在可重組網狀式多處理機上 找尋對稱字串和週期字串的平行演算法
Partitionable Bus-Based Parallel Algorithms for Finding all Initial Palindromes and Periods of a String

鍾國亮
Kuo-Liang Chung
臺灣科技大學資管系
Department of Information Management
Graduate Program of Information Engineering
National Taiwan University of Science and Technology
klchung@cs.ntust.edu.tw

陳秀娘
Hsiu-Niang Chen
萬能工商專校資管科
Department of Information Management
Van-Nung Institute of Technology
csn@a3.im.vit.edu.tw

摘要

給長度為 n 的字串，本篇論文提出在可重組網狀式多處理機上 O(1)時間找尋對稱字串和週期字串的平行演算法，然後，在相同的成本下，提出了一個分割的策略來解決處理器不足的情形.

關鍵字：對稱字串，平行演算法，週期字串

Abstract

Given a string of length n, this paper first presents an O(1) time parallel algorithm for finding all initial palindromes and periods of the string on an n×n reconfigurable mesh (RM). Then, we provide a partitionable strategy when the RM doesn't offer sufficient processors.

Keywords: palindromes, parallel algorithms, periods

1. Introduction

Palindromes are often studied in word puzzles [1] and complexity theory ([8], [9]). A palindrome is a symmetric string, it may be read the same forward and backward. Formally, a string $S[1..k]$ is said to be a palindrome if $S[i] = S[k-i+1]$ for $i = 1, .., k$. If the prefix $S[1..k]$ of the string $S[1..n]$ is a palindrome, we call the string $S[1..n]$ has an initial palindrome of length k. Given such a string S of length n, we want to find all initial palindromes for $k > 1$ and the shortest initial palindrome among these palindromes. For example, given a string $S[1..11] = aabccbaaefa$, its all initial palindromes are aa and $aabccbaa$; the shortest initial palindrome is aa.

A string $S[1..n]$ is said to have a period $S[1..p]$ of length $p\ (> 0)$ if $S[i] = S[i+p]$ for $i = 1, .., n-p$. In many string-matching (SM) applications, we often want to find all periods and the longest period of $S[1..n]$. For example, given a string $S = ababbcaba$, its all periods are $ababbc$ and $ababbcab$; the longest period is of length 8.

Some fast linear-time sequential algorithms have been developed for finding all initial palindromes and periods. Galil presented a parallel algorithm [7] for finding all initial palindromes and periods in $O(\log n)$ time on a CRCW-PRAM model using $O(n)$ processors. Recently, Breslauer and Galil [2] presented an optimal $O(\log\log n)$-time CRCW-PRAM algorithm using $O(n/\log\log n)$ processors for finding all initial palindromes and periods of a string.

Reconfigurable mesh (RM) is a powerful and promising parallel model. On RMs, some $O(1)$-time parallel algorithms for solving some matching problems ([3], [4], [5], [6]) have been developed. This paper first presents an $O(1)$ time parallel algorithm for finding all initial palindromes and the shortest initial palindrome on the RM using $O(n^2)$ processors and presents an $O(1)$ time parallel algorithm for finding all periods and the longest period on the same RM. To the best of our knowledge, this is the first time that such constant-time parallel algorithms are being presented for solving these problems. Then, a partitionable strategy is presented while preserving the same cost $O(n^2)$ if the RM doesn't provide sufficient processors. This overcomes the hardware limitation and is suitable for VLSI implementation.

The remainder of the paper is organized as follows. Section 2 introduces the RM computational model. In Section 3, we present the $O(1)$ time parallel algorithms for finding all initial palindromes and periods on the $n \times n$ RM, respectively. In Section 4, the partitionable strategy is presented when the RM doesn't provide sufficient processors. Some concluding remarks are included in Section 5.

2. Computational Model: RM

The parallel computational model used in this research is the RM. A RM has a reconfigurable bus system whose configuration can be dynamically changed. An $m \times n$ RM consists of $m \times n$ (m rows and n columns) identical processors arranged in a 2-D rectangular array with a reconfigurable bus system. The processor located in row i and column j for $1 \leq i \leq m$ and $1 \leq j \leq n$ is referred to as PE(i,j). Each processor has four ports denoted by N (north), S (south), E (east), and W (west), respectively.

We assume that the RM is operated in a SIMD model. All processors can work synchronously and setting internal connections, performing an arithmetic or boolean operation, broadcasting a value on a bus, or receiving a value from a specific bus only need $O(1)$ time. If there exists a bus between two ports, we assume that it takes $O(1)$ time to broadcast the data from one port to the other. In addition, at any given time only one value can be sent to a bus and processors read the bus if instructed to do so.

3. The $O(1)$ Time Parallel Algorithms

3.1 Finding all Initial Palindromes and the Shortest Initial Palindrome

We take the string $S = aabccbaaefa$ to demonstrate our $O(1)$ time parallel algorithm. On the $n \times n$ RM, our $O(1)$ time parallel algorithm for finding all initial palindromes is presented in the following five steps. Initially, suppose the string $S[1..n]$ has been fed into row 1 of the RM. That is, processor PE$(1,j)$ stores the data $S[j]$ for $1 \leq j \leq n$. Here, $n = 11$.

Algorithm_1:

Step 1: Establish a vertical bus system for each column. This configuration can be built by connecting the N and S ports of each processor. Then, each processor PE$(1,j)$ for $1 \leq j \leq n$ broadcasts its own data $S[j]$ to the others in the same column via the vertical bus system.

Step 2: Establish a horizontal bus system for each row. This configuration can be built by connecting the W and E ports of each processor. Then, each processor PE(j,j) for $1 \leq j \leq n$ broadcasts its own data $S[j]$ to the others in the same row via the horizontal bus system.

Step 3: Each processor first disconnects its connections. Then, except the processors in the first row, each processor connects its N and W ports.

Step 4: Each processor PE(i,j), $1 \leq i \leq n$ and $1 \leq j \leq n$, connects its E and S ports when $S[i] = S[j]$. Otherwise do nothing.

Step 5: Along the corresponding stairlike bus system, each processor PE$(1,j)$, $1 < j \leq n$, with the connection linking the E and S ports sends the value j from the E port to the processor PE$(i,1)$ for $1 \leq i \leq n$. Finally, each processor PE$(i,1)$ reports the value j $(=i)$ as the length of the initial palindrome if the S port of that processor receives the value. Fig. 1 illustrates the configuration of the RM after performing this step.

In our example, the processor PE(2, 1) reports 2 as the length of the initial palindrome; PE(8, 1) reports 8 as the length of the other initial palindrome. The following two additive steps are appended to find the shortest initial palindrome.

Step 6: Each processor PE$(i,1)$ for $1 \leq i \leq n$ first disconnects its connections. Then, each processor PE$(i,1)$ that doesn't receive the value j $(=i)$ connects its N and S ports.

Step 7: Each processor PE$(i,1)$ holding the value j $(=i)$ first sends the value j from the N port to the bus, then processor PE$(1,1)$ reports the received value j as the length of the shortest initial palindrome if the S port of PE$(1,1)$ receives the value j. In this example, processor PE$(1,1)$ reports 2 as the length of the shortest initial palindrome. By the same argument, we can also find the longest initial palindrome.

Since each step in Algorithm_1 takes $O(1)$ time, we have the following result.

Theorem 1. *The problem of finding all initial palindromes and the shortest (longest) initial palindrome can be solved in $O(1)$ time on the $n \times n$ RM.*

3.2 Finding all Periods and the Longest Period

We take the string $S = abaabcaba$ to demonstrate our $O(1)$ time parallel algorithm for finding all periods on the $n \times n$ RM. The algorithm consists of the following five steps.

Algorithm_2:

Step 1, Step 2: The same as Step 1 and 2 in Algorithm_1.

Step 3: Each processor first disconnects its connections. Then, except the processors in the first row, each processor connects its N and E ports.

Step 4: Each processor PE(i,j), $1 \leq i \leq n$ and $1 \leq j \leq n$, connects its W and S ports when $S[i] = S[j]$. Otherwise do nothing.

Step 5: Along the corresponding stairlike bus system, each processor PE$(1,j)$, $1 < j \leq n$, with the connection linking the W and S ports sends the value $j-1$ from the W port to the processor PE(i,n) for $1 \leq i \leq n$. Finally, each processor

PE(i,n) reports the value $j-1$ $(=n-i)$ as the length of the period if the S port of that processor receives the value. Fig. 2 illustrates the configuration of the RM after performing this step.

In our example, the processor PE$(1,9)$ reports 8 as the length of the period; PE$(3,9)$ reports 6 as the length of the other period. Similar to Step 6 and 7 in Algorithm_1, we can find the longest and shortest periods. In our example, PE$(1,9)$ reports 8 as the length of the longest period.

Since each step in Algorithm_2 takes $O(1)$ time, we have the following result.

Theorem 2. *The problem of finding all periods and the longest (shortest) period can be solved in $O(1)$ time on the $n \times n$ RM.*

4. The Partitionable Parallel Algorithms

Consider the case when the number of processors available is not enough. Given an RM consisting of $M \times N$ (M rows and N columns) processors, if the string is of length n, where $N < n$ or $M < n$, this section proposes a partitionable strategy to overcome this hardware limitation. As a result, our results are suitable for VLSI implementation.

Since finding all periods and the longest period is quite similar to finding all initial palindromes and the shortest initial palindrome, we only explore the partitionable strategy for the latter case. Without loss of generality, for finding all initial palindromes, we focus on the following two cases: (1)$M < n$; (2)$N < n$ and $M < n$.

A. Case 1: $M < n$

Assume the RM consists of 7×11 processors and the string used is the same as in Section 3.1. That is, $M = 7$, $N = 11$, and $n = 11$. Initially, suppose processor PE$(1,j)$ stores the data $S[j]$ for $1 \le j \le n$.

We partition the input string into $\hat{X}$ pipes, where $\hat{X} = \lceil \frac{n-1}{M-1} \rceil$, and each pipe has one overlapping entry shared with the last pipe. Besides this entry-sharing feature, the input string is arranged into a snakelike column-major order.

Our parallel algorithm processes these $\hat{X}$ pipes from the first pipe to the $\hat{X}$th pipe successively. For finding all initial palindromes, our partitionable parallel algorithm for $M < n$ case is described below.

Algorithm_3:

Step 1: The same as Step 1 in Algorithm_1.

For $X = 1$ **to** $\hat{X}$ /* we process these $\hat{X}$ pipes from the first one to the last one */

begin

Step 2: Establish a horizontal bus system for each row. Then, for odd (even) X each processor PE$(i, 2\lfloor \frac{X}{2} \rfloor (M-1)+i)$ (PE$(i, X(M-1)-i+2)$) for $1 \le i \le M$ broadcasts its own data $S[2\lfloor \frac{X}{2} \rfloor (M-1)+i]$ ($S[X(M-1)-i+2]$) to the others in the same row via the horizontal bus system.

Step 3: Each processor first disconnects its connections. Then, except the processors in the first (Mth) row for odd (even) X, each processor connects its N (S) and W ports.

Step 4: Each processor PE(i,j), $1 \le i \le M$ and $1 \le j \le n$, connects its E and S (N) ports for odd (even) X when $S[2\lfloor \frac{X}{2} \rfloor (M-1)+i] = S[j]$ ($S[X(M-1)-i+2] = S[j]$). Otherwise do nothing.

Step 5: *Case* $X = 1$: Along the corresponding stairlike bus system, with the connection linking the E and S ports, each processor PE$(1,j)$, $1 < j \le n$, sends the value j from the E port to the processor PE$(i,1)$ for $1 \le i \le M$ or PE(M,k) for $1 \le k \le n-M+1$. Finally, each processor PE$(i,1)$ reports the value j $(=i)$ as the length of the initial palindrome if the S port of that processor receives the value; PE(M,k) keeps the received value if the S port of that processor receive the value.

Case $X > 1$: Along the corresponding stairlike bus system, with the connection linking the E and S (N) ports for odd (even) X, each processor PE$(1,j)$ (PE(M,j)) holding the value sent from pipe $X-1$, $1 < j \le n-(X-1)*(M-1)$, sends the received value from the E port to the processor PE$(i,1)$ for $1 \le i \le M$ or to PE(M,k) (PE$(1,k)$) for $1 \le k \le n - X*(M-1)$. Finally, each processor PE$(i,1)$ reports the received value ($= 2\lfloor \frac{X}{2} \rfloor (M-1)+i$ when X is odd; $=X(M-1)-i+2$ when X is even) as the length of the initial palindrome if the S (N) port of that processor receives the value; PE(M,k) (PE$(1,k)$) keeps the received value if the S (N) port of that processor receive the value.

end

For $X = 1$ in our example, PE$(2,1)$ reports 2 as the length of initial palindrome; processor PE$(7,2)$ keeps the value 8, which will be used by pipe 2 . For $X = 2$, PE$(6,1)$ reports 8

($=X*(M-1)-i+2$) as the initial palindrome.

Modifying Algorithm_3 slightly, the problem of finding the shortest initial palindrome can also be solved in $O(\hat{X})$ time on the RM using $O(\frac{n^2}{\hat{X}})$ processors. Under the same cost $O(n^2)$ as in Theorem 1, we have the following result.

Theorem 3. *For $M < n$, the problem of finding all initial palindromes and the shortest initial palindrome can be solved in $O(\hat{X})$ time on the RM using $O(Mn)$ ($=O(\frac{n^2}{\hat{X}})$) processors, where $\hat{X} = \lceil \frac{n-1}{M-1} \rceil$.*

B. Case 2: $N < n$ and $M < n$

Without loss of generality, suppose $M = N < n$. Following the same string used in Section 3.1, the RM consists of 7×7 processors. That is, $M = 7$ and $n = 11$. Initially, suppose processor PE$(1, j)$, $2 \le j \le M-1$, stores the data $S[2J(M-1)+j]$ and $S[2J(M-1)+2M-j]$ for $0 \le J \le \lfloor \frac{n-2}{2(M-1)} \rfloor$. Specifically, PE$(1,1)$ stores the data $S[2J(M-1)+1]$; PE$(1, M)$ stores the data $S[2J(M-1)+M]$. That is, the input string is arranged into the snakelike row-major order for the first row and totals $\hat{X}$ pipes, where $\hat{X} = \lceil \frac{n-1}{M-1} \rceil$.

In our example, $\hat{X} = 2$ and the string=*aabccbaaefa* is arranged into the snakelike row-major order for rows and is arranged into the snakelike column-major order for columns. Our parallel algorithm processes these $\hat{X}^2$ pipes from pipe $\hat{X}$ to pipe 1 for each fixed pipe X, $1 \le X \le \hat{X}$, successively. Our partitionable parallel algorithm for this case is listed below.

Algorithm_4:

```
For $X = 1$ to $\hat{X}$
begin
    Step 1: Establish a vertical bus system for each column. For odd (even) $X$, each processor PE$(1,j)$ for $1 \le j \le M$ broadcasts its own data $S[2\lfloor \frac{X}{2} \rfloor (M-1)+j]$ ($S[X(M-1)-j+2]$) to the others in the same column via the vertical bus system.
end
For $X = 1$ to $\hat{X}$
begin
    Step 2: Establish a horizontal bus system for each row. For odd (even) $X$, each processor PE$(i,i)$ for $1 \le i \le M$ broadcasts its own data $S[2\lfloor \frac{X}{2} \rfloor (M-1)+i]$ ($S[X(M-1)-i+2]$) to the others in the same row via the horizontal bus system.
end
For $X = 1$ to $\hat{X}$ /* we process these $\hat{X}$ pipes from the first one to the last one */
  For $Y = \hat{X}$ to 1 /* we process these $\hat{X}$ pipes from the last one to the first one */
  begin
      Step 3: Each processor first disconnects its connections. Then, except the processors in the first ($M$th) row for odd (even) $X$, each processor connects its $N$ ($S$) and $W$ ports when $Y$ is odd; connects its $N$ ($S$) and $E$ ports when $Y$ is even.
      Step 4: Each processor PE$(i,j)$, $1 \le i \le M$ and $1 \le j \le M$, connects its $E$ and $S$ ($N$) ports for odd (even) $X$ and odd $Y$ when $S[2\lfloor \frac{X}{2} \rfloor (M-1)+i] = S[2\lfloor \frac{Y}{2} \rfloor (M-1)+j]$ ($S[X(M-1)-i+2] = S[2\lfloor \frac{Y}{2} \rfloor (M-1)+j]$); each processor PE$(i,j)$, $1 \le i \le M$ and $1 \le j \le M$, connects its $W$ and $S$ ($N$) ports for odd (even) $X$ and even $Y$ when $S[2\lfloor \frac{X}{2} \rfloor (M-1)+i] = S[Y(M-1)-j+2]$ ($S[X(M-1)-i+2] = S[Y(M-1)-j+2]$). Otherwise do nothing.
      Step 5: Case $X = 1$: Along the corresponding stairlike bus system, with the connection linking the $E$ ($W$) and $S$ ports for odd (even) $Y$, each processor PE$(1,j)$, $1 \le j \le M$, sends the value $2\lfloor \frac{Y}{2} \rfloor (M-1)+j$ ($Y(M-1)-j+2$) from the $E$ ($W$) port to the processor PE$(i,1)$ (PE$(i,M)$) for $1 \le i \le M$ and PE$(i,M)$ (PE$(i,1)$) holding the value sent from pipe $Y+1$, $1 \le i \le M$, send the received value from the $E$ ($W$) port to PE$(M,k)$ for $1 \le k \le M$. Finally, each processor PE$(i,1)$ reports the value $2\lfloor \frac{Y}{2} \rfloor (M-1)+j$ as the initial palindrome if the $S$ port of that processor receives the value and $Y = 1$; otherwise PE$(i,1)$ (PE$(i,M)$) and PE$(M,k)$ keep the received value if the $S$ ports of those processors receive the value.
              Case $X > 1$ and $Y$ is odd: Along the corresponding stairlike bus system, with the connection linking the $E$ and $S$ ($N$) ports for odd (even) $X$ and odd $Y$, processor PE$(1,j)$ (PE$(M,j)$) holding the value sent from pipe $X-1$ and processor PE$(i,M)$ holding the value sent from pipe $Y+1$, where $1 \le j \le M$ and $1 \le i \le M$, send the received value from the $E$ port to the processor PE$(i,1)$ for $1 \le i \le M$ or PE$(M,k)$ (PE$(1,k)$) for $1 \le k \le M$. Finally, each processor PE$(i,1)$ reports the received value ($= 2\lfloor \frac{X}{2} \rfloor (M-1)+i$ when
```

X is odd; $=X(M-1)-i+2$ when X is even) as the length of the initial palindrome if the S (N) port of that processor receives the value and $Y=1$; otherwise PE$(i,1)$ and PE(M,k) (PE$(1,k)$) keep the received value if the S (N) ports of those processors receive the value.

Case $X>1$ and Y is even: Along the corresponding stairlike bus system, with the connection linking the W and S (N) ports for odd (even) X and even Y, processor PE$(1,j)$ (PE(M,j)) holding the value sent from pipe $X-1$ and processor PE$(i,1)$ holding the value sent from pipe $Y+1$, where $1\le j\le M$ and $1\le i\le M$, send the received value from the W port to the processor PE(i,M) for $1\le i\le M$ or PE(M,k) (PE$(1,k)$) for $1\le k\le M$. Finally, PE(i,M) and PE(M,k) (PE$(1,k)$) keep the received value if the S (N) ports of those processors receive the value.

end

For $X=1$ and $Y=2$ in our example, processor PE(1,7) keeps the value 7 and PE(2,7) keeps the value 8 (see Fig. 1(a)). For $X=1$ and $Y=1$, PE(2,1) reports 2 as the length of the initial palindrome; PE(7,2) keeps the value 8 (see Fig. 1(b)). For $X=2$ and $Y=2$, no processor reports or keeps the value (see Fig. 1(c)). For $X=2$ and $Y=1$, PE(6,1) reports 8 ($=X*(M-1)-i+2$) as the length of the initial palindrome (see Fig. 1(d)).

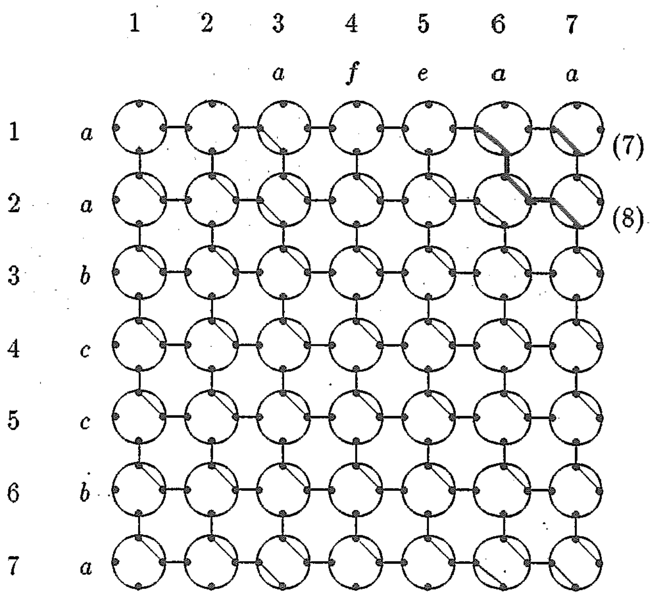
Modifying Algorithm_4 slightly, the problem of finding the shortest initial palindrome can also be solved in $O(\hat{X}^2)$ time on the RM using $O(\frac{n^2}{\hat{X}^2})$ processors. Under the same cost $O(n^2)$ as in Theorem 1, we have the following results.

Theorem 4. *For $N<n$, $M<n$ and $M<=N$, the problem of finding all initial palindromes and the shortest initial palindrome can be solved in $O(\hat{X}^2)$ time on the RM using $O(M^2)$ ($=O(\frac{n^2}{\hat{X}^2})$) processors, where $\hat{X}=\lceil\frac{n-1}{M-1}\rceil$.*

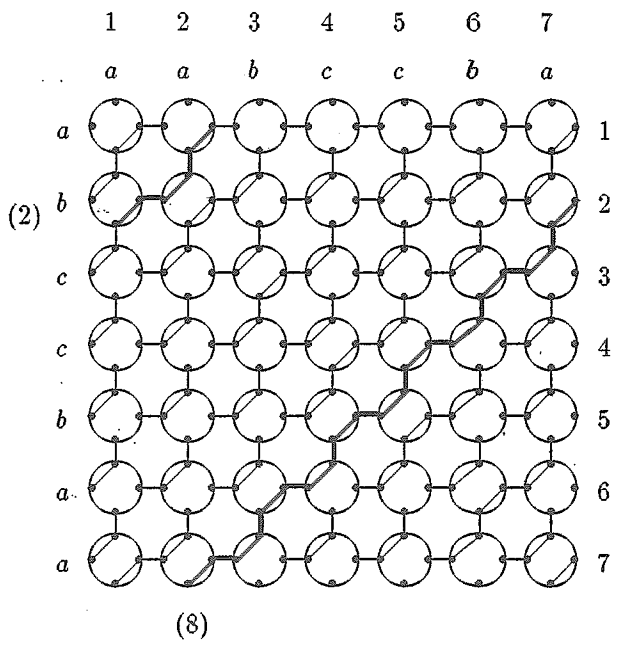
Corollary 5. *For $N<n$, $M<n$ and $M<=N$, the problem of finding all periods and the longest period can be solved in $O(\hat{X}^2)$ time on the RM using $O(M^2)$ ($=O(\frac{n^2}{\hat{X}^2})$) processors, where $\hat{X}=\lceil\frac{n-1}{M-1}\rceil$.*

5. Conclusions

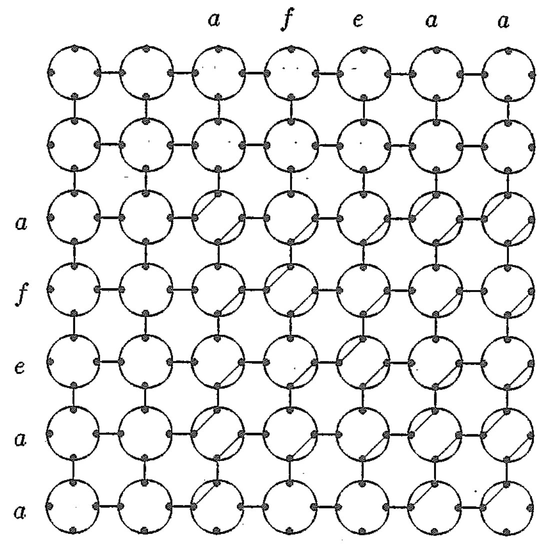
Finding all palindromes and periods often arises in word puzzles and complexity theory. Given a string of length n, the main contributions of this paper are twofold: first we have presented $O(1)$ time parallel algorithms for finding all initial palindromes, the shortest initial palindrome, all periods, and the longest period on the $n\times n$ RM; second, under the same cost $O(n^2)$, we have presented a partitionable strategy when the RM doesn't offer sufficient processors, and this result is very suitable for VLSI implementation. Applying the result in [3], the results of this paper can be applied to solve the same problems involved in this paper for the run-length coded strings.



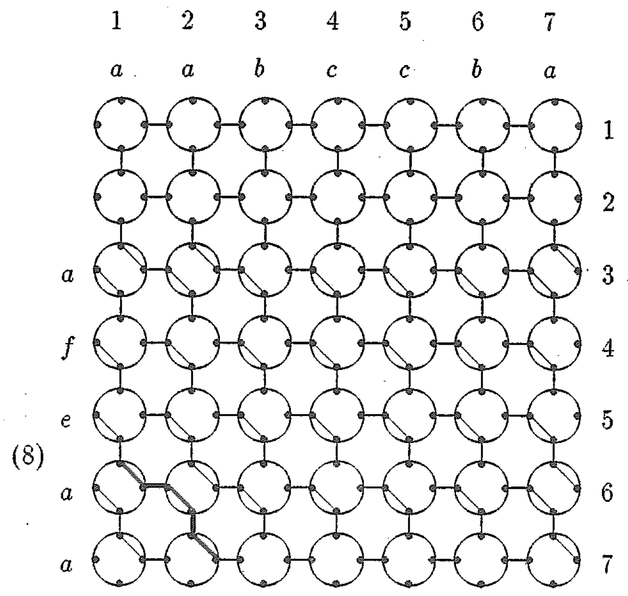
(a) X=1 and Y=2



(b) X=1 and Y=1



(c) X=2 and Y=2



(d) X=2 and Y=1

Fig. 1. Some snapshots for simulating Algorithm_4.

References

[1] Bergerson, H. W.: Palindromes and anagrams. Dover. New York, 1973.

[2] Breslauer, D., Galil, Z.: Finding all periods and initial palindromes of a string in parallel. Algorithmica *14* 355–366 (1995).

[3] Chen, H. N., Chung, K. L.: Partitionable bus-based string-matching algorithm for run-length coded strings with VLDCs. Special Issue of VLSI Design on High-Performance Bus-Based Architectures, (1997), to appear.

[4] Chung K. L.: Fast string matching algorithms for run-length coded strings. Computing *54*, 119–125 (1995).

[5] Chung K. L.: $O(1)$ time parallel string-matching algorithm with VLDCs. Pattern Recognition Letters *17*, 475–479 (1996)

[6] Chung, K. L.: Image template matching on reconfigurable meshes. Parallel Processing Letters, (1996), to appear.

[7] Galil, G.: Optimal parallel algorithms for string matching. Inform. and Control *67*, 144–157 (1985).

[8] Hopcroft, J. E., Ullman, J. D.: Introduction to automata theory, languages and computation. Addison-Wesley, Reading, MA, 1979.

[9] Lothaire, M.: Combinatorics on words. Addison-Wesley, Reading, MA, 1983.