EPREUVE COMMUNE DE TIPE

TITRE: LES ARCHITECTURES SYSTOLIQUES

Temps de préparation : 2 h 15

Temps de présentation devant le jury : 10 minutes

Entretien avec le jury : 10 minutes

GUIDE POUR LE CANDIDAT :

Le dossier ci-joint comporte :

Document principal : 7 pages

Documents complémentaires : 4 pages

Travail suggéré au candidat:

- Montrer les premières pulsations du réseau de calcul de convolution récursive.
- Mettre en évidence, en s'appuyant sur les exemples donnés, l'intérêt et les limites des architectures systoliques.

CONSEILS GENERAUX POUR LA PREPARATION DE L'EPREUVE :

- Lisez le dossier en entier dans un temps raisonnable
- Réservez du temps pour préparer l'exposé devant le jury.

LES ARCHITECTURES SYSTOLIQUES

I. Introduction

Pour augmenter les performances des calculateurs, l'approche traditionnelle consiste à rechercher une augmentation de la vitesse des composants matériels et repose sur un système d'exploitation sophistiqué pour minimiser les périodes d'horloge. Aujourd'hui, cette approche est remise en cause. Tout d'abord l'industrie des semi-conducteurs est d'une manière générale plus à même de développer des technologies à très haute densité d'intégration que de réaliser des circuits ultra rapides. Ensuite, on a atteint des limites physiques qui semblent incontournables jusqu'à l'arrivée sur le marché des technologies très rapides de la prochaine génération (à l'arséniure de gallium, GaAs) ; de plus, le prix à payer pour gagner un ordre de grandeur sur la vitesse des composants en utilisant ces nouvelles technologies sera vraisemblablement prohibitif pour la plupart des applications.

Plutôt que de compter seulement sur l'augmentation de la vitesse des composants, une autre possibilité est d'opter pour des systèmes parallèles qui pourront être implantés efficacement sous forme de circuit intégrés VLSI (Very Large Scale Integration) : la complexité de la conception de ces circuits doit rester dans le cadre des meilleures possibilités industrielles.

De fait, la complexité des circuits intégrés disponibles à l'heure actuelle rend possible la réalisation à un faible coût de tels systèmes parallèles. Deux restrictions cependant :

- il s'agit de processeurs spécialisés que l'on adjoint à un processeur hôte de type conventionnel ;
- la classe d'application est bien délimitée : les problèmes où le volume de calculs à effectuer prime largement sur les transferts de données à réaliser.

Le modèle systolique, introduit en 1978 par Kung et Leiserson, s'est révélé être un outil puissant pour la conception de processeurs intégrés spécialisés. En un mot, une architecture systolique est agencée en forme de réseau. Ces réseaux se composent d'un grand nombre de

cellules élémentaires identiques et localement interconnectées. Chaque cellule reçoit des données en provenance des cellules voisines, effectue un calcul simple, puis transmet les résultats, toujours aux cellules voisines, un temps de cycle plus tard. Pour fixer un ordre de grandeur, disons que chaque cellule a la complexité, au plus, d'un petit microprocesseur.

Les cellules évoluent en parallèle, en principe sous le contrôle d'une horloge globale (synchronisme total) : plusieurs calculs sont effectués simultanément sur le réseau, et on peut "pipeliner" la résolution de plusieurs instances du même problème sur le réseau. Le "pipelining" est une technique de parallélisation des tâches dont on peut donner un exemple dans le domaine de la construction automobile avec les chaînes de fabrication : chaque tâche est réalisée sur un poste différent, et ainsi un grand nombre d'automobiles sont simultanément en cours de fabrication sur la chaîne.

La dénomination "systolique" provient d'une analogie entre la circulation des flux de données dans le réseau et celle du sang humain, l'horloge qui assure la synchronisation constituant le "cœur" du système.

II. Deux exemples simples

II.1 Convolution non-récursive

Partons du problème suivant : étant donné une suite x_1, x_2, x_3, … de données, calculer pour tout $i \geq k$ la valeur de $y_i = a_1 \times x_i + a_2 \times x_{i-1} + a_3 \times x_{i-2} + \ldots + a_k \times x_{i-k+1}$, où les poids a_1, a_2, …, a_k sont des coefficients fixés.

Une solution est décrite sur la figure 1 avec $k = 4$. Notons qu'un nouvel y_i est calculé tous les temps de cycle, τ_k, mais que ce temps de cycle doit être assez long pour permettre la réalisation d'une multiplication et de k-1 additions.

Sur le réseau systolique de la figure 2, composé de k cellules, un nouvel y_i n'est délivré en sortie qu'un temps de cycle sur deux, mais le temps de cycle τ_{syst} du réseau est celui de chaque

cellule, correspondant à une multiplication suivie d'une addition. Pour k assez grand, $\tau_k > 2\tau_{syst}$, et le réseau systolique est plus performant. Bien mieux, τ_{syst} ne dépend pas de k.

Quelques pulsations de ce réseau sont données figure 3 pour illustrer le principe du calcul "par accumulations successives": dans chaque cellule du réseau est effectué un calcul partiel $y_i := y_i + a_j \times x_{i-j+1}$, et plusieurs variables y_i sont calculées en parallèle (le réseau fonctionne en pipe-line).

II.2 Convolution récursive

Soit maintenant à calculer l'expression récurrente suivante :

$$y_i = a_1 \times y_{i-1} + a_2 \times y_{i-2} + a_3 \times y_{i-3} + \ldots + a_k \times y_{i-k},$$

pour $i \geq k+1$, et où $y_1, y_2, \ldots, y_k$ sont des valeurs initiales données.

Comme indiqué figure 4, moyennant l'ajout d'une cellule de retard, une initialisation adéquate des registres de délai, ainsi qu'une inhibition des multiplieurs par a_1 et a_2 durant les deux premiers cycles, le même réseau systolique peut être utilisé : quand une variable y_i circule de droite à gauche, elle est en train d'accumuler sa valeur définitive, puis elle rebondit dans la cellule de retard D_5 pour circuler inchangée en sens inverse, jouant le rôle des x_i dans le réseau précédent.

III. Triangularisation systolique de systèmes linéaires

Soit A une matrice carrée d'ordre n, et $AX = B$ un système linéaire à résoudre. La plupart des méthodes directes sont des algorithmes séquentiels qui procèdent en deux étapes : triangularisation de la matrice A bordée du vecteur B par prémultiplication par des matrices convenables, puis résolution du système linéaire obtenu. Formellement, on peut écrire :

Programme de triangularisation du système

Soit $A' = (A,B)$ matrice de taille $n \times (n + 1)$

pour i : = 1 à n faire

pour j : = i+1 à n faire la tâche T_{ij} : $\begin{pmatrix} \text{ligne } i \\ \text{ligne } j \end{pmatrix} := M_{ij} \begin{pmatrix} \text{ligne } i \\ \text{ligne } j \end{pmatrix}$

où M_{ij} est déterminée de manière à annuler le coefficient a_{ji}.

Par exemple en choisissant $M_{ij} = \begin{pmatrix} 1 & 0 \\ l_{ji} & 1 \end{pmatrix}$, avec $l_{ji} = -\frac{a_{ji}}{a_{ii}}$, on retrouve la méthode de Gauss sans pivotage. Ces méthodes ont un coût de $n^3/3$ opérations élémentaires.

Ces algorithmes peuvent être parallélisés en recherchant le maximum de tâches indépendantes, que l'on mènera simultanément. Les contraintes de séquencement imposées par la méthode de Gauss sans pivotage sont : T_{ij} précède $T_{i+1,j}$ pour $j > i$. Par exemple, T_{12}, T_{13}, …, T_{1n} peuvent être exécutés en parallèle.

Pour définir une architecture systolique réalisant la triangularisation, il faut se doter de trois types de cellules (voir la figure 5) :

- des cellules de générations, capables de calculer les coefficients des matrices M_{ij},
- des cellules de combinaison, capables de recevoir, combiner et transmettre les informations en provenance des cellules voisines,
- et des cellules de retard.

Ainsi, pour la triangularisation d'une matrice A de taille n, bordée du vecteur B dont les composantes sont notées $b_i = a_{i,n+1}$, on peut définir un réseau comportant $n(n+1)/2$ cellules de combinaison, n cellules de génération et n-1 cellules de retard (voir la figure 6). La matrice A est introduite par la partie supérieure du réseau, un port d'entrée recevant tous les éléments d'une même colonne. Lorsqu'au cycle i, a_{1i} ($1 \leq i \leq n+1$) arrive dans le réseau, il est stocké dans le registre interne de la cellule qui le reçoit en entrée ($i^{\text{ième}}$ cellule de la première ligne); l'instant d'après, cette cellule dispose de a_{1i} et a_{2i} comme données, et aussi de la matrice M_{12}

générée par la cellule recevant a_{11} : elle dispose ainsi de tous les éléments pour modifier a_{2i} suivant la transformation M_{12} conduisant à l'annulation de a_{21}. Le même raisonnement s'applique pour les M_{1i}, $i \geq 3$. De manière analogue, les cellules de la ligne j permettront l'annulation des coefficients a_{kj}, $k > j$, et la modification correspondante des éléments a_{lm} avec $l,m \geq j+1$. Le système est ainsi triangularisé en $3n$ temps de cycles.

IV. Pourquoi des architectures systoliques ?

IV.1. Concurrence et communication

Comme on l'a vu dans les exemples précédents, plusieurs calculs sont effectués sur une même donnée à l'intérieur du réseau : une fois qu'une donnée en provenance de la mémoire externe est lue par le réseau, elle passe de cellule en cellule et peut donc être utilisée de nombreuses fois. Ce mode de calcul est à l'opposé du fonctionnement basé sur un accès mémoire à chaque utilisation d'une donnée, caractéristique des architectures séquentielles traditionnelles.

En conséquence, un réseau systolique peut être étendu (en adjoignant des cellules) pour traiter un problème coûteux en nombre d'opérations sans qu'il faille imposer pour autant une augmentation correspondante du débit de la mémoire externe. Cette propriété confère aux réseaux systoliques un avantage majeur sur les architectures traditionnelles.

IV.2. Simplicité et régularité

Réaliser des circuits spécialisés à un coût raisonnable est une préoccupation majeure pour les concepteurs VLSI : le coût de conception d'un tel circuit doit être assez bas pour pouvoir être amorti sur un faible volume de production. Comme on l'a vu dans les exemples précédents, les architectures systoliques sont composées à partir d'un petit nombre de cellules de base, premier avantage sur une architecture composée d'une grande variété de cellules complexes. Deuxième avantage, l'interconnexion locale et régulière des cellules facilite grandement l'implantation topologique de celles ci sur le circuit. Reprenons l'exemple de la figure 1 : les

données $(x_i)_{i \geq 1}$ sont propagées le long d'un bus de données, et dupliquées pour alimenter les quatre multiplieurs. Le dessin d'un circuit de convolution avec 5 coefficients ou plus, à partir du précédent, n'est pas aisé, alors qu'avec la solution systolique de la figure 2, c'est immédiat. D'une manière générale, la modularité et le faible "fan-out" (nombre maximal de copies d'une même variable créées à l'intérieur du réseau) des réseaux systoliques leur donne une grande reconfigurabilité : elles sont adaptables à la taille et à la nature du problème traité.

Enfin, il importe de dire un mot de la testabilité et de la forte résistance aux pannes des architectures systoliques. Les mêmes arguments que précédemment jouent en leur faveur dans ces domaines :

- pour le test : si les cellules sont identiques, il suffit d'en tester une seule;
- pour la résistance aux pannes : considérons un circuit comportant plusieurs rangées de cellules. On peut prévoir des mécanismes d'interrupteurs permettant de contourner les cellules qui se révéleraient défaillantes après réalisation. Il s'agit alors de reprogrammer le réseau en n'utilisant que les cellules valides, ce qui est facilité par la localité des connexions entre cellules.

IV.3. Calculs intensifs

Les systèmes VLSI sont adaptés à l'implantation d'algorithmes "compute-bound" (c'est-à-dire où le nombre de calculs élémentaires est plus grand que le nombre de données en entrée-sortie) plutôt qu'à des problèmes "I/O-bound" (où la situation est inversée) car le nombre de ports d'entrée-sortie est limité. Par exemple, la multiplication de deux matrices de taille n, qui nécessite $O(n^3)$ multiplications et additions pour $O(n^2)$ données, est un problème "compute-bound", tandis que l'addition de deux matrices de taille n, qui nécessite n^2 additions pour $3n^2$ opérations d'entrée-sortie est un problème "I/O-bound".

Les caractéristiques des architectures systoliques conduisent dans la plupart des problèmes "compute-bound" à des calculs en temps réel, c'est-à-dire où les sorties sont délivrées au

même rythme que les entrées. De fait, de telles architectures se sont révélées très performantes pour la résolution de nombreux problèmes où le volume de calcul est très important, et le traitement régulier.

IV.4. Peu de problèmes ont résisté à la systolisation

Même si tous les algorithmes proposés dans la littérature ne sont pas implémentés (ou implémentables) en VLSI, un tour d'horizon des réseaux existants a le mérite de montrer l'étendue du champ d'application de l'algorithmique systolique :

- traitement du signal et de l'image : filtrage; convolution 1D et 2D; corrélation; lissage par médiane 1D et 2D; transformée de Fourier discrète; projections géométriques; encodage/décodage pour les corrections d'erreurs; …
- arithmétique matricielle : multiplication matrice-vecteur, matrice-matrice; triangularisation (résolution de systèmes linéaires, calcul de l'inverse d'une matrice); décomposition QR (calcul aux moindres carrés, inversion de matrice de covariance); problèmes aux valeurs propres; …
- applications non numériques :

- structures de données : piles; files d'attente; recherche dans un dictionnaire; tri; …
- graphes et algorithmes géométriques : fermeture transitive; arbre de degré minimum; composantes connexes; enveloppes convexes; …
- manipulation de chaînes de caractères : occurrences d'un mot; plus longue sous-suite; reconnaissance de langages réguliers; …
- programmation dynamique;
- opérations sur des bases de données relationnelles;
- algèbre des polynômes : multiplication; division euclidienne; PGCD; …
- arithmétique entière et dans des corps finis : multiplication; division; PGCD; …
- simulation de type Monte-Carlo.

DOCUMENT COMPLEMENTAIRE : FIGURES

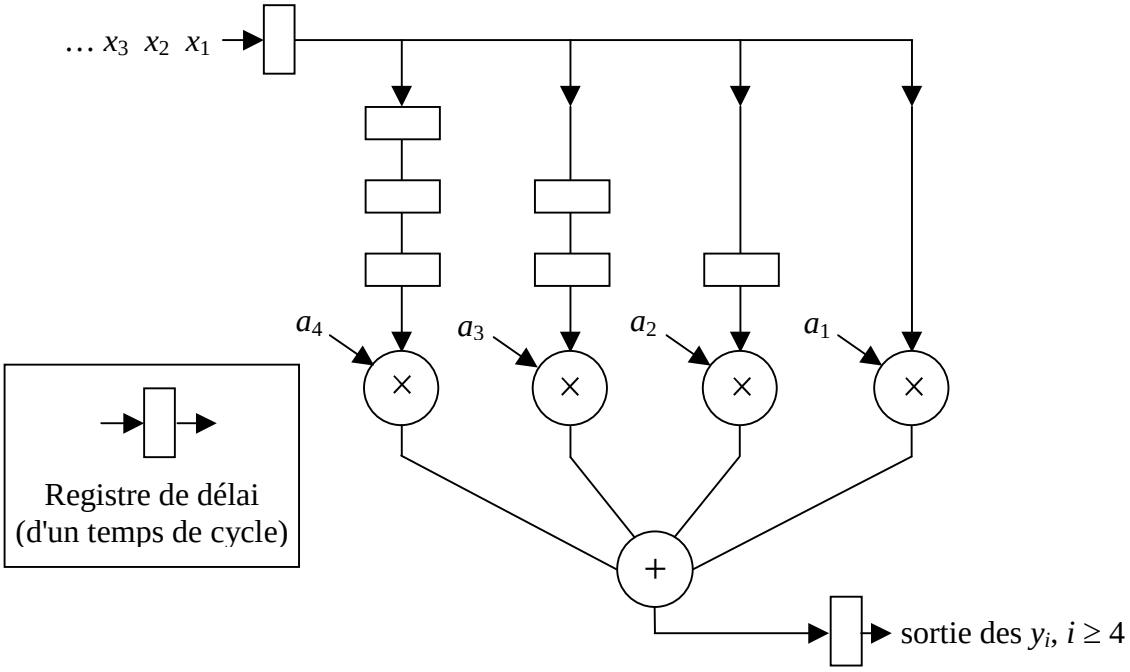


Figure 1 : Première solution

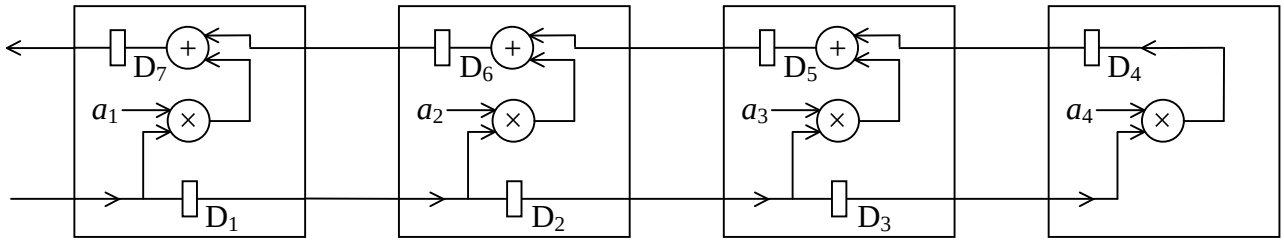


Figure 2 : Convolution non récursive

	D_1	D_2	D_3	D_4	D_5	D_6	D_7
1	x_1	0	0	0	0	0	$a_1.x_1$
2	0	x_1	0	0	0	$a_2.x_1$	0
3	x_2	0	x_1	0	$a_3.x_1$	0	$a_1.x_2 + a_2.x_1$
4	0	x_2	0	$a_4.x_1$	0	$a_2.x_2 + a_3.x_1$	0
5	x_3	0	x_2	0	$a_3.x_2 + a_4.x_1$	0	$a_1.x_3 + a_2.x_2 + a_3.x_1$
6	0	x_3	0	$a_4.x_2$	0	$a_2.x_3 + a_3.x_2 + a_4.x_1$	0
7	x_4	0	x_3	0	$a_3.x_3 + a_4.x_2$	0	$a_1.x_4 + a_2.x_3 + a_3.x_2 + a_4.x_1$
8	0	x_4	0	$a_4.x_3$	0	$a_2.x_4 + a_3.x_3 + a_4.x_2$	0
9	x_5	0	x_4	0	$a_3.x_4 + a_4.x_3$	0	$a_1.x_5 + a_2.x_4 + a_3.x_3 + a_4.x_2$
10	0	x_5	0	$a_4.x_4$	0	$a_2.x_5 + a_3.x_4 + a_4.x_3$	0
11	x_6	0	x_5	0	$a_3.x_5 + a_4.x_4$	0	$a_1.x_6 + a_2.x_5 + a_3.x_4 + a_4.x_3$
12	0	x_6	0	$a_4.x_5$	0	$a_2.x_6 + a_3.x_5 + a_4.x_4$	0

Figure 3 : Quelques pulsations

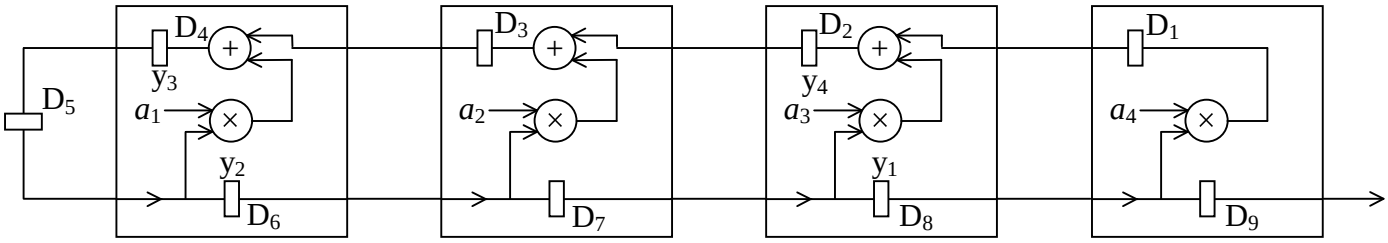
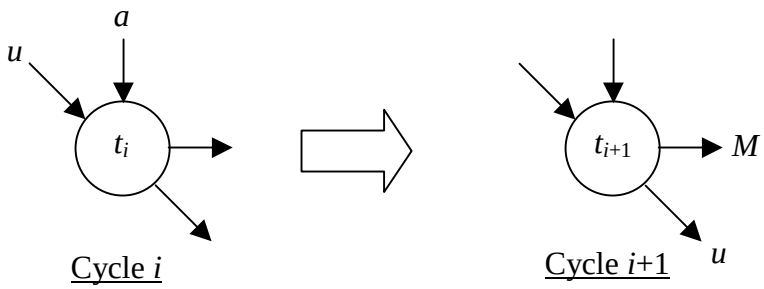


Figure 4 : Convolution récursive (état initial)

Cellule de génération :



$a \in \mathbb{R}$, $M \in \mathbf{M}_{22}(\mathbb{R})$, $u \in \{ I, T, N\}$ où *I*, *T*, *N* sont respectivement les abréviations de "Initialisation", "Triangularisation" et "Néant".

Algorithme :

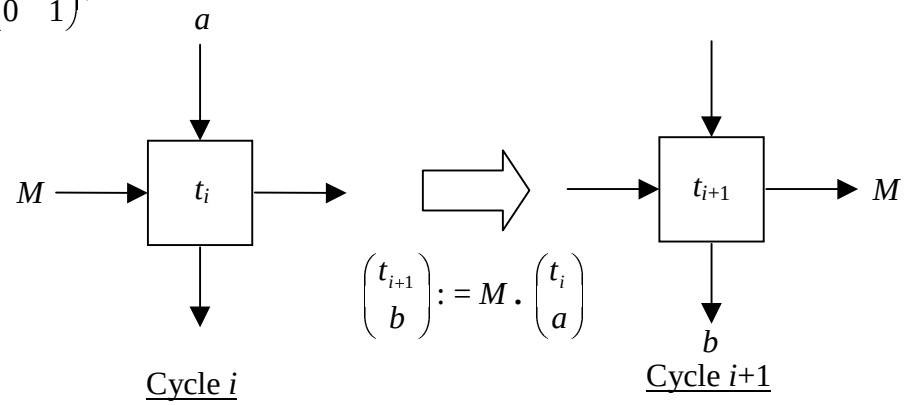
Si $u = I$ alors $M = M_I = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$;

si $u = T$ alors $M = \begin{pmatrix} 1 & 0 \\ -a/t_i & 1 \end{pmatrix}$;

si $u = N$ alors $M = M_{Id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$;

$t_{i+1} := m_{11}.t_i + m_{12}.a$;

Cellule de combinaison :



Cellule de retard :

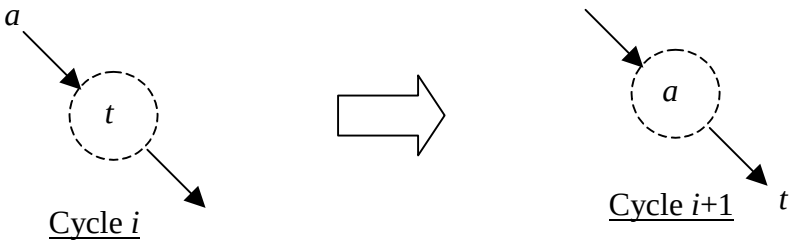
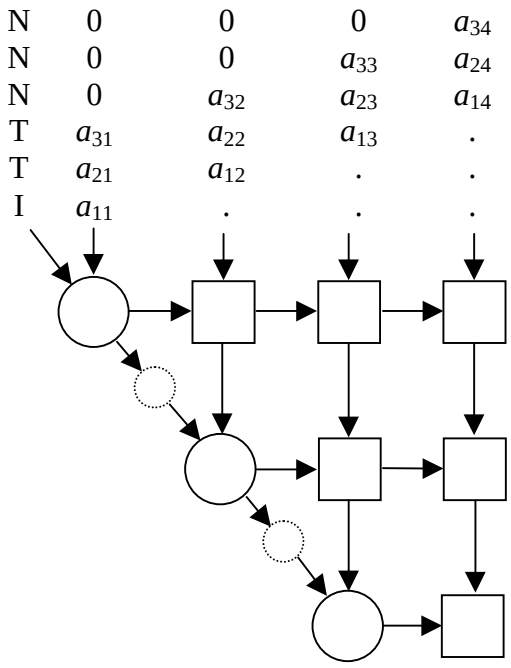
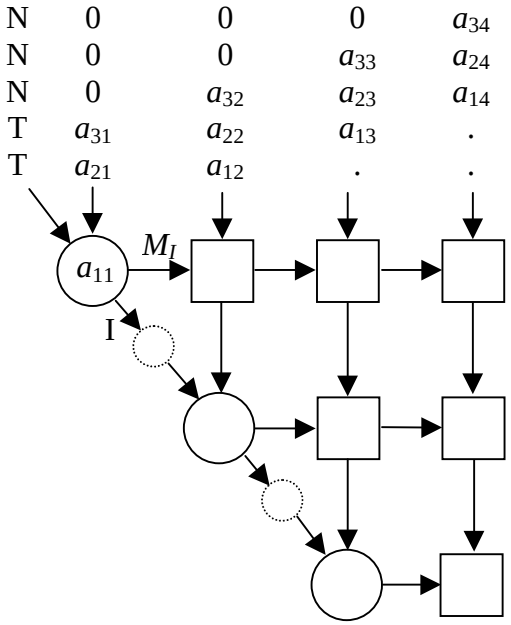


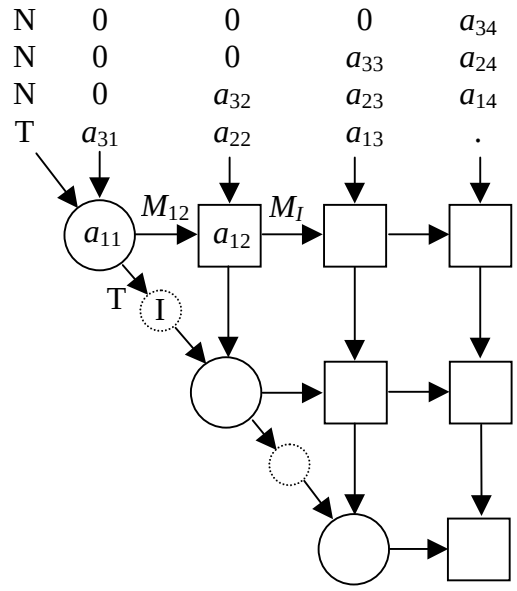
Figure 5 : Structure des cellules



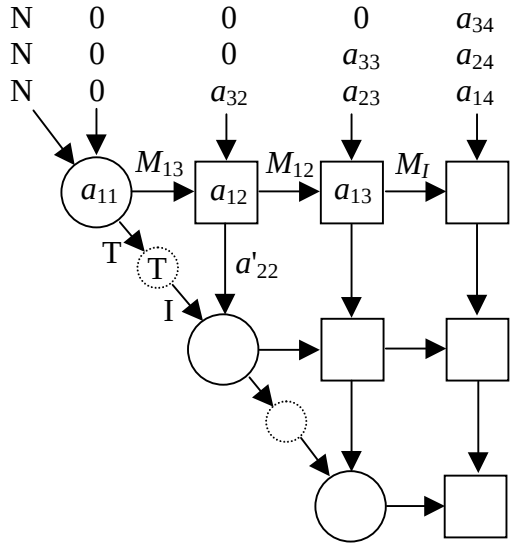
Cycle 1 :
I : Initialisation ;
T : Triangularisation ;
N : Néant.



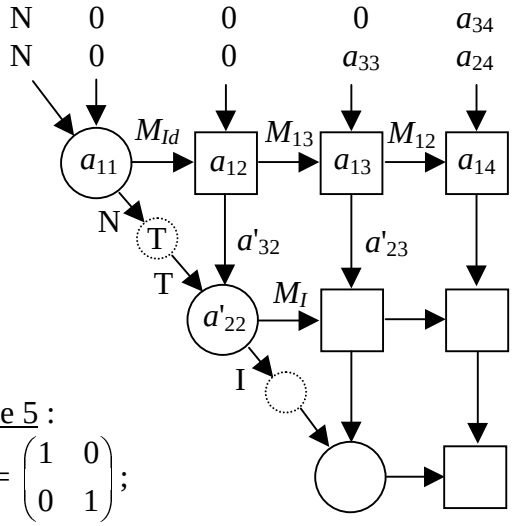
Cycle 2 :
$M_I = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$, matrice d'initialisation



Cycle 3 :
$M_{12} = \begin{pmatrix} 1 & 0 \\ l_{21} & 1 \end{pmatrix}$, $l_{21} = -\frac{a_{21}}{a_{11}}$,
matrice d'annulation du coefficient a_{21}



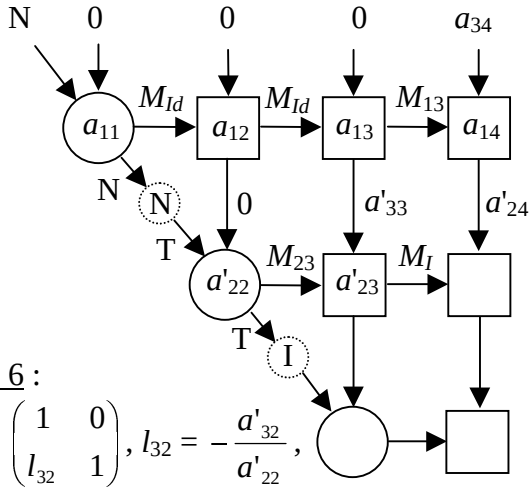
Cycle 4 :
$M_{13} = \begin{pmatrix} 1 & 0 \\ l_{31} & 1 \end{pmatrix}$, $l_{31} = -\frac{a_{31}}{a_{11}}$,
matrice d'annulation du coefficient a_{31} ;
$a'_{22} = -\frac{a_{21}}{a_{11}} a_{12} + a_{22}$;



Cycle 5 :

$M_{Id} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$;

$a'_{32} = -\frac{a_{31}}{a_{11}} a_{12} + a_{32}$; $a'_{23} = -\frac{a_{21}}{a_{11}} a_{13} + a_{23}$;

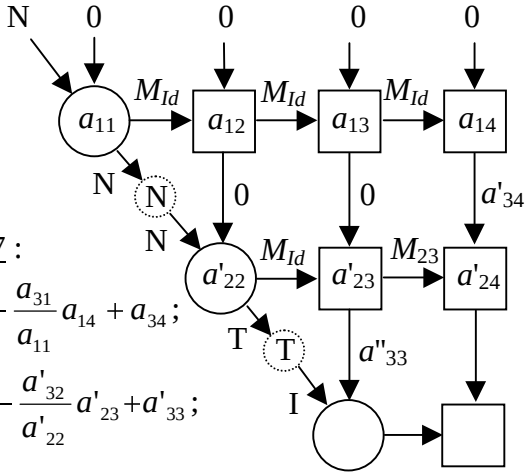


Cycle 6 :

$M_{23} = \begin{pmatrix} 1 & 0 \\ l_{32} & 1 \end{pmatrix}$, $l_{32} = -\frac{a'_{32}}{a'_{22}}$,

matrice d'annulation du coefficient a_{32} ;

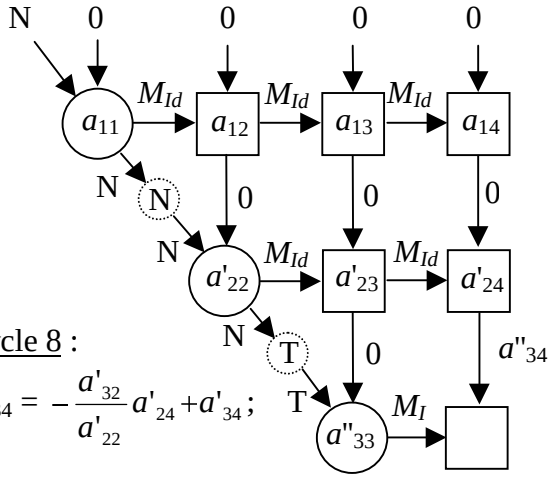
$a'_{33} = -\frac{a_{31}}{a_{11}} a_{13} + a_{33}$; $a'_{24} = -\frac{a_{21}}{a_{11}} a_{14} + a_{24}$;



Cycle 7 :

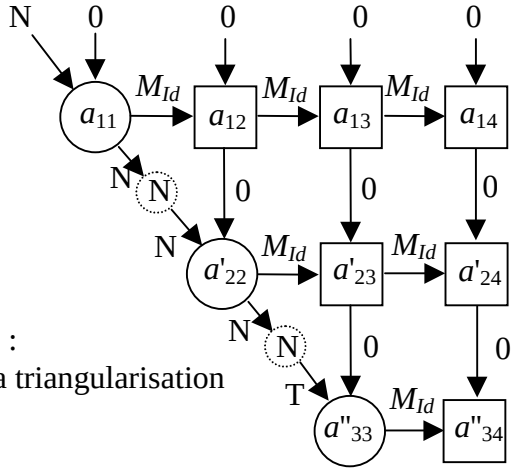
$a'_{34} = -\frac{a_{31}}{a_{11}} a_{14} + a_{34}$;

$a''_{33} = -\frac{a'_{32}}{a'_{22}} a'_{23} + a'_{33}$;



Cycle 8 :

$a''_{34} = -\frac{a'_{32}}{a'_{22}} a'_{24} + a'_{34}$;



Cycle 9 :

fin de la triangularisation

Figure 6 : Triangularisation d'une matrice 3 × 3