STORING TREES INTO PARALLEL MEMORIES

(extended abstract)

H.A.G. Wijshoff

RUU-CS-85-29

October 1985

Rijksuniversiteit Utrecht

Vakgroep informatica

Budapestlaan 6 3584 CD Utrecht
Corr. adres: Postbus 80.012 3508 TA Utrecht
Telefoon 030-53 1454
The Netherlands

STORING TREES INTO PARALLEL MEMORIES
(extended abstract)

H.A.G. Wijshoff

Technical Report RUU-CS-85-29
October 1985

Department of Computer Science
University of Utrecht
P.O.Box 80.012, 3508 TA Utrecht
the Netherlands

STORING TREES INTO PARALLEL MEMORIES

(extended abstract)

H.A.G. Wijshoff

Department of Computer Science, University of Utrecht

P.O. Box 80.012, 3508 TA Utrecht, the Netherlands

Parallel memories require special storage schemes from a given data-structure into a set of memory banks. These storage schemes, also known as skewing schemes, received considerable attention in the literature recently. However, until now only storage schemes for d-dimensional arrays have been considered. In this paper we are concerned with storage schemes for tree data-structures. We give bounds on the minimal number of memory banks needed for certain applications, and furthermore we introduce the regular skewing schemes which have the property of being compactly representable and which can be computed fast.

INTRODUCTION

The availability of data can heavily affect the performance of parallel computers. This is a result of (i) the increased ratio between memory cycle time and execution cycle time and (ii) the need to supply data to several processing units simultaneously. An efficient solution to the data-distribution problem is providing a parallel computer with a number of (interleaved) memory banks. This approach can be found in e.g., the BSP, the CDC 205, and the CRAY 1. The structure of such a computer architecture is depicted in figure 1.

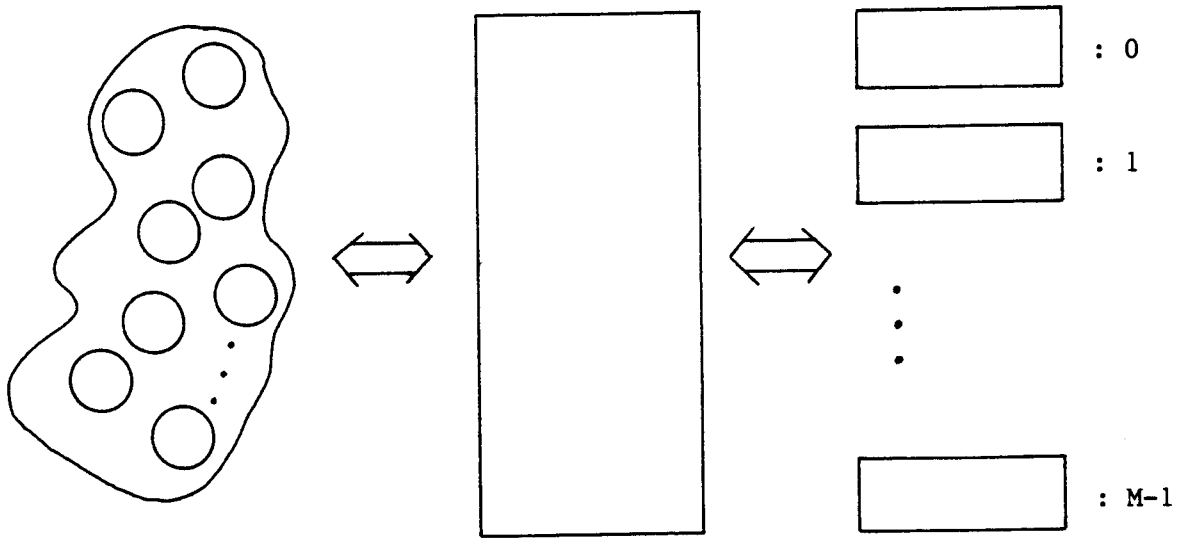


Figure 1.

Whenever the processors want access to a set of data elements, they send requests to the memory banks in which the data elements are stored. The time needed for handling these requests is proportional to the maximum number of data elements requested that are stored within the same memory bank. So the problem arises how to distribute the data over a number of memory banks such that certain parts of the data can be fetched conflict-free, i.e., such that they are stored in different banks. This problem is also known as the granularity problem of parallel memories. Mappings that give such a distribution are called skewing schemes. Thus, a skewing scheme merely is a mapping from some kind of data-structure into the set {0,1,...,M-1} corresponding to the names of the M memory banks. A part T of the data to which the processors want access in called a template.
An important issue is that skewing schemes are not of much practical interest, if they do not allow for a short representation in a formula, or cannot otherwise be computed fast. This is so because all processors have to know where the particular data elements they want access to are stored [3,5,6]. Recently new advances were made in distributing d-dimensional arrays over a number of memory banks [1,2,3,4,5,6]. Other kinds of data-structures, however, have not been considered yet.
In this paper we will deal with storage schemes for tree data-structures, which are fundamental in many non-numerical computations. First we discuss general skewing schemes for tree data-structures, and we give a characterisation of the minimal number of memory banks needed for a skewing scheme to be conflict-free for a collection of templates. In the last section we introduce a class of skewng schemes, the regular skewing schemes, which appear to be representable in a compact way and can be computed fast. Moreover, they are very feasible for some interesting sets of templates.

In this paper we just give a short impression of the results obtained. In [7] the results are given in full detail.

GENERAL SKEWING SCHEMES FOR TREE DATA-STRUCTURES

In order to define skewing schemes for trees we may pose the question how to represent these trees. First we make the assumption that trees are infinite, complete, and k-ary, and that they have a fixed orientation (i.e., the trees are "rooted"). So when we mention trees throughout this paper, we mean trees which have these restrictions. Note that an arbitrary finite k-ary tree can be seen as embedded in such a tree. In the following definitions we give a manner of labeling (numbering) the nodes (elements) of a tree.

Definition 2.1. Given a (k-ary) tree T with root α_0. Let $e_1,e_2,\ldots,e_k$ denote the edges from a node to its sons numbered from left to right. Then lab: T → {all strings over the alfabet $\{e_1,e_2,\ldots,e_k\}$} is defined by lab $(\alpha) = \lambda_1 \lambda_2 \ldots \lambda_n$, with for all $1 \leq i \leq n$: $\lambda_i \in \{e_1,e_2,\ldots,e_k\}$ and $\lambda_1\lambda_2..\lambda_n$ constitutes the labeled path from α_0 to α.
lab $(\alpha_0) = \varepsilon$ (the empty string).

We usually identify a node with the sequence lab(α), which uniquely defines it. In particular T can be seen as the infinite set of strings over the alfabet $\{e_1,e_2,\ldots,e_k\}$.

Definition 2.2.
(i) A template P on a tree T is any finite set of nodes $\{\alpha_1,\alpha_2,\ldots,\alpha_N\} \subseteq T$.
(ii) An instance $P(\gamma)$, $\gamma \in T$, of a template P is the set $\{\gamma\alpha_1,\gamma\alpha_2,\ldots,\gamma\alpha_N\}$.

Definition 2.3.
(i) A skewing scheme s on a tree T is a surjective map from T into {0,1,...,M-1}.
(ii) A skewing scheme s is valid for a collection of templates C = $\{P_1,P_2,\ldots,P_s\}$, if $\forall\, 1 \leq i \leq s\ \forall \gamma \in T\ \forall m \in \{0,1,\ldots,M-1\}$: $|P_i(\gamma) \cap s^{-1}(m)| \leq 1$.
(iii) $m(P_1,P_2,\ldots,P_s)$ is equal to the minimal number M such that there exists a skewing scheme s: T → {0,1,...,M-1} which is valid for $\{P_1,P_2,\ldots,P_s\}$.

An example of a binary tree, a template with its instances, and a skewing scheme which is valid for this template is given in figure 2.

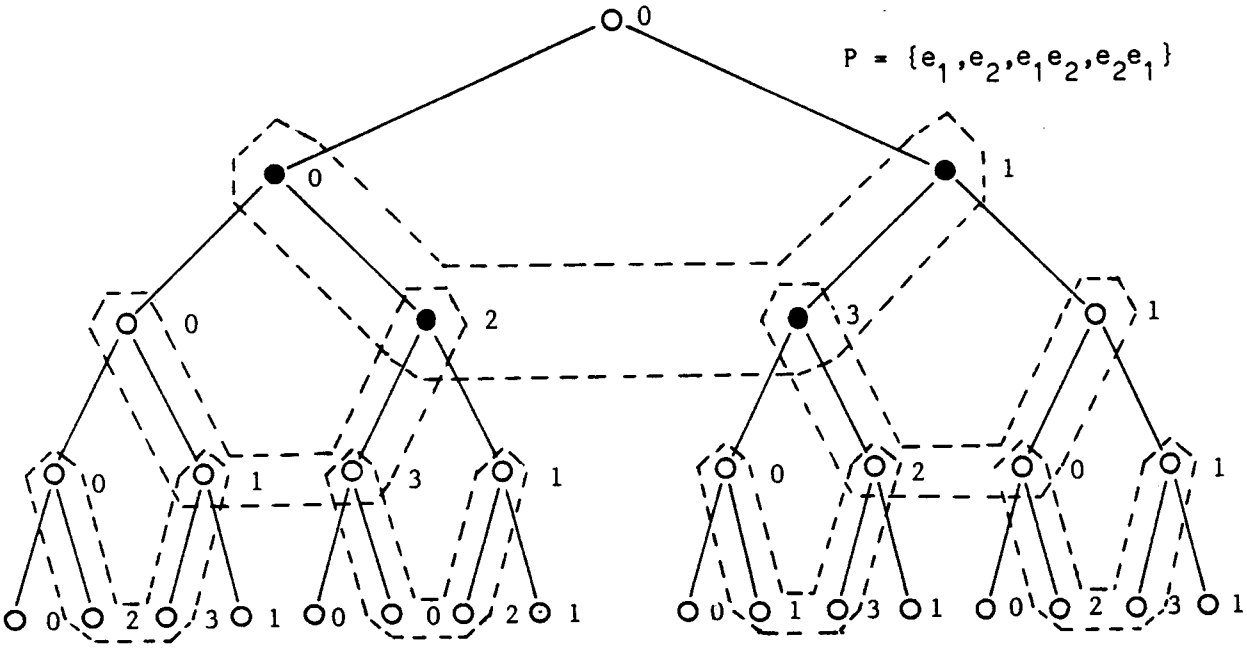


Figure 2.

The notion of the transposed template given in definition 2.4 will appear to be very convenient for determining the validity of skewing schemes.

Definition 2.4. Let the mapping tr:T→T be defined by $tr(\lambda_1\lambda_2\ldots\lambda_n) = \lambda_n\lambda_{n-1}\ldots\lambda_1$. Then

(i) $\alpha^* = tr(\alpha)$ is the transposed node of α

(ii) $P^* = tr(P)$ is the transposed template of P.

Lemma 2.5. In general it is not true that $m(P) = M$ iff $m(P^*) = M$.

Proof.
Consider a binary tree T. Take $P = \{\varepsilon, e_1e_1, e_1e_1e_1, e_2, e_2e_1e_1, e_2e_1e_1e_1\}$. Then $P^* = \{\varepsilon, e_1e_1, e_1e_1e_1, e_2, e_1e_1e_2, e_1e_1e_1e_2\}$ (See figure 3). Then $m(P)=6$ and $m(P^*)=8$. □

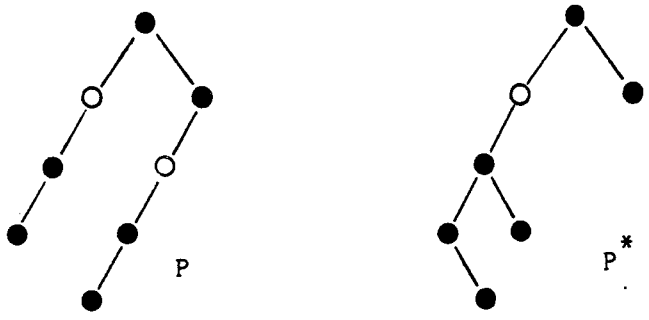


Figure 3.

With the aid of P^* we can define the conflict-number $C_p(\alpha)$ of a node α, which denote the number of different ways in which the template P can be laid on the node α.

Definition 2.6.
Given a template P. The conflict-number $C_p(\alpha)$ of a node α, with respect to P, is defined as $C_p(\alpha) = |\{\beta, \beta \in P^* \text{ and } \beta \text{ lies on the path from the root } \alpha_0 \text{ to } \alpha^*\}|$.

The following theorem gives a rather rough estimate on the number m(P).

Theorem 2.7. For any template P, $m(P) \leq |P| . \max_{\alpha \in T} C_p(\alpha)$.

Note that $|P| . \max_{\alpha \in T} C_p(\alpha)$ is bounded by $|P|^2$.

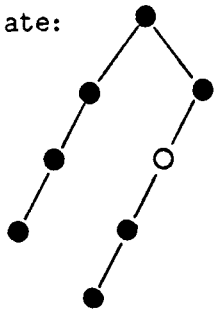
An exact characterisation of $m(P_1, P_2, \ldots, P_s)$ is obtained by reducing the problem of finding a valid skewing scheme for a tree to the problem of finding a valid skewing scheme for a strip data-structure, which is in some sense related to a d-dimensional array. Because of the fact that this reduction is very involved, we

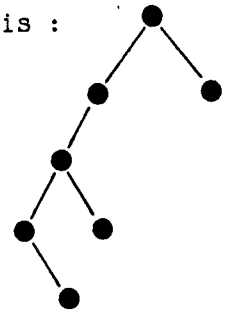
confine ourselves to state one of the results that follows from this characterisation. For a detailed analysis see [7].

A template P is called connected if for all $\alpha \in P$ and for all β on the path from the root α_0 to α^* : $\beta \in P$.

Theorem 2.8. Given a template P. If P^* is connected, then $m(P) = |P|$.

Note that the implication: if P is connected, then $m(P) = |P|$ is not true. See, e.g., the following example.

Example (T is a binary tree). Let P be the template:  .

So P^* is :  .

Then it follows that $|P| = 7 (= |P^*|)$, $m(P)=7$, but $m(P^*)=8$.

REGULAR SKEWING SCHEMES

Skewing schemes are only of practical interest if they can be represented in a simple and compact manner. For d-dimensional arrays the so-called periodic (regular) skewing schemes fulfill this condition [3,5,6]. A skewing scheme is called periodic (regular) if the following is true: whenever two points x and y of a data-structure (e.g. a d-dimensional array) are mapped into the same memory bank, then any two points which are in the same relative position to each other as x and y, are mapped into the same bank also. If we formalise this for trees, we get the following definition.

Definition 3.1.

(i) A skewing scheme $s: T \to \{0,1,\ldots,M-1\}$ is called semi-regular if the following condition holds: if for some $\alpha, \gamma \in T$: $s(\alpha) = s(\alpha\gamma)$, then for all $\delta \in T$: $s(\delta) = s(\delta\gamma)$.

(ii) A basis for a semi-regular skewing scheme s is the minimal set $B_s = \{\gamma_1, \gamma_2, \ldots\}$ such that

1. for all $\alpha \in T$, $i \geq 1$: $s(\alpha) = s(\alpha\gamma_i)$, and
2. for all skewing scheme s' : $T \to \{0,1,\ldots,M-1\}$ there exists a set $F_{s'} \subseteq T$, $|F_{s'}|$ is finite, such that from the condition

for all $\alpha \in T$, $i \geq 1$: $s'(\alpha) = s'(\alpha\gamma_i)$ follows that
for all $\alpha \in T$ there exists a $\delta \in F_{s'}$: $s'(\alpha) = s'(\delta)$.

In order to provide that a regular skewing scheme s is compactly representable we need that $|B_s|$ is finite. However, the following theorem shows that this approach does not yield the desired result.

Theorem 3.2. If s is a semi-regular skewing scheme and $|B_s|$ is finite, then $B_s = \{\alpha, |\alpha| = d\}$, for some d. ($|\alpha|$ denotes the length of the string α.)

So we see that the semi-regular skewing schemes s, with $|B_s|$ is finite, are only a very restrictive set of skewing schemes. From this we may conclude that we need a stronger notion than semi-regularity. Therefore we introduce the notion of regularity in definition 3.3.

Definition 3.3.
(i) A skewing scheme $s : T \to \{0,1,\ldots,M-1\}$ is called regular if the following condition holds: if for some $\alpha, \beta \in T$: $s(\alpha) = s(\beta)$, then for all $\gamma \in T$: $s(\gamma\alpha) = s(\gamma\beta)$.
(ii) A basis for a regular skewing scheme is the minimal set $B_s = \{[\alpha_1,\beta_1],[\alpha_2,\beta_2],\ldots\}$ such that
1. for all $\gamma \in T$, $i \geq 1$: $s(\gamma\alpha_i) = s(\gamma\beta_i)$, and
2. for all skewing scheme $s' : T \to \{0,1,\ldots,M-1\}$ there exists a set $F_{s'} \subseteq T$, $|F_{s'}|$ is finite, such that from the condition
for all $\gamma \in T$, $i \geq 1$: $s'(\gamma\alpha_i) = s'(\gamma\beta_i)$ follows that
for all $\alpha \in T$ there exists a $\delta \in F_{s'}$: $s'(\alpha) = s'(\delta)$.

Lemma 3.4. If s is regular then s is semi-regular.

These regular skewing schemes are very feasible for skewing a collection of templates. See, for instance, the next example.

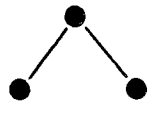
Example (T is again the complete and infinite binary tree).
Consider $B_s = \{[\varepsilon, e_1e_1], [\varepsilon, e_2e_2], [e_1,e_2e_1], [e_2,e_1e_2]\}$ and let $F_s = \{\varepsilon, e_1,e_2\}$. In addition let $s(\varepsilon)=0$, $s(e_1)=1$, $s(e_2) = 2$. Then for all $\alpha \in T$ $s(\alpha)$ is fixed. Hence s is a regular skewing scheme.
Furthermore it follows that s is valid for the template  .

Figure 4 shows the elements α of T with $s(\alpha) = 0$.

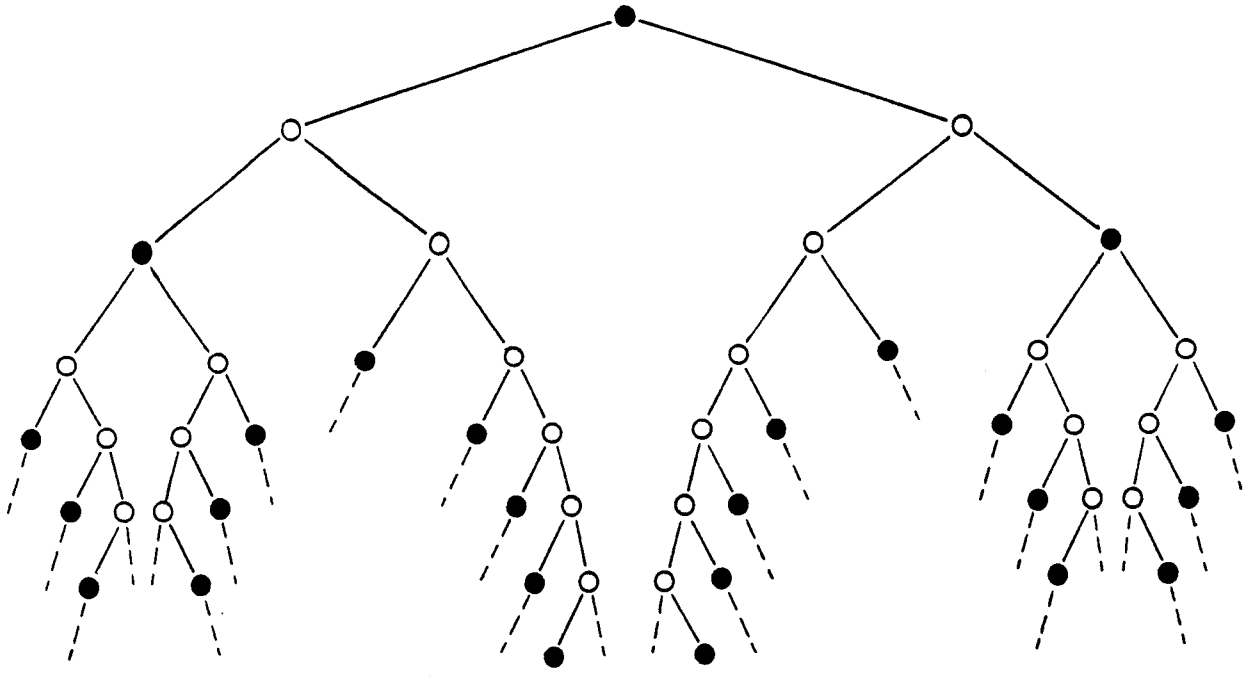


Figure 4.

In [7] conditions are given for a regular (and semi-regular) skewing scheme to be valid for an arbitrary collection of templates. With the help of these conditions the next two theorems are proven.

Theorem 3.5. Given a binary tree T, and the template $P = \{\alpha, |\alpha| \leq d\}$, for some $d \geq 0$. (Thus P is the complete subtree of depth d.) Then there exists a regular skewing scheme $s: T \to \{0,1,\ldots,2^{d+1}-1\}$ ($2^{d+1} = m(P)$), that is valid for P, and where $|B_s| = 2^{d+1}$.

A template P is called a double-path of length d on a binary tree, if there exists a path α_0(=the root), $\alpha_1, \alpha_2, \ldots, \alpha_{d-1}$ of length d-1, such that $P = \{\beta, \exists \alpha_i$ $(0 \leq i \leq d-1)$: $\beta = \alpha_i e_1$ or $\beta = \alpha_i e_2\}$.

Theorem 3.6. Given a binary tree T and C the collection of all double-paths of length d, for some $d \geq 1$. Then there exists a regular skewing scheme $s: T \to \{0,1,\ldots,2d-1\}$ ($2d = m(C)$), that is valid for C, where $|B_s| = 2(d+1)$.

These two theorems can be generalized in a natural way for k-ary trees, with $k>2$.

As final result we show that each regular skewing scheme can indeed be represented at a low cost and computed in a fairly short time.

Theorem 3.7. Given a regular skewing scheme s with $|B_s| = b$ and $\max_{[\alpha,\beta]\in B_s} |\beta| = d$. Then s can be represented in space $\leq (d+3).b$, such that for all $\alpha \in T$: $s(\alpha)$ can be computed in time $\leq b.d.|\alpha|$.

Proof.
Consider the subtree $\tilde{T}$ of T which is defined by $\tilde{T} = \{\gamma, \exists[\alpha,\beta] \in B_s: \gamma\gamma' = \beta^*, \text{ for some } \gamma' \in T\}$. Call the nodes γ of $\tilde{T}$, for which there exists a $[\alpha,\beta] \in B_s$, $\gamma' \in T$ with $\gamma\gamma' = \beta^*$, the leaves of $\tilde{T}$. Assign to each leaf γ of $\tilde{T}$, the node $B(\gamma) = \alpha$, if $[\alpha,\beta] \in B_s$ and $\gamma\gamma' = \beta^*$, for some γ'. Assign to each node $\in \tilde{T}$, with $\gamma^* \in F_s$, the number $f(\gamma) = s(\gamma^*)$. Then the following algorithm A computes $s(\alpha)$, for arbitrary α.

```
Algorithm A :
   γ := the root of T̃ ;
   ξ := α*    ;
   repeat until ξ = ε
       if γ is a leaf
         then  ξ := B(γ) ξ ;
               γ := the root of T̃
         else # ξ = e_j ξ'#
               ξ := ξ'    ;
               γ := γ e_j
       endif
   end repeat
   result := f(γ).
```

The algorithm A is correct and costs at most b.d. $|\alpha|$ time. Furthermore the representation costs are

$\leq$ "the size of $\tilde{T}$" + "the value's $f(\gamma)$" + "the value's $B(\gamma)$"

$\leq |\tilde{T}| + |F_s| + |B_s|.d = 2.|B_s| + |B_s| + |B_s|.d = (d+3).|B_s|$. □

REFERENCES

[1] P. Budnik and D.J.Kuck, "The organization and use of parallel memories", IEEE Trans. Comput., vol. C-20, pp.1566-1569, 1971.

[2] D.H.Lawrie, "Access and alignment in an array processor", IEEE Trans. Comput., vol. C-24, pp. 1145-1155, 1975.

[3] H.D. Shapiro, "Theoretical limitations on the use of parallel memories", IEEE Trans. Comput., vol. C-27, pp. 241-248, 1978.

[4] J. van Leeuwen and H.A.G. Wijshoff, "Data mappings in large parallel computers", Dep. Comput. Sci., Univ. Utrecht, Utrecht, the Netherlands, Tech. Rep. RUU-CS-83-11, 1983. (Also appeared in I.Kupka (ed.), GI-13 Jahrestagung, Informatik Fb 73, Springer Verlag, 1983, pp 8-20.)

[5] H.A.G. Wijshoff and J. van Leeuwen, "Periodic storage schemes with a minimum number of memory banks", Dep. Comput. Sci., Univ. Utrecht, Utrecht, the Netherlands, Techn. Rep. RUU-CS-83-4, 1983.

[6] H.A.G. Wijshoff and J. van Leeuwen, "The structure of periodic storage schemes for parallel memories", IEEE Trans. Comput., vol. C-34, pp. 501-505, 1985.

[7] H.A.G. Wijshoff, "Storing trees into parallel memories", Dep. Comput. Sci., Univ. Utrecht, Utrecht, the Netherlands, Techn. Rep. RUU-CS-85-.... (forthcoming), 1985.