N° d'ordre: 1018

THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le grade de

DOCTEUR TROISIEME CYCLE

Spécialité : Electronique

par

EL miloud AMAMOU

maitre ès sciences

CONTRIBUTION A L'ETUDE DE FAISABILITE D'UN SYSTEME A OPERATEURS NUMERIQUES

Application a la simulation hybride et au traitement paralléle

AVANT PROPOS

Le travail présenté dans ce mémoire a été effectué au Laboratoire d'Informatique Industrielle de l'INSTITUT INDUSTRIEL DU NORD et au Laboratoire de Systématique de l'UNIVERSITE DES SCIENCES ET TECHNIQUES de LILLE I.

Je tiens tout d'abord, à exprimer à Monsieur le Professeur F. LAURENT, Directeur du Laboratoire de Systématique, combien je suis honoré qu'il ait accepté de présider mon Jury de thèse. Qu'il trouve ici, le témoignage de ma profonde reconnaissance pour m'avoir accueilli et intégré au sein de son équipe.

Que Messieurs le Professeur J.C. GENTINA, Directeur de l'Institut Industriel du Nord, et D. CORBEEL, Maître-Assistant à l'Institut Industriel du Nord, trouvent ici l'expression de toute ma reconnaissance, pour leur aide constante, leurs idées, leurs conseils et leurs encouragements qui m'ont toujours été précieux, tant au cours de mes recherches que dans la rédaction de cette thèse.

Je tiens à adresser mes vifs remerciements à Monsieur le Professeur P. BORNE de l'Institut Industriel du Nord, pour l'honneur qu'il me fait en acceptant de participer à mon Jury de thèse.

Qu'il me soit permis de rendre hommage à tous les techniciens et chercheurs du Laboratoire d'Informatique Industrielle de l'I. D. N. et du Laboratoire de Systématique de Lille I, pour l'aide précieuse qu'ils m'ont apportée, tant sur le plan scientifique que sur le plan humain.

Que Madame TRICOT qui a eu la lourde charge de dactylographier ce document, Madame DELTOUR et Monsieur SOYEZ qui en ont assuré, avec gentillesse, la réalisation matérielle, soient remerciés et assurés de toute ma gratitude.

SOMMAIRE

CHAPITRE I

page

CHAPITRE II

CHAPITRE III

INTRODUCTION GENERALE

La simulation peut être considérée comme un outil de base permettant la résolution des systèmes complexes qui ne peuvent être complètement appréhendés sur le plan théorique. Toutefois, il convient de souligner qu'une simulation n'est possible que dans l'hypothèse où le système peut être décrit par un modèle représentatif et s'il existe des moyens techniques capables de le manipuler.

Généralement, un système est décrit à l'aide d'un modèle à caractère mathématique, en vue de sa manipulation par simulation électronique, dont deux classes sont à distinguer :

- Le calculateur analogique :

Sa puissance réside dans le fait qu'il traite les informations en continu et de façon parallèle. Le temps de résolution d'un problème est indépendant du nombre d'opérateurs mis en jeu. Cependant, la précision n'est pas grande (on atteint 0,01 % pour des calculateurs très perfectionnés) et présente une contrainte pour la modélisation des grands systèmes où le nombre d'amplificateurs opérationnels à interconnecter se trouve limité. Les opérations spatiales et la mémorisation ne peuvent se faire sans aménagement coûteux.

- Le calculateur numérique :

Il possède la capacité de manipuler un très grand nombre de données avec une précision sans limite théorique. La possibilité de résolution en virgule flottante permet d'éviter le problème de changement d'échelle. La simulation spatiale se fait sans problème. Cependant, les intégrations et dérivations se font par approximation théorique, et les informations sont traitées selon un modèle séquentiel, ce qui alourdit généralement la résolution des modèles continus.

Avec des possibilités ainsi complémentaires, le couplage des deux calculateurs paraît naturel. L'ensemble ainsi obtenu est appelé calculateur hybride de type II.

Dans un tel ensemble, la partie analogique justifie son intérêt par la résolution parallèle en temps réel, accéléré ou ralenti, des équations différentielles.

Toutefois, cette partie risque d'être difficilement utilisable pour certains problèmes (cas des grands systèmes, grande précision, le temps n'étant pas la seule variable indépendante).

Le recours à un calculateur purement numérique, mais possédant cette fois-ci, la faculté temporelle de l'analogique, devient la seule alternative.

La recherche d'implémenter de tels calculateurs, qui est notre préoccupation, se situe dans le cadre des tentatives actuelles de construire de vastes ensembles à traitement réparti fonctionnellement.

Le modèle que nous proposons consiste à conserver la structure de l'hybride de type II, où la partie analogique sera substituée par des éléments numériques, avec le critère principal de maintenir son caractère de calcul parallèle. En effet, les possibilités apportées par l'amélioration quasi permanente des techniques L. S. I. et V. L. S. I. (microprocesseur, circuits spécialisés, réseaux logiques programmables) permettent d'argumenter cette voie.

Le point essentiel développé dans cette thèse, est l'étude des performances d'opérateurs numériques, en utilisant des techniques et des technologies différentes.

Dans le premier chapitre, c'est la technique programmée, selon une architecture multiprocesseur qui est proposée et présentée en trois étapes :

- Description du calculateur central (l'analogue du calculateur numérique dans un modèle hybride de type II), qui est le microcalculateur d'INTEL, l'ISBC^{MT} 86/12A.

- Définition d'un processeur pour une structure multiprocesseur à "Bus Commun".

- Dans la 3ème partie de ce chapitre, la simulation d'un pas d'intégration numérique sur un processeur, nous a permis de conclure que notre processeur ne dispose pas d'une base de temps suffisamment rapide pour assumer cette tâche.

Dans le deuxième chapitre, c'est la technique cablée qui est envisagée. Cette approche consiste à imaginer un réseau d'opérateurs numériques. Un moyen de communication adéquat entre les opérateurs permettra le cablage d'un processus selon le modèle analogique.

Dans ce deuxième chapitre et dans un premier temps, nous étudions quelques notions théoriques de base, des opérateurs arithmétiques primitifs (addition, multiplication). Dans un deuxième temps, les performances de ces éléments matérialisés à l'aide des technologies S.S.I. et F.P.L.A. sont données.

Le troisième chapitre se place dans l'hypothèse où nous disposons d'opérateurs numériques rapides satisfaisant le critère énoncé plus haut. Il convient d'intégrer ces opérateurs dans un ensemble matériel et logiciel, qui forme notre calculateur appelé "simulateur digital". L'utilisateur proposera un modèle mathématique et le cablage sera pris en charge par la machine. Dans ce contexte, le schéma de cablage permettant la description formelle d'un cablage, facilitera vraisemblablement la mise au point du langage de simulation et d'un moyen de communication entre les opérateurs.

o

o o

CHAPITRE I

LES MULTIPROCESSEURS

LES MULTIPROCESSEURS

INTRODUCTION

Il a été spécifié dans l'introduction générale, le but de ce mémoire, qui se traduit par la recherche d'un matériel et de son organisation autour d'un calculateur numérique, ici l'ISBC ^{MT}86/12A.

Le système qui en résulterait, où toutes les variables apparaissent sous forme quantifiées, aura toutes les caractéristiques du calcul numérique, il doit de plus conserver le caractère parallèle des opérations du calcul analogique.

Dans ce contexte, les multiprocesseurs semblent être une première voie d'investigation. Dans ce chapitre, nous proposons d'étudier cette solution, selon trois phases.

La première partie est consacrée à la description du calculateur ISBC ^{MT}86/12A.

La seconde, à la présentation de quelques modèles d'architectures multiprocesseurs et leurs caractéristiques.

Enfin, nous étudierons la structure "bus-commun", simple, mais qui nous permettra de conclure quant au choix d'une solution multiprocesseur.

I.1 - DESCRIPTION ET PRESENTATION DE L'ISBC ^{MT}86/12A

I.1.1 - Généralités

Conçue par INTEL et contrôlée par le microprocesseur 8086 A, la carte ISBC 86/12A, baptisée "Multibus", est un microcalculateur standard complet sur un seul circuit imprimé.

Il permet d'adresser 1 million d'octets de mémoire et assure le transfert de données de 8 et 16 bits. Il comporte une capacité de mémoire vive dynamique de 32 k, une interface de communication série, 3 ports d'E/S parallèles programmables, 3 compteurs programmables, un contrôleur de priorité d'interruptions et peut supporter jusqu'à 16 k Octets de mémoire morte.

Le débit maximal est de 5 millions de transferts (octet ou mot de 16 bits) par seconde. Une des caractéristiques intéressantes du multibus, est la possibilité de connecter plusieurs maîtres sur le même bus et pouvant accéder à sa mémoire locale.

L'allocation du bus se fait soit par simple priorité (Daisy-Chain), soit par priorité parallèle.

I.1.2 - L'unité centrale |1| |2| |5| |8|

L'unité centrale de l'ISBC 86/12A est le microprocesseur 8086 de INTEL. Il travaille à une fréquence de 5 MHz et traite des mots de 8 et 16 bits. Son architecture interne est divisée en deux parties : l'unité d'exécution (EU) et l'unité d'interface du bus (BIU).

L'EU contient les registres de données et l'ALU. Le BIU effectue une pré-recherche des instructions et les stocke dans une file d'attente de 6 octets, il permet de réduire le cycle minimum d'une instruction de 1,2 µs à 400 ns sur le principe pipe-line. Elle effectue le calcul des adresses et fournit le bus de contrôle.

Ses registres sont au nombre de 14 et répartis comme suit :

* 1 registre de conditions de 16 bits dont 9 seulement sont significatifs
* 1 compteur ordinal de 16 bits
* 12 registres de 16 bits organisés en 3 groupes :

. Un groupe de 4 registres nommés registres généraux ou groupe HL, dont les parties haute et basse peuvent être adressées séparément comme des registres de 8 bits

. Un groupe de 4 registres nommés registres pointeurs et indexes,

ou groupe PI

. Un groupe de 4 registres nommés registres de segment ou groupe S.

Ces registres sont utilisés pour tous les calculs des adresses mémoires.

I.1.3 - Les instructions |1| |2|

Au nombre de 113 et de longueurs différentes (8, 16 ou 32 bits), les instructions du 8086 sont répertoriées en 6 types d'opérations :

* Opérations de transfert
* Opérations arithmétiques incluant la multiplication et la division
* Manipulation au niveau du bit
* Manipulation de chaîne de caractères
* Transfert de programme
* Commande et contrôle du processeur (manipulation du registre d'état et synchronisation avec l'extérieur).

I.1.4 - Structure du bus de l'ISBC 86/12A |3| |4| |7|

L'architecture de l'ISBC 86/12A est organisée autour de 3 bus hiérarchisés (Figures I.1 et I.2).

a) Le bus local qui connecte l'unité centrale à tous les circuits d'Entrées/Sorties et de la mémoire morte (ROM / EPROM) résidant (sur la carte) et au bus double accès de la mémoire vive.

b) Le bus double accès qui contrôle la mémoire vive (RAM dynamique) et communique avec le multibus et le bus local. Il peut être dans l'un des 3 états suivants :

- Contrôlé par le bus local sans l'utiliser (non accupé)
- Contrôlé et utilisé par le bus local (occupé)
- Contrôlé et utilisé par le multibus (occupé).

c) Multibus ou bus système, qui permet à un autre maître d'accéder à la mémoire de l'ISBC 86/12A, ou à l'unité centrale de communiquer avec des modules connectés sur le multibus.

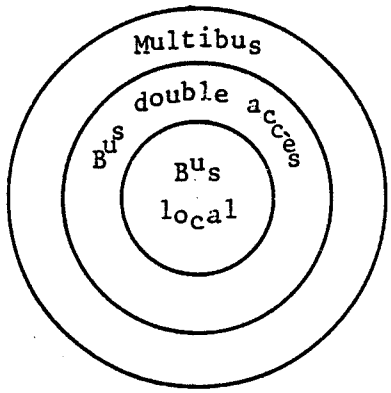


Figure I.1

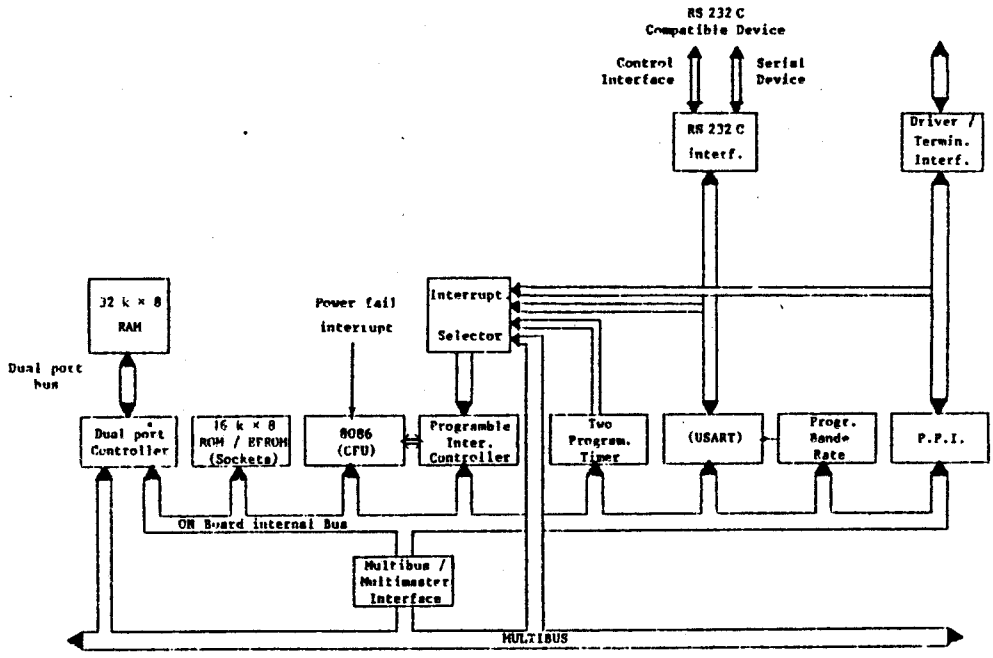


Figure I.2

I.1.5 - Organisation de la mémoire |3| |7|

I.1.5.1 - Mémoire vive

L'ISBC 86/12A contient 32 k Octets de RAM dynamique accessible par l'unité centrale et par un autre maître connecté sur le bus système.

Pour l'unité centrale, la zone mémoire est fixée de 1'adresses OH jusqu'à 07FFFH.

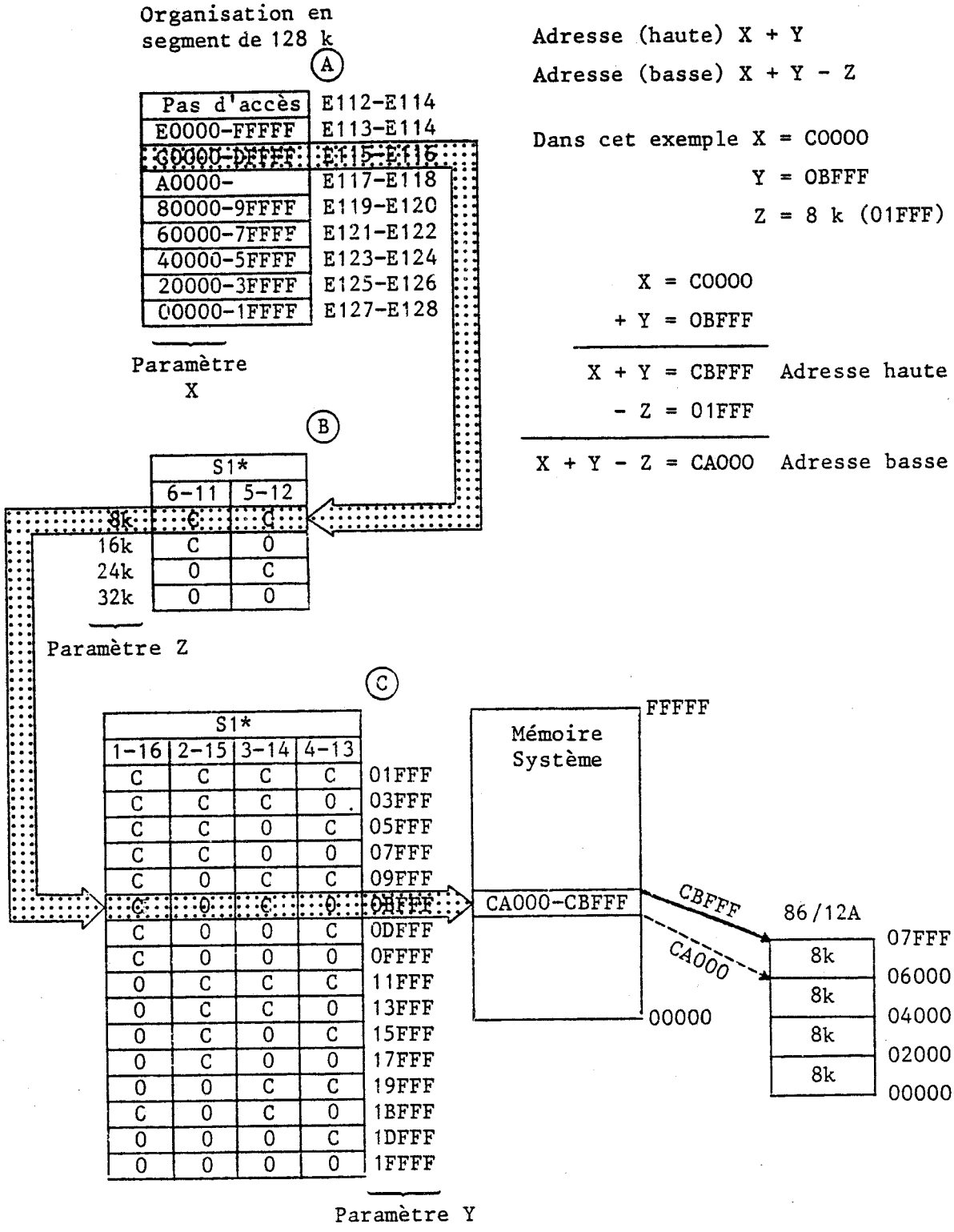
Pour un maître extérieur, l'utilisateur doit spécifier les paramètres X, Y et Z par un ensemble de straps (E112 - E198) et un sélecteur S1 à 8 pôles indépendants.

Le paramètre X sélectionne un segment de 128 k Octets parmi 8 possibles. Le paramètre Z sélectionne la taille mémoire accessible par le maître extérieur qui peut être de 8, 16, 24 ou 32 k Octets. Le paramètre Y fixe la borne supérieure de la zone mémoire accessible.

La figure I.3 donne un exemple de configuration de la mémoire. Dans cet exemple, l'adresse physique de la mémoire est de OH à 07FFFH. Pour accéder à la mémoire, l'unité centrale doit générer une adresse comprise entre OH et 07FFFH. Par contre, un maître extérieur ne peut accéder qu'à la zone mémoire de 06000H à 07FFFH et pour se faire, il doit générer une adresse comprise entre CA000H et CBFFFH.

La zone mémoire commune occupe toujours les Z k Octets de la partie haute de la mémoire.

Exemple de configuration de la RAM



Adresse (haute) X + Y

Adresse (basse) X + Y - Z

Dans cet exemple X = C0000

Y = 0BFFF

Z = 8 k (01FFF)

X = C0000		
+ Y = 0BFFF		
X + Y = CBFFF	Adresse haute	
- Z = 01FFF		
X + Y - Z = CA000	Adresse basse	

* C : Fermé
O : Ouvert

Figure I.3

I.1.5.2 - Mémoire morte |7|

La carte ISBC 86/12A peut supporter jusqu'à 4 boitiers de mémoire morte de type 2758 (2 k), 2716 / 2316E (4 k), 2732 / 2332A (8 k) |8|.

Un jeu de straps permet leurs configurations de façon à avoir un adressage de :

FF000H à FFFFFH en utilisant des boitiers de 2 k
FE000H à FFFFFH en utilisant des boitiers de 4 k
FC000H à FFFFFH en utilisant des boitiers de 8 k.

I.1.6 - Organisation des Entrées/Sorties |7|

I.1.6.1 - Entrées/Sorties parallèles

En nombre de 24 lignes programmables et gérées par un 8255 A |8|, les entrées/sorties parallèles sont organisées en 3 ports (A, B, C) de 8 bits. Ils sont programmables indépendamment, comme des ports d'entrées/sorties unidirectionnels ou bidirectionnels. Leur liaison avec les signaux extérieurs est assurée par l'existance de supports pour des circuits adaptateurs interchangeables (amplificateurs de lignes / récepteurs). Ces lignes d'E/S sont disponibles sur le connecteur J1.

I.1.6.2 - Entrées/Sorties séries

La carte ISBC 86/12A est équipée d'un USART, le 8251 A |8|. La fréquence et la technique (synchrone/asynchrone) de transmission sont contrôlées par programme et servent d'interface pour tout périphérique traitant des données séries. Les lignes d'E/S séries sont disponibles sur le connecteur J2.

I.1.6.3 - Horloge programmable

La carte ISBC 86/12A contient un 8253 |8| qui incorpore 3 compteurs 16 bits indépendants et totalement programmables. Deux de ces compteurs peuvent être utilisés selon les besoins et un compteur fournit la fréquence de transmission et réception au 8251 A.

Le contenu de chaque compteur peut être lu et écrit à tout moment pendant le fonctionnement du système.

I.1.7 - Organisation du multibus |7|

Le multibus comprend les lignes de contrôle, les lignes d'adresses et d'inhibition, les lignes de données, les lignes d'interruption, les lignes de contrôle du bus, les alimentations et les protections contre les coupures de courant.

Tous les signaux, exceptés ceux de protections contre les coupures de courant, sont disponibles sur le connecteur P1. Les signaux de protection existent sur le connecteur auxiliaire P2 (Annexe I).

I.1.7.1 - Lignes de contrôle

a) Les horloges :

* BCLK est l'horloge du multibus. Elle assure la synchronisation des logiques de gestion de priorités sur chaque module maître.

* CCLK est l'horloge générale, destinée aux modules maîtres ou esclaves. Si le système dispose de plusieurs maîtres, seul un d'entre eux générera l'horloge générale.

b) Commande de bus :

Elles permettent au maître de contrôler les esclaves. Ces commandes sont :

MRDC/ : Commande de lecture de la mémoire
MWRC/ : Commande d'écriture de la mémoire
IORC/ : Entrée à partir d'un périphérique
IOWC/ : Sortie vers un coupleur

c) Synchronisation avec la périphérie :

XACK/ : ce signal indique au maître la fin d'un transfert au niveau de la mémoire ou des entrées/sorties.

d) Initialisation :

INIT/ : permet d'initialiser le système complet. Ce signal est en général généré avant le démarrage de toute opération. Il est élaboré

par n'importe lequel ou par tous les maîtres. Il peut provenir d'une source extérieure.

I.1.7.2 - Lignes d'adresses et d'inhibition

Le multibus permet d'adresser jusqu'à 1 million d'octets de mémoire. Il dispose pour réaliser cet adressage, des lignes suivantes :

* ADR0/ - ADR13/ : bus de 20 lignes d'adresses. Ces lignes sont référenciées en hexadécimal (0 - 9, A - F, 10 - 13). Dans le cas de l'adressage des entrées/sorties, seules les 16 premières lignes sont utilisées.

* INH1/ : ligne d'inhibition. Si la RAM et la ROM occupent le même espace d'adressage, INH1/ valide la ROM et inhibe la RAM.

* BHEN/ : validation de l'octet de poids fort dans un mot de 16 bits.

I.1.7.3 - Lignes de données

DAT0/ - DATF/ : lignes de données référenciées en hexadécimal. Elles assurent la transmission ou la réception des données sur 8 ou 16 bits. Dans le cas de transferts sur 8 bits, seules 8 lignes de poids faible DAT0/ - DAT7/ sont utilisées.

I.1.7.4 - Lignes d'interruptions

INT0/ - INT7/ : 8 lignes de demandes d'interruptions 0 à 7 envoyées vers le module maître. Le niveau 0 est le niveau le plus prioritaire.

I.1.7.5 - Lignes de contrôle et de gestion du bus

Elles se chargent de la gestion du bus dans un système multiprocesseur, sur le principe demande et réponse par priorité. La gestion de priorité se fait en deux modes (examinés dans la suite de ce chapitre) : série ou parallèle.

Les lignes de contrôle contiennent les signaux suivants :

* BREQ/ : demande d'accès au bus
* BPRN/ : entrée qui indique au module maître qu'il est le plus prioritaire
* BPRO/ : sortie à connecter au BPRN/ du maître de priorité inférieure, dans le cas d'une gestion série des priorités
* BUSY/ : indique que le multibus est occupé
* CBRQ/ : prévient le maître qui utilise le multibus qu'un autre maître est en attente.

I.1.8 - Opérations de transfert de données par multibus |7|

La description des signaux du multibus telle que faite précédemment en conjonction avec les chronogrammes (Annexe I), donne d'une façon générale la technique de transfert de données entre les modules.

Tout transfert débute par l'envoi d'une adresse de la ressource désirée. Un système de décodage et une logique de contrôle permettent l'accès ou non à cette ressource. Quand la ressource est accordée, les commandes sont autorisées et le transfert aura lieu.

I.1.9 - Système d'interruptions (Figure I.7)

Deux modes d'interruptions sont possibles : interruptions vectorisées (BVI) et non vectorisées (NBVI). Pour les deux modes, c'est le 8259 A |8| qui existe sur la carte, qui joue le rôle de maître.

I.1.9.1 - Interruptions non vectorisées NBVI

Dans ce mode d'interruptions, le vecteur d'interruption est généré par le contrôleur d'interruption localisé sur le module maître et envoyé vers l'unité centrale par le bus local. Ce sont les 8 lignes (INT0/ - INT7/) qui sont utilisées pour demander une interruption.

I.1.9.2 - Interruptions vectorisées BVI

Dans ce type d'interruption, c'est le module esclave demandant l'interruption qui génére le vecteur d'interruption et l'envoie au maître par le multibus. L'extension jusqu'à 64 niveaux d'interruptions est possible.

Remarque : Dans les deux cas, le module maître doit être configuré de façon à avoir la possibilité d'accéder au multibus.

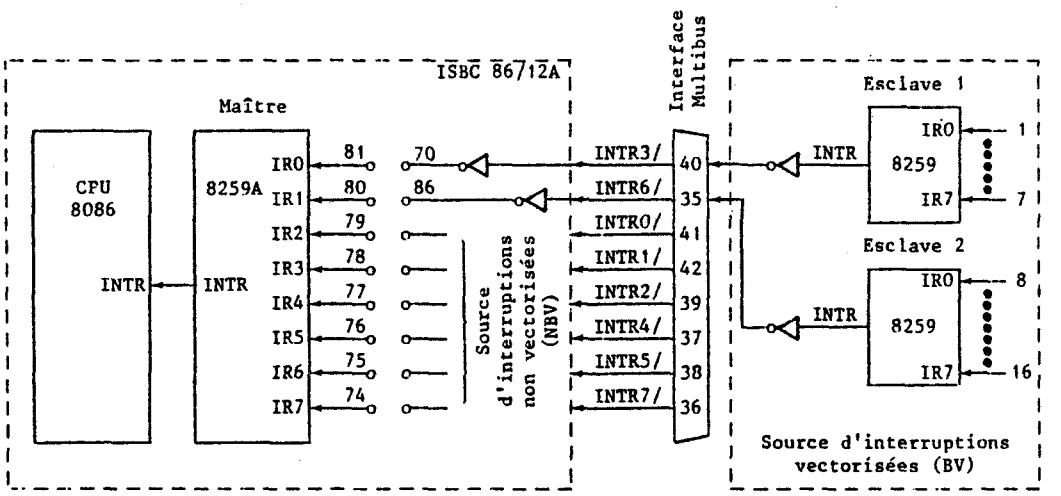
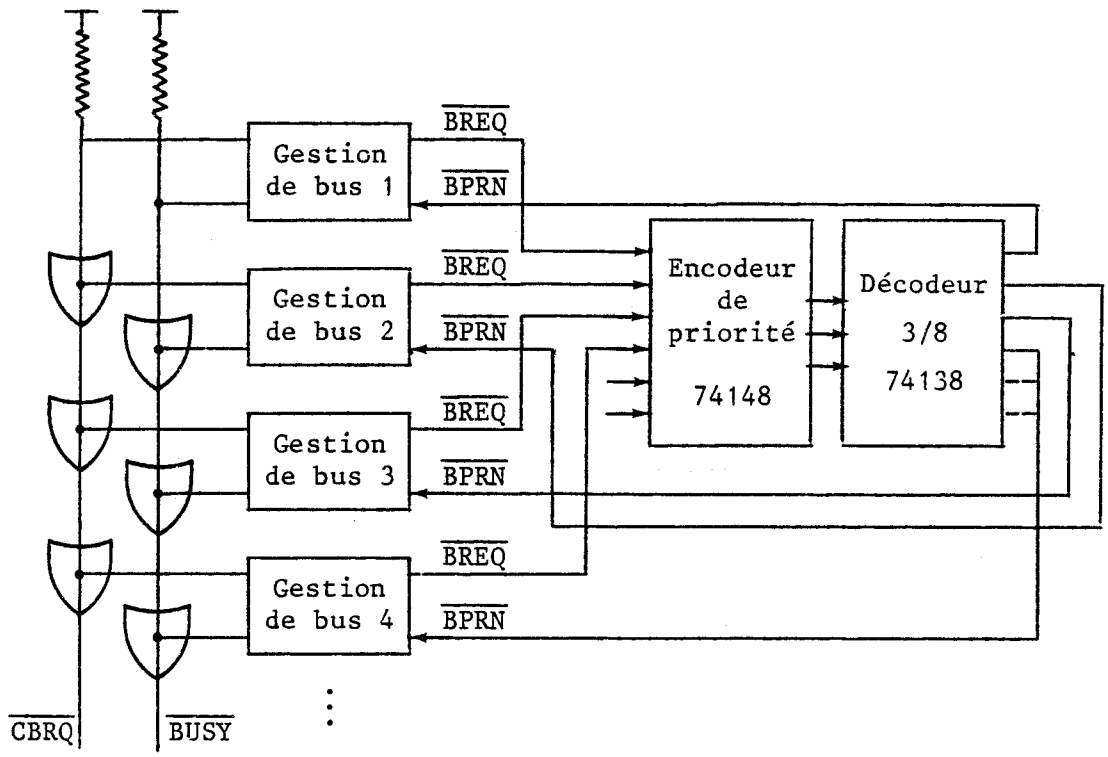


Figure I.7

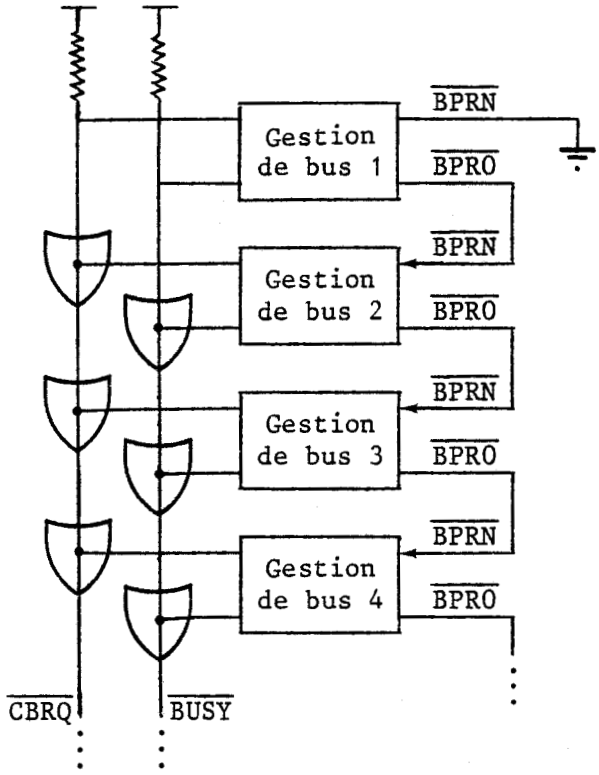
I.1.10 - Gestions de priorités et d'accès au multibus

C'est un circuit spécialisé, le 8289 6 d'INTEL, qui est chargé de la gestion du multibus. Suivant l'état du processeur, il formule la demande d'accès au multibus par le signal BREQ/. Cette demande est traitée par un circuit de gestion de priorités, qui répond par un signal BPRN/, indiquant qu'il est le plus prioritaire et doit attendre la disponibilité du multibus qui lui sera signalée par BUSY/.



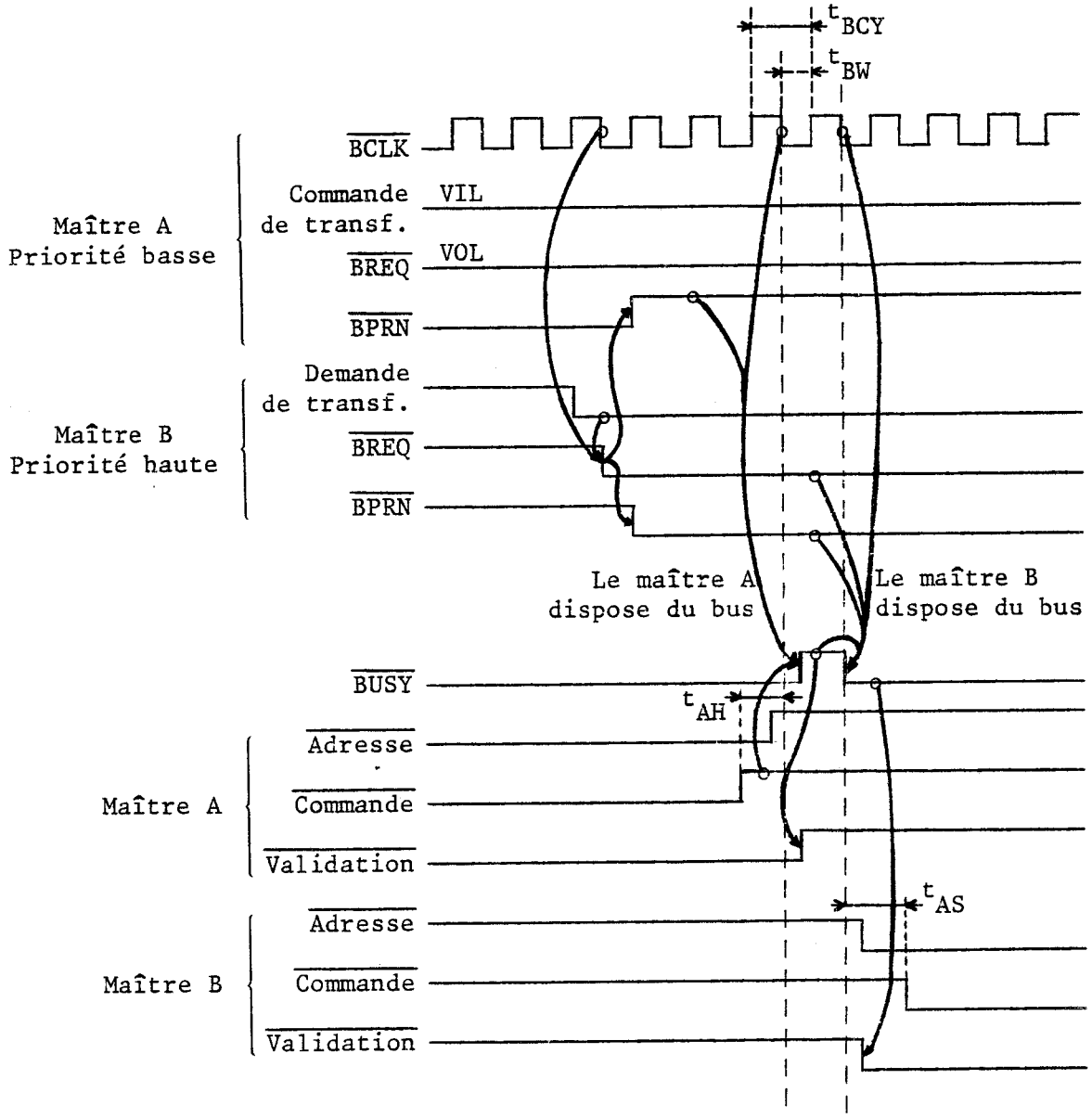
GESTION DES PRIORITES PARALLELES

Figure I.8



GESTION DES PRIORITES SERIES

Figure I.9



CHRONOGRAMME DE LA PRISE DE CONTROLE DU BUS
PAR UN MAITRE PLUS PRIORITAIRE

Figure I.10

t_{AS} = 50 ns ; t_{AH} = 50 ns ; t_{BCY} = 108 ns ; t_{BW} = 35 ns à 74 ns

I.1.11 - Conclusion

La présentation et la description du calculateur ISBC 86/12A nous ont conduit à étudier tous les circuits composant la carte. Parmi les plus intéressants, le 8288 et le 8289 (Annexe I) permettent une large

flexibilité dans la configuration des systèmes, dans un environnement multiprocesseurs. Nous verrons par la suite, la facilité offerte par ces circuits pour concevoir un système multiprocesseur autour de l'ISBC 86/12A.

I.2 - LES SYSTEMES A HAUTE PERFORMANCE : LES MACHINES PARALLELES ET LES MULTIPROCESSEURS

Différentes interprétations, concepts et modèles ont été introduits pour caractériser les systèmes hautement performants. Une caractéristique commune à tous ces systèmes est la recherche de l'exploitation à tous les niveaux du parallèlisme apparaissant dans un programme de façon implicite ou explicite. Cette recherche, qui ne se fait pas sans problème, ne résulte pas d'une simple curiosité intellectuelle, mais constitue un outil indispensable à la résolution de nombreux problèmes, de natures différentes et complexes, qui nécessite le traitement rapide d'un très grand nombre d'équations simultanées et des techniques de simulation et de contrôle avancées.

I.2.1 - Domaines d'application

Trois grandes catégories de domaines de mise en œuvre peuvent être distinguées :

1. Traitement du signal (météorologie, radar de poursuite, étude des séismes, traitement d'image, analyse biomédicale, détection d'anomalies magnétiques, ...)

2. Planification et optimisation (analyse nucléaire, analyse structurale, programmation dynamique, programmation linéaire, ...)

3. Calcul économique et scientifique (économétrie, aérodynamique, géophysique, optimisation et modélisation de ressources, analyse de la pollution, analyse médicale, ...).

Les méthodes typiques de résolution de tels problèmes nécessitent la programmation linéaire, le calcul vectoriel, le calcul de la trans-

formée de Fourrier rapide ou de la convolution et le filtrage numérique.

Les grandes classes de calculateurs qui permettent de tels traitements sont les machines pipe-lines et les multiprocesseurs.

I.2.2 - Les différents modèles d'architectures

De nombreuses approches de classification des architectures de calculateurs sont possibles. Nous citons ci-après quelques techniques de classification proposées par différents auteurs.

Flyn |9| propose une technique de classification par flot de données et d'instructions comme indiqué ci-dessous :

SISD : un flot de données et un flot d'instructions ; c'est la structure d'une machine classique uniprocesseur (ex : IBM System / 360).
SIMD : un flot d'instructions et plusieurs flots de données ; c'est la structure de machines telles que Illiac IV |10|, et STAR |11|.
MISD : un flot de données et plusieurs flots d'instructions ; peu de machines ont véritablement cette structure, interprétée par certains comme un système pipe-line |12|.
MIMD : plusieurs flots de données et plusieurs flots d'instructions ; c'est la structure multiprocesseurs telle que UNIVAC 1108.

On trouve dans |13| une technique de classification basée sur les propriétés du parallèlisme, proposée par Murtha et Beadles (unité de contrôle commune et centralisée, processeurs identiques, parallèlisme global, parallèlisme local).

Hobbs et al |14| suggèrent une classification d'après les ressources mises en parallèle (flot de données, unités de contrôle, unités de traitement)et d'après les réalisations (multiprocesseurs, processeurs associatifs, réseaux et tableaux de processeurs, machines fonctionnelles).

Handler |15| propose une classification des processeurs à haute performance en décrivant une machine à l'aide de la fonction t(machine) :

$$t(\text{machine}) = (k, d, w)$$

où k représente le nombre d'unités de contrôle ; d et w, le nombre d'unités arithmétiques et logiques et la complexité du circuit logique respectivement.

Exemples :

t (Bourrough - Illiac IV) = (1, 64, 64)
t (Good Year - Staran B) = (1, 8192, 1)
t (Cannegie Mellon - Cmmp |16|) = (16, 1, 16)

Bien d'autres classifications ont été proposées : Higbie |17|, Shore |18|, Enslow |19|, Fuller et al |20|, Bell |21|.

La multiplicité des techniques de classification montre la difficulté d'établir des modèles de machines significatifs et précis. Ceci est d'autant plus vrai qu'un bon nombre de machines actuelles ne correspond pas d'une manière assez nette à tel ou tel critère.

Ces modèles sont nécessaires, non seulement pour la conception et la simulation, mais aussi pour la sélection des systèmes compétitifs du futur.

I.2.3 - Les systèmes multiprocesseurs

I.2.3.1 - Définition

Là encore, plusieurs définitions sont possibles. D'une manière générale, un multiprocesseur est un système informatique dans lequel plusieurs processeurs fonctionnent en parallèle en se partageant un même ensemble de mémoires et d'unités périphériques. Les points de vue sont différents sur la définition et les types de processeurs utilisés, et la manière dont est géré le système global (système d'exploitation régissant l'intéraction entre les processeurs, les ressources communes, les travaux à exécuter).

Ces considérations relèvent de chaque concepteur et du type d'application visé. Au lieu de donner une définition dogmatique d'un système

multiprocesseurs, nous proposerons plutôt quelques techniques de communication entre processeurs qui est un des problèmes essentiel posé par ces systèmes.

I.2.3.2 - Communication entre processeurs

La procédure classique et la plus utilisée de communication entre processeurs, est celle de la mémoire commune.

* Utilisation de la mémoire circulante :

Plusieurs modules de mémoire circulante |22|, |23|, disposés en série avec bouclage du dernier module sur le premier afin de former un anneau, constituent la mémoire commune. La liaison entre deux modules constitue la fenêtre d'accès. Les processeurs sont répartis autour de l'anneau et placés devant les fenêtres d'accès (Figure I.2.1). L'exemple d'une telle réalisation est donnée par Maud |24|.

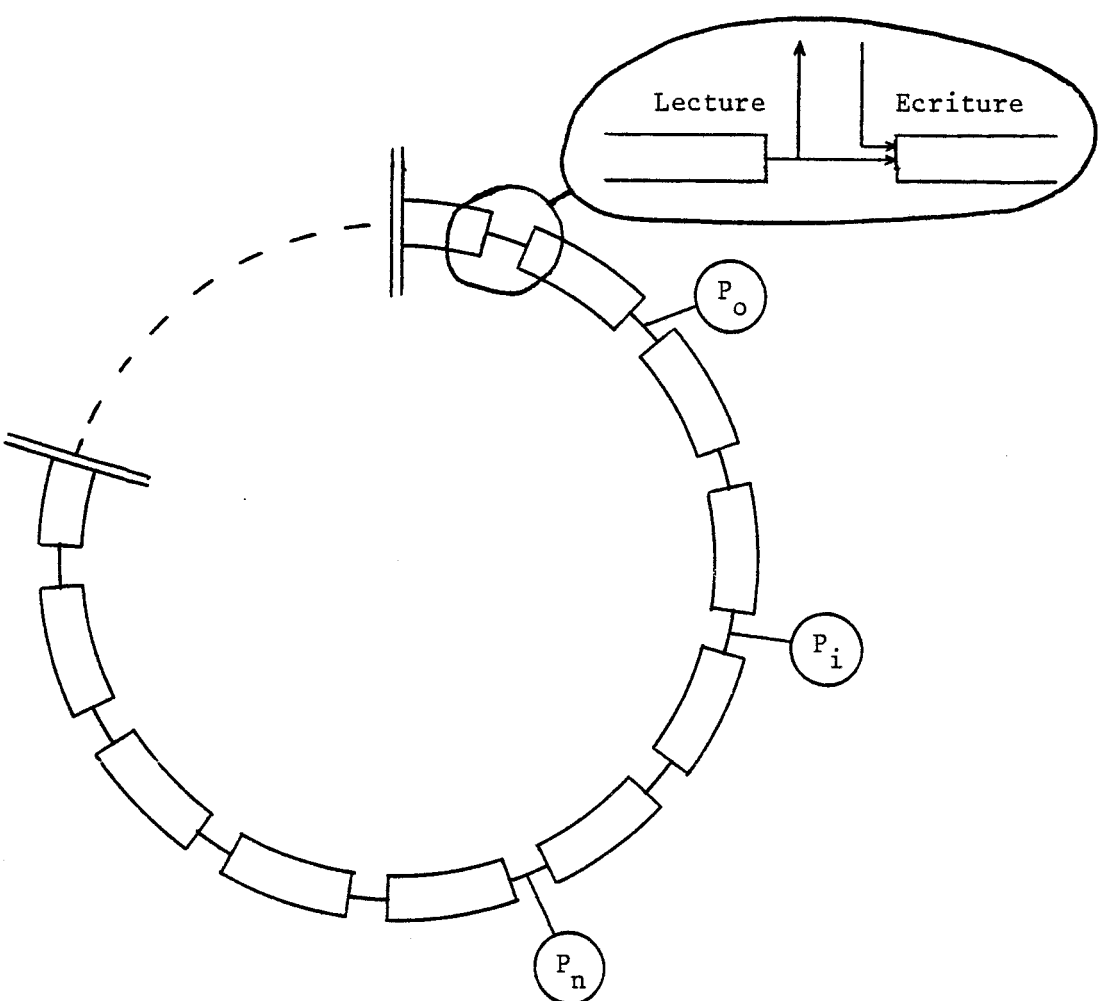
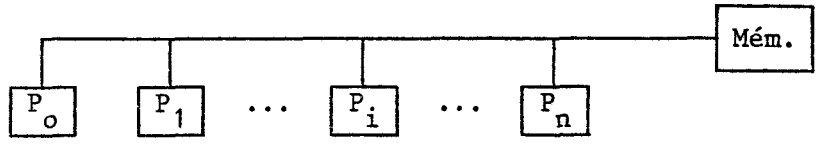


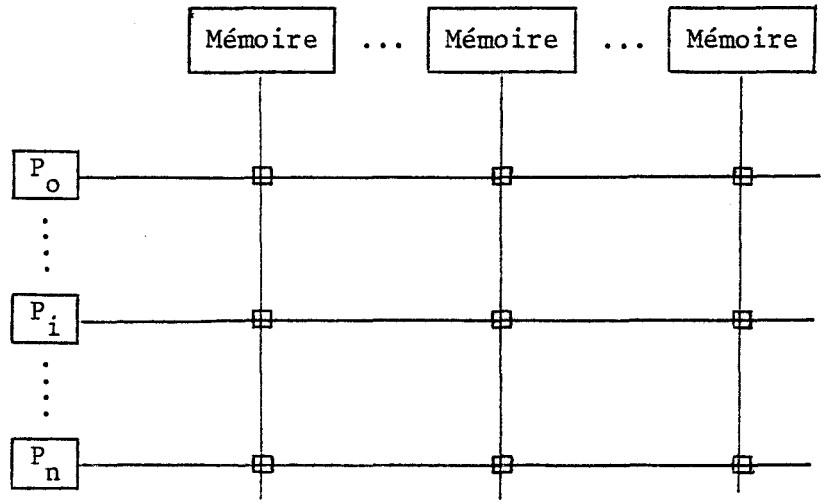
Figure I.2.1

* Accès à la mémoire par un bus commun :



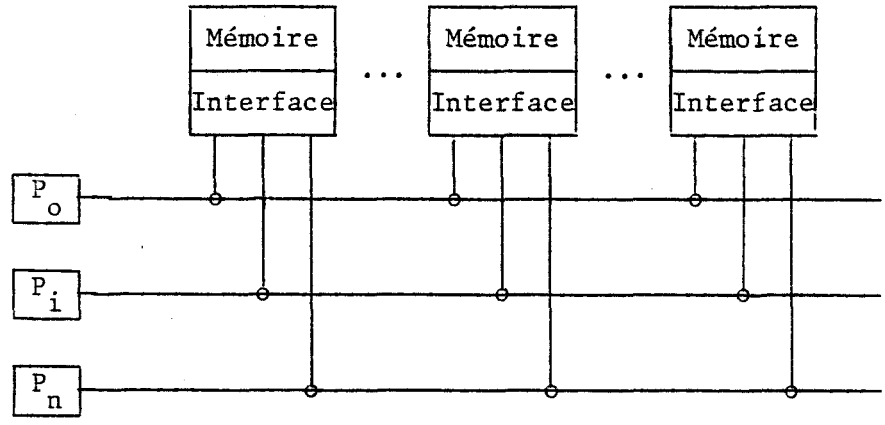
Cette structure est caractérisée par une faible complexité fonctionnelle, un faible coût d'interconnexion, une expansion facile, mais se trouve rapidement saturée lorsque croît le nombre de processeurs. L'accès au bus commun est usuellement contrôlé par un arbitre |25|.

* Accès à la mémoire par un Cross-Bar :



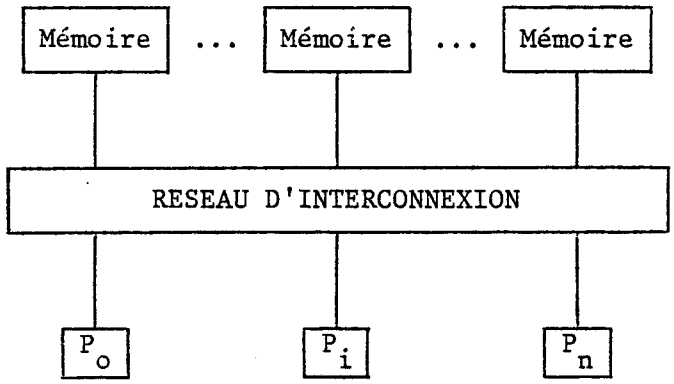
Ici, la mémoire est divisée en blocs. L'accès à chaque bloc se fait au moyen d'un bus commun à tous les processeurs. La difficulté essentielle du Cross-Bar est la croissance de sa complexité comme le produit du nombre de processeurs par le nombre de blocs mémoire.

* Utilisation de mémoires multiports :



Ce sont les interfaces associées à chaque bloc mémoire sui gérent les conflits d'accès. La centralisation de la logique d'arbitrage et de priorité au niveau de chaque interface permet une large variété de configurations des systèmes. Mais c'est une solution très lourde et difficilement extensible.

* Utilisation d'un réseau d'interconnexion :



Le réseau d'interconnexion permet d'établir le lien entre n'importe quel processeur et n'importe quel bloc mémoire. Il est caractérisé par un débit élevé, mais cependant, le degré de simultanéité d'accès est limité par le nombre de blocs mémoire indépendants. Dans ce cas, chaque processeur élémentaire peut disposer d'une mémoire locale.

I.3 - ETUDE D'UNE ARCHITECTURE MULTIPROCESSEURS AUTOUR DE L'ISBC 86/12A

Compte tenu de la structure même de 1'ISBC 86/12A décrite dans le paragraphe I.1, où l'accès à la mémoire commune est géré par un arbitre, le 8289 |6|, et pour des raisons de compatibilité et de simplicité, nous proposons dans un premier temps d'analyser les caractéristiques et performances de la structure "bus commun" (Figure I.3.1).

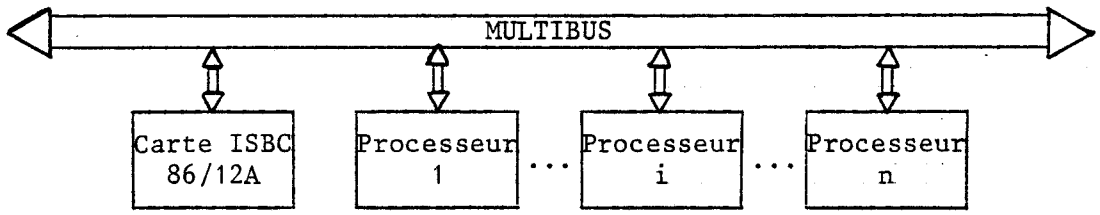


Figure I.3.1

Le nombre de processeurs peut aller jusqu'à trois si la technique des gestion de priorité utilisée est série et jusqu'à 16 si elle est parallèle.

Pour augmenter le nombre de processeurs et éviter la saturation rapide de la machine, une structure hiérarchisée peut être envisagée et confère à la machine la structure du calculateur SMS 201-203 |26| (Figure I.3.2).

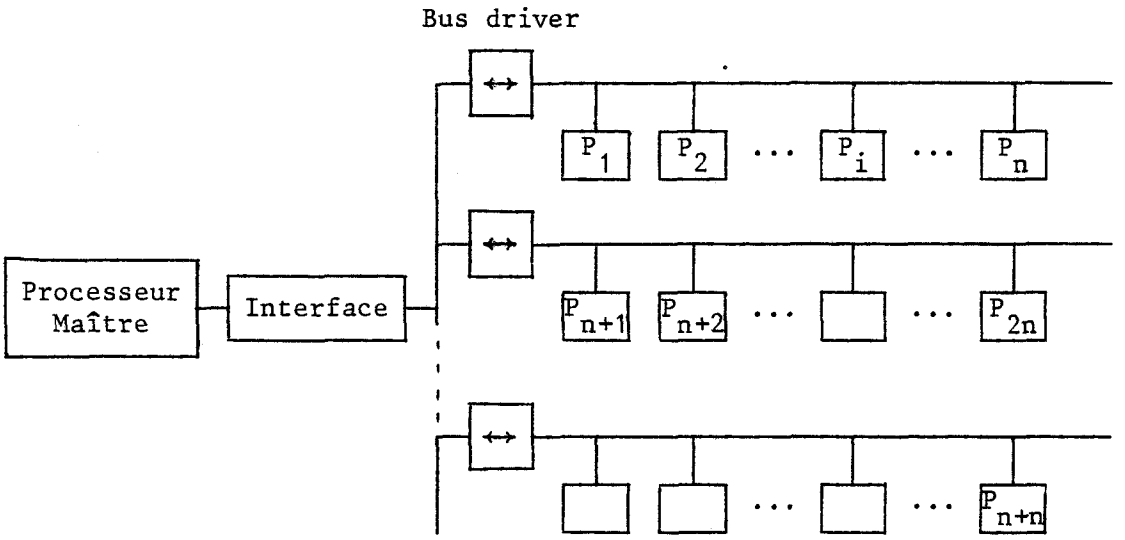


Figure I.3.2

Au niveau de notre étude, nous nous sommes limités à l'élaboration du schéma de réalisation d'un processeur local ; dans ce cas, tous les processeurs sont supposés identiques.

I.3.1 - Définition d'un processeur

Un processeur doit être capable d'opérer seul, hors de tout environnement, pendant un bref laps de temps, ce qui suppose qu'un processeur doit réunir une unité arithmétique et logique, une unité de contrôle, une mémoire locale et des dispositifs d'Entrées/Sorties. C'est un microprocesseur qui va jouer le rôle de l'unité arithmétique et logique et de l'unité de contrôle. Cette définition confère au processeur la configuration suivante (Figure I.3.3).

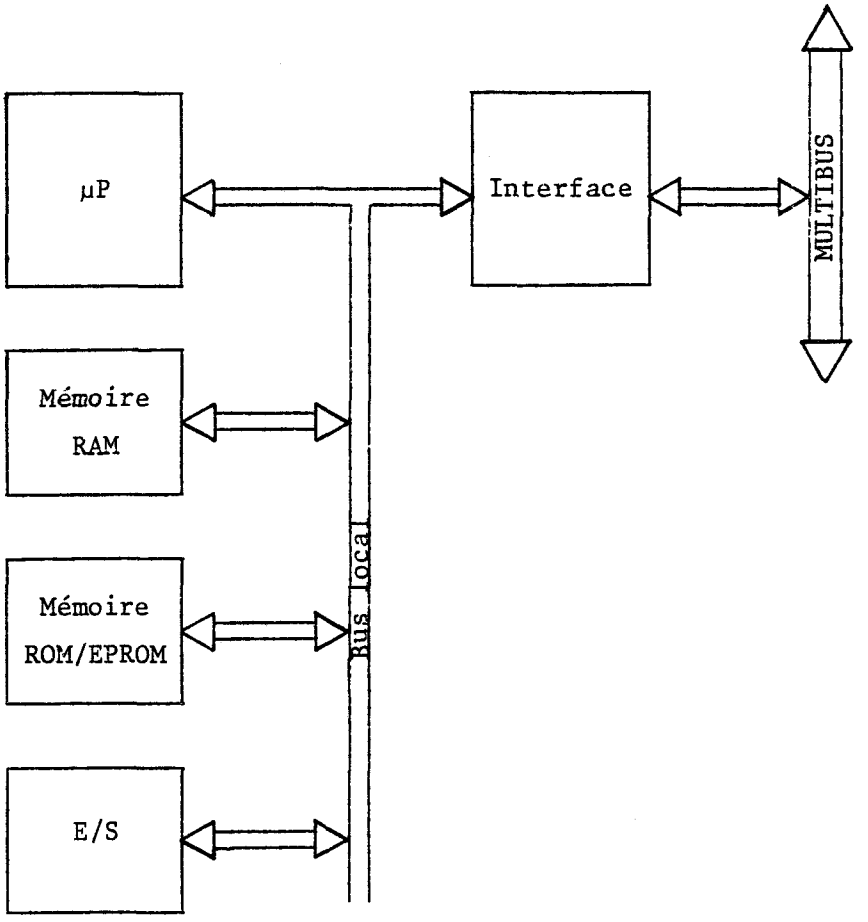


Figure I.3.3

I.3.2 - Etude d'un schéma de réalisation

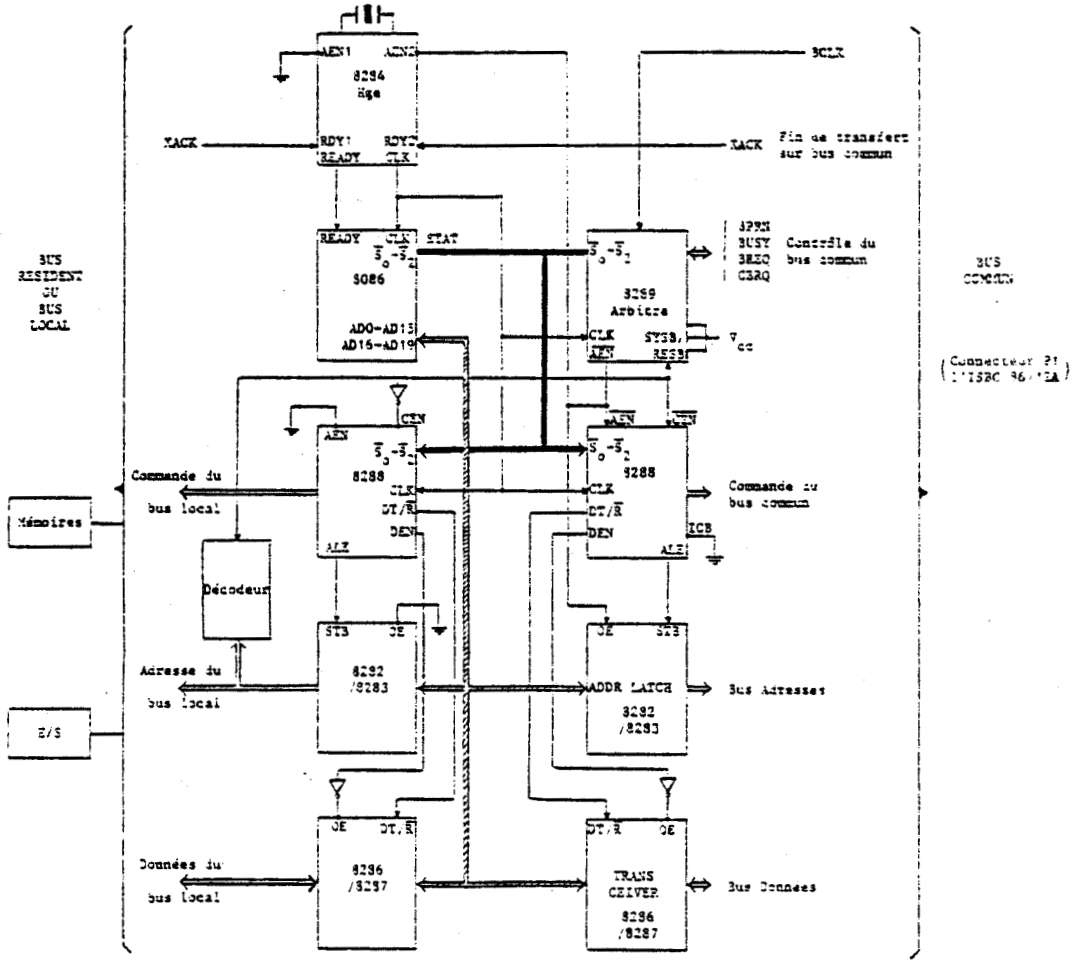
I.3.2.1 - Choix du microprocesseur

Compte tenu des matériels de développement disponibles au laboratoire, le choix du microprocesseur se fait au sein de la famille INTEL (8080, 8085, 8086, 8088).

Le premier objectif souhaité concerne la transposition des opérateurs analogiques d'un calculateur hybride en opérateurs numériques. Notamment, l'opération intégrale et sa génaralisation par rapport à une variable quelconque exige du processeur d'être le plus rapide possible, d'une bonne précision, il doit donc être capable d'effectuer les opérations de base, à savoir l'addition et la multiplication. Ces caractéristiques, ajoutées à des considérations d'ordre technologiques (compatibilité avec le multibus, facilité de réalisation de l'interface, ...) font que notre choix s'est porté sur 8086.

I.3.2.2 - Schéma proposé

Il ne nous paraît pas nécessaire d'établir ici le schéma complet du processeur et sa description détaillée. Nous nous contentons de donner ci-après, son schéma synoptique (Figure I.3.4) et quelques explications fonctionnelles.



SCHEMA SYNOPTIQUE D'UN PROCESSEUR

Figure I.3.4

I.3.2.3 - Principe de fonctionnement

On trouve dans |1|, |2|, |5|, |6|, |8|, le mode de fonctionnement de tous les composants qui figurent sur le schéma synoptique de la figure I.3.4.

Pour effectuer une lecture ou une écriture, le processeur génère l'adresse de la ressource considérée. Le 8288 qui contrôle le bus local, valide cette adresse sur le bus d'adresse local en envoyant un signal ALE au 8282 correspondant. Cette adresse est décodée par une logique de décodage. Si la ressource adressée est sur le bus commun, elle est indiquée à l'arbitre du bus commun (le 8289) par un signal de niveau logique haut sur son entrée "SYSB/$\overline{\text{RESB}}$". A ce moment, le 8289 génère une demande d'accès au bus commun, par le signal "BREQ". Ce signal est ensuite traité par le circuit de gestion de priorité qui répondra par le signal "BPRN" dès qu'il n'y a plus de processeur de priorité haute. Le 8289 attend la disponibilité du bus qui lui sera signalé par "BUSY" et l'autorise à valider le 8288 qui contrôle le bus commun, par le signal "$\overline{\text{AEN}}$". Ce même 8288 validera, à son tour, le bus d'adresses, de données et génère les signaux de commande. Le bus de données et les signaux de commande du bus local sont inhibés par le complément du même signal délivré par la logique de décodage et qui a alerté le 8289.

Si, au contraire, la ressource adressée est sur le bus local, la logique de décodage délivre un signal logique de niveau bas qui indique au 8289 qu'il s'agit d'une ressource locale. Ce signal complémenté, permet au 8288 qui contrôle le bus local, de valider le bus de données et de générer les signaux de commande.

I.3.3 - Evalutation des performances du système

L'opération intégrale est à la base de la résolution de la plupart des systèmes physiques qui généralement, sont régis par un système d'équations différentielles. Cette opération est facilement réalisable en calcul analogique, lorsque les grandeurs mises en jeu sont des variables qui ne dépendent que du temps. L'intégrale d'une grandeur s'effectue d'une façon continue et sans approximation théorique. En calcul numérique, cette opération s'effectue par approximation et nécessite plusieurs

opérations suivant l'algorithme utilisé. Dans le cas le plus simple, qui utilise la méthode dite des rectangles ou d'Euler, l'intégrale d'une fonction x(t), ayant les valeurs (x_0, x_1, ..., x_i, ..., x_n) aux instants (t_0, t_1, ..., t_i, ..., t_n) séparés par des intervalles de temps Δt, s'écrit :

$$I = I_0 + \sum_{i=1}^{N} x_n.\Delta t$$

Pour calculer l'expression de I numériquement, il convient de l'écrire sous la forme récurrente :

$$I_{n+1} = I_n + \Delta t.x_{n+1} \qquad (1)$$

Si, par ailleurs, on impose un gain k, l'équation (1) devient :

$$I_{n+1} = I_n + k.\Delta t.x_{n+1} \qquad (2)$$

où I_{n+1} constitue une approximation de l'intégrale :

$$I = k \int_{t_o}^{t_{n+1}} x(t).dt$$

Pour évaluer les performances du système et surtout les comparer au calcul analogique, nous plaçons le processeur dans le cas le plus défavorable, où il joue le rôle d'un intégrateur, et nous nous intéressons plus particulièrement à la vitesse de calcul et à la précision. Le processeur intégrateur reçoit à chaque intervalle de temps ΔT_c, les données x_n, Δt, k et délivre I_n.

Les valeurs x_n et I_n transitent par l'intermédiaire du bus commun. Δt et k sont supposées constantes et donc rangées dans la mémoire locale du processeur, lors de la mise en conditions initiales.

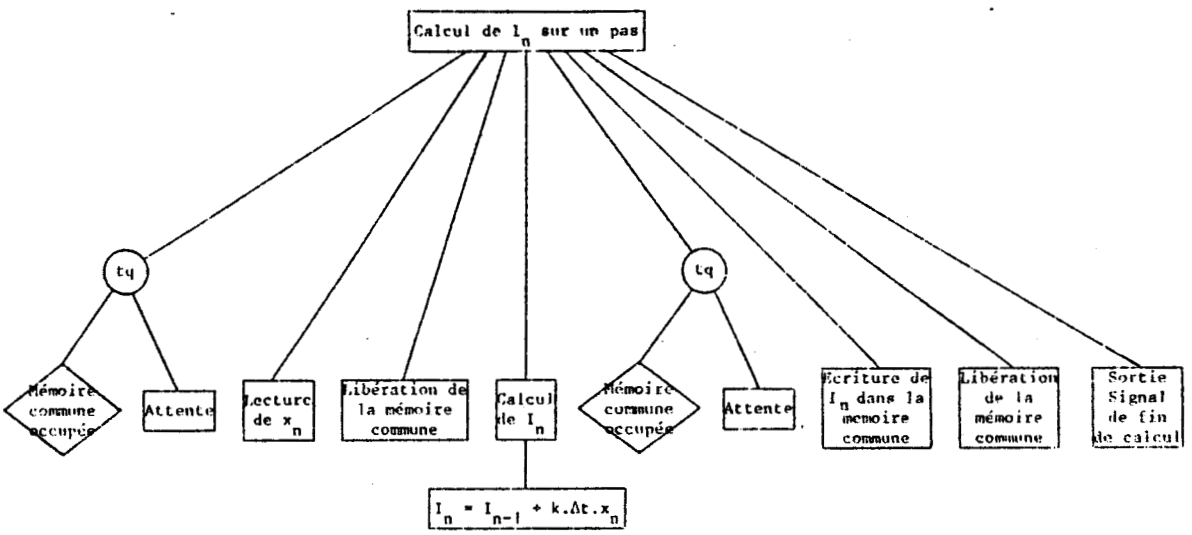
Dans ces conditions, une résolution élémentaire se déroule en 8 grandes étapes :

I
1) Gel de la mémoire commune
2) Lecture de x_n
3) Libération de la mémoire commune

II
4) Calcul de I_n

III
5) Gel de la mémoire commune
6) Ecriture de I_n dans la mémoire commune
7) Libération de la mémoire commune

8) Eventuellement, sortie d'un signal de fin de calcul



ORGANIGRAMME DU CALCUL DE I_n SUR UN PAS

Le programme écrit en assembleur, relatif à un tel traitement est le suivant :

```
PINTEG : MOV AL, 1
WAIT 1 : LOCK XCHG SEMA, AL     } Réservation de la mémoire
         TEST AL, AL            } commune
         JNZ WAIT 1             }
CAL I_n : MOV AX, COMM [x_n]    Lecture de x_n
         MOV SEMA, 0            Libération de la mémoire commune
MULT 1 : IMUL ALPHA [Δt]        x_n.Δt → DX.AX
         JNO NULL               [DX] non significatif
         RCL AX, 1              } Cadrage du résultat
         RCL DX, 1              }
         JO Corrige 1           { Test conservation du signe après cadrage
         MOV AX, DX             [DX] → [AX]
MULT 2 : IMUL ALPHA [k]         x_n.Δt.k → DX.AX
         JNO NULL               [DX] non significatif
         RCL AX, 1              } Cadrage du résultat
         RCL DX, 1              }
         JO Corrige 2           { Test conservation du signe après cadrage
         MOV AX, DX             [DX] → [AX]
MULT 3 : ADD AX, ALPHA [I_n]    I_n-1 + k.Δt.x_n → AX
         JO FAUT                Test débordement
         MOV BL, 1              }
WAIT 2 : LOCK XCHG SEMA, BL     } Réservation de la mémoire
         TEST BL, BL            } commune
         JNZ WAIT 2             }
         MOV COMM [I_n], AX     I_n → Mémoire commune
         MOV SEMA, 0            Libération de la mémoire commune
         MOV ALPHA [I_n], AX    { Conservation de I_n pour la prochaine résolution
SOR SIGNL : MOV AL, 01          } Sortie d'un signal de fin de calcul
         OUT RC                 }
CORRIGE 1 : XOR DX, 8000H       } Correction du signe du résultat de
         MOV AX, DX             } la première multiplication s'il est
         JMP MUL 2              } changé après cadrage
CORRIGE 2 : XOR DX, 8000        } Correction de signe du résultat de
         MOV AX, DX             } la deuxième multiplication s'il est
         JMP MUL 3              } changé après cadrage
```

Programme (suite) :

```
FAUT : MOV AL, 02
       OUT RC        } Sortie d'un signal indiquant un dépassement de capacité

NULL : MOV AL, 04
       OUT RC        } Sortie d'un signal indiquant un résultat nul ou non significatif à la suite d'une multiplication
```

avec :

COMM $[x_n]$, COMM $[I_n]$ les adresses de x_n et I_n dans la mémoire commune

ALPHA $[k]$, ALPHA $[\Delta t]$, ALPHA $[I_n]$ les adresses de k, Δt et I_n dans la mémoire locale.

Rappel : Le processeur traite des données sur 16 bits.
Le résultat de la multiplication de deux mots de 16 bits est donné sur 32 bits dans les registres DX . AX avec le contenu de DX comme partie la plus significative.
L'échelle est définie telle que toutes les grandeurs soient comprises entre + 1 et - 1.
+ 1 est représenté par 7FFFH = (+ 32767) décimal
- 1 est représenté par 8001H = (- 32767) décimal
La virgule est supposée fixe, juste à droite du bit de signe ; ce qui donne pour 1LSB : 2^{-15}.

Le cadrage du résultat de la multiplication se fait par décalage à gauche d'un bit du double registre DX, AX, suivi d'une correction du bit de signe s'il n'est pas conservé après l'opération de décalage. Pour le résultat final, nous prenons la partie la plus significative qui est dans DX.

Ainsi, on définit une précision de 2^{-15} sur la valeur approchée de l'intégrale.

Le temps moyen nécessaire au calcul d'un pas de I_n est de l'ordre de 100 µs.

Pour comparer les performances du système numérique et du calculateur analogique sur la base de l'intégration, il convient de définir la notion de temps de calcul et de précision.

Temps de calcul : c'est le temps nécessaire à la résolution d'un problème, pendant lequel la machine partant des conditions initiales déterminées, fournira l'évolution dynamique des grandeurs physiques.

- En calcul analogique :

Une grandeur x(t) évoluant en fonction du temps, passe de x(0) à x(T) pendant le temps T ; son intégrale I(t) partant de I(0) atteint la valeur I(T) au bout d'un temps T. On dit que T est le temps de calcul nécessaire pour passer de I(0) à I(T).

- En calcul numérique :

Si la même grandeur x(t) prend les valeurs x_0, x_1, ..., x_i, ...x_N aux instants $t_0 = 0$, t_1, ..., t_i, ..., t_N, son intégrale approchée prend les valeurs I_0, I_1, ..., I_i, ..., I_N aux mêmes instants $t_0 = 0$, t_1, ..., t_i, ..., t_N. Le temps de calcul nécessaire pour aller de I_0 à I_N est :

$$N.T_c$$

avec T_c = 100 µs dans ce cas.

Pour que notre système conserve le caractère temporel du calcul analogique, il faut que :

$$T = N.T_c \Longrightarrow N = T / T_c$$

Cette relation concerne à la fois le temps et la précision.

Du point de vue temps, elle traduit le fait qu'il faut prendre un échantillon toutes les 100 µs, c'est à dire que :

$$\Delta t_{min} = 100\ \mu s$$

et que T soit supérieur à $2\ T_c$ soit 200 µs

en raison du théorème de Shannon, ce qui autorise une fréquence f inférieure ou égale à 5 kHz.

Du point de vue précision, pour avoir une bonne approximation de l'intégrale, il faut que $\Delta x/\Delta t$ soit le plus faible possible pour Δt_{min} avec :

$$\Delta x = x_{n+1} - x_n$$

Avec l'échelle que nous avons adoptée :

$$|\Delta x|_{min} = 1 \text{ LSB} = 2^{-15}$$

ce qui donne une variation dynamique de :

$$\frac{\Delta x}{\Delta t} = \frac{2^{-15}}{100\ \mu s} \neq 1/3 \ \text{V/s}$$

si x(t) est une tension.

Quant à la précision, en calcul numérique, on atteint :

$$2^{-15} \neq 10^{-4}/3$$

qui peut toujours être améliorée grâce au calcul en virgule flottante, et l'amélioration de l'algorithme d'approximation et qui entrainent automatiquement une augmentation du temps de calcul non négligeable.

En calcul analogique, avec un bon amplificateur, on atteint une précision de 10^{-4} avec une bande passante de 100 kHz.

Le tableau ci - après résume les caractéristiques de chaque type de calculateur.

	Précision	Fréquence	$Gain_{max}$
Calcul analogique	10^{-4}	100 kHz	1000
Calcul numérique	$\frac{1}{3}.10^{-4}$	5 kHz	$2^{15} - 1$

On remarque que le calculateur numérique apporte une faible amélioration de précision (facteur 3) mais on réduit d'un facteur 20 la fréquence de travail.

En revanche, il permet la résolution d'une intégration par rapport à un argument non temporel sans aucune difficulté, contrairement au calcul analogique où seul le temps peut servir d'argument.

CONCLUSION

Ce chapitre a permis de définir et d'évaluer une architecture multiprocesseur autour de l'ISBC 86^{MT}/12A, adaptée à un type particulier d'application (résolution de systèmes d'équations différentielles et aux dérivées partielles).

Ceci nous a conduit à présenter le calculateur ISBC 86^{MT}/12A dans une première partie. La deuxième partie a été consacrée à la définition d'une structure multiprocesseur et au choix d'un processeur adéquat.

L'étude des caractéristiques de la primitive d'intégration (temps de calcul, précision) nous permet d'affirmer que nous ne disposons pas sur cette base, d'un processeur élémentaire suffisamment performant, notamment sur le plan de la dynamique.

Toutefois, l'architecture proposée permet dans de bonnes conditions, d'organiser et gérer, et de répartir la charge de calcul sur un ensemble d'opérateurs numériques, que nous proposons d'étudier dans les prochains chapitres.

BIBLIOGRAPHIE DU CHAPITRE I

|1| The 8086 Family USER'S Manual
INTEL, Octobre 1978.

|2| MCS 86™ USER'S Manual
INTEL, Juillet 1978.

|3| ISBC™ Application Manual
INTEL.

|4| Robert GARROW, Jim JOHNSON and Les SOLTESZ
"16 bit Single-Board Computer maintains 8 bit Family ties"
Electronics, 12 Octobre 1978.

|5| El Miloud AMAMOU
"Multiprocesseurs numériques : application à la définition d'un opérateur numérique en calcul hybride"
Mémoire de D. E. A., 1980, Université des Sciences et Techniques de Lille I.

|6| "Application Note" AP-51
"Designing 8086, 8088, 8089 Multiprocessing"
Systems with the 8289 Bus Arbiter
INTEL, Mars 1979.

|7| "ISBC 86MT/12A : Single-Board Computer Hardware Reference Manual"
INTEL, Manual Order Number : 9803074-01.

|8| "Component DATA Catalog"
INTEL, 1980.

|9| M.J. FLYN
"Some Computer organization and their effectiveness"
IEEE Transaction on Computer, Septembre 1972.

|10| D.L. SLOTNICK
"Unconventional systems"
AFIPS Conference, Proceedings, SJCC 1967.

|11| R.G. HINTZ and D.P. TATE
"Control Data Star-100 Processor design"
IEEE Computer Society Internat' Conf., 1972, pp. 1-7.

|12| K.J. THURBER
"Large Scale Computer architecture : Parallel and associative processors"
Hayden, Rochelle Park, NJ 1976.

|13| J.C. MURTHA and R.L. BEADLES
"Survey of the highly Parallel information Processing systems"
Office of Naval Research, Repport n° 4755, 1964.

|14| L.C. HOBBS and D.J. THEIS
"Survey of parallel Processor approches and tachniques"
Parallel Processor systems, Technologies and applications, Spartan Book's, NY 1970.

|15| W. HANDLER
"The impact of classification shemes on Computer architecture"
Parallel Processing Symposium, Août 1977.

|16| W.A. WULF and C.G. BELL
"Cmmp : a multi mini Processors"
AFIPS Conference Proceedings, FJCC 1972.

|17| L.C. HIGBIE
"Super Computer Architecture"
Computer, Décembre 1973.

|18| J.E. SHORE
"Second Thoughts on parallel Processing"
Computing and Electrical Engineering, Vol. 1, Pergamon Press, Oxford, 1973.

|19| P.H. ENSLOW Jr
"Multiprocessors organizations : a survey"
Computer Survey, Vol. 9, n° 1, Mars 1977.

|20| S.H. FULLER and al
"Multi microprocessors : an overview and working examples"
Proceedings on the IEEE, Vol. 66, n° 2, Février 1978.

|21| G. PANIGRAPHI
"The implementation of Electronic Senal memories"
Computer, Juillet 1977.

|22| V. CORDONNIER
"L'emploi des mémoires circulantes dans l'architecture des ordinateurs"
Publication du Laboratoire de Calcul de Lille I, 1978.

|23| V. CORDONNIER
"La mémoire circulante de Maud"
Publication du Laboratoire de Calcul de Lille I, 1979.

|24| M.P. LECOUFFE
"MAUD : Une machine d'Assignation Unique Dynamique"
Publication du Laboratoire de Calcul, Lille, n° 116, septembre 78.

|25| W.W. PLUMER
"Asynchronous Arbiters"
IEEE Transaction on Computer, Janvier 1972.

|26| R. KOBER, C. KUZNIA
"SMS 201 : a powerful parallel processor with 128 microprocessors"
Euromicro Journal, n° 5, 1979.

CHAPITRE II

ANALYSE ET SYNTHESE DES OPERATEURS NUMERIQUES

ANALYSE ET SYNTHESE DES OPERATEURS NUMERIQUES

INTRODUCTION

Dans le calcul d'une intégrale, tel que décrit dans le chapitre précédent, apparaissent des opérations arithmétiques (addition, multiplication) et des opérations logiques (test de signe, test de débordement, décalage). Le traitement de ces opérations, nécessaires pour un pas d'intégration, sur un microprocesseur, le 8086 en l'occurrence, s'est avéré trop long.

Ceci résulte du fait que les opérations sont encore lentes (2,8 µs pour l'addition et 28,8 µs pour la multiplication) et se déroulent d'une façon séquentielle.

Il est légitime de penser que l'amélioration du temps de calcul peut se faire de deux façons au moins :

* utiliser des circuits logiques plus rapides et vraisemblablement à moins haut niveau d'intégration

* organiser ces circuits (interconnexion et gestion traitées dans le dernier chapitre), de manière à rendre le temps de calcul sur un pas, proche de celui de l'opérateur le plus lent.

Dans ce chapitre, nous traiterons le premier point, concernant la conception de dispositifs numériques rapides en trois paragraphes.

Le premier paragraphe constitue l'analyse et la synthèse des opérateurs primitifs (+, -, ×).

Dans le deuxième paragraphe, nous étudierons la réalisation matérielle de ces opérateurs, par deux techniques différentes :

* à base de portes logiques

* à base de réseaux logiques programmables.

Le troisième paragraphe contient une étude comparative des performances des deux techniques proposées.

II.1 - ANALYSE ET SYNTHESE DES OPERATEURS PRIMITIFS

II.1.1 - Définition

Dans une unité arithmétique et logique, on rencontre plusieurs types d'opérations. On appelle opérateur primitif, tout élément effectuant un type d'opération élémentaire de base. Il existe deux catégories d'opérateurs :

* Monadiques : il y a une seule information à traiter (un seul opérande) et une seule information résultante

* Dyadiques : il y a deux informations à traiter (deux opérandes) et une seule information résultante.
Exemples : ET, OU logiques, +, *, +, -, >, <, ...

II.1.2 - Représentation de l'information

Une information est un nombre variable, et sera représentée par des variables binaires sous forme de mot. Chaque varibale binaire portera l'indice représentant le rang dans le mot.

On écrira, par exemple :

le mot $X = x_3x_2x_1x_0$ $(x_0, x_1, x_2, x_3) \in \{0 , 1\}$

qui représente le nombre x :

$$x = x_3.2^3 + x_2.2^2 + x_1.2^1 + x_0.2^0$$

avec + : addition arithmétique
. : multiplication arithmétique.

X est le code binaire pur de x |1| |2|.

II.1.3 - Les opérateurs arithmétiques

Ils sont de type dyadiques, et au nombre de quatre :

* addition,
* soustraction,
* multiplication,
* division.

II.1.3.1 - Addition binaire

L'analyse classique d'un additionneur binaire consiste à considérer les bits et la retenue entrante de plus faible poids, comme des variables, et élaborer les bits somme et retenue sortante en fonction de ces variables.

Soient x_i et y_i, les bits de rang i des nombres X et Y à ajouter, et r_i la retenue entrante ; S_i, le digit somme élaboré au rang i, et r_{i+1} le report destiné au rang i+1.

Les quantités S_i et r_{i+1} sont toujours exprimées sous formes d'expressions booléennes en fonction de x_i, y_i et r_i à partir de la table de vérité, Fig. II.1 |1| |3| |4| |5|, et parfois traduites sous forme d'expressions arithmétiques en x_i, y_i, r_i |6| |7| |2|.

x_i	y_i	r_i	r_{i+1}	S_i
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Figure II.1

Plusieurs expressions de S_i et r_{i+1} peuvent être établies, en voici quelques unes :

* Pour S_i :

(2.1) $S_i = x_i \bar{y}_i \bar{r}_i + \bar{x}_i y_i \bar{r}_i + \bar{x}_i \bar{y}_i r_i + x_i y_i r_i$ (forme canonique)

(2.2) $S_i = (x_i \oplus y_i) \oplus r_i$ (ou exclusif à 2 entrées)

(2.3) $S_i = x_i \oplus y_i \oplus r_i$ (ou exclusif à 3 entrées)

* Pour r_{i+1} :

(2.4) $r_{i+1} = x_i y_i \bar{r}_i + x_i \bar{y}_i r_i + \bar{x}_i y_i r_i + x_i y_i r_i$

(2.5) $r_{i+1} = x_i y_i + y_i r_i + r_i x_i$

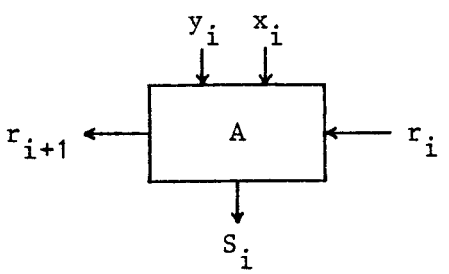
(2.6) $r_{i+1} = x_i y_i + (x_i + y_i) r_i$

(2.7) $r_{i+1} = x_i y_i \oplus (x_i \oplus y_i) r_i$

(2.8) $r_{i+1} = x_i y_i \oplus y_i r_i \oplus r_i x_i$

(2.7)' $r_{i+1} = \text{Maj}\ (x_i\ ,\ y_i\ ,\ r_i)$

Le dispositif permettant d'élaborer S_i et r_{i+1} en fonction des variables x_i, y_i et r_i est appelé additionneur complet ou étage d'additionneur (Fig. II.2).

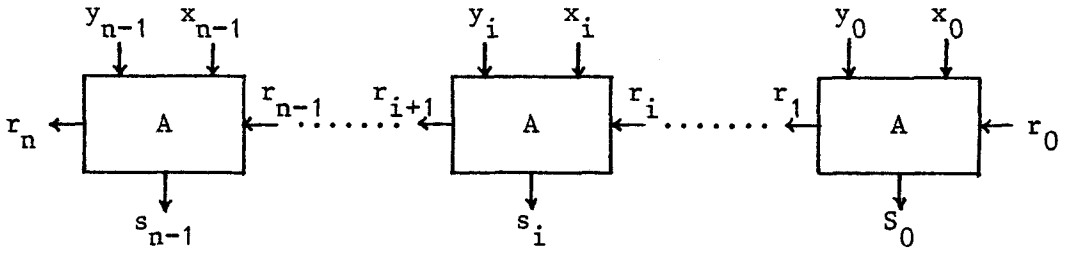


ETAGE D'ADDITIONNEUR

Figure II.2

Il peut être matérialisé à l'aide de quelques portes logiques, de plusieurs façons |8|.

Une liaison en cascade ou en ligne de plusieurs étages d'additionneurs (Fig. II.3), constitue un additionneur de nombres naturels dit en ligne ou en cascade |2|.



ADDITIONNEUR EN LIGNE

Figure II.3

Sa structure est simple, mais le temps de calcul peut devenir prohibitif pour des mots longs. En fait, si t_p est le temps de propagation d'un étage, et n le nombre d'étages, le temps de calcul de la somme est de :

$$n.t_p$$

Il convient donc de chercher d'autres formes d'additionneurs où les effets de propagation de la retenue soient réduits.

L'idée qui consiste à exprimer tous les digits S_i en fonction des $2(i+1)$ variables x_k, y_k ($k = 0, 1, \ldots, i$), de r_0 et de leurs compléments sous forme canonique (sommes de produits), peut effectivement conduire à la matérialisation d'additionneurs rapides, si on dispose de portes "ET" et "OU" logiques ayant un nombre d'entrées suffisant, pour élaborer tout S_i en deux couches logiques, si les compléments des x_i, y_i et r_0 sont disponibles.

En fait, pour élaborer S_i en deux couches logiques, il faut des portes "ET" de 4 à (3+i) entrées et une porte "OU" à $\sum_{j=0}^{i} 2^{j+2}$ entrées.

En effet, à partir des expressions canoniques de S_i, r_{i+1} et $\overline{r}_{i+1}$:

$$S_i = \overline{x}_i y_i \overline{r}_i + \overline{x}_i \overline{y}_i r_i + x_i \overline{y}_i \overline{r}_i + x_i y_i r_i \qquad (2.9)$$

$$r_{i+1} = x_i y_i + y_i r_i + r_i x_i \qquad (2.10)$$

$$\overline{r}_{i+1} = \overline{x}_i \overline{y}_i + \overline{y}_i \overline{r}_i + \overline{r}_i \overline{x}_i \qquad (2.11)$$

on en déduit que le nombre de monômes dans S_i est égal à deux fois le nombre de monômes dans r_i plus deux fois le nombre de monômes dans $\overline{r}_i$.

(2.10) et (2.11) montrent que r_i et $\overline{r}_i$ ont même nombre de monômes et par conséquent, on peut dire que dans S_i il y a 4 fois le nombre de monômes dans r_i.

Dans r_i, on montre par récurrence qu'il y a $\sum_{j=0}^{i} 2^j$ monômes.

Pour i=0, $r_0 = r_0$ un seul terme

et $\sum_{j=0}^{0} 2^j = 1$

Supposons la relation vraie pour r_i et soit $N_i = \sum_{j=0}^{i} 2^j$ monômes dans r_i.

Montrons que c'est vrai pour r_{i+1} :

$$r_{i+1} = x_i y_i + y_i r_i + r_i x_i = x_i y_i + (x_i + y_i) r_i$$

Par simple comptabilité, nous avons :

$$N_{i+1} = 1 + 2 N_i = 1 + 2 \sum_{j=0}^{i} 2^j = 1 + \sum_{j=0}^{i} 2^{j+1}$$

$$\boxed{N_{i+1} = \sum_{j=0}^{i+1} 2^j}$$ avec N_{i+1} : nombre de monômes dans r_{i+1}

Par le même raisonnement, on montre que le nombre de variables maximum d'un monôme dans S_i est bien (3+i).

Vérifions que c'est vrai pour i=0 :

$$S_0 = x_0 \bar{y}_0 \bar{r}_0 + \bar{x}_0 \bar{y}_0 r_0 + \bar{x}_0 y_0 \bar{r}_0 + x_0 y_0 r_0$$

on voit bien que tous les monômes contiennent 3 variables soit :

$$3 + 0 = 3.$$

Supposons que c'est vrai pour S_i et montrons que c'est vrai également pour S_{i+1} :

$$S_i = x_i \bar{y}_i \bar{r}_i + \bar{x}_i \bar{y}_i r_i + \bar{x}_i y_i \bar{r}_i + x_i y_i r_i$$

Compte tenu des expressions (2.10) et (2.11) de r_{i+1} et $\bar{r}_{i+1}$, on peut raisonner indifféremment par rapport à r_i ou $\bar{r}_i$ en ce qui concerne le nombre de variables constituant les termes produits dans S_i. Prenons par exemple le terme contenant r_i (c.à.d. $x_i y_i r_i$), il contient :

$$2 + (\text{nombre de variables dans } r_i) = 3 + i$$

et donc, le nombre de variables dans r_i est égal à $3 + i - 2 = i + 1$.

$$r_{i+1} = x_i y_i + x_i r_i + y_i r_i$$

montre que le nombre de variables maximum des monômes est égal à 1 + (celui dans r_i).

(2.9) écrite au rang (i+1) montre que ce nombre pour S_{i+1} est égal à 2 + (celui de r_{i+1}) = 2 + (1 + (celui dans r_i))

$$= 2 + 1 + i + 1$$
$$= (i + 1) + 3$$

En résumé, pour élaborer le bit s_i, il faut une porte OU à

$\sum_{j=0}^{i} 2^{j+2}$ entrées et $\sum_{j=0}^{i} 2^{j+2}$ portes "ET" de 4 à (3 + i) entrées.

A titre d'exemple :

pour i = 15 (mot de 16 bits)

il faut une porte OU à $\sum_{j=0}^{15} 2^{j+2} = 4 \sum_{j=0}^{15} 2^{j} = 4.(2^{16} - 1)$
$= 262\,140$ entrées

ce qui est énorme et rend inconcevable des additionneurs sur ce principe.

a) Réduction du nombre de monômes dans S_i :

Le principe consiste à mettre en évidence les termes classiques de propagation, génération et transmission |9| |10| |11| |12| |13| |14| |15| et d'exprimer S_i et r_{i+1} en fonction de ces termes.

Dans |9| |15|, plus particulièrement, l'auteur a défini une fonction de transfert T_i pour l'étage i, par la condition :

$$(2.12) \quad \begin{cases} T_i = 1 \iff (r_{i+1} = r_i) \\ T_i = r_{i+1}\, r_i + \overline{r}_{i+1}\, \overline{r}_i \\ T_i = (x_i + y_i)\, r_i + (\overline{x}_i + \overline{y}_i)\, \overline{r}_i \end{cases}$$

Les termes $T_i^0 = \overline{x}_i + \overline{y}_i$

$T_i^1 = x_i + y_i$

caractérisent respectivement les conditions de transfert d'une retenue entrante égale à zéro ou à un.

$\overline{T}_i^0 = x_i\, y_i = G_i^1$ caractérise la condition de non-transfert d'une retenue entrante égale à "0". C'est le terme "génération de un".

$\overline{T}_i^1 = \overline{x}_i\, \overline{y}_i = G_i^0$ caractérise la condition de non-transfert d'une retenue égale à un. C'est le terme "génération de zéro".

Les termes G_i^ℓ, T_i^k (ℓ, $k \in \{0 , 1\}$) qui sont des fonctions des variables x_i, y_i ne sont pas indépendants et vérifient certaines relations |15|, en particulier :

$\overline{G}_i^0 . T_i^0 = \overline{G}_i^1 . T_i^1 = x_i \oplus y_i$ qui définit d'ailleurs le terme

"propagation" |15| |16| |17| |18|.

Pour des raisons de compréhension, nous partons des expressions booléennes exprimant r_{i+1} et S_i en fonction des variables x_i, y_i et r_i, soient :

$$(2.13) \quad \begin{cases} S_i = x_i \oplus y_i \oplus r_i \\ S_i = (x_i \oplus y_i) \oplus r_i \end{cases}$$

$$(2.14) \quad \begin{cases} r_{i+1} = x_i y_i + (x_i \oplus y_i) r_i \\ r_{i+1} = x_i y_i + (x_i + y_i) r_i \\ r_{i+1} = x_i y_i \oplus (x_i \oplus y_i) r_i \end{cases}$$

Quant aux termes de transmission, propagation et génération, nous adoptons :

$$(2.15) \quad \begin{cases} T_i = x_i + y_i & \text{comme transmission} \\ G_i = x_i y_i & \text{comme génération} \\ P_i = x_i \oplus y_i & \text{comme propagation} \end{cases}$$

Par substitution de (2.15) dans (2.13) et (2.14), il vient :

$$(2.16) \quad S_i = P_i \oplus r_i$$

$$(2.17) \quad r_{i+1} = G_i + P_i r_i$$

$$(2.18) \quad r_{i+1} = G_i \oplus P_i r_i$$

$$(2.19) \quad r_{i+1} = G_i + T_i r_i$$

Considérons la relation (2.17). Un calcul explicite de r_{i+1} en fonction des variables G_k, P_k ($k = 0, 1, \ldots, i$) et de r_0 donne :

$$(2.20) \quad r_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + \ldots + P_i P_{i-1} \ldots P_j \ldots P_1 G_0 + P_i P_{i-1} \ldots P_j \ldots P_0 r_0$$

Posons :

$$(2.21)\qquad G_{0,i}^{P} = G_i + \sum_{j=0}^{i-1} \left(\prod_{k=j+1}^{i} P_k \right) G_j = \sum_{j=0}^{i} \left(\prod_{k=j+1}^{i} P_k \right) G_j$$

avec $\prod_{k=i+1}^{i} P_k = 1$

$$(2.22)\qquad P_{0,i} = \prod_{k=0}^{i} P_k$$

(2.20) devient :

$$(2.23)\qquad \boxed{r_{i+1} = G_{0,i}^{P} + P_{0,i}\, r_0}$$

et

$$(2.24)\qquad \overline{r}_{i+1} = \overline{G_{0,i}^{P} + P_{0,i}\, r_0} = \overline{G}_{0,i}^{P}\, \overline{P}_{0,i} + \overline{G}_{0,i}^{P}\, \overline{r}_0$$

$$\overline{G}_{0,i}^{P} = \overline{\sum_{j=0}^{i} \left(\prod_{k=j+1}^{i} P_k \right) G_j}$$

$$= \overline{G}_i \prod_{j=0}^{i-1} \overline{\left(\prod_{k=j+1}^{i} P_k \right) G_j}$$

$$= \overline{G}_i \prod_{j=0}^{i-1} \left(\overline{\prod_{k=j+1}^{i} P_k} + \overline{G}_j \right)$$

Posons $\prod_{k=j}^{i} P_k = P_{j,i}$

$$\Longrightarrow \quad \overline{G}_{0,i}^{P} = \overline{G}_i \prod_{j=0}^{i-1} (\overline{P}_{j+1,i} + \overline{G}_j)$$

avec un calcul explicite de $\overline{G}_{0,i}^{P}$ et des simplifications successives souvent de type :

$$\overline{P}_{k,i} \cdot \overline{P}_{k+1,i} = \overline{P}_{k+1,i}$$

$$\overline{P}_{k+1,i} + \overline{P}_{k+1,i} \cdot \overline{G}_{k-1} = \overline{P}_{k+1,i}$$

$$\overline{P}_{k+1,i} + \overline{P}_{k,i} \cdot \overline{G}_k = \overline{P}_{k+1,i} + \overline{P}_k \cdot \overline{G}_k$$

on aboutit au résultat :

$$(2.25) \qquad \overline{G}_{0,i}^{P} = \sum_{j=1}^{i} \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j + \prod_{j=0}^{i} (\overline{G}_j)$$

$$\overline{G}_{0,i}^{P} . \overline{P}_{0,i} = \left(\sum_{j=1}^{i} \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j + \prod_{j=0}^{i} \overline{G}_j \right) . \overline{P}_{0,i}$$

$$= \overline{P}_{0,i} \left(\sum_{j=1}^{i} \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j \right) + \overline{P}_{0,i} \left(\prod_{j=0}^{i} \overline{G}_j \right)$$

$$= \sum_{j=1}^{i} \left(\left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j . \overline{P}_{0,i} \right) + \overline{P}_{0,i} \prod_{j=0}^{i} \overline{G}_j$$

$$\overline{P}_j . \overline{P}_{0,i} = \overline{P}_j$$

$$\overline{P}_{0,i} \prod_{j=0}^{i} \overline{G}_j = \overline{P}_0 \prod_{j=0}^{i} \overline{G}_j + \overline{P}_{1,i} \prod_{j=0}^{i} \overline{G}_j$$

$$\overline{G}_{0,i}^{P} . \overline{P}_{0,i} = \sum_{j=1}^{i} \left(\prod_{k=j}^{i} \overline{G}_k . \overline{P}_j \right) + \overline{P}_{1,i} \prod_{j=0}^{i} \overline{G}_j + \overline{P}_0 \prod_{j=0}^{i} \overline{G}_j$$

$$\overline{P}_{1,i} \prod_{j=0}^{i} \overline{G}_j = \left(\sum_{j=1}^{i} \overline{P}_j \right) . \prod_{k=0}^{i} \overline{G}_k = \sum_{j=1}^{i} \left(\left(\prod_{k=0}^{i} \overline{G}_k \right) \overline{P}_j \right)$$

$$\overline{G}_{0,i}^{P} . \overline{P}_{0,i} = \sum_{j=1}^{i} \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j + \sum_{j=1}^{i} \left(\prod_{k=0}^{i} \overline{G}_k \overline{P}_j \right) + \overline{P}_0 \prod_{j=0}^{i} \overline{G}_j$$

$$= \sum_{j=1}^{i} \left(\left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j + \left(\prod_{k=0}^{i} \overline{G}_k \right) \overline{P}_j \right) + \overline{P}_0 \prod_{j=0}^{i} \overline{G}_j$$

$$\prod_{k=0}^{i} \overline{G}_k = \prod_{\ell=0}^{j-1} \overline{G}_\ell . \prod_{k=j}^{i} \overline{G}_k$$

et $$\left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j + \left(\prod_{k=0}^{i} \overline{G}_k \right) \overline{P}_j = \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j \left(1 + \prod_{\ell=0}^{j-1} \overline{G}_\ell \right)$$

$$= \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j$$

et finalement :

$$(2.26) \qquad \overline{G}^{P}_{0,i} \cdot \overline{P}_{0,i} = \sum_{j=1}^{i} \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j + \overline{P}_0 \prod_{j=0}^{i} \overline{G}_j$$

Posons :

$$\alpha_p = \sum_{j=1}^{i} \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j$$

et

$$\beta_p = \prod_{j=0}^{i} \overline{G}_j$$

Par substitution de α_p et β_p dans (2.25) et (2.26), nous avons :

$$(2.27) \qquad \begin{cases} \overline{G}^{P}_{0,i} = \alpha_p + \beta_p \\ \overline{G}^{P}_{0,i} \cdot \overline{P}_{0,i} = \alpha_p + \overline{P}_0 \, \beta_p \end{cases}$$

(2.27) dans (2.24) donne :

$$\begin{aligned} \overline{r}_{i+1} &= (\alpha_p + \overline{P}_0 \, \beta_p) + (\alpha_p + \beta_p)\overline{r}_0 \\ &= \alpha_p + \overline{P}_0 \, \beta_p + \alpha_p \overline{r}_0 + \beta_p \overline{r}_0 \\ &= \alpha_p \, (1 + \overline{r}_0) + \overline{P}_0 \beta_p + \beta_p \overline{r}_0 \\ &= \alpha_p + \overline{P}_0 \beta_p + \beta_p \overline{r}_0 \end{aligned}$$

$$\alpha_p + \overline{P}_0 \beta_p = \sum_{j=1}^{i} \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j + \overline{P}_0 \prod_{j=0}^{i} \overline{G}_j = \sum_{j=0}^{i} \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j$$

$$\Longrightarrow \quad (2.28) \qquad \boxed{\overline{r}_{i+1} = \sum_{j=0}^{i} \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{P}_j + \overline{r}_0 \left(\prod_{j=0}^{i} \overline{G}_j \right)}$$

$$(2.29) \qquad \boxed{r_{i+1} = \sum_{j=0}^{i} \left(\prod_{k=j+1}^{i} P_k \right) G_j + r_0 \prod_{k=0}^{i} P_k}$$

Le même calcul de r_{i+1} et $\bar{r}_{i+1}$ peut être refait à partir de $r_{i+1} = G_i + T_i r_i$ et on retrouvera des relations analogues à (2.28) et (2.29) par simple substitution de T à P :

$$\begin{cases} r_{i+1} = G_{0,i}{}^{T} + T_{0,i} r_0 \\ \bar{r}_{i+1} = \bar{G}_{0,i}{}^{T} . \bar{T}_{0,i} + \bar{G}_{0,i}{}^{T} . \bar{r}_0 \end{cases}$$

(2.30) $$\boxed{r_{i+1} = \sum_{j=0}^{i} \left(\prod_{k=j+1}^{i} T_k \right) G_j + r_0 \prod_{k=0}^{i} T_k}$$

(2.31) $$\boxed{\bar{r}_{i+1} = \sum_{j=0}^{i} \left(\prod_{k=j}^{i} \bar{G}_k \right) \bar{T}_j + \bar{r}_0 \prod_{j=0}^{i} \bar{G}_j}$$

Nous venons de définir deux types de relations de r_{i+1} et $\bar{r}_{i+1}$ selon qu'on prend la paire de variables (G_k , P_k) ou (G_k , T_k) que nous écrirons sous la forme :

(2.32) $$\begin{cases} r_{i+1} = G_{0,i}{}^{P} + P_{0,i} r_0 \\ \bar{r}_{i+1} = \gamma_{0,i}{}^{P} + \delta_{0,i}{}^{P} \bar{r}_0 \end{cases} \qquad \text{paire } (G_k , P_k)$$

(2.33) $$\begin{cases} r_{i+1} = G_{0,i}{}^{T} + T_{0,i} r_0 \\ \bar{r}_{i+1} = \gamma_{0,i}{}^{T} + \delta_{0,i}{}^{T} \bar{r}_0 \end{cases} \qquad \text{paire } (G_k , T_k)$$

avec :

$$G_{0,i}{}^{P} = \sum_{j=0}^{i} \left(\prod_{k=j+1}^{i} P_k \right) G_j$$

$$P_{0,i} = \prod_{k=0}^{i} P_k$$

$$\gamma_{0,i}{}^{P} = \sum_{j=0}^{i} \left(\prod_{k=j}^{i} \bar{G}_k \right) \bar{P}_j$$

$$\delta_{0,i}{}^{P} = \prod_{j=0}^{i} \bar{G}_j$$

$$G_{0,i}^{T} = \sum_{j=0}^{i} \left(\prod_{k=j+1}^{i} T_k \right) G_j$$

$$T_{0,i} = \prod_{k=0}^{i} T_k$$

$$\gamma_{0,i}^{T} = \sum_{j=0}^{i} \left(\prod_{k=j}^{i} \overline{G}_k \right) \overline{T}_j$$

$$\delta_{0,i}^{T} = \prod_{j=0}^{i} \overline{G}_j$$

A partir des expressions explicites de r_i et $\overline{r}_i$, on peut exprimer S_i en fonction de P_k, G_k, $\overline{P}_k$, $\overline{G}_k$ ($k = 0, 1, \ldots, i$), de r_0 et de $\overline{r}_0$.

On sait que : $S_i = P_i \oplus r_i = \overline{P}_i r_i + \overline{r}_i P_i$

(2.34)

$$\boxed{\begin{aligned} S_i = \overline{P}_i . \Big(\sum_{j=0}^{i-1} \Big(\prod_{k=j+1}^{i-1} P_k \Big) G_j + r_0 \prod_{k=0}^{i-1} P_k \Big) \\ + P_i . \Big(\sum_{j=0}^{i-1} \Big(\prod_{k=j}^{i-1} \overline{G}_k \Big) \overline{P}_j + \overline{r}_0 \prod_{j=0}^{i-1} \overline{G}_j \Big) \end{aligned}}$$

On remarque bien une réduction considérable du nombre de monômes dans S_i, qui est en fait, égal à $2(i+1)$.

Pour $i = 15$ par exemple, il faut une porte "OU" à 32 entrées au lieu de 262 140 dans le cas précédent.

Faut-il encore dans ce cas :

* accepter d'augmenter le nombre de couches logiques de 2 pour le calcul de G_i et P_i

* un nombre de portes "ET" de $2(i+1)$ pour chaque digit S_i

* disposer de portes "ET" à $i+2$ entrées

* disposer de portes "OU" à $2(i+1)$ entrées.

Nous reviendrons sur toutes ces questions dans le paragraphe concernant la matérialisation des opérateurs.

II.1.3.2 - La soustraction binaire

Faire la différence X - Y des deux nombres binaires X et Y consiste à trouver un mot binaire D, s'il existe, tel que :

$$D = X - Y \iff X = Y + D$$

D'après les règles de l'addition définies précedemment (2.3), (2.7), (2.7)', nous avons :

$$\begin{cases} x_i = y_i \oplus d_i \oplus R_i & \text{avec } R_0 = 0 \quad (2.35) \\ R_{i+1} = \text{Maj}\,(y_i\,,\ d_i\,,\ R_i) \quad (2.36) \\ \qquad = y_i\ d_i \oplus (y_i \oplus d_i)\ R_i \end{cases}$$

$$(2.35) \Longrightarrow (2.37) \qquad d_i = x_i \oplus y_i \oplus R_i$$

(2.37) dans (2.36) donne :

$$R_{i+1} = \text{Maj}\,(y_i\,,\ x_i \oplus y_i \oplus R_i\,,\ R_i)$$

$$\begin{aligned} R_{i+1} &= y_i\,(x_i \oplus y_i \oplus R_i) \oplus (y_i \oplus x_i \oplus y_i \oplus R_i)\ R_i \\ &= (y_i\ x_i \oplus y_i \oplus y_i\ R_i) \oplus (y_i \oplus y_i \oplus x_i \oplus R_i)\ R_i \\ &= (x_i \oplus 1)\ y_i \oplus y_i\ R_i \oplus y_i\ R_i \oplus x_i\ R_i \oplus R_i \\ &= (x_i \oplus 1)\ y_i \oplus y_i\ R_i \oplus x_i\ R_i \oplus R_i \\ &= \bar{x}_i\ y_i \oplus (y_i \oplus x_i \oplus 1)\ R_i \\ &= \bar{x}_i\ y_i \oplus (y_i \oplus x_i)\ R_i \\ &= \text{Maj}\,(\bar{x}_i\,,\ y_i\,,\ R_i) \end{aligned}$$

D'où les lois de la soustraction binaire :

$$\begin{cases} d_i = x_i \oplus y_i \oplus R_i & (2.38) \\ R_{i+1} = \text{Maj}\,(\bar{x}_i\,,\ y_i\,,\ R_i) & (2.39) \end{cases} \qquad (R_0 = 0)$$

Jusqu'ici, nous n'avons considéré que des nombres naturels (sans signe). L'introduction de la soustraction ne se fait pas sans conséquence. En effet, dans l'opération X - Y, on sait sue si X < Y, le résultat sera négatif, il convient de les distinguer des nombres positifs.

a) Représentation des nombres négatifs :

Le nombre binaire

$$X = x_{n-1}\ x_{n-2} \cdots x_i \cdots x_0$$

est aussi égal à :

$$X = 0\ x_{n-1} \cdots x_i \cdots x_0$$
$$= 0\ 0\ x_{n-1} \cdots x_i \cdots x_0$$

On peut ajouter autant de zéros à gauche de x_{n-1} que l'on veut.

Soient 2 nombres binaires X et Y :

$$\begin{cases} X = 0 \ldots 0\ x_{n-1}\ x_{n-2} \cdots x_i \cdots x_0 \\ Y = 0 \ldots 0\ y_{n-1}\ y_{n-2} \cdots y_i \cdots y_0 \end{cases}$$

D'après les lois de la soustraction établies précédemment, on sait calculer D = X - Y.

Au rang n, nous avons :

$$\begin{cases} d_n = x_n \oplus y_n \oplus R_n \\ R_{n+1} = \text{Maj}\ (\overline{x}_n\ ,\ y_n\ ,\ R_n) \end{cases}$$

Or, ici $x_n = y_n = 0$

$$d_n = 0 \oplus 0 \oplus R_n = R_n$$
$$R_{n+1} = \text{Maj}\ (1\ ,\ 0\ ,\ R_n) = R_n$$

et $$D = R_n \ldots R_n\ R_n\ d_{n-1}\ d_{n-2} \cdots d_i \cdots d_0$$

R_n peut être égale à 0 ou 1 et D prend alors les formes suivantes :

(2.40) $\quad D = 0 \ldots 0\ d_{n-1} \ldots d_i \ldots d_0$

(2.41) $\quad D = 1 \ldots 1\ d_{n-1} \ldots d_i \ldots d_0$

Une simple vérification sur des nombres quelconques X et Y montre que la première forme (2.40) correspond aux cas $X \geq Y$, et la forme (2.41) aux cas $X < Y$, et d'ailleurs, la deuxième forme ne correspond pas à un nombre naturel fini puisqu'elle se termine par une infinité de 1 à gauche de d_{n-1}. Avec ces considérations, et ce qu'on sait déjà sur les nombres binaires négatifs, tous les nombres seront désormais considérés comme signés et représentés par :

$$X = a^* x_{n-1} \ldots x_i \ldots x_0 \qquad \text{où} \qquad a\ ,\ x_i \in \{0\ ,\ 1\}$$

avec $\quad a = 0 \quad$ pour $\quad X \geq 0$

$\quad\quad a = 1 \quad$ pour $\quad X < 0$

et $\quad a^* = a \ldots a \quad$ (suite de a)

Cette représentation est appelée représentation dyadique |9|. Elle permet de faire la théorie de diverses opérations plus facilement que celle utilisant des suites tronquées. Le passage aux suites tronquées, pour la réalisation matérielle, peut se faire en fin d'étude.

D'après ce que nous venons de voir, on peut définir l'opposé de X (c'est à dire - X) par le calcul de :

$$-X = 0 - X$$

(- X est aussi appelé le complément vrai de X ou encore le complément algébrique de X et sera noté $\hat{X}$ |2|.

$$\hat{X} = 0 - X = 0^* - a^* x_{n-1} \ldots x_i \ldots x_0$$

avec $\quad a\ ,\ x_j\ (j = 0,\ 1,\ \ldots,\ n-1) \in \{0\ ,\ 1\}$

En appliquant les relations de soustraction, nous avons :

$$(2.42)\qquad \hat{x}_i = 0 \oplus x_i \oplus R_i = x_i \oplus R_i = \overline{x}_i\, R_i + x_i\, \overline{R}_i \qquad (R_0 = 0)$$
$$R_{i+1} = \mathrm{Maj}\,(\overline{0}\ ,\ x_i\ ,\ R_i) = \mathrm{Maj}\,(1\ ,\ x_i\ ,\ R_i)$$

$$(2.43)\qquad R_{i+1} = x_i + R_i \Longrightarrow R_{i+1} = \sum_{j=0}^{i} x_j$$

$$(2.42) \Longrightarrow \begin{cases} \hat{x}_i = \overline{x}_i & \text{si} \quad R_i = 1 \\ \hat{x}_i = x_i & \text{si} \quad R_i = 0 \end{cases}$$

et (2.43) $\Longrightarrow$ $R_i = 1$ si un x_j de rang inférieur ($j < i$) est égal à 1 d'où la règle :

Pour passer de X à - X ($X = a^* x_{n-1} \ldots x_i \ldots x_0$), il suffit de scruter tous les x_i de la droite vers la gauche à partir de $i = 0$ et de complémenter (complément booléen) tous les x_j se trouvant à gauche du premier 1 rencontré.

Les relations (2.42) et (2.43) peuvent aussi s'écrire :

$$\hat{x}_i = x_i \oplus R_i = \overline{x}_i \oplus \overline{R}_i \qquad (R_0 = 0)$$
$$R_i = \sum_{j=0}^{i-1} x_j \Longrightarrow \overline{R}_i = \prod_{j=0}^{i-1} (\overline{x}_j)$$

$$\Longrightarrow \hat{x}_i = \overline{x}_i \oplus \prod_{j=0}^{i-1} (\overline{x}_j)$$

Relation de comptage |2| ou encore de l'opération $\overline{X} + 1$, où $\overline{X}$ est le complément booléen de X.

D'où <u>la règle classique</u> :

Pour obtenir le complément vrai de X, complémenter X bit par bit et ajouter une unité de plus faible poids.

Les opérations d'addition, de soustraction et de complémentation sur des mots de longueur finies, conduit à définir la longueur du mot résultant par rapport à celle des mots traités, et par la même, la position du premier digit signe dans une représentation dyadique. Ceci nécessite l'étude de ces opérations sur des nombres signés, et la détermination des lois de validité du résultat.

II.1.3.3 - Traitement des nombres signés

a) Complémentation algébrique :

Soit un nombre X signé et représenté par :

$$X = a^* \; x_{n-1} \cdots x_i \cdots x_0$$

Le complément vrai de X est défini par :

$$(2.44) \quad \begin{cases} \hat{x}_i = x_i \oplus r_i \\ r_{i+1} = x_i + r_i \end{cases} \qquad (r_0 = 0)$$

Le problème de validité du résultat (débordement) se pose pour les derniers bits. Il s'agit de savoir à partir de quelle position on voit apparaître la séquence de 0 ou de 1 qui, à priori, est représentative du signe.

En appliquant (2.44) pour $i \geq n$, on aboutit au résultat suivant :

$$\hat{X} = (\overline{a}\, r_n)^* \; (a \oplus r_n) \; \hat{x}_{n-1} \cdots \hat{x}_i \cdots \hat{x}_0 \qquad |2|$$

Ceci montre que le signe de $\hat{X}$ ne commence pas obligatoirement au rang n comme pour X, mais sûrement au rang (n + 1).

Dans le cas courant, et c'est le cas que nous considérons, où $\hat{X}$ est défini sur la même longueur que X, on dira qu'il y a débordement si :

$$\overline{a}\, r_n \neq a \oplus r_n$$

d'où la condition :

$$\text{deb } \hat{X} = \overline{a}\, r_n \oplus (a \oplus r_n) = (\overline{a}\, r_n \oplus a) \oplus r_n$$

$$\text{deb } \hat{X} = (\overline{\overline{a}\, r_n}\; a + \overline{a}\, r_n\, \overline{a}) \oplus r_n = (a + r_n) \oplus r_n$$

(2.45) $\boxed{\text{deb } \hat{X} = a\ \overline{r}_n}$ où deb $\hat{X}$ est une variable qui signale un débordement lorsqu'elle vaut 1.

et

(2.46) $\boxed{\text{Sign } \hat{X} = \overline{a}\ r_n}$ Sign $\hat{X}$ = Signe de $\hat{X}$

= 0 si $\hat{X}$ positif

= 1 si $\hat{X}$ négatif

b) Addition :

Les règles d'addition de deux nombres signés X et Y sont les mêmes que celles des nombres naturels. Il convient seulement de déterminer les expressions de deb (X + Y) et Sign (X + Y) où :

$$\begin{cases} X = a^* \ x_{n-1} \cdots x_i \cdots x_0 \\ Y = b^* \ y_{n-1} \cdots y_i \cdots y_0 \end{cases}$$

En ne s'intéressant qu'aux rangs supérieurs ou égaux à n, en appliquant les relations (2.3) et (2.7)', nous avons |2| :

$$S = X + Y = \left[\text{Maj } (a\ ,\ b\ ,\ \overline{r}_n)\right]^* \ (a \oplus b \oplus r_n)\ S_{n-1} \cdots S_i \cdots S_0$$

soit :

$$S = S^*_{n+1}\ S_n\ S_{n-1} \cdots S_i \cdots S_0$$

avec :

$$\begin{cases} S_{n+1} = \text{Maj } (a\ ,\ b\ ,\ \overline{r}_n) \\ S_n = a \oplus b \oplus r_n \end{cases}$$

Là encore, on remarque que le signe est donné par S_{n+1} et le débordement par $S_n \neq S_{n+1}$. D'où les expressions :

(2.47) $\boxed{\text{Sign } (X + Y) = \text{Maj } (a\ ,\ b\ ,\ \overline{r}_n)}$

(2.48) $\boxed{\text{deb } (X + Y) = \overline{r}_n\ a\ b + r_n\ \overline{a}\ \overline{b}}$

c) Soustraction :

De la même façon, nous avons pour D = X - Y :

$$D = d^{*}_{n+1}\ d_n\ d_{n-1} \cdots d_i \cdots d_0$$

avec :

$$\begin{cases} d_{n+1} = \text{Maj}\ (a\ ,\ \overline{b}\ ,\ R_n) \\ d_n = a \oplus b \oplus R_n \end{cases}$$

(2.49) $$\boxed{\text{Sign}\ (X - Y) = d_{n+1} = \text{Maj}\ (a\ ,\ \overline{b}\ ,\ R_n)}$$

et

(2.50) $$\boxed{\text{deb}\ (X - Y) = d_{n+1} \oplus d_n = \overline{a}\ b\ \overline{R}_n + a\ \overline{b}\ R_n}$$

d) La multiplication binaire :

Soient deux nombres signés x et y, représentés respectivement par les mots binaires X et Y :

$$\begin{cases} X = a^{*}\ x_{n-1} \cdots x_i \cdots x_0 \\ Y = b^{*}\ y_{n-1} \cdots y_i \cdots y_0 \end{cases}$$

a = signe de X

b = signe de Y

$$(a,\ b,\ x_j,\ y_j) \in \{0\ ,\ 1\} \qquad j = (0,\ 1,\ \ldots,\ n-1)$$

avec

$$\begin{cases} x = -\ a\ 2^n + \sum_{i=0}^{n-1} x_i\ 2^i \\ y = -\ b\ 2^n + \sum_{i=0}^{n-1} y_i\ 2^i \end{cases}$$

Le produit X.Y des mots binaires X et Y sera, par définition : trouver un mot binaire P, représentatif du nombre p, tel que : p = x.y.

En remplaçant x et y par leur valeur dans p, il vient :

$$p = (-a\,2^n + \sum_{i=0}^{n-1} x_i\,2^i)\,(-b\,2^n + \sum_{i=0}^{n-1} y_i\,2^i)$$

$$p = a\,b\,2^{2n} - a\,2^n \sum_{i=0}^{n-1} y_i\,2^i - b\,2^n \sum_{i=0}^{n-1} x_i\,2^i + (\sum_{i=0}^{n-1} x_i\,2^i)(\sum_{i=0}^{n-1} y_i\,2^i)$$

$$p = a\,b\,2^{2n} - a\,2^n \sum_{i=0}^{n-1} y_i\,2^i - b\,2^n \sum_{i=0}^{n-1} x_i\,2^i + \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} y_i\,x_j\,2^{i+j}$$

$$p = a\,b\,2^{2n} - (\sum_{i=0}^{n-1} a\,y_i\,2^{i+n} + \sum_{i=0}^{n-1} b\,x_i\,2^{i+n}) + \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} y_i\,x_j\,2^{i+j}$$

Par passage en représentation binaire et disposition des termes de même poids sur la même colonne, nous avons (Fig. II.4) pour n = 4 :

$$X = a\,x_3\,x_2\,x_1\,x_0 \qquad \text{et} \qquad Y = b\,y_3\,y_2\,y_1\,y_0$$

Figure II.4

						y_0x_3	y_0x_2	y_0x_1	y_0x_0	$+ \sum_{j=0}^{4-1} x_j\,y_0\,2^j$
					y_1x_3	y_1x_2	y_1x_1	y_1x_0	0	$+ \sum_{j=0}^{4-1} x_j\,y_1\,2^{j+1}$
				y_2x_3	y_2x_2	y_2x_1	y_2x_0	0	0	$+ \sum_{j=0}^{4-1} x_j\,y_2\,2^{j+2}$
			y_3x_3	y_3x_2	y_3x_1	y_3x_0	0	0	0	$+ \sum_{j=0}^{4-1} x_j\,y_3\,2^{j+3}$
0	0	$-ay_3$	$-ay_2$	$-ay_1$	$-ay_0$	0	0	0	0	$+ -\sum_{i=0}^{4-1} a\,y_i\,2^{i+4}$
0	0	$-bx_3$	$-bx_2$	$-bx_1$	$-bx_0$	0	0	0	0	$+ -\sum_{i=0}^{4-1} b\,x_i\,2^{i+4}$
	ab	0	0	0	0	0	0	0	0	$+ a\,b\,2^8$
$P = P_9$	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	$= p$

$$P = -P_9\,2^9 + \sum_{i=0}^{8} P_i\,2^i$$

La figure II.4 montre que le produit P est obtenu par addition des produits partiels en tenant compte des signes, en particulier des $a\,y_i$ et $b\,x_i$ ($i = 0, 1, \ldots, n-1$). Ceci peut être gênant lors de la conception d'un multiplieur, où la régularité du processus de multiplication est toujours recherchée. Il est donc intéressant de transformer la figure II.4 de façon à réduire le nombre de soustractions ou de les supprimer totalement.

- *Transformation* |19| :

En vertu de $-X = \overline{X} + 1$, la soustraction de :

$$(2.51) \qquad 2^n \left(-0\,2^{n+1} + 0\,2^n + \sum_{i=0}^{n-1} a\,y_i\,2^i\right)$$

peut être remplacée par l'addition de :

$$(2.52) \qquad 2^n \left(-1\,2^{n+1} + 1\,2^n + 1 + \sum_{i=0}^{n-1} \overline{a\,y_i}\,2^i\right.$$

(2.52) prend les valeurs :

$$(2.53) \quad \begin{cases} 0 & \text{pour} \quad a = 0 \\ 2^n \left(-2^{n+1} + 2^n + 1 + \sum_{i=0}^{n-1} \overline{y}_i\,2^i\right) & \text{pour} \quad a = 1 \end{cases}$$

Ainsi, (2.52) peut être écrite comme :

$$(2.54) \qquad 2^n \left(-2^{n+1} + 2^n + \overline{a}\,2^n + a + \sum_{i=0}^{n-1} a\,\overline{y}_i\,2^i\right)$$

D'une façon similaire, la soustraction de :

$$2^n \left(-0\,2^n + \sum_{i=0}^{n-1} b\,x_i\,2^i\right)$$

peut être remplacée par l'addition de :

$$(2.55) \qquad 2^n \left(-2^{n+1} + 2^n + \overline{b}\,2^n + b + \sum_{i=0}^{n-1} b\,\overline{x}_i\,2^i\right).$$

Et la figure II.4 se trouve ainsi transformée en la figure II.5 :

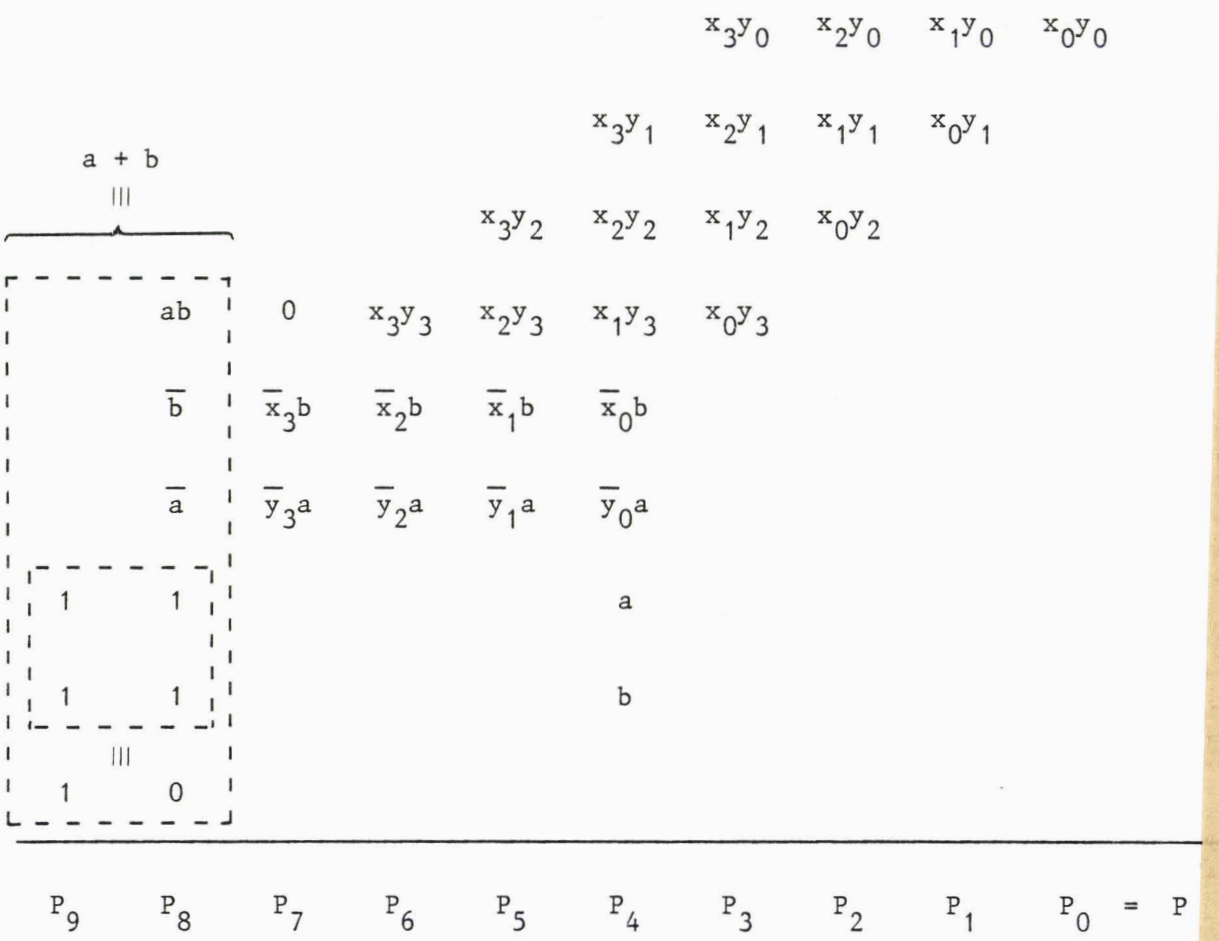


Figure II.5

La figure II.5 met en évidence la régularité du processus de multiplication en deux points :

- tous les bits des produits partiels peuvent être généré par des portes "ET" uniquement,

- chaque bit du produit partiel a un coefficient positif.

Ce qui permet, dans un réseau de multiplication cellulaire, de n'utiliser que les fonctions "ET" et ADD (addition).

- *Transformation :*

En raison de la commutativité de l'addition et de la soustraction, la figure II.4 peut s'écrire :

					$-ay_0$	x_3y_0	x_2y_0	x_1y_0	x_0y_0	
				$-ay_1$	x_3y_1	x_2y_1	x_1y_1	x_0y_1	0	
			$-ay_2$	x_3y_2	x_2y_2	x_1y_2	x_0y_2	0	0	
		$-ay_3$	x_3y_3	x_2y_3	x_1y_3	x_0y_3	0	0	0	
(-a)	(-b)	$-bx_3$	$-bx_2$	$-bx_1$	$-bx_0$	0	0	0	0	
P_9	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	= P

Figure II.6

Cette forme n'est autre que celle obtenue par le procédé manuel de multiplication de deux nombres en représentation tronquée et où les bits signés sont affectés du signe moins.

Les opérations de soustraction peuvent être réduites à des opérations d'addition en utilisant pour les X décalés, des signes étirés suffisamment loin, et en gardant Y sous forme compacte, avec des poids positifs avant la position de signe (b) et négatif pour celle-ci. En fait, le signe de X est étiré jusqu'à la position 2(n+1), si n est la longueur de X signe non compris. En effet :

- lors de l'addition de deux termes à signes dédoublés et traités comme des bits ordinaires, on sait qu'on obtient toujours la somme signe compris, même s'il y a débordement,

- dans la multiplication de deux nombres avec signe $|2|$, de longueurs (n+1) signes compris, le résultat du produit a toujours un signe dédoublé, s'il est exprimé sur 2(n+1) bits, sauf dans le cas $(-2^n)\times(-2^n)$, qui pourrait être considéré comme un cas de dépassement de

capacité et donc à détecter et qui permettrait l'arrêt systématique des produits après 2n+1 bits.

Cette propriété de la représentation dyadique nous permet d'écrire la matrice des produits partiels sous la forme (Fig. II.7) :

						a	x_3	x_2	x_1	x_0	×
						b	y_3	y_2	y_1	y_0	
ay_0	ay_0	ay_0	ay_0	ay_0	ay_0	x_3y_0	x_2y_0	x_1y_0	x_0y_0		
ay_1	ay_1	ay_1	ay_1	ay_1	x_3y_1	x_2y_1	x_1y_1	x_0y_1	0		
ay_2	ay_2	ay_2	ay_2	x_3y_2	x_2y_2	x_1y_2	x_0y_2	0	0		
ay_3	ay_3	ay_3	x_3y_3	x_2y_3	x_1y_3	x_0y_3	0	0	0		
-ab	-ab	$-bx_3$	$-bx_2$	$-bx_1$	$-bx_0$	0	0	0	0		X.b*
P_9	P_8	P_7	P_6	P_5	P_4	P_3	P_2	P_1	P_0	=	P

Figure II.7

(2.56) $$Xb^* = \begin{cases} \hat{X} & \text{si } b=1 \\ 0 & \text{si } b=0 \end{cases}$$

$$Xb^* = b.\overline{X} + b$$

arithmétique

complément booléen de X

$\Longrightarrow$ la ligne Xb* s'écrit en deux lignes :

$b\overline{a}$	$b\overline{a}$	$b\overline{x}_3$	$b\overline{x}_2$	$b\overline{x}_1$	$b\overline{x}_0$	0	0	0	0
0	0	0	0	0	b	0	0	0	0

et la soustraction se trouve ainsi réduite à une simple complémentation booléenne et à l'addition d'un terme de plus faible poids (c.à.d. b).

Il est aussi possible de supprimer l'irrégularité dûe au signe du multiplieur Y, en utilisant des algorithmes qui consistent à recoder le multiplieur Y par un moyen ne faisant plus la distinction entre le bit de signe et les autres bits, tels que algorithmes de Booth et de Wallace, fréquemment utilisés pour les multiplieurs cellulaires |20| |21| |22| que nous exposerons dans le prochain paragraphe.

II.2 - MATERIALISATION

II.2.1 - L'additionneur

Avant daborder directement le principe de conception de l'additionneur, il convient de rappeler les relations (2.32) et (2.33) qui expriment r_{i+1} et $\bar{r}_{i+1}$ en fonction de la variable r_0 et d'interpréter les termes $A^{\ell}_{0,i}$.

$$A \in \{G\ ,\ P\ ,\ T\ ,\ \delta\ ,\ \gamma\}$$

$$\ell \in \{P\ ,\ T\}$$

$$(2.32)\quad \begin{cases} r_{i+1} = G_{0,i}^{P} + P_{0,i}\, r_0 \\ \bar{r}_{i+1} = \gamma_{0,i}^{P} + \delta_{0,i}^{P}\, \bar{r}_0 \end{cases} \qquad \text{paire } (G_k\ ,\ P_k)$$

$$(2.33)\quad \begin{cases} r_{i+1} = G_{0,i}^{T} + T_{0,i}\, r_0 \\ \bar{r}_{i+1} = \gamma_{0,i}^{T} + \delta_{0,i}^{T}\, \bar{r}_0 \end{cases} \qquad \text{paire } (G_k\ ,\ T_k)$$

Les relations (2.32) et (2.33) peuvent être spécialisées par rapport à r_0 et nous avons, pour r_0=0 et r_0=1 :

$$(2.57)\quad \begin{cases} r_{i+1}\ (r_0=1) = G_{0,i}^{P} + P_{0,i} \\ \bar{r}_{i+1}\ (r_0=1) = \gamma_{0,i}^{P} \\ r_{i+1}\ (r_0=0) = G_{0,i}^{P} \\ \bar{r}_{i+1}\ (r_0=0) = \gamma_{0,i}^{P} + \delta_{0,i}^{P} \end{cases} \qquad \text{paire } (G_k\ ,\ P_k)$$

$$(2.58)\quad \begin{cases} r_{i+1}\ (r_0=1) = G_{0,i}^{T} + T_{0,i} \\ \bar{r}_{i+1}\ (r_0=1) = \gamma_{0,i}^{T} \\ r_{i+1}\ (r_0=0) = G_{0,i}^{T} \\ \bar{r}_{i+1}\ (r_0=0) = \gamma_{0,i}^{T} + \delta_{0,i}^{T} \end{cases} \qquad \text{paire } (G_k\ ,\ T_k)$$

II.2.1.1 - Interprétation et définition

Considérons le groupe de relations (2.57) :

$$r_{i+1}\ (r_0=1) = 1$$

signifie que lorsqu'un report entrant r_0 est égal à 1, il se trouve inchangé à la sortie. On dit que le bloc transmet un "1".

D'où la condition de transmission de "1" :

$$(2.59)\qquad \boxed{T_{0,i}^{1*} = G_{0,i}^{P} + P_{0,i}}$$

où $T_{0,i}^{1*}$ est une variable booléenne qui, lorsqu'elle vaut "1", indique que le bloc transmet un "1".

$$\bar{r}_{i+1}\ (r_0=0) = 1$$

signifie que lorsqu'un report entrant r_0 est égal à 0, il se trouve inchangé à la sortie. On dit que le bloc transmet un "0".

D'où la condition de transmission de "0" :

$$(2.60)\qquad \boxed{T_{0,i}^{0*} = \gamma_{0,i}^{P} + \delta_{0,i}^{P}}$$

où $T_{0,i}^{0*}$ est une variable booléenne qui, lorsqu'elle vaut "1", indique que le bloc transmet un "0".

On peut définir une fonction de <u>transfert</u> ou <u>transmission de bloc</u> par :

$$T_{0,i}^{*} = T_{0,i}^{0*}\,\bar{r}_0 + T_{0,i}^{1*}\,r_0 \qquad (2.61)$$

où $T_{0,i}^{*}$ est une variable booléenne qui, lorsqu'elle vaut "1" indique qu'un report entrant r_0 ressort inchangé.

Lorsqu'un bloc <u>ne transmet pas</u>, un report entrant r_0 ressort modifié et le bloc devient <u>générateur</u> :

- <u>de zéro</u> : si $r_{i+1}\,(r_0=1) = 0$ ou $\bar{r}_{i+1}\,(r_0=1) = 1$

d'où la condition de génération de "0" :

$$G_{0,i}^{0} = \gamma_{0,i}^{P} \qquad (2.62)$$

où $G_{0,i}^{0}$ est une variable booléenne qui, lorsqu'elle vaut "1", indique que le bloc génère un "0" pour un report entrant égal à "1".

- <u>de un</u> : si $r_{i+1}\,(r_0=0) = 1$

d'où la condition de génération de "1" :

$$G_{0,i}^{1} = G_{0,i}^{P} \qquad (2.63)$$

où $G_{0,i}^{1}$ est une variable booléenne qui, lorsqu'elle vaut "1", indique que le bloc génère un "1" pour un reprot entrant égal à "0".

Et la fonction génération peut être définie par :

$$G_{0,i} = G_{0,i}^{1}\,\bar{r}_0 + G_{0,i}^{0}\,r_0 \qquad (2.64)$$

ou

$$\boxed{G_{0,i} = \overline{T}_{0,i}^{\,*}} \tag{2.65}$$

condition de non transfert.

Enfin, le terme propagation est défini par une variable booléenne, notée $P_{0,i}^{\,*}$ qui, lorsqu'elle vaut "1", indique qu'une variation du report entrant r_0 est représentée à la sortie et donnée par :

$$P_{0,i}^{\,*} = r_{i+1}\,(r_0) \oplus r_{i+1}\,(\overline{r}_0) \tag{2.66}$$

$$r_0 = 0 \text{ ou } 1 \implies P_{0,i}^{\,*} = r_{i+1}\,(r_0=1) \oplus r_{i+1}\,(r_0=0)$$

$$P_{0,i}^{\,*} = T_{0,i}^{\,1*} \oplus G_{0,i}^{\,1} \tag{2.67}$$

$$\begin{aligned}
P_{0,i}^{\,*} &= r_{i+1}\,(r_0=1).\overline{r}_{i+1}\,(r_0=0) + r_{i+1}\,(r_0=1).r_{i+1}\,(r_0=0) \\
&= (G_{0,i}^{\,P} + P_{0,i})\,(\gamma_{0,i}^{\,P} + \delta_{0,i}^{\,P}) + \gamma_{0,i}^{\,P}.G_{0,i}^{\,P} \\
&= G_{0,i}^{\,P}.\gamma_{0,i}^{\,P} + G_{0,i}^{\,P}.\delta_{0,i}^{\,P} + P_{0,i}.\gamma_{0,i}^{\,P} + P_{0,i}.\delta_{0,i}^{\,P} + \gamma_{0,i}^{\,P}.G_{0,i} \\
&= G_{0,i}^{\,P}\,(\gamma_{0,i}^{\,P} + \delta_{0,i}^{\,P}) + \gamma_{0,i}^{\,P}\,(G_{0,i}^{\,P} + P_{0,i}) + P_{0,i}.\delta_{0,i}^{\,P} \\
&= r_{i+1}\,(r_0=0).\overline{r}_{i+1}\,(r_0=0) + \overline{r}_{i+1}\,(r_0=1).r_{i+1}\,(r_0=1) + P_{0,i}.\delta_{0,i}^{\,P} \\
&= P_{0,i}.\delta_{0,i}^{\,P}
\end{aligned}$$

$$P_{0,i}^{\,*} = \prod_{k=0}^{i} P_k . \prod_{j=0}^{i} \overline{G}_j = \prod_{j=0}^{i} P_j.\overline{G}_j$$

$$\left.\begin{aligned} P_j &= x_j \oplus y_j \\ \overline{G}_j &= \overline{x_j.y_j} \end{aligned}\right\} \implies P_j.\overline{G}_j = P_j$$

$$\boxed{P_{0,i}^{\,*} = P_{0,i}} \tag{2.68}$$

Le même raisonnement peut être fait avec le groupe de relations (2.58) et nous avons les deux groupes de relations :

Paires (G_k , P_k)			Paires (G_k , T_k)
$T_{0,i}^{1*} = G_{0,i}^{P} + P_{0,i}$	$= r_{i+1}(r_0=1)$		$T_{0,i}^{1*} = G_{0,i}^{T} + T_{0,i}$
$T_{0,i}^{0*} = \gamma_{0,i}^{P} + \delta_{0,i}^{P}$	$= \bar{r}_{i+1}(r_0=0)$		$T_{0,i}^{1*} = \gamma_{0,i}^{T} + \delta_{0,i}$
$G_{0,i}^{1} = G_{0,i}^{P}$	$= r_{i+1}(r_0=0)$		$G_{0,i}^{1} = G_{0,i}^{T}$
$G_{0,i}^{0} = \gamma_{0,i}^{P}$	$= \bar{r}_{i+1}(r_0=1)$		$G_{0,i}^{0} = \gamma_{0,i}^{T}$
$P_{0,i}^{*} = P_{0,i}\cdot\delta_{0,i}^{P}$			$P_{0,i}^{*} = T_{0,i}\cdot\delta_{0,i}^{T}$
$P_{0,i}^{*} = P_{0,i}$			$P_{0,i}^{*} = \prod_{j=0}^{i} \bar{G}_j.T_j = \prod_{j=0}^{i} P_j$
			$P_{0,i}^{*} = P_{0,i}$

avec, en conséquence de $r_{i+1}(r_0).\bar{r}_{i+1}(r_0) = 0$:

$$(2.69)\quad \begin{cases} T_{0,i}^{1*}.G_{0,i}^{0} = 0 \iff T_{0,i}^{1*} = \bar{G}_{0,i}^{0} \\ T_{0,i}^{0*}.G_{0,i}^{1} = 0 \iff T_{0,i}^{0*} = \bar{G}_{0,i}^{1} \end{cases}$$

$$(2.70)\quad \begin{cases} P_{0,i}^{*} = P_{0,i} = G_{0,i}^{1} \oplus T_{0,i}^{1*} \\ \qquad\qquad = T_{0,i}^{0*} \oplus G_{0,i}^{0} \\ \qquad\qquad = T_{0,i}^{0*} \oplus G_{0,i}^{0} \end{cases}$$

$$(2.71)\quad \begin{cases} T_{0,i}^{0*} + G_{0,i}^{0} = T_{0,i}^{0*} \\ T_{0,i}^{1*} + G_{0,i}^{1} = T_{0,i}^{1*} \end{cases}$$

$$(2.72)\quad \begin{cases} r_{i+1} = G_{0,i}^{1}.\bar{r}_0 + T_{0,i}^{1*}.r_0 = G_{0,i}^{1} + T_{0,i}^{1*}.r_0 \\ \qquad = G_{0,i}^{1} + P_{0,i}.r_0 = G_{0,i}^{1} + T_{0,i}.r_0 \\ \bar{r}_{i+1} = T_{0,i}^{0*}.\bar{r}_0 + G_{0,i}^{0}.r_0 = G_{0,i}^{0} + T_{0,i}^{0*}.\bar{r}_0 \end{cases}$$

Les fonctions transmission, génération et propagation, que nous venons de définir pour un bloc $[0 , i]$, d'origine zéro et de longueur i+1, qui reçoit r_0 comme variable d'entrée et les paramètres x_j et y_j

(j = 0, 1, ..., i), peuvent être tranposées pour un bloc d'origine et de longueur quelconque B [k , m], qui reçoit r_k comme variable d'entrée et les paramètres x_ℓ et y_ℓ (ℓ = k, k+1, ..., m) et notés :

$$C^{\ell}_{k,m} \qquad C \in \{G , T , P\} \qquad \ell \in \{0 , 1 , 0^* , 1^*\}$$

II.2.1.2 - Méthode générale de conception

La conception d'additionneur rapide, dont le principe consiste à générer tous les reports r_i, peut être faite en quatre étapes :

- décomposition en blocs adjacents du mot binaire

- génération du report entrant pour chaque bloc

- génération des reports de tout rang r_i

- recombinaison des reports r_i avec les paires x_i, y_i ou G_i et P_i (ou T_i), pour calculer les sommes S_i.

II.2.1.2.1 - Caractéristiques des blocs

Soit le bloc B (0 , i) d'origine 0 et de longueur i+1, le report sortant r_{i+1} est fonction du report entrant r_0 et des paramètres $G_{0,i}^{1}$, $T_{0,i}^{1*}$ ou $T_{0,i}$ ou $P_{0,i}$ (2.59), selon que l'on prend la paire G_k , P_k ou la paire G_k , T_k.

Considérons le cas de la paire G_k , P_k ; nous avons :

$$(2.73) \qquad r_{i+1} = G_{0,i} + P_{0,i} \cdot r_0 \qquad \text{où} \qquad G_{0,i} = G_{0,i}^{1}$$

Le calcul de r_{i+1} se ramène au calcul de $G_{0,i}$ et $P_{0,i}$ qui peut se faire de deux façons :

- récurrente par les relations :

$$(2.74)\quad \begin{cases} G_{0,i} = G_i + P_i . G_{0,i-1} \\ P_{0,i} = P_i . P_{0,i-1} \end{cases}$$

- explicite par les relations :

$$(2.75)\quad \begin{cases} G_{0,i} = \sum_{j=0}^{i} \left(\prod_{k=j+1}^{i} P_k \right) G_j \\ P_{0,i} = \prod_{k=0}^{i} P_k \end{cases}$$

Il est évident que la forme récurrente n'est pas intéressante dans le contexte où on cherche à matérialiser des additionneurs rapides. Nous nous intéressons donc qu'à la forme explicite.

a) Structure générale d'un bloc :

La figure II.8 donne la structure générale d'un bloc de longueur "4" et d'origine "0".

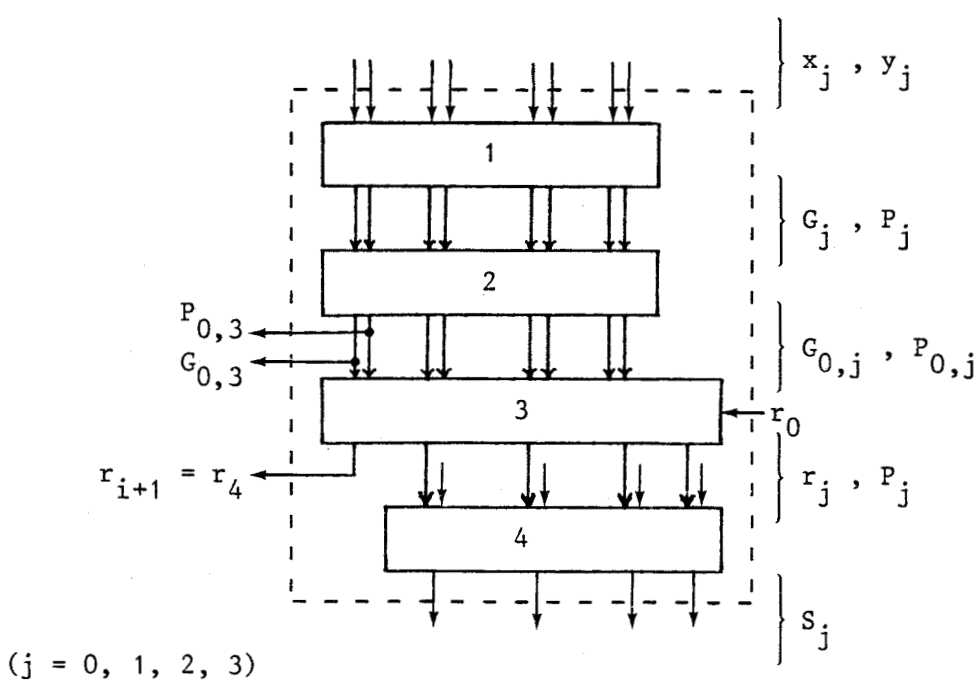


Figure II.8

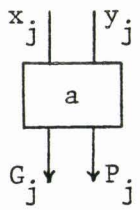
Un bloc est constitué de 4 réseaux numérotés de 1 à 4 :

- Réseau 1 :

Le réseau 1 reçoit les x_j, y_j et calcule : (Fig. II.9)

$$\begin{cases} P_j = x_j \oplus y_j \\ G_j = x_j . y_j \end{cases}$$

par a :



selon le schéma suivant :

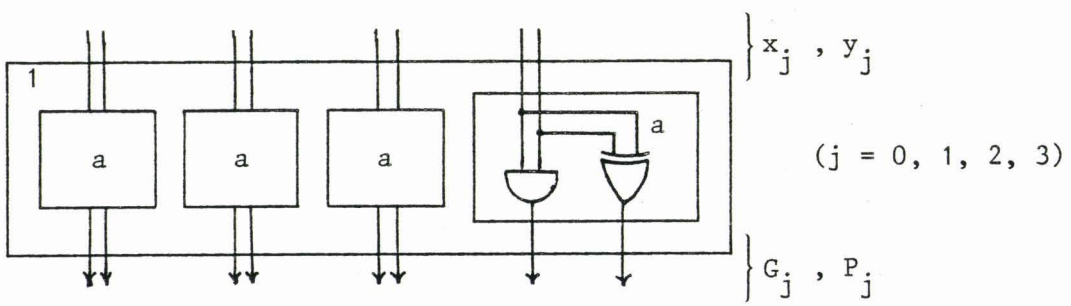


Figure II.9

- Réseau 2 :

Le réseau 2 reçoit les G_j, P_j et calcule : (Fig. II.10)

$$\begin{cases} G_{0,0} = G_0 \\ P_{0,0} = P_0 \end{cases} \quad \text{par } R_1$$

$$\begin{cases} G_{0,1} = G_1 + P_1 G_0 \\ P_{0,1} = P_0 P_1 \end{cases} \quad \text{par } R_2$$

$$\begin{cases} G_{0,2} = G_2 + P_2 G_1 + P_2 P_1 G_0 \\ P_{0,2} = P_0 P_1 P_2 \end{cases} \quad \text{par } R_3$$

$$\begin{cases} G_{0,3} = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 \\ P_{0,3} = P_0 P_1 P_2 3 \end{cases} \quad \text{par } R_4$$

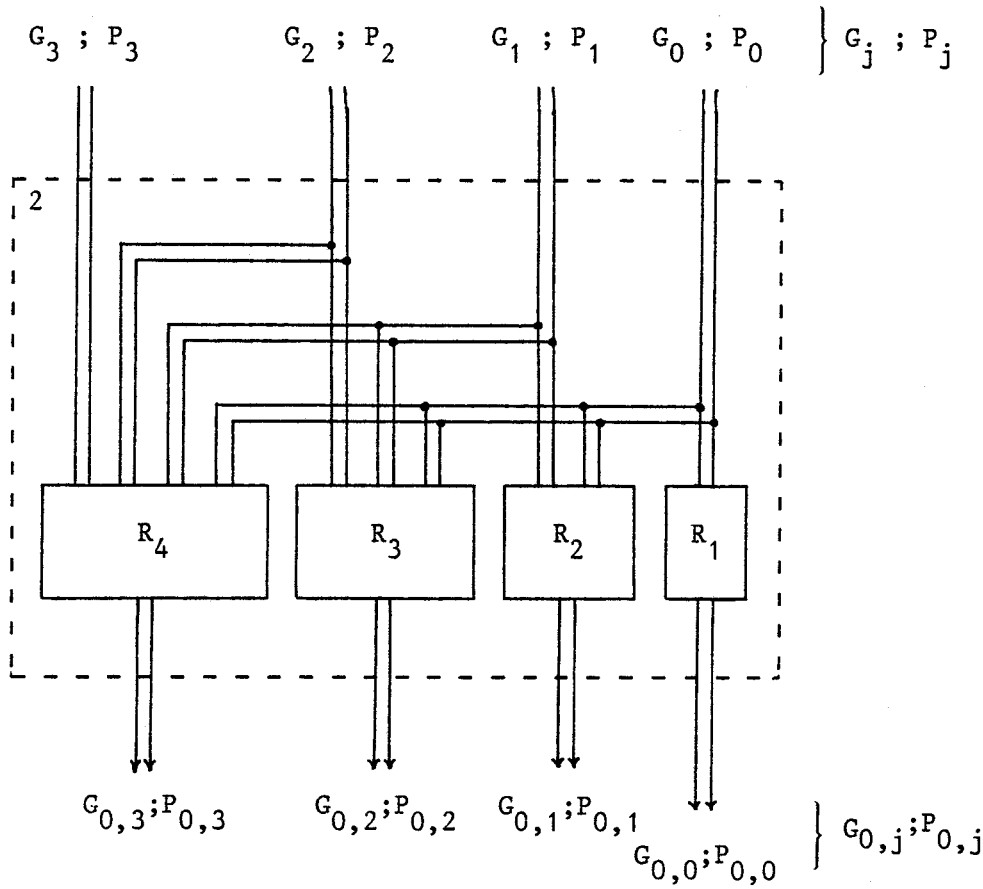
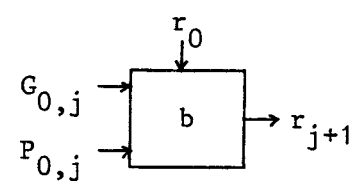


Figure II.10

- Réseau 3 :

Le réseau 3 reçoit les $G_{0,j}$, $P_{0,j}$ et r_0 et calcule :

$r_{j+1} = G_{0,j} + P_{0,j}\, r_0$ par b :



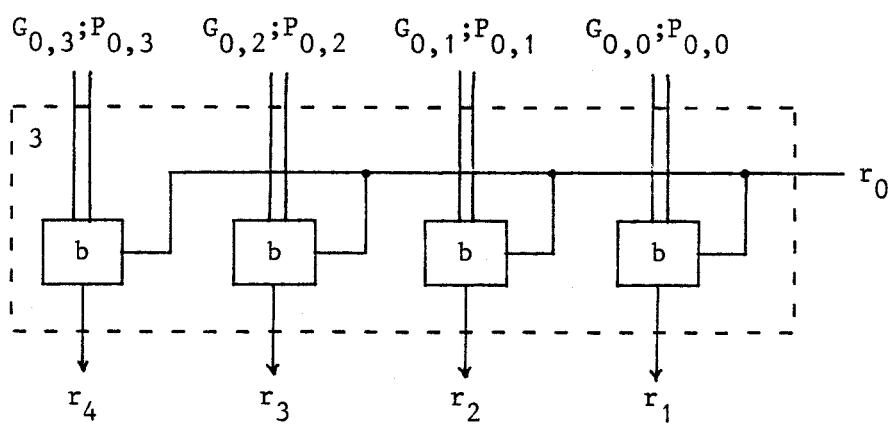
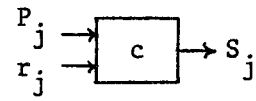


Figure II.11

- Réseau 4 :

Le réseau 4 reçoit les r_j, P_j et calcule :

$S_j = P_j \oplus r_j$ par c :



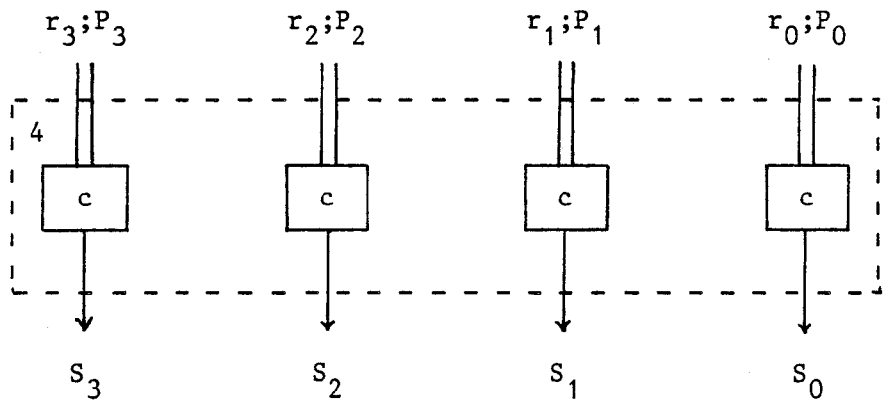


Figure II.12

Le bloc $B_{[0,3]}$ étant complètement défini. Nous convenons de le noter B_0 et de le schématiser par la figure II.13 :

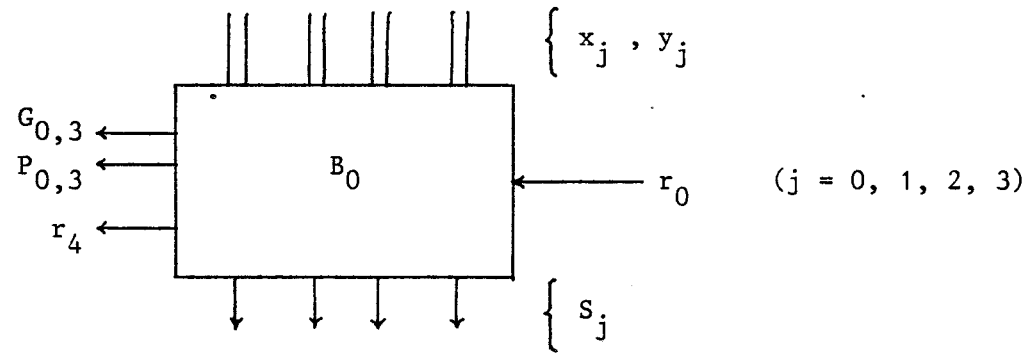


Figure II.13

Le problème maintenant, consiste à interconnecter les blocs entre eux, de manière à avoir l'additonneur compelt ou le bloc $B_{[0,n]}$.

b) Différents types d'interconnexions :

Considèrons le cas particulier où n = 15 (mot binaire de 16 bits).

Le bloc $B_{[0,15]}$ peut être partitionné en 4 blocs adjacents de type B_0, notés successivement B_0, B_1, B_2 et B_3 avec B_0 le bloc de plus faible poids.

- connexion série ou cascade :

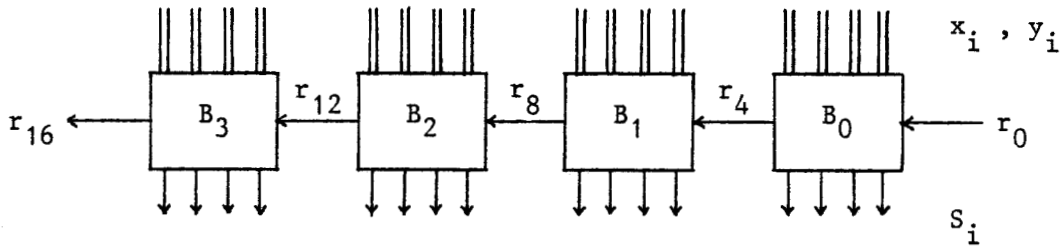


Figure II.14

- connexion cascade et parallèle :

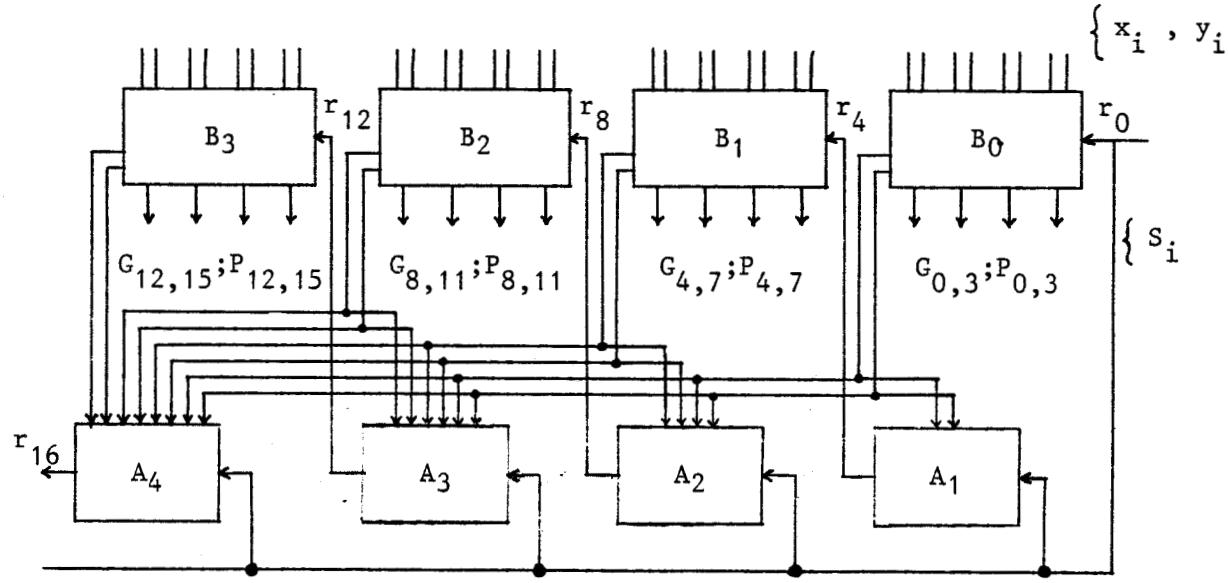


Figure II.15

Posons :

$$\begin{cases} G_{0,3} = G(B_0) \\ P_{0,3} = P(B_0) \end{cases} \qquad \begin{cases} G_{8,11} = G(B_2) \\ P_{8,11} = P(B_2) \end{cases}$$

$$\begin{cases} G_{4,7} = G(B_1) \\ P_{4,7} = P(B_1) \end{cases} \qquad \begin{cases} G_{12,15} = G(B_3) \\ P_{12,15} = P(B_3) \end{cases}$$

D'après la relation (2.73), nous avons :

$$(2.76) \qquad \begin{cases} r_4 = G_{0,3} + P_{0,3}\, r_0 & \text{calculé par } A_1 \\ r_8 = G_{0,7} + P_{0,7}\, r_0 & \text{calculé par } A_2 \\ r_{12} = G_{0,11} + P_{0,11}\, r_0 & \text{calculé par } A_3 \\ r_{16} = G_{0,15} + P_{0,15}\, r_0 & \text{calculé par } A_4 \end{cases}$$

Les entrées de A_1 sont $G_{0,3}$, $P_{0,3}$ et r_0 et le calcul de r_4 se fait sans problème.

Les entrées de A_2 sont r_0, $G_{0,3}$, $P_{0,3}$ et $G_{4,7}$ et $P_{4,7}$ ou encore $G(B_0)$, $P(B_0)$ et $G(B_1)$, $P(B_1)$ et pour calculer r_8 il faut d'abord calculer $G_{0,7}$ et $P_{0,7}$ que nous notons $G(B_1B_0)$ et $P(B_1B_0)$ du bloc B_1B_0 formé de la réunion des blocs B_1 et B_0.

D'une façon générale, pour le bloc $B_{i+1}B_i$ formé des deux blocs adjacents B_{i+1} et B_i :

$$(2.77) \qquad \begin{cases} G(B_{i+1}B_i) = G(B_{i+1}) + P(B_{i+i})\, G(B_i) \\ P(B_{i+1}B_i) = P(B_{i+1}).P(B_i) \end{cases}$$

Démonstration :

Pour simplifier les notations, prenons $B_i = B_{0,3} = B_0$

et $B_{i+1} = B_{4,7} = B_1$

et donc $B_1B_0 = B_{0,7}$

schématisé sur la figure II.16.

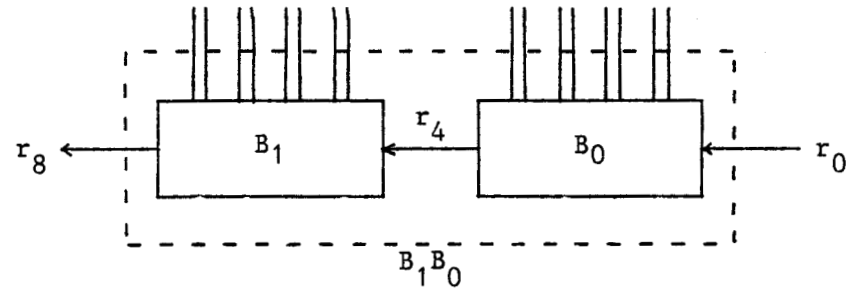


Figure II.16

Les relations (2.77) peuvent être démontrées de 3 façons :

- par récurrence

- à partir des relations (2.73) et (2.76) en écrivant :

$$r_4 = f\ (G(B_0),P(B_0),r_0)$$

$$r_8 = f\ (G(B_1),P(B_1),r_4)$$

$$r_8 = f\ (G(B_1),P(B_1),f(G(B_0),P(B_0),r_0) \tag{2.78}$$

$$r_8 = G_{0,7} + P_{0,7}\ r_0 \tag{2.79}$$

et par identification de (2.78) et (2.79) par rapport à r_0, on détermine :

$$G_{0,7} = G(B_1B_0) = g\ (G(B_0),G(B_1),P(B_0),P(B_1))$$

$$P_{0,7} = P(B_1B_0) = h\ (G(B_0),G(B_1),P(B_0),P(B_1))$$

- à partir des relations (2.75) que nous utilisons :

$$G(B_0) = \sum_{j=0}^{3} \left(\prod_{k=j+1}^{3} P_k \right) G_j \tag{2.80}$$

$$P(B_0) = \prod_{k=0}^{3} P_k \tag{2.81}$$

} pour B_0

$$G(B_1) = \sum_{j=4}^{7} \left(\prod_{k=j+1}^{7} P_k \right) G_j \tag{2.82}$$

$$P(B_1) = \prod_{k=4}^{7} P_k \tag{2.83}$$

} pour B_1

$$(2.84)\qquad G(B_1B_0) = \sum_{j=0}^{7} \left(\prod_{k=j+1}^{7} P_k \right) G_j$$

$$(2.85)\qquad P(B_1B_0) = \prod_{k=0}^{7} P_k$$

} pour B_1B_0

La relation (2.84) peut s'écrire :

$$G(B_1B_0) = \sum_{j=0}^{7} \left(\prod_{k=j+1}^{7} P_k \right) G_j = \sum_{j=0}^{3} \left(\prod_{k=j+1}^{7} P_k \right) G_j + \sum_{j=4}^{7} \left(\prod_{k=j+1}^{7} P_k \right) G_j$$

$$= \sum_{j=0}^{3} \left(\prod_{k=j+1}^{3} P_k \right) G_j . \prod_{\ell=4}^{7} P_\ell + G(B_1)$$

$$(2.86)\qquad \boxed{G(B_1B_0) = G(B_0).P(B_1) + G(B_1)}$$

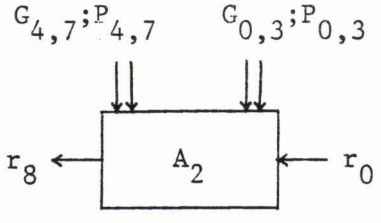
La relation (2.85) peut s'écrire :

$$P(B_1B_0) = \prod_{k=0}^{7} P_k = \prod_{k=0}^{3} P_k . \prod_{j=4}^{7} P_j$$

$$(2.87)\qquad \boxed{P(B_1B_0) = P(B_0).P(B_1)}$$

D'où le réseau A_2 :

$$r_8 = G(B_1B_0) + P(B_1B_0)\, r_0$$



$$(2.88)\qquad \boxed{r_8 = G_{0,3}.P_{4,7} + P_{0,3}.P_{4,7}\, r_0}$$

Les relations (2.86) et (2.87) nous permettent d'écrire pour le bloc $B_2B_1B_0$ formé de la réunion des blocs B_2, B_1 et B_0 :

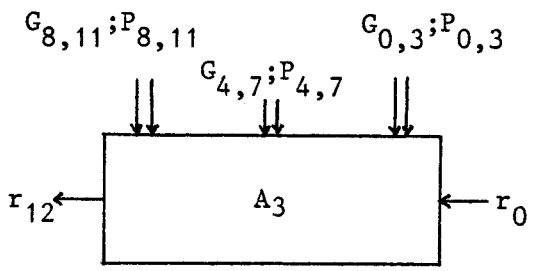
$$G(B_2B_1B_0) = G((B_2)(B_1B_0)) = G(B_2) + P(B_2).G(B_1B_0)$$
$$= G(B_2) + P(B_2).(G(B_0).P(B_1) + G(B_1))$$
$$= G(B_2) + P(B_2).G(B_0).P(B_1) + P(B_2).G(B_1)$$

(2.89) $$\boxed{G(B_2B_1B_0) = G(B_2) + P(B_2).G(B_1) + P(B_2).P(B_1).G(B_0)}$$

et $$P(B_2B_1B_0) = P(B_2).P(B_1B_0) = P(B_2).P(B_1).P(B_0)$$

(2.90) $$\boxed{P(B_2B_1B_0) = P(B_2).P(B_1).P(B_0)}$$

D'où le réseau A_3 :



$$r_{12} = G(B_2B_1B_0) + P(B_2B_1B_0)\ r_0$$

(2.91) $$\boxed{r_{12} = G_{8,11} + P_{4,7}.G_{4,7} + P_{8,11}.P_{4,7}.G_{0,3} + P_{8,11}.P_{4,7}.P_{0,3}.r_0}$$

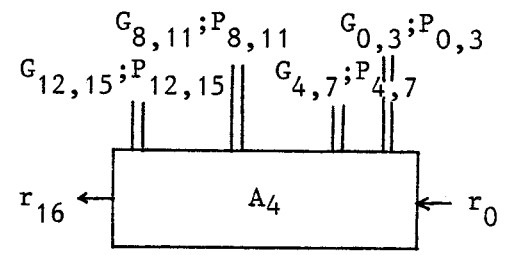
Les relations (2.73), (2.74), (2.76) et (2.77) nous permettent d'écrire pour le bloc $B_3B_2B_1B_0$:

$$G(B_3B_2B_1B_0) = G(B_3) + P(B_3)\left[G(B_2) + P(B_2).G(B_1) + P(B_2).P(B_1).G(B_0)\right]$$

(2.92) $$\boxed{\begin{aligned} G(B_3B_2B_1B_0) = {} & G(B_3) + P(B_3).G(B_2) + P(B_3).P(B_2).G(B_1) \\ & + P(B_3).P(B_2).P(B_1).G(B_0) \end{aligned}}$$

(2.93) $$\boxed{P(B_3B_2B_1B_0) = P(B_3).P(B_2).P(B_1).P(B_0)}$$

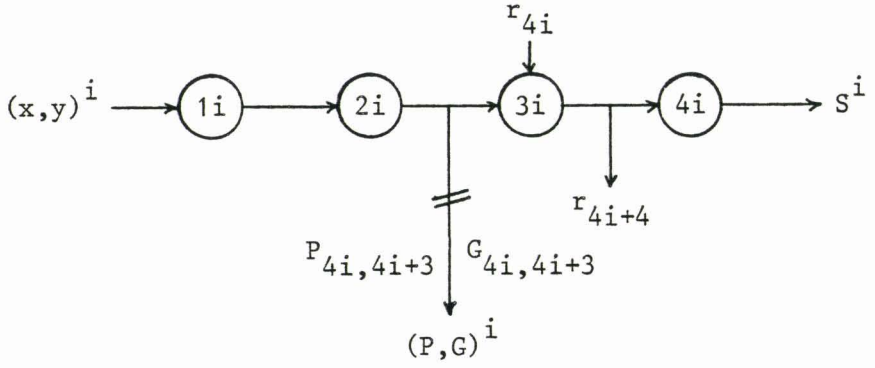
D'où le réseau A_4 :



$$r_{16} = G(B_3B_2B_1B_0) + P(B_3B_2B_1B_0)\ r_0$$

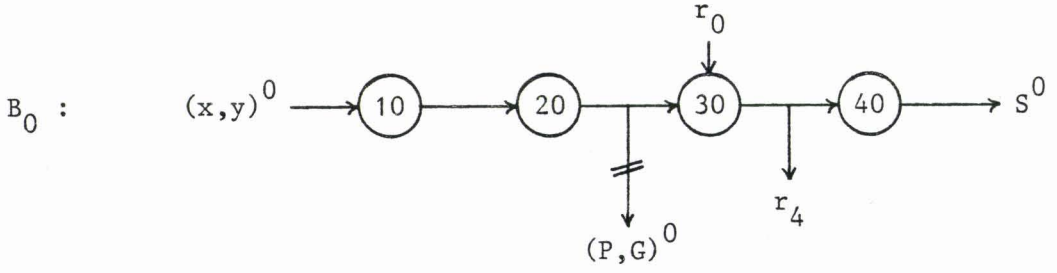
(2.94)
$$r_{16} = G_{12,15} + P_{12,15}\cdot G_{8,11} + P_{12,15}\cdot P_{8,11}\cdot G_{4,7} + P_{12,15}\cdot P_{8,11}\cdot P_{4,7}\cdot G_{0,3} + P_{12,15}\cdot P_{8,11}\cdot P_{4,7}\cdot P_{0,3}\cdot r_0$$

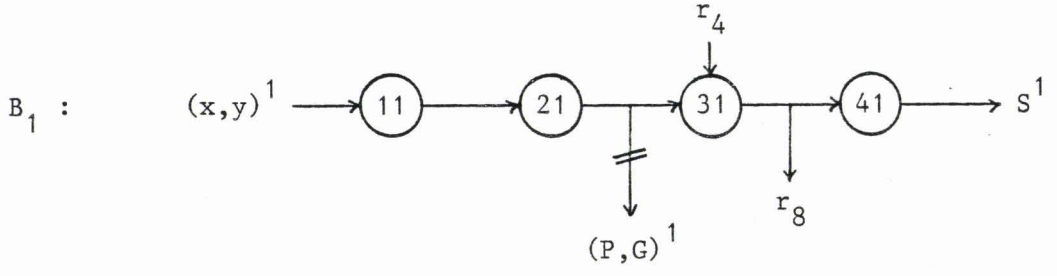
Convenons de représenter le bloc B_i de longueur 4 par :



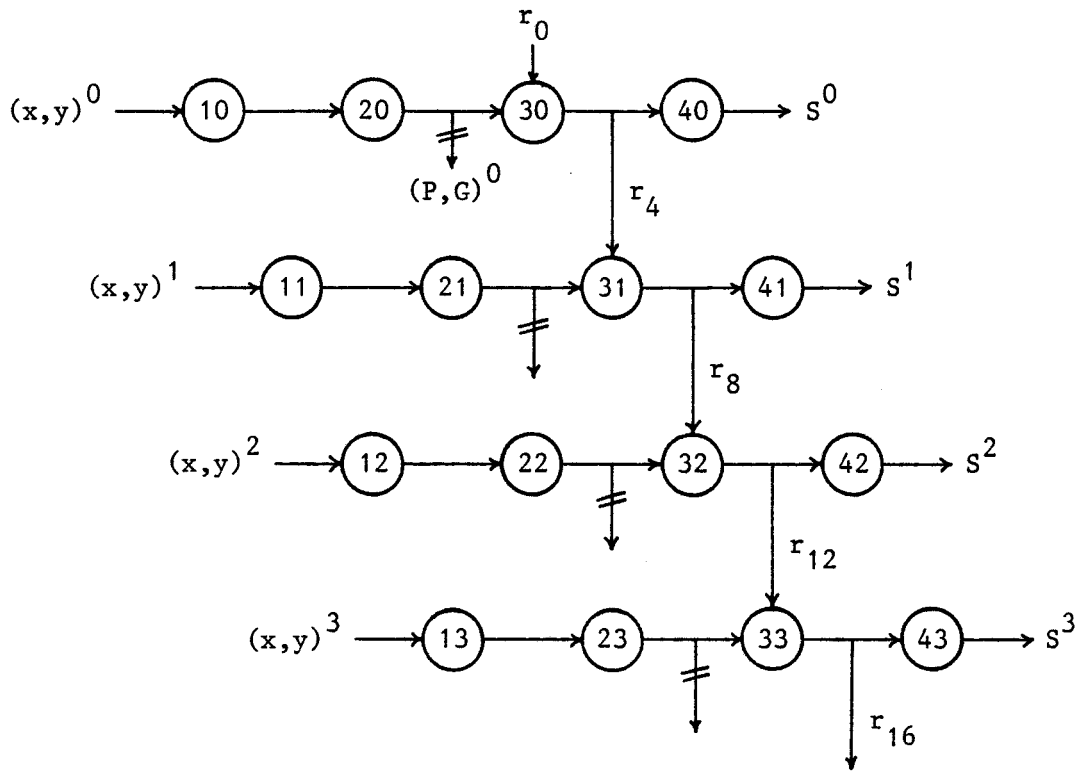
où 1i, 2i, 3i, 4i sont les réseaux 1, 2, 3 et 4 de la figure II.8.

Nous avons, par exemple, pour :



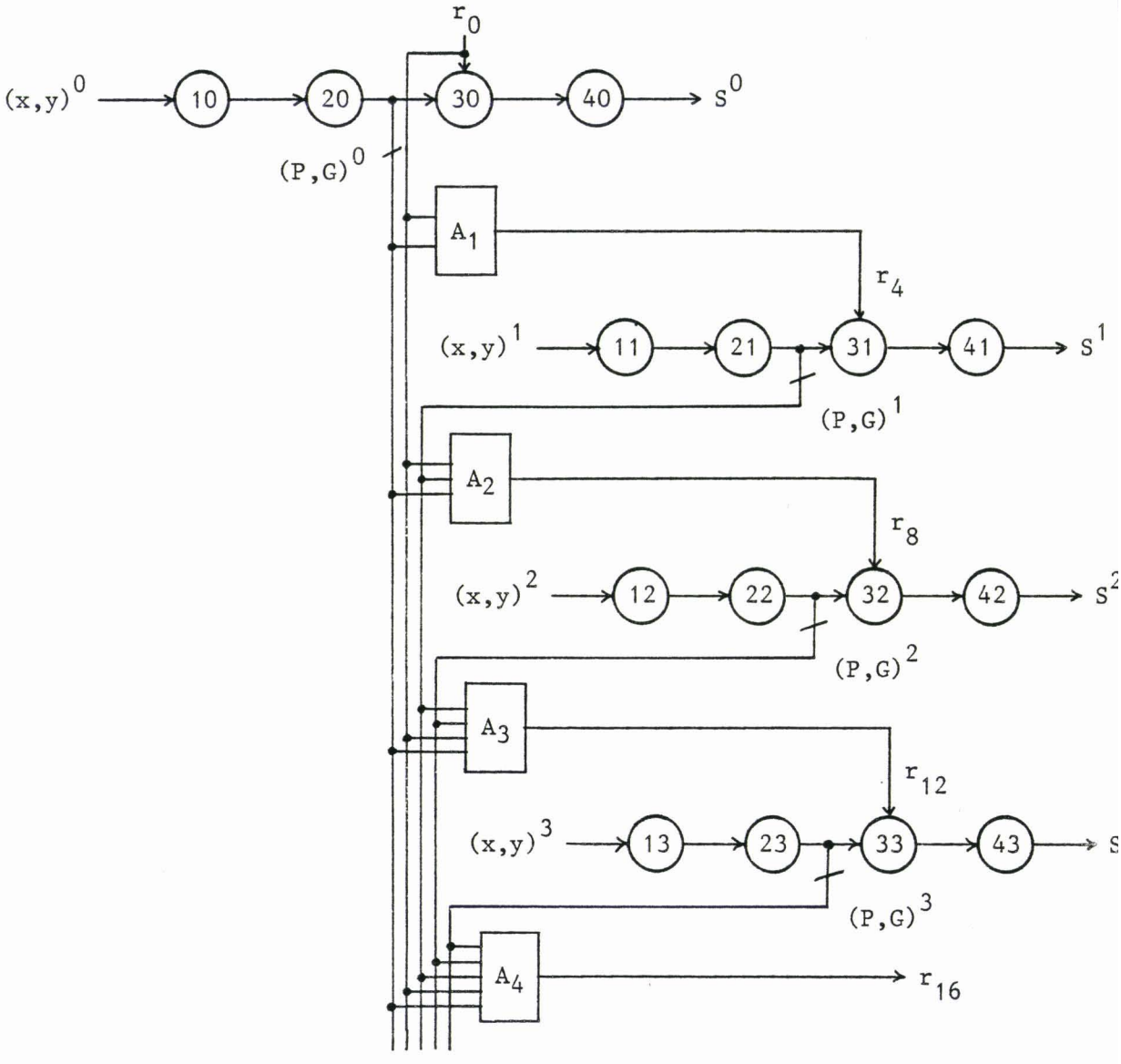


Les figures II.14 et II.15, selon cette nouvelle représentation des blocs, prennent les formes respectives des figures II.16 et II.17 :



CONNEXION CASCADE

Figure II.16



CONNEXION CASCADE ET PARALLELE

Figure II.17

Dans le cas de la Figure II.16 (connexion cascade), le chemin le plus long pour aller de $(x,y)^0$, r_0 à S^3 est le suivant :

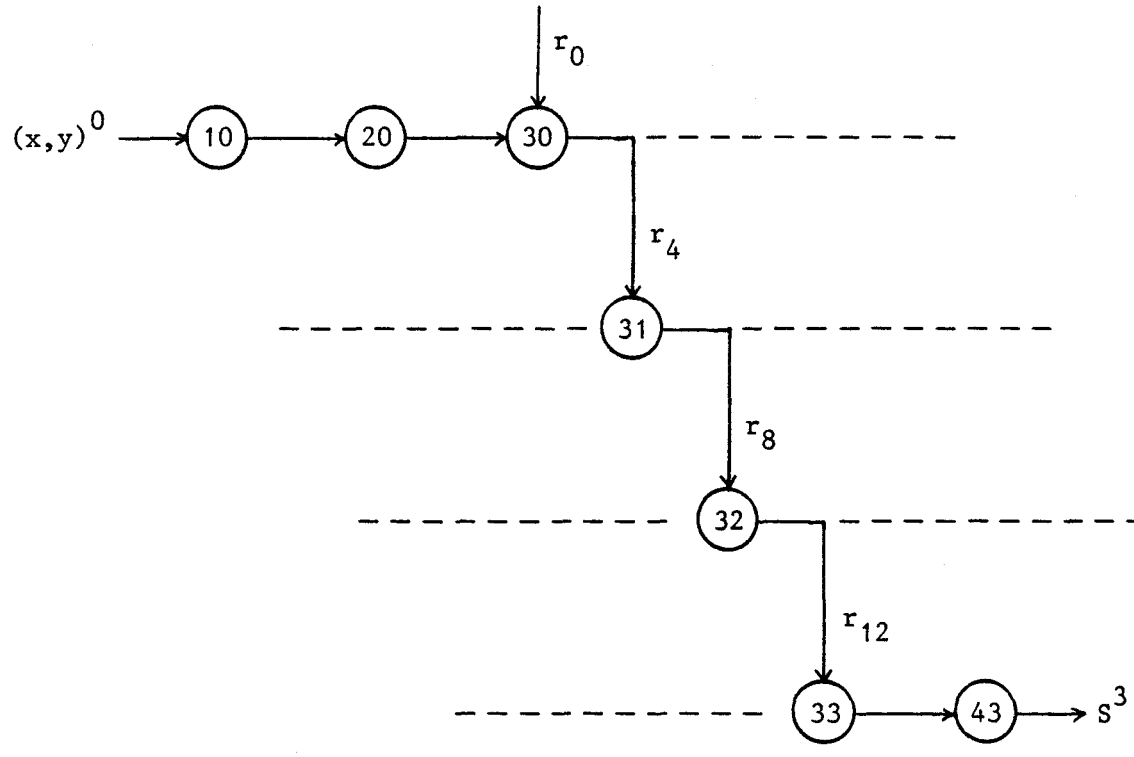


Figure II.18

Si chaque circuit de type —(ij)— (i = 1, 2, 3, 4 ; j = 0, 1, 2, 3) calcule des fonctions logiques en deux couches logiques ; le temps de calcul de tous les S_i (i = 0, 1, ..., 15) est donc de 7 × 2 = 14 couches logiques.

Dans le cas de l'interconnexion cascade et parallèle, il vient le graphe suivant, mettant en évidence le rôle des blocs A_i. Le flux de données $(x,y)^j$ traverse simultanèment la couche (1,i) puis les couches (2,i), (A_i), (3,i) et (4,i).

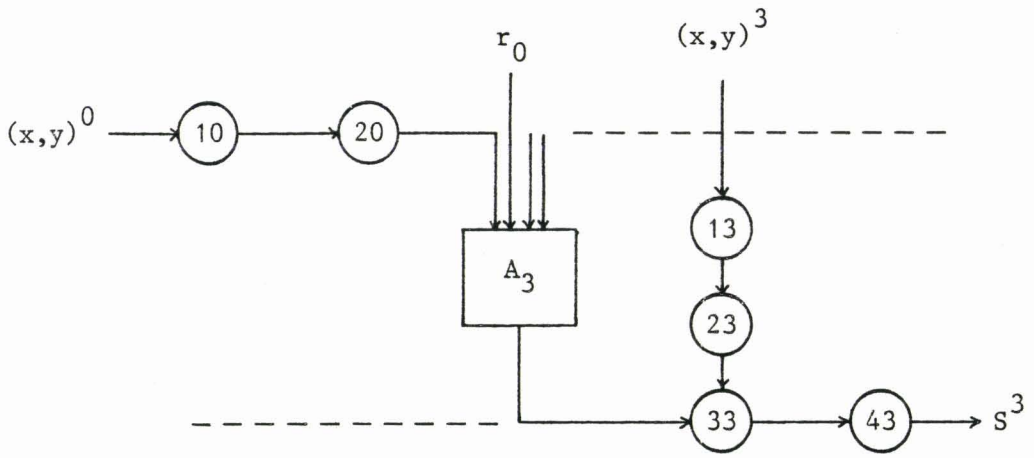


Figure II.19

Le circuit [A3] comme les circuits —(ij)—, calcule des fonctions logiques en deux couches logiques. Le temps de calcul de tous les S_i ($i = 0, 1, \ldots, 15$) est de $5 \times 2 = 10$ couches logiques.

Deux conclusions immédiates se dégagent :

- La structure connexion cascade et parallèle réduit de 4 couches logiques, le temps de calcul par rapport à la structure connexion cascade.

- La structure connexion cascade et parallèle augmente le prix de l'additionneur de 10 portes logiques "ET" et de 4 portes logiques "OU" pour la réalisation des circuits A_i ($i = 1, 2, 3, 4$).

Ces deux conclusions nous conduisent à poser deux questions :

- Si aucune structure ne satisfait le temps de calcul, peut-on encore améliorer la structure la plus rapide ?

- Si une structure satisfait le critère de temps, mais non le critère de coût, peut-on la conserver par la mise en œuvre de technologies spécifiques ?

Nous examinons tout de suite la première question, dans le cas où aucune structure n'est satisfaisante. Quant à la deuxième question, comme mentionné dans l'introduction, elle fera l'objet du 3ème paragraphe de ce chapitre.

c) Amélioration du temps de calcul :

Le problème est de réduire le parcours des informations depuis $(x,y)^0$, r_0 à S^3.

Pour un bloc (bloc 0), le chemin $(x,y)^0$, $r_0 \rightarrow S^0$, donné par la figure II.20 et traduit en circuit logique (Fig. II.21), va nous permettre, par des transformations successives, de dégager les degrés de réduction du temps de calcul et d'augmentation de complexité.

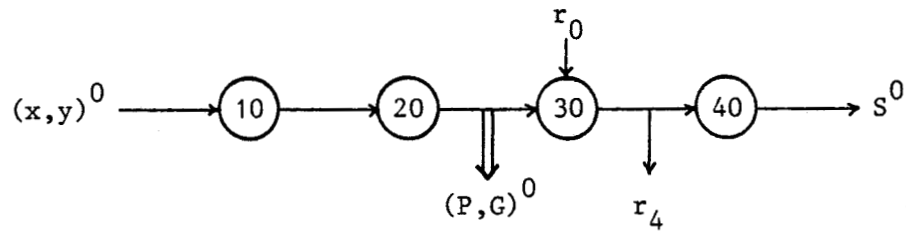


Figure II.20

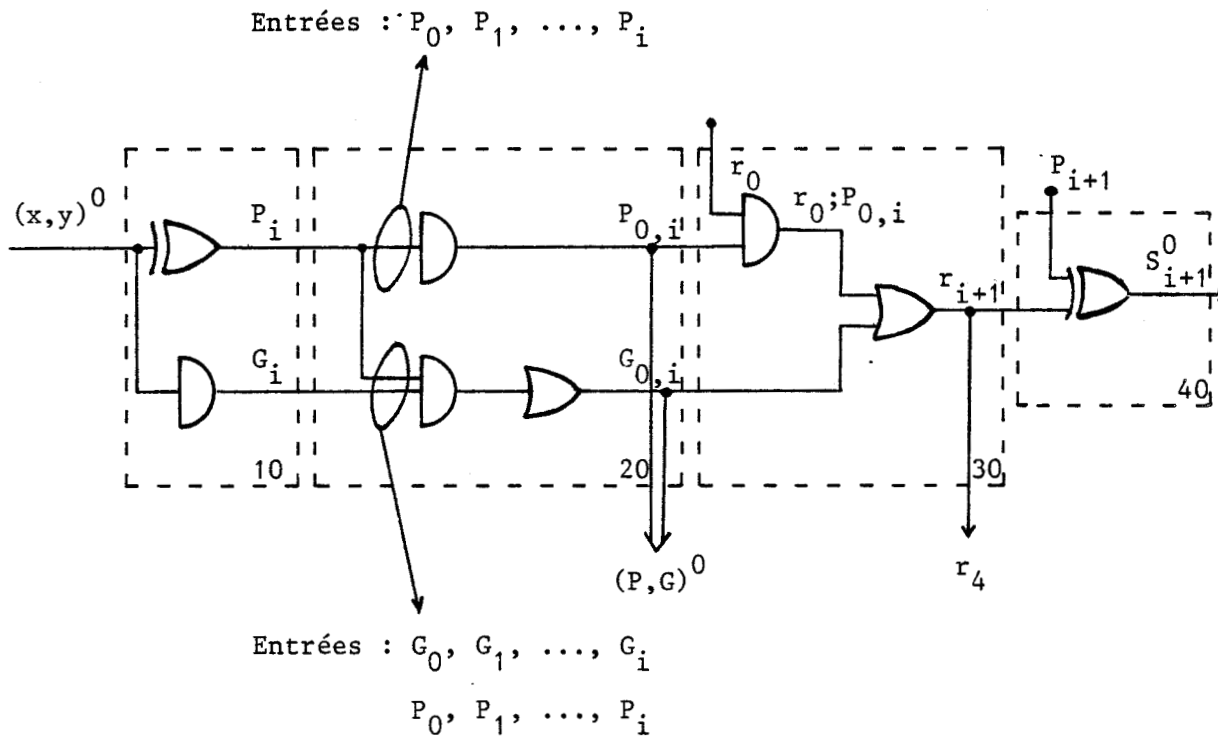
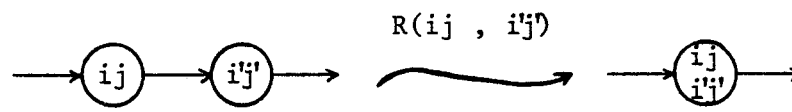


Figure II.21

Le temps de calcul de r_4 est ici de 5 couches logiques (la porte "OU" exclusif est considérée comme équivalente à 2 couches logiques).

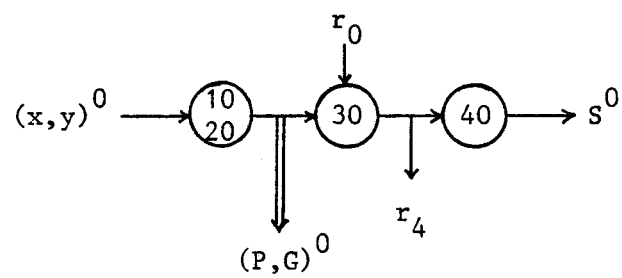
Le principe général de réduction du temps de calcul consiste à grouper plusieurs circuits de type —(ij)— en un seul. Ceci est permis par les propriétés de commutativité, d'associativité et de distributivité des opérations "ET", "OU" logiques.

Notons R(i j , i'j'), l'opération de groupement des circuits →(ij)→ et →(i'j')→ définie par :



- Transformation : R(10,20) :

R(10,20) appliquée à la figure II.20 donne :



dont le schéma logique, qui est l'application de R(10,20) à la figure II.21, est le suivant :

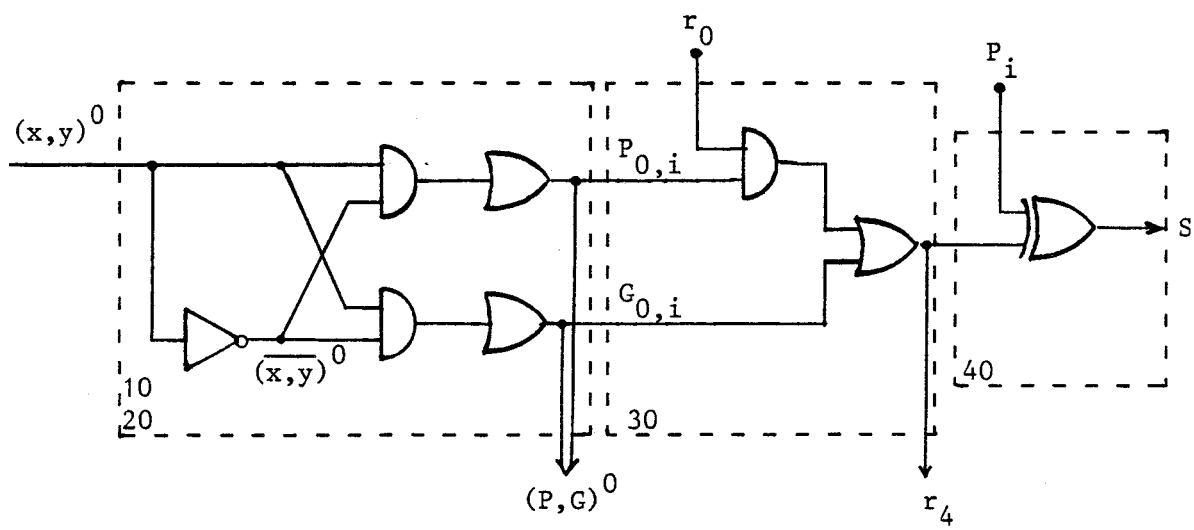


Figure II.22

$(\overline{x,y})^0$ est le complément booléen de $(x,y)^0$.

Ici aussi, le temps de calcul de r_4 est de 5 couches logiques et peut être réduit d'une couche logique si on utilise la paire (G_k , T_k) au lieu de (G_k , P_k) auquel cas nous avons la figure II.23 :

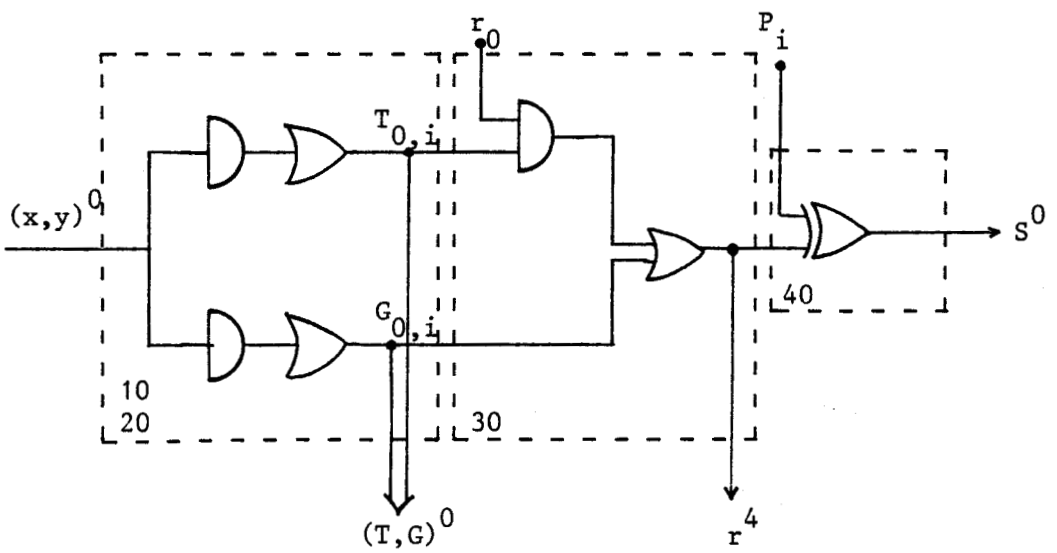


Figure II.23

Avant l'opération R(10,20), le nombre de portes nécessaires à l'élaboration des $P_{0,i}$ et $G_{0,i}$ (i = 0, 1, 2, 3) était de :

- 4 portes "OU" exclusif
- 3 portes "OU"
- 13 portes "ET".

Après l'opération R(10,20), il est de :

- 4 portes "OU" exclusif (calcul de P_i pour le circuit 40)
- 7 portes "OU" jusqu'à 16 entrées
- 44 portes "ET" à 5 entrées maximum.

Le temps de calcul global $(x,y)^0$, $r_0 \rightarrow S^3$ est donné par la table II.1 pour les deux types de connexion, avant et après l'opération R(1i,2i).

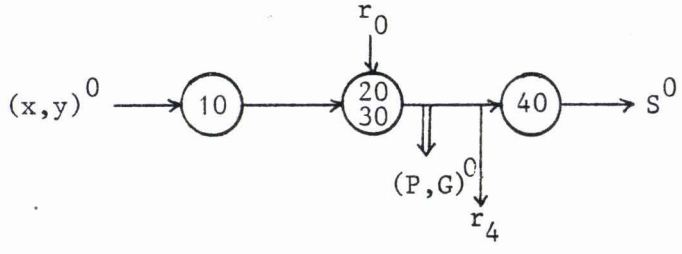
	Avant l'opération R(1i,2i)	Après l'opération R(1i,2i)
Connexion cascade	13 couches logiques	12 couches logiques
Connexion cascade et parallèle	10 couches logiques	8 couches logiques

Table II.1

Remarque : Il n'est pas nécessaire d'appliquer l'opération R(1i,2i) pour les blocs B_1, B_2 et B_3 dans le cas de la connexion cascade car, seule r_4 est responsable de la réduction du temps de calcul d'une couche logique. En effet, r_4 est calculée en 4 couches logiques après R(10,20) et va être combiné à $(P,G)^1$ qui est aussi calculée en 4 couches logiques avant l'opération R(11,21).

- Transformation R(20,30) :

R(20,30) appliquée à la figure II.20 donne :



dont le schéma logique, qui est l'application de R(20,30) au circuit de la figure II.21, est le suivant (Fig. II.24) :

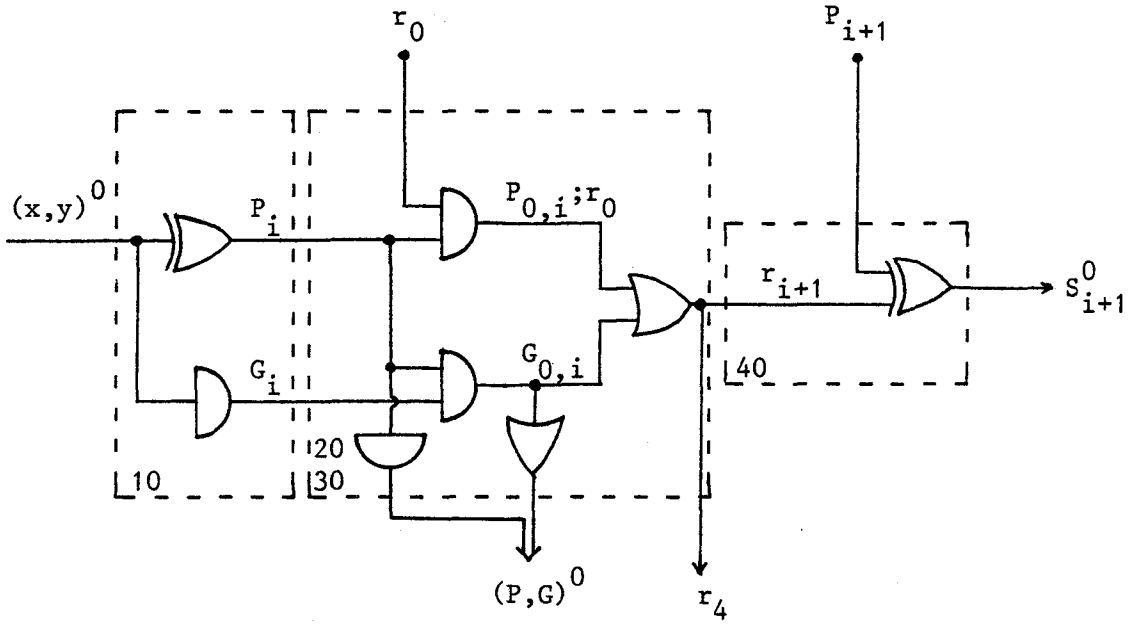


Figure II.24

La table II.2 donne le temps de calcul global et le nombre de portes nécessaires à l'élaboration des retenues r_i et de la paire de paramètres $(P,G)^i$ à partir des P_i, G_i pour les deux types de connexions, avant et après l'opération R(2i,3i).

	Avant R(2i,3i)		Après R(2i,3i)	
	cascade	cascade et parallèle	cascade	cascade et parallèle
Temps de calcul global en couches logiques	13	10	12	10
Nombre de portes "OU"	7	7	5	5
Nombre de portes "ET"	13	13	11	11

Table II.2

La table II.2 montre que l'opération R(2i,3i) réduit le nombre de portes logiques de 4 et le temps de calcul d'une couche logique pour la connexion cascade, mais pas pour la connexion cascade et parallèle qui reste le même (10 couches logiques).

- Transformation R(30,40) :

La relation $r_{i+1} = G_{0,i} + P_{0,i} \cdot r_0$ peut aussi s'écrire :

$$r_{i+1} = G_{0,i} \oplus P_{0,i} \cdot r_0 \qquad \text{et}$$

$$\begin{aligned} S_i &= P_i \oplus r_i \\ &= P_i \oplus (G_{0,i} \oplus P_{0,i-1} \cdot r_0) \\ &= P_i \oplus G_{0,i-1} \oplus P_{0,i-1} \cdot r_0 \end{aligned}$$

ce qui permet la transformation, par l'application de R(30,40), de la figure II.21 en la figure II.25 :

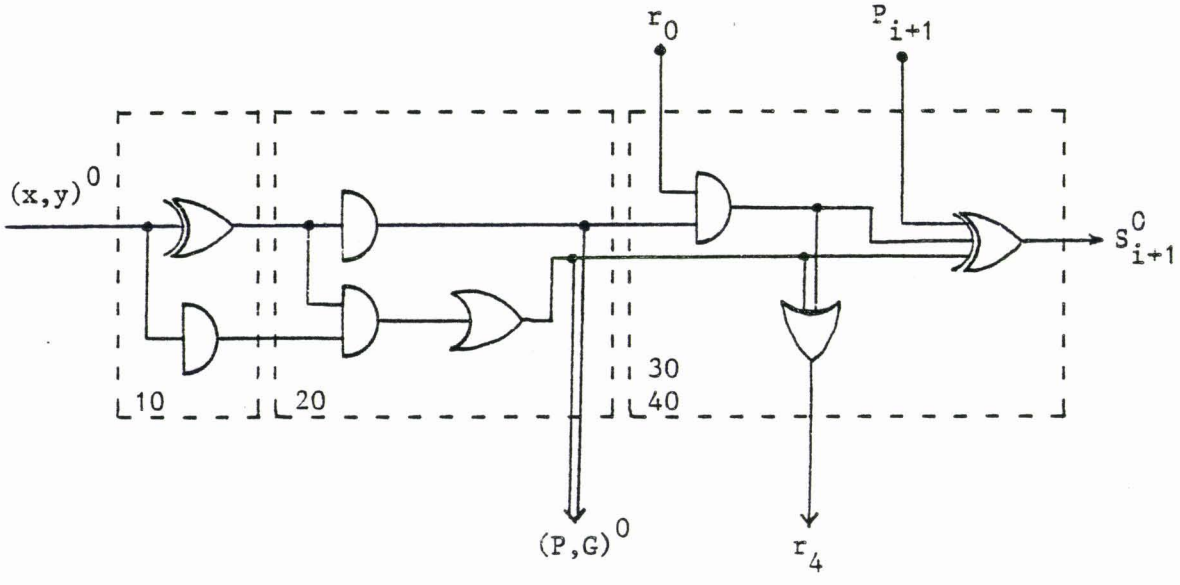


Figure II.25

Ici, le temps de calcul est de 10 couches logiques pour la connexion cascade et de 9 couches logiques pour la connexion cascade et parallèle. Le nombre de portes logiques est réduit de 4 pour un bloc, par l'utilisation de portes "OU" exclusif à 3 entrées.

Jusqu'ici, nous ne nous sommes intéressés qu'à la réduction du parcours $(x,y)^i \rightarrow S^i$. Une autre façon de réduire le temps de calcul consiste à augmenter la longueur des blocs. Cette solution n'apporte rien au cas de la connexion cascade et parallèle. En effet, pour des blocs de longueur 8 (c.à.d. le double des blocs étudiés), nous avons :

- Pour le cas de connexion cascade et parallèle :

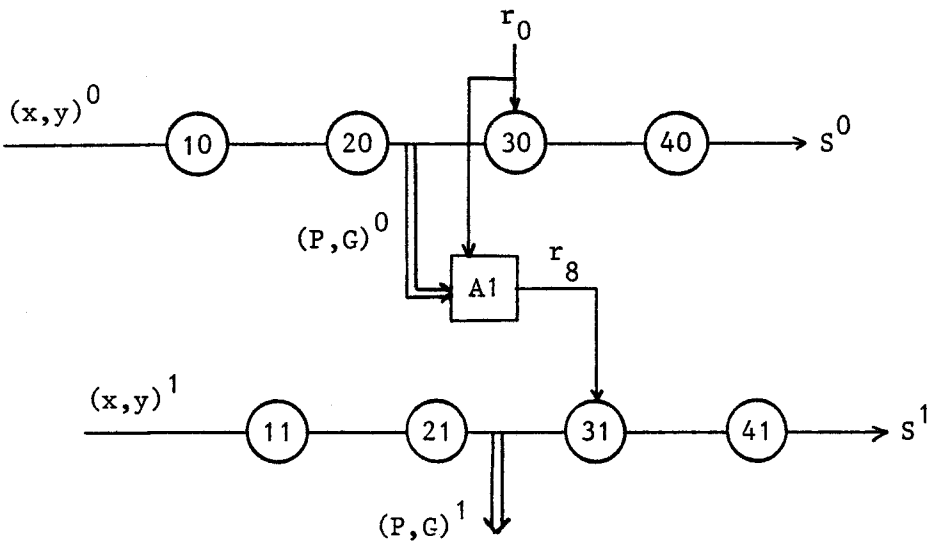


Figure II.26

- Pour le cas de connexion cascade :

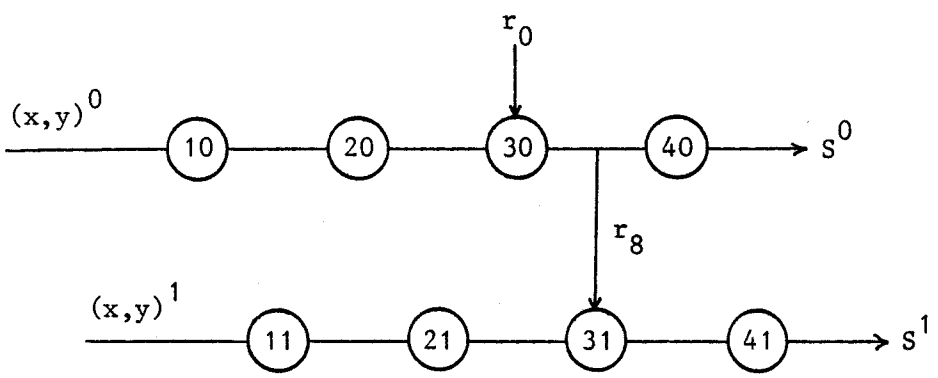
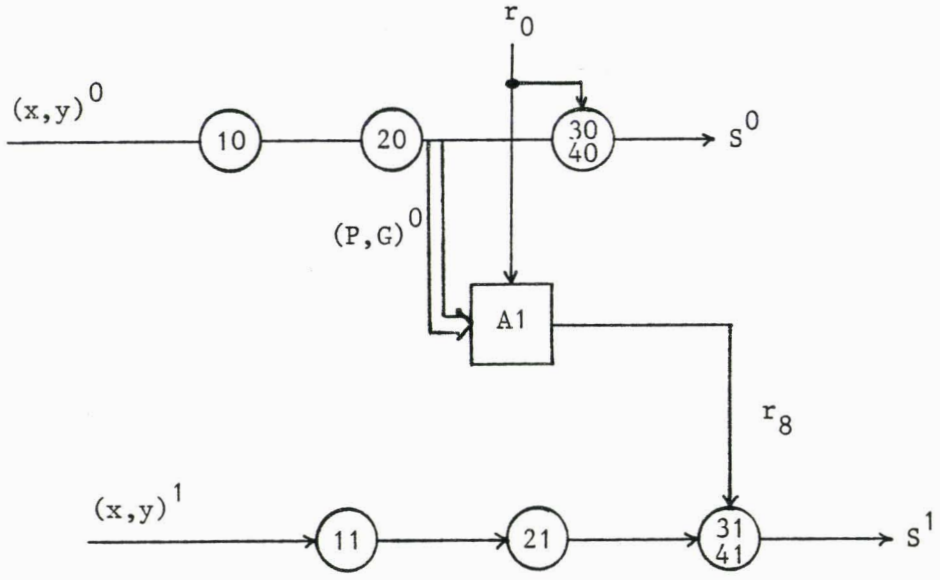


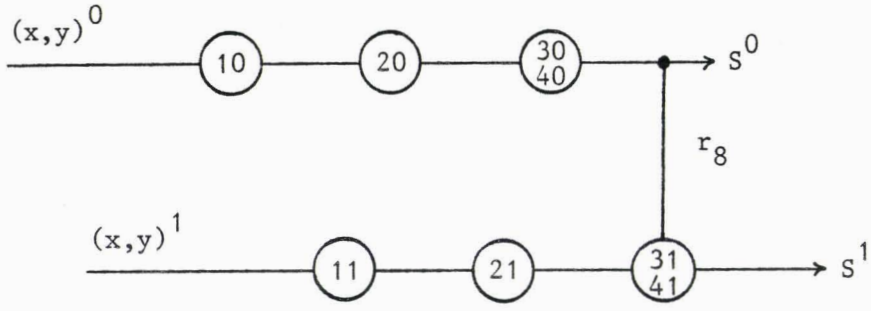
Figure II.27

Les figures II.26 et II.27 montrent que le temps de calcul est le même pour les deux types de connexion, c'est à dire de 10 couches logiques. Ce temps de calcul peut être réduit d'une couche logique par l'application de R(3i,4i) aux figures II.26 et II.27 qui deviennent sous l'effet de R(3i,4i) :



CONNEXION CASCADE ET PARALLELE

Figure II.28



CONNEXION CASCADE

Figure II.29

Il est aussi possible de réduire le temps de calcul global d'une couche logique dans le cas de la connexion cascade et parallèle, par le groupement des circuits de type —[Ai]— et —(3i/4i)— ou par l'application de l'opération $R(Ai, \frac{3i}{4i})$.

En raison des contraintes technologiques (entrance - sortance des portes logiques), les transformations que nous venons de présenter ne sont pas toutes possibles pour n'importe quelle technologie utilisée.

II.2.1.3 - Réalisation matérielle à l'aide de portes logiques élémentaires

a) Matériel disponible |23| :

La technologie que nous proposons d'utiliser, en l'occurrence la TTL, ne peut être un choix définitif ; néanmoins, elle servira de support pour évaluer le temps de calcul et la complexité de l'additionneur à base de portes élémentaires. Ce choix est aussi fait pour des raisons technologiques : compatibilité avec plusieurs sortes de microprocesseurs.

Par ailleurs, la technologie TTL nous permet de disposer d'une grande diversité de portes logiques et ne pose pas trop de problèmes pour la mise en œuvre.

Actuellement, en technologie TTL, on dispose des circuits suivants :

- Portes ET à 2, 3 et 4 entrées
- Portes OU à 2 entrées
- Portes ET Non (NAND) à 2, 3, 4, 8 et 13 entrées
- Portes OU Non (NOR) à 2, 3 et 5 entrées
- Portes OU exclusif à 2 et 3 entrées.

Les deux cas de figures intéressants au point de vue temps de calcul ont été décrits sur les figures II.23 et II.29, ils permettent un temps de calcul global de 8 couches logiques.

- Cas de la figure II.23 :

Nous avons vu que cette structure nécessite des portes OU à 16 entrées. Avec le matériel disponible, on peut réaliser des fonctions OU à 13 entrées seulement en utilisant les portes Nand et en exprimant une fonction F :

$$F = \sum_i \prod_j \alpha_j$$

Comme :

$$F = \overline{\overline{\sum_i \prod_j \alpha_j}} = \overline{\prod_i \overline{\prod_j \alpha_j}} \quad \text{et} \quad \overline{\prod_j \alpha_j} = \text{Nand}_j\ \alpha_j$$

$$F = \overline{\prod_i (\text{Nand } \alpha_j)_j} = \text{Nand}_i (\text{Nand}_j \alpha_j)_j .$$

Pour pouvoir utiliser la transformation R(1i,2i), il faut prendre des blocs de longueur inférieure à 4 avec la possibilité de réaliser tous les circuits Ai en deux couches logiques.

Avec des blocs de longueur 3 et un mot binaire de 16 bits, il faut 5 circuits de type Ai (c.à.d. 6 blocs).

Un circuit de type Ai nécessite une porte OU de (i+1) entrées et (i+1) portes ET de (1, 2, 3, ..., i+1) entrées pour réaliser les retenues entrantes des blocs.

Il est donc possible de réaliser A_5 à l'aide du matériel disponible.

. Pour le bloc 0 :

$$(T_{0,i})^0 \begin{cases} T_{0,0} = T_0 = x_0 + y_0 \\ T_{0,1} = T_0 T_1 = (x_0 + y_0)(x_1 + y_1) = x_0x_1 + x_0y_1 + y_0x_1 + y_0y_1 \\ T_{0,2} = T_0T_1T_2 = x_0x_1x_2 + x_0y_1x_2 + y_0x_1x_2 + y_0y_1x_2 + x_0x_1y_2 \\ \qquad\qquad + x_0y_1y_2 + y_0x_1y_2 + y_0y_1y_2 \end{cases}$$

$$(G_{0,i})^0 \begin{cases} G_{0,0} = G_0 = x_0y_0 \\ G_{0,1} = G_1 + T_1G_0 = x_1y_1 + (x_1 + y_1)\, x_0y_0 \\ \qquad\qquad = x_1y_1 + x_1x_0y_0 + y_1x_0y_0 \\ G_{0,2} = G_2 + T_2G_1 + T_2T_1G_0 = x_2y_2 + (x_2 + y_2)\, x_1y_1 \\ \qquad\qquad + (x_2 + y_2)\,(x_1 + y_1)\, x_0y_0 \end{cases}$$

$$= x_2y_2 + x_2x_1y_1 + y_2x_1y_1 + x_2x_1x_0y_0 + x_2y_1x_0y_0 + y_2x_1x_0y_0 + y_2y_1x_0y_0$$

Ces équations sont identiques pour tous les blocs B_i avec des indices correspondants :

$$3i + j \qquad (i = 0, 1, 2, 3, 4, 5 \; ; \; j = 0, 1, 2)$$

La transformation R(3i,4i) peut aussi être appliquée et porter à

7 couches logiques le temps de calcul global de l'addition de deux mots binaires de 16 bits.

La table II.3 donne le nombre de portes logiques nécessaires à l'élaboration de tous les bits somme de deux mots binaires de 16 bits, avec sortie de signaux Sign (X + Y) et deb (X + Y).

- Cas de la figure II.29 :

Le temps de calcul global, selon la figure II.29 où le mot à traiter est partitionné en deux blocs de longueur 8, est de 9 couches logiques. Il est possible de reduire ce temps de deux couches logiques par l'application de R(2i,3i) au 1er bloc et R(3i,4i) au 2ème bloc (Fig. II.30) :

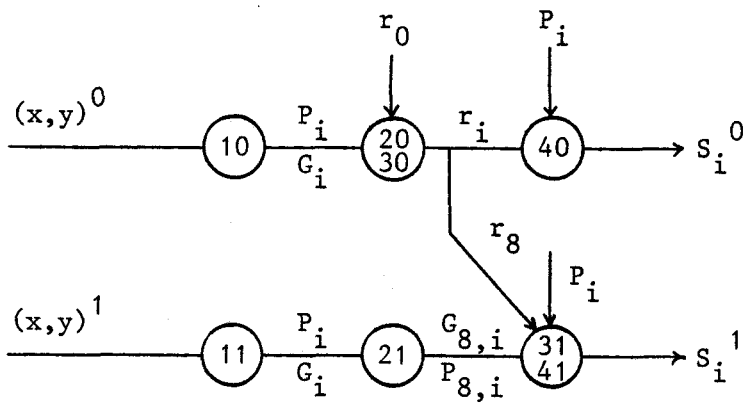


Figure II.30

Cependant, pour respecter l'entrance de la TTL, il faut tantôt prendre la paire (G_k,P_k), tantôt la paire (G_k,T_k) pour le calcul des $G_{0,i}$, $T_{0,i}$ ou $P_{0,i}$ et r_i.

Ainsi, pour tout (x_i,y_i), les variables G_i, $\overline{G_i}$, T_i, P_i sont calculées selon le schéma suivant (Fig. II.31) :

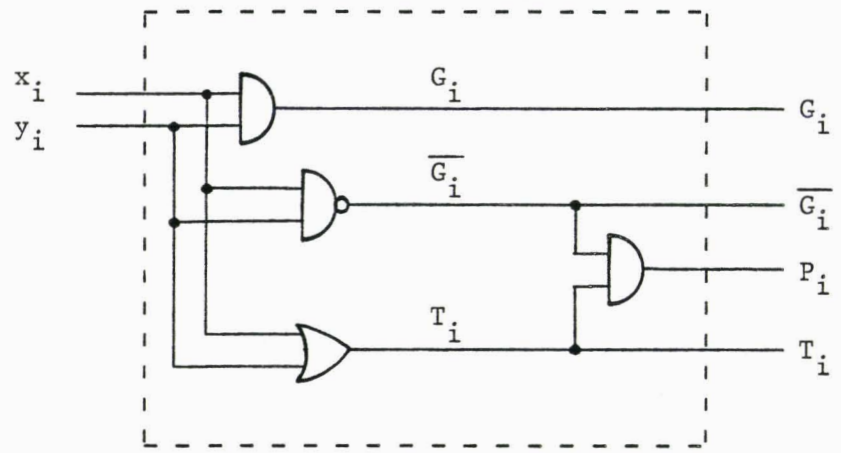


Figure II.31

Pour le bloc 0 (calcul des r_i, S_i ; i=0,1,...,8) :

On calcule toutes les retenues r_i (i = 1, 2, 8) et ensuite vient l'assimilation (c.à.d. le calcul des S_i (i = 0, 1, ..., 7) par les équations logiques :

pour les digits sommes : $S_i = P_i \oplus r_i$

pour les retenues : $r_{i+1} = G_{0,i} + P_{0,i}\, r_0$

$r_{i+1} = G_{0,i} + T_{0,i}\, r_0$

Les r_i seront exprimées sous forme explicite en fonction des G_i, T_i ou P_i et $\overline{G_i}$ pour être réalisées par des portes Nand. Ainsi, nous avons :

$$r_0 = r_0$$

$$r_1 = \overline{\overline{G_0}.\overline{P_0.r_0}}$$

$$r_2 = \overline{\overline{G_1}.\overline{P_1G_0}.\overline{P_0P_1r_0}}$$

$$r_3 = \overline{\overline{G_2}.\overline{P_2G_1}.\overline{P_2P_1G_0}.\overline{P_2P_1P_0r_0}}$$

$$\vdots$$

$$r_8 = \overline{\overline{G_7}.\overline{P_7G_6}.\overline{P_7P_6G_5}.\overline{P_7P_6P_5G_4}.\overline{P_7P_6P_5P_4G_3}.\overline{P_7P_6P_5P_4P_3G_2}.\overline{P_7P_6P_5P_4P_3P_2G_1}.\overline{\overline{P_7P_6P_5P_4P_3P_2P_1G_0}.\overline{P_7P_6P_5P_4P_3P_2P_1P_0r_0}}}$$

Pour le bloc 1 :

On calcule d'abord les $G_{8,i}$ et $P_{8,i}\,r_8$ sous forme explicite par des portes Nand, en fonction des G_i, T_i ou P_i et $\overline{G_i}$ et enfin, l'assimilation se fait par une porte OU exclusif à 3 entrées qui calcule :

$$S_i = P_i \oplus G_{8,i} \oplus P_{8,i}\,r_8$$

La table II.4 donne le nombre de portes nécessaires au calcul de tout S_i.

Les deux tables (II.3 et II.4), qui mettent en évidence les complexités des structures considérées comme les plus rapides et de même vitesse, ont été établies pour des mots de 16 bits et avec la même technologie (la TTL).

Il est évident que notre choix se portera sur la structure la moins complexe, à savoir le cas de la figure II.30.

Si les mots à traiter sont de longueurs supérieures à 16, la structure cascade et parallèle est visiblement la plus intéressante, en ce qui concerne le facteur temps.

	TYPE DE PORTES LOGIQUES								
	ET 2 entrées	OU 2 entrées	Nand 2 entrées	Nand 3 entrées	Nand 4 entrées	Nand 8 entrées	OU exclusif 3 entrées	Total des portes logiques	Temps de calcul en couches logiques
Nombre de chaque type	26	16	46	72	28	15	16	219	7

MATERIEL ET TEMPS DE CALCUL POUR L'ADDITIONNEUR "CONNEXION CASCADE ET PARALLELE" 16 BITS

Table II.3

	TYPE DE PORTES LOGIQUES									
	ET 2 entrées	OU 2 entrées	Nand 2 entrées	Nand 3 entrées	Nand 4 entrées	Nand 8 entrées	Nand 13 entrées	OU exclusif 2 entrées	OU exclusif 3 entrées	Total des portes logiques	Temps de calcul en couches logiques
Nombre de chaque type	32	16	34	16	18	31	4	9	7	167	7

MATERIEL ET TEMPS DE CALCUL POUR L'ADDITIONNEUR "CONNEXION CASCADE" 16 BITS

Table II.4

II.2.1.4 - Réalisation matérielle à l'aide de réseaux logiques programmables

Les réseaux logiques ont été introduits par signetics et MMI, pour remplacer les circuits MSI ou SSI traditionnels dans les équipements où l'approche microprogrammée est souvent rejetée à cause de sa lenteur. Nous estimons utile et logique d'examiner cette alternative, vue la complexité d'un additionneur 16 bits à base de circuits SSI, exprimée par les tables II.3 et II.4 (environ 200 portes).

a) Structure des réseaux logiques programmables :

Il existe plusieurs types de réseaux logiques programmables. La figure II.32 illustre leur structure générale.

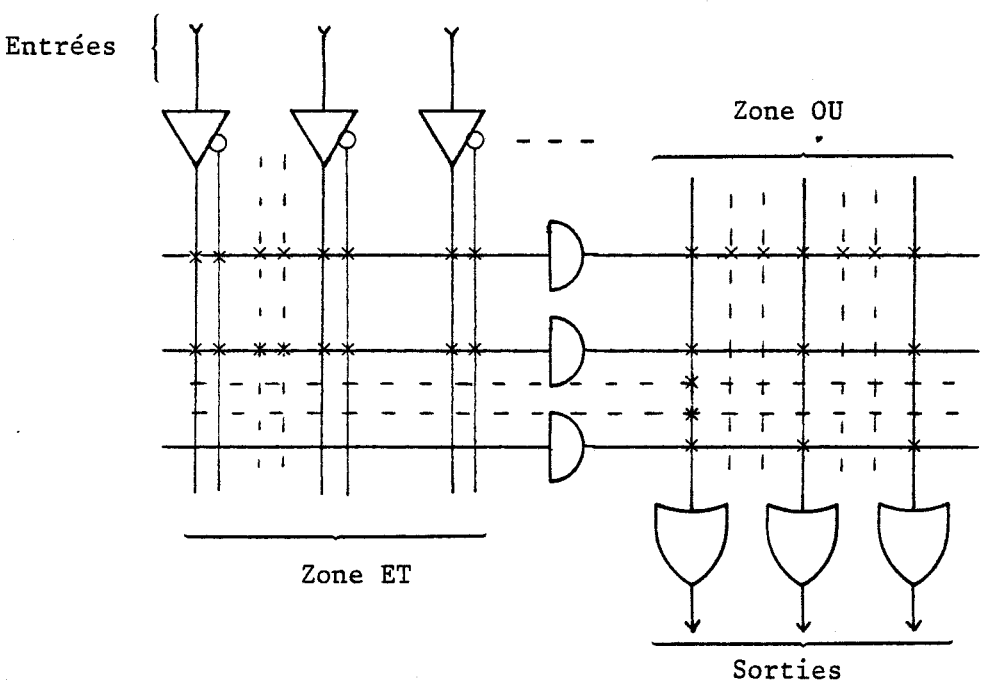


Figure II.32

× : représente un fusible programmable ou non selon le type de réseaux.

Exemples : La zone ET est figée pour les PROM
La zone OU est figée pour les PAL
Les deux zones sont programmables pour les PLA, FPLA, FPLS.

Après avoir fait le tour d'horizon d'une grande partie de ces réseaux (voir Annexe II), nous avons retenu le réseau FPLA de type 16 × 48 × 8, c'est à dire 16 entrées, 8 sorties et des portes OU à 48 entrées maximum.

b) Schémas de réalisation d'un additionneur 16 bits :

Nous avons établi, dans un résultat antérieur, que pour élaborer le bit somme S_i de rang i d'une façon explicite, à partir des variables x_j, y_j et r_0 (j = 0, 1, 2, ..., i), il faut une porte "OU" ayant :

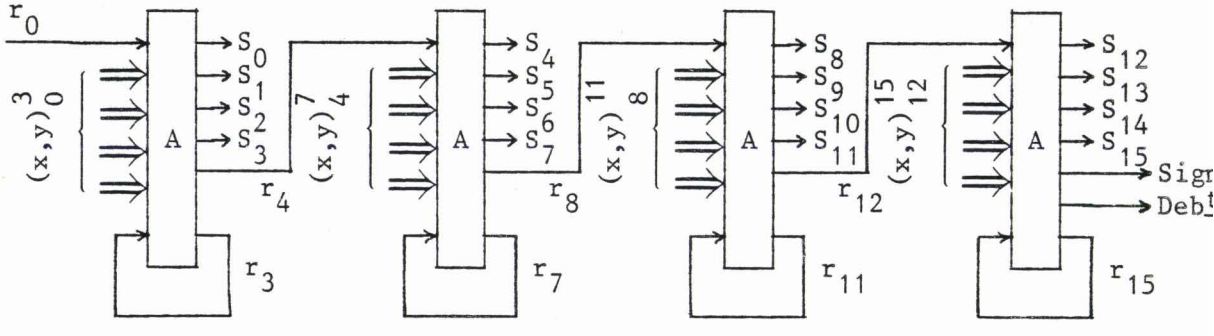
$$\sum_{j=0}^{i} 2^{j+2} \text{ entrées} \quad \text{et} \quad \text{autant de portes "ET".}$$

Il faut également des portes "ET" et une porte "OU" ayant :

$$\sum_{j=0}^{i+1} 2^{j} \text{ entrées pour élaborer la retenue sortante } r_{i+1}.$$

A l'aide du FPLA 16 × 48 × 8, choisi à cause du nombre 48 (nombre d'entrées des portes OU) qui est le plus élevé par rapport aux autres types de réseaux, il est possible d'exprimer d'une façon explicite, les bits sommes : S_0, S_1, S_2 et les retenues r_1, r_2, r_3 et r_4.

Selon la structure cascade (Fig. II.14) et les considérations ci-dessus, un premier schéma possible est donné par la figure II.33.



A : FPLA 16 × 48 × 8

Figure II.33

Notations :

$$\begin{cases} (x,y)_i^j = ((x_i,y_i)\ ,\ (x_{i+1},y_{i+1})\ ,\ \ldots\ ,\ (x_j,y_j)) \\ (P,G)_i^j = (P_{i,j}\ ,\ G_{i,j}) \end{cases}$$

Ici, avec 4 FPLA, le temps de calcul de tous les bits sommes S_i (i = 0, 1, ..., 15) est de 5 fois le temps de propagation d'un FPLA.

On peut remarquer que les entrées et les sorties ne sont pas toutes utilisées. Il est possible de réduire le nombre des FPLA d'une unité, par l'élaboration de 6 bits sommes au lieu de 3 au niveau de chaque réseau. D'où le schéma de la figure II.34.

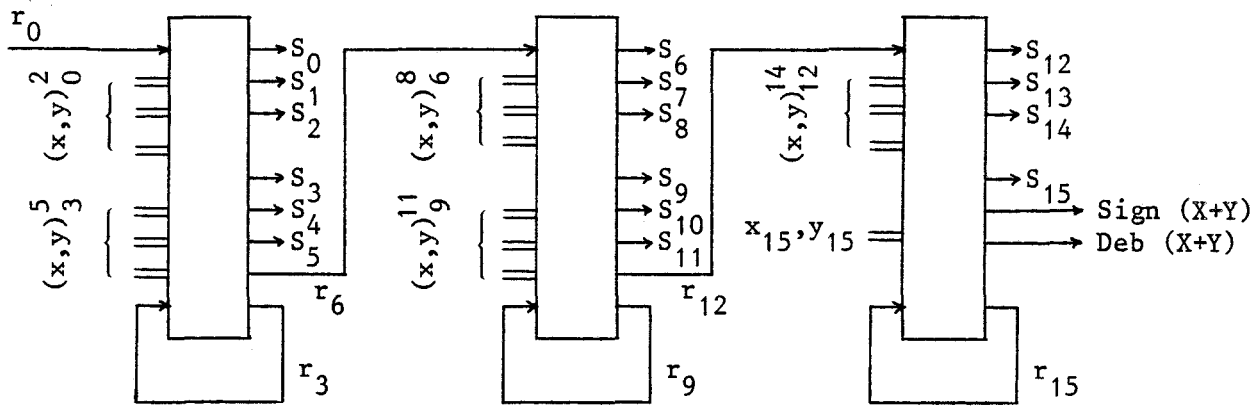


Figure II.34

Ici, avec 3 FPLA, nous avons un additionneur 16 bits avec un temps de calcul égal à 6 fois le temps de propagation d'un FPLA.

Les deux schémas présentés par les figures II.33 et II.34 sont de type cascade. La structure cascade et parallèle est aussi envisageable.

En raison des contraintes technologiques des FPLA mentionnés plus haut, la longueur des blocs ne peut être supérieure à 3, tout au moins au niveau de l'assimilation, avec un calcul préalable des retenues entrantes des blocs. La figure II.35 décrit ce cas pour un mot de 16 bits.

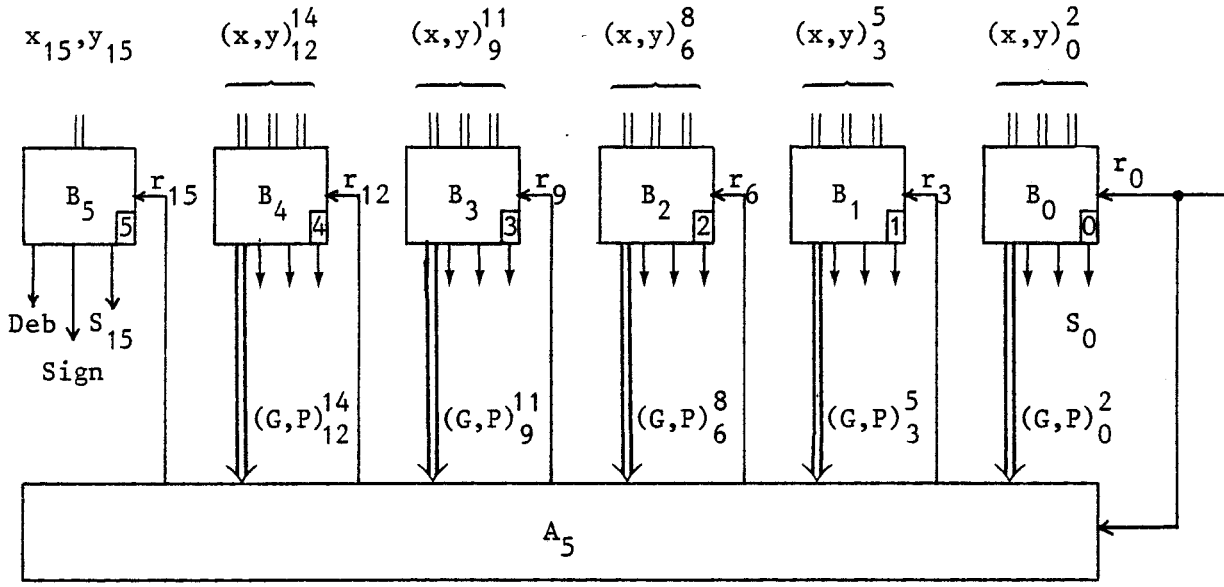


Figure II.35

Si chaque bloc B_i (i = 0, 1, 2, 3, 4, 5) et le réseau A_5 est matérialisé par un FPLA, nous obtenons, selon la figure II.35, un additionneur 16 bits, réalisé à l'aide de 7 FPLA, avec un temps de calcul de 3 fois le temps de propagation d'un FPLA.

Vu la structure des FPLA, il nous paraît difficile de réduire la vitesse de calcul à moins de 3 couches FPLA ; par contre, il est possible de réduire le nombre de réseaux. Nous avons étudié plusieurs configurations, et nous sommes arrivés à réduire à 4 le nombre de réseaux; avec un temps de calcul égal à 4 fois le temps de propagation d'un FPLA, et à 5 puis à 4 réseaux, avec un temps de calcul égal à 3 fois le temps de propagation d'un FPLA.

Les figures II.36, II.37 et II.38 donnent ces 3 configurations.

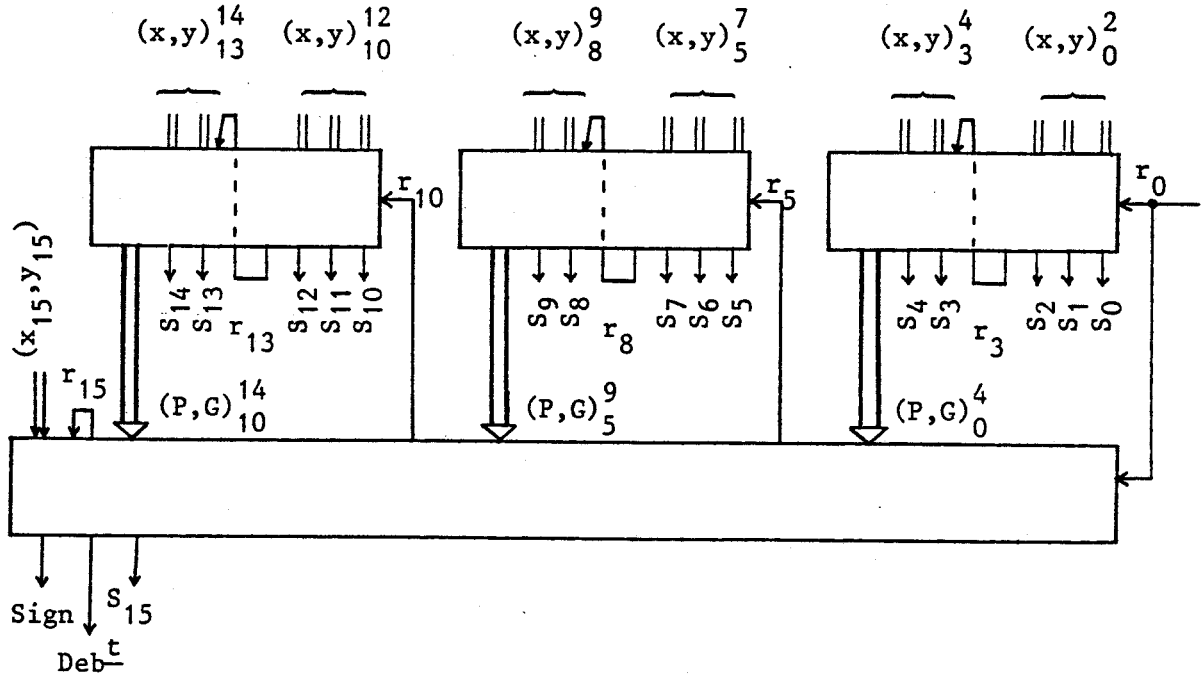


Figure II.36

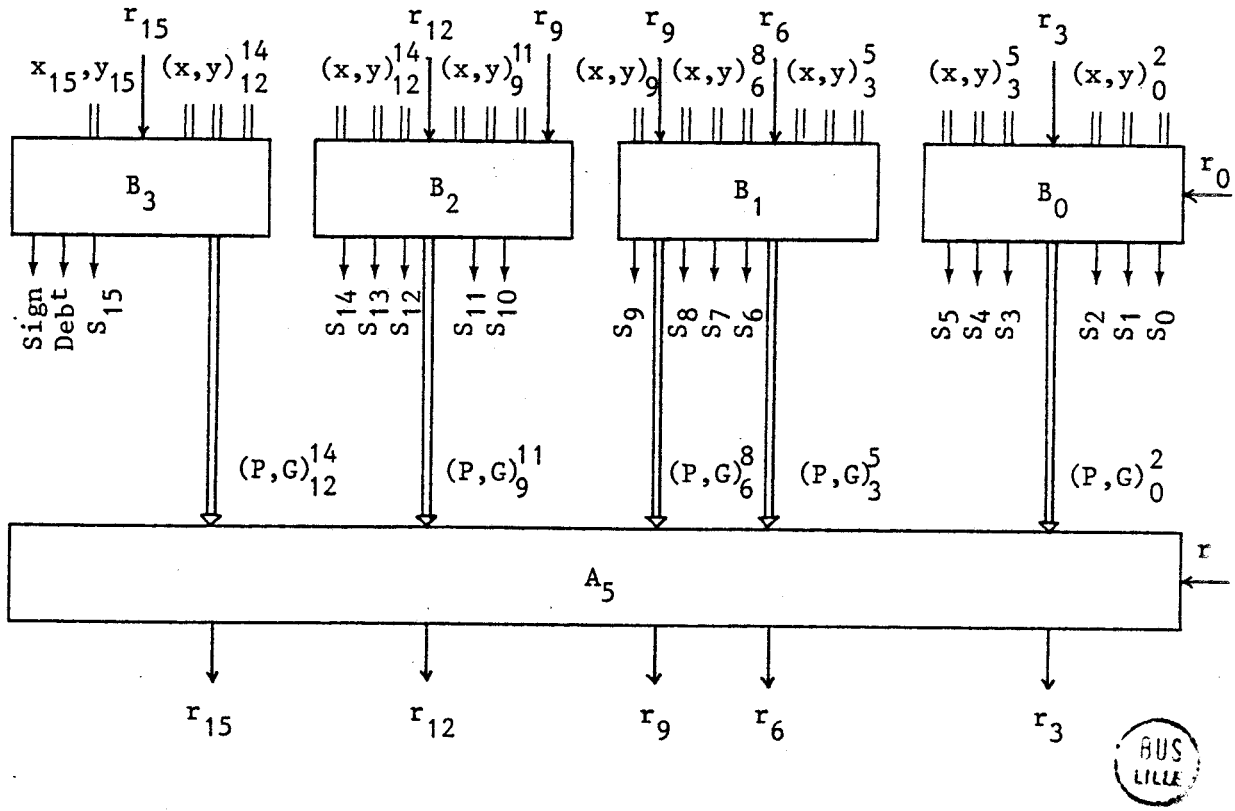


Figure II.37

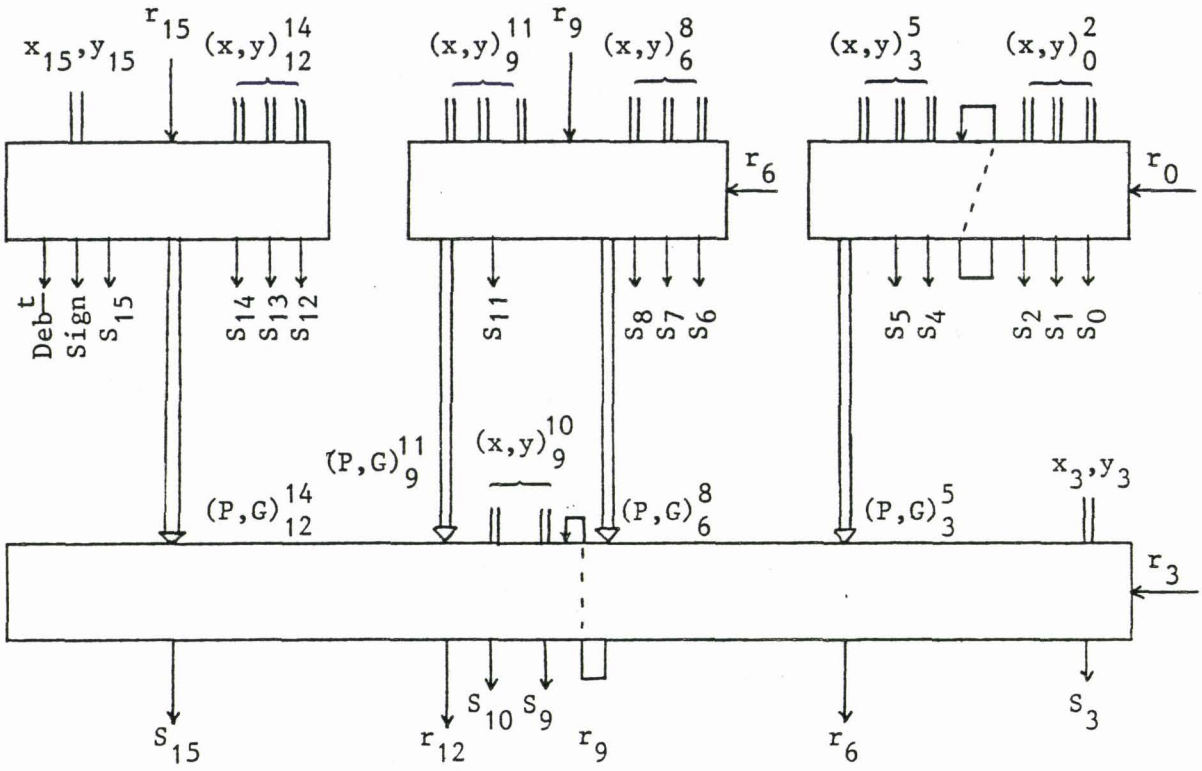


Figure II.38

II.2.2 - Le multiplieur

Il existe deux grandes classes de multiplieurs : les multiplieurs séquentiels et les multiplieurs parallèles.

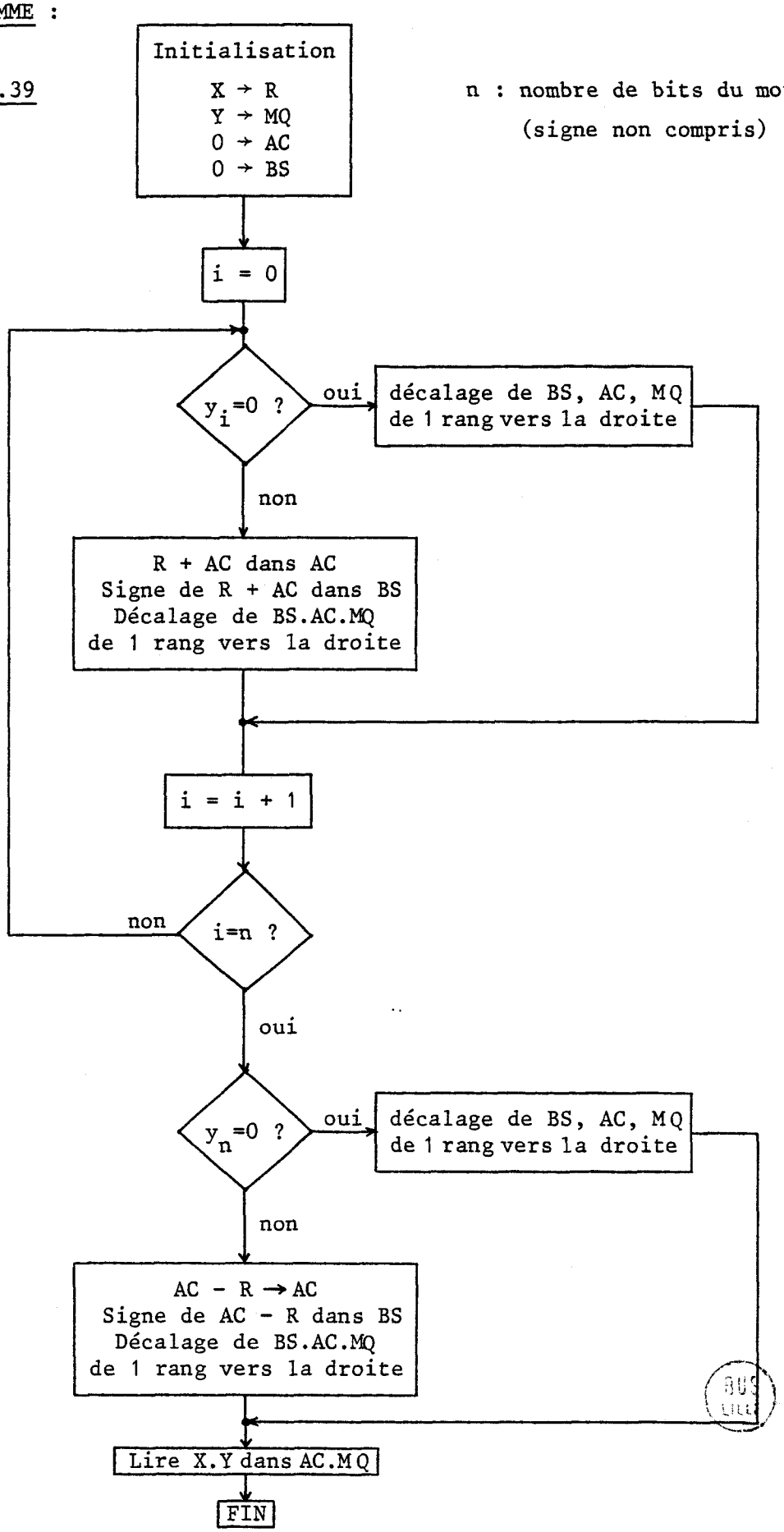
II.2.2.1 - Multiplieurs séquentiels

Contrairement au calcul manuel, le principe de la multiplication séquentielle consiste à ramener l'addition sur deux nombres de longueur n+1, selon l'algorithme d'addition décalage. Le décalage porte sur le premier produit partiel d'un rang vers la droite, plutôt que sur le deuxième produit partiel d'un rang vers la gauche.

L'addition ordinaire peut déborder, il convient donc de prévoir une bascule de débordement, qui en fait, contiendra le bit signe de l'addition. La figure II.39 donne l'organigramme du déroulement d'une multiplication séquentielle, et les figures II.40 et II.41 décrivent deux exemples de multiplication de deux nombres binaires de 5 bits (signes compris).

ORGANIGRAMME :

Figure II.39



n : nombre de bits du mot Y
(signe non compris)

1er Exemple : X = 1 0 1 1 0 = - 1 0 X < 0

Y = 0 1 0 0 1 = 9 Y > 0

X Y = 1 1 1 0 1 0 0 1 1 0 = - 9 0

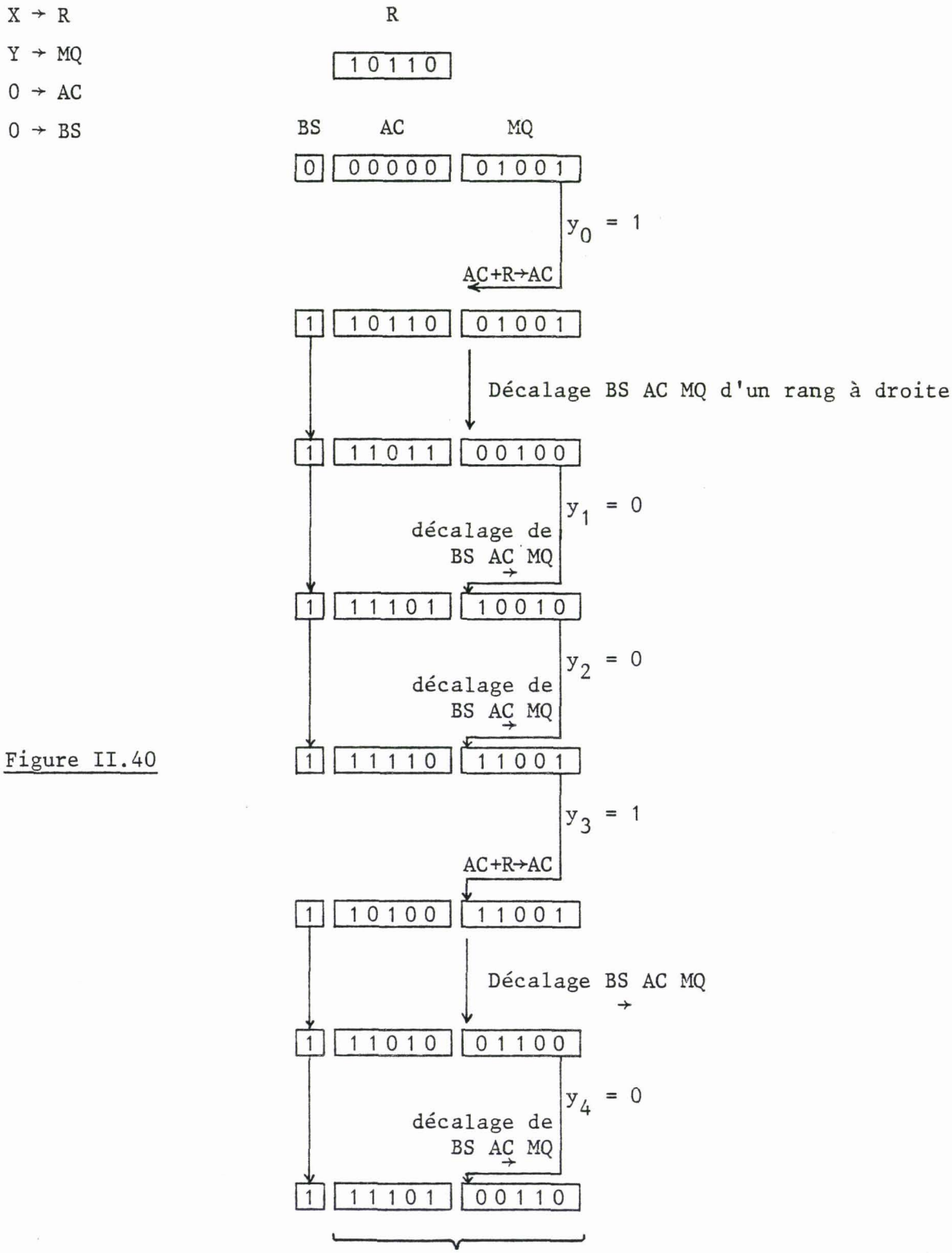


Figure II.40

2ème Exemple : X = 1 0 1 1 0 = - 1 0

Y = 1 1 0 1 0 = - 6

X Y = 0 0 0 0 1 1 1 1 0 0 = + 6 0

X → R

Y → MQ

0 → AC

0 → BS

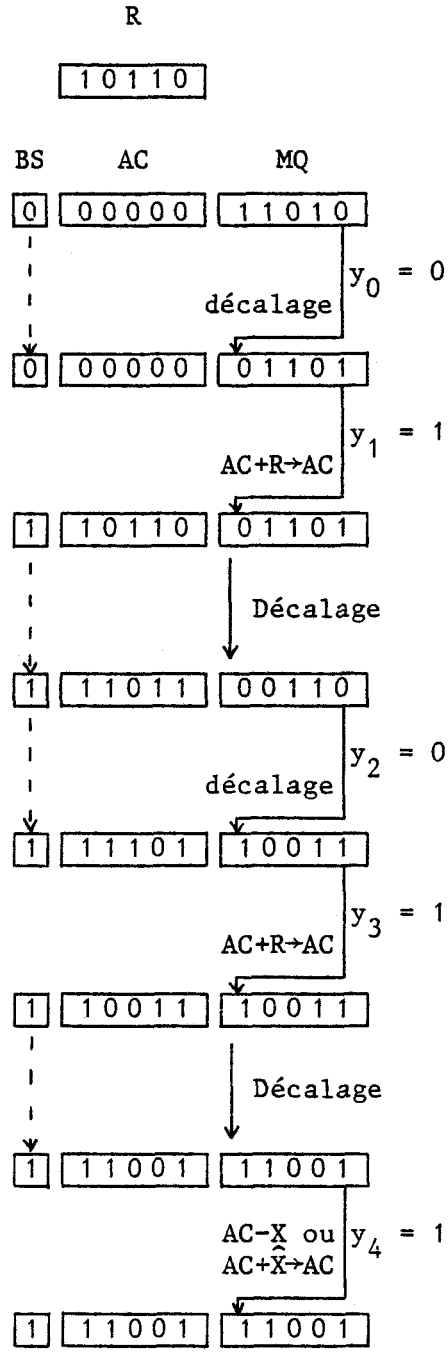
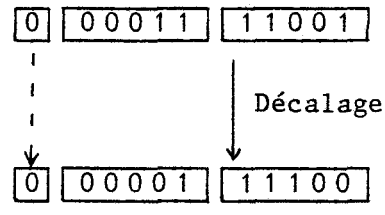


Figure II.41

$01010\hat{X}^+$



Résultat du produit X Y

Le multiplicande X est rangé dans un registre R durant tout le déroulement de la multiplication.

Le multiplieur Y est rangé dans un registre MQ. Un registre AC ou accumulateur, initialement à zéro, est nécessaire pour ranger les additions cumulées des produits partiels. Une bascule de signe BS, portera la longueur de l'accumulateur à n+2 (n = longueur des mots à traiter, signes non compris).

Le résultat du produit XY, se trouve en fin de calcul, dans le double registre AC : MQ.

Le temps de calcul d'une multiplication effectuée selon l'algorithme que nous venons de décrire, peut être estimé à (n+1) fois le temps d'addition et (n+1) fois le temps de décalage.

Il est clair qu'une multiplication fondée sur cet algorithme est très lente, dans un système où nous parlons d'opérateurs rapides. Il est donc nécessaire de chercher à améliorer le temps de multiplication.

II.2.2.2 - Amélioration du temps de calcul de la multiplication

Dans le cas où la multiplication est toujours basée sur l'analyse du multiplicateur, le principe général pour accélérer la multiplication consiste à réduire le nombre d'additions effectives ($\neq +0$) par un recodage approprié du multiplicateur.

Les recodages les plus utilisés sont celui de Booth et celui de Wallace.

a) Recodage de Booth :

Soit le multiplicateur $Y = y_n\ y_{n-1} \cdots y_i \cdots y_0$

$y_i \in \{0\ ,\ 1\}$ pour $i = 0, 1, \ldots, n$
y_n : bit de signe.

Le principe consiste à recoder Y en $Y' = y'_n\ y'_{n-1} \cdots y'_i \cdots y'_0$ dans

le code binaire signé (0, 1, $\overline{1}$), par l'examen du multiplicateur par paires entrelacées $y_{i+1}y_i$ pour (i = -1, 0, 1, ..., n-1) avec y_{-1} = 0 selon le tableau suivant :

y_{i+1}	y_i	y'_{i+1}
0	0	0
0	1	1
1	1	$\overline{0}$
1	0	$\overline{1}$

Table II.5

Ceci revient en fait, à dédoubler tous les bits y_i (sauf le bit signe) en écrivant 1 comme 1 $\overline{1}$ et 0 comme 0 0 .

En effet :

$$y = - y_n 2^n + \sum_{i=0}^{n-1} y_i 2^i$$

$$= - y_n 2^n + 2 \sum_{i=0}^{n-1} y_i 2^i - \sum_{i=0}^{n-1} y_i 2^i$$

$$= - \sum_{i=0}^{n} y_i 2^i + \sum_{i=0}^{n-1} y_i 2^{i+1}$$

Posons : i + 1 = j il vient :

$$\sum_{i=0}^{n-1} y_i 2^{i+1} = \sum_{j=1}^{n} y_{j-1} 2^j$$

avec l'hypothèse $y_{-1} = 0$:

$$\sum_{j=1}^{n} y_{j-1} 2^j = \sum_{j=0}^{n} y_{j-1} 2^j$$

et y s'écrit alors :

$$y = \sum_{j=0}^{n} y_{j-1}\, 2^{j} - \sum_{i=0}^{n} y_i\, 2^{i}$$

ou encore, avec $j = i$:

$$y = \sum_{i=0}^{n} (- y_i + y_{i-1})\, 2^{i}$$

D'où le recodage de Booth, en posant $y'_i = - y_i + y_{i-1}$:

$$y' = \sum_{i=0}^{n} y'_i\, 2^{i} \qquad y'_i \in \{0, 1, \overline{1}\} \qquad i = (0, 1, \ldots, n)$$

Les opérations à effectuer au rang i, en fonction des bits examinés y_i et y_{i-1}, sont les suivantes :

y_i	y_{i-1}	Opérations à effectuer
0	0	Décalage de 1 rang vers la droite
0	1	Addition du multiplicande, puis décalage de 1 rang vers la droite
1	0	Soustraction du multiplicande, puis décalage de 1 rang vers la droite
1	1	Décalage de 1 rang vers la droite

Table II.6

Le décalage s'effectue comme dans le cas précédent, c'est à dire avec conservation du signe de chaque produit partiel cumulé.

Exemple : X = 0 1 0 0 1 = + 9
Y = 1 1 1 0 0 = - 4

Supposons : X rangé dans un registre R
Y rangé dans un registre MQ.

Une bascule Q_{-1} qui contiendra y_{i-1}, initialement à zéro ($y_{-1} = 0$), placée à droite de MQ.

Un accumulateur AC qui contiendra le produit partiel cumulé, nécessaire pour les opérations AC ± R → AC.

Une bascule de signe BS, qui contiendra le bit signe de l'opération AC ± R, placée à gauche de AC.

Le résultat final est donné par le contenu du double registre AC MQ, en complément à 2.

La table ci-dessous décrit les différentes phases du produit X Y, effectué selon l'algorithme de Booth.

	Examen des bits y_i | y_{i-1}		y'_i	Phase n°	Opérations à effectuer	BS	AC	MQ	Q_{-1}
état initial						0	0 0 0 0 0	1 1 1 0 0	0
i = 0	0	0	0	1	Décalage à droite de BS AC MQ Q_{-1}	0 ↓	0 0 0 0 0	0 1 1 1 0	0
i = 1	0	0	0	2	Décalage à droite de BS AC MQ Q_{-1}	↓ 0	0 0 0 0 0	0 0 1 1 1	0
i = 2	1	0	$\overline{1}$	3	AC−R dans AC Décalage à droite de BS AC MQ Q_{-1}	− 1 ↓ 1	0 1 0 0 1 1 0 1 1 1 1 1 0 1 1	1 0 0 1 1	1
i = 3	1	1	0	4	Décalage à droite de BS AC MQ Q_{-1}	↓ 1	1 1 1 0 1	1 1 0 0 1	1
i = 4	1	1	0	5	Décalage à droite de BS AC MQ Q_{-1}	↓ 1	1 1 1 1 0	1 1 1 0 0	1

Table II.7

Le résultat est donné par le contenu de AC MQ, soit :

X Y = 1 1 1 1 0 1 1 1 0 0 = - 36 = 9 × (- 4)

Ici aussi, la multiplication s'effectue en (n+1) phases, si n est le nombre de bits du multiplicateur (signe non compris). Par contre, le bit signe est traité comme un bit ordinaire. En effet, la présence d'une suite de 1 ou de 0 conduit à de simples décalages (pas d'additions ou de soustractions).

b) Recodage de Wallace :

Le principe est le même que celui de Booth, mais l'examen du multiplicateur se fait par 3 bits au lieu de 2 ; ainsi Y recodé sera exprimé dans le code $(\overline{2}, \overline{1}, 0, 1, 2)$, selon le tableau suivant :

y_{i+1}	y_i	y_{i-1}	y'_i
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	2
1	0	0	$\overline{2}$
1	0	1	$\overline{1}$
1	1	0	$\overline{1}$
1	1	1	0

Table II.8

Pour ($i = 0, 2, 4, \ldots, n-1$)
avec $y_{-1} = 0$
n impair et égal à la longueur du multiplicateur Y (signe non compris)

Si n est pair, il faut :
- soit dédoubler le bit signe et faire ($i = 0, 2, 4, \ldots, n$)
- soit traiter la paire $y_n y_{n-1}$ selon la table suivante :

y_n	y_{n-1}	y'_n
0	0	0
0	1	1
1	0	$\overline{1}$
1	1	0

Table II.9

Le recodage de Wallace repose sur le fait qu'on peut écrire :

$$y_i \, 2^i = y_i \, 2^{i+1} - 2 \, y_i \, 2^{i-1} \qquad (2.95)$$

Par ailleurs, nous avons, pour n impaire :

$$y = - y_n \, 2^n + \sum_{i=0}^{n-1} y_i \, 2^i$$

$$= - y_n \, 2^n + \sum_{i=0,2,4\ldots}^{n-1} y_i \, 2^i + \sum_{i=1,3,5\ldots}^{n-2} y_i \, 2^i \qquad (2.96)$$

La relation (2.95) nous permet d'écrire :

$$(2.97) \qquad \sum_{i=1,3,5\ldots}^{n-2} y_i \, 2^i = \sum_{i=1,3\ldots}^{n-2} y_i \, 2^{i+1} - 2 \sum_{k=1,3\ldots}^{n-2} y_k \, 2^{k-1}$$

Posons : $i + 1 = j$

$k - 1 = \ell$

$y_{-1} = 0$

alors, (2.97) s'écrit :

$$(2.98) \qquad \sum_{i=1,3\ldots}^{n-2} y_i \, 2^i = \sum_{j=0,2,4\ldots}^{n-1} y_{j-1} \, 2^j - \sum_{\ell=0,2,4\ldots}^{n-3} 2 \, y_{\ell+1} \, 2^\ell$$

Reportons (2.98) dans (2.96), il vient :

$$(2.99) \qquad y = -y_n \, 2^n + \sum_{i=0,2,4\ldots}^{n-1} y_i \, 2^i + \sum_{j=0,2,4\ldots}^{n-1} y_{j-1} \, 2^j - \sum_{\ell=0,2,4\ldots}^{n-3} 2 \, y_{\ell+1} \, 2^\ell$$

$$- y_n \, 2^n - \sum_{\ell=0,2,4}^{n-3} 2 \, y_{\ell+1} \, 2^\ell = - 2 \, y_n \, 2^{n-1} - \sum_{\ell=0,2,4}^{n-3} 2 \, y_{\ell+1} \, 2^\ell$$

$$= - \sum_{\ell=0,2,4}^{n-1} 2 \, y_{\ell+1} \, 2^\ell$$

(2.99) s'écrit alors :

$$y = \sum_{i=0,2,4}^{n-1} y_i \, 2^i + \sum_{j=0,2,4}^{n-1} y_{j-1} \, 2^j - \sum_{k=0,2,4}^{n-1} 2 \, y_{\ell+1} \, 2^\ell$$

ou encore, avec $i = j = \ell$:

$$(2.100) \qquad y = \sum_{i=0,2,4}^{n-1} (- 2 \, y_{i+1} + y_i + y_{i-1}) \, 2^i$$

D'où le recodage de Wallace, en posant $y'_i = - 2 \, y_{i+1} + y_i + y_{i-1}$:

$$y' = \sum_{i=0,2,4}^{n-1} y'_i \, 2^i \qquad y'_i \in \{\overline{2}, \overline{1}, 0, 1, 2\}$$

Les opérations à effectuer en fonction des bits examinés y_{i+1}, y_i et y_{i-1}, sont les suivantes :

y_{i+1}	y_i	y_{i-1}	y'_i	Opération à effectuer
0	0	0	0	Décalage de deux rangs vers la droite
0	0	1	1	Addition du multiplicande puis décalage de deux rangs vers la droite
0	1	0	1	Addition du multiplicande puis décalage de deux rangs vers la droite
0	1	1	+2	Addition de deux fois le multiplicande puis décalage de deux rangs vers la droite
1	0	0	-2	Soustraction de deux fois le multiplicande puis décalage de deux rangs vers la droite
1	0	1	-1	Soustraction du multiplicande puis décalage de deux rangs vers la droite
1	1	0	-1	Soustraction du multiplicande puis décalage de deux rangs vers la droite
1	1	1	0	Décalage de deux rangs vers la droite

Table II.10

Exemple : X = 1 1 0 0 0 1 = - 15

Y = 1 1 0 1 1 0 = - 10

	Examen des bits y_{i+1} y_i y_{i-1}	y'_i	Phase n°	Opérations à effectuer
Initialisation				AC : Accumulateur de longueur 7 à zéro MQ : Registre de longueur 6 à zéro BS : Bascule de signe à zéro
i = 0	1 0 0	-2	1	(AC) - 2 X dans AC et décalage de deux rangs vers la droite de BS, AC, MQ. (AC) = 0 0 0 0 0 0 0 - 2 X = 1 1 0 0 0 1 0 = |0| |0 0 1 1 1 1 0| |0 0 0 0 0 0| (BS, AC, MQ) Décalage |0| |0 0 0 0 1 1 1| |1 0 0 0 0 0| (BS, AC, MQ)
i = 2	0 1 1	+2	2	(AC) + 2 X dans AC et décalage de deux rangs vers la droite de BS, AC, MQ. (AC) = 0 0 0 0 1 1 1 + 2 X = 1 1 0 0 0 1 0 = |1| |1 1 0 1 0 0 1| |1 0 0 0 0 0| (BS, AC, MQ) Décalage |1| |1 1 1 1 0 1 0| |0 1 1 0 0 0| (BS, AC, MQ)
i = 4	1 1 0	-1	3	(AC) - X dans AC puis décalage de deux rangs vers la droite de BS, AC, MQ. (AC) = 1 1 1 1 0 1 0 - X = 1 1 1 0 0 0 1 |0| |0 0 0 1 0 0 1| |0 1 1 0 0 0| (BS, AC, MQ) Décalage |0| |0 0 0 0 0 1 0| |0 1 0 1 1 0| (BS, AC, MQ) Résultat final Résultat tronqué : 0 0·0 0 1 0 0 1 0 1 1 0 = + 150

Table II.11

Dans les deux algorithmes, les bits signes sont traités comme des bits ordinaires, mais avec l'algorithme de Wallace, le nombre de phases est réduit de moitié.

L'aspect séquentiel, selon lequel les deux algorithmes ont été décrits, présuppose leur application aux multiplieurs séquentiels. Nous verrons qu'ils sont aussi applicables aux multiplieurs cellulaires.

II.2.2.3 - Les multiplieurs cellulaires

Ce sont des circuits logiques, purement combinatoires, effectuant le calcul et la sommation des produits partiels composant le produit de deux nombres X et Y.

On distingue deux grandes classes de multiplieurs cellulaires :

- Réseaux cellulaires itératifs

- Génération de la matrice produits partiels, avec réduction à plusieurs niveaux de la matrice par des pseudo-additionneurs.

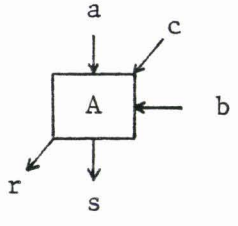
a) Réseaux cellulaires itératifs :

Le principe consiste à définir des cellules souvent identiques et des lois de liaison de ces cellules, de façon à donner au multiplieur, une structure généralement régulière et adaptable à une intégration à grande échelle (LSI) |20| |21| |22| |24| |25| |26| |27|.

Nous présentons 3 types de multiplieurs, mettant en évidence 3 types de cellules.

- Multiplieur cellulaire selon l'algorithme de Baugh et Wooley :

La figure II.42 décrit ce type de multiplieur pour un mot de 5 bits (signe compris).



$$\begin{cases} s = a \oplus b \oplus c \\ r = ab + bc + ca \end{cases}$$

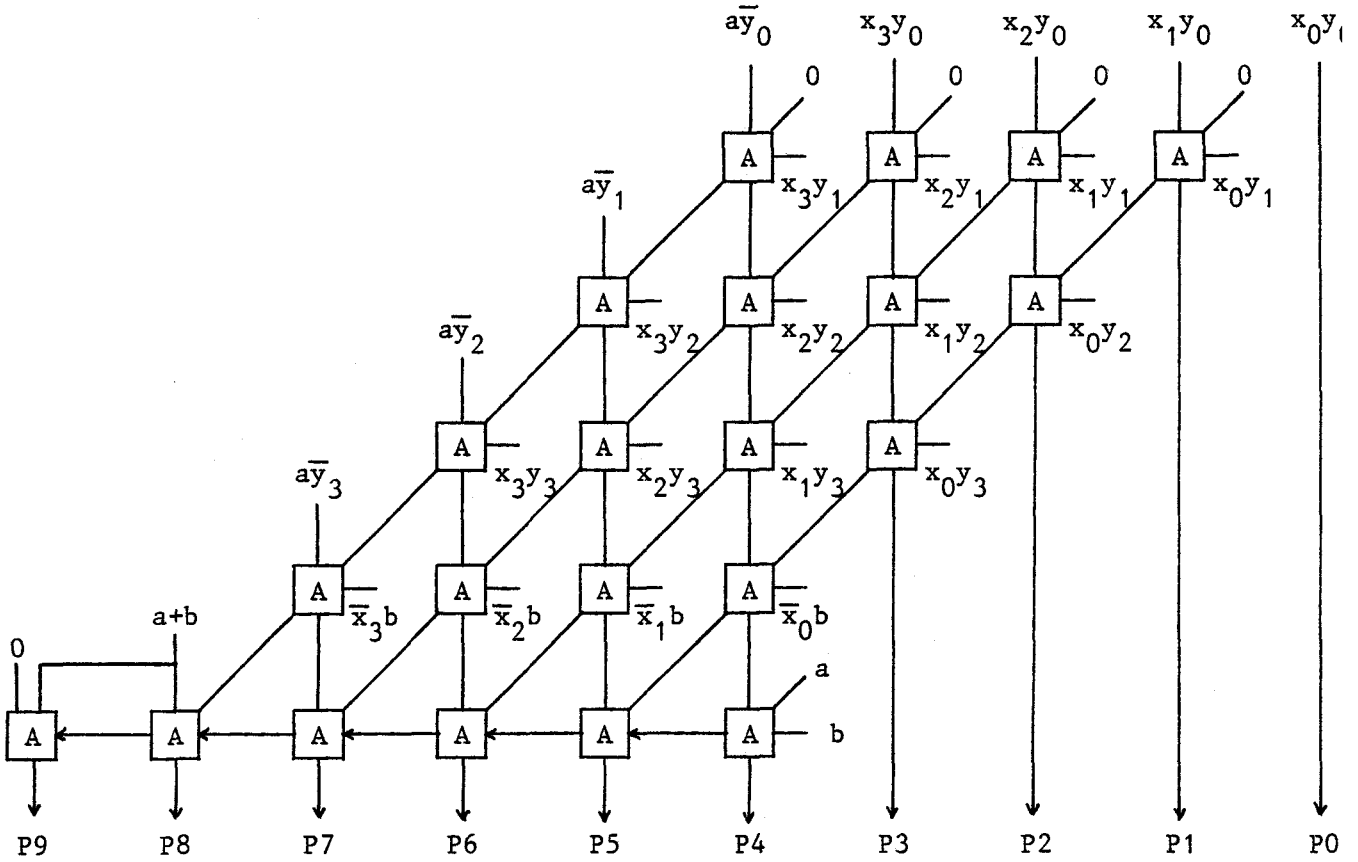


Figure II.42

La cellule de base A est un étage d'additionneur. Les entrées des cellules sont les microproduits x_iy_j, $a\bar{y}_i$, $b\bar{x}_i$ et les reports qui sont transmis obliquement. En tenant compte du réseau de calcul des microproduits qui ne figure pas sur le dessin, le temps de multiplication pour un mot de n bits, est de 2(n+1) traversées de cellule, plus deux couches logiques (une pour le calcul des compléments $\bar{x}_i$, $\bar{y}_j$ et une pour le calcul des x_iy_j).

Le matériel nécessaire pour l'élaboration d'un tel multiplieur est de :

n (n+1) + 2 cellules de type A

(n+1) (n+1) portes "ET" à deux entrées

et 2 n portes "NON"

où n est la longueur des mots X et Y (signes non compris).

- Multiplieur cellulaire selon l'algorithme de Booth :

Les différentes opérations à effectuer sur les produits partiels, sont l'addition, la soustraction ou pas d'opération.

Dans |21|, l'auteur a proposé une cellule de type addition/pas d'opération (Fig. II.43). La cellule possède 5 entrées et 5 sorties.

Les entrées P et Q contrôlent le mode d'opération :

$Q = 1 \Longrightarrow$ Faire $S = \dot{X} + R + C_{IN}$

$P = 1 \Longrightarrow$ Faire $\dot{X} = \overline{X}$

$P = 0 \Longrightarrow$ Faire $\dot{X} = X$

$Q = 0 \Longrightarrow$ Faire $S = R + 0 + C_{IN}$

avec C_{IN_0} = P.Q (Fig. II.44), ce qui correspond à ramener l'opération de soustraction R - X à une opération d'addition $R + \overline{X} + 1$ lorsque P = Q = 1.

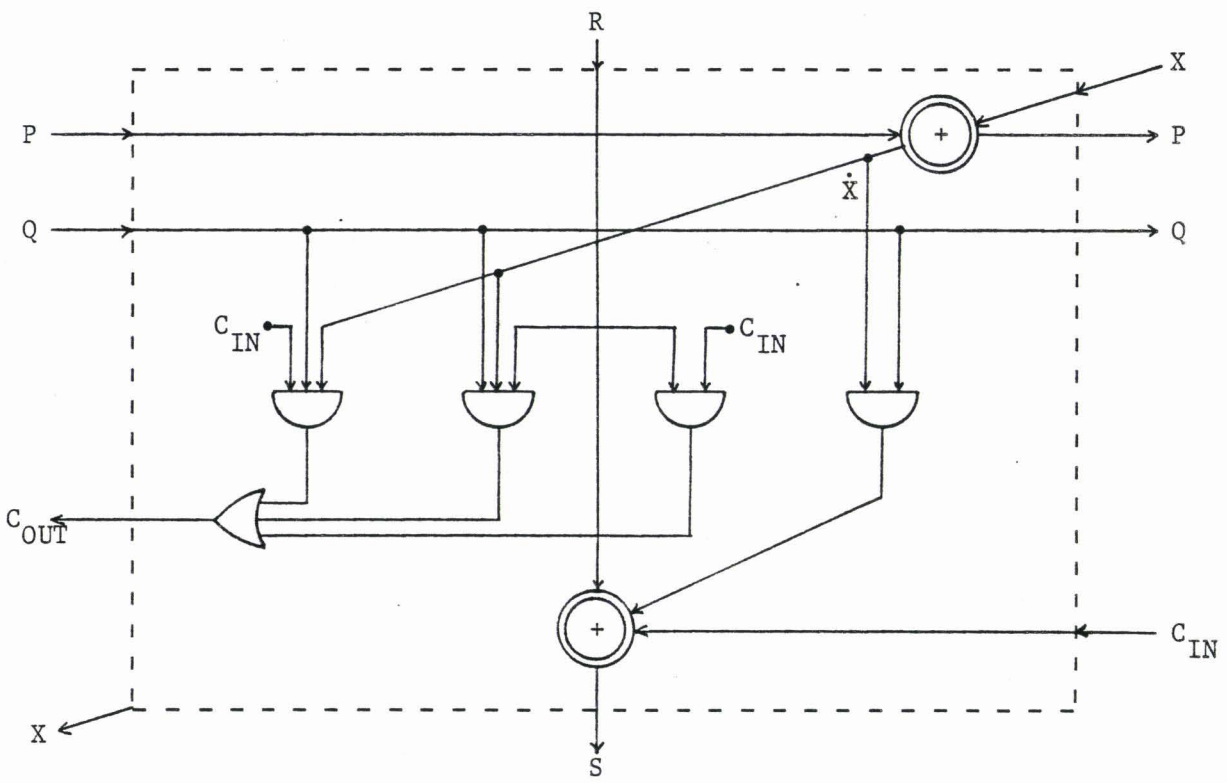


Figure II.43

Entrées : P, Q, X, R, C_{IN} Sorties : P, Q, S, X, C_{OUT}

⊕ ≡ OU Exclusif

La figure II.44 donne le schéma d'un multiplieur pour deux mots de 3 bits (signes compris), à l'aide de la cellule décrite par la figure II.43.

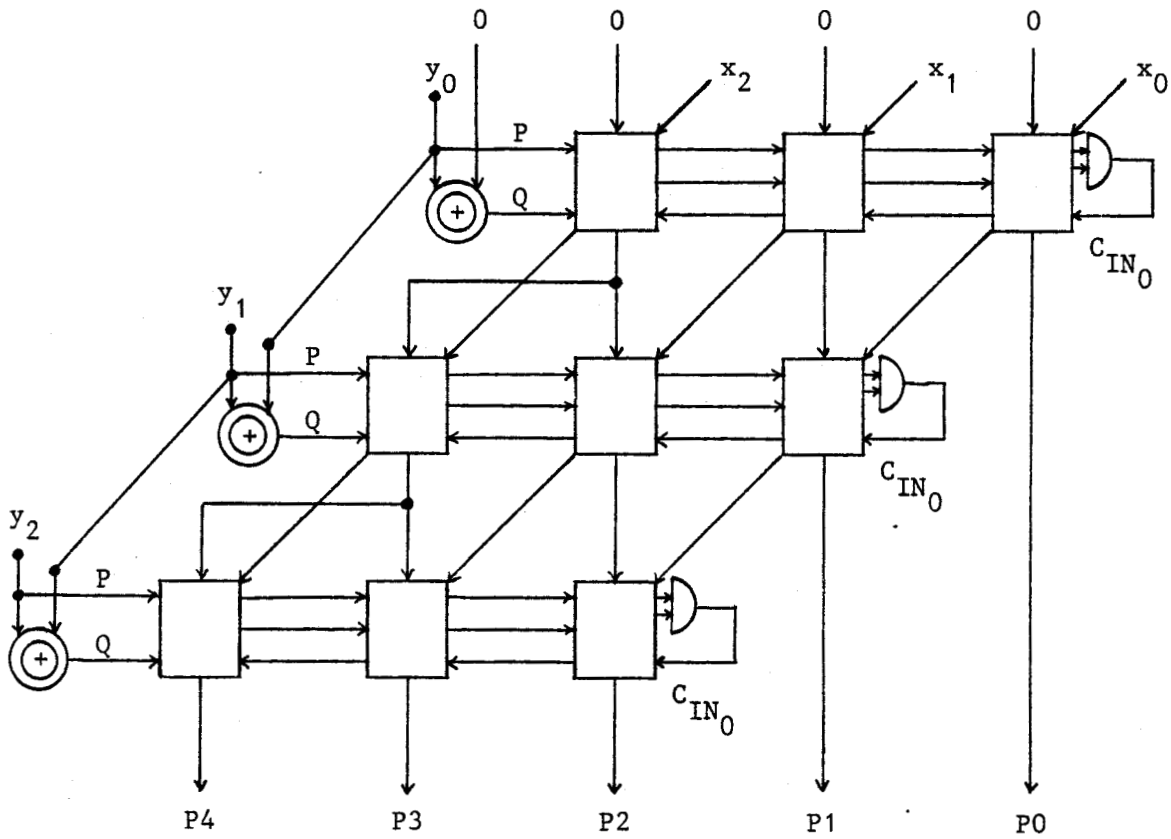


Figure II.44

$P = y_i$

$Q = y_i \oplus y_{i-1}$

$y_i\ y_{i-1} = (1\ 0$ ↓ $X=\overline{X}$ et $C_{IN_0}=1$ ou $0\ 1)$ ↓ $\dot{X}=X$ et $C_{IN_0}=0$ ⟶ addition de $\dot{X}$

$y_i\ y_{i-1} = (0\ 0$ ↓ $Q=0$ et $C_{IN_0}=0$ ou $1\ 1)$ ↓ $Q=0$ et $C_{IN_0}=0$ ⟶ addition de $\dot{X}.Q$

Le matériel nécessaire pour réaliser un multiplieur de deux mots de longueur n (signe compris), est de :

n^2 cellules

n portes "ET" à deux entrées

n portes "OU" exclusif à deux entrées.

Le temps de multiplication est égal à 3n-2 traversées de cellule, plus une traversée de porte OU Exclusif et une porte "ET".

- Multiplieur cellulaire selon l'algorithme de Wallace :

Les différentes opérations à effectuer sur les produits partiels P^j sont :

$$P^j + X \ , \ P^j - X \ , \ P^j + 2X \ , \ P^j - 2x \text{ ou } P^j + 0$$

où X est le multiplicande et P^j le produit partiel de l'étape j.

La cellule (multiplieur 4 × 2), proposé par C.I. TOMA |22| (Fig. II.45), permet de réaliser un multiplieur n×n bits (signes compris), à l'aide de $\frac{n}{2}\left(\frac{n}{4} + 1\right)$ cellules, plus :

$\frac{n}{2}$ OU Exclusif, deux entrées

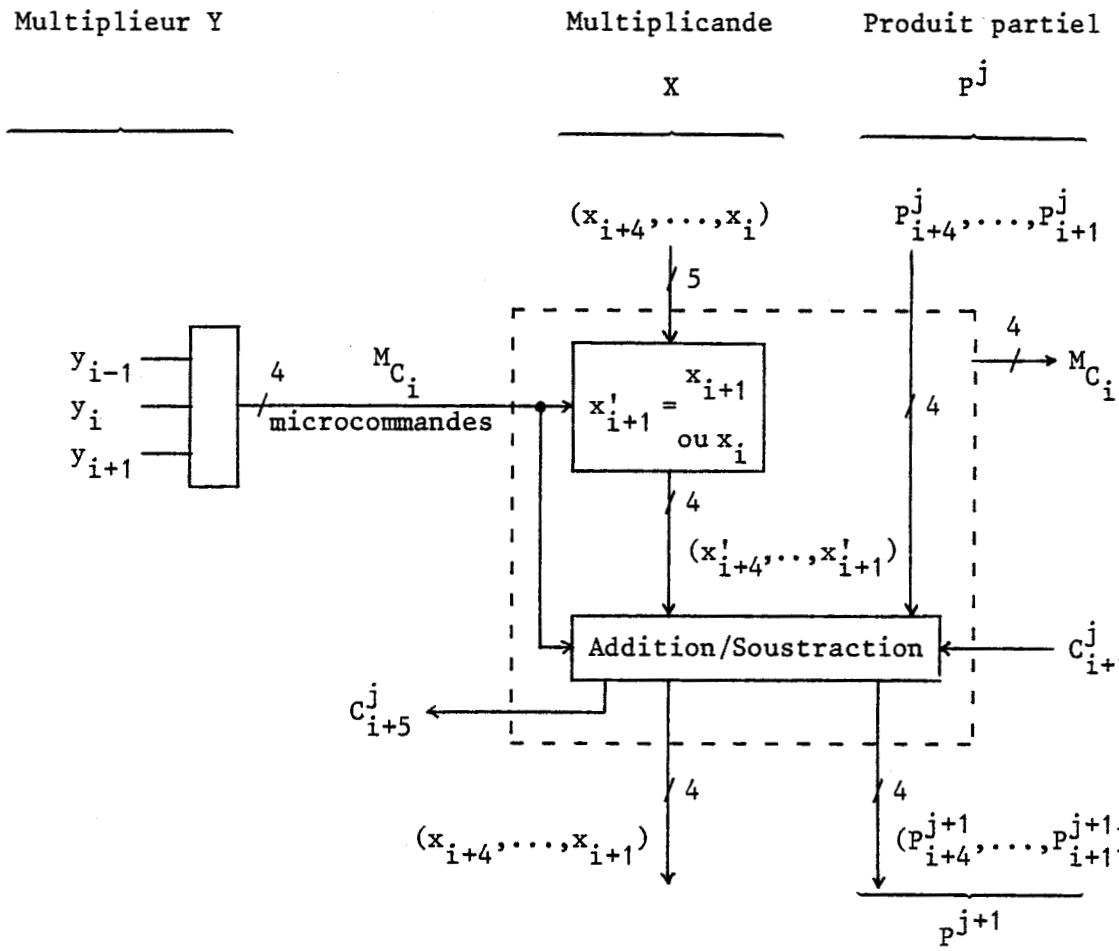
$\frac{n}{2}$ non OU Exclusif, deux entrées

n portes OU, deux entrées

n portes ET, deux entrées

$3\frac{n}{2}$ portes NON

} nécessaires au calcul des microcommandes des cellules



Synoptique de la cellule : Multiplieur 4×2

Figure II.45

avec T_{C_i} : temps de calcul des microcommandes à partir des y_i

T_{x_i} : temps de calcul des x'_i à partir des x_i

$T_{a/s}$: temps de calcul des P_i^{j+1} à partir des x'_i et des P_i^j

Le temps de multiplication de deux nombres de n bits (signes compris) est de l'ordre de :

$$(5\,\frac{n}{4} - 1)\,T_{a/s} + T_{C_i} + T_{x_i}$$

b) Génération et réduction de la matrice produits partiels :

La matrice produits partiels de deux nombres X, Y de n bits, comporte n lignes, dont la somme est le résultat du produit X Y. Un circuit qui reçoit comme entrée, les n lignes, et fournit comme sortie, leur somme (additionneur multiopérandes), est nommé "Compteur parallèle généralisé".

Introduit par L. DADDA |28|, comme approche pour la réalisation des multiplieurs parallèles, et qui a fait l'objet de plusieurs études concernant particulièrement la réalisation de grands compteurs parallèles à base d'un petit |29| |30| |31| |32| |33|, un compteur généralisé est représenté par la notation suivante :

$$(C_{n-1}, C_{n-2}, \ldots, C_0 ; d)$$

où : n est le nombre de colonnes
C_i, le nombre de bits de la colonne i et de poids 2^i
d, le nombre de bits représentatifs de la somme.

Soit C_{ij}, l'entrée j de la colonne i ($j = 0, 1, \ldots, C_i-1$)
$C_{ij} = \{0 , 1\}$

Soit S_k : bit somme de rang k ($k = 0, 1, \ldots, d-1$)
$S_k = \{0 , 1\}$

alors :

$$s = \sum_{i=0}^{d-1} S_k 2^k = \sum_{i=0}^{n-1} \sum_{j=0}^{C_i-1} C_{ij} 2^i$$

La capacité C du compteur est de :

$$C = \sum_{i=0}^{n-1} C_i 2^i \qquad \text{et donc} \qquad s \leqq C$$

Exemple :

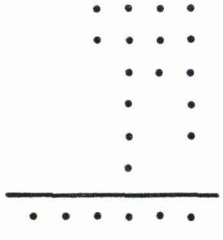


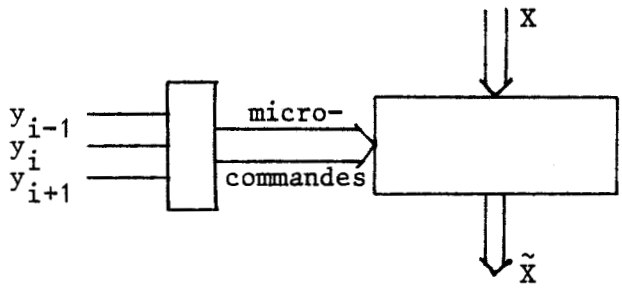
Figure II.46 Compteur parallèle généralisé (2 , 6 , 3 , 5 ; 6).

Nous avons vu que l'algorithme de Wallace, réduit de moitié, la matrice produits partiels du produit de deux nombres X et Y.

Soit M_w cette matrice, et proposons nous d'appliquer cette notion de compteur parallèle à M_w.

- Détermination de la matrice M_w :

Le calcul de M_w repose sur le même principe de la cellule 4×2 |22|, décrit par le schéma synoptique suivant :



où $\tilde{X} =$

$$\begin{cases} 0 & \text{si } \bar{y}_{i+1}\ \bar{y}_i\ \bar{y}_{i-1} + y_{i+1}\ y_i\ y_{i-1} = m_{i_0} = 1 \\ X & \text{si } \bar{y}_{i+1}\ y_i\ \bar{y}_{i-1} + \bar{y}_{i+1}\ \bar{y}_i\ y_{i-1} = m_{i_1} = 1 \\ -X & \text{si } y_{i+1}\ y_i\ \bar{y}_{i-1} + y_{i+1}\ \bar{y}_i\ y_{i-1} = m_{i_{\bar{1}}} = 1 \\ +2X & \text{si } \bar{y}_{i+1}\ y_i\ y_{i-1} = m_{i_2} = 1 \\ -2X & \text{si } y_{i+1}\ \bar{y}_i\ \bar{y}_{i-1} = m_{i_{\bar{2}}} = 1 \end{cases}$$

Figure II.46

On peut écrire :

$$\begin{aligned} -X &= \overline{X} + 1 \\ -2X &= -2\overline{X} + 1 \end{aligned} \qquad \text{(+ arithmétique)}$$

Ceci correspond au cas : $m_{i_{\overline{1}}} = 1$ ou $m_{i_{\overline{2}}} = 1$.

Posons $\alpha_i = m_{i_{\overline{1}}} + m_{i_{\overline{2}}}$ et écrivons $\tilde{X}$ comme :

$$\tilde{X} = \tilde{X}' + \alpha_i \, 2^0$$

où :

$$(2.101) \qquad \tilde{x}'_i = m_{i_1} x_i + m_{i_{\overline{1}}} \overline{x}_i + m_{i_2} x_{i-1} + m_{i_{\overline{2}}} \overline{x}_{i-1}$$

ou encore :

$$(2.102) \qquad \tilde{x}'_i = (m_{i_1} + m_{i_{\overline{1}}}) x_i \oplus (m_{i_2} + m_{i_{\overline{2}}}) x_{i-1} \oplus \alpha_i$$

qui donne :

pour $\alpha_i = 0$:

$$\begin{cases} m_{i_{\overline{1}}} + m_{i_{\overline{2}}} = 0 & \tilde{x}'_i = m_{i_1} x_i \oplus m_{i_2} x_{i-1} \oplus 0 \\ m_{i_1} \cdot m_{i_2} = 0 & \tilde{x}'_i = 0 \quad \text{si} \quad m_{i_1} = m_{i_2} = 0 \end{cases}$$

$$\tilde{x}'_i = x_i \quad \text{si} \quad m_{i_1} = 1 \quad \text{et} \quad m_{i_2} = 0$$

$$\tilde{x}'_i = x_{i-1} \quad \text{si} \quad m_{i_1} = 0 \quad \text{et} \quad m_{i_2} = 1$$

$$\tilde{x}'_i = x_i \oplus x_{i-1} \quad \text{si} \quad m_{i_1} = m_{i_2} = 1$$

(cas non possible)

pour $\alpha_i = 1$:

$$\begin{cases} m_{i_1} + m_{i_2} = 0 & \tilde{x}'_i = m_{i_{\overline{1}}} \cdot x_i \oplus m_{i_{\overline{2}}} x_{i-1} \oplus 1 \\ m_{i_{\overline{1}}} + m_{i_{\overline{2}}} = 1 & \end{cases}$$

$\tilde{x}'_i = x_i \oplus 0 \oplus 1 = \overline{x}_i$ si $m_{i_{\overline{1}}} = 1$ et $m_{i_{\overline{2}}} = 0$

$\tilde{x}'_i = 0 + x_{i-1} \oplus 1 = \overline{x}_{i-1}$ si $m_{i_{\overline{1}}} = 0$ et $m_{i_{\overline{2}}} = 1$

$\tilde{x}'_i = 0 \oplus 0 \oplus 1 = 1$ si $m_{i_{\overline{1}}} = m_{i_{\overline{2}}} = 0$

(cas impossible $\alpha_i = 1$)

$\tilde{x}'_i = x_i \oplus x_{i-1} \oplus 1$ si $m_{i_{\overline{1}}} = m_{i_{\overline{2}}} = 1$

(cas non possible)

Les microcommandes peuvent alors être calculées sous trois formes :

1) m_{i_1} , m_{i_2} , $m_{i_{\overline{1}}}$, $m_{i_{\overline{2}}}$, α_i dont les équations logiques sont données Fig. II.46, et $\tilde{x}'_i$ calculé à partir de (2.101)

2) $m_{i_{1\overline{1}}} = m_{i_1} + m_{i_{\overline{1}}} = y_i \oplus y_{i-1}$ $\quad m_{i_{2\overline{2}}} = m_{i_2} + m_{i_{\overline{2}}} = \overline{y_i \oplus y_{i-1}}$

$\alpha_i = m_{i_{\overline{1}}} + m_{i_{\overline{2}}} = y_{i+1} \overline{y}_i + y_{i+1} \overline{y}_{i-1} = y_{i+1} \cdot \overline{y_i \, y_{i-1}}$

et le calcul de $\tilde{x}'_i$ se fait par la relation (2.102).

3) Sous forme explicite :

$$\tilde{x}'_i = \overline{y}_{i+1} \, y_i \, \overline{y}_{i-1} \, x_i + \overline{y}_{i+1} \, \overline{y}_i \, y_{i-1} \, x_i + y_{i+1} \, y_i \, \overline{y}_{i-1} \, \overline{x}_i + y_{i+1} \, \overline{y}_i \, y_{i-1} \, \overline{x}_i + \overline{y}_{i+1} \, y_i \, y_{i-1} \, x_{i-1} + y_{i+1} \, \overline{y}_i \, \overline{y}_{i-1} \, \overline{x}_{i-1}$$

$$\alpha_i = y_{i+1} \, \overline{y}_i + y_{i+1} \, \overline{y}_{i-1}$$

Cette forme est intéressante si on dispose des compléments $\overline{y}_j$ et $\overline{x}_j$, ou si on utilise des réseaux logiques programmables par exemple.

La figure II.47 donne le schéma logique, de la génération de la ligne i de M_w, selon la 2ème forme, avec :

$X = a\ x_{n-1}\ x_{n-2} \cdots x_i \cdots x_0$ comme multiplicande

$Y = b\ y_{n-1}\ y_{n-2} \cdots y_i \cdots y_0$ comme multiplieur

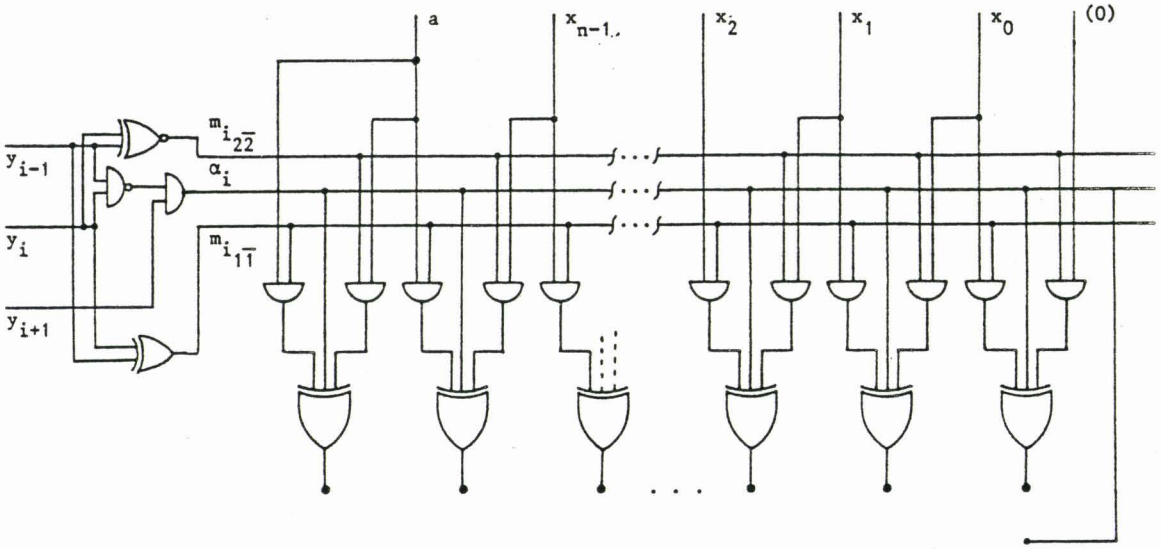


Figure II.47

D'où M_w pour n=16 (n = longueur des mots X et Y, signes compris).

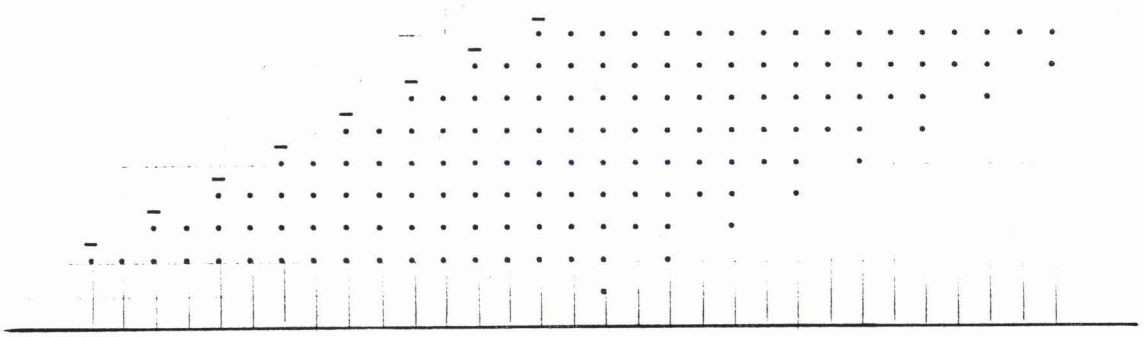


Figure II.48

Le produit XY est la somme des 9 lignes, où $\bar{.}$ est affecté du signe -, et . est affecté du signe +.

$. \in \{0\ ,\ 1\}$

$\bar{.} = a \in \{0\ ,\ 1\}$

Pour éviter cette irrégularité du signe, il suffit d'adopter soit la représentation dyadique, soit le complément vrai.

En représentation dyadique, M_w devient :

```
a a a a a a a a a a a a a a a . . . . . .
a a a a a a a a a a a a a a . . . . . .  .
a a a a a a a a a a a a a . . . . . .  .
a a a a a a a a a a a . . . . . .  .
a a a a a a a a . . . . . .  .
a a a a a a . . . . . .  .
a a a a . . . . . .  .
a a . . . . . .  .
                 .
```

Figure II.49

En complément vrai, il s'agit sur la figure II.48, de déplacer la diagonale d'extrême gauche des $\bar{.}$, sur la dernière ligne et de prendre son complément booléen augmenté d'une unité de plus faible poids.

La dernière ligne devient :

a o a o a o a o a o a o a o .

(où le a sont affectés du signe -), que nous pouvons remplacer par :

```
1 1 ā 1 ā 1 ā 1 ā 1 ā 1 ā 1 ā 1 ā o .
                                1   +
-------------------------------------
a a o a o a o a o a o a o a o a a o . =
```

La matrice M_w prend alors la forme suivante :

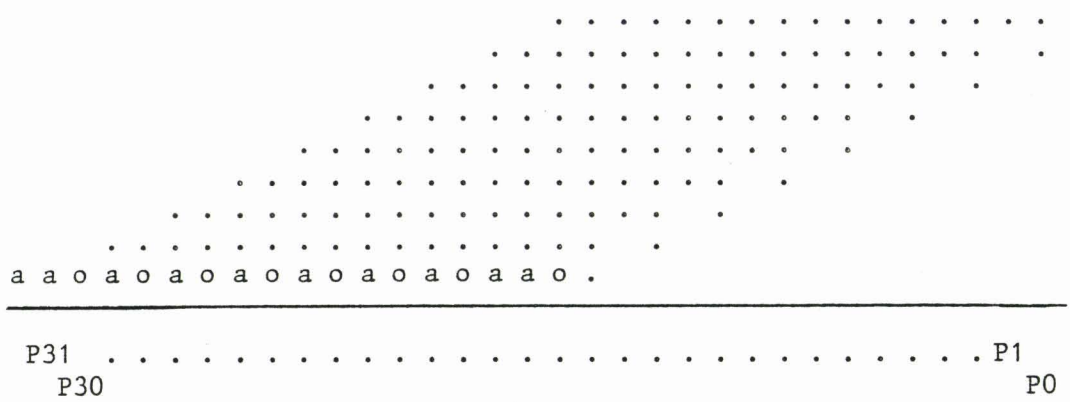


Figure II.50

- Réduction de la matrice M_w :

C'est la matrice M_w, donné par la Figure II.50, que nous considérons comme matrice d'origine, et qu'il faut réduire en deux lignes.

Les deux lignes sont ensuite additionnées à l'aide d'un additionneur rapide (série parallèle, par exemple).

La réduction de la matrice se fait par des compteurs parallèles (réalisés à l'aide d'étages d'additionneur |28|, ou des mémoires mortes, ROM |34| |35|) généralement de même type, avec :

$$C_i = C_j \quad \forall \; i \quad \text{et} \quad j$$

et noté :

$(n \times N , d)$ où n : nombre de colonnes
N : nombre d'élément des colonnes ou nombre de lignes
d : longueur du mot somme des N opérandes.

Avec ce type de compteur, il est possible de réduire en deux lignes une matrice $a.n \times N$ $(a > 2)$ en une seule étape si $d \leq 2n$.

La matrice M_w, pour être réduite en deux lignes et en une seule étape, nécessite des compteurs parallèles de type (3 × 9 ; 6) qui, pour être réalisés, demandent des "ROMS" de 2^{27} comme capacité mémoire, ou 21 étages d'additionneur (1 × 3 ; 2) en 6 étapes |33|.

Or, il est possible de réduire la matrice M_w en 4 étapes seulement à l'aide d'étages d'additionneurs et de 1/2 additionneur. C'est cette dernière solution que nous analysons.

Pour alléger l'écriture de la matrice, nous la représentons par une suite de 32 chiffres :

$$\underset{ij}{k}$$

k : nombre de bits de la colonne correspondante, au niveau ℓ
i : nombre d'étages d'additionneurs
j : nombre de 1/2 additionneur
} permettant le passage au niveau $\ell + 1$

Au niveau 0, nous avons M_{w_0}, matrice d'origine.

$\frac{1}{00}\frac{0}{00}\frac{2}{00}\frac{1}{00}\frac{3}{00}\frac{2}{00}\frac{4}{00}\frac{3}{00}\frac{5}{00}\frac{4}{00}\frac{6}{10}\frac{5}{10}\frac{7}{20}\frac{6}{20}\frac{8}{21}\frac{8}{21}\frac{8}{21}\frac{9}{30}\frac{7}{20}\frac{8}{20}\frac{6}{10}\frac{7}{10}\frac{5}{00}\frac{6}{00}\frac{4}{00}\frac{5}{00}\frac{3}{00}\frac{4}{00}\frac{2}{00}\frac{3}{00}\frac{1}{00}\frac{2}{00}$ $\ell = 0$

$\frac{1}{00}\frac{0}{00}\frac{2}{00}\frac{1}{00}\frac{3}{00}\frac{2}{00}\frac{4}{10}\frac{3}{10}\frac{5}{11}\frac{5}{11}\frac{5}{11}\frac{5}{11}\frac{5}{11}\frac{6}{20}\frac{6}{20}\frac{6}{20}\frac{5}{11}\frac{5}{11}\frac{5}{11}\frac{5}{11}\frac{5}{11}\frac{5}{11}\frac{6}{20}\frac{4}{10}\frac{5}{10}\frac{3}{00}\frac{4}{00}\frac{2}{00}\frac{3}{00}\frac{1}{00}\frac{2}{00}$ $\ell = 1$

$\frac{1}{00}\frac{0}{00}\frac{2}{00}\frac{1}{00}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{4}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{4}{00}\frac{2}{00}\frac{3}{00}\frac{1}{00}\frac{2}{00}$ $\ell = 2$

$\frac{1}{00}\frac{0}{00}\frac{2}{01}\frac{2}{01}\frac{2}{01}\frac{2}{01}\frac{2}{01}\frac{2}{01}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{3}{10}\frac{2}{01}\frac{2}{01}\frac{2}{01}\frac{2}{01}\frac{2}{01}\frac{2}{01}\frac{3}{10}\frac{1}{00}\frac{2}{00}$ $\ell = 3$

1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2 $\ell = 4$

Le temps de multiplication est ici de $T_m = T_c + 4\,T_a + T_f$ avec :

T_c : Temps de calcul de M_w (environ 5 couches logiques)
T_a : Temps de traversée d'un étage d'additionneur
T_f : Temps d'addition de deux mots de 32 bits.

Le matériel nécessaire pour la réalisation d'un tel multiplieur est de :

- 330 portes "ET" deux entrées
- 8 portes "Nand" deux entrées
- 8 portes "Non OU exclusif" deux entrées
- 8 portes "OU exclusif" deux entrées
- 136 portes "OU exclusif" trois entrées
- 85 étages d'additionneurs ou compteur (1 × 3 ; 2)
- 27 demi-additionneurs ou compteurs (1 × 2 ; 2)
- 1 additionneur 32 bits

Dans le cas général, pour des opérandes de n bits (signes compris) : un étage d'additionneur réduit d'un bit la matrice M_w et, comme elle est réduite à deux mots de 2n bits, le nombre d'étages peut être estimé à :

$N(M_w) - 4n$ où $N(M_w)$: nombre de bits de la matrice M_w

$$N(M_w) = \frac{n^2}{2} + n + 1$$

$$\Longrightarrow \quad N_{ea} \cong \frac{n^2}{2} - 3n + 1$$

N_{ea} : nombre d'étages d'additionneurs

Le nombre d'étapes permettant la réduction de M_w à deux opérandes est donné par j tel que :

$$d_j < \frac{n}{2} + 1 \leqq d_{j+1}$$

où $d_j = \left\lfloor \frac{3}{2} d_{j-1} \right\rfloor$ et $d_1 = 2$

$\lfloor X \rfloor$ = valeur entière de X, prise par défaut.

Ceci vient du fait que des colonnes adjacentes de même nombre d'éléments d_j vont être réduites, à l'aide de $\left\lfloor d_j/3 \right\rfloor$ étages, pour chaque colonne, à :

$$d_{j-1} = d_j + \underbrace{\left\lfloor \frac{d_j}{3} \right\rfloor}_{\substack{\downarrow \\ \text{retenues de} \\ \text{la colonne précédente}}} - 2 \left\lfloor \frac{d_j}{3} \right\rfloor = d_j - \left\lfloor \frac{d_j}{3} \right\rfloor = \left\lfloor 2 \frac{d_j}{3} \right\rfloor \qquad d_j = \left\lfloor 3 \frac{d_{j-1}}{2} \right\rfloor$$

Le temps de multiplication T_m devient :

$$T_m = T_c + j\ T_a + T_f\ (2n)$$

où $T_f(2n)$ est le temps d'addition de deux mots de 2n bits.

II.3 - ETUDE COMPARATIVE DES DEUX TECHNOLOGIES (SSI - FPLA)

Ce paragraphe concerne l'étude des performances d'un additionneur et d'un multiplieur de mots de 16 bits, réalisés à l'aide de portes lo--iques et de réseaux logiques programmables.

Pour l'addition, c'est la table II.4 et la figure II.38 qui nous serviront de base de comparaison.

Pour la multiplication, c'est le multiplieur cellulaire étudié dans le paragraphe (II.2.2.3 - b), qui permettra cette comparaison. Il convient donc de chiffrer le nombre de FPLA nécessaires à sa réalisation et le temps de multiplication correspondant.

II.3.1 - Remarque

Nous avons besoin d'un additionneur de 32 bits pour la sommation des deux dernières lignes après la réduction de M_w.

Le temps de calcul et le matériel nécessaire à sa réalisation peuvent être estimés au double d'un additionneur de 16 bits dans le cas de la technologie SSI.

Dans le cas des réseaux logiques programmables, le nombre de FPLA devient :

17 pour la génération de M_w
27 pour la réduction de M_w en deux lignes
7 pour l'additionneur 32 bits selon la figure II.36.

Le temps de multiplication est alors de : $T_m = 1 + 4 + 4$
$= 9$ traversées FPLA

II.3.2 - Paramètres de comparaison

Les paramètres qu'il est facile de chiffrer et qui permettent en effet de se faire une bonne idée sur les performances du système sont :

- le temps de calcul*
- la puissance dissipée*
- la complexité, mesurée en nombre de boitiers, ou plus précisément, en nombre de points de connexion.

* Caractéristiques des circuits considérés ici :
TTL (S ou H) : temps de propagation typ. = 5 ns
consommation par porte typ. = 20 mW
FPLAN 82S 100/101 : temps de propagation typ. = 50 ns
consommation par circuit typ. = 600 mW

Les tables II.12 et II.13 donnent ces paramètres pour l'additionneur et le multiplieur de mots de 16 bits, pour les deux technologies.

	Nombre de boitiers	Nombre de connexions	Temps de l'addition	Puissance dissipée
SSI	80	80×14=1120	7×5=35 ns	170 × 20 = 3400 mW = 3,4 W
FPLA	4	4×28=112	3×50=150 ns	4 × 600 = 2400 mW = 2,4 W

Table II.12 Paramètres de l'additionneur

	Nombre de boitiers	Nombre de connexions	Temps de l'addition	Puissance dissipée
SSI	373	373×14=5222	150 ns	28,76 W
FPLA	51	51×28=1428	9×50=450 ns	51 × 600 = 30600 mW = 30,6 W

Table II.13 Paramètres du multiplieur

II.3.3 - Conclusion

Les tables II.12 et II.13 établies pour un cas particulier (mots de 16 bits), nous permettent de constater que le facteur "temps" peut être considéré comme satisfaisant, pour les circuits SSI, mais leur complexité rend fastudieus la matérialisation d'opérateurs numériques sur cette base. De l'autre côté, les réseaux logiques programmables, qui réduisent cette complexité, demeurent encore lents, de par leur structure même (nombre de mintermes limité (48)) inadaptée à ce domaine.

Cependant, une solution envisageable consisterait à trouver un compromis entre les deux technologies. Nous pensons par exemple, à l'utilisation d'un FPLA à la place des circuits $\boxed{A_i}$, dans un additionneur à connexion série parallèle, et des étages d'additionneurs pour réduire la matrice M_w, dans un multiplieur à base de réseaux logiques programmables.

CONCLUSION

Nous avons, au cours de ce chapitre, présenté quelques principes de conception d'additionneurs et de multiplieurs des nombres binaires en complément à 2.

Cette étude a été entreprise dans l'espoir de matérialiser des opérateurs numériques rapides, à base de réseaux logiques programmables qui nous paraissaient une voie prometteuse.

Or, les différents schémas de réalisation étudiés, montrent leurs faiblesses dans ce domaine (lents, taille insuffisante). Ceci n'empêche pas leur intégration comme circuits annexes, dans un tel système, d'autant plus que leur vitesse de propagation augmente dans le temps.

Le problème reste posé de nouveau, ce qui est dû au caractère complexe des opérateurs numériques, auquel nous suggérons deux solutions :

1) entreprendre une étude profonde des opérateurs arithmétiques, probablement à long terme, débouchant certainement à des circuits complexes et vraisemblablement à très haut niveau d'intégration,
2) faire appel à des éléments arithmétiques standards existants (voir Annexe II).

BIBLIOGRAPHIE DU CHAPITRE II

|1| M. AUMIAUX
"Fonctions logique et arithmétique binaire"
MASSON, Paris, 1979.

|2| J. CHINAL
"Circuits logiques de traitement numérique de l'information"
SUP'AERO, Cepadues Editions.

|3| H. BOUCHER
"Machines informatiques"
I, II, III, ENSTA, Paris, 1970.

|4| N.R. SCOTT
"Analog and digital computer technology"
Mc Graw-Hill, New York, 1960.

|5| H. BOUCHER
"Architecture de l'ordinateur"
Tome 1, ENSTA, Cepadues Editions.

|6| J.P. CHINAL
"Techniques booléennes et calculateurs arithmétiques"
DUNOD, Paris, 1967.

|7| K.J. DEAN
"An introduction to counting techniques and transistor logic"
Chapman and Hall, London, 1964.

|8| T.S. LIU, H.R. HOHULNI, L.E. SHIAN, S. MUROGA
"Optimal one bit full adder with different types of gates"
IEEE Trans. on Computer, Vol. C 23, n° 1, pp. 63-69, janvier 1974.

|9| J. CHINAL
"Théorie microanalytique des processeurs arithmétiques"
Thèse Doctorat ès Sciences Mathématiques, Univ. P. Sabatier, Toulouse, 1973.

|10| A. WEINBERGER, J.L. SMITH
"A one micro-second adder using Megacycle circuitry"
IRE Trans. Elect. Comp., Vol. EC 5, june 1956, pp. 67-73.

|11| B. GILCHRIST, J.H. POMERENE, S.Y. WONG
"Fast carry logic for digital computers"
IRE Trans. Elect. Comp., pp. 133-136

|12| J. SKLANSKY
"Conditional sum addition logic"
IRE Trans. Elect. Comp., Vol. EC 9, june 1960, pp. 226-231.

|13| J. SKLANSKY
"An evaluation of several two summ and binary adders"
IRE Trans. Elect. Comp. Vol. EC 9, pp. 213-226; 1960.

|14| J. CHINAL
"Multilevel look-ahead binary comparators"
ENSA, 1972.

|15| J. CHINAL
"Microsystèmes numériques"
ENSA, 1972.

|16| I. FLORES
"The logic of computer arithmetic"
Prentice-Hall, 1963.

|17| F.F. SELLERS, M.Y. HSIAO, L.W. BEARSON
"Error-detecting logic for digital computers"
Mc Graw-Hill, New York, 1968.

|18| G.C. LANGDON, C.K. TANG
"Concurrent error detecting for group look ahead binary adders"
IBM Journal of Research and Development, septembre 1970, pp. 563-573.

|19| C.R. BAUGH, B.A. WOOLEY
"A tow's complement parallel array multiplication algorithm"
IEEE Trans. on Comp., Vol. C 22, n° 12, pp. 1045-1047, december 1973.

|20| J.C. MAJITHIA, R. KITAI
"An iterative array for multiplication of signed binary numbers"
IEEE Trans. on Comp., Vol. C 20, n° 2, pp. 214-216, feb. 1971.

|21| S. BANDYOPADHYAN, S. BASU, A.K. HOUDHURY
"An iterative array for multiplication of signed binary numbers"
IEEE Trans. on Comp., Vol. C 21, pp. 921-922, august 1972.

|22| C.I. TOMA
"Cellular logic array for high speed signed binary number multiplication"
IEEE Trans. on Comp., Vol. C 24, n° 9, september 1975.

|23| TEXAS INSTRUMENTS
"The TTL Data book for design engineers"
Dallas, 1976.

|24| S.D. PEZARIS
"A 40 ns, 17 bits multipliers"
IEEE Trans. on Comp., Vol. C 20, pp. 442-447, april 1971.

|25| J. DEVERELL
"Multiplication of complex numbers using iterative arrays"
Electronics Letters, Vol. 7, n° 9, pp. 205-207, may 1971.

|26| T.J. CHUNG, S.D. DERDOSIAN
"Fast digital multiplier based on iterative cellular arrays of ROMS"
Electronics Letters, Vol. 11, n° 18, pp. 426-428, september 1975.

|27| I.D. DEEGAN
"Cellular multiplier for signed binary numbers"
Electronics Letters, Vol. 7, pp. 436-437, 1971.

|28| L. DADDA
"Some schemes for parallel multipliers"
Alta Freq., Vol. 34, pp. 349-356, 1965.

|29| C.C. FOSTER, F.D. STOCKTON
"Counting responders in an associative memory"
IEEE Trans. on Comp., Vol. C 20, pp. 1580-1583, 1971.

|30| E.E. SWARTZLANDER Jr
"Parallel Counters"
IEEE Trans. on Comp., Vol. C 22, pp. 1021-1024, 1973.

|31| H. KOBAYASHI, H. OHARA
"A synthesizing for large parallel counters with a net work of smaller ones"
IEEE Trans. on Comp., Vol. C 27, pp. 753-757, 1978.

|32| A. MEO
"Arithmetic networks and their minimization using a new line of elementary units"
IEEE Trans. on Comp., Vol. C 24, pp. 258-280, 1975.

|33| S. DOMIDO, H.A. CANTO
"Synthesis of generalized parallel counters"
IEEE Trans. on Comp., Vol. C 30, pp. 699-703, 1981.

CHAPITRE III

SCHEMA DE CABLAGE

ET PROTOCOLES DE COMMUNICATION

SCHEMA DE CABLAGE ET PROTOCOLES DE COMMUNICATION

INTRODUCTION

La chaîne de traitement fréquemment suivie pour simuler un processus P_i sur un calculateur analogique, digital ou hybride, est la suivante :

{Processus P_i} ⟷ (Modélisation) {Modèle Mathématique} ⟷ (Traduction / Simulation) {Programme ET/OU Cablage}

D. CORBEEL |1| a introduit un intermédiaire formel, reposant sur la structure algébrique du magmoïde |2| |3| |4| |5|, entre le modèle mathématique et le cablage physique du calculateur, appelé schéma de cablage. La chaîne de traitement devient alors :

{Processus P_i} ⟷ (Modélisation) {Modèle Mathématique} ⟷ {Schéma de cablage} ⟷ {Cablage}

Le premier intérêt du schéma de cablage est de permettre le passage d'un modèle mathématique à un autre modèle mathématique du même processus, d'une façon aisée, en se servant de ses propriétés combinatoires |1| (une autre conséquence sera le passage d'un schéma de cablage à un autre).

Un deuxième intérêt du schéma de cablage est d'être une approche de l'automatisation du cablage.

Nous proposons dans ce chapitre de donner quelques idées de reflexion sur l'élaboration d'un simulateur digital ou à opérateurs discrets, par la présentation du schéma de cablage et de quelques protocoles de communications entre opérateurs.

III.1 - DEFINITIONS |1|

III.1.1 - Mémoire de degré p

Une mémoire de degré p est une séquence de p cellules mémoires notée par l'une des écritures suivantes :

* ——•——•——•—— Mémoire de degré 4

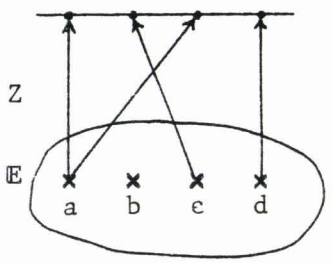
* $[p]$ Intervalle des entiers compris entre 1 et p

* $\langle p ; X_1, X_2, \ldots, X_i, \ldots, X_p \rangle$ Notation en système de variables

III.1.2 - Assignation de degré $[p]$

L'assignation de degré p est l'application Z de $[p]$ dans $\mathbb{E}$, où Z noté $(Z_1, Z_2, \ldots, Z_p)$ est le contenu d'une mémoire de degré p.

Exemple :



$Z : [4] \rightarrow \mathbb{E}$ et $Z(1) = Z_1 = a$

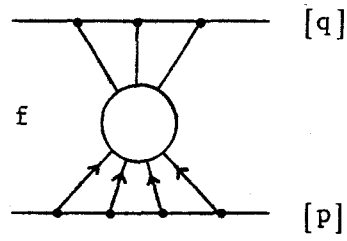
$Z(2) = Z_2 = c$

$Z(3) = Z_3 = a$

$Z(4) = Z_4 = d$

III.1.3 - Cellules opérateurs

Les cellules opérateurs sont des symboles fonctionnels de type $\binom{q}{p}$ et une cellule opérateur f possédant p entrées et q sorties est décrite par :



f : symbole fonctionnel de type $\begin{pmatrix} 3 \\ 4 \end{pmatrix}$.

Remarque :

$f \in F_p^q$ où f_p^q représente l'ensemble des opérateurs ayant p variables d'entrée et q variables de sortie.

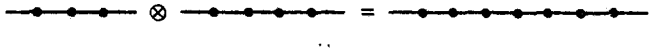
III.1.4 - Produit tensoriel

Le produit tensoriel, noté $\otimes$, permet la composition des cellules mémoires des cellules opérateurs et des assignations.

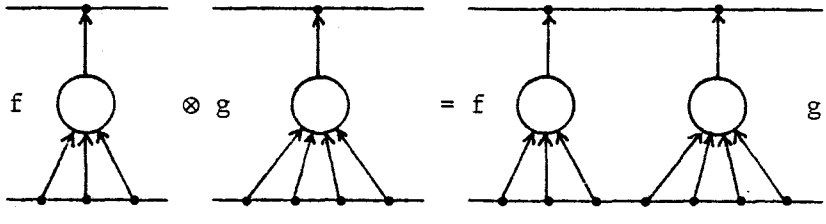
a) Cellules mémoires (concaténation) :

$[p] \otimes [q] = [p + q]$

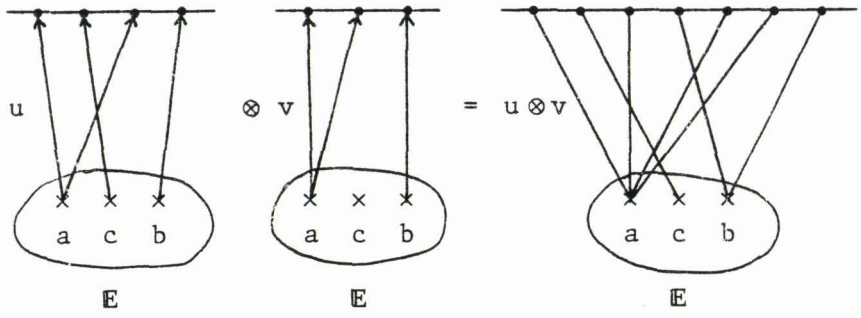
$< p ; X_1, X_2, \ldots, X_p > \otimes < q, X_1, X_2, \ldots, X_p > = < p + q ; X_1, X_2, \ldots, X_{p+q} >$



b) Cellules opérateurs :



c) Assignations :



L'application $u \oplus v : [p + q] \rightarrow \mathbb{E}$ est définie par :

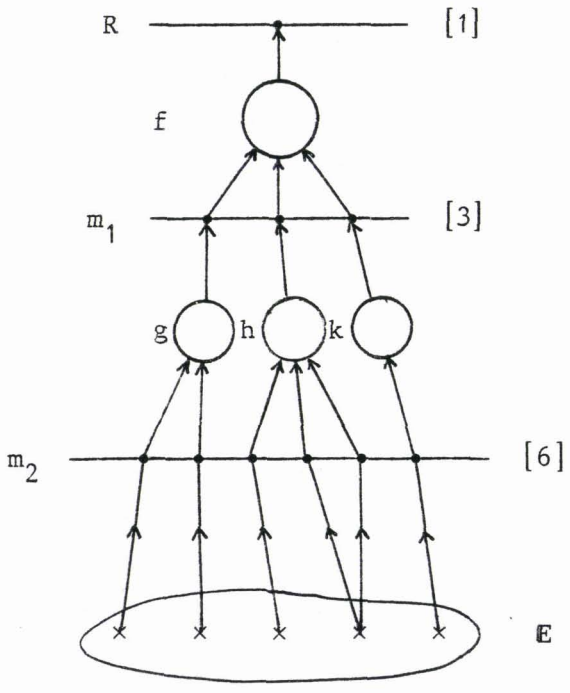
$$(u \otimes v)(i) = u(i) \quad \text{si} \quad i \in [p]$$

$$(u \otimes v)(p + j) = v(j) \quad \text{si} \quad j \in [q]$$

III.1.5 - Produit de composition (Représentation fonctionnelle)

Le produit de composition noté •, consiste à composer en série les machines abstraites, composées de cellules opérateurs et de cellules mémoires.

Exemple : Soit la machine abstraite T suivante :



Si les contenus des cellules mémoires m_1 et m_2 sont respectivement Y et Z, avec :

$$Y = < Y_1, Y_2, Y_3 >$$

et

$$Z = < Z_1, Z_2, Z_3, Z_4, Z_5, Z_6 >$$

nous avons les relations :

$$Y_1 = g(Z_1, Z_2)$$
$$Y_2 = h(Z_3, Z_4, Z_5)$$
$$Y_3 = k(Z_2)$$

et la machine abstraite T sera définie par :

$$T = f(Y) = f(Y_1, Y_2, Y_3)$$
$$T = f(g(Z_1, Z_2), h(Z_3, Z_4, Z_5), k(Z_6))$$

On voit que la description de la machine abstraite T fait intervenir tous les paramètres. Le produit tensoriel permet de corriger ce défaut. Ainsi, T peut être définie d'une manière équivalente :

$$T = f.(g(Z_1, Z_2) \otimes h(Z_1, Z_2, Z_3) \otimes k(Z_1))$$

ou

$$T = f.(g \otimes h \otimes k)$$

III.1.6 - Les torsions

Lorsque nous devons réaliser un cablage, il apparaît souvent la situation suivante :

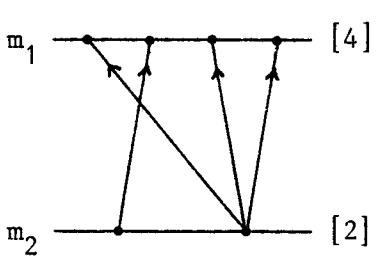


Figure III.1

La torsion peut être définie comme l'application $\theta : [p] \rightarrow [q]$ permettant de recopier le contenu des cellules mémoires d'une mémoire à l'autre, avec éventuellement des oublis des modifications dans l'ordre, des duplications.

Le cablage de la figure III.1 est décrit par la torsion $\theta : [4] \rightarrow [2]$, d'où $\theta \in \textcircled{H}_2^4$ (ensemble des applications de $[4] \rightarrow [2]$).

Dans l'exemple : $\theta(1) = \theta(3) = \theta(4) = 2$

$\theta(2) = 1$

et la tosion θ sera définie par :

$< 2 ; 2, 2, 1, 2 >$

c'est à dire

$< 2 ; \theta(1), \theta(2), \theta(3), \theta(4) >$

D'une manière générale, si θ est une application de $\textcircled{H}_p^q$, alors :

$$\theta = < p ; \theta(1), \theta(2), \ldots, \theta(q) >$$

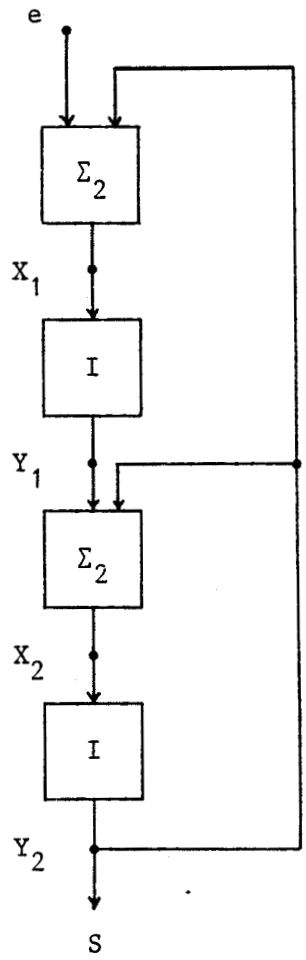
III.2 - SCHEMA DE CABLAGE ASSOCIE A UN PROCESSUS |1|

Considérons un processus P_1 et le modèle mathématique le décrivant :

$$(1) \quad \begin{cases} \dot{Y}_1 = Y_2 + e \\ \dot{Y}_2 = Y_1 + Y_2 \\ S = Y_2 \end{cases}$$

Si nous disposons d'opérateurs d'intégration et de sommation, le modèle mathématique (1) peut être représenté par le schéma de cablage suivant :

(S_1)



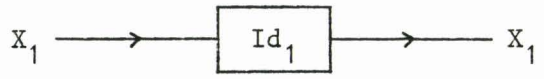
Ce schéma peut être décrit par le système formel d'équations :

$$(2) \quad \begin{cases} X_1 = \Sigma_2.(e \otimes Y_2) \\ Y_1 = I.(X_1) \\ X_2 = \Sigma_2.(Y_1 \otimes Y_2) \\ Y_2 = I.(X_2) \\ S = Y_2 \end{cases}$$

Les produits tensoriel et de composition permettent d'écrire le système (2) sous la forme :

$$(3) \quad < Y_1, Y_2, S > = < I.\Sigma_2.(e \otimes Id_1) \otimes I.\Sigma_2.(Id_1 \otimes Id_1) \otimes Id_1 > \\ < 3\,;2, 1, 2, 2 > < Y_1, Y_2, S >$$

où Id_1 représente l'opérateur identité de F_1^1 schématisé :



L'équation (3) décrit formellement le schéma de cablage (S_1) associé au processus P_1.

III.3 - APPLICATION DE LA TORSION AU CABLAGE

Si les entrées des cellules opérateurs sont les cellules mémoires d'une mémoire m_1, et les sorties des cellules opérateurs sont des cellules mémoires d'une mémoire m_2, selon le schéma suivant :

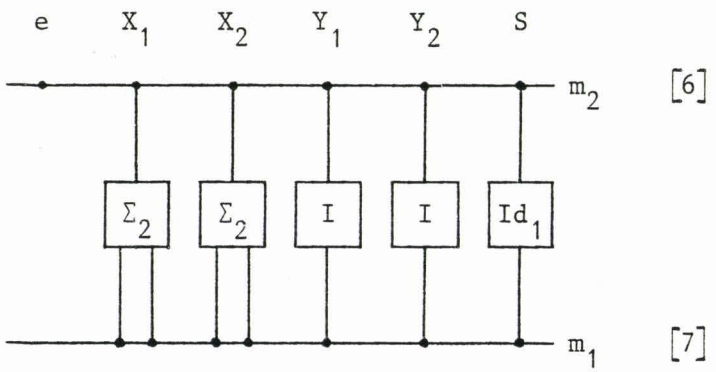
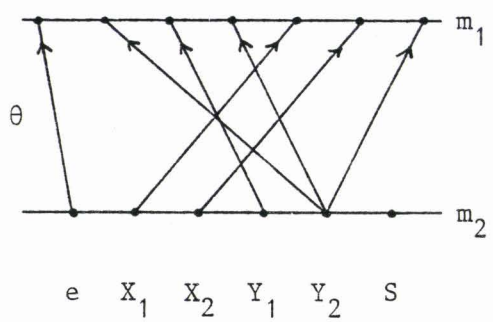


Figure III.2

le cablage du processus P_1, sur une calculatrice analogique consiste à réaliser la torsion θ : [7] → [6] suivante :



Le processus de simulation devient alors le suivant :

i) Réalisation de la torsion θ (cablage physique)

ii) Mise en conditions initiales

iii) Calcul

Sur un calculateur digital, la nature de l'information à traiter (codée) ne permet pas le même processus de simulation qu'en calcul analogique. Certaines phases, notamment la 3ème phase ou la phase de calcul, se font par étape. On peut envisager trois processus de simulation qui nous servirons de support pour le paragraphe suivant où quelques approches de communication entre opérateurs sont présentées.

III.4 - PROTOCOLES DE COMMUNICATION

Les opérateurs peuvent être de différents types, dont les plus utilisés sont de type F^1, F_1^1 et F_2^1.

Soit : NE, le nombre total des entrées des opérateurs et
NS, le nombre total des sorties des opérateurs.

Si on associe à chaque entrée des opérateurs, une cellule mémoire d'une mémoire M_1 et à chaque sortie une cellule mémoire d'une mémoire M_2 (une cellule mémoire peut être, à priori, un registre), alors :

M_1 est de degré NE et

M_2 est de degré NS.

La figure III.3 donne la représentation du système d'opérateurs où les opérateurs sont numérotés de 1 à N.

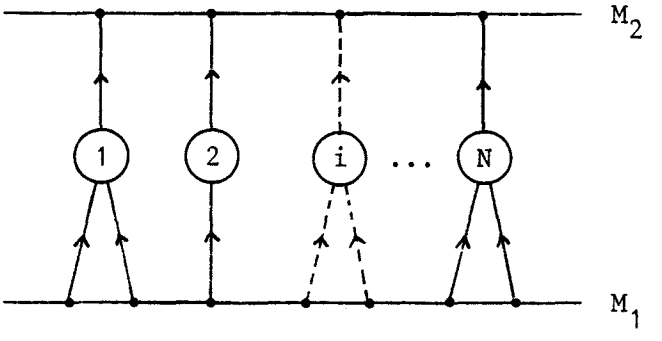


Figure III.3

On peut penser que le cablage d'un processus P_i se ramène toujours à la réalisation de torsion de [NE] → [NS]. Or, il peut arriver la situ-

ation suivante (lorsque des entrées d'opérateurs ne sont pas forcément liées à des sorties) :

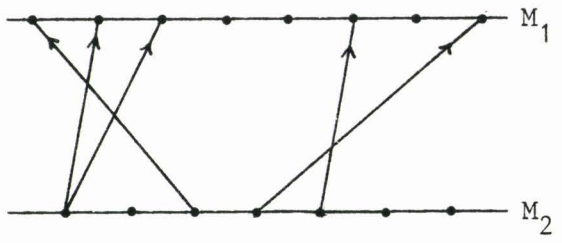


Figure III.4

On ne peut pas parler ici de torsion de [NE] → [NS], en ce sens su'une torsion est définie comme étant une application de [p] → [q].

Pour des raisons de compatibilité terminologique, nous introduisons une mémoire fictive M_3 de degré n variable, qui servira d'intermédiaire entre M_2 et M_1.

L'introduction de M_3 permet la réalisation du cablage de la figure III.4, selon le schéma suivant :

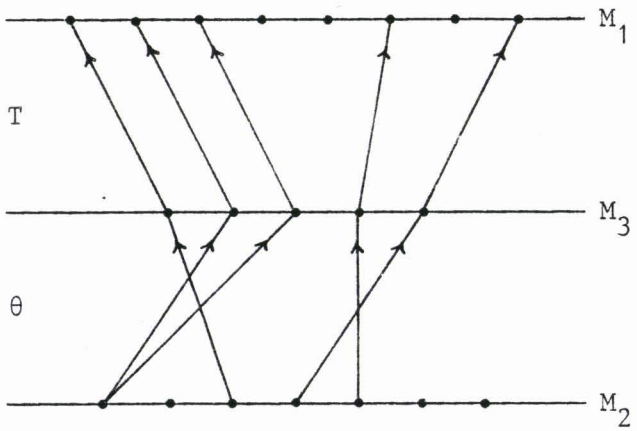


Figure III.5

Dans cet exemple :

θ est une torsion de [5] → [7] ou de $[M_3]$ → $[M_2]$

T est une application de [5] → [8] ou de $[M_3]$ → $[M_1]$

θ recopie le contenu des cellules mémoires de M_2 dans les cellules mémoires de M_3.

T recopie le contenu des cellules mémoires de M_3 dans les cellules mémoires de M_1.

III.4.1 - 1ère Approche du processus de simulation du système S_1

La première approche du processus de simulation du système S_1 décrit précédemment, consiste en une analogie avec le processus de simulation sur calculatrice analogique.

Les différentes étapes sont :

1 - Réalisation de la torsion θ

2 - Réalisation de l'application T

3 - Mise en conditions initiales

4 - Activations des opérateurs sommateurs Σ_2^1 et Σ_2^2
(calcul de $Y_1 + Y_2$ et de $Y_2 + e$)

5 - Activations des opérateurs intégrateurs I_1^1 et I_1^2
(calcul de $I.(X_1)$ et de $I.(X_2)$)

6 - Retour à l'étape n° 4 pour un nouveau pas de calcul

La configuration des opérateurs, de θ et T, est décrite ci-après (Fig. III.6).

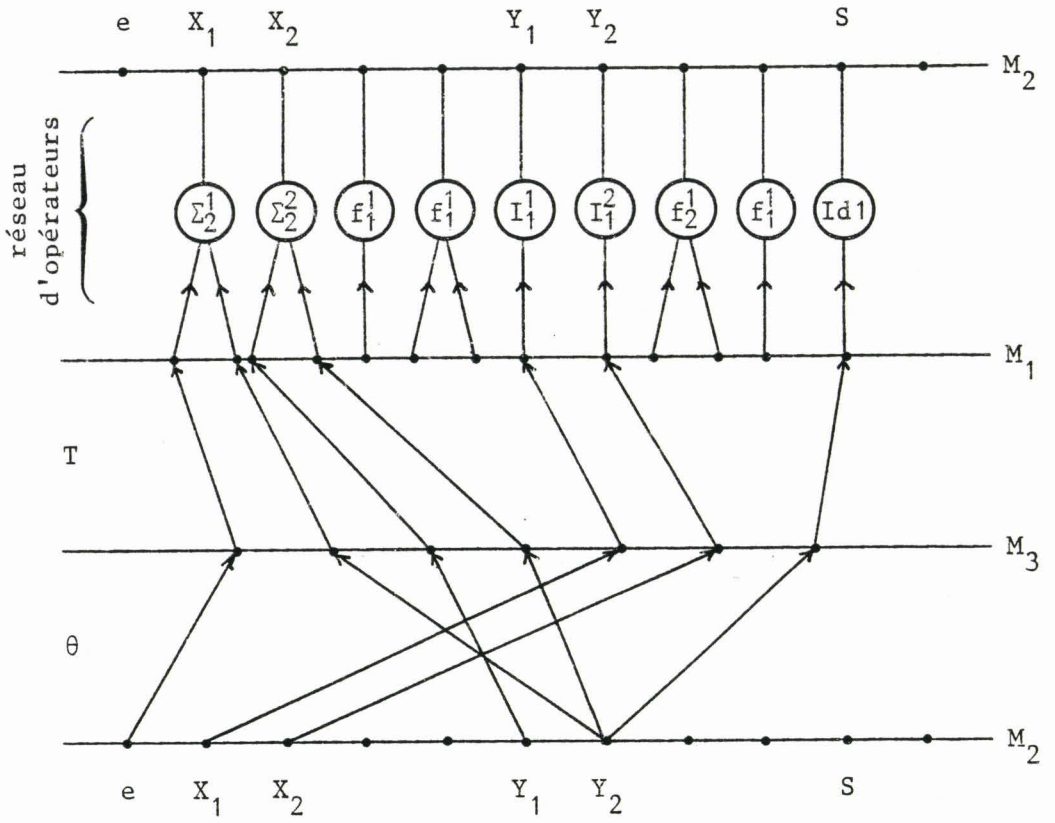


Figure III.6

Ce processus de simulation que nous venons de décrire suppose que la torsion θ et l'application T soient réalisées une fois pour toute et de façon permanente jusqu'à la fin d'un essai. Nous dirons que la communication entre M_2 et M_1 est directe ou physique : d'où les réseaux interconnexions comme première approche de communication entre opérateurs.

Un réseau d'interconnexion permute les données qui sont à ses entrées. Lors de la permutation de $N = 2^n$ éléments, par un réseau d'interconnexion à N entrées et N sorties, un chemin relie chaque entrée du réseau à une seule sortie. Le réseau peut être vu comme réalisant une bijection de l'ensemble

$$E = \{0, 1, \ldots, 2^n-1 = N-1\}$$

sur lui-même |6| ; ainsi, il réalise la permutation f des données présentées à ses entrées.

Lorsque f est activée, un chemin relie l'entrée i du réseau à la sortie f(i) pour tout i ($0 \leq i \leq 2^n-1$).

Il convient toutefois, de rappeler que dans un cablage, nous avons affaire à des torsions qui sont des applications généralement non bijectives. Ceci implique que l'utilisation d'un réseau d'interconnexion défini plus haut, comme moyen de cablage, doit s'accompagner d'un certain aménagement d'adaptation du réseau au système, ou inversement.

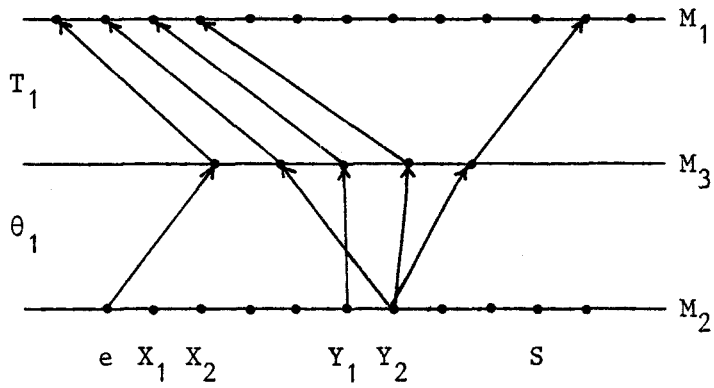
III.4.2 - 2ème Approche du processus de simulation du système S_1

Nous pouvons remarquer que les étapes 4 et 5 du processus de simulation décrit précédemment se font de manière séquentielle (ceci est du à la nature de l'information à traiter (codée)). Cette remarque peut également s'appliquer au cablage ; le processus de simulation devient donc :

1 - Mise en conditions initiales

2 - Réalisation de la torsion θ_1 de $[M_3] \rightarrow [M_2]$

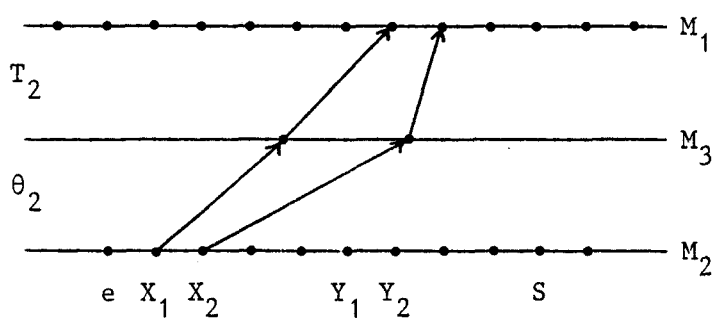
3 - Réalisation de l'application T_1 de $[M_3] \rightarrow [M_1]$:



4 - Activation des opérateurs sommateurs Σ_2^1 et Σ_2^2
(calcul de $Y_1 + Y_2$ et de $Y_2 + e$)

5 - Réalisation de la torsion θ_2 : $[M_3] \rightarrow [M_2]$

6 - Réalisation de l'application T_2 : $[M_3] \rightarrow [M_1]$



7 - Activation des opérateurs intégrateurs I_1^1 et I_1^2
(calcul de $I.(X_1)$ et de $I.(X_2)$)

8 - Retour à l'étape n° 2, pour un nouveau pas.

Remarque : A tout cablage ou transfert de données entre opérateurs, correspond deux opérations successives θ_i et T_i. Nous regroupons ces deux opérations en une seule, notée C_i^p : de $[M_2] \rightarrow [M_1]$, où p est le degré de M_3. D'où une deuxième approche de communication entre opérateurs :

Un processeur P_c que nous appelerons "processeur de cablage", jouant le rôle de M_3, c'est à dire procédant à la lecture de M_2 et à l'écriture de M_1 conformément à l'opération C_i^p. Le processeur P_c peut être mis sous le contrôle d'un processeur maître lui communiquant l'ordre et les informations nécessaires à la réalisation de C_i^p.

Voici un programme typique que le processeur P_c sera amené à exécuter :

```
Faire pour n = 1 à p
           j = k(n)
           i = ℓ(n)
           M3(n) = M2(j)
           M1(i) = M3(n)

      fait
```

où $\beta(x)$: $\beta \in \{k, \ell, M_3, M_1, M_2\}$

$x \in \{i, j, n\}$

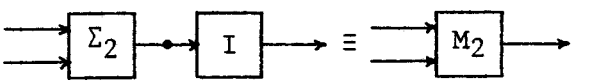
désigne le contenu de la cellule mémoire d'adresse x de la mémoire β.

Le scalaire p et les vecteurs k(n), ℓ(n) sont communiqués par le processeur maître au processeur de cablage P_c.

Nous appelerons cette technique de communication : "communication indirecte".

III.4.3 - 3ème Approche du processus de simulation du système S_1

Cette approche consiste en une composition d'opérateurs. Ainsi, par exemple, l'opération $I.\Sigma$ exécutée jusqu'à présent par un opérateur sommateur et un opérateur intégrateur, peut être exécutée par un opérateur équivalent $M = I.\Sigma$ représenté par le schéma suivant :



Le processus de simulation du système S_1 suit la chaîne de traitement suivante :

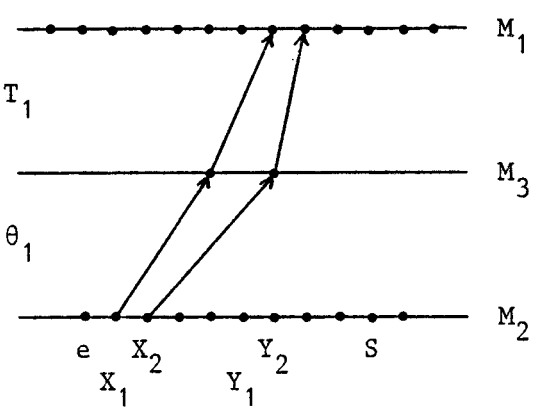
1 - Composition d'opérateurs :

Il s'agit de réaliser ici

$$M_2^1 = I_1^1.\Sigma_2^1 \quad \text{et} \quad M_2^2 = I_1^2.\Sigma_2^2$$

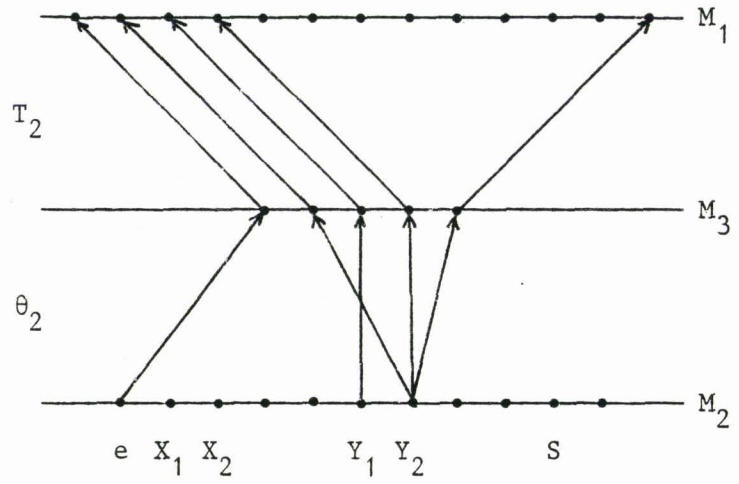
comme suit :

Réalisation de la torsion θ_1 de $[M_3] \rightarrow [M_2]$ et de l'application T_1 de $[M_3] \rightarrow [M_1]$ ou de l'opération C_1^2 suivante :



2 - Mise en conditions initiales

3 - Réalisation de l'opération C_2^5 suivante :



4 - Activation des opérateurs M_2^1 et M_2^2

5 - Retour à l'étape n° 3, pour un nouveau pas.

Dans ce processus de simulation, C_1^2 est réalisée de façon permanente durant la totalité d'un essai, selon la première approche de communication. Par contre, C_2^5 est réalisée à chaque pas de calcul selon la deuxième approche de communication. D'où une troisième approche possible :

- . Décomposer l'ensemble des opérateurs en sous-ensembles
- . Un sous-ensemble d'opérateurs formera un module
- . Un module peut être composé d'opérateurs de même type ou de types différents
- . Selon le choix des composants d'un module, la communication entre les opérateurs d'un module se fait selon la première approche, et la communication entre les modules se fait selon la deuxième approche ou inversement.

Nous appelerons ce type de communication : "communication mixte".

Cette méthode induit la possibilité d'agrégation ou de simplification du schéma |7|, mettant en évidence les sous-ensembles indépendants ou parallèles sur un pas de calcul.

La torsion est alors utilisée pour le recyclage des données selon un algorithme cablé ou programmé.

CONCLUSION

Nous avons donné dans ce chapitre, un bref rappel sur le schéma de cablage, dans le souci de souligner son intérêt dans l'élaboration d'un simulateur digital, où l'utilisateur proposera un modèle mathématique : la cablage et le choix des opérateurs lui seront transparents.

Dans l'expression formelle d'un schéma de cablage, les produits de composition et tensoriel mettent en évidence les opérateurs évoluant d'une façon séquentielle et ceux évoluant d'un façon parallèle. C'est d'ailleurs cette caractéristique qui nous a conduit à proposer l'approche de "communication mixte".

Ceci n'est pas sans rappeler les langages de programmation structurés où les procédures correspondent aux opérations C_i^p.

Rappelons enfin, que le schéma de contrôle |1|, ayant fait l'objet de nombreuses études, notamment dans le cadre du Laboratoire d'Informatique Industrielle de l'I. D. N. |8| |9| |10|, est nécessaire à l'implémentation d'une logique de commande efficace. Nous pensons par là à une éventuelle modélisation du simulateur par réseau de Pétri.

BIBLIOGRAPHIE DU CHAPITRE III

|1| D. CORBEEL
"Schéma de cablage et schéma de contrôle. Application à la simulation et à la gestion de processus industriels"
Thèse de Docteur de Spécialité (Informatique), Lille, 1979.

|2| A. ARNOLD
"Systèmes d'équations dans le magmoïde. Ensembles rationnels et algébriques d'arbres"
Thèse ès Sc. Math., Lille, 1977.

|3| A. ARNOLD, M. DACHET
"Théorie des magmoïdes"
Introductions communes aux Thèses ès Sc. Math. d'Arnold et de Dauchet, Lille, 1977.

|4| A. ARNOLD, M. DAUCHET
"Théorie des magmoïdes"
Colloque de Lille "Les arbres en algèbre et en programmation", 1976.

|5| M. DAUCHET
"Transductions de forêts. Biomorphismes et Magmoïdes"
Thèse d'Etat, Lille, 1977.

|6| H.J. SIEGEL
"Analysis techniques for SIMD machine interconnection networks and the effects of processor address masks"
IEEE Trans. on Comp., Vol. C-26, pp. 153-161, feb. 1977.

|7| D. CORBEEL, J.C. GENTINA
"Formal description of process : application to simulation"
U. K. S. C. Conference on Computer Simulation, Chester, avril 1978.

|8| C. VERCAUTER
"Sur un ensemble d'outils d'aide à la spécification et à la conception des systèmes industriels"
Thèse de Docteur de 3ème Cycle, Lille, 1982.

|9| N. LAHMAR
"Sur un système de gestion de processus multitâches : application à la commande et à la simulation"
Thèse de Docteur de 3ème Cycle, Lille, 1982.

|10| D. CORBEEL, C. VERCAUTER, J.C. GENTINA
"Méthodologie de description des systèmes de processus et de gestion d'erreur"
Convention Informatique Latine, 6/81, Barcelone.

"Formal description of processes system and execution handling"
IASTED International Symposium Modelling, Identification and Control, 18/21 fev. 1981, Davos.

"Specification and conception of real time control systems"
IASTED 09/81, Le Caire.

"Executif d'émulation du contrôle des processus en temps réel"
AMSE, 09/81, Lyon.

CONCLUSION GENERALE

L'objectif que nous nous sommes fixé, était d'apporter une réponse à la question suivante :

> Dispose-t-on, actuellement, de moyens techniques et théoriques, permettant d'élaborer des opérateurs numériques rapides, s'intégrant dans un ensemble informatique, en vue de la simulation de systèmes complexes, selon un modèle parallèle ou analogique ?

(Modèle analogique sous-entend ici, que la description du système, le câblage et le temps de calcul, soient proches de l'analogique).

A juger des différents cas étudiés, dans des situations très particilières (approche programmée à base du microprocesseur 8086, approche cablée à base de circuits S. S. I. et F. P. L. A.), la réponse est NON si on se limite à ces moyens. Autrement dit, le champ d'exploration des possibilités offertes par le développement de la micro-électronique reste ouvert (les éléments de calcul standards présentés en Annexe II en témoignent).

Toutefois, cette étude nous a permis de nous rendre compte des difficultés que posent encore les opérateurs arithmétiques sur le plan pratique, et qu'il convient de décrire par des algorithmes en vue d'une intégration probable à grande échelle. C'est là une de nos préoccupations futures.

Si on examine le problème en terme de temps de calcul, la satisfaction du critère temporel imposé aux éléments numériques, revient à dire qu'un pas de calcul (qui peut comporter plusieurs opérations en série) doit se faire en un temps T_N tel que :

$$T_N \leqq T_A/2 \quad \text{où} \quad T_A = 1/f_A$$

avec f_A : bande passante du calculateur analogique, qui est de l'ordre de 100 kHz.

Soit :

$$T_N \leq 1/2f_A = 1 / 2.100.10^3 = 50.10^{-6} = 50\ \mu s$$

Le temps T_N comporte les temps de transfert de données entre opérateurs (cablage), de commande, de calcul et de contrôle. Il serait, à notre avis, illusoire de penser qu'actuellement, on peut satisfaire cette contrainte.

Cependant, nous restons optimiste et continuons nos travaux dans le même esprit, d'une éventuelle réalisation du "simulateur digital", avec le souci constant de minimiser T_N, à tous les niveaux.

o

o o

ANNEXE I

Table A I.1 - Le connecteur P1 du Multibus

	(COMPONENT SIDE)			(CIRCUIT SIDE)		
	PIN[1,2]	MNEMONIC	DESCRIPTION	PIN[1,2]	MNEMONIC	DESCRIPTION
POWER SUPPLIES	1	GND	Signal GND	2	GND	Signal GND
	3	+5V	+5Vdc	4	+5V	+5Vdc
	5	+5V	+5Vdc	6	+5V	+5Vdc
	7	+12V	+12Vdc	8	+12V	+12Vdc
	9	-5V	-5Vdc	10	-5V	-5Vdc
	11	GND	Signal GND	12	GND	Signal GND
BUS CONTROLS	13	BCLK/	Bus Clock	14	INIT/	Initialize
	15		Reserved	16		Reserved
	17	BUSY/	Bus Busy	18		Reserved
	19	MRDC/	Mem Read Cmd	20	MWTC/	Mem Write Cmd
	21	IORC/	I/O Read Cmd	22	IOWC/	I/O Write Cmd
	23	XACK/	XFER Acknowledge	24	INH1/	Inhibit 1 disable RAM
BUS CONTROLS AND ADDRESS	25		Reserved	26		Reserved
	27	BHEN/	Byte High Enable	28	AD10/	
	29	CBRQ/	Common Bus Request	30	AD11/	Address
	31	CCLK/	Constant Clk	32	AD12/	Bus
	33	INTA/	Interrupt Acknowledge	34	AD13/	
INTERRUPTS	35	INT6/	Parallel	36	INT7/	Parallel
	37	INT4/	Interrupt	38	INT5/	Interrupt
	39	INT2/	Requests	40	INT3/	Requests
	41	INT0/		42	INT1/	
ADDRESS	43	ADRE/		44	ADRF/	
	45	ADRC/		46	ADRD/	
	47	ADRA/	Address	48	ADRB/	Address
	49	ADR8/	Bus	50	ADR9/	Bus
	51	ADR6/		52	ADR7/	
	53	ADR4/		54	ADR5/	
	55	ADR2/		56	ADR3/	
	57	ADR0/		58	ADR1/	
DATA	59	DATE/		60	DATF/	
	61	DATC/		62	DATD/	
	63	DATA/		64	DATB/	
	65	DAT8/	Data	66	DAT9/	Data
	67	DAT6/	Bus	68	DAT7/	Bus
	69	DAT4/		70	DAT5/	
	71	DAT2/		72	DAT3/	
	73	DAT0/		74	DAT1/	
POWER SUPPLIES	75	GND	Signal GND	76	GND	Signal GND
	77		Reserved	78		Reserved
	79	-12V	-12Vdc	80	-12V	-12Vdc
	81	+5V	+5Vdc	82	+5V	+5Vdc
	83	+5V	+5Vdc	84	+5V	+5Vdc
	85	GND	Signal GND	86	GND	Signal GND

1. All odd-numbered pins (1, 3, 5 . . . 85) are on component side of the board. Pin 1 is the left-most pin when viewed from the component side of the board with the extractors at the top. All unassigned pins are reserved.

Table A I.2 - Description des signaux du Multibus

Signal	Functional Description
ADR0/-ADRF/ ADR10/-ADR13/	*Address.* These 20 lines transmit the address of the memory location or I/O port to be accessed. For memory access, ADR0/ (when active low) enables the even byte (DAT0/-DAT7/) on the Multibus interface; i.e., ADR0/ is active low for all even addresses. ADR13/ is the most significant address ibt.
BCLK/	*Bus Clock.* Used to synchronize the bus contention logic on all bus masters. When generated by the iSBC 86/12A board, BCLK/ has a period of 108.5 nanoseconds (9.22 MHz) with a 35-65 percent duty cycle.
BHEN/	*Byte High Enable.* When active low, enables the odd byte (DAT8/-DATF/) onto the Multibus interface.
BPRN	*Bus Priority In.* Indicates to a particular bus master that no higher priority bus master is requesting use of the bus. BPRN/ is synchronized with BCLK/.
BPRO/	*Bus Priority Out.* In serial (daisy chain) priority resolution schemes, BPRO/ must be connected to the BPRN/ input of the bus master with the next lower bus priority.
BREQ/	*Bus Request.* In parallel priority resolution schemes, BREQ/ indicates that a particular bus master requires control of the bus for one or more data transfers. BREQ/ is synchronized with BCLK/.
BUSY/	*Bus Busy.* Indicates that the bus is in use and prevents all other bus masters from gaining control of the bus. BUSY/ is synchronized with BCLK/.
CBRQ/	*Common Bus Request.* Indicates that a bus master wishes control of the bus but does not presently have control. As soon as control of the bus is obtained, the requesting bus controller raises the CBRQ/ signal.
CCLK/	*Constant Clock.* Provides a clock signal of constand frequency for use by other system modules. When generated by the iSBC 86/12A board, CCLK/ has a period of 108.5 nanoseconds (9.22 MHz) with a 35-65 percent duty cycle.
DAT0-DATF/	*Data.* These 16 bidirectional data lines transmit and receive data to and from the addressed memory location or I/O port. DATF/ is the most-significant bit. For data byte operations, DAT0/-DAT7/ is the even byte and DAT8/-DATF/ is the odd byte.
INH1/	*Inhibit RAM.* For system applications, allows iSBC 86/12A board dual port RAM addresses to be overlayed by ROM/PROM or memory mapped I/O devices. This signal has no effect on local CPU access of its dual port RAM.
INIT/	*Initialize.* Resets the entire system to a known internal state.
INTA/	*Interrupt Acknowledge.* This signal is issued in response to an interrupt request.
INT0-INT7/	*Interrupt Request.* These eight lines transmit Interrupt Requests to the appropriate interrupt handler. INT0 has the highest priority.
IORC/	*I/O Read Command.* Indicates that the address of an I/O port is on the Multibus interface address lines and that the output of that port is to be read (placed) onto the Multibus interface data lines.
IOWC/	*I/O Write Command.* Indicates that the address of an I/O port is on the Multibus interface interface address lines and that the contents on the Multibus interface data lines are to be accepted by the addressed port.
MRDC/	*Memory Read Command.* Indicates that the address of a memory location is on the Multibus interface address lines and that the contents of that location are to be read (placed) on the Multibus interface data lines.
MWTC/	*Memory Write Command.* Indicates that the address of a memory location is on the Multibus interface address lines and that the contents on the Multibus interface data lines are to be written into that location.
XACK/	*Transfer Acknowledge.* Indicates that the addressed memory location has completed the specified read or write operation. That is, data has been placed onto or accepted from the Multibus interface data lines.

Table A I.3 - Le connecteur P2 du Multibus

Pin [1,2]	Signal	Definition
1	GND	Auxiliary common
2	GND	
3	+5V AUX	Auxiliary backup battery supply
4	+5V AUX	
7	-5V AUX	
8	-5V AUX	
11	+12V AUX	
12	+12V AUX	
19	PFI/	*Power Fail Interrupt.* This externally generated signal, which is input to the priority interrupt jumper matrix, should normally be connected to the 8085 CPU NMI input.
20	MEM PROT/	*Memory Protect.* This externally generated signal prevents access to the dual port RAM during backup battery operation.
21	GND	Auxiliary common
22	GND	
32	ALE	*Address Latch Enable.* This iSBC 86/12A board activates ALE during T1 of every CPU/ machine cycle. This signal may be used as an auxiliary address latch.
38	AUX RESET/	*Reset.* The externally generated signal initiates a power-up sequence; i.e., initializes the iSBC 86/12A board and resets the entire system to a known internal state.

1. All odd-numbered pins (1, 3, 5 . . . 59) are on component side of the board. Pin 1 is the left-most pin when viewed from the component side of the board with the extractors at the top.
2. Cable connector numbering convention may not agree with board connector numbering convention.

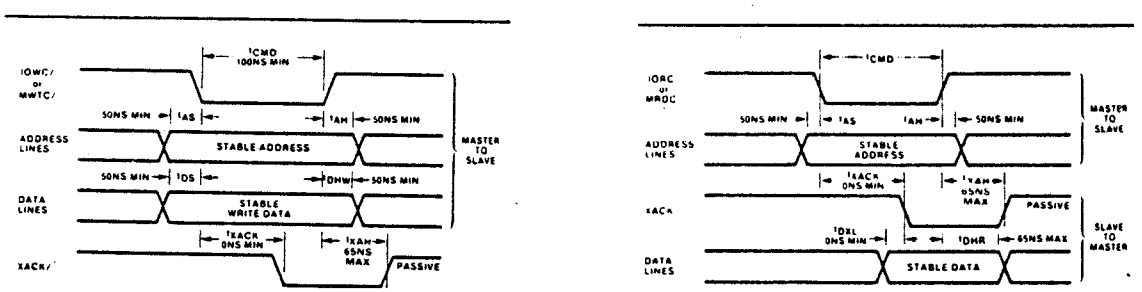


Figure A I.1 - Chronogrammes d'un cycle de lecture et d'écriture d'une donnée

N.B. : S/ signifie que le signal S est actif à l'état bas

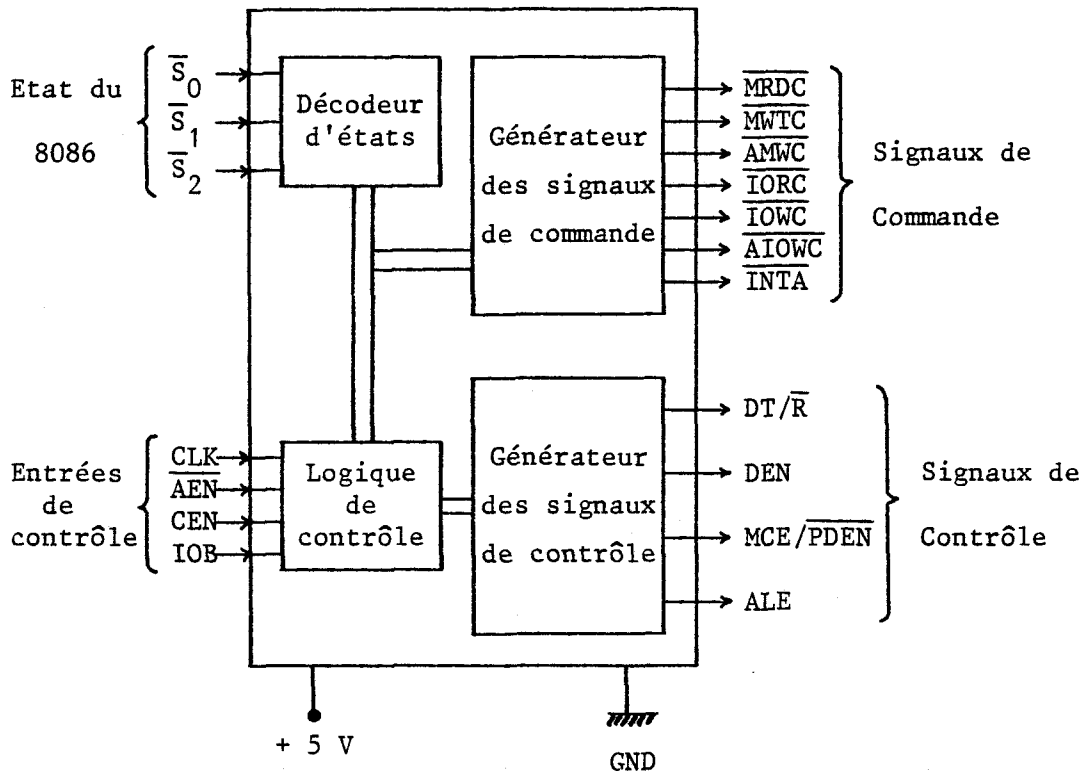
TRANSFERT DE DONNEES SUR MULTIBUS

La description de l'organisation du bus telle que faite dans le Chapitre I, et illustrée par les Tables A I.1 et A I.2, indique le rôle des différentes broches. La fonction principale de ces signaux est d'assurer des transferts de données entre les modules.

Un transfert (Fig. A I.1) débute par l'envoi de l'adresse d'une position de mémoire ou d'un port sur le bus d'adresse. Quand il s'agit d'une écriture, la donnée est placée simultanément sur le bus de données. Le module maître génère ensuite une commade de bus (Entrée ou Sortie pour un périphérique, lecture ou écriture pour une mémoire), qui va valider le module esclave concerné. Ce module envoie ensuite un signal de fin de transfert (après avoir lu ou déposé une donné) permettant au module maître, de terminer son cycle de lecture ou d'écriture.

En présence de deux modules maîtres, la même technique de transfert aura lieu après que le module maître demandant ce transfert acquiert l'accès au bus commun, selon le chronogramme de la figure I.10 du chapitre I.

LE 8288 : CONTROLEUR DE BUS



SCHEMA BLOC

Les signaux $\overline{S}_0$, $\overline{S}_1$ et $\overline{S}_2$ donnent l'état du processeur selon la table suivante :

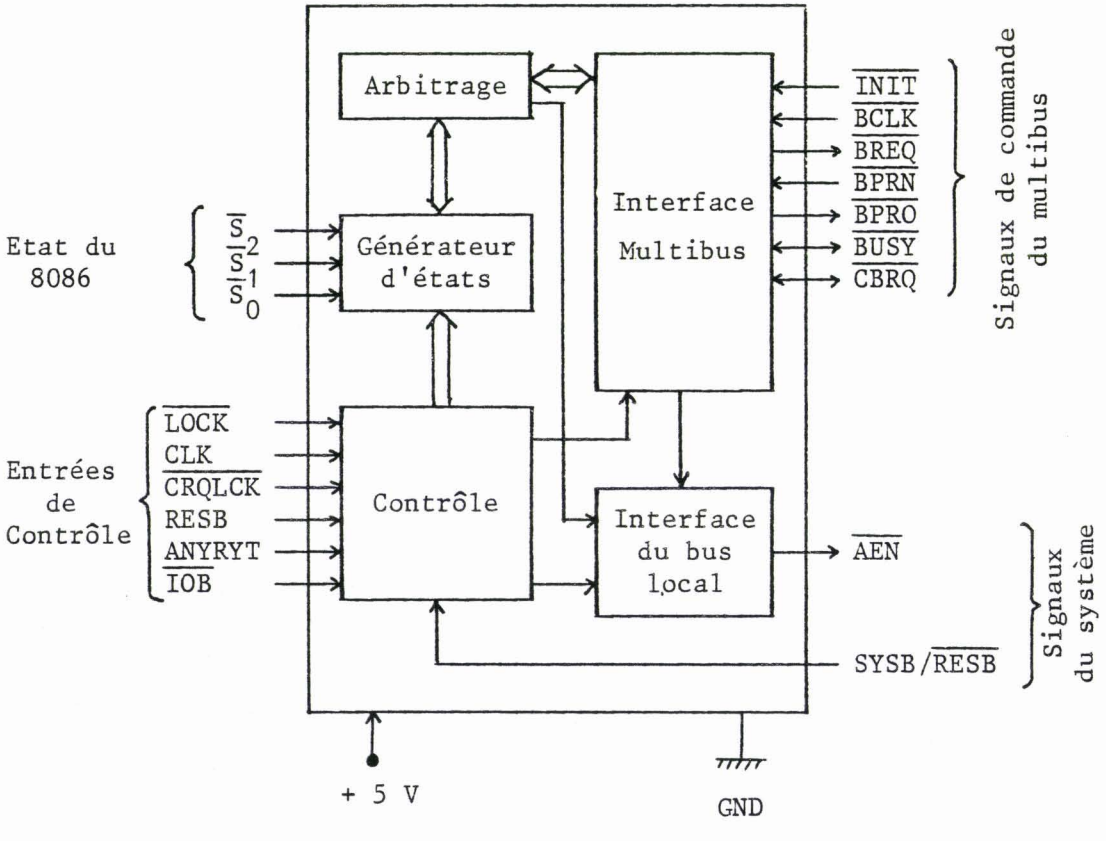
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	Etat du processeur	Commande du 8288
0	0	0	Demande d'interruption	$\overline{INTA}$
0	0	1	Lecture d'un port d'E/S	$\overline{IORC}$
0	1	0	Ecriture d'un port d'E/S	$\overline{IOWC}$, $\overline{AIOWC}$
0	1	1	Halt	Aucun signal de commande n'est actif
1	0	0	Accès au code	$\overline{MRDC}$
1	0	1	Lecture mémoire	$\overline{MRDC}$
1	1	0	Ecriture mémoire	$\overline{MWTC}$, $\overline{AMWC}$
1	1	1	Etat passif	Aucun signal n'est actif

Les signaux d'arbitrage sont :

$\overline{AEN}$: lorsque ce signal est actif (état bas), les signaux de commande peuvent être autorisés au bout de 150 ns minimum.

CEN : lorsque ce signal est à l'état bas, tous les signaux de commande ainsi que les signaux de contrôle DEN et $\overline{PDEN}$ sont forcés à leur état inactif. Lorsque ce signal est à l'état haut, tous les signaux de commande et de contrôle sont autorisés.

LE 8289 : ARBITRE DU BUS COMMUN



SCHEMA BLOC

Le mode de fonctionnement du 8289 est donné par les tableaux suivants :

	Signaux d'état du 8086			Mode d'E/S seulement	Mode bus résident seulement		Mode d'E/S et Mode bus résident		Mode bus unique
				$\overline{IOB}$ = 0	$\overline{IOB}$ = 1 , RESB = 1		$\overline{IOB}$ = 0 , RESB = 1		$\overline{IOB}$ = 1 , RESB = 0
	$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$		SYSB/$\overline{RESB}$ = 0	SYSB/$\overline{RESB}$ = 1	SYSB/$\overline{RESB}$ = 0	SYSB/$\overline{RESB}$ = 1	
COMMANDES E/S	0	0	0	X	✓	X	X	X	✓
	0	0	1	X	✓	X	X	X	✓
	0	1	0	X	✓	X	X	X	✓
HALT	0	1	1	X	X	X	X	X	X
COMMANDES MEMOIRE	1	0	0	✓	✓	X	✓	X	✓
	1	0	1	✓	✓	X	✓	X	✓
	1	1	0	✓	✓	X	✓	X	✓
PASSIF	1	1	1	X	X	X	X	X	X

MODE DE CONFIGURATION DU 8289	CONFIGURATION DES ENTREES DE CONTROLE	ETAT DU BUS COMMUN	
		Demandé**	Restitué*
Bus Unique Mode Multimaîtres	$\overline{IOB}$ = 1 RESB = 0	Pendant l'état actif des signaux d'états du microprocesseur	HLT + TI . CBRQ + HPBRQ†
Mode Bus Résident seulement	$\overline{IOB}$ = 1 RESB = 1	SYSB/$\overline{RESB}$ = 1 et les signaux d'états actifs	(SYSB/$\overline{RESB}$ = 0 + TI) . (CBRQ) + HLT + HPBRQ
Mode E/S seulement	$\overline{IOB}$ = 0 RESB = 0	Demande de lecture ou écriture de la mémoire	(état E/S) + TI) . CBRQ + HLT + HPBRQ
Mode E/S et Bus Résident	$\overline{IOB}$ = 0 RESB = 1	(Commande de la mémoire) . (SYSB/$\overline{RESB}$ = 1)	(Commande E/S + SYSB/$\overline{RESB}$ = 0) . CBRQ + HBRQ† + HLT

Notes :

X : le bus commun peut être restitué à un autre maître.

✓ : le bus commun est demandé par le processeur maître.

* : $\overline{\text{LOCK}}$ interdit la restitution du bus à n'importe quel autre processeur maître.
$\overline{\text{CRQLCK}}$ interdit la restitution du bus à un autre maître de plus faible priorité.

** : sauf si le processeur est à l'état HALT ou Passif.

† : bus commun demandé par un autre maître de plus haute priorité ou $\overline{\text{BPRN}} = 1$.

Le + est le OU logique.

Le • est le Produit logique.

TI correspond à l'état passif du processeur (S_0, S_1, S_2 = 1 1 1).

HLT correspond à l'état HALT du processeur (S_2, S_1, S_0 = 0 1 1).

Le dialogue entre le 8289 et le 8288 se fait par simple liaison de la sortie $\overline{\text{AEN}}$ du 8289. Lorsque le 8289 détermine que le processeur qui lui est associé demande l'accès au bus commun, et que celui-ci est libre ($\overline{\text{BUSY}} = 1$), il autorise le 8288 à valider les signaux de commande sur le bus commun par le signal $\overline{\text{AEN}}$.

ANNEXE II

LES RESEAUX LOGIQUES PROGRAMMABLES

L'utilisation généralisée de la logique programmée, fait quelquefois perdre de vue qu'elle n'est pas toujours la meilleure solution à un problème donné.

Son défaut majeur réside dans le fait qu'on bute, avec les microprocesseurs monolithiques, sur le problème de la vitesse. Cependant, si la complexité des opérations n'est pas importante, la logique cablée peut suffire.

Des constructeurs tels que Signetics et MMI proposent des outils commodes pour ce genre de technique de conception : les réseaux de portes programmables et les réseaux séquentiels programmables.

A - LES RESEAUX DE PORTES PROGRAMMABLES

a) Les PROM :

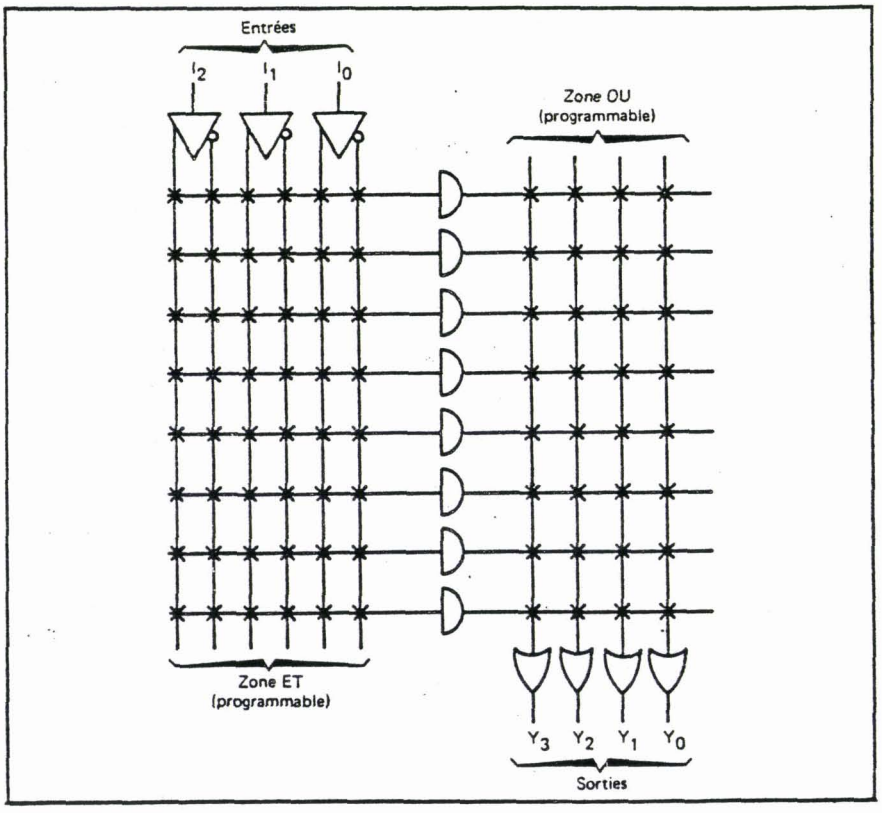
La Figure 1 montre la structure d'une PROM de 8 mots de 4 bits. Toutes les combinaisons booléennes des variables d'entrées sont décodées par le réseau de portes ET à 3 entrées de manière figée.

b) Les P L A (Programmable Logic Array) :

La structure d'un P L A est donnée par la Figure 2.

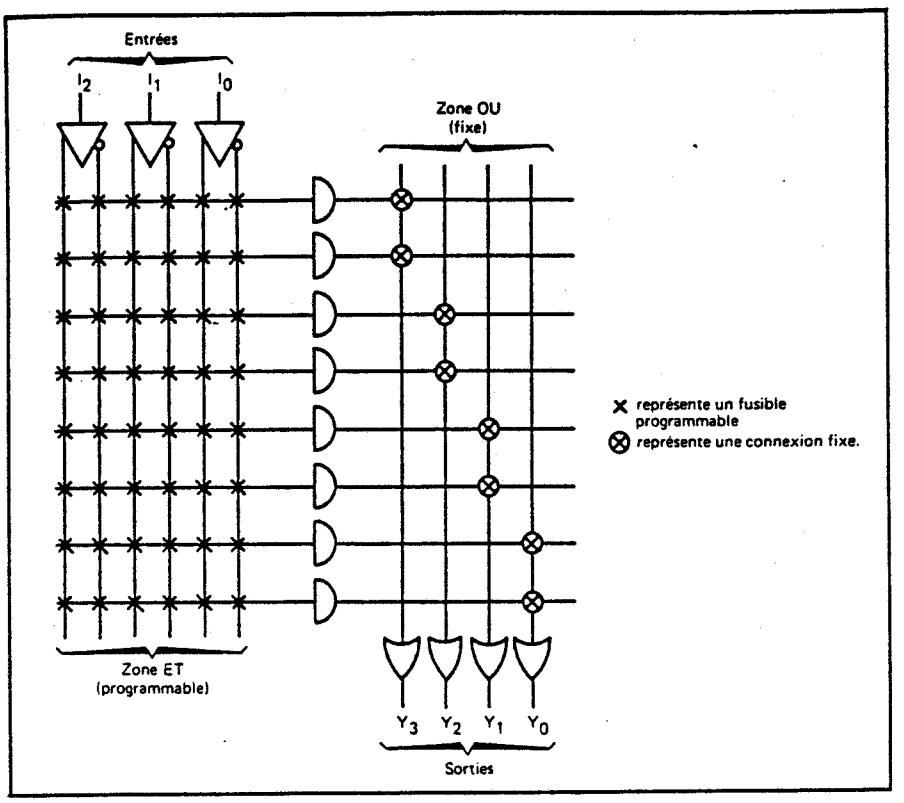
Dans les P L A, contrairement aux PROM, le réseau de portes ET est programmable par l'utilisateur. Dans une PROM de n variables d'adresse, il y a $N = 2^n$ produits, tous utilisables pour le calcul des fonctions. Dans un P L A de n variables d'entrées, il y a $N \ll 2^n$ termes produits utilisables : pour n = 16 entrées par exemple, il y a 48 termes produits seulement.

Fig. 2 - Représentation d'un réseau de portes ET-OU programmable (PLA). Le nombre de portes ET n'est plus lié au nombre de combinaisons d'entrées.



c) Le P A L (Programmable Array Logic) :

La structure du P A L est un peu plus simple que celle des P L A (voir Fig. 3). On a toujours le réseau programmable de portes ET comme dans un P L A ; par contre, le réseau de portes OU en sortie n'est plus programmable. Une fonction de sortie ne peut recevoir que certains monômes.



▲

Fig. 3 - Représentation d'un PAL dans lequel les produits sont distribués seulement sur certaines sorties.

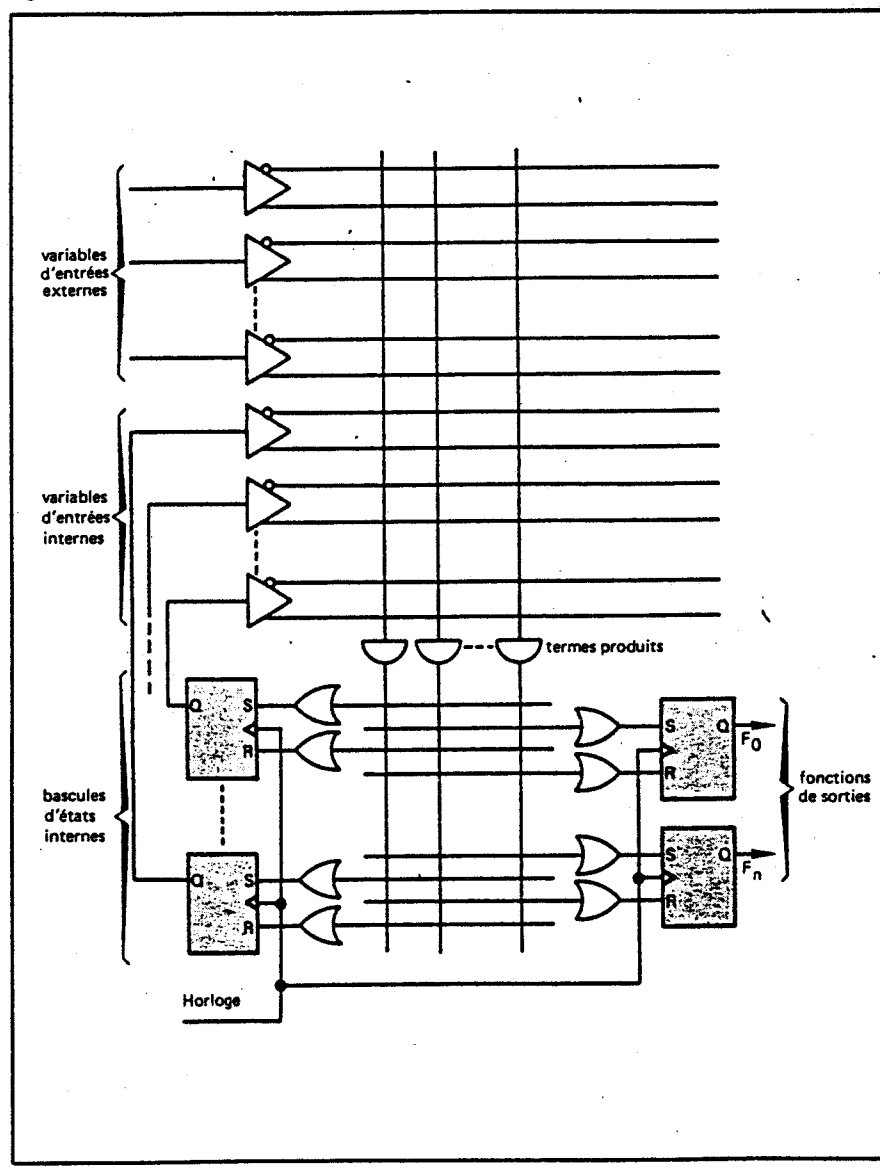
B - LES PRODUITS SEQUENTIELS PROGRAMMABLES

Ces circuits comprennent :

- Un réseau programmable de portes ET
- Un réseau de portes OU programmable entièrement ou fixe, selon les constructeurs
- Des fonctions de sortie sous forme combinatoire (sorties du réseau de portes OU) ou sous forme mémorisée (les sorties du réseau de portes OU alimentent des bascules de sortie)
- Un réseau de bascules d'état interne, recevant des fonctions de commande issues du réseau de portes OU
- Les états internes des bascules sont réinjectés dans le réseau de portes ET au même titre que les variables externes.

Les Figures 4 et 5 donnent les schémas de principe d'un F. P. L. S. de type Signetics et MMI respectivement.

Fig. 4 - Schéma de principe d'un PLS de type Signetics.



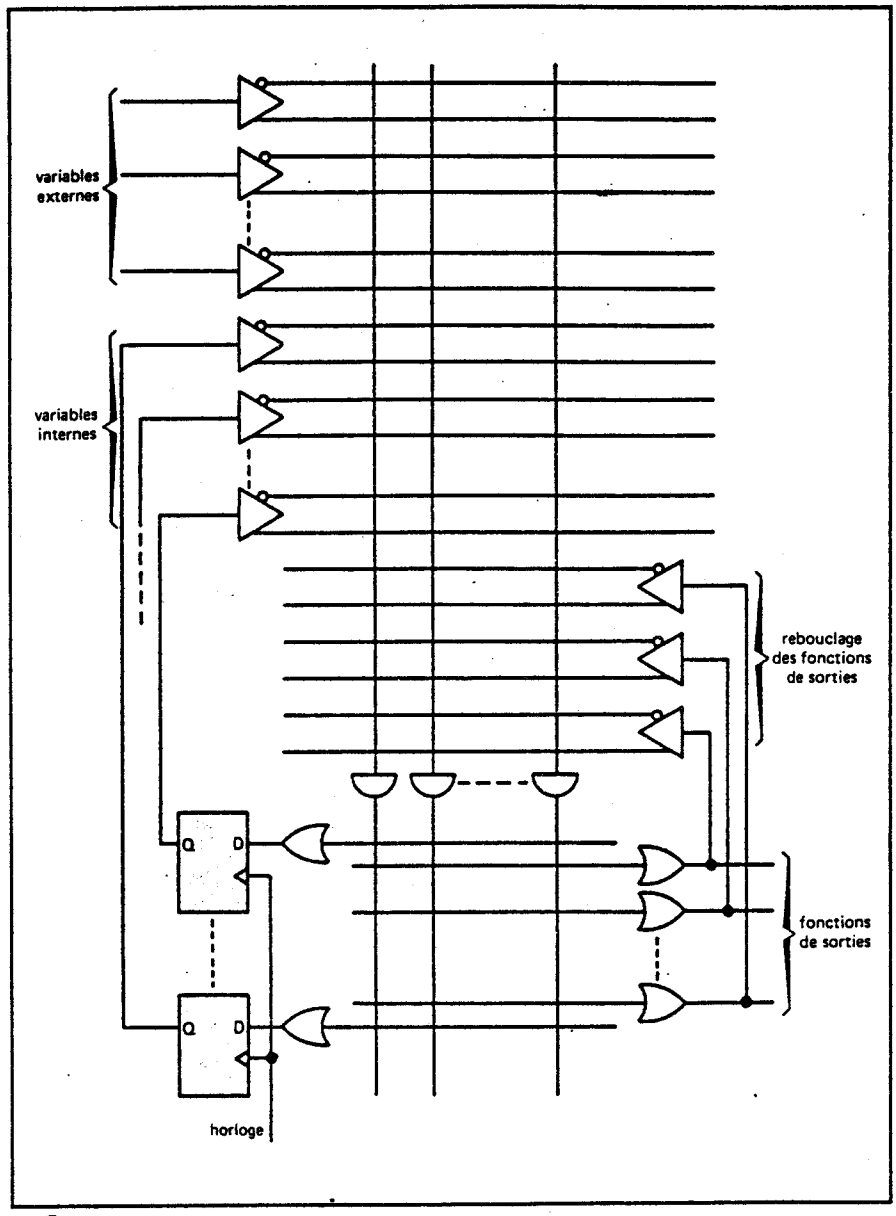
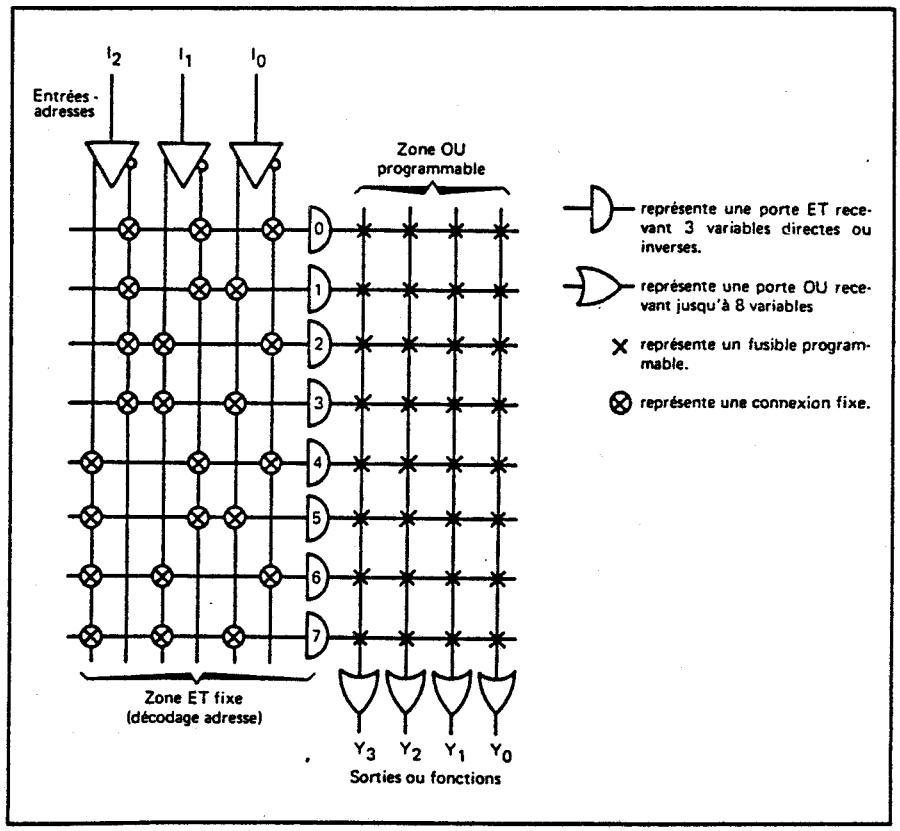


Fig.5 - Schéma de principe d'un type MMI.

Fig. 1 - Représentation simplifiée d'une mémoire Prom comme un réseau de portes ET à trois entrées et de portes OU connectées à chacune de leur sortie.



La porte "0" décode la valeur $\overline{I}_2$, $\overline{I}_1$, $\overline{I}_0$ (où $I_2 = I_1 = I_0 = 0$).

La porte "4" décode la valeur I_2, $\overline{I}_1$, $\overline{I}_0$ (soit 1 0 0), etc...

Ensuite, les sorties des portes ET sont reliées au réseau de portes OU, au moyen de fusibles programmables par l'utilisateur.

Le nombre de fonctions de n variables est de la forme :

$$N = 2^{2^n}.$$

Ainsi, avec le réseau de la Figure 1, on peut réaliser :

$$2^{2^3} = 2^8 = 256 \text{ fonctions différentes.}$$

LES ELEMENTS ARITHMETIQUES STANDARDS

A - L'ADDITION

A l'aide de 4 unités arithmétiques et logiques de type 74 S 381 et d'un générateur de retenue anticipée de type 74 S 182, on peut réaliser un additionneur 16 bits avec un temps de calcul typique, égal à 27 ns.

La table ci-dessous donne les caractéristiques d'un additionneur 16 bits et 32 bits, à l'aide des circuits 74 S 381 et 74 S 182.

	Nombre de boitiers	Nombre de connexions	Temps de calcul typique	Puissance consommée typique
Additionneur 16 bits	4 × 74 S 381 1 × 74 S 182	4 × 20 + 1 × 16 = 96	27 ns	2,36 W
Additionneur 32 bits	8 × 74 S 381 3 × 74 S 182	8 × 20 + 3 × 16 = 208	37 ns	4,98 W

TABLE A II.1

B - LA MULTIPLICATION

Rappelons que les multiplieurs se décomposent en deux grandes familles de produits :

- Les multiplieurs parallèles très rapides (50 à 200 ns)
- Les multiplieurs séquentiels plus lents (1 à 2 µs).

Les Tables A II.2 et A II.3, extraites de Minis et Micros (n° 136), établies par M. BACONNIER, donnent respectivement les principaux produits commercialisés et les possibilités d'extension cablée offertes par les principaux multiplieurs.

Type	Modèle	Fabricant	Configuration	Vitesse (typ)		Consom. en W		Boîtier	Représentation des données	Observations
				Fréquence max	Nombre de cycles d'horloge	typ.	max			
Séquentiel	25 LS 14	AMD	8 x 1	25 MHz	8 + n	0,45	0,75	16	Compl. à 2	— Type série parallèle — Multiplicateur de n bits sur entrée série — Sortie série pour le résultat
	25 LS 2516	AMD	8 x 8	20 MHz	8	1,1	1,6	40	Compl. à 2	— Multiplication avec accumulation pour une mise en cascade — Séquence externe
	67508	MMI	8 x 8	8 MHZ	4	1,1		24	Compl. à 2	— Pas d'extension câblée — Séquenceur incorporé — Offrent également la division — Bus d'E/S unique orienté microprocesseurs 8 ou 16 bits
	67516	MMI	16 x 16	8 MHZ	8	1,6		24	Compl. à 2	
Parallèle	25 S 05	AMD	2 x 4	25 ns		0,6	0,9	24 br.	Compl. à 2	— Multiplieur avec accumulation (X.Y + K) pour mise en cascade
	93 S 43	Fairchild								
	10183	Motorola	2 x 4	11 ns			0,9	24 br.	Compl. à 2	— Technologie ECL — Multiplieur avec accumulation (X.Y + K) pour mise en cascade
	74 S 274	Texas	4 x 4	50 ns		0,5	0,8	20 br.	Binaire pur	— L'extension du format fait appel aux circuits 74 S 275 (arbre de Wallace)
	67558	MMI	8 x 8	100 ns		0,9	1,4	40 br.	Binaire pur ou Compl. à 2	— Les 2 circuits sont compatibles — Extension possible du format par batterie d'additionneurs
	25 S 558	AMD		45 ns						
	25 S 557	AMD	8 x 8	50 ns			1,4	40 br.	id.	Idem 25 S 558 sauf sortie latchée
	MPY-8HJ	TRW	8 x 8	45 ou 65 ns		1	1,3	40 br.	Compl. à 2	— Registres d'E/S incorporés — Pas d'extension câblée
	MPY-12HJ	TRW	12 x 12	80 ns		2	2,7	64 br.	Binaire pur ou Compl. à 2	— Registres d'E/S — Extension aisée du format avec avec les additionneurs — Possibilité de décalage sur le résultat (MPY-24HJ uniquement) — Boîtier pourvu d'un radiateur
	MPY-16HJ	TRW	16 x 16	100 ns		3	4	64 br.		
	MPY-24HJ	TRW	24 x 24	200 ns		3,5	4,2	64 br.		
	TDC 1008 J	TRW	8 x 8	70 ns		1,2	1,6	48 br.	Binaire pur ou Compl. à 2	Il s'agit de multiplieurs avec accumulation (3 bits) orientés vers le traitement numérique du signal
	TDC 1009 J	TRW	12 x 12	95 ns		2,4	3,2	64 br.		
	TDC 1010 J	TRW	16 x 16	115 ns		3,4	4,5	64 br.		

Table A II.2 - **Panorama des principaux circuits intégrés multiplieurs du marché**

	Multiplieur de base employé	Format de l'opérateur de multiplication à réaliser											
		8 × 8			16 × 16			24 × 24			32 × 32		
		Nb de boîtiers	Vitesse (typ)	Consom.	Nb de boîtiers	Vitesse (typ)	Consom.	Nb de boîtiers	Vitesse (typ)	Consom.	Nb de boîtiers	Vitesse (typ)	Consom.
AMD	25 LS 14	1 + 2	650 ns	0,8 W typ	2 + 4	1,3 µs	1,6 W typ						
AMD	25 LS 2516	1	400 ns	1,1 W typ	2	800 ns	2,3 W typ						
MMI	67508	1	500 ns	1,1 W typ									
MMI	67516	—	—	—	1	1 µs	1,6 W typ						
AMD	25 S 05	8	75 ns	5 W typ	32	110 à 150 ns	30 W max						
MOT	10183	8	50 ns	7 W max	32	90 à 120 ns	30 W max						
TEXAS	74 S 274	4 + 8	75 ns	6 W typ	16 + 29	120 ns	30 W max						
MMI	67558	1	100 ns	0,9 W typ	4 + 13	135 ns	18 W max	9 + 33	170 ns	45 W max	16 + 63	190 ns	85 W max
AMD	25 S 558	1	45 ns	0,9 W typ	4 + 13	80 ns	18 W max	9 + 33	115 ns	45 W max	16 + 63	135 ns	85 W max
TRW	MPY-8 HJ	1	45 ns ou 65 ns	1 W typ									
TRW	MPY-12 HJ	—	—	—	—	—	—	4 + 19	125 ns	30 W max	—	—	—
TRW	MPY-16 HJ	—	—	—	1	100 ns	4 W max	—	—	—	4 + 25	140 ns	41 W max
TRW	MPY-24 HJ	—	—	—	—	—	—	1	200 ns	4,2 W max			

Table A II.3 - **Eléments de choix pour la constitution d'opérateurs de multiplication allant de 8 x 8 bits à 32 x 32 bits**

En ce qui concerne le nombre des boitiers dans la Table A II.3, le premier chiffre indique le nombre de multiplieurs de base. Le second chiffre indique le nombre approximatif de circuits MSI Standards (additionneurs, générateurs de retenue anticipée, registres à décalage, ...) nécessaires pour la sommation des produits partiels.

Il est à rajouter sur la liste, le circuit 29516 apparu chez AMD, qui est un multiplieur parallèle 16 × 16 bits combinatoire (50 ns Typ.) en un boitier 64 broches et compatible avec le MPY 16 HJ de TRW.