RAIRO. INFORMATIQUE THÉORIQUE

YVES ROBERT
MAURICE TCHUENTE

Réseaux systoliques pour des problèmes de mots

RAIRO. Informatique théorique, tome 19, n° 2 (1985), p. 107-123
<http://www.numdam.org/item?id=ITA_1985__19_2_107_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 19, n° 2, 1985, p. 107 à 123)

RÉSEAUX SYSTOLIQUES POUR DES PROBLÈMES DE MOTS (*)

par Yves ROBERT (1) et Maurice TCHUENTE (1)
Communiqué par J. BERSTEL

Résumé. — *On présente des réseaux systoliques pour la résolution en temps réel de problèmes de convolution généralisée. On a en vue la définition d'architectures systoliques d'efficacité maximale à partir d'une approche de type « divide-and-conquer » (approche bien connue dans le cas séquentiel). On donne trois applications à des problèmes de mots :*

- *le calcul du nombre d'occurrences d'un mot dans une chaîne;*
- *la reconnaissance des palindromes;*
- *la reconnaissance des carrés.*

Cette technique « divide-and-conquer » apparaît comme le complément naturel de la méthodologie de systolisation des circuits développée par Leiserson et Saxe [11].

Abstract. — *We introduce systolic arrays for the real time solution of some general convolution problems. This paper is devoted to illustrate the power of the divide-and-conquer approach (which is well-known for sequential algorithms) for the design of efficient systolic architectures. We give three applications to words problems:*

- *the computation of the occurrences of a given word in a string;*
- *the real-time palindrome recognizer;*
- *the real-time square acceptor.*

This divide-and-conquer technique can be viewed as the last step of the methodology introduced by Leiserson and Saxe [11] *for the design of systolic systems.*

1. INTRODUCTION

Le modèle systolique a été introduit en 1978 par Kung et Leiserson [10] pour tirer parti des nouvelles technologies d'intégration à haute densité; en 5 ans, ce modèle s'est révélé être un outil puissant pour la conception de processeurs intégrés spécialisés.

Un circuit basé sur une architecture systolique comprend, pour l'essentiel, un réseau régulier de cellules élémentaires, localement interconnectées et globalement synchronisées, et réalise de hautes performances « en temps réel » par une utilisation simultanée et pipe-line des cellules.

(*) Reçu décembre 1983, révisé juin 1984.
(1) C.N.R.S., Laboratoire T.I.M. 3/I.M.A.G., B.P. n° 68, 38402, Saint-Martin d'Hères Cedex.

0399-0540/85/02 107 17/$ 3,40/

Kung décrit en détail dans [8] tout l'intérêt du modèle pour la réalisation de circuits spécialisés :

— Les architectures systoliques sont faciles à implémenter à cause de leur dessin simple et régulier. Tout algorithme systolique conduit directement à un schéma de réalisation sur silicium;

— ces architectures sont aisément reconfigurables en raison de leur modularité;

— elles permettent de nombreux calculs pour chaque accès-mémoire, ce qui accélère le traitement de problèmes coûteux en nombre d'opérations sans augmenter les exigences en matière d'entrées/sorties.

De fait, de telles architectures se sont avérées très performantes pour la résolution de nombreux problèmes issus du traitement du signal [7], de l'algèbre numérique linéaire [8, 10], ou encore des bases de données [9]. Plus récemment, des réseaux systoliques ont été proposés pour la reconnaissance des langages [3, 4, 5 et 14].

Une définition précise du terme systolique a été introduite en 1981 par Leiserson et Saxe [11]. De plus, ces deux auteurs présentent une démarche constructive pour définir une architecture systolique à partir d'une architecture comprenant de la logique combinatoire se propageant en cascade. Le seul inconvénient de leur méthodologie est qu'elle débouche le plus souvent sur des réseaux où chaque cellule n'est activée qu'un top d'horloge sur k, $k \geqq 2$, ce qui limite par un facteur k le degré de parallélisme obtenu et augmente le temps d'exécution.

Plusieurs solutions ont été proposées pour remédier à cette situation : Kung et Leiserson [10] proposent de fondre k cellules adjacentes en une seule, ce qui diminue la surface par un facteur k mais n'améliore pas le temps d'exécution. Une autre solution consisterait à entrelacer la résolution de k problèmes indépendants (de même nature) sur le réseau, mais ceci n'accélère pas la résolution de chaque problème.

Nous présentons ici une approche basée sur la technique classique « divide-and-conquer » qui peut s'énoncer ainsi [6] : étant donné un problème à résoudre, diviser ce problème en k sous-problèmes que l'on résoud séparément, puis combiner la solution de ces k sous-problèmes pour reconstruire une solution du problème initial. Cette technique est bien connue pour les algorithmes usuels, et nous allons illustrer son application aux architectures systoliques (pour augmenter la vitesse d'exécution) sur des exemples issus de problèmes de mots.

Nous commençons par rappeler brièvement la construction de Leiserson et Saxe [11].

2. UNE MÉTHODE DE SYSTOLISATION

Un circuit est défini formellement comme un graphe orienté $G=(V, U)$, dont les sommets représentent les éléments fonctionnels du circuit. Un sommet particulier représente la structure hôte, qui permet au circuit de communiquer avec le monde extérieur.

Chaque sommet v de G est affecté d'un poids $d(v)$ positif qui est le temps de cycle de la cellule qu'il représente; chaque arc $e=(v, v')$ de U est affecté d'un poids entier $w(e)$ qui représente le nombre de registres que doit traverser une donnée pour aller de v à v'. Les circuits systoliques sont alors ceux pour lesquels chaque arc porte au moins un registre; leur synchronisation peut ainsi s'effectuer globalement au rythme d'une horloge dont le temps de cycle est $\operatorname{Max}_{v \in V} d(v)$.

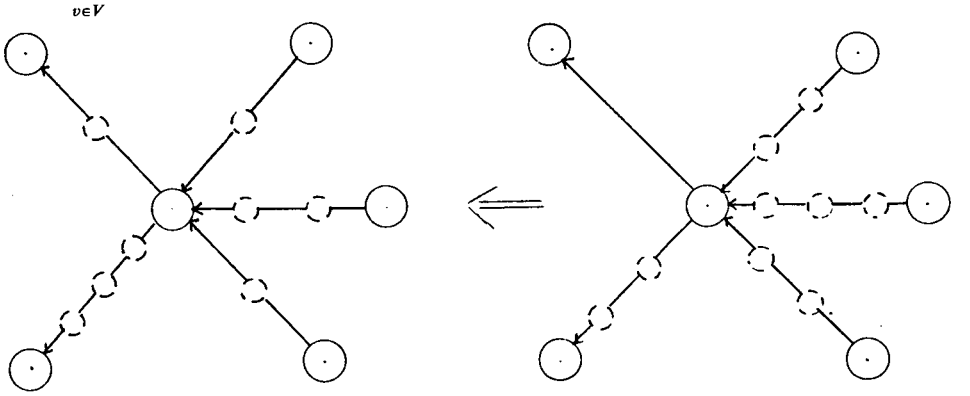


Figure 1. – Transformation élémentaire du graphe de communication.

Il est clair que la transformation qui consiste à retirer un registre sur chaque arc entrant dans une cellule donnée, et à en ajouter un sur chaque arc sortant de cette cellule, ne modifie en rien le comportement de la cellule vis-à-vis des éléments voisins (cf. *fig.* 1). Par ailleurs, on vérifie aisément que de telles transformations laissent invariant le nombre de registres sur tout circuit élémentaire; en conséquence, une condition nécessaire pour que de telles transformations conduisent à un circuit systolique, est que, sur chaque circuit élémentaire du graphe initial, le nombre de registres soit supérieur ou égal au nombre d'arcs. Leiserson et Saxe ont montré que cette condition nécessaire est aussi suffisante. Ceci les amène à proposer la méthodologie suivante pour la conception d'architectures systoliques :

– définir un réseau simple R (en général non systolique) où à chaque top d'horloge le résultat doit s'accumuler le long d'un chemin ne comportant pas de registres (logique combinatoire se propageant en cascade);

— déterminer le plus petit entier k tel que le réseau R_k obtenu à partir de R en multipliant par k le poids de tous les arcs soit systolisable; R_k a alors le même comportement externe que R mais sa vitesse est divisée par k;

— systoliser R_k, à l'aide des transformations précédentes (on peut définir des algorithmes polynomiaux performants pour cela).

Cette méthodologie conduit donc en général à un réseau qui délivre un résultat seulement une pulsation sur k. Nous proposons ici une approche purement algorithmique basée sur la stratégie classique du « divide-and-conquer » qui permet, à partir de R_k, d'obtenir un réseau R^* de même vitesse que R. Cette approche peut ainsi être considérée comme la dernière étape de la méthodologie de systolisation que nous venons de décrire brièvement.

3. CONVOLUTION GÉNÉRALISÉE

Soit E un ensemble non vide, $(B, *)$ un monoide commutatif, et $\top$ une loi de composition externe sur E à valeurs dans B :

$$E \times E \to B,$$
$$(e, e') \mapsto e \top e'.$$

Dans la suite, $\top$ est appelé opérateur de base.

La convolution généralisée se définit comme suit :

Données : — une suite de E, $(a_i; i \geqq 1)$; un élément a_i est appelé symbole d'entrée;

— une suite d'entiers $(n_i; i \geqq 1)$;

— une suite de vecteurs $W = (w^{(i)} \in E^{n_i}; i \geqq 1)$; un vecteur $w^{(i)}$ est appelé vecteur de référence.

Problème : Calcul de la suite $(y_i; i \geqq 1)$ définie par :

$$y_i = (a_i \top w_1^{(i)}) * (a_{i-1} \top w_2^{(i)}) * \ldots * (a_{i-n_i+1} \top w_{n_i}^{(i)}).$$

La convolution classique :

$$y_i = (a_i \top w_1) * (a_{i-1} \top w_2) * \ldots * (a_{i-k+1} \top w_k);$$
$$i \geqq k,$$

correspond au cas particulier où la suite W est constante et égale à $(w_1, w_2, \ldots, w_k)$.

L'approche « divide-and-conquer » consiste à séparer les calculs en deux parties :

$$y_i^1 = (a_i \top w_1^{(i)}) * (a_{i-2} \top w_3^{(i)}) * \ldots,$$
$$y_i^2 = (a_{i-1} \top w_2^{(i)}) * (a_{i-3} \top w_4^{(i)}) * \ldots$$

On construit alors un réseau R comportant :

- un sous-réseau $R1$ pour le calcul de y_i^1;
- un sous-réseau $R2$ pour le calcul de y_i^2;
- une cellule qui reçoit y_i^1, y_i^2 en entrée et délivre en sortie :

$$y_i = y_i^1 * y_i^2.$$

La raison de ce découpage est que, dans les réseaux systoliques proposés pour résoudre divers problèmes de convolution (au sens large où nous l'avons définie), les variables a_i et y_j circulent dans des directions opposées. Or on vérifie aisément (cf. *fig.* 2) que, dans un réseau linéaire, une variable y_j ne peut pas rencontrer deux éléments a_i, a_{i+1} qui se suivent. En conséquence, toute solution basée sur un réseau linéaire et où les variables a_i, y_j circulent dans des directions opposées, est d'efficacité 1/2, c'est-à-dire que deux variables consécutives y_j, y_{j+1} sont séparées par deux tops d'horloge [10] (cf. *fig.* 5).

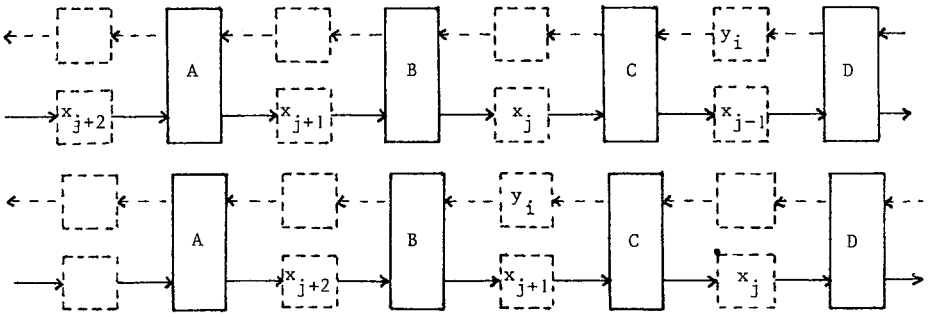


Figure 2. — Instant $t+1$: x_{j+2} et y_i sont des entrées de la cellule B.

L'approche « divide-and-conquer » permet de procéder ainsi :

— calcul de $y_i^1 = (a_i \top w_1^{(i)}) * (a_{i-2} \top w_3^{(i)}) * \ldots$;

sur un réseau $R1$ d'efficacité 1,

— calcul de $y_i^2 = (a_{i-1} \top w_2^{(i)}) * (a_{i-3} \top w_3^{(i)}) * \ldots$;

sur un réseau $R2$ d'efficacité 1,

— calcul de $y_i = y_i^1 * y_i^2$ avec une efficacité 1.

Dans la suite, nous allons étudier les cas suivants qui sont de difficulté croissante :

cas 1 : La suite de référence W est constante : $w^{(i)}=(w_1, w_2, \ldots, w_k)$ pour tout i;

cas 2 : Les vecteurs de référence s'allongent par la droite au cours du temps :

$$w^{(i)}=(w_1, w_2, \ldots, w_p); \qquad p=[i/2],$$

$w^{(i)}$ correspond aux p premiers termes d'une suite fixée $(w_j; j \geqq 1)$;

cas 3 : Les vecteurs de référence s'allongent par la gauche au cours du temps :

$$w^{(i)}=(w_p, w_{p-1}, \ldots, w_1); \qquad i=2p,$$

$w^{(i)}$ correspond aux p premiers termes, pris dans l'ordre inverse, d'une suite fixée $(w_j; j \geqq 1)$.

Chacun de ces cas sera illustré par un problème de mots.

4. NOMBRE D'OCCURRENCES D'UN MOT DANS UNE CHAÎNE

Le premier cas, celui de la convolution avec une référence constante, peut s'illustrer par le problème de détection des occurrences d'un mot dans une chaîne de caractères.

Soit donc $A=a_1 \ldots a_N$ et $W=w_n \ldots w_2 w_1$, $n<N$, deux mots sur un alphabet S. Une occurrence de W dans A est une sous-chaîne $a_i a_{i+1} \ldots a_{i+n-1}$ qui est égale à W. Le problème posé ici est celui de la détection des occurrences de W dans A. Par exemple, si $A=abababa$ et $W=aba$, alors $a_1 a_2 a_3$, $a_3 a_4 a_5$, $a_5 a_6 a_7$ sont des occurrences de W dans A.

Ce problème est équivalent au calcul des variables booléennes :

$$(1) \qquad y_i = \begin{cases} [a_i=w_1] \wedge [a_{i-1}=w_2] \wedge \ldots \wedge [a_{i-n+1}=w_n] \\ \qquad\qquad\qquad\qquad\qquad\qquad \text{pour} \quad i \geqq n, \\ 0, \qquad \text{pour} \quad i<n. \end{cases}$$

Une solution évidente (non systolique) est donnée par le schéma de la figure 3 [8]. Étant donnée la présence dans ce réseau de circuits comportant deux arcs et un seul registre, on commence par doubler le nombre de registres sur chaque arc, ce qui conduit au réseau deux fois plus lent de la figure 4. L'application de la méthode de [11] conduit alors au réseau de la figure 5.

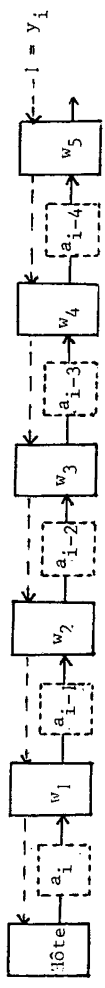


Figure 3. — Solution non systolique.

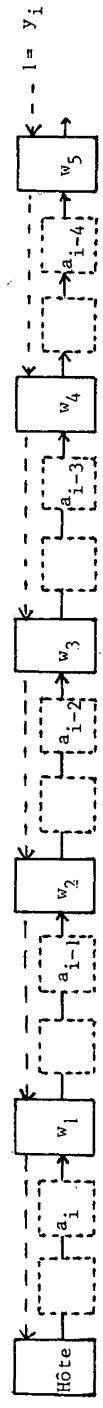


Figure 4. — Solution de la figure 1 avec vitesse divisée par 2.

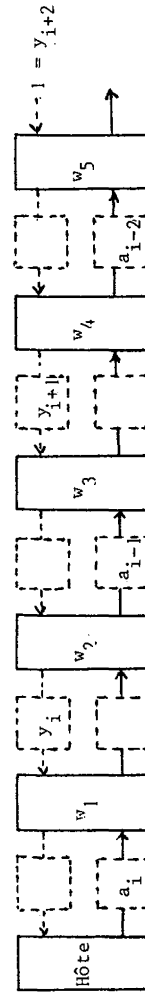


Figure 5. — Version systolisée du réseau de la figure 3.

Il s'agit bien d'un problème de convolution pour lequel :

— l'opérateur de base est l'égalité;

— $\wedge$ désigne la conjonction;

— la référence est constante et égale à $w_1 w_2 \ldots w_n$.

La technique « divide-and-conquer » peut s'appliquer ainsi :

— calcul dans un réseau $R1$ de :

$$y_i^1 = [a_i = w_1] \wedge [a_{i-2} = w_3] \wedge \ldots;$$

— calcul dans un réseau $R2$ de :

$$y_i^2 = [a_{i-1} = w_2] \wedge [a_{i-3} = w_4] \wedge \ldots;$$

— calcul de $y_i = y_i^1 \wedge y_i^2$ à la sortie de $R1$ et $R2$.

On obtient ainsi la solution présentée à la figure 6.

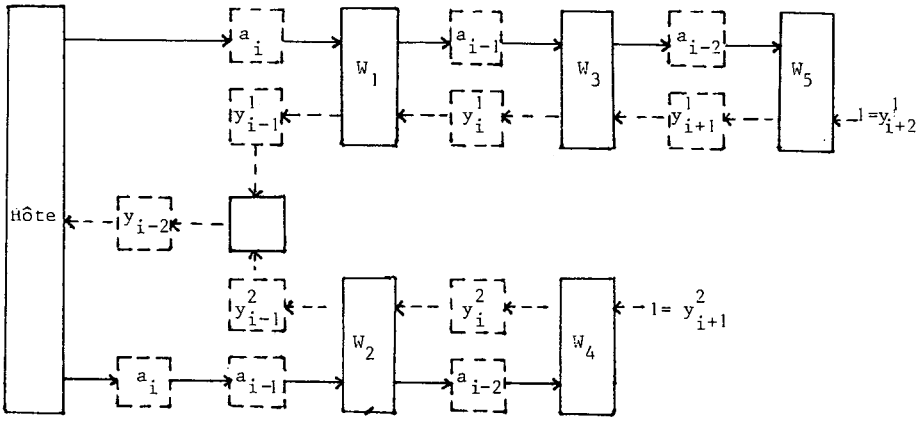


Figure 6. — Détection en temps réel des occurrences d'un mot.

Calcul du nombre d'occurences disjointes

Appelons occurences disjointes de W dans A deux occurences $a_i a_{i+1} \ldots a_{1+n-1}$, $a_j a_{j+1} \ldots a_{j+n-1}$ qui n'ont aucune position commune, c'est-à-dire telles que les intervalles $[i, i+n-1]$ et $[j, j+n-1]$ sont disjoints. Dans l'exemple examiné au début de ce paragraphe, avec $A = ababaaba$ et $B = aba$, le nombre d'occurences disjointes de B dans A est 2.

Le problème revient à calculer la suite $(y_i;\ i \geqq 1)$ définie par :

$$(2) \qquad y_i = \begin{cases} [a_i = w_1] \wedge \ldots \wedge [a_{i-n+1} = w_n] \\ \qquad \wedge \bar{y}_{i-1} \wedge \ldots \wedge \bar{y}_{i-n+1} \qquad \text{pour} \quad i \geqq n, \\ 0, \qquad \text{pour} \quad i < n, \end{cases}$$

où $\bar{y}$ désigne le complémentaire de y.

Pour résoudre ce problème à partir du réseau de la figure 6, il suffit que chaque variable de rang i, une fois calculée, rencontre celles de rang $i+k$, $1 \leqq k \leqq n-1$, afin de les faire basculer à 0 si elle-même vaut 1. Pour ce faire, on procède comme suit (cf. *fig.* 7) : chaque y_i, une fois calculée, est placée dans un registre d'entrée de la cellule qui calcule $y_i = y_i^1 \wedge y_i^2$, afin de rencontrer y_{i+1}, et est aussi renvoyée dans le réseau à la rencontre des y_{i+k}, $1 \leqq k \leqq n-1$; il est à noter que cette variable renvoyée dans le réseau ne doit pas atteindre toutes les cellules, car elle ferait alors basculer indûment à 0 des variables y_{i+m}, $m \geqq n$.

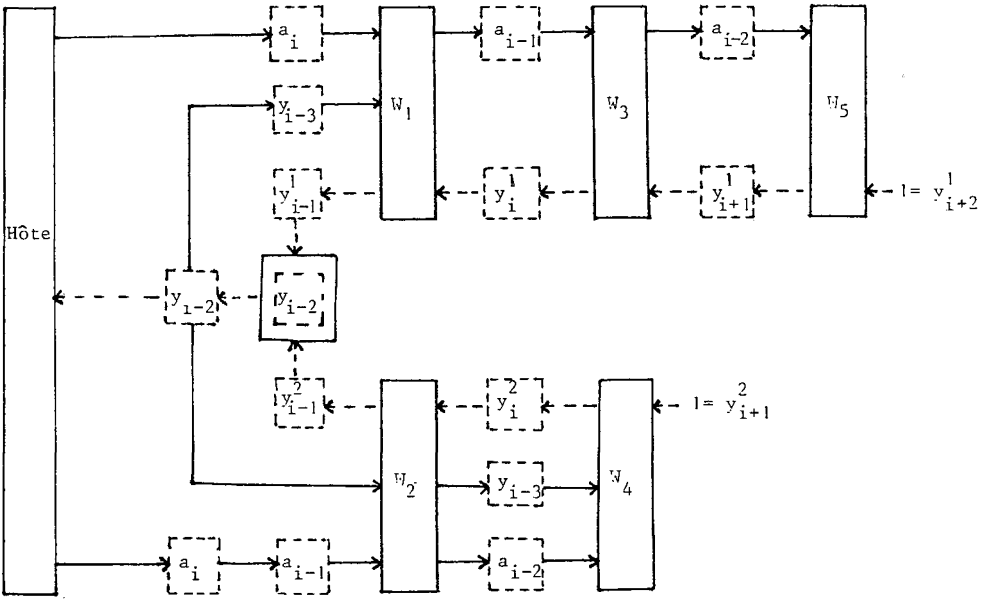


Figure 7. — Calcul en temps réel des occurrences disjointes.

Remarques 1. Le fait de placer y_i dans un registre d'entrée de la cellule qui calcule $y_i = y_i^1 \wedge y_i^2$ n'augmente pas le temps de cycle du réseau, car cette cellule est alors amenée à calculer :

$$y_i = y_i^1 \wedge y_i^2 \wedge \bar{r},$$

où r est la valeur du registre interne, et reste donc de complexité nettement inférieure aux autres cellules du réseau.

2. Le problème que nous venons de traiter est très semblable dans sa formulation à la convolution récursive :

$$y_i = w_1 \cdot x_i + \ldots + w_n \cdot x_{i-n+1} + r_1 \cdot y_{i-1} + \ldots + r_m \cdot y_{i-m},$$

w_j, r_j, x_i réels; $y_1, \ldots, y_m$ réels donnés.

Cependant la remarque 1 ne s'applique pas à la convolution récursive car la cellule qui calcule y_i aurait alors à effectuer une addition supplémentaire par rapport aux autres cellules; inversement, la technique exposée dans [12] et qui consiste à réécrire la convolution récursive sous la forme :

$$y_i = w_1^* \cdot x_i + \ldots + w_n^* \cdot x_{i-n+1} + r_2^* \cdot y_{i-2} + \ldots + r_{m+1}^* \cdot y_{i-m-1},$$

ne peut être utilisée ici car elle conduirait à des formules inexploitables.

5. RECONNAISSANCE DES PALINDROMES

Le deuxième cas que nous allons traiter est celui où la référence s'allonge par la droite au cours du temps. Ceci correspond à la situation où $w^{(i-1)}$ est préfixe de $w^{(i)}$, et peut s'illustrer à l'aide de l'exemple de la reconnaissance des palindromes.

Une chaîne $a_1 a_2 \ldots a_n$ est appelée un palindrome si elle est symétrique, c'est-à-dire si $a_i = a_{n-i+1}$ pour tout i. Les palindromes constituent un langage hors contexte très simple, et leur reconnaissance est un problème classique. Barzdin [1] a montré que la reconnaissance d'un palindrome de taille n sur une machine de Turing ayant une seule tête de lecture-écriture nécessite un temps de l'ordre de n^2. Plus tard, Smith [13] a exhibé un automate cellulaire uniforme de dimension un, capable de reconnaître un palindrome de longueur n en temps linéaire $t_n \leqq n/2$; le déploiement du diagramme espace-temps de cet algorithme donne la solution proposée récemment par Culik *et al.* [4]. Cependant, le modèle de Smith n'est pas adapté au pipe-line et ne permet donc pas, pour une suite $(a_i;\ i \geqq 1)$ donnée, le calcul en temps réel de la suite booléenne $(y_i;\ i \geqq 1)$ définie par :

$$y_i = \begin{cases} 1 & \text{si } a_1 a_2 \ldots a_i \text{ est un palindrome,} \\ 0 & \text{sinon.} \end{cases}$$

Ce dernier problème a été résolu pour la première fois par Cole [2]; Leiserson et Saxe [11] ont proposé récemment une solution beaucoup plus simple mais dans laquelle deux symboles a_i consécutifs sont séparés par deux tops d'horloge, et ce pour les raisons que nous avons exposées au paragraphe précédent.

La formule de calcul d'une variable y_i est :

$$y_i = [a_i = a_1] \wedge [a_{i-1} = a_2] \wedge \ldots \wedge [a_{i-[i/2]+1} = a_{[i/2]}],$$

où $[u]$ désigne le plus petit entier supérieur ou égal à u.

Il s'agit donc d'un problème de convolution pour lequel :

— l'opérateur de base est l'égalité;

— $\wedge$ désigne la conjonction;

— $w^{(i)} = a_1 a_2 \ldots a_{[i/2]}$ pour tout $i \geqq 1$.

Comme les mots de référence $w^{(i)}$, $i \geqq 1$, dépendent de la suite des symboles d'entrée $(a_i;\ i \geqq 1)$, le traitement d'un symbole a_i comporte trois phases :

— dans une première phase, tant que a_i rencontre y_j, $j \leqq 2(i-1)$, il joue le rôle de symbole d'entrée;

— dans une seconde phase, lorsqu'il rencontre y_{2i-1}, il sert simultanément de symbole d'entrée et de symbole de référence;

— enfin, dans la troisième phase, lorsqu'il rencontre y_j, $j \geqq 2i$, il joue le rôle de symbole de référence.

La solution au problème de la reconnaissance des palindromes en temps réel s'obtient à partir du réseau de reconnaissance des occurences d'un mot (cf. *fig.* 5), en adjoignant un dispositif permettant de générer dynamiquement les mots de référence $w^{(i)}$, $i \geqq 1$. On procède comme suit :

— a_{2j+1} ne sert de symbole de référence que dans le sous-réseau $R1$; en conséquence, on adjoint une variable booléenne Test_{2j+1}, qui vaut 1 dans la copie de a_{2j+1} envoyée dans $R1$, et 0 dans la copie envoyée dans $R2$. Un traitement analogue est appliqué aux variables a_{2j}, $j \geqq 1$;

— le problème important de détermination du début de la phase 3 pour un couple $(a_{2j}, 1)$ ou $(a_{2j+1}, 1)$ se résoud en remarquant tout simplement que celà correspond à la rencontre d'une cellule dont le registre interne est vide.

Ceci conduit à la solution de la figure 8.

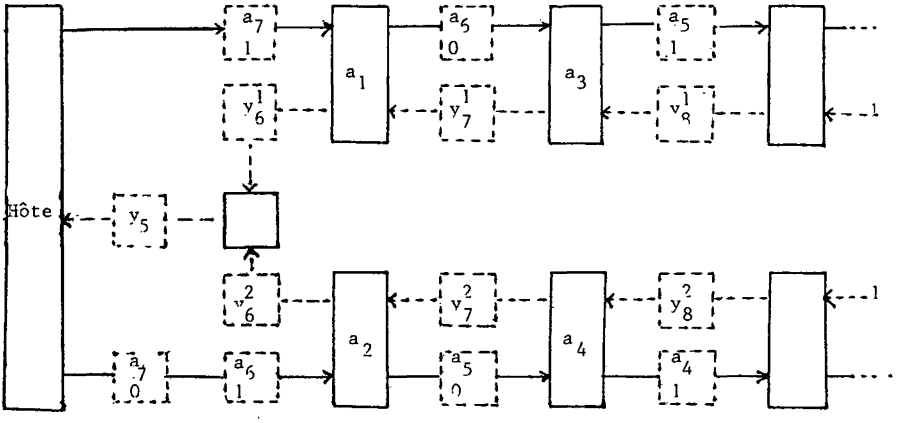


Figure 8. — Reconnaissance des palindromes en temps réel.

6. RECONNAISSANCE DES CARRÉS

Le troisième cas que nous étudions est celui où la référence s'allonge par la gauche au cours du temps. Ceci correspond à la situation où, pour tout $i \geqq 1$, $w^{(i-1)}$ est un suffixe de $w^{(i)}$, et peut s'illustrer par le problème de la reconnaissance des carrés.

Une chaîne $A = a_1 a_2 \ldots a_n$, $n = 2p$ entier pair, est appelée un carré si $a_i = a_{i+p}$ pour $1 \leqq i \leqq p$. Soit donc à évaluer, pour $(a_i; i \geqq 1)$ donnée, la suite booléenne définie par :

$$y_i = \begin{cases} 1 & \text{si } a_1 a_2 \ldots a_i \text{ est un carré,} \\ 0 & \text{sinon.} \end{cases}$$

La formule de calcul de y_i est :

$$y_i = \begin{cases} [a_i = a_p] \wedge [a_{i-1} = a_{p-1}] \wedge \ldots \\ \qquad \wedge [a_{p+1} = a_1] & \text{si } i = 2p, \\ 0 \quad \text{sinon.} \end{cases}$$

Il s'agit bien d'un problème de convolution pour lequel :

— l'opérateur de base est l'égalité;

— $\wedge$ désigne la conjonction;

— $w^{(i)} = a_p a_{p-1} \ldots a_1$ pour $i = 2p$; ($w^{(i)}$ est indéfini pour i impair).

Comme dans le cas du palindrome, chaque symbole a_i est amené à jouer deux rôles :

— lorsqu'il est comparé à a_j, $j < i$, pour le calcul d'une variable y_k, $k < 2i$, il sert de symbole d'entrée;

— lorsqu'il est comparé à a_j, $j > i$, pour le calcul d'une variable y_k, $k \geqq 2i$, il joue le rôle d'un caractère de référence.

Pour que le calcul s'effectue en temps réel, il est nécessaire qu'à tout instant $t = 2i$, le symbole d'entrée a_{2i}, le symbole de référence a_i et la variable y^1_{2i} soient en entrée de la première cellule de $R1$, de manière à ce que l'accumulation :

$$y^1_i := y^1_i \wedge [a_{2i} = a_i],$$

puisse s'effectuer.

Pour des raisons de clarté, nous allons d'abord supposer que la structure hôte fournit les données de la manière suivante :

— à tout instant $t = 2i + 1$, elle délivre le symbole d'entrée a_{2i+1};

— à tout instant $t = 2i$, elle délivre le symbole d'entrées a_{2i} et le symbole de référence a_i.

Il faut alors définir un mécanisme qui, à partir des caractères d'entrée fournis par la structure hôte, génère dynamiquement les mots de référence :

$$w^{(i)} = a_p a_{p-1} \ldots a_1, \qquad i = 2p.$$

Considérons la copie du caractère de référence a_i qui a l'instant $t = 2i$ est envoyée dans le sous-réseau $R1$:

$$y_{2i} = [a_{2i} = a_i] \wedge [a_{2i-1} = a_{i-1}] \wedge \ldots \wedge [a_{i+1} = a_1],$$

elle joue le rôle de $w_1^{(2i)}$ et se trouve donc en entrée de la première cellule;

— comme les caractères de référence du sous-réseau $R1$ ne sont amenés à jouer les rôles de $w_k^{(j)}$ que pour k impair, a_i intervient dans la deuxième cellule de $R1$ pour jouer le rôle de $w_3^{(j)}$, ce qui correspond à $j = 2i+4$:

$$y_j = [a_{2i+4} = a_{i+2}] \wedge [a_{2i+3} = a_{i+1}] \wedge [a_{2i+2} = a_i] \wedge \ldots \wedge [a_{i+3} = a_1].$$

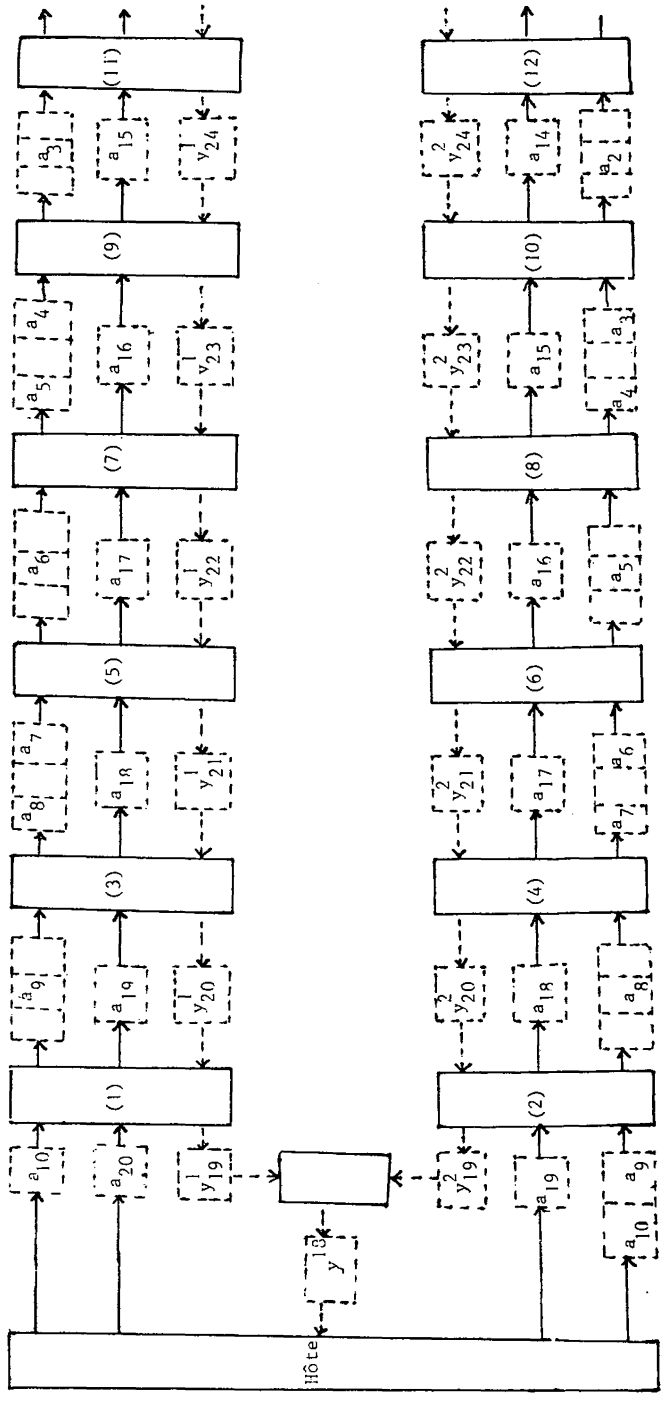
Comme l'accumulation :

$$y^1_{2i+4} := y^1_{2i+4} \wedge [a_{2i+2} = a_i],$$

s'effectue dans la deuxième cellule de $R1$ à l'instant $t = 2i+3$, on déduit que a_i doit traverser trois cellules de retard pour passer d'une cellule de $R1$ à la suivante.

Un raisonnement analogue s'applique à la copie du caractère de référence a_i qui est envoyée dans le sous-réseau $R2$. En outre, comme nous l'avons remarqué plus haut, la contrainte temps-réel impose que les caractères de référence soient espacés de deux tops d'horloge. On est ainsi conduit au réseau de la figure 9.

Intéressons-nous maintenant au réseau qui doit recevoir en entrée a_i à l'instant i et le délivrer à la première cellule à l'instant $2i$. Ce réseau est implicite dans la construction proposée par Cole [2] pour la reconnaissance des carrés en temps réel. Il suffit de considérer un réseau linéaire où chaque cellule comporte deux couloirs de circulation. Le principe de fonctionnement est le suivant : chaque symbole a_i circule dans le couloir du bas, en avançant à chaque top d'une cellule vers la droite, jusqu'à l'instant t_0 où dans la cellule voisine à sa droite les deux registres sont vides. A l'instant t_0+1 le symbole gagne le couloir du haut, et à partir de l'instant t_0+2 circule dans ce couloir vers la gauche, toujours en avançant d'une cellule à chaque top. Quelques pulsations consécutives sont représentées figure 10 pour illustrer la marche de ce dispositif.



Les tests $a_{i-k+1} = w_k^{(i)}$ s'effectuent dans la cellule (k).
Figure 9. — Reconnaissance des carrés en temps réel.

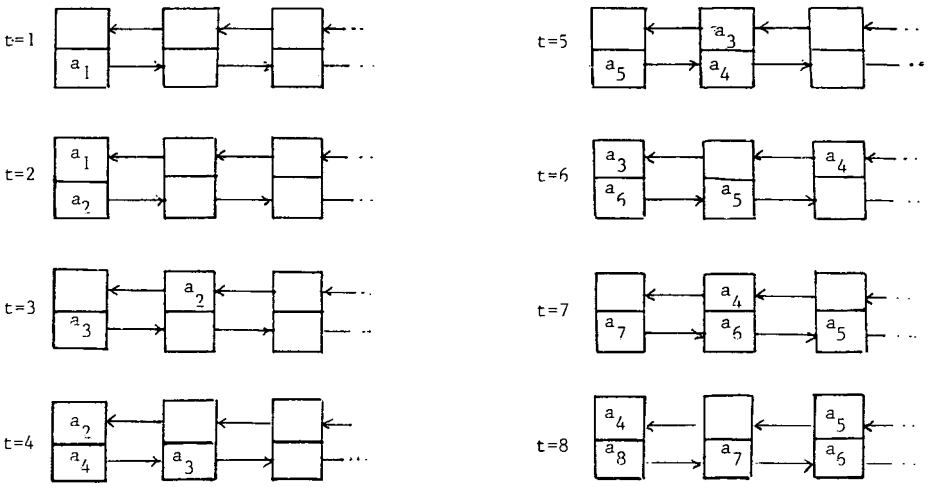


Figure 10

REMARQUE : Dans tous les exemples que nous avons traités, l'approche « divide-and-conquer » conduit à une solution où chaque variable d'entrée comporte deux copies dans le réseau. Lors d'une réalisation effective, cette duplication des entrées serait effectuée par une cellule spéciale à l'intérieur du circuit. Cependant, pour une raison de clarté, nous avons supposé que c'est la structure hôte qui effectue cette duplication des données.

7. CONCLUSION

La méthode de systolisation de Leiserson et Saxe [11], complétée par une transformation de type « divide-and-conquer » à son dernier stade, se révèle être un outil très efficace : ainsi pour le problème de convolution généralisée est-on conduit à une solution temps réel où le circuit reçoit une nouvelle donnée et délivre un nouveau résultat tous les tops d'horloge.

Des techniques de type « divide-and-conquer » avaient été introduites par les auteurs dans [12] pour améliorer les performances de réseaux systoliques pour la convolution récursive et le produit de matrices. De fait, cette approche peut être utilisée à partir de circuits déjà systoliques. Mais d'un point de vue méthodologique, elle s'avère constituer le complément naturel (la dernière étape) d'une démarche qui, partant d'une solution évidente mais non adaptée au calcul en temps réel, conduit à un circuit systolique de performance maximale.

D'autre part, l'introduction de la notion de convolution généralisée présente un intérêt en elle-même. En effet, comme nous l'avons montré, c'est un

contexte qui englobe des problèmes très variés sous un formalisme unique, et qui permet de résoudre ces problèmes en temps réel, en adjoignant au réseau de convolution classique un mécanisme de génération dynamique des vecteurs de référence. D'ailleurs, cette approche ne se limite pas aux problèmes de mots : ainsi la conception d'un réseau systolique d'efficacité maximale pour la multiplication d'une matrice A à structure bande par un vecteur ou encore pour la résolution d'un système triangulaire de matrice A :

$$A x = b,$$

rentrent directement dans ce formalisme et correspond au cas où les vecteurs de référence évoluent dynamiquement au cours du temps, mais tout en restant de longueur fixe.

Notons enfin que, pour la reconnaissance des palindromes et des carrés, on peut retrouver les réseaux proposés par Cole [2] à partir des réseaux présentés ici, en effectuant des regroupements entre des cellules voisines. Notre approche cumule ainsi deux avantages :

— elle évite les preuves compliquées de Cole [2], grâce à l'utilisation du formalisme de Leiserson et Saxe [11];

— elle conduit à une solution d'efficacité maximale, grâce à l'emploi d'une approche « divide-and-conquer », et remédie ainsi au manque de performance des solutions obtenues dans [11].

REMERCIEMENTS

Nous remercions le professeur Jean Berstel dont les suggestions nous ont permis de clarifier certains points délicats, ainsi que le rapporteur dont les remarques ont conduit à une amélioration de la présentation de cet article.

BIBLIOGRAPHIE

1. Y. M. Barzdin, *Complexity of Recognition of Symmetry in Turing Machines*, Problemy Kibernetiki, vol. 15, 1965.
2. S. N. Cole, *Real-time Computation by n-Dimensional Iterative Arrays of Finite-state Machines*, I.E.E.E. Trans. on Computers, vol. C 18, avril 1979, p. 349-365.
3. K. Culik II, J. Gruska et A. Salomaa, *On a Family of L. Languages Resulting from Systolic Tree Automata*, Research Report CS-81-36, University of Waterloo.
4. K. Culik II, J. Gruska et A. Salomaa, *Systolic Treillis Automata for (VLSI)*, Research Report CS-81-36, University of Waterloo.
5. M. J. Foster and H. T. Kung, *Recognize Regular Langages with Programmable Building-blocks, in the Proceedings of the VLSI*, 1981, Conference, Edinburgh.
6. E. Horowitz et A. Zora, *Divide-and-conquer for Parallel Processing*, I.E.E.E. Trans. on Computers, vol. C 32, (6), juin 1983.

7. A. V. KULKARNI et D. W. L. YEN, *Systolic Processing and an Implementation for Signal and Image Processing*, I.E.E.E. Trans. on Computers, vol. C31, octobre 1982, p. 1000-1009.
8. H. T. KUNG, *Why Systolic Architectures*, I.E.E.E. Computer, vol. 15, n° 1, janvier 1982, p. 37-46.
9. H. T. KUNG et P. L. LEHMAN, *Systolic V.L.S.I. Arrays for Relational Database Operations*, Proceedings of the 1980 A.C.M./SIGMOD International Conference on Management of data, Los Angeles, U.S.A.
10. H. T. KUNG et C. E. LEISERSON, *Systolic Arrays for (VLSI)*, in the Proceedings of the Symposium on Sparse Matrix Computations and their Applications, Knoxville, 1978.
11. C. E. LEISERSON et J. D. SAXE, *Optimizing Synchronous Systems*, 22-th Annual Symposium on Foundations of Computer Science, I.E.E.E., octobre 1981, p. 23-36.
12. Y. ROBERT et M. TCHUENTE, *Designing Efficient Systolic Algorithms*, Rapport de Recherche 393, *Lab. I.M.A.G., Grenoble*, septembre 1983.
13. A. R. SMITH III, *Cellular Automata theory*, Technical Report 2, *Stanford Electronics Laboratories*, décembre 1969.
14. M. STEINBY, *Systolic Tree and Systolic Langage Recognition by Tree Automata*, Theoretical Computer Science, vol. 22, 1983, p. 219-232.