiques à l'intérieur (figure 2.7) mais il n'y a pas de liaisons préétablies pour connecter les cellules entre elles, ni même de connexions entre les éléments d'une cellule.



E-S : Réseau d'entrée ou sortie
CL: cellule logique

figure 2.7: Réseau prédiffusée vierge

Celles ci sont diffusées par le fabricant lorsque le client a terminé son schéma et en a établi la liste avec outils informatiques.
Le réseau de cellules précaractérisées (Full-Custom) offre un degré de souplesse encore superieur aux prédiffusées (figure 2.8)



E-S:cellule d'entrée ou sortie
fi: fonction précaractérisée

figure 2.8: circuit à cellules précaractérisées

Il n' y a pas de composants prédéterminés avec des emplacements fixées pour des cellules contenant des portes logiques identiques mais seulement des schémas électriques et topologiques dans une bibliotheque d'ordinateur. Ceux-ci correspondent à des fonctions d'usage courant (registres,compteurs,mémoires..). La forme géométrique de ces fonctions peut varier. Les fonctions sont dites précaractérisées car leur consommation électrique, leur fréquence de fonctionnement, leur temps de propagation, etc, sont connus au préalable. En fait cela revient à disposer d'un catalogue de fonctions diffusables à la demande sur silicium comme on dispose d'un catalogue de circuits standards prêts à l'emploi

Les prédiffusés ou précaractérisés portent le nom générique d'ASIC parcequ'ils sont configurés spécialement pour une application. Ils offrent une grande souplesse de configuration, de connexions et d'emplacement que les PLD et les réseaux de cellules LCA. Leur densité est aussi plus grande: il y'a aujourd'hui des milliers de portes dans un seul boitier.

1.5.3.Avantage des ASICs

Concevoir un systeme électronique en utilisant des ASICs plutôt que des composants en logique discrete fournit la possibilité d'échapper à quelques-uns des travers des pratiques de conception qui se sont construites au cours des années.

La médiocrité de ces méthodes est en partie due aux choix inadequats des composants, en particulier de la série 7400. Il n y'a plus de contraintes sur le

blocage des entrées-sorties pour les blocs fonctionnels internes à un ASIC, et où l'on dispose d'une conception hiérarchique.

Grâce aux outils de CAO, les fonctions requises pour un circuit peuvent être exprimées sous formes d'un assemblage hiérarchique de modules concus spécifiquement pour ce circuit. La prise en compte de la testabilité peut faire partie intégrante de la conception fonctionnelle. Les étages redondants sont détruits lors de la conception d'un réseau optimisé, ce qui donne un circuit intégré plus compact, qui utilise moins de silicium, consomme moins, et est plus rapide.

CHAPITRE II

LES MULTIPLIEURS

2.1. La multiplcation

La multiplication câblé sur circuits intégrés est l'une des applications vitales qui jouent un rôle important dans divers systèmes dipitaux rapides, tels les ordinateurs et les processeurs de signal (traitement en temps réel de la parole et de l'image, ff) Pour ce dernier domaine la multiplication à très grande vitesse est indispensable

La fonction de multiplication peut être vue comme la combinaison de deux opérations majeures : la génération des produits partiels à partir du multiplicande et du multiplicateur, et la sommation de ces produits partiels donnant le produit final

Un ensemble d'algorithmes de generation de produits partiels existe, allant du plus simple au plus élaboré. On distingue fondamentalement deux types d'algorithmes : les algorithmes simples sans recodage qui comprennent notamment l'algorithme élémentaire "Shift and add", l'algorithme de Pezari [2] et celui de Baugh-Wooley [10] et les algorithmes avec recodage tels que l'algorithme de Takagi [3], de Booth et Booth modifié.

De la méme manière, un grand nombre de méthodes de sommation sont disponibles, elles diffèrent par le mode de propagation selon qu'il soit linéaire ou logarithmique Parmi les méthodes à propagation linéaire, on distingue les reseaux d'additionneurs à propagation de retenue "Carry Propagate Adder Tree C.P.A.T", à sauvegarde de retenue "Carry Save Adder Tree C.S.A.T" et a propagation alternée "Alternating Carry Save Adder Tree A.C.S.A.T". Entre les méthodes à propagation logarithmique, on relève l'arbre de Wallace, de Dadda et les différentes combinaisons entre ces méthodes.

Il existe aussi des options qui permettent d'améliorer l'efficacité des techniques déjà mentionnées, tels que le "piplining" des calculs, l'emploi d'additionneurs à anticipation de retenue "Carry look-Ahead Adder C.L.A.A" ou la technique "Manchester".

Par contre les multiplieurs se divisent en deux grandes catégories selon le mode de traitement de données numériques qui sont.

- Les multiplieurs bit série en VLSI
- Les multiplieurs parallèles monolithiques en VLSI

2.2 Multiplieurs bit série en VLSI

2.2.1. Introduction

Beaucoup de systèmes de traitement de données numériques utilisent en représentation "bit série" des séquences de données ou d'échantillons d'un signal. Le problème majeur avec l'algorithme série, dans le cas de la multiplication, a été généralement le manque de rapidité ou la complexité de la circuiterie nécessaire pour l'implémentation

Dans un multiplieur "bit série", on suppose qu'au moins un des opérandes arrive en série typiquement un bit à la fois. L'autre opérande est typiquement disponible en parallèle . le résultat est calculé en série.

2.2.1. Multiplication "bit série" pour des opérandes non signés

2.2.2.1 Multiplicateur bit série à sauvegarde de retenue (carry-save)

La figure 2.1 montre un multiplieur bit-série dont la retenue est stockée dans un registre. Ce circuit est plus rapide qu'un multiplieur bit-série où la retenue se propage à travers la chaîne. Pour un opérande (multiplicande) de N bits, le temps d'execution de l'operande est proportionnel à la somme (N+K) des deux opérandes. Mais un certain nombre de cycles d'horloge est nécessaire pour qu'après le passage de tous les bits de l'opérande série, le résultat sera calculé complètement et il y à remise à zéro (self-reset).

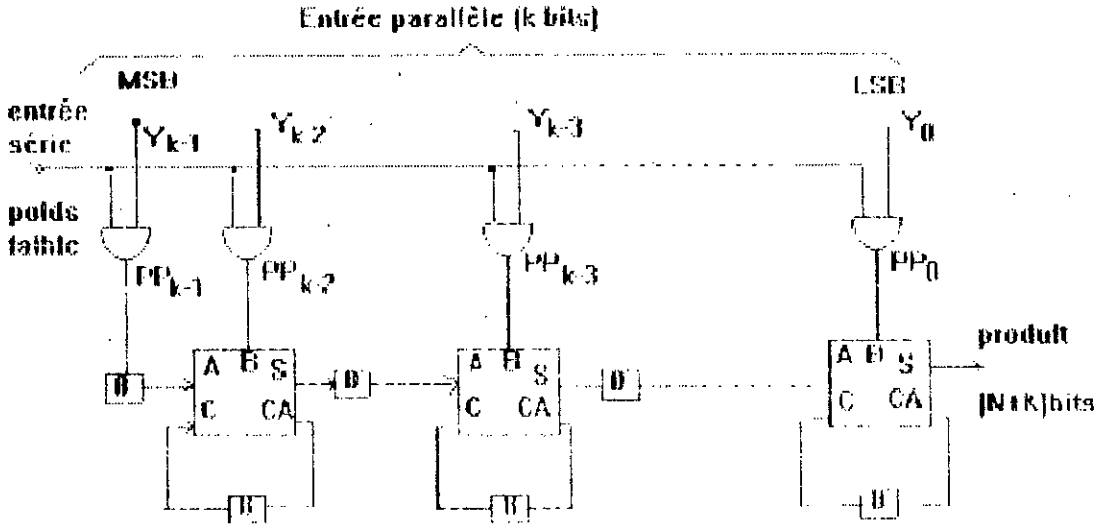


fig2.1 Série-Parallèle

Légende :

D : retard d'un temps bit

A B S
C CA
D

: additionneur complet d'un bit avec bascule de retenue.

2.2.2.2. Multiplieur PipeLine de Lyon

Le problème principal du multiplieur précédent est le fait qu'il utilise une chaîne de zéros, après le passage du bit de poids fort (MSB) du multiplieur série Lyon en 1976 a proposé une version en "Pipe-Line" ou les opérandes séries sont traités par le multiplieur avec aucune séparation de zéros entre eux Ce multiplieur consiste en K modules, avec des interconnections en série pour l'opérande multiplicande, et la somme des produits partiels. Des retards sous contrôle d'horloge sont incorporés sur outres toutes les interconnections série, pour augmenter la vitesse de propagation des bits et donc la vitesse de l'opération.

- Au lieu d'appliquer des signaux de commande en parallèle, on incorpore dans le module un autre registre-serie pour propager le signal de commande à travers chaque module.
- Pour que l'opérande multiplicateur Y puisse être entrer en série, en même temps que l'opérande multiplicande X et le signal de commande, on inclut un registre-série avec des bascules.

La figure 2.2 montre l'architecture du multiplieur de Lyon décrit ci-dessus.

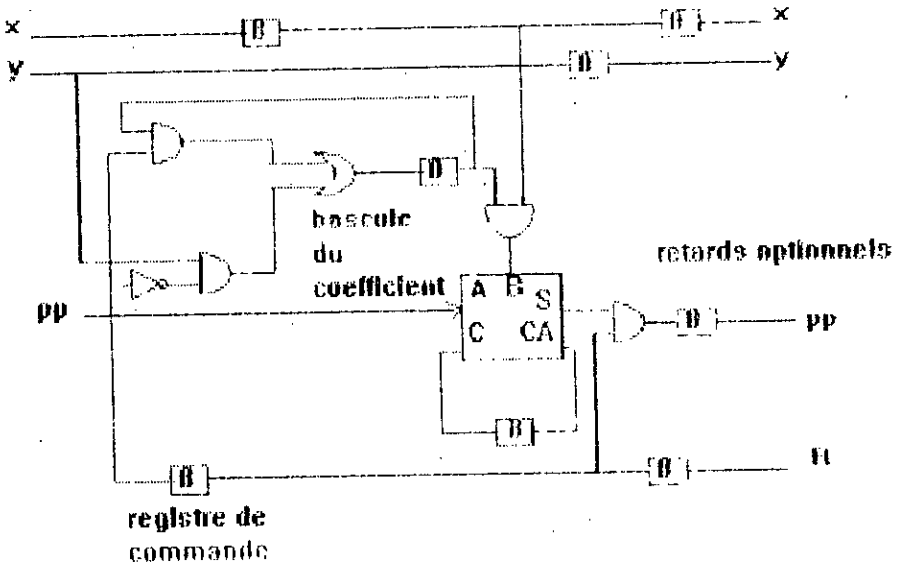


fig 2.2 Une cellule de multiplieur de Lyon

2.2.3. Multiplication "Bit-Série" pour opérandes signés l'algorithme de Booth

Les algorithmes précédents permettent de faire la multiplication en utilisant des additions répétitives pour des opérandes non signés.

L'algorithme de Booth permet:

i)de réduire le nombre de commandes

ii)d'améliorer le rendement (temps x surface de la multiplication).

Ceci en particulier dans le cas des multiplieurs parallèles (cf.chapitre II). Cependant, ceci n'est pas évident dans le cas des multiplieurs séries. La raison principale de l'utilisation de cet algorithme dans ce type de multiplieurs est qu'elle permet à l'utilisateu une circuiterie additionnelle pour traiter les bits signés des opérandes.

2.2.3.1. L'algorithme de Booth modifié, Multiplieur de Lyon

On peut réduire le temps de propagation d'un multiplieur en utilisant l'algorithme de Booth modifié. Cette algorithme recode l'opérande coefficient de manière à réduire par deux le nombre d'étapes de calcul et par conséquent, réduire par deux la longueur du réseau de traitement. L'algorithme de Booth modifié recode chaque paire de bits de l'opérande coefficient dans un système à 5 digits: -2,-1,0,+1,+2. L intérêt de ce plan de recodage est que les produits de ces digits avec les opérandes multiplicandes, sont implementés simplement par des décalages optionnels. De cette façon, il n'est génére qu'un seul produit partiel pour chaque paire de bits de l'opérande coefficient et le

nombre des sommes de produits partiels est réduit par conséquent. Les nouveaux produits partiels et sommes de produits partiels deviennent plus longs d'un bit, alors, on a besoin d'une extension de signes d'un bit de l'opérande multiplicande. L'algorithme de recodage consiste en l'éxamination de triplets de bits de l'opérande, comprenant la paire de bits considérée ainsi que le bit de poids faible suivant. Le bit initial de poids faible est un zéro inséré.

Le coefficient d'origine est $Y=\sum_{i=1}^{k-1} Y_i 2^{i-k+1} - Y_k 2$ i pair

qui est recodé comme suit: $Y= \sum_{i=1}^{k-1} Y'_i 2^{i-k+1}$ i impaire

où Y'_i {-2,-1,0,1,2}

opérandes coefficient d'origine			opérandes coefficient recodé	opération
Y_{i+1}	Y_i	Y_{i-1}	Y'_i	
0	0	0	0	ajouter zéro
0	0	1	+1	ajouter X (multiplicande)
0	1	0	+1	ajouter X
0	1	1	+2	ajouter 2X
1	0	0	+2	soustraire 2X
1	0	1	-1	soustraire 1X
1	1	0	-1	soustraire 1X
1	1	1		soustraire zéro

Tableau 2.1

La cellule du nouveau multiplieur utilisant cet algorithme est montré en figure 2.3.

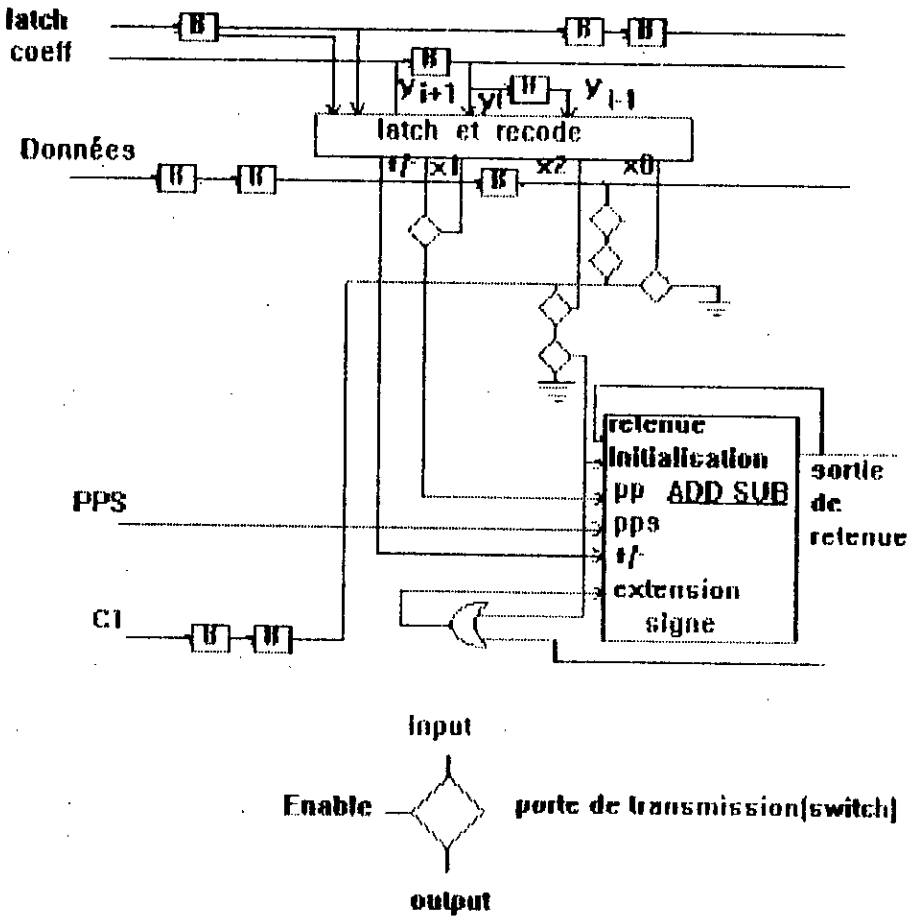


fig 2.3: Une cellule du multiplieur de Booth modifié

L'élément principal de calcul est un add/soust. série programmable avec retour interne de la retenue. l'opérande coefficient est propagé sur une ligne de retard à 1 bit par étage, modifiée de façon à permettre aux trois bits concernés à être mis en attente et recodés sous la commande du signal "Latch". Ceci est relié mais séparé de C1 pour permettre à un seul opérande coefficient d'être en permanence sauvegardé si ceci est requis. En connectant "Latch" à C1 à l'entrée du multiplieur, on peut effectuer une opération continue avec des opérandes variables.

Les opérandes-multiplicandes sont propagés à travers une ligne de retard à 3 bits par étage (pour accommoder le retard de 1 bit entre les sommes de produits partiels) et ceci est connecté de façon à effectuer la formation de produits partiels 1x, 2x.

- La cellule add/substract est initialisé à partir de C1 mais l'extension de signe est séparément contrôlé pour effectuer une extension à 2 bits de sommes de produits partiels.

- L'extension de signes est omise dans la dernière cellule, qui est identique aux autres cellules du point de vue opération arithmétique.
- La première cellule de réseau est différente du reste, concernant l'entrée de faible poids du recodeur, qui est fixée à zéro. La dernière cellule du réseau diffère des autres, dans le sens où un dimensionnement correct ne requiert qu'un seul bit extension et décalage.

2.2.4. Conclusion

Les multiplieurs bit-série présentent plusieurs avantages, notamment une réduction dans la surface occupée.
Pour la multiplication en complément à deux, par l'utilisation simultanée de deux bits de l'opérande série, recodés par l'algorithme de Booth modifié, on aboutit à une configuration qui a des performances élevés dans un environnement VLSI. Cette configuration est très utilisée actuellement, où il s'agit d'intégrer le multiplieur sur la même puce avec d'autres modules, pour former des processeurs de traitement du signal.

2.3. Multiplieurs parallèles monolithiques en VLSI

2.3.1. Introduction

Une amélioration de la vitesse de la multiplication est possible par l'utilisation des multiplieurs parallèles qui génèrent à la fois tous les produits partiels de deux opérandes. Ces multiplieurs parallèles n'ont pas de compromis (surface x temps) optimum, comparés aux multiplieurs "bit-série". Cependant dans la majorité des cas pratiques, ils sont utilisés car ils offrent un temps d'exécution minimal.

2.3.2. Génération des produits partiels

La représentation des entiers signés la plus utilisée est le complément à 2. Ceci nécessite que les algorithmes de calcul soient conformes à cette représentation. Afin d'atteindre des performances meilleures, certains algorithmes emploient d'autres types de représentations. Ceux-ci doivent effectuer un recodage des données avant leur traitement, c'est-à-dire, passer du code complément à deux employé. C'est le cas de l'algorithme de Takagi [4] qui utilise la représentation binaire redondante SD "Signed Digit" et aussi l'algorithme de Booth modifié qui repose sur la technique connue sous le nom "String Recoding" [5].

2.3.3. Algorithmes simples (sans recodage)

2.1.1. L'algorithme élémentaire "Shift and Add" (Multiplication non signée)

Soient X, Y deux entiers positifs, codés sur N bits chacun.

$X = X_{n-1}, X_{n-2}, \ldots\ldots X_1, X_0 = \sum_{i=0}^{n-1} X_i 2^i$: le multiplicande

$Y = Y_{n-1}, Y_{n-2}, \ldots\ldots Y_1, Y_0 = \sum_{j=0}^{n-1} Y_j 2^j$: le multiplicateur

Soit P le produit: $P = XY = P_{2n-1}, P_{2n-2}, \ldots\ldots P_1, P_0$ qui peut se mettre sous la forme :

$$P = X.\sum_{j=0}^{n-1} Y_j 2^j$$

$$= \sum_{j=0}^{n-1} Y_j (X 2^j) \qquad (1)$$

$$= \sum_{j=0}^{n-1} \left[\sum_{i=0}^{n-1} (Y_j X_i) 2^i \right] 2^j \qquad (2)$$

L'expression (1) montre que l'on calcule produit comme une suite de produits partiels $Y_j(X.2^j)$ qui ne sont rien d'autres que le multiplicande X décalé de i positions à gauche multiplié par le bit (i) du multiplicateur.

L'exemple suivant illustre cette méthode très simple

soit X= 1 0 0 1 1 = 19 , n=5
Y= 1 1 0 0 1 = 25

$$\begin{aligned}
1\,0\,0\,1\,1 &= Y_0(X.2^0)\\
+0\,0\,0\,0\,0\,. &= Y_1(X.2^1)\\
+0\,0\,0\,0\,0\,.\,. &= Y_2(X.2^2)\\
+1\,0\,0\,1\,1\,.\,.\,. &= Y_3(X.2^3)\\
+1\,0\,0\,1\,1\,.\,.\,.\,. &= Y_4(X.2^4)\\
\hline
P = 1\,1\,1\,0\,1\,1\,0\,1\,1 &= 475
\end{aligned}$$

La matérialisation sous forme de circuit est pratiquement immédiate (fig 2.4), cette architecture est très régulière s'adaptant bien à l'implantation VLSI, mais elle est très lente à cause de la propagation horizontale de la retenue, cependant la vitesse peut être améliorer par l'utilisation d'un réseau à sauvegarde de retenue suivi d'une chaîne à propagation de

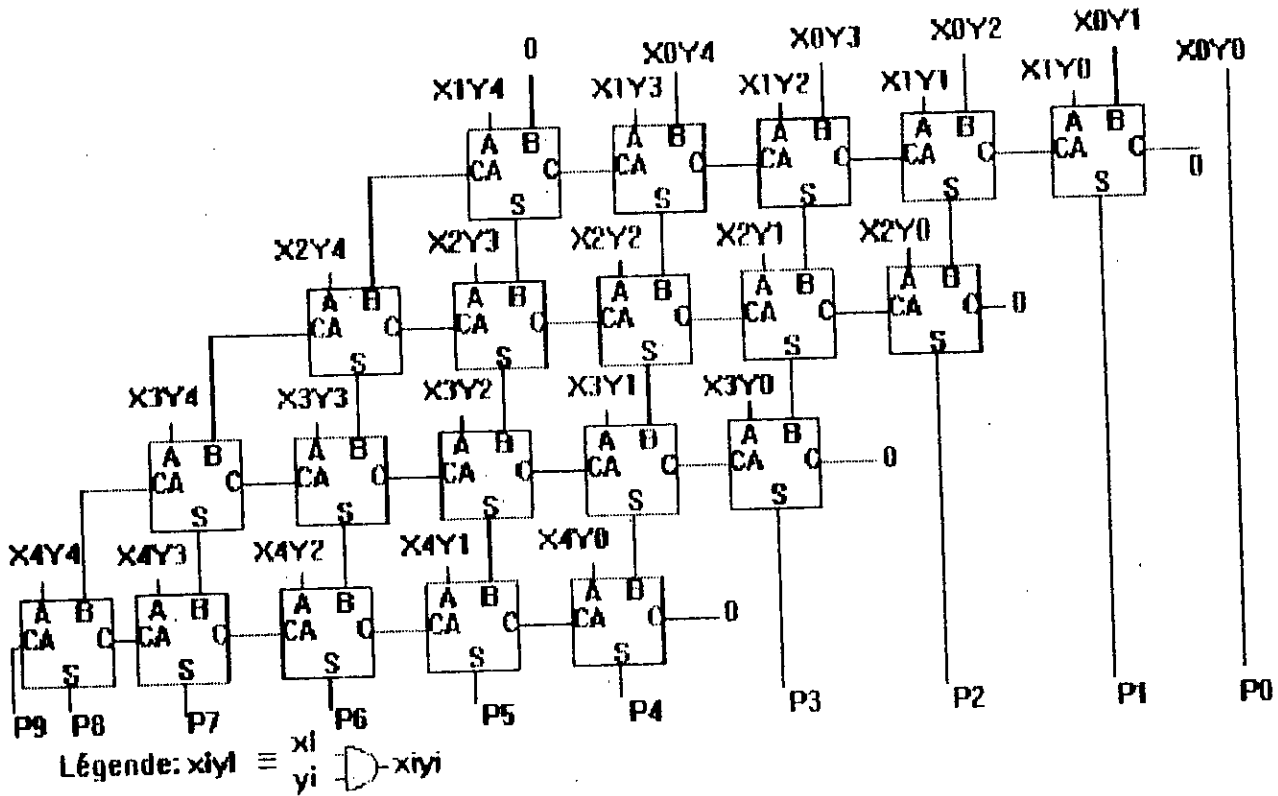


figure2.4: Architecture du multiplieur "Shift and Add"
5 bits x 5 bits

retenue munie d'une circuiterie "Carry-LookAhead" délivrant le résultat final.
De même, on peut utiliser un réseau d'additionneurs à propagation alternée résultant en un gain de vitesse plus élevé, mais aussi en une affectation sensible de la régularité (comme l'utilisation de l'arbre de Wallace ou de Dadda).

2.3.3.1. Multiplieurs des nombres signés (complément à 2)

Quand les opérandes sont sous forme complément à deux, la conception de tels multiplieurs devient très difficile puisque le bit de signe se trouve imbriqué dans l'opérande lui-même.
Des réseaux pour la multiplication directe ont été proposés [6] , [7] les cellules utilisées par ces configurations sont:
i) plus complexes.
ii) leur nombres est plus important par rapport à un réseau de multiplieurs de nombres positifs.
Deux autres configurations qui réduisent les problèmes cités ci-dessus sont:
i) le réseau de Pezaris [2]
ii) l'algorithme de Badeira [8].
La configuration de Pezaris utilise différents types de cellules dans le réseau et par conséquent, la modularité et la régularité du réseau sont détruites par contre l'algorithme de Baiera utilise un seul type de cellules d'addition et les valeurs vraies des opérandes. Les seuls problèmes de cette configuration sont les irrégularités topologiques qui peuvent créer certains problèmes d'implémentation en VLSI.
La table suivante résume les performances de ces deux approches pour une multiplication de m x n en complément à deux.

	Pezaris	Badeira
Nombre total de cellules	n(m-1)	m(n-1)
temps de retard total	2(m+n-1)Δ	2(m+n-1)Δ

Tableau 2.2

Δ: Temps de retard d'une cellule.

2.3.4. Algorithme avec recodage

2.3.4.1. Algorithme de Booth et Booth modifie

Comme l'algorithme de Baugh-Wooley, l'algorithme de Booth apporte une solution au problème pose par la multiplication signée, en se basant sur la multiplication mathématique des deux opérandes X et Y. Soit Y le multiplicateur et X le multiplicande, représentés en notation complément a deux:

$$Y = -Y_{n-1}2^{n-1} + \sum_{j=0}^{n-2} Y_j 2^j$$

Cette équation peut être aussi écrite de la façon suivante

$$Y = \sum_{j=0}^{(n/r)-1} (Y_{rj-1} + Y_{rj}2^0 + Y_{rj+1}2^1 + Y_{rj+2}2^2 + \ldots\ldots + Y_{rj+r-2}2^{r-2} + Y_{rj+r-1}2^{r-1})2^{rj}$$

Où Y-1=0 , r entier positif

Posons :

$$Y = \sum_{j=0}^{(n/r)-1} Q_j 2^{rj} \qquad Q_j \in \left[-2^{r-1}, +2^{r-1}\right]$$

De cette nouvelle équation, il devient clair que le multiplicateur Y peut être divise en (n/r) groupements ayants chacun n+1 bits. Deux groupements contigus ont 1 bit commun:

$$X.Y = \sum_{j=0}^{(n/r)-1} (Q_j X)2^{rj}$$

Ce qui correspond a une somme de n/r produits décalés l'un de l'autre de n positions. Ces deux observations ont permis d'annoncer un algorithme selon les étapes suivantes:

1- Diviser le multiplicateur en n/r groupements.
2- Evaluer chaque groupement (calculer les facteurs Q_j).
3- Générer les produits partiels en effectuant le produit (Q_j.X) .
4- Additionner les produits partis.

Cet algorithme s'applique aussi bien a une multiplication signée qu'à une multiplication non-signeé. Seulement, dans ce dernier cas, il devient impératif d'insérer deux bits a zéro à gauche de la partie la plus significatif du multiplicateur ; Un pour que le multiplicateur ne soit pas confondu avec un nombre signé négatif, et l'autre pour que le groupement soit complet . Ces deux bits ne sont donc pas nécessaires pour un multiplicateur représenté en complément à deux . Notons aussi qu'on doit dans les deux cas insérer un bit à zéro (Y-1) à la partie la moins significative du multiplicateur et que l'un doit s'assurer de l'éxtension du signe du multiplicande, ainsi que l'intégrité de la notation l'éxige.

Pour la multiplication n×n signée et non-signée, le résultat est toujours donné sur 2n bits, alors que pour le mode mixte le résultat est donné sur 2n+1 bits; car, la donnée non-signeé est convertie vers la notation en complément à deux nécessitant n+1 bits pour sa représentation.

2.3.4.2 Algorithme de Booth

Dans cette algorithme la valeur de r est égale à 1, ce qui se traduit par une division du multiplicateur en n groupements de deux bits chacun. Pour r=1 :

$$Y = \sum_{j=0}^{n-1} (Y_{j-1} - Y_j) 2^j \quad \text{et} \quad Q_j \in \{-1, 0, +1\}$$

Par conséquent, l'algorithme de Booth est équivalent à recoder le multiplicateur dans un système de nombre ternaire à valeurs -1,0,+1, et la multiplication consiste à ajouter les produits partiels négatifs et positifs, formés en multipliant chaque ó Qj par l'opérande multiplicande.
La table suivante énumère les différentes possibilités :

Y_{j-1}	Y_j	Q_j	opération
0	0	0	+0
0	1	-1	-x
1	0	+1	+x
1	1	0	-0

Tableau 2.3

Exemple: Soit à multiplier deux nombres signés X et Y représentés sur quatre bits en complément à deux.
X=3,Y=-5; en binaire: X=0011, Y=1011. Soit Y le multiplicateur.
La première opération consiste à diviser le multiplicateur en n groupements, n=4.

1 0 1 1 0
Y-1

La seconde opération consiste à recoder les groupements ainsi obtenus suivant la table ci-dessus. on obtient :
$q_0 = -1;\ q_1 = 0;\ q_2 = +1;\ q_3 = -1$
L'opération suivante permet de générer les produits partiels en effectuant les produits bj X
ce qui donne : A = -X, B = 0, C = +X, D = -X.

Pour obtenir -X ,il suffit d'inverser les bits X puis leur ajouter 1.

3 =→ 0011 → 1100 + 1 → 1101 = −3

Finalement, la dernière opération réalise la sommation des produits partiels pour obtenir le produit final.

```
1 1 1 1 1 0 1 : A
0 0 0 0 0 0 . : B
0 0 0 1 1 .   : C
1 1 0 1 .     : D
-----------------
1 1 1 1 0 0 0 1 = -15
```

IL faut prendre certaines précautions lors de l'addition de deux nombres représentés en complément à deux. Pour que le résultat soit correct, il faut répéter le bit de signe de chaque produit partiel jusqu'à ce qu'il ait le même nombre de bits que le produit final. la matrice des produits partiels prend donc une forme trapézoïdale.

2.3.4.3. Algorithme de Booth modifié

Le seul et unique avantage que présente cet algorithme sur celui de Booth est qu'il permet une réduction d'au moins 50% du nombres de produits partiels à additionner, ce qui se traduit par une amélioration conséquente de la vitesse. Cette réduction est réalisée en considérant des groupements de taille supérieure ou égale à trois. Donc, l'algorithme de Booth modifié s'applique pour des valeurs de r supérieure à deux.

Pour r=2 :

$$Y = \sum_{j=0}^{(n/2)-1} (Y_{z_{j-1}} + Y_{z_j} - 2XY_{z_{j+1}})4^j \quad \text{et } Q_j \in \{-2,-1,0,1,2\}$$

Dans ce cas, l'algorithme de Booth modifié recode chaque paire de bits du multiplicateur dans un système à 5 digits: -2, -1, 0, 1, 2.

L'intérêt de ce plan de recodage est que les produits de ces digits avec les opérandes multiplicandes sont implantés simplement par décalages optionnels. De cette façon, il n'est généré qu'un seul produit partiel pour chaque paire de bits du multiplicande et le nombre de produits partiels devient plus long d'un bit à cause du décalage éventuel.

L'algorithme de recodage consiste en l'examen de triplet de bits du multiplicateur selon la table suivante :

Y_{j-1}	Y_j	Y_{j+1}	Q_j	opération
0	0	0	0	+0
1	0	0	+1	+X
0	1	0	+1	+X
1	1	0	+2	+2X
0	0	1	-2	-2X
1	0	1	-1	-X
0	1	1	-1	-X
1	1	1	0	-0

Tableau 2.4

Exemple : Soit à multiplier deux nombres signés X et Y représentés sur quatre bits en complément à deux.

X=3, Y=-5 en binaire X=011, Y= 1011 1 0 1 1 0.....y-1

Après recodage, on obtient : 0 = - 1 ; 1 = - 1 . Les produits partiels sont donc :

A= -X; B= -X

1 1 1 1 0 1 : A
1 1 0 1 . . :

1 1 1 1 0 0 0 1 = - 15

Nous terminons cette discussion par une réflexion sur le choix de la taille des groupements . Sans doute, il peut paraître que choisir des tailles de groupements plus grandes en résulte des performances meilleures. Ceci n'est pas vrai; du moins pour des multiplieurs de taille moyenne (4 bits et moins) , car, avec des groupements de trois bits, on a besoin des termes ± X qui peuvent être facilement obtenus par décalage et/ou une complémentation. Par contre pour des groupements de 4 bits, par exemple, il serait nécessaire de calculer des termes tels que ± 3X, ± 4X,. Le temps nécessaire pour calculer le dernier terme ,surtout pour des opérandes longs, empêche l'emploi de plans de recodage plus complexe, au moins dans le domaine du VLSI. Une exception est faite dans le cas où le multiplieur est une composante d'une structure plus complexe. Dans ce cas, il pourrait y avoir un temps «morts»

suffisamment long, du à des contraintes architecturales pour permettre le calcul de ces termes.

2.4 Les nouveaux algorithmes

Les contraintes de la technologie VLSI actuelle ont permis la conception de nouveaux algorithmes pour la multiplication en notation complément à deux [11][12]. Actuellement, ces nouveaux algorithmes peuvent êtres classés en deux catégories:

1. Algorithmes utilisant l'arbre d'addition binaire redondante.
   RBAT (Redundant Binary Addition Tree).
2. Algorithme de Hurson basé sur l'architecture systolique [13] [14].

2.4.1 L'algorithme de Takagi (RBAT)

Dans cet algorithme, on utilise une représentation redondante ou chaque digit peut être 0,+1,-1. C'est l'une des représentations des nombres a digit signés (SD) proposée par Avizienis.
Un exemple d'un multiplieur n×n bits, basé sur cet algorithme est montré dans la figure 2.4
Le multiplieur basé sur cet algorithme est formé d'un arbre binaire d'additionneurs binaires redondants. Cet algorithme est décrit comme suit :

Etape 1 :

Conversion du multiplicande A et du multiplicateur ou coefficient B, en représentation équivalente binaire redondante.
(Le temps d'exécution est indépendant de n, nombre de bits du multiplieur, et le nombre des éléments de calcul nécessaires est aussi indépendant de n).

Etape 2 :
génération des produits partiels P0j avec (j=0,1,.....n-1) de n bits pour n bits du multiplicateur B. Le temps d'exécution est indépendant de n puisque tous les produits partiels des n bits sont générés en parallèle, alors que le nombre d'éléments de calcul nécessaires est proportionnel à n^2.

Etape 3 :
Addition des produits deux à deux, au moyen d'un arbre binaire constitué d'additionneurs binaires redondants jusqu'à l'obtention du produit $P_{K,0}$ ou $K = \log n^2$.
Réalisation des additions binaires redondantes à chaque niveau de l'arbre en parallèle, c'est à dire :

Toutes les additions

$$P_{i,j} = P_{i-1,2j} + P_{i-1,2j+1} \quad \text{pour} \quad j=0,1,.....(n-1)/2i$$

avec i=1,2,.....k.

se font en parallèle.

Pij :représente le jième résultat intermédiaire au niveau de l'arbre.

Le temps d'exécution dans cette étape est proportionnel à $K = \log n^2$ et le nombre des éléments de calcul nécessaires est proportionnel à n^2.

Etape 4 :

Conversion du produit $P_{k,0}$ dans la représentation équivalente en complément à deux.

(La conversion est réalisée au moyen d'un additionneur à anticipation de retenue [15],donc le temps d'exécution de cette étape est proportionnel à $\log n^2$,alors que le nombre d'élément d'un additionneur à anticipation de retenue (CLA) est proportionnel à n.)

Par conséquent, l'exécution totale de l'algorithme de multiplication se réalise dans un temps réalise de $O(\log n^2)$ avec $O(n^2)$ éléments de calcul. Les étapes de cet algorithme nous font constater que la structure du réseau cellulaire est régulière et par conséquent, elle est apte pour une implémentation VLSI.

Nous noterons que l'utilisation de l'algorithme de Booth est possible pour générer les produits partiels. Ainsi , le temps de calcul et le nombre des éléments nécessaires seront réduits. La réalisation pratique d'un multiplieur sur cet algorithme se trouve dans [16].

2.4.2 L'algorithme de Hurson :[13][14]

Cet algorithme est basé sur la considération des données d'un point de vue logique, plutôt que sur leur addition. En général, la multiplication de deux nombres X et Y de n bits génère une matrice M de sommes de produits partiels (Summards) dont les éléments mij sont donnés par la formule (1) suivante :

$$\begin{cases} 1 & \text{si } X_i = Y_j = 1 \\ 0 & \text{ailleurs} \end{cases}$$

Le résultat de la multiplication (P2n-1.....P1) est aussi donné par la formule (2) suivante :

$$P_k = S_k \bmod 2 \quad \text{où } S_k = \sum_{i=0}^{k} (X_i Y_{k+1-i} + C_i) \qquad X_i = Y_j = 0 \quad \text{si } i > n$$

$$\text{et} \begin{cases} C_{k-1} = (S_{k-1} / 2) & \text{pour } 2 < k < 2n \\ C_0 = 0 \end{cases}$$

Ck est retenue de la colonne k.
Les équations (1) et (2) indiquent :

i) Les éléments Xi Yj = 0 n'affectent pas Sk.

ii) Chaque paire de 1 dans la somme des (Xi Yk+1-i) donne un 1 au terme Ck+1 qui peut à son tour affecter Si avec k+2 < i < 2n.

Ces observations i) ii) ont permis d'annoncer un algorithme selon les étapes suivantes :

1- Seuls les éléments (Xi Yj) différents de zéro sont générés.

2- Pour chaque paire de 1, un 1 est propagé à la prochaine colonne.

3 - Si le nombre de 1 dans chaque colonne k ,avec 1 < k < 2n , est un nombre impair alors, Pk = 1 sinon Pk = 0.

La figure 2.5 montre un exemple d'un tel algorithme.
Dans l'approche de Hurson, les observations précédentes ont été modifié pour aboutir au étape suivante de l'algorithme en architecture pipeline :

i) Les produits partiels sont générés en série, une ligne à chaque impulsion pipeline.

ii) Durant chaque impulsion, un bit du résultat est généré.

iii) La multiplication de deux nombres de n bits est effectué en 3n impulsion.

Durant chaque impulsion, trois opération tiennent place. La topologie contient trois modules:

- un jeu de registres.
- un contrôleur de décalage.
- un générateur de sortie.

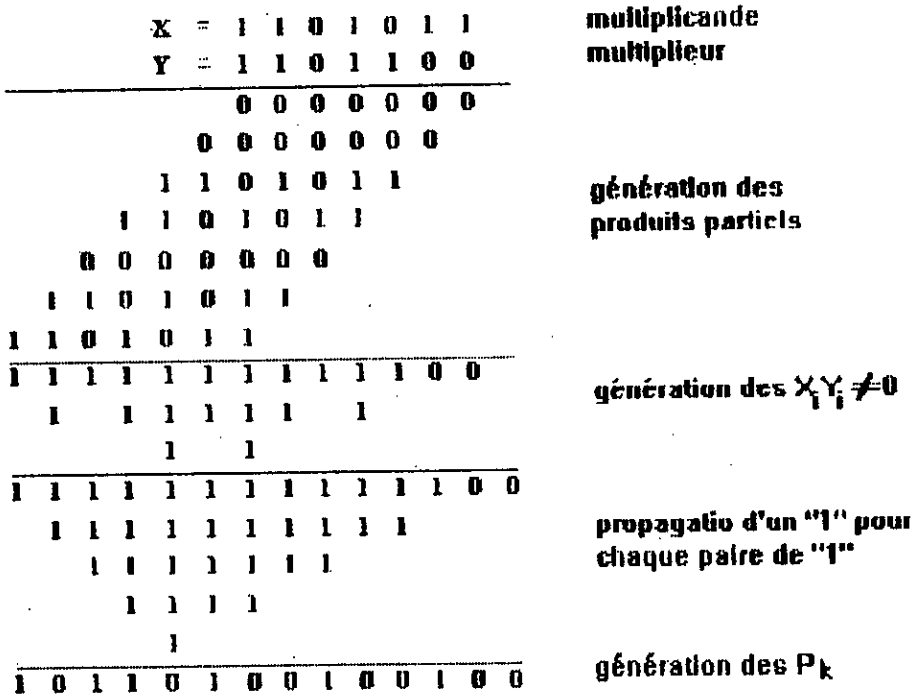


Figure 2.5 : Exemple de l'algorithme de Hurson

2.4.2. L'algorithme de Hurson basé sur l'architecture systolique et sur la méthode de Booth modifié

Pour un multiplieur en complément à deux, l'algorithme de Booth modifié a été incorporé dans l'architecture de Hurson [14] comme suit :

i) A chaque impulsion, un produit partiel est généré. Ce produit partiel est égal à l'une des valeurs 0, X, X', 2X, 2X'. où X' est le complément de X.

ii) L'architecture précédente traite n colonnes et pour tenir compte du bit de signe ,on a n colonnes en plus à traiter.

iii) Des modifications ont été portés sur la génération des bits de résultat; la première colonne atteint le générateur de sortie après n/2 impulsions pipeline.

Après, deux bits par cycle sont générés dans n itérations prochaines. Donc, le système a besoin de 1.5n impulsions pipeline pour multiplier deux opérandes de n bits.

2.5 Sommation des produits partiels

Tels qu'ils sont présentés, les algorithmes de Baugh-Wooley, Booth ainsi que d'autres variantes, ne sont que des techniques de génération de produits partiels adaptées aux exigences de la notation signée. Les méthodes de sommation qui réalisent la seconde opération de la multiplication offrent un éventail beaucoup plus large quant aux applications de ces algorithmes. Un même algorithme pourrait être combiné à plusieurs méthodes de sommation, ce qui lui confère des architectures diverses et donc performances différentes. Ainsi, on dispose d'une panoplie de solutions répondant aux diverses exigences fixées par les spécifications de conception. (régularité, vitesse, surface). Toutes les méthodes de sommation sont des schémas d'additions parallèles. On distingue deux types de méthodes selon que la propagation se fait d'une manière linéaire ou logarithmique.
Considérons la matrice des produits partiels suivante et voyons comment se fait sa réduction en un produit final à travers les différentes méthodes de sommation.

					$A_1.B_5$	$A_1.B_4$	$A_1.B_3$	$A_1.B_2$	$A_1.B_1$
				$A_2.B_5$	$A_2.B_4$	$A_2.B_3$	$A_1.B_2$	$A_2.B_1$	
			$A_3.B_5$	$A_3.B_4$	$A_3.B_3$	$A_3.B_2$	$A_3.B_1$		
		$A_4.B_5$	$A_4.B_4$	$A_4.B_3$	$A_4.B_2$	$A_4.B_1$			
	$A_5.B_5$	$A_5.B_4$	$A_5.B_3$	$A_5.B_2$	$A_5.B_1$				
P =	P_9	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1

2.5.1 Méthodes de sommation à propagation linéaire

2.5.1.1 Réseau d'additionneurs à propagation de retenue

Ce réseau est de tous les réseaux existants ,celui qui s'offre le mieux à l'implémentation VLSI. (figure 2.4) .Ceci en raison de sa très grande régularité. toutefois, sa lenteur inhérente à la propagation de la retenue rend nécessaire l'utilisation de méthodes plus rapides dans des application où la vitesse est un facteur primordial.

2.5.1.2. Réseau d'additionneurs à sauvegarde de retenue CSA

Afin de réduire le délai dû principalement à la propagation horizontale de la retenue, une technique de parallélisme est introduite dans le réseau. Dans la technique de sauvegarde de retenue, illustrée à la (figure 2.6), la retenue se propage en diagonale. Cette disposition est fonctionnellement identique à celle déjà décrite. En effet, la retenue d'une cellule d'additionneurs peut être envoyée n'importe où dans la même colonne, où l'on additionne des bits de même poids.

Marginalement très faible, le prix payé sera la présence d'une rangée supplémentaire que nous appellerons "Accumulateur" où la retenue continue à se propager horizontalement. Cette augmentation de surface se trouve néanmoins compensée en grande partie par l'emploi de demi-additionneurs sur les bords haut et gauche du réseau. Cette nouvelle technique apporte un gain de vitesse de 30% sans altérer pour autant la régularité.

C'est la raison pour laquelle en pratique, on utilise toujours la propagation en diagonale de la retenue. Ce gain peut aller jusqu'à 60% si le délai de l'accumulateur est très inférieur à celui des additionneurs du réseau. On a donc toujours intérêt à optimiser le délai de l'accumulateur. Nous montrerons plus loin quelques techniques permettant d'atteindre ce but.

2.5.1.3. Réseau d'additionneurs à propagation alternée (ACSA)

Combinée à la technique de sauvegarde de retenue, la technique de propagation alternée permet d'accroître d'avantage le parallélisme dans le réseau, Figure 2.7. Au lieu que la somme se propage successivement à travers les additionneurs d'une colonne, elle est alternée entre eux, divisant ainsi par deux le nombre d'étages parcourus par la somme. Ce qui se traduit par une nette amélioration de la vitesse. Toutefois, des irrégularités sont introduites dans le réseau, mais, celles-ci deviennent négligentes devant l'avantage apporté par cette technique.

2.5.2 Méthodes de sommation à propagation logarithmique

Dans le souci d'une grande vitesse et tout en se basant sur la technique de sauvegarde de retenue, certaines méthodes de sommation permettent de réduire le délai d'une manière logarithmique. Ces méthodes reposent sur l'emploi de compteurs parallèles qui exigent la disponibilité simultanée de tous les produits partiels. Leur principe est donné par la (Figure 2.8) pour un multiplieur 8×8.

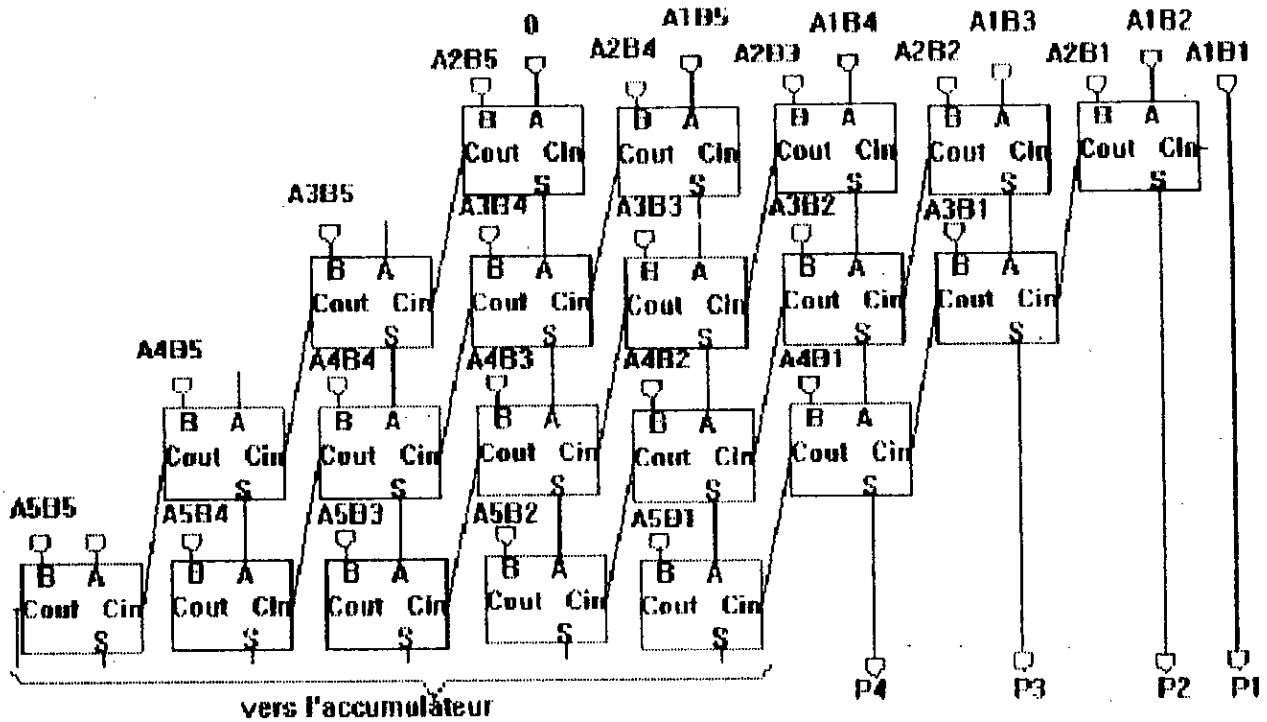


FIGURE 2.6:Réseau d'additionneurs à sauvegarde de retenue (CSA)

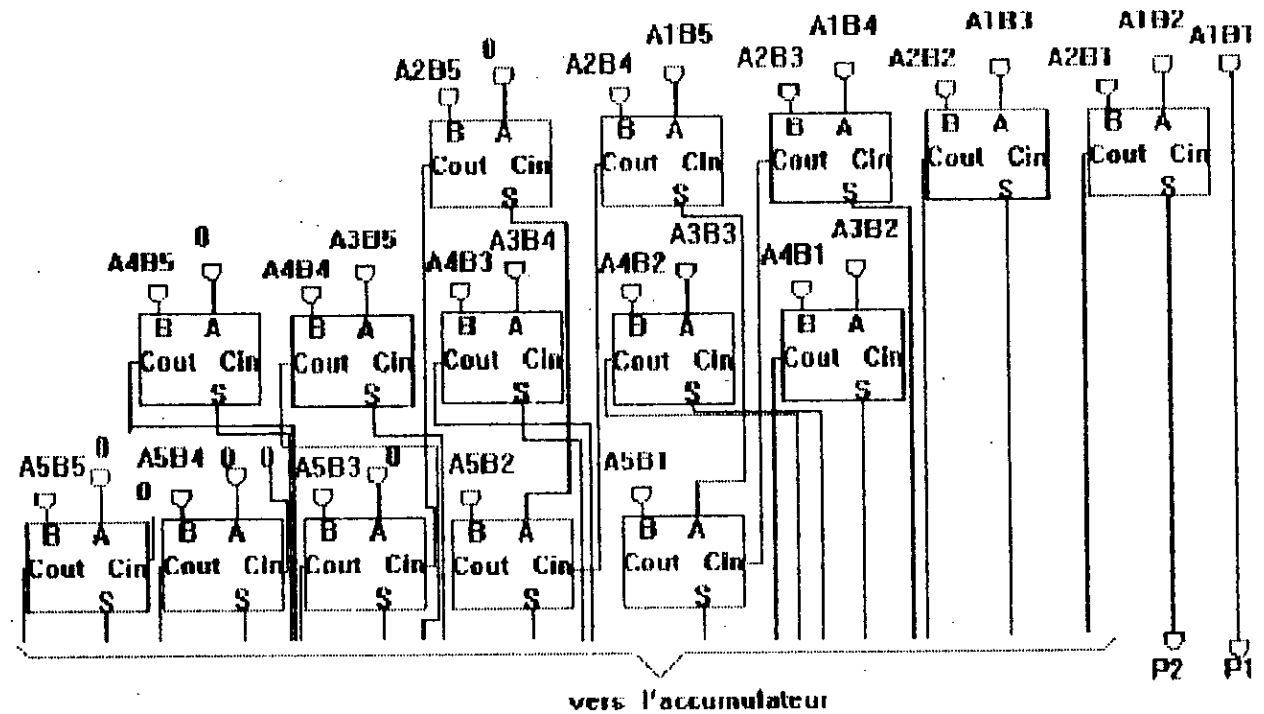


figure 2.7: Réseau d'additionneurs à propagation alternée (ACSA)

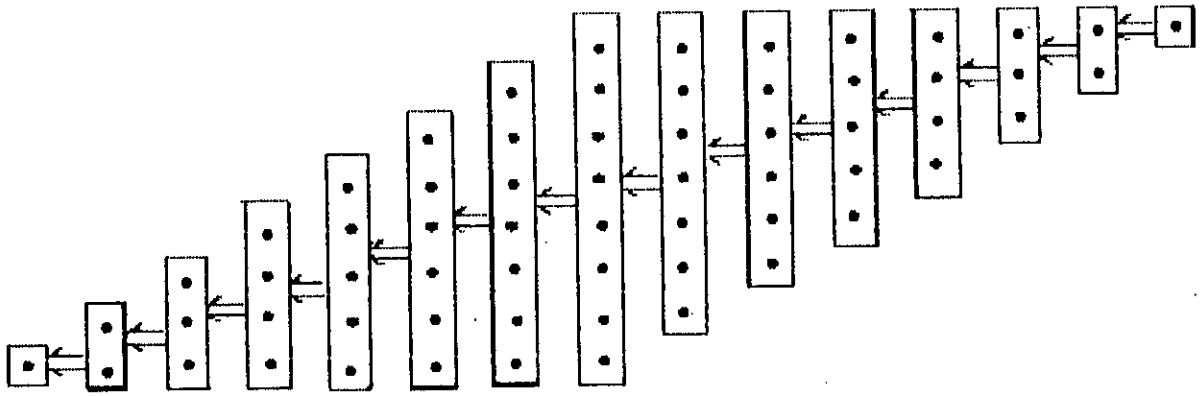


figure 2.8

Chaque rond représente un bit du produit partiel. les rectangles sont des compteurs parallèles qui comportent des bits de même poids. Les flèches horizontales sont un ensemble de retenues d'une colonne à l'autre . Plusieurs manières de matérialiser ce principe sous forme d'arbre d'additionneurs sont possibles. En voici quelques unes :

2.5.2.1. Arbre de Wallace

La méthode de Wallace [17] n'est qu'une optimisation du point de vue vitesse et surface du réseau d'additionneurs à sauvegarde de retenue (Figure 2.9). Dans cette méthode, chaque additionneur du premier étage reçoit trois éléments de chaque colonne de la matrice des produits partiels, ce qui exige la disponibilité simultanée des produits partiels. Ces trois éléments sont réduits à deux, en une somme qui se propage verticalement et une somme qui se propage selon le principe de sauvegarde de retenue. Ainsi, pour les étages qui suivent, chaque additionneur ne reçoit qu'un seul élément. Parfois, l'emploi d'un demi-additionneur suffit quand il n'y a pas de retenue de l'étage précédent. Cette façon d'organiser le réseau a une conséquence négative sur la régularité à cause des interconnexions. Néanmoins, cette architecture exige moins d'espace et plusieurs additions sont élaborées en parallèle augmentant ainsi la vitesse.

2.5.2.2. Arbre de Dadda

Une autre façon d'organiser le réseau d'additionneurs est décrite par le méthode de Dadda , (Figure 2.10). Cette méthode [18] est très semblable à celle de Wallace du fait que toutes les deux ont le même nombre d'étages et que toutes les deux exigent le même nombre d'additionneurs, Table 5.

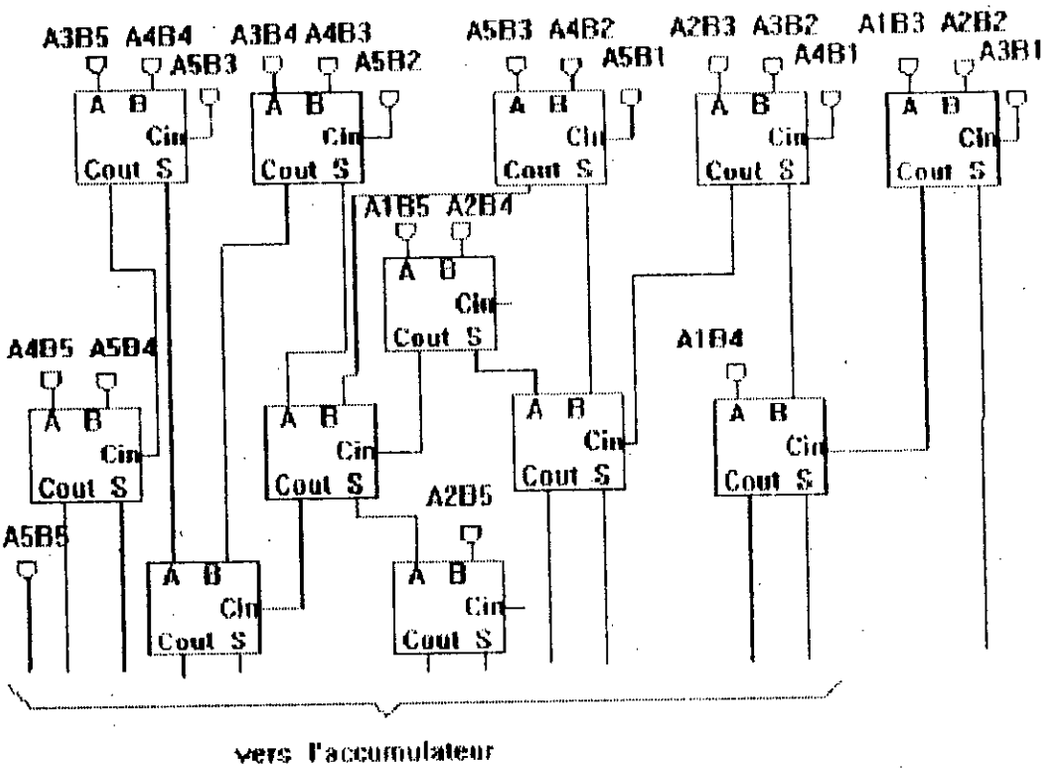


figure 2.9

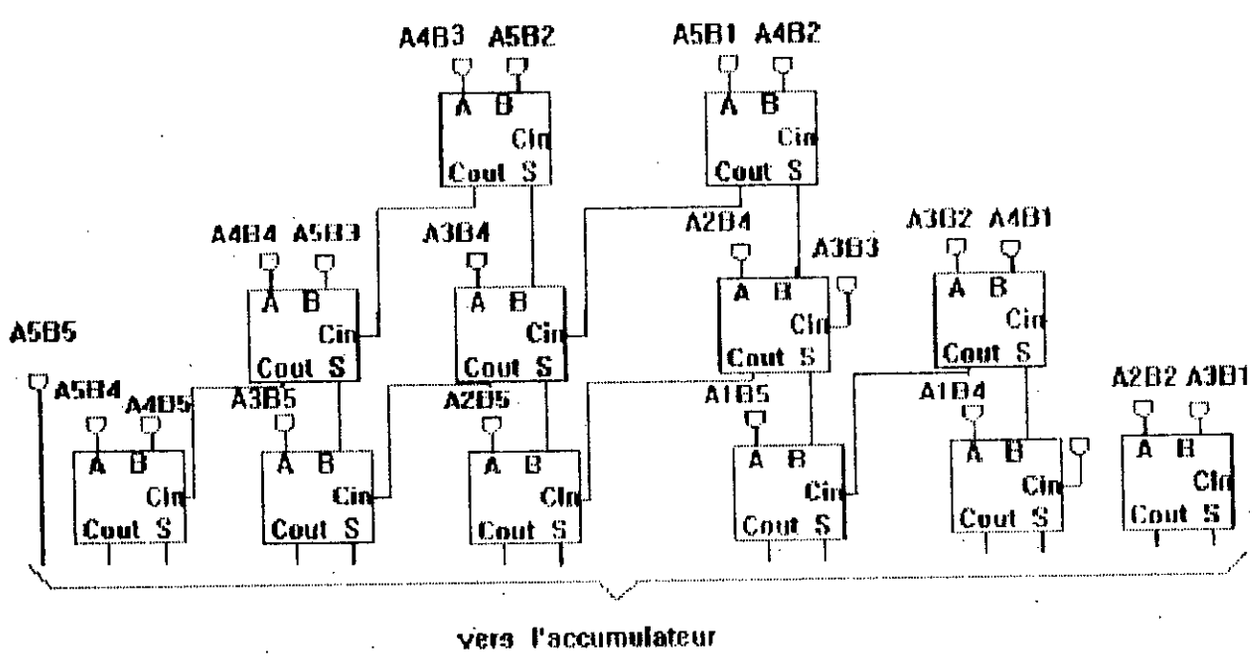


figure 2.10: L'arbre de Dadda

Composée à la méthode de Wallace, cette méthode n'apporte aucune différence en terme de vitesse ou surface à part une plus grande complexité d'interconnexions qui altère d'une manière très significative la régularité et rend de ce fait préférable l'utilisation de la méthode de Wallace.

2.5.2.3. Arbre mixte

De toutes les méthodes de sommation présentées jusqu'ici, cette méthode est la plus performante côté vitesse du fait qu'elle met en combinaison plusieurs techniques de parallélisme, Figure 2.11.
Elle permet de réduire successivement en une somme et une retenue une colonne j de la matrice des produits partiels à travers un nombre d'étages relativement petit, et qui grâce à une technique nouvelle de sauvegarde de retenue qui consiste à regrouper les retenues générées à un étage i de la colonne j pour les injecter à l'étage i+1 de la colonne j+1, éliminant ainsi d'une façon plus astucieuse le besoin de propagation de la retenue et augmentant du même coup le parallélisme qui se traduit par une amélioration de la vitesse , mais aussi par une très grande altération sur la régularité, conséquence de l'enchevêtrement des interconnexions nécessaires à l'acheminement des retenues. La technique de propagation alterné pourrait être combiné à cette dernière en faveur de la vitesse, mais ceci introduit d'avantage d'irrégularités et rend l'implémentation VLSI très difficile.

De cette étude de quelques méthodes de sommation représentatives de l'ensemble des méthodes existantes se dégage certaines remarques et observations qui suscitent quelques réflexions.

Méthode de sommation	Nombred'étages n:nbre de bits	Nbre d'add du réseau	Taille Accumulateur	Régularité
CPA	3n-1	n(n-1)	0	très bonne
CSA	n-1	n(n-1)	n	bonne
CASA	n/2	n(n-2)	2n-2	médiocre
Wallace	log n/log (1.5)	(n-1)(n-2)	2n-2	mauvaise
Dadda	log n/log(1.5)	(n-1) (n-2)	2n-2	mauvaise
Mixte	log n/log (1.7)	(n-1) (n-2)	2n-2	très médiocre

Tableau 2.5

De part la diversité de ces méthodes et les spécificités de chacune, il est à noter qu'elles diffèrent fondamentalement par deux critères ,à savoir :Vitesse et Régularité. Ces deux critères vont toujours de pair et l'amélioration de l'un

ne se fait qu'au détriment de l'autre (Table 5). Notons que l'absence de régularité dans une architecture a les conséquences suivantes :

- La disposition de cellules n'est pas compacte. La perte de surface lors de l'implémentation sera donc importante.
- Les connexions entre cellules seront très longues et donc introduisent des délais non négligeables à cause des capacités parasites.
- Le temps exigé par l'implémentation sera plus long.
- La génération automatique deviendra très difficile.

Pour une implémentation VLSI, ce dont on a besoin est une architecture rapide ,mais aussi régulière permettant une occupation moindre de surface et un temps d'implémentation réduit. Si on considère la vitesse comme seul critère, l'arbre mixte serait à notre avis le meilleur choix, ce qui n'est pas le cas pour notre application où la régularité est aussi importante que la vitesse.

CHAPITRE III

LES DIVISEURS

3.1. La division

Les développements récents de la technologie VLSI ont motivé la conception d'algorithmes puissants aptes pour une implantation dans un environnement VLSI.
La division est l'un de ces algorithmes qui jouent un rôle important dans divers domaines tels les calculateurs et les ordinateurs . Une division à très grande vitesse est donc indispensable à ces deux systèmes digitaux.
Notre étude s'est portée sur deux algorithmes.

3.2. Division binaire sans restauration utilisant un réseau cellulaire (Non restaurating binary division using à cellular array) [19]

Ce circuit utilise une cellule arithmétique qui est un additionneur complet (Full adder) . Cette cellule peut aussi être utiliser dans un réseau itératif pour une division sans restauration . La fig 3.1 illustre cette cellule, dont les équations sont les suivantes.

$S = A \oplus C \oplus (B \oplus D)$
$T = AC \oplus (B \oplus D)(A \oplus C)$
$P = B$
$F = D$

S: Sortie de l'additionneur (output)
P: Bit du diviseur
T: Bit du carry-out
C: Bit du carry-in
F,D: Bits de contrôle

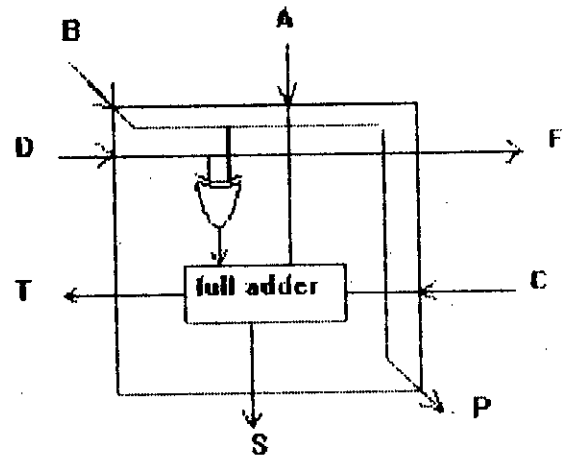


figure 3.1: Cellule de Guild

Cette cellule présente quatre sorties et quatre entrées avec C étant une entrée retenue (carry in). Cette cellule peut être réaliser par

un full adder et une porte ou-exclusive comme indiqué dans le schéma.
Le circuit de la division sans restauration est illustré en fig 3.2

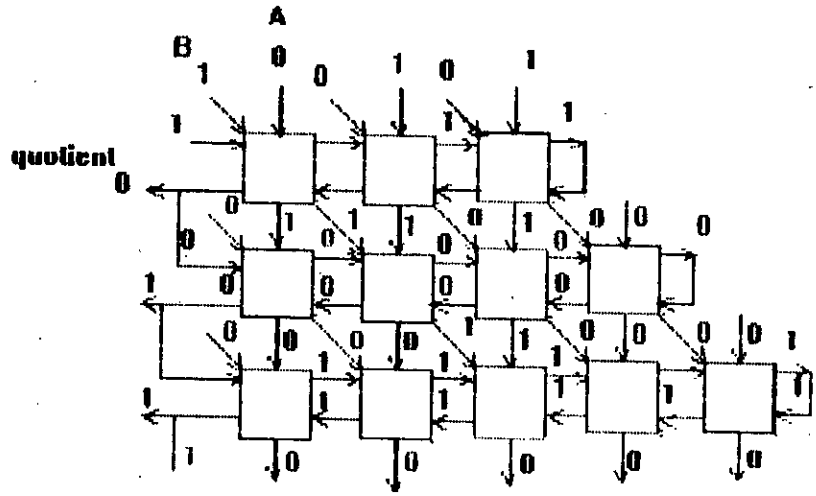


figure3.2: circuit diviseur 3 bits

La soustraction dans chaque ligne est réalisée, en utilisant l'arithmétique complément à 2. Ainsi X - Y est traité comme $X+\overline{Y}+1$. La complémentation de Y, quand elle est nécessaire, est accomplie par l'état de contrôle D. L'addition de 1 est accomplie en mettant C=1 pour une cellule de la ligne, ayant le bit le moins significatif. L'algorithme de division sans restauration peut être résumé comme suit:
a) Soustraire le diviseur B du dividende A, c'est à dire réaliser $A+\overline{B}+1$. pour donner une somme R1.
b) Si on n'a pas de résultat du carry ou le reste R1 est négatif, 0 est introduit comme étant un bit quotient. c) Si carry out a un résultat, on introduit 1 comme étant un bit quotient.

Dans le circuit de la fig.3.2, les carry-out des cellules de bits les plus significatifs produisent les bits quotients et aussi les bits D de contrôle des lignes suivantes. Le carry in de la cellule de bit le moins significatif de chaque ligne est obtenue en connectant les sorties F aux entrées C de la cellule.

A la fig 3.2, la division de 011 par 100 est traitée. tous les états intermédiaires sont aussi illustrés.
Le nombre de cellule requises pour diviser deux nombres de n bits et produisent un quotient de m bits et donnée par :

$$\frac{m(m+2n-1)}{2}$$

Si ,dans chaque cellule , le full adder a un retard Ta secondes and et la porte ou-exclusive a un retard de Tg secondes , le délai total pour obtenir un quotient de m bits est :

$$\frac{m(m+2n-1)}{2} Ta + m\,Tg \quad \text{secondes}$$

3.3. Division binaire sans restauration avec les techniques retenue (carry save) et carry lookahead [20]

3.3.1. Préambule :

Cet algorithme apporte une amélioration du diviseur sans restauration en utilisant les techniques de carry save et carry lookahead (introduites par Cappa - Hamacher [4] .Cette amélioration apparaît au niveau de la vitesse du circuit et du nombre de portes utilisées .

Le diviseur binaire classique sans restauration peut être implémenté avec un circuit itératif construit à partir de cellules conventionnelles additionneur/soustracteur contrôlées (C.A.S) [19] .La ligne de contrôle K est mise à 1 à chaque ligne pour réaliser la soustraction et mise à 0 pour réaliser l'addition .Le carry out de la cellule C.A.S la plus à gauche détermine la valeur du quotient , et détermine en même temps l'opération add/soustraction pour la ligne suivante .

L'inconvénient de ce circuit diviseur sans restauration est la vitesse lente du temps d'exécution due au fait que les lignes sont connectées en série et la retenue est propagée comme une ondulation le long de la ligne .

Une amélioration de la construction de ce circuit a été suggéré par Cappa - Hamacher .Les techniques de carry - save et carry - lookahead sont utilisées pour éliminer la propagation de la retenue sous forme d'ondulation .Cependant cette construction nécessite trois types de cellules .

Ceci augmente la complexité de la circuiterie et réduit l'aptitude d'une implementation VLSI , à cause du manque d'homogénéité et de modularité dans la structure cellulaire.

Dans ce chapitre, deux types de cellules de base sont nécessaires pour la construction du circuit suggéré . Ceci convient pour une implémentation en VLSI .De plus , la réduction du nombre de portes logiques et de pins d'Entrées/Sorties dans la cellule de base améliore la vitesse d'exécution.

3.3.2. Structure des cellules de base

Deux types de cellules logiques de base sont utilisés pour la construction du circuit diviseur à cellules cary save . Une cellule notée A, et une autre cellule notée CLA (carry lookahead); la cellule A est utilisée pour l'arithmétique alors que CLA est utilisée pour accélérée la propagation du carry.

La fig 3.1. montre le schéma logique de la cellule A' .

Les équations logiques du digit de sortie somme S et le carry out C de la ième ligne et jème colonne sont données par :

$$Sij = Sji\text{-}1 \oplus Cji\text{-}1 \oplus (Dj \oplus Ki) \qquad (1)$$

$$Cij\text{-}1 = Sji\text{-}1\ Cji\text{-}1 + (Dj \oplus Ki)(Sji\text{-}1 \oplus Cji\text{-}1) \qquad (2)$$

où Sij-1 et Cij-1 sont la jème somme et carry out de la (i-1)ème ligne ,Dj est le jème bit du diviseur ,et Ki le signal de contrôle de l'opération de la ième ligne .

Ki = 0 pour une addition

Ki = 0 pour une soustraction

La seconde cellule de base, est la cellule CLA comme indiqué dans la fig 3.2 .La sortie de la cellule du carry lookahead ‹(CLA), C peut être exprimée par:

$$Cil = Ci0 + Ci1Si1 + Si1Si2Ci2 + \ldots\ldots\ldots + Si1Si2\ldots Sin\text{-}1Cin\text{-}1 \qquad (3)$$

où Cij et Sij , i=0,1,.....,n-1 et j=0,1,.....,n-1 .sont définis comme précédemment. Cependant ils diffèrent dans le sens .Ici Ci peut être considérer comme un digit de génération de carry alors que Si est considérer comme étant un bit de propagation du carry .

A partir de ce point de vue ,l'équation (3) peut être considéré comme une union de chaînes de propagation de carry .Chaque chaîne est débutée par les digits de propagation du carry et doit être terminée par un digit de génération du carry .Le premier terme dans l'équation (3) est considéré comme un digit de génération direct du carry . Tout les autres termes sont des générations indirects du carry car ils sont propagés à travers une chaîne de digits sommes .Si dans une chaîne de digits de propagation ,Si1Si2...Sik est interrompue par un digit "0" ,alors cette chaîne et les autres qui s'en suivent seront nulles .Ce fait permet à la CLA d'être facilement détectable à partir de deux vecteurs Ci et Si dans le circuit .

3.4. Circuit diviseur à additionneur/soustracteur à sauvegarde de retenue (carry save adder/substractor).

Avec les cellules de bases spécifiées dans la section précédente ,le schéma de l'additionneur ou soustracteur à carry save avec des cellules de carry lookahead est montré en fig 3.3.

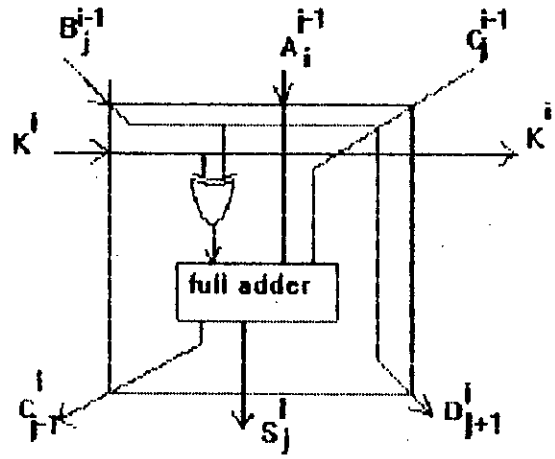


figure 3.3: Cellule de Guild

Il utilise les cellules A' et les cellules CLA (fig 3.4) ,en plus d'un certains nombre de portes XOR et d'inverseurs .La première ligne du circuit (fig3.5) réalise la soustraction ou l'addition suivant le signe du dividende .L'addition est réalisée si le dividende est positif .Cette opération arithmétique est contrôlée par la ligne K qui est connectée à l'entrée de la retenue initiale la plus à droite de chaque ligne .
Ceci signifie que la complémentation à deux est réalisée quand K=1 .Il est à noter que le digit quotient qi de la ième ligne ,devra être de même signe que la (i+1)ème ligne de contrôle ,Ki+1 .

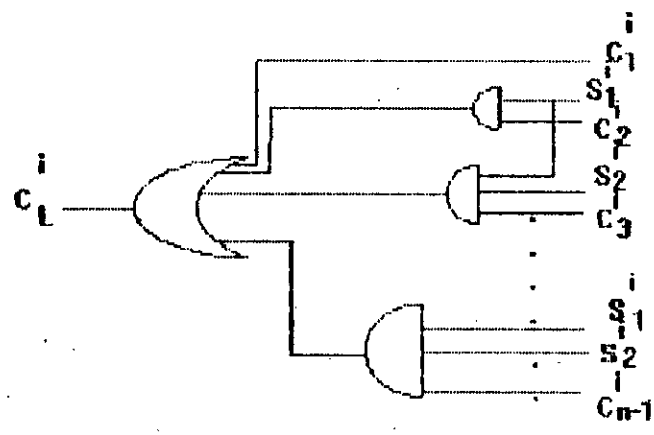


figure 3.4: cellule carry lookahead

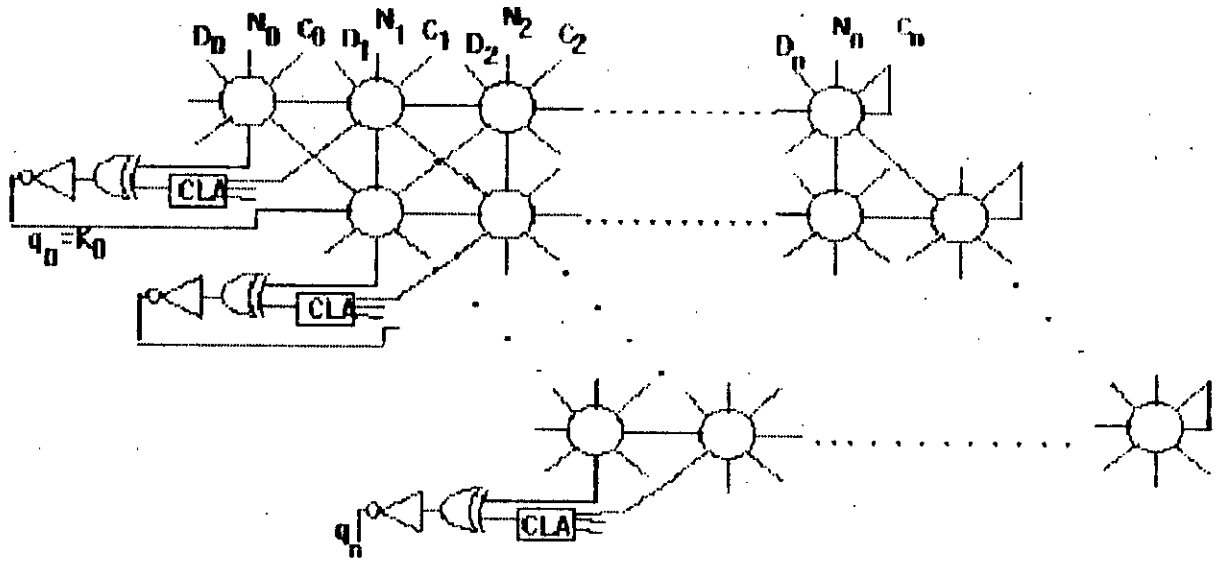


fig3.5: Amélioration du circuit diviseur avec CLA

3.5. Comparaison de la vitesse de l'opération et les besoins hardware avec d'autres cellules de circuits diviseurs

La vitesse peut être mesurer en termes de nombres de retard des portes de bases , et les besoins du hardware peuvent être évaluer en termes de nombres de portes logiques nécessaires à la construction du circuit . Pour un circuit diviseur 2n par 2n sans restauration et sans circuit de CLA , $(n+1)^2$ cellules sont nécessaires , où n est le nombre de bits du diviseur ou du quotient , les cellules CAS utilisées dans la construction des diviseurs sans restaurations nécessitent un retard de 3T par cellule . Le plus mauvais cas de retard du signal est le temps nécessaire pour générer les derniers bits quotients significatifs (LSQB) qn de la dernière ligne du circuit . A cause de la nécessité que les restes partiels successifs , d'être générés ligne par ligne de manière sérielle , le temps total nécessaire pour générer le LSQB à la dernière ligne est

$D_{non\text{-}restauring} = 3(n+1)2\,T$ sans CLA

Pour un circuit de division 2n par 2n sans restauration implementé avec des cellules carry -save et augmenté dans chaque ligne par la logique CLA (cappa Hammacher) [2] , il nécessite n (n+1) cellules A , (n+1) circuits CLA , et (n+1) portes XOR . Le temps total de divisions

devra être (n+1) fois le temps de retard de chaque ligne en plus de l'inverseur .

$$TCLA\ (\text{circuit diviseur}) = (n+1)\ T_{ligne} + T$$

où Tligne est le retard de la ligne et T est attribué à l'inverseur de la dernière ligne pour produire le dernier bit quotient . Si on suppose un seul niveau du CLA , on a :

$$T_{ligne}\ (1\ \text{seul niveau de CLA}) = 3T + 6T = 9T.$$

où 3T provient de la sortie de la porte XOR et 6T provient de la cellule Carry Lookahead logique.

d'où le temps total de la division pourra être :

$$TCLA\ (\text{circuit diviseur}) = (n+1)\ 9\ T+T= (9n+10)T$$

Pour le cas de notre projet de fin d'études le retard de chaque ligne pour une seule CLA est

$$T_{ligne}\ (\text{un seule niveau CLA}) = 3T +4T = 7T$$

où 3T est dû à la sortie de la porte XOR et 4T est dû au circuit logique CLA . De plus , du moment que le vecteur C est égal à zéro au début du circuit , C_0^L peut être appliquer directement à la l'entrée de S^0_0 (voir fig 3) , et ainsi le retard de la porte XOR (3T) peut être économiser . L e temps total de la division est dans ce cas :

$$D\ CLA\ (\text{circuit de division}) = (n+1)\ 7T + T -3T$$

Pour évaluer le coût des portes des cellules CLA pour le circuit de Cappa - Hammacher des circuits considérés . Un - niveau CLA nécessite au plus (n -1) portes . En sommant ces nombres de portes , le nombre total de portes pour un diviseur 2n par 2n devient :

$$\text{nombre de portes (un niveau de CLA)} = 17n\ (n+1) + (n+1)\ (n+9+1) +1$$
$$= 18\ n2 + 28n +11$$

Pour le cas du circuit suggéré , chaque cellule A contient 15 portes , pas de cellules de signes requises (cellule S) . Le nombre total de portes pour une division 2n par 2n avec un niveau de CLA est :

$$\text{nombre de portes (un niveau de CLA)} = 15\ (n+1)2+ (n+1)\ (n) +1$$
$$= 16\ n2+ 31n +16$$

En comparant ces 2 dernières équations , on peut voir que si n (longueur du diviseur) est large , le profil serait plus grand pour la dernière équation .

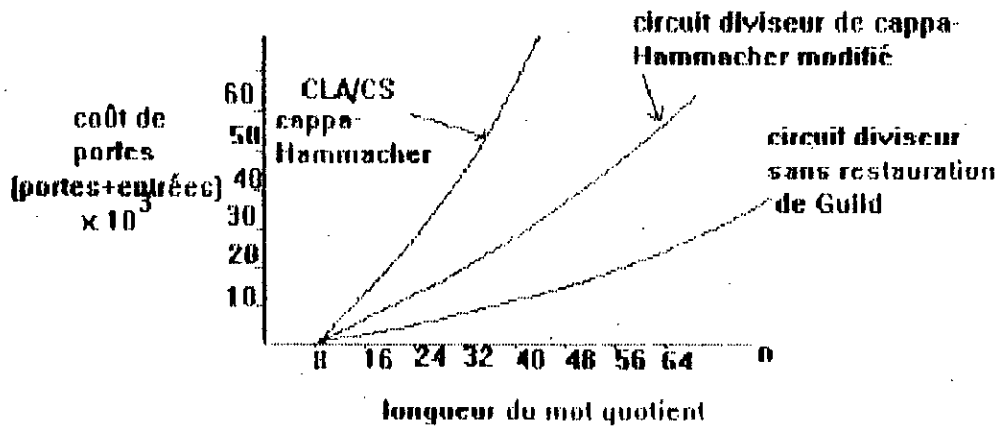
La table 1 résume le nombre total de portes et le temps de division associés à chacun des circuits diviseurs .

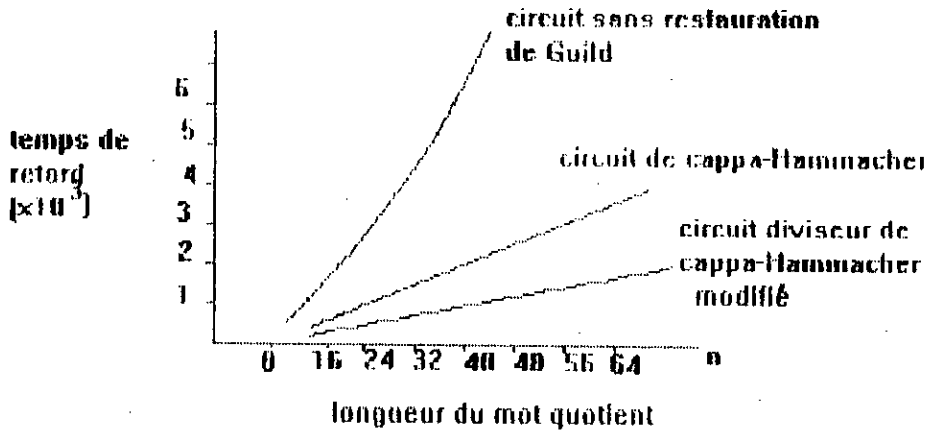
Les courbes graphiques des vitesses et coûts de portes sont aussi donnés :

Type de circuit diviseur	Nombre de portes	Temps de division
Sans restauration (GUILD)	17(n+1)2	$3(n+1)^2T$
Sans restauration avec 1-niveau CLA et sauvegarde de retenue (Cappa-Hammacher)	18n2+28n+11	9(n+10)T
Sans restauration modifié avec 1 niveau CLA et sauvegarde de retenue	16n2+31n+16	7(n+5)T

tableau 3.1

Table 1 : Nombre de portes et retards de trois types de diviseurs
n= longueur du quotient (diviseur)




circuit sans restauration de Guild
circuit de cappa-Hammacher
circuit diviseur de cappa-Hammacher modifié
temps de retard (×10⁻³)
5
4
3
2
1
0
16
24
32
40
48
56
64
n
longueur du mot quotient

CHAPITRE IV

CONCEPTION DU MULTIPLIEUR ET DU DIVISEUR

4.1. Généralités sur les contraintes de conception

comme toute conception de système VLSI, la conception d'un multiplieur ou d'un diviseur est directement influencée par trois facteurs interdépendants: régularité, vitesse et surface.
Pour chacun de ces facteurs, l'influence s'exerce d'une façon plus ou moins prononcée suivant la nature de la conception et les objectifs que l'on se fixe. Ainsi la conception d'un multiplieur ou d'un diviseur impose le choix d'un algorithme de multiplication ou de division dont l'architecture respecte la contrainte de régularité sans pour autant sacrifier les deux autres contraintes. En somme, il s'agit d'établir un bon compromis permettant d'avoir un multiplieur ou un diviseur ayant une bonne vitesse de calcul et une surface minimale. Pour concilier ces différentes contraintes, une approche consiste à décomposer la multiplication ou la division en sous-pérations complémentaires, puis à déterminer l'action qu' exerce chaque contrainte sur la sous-opération concernée.
Comme il a été dit dans le chapitre précédant, la multiplication peut être décomposée en génération de produits partiels et en sommation de ces produits partiels. A ce niveau là, certaines conclusions peuvent être déduites: a) le temps de multiplication est proportionnel au nombre des additionneurs.
b) l'architecture du multiplieur découle principalement du choix de l'arbre de sommation. Ainsi, pour améliorer le temps de multiplication il suffit de réduire le nombre des additions. Il existe généralement deux manières d'atteindre cet objectif: la première consiste à réduire le nombre des produits partiels. Cette approche peut être réaliser par la technique de recodage (vue au chapitre III), mais cette technique augmente la complexité du système à intégrer car elle nécessite d'autres modules qui réalisent la fonction recodage et qui sont généralement très difficiles à intégrer.

L'autre manière de réduire le nombre des additions est de réduire le nombre des étages additionneurs dans l'arbre de sommation. Cette approche peut être implémenter en utilisant l'arbre mixte, l'arbre de Dadda ou de Wallace. Cependant ce genre de choix introduit une irrégularité telle qu'elle rend la structure du multiplieur non régulière. A ce niveau là, deux choix semblent les plus prometteurs: Le réseau à sauvegarde de retenue et le réseau à propagation alternée. La raison pour laquelle notre choix s'est porté sur le réseau à sauvegarde de retenue est la régularité de ce réseau. Celle ci réduit considérablement la complexité des interconnexions par rapport a un réseau à propagation alternée, ainsi qu'elle s'accompagne d'une utilisation optimale de la surface et une diminution de l'effet des capacités parasites sur le temps de réponse du multiplieur. En effet, le choix de l'algorithme de Baugh-Wooley combiné à un réseau à sauvegarde de retenue nous semble le plus apte à concilier des contraintes plus ou moins antagonistes et qui impose un choix très restrictif sur le nombre des solutions viables.

Pour en ce qui concerne le diviseur on a choisi un diviseur sans restauration pour diminuer le nombre de cellules utilisées combiné à une technique de retenue anticipée "Carry-Lookahead" (CLA) pour satisfaire aux contraintes citées auparavant. Une nette amélioration a été apporté au diviseur de Cappa-Hammacher sans restauration en utilisant les techniques de carry-save et carry-Lookahead. Comparée à d'autres diviseurs, cette amélioration apparaît au niveau de la vitesse d'exécution , ainsi que la réduction du nombre de portes logiques. De plus deux types de cellules sont utilisées au lieu de trois dans d'autres diviseurs.

Il est clair que les systèmes VLSI, quelque soit leur nature, intégrant très étroitement avec les contraintes de régularités, de vitesse et de surface et que cette interaction est déterminante pour le succès ou l'échec de toute conception.

4.2.Multiplieur

Nous avons éxaminé dans les chapitres précédents le problème de la multiplication et leurs algorithmes offrants les meilleures réponses à ce problème.
Notre choix s'étant porté sur l'algorithme de Baugh-Wooley [10] combiné à un réseau "Carry-Save", nous allons dans ce chapitre étudier en détail cet algorithme et déterminer les différents éléments qui constituent l'architecture du multiplieur. Nous allons essayer également d'apporter une réponse à deux problèmes importants: l'éxtention de signe et la propagation de la retenue dans l'accumulateur.

4.2.1.Présentation des différentes céllules

par respect à l'esprit de la méthode développée dans l'article "conception structurée" [1], il nous a semblé pratique de présenter l'architecture comme étant un ensemble de modules élémentaires (voir annexe) communiquant et coopérant pour la réalisation d'une même tâche. Ces modules sont formés à partir de céllules de base de la bibliothèque du MCE. La tâche attribuée à chaque module représente une sous-fonction de la multiplication.

4.2.2. Les additionneurs

La céllule FA (Full Adder) intervient pour une bonne partie dans le délai de réponse du multiplieur. L'additionneur FA doit à la fois être compact et présenter un délai le plus faible possible.
Malheureusement ce délai est fixé par le fondeur de silicium (firme MCE), mais cela n'empêche pas de donner une optimisation qui portera sur deux points selon que la céllule FA se trouve dans le réseau "Carry-Save" ou fait partie de l'accumulateur, étant donné que c'est ce délai qui fixe le temps de réponse de l'accumulateur.

4.1.3. Problème de l'extension du signe

La méthode de Baugh-Wooley fournit un moyen direct pour la génération des produits partiels. et cela pour

Dans la multiplication binaire le produit de (n+m) bits $P=(P_{N+M-1},P_{N+M-2},\ldots,P_0)$ est formé à partir de la multiplication d'un multiplicande de m-bits $y=(Y_{M-1},Y_{M-2},\ldots,Y_0)$ par le multiplicateur de n-bits $x=(X_{N-1},X_{N-2},\ldots,X_0)$. L'opération de multiplication est souvent éffectuée selon la figure 6.1. Le AND de chaque bit du multiplicateur et de chaque bit du multiplicande est formé pour générer les bits du produit partiel qui sont par la suite sommés pour former le produit final.

La difficulté en complément à deux est lié aux signes du multiplicande et du multiplicateur. Supposons Y_v la valeur du multiplicande y, et X_v la valeur du multiplicateur x. Pour la représentation en complément à deux de X_v et Y_v est donné par:

$$Y_v = Y_{m-1}2^{m-1} + \sum_{i=0}^{m-2} Y_i 2^i \qquad (1a)$$

$$X_v = X_{n-1}2^{n-1} + \sum_{i=0}^{n-2} X_i 2^i \qquad (1b)$$

La valeur final du produit est:

$$P_v = P_{m+n-1}2^{m+n-1} + \sum_{I=0}^{m+n-2} P_i 2^i = X_v Y_v \qquad (2)$$

$$P_v = (X_{n-1}Y_{m-1}2^{m+n-1} + \sum_{I=0}^{n-2}\sum_{j=0}^{m-2} X_i Y_j 2^{i+j}) - (\sum_{i=0}^{m-2} X_{n-1}Y_i 2^{n-i+1} + \sum_{i=0}^{n-2} Y_{m-1}X_i 2^{m-i+1})$$

Quand on forme le produit P par addition des produits partiels, le signe du bit du produit partiel doit être considéré. En particulier le signe de: $X_{n-1}Y_i$ et $Y_{m-1}X_i$ pour i=0,1,....,n-2 qui sont négatifs.

Tous les produits partiels avec un signe négatif sont placés à la derniere lignes. Le produit est formé à partir de l'additionneur de n-2 premiers produits partiels et de soustraire les deux dernieres lignes.

Au lieu de soustraire le produit partiel dont le signe est négatif, la négation des produits partiels peut être additionner.

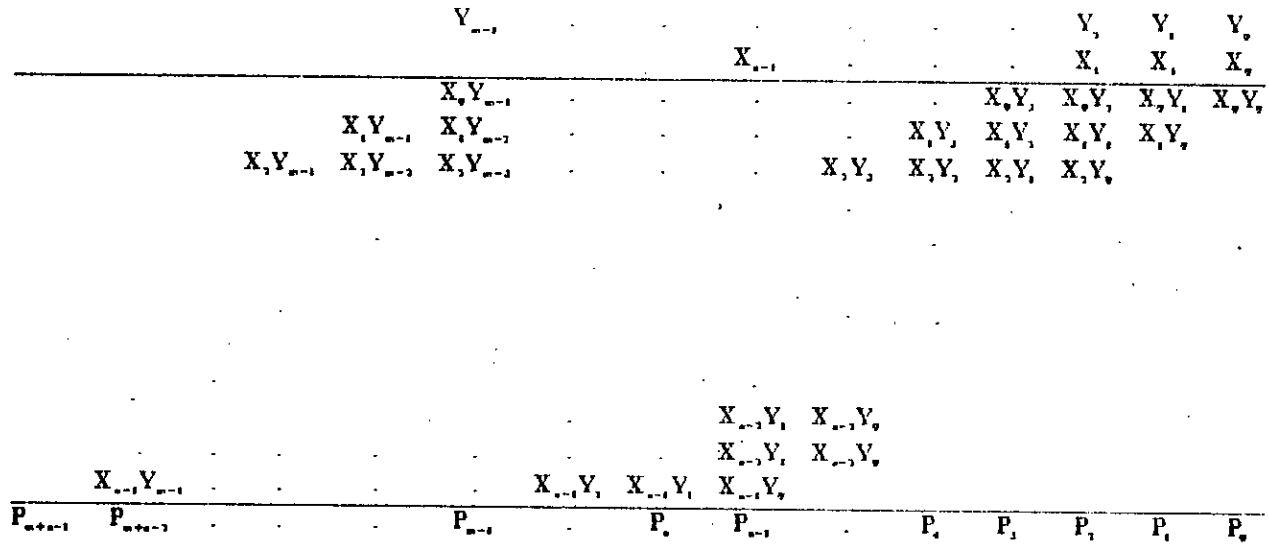


figure 4.1: La multiplication conventionnelle en complément à de deux m x n bits

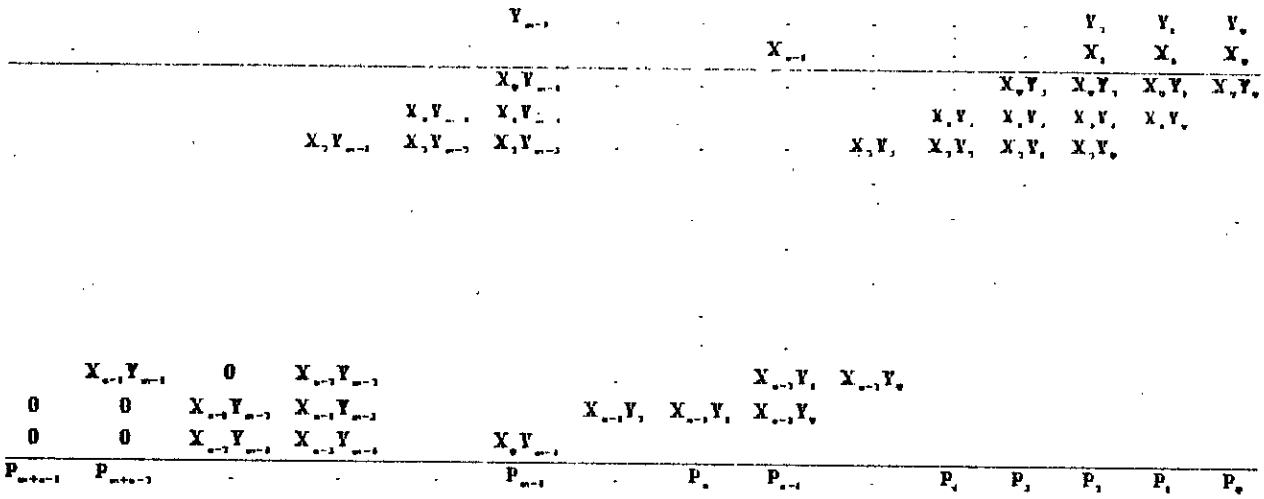


figure 4.2: Séparation des bits du produit partiel positif/négatif

Comme étant connu la négation en complément à deux d'un nombre $Z=(Z_{h-1},Z_{h-2},.....,Z_0)$ où sa valeur est Z_v est

$$-Z_v = 1-\overline{Z}_{k-1}2^{k-1}+\sum_{i=0}^{k-2}\overline{Z}_i 2^i \qquad (3)$$

où $\overline{Z}_i$ est le complément de Zi , ainsi, la soustraction de

$$2^{n-1}(-1.2^m+1.2^{m-1}+1+\sum_{i=0}^{m-2}\overline{Z}_i 2^i \qquad (4)$$

peut être remplcée par l'addition de:

$$2^{n-1}(-1.2^m+1.2^{m-1}+1+\sum_{i=0}^{m-2}\overline{X_{n-1}Y_i}\,2^i \qquad (5)$$

Ainsi le produit partiel de la derniere ligne de la (figure 4.2) contenant $0\ 0\ X_{n-1}Y_{m-2}\ X_{n-1}Y_{m-3}$ .....$X_{n-1}Y_0$ est remplacée par:
$1\ 1\ \overline{X_{n-1}Y_{m-2}}\ \ \overline{X_{n-1}Y_{m-3}}$ .....$\overline{X_{n-1}Y_0}$ avec un "1" additionné à la colonne P_{n-1}. De même pour la ligne contenant
$0\ 0\ X_{n-2}Y_{m-1}\ X_{n-3}Y_{m-1}$ .....X_0Y_{m-1} est remplacée par:
$1\ 1\ \overline{X_{n-2}Y_{m-1}}\ \ \overline{X_{n-3}Y_{m-1}}$ .....$\overline{X_0Y_{m-1}}$ avec un "1" additionné à la colonne P_{m-1} Le résultat de la substitution de (5) dans (4) dans la (figure 4.2) est une non uniformité de génération du produit partiel ce qui affecte évidement la régularité, car une partie des produits partiels du multiplieur sont formés à partir de portes NAND alors que d'autres nécéssitent seulement des portes AND. Pour simplifier cette situation, les équivalences suivantes sont utilisées: Notons d'abord que (5) a les valeurs suivantes

$$\begin{cases} 0 & \text{pour } X_{n-1}=0 \\ 2^{n-1}(-2^m+2^{m-1}+\overline{X}_{n-1}2^{m-1}+X_{n-1}+\sum_{i=0}^{m-2}X_{n-1}\overline{Y}_i 2^i) & \text{pour } X_{n-1}=1 \end{cases}$$

à partir de (6) l'équation (5) peut être écrite:

$$2^{n-1}(-2^m+2^{m-1}+\overline{X}_{n-1}2^{m-1}+X_{n-1}+\sum_{i=0}^{m-2}X_{n-1}\overline{Y}_i 2^i) \qquad (7)$$

Comme les deux derniers lignes du produit partiel dans la (figure 4.2) sont de même forme que (5), alors l'équation (7) est substituée dans ces lignes d'où le nouveau algorithme de production (figure 4.3).

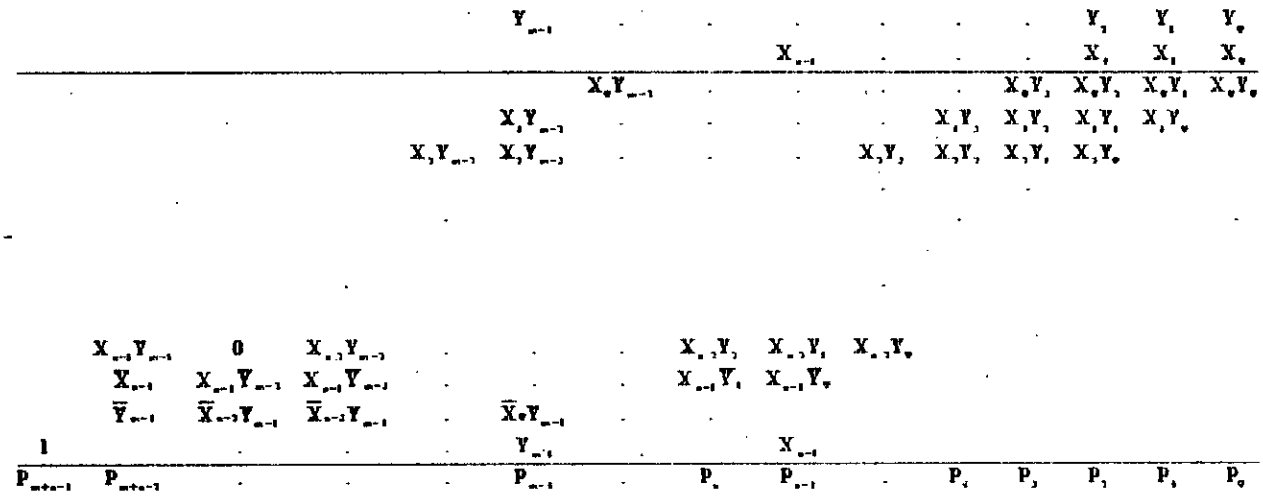


figure 4.3: Algorithme avec bits du produit partiel tous positifs

La principale caracteristique des bits du produit partiel de l'algorithme (figure 4.3) est l'uniformité (régularité). Les deux avantages de cette uniformité sont:

1) Les produits partiels sont obtenus par un simple AND du multiplieur avec le multiplicande

2) Chaque produit partiel a un coefficient positif.

Ainsi le produit est formé uniquement avec les portes AND et les portes FA (Additionneur). Aucune soustraction ni utilisation de portes NAND n'est nécéssaire pour former $\overline{X_i Y_j}$ qui affecte la régularité. Aussi les cinq extra produit partiel de l'algorithme qui sont $X_{n-1}, \overline{X}_{n-1}, Y_{n-1}, \overline{Y}_{n-1}$ et "1" peuvent être inclu dans l'implémentation de l'algorithme sans augmenter le temps de réponse du multiplieur.

On peut aussi déduire du cheminement qui précède les termes correcteurs opérations mixtes. (la table 4.1) résume les differntes combinaisons.

Une remarque reste à faire est que cette discussion demeure valable pour des opérandes de tailles differentes ou de même taille.

MODE	Bit 2n-1 2n-2 2n-3 ..... n-1
SIGNE	$\overline{X}_{n-1}$ $\overline{Y}_{n-1}X_{n-1}$ ..... $\overline{Y}_0X_{n-1}$ 1 1 0 ..... 0 X_{n-1} $\overline{Y}_{n-1}$ $\overline{X}_{n-2}Y_{n-1}$ ..... $\overline{X}_0Y_{n-1}$ 1 1 0 ..... 0 Y_{n-1} $X_{n-1}Y_{n-1}$
NON SIGNE	$X_{n-1}Y_{n-2}$ ..... $X_{n-1}Y_0$ $X_{n-1}Y_{n-1}$ $X_{n-2}Y_{n-1}$ ..... X_0Y_{n-1}
X SIGNE Y NON SIGNE	$\overline{X}_{n-1}$ $\overline{Y}_{n-2}X_{n-1}$ ..... $\overline{Y}_0X_{n-1}$ $X_{n-1}Y_{n-1}$ $X_{n-1}Y_{n-1}$ 1 1 0 ..... 0 X_{n-1}
X NON SIGNE Y SIGNE	$\overline{Y}_{n-1}$ $\overline{X}_{n-2}Y_{n-1}$ ..... $\overline{X}_0Y_{n-1}$ $X_{n-1}Y_{n-1}$ $X_{n-1}Y_{n-1}$ 1 1 0 ..... 0 Y_{n-1}

table 4.1: Les termes correcteurs de l'algorithme de Baugh-Wooley

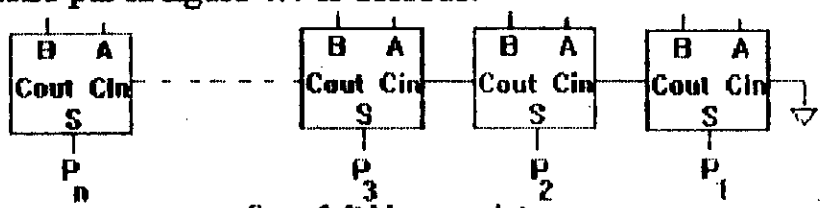
4.2.4. L'accumulateur

L'importance de la propagation de la retenue dans l'accumulateur est capitale. En effet l'analyse de l'architecture du multiplieur révèle que le chemin critique, c'est à dire, le chemin le plus long passe nécéssairement par l'accumulateur ou une partie de l'accumulateur. Aussi, il est raisonable d'étudier certaines techniques d'optimisation de la propagation de la retenue ayant une très bonne régularité ainsi qu'un bon compromis éspace/surface.
Des techniques classiques comme les additionneurs à anticipation de retenue (CLA) ou la chaîne de "Manchester" permettent d'améliorer la vitesse. Leur intégration est cependant couteuse en surface et en compléxité, il est cependant possible de concevoir la même cellule que celle du réseau et d'obtenir un accumulateur très rapide.

4.2.4.1. Principe de réalisation de l'accumulateur

La retenue commence à se propager dans l'accumulateur lorsque toutes les données (somme,retenue) de la derniere rangée du réseau sont stables.

Deux des trois entrées du FA de l'accumulateur (A et B) sont stables et une seule se propage (retenue).
L'accumulateur est construit en méttant en série des additionneurs comme il est illustré par la figure 4.4 ci-dessous.



figur 4.4: L'accumulateur

Solution [21]
Afin de réduire le délai, il suffit d'insérer amplificateur (double inverseur) de façon periodique dans la chaîne tel qu'il est illustré à la figure suivante (fig 4.5).

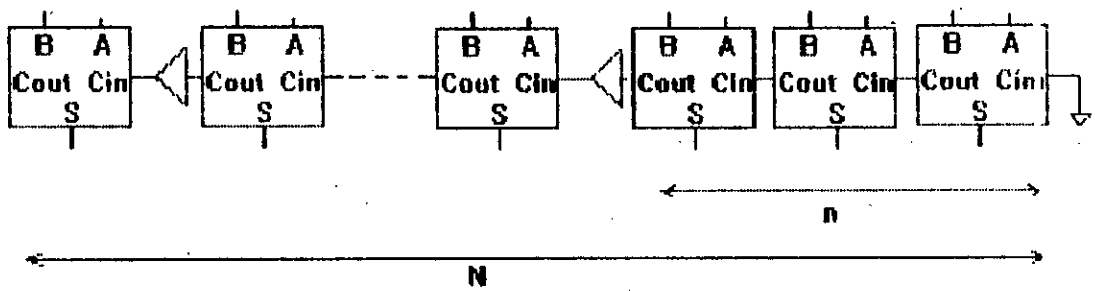


figure 4.5

Il est clair que le délai total peut être réduit en réduisant n, lorsque N est grand. Cependant, si n est trop petit, ajouter des amplificateurs peut faire augmenter le délai au lieu de le réduire.
Cette discussion suggère l'existence d'un optimum. Un calcul théorique [21] montre que l'optimum est obtenu lorsque le délai au travers de l'amplificateur est égal au travers de la chaîne des FA.

4.3.Diviseur

Nous avons examiné dans dans le chapitre III deux types d'algorithmes de division, celui de Guild et de Cappa-Hammacher modifié, ainsi que les caractéristiques de chacun d'eux (extension de signe, nombre de cellules, vitesse d'exécution,...) voir tableau comparatif 3.1.

Notre choix s'étant porté sur l'algorithme de Cappa-Hammacher modifié [20] combiné à des techniques "carry-save" et "carry-lookahead",

étant donné qu'ils apportent une solution aux deux problemes majeurs de la division à savoir: nombre de cellules logiques nécessaires (élevé dans le cas de certains algorithmes tel que celui de Guild) et la propagation de la retenue. Dans le cas de notre projet de fin d'études cet algorithme a été appliqué pour la réalisation d'un diviseur 16/8 Bits.

4.3.1. Présentation des differentes cellules

L'architecture du diviseur étant basée sur basé sur un ensemble de modules élémentaires (voir annexe). Ces modules sont formés à partir de macro-cellules de FA (full adder) et de potes XOR (voir figure 3.3) existant dans la bibliotheque de MCE.

4.3.2. Présentation de l'algorithme

L'algorithme utilise une structure parallele, constituée du réseau de cellules notées A et CLA, dont leurs foctions pricipales sont : l'arithmétique et l'accelération de la propagation du carry. La structure de cet algorithme et les schémas qui lui sont associés sont illustrés en détail au chapitre III, paragraphe 3.

CHAPITRE V

DESCRIPTION DU LOGICIEL DE SIMULATION (MCE)

5.1. Introduction

Il y a quelques années, la conception d'un circuit était essentiellement effectuée à la main, il était alors très important de choisir une méthode de conception très structurée, de manière à réduire les coûts de conception. Aujourd'hui, de très nombreux outils de conception sont disponibles, et on a tendance à croire qu'une bonne méthodologie de conception n'est pas essentielle. En particulier, de nombreux ingénieurs système, sont parfaitement capable de concevoir un circuit intégré (après sa conception logique par des méthodes de synthèse) et utiliser par la suite à des outils modernes permettant de "compiler" de manière automatique le schéma logique en un layout. Il arrive parfois que la méthode de conception ainsi que les outils C.A.O utilisés ne conviennent pas pour un type de circuit intégré, vue ses performances, en particulier la surface et la vitesse, qui peuvent passablement différer en fonction de la méthode de conception utilisée.

5.2. Objectifs d'une méthode de conception et choix de l'outil CAO

Une méthode de conception est définie par les étapes que le concepteur décide de suivre depuis le cahier des charges jusqu'au layout. Les objectifs sont par ordre d'importance.

- La sécurité de conception, soit l'obtention d'un circuit correct au premier tour d'intégration
- La réduction du temps nécessaire à la conception ainsi que la réduction des coûts.

Une densité d'intégration satisfaisante pour le type de circuit à concevoir

Un autre critère important dans la conception d'un circuit intégré est le choix adéquat de l'outil CAO. En effet les performances diffèrent d'un outil à l'autre en particulier en ce qui concerne les librairies des éléments de base et l'architecture avec laquelle ils ont été conçus.

En ce qui concerne notre projet de fin d'étude, notre choix s'est porté sur le logiciel BX de MCE qu'on détaillera ses performances par la suite.

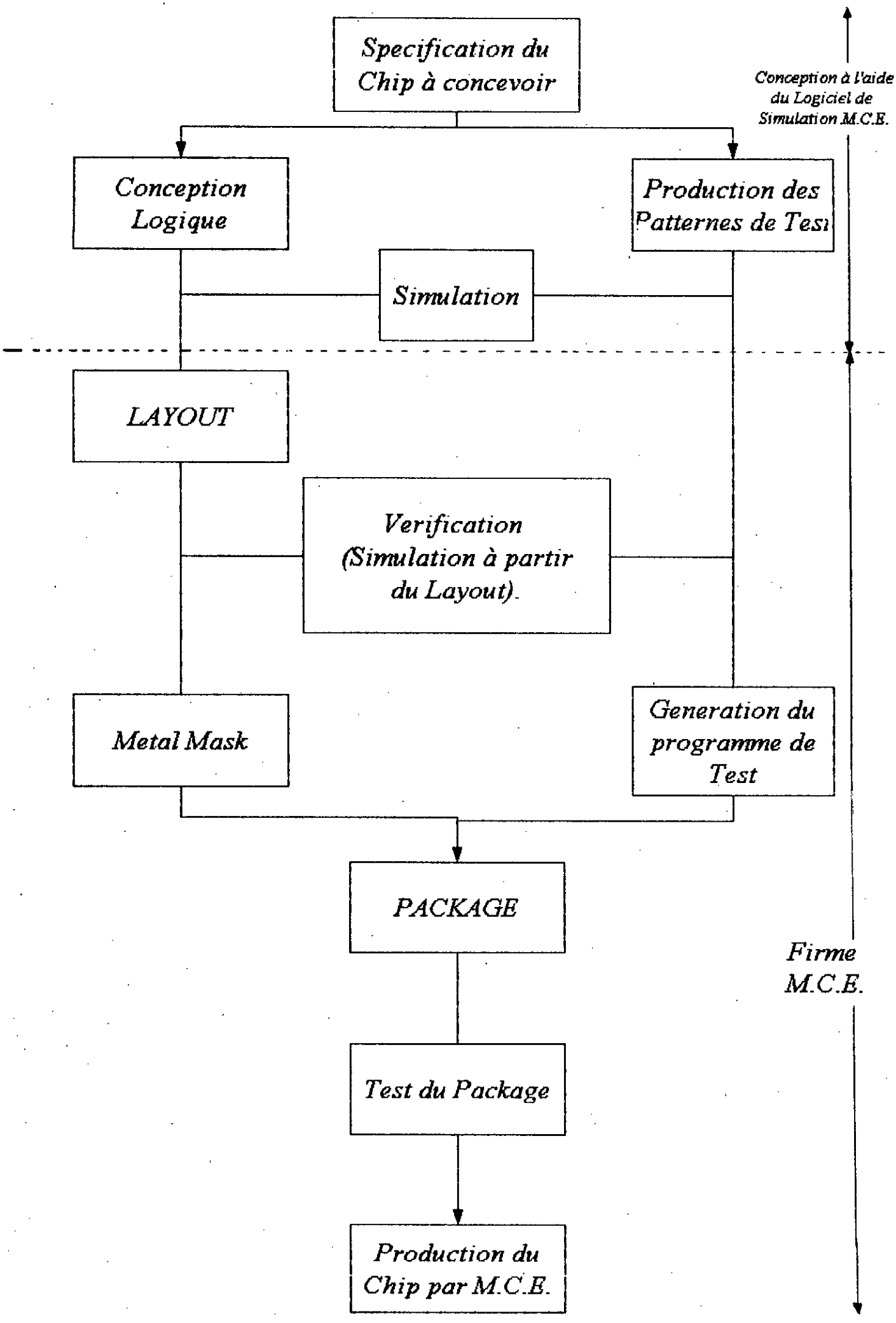
5.3. Simulation par le logiciel de M.C.E

Le logiciel BX de la firme M.C.E (micro circuits engineering), est un puissant outil de conception assistée par ordinateur (C.A.O)

utilisé pour concevoir et tester des circuits pouvant être implémenter sur des circuits intégrés par M.C.E, et de simuler leurs fonctionnements.
C'est donc un précieux outil pour le concepteur d'un système avant d'entreprendre sa réalisation.
Principalement BX est utilisé pour:
- Introduire une description textuelle ou schéma d'un circuit logique
- Construire une description de formes d'ondes permettant à la conception logique d'être complètement validée pendant la phase de simulation. Celui-ci prévoit aussi les moyens de production des vecteurs de tests du circuit après sa fabrication.
- Simuler l'opération du circuit logique.
- Prévoir les donnée pour l'emplacement et le routage automatique du layout.
Le procédé de simulation du circuit logique à l'aide du logiciel BX peut remplacer l'étape du montage expérimental utilisée pour le développement d'un produit. Ceci a plusieurs avantages.
- La conception d'un circuit est construite et validée plus rapidement que la méthode du montage expérimental.
- La simulation sur ordinateur est plus consistante que le prototype à montage expérimental, ce qui permet d'éviter les problèmes, tels les problèmes de fiabilités des composants, pertes de connections, effets de charges...

5.4. Etapes de conception

L'organigramme donné ci-après illustre l'acheminement de l'étape de conception d'un circuit intégré :
1- Spécifications du système
Le point de départ pour une conception est la spécification du concepteur de son système complet.
2- Partitionnement du système
La prochaine étape consiste à partitionner le système en circuit, et autre circuit intégré et composant.
3- Spécification du chip
Le chip est spécifié en définissant ses logiques
4- conception logique
La conception de circuits logiques passe essentiellement par deux étapes à savoir:
- Utilisation d'une méthode de synthèse (synthèse logique, architecturale, intermédiaire et enfin une synthèse utilisant un langage de haut niveau).


Specification du Chip à concevoir
Conception à l'aide du Logiciel de Simulation M.C.E.
Conception Logique
Production des Patternes de Test
Simulation
LAYOUT
Verification (Simulation à partir du Layout)
Metal Mask
Generation du programme de Test
PACKAGE
Test du Package
Production du Chip par M.C.E.
Firme M.C.E.

- Saisie du schéma logique :

Deux méthodes sont utilisées pour la saisie d'un schéma dans le simulateur BX:

- Une primitive de conception logique nécessitant une capture schématique utilisant les cellules standard de la bibliothèque (NAND, NOR, Inverseur etc...) ou une combinaison de primitives de base.
- Création d'une description textuelle du circuit avec un éditeur de texte: c'est un modèle comportemental

5- Production des patternes de test:

La production des patternes de test doit être envisager par le concepteur pour permettre le test du chip pendant le procédé de fabrication. En terme de simulation ces modèles de test vérifient la compatibilité avec la logique.

6- Saisie du schéma ou du texte pour la simulation

Cette étape permet de simuler le circuit conçu et le mettre au point. Cette simulation logique est réalisée en comparant les sorties du circuit aux sorties déjà définies dans la description des formes d'ondes.

7- Conception de layout géométrique

Le procédé de conception du layout géométrique consiste en l'interconnexion du layout des cellules standard du circuit. Cette procédure est similaire au tracé du circuit imprimé et peut être faite soit par le concepteur soit par MCE (Dans la version disponible à l'ENP c'est au MCE de faire la simulation du layout). Enfin la tâche du tracé physique est confiée aux "fondeurs de silicium" et est réalisée par M.C.E.

8- Simulation à partir du layout

Cette seconde simulation tient compte des effets de charge dus aux connexions et elle et réalisée après conception du layout. Ceci assure une bonne marge de conception du produit final.

9- Package

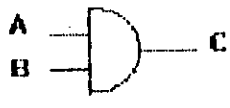
Le produit final devra être présenter sous forme de boite avec des pins constituants les connexions d'entrées sorties, ce qu'on appelle package.

5.5 testabilité et modèles de fautes

5.5.1. Modèles de fautes

Les modèles de fautes, pour les circuits intégrés sont des abstractions de mécanismes qui peuvent causer l'échec de ces dispositifs.

La qualité du test est fortement corrélée avec la "qualité" du modèle de fautes utilisé. Parmi les modèles de fautes les plus prépondérant' au niveau des portes logiques, on cite:
Modèle "Stuck at fault"
D'après Mc Clusky et Clegg, certains modèles "stuck at fault peuvent être diviser en classes d'équivalences. Toutes les fautes appartenant à une classe sont dites de même classe d'équivalence. Soit une porte logique simple (figure 6.1), on note les fautes A stuck at-0, B stuck at-0, C stuck at-0 respectivement par A_0,B_0, C_0 .
Le tableau de Karnaugh indique que les fautes A0,B0 et C0 causent les mêmes perturbations ou changements de la fonction f(C) réalisée à la sortie de la porte, C. Les fautes A0,B0 et C0 sont dites alors équivalentes. D'autres fautes similaires peuvent être équivalentes sans qu'elles n'appartiennent à la même porte, mais ces types d'équivalences sont en général plus difficiles à localiser. Ces équivalences de fautes sont applicables aussi bien aux circuits combinatoires qu'aux circuits séquentiels.



B \ A	0	1
0	C_1 0	B_1 C_1 0
1	A_1 C_1 0	A_0 B_0 C_0 1

figure 6.1

Une idée similaire à "l'équivalence des fautes" mais limité aux circuits combinatoires est la "dominance de fautes". Revenons une deuxième fois à la (figure 6.1), où on note T_{A1},T_{C1} l'ensemble des fautes A1,C1 respectivement. Puisque $T_{A1} \subset T_{C1}$, chaque test qui détecte C1 détecte A1, on dit alors que C1 domine A1.
Ces deux idées d'équivalences de fautes et de dominances de fautes peuvent réduire le nombre des tests du "Vecteur de test" qui doit être générer pour la détection de ces types d'erreurs.

5.5.2. Testabilité des circuits intégrés

Les premiers échantillons prototypes qui suivront le tracé physique, doivent être nécessairement testés et vérifiés pour s'assurer de leurs bons fonctionnements .

Les tests du chip sont réalisés en véhiculant une certaine séquence d'états fixe à travers les pins d'entrée et en même temps vérifier si les états des pins des sorties sont corrects. La séquence d'états d'entrées sorties est définie par les patternes de test. Chaque groupe d'état d'entrées sorties est connu comme un "Vecteur de test" et peut être appliquer à un ou plusieurs "cycles de tests".
Ces patterns de test sont divisés en un nombre de "groupes", chaque groupe consiste en un maximum de 256 "vecteurs de tests". Le vecteur de test est une série d'états, logiques prévus et peuvent être appliquer pour un ou plusieurs cycles de test.
Le cycle de test et un créneau temporel sur lequel l'equipement de test opère et pendant lequel les entrées sont appliquées et les sorties sont récupérées. Chaque entrée du chip doit changer d'état une seule fois pendant le test ou bien recevoir une seule impulsion d'horloge. Chaque sortie du chip doit être échantillonner une seule fois pendant le cycle de test. Une autre séquence de tests standard est prévue par MCE , pour realiser les quatre test suivant:

* test de courant statique fournit :

Le courant statique fournit est mesuré afin de s'assurer qu'il est dans une marge acceptable

* test paramétriques D.C :

Ces tests sont générés par M.C.E en utilisant les définitions des entrées sorties du chip, avec les données caractéristiques des cellules périphériques. Ils vérifient que le chip peut opérer à des niveaux haut minimaux de tensions d'entrée ($V_{IH\_\_}$) et avec des niveaux bas maximaux de tension ($V_{IL\_\_}$) appliquée. Ils s'assurent aussi que le chip fonctionnent à des tensions appliquées minimales ($V_{DD\_\_}$) et maximales ($V_{DD\_\_}$).

* test paramétrique A.C:

Ces tests sont optionnels seulement pour les circuits à grandes vitesses, et vérifient que le circuit peut opérer à ces vitesses bien définies.

CHAPITRE VI

IMPLEMENTATION DU MUTIPLIEUR ET DU DIVISEUR ET LEURS PERFORMANCES

6.1. Introduction

Les étapes par lesquelles doit passer un circuit implemente selon le logiciel M C E sont montrées figure 6.1 qui sont :

1.Introduction du circuit sous forme textuelle ou graphique
2.Introduction des formes d'ondes aussi sous forme graphique ou textuelle pour verifier la validité du circuit durant la simulation et aussi aprés la fabrication .
3.Simulation
4.Fournir des resultats qui seront utilisés par la suite pour élaborer le "layout" ainsi que le routage sur silicium .

I. Introduction du schéma logique du multiplieur et du diviseur
Pour l'implementation du circuit multiplieur ainsi que du diviseur on a suivi l'esprit hierarchique de conception , c'est à dire à partir des portes logiques existant au niveau de la bibliothèque M C E telles que : portes AND ,NAND , Additionneurs etc... .On a crée nos propres modules ou si on peut dire nos macros-cellule .
Ces modules seront ensuites connectés par des fils entre eux et entre les "PADs" ou les pins d'entrées sorties pour former soit le multiplieur soit le diviseur .
Mais il faut se méfier que l'utilisation totale de la surface associée à l'implementation ne dépasse pas 80% pour permettre le passage des connexions lors de la fabrication , par la suite on doit compiler ce circuit pour avoir une base de donné qui sera utilisé par le simulateur du MCE . Signalons içi que le MCE comporte 8 pages "sheets", chacune comporte 64 portions de surface ou "AREAS". Si le circuit occupe plus d'un sheet comme le cas du multiplieur et du diviseur, on doit utiliser des interconnexions ponctuelles, en attribuant le même nom pour chaque interconnexion dans les deux pages.
3- Introduction des formes d'ondes
Les formes d'ondes sont utilisées durant la simulation et sont automatiquement converties pour être utiliser aussi par les équipements de test automatique "ATE" une fois toute la conception finie. De même pour les formes d'ondes deux approches sont possibles: une approche textuelle et une approche graphique.
Cette derniere approche est constituée de:

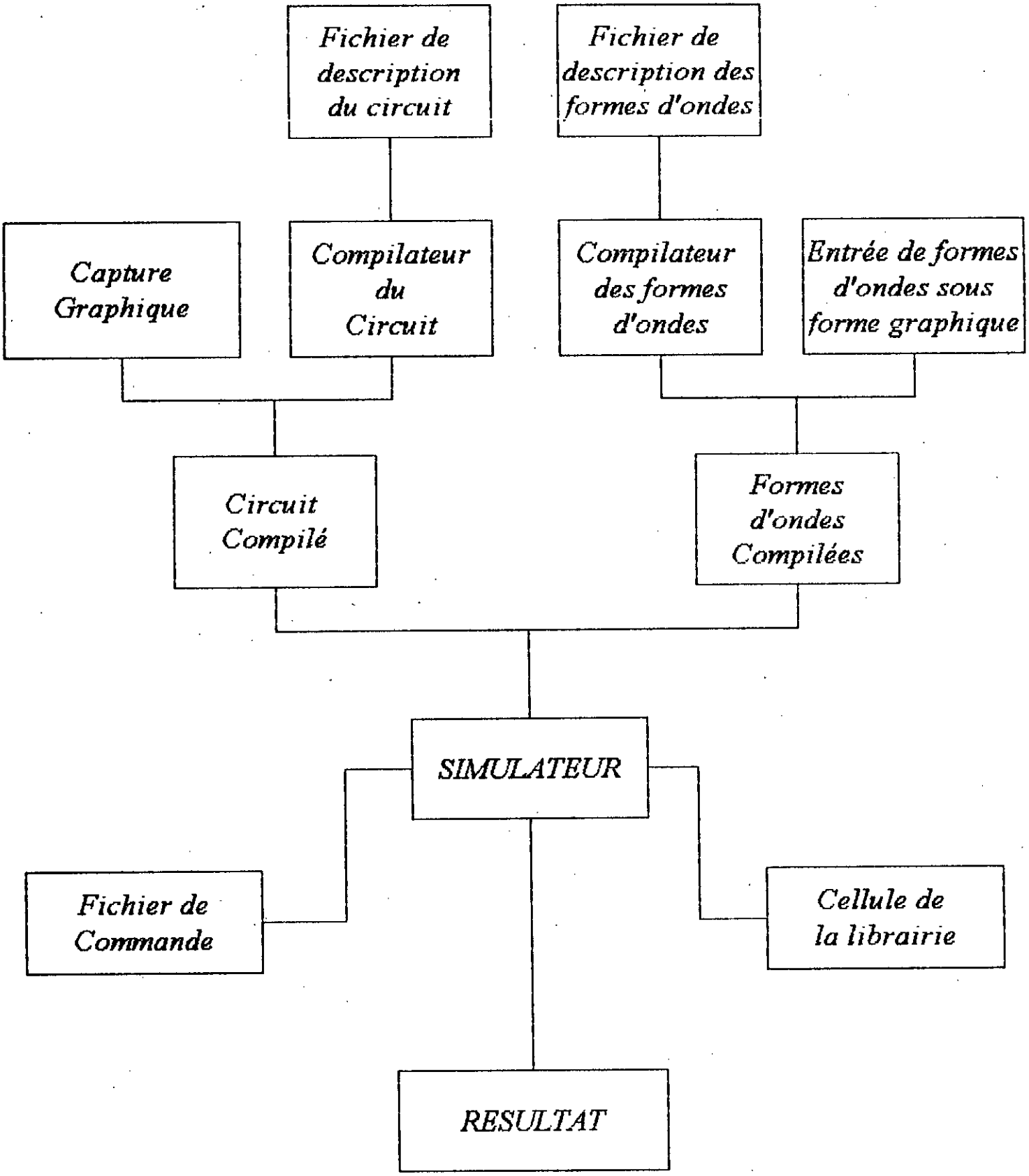


figure 6.1 : *Etapes d'implementation d'un Circuit electronique avec le M.C.E.*

- Mise en forme "Set up form", utilisée pour définir la liste de l'ensemble des temps et celles des ondes.

- La fenêtre d'onde,"Waveform window" utilisée pour voir les ondes déja définies auparavant.

- L'éditeur ou " Edit window" utilisé pour éffectuer des changements sur les états d'une seule forme d'onde.
Chaque fenêtre cité auparavant a son propre texte d'aide "help display text".

3.1. Set up-form
Utilisée pour donner: la durée du test d'un cycle,la liste des générateurs de temps "Timing generator : TG" et la liste des formes d'ondes.

3.2. La periode
La forme d'onde est divisée en un ensemble de groupes, chacun peut être utiliser pour tester une zone ou une fonction particuliere du circuit. Le groupe est lui même formé d'un ensemble de cycles de test de même durée, la longueur de chaque cycle est appelé une période.
Dans cette mise en forme on doit spécifier la nature de chaque onde, le MCE permet sept types d'ondes: Input, Output, Clock, Not-clock, Bi-directional, Bi-directional clock, et Bi-directional Not-clock.

6.1.2 La fenêtre d'onde "Waveform window"
Cette fenêtre affiche toute les formes d'ondes déja définies avec leurs états . Un seul groupe est affiché.

6.1.3. L'éditeur "Edit window"
Cette fenêtre est utilisée pour compléter ou réetablir la mise en forme des états de l'onde séléctionnée. Une fois toutes ces étapes éffectuées avec succés, on doit compiler ces formes d'ondes avec "Waveform compiler" existant au menu principal pour passer à la simulation.

CONCLUSION

BIBLIOGRAPHIE ET ANNEXE

PRESENTATION ET RESULTATS DE LA SIMULATION

Annexes 1

Les courbes de variation

Les caracteristiques de la famille 5 micron.

Les valeurs typiques de cette famille (voir manuel MCE) sont données à 25°C et 5V d'alimentation par contre ces caractristiques varient selon les graphes ci-dessous.

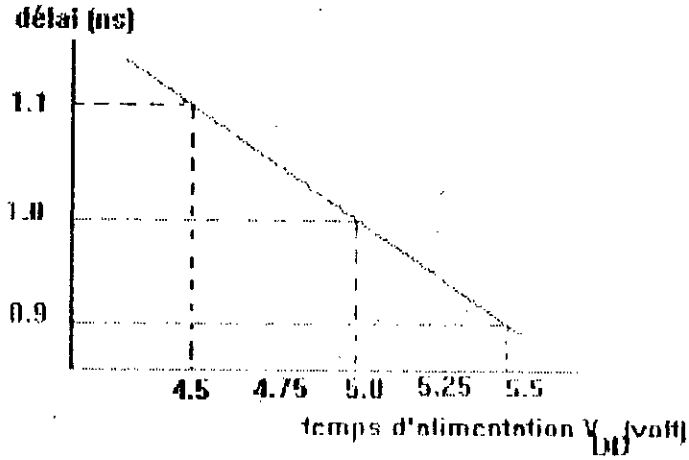


Figure 1: Variation du temps de réponse en fonction de l'alimentation

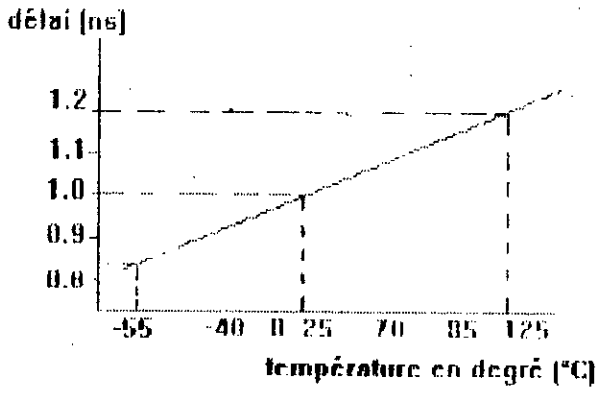


figure 2. Variation du temps de reponse en fonction des variations de la température

ANNEXE 2

I. Calcul du temps de reponse

I.1 definition du temps de reponse d'une cellule

Le temps de réponse de l'entrée vers la sortie pour toutes les cellules de la bibliothèque MCE est donné par :

[illegible]

où :

a) Dout = le delai de l'entrée vers la sortie sans charge

b) Dld = le delai additionnel dû à 1 unité de charge à la sortie en (ns)

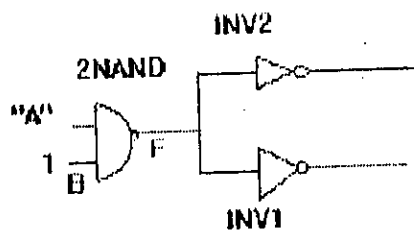
c) n = le nombre total des charges (sortance) à la sortie

d) load = nombre equivalent de charge d'une cellule à l'entrée

(où 1 seul INV2 equivalent à une unité de charge).

I.2 Un exemple de cellul de delai

On considère un cellule 2NAND commande un seul INV2 et unseulINV2 (avec l'entrée B du 2NAND forcée à "1").



En regardant les tableaux des délais pour la cellule 2NAND on trouve que pour la sortie (F) passant du niveau bas au niveau haut (L>H).

Dout = 3 ns et Dld = 1.6 ns

Celle du INV1 et INV2

pour leur charge équivalente toujours donnée par les tableaux

pour INV1 Load = 2; pour INV2 Load =1

Ainsi n = load (1) + load (2) = 1+2 = 3

Le délai total est : Dout = Dout + (n*Dld) = 3+(3*1.6) = 7.8 ns

Le maximum et le minimum de delai dans la gamme (4.5v à 5.5v) et pour une gamme de temperature de (-55°C à + 125°C) varient de cette manière

minimum delai = 0.3 * le delai typique ou nominal
maximum delai = 1.7 * le delai typique

Le maximum et le minimum de delai dans la gamme (4.5v à 5.5v) et pour une gamme de temperature de (-55°C à + 125°C) varient de cette manière

minimum delai = 0.3 * le delai typique ou nominal
maximum delai = 1.7 * le delai typique

Commentaires des resultats

Comme application de ce travail, on a choisi six exemples de multiplications pour des nombres non-signés,signés et mixtes. Les valeurs de ces opérandes pour le cas de la multiplication sont les suivantes:

(-17465)*(-31759)=554670935
(-16220)*(21930)=-355704600
(30260)*(-15711)=-475414860
(32450)*(18925)= 614116250
2487*1500= 3730500
(-1250)*(790)= -987500

Les résultats de cet exemple sont donnés ci-aprés (voir listing N°1)

Trois groupes ont été utilisé pour les six exemples, chacun partagé en cycles, sur chaque cycle on introduit un seul exemple.

Le multiplieur ainsi conçu a les spécifications suivantes:

Tr=882 ns surface= 118.875 mm^2

En ce qui concerne le diviseur deux exemples ont été choisis. Leurs valeurs sont les suivantes:

27280/151=176
(-11858)/121= 98

Le diviseur ainsi conçu a les spécifications suivantes:

Tr= 813 ns surface= 118,875 mm^2

Les résultats sont donnés ci-aprés (voir listing N°2).

Les résultats simulés sur le MCE sont lus verticalement de gauche à droites pour les valeurs d'entrées du multiplieur et du diviseur, et de droite à gauche pour les valeurs de sortie de ces deux circuits.
Ainsi on lit sur la première colonne le temps ou "TIME" et sur la deuxieme colonne on lit les entrées
Exemple:

les valeurs X0,X1,Y0,Y1,P0,P1,ZERO,UN sont écrits d'après le format suivant

```
X X Y Y P P Z U
0 1 0 1 0 1E N
          R
          O
```

```
                *** MCE GATE ARRAY DESIGN SOFTWARE *** (Ref.No:1534)
              (C) Copyright Micro Circuit Engineering Ltd. 1989
                        Customer Ref: ABA         362/5.0

                        BX  Simulator  Ver. 4H11
                        ------------------------
                        Run on 23-APR-94 at 18:02:32

Using   Circuit: MULT              Produced on 13-APR-94 at 15:02:14
Array Family: 70000 series 5 micron single layer
       Library: Ver. 57
and Waveforms: FTMULAA        .   Produced on 23-APR-94 at 18:02:04
Initial Timing Data :-
Period          1000 nano-seconds.
Timing-generator no. 0    Delay       50      Width      600
Timing-generator no. 1    Delay      180      Width      700

   Starting new test pattern - Group number:   1 at time      0
   ------------------------------------------------------------------------


    ** Delay scaling factor of  1.00 used.
        XXXXXXXXXXXXXXXXZUYYYYYYYYYYYYYYYY  PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP
        0123456789111111EN0123456789111111  01234567891111111111222222222233
 TIME:           012345R          012345              012345678901234567890 12
 ----                 O

      0- Cycle       0
@     0: 1001110000100011011111000000111111  XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
@     6: 1001110000100011011111000000111111  1XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
@    49: 1001110000100011011111000000111111  11XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
@    88: 1001110000100011011111000000111111  111XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
@   115: 1001110000100011011111000000111111  1110XXXXXXXXXXXXXXXXXXXXXXXXXXXX
@   120: 1001110000100011011111000000111111  11101XXXXXXXXXXXXXXXXXXXXXXXXXXX
@   201: 1001110000100011011111000000111111  111010XXXXXXXXXXXXXXXXXXXXXXXXXX
@   259: 1001110000100011011111000000111111  1110101XXXXXXXXXXXXXXXXXXXXXXXXX
@   283: 1001110000100011011111000000111111  1110101OXXXXXXXXXXXXXXXXXXXXXXXX
@   289: 1001110000100011011111000000111111  11101010XXXXXXXXXXXXXXXXXXXXXXXX
@   293: 1001110000100011011111000000111111  11101011XXXXXXXXXXXXXXXXXXXXXXXX
@   449: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   519: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   546: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   622: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   649: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   802: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   816: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   818: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   824: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   825: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   826: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   828: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   833: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   834: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   850: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   851: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   856: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   858: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   866: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   868: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
@   872: 1001110000100011011111000000111111  11101010110XXXXXXXXXXXXXXXXXXXXX
```

```
                1000- Cycle       1
@   1050: 0011101011111110101010101011010101   0110101011011001111100001000010000
@   1056: 0011101011111110101010101011010101   0110101011011001111100001000010000
@   1085: 0011101011111110101010101011010101   0010101011011001111100001000010000
@   1110: 0011101011111110101010101011010101   0010101011011001111100001000010000
@   1145: 0011101011111110101010101011010101   0001101011011001111100001000010000
@   1153: 0011101011111110101010101011010101   0001101011011001111100001000010000
@   1175: 0011101011111110101010101011010101   0001100011011001111100001000010000
@   1183: 0011101011111110101010101011010101   0001100011011001111100001000010000
@   1191: 0011101011111110101010101011010101   0001100011011001111100001000010000
@   1198: 0011101011111110101010101011010101   0001100011011001111100001000010000
@   1203: 0011101011111110101010101011010101   0001100011011001111100001000010000
@   1218: 0011101011111110101010101011010101   0001100011011001111100001000010000
@   1227: 0011101011111110101010101011010101   0001100011111001111100001000010000
@   1238: 0011101011111110101010101011010101   0001100011111001111100001000010000
@   1239: 0011101011111110101010101011010101   0001100011111001111100001000010000
@   1247: 0011101011111110101010101011010101   0001100011111001111100001000010000
@   1266: 0011101011111110101010101011010101   0001100011111001111100001000010000
@   1268: 0011101011111110101010101011010101   0001100011111001111000001000010000
@   1286: 0011101011111110101010101011010101   0001100011111001111000001000010000
@   1297: 0011101011111110101010101011010101   0001100011111001111001000100001000
@   1299: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1306: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1327: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1329: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1330: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1345: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1347: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1355: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1368: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1371: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1373: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1377: 0011101011111110101010101011010101   0001100011111001111001100100001000
@   1400: 0011101011111110101010101011010101   0001100011111001111001100101001000
@   1494: 0011101011111110101010101011010101   0001100011111001111001100101001000
@   1612: 0011101011111110101010101011010101   0001100011111001111001100101011000
@   1738: 0011101011111110101010101011010101   0001100011111001111001100101010000
@   1861: 0011101011111110101010101011010101   0001100011111001111001100101010000
@   1866: 0011101011111110101010101011010101   0001100011111001111001100101010000
@   1909: 0011101011111110101010101011010101   0001100011111001111001100101010001
      2000- Cycle       2
@   2050: 0010110001101110011111101010111101   0001100011111001111001100101010001
@   2059: 0010110001101110011111101010111101   0001100011111001111001100101010001
@   2070: 0010110001101110011111101010111101   0001100011111001111001100101010001
@   2087: 0010110001101110011111101010111101   0011100011111001111001100101010001
@   2098: 0010110001101110011111101010111101   0011100011111001111001100101010001
@   2129: 0010110001101110011111101010111101   0011100011111001111001100101010001
@   2166: 0010110001101110011111101010111101   0011000011111001111001100101010001
@   2199: 0010110001101110011111101010111101   0011001011111001111001100101010001
@   2226: 0010110001101110011111101010111101   0011001011111001111001100101010001
@   2262: 0010110001101110011111101010111101   0011001011111001111001100101010001

@   2299: 0010110001101110011111101010111101   0011001010111001111001100101010001
@   2332: 0010110001101110011111101010111101   0011001010011001111001100101010000
@   2366: 0010110001101110011111101010111101   0011001010001001111001100101010001
@   2394: 0010110001101110011111101010111101   0011001010000001111001100101010001
```

```
@   2432: 0010110001101110011111101010111101  0011001010000001110011001010001
@   2462: 0010110001101110011111101010111101  0011001010000001110011001010001
@   2508: 0010110001101110011111101010111101  0011001010000001011001100101010001
@   2556: 0010110001101110011111101010111101  0011001010000001001001100101010001
@   2596: 0010110001101110011111101010111101  0011001010000001001001100101010001
@   2630: 0010110001101110011111101010111101  0011001010000001001101100101010001
@   2685: 0010110001101110011111101010111101  0011001010000001001101100101010001
@   2715: 0010110001101110011111101010111101  0011001010000001001101100101010001
@   2741: 0010110001101110011111101010111101  0011001010000001001101000101010001
@   2784: 0010110001101110011111101010111101  0011001010000001001101010101010001
@   2791: 0010110001101110011111101010111101  0011001010000001001101010101010001
@   2813: 0010110001101110011111101010111101  0011001010000001001101010001010001
@   2825: 0010110001101110011111101010111101  0011001010000001001101010001010001
@   2839: 0010110001101110011111101010111101  0011001010000001001101010001010001
@   2861: 0010110001101110011111101010111101  0011001010000001001101010001110001
@   2871: 0010110001101110011111101010111101  0011001010000001001101010001110001
@   2875: 0010110001101110011111101010111101  0011001010000001001101010001110001
@   2877: 0010110001101110011111101010111101  0011001010000001001101010001110001
@   2880: 0010110001101110011111101010111101  0011001010000001001101010001110001
@   2882: 0010110001101110011111101010111101  0011001010000001001101010001110001
    3000- Cycle       3
@   3050: 0100001101111111001101101111110010010  0011001010000001001101010001110001
@   3083: 0100001101111111001101101111110010010  0111001010000001001101010001110001
@   3087: 0100001101111111001101101111110010010  0101001010000001001101010001110001
@   3098: 0100001101111111001101101111110010010  0101001010000001001101010001110001
@   3124: 0100001101111111001101101111110010010  0101101010000001001101010001110001
@   3130: 0100001101111111001101101111110010010  0101101010000001001101010001110001
@   3138: 0100001101111111001101101111110010010  0101100010000001001101010001110001
@   3139: 0100001101111111001101101111110010010  0101100110000001001101010001110001
@   3141: 0100001101111111001101101111110010010  0101100110000001001101010001110001
@   3144: 0100001101111111001101101111110010010  0101100111000001001101010001110001
@   3152: 0100001101111111001101101111110010010  0101100111000001001101010001110001
@   3170: 0100001101111111001101101111110010010  0101100111010010011010100011110001
@   3221: 0100001101111111001101101111110010010  0101100111010010011010100011110001
@   3277: 0100001101111111001101101111110010010  0101100111010110011010100011110001
@   3336: 0100001101111111001101101111110010010  0101100111010100011010100011110001
@   3362: 0100001101111111001101101111110010010  0101100111010101011010100011110001
@   3394: 0100001101111111001101101111110010010  0101100111010101011010100011110001
@   3426: 0100001101111111001101101111110010010  0101100111010101011010100011110001
@   3477: 0100001101111111001101101111110010010  0101100111010101010010100011110001
@   3507: 0100001101111111001101101111110010010  0101100111010101011010100011110001
@   3534: 0100001101111111001101101111110010010  0101100111010101011010100011110001
@   3562: 0100001101111111001101101111110010010  0101100111010101011010100011110001
@   3606: 0100001101111111001101101111110010010  0101100111010101011000001110001
@   3627: 0100001101111111001101101111110010010  0101100111010101011001001110001
@   3671: 0100001101111111001101101111110010010  0101100111010101011001001110001
@   3701: 0100001101111111001101101111110010010  0101100111010101011001001110001
@   3733: 0100001101111111001101101111110010010  0101100111010101011001001110001
@   3765: 0100001101111111001101101111110010010  0101100111010101011001001010001
@   3808: 0100001101111111001101101111110010010  0101100111010101011001001000001
@   3837: 0100001101111111001101101111110010010  0101100111010101011001001001001
@   3847: 0100001101111111001101101111110010010  0101100111010101011001001001001
@   3851: 0100001101111111001101101111110010010  0101100111010101011001001001001
@   3853: 0100001101111111001101101111110010010  0101100111010101011001001001000
Sarting new test pattern - Group number:   2 at time          4000
-------------------------------------------------------------------
    4000- Cycle       0
@   4050: 0000000000000000000000000000000000  0101100111010101011001001001000
@   4076: 0000000000000000000000000000000000  0101100111010101011001001001000
@   4087: 0000000000000000000000000000000000  0001100111010101011001001001000
```

```
@   4109: 00000000000000000000000000000000  00011001110101010101100100100100
@   4113: 00000000000000000000000000000000  00001001110101010101100100100100
@   4166: 00000000000000000000000000000000  00000001110101010101100100100100
@   4192: 00000000000000000000000000000000  00000001110101010101100100100100
@   4279: 00000000000000000000000000000000  00000001110101010101100100100100
@   4309: 00000000000000000000000000000000  00000001110101010101100100100100
@   4352: 00000000000000000000000000000000  00000001110101010101100100100100
@   4378: 00000000000000000000000000000000  00000000110101010101100100100100
@   4407: 00000000000000000000000000000000  00000000010101010101100100100100
@   4438: 00000000000000000000000000000000  00000000000101010101100100100100
@   4455: 00000000000000000000000000000000  00000000000101010101100100100100
@   4488: 00000000000000000000000000000000  00000000000001010101100100100100
@   4581: 00000000000000000000000000000000  00000000000001010101100100100100
@   4582: 00000000000000000000000000000000  00000000000000010101100100100100
@   4611: 00000000000000000000000000000000  00000000000000010101100100100100
@   4613: 00000000000000000000000000000000  00000000000000000101100100100100
@   4629: 00000000000000000000000000000000  00000000000000000101100100100100
@   4640: 00000000000000000000000000000000  00000000000000000011001001001000
@   4645: 00000000000000000000000000000000  00000000000000000011001001001000
@   4661: 00000000000000000000000000000000  00000000000000000010010010010000
@   4663: 00000000000000000000000000000000  00000000000000000000010010010000
@   4665: 00000000000000000000000000000000  00000000000000000000010010010000
@   4681: 00000000000000000000000000000000  00000000000000000000010010010000
@   4697: 00000000000000000000000000000000  00000000000000000000000010010000
@   4759: 00000000000000000000000000000000  00000000000000000000000010010000
@   4775: 00000000000000000000000000000000  00000000000000000000000010010000
@   4804: 00000000000000000000000000000000  00000000000000000000000000010000
@   4850: 00000000000000000000000000000000  00000000000000000000000000010000
@   4862: 00000000000000000000000000000000  00000000000000000000000000010000
@   4865: 00000000000000000000000000000000  00000000000000000000000000000000
@   4893: 00000000000000000000000000000000  00000000000000000000000000000000
@   4916: 00000000000000000000000000000000  00000000000000000000000000000010
@   4945: 00000000000000000000000000000000  00000000000000000000000000000010
    8000- Cycle       4
@   8050: 11101101100100000100111011101000000  00000000000000000000000000000010
@   8060: 11101101100100000100111011101000000  00000000000000000000000000000010
@   8101: 11101101100100000100111011101000000  00000000000000000000000000000010
@   8115: 11101101100100000100111011101000000  00000000000000000000000000000010
@   8130: 11101101100100000100111011101000000  00100000000000000000000000000010
@   8145: 11101101100100000100111011101000000  00100000000000000000000000000010
@   8188: 11101101100100000100111011101000000  00100000000000000000000000000010
@   8215: 11101101100100000100111011101000000  00100000000000000000000000000010
@   8239: 11101101100100000100111011101000000  00100010000000000000000000000010
@   8241: 11101101100100000100111011101000000  00100010000000000000000000000010
@   8261: 11101101100100000100111011101000000  00100010000000000000000000000010
@   8266: 11101101100100000100111011101000000  00100010000000000000000000000010
@   8269: 11101101100100000100111011101000000  00100010000000000000000000000010
@   8285: 11101101100100000100111011101000000  00100010000000000000000000000010
@   8298: 11101101100100000100111011101000000  00100010000000000000000000000010
@   8309: 11101101100100000100111011101000000  00100010001000000000000000000010
@   8328: 11101101100100000100111011101000000  00100010001100000000000000000010
@   8329: 11101101100100000100111011101000000  00100010001100000000000000000010
@   8338: 11101101100100000100111011101000000  00100010001100000000000000000010
@   8344: 11101101100100000100111011101000000  00100010001100000000000000000010
@   8355: 11101101100100000100111011101000000  00100010001101000000000000000010
@   8358: 11101101100100000100111011101000000  00100010001101000000000000000010
@   8379: 11101101100100000100111011101000000  00100010001101100000000000000010
@   8393: 11101101100100000100111011101000000  00100010001101110000000000000010
@   8413: 11101101100100000100111011101000000  00100010001101110000000000000010
@   8428: 11101101100100000100111011101000000  00100010001101110000000000000010
```

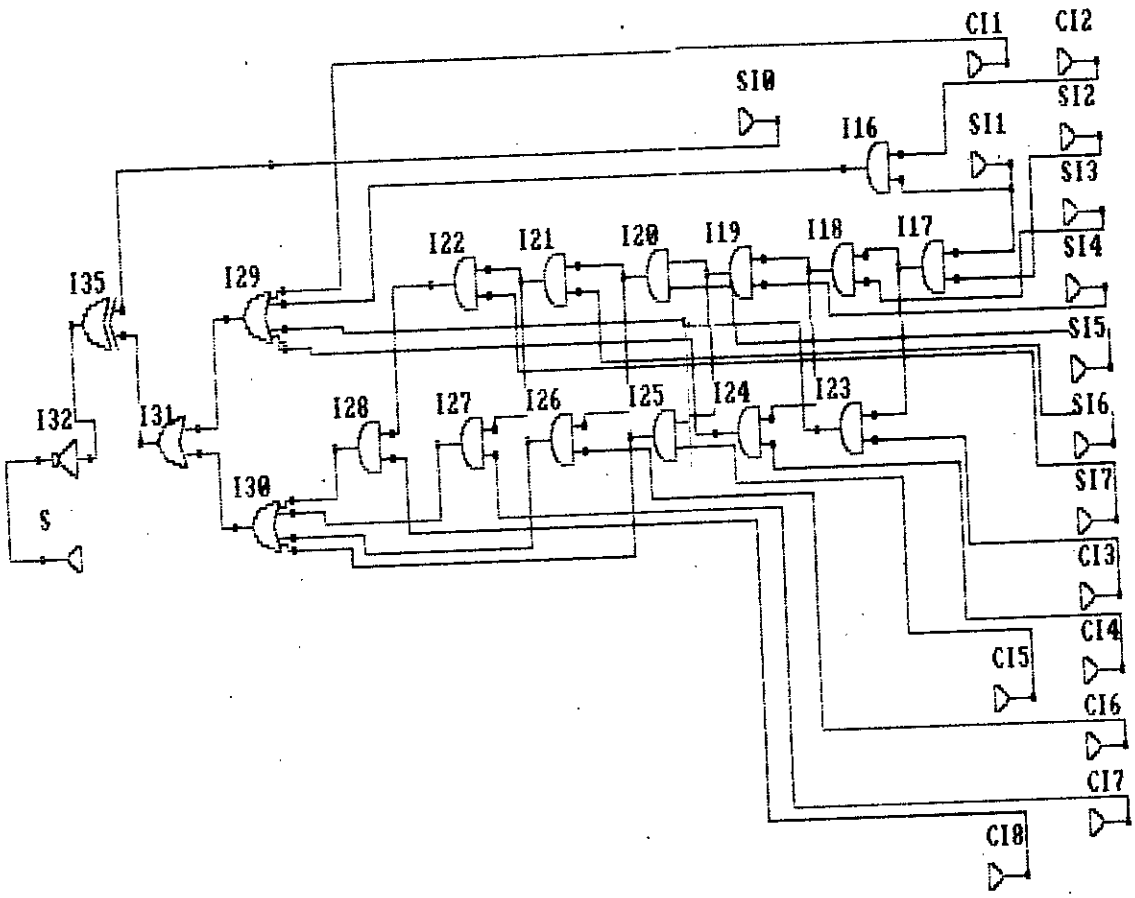
```
@   8434: 1110110110010000010011101110100000  0010001000110111000000000000000010
@   8438: 1110110110010000010011101110100000  0010001000110111000000000000000010
@   8442: 1110110110010000010011101110100000  0010001000110111000000000000000010
@   8450: 1110110110010000010011101110100000  0010001000110111000100000000000010
@   8463: 1110110110010000010011101110100000  0010001000110111000100000000000010
@   8473: 1110110110010000010011101110100000  0010001000110111000110000000000010
@   8482: 1110110110010000010011101110100000  0010001000110111000110000000000010
@   8548: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8554: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8586: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8588: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8595: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8602: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8639: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8645: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8663: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8678: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8686: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8715: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8744: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8773: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8802: 1110110110010000010011101110100000  0010001000110111000111000000000010
@   8845: 1110110110010000010011101110100000  0010001000110111000111000000000000
Starting new test pattern - Group number:   3 at time           9000
-------------------------------------------------------------------
    9000- Cycle       0
@   9050: 0000000000000000000000000000000000  0010001000110111000111000000000000
@   9076: 0000000000000000000000000000000000  0010001000110111000111000000000000
@   9085: 0000000000000000000000000000000000  0000001000110111000111000000000000
@   9087: 0000000000000000000000000000000000  0000001000110111000111000000000000
@   9113: 0000000000000000000000000000000000  0000001000110111000111000000000000
@   9139: 0000000000000000000000000000000000  0000001000110111000111000000000000
@   9167: 0000000000000000000000000000000000  0000000000110111000111000000000000
@   9216: 0000000000000000000000000000000000  0000000000110111000111000000000000
@   9253: 0000000000000000000000000000000000  0000000000110111000111000000000000
@   9319: 0000000000000000000000000000000000  0000000000110111000111000000000000
@   9324: 0000000000000000000000000000000000  0000000000010111000111000000000000
@   9335: 0000000000000000000000000000000000  0000000000000111000111000000000000
@   9366: 0000000000000000000000000000000000  0000000000000111000111000000000000
@   9382: 0000000000000000000000000000000000  0000000000000011000111000000000000
@   9398: 0000000000000000000000000000000000  0000000000000001000111000000000000
@   9414: 0000000000000000000000000000000000  0000000000000000000111000000000000
@   9430: 0000000000000000000000000000000000  0000000000000000000111000000000000
@   9432: 0000000000000000000000000000000000  0000000000000000000111000000000000
@   9464: 0000000000000000000000000000000000  0000000000000000000111000000000000
@   9582: 0000000000000000000000000000000000  0000000000000000000011000000000000
@   9611: 0000000000000000000000000000000000  0000000000000000000010000000000000
@   9640: 0000000000000000000000000000000000  0000000000000000000000000000000000
@   9665: 0000000000000000000000000000000000  0000000000000000000000000000000000
@   9683: 0000000000000000000000000000000000  0000000000000000000000000000000000
@   9746: 0000000000000000000000000000000000  0000000000000000000000000000000000
@   9759: 0000000000000000000000000000000000  0000000000000000000000000000000000
@   9775: 0000000000000000000000000000000000  0000000000000000000000000000000000
@   9784: 0000000000000000000000000000000000  0000000000000000000000000000000000
@   9799: 0000000000000000000000000000000000  0000000000000000000000000000000000
@   9838: 0000000000000000000000000000000000  0000000000000000000000000000000000
@   9864: 0000000000000000000000000000000000  0000000000000000000000000000000010
@   9871: 0000000000000000000000000000000000  0000000000000000000000000000000010
   14000- Cycle       5
@  14050: 0100011100100001010110100011000000  0000000000000000000000000000000010
```

```
@  14060: 01000111001000010101101000110000000  00000000000000000000000000000010
@  14087: 01000111001000010101101000110000000  00000000000000000000000000000010
@  14107: 01000111001000010101101000110000000  00000000000000000000000000000010
@  14131: 01000111001000010101101000110000000  00100000000000000000000000000010
@  14140: 01000111001000010101101000110000000  00100000000000000000000000000010
@  14178: 01000111001000010101101000110000000  00110000000000000000000000000010
@  14183: 01000111001000010101101000110000000  00110000000000000000000000000010
@  14203: 01000111001000010101101000110000000  00110000000000000000000000000010
@  14219: 01000111001000010101101000110000000  00110000000000000000000000000010
@  14225: 01000111001000010101101000110000000  00110000000000000000000000000010
@  14240: 01000111001000010101101000110000000  00110100000000000000000000000010
@  14243: 01000111001000010101101000110000000  00110100000000000000000000000010
@  14282: 01000111001000010101101000110000000  00110110000000000000000000000010
@  14283: 01000111001000010101101000110000000  00110110000000000000000000000010
@  14295: 01000111001000010101101000110000000  00110110000000000000000000000010
@  14300: 01000111001000010101101000110000000  00110110000000000000000000000010
@  14302: 01000111001000010101101000110000000  00110110000000000000000000000010
@  14311: 01000111001000010101101000110000000  00110110000000000000000000000010
@  14337: 01000111001000010101101000110000000  00110111010000000000000000000010
@  14342: 01000111001000010101101000110000000  00110111010000000000000000000010
@  14345: 01000111001000010101101000110000000  00110111010000000000000000000010
@  14346: 01000111001000010101101000110000000  00110111010000000000000000000010
@  14378: 01000111001000010101101000110000000  00110111010000000000000000000010
@  14385: 01000111001000010101101000110000000  00110111010000000000000000000010
@  14387: 01000111001000010101101000110000000  00110111010000000000000000000010
@  14398: 01000111001000010101101000110000000  00110111010000000000000000000010
@  14413: 01000111001000010101101000110000000  00110111010000000000000000000010
@  14418: 01000111001000010101101000110000000  00110111010000000000000000000010
@  14427: 01000111001000010101101000110000000  00110111010001000000000000000010
@  14437: 01000111001000010101101000110000000  00110111010001000000000000000010
@  14438: 01000111001000010101101000110000000  00110111010001000000000000000010
@  14445: 01000111001000010101101000110000000  00110111010001000000000000000010
@  14452: 01000111001000010101101000110000000  00110111010001000000000000000010
@  14461: 01000111001000010101101000110000000  00110111010001000000000000000010
@  14498: 01000111001000010101101000110000000  00110111010001000100000000000010
@  14522: 01000111001000010101101000110000000  00110111010001000100000000000010
@  14529: 01000111001000010101101000110000000  00110111010001000100000000000010
@  14532: 01000111001000010101101000110000000  00110111010001000110000000000010
@  14543: 01000111001000010101101000110000000  00110111010001000110000000000010
@  14559: 01000111001000010101101000110000000  00110111010001000111000000000010
@  14575: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14591: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14597: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14624: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14627: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14634: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14640: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14645: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14651: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14654: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14656: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14667: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14673: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14677: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14689: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14691: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14714: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14734: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14760: 01000111001000010101101000110000000  00110111010001000111100000000010
@  14763: 01000111001000010101101000110000000  00110111010001000111100000000010
```

```
@  14787: 0100011100100001010110100011000000  0011011010001000111100000000000010
@  14789: 0100011100100001010110100011000000  0011011010001000111100000000000010
@  14792: 0100011100100001010110100011000000  0011011010001000111100000000000010
@  14803: 0100011100100001010110100011000000  0011011010001000111100000000000010
@  14810: 0100011100100001010110100011000000  0011011010001000111100000000000010
@  14818: 0100011100100001010110100011000000  0011011010001000111100000000000010
@  14835: 0100011100100001010110100011000000  0011011010001000111100000000000010
@  14839: 0100011100100001010110100011000000  0011011010001000111100000000000010
@  14863: 0100011100100001010110100011000000  0011011010001000111100000000000000
@  14868: 0100011100100001010110100011000000  0011011010001000111100000000000000
@  14911: 0100011100100001010110100011000000  0011011010001000111100000000000001
END OF PATTERNS

*** Check mode - no differences found ***
```

```
          *** MCE GATE ARRAY DESIGN SOFTWARE *** (Ref.No:1534)
    (C) Copyright Micro Circuit Engineering Ltd. 1989
          Customer Ref: ABA          362/5.0

          BX  Simulator  Ver. 4H11
          ------------------------

          Run on 28-APR-94 at 11:52:16

Using   Circuit: DIVIS                    Produced on 27-APR-94 at 15:10:39
Array Family: 70000 series 5 micron single layer
     Library: Ver. 57
and Waveforms: FTDIVAA                   Produced on 28-APR-94 at 11:51:58

Initial Timing Data :-
Period          1000 nano-seconds.
Timing-generator no. 0    Delay       100       Width       200
Timing-generator no. 1    Delay       600       Width       100

Starting new test pattern - Group number:    1 at time                0
-------------------------------------------------------------------------
** Delay scaling factor of  1.00 used.
          NNNNNNNNNNNNNNNNCKDDDDDDDDD  SSSSSSSSCCCCCCCCDDDDDDDDDS
          012345678911111100012345678  012345670000000000000000OI
     TIME:          012345                        012345670123456 78J
     ----                                                        8

        0- Cycle       0
@       0: 000010010101011001110110010  XXXXXXXXXXXXXXXXXXXXXXXXX
@      26: 000010010101011001110110010  0XXXXXXX0XXXXXXX0XXXXXXXX
@      55: 000010010101011001110110010  00XXXXXX01XXXXXX00XXXXXXX
@     107: 000010010101011001110110010  00XXXXXX01XXXXXX00XXXXXXX
@     197: 000010010101011001110110010  000XXXXX011XXXXX001XXXXXX
@     340: 000010010101011001110110010  000XXXXX011XXXXX001XXXXXX
@     367: 000010010101011001110110010  0000XXXX0110XXXX0010XXXXX
@     426: 000010010101011001110110010  0000XXXX0110XXXX0010XXXXX
@     445: 000010010101011001110110010  0000XXXX0110XXXX0010XXXXX
@     505: 000010010101011001110110010  00001XXX01101XXX00100XXXX
@     507: 000010010101011001110110010  00001XXX01101XXX00100XXXX
@     695: 000010010101011001110110010  000011XX011010XX0010010XX
@     740: 000010010101011001110110010  000011XX011010XX0010010XX
@     778: 000010010101011001110110010  0000110X0110100X0010010XX
@     813: 000010010101011001110110010  00001101011010000010010000
      1000- Cycle       1
@     1011: 000010010101011001110110010  00001101011010000010010000
@     1056: 000010010101011001110110010  00001101111010001010010000
@     1100: 010010100111010101100111100  00001101111010001010010000
@     1155: 010010100111010101100111100  00001101111010001010010000
@     1158: 010010100111010101100111100  00001101111010001010010000
@     1168: 010010100111010101100111100  00001101111010001010010000
@     1199: 010010100111010101100111100  00001101111010001010010000
@     1212: 010010100111010101100111100  00001101111010001010010000
@     1214: 010010100111010101100111100  01001101101010001010010000
@     1223: 010010100111010101100111100  01001101101010001010010000
@     1228: 010010100111010101100111100  01001101101010001010010000
@     1240: 010010100111010101100111100  01001101101010001010010000
```

```
@       1241: 0100101001110101011001111000  0100110110101000101001000 0
@       1247: 0100101001110101011001111000  0100110110101000101001000 0
@       1248: 0100101001110101011001111000  0100110110101000101001000 0
@       1251: 0100101001110101011001111000  0100110110001000100001000 0
@       1253: 0100101001110101011001111000  0100110110001000100001000 0
@       1254: 0100101001110101011001111000  0100110110001000100001000 0
@       1267: 0100101001110101011001111000  0100110110001000100001000 0
@       1276: 0100101001110101011001111000  0100110110001000100001000 0
@       1280: 0100101001110101011001111000  0100110110001000100001000 0
@       1282: 0100101001110101011001111000  0100110110001000100001000 0
@       1292: 0100101001110101011001111000  0100110110001000100001000 0
@       1293: 0100101001110101011001111000  0100110110011000100001000 0
@       1300: 0100101001110101011001111000  0100110110011000100001000 0
@       1309: 0100101001110101011001111000  0100110110011000100001000 0
@       1321: 0100101001110101011001111000  0100110110011000100001000 0
@       1329: 0100101001110101011001111000  0100110110011000100001000 0
@       1352: 0100101001110101011001111000  0100110110011000100001000 0
@       1369: 0100101001110101011001111000  0100110110011000100001000 0
@       1370: 0100101001110101011001111000  0100110110010000100011000 0
@       1379: 0100101001110101011001111000  0100010110010000100011000 0
@       1390: 0100101001110101011001111000  0100010110010000100011000 0
@       1395: 0100101001110101011001111000  0100010110010000100011000 0
@       1397: 0100101001110101011001111000  0100010110010000100011000 0
@       1401: 0100101001110101011001111000  0100010110010000100011000 0
@       1406: 0100101001110101011001111000  0100010110010000100011000 0
@       1422: 0100101001110101011001111000  0100010110010000100011000 0
@       1428: 0100101001110101011001111000  0100010110010100100011000 0
@       1438: 0100101001110101011001111000  0100010110010100100011000 0
@       1440: 0100101001110101011001111000  0100010110010100100011000 0
@       1451: 0100101001110101011001111000  0100010110010100100011000 0
@       1465: 0100101001110101011001111000  0100010110010100100011000 0
@       1487: 0100101001110101011001111000  0100010110010100100011000 0
@       1526: 0100101001110101011001111000  0100010110010100100011000 0
@       1530: 0100101001110101011001111000  0100010110010100100011000 0
@       1534: 0100101001110101011001111000  0100010110010100100011000 0
@       1579: 0100101001110101011001111000  0100011110010100100011000 0
@       1615: 0100101001110101011001111000  0100011110010100100011000 0
@       1633: 0100101001110101011001111000  0100011110010100100011000 0
@       1660: 0100101001110101011001111000  0100011110010100100011000 0
@       1672: 0100101001110101011001111000  0100011110010100100011000 0
@       1678: 0100101001110101011001111000  0100011110010100100011000 0
@       1720: 0100101001110101011001111000  0100011110010101100011011 1
@       1722: 0100101001110101011001111000  0100011010010101100011011 1
@       1738: 0100101001110101011001111000  0100011010010101100011011 1
@       1765: 0100101001110101011001111000  0100011010010101100011011 1
@       1771: 0100101001110101011001111000  0100011010010101100011011 1
@       1780: 0100101001110101011001111000  0100011010010101100011011 1
@       1817: 0100101001110101011001111000  0100011010010101100011011 1
END OF PATTERNS

 *** Check mode - no differences found ***
```


CI1
CI2
SI0
SI2
I16
SI1
SI3
I22
I21
I20
I19
I18
I17
SI4
I35
I29
SI5
I28
I27
I26
I25
I24
I23
SI6
I32
I31
SI7
I30
S
CI3
CI4
CI5
CI6
CI7
CI8

BIBLIOGRAPHIE

[1]: C.Mead et L.Conway, "Introduction aux systemes VLSI", Intereditions,1983.

[2]: S.D.Pezaris, "A 40 ns 17x17 Bits array multiplier" IEEE Trans.Computer.,Vol C-21, pp 442-447, April1971.

[3]: N.Takagi, H.Yas and S.Yajima, "High-speed VLSI multiplication with a redundant binary addition tree" IEEE Trans.Computer.,Vol C-34,n°9,pp 789-796, 1985

[4]: K.Hwang, "Computer arithmetic:principles, architcture and design", John Wiley and Sons, 1979

[5]: D.A.Henlin, M.T.Frestsch, M.Mazin and E.T.Leuwis, "A 16 x 16 Bits pipelined multiplier macrocell".

[6]: R.de Mori and A.Serre, "A parallel structure for signed number multiplication and addition", IEEE Trans.Computer.,Vol C-21, pp 1453-1454, Dec.1972

[7]: J.C.Madjithia and R.Kitai, "An iterative array for multiplication of signed binary numbers", IEEE Trans.Computer.,Vol C-20,pp 214-216, Feb.1971

[8]: N.Badeira, K.Vaccaro and J.A.Howard, "A two's complement array multiplier using true values of operands", IEEE Trans.Computer.,Vol C-33, n°8 August 1983

[9]: Y.Harata et al, "High-speed LSI multiplier using a redundant binary adder tree" Proc-IEEE ICC D 84, pp 165-170, oct.1984.

[10]: C.R.Baugh and B.A.Wooley, "A two's complement parallel array multiplier algorithm", IEEE Trans.Computer.,Vol C-22, pp 1045-1047, Dec.1973

[11]: R.F.Lyon "Two's complement pipeline multipliers", IEEE Trans. on communication, April 1976

[12]: "Premire revue internaionale Algerienne des technologies avancées", vol 1, n°1,Juin 1991

[13]: A.R.Hurson and B.Shirazi, "A systolic multiplier unit and its VLSI design"Proc. of the 12th International conference computer architecture, June 1985

[14]: A.R.Hurson ,B.Shirazi and S.H.Pakzad, "Layout Design of a Systolic Multiplier Unit for 2's complement numbers", VLSI Systems Design, pp 78-84, Feb.1986

[16]: L.R.Rubinfield, "Aproof of the modified Booth's algorithms for multiplication" IEEE Trans.Computer, pp 1014-1017, octobre 1975

[17]: C.S.Wallace, "A suggestion for a fast multiplier", IEEE Trans. on Electronic Computers, pp 14-17, Feb.1964

[18]: L.Dadda, "Some schemes for parallel multiplier", Alta frequenza, Vol 34, pp349-356, Vol 34,, 1965

[19]: H.H.Guild, "Some cellular logic arrays for nonrestauring binary division",The Radio and Elec. Engr., Vol. 39, pp 345-348 June 1970.

[20]: M.Cappa and V.C.Hammacher, "An augmented iterative array for highspeed binary division", IEEE Trans.Computer.,Vol C-22 pp 172-175 Feb.1973