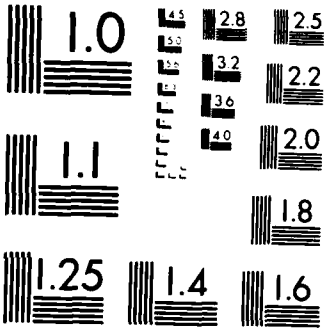
AD-A171 686 A PARALLEL FIRST-ORDER LINEAR RECURRENCE SOLVER(U) 1/1
JOHNS HOPKINS UNIV BALTIMORE MD DEPT OF ELECTRICAL
ENGINEERING AND COMPUTER SCIENCE G G MEYER ET AL.
UNCLASSIFIED SEP 86 JHU/EECS-86/07 AFOSR-85-0097 F/G 9/2 NL



A PARALLEL FIRST-ORDER LINEAR RECURRENCE SOLVER

Gerard G. L. Meyer and Louis J. Podrazik

REPORT JHU/EECS-86/07

ELECTRICAL ENGINEERING & COMPUTER SCIENCE

(12)

A PARALLEL FIRST-ORDER LINEAR RECURRENCE SOLVER

Gerard G. L. Meyer and Louis J. Podrazik

REPORT JHU/EECS-86/07

Electrical Engineering and Computer Science Department

The Johns Hopkins University

Baltimore, Maryland 21218

This work was supported by the Air Force Office of Scientific Research under Contract AFOSR-85-0097.

ABSTRACT

In this paper we present a parallel procedure for the solution of first-order linear recurrence systems of size N when the number of processors p is small in relation to N. We show that when $1 < p^2 \leq N$, a first-order linear recurrence system of size N can be solved in $5(N-1)/(p+1)$ steps on a p processor SIMD machine and at most $5(N - \frac{1}{2})/(p + \frac{3}{2})$ steps on a p processor MIMD machine. As a special case, we further show that our approach precisely achieves the lower bound $2(N-1)/(p+1)$ for solving the parallel prefix problem on a p processor machine.

I. INTRODUCTION

In this paper we present a parallel algorithm to solve the well-known first-order linear recurrence system $R\langle N,1\rangle$ when the number of processors p is small in relation to N, and where $R\langle N,1\rangle$ is defined as follows:

$R\langle N,1\rangle$: Given N, given $b = (b_1, b_2, \ldots, b_N)$, and given $a = (a_2, a_3, \ldots, a_N)$, compute $x = (x_1, x_2, \ldots, x_N)$ such that $x_1 = b_1$ and $x_i = a_i x_{i-1} + b_i$ for $i = 2,3,\ldots,N$.

We present both SIMD and MIMD versions of the algorithm. We analyze the SIMD version by first considering a simplified shared memory model of parallel computation that facilitates comparison with previous work. In that model the parallelism exhibited by the proposed algorithm is examined in terms of data dependencies only, therefore allowing us to determine the idealized performance of the procedure. We then consider a second model of computation which consists of a SIMD p processor ring configuration with a broadcasting capability. In that model interprocessor communication is taken into account, and a more realistic analysis of the algorithm is performed. The MIMD version of the algorithm is analyzed by considering the same simplified model as in the SIMD case, with the exception that the same operation need not be performed by all processors at the same time. Finally, we observe that the algorithm can be efficiently mapped to an MIMD p processor ring configuration with a broadcasting capability.

Many algorithms have been proposed to solve linear recurrences in parallel, each with different objectives. Earlier results assumed the availability of an unlimited number of processing elements and were concerned with determining the number of processors necessary to achieve minimal computational time

[KOG73], [CHE75], [KUC76], [SAM77]. Later, limited processor solutions were considered. Chen, Kuck and Sameh [CHE78] presented a SIMD algorithm that solved M-th order systems in $(\frac{N}{p})(2M^2+3M) + O(M^2\log_2 \frac{p}{M})$ steps, but did not discuss any specific parallel implementation. Gajski [GAJ81] improved upon this result by performing the SIMD computation in less than $(\frac{N}{p})(2M^2+3M)$ steps using $p \leq N^{1/2}$ processors in a shared memory architecture. In this paper we show that by using a SIMD p processor ring network modified to support global broadcasting, the number of steps required to solve a first-order linear recurrence of size $N \geq p^2$ is $5(N-1)/(p+1)$. This improves upon the results of [CHE78] and [GAJ81] for the first-order case when $N > p^2$. Our approach is a generalization of the matrix factorization technique presented in [MEY85], and it reduces to the SIMD procedure presented in [GAJ81] when $N = p^2$.

Moreover, when $a_i = 1$, for all i in [1,2,...,N], $R<N,1>$ reduces to a particular form of the parallel prefix problem. When $N > p$, Kruskal, Rudolph and Snir [KRU85] present an algorithm which solves the parallel prefix problem in $2N/p + 2log_2p - 2$ steps. Snir [SNI86] improves upon this approach when $N \geq p^2$ by solving the problem in $2N/(p+1) + O(1)$ steps, which is within a constant additive term of the lower bound. In case of parallel prefix, we show that our algorithm precisely achieves the lower bound $2(N-1)/(p+1)$ established by Snir [SNI86] when $N \geq p^2$.

Carlson [CAR84] considered mapping the computation of first-order linear recurrence systems to perfect shuffle and cube-connected systems. A common feature of both Carlson's algorithm and our algorithm is that the computation is organized so that the transfer of input and output data can be performed

concurrently with the execution of the algorithm, thus providing a balance between I/O and processing loads.

An important problem related to the parallel solution of $R<N,1>$ is the parallel evaluation of general arithmetic expressions [BRE74]. In the case of first-order linear recurrences of size N, the problem is to efficiently evaluate x_N in parallel using p processors. In that case, we observe that when applied to computing x_N only, a straightforward application of our approach does not improve upon existing results [BRE74], [CHE78].

In order to specify the parallelism exhibited by our algorithms, we augment those statements which can be executed in parallel. We use the following syntax:

```
FORALL i ε S DO IN PARALLEL
    BODY                                  /* Comments */
END FORALL
```

which indicates that the BODY may be executed concurrently for each i in the set S.

The SIMD procedure to solve $R<N,1>$ is presented in Section II; in Section III we discuss a parallel implementation of the algorithm, and in Section IV we present an MIMD version of our algorithm to solve $R<N,1>$. Finally, our conclusions comprise Section V.

II. THE SIMD ALGORITHM

The abstract model of SIMD parallel computation (Figure 1) considered in this section consists of a global parallel memory, p parallel processors, and an interconnection network, where all processors perform the same operation at each time step. We further simplify the model by making the following assump-

tions:

A1. Each arithmetic operation (addition or multiplication) is performed in unit time, referred to as a step.

A2. There are no accessing conflicts in global memory; furthermore, all data is assumed to initially reside in global memory.

A3. There is no time required to access global memory.

This simplistic model allows the parallelism of the proposed algorithm to be analyzed without introducing the added complexity of the implementation. In Section III we map the algorithm to a specific computational network and we analyze the corresponding implementation.

Given N and p, our approach to solving $R\langle N,1\rangle$ consists of partitioning $R\langle N,1\rangle$ into a sequence of $\left\lceil \frac{N-1}{p^2-1} \right\rceil$ reduced recurrence systems $R\langle n,1\rangle$, each of size $n = p^2$, except for the last recurrence system which may be of size less than p^2. Each $R\langle n,1\rangle$ is then solved in parallel with its initial value taken as the final value obtained from the previously solved reduced recurrence system, except for the first recurrence that uses $x_1 = b_1$. The first $R\langle n,1\rangle$ is solved as follows: (i) each of the p processors concurrently computes a partial solution for a different x_i; (ii) after p parallel iterations p^2 partial solutions have been computed, one for each x_i, i in $[1,2,...,p^2]$, where the partial solutions for x_i, i in $[1,2,...,p]$, are precisely the solutions for $R\langle N,1\rangle$; (iii) based upon x_p the next p partial solutions x_i, i in $[p+1,p+2,...,2p]$ are then updated in parallel to their correct values. After $p-1$ parallel update iterations $R\langle n,1\rangle$ is solved. The next reduced recurrence system of size p^2 is then solved with x_{p^2} as its initial value. We continue in this manner until the last $R\langle n,1\rangle$ is solved. Since the initial and final values of each $R\langle n,1\rangle$ overlap, the complete

solution of $R<N,1>$ requires solving $\left\lceil \frac{N-1}{p^2-1} \right\rceil$ reduced recurrence systems.

We now describe the SIMD algorithm to solve $R<N,1>$ when $\frac{N-1}{p^2-1}$ is an integer. For ω in $[0,1,...,(N-1)/(p^2-1)]$, let the index sets S_ω and T_ω be defined as

$$S_\omega = \{1 + mp + \omega(p^2-1) : m = 1,2,...,p-1\}$$

and

$$T_\omega = \{1 + mp + \omega(p^2-1) : m = 0,1,...,p-1\}.$$

```
1. PROCEDURE R(N,p,a,b)
2.   x_1 := b_1
3.   FOR ω := 0 TO (N-1)/(p^2-1) - 1 DO          /* Solve each R<n,1> */
4.     FORALL i ∈ S_ω DO IN PARALLEL              /* Begin Coefficient Computation Phase */
5.       A[i,i] := a_i;
6.     END FORALL
7.     FOR i := 1 TO (p-1) DO
8.       FORALL j ∈ S_ω DO IN PARALLEL
9.         A[i+j,j] := a_{i+j} A[i+j-1,j];
10.      END FORALL
11.    END FOR                                    /* End Coefficient Computation Phase */
12.    FORALL i ∈ S_ω DO IN PARALLEL              /* Begin Partial Solution Phase */
13.      x_i := b_i;
14.    END FORALL
15.    FOR i := 1 TO (p-1) DO
16.      FORALL j ∈ T_ω DO IN PARALLEL
17.        x_{i+j} := a_{i+j} x_{i+j-1} + b_{i+j};
```

```
18.       END FORALL
19.     END FOR                                   /* End Partial Solution Phase */
20.     FOR i ∈ S_ω DO                            /* Begin Solution Update Phase */
21.       FORALL j := 0 TO (p−1) DO IN PARALLEL
22.         x_{i+j} := A[i+j,i] x_{i−1} + x_{i+j};
23.       END FORALL
24.     END FOR                                   /* End Solution Update Phase */
25.   END FOR
26. END PROCEDURE
```

The preceding algorithm sequentially solves $\frac{N-1}{p^2-1}$ reduced recurrence systems of size p^2, each in parallel. Each reduced system is solved in three phases: the coefficient computation phase consists of the execution of loops 4 and 7 and computes all coefficients of the form $a_{i+j}a_{i+j-1}\cdots a_j$ which are needed later during the solution updates; the partial solution phase consists of the execution of loops 12 and 15 and computes p^2 partial solutions, in which the first p partial solutions are the actual solutions; and finally, the solution update phase consists of the execution of loop 20 in which the coefficients computed in the first phase are used to update the next p estimates at each iteration. The complete solution to $R\langle N,1\rangle$ is therefore obtained after executing $\frac{N-1}{p^2-1}$ iterations of loop 3. An example illustrating the computations performed by the algorithm is given in Figure 2 for the case $N = 17$ and $p = 3$ where the notation x_i is used to indicate a correct value for the solution. Note that each computational level may be performed in parallel using at most three processors.

Our model assumptions imply that we need only consider computational operations when determining the number of steps required by the algorithm.

Therefore we must examine those computations performed in loops 7, 15 and 20. The execution of loop 7 is performed $p-1$ times, each iteration requiring $p-1$ processors to concurrently perform a single multiplication, thus loop 7 requires $p-1$ steps. Both loops 15 and 20 are iterated $p-1$ times, each iteration requiring p processors to concurrently perform a multiplication followed by an addition. Thus, loops 15 and 20 each require $2(p-1)$ steps. The total number of steps required to solve each reduced recurrence system $R<p^2,1>$ using p processors is therefore $5(p-1)$, and hence the resulting theorem follows.

Theorem 1: Given N and p such that $\frac{N-1}{p^2-1}$ is an integer, the number of steps required to solve the linear recurrence system $R<N,1>$ using a SIMD parallel computer with p processors is $\frac{5(N-1)}{p+1}$.

If $\frac{N-1}{p^2-1}$ is not an integer, our approach to solving $R<N,1>$ requires that we solve the reduced recurrence system $R<n_r,1>$, where $n_r < p^2$. One approach to solving $R<n_r,1>$ is to use a technique which is known to be efficient whenever $n_r < p^2$. Applicable techniques include the algorithms presented by Chen, et al. [CHE78] and Kogge and Stone [KOG73]; however, these techniques are not desirable because they require the machine to store and execute multiple algorithms based upon the size of the recurrence system. A less efficient but more practical approach to solving $R<n_r,1>$ consists of using the proposed technique to solve the augmented system $R<p^2,1>$ and simply terminate the computation when the last x_i is computed. In that case the number of steps required by the algorithm is at most $\left\lceil \frac{N-1}{p^2-1} \right\rceil 5(p-1)$.

Finally, we make the observation that the above SIMD algorithm most notably differs from the approaches presented in [CHE78] and [GAJ81] in that our

approach partitions the problem and sequentially solves a series of reduced recurrences of size p^2. However, when $N = p^2$, our approach reduces to that of [GAJ81], except that Gajski presents the coefficient computation and partial solution phases as a single computational phase. Moreover, when $N \geq p^2$, the algorithm of Chen, et al. [CHE78] is less efficient than Gajski's as a result of implementing an extra computational phase in which a separate first-order recurrence of size p is solved using p processors, requiring an additional $2log_2p$ steps.

When N and p are powers of two, the algorithm of Chen, et al. requires $\frac{5N}{p} + 2\log_2 p - 5$ steps [CHE78] and when $\frac{N}{p^2}$ is an integer, Gajski's SIMD algorithm requires $\frac{N}{p^2}(5p-3) - 2$ steps [GAJ81], whereas when $\frac{N-1}{p^2-1}$ is an integer, our SIMD algorithm requires only $\frac{5(N-1)}{p+1}$ steps. For example, when $N = 2^{18}$ and $p = 2^3$, the number of steps required by the SIMD algorithms presented in [CHE78], [GAJ81] and this paper are 163,841, 151,550 and 145,635, respectively.

Finally, when $a_i = 1$, for all i in $[1,2,...,N]$, $R<N,1>$ is a particular form of the parallel prefix problem and reduces to computing the cascade sums (b_1+b_2), $(b_1+b_2+b_3)$, ..., $(b_1+b_2+ \cdots +b_N)$ in parallel using p processors. The following corollary is a direct consequence of Theorem 1.

Corollary 1: Given N and p such that $\frac{N-1}{p^2-1}$ is an integer, the number of steps required to solve the parallel prefix problem using a SIMD parallel computer with p processors is $\frac{2(N-1)}{p+1}$.

Thus, when $\frac{N-1}{p^2-1}$ is an integer, our SIMD algorithm precisely achieves

the parallel prefix computational lower bound $\frac{2(N-1)}{p+1}$ established by Snir [SNI86]. This result improves upon existing approaches to solving the parallel prefix problem when $N > p^2$. In that case the parallel prefix problem is solved in $\frac{2N}{p+1} + O(1)$ steps by Snir's algorithm [SNI86], $2\frac{N}{p} + \log_2 p$ - 2 steps by the data independent algorithm presented by Kruskal, et al. [KRU85], and $\frac{N}{p^2}(2p-1) - 1$ steps by Gajski's algorithm [GAJ81].

III. THE SIMD PARALLEL IMPLEMENTATION

The abstract model of SIMD parallel computation presented in Section II neglected the issues of data organization and alignment as well as communication overhead, all of which are highly machine dependent. We now present a parallel implementation of the proposed algorithm that takes these issues into account. The SIMD model of computation considered in this section (Figure 3) consists of p processors executing the same operation in lock step, with each processor containing its own local storage. The processors are interconnected by a unidirectional ring network in which processor i transfers data to processor $i+1$, i in $[1,2,...,p-1]$ and processor p transfers data to processor 1. Furthermore, we assume that the network possesses a broadcasting capability that allows any processor to broadcast data to all other processors. The time required by the algorithm will be determined under the following assumptions:

A1. Each arithmetic operation (addition or multiplication) is performed in unit time, referred to as a step.

A2. Interprocessor transfers require one step.

A3. Data broadcasts require one step.

A4. Each a_i and b_i required by a processor is assumed to initially reside in the local memory of that processor.

In order to determine an efficient processor assignment, we first make the observation that the p consecutive partial solutions updated at each iteration of the update phase of the algorithm must reside in a different processor. Furthermore, both the coefficient computation phase and the partial solution phase of the algorithm exhibit explicit data dependencies which must be preserved. These constraints can be satisfied if we rotate the processor assignment at each parallel iteration of the algorithm, and in that case, the algorithm can be directly mapped to a SIMD p processor unidirectional ring network with broadcasting capability. Figure 4 illustrates such a processor assignment and the corresponding communication requirements for the case $N = 17$ and $p = 3$.

We now present the algorithm to solve $R<N,1>$ as executed by processor k, for all k in $[1,2,...,p]$.

1. PROCEDURE $R_k(N,p,a,b)$
2. $x_1 := b_1$
3. FOR $\omega := 0$ TO $(N-1)/(p^2-1) - 1$ DO /* Solve each $R<n,1>$ */
4. $A[i,i] := a_i$; /* Begin Coefficient Computation Phase */
   /* $i = 1 + (k-1)p + \omega(p^2-1)$ */
5. FOR $i := 1$ TO $(p-1)$ DO
6. $A[i+j,j] := a_{i+j}\ A[i+j-1,j]$;
   /* $j = (1+(k-i-1)p)\ mod\ p^2 + \omega(p^2-1)$ */
7. END FOR /* End Coefficient Computation Phase */
8. IF $k = 1$ THEN $x_i := x_i$ ELSE $x_i := b_i$; /* Begin Partial Solution Phase */
   /* $i = 1 + (k-1)p + \omega(p^2-1)$ */
9. FOR $i := 1$ TO $(p-1)$ DO

10. $x_{i+j} := a_{i+j}\ x_{i+j-1} + b_{i+j}$;

/* $j = (1+(k-i-1)p)\ mod\ p^2 + \omega(p^2-1)$ */

11. END FOR /* End Partial Solution Phase */

12. FOR $m := 1$ TO $(p-1)$ DO /* Begin Solution Update Phase */

13. $x_{i+j} := A[i+j,i]\ x_{i-1} + x_{i+j}$;

/* $i = 1+mp+\omega(p^2-1)$, $j = (p-m+k-1)\ mod\ p$ */

14. END FOR /* End Solution Update Phase */

15. END FOR

16. END PROCEDURE

Our model assumptions imply that we must consider interprocessor communication in addition to operational count when determining the number of steps required by the algorithm. Therefore, we must examine the computations and interprocessor transfers performed in loops 5, 9 and 12. Each iteration of loop 5 requires an interprocessor transfer of $A[i+j-1,j]$ followed by a single multiplication. Thus, loop 5 requires $2(p-1)$ steps. Loop 9 is iterated $p-1$ times, each iteration requiring an interprocessor transfer of x_{i+j-1} followed by a multiplication and addition. Thus, loop 9 requires $3(p-1)$ steps. Loop 12 is also iterated $p-1$ times, each iteration requiring a data broadcast of x_{i-1} followed by a multiplication and addition. Thus, loop 12 also requires $3(p-1)$ steps. The total number of steps required to solve each reduced recurrence $R<p^2,1>$ using p processors is therefore $8(p-1)$, and hence, the resulting theorem follows.

Theorem 2: Given N and p such that $\frac{N-1}{p^2-1}$ is an integer, the number of steps required to solve a linear recurrence system $R<N,1>$ using a SIMD parallel computer with p processors is $\frac{8(N-1)}{p+1}$.

Among the existing SIMD algorithms to solve $R<N,1>$, the SIMD algorithm presented by Gajski [GAJ81] can be most efficiently mapped to a unidirectional ring network with broadcasting capability. Based upon the assumptions made in this section, when $\frac{N}{p^2}$ is an integer the number of steps required by Gajski's approach to solve $R<N,1>$ is $\frac{N}{p^2}(8p-5)-3$, and therefore when $N > p^2$ our approach is more efficient than Gajski's when implemented upon a ring network capable of broadcasting.

Finally, we make the observation that the algorithm does not require all of the inputs a_i and b_i in order for the processing to begin. Specifically, the algorithm requires p^2-1 a_i and p^2 b_i for every $5(p-1)$ computational steps, corresponding to solving each $R<p^2,1>$ in sequence. Similarly, the outputs x_i are produced in blocks of p^2-1 at a time. This suggests that I/O could be overlapped with the computation, providing a balance between I/O and processing loads, and therefore the deletion of assumption A4 has a negligible effect if one assumes that I/O and processing can be done concurrently.

IV. THE MIMD ALGORITHM

In this section we again consider the simplistic model of computation given in Section II with the exception that we no longer require all processors to execute the same operation at each step; that is, we now consider a MIMD implementation in which we neglect the issues of data organization and alignment as well as communication overhead.

The MIMD approach for solving $R<N,1>$ is based upon the observation that only $p-1$ processors are needed at each iteration of the coefficient computation phase. Assuming $\frac{N-1}{p^2-1}$ to be an integer, the total number of multipli-

cations required to compute all necessary coefficients is $\frac{(N-1)(p-1)}{p+1}$, $p-1$ of which may be performed concurrently at each step. Therefore, all of the required coefficients can be computed in $\frac{N-1}{p+1}$ steps using $p-1$ processors. This leaves one processor free for $(N-1)/(p+1)$ steps, allowing us to expand the size of the recurrence by at most $n_0 = \left\lfloor \frac{N-1}{2(p+1)} \right\rfloor$ and use the free processor to concurrently solve the reduced system $R<n_0,1>$. Thus, using an MIMD approach we can solve the entire system $R<N+n_0,1>$ in $\frac{5(N-1)}{p+1}$ steps.

Given a recurrence system of size N and the number of processors p the following lemma expresses n_0 in terms of N and p only.

Lemma 1: Given N and p, $n_0 = \left\lfloor \frac{N-(p+2)}{2p+3} \right\rfloor$.

Given N and p, our MIMD approach to solving $R<N,1>$ consists of partitioning $R<N,1>$ into a sequence of $\left\lceil \frac{N-n_0-1}{p^2-1} \right\rceil + 1$ reduced recurrences. The first recurrence is of size n_0+1 and all others are of size p^2, except for the last recurrence that may be of size less than p^2. The coefficient computation phase of the algorithm uses $p-1$ processors to compute all needed coefficients for all of the reduced systems. Concurrent with this computation, the free processor computes the solution to $R<n_0+1,1>$. Each subsequent $R<n,1>$ is then solved in the same manner as in the SIMD algorithm by executing a partial solution phase followed by a solution update phase. The complete solution is obtained after solving all $\left\lceil \frac{N-n_0-1}{p^2-1} \right\rceil + 1$ reduced recurrences.

We now present the MIMD algorithm to solve $R<N,1>$ when $\frac{N-n_0-1}{p^2-1}$ is an integer. As in the SIMD case, it is not difficult to modify the algorithm if

the above assumption is not satisfied by simply terminating the computation at the point when the last x_i is updated. For ω in $[0,1,...,(N-n_0-1)/(p^2-1)]$, we now define the index sets U_ω and V_ω as

$$U_\omega = \{1 + n_0 + mp + \omega(p^2-1) : m = 1,2,...,p-1\}$$

and

$$V_\omega = \{1 + n_0 + mp + \omega(p^2-1) : m = 0,1,...,p-1\}.$$

```
1. PROCEDURE R(N,p,a,b)
2.   x_1 := b_1
3.   FOR i := 2 to n_0+1 DO                          /* Solve R<n_0,1> */
4.     x_i = a_i x_{i-1} + b_i
5.   END FOR                                         /* End R<n_0,1> Solution */
6.   FOR ω := 0 TO (N-n_0-1)/(p^2-1) - 1 DO          /* Begin Coefficient Computation Phase */
7.     FORALL i ε U_ω DO IN PARALLEL
8.       A[i,i] := a_i;
9.     END FORALL
10.    FOR i := 1 TO (p-1) DO
11.      FORALL j ε U_ω DO IN PARALLEL
12.        A[i+j,j] := a_{i+j} A[i+j-1,j];
13.      END FORALL
14.    END FOR
15.  END FOR                                         /* End Coefficient Computation Phase */
16.  FOR ω := 0 TO (N-n_0-1)/(p^2-1) - 1 DO          /* Solve each R<n,1> */
17.    FORALL i ε U_ω DO IN PARALLEL                 /* Begin Partial Solution Phase */
18.      x_i := b_i;
19.    END FORALL
```

```
20.    FOR i := 1 TO (p-1) DO
21.      FORALL j ∈ V_ω DO IN PARALLEL
22.        x_{i+j} := a_{i+j} x_{i+j-1} + b_{i+j};
23.      END FORALL
24.    END FOR                                  /* End Partial Solution Phase */
25.    FOR i ∈ U_ω DO                           /* Begin Solution Update Phase */
26.      FORALL j := 0 TO (p-1) DO IN PARALLEL
27.        x_{i+j} := A[i+j,i] x_{i-1} + x_{i+j};
28.      END FORALL
29.    END FOR
30.  END FOR                                    /* End Solution Update Phase */
31. END PROCEDURE
```

Note that (i) the coefficient computation phase of the SIMD algorithm has been modified so as to compute the necessary coefficients for all $R<n,1>$ before the first reduced recurrence is solved in parallel; and (ii) the processor that is idle during the SIMD coefficient computation phase is now used to concurrently compute the solution to $R<n_0+1,1>$. An example illustrating the computations performed by the MIMD algorithm is given in Figure 5 for the case $N = 19$ and $p = 3$.

Based upon the MIMD model considered in this section, we conclude that the time required by the MIMD algorithm is determined by the computational operations performed in loops 3, 6 and 16. Loops 3 and 6 are executed concurrently, using 1 and $p-1$ processors, respectively. Loop 6 requires $\frac{N-n_0-1}{p+1}$ steps, and the quantity n_0 has been defined so that loop 3 requires at most the same number of steps as loop 6. All p processors are used in exe-

cuting loop 16, and thus loop 16 requires $\frac{4(N-n_0-1)}{p+1}$ steps. The number of steps required by the MIMD algorithm is therefore $\frac{5(N-n_0-1)}{p+1}$ steps. Thus, the resulting theorem follows.

Theorem 3: Given N and p such that $N \geq p^2 + \frac{1}{2}p - 1$, the number of steps required to solve a linear recurrence system $R\langle N,1\rangle$ using a MIMD parallel computer with p processors is at most $\left\lceil \frac{N-\frac{1}{2}}{(p+3/2)(p-1)} \right\rceil 5(p-1)$.

When $N = p^2 + \frac{1}{2}p - 1$, our approach reduces to the MIMD algorithm presented in [GAJ81] in which the number of steps required to solve $R\langle N,1\rangle$ is at most $\left\lceil \frac{N-1}{p^2+\frac{1}{2}p-2} \right\rceil 5(p-1)$. When $N > p^2 + \frac{1}{2}p - 1$, our MIMD approach differs from [GAJ81] by organizing a single coefficient computation phase to compute the necessary coefficients for all $R\langle n,1\rangle$ before the first reduced recurrence is solved in parallel, rather than including a coefficient computation phase as part of solving each reduced recurrence.

Finally, we note that, like the SIMD algorithm, the MIMD algorithm can also be directly mapped to a p processor unidirectional ring network with broadcasting capability. Figure 6 illustrates such a processor assignment and the corresponding communication requirements for the case $N = 19$ and $p = 3$.

V. CONCLUSIONS

The algorithm presented in this paper exploits the fact that for a fixed number of processors p, the parallel approach presented in [MEY85] to solve $R\langle N,1\rangle$ attains maximum speedup $\frac{2}{5}(p+1)$ when $N = p^2$. When $N \geq p^2$, the structure of $R\langle N,1\rangle$ allows the solution to be obtained by sequentially solving a series of reduced recurrences, each of size p^2, except for the last

recurrence system which may be of size less than p^2. As a result, we are able to improve upon existing approaches for solving $R<N,1>$ whenever $N > p^2$.

REFERENCES

[BRE74] Brent, R.P., The Parallel Evaluation of General Arithmetic Expressions, *JACM*, Vol. 21, No.2, April 1974, pp. 201-206.

[CAR84] Carlson, D.A., and Sugla, B., Time and Processor Efficient Parallel Algorithms for Recurrence Equations and Related Problems, *Proceedings of the 1984 International Conference on Parallel Processing*, August 21-24, 1984, pp. 310-314.

[CHE75] Chen, S.C. and Kuck, D.J., Time and Parallel Processor Bounds for Linear Recurrence Systems, *IEEE Trans. Computers.*, Vol. C-24, No.7, July 1975, pp. 701-717.

[CHE78] Chen, S.C., Kuck, D.J. and Sameh, A.H., Practical Parallel Band Triangular System Solvers, *ACM Trans. on Mathematical Software*, Vol. 4, No. 3, September 1978, pp. 270-277.

[GAJ81] Gajski, D. J., An Algorithm for Solving Linear Recurrence Systems on Parallel and Pipelined Machines, *IEEE Trans. Computers.*, Vol. C-30, No. 3, March 1981, pp. 190-206.

[KOG73] Kogge, P.M., and Stone, H.S., A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations, *IEEE Trans. Computers.*, Vol. C-22, No. 8, August 1973, pp. 786-793.

[KRU85] Kruskal, C.P., Rudolph, L., and Snir, M., The Power of Parallel Prefix, *IEEE Trans. Computers.*, Vol. C-34, No. 10, October 1985, pp. 965-968.

[KUC76] Kuck, D. J., Parallel Processing of Ordinary Programs, *Advances in Computers.*, Vol. 15, Academic Press, New York, 1976, pp. 119-179.

[MEY85] Meyer, G.G.L., and Podrazik, L.J., A Matrix Factorization Approach to the Parallel Solution of First-Order Linear Recurrences, *Proceedings of the 23-rd Annual Allerton Conference on Communication, Control and Computing,* October 2-4, 1985, pp. 243-250.

[SAM77] Sameh, A.H., and Brent, R.P., Solving Triangular Systems on a Parallel Computer, *SIAM J. Numer. Anal.,* Vol. 14, No. 6, December 1977, pp. 1101-1113.

[SNI86] Snir, M., Depth-Size Trade-offs for Parallel Prefix Computation, *Journal of Algorithms,* Vol. 7, No. 2, June 1986, pp. 185-201.

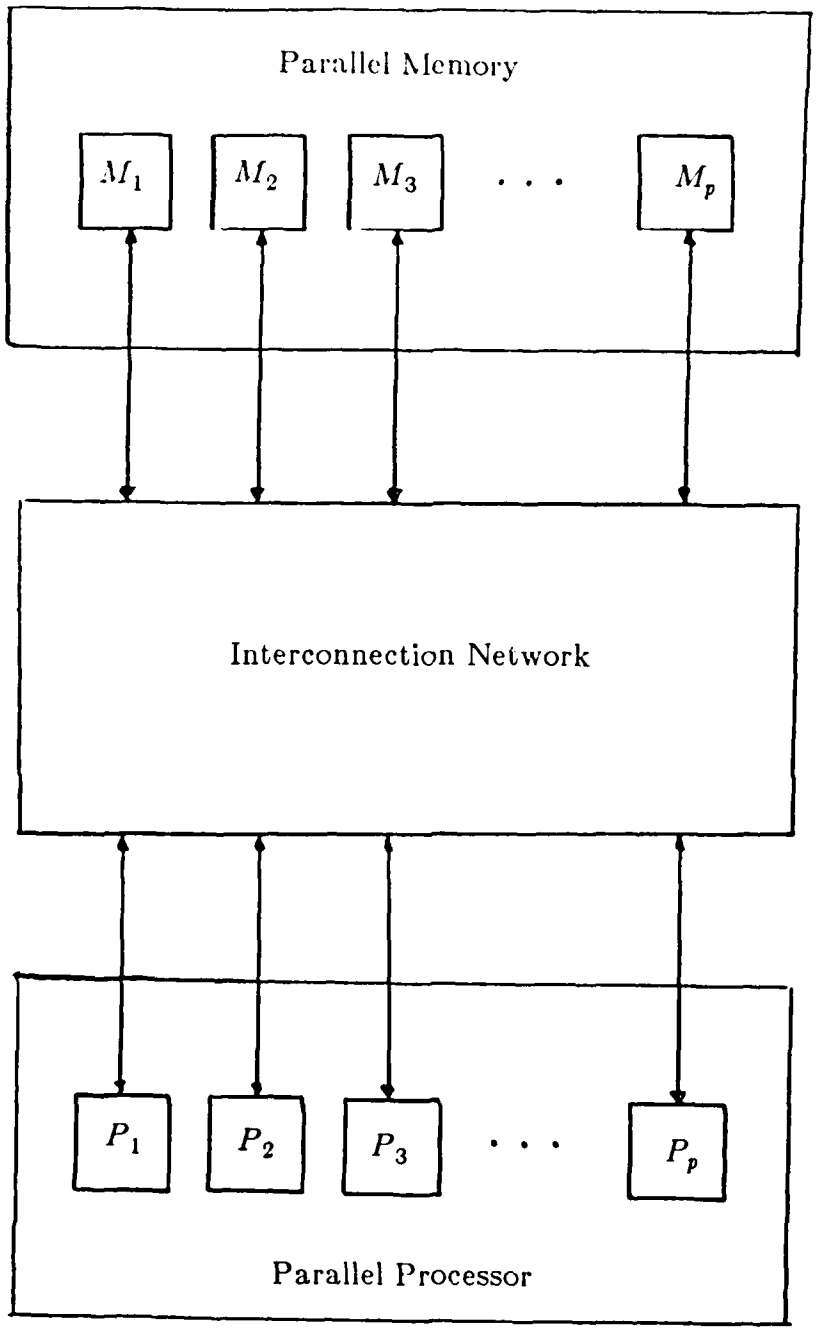


Figure 1. The Abstract Parallel Computational Model

Computational Phase	Time Step	Parallel Computations		
Coefficient Computation Phase		---	$A[4,4] = a_4$	$A[7,7] = a_7$
	1	---	$A[5,4] = a_5A[4,4]$	$A[8,7] = a_8A[7,7]$
	2	---	$A[6,4] = a_6A[5,4]$	$A[9,7] = a_9A[8,7]$
Partial Solution Phase		$x_1 = b_1$	$x_4 = b_4$	$x_7 = b_7$
	3,4	$x_2 = a_2x_1 + b_2$	$x_5 = a_5x_4 + b_5$	$x_8 = a_8x_7 + b_8$
	5,6	$x_3 = a_3x_2 + b_3$	$x_6 = a_6x_5 + b_6$	$x_9 = a_9x_8 + b_9$
Solution Update Phase	7,8	$x_4 = A[4,4]x_3 + x_4$	$x_5 = A[5,4]x_3 + x_5$	$x_6 = A[6,4]x_3 + x_6$
	9,10	$x_7 = A[7,7]x_6 + x_7$	$x_8 = A[8,7]x_6 + x_8$	$x_9 = A[9,7]x_6 + x_9$
Coefficient Computation Phase		---	$A[12,12] = a_{12}$	$A[15,15] = a_{15}$
	11	---	$A[13,12] = a_{13}A[12,12]$	$A[16,15] = a_{16}A[15,15]$
	12	---	$A[14,12] = a_{14}A[13,12]$	$A[17,15] = a_{17}A[16,15]$
Partial Solution Phase		$x_9 = x_9$	$x_{12} = b_{12}$	$x_{15} = b_{15}$
	13,14	$x_{10} = a_{10}x_9 + b_{10}$	$x_{13} = a_{13}x_{12} + b_{13}$	$x_{16} = a_{16}x_{15} + b_{16}$
	15,16	$x_{11} = a_{11}x_{10} + b_{11}$	$x_{14} = a_{14}x_{13} + b_{14}$	$x_{17} = a_{17}x_{16} + b_{17}$
Solution Update Phase	17,18	$x_{12} = A[12,12]x_{11} + x_{12}$	$x_{13} = A[13,12]x_{11} + x_{13}$	$x_{14} = A[14,12]x_{11} + x_{14}$
	19,20	$x_{15} = A[15,15]x_{14} + x_{15}$	$x_{16} = A[16,15]x_{14} + x_{16}$	$x_{17} = A[17,15]x_{14} + x_{17}$

FIGURE 2 : Solution of $R<17,1>$ for 3 processor SIMD computer

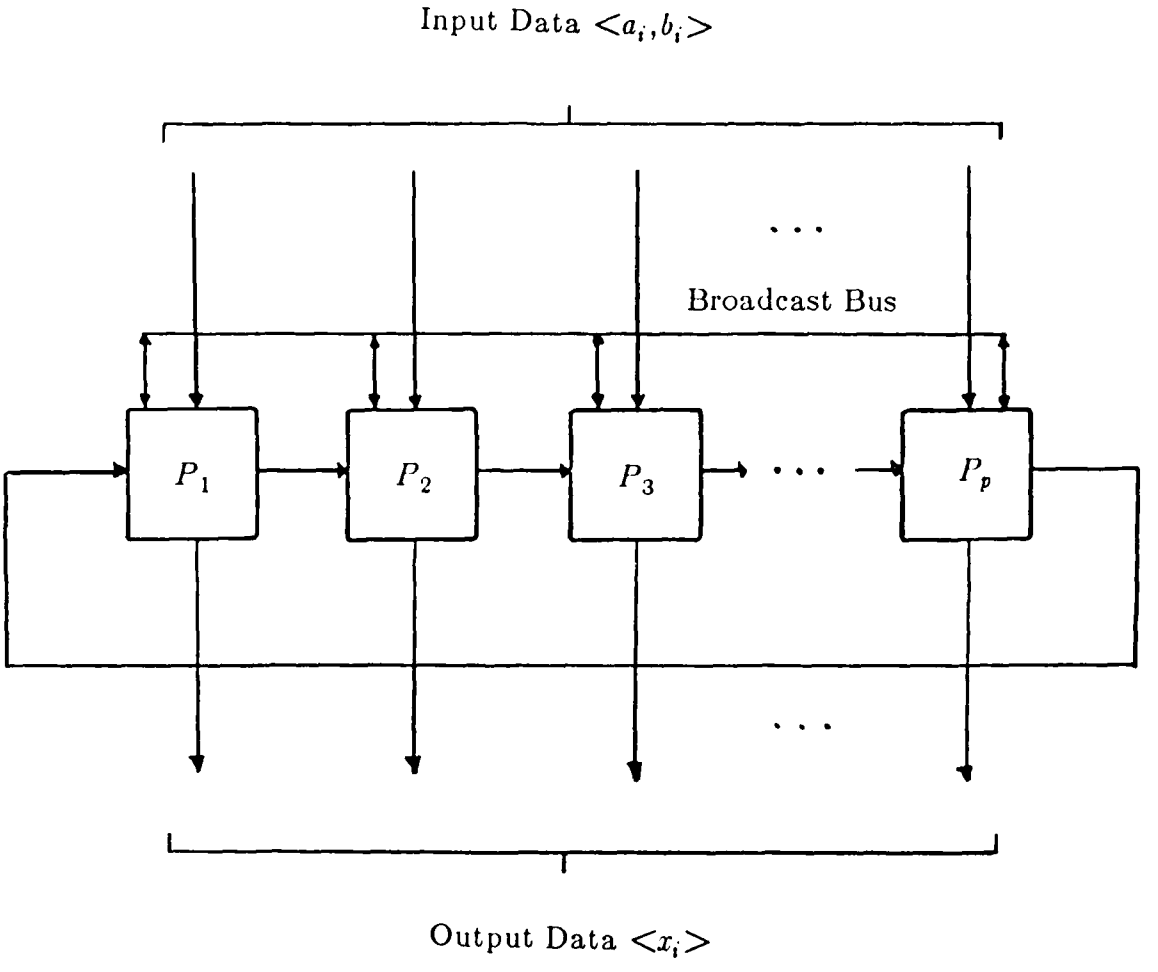


FIGURE 3. The Practical Parallel Computational Model

Computational Phase	Time Step	Parallel Computations		
		PROCESSOR #1	PROCESSOR #2	PROCESSOR #3
Processor Resident Data		a_i, i = 6,8,14,16 b_i, i = 1,6,8,14,16	a_i, i = 2,4,9,10,12,17 b_i, i = 2,4,9,10,12,17	a_i, i = 3,5,7,11,13,15 b_i, i = 3,5,7,11,13,15
Coefficient Computation Phase	1 2,3 4	--- A[8,7] = a_8A[7,7] → A[6,4] = a_6A[5,4]	A[4,4] = a_4 → --- A[9,7] = a_9A[8,7]	A[7,7] = a_7 → A[5,4] = a_5A[4,4] → ---
Partial Solution Phase	5 6,7,8 9,10,11	$x_1 = b_1$ → $x_8 = a_8x_7 + b_8$ → $x_6 = a_6x_5 + b_6$	$x_4 = b_4$ → $x_2 = a_2x_1 + b_2$ → $x_9 = a_9x_8 + b_9$	$x_7 = b_7$ → $x_5 = a_5x_4 + b_5$ → $x_3 = a_3x_2 + b_3$ ⇉
Solution Update Phase	12,13,14 15,16	x_6 = A[6,4]$x_3 + x_6$ ⇉ x_8 = A[8,7]$x_6 + x_8$	x_4 = A[4,4]$x_3 + x_4$ x_9 = A[9,7]$x_6 + x_9$	x_5 = A[5,4]$x_3 + x_5$ x_7 = A[7,7]$x_6 + x_7$
Coefficient Computation Phase	17 18,19 20	--- A[16,15] = a_{16}A[15,15] → A[14,12] = a_{14}A[13,12]	A[12,12] = a_{12} → x_9 → A[17,15] = a_{17}A[16,15]	A[15,15] = a_{15} → A[13,12] = a_{13}A[12,12] → x_9 →
Partial Solution Phase	21 22,23,24 25,26,27	x_9 → $x_{16} = a_{16}x_{15} + b_{16}$ → $x_{14} = a_{14}x_{13} + b_{14}$	$x_{12} = b_{12}$ → $x_{10} = a_{10}x_9 + b_{10}$ → $x_{17} = a_{17}x_{16} + b_{17}$	$x_{15} = b_{15}$ → $x_{13} = a_{13}x_{12} + b_{13}$ → $x_{11} = a_{11}x_{10} + b_{11}$ ⇉
Solution Update Phase	28,29,30 31,32	x_{14} = A[14,12]$x_{11} + x_{14}$ ⇉ x_{16} = A[16,15]$x_{14} + x_{16}$	x_{12} = A[12,12]$x_{11} + x_{12}$ x_{17} = A[17,15]$x_{14} + x_{17}$	x_{13} = A[13,12]$x_{11} + x_{13}$ x_{15} = A[15,15]$x_{14} + x_{15}$
Processor Resident Results		x_i, i = 1,6,8,14,16	x_i, i = 2,4,9,10,12,17	x_i, i = 3,5,7,11,13,15

Notation: → indicates that the data evaluated at the current step is transferred to the processor on the right.

⇉ indicates that the data evaluated at the current step is broadcast to all processors.

FIGURE 4 : Processor Assignment for SIMD Solution of $R_{<17,1>}$ with $p = 3$

Computational Phase	Time Step	Parallel Computations		
Coefficient Computation + $R<n_0+1,1>$ Solution Phase		$x_1 = b_1$	A[6,6] = a_6	A[9,9] = a_9
	1	$x_2 = a_2x_1$	A[7,6] = a_7A[6,6]	A[10,9] = a_{10}A[9,9]
	2	$x_2 = x_2 + b_2$	A[8,6] = a_8A[7,6]	A[11,9] = a_{11}A[10,9]
		---	A[14,14] = a_{14}	A[17,17] = a_{17}
	3	$x_3 = a_3x_2$	A[15,14] = a_{15}A[14,14]	A[18,17] = a_{18}A[17,17]
	4	$x_3 = x_3 + b_3$	A[16,14] = a_{16}A[15,14]	A[19,17] = a_{19}A[18,17]
Partial Solution Phase		$x_3 = x_3$	$x_6 = b_6$	$x_9 = b_9$
	5,6	$x_4 = a_4x_3 + b_4$	$x_7 = a_7x_6 + b_7$	$x_{10} = a_{10}x_9 + b_{10}$
	7,8	$x_5 = a_5x_4 + b_5$	$x_8 = a_8x_7 + b_8$	$x_{11} = a_{11}x_{10} + b_{11}$
Solution Update Phase	9,10	x_6 = A[6,6]$x_5 + x_6$	x_7 = A[7,6]$x_5 + x_7$	x_8 = A[8,6]$x_5 + x_8$
	11,12	x_9 = A[9,9]$x_8 + x_9$	x_{10} = A[10,9]$x_8 + x_{10}$	x_{11} = A[11,9]$x_8 + x_{11}$
Partial Solution Phase		$x_{11} = x_{11}$	$x_{14} = b_{14}$	$x_{17} = b_{17}$
	13,14	$x_{12} = a_{12}x_{11} + b_{12}$	$x_{15} = a_{15}x_{14} + b_{15}$	$x_{18} = a_{18}x_{17} + b_{18}$
	15,16	$x_{13} = a_{13}x_{12} + b_{13}$	$x_{16} = a_{16}x_{15} + b_{16}$	$x_{19} = a_{19}x_{18} + b_{19}$
Solution Update Phase	17,18	x_{14} = A[14,14]$x_{13} + x_{14}$	x_{15} = A[15,14]$x_{13} + x_{15}$	x_{16} = A[16,14]$x_{13} + x_{16}$
	19,20	x_{17} = A[17,17]$x_{16} + x_{17}$	x_{18} = A[18,17]$x_{16} + x_{18}$	x_{19} = A[19,17]$x_{16} + x_{19}$

FIGURE 5 : Solution of $R<19,1>$ for 3 processor MIMD computer

Computational Phase	Time Step	Parallel Computations		
		PROCESSOR #1	PROCESSOR #2	PROCESSOR #3
Processor Resident Data		a_i, i = 8,10,16,18 b_i, i = 1,8,10,16,18	a_i, i = 2,3,4,6,11,12,14,19 b_i, i = 4,6,11,12,14,19	a_i, i = 5,7,9,13,15,17 b_i, i = 2,3,5,7,9,13,15,17
Coefficient Computation + $R<n_0+1,1>$ Solution Phase	1 2,3 4,5 6 7,8 9,10	$x_1 = b_1 \rightarrow$ $A[10,9] = a_{10}A[9,9] \rightarrow$ $A[8,6] = a_8A[7,6]$ $x_2 \rightarrow$ $A[18,17] = a_{18}A[17,17] \rightarrow$ $A[16,14] = a_{16}A[15,14]$	$A[6,6] = a_6 \rightarrow$ $x_2 = a_2x_1 \rightarrow$ $A[11,9] = a_{11}A[10,9]$ $A[14,14] = a_{14} \rightarrow$ $x_3 = a_3x_2 \rightarrow$ $A[19,17] = a_{19}A[18,17]$	$A[9,9] = a_9 \rightarrow$ $A[7,6] = a_7A[6,6] \rightarrow$ $x_2 = x_2 + b_2 \rightarrow$ $A[17,17] = a_{17} \rightarrow$ $A[15,14] = a_{15}A[14,14] \rightarrow$ $x_3 = x_3 + b_3 \rightarrow$
Partial Solution Phase	11 12,13,14 15,16,17	$x_3 \rightarrow$ $x_{10} = a_{10}x_9 + b_{10} \rightarrow$ $x_8 = a_8x_7 + b_8$	$x_6 = b_6 \rightarrow$ $x_4 = a_4x_3 + b_4 \rightarrow$ $x_{11} = a_{11}x_{10} + b_{11}$	$x_9 = b_9 \rightarrow$ $x_7 = a_7x_6 + b_7 \rightarrow$ $x_5 = a_5x_4 + b_5 \Rightarrow$
Solution Update Phase	18,19,20 21,22	$x_8 = A[8,6]x_5 + x_8 \Rightarrow$ $x_{10} = A[10,9]x_8 + x_{10}$	$x_6 = A[6,6]x_5 + x_6$ $x_{11} = A[11,9]x_8 + x_{11}$	$x_7 = A[7,6]x_5 + x_7$ $x_9 = A[9,9]x_8 + x_9$
Partial Solution Phase	23 24,25,26 27,28,29	--- $x_{18} = a_{18}x_{17} + b_{18} \rightarrow$ $x_{16} = a_{16}x_{15} + b_{16}$	$x_{14} = b_{14} \rightarrow$ $x_{12} = a_{12}x_{11} + b_{12} \rightarrow$ $x_{19} = a_{19}x_{18} + b_{19}$	$x_{17} = b_{17} \rightarrow$ $x_{15} = a_{15}x_{14} + b_{15} \rightarrow$ $x_{13} = a_{13}x_{12} + b_{13} \Rightarrow$
Solution Update Phase	30,31,32 33,34	$x_{16} = A[16,14]x_{13} + x_{16} \Rightarrow$ $x_{18} = A[18,17]x_{16} + x_{18}$	$x_{14} = A[14,14]x_{13} + x_{14}$ $x_{19} = A[19,17]x_{16} + x_{19}$	$x_{15} = A[15,14]x_{13} + x_{15}$ $x_{17} = A[17,17]x_{16} + x_{17}$
Processor Resident Results		x_i, i = 1,8,10,16,18	x_i, i = 4,6,11,12,14,19	x_i, i = 2,3,5,7,9,13,15,17

Notation: $\rightarrow$ indicates that the data evaluated at the current step is transferred to the processor on the right.

$\Rightarrow$ indicates that the data evaluated at the current step is broadcast to all processors.

FIGURE 6 : Processor Assignment for MIMD Solution of $R<19,1>$ with $p = 3$

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER: JHU/EECS-86/07	2. GOVT ACCESSION NO.: AD-A171686
	3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): A PARALLEL FIRST-ORDER LINEAR RECURRENCE SOLVER	5. TYPE OF REPORT & PERIOD COVERED: TECHNICAL
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): GERARD G.L. MEYER AND LOUIS J. PODRAZIK	8. CONTRACT OR GRANT NUMBER(s): AFOSR-85-0097
9. PERFORMING ORGANIZATION NAME AND ADDRESS: THE JOHNS HOPKINS UNIVERSITY, BALTIMORE, MARYLAND 21218	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS: AIR FORCE OFFICE OF SCIENTIFIC RESEARCH /NM, BOLLING AFB, DC 20332-6448	12. REPORT DATE: SEPTEMBER 1986
	13. NUMBER OF PAGES: 27
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

LINEAR RECURRENCE; ALGORITHM COMPLEXITY; PARALLEL EVALUATION; PARALLEL PROCESSORS; PARALLEL PREFIX

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this paper we present a parallel procedure for the solution of first-order linear recurrence systems of size N when the number of processors p is small in relation to N. We show that when $1 < p^2 \leq N$, a first-order linear recurrence system of size N can be solved in $5(N-1)/(p+1)$ steps on a p processor SIMD machine and less than $5N/(p+\frac{3}{2})$ steps on a p processor MIMD machine.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

END

10-86

DTIC