PARTITIONED ALGORITHMS AND VLSI STRUCTURES FOR LARGE-SCALE MATRIX COMPUTATIONS

Kai Hwang
School of Electrical Engineering
PURDUE UNIVERSITY
West Lafayette, Indiana USA

Yen-Heng Cheng
Department of Computer Engineering and Science
TSING-HUA UNIVERSITY
Peking, China

ABSTRACT:

VLSI modular arithmetic structures and new partitioned matrix algorithms are developed in this paper to perform hardware matrix computations in solving large-scale linear system of equations. Gaussian elimination and inversion of triangular matrices are shown systematically partitionable. All the partitioned algorithms being developed can achieve linear computation time $O(n)$, where n is the order of the linear system. The partitioned matrix computations are feasible for modular VLSI implementation with constrained I/O terminals. Performance analysis and design tradeoffs of the partitioned VLSI arithmetic structures are also provided.

1. INTRODUCTION

Large-scale matrix computations are needed in connection with solving high-order Linear System of Equations (LSE), $\underline{A} \cdot \underline{x} = \underline{b}$, in many important application areas, such as structural analysis, linear programming, seismic signal processing, image understanding, numerical weather forecasting, and artificial intelligence, etc. At present, most large scale LSEs are solved on SIMD array processors or on pipelined vector computers using predeveloped software packages [7]. Fast matrix algorithms for solving LSEs on parallel computers have been studied by Csanky [1], Kant and Kimura [8], Sameh and Kuck [13], and Wing and Huang [17] among many other authors.

The rapid advances in Very Large-Scale Integration (VLSI) microelectronic technology have created new architectural horizons to implement large-scale vector/matrix computation algorithms directly in hardware. Mead and Conway [11] have studied the cost and performance of highly concurrent VLSI computing structures. Kung and Leiserson [9], Kung [10] have suggested the systolic arrays for matrix and vector computations. Foster and Kung [3] and Swartzlander [15] discussed the design problems of special-purpose VLSI chips for pattern matching or for signal processing. The pioneering systolic arrays have opened the research area of VLSI computing structures. Recently, Horowitz [2], Hwang and Cheng [6], and Preparata and Vuillemin [12] studied globally structured VLSI arrays for matrix computations. In the theoretical domain, Savage [14] and Thompson [16] developed the complexity theory and computation models for VLSI computing structures.

*This research was supported by the NSF under grant MCS78-18096A02.

Due to the limitations on projected VLSI chip density and chip packaging technology, we can only expect monolithic VLSI computing devices with regularly structured functions and limited I/O terminals. Modular approach to develop VLSI devices is amenable from the viewpoints of feasibility and applicability. In this paper, we developed partitioned iterative algorithms and VLSI modular structures for the following matrix computations.

- Partitioned L-U Decomposition of Matrices.
- Partitioned Inversion of Triangular Matrices.
- Partitioned Multiplication of matrices.
- Partitioned solution of triangularized LSEs.

We reserve the parameter, n, for the order of a given dense matrix $\underline{A}$ and the parameter, m, as the size of available VLSI computing modules. Note that $m \ll n$ in practical applications. We shall consider those cases in which $k = n/m$ is an integer. All analytical results on complexity and performance are expressed in terms of these three parameters. Tradeoff studies between computation times and hardware complexities are also provided.

2. MATRIX COMPUTATIONS IN LINEAR SYSTEMS

An LSE of order n is characterized by a pair $(\underline{A}, \underline{b})$, where $\underline{A} = (a_{ij})$ is an $n \times n$ matrix, $\underline{b} = (b_1, b_2, \cdots, b_n)^T$ is a column vector. The problem of solving an LSE is to find a vector $\underline{x} = (x_1, x_2, \cdots, x_n)^T$ which satisfies $\underline{A} \cdot \underline{x} = \underline{b}$. The solution vector $\underline{x}$ is unique, if and only if $\underline{A}$ is nonsingular. We shall consider only strongly nonsingular LSE's, in which all the diagonal submatrices of $\underline{A}$ are nonsingular. This strong nonsingularity is necessary and sufficient in three of the partitioned matrix algorithms being presented, except the algorithm for partitioned matrix multiplication.

Using the Gaussian elimination method, one can systematically decompose $\underline{A}$ into two triangular matrices $\underline{L}$ and $\underline{U}$ such that $\underline{A} = \underline{L} \cdot \underline{U}$, where $\underline{L} = (\ell_{ij})$ is a lower triangular matrix with all diagonal elements equal to 1, and $\underline{U} = (u_{ij})$ is an upper triangular matrix with nonzero diagonal elements. Such an L-U decomposition is unique, if and only if $\underline{A}$ is strongly nonsingular. Let us define the following recursive computations.

$$a_{ij}^{(1)} = a_{ij} \text{ for all } 1 \le i,j \le n$$

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - (a_{ik}^{(k)}/a_{kk}^{(k)}) \cdot a_{kj}^{(k)} \qquad (1)$$

$$\text{for } k = 1,2,\ldots,n-1$$

$$\text{and } i,j = k,k+1,\ldots,n$$

The entries of $\underline{L}$ and $\underline{U}$ are computed by

$$\ell_{ik} = a_{ik}^{(k)}/a_{kk}^{(k)} \text{ for all } i > k$$

$$u_{kj} = a_{kj}^{(k)} \text{ for all } k \le j \qquad (2)$$

We shall show in section 4 how to partition the above recursive computations in order to generate the decomposed matrices $\underline{L} = (\ell_{ik})$ and $\underline{U} = (u_{kj})$ for arbitrary order n using fixed size VLSI modules. In this paper, Gaussian elimination with natural ordering is presented for clarity purpose. One can always use the pivoting technique to improve accuracy, as suggested in our earlier paper [6].

The sequence of Gaussian elimination operations will transform the dense system $\underline{A} \cdot \underline{x} = \underline{b}$ into an equivalent triangular LSE, characterized by $\underline{U} \cdot \underline{x} = \underline{L}^{-1} \cdot \underline{b} = \underline{d}$. With this triangularized system, one can compute the solution vector $\underline{x}$ by $\underline{x} = \underline{U}^{-1} \cdot (\underline{L}^{-1} \cdot \underline{b}) = \underline{U}^{-1} \cdot \underline{d}$. The inverse matrices $\underline{U}^{-1}$ and $\underline{L}^{-1}$ always exist, because $\underline{U}$ and $\underline{L}$ are nonsingular.

Consider a nonsingular upper triangular matrix $\underline{U}$ of order n. The inverse matrix $\underline{V} = (v_{ij}) = \underline{U}^{-1}$ is also an upper triangular matrix. The following recursive computations are needed to generate the entries of the inverse matrix $\underline{V}$.

$$v_{kk} = u_{kk}^{-1} \text{ for all } k = 1,2,\ldots,n$$

$$v_{ij} = -\left(\sum_{k=i+1}^{j} u_{ik} \cdot v_{kj}\right) \cdot u_{ii}^{-1} \text{ for all } j > i \qquad (3)$$

In Section 5, we shall partition the above procedures to enable modular generation of the (v_{ij}) entries for all $1 \le i,j \le n$.

In fact, once the entries (u_{ij}) of the triangular matrix $\underline{U}$ and the elements (d_i) of the transformed vector $\underline{d}$ are determined, the solution vector $\underline{x}$ can be recursively generated by

$$x_n = d_n \cdot u_{nn}^{-1}$$

$$x_i = \left(d_i - \sum_{j=i+1}^{n} u_{ij} \cdot x_j\right) \cdot u_{ii}^{-1} \qquad (4)$$

$$\text{for } i = n-1,\ldots,2,1$$

In Section 6, we shall show a partitioned procedure to generate sections of the solution vector $\underline{x} = (x_n, x_{n-1}, \ldots, x_1)^T$ sequentially by VLSI modules.

3. BASIC VLSI ARITHMETIC MODULES

Before presenting the partitioned algorithms for iterative L-U decomposition and matrix inversion and multiplication, we describe below several primitive VLSI computing modules. These modules will be used as building blocks in implementing those partitioned matrix algorithms in subsequent sections.

The Type-I VLSI computing modules are designed for local L-U decomposition of any m x m submatrix $\underline{A}_m$ into two triangular matrices $\underline{L}_m$ and $\underline{U}_m$ such that $\underline{A}_m = \underline{L}_m \cdot \underline{U}_m$, where $\underline{L}_m$ is a lower triangular matrix and $\underline{U}_m$ is an upper triangular matrix as shown below for the case of m=3.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{32} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ \ell_{21} & 1 & 0 \\ \ell_{31} & \ell_{32} & 1 \end{bmatrix} \cdot \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix} \qquad (5)$$

Equations 1 and 2 can be used to compute $\underline{L}_m = (\ell_{ij})$ and $\underline{U}_m = (u_{ij})$ for all $1 \le i,j \le m$. In Fig. 1, we show the schematic design of a 4 x 4 Type-I arithmetic module with m = 4. Two types of arithmetic cells are needed in all VLSI module designs. The D cell performs the division of two matrix elements, say a/b, and the M cell performs the accumulative multiplication, say a-b*c. Typical array structures of such cellular arithmetic cells can be found in Hwang [4,5]. Three levels of fast latches are used to control the timing and data flow. The structure is pipelined with feedback lines controlled by multiplexers (MPX), Demultiplexers (DMX), and the latches. The time delay of the Type-I module equals 2m units of time.

The Type-II VLSI array modules are used for the inversion of triangular submatrices of order m. In Fig. 2, we show the schematic design of the Type II VLSI module with size m=4. This module performs the iterative computations specified in Eq. 3. It is also pipelined with a total compute delay of 2m units of time. The thick lines between the arithmetic cells are high-speed latches. Similar triangular arrays were also independently suggested by Preparata and Vuillemin [12] for matrix inversion.

The Type-III VLSI arithmetic module is for the additive matrix multiplications specified below for the case of m = 2 and any integer $r \ge 1$.

$$\begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} - \sum_{k=1}^{r} \begin{bmatrix} b_{11}^{(k)} & b_{12}^{(k)} \\ b_{21}^{(k)} & b_{22}^{(k)} \end{bmatrix} \cdot \begin{bmatrix} c_{11}^{(k)} & c_{12}^{(k)} \\ c_{21}^{(k)} & c_{22}^{(k)} \end{bmatrix} \qquad (6)$$

In Fig. 3, we show the design of a Type-III ar-

ithmetic module for the case of m=2. The accumulative multiplications are implemented with the M cells, which keep performing the additive multiply operations. At time t_1, matrix element a_{ij} enters one Accumulative Multiply Unit (AMU). Starting from time t_2, the feedback loop is established inside each AMU. The output d_{ij} is generated at cycle t_{mr+1}. In other words, the total time delay of a Type-III module is m•r+1, in order to complete the computations specified in Eq.6.

All three types of computing modules will be used in partitioned L-U decomposition. Only Type-II and Type-III modules will be used in partitioned matrix inversion and in the VLSI solution of triangular LSE. The partitioned multiplication of two large matrices requires the use of only Type-III modules. It is assumed that there are large I/O data (latches) in each VLSI module to allow the outputs of a VLSI module be connected directly to the inputs of other VLSI modules to form a synchronous modular network. In other words, all the intermediate results are routed directly from modules to modules without storing back to the main memory.

4. PARTITIONED L-U DECOMPOSITION OF A DENSE MATRIX

Consider the L-U decomposition of an n x n nonsingular dense matrix $\underline{A} = \underline{L} \cdot \underline{U}$. A systolic array of n^2 step-processors can compute the triangular matrices $\underline{L}$ and $\underline{U}$ in 4n units of time [9]. Such a systolic array may require 2w x (2n-1) input/output terminals, where w is the word length of the matrix elements. For large n (say n > 1000) with typical operand length w (say 32 bits per word), it is rather impractical to consider implementing an n x n systolic array on a single VLSI chip. The limitation lies in the projected VLSI density and by the I/O packaging constraints.

The given n x n dense matrix $\underline{A} = (a_{ij})$ and the two decomposed triangular matrices $\underline{L} = (\ell_{ij})$ and $\underline{U} = (u_{ij})$ can be each partitioned into k^2 submatrices of order m x m each. The partitioned operations on submatrices can be best illustrated by an example matrix $\underline{A}$ of order n = 6 using size m=2 modules, and thus the ratio k = 6/2 = 3. There are in total k•(k + 1) = 3 x 4 = 12 nontrivial 2 x 2 submatrices in $\underline{L}$ and $\underline{U}$ such that $\underline{A} = \underline{L} \cdot \underline{U}$. These submatrices must be generated in a specific order to be described below.

$$\underline{A} = \left[\begin{array}{cc|cc|cc} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ \hline a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & a_{36} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} \\ \hline a_{51} & a_{52} & a_{53} & a_{54} & a_{55} & a_{56} \\ a_{61} & a_{62} & a_{63} & a_{64} & a_{65} & a_{66} \end{array}\right] = \left[\begin{array}{ccc} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{array}\right] \qquad (7)$$

$$\underline{L} = \left[\begin{array}{cc|cc|cc} 1 & 0 & 0 & 0 & 0 & 0 \\ \ell_{21} & 1 & 0 & 0 & 0 & 0 \\ \hline \ell_{31} & \ell_{32} & 1 & 0 & 0 & 0 \\ \ell_{41} & \ell_{42} & \ell_{43} & 1 & 0 & 0 \\ \hline \ell_{51} & \ell_{52} & \ell_{53} & \ell_{54} & 1 & 0 \\ \ell_{61} & \ell_{62} & \ell_{63} & \ell_{64} & \ell_{65} & 1 \end{array}\right] = \left[\begin{array}{ccc} L_{11} & 0 & 0 \\ L_{21} & L_{22} & 0 \\ L_{31} & L_{32} & L_{33} \end{array}\right] \qquad (8)$$

$$\underline{U} = \left[\begin{array}{cc|cc|cc} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} & u_{16} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} & u_{26} \\ \hline 0 & 0 & u_{33} & u_{34} & u_{35} & u_{36} \\ 0 & 0 & 0 & u_{44} & u_{45} & u_{46} \\ \hline 0 & 0 & 0 & 0 & u_{55} & u_{56} \\ 0 & 0 & 0 & 0 & 0 & u_{66} \end{array}\right] = \left[\begin{array}{ccc} U_{11} & U_{12} & U_{13} \\ 0 & U_{22} & U_{23} \\ 0 & 0 & U_{33} \end{array}\right] \qquad (9)$$

First of all, we perform the local L-U decomposition on submatrix A_{11} using a Type-I module to generate the first two triangular submatrices L_{11} and U_{11} such that $A_{11} = L_{11} \cdot U_{11}$. Two Type-II VLSI modules are then used to compute the inverse submatrices L_{11}^{-1} and U_{11}^{-1}. We then perform the following matrix multiplications using 2(k-1) Type III modules to generate all the L_{p1} and U_{1q} submatrices in parallel.

$$L_{p1} = A_{p1} \cdot U_{11}^{-1} \quad \text{for } p = 2, 3, \ldots, k$$
$$U_{1q} = L_{11}^{-1} \cdot A_{1q} \quad \text{for } q = 2, 3, \ldots, k \qquad (10)$$

For the example 6 x 6 matrix, submatrices L_{21}, L_{31}, U_{12} and U_{13} are generated using Eq.10.

In each iteration, we need to generate the following intermediate submatrices using Type-III modules for r = min (p,q).

$$\hat{A}_{pq} = A_{pq} - \sum_{s=1}^{r-1} L_{ps} \cdot U_{sq} \qquad (11)$$
$$\text{for } p,q=2,3,\ldots,k.$$

The local L-U decompositions is then performed on $\hat{A}$ in each iteration.

$$L_{qq} \cdot U_{qq} = \hat{A}_{qq} \quad \text{for } q = 2, 3, \ldots, k \qquad (12)$$

The remaining off-diagonal submatrices L_{pq} and U_{pq} are computed by inverting the diagonal submatrices U_{qq} and L_{qq} and then multiplying them by the corresponding intermediate submatrices $\hat{A}_{pq}$ and $\hat{A}_{qp}$ as follows for p = q+1, q+2, ···, k and r = 2, 3, ···, k.

$$L_{pq} = \hat{A}_{pq} \cdot U_{qp}^{-1}$$
$$U_{qp} = L_{qq}^{-1} \cdot \hat{A}_{qp} \qquad (13)$$

For the example 6 x 6 matrix, we first compute the following intermediate matrix $\hat{A}_{22}$ with r = 2. By performing $L_{22} \cdot U_{22} = \hat{A}_{22}$, we obtain the two triangular submatrices L_{22} and U_{22}. Substituting these submatrices, we obtain two additional submatrices $U_{23} = L_{22}^{-1} \cdot \hat{A}_{23}$ and $L_{32} = \hat{A}_{32} \cdot U_{22}^{-1}$. The next intermediate dense matrix can then be calculated $\hat{A}_{33} = A_{33} - (L_{31} \cdot U_{13} + L_{32} \cdot U_{23})$. Performing L-U decomposition on $\hat{A}_{33}$, we obtain the last two submatrices L_{33} and U_{33}.

The above iterative procedures are summarized in Algorithm 1 for partitioned L-U decomposition of any strongly nonsingular dense matrix $\underline{A}$.

ALGORITHM 1

Inputs:

The n x n dense matrix $\underline{A} = (a_{ij})$ partitioned into k^2 m x m submatrices A_{ij} for i,j=1,2,•••,k, where n = k•m.

Outputs:

k•(k+1) submatrices of order m x m each: L_{pq} for $q \leq p = 1, 2, \ldots, k$ and U_{rs} for $s \geq r = 1, 2, \ldots, k$

Procedures:

(1) Decompose A_{11} into L_{11} and U_{11} such that $L_{11} \cdot U_{11} = A_{11}$.

(2) Find the inverse matrices L_{11}^{-1} and U_{11}^{-1}

Compute $L_{p1} = A_{p1} \cdot U_{11}^{-1}$ and $U_{1p} = L_{11}^{-1} \cdot A_{1p}$ for p = 2,3,...,k.

(3) For q = 2 to (k-1) step 1 do

Compute $\hat{A}_{qq} = A_{qq} - \sum_{s=1}^{q-1} L_{qs} \cdot U_{sq}$;

Decompose $\hat{A}_{qq} = L_{qq} \cdot U_{qq}$;

Find inverse matrices L_{qq}^{-1} and U_{qq}^{-1}.

For p = (q+1) to k step 1 do

Compute $\hat{A}_{pq} = A_{pq} - \sum_{s=1}^{r-1} L_{ps} \cdot U_{sq}$

and $\hat{A}_{qp} = A_{qp} - \sum_{s=1}^{r-1} L_{qs} \cdot U_{sp}$

for r = min(p,q);

Compute $L_{pq} = \hat{A}_{pq} \cdot U_{qq}^{-1}$; $U_{qp} = L_{qq}^{-1} \cdot \hat{A}_{qp}$.

Repeat

Repeat

(4) Compute $\hat{A}_{kk} = A_{kk} - \sum_{s=1}^{k-1} L_{ks} \cdot U_{sk}$;

Decompose $\hat{A}_{kk} = L_{kk} \cdot U_{kk}$

5. PARTITIONED MATRIX INVERSION AND MULTIPLICATION

Partitioned algorithms are developed in this section for iterative inversion of an n x n strongly nonsingular triangular matrix using small m x m Type-II and Type-III VLSI array modules. For clarity, we demonstrate the partitioning methods by finding the inverse of an example 6 x 6 upper triangular matrix, $\underline{U} = (u_{ij})$ with 2 x 2 array modules (for n = 6 and m = 2). The inverse matrix $\underline{V} = (v_{ij}) = \underline{U}^{-1}$ is partitioned as follows:

$$\underline{V} = \begin{bmatrix} V_{11} & V_{12} & V_{13} \\ 0 & V_{22} & V_{23} \\ 0 & 0 & V_{33} \end{bmatrix} = \left[\begin{array}{cc|cc|cc} v_{11} & v_{12} & v_{13} & v_{14} & v_{15} & v_{16} \\ 0 & v_{22} & v_{23} & v_{24} & v_{25} & v_{26} \\ \hline 0 & 0 & v_{33} & v_{34} & v_{35} & v_{36} \\ 0 & 0 & 0 & v_{44} & v_{45} & v_{46} \\ \hline 0 & 0 & 0 & 0 & v_{55} & v_{56} \\ 0 & 0 & 0 & 0 & 0 & v_{66} \end{array}\right]$$

Note that $\underline{U} \cdot \underline{V} = \underline{I}$ and $\underline{V} \cdot \underline{U} = \underline{I}$. By equating the corresponding submatrices in the product matrix to the identity matrix $\underline{I}$, we obtain three types of relations between submatrices.

$$U_{11} \cdot V_{11} = \underline{I}$$
$$U_{22} \cdot V_{22} = \underline{I} \qquad (14)$$
$$U_{33} \cdot V_{33} = \underline{I}$$

$$U_{11} \cdot V_{12} + U_{12} \cdot V_{22} = \underline{0}$$
$$U_{22} \cdot V_{23} + U_{23} \cdot V_{33} = \underline{0} \qquad (15)$$

$$U_{11} \cdot V_{13} + U_{12} \cdot V_{23} + U_{13} \cdot V_{33} = \underline{0} \qquad (16)$$

First of all, we perform the local matrix inversions of all diagonal submatrices to generate $V_{pp} = U_{pp}^{-1}$ for p=1,2,3. Such inversions are always possible due to the strong nonsingularity of the given matrix $\underline{U}$.

Using the inverse submatrices just generated, we generate two additional submatrices of $\underline{V}$ by Eq. 15 as follows:

$$V_{12} = -V_{11} \cdot (U_{12} \cdot V_{22})$$
$$V_{23} = -V_{22} \cdot (U_{23} \cdot V_{23}) \qquad (17)$$

The last submatrix of V_{33} can be computed by rearranging Eq. 16 as follows using the two submatrices generated in Eq. 17.

$$V_{13} = -V_{11} \cdot (U_{12} \cdot V_{23} + U_{13} \cdot V_{33}) \qquad (18)$$

The above recursive steps of generating submatrices, V_{pq} for $1 \leq p,q \leq k$, of the inverse matrix

$\underline{V} = \underline{U}^{-1}$ are summarized in Algorithm 2.

ALGORITHM 2

Inputs:

Submatrices U_{pq} of matrix $\underline{U} = (u_{ij})$ for all $q \geq p = 1, 2, \ldots, k$.

Outputs:

$k \cdot (k+1)/2$ submatrices V_{pq} of the inverse matrix $\underline{V} = \underline{U}^{-1}$ for all $q \geq p = 1, 2, \ldots, k$.

Procedures:

1. For p ← 1 to k step 1 do
   $V_{pp} = U_{pp}^{-1}$
   Repeat
2. For q ← 1 to (k-1) step 1 do
   For p ← 1 to k-q step 1 do

$$W_{p,p+q} = \sum_{r=1}^{q} U_{p,p+r} \cdot V_{p+r,p+q}; \qquad (19)$$

$$V_{p,p+q} = - V_{pp} \cdot W_{p,p+q}.$$

   Repeat
   Repeat

The partitioned multiplication of two large n x n matrices, say $\underline{A} \times \underline{B} = \underline{C}$, is rather straightforward. We include it in Algorithm 3 for completeness. Basically, each m x m submatrices C_{pq} of the product matrix $\underline{C}$ is obtained by performing the accumulative multiplications in a Type-III module. All k^2 modules operate in parallel.

ALGORITHM 3

Inputs:

m x m submatrices A_{pr} and B_{rq} of the matrix $\underline{A} = (a_{ij})$ and matrix $\underline{B} = (b_{ij})$, for $p,q,r = 1,2,\ldots,k$.

Outputs:

m x m submatrices C_{pq} of the resulting product matrix $\underline{C} = (c_{ij})$, for $p,q = 1,2,\ldots,k$.

Procedures:

For p ← 1 to k step 1 do
  For q ← 1 to k step 1 do

$$C_{pq} = \sum_{r=1}^{k} A_{pr} \cdot B_{rq} \qquad (20)$$

  Repeat
Repeat

6. PARTITIONED VLSI SOLUTION OF TRIANGULAR SYSTEMS

After L-U decomposition of a dense system $\underline{A} \cdot \underline{x} = \underline{b}$, we obtain $(\underline{L} \cdot \underline{U}) \cdot \underline{x} = \underline{L} \cdot (\underline{U} \cdot \underline{x}) = \underline{b}$. This actually represents the solution of two triangular subsystems. The forward elimination is specified by $\underline{L} \cdot \underline{d} = \underline{b}$ and the backforward substitution corresponds to $\underline{U} \cdot \underline{x} = \underline{d}$. The solutions to these two subsystems lead to the solution of the dense system $\underline{A} \cdot \underline{x} = \underline{b}$. A systematic partitioning procedure is described below to achieve the modular VLSI solution of any triangularized system.

By denoting the old coefficient vector $\underline{b} = (b_1, b_2, \ldots, b_n)^T = (a_{1,n+1}, a_{2,n+1}, \ldots, a_{n,n+1})^T$, we can expand the n x n characteristic matrix $\underline{A} = (a_{ij})$ into an n x (n+1) matrix $\underline{A}' = (\underline{A}, \underline{b})$ with $\underline{b}$ as the rightmost column. The iterative computations of $a_{ij}^{(k)}$ for $k = 1,2,\ldots,n$ in Eq. 1 are extended to cover all $n \cdot (n+1)$ elements for $1 \leq i \leq n$ and $1 \leq j \leq n+1$. The transformed coefficient vector $\underline{d} = (d_1, d_2, \ldots, d_n)^T$ is generated by solving the forward system $\underline{L} \cdot \underline{d} = \underline{b}$ as follows:

$$d_k = a_{k,n+1}^{(k)} \quad \text{for } k = 1,2,\ldots,n \qquad (21)$$

Comparing Eq. 2 and Eq. 21, we realize the similarity between computing u_{kj} for all $k \leq j$ and computing d_k for $k = 1,2,\ldots,n$. Therefore, the same L-U decomposition hardware used to generate the triangular matrix $\underline{U} = (u_{ij})$ can be used, with minor modification, to generate the new column vector $\underline{d}$ to be used in the backward system $\underline{U} \cdot \underline{x} = \underline{d}$. We use the following system with order $n = 6$ to illustrate the partitioned solution method of $\underline{U} \cdot \underline{x} = \underline{d}$ with known $\underline{U} = (u_{ij})$ and $\underline{d} = (d_1, d_2, \ldots, d_6)^T$.

$$\left[\begin{array}{cc|cc|cc} u_{11} & u_{12} & u_{13} & u_{14} & u_{15} & u_{16} \\ 0 & u_{22} & u_{23} & u_{24} & u_{25} & u_{26} \\ \hline 0 & 0 & u_{33} & u_{34} & u_{35} & u_{36} \\ 0 & 0 & 0 & u_{44} & u_{45} & u_{46} \\ \hline 0 & 0 & 0 & 0 & u_{55} & u_{56} \\ 0 & 0 & 0 & 0 & 0 & u_{66} \end{array}\right] \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{bmatrix} \qquad (22)$$

Three ($k = n/m = 6/2 = 3$) partitioned solution subvectors are generated sequentially in the following order.

$$\begin{bmatrix} x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} u_{55} & u_{56} \\ 0 & u_{66} \end{bmatrix}^{-1} \cdot \begin{bmatrix} d_5 \\ d_6 \end{bmatrix}$$

$$\begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} u_{33} & u_{34} \\ 0 & u_{44} \end{bmatrix}^{-1} \cdot \left(\begin{bmatrix} d_3 \\ d_4 \end{bmatrix} - \begin{bmatrix} u_{35} & u_{36} \\ u_{45} & u_{46} \end{bmatrix} \cdot \begin{bmatrix} x_5 \\ x_6 \end{bmatrix} \right)$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}^{-1} \cdot \left(\begin{bmatrix} d_1 \\ d_2 \end{bmatrix} - \begin{bmatrix} u_{13} & u_{14} \\ u_{23} & u_{24} \end{bmatrix} \cdot \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} - \begin{bmatrix} u_{15} & u_{16} \\ u_{25} & u_{26} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right)$$

In general, the matrix $\underline{U}$ can be partitioned into $k\cdot(k+1)/2$ m x m submatrices as shown in Eq. 11. Similarly, we can partition the solution vector $\underline{x}$ into k subvectors and so can the transformed vector $\underline{d}$. The above partitioned VLSI solution procedures are summarized in Algorithm 4 for a generalized triangular LSE of order n.

ALGORITHM 4
Inputs:

m x m submatrices U_{pq} of $\underline{U}$ for $q \geq p = 1,2,\ldots,k$. The coefficient subvectors $\underline{d}_p$ for $p = 1,2,\ldots,k$, each having m consecutive elements of the vector $\underline{d}$.

Outputs:

The subvectors $\underline{x}_p$ for $p=1,2,\cdots,k$, of the solution vector $\underline{x} = [x_1,x_2,\ldots,x_n]^T$, where each $\underline{x}_p$ has m consecutive elements of $\underline{x}$.

Procedures:

For p ← k to 1 in step (-1) Compute

$$U_{pp}^{-1} \text{ from } U_{pp};$$

$$\hat{\underline{d}}_p = \underline{d}_p - \sum_{q=p+1}^{k} U_{pq} \cdot \underline{x}_q; \qquad (23)$$

$$\underline{x}_p = U_{pp}^{-1} \cdot \hat{\underline{d}}_p.$$

Repeat

It should be noted that $U_{p,k+1} = \underline{0}$ and $\underline{x}_{k+1} = \underline{0}$ were assumed in computing the subvector $\hat{\underline{d}}_p$ in Eq.23. Type-II VLSI modules are needed to perform the local matrix inversion in Algorithm 4. The computations specified in Eq. 23 are very similar to those in Eq. 6, except the matrices (a_{ij}) and (c_{ij}) has been replaced by the column vectors $\underline{d}_p$ and $\underline{x}_p$ respectively. Modified Type-III VLSI modules reduced from the design shown in Fig. 3 can be used to implement these accumulative matrix-vector multiplications.

7. PERFORMANCE ANALYSIS AND DESIGN TRADEOFFS

It has been projected by Mead and Conway [11] that by the late 80's it will be possible to fabricate IC chips, each of which contains 10^7 or 10^8 individual transistors. The VLSI computing structure requires not only large number of processing cells and latch memories, but also large number of interconnection paths throughout the integrated chip. The length and organization of these communication paths set a lower bound on the chip area and time delay required for system operations. Furthermore, the I/O and packaging constraints of monolithic IC chip set further limitation on the applicability of VLSI chips in digital system design.

Speed performance and hardware complexity of the above partitioned VLSI matrix computation algorithms are analyzed below. The time delays of Type-I, Type-II and Type-III VLSI arithmetic modules are respectively 2m, 2m, and m•r+1 units of time, where each time unit is essentially equal to the delay of one multiply cell (M cell) or of one divide cell (D cell) in those designs shown in Figs. 1-3. With typical operand length of 32 bits, the cell delays can be as low as 100 nanoseconds with projected bipolar technology. This cell delay determines the period of the clocked VLSI arithmetic devices.

To implement Algorithm 1 in hardware, only one Type-I VLSI module is sufficient to perform the successive local L-U decompositions of all diagonal submatrices L_{pp} and U_{pp} (for $p=1,2,\ldots,k$). Two Type-II modules are needed to compute the inverse matrices L_{pp}^{-1} and U_{pp}^{-1} of those diagonal submatrices being decomposed. As shown in Table 1, the number of required Type-III modules varies from 2(k-1) units in Step 2, to 2(k-2) units in Step 3_2, and eventually down to 2 units in Step $3_{(k-1)}$. Besides, Type III modules are needed to compute $\hat{A}_{pp}$ for $p=2,3,\ldots,k$. Of course, many of the Type-III modules can be shared by the successive steps. Even without resource sharing, at most $2[1+2+\ldots+(k-1)] + k = 2k(k-1)/2 + k = k^2 = (n/m)^2$ Type-III modules are needed in Algorithm 1.

A minimum-delay analysis is performed in Table 1. In other words, as many as parallel VLSI modules are employed in each computation step of Algorithm 1. The start times and time delays of the computation steps are given. Many of the matrix computation steps can be performed in a lookahead fashion, as long as there is no data dependence problem. The lookahead operations can be seen by the overlapped start times in Table 1. The minimum time delay of Algorithm 1, is obtained with maximized overlapping operations in the VLSI modules.

$$T_1 = 6n + \frac{2n}{m} - (4m+2) = 6n, \text{ if } n >> m >> 1 \qquad (24)$$

With overlapped operations, many of the Type-III modules are shared by the successive computation steps in Algorithm 1. Figure 4 shows the actual Type-III module counts in the recursive substeps 3_q for $q=2,3,\ldots,k-1$. For $q > 13$, the effect of the resource sharing becomes apparent. The hardware demand increases essentially linearly after the initial transient. The peak of each curve indicates the necessary hardware M_1 required to achieve the minimum delay T_1 in Eq.24. Through some tedious algebraic derivation, we obtain the following result on the minimum number of VLSI modules required to achieve the minimum delay T_1.

$$M_1 = \frac{1}{11} \cdot (n/m)^2, \quad \text{if } n >> m. \qquad (25)$$

In Table 2, we show the time delays and module requirements in using Algorithm 2 to compute the

inverse of a triangular matrix of order n. The total time delay is a linear function of the system order n.

$$T_2 = 2(1 + \frac{1}{m})n - 2 = 2n, \text{ if } n >> m >> 1 \quad (26)$$

To implement Algorithm 2 requires k Type-II modules to invert all the diagonal submatrices, U_{pp}^{-1} for p=1,2,...,k in Step 1. As many as k-q Type-III modules are needed in each Substep 2_q for q=1,2,...,(k-1). Therefore, at most k(k-1)/2 Type-III modules are needed. With overlapped operations in all Substeps, many of the Type-III modules are shared to yield the following minimum package count.

$$M_2 = \frac{1}{6}(n/m)^2, \text{ if } n >> m. \quad (27)$$

The variation of the required numbers of Type-III modules in successive recursive steps of Algorithm 2 is demonstrated in Fig. 5. The peak of each curve indicates the minimum module count M_2.

The partitioned matrix multiplication specified in Algorithm 3 requires T_3 delays and M_3 Type-III modules, both are minimum values.

$$T_3 = m \cdot k + 1 = n + 1 = n, \text{ if } n >> 1$$
$$M_3 = k^2 = (n/m)^2, \text{ if } n >> m \quad (28)$$

The detailed matrix computations in Algorithm 4 are illustrated in Table 3. The minimum time delay T_4 and minimum number of required Type-III modules M_4 for Algorithm 4 are obtained as

$$T_4 = 2n(1 + \frac{1}{m}) + (m-1) = 2n,$$
$$M_4 = \frac{1}{2}(n/m), \text{ if } n >> m >> 1 \quad (29)$$

The number M_4 is obtained by considering all possible ways of sharing the Type-III modules in successive cycles in Table 3. The fact M_4 being linearly proportional to n/m is due to the multiplication of a matrix by a column vector.

In the above complexity analysis, the minimum time delays were obtained at the expense of using as many parallel VLSI modules as possible as demonstrated in Figs. 4 and 5. In what follows, we conduct a tradeoff study between the computation time and the required hardware packages. In each of the four partitioned matrix algorithms the hardware module requirement is primarily determined by the use of Type-III VLSI modules. Therefore, it suffices to study the variation of computation time delays versus the available number of Type-III modules.

Our tradeoff study was carried out by simulating all the possible ways of carrying out the required computations among submatrices with various numbers of available VLSI modules. The simulation results are plotted in Fig.6 for Algorithm 1. All the time delays are monotonic decreasing functions with respect to the increase of available VLSI modules. When the module counts exceed the lower bound M_1, the minimum time delays are achieved as shown by the lower flat portions of the curves. These results demonstrate the possibility of design tradeoffs between speed and hardware cost. By presetting a speed requirement, one can always use the curves in Fig. 6 to decide the minimum hardware required to achieve the desired performance.

On the other hand, one can predict the speed performance of the above partitioned matrix algorithms under prespecified hardware allowance. The speed performance (time delays), as a function of the system order, k = n/m (for fixed m), can be visualized by drawing vertical lines which intersect all the curves in Fig. 6. The intersections correspond to speed variations with respect to the growing orders k. Such a speed function for Algorithm 1 is illustrated in Fig.7 under four imposed hardware bounds.

For small systems, say $k \leq \sqrt{11N}$ (derived from Eq. 25 and $k^2/11 \leq N$, where N is the maximally allowed number of VLSI modules), the computation time varies linearly with increasing k and is almost independent of the imposed hardware limitations. As the system order increases, the time delay increases quadratically. This information should be extremely useful to designers of hardware linear system solvers, especially when the design has to be conducted with limited hardware resources.

8. CONCLUSIONS

All the partitioned algorithms being developed have linear computation time $\mathbf{O}(n)$. Hardware modular requirements are proportional to $\mathbf{O}(n^2/m^2)$ in partitioned L-U decomposition, matrix inversion and multiplication. The partitioned VLSI solution of a triangularized linear system requires only $\mathbf{O}(n/m)$ VLSI modules. For n >> m >> 1, the minimum delays and package counts for the four algorithms are summarized in Table 4.

The systolic arrays [9,10,11] and the pipelined VLSI structure presented in our earlier paper [6] are both designed for global matrix computations. The new partitioned approach presented in this paper compares favorably over those global VLSI arrays in at least three practical aspects:

(a) Linear matrix computation time, $\mathbf{O}(n)$, is preserved in the modularization process with only minor decrease in speed. For example, our partitioned L-U decomposition can be done in $\mathbf{O}(6n)$ time, while the corresponding global systolic array of size n x n has a delay of $\mathbf{O}(4n)$.

(b) The partitioned matrix algorithms are shown implementable by VLSI array modules with feasible sizes. By feasibility, we have considered the practical constraints imposed by limited chip capacity (area) and limited I/O leads on VLSI chips.

(c) Our partitioned approach offers better expandability, maintainability and flexibility to system designers. The design tradeoffs between speed and hardware can be used to optimize the performance/cost ratio in developing special-purpose vector/matrix computers.

Toward the eventual realization of VLSI architecture for large-scale matrix computations, there are still many practical issues yet to be answered such as the layout of VLSI circuits, the I/O packaging constraints, and the operand buffering problem etc. We firmly believe that the partitioned matrix computations and modular VLSI hardware development are and will be the logical and feasible approach to the design of special-purpose matrix computing machines, until innovative schemes can be developed to alleviate the I/O bottleneck problem associated with any globally-structured VLSI computing structures.

REFERENCES

[1] Csanky, L., "Fast Parallel Matrix Inversion Algorithms", SIAM J. Computing, Vol. 5, 1976, pp. 618-623.

[2] Horowitz, E., "VLSI Architectures for Matrix Computations," Proc. of Int'l Conf. on Parallel Processing, IEEE Catalog No. 79 CH1433-2C, Aug. 21-24. 1979, pp. 124-127.

[3] Foster, M. J. and Kung, H. T., "The Design of Special-Purpose VLSI Chips", Computer Magazine, IEEE Computer Society, Jan. 1980, pp. 26-40.

[4] Hwang, K., "Global and Modular Two's Complement Array Multipliers", IEEE Trans. Computers, Vol. C-28, No. 4, April 1979, pp. 300-306.

[5] Hwang, K., Computer Arithmetic: Principles, Architecture, and Design, John Wiley, New York, 1979. Chaps. 6 and 8.

[6] Hwang, K. and Cheng, Y. H., "VLSI Computing Structures for Solving Large Scale Linear System of Equations," Proc. of Int'l Conf. Parallel Processing, IEEE Catalog No. 80 CH1569-3, Aug. 26-29, 1980, pp. 217-230.

[7] Hwang, K., Su, S. P. and Ni, L. M., "Vector Computer Architecture and Processing Techniques," Advances in Computers, Vol. 20 (M. C. Yovits, editor), Academic Press, New York, 1981.

[8] Kant, R. M. and Kimura, T., "Decentralized Parallel Algorithms for Matrix Computations", Proc. of the Fifth Annual Symp. on Computer Architecture, Palo Alto, CA, April 1978, pp. 96-100.

[9] Kung, T. H. and Leiserson, C. E., "Systolic Arrays (for VLSI)," in Sparse Matrix Proc., (Duff. I. S. et al. editors), Society for Indust. and Appl. Math., Pa. 1979, pp. 256-282.

[10] Kung, H. T., "Let's Design Algorithms for VLSI Systems," Proc. Caltech. Conf. VLSI, Cal. Tech. Pasadena, Calif., Jan. 1979, pp. 65-90.

[11] Mead, C. and Conway, L., Introduction to VLSI Systems, Addison Wesley Pub. Co., Reading, Mass. 1980, pp. 263-332.

[12] Preparata, F. P. and Vuillemin, J., "Optimal Integrated-Circuit Implementation of Triangular Matrix Inversion," Proc. of Int'l Conf. Parallel Processing, Aug. 26-29, 1980, IEEE Catalog No. 80 CH1569-3, pp. 211-216.

[13] Sameh, A. and Kuck, D., "On Stable Parallel Linear System Solvers", J. of ACM, Vol. 25, No. 1, Jan. 1978, pp. 81-91.

[14] Savage, J. E., "Area-Time Tradeoffs for Matrix Multiplication and Related Problems in VLSI Models," Tech. Report No. CS-50, Dept. of Computer Science, Brown University, August 1979. (19 pages).

[15] Swartzlander, E. E., "VLSI Architecture," in Very Large Scale Integration (VLSI): Fundamentals and Applications, (Edited by D. F. Barbe), Springer-Verlag, New York, 1980.

[16] Thompson, C. D., "A Complexity Theory for VLSI," Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon Univ., Pittsburgh, Penn., Sept. 1979.

[17] Wing, O. and Huang, J. W., "A Computation Model of Parallel Solution of Linear Equations," IEEE Trans. Comp., July 1980, pp. 632-638.

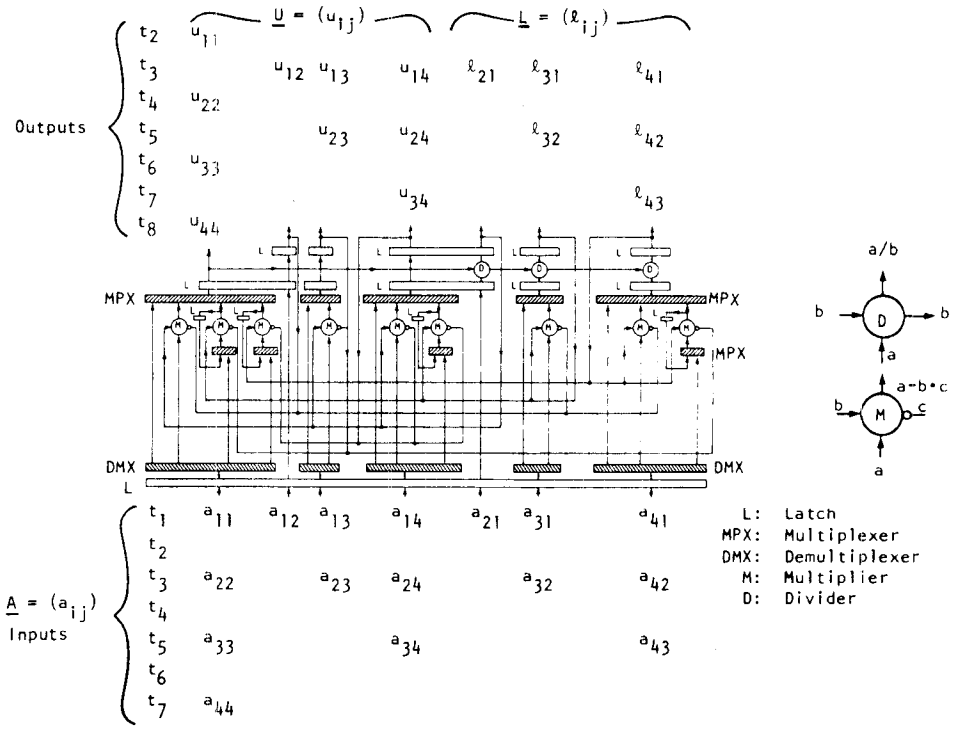


Fig. 1 Type-I VLSI Computing Module for Local (4×4) L-U Decomposition

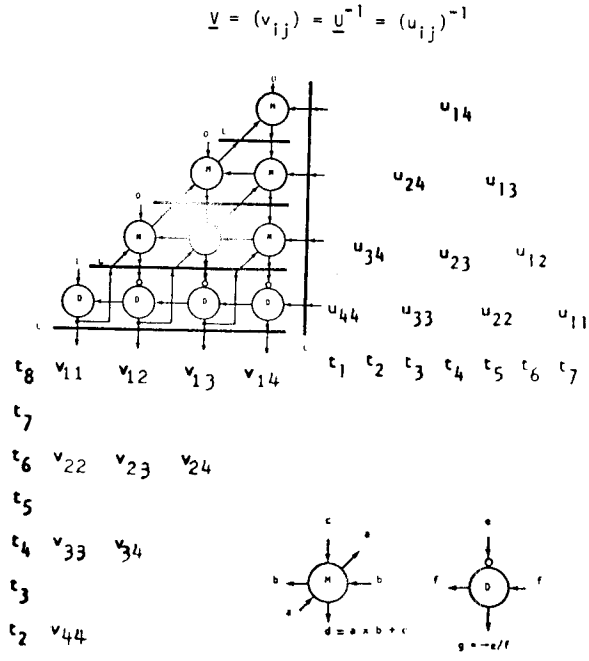
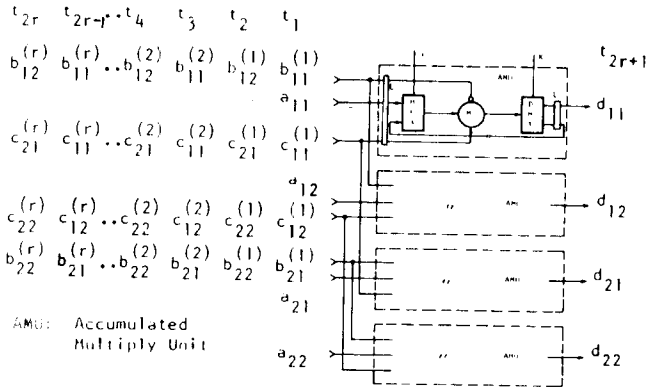


Fig. 2 Type-II VLSI computing module for the inversion of 4×4 triangular matrix.



$$\begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} - \sum_{i=1}^{r} \begin{bmatrix} b_{11}^{(i)} & b_{12}^{(i)} \\ b_{21}^{(i)} & b_{22}^{(i)} \end{bmatrix} \cdot \begin{bmatrix} c_{11}^{(i)} & c_{12}^{(i)} \\ c_{21}^{(i)} & c_{22}^{(i)} \end{bmatrix}$$

$$d_{11} = a_{11} - \sum_{i=1}^{r} b_{11}^{(i)} \cdot c_{11}^{(i)} + b_{12}^{(i)} \cdot c_{21}^{(i)}$$

$$d_{12} = a_{12} - \sum_{i=1}^{r} b_{11}^{(i)} \cdot c_{12}^{(i)} + b_{12}^{(i)} \cdot c_{22}^{(i)}$$

$$d_{21} = a_{21} - \sum_{i=1}^{r} b_{21}^{(i)} \cdot c_{11}^{(i)} + b_{22}^{(i)} \cdot c_{21}^{(i)}$$

$$d_{22} = a_{22} - \sum_{i=1}^{r} b_{21}^{(i)} \cdot c_{12}^{(i)} + b_{22}^{(i)} \cdot c_{22}^{(i)}$$

Fig. 3 Type-III VLSI computing module for accumulative multiplication of 2×2 submatrices.

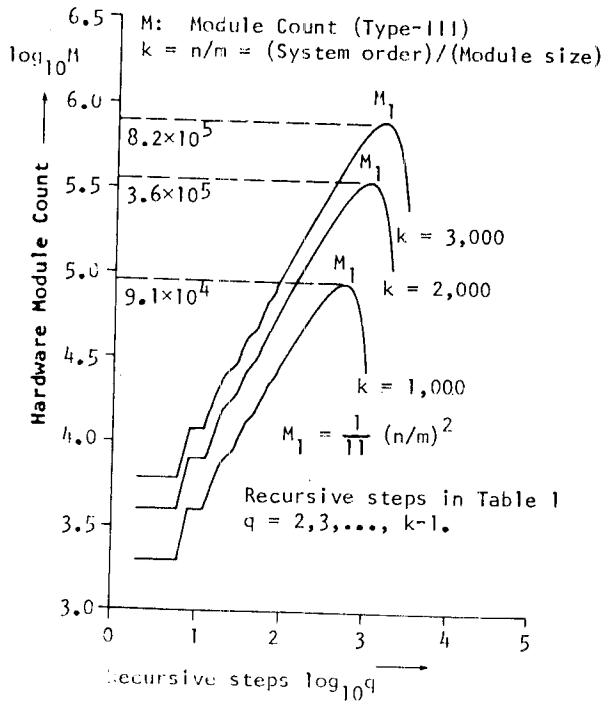


Fig. 4 Hardware demand in successive recursive steps of Algorithm 1.

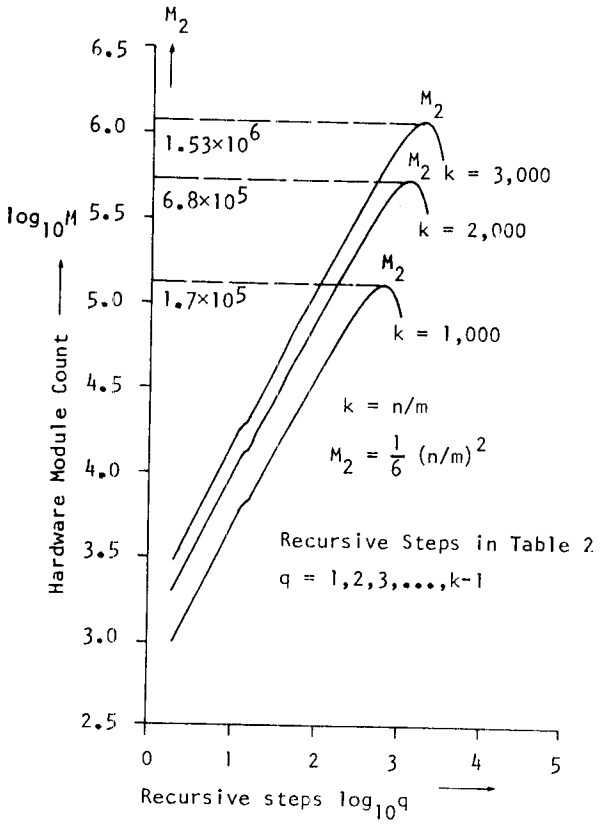


Fig. 5 Hardware demand in successive recursive steps of Algorithm 2

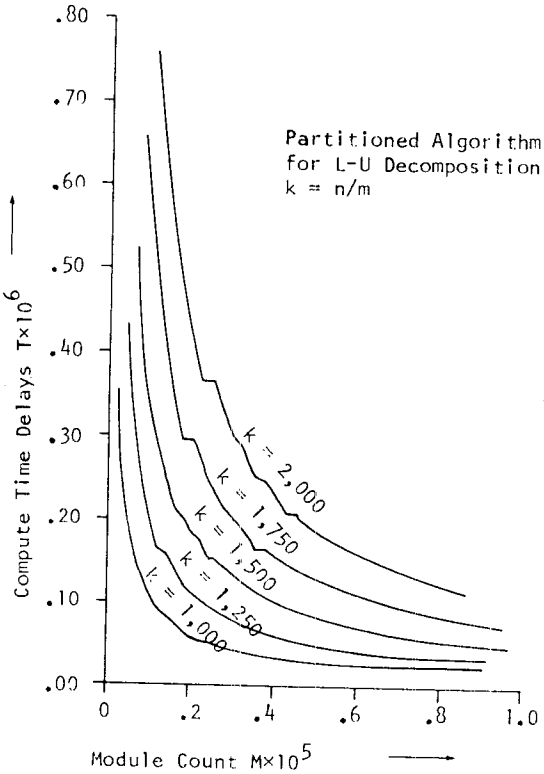


Fig. 6 Tradeoffs between computation time and available hardware modules for Algorithm 1

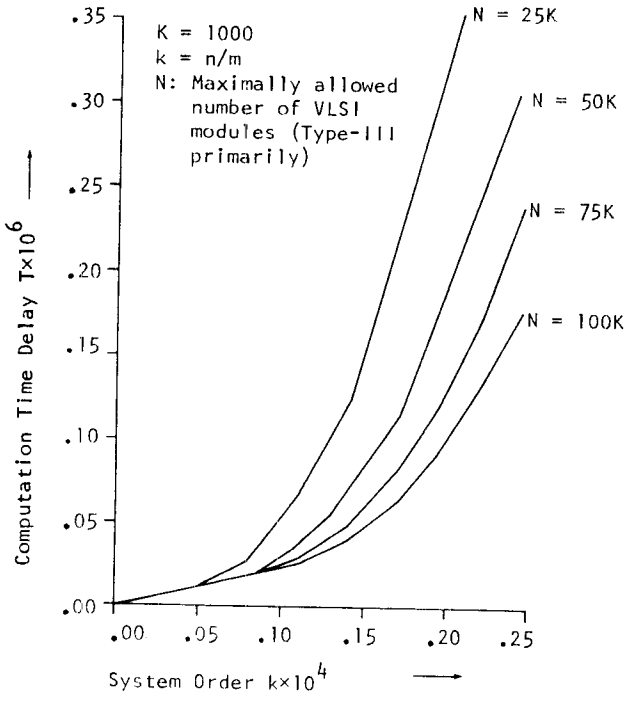


Fig. 7 Speed performance of Algorithm 1 using limited number of VLSI computing modules.

Tabel 1. Time and Hardware Complexities of the Partitioned L-U Decomposition (Algorithm 1)

Step	Submatrix Computations	Time Complexity*		VLSI Module Count		
		Start Time	Delay	Type I	Type II	Type III
1	$A_{11} = L_{11} \cdot U_{11}$	0	2m	1		
2	L_{11}^{-1}, U_{11}^{-1}	2m	2m		2	
	$L_{p1} = A_{p1} \cdot U_{11}^{-1}$ $U_{1p} = L_{11}^{-1} \cdot A_{1p}$	4m (for p=2,3,...,k)	m+1			2(k-1)
3_2 (q=2)	$\hat{A}_{22}$	5m+1	m+1			1
	$\hat{A}_{22} = L_{22} \cdot U_{22}$	6m+2	2m	1		
	L_{22}^{-1}, U_{22}^{-1}	8m+2	2m		2	
	$\hat{A}_{p2}$, $\hat{A}_{2p}$	9m+1 (for p=3,4,...,k)	m+1			2(k-2)
	L_{p2}, U_{2p}	10m+2 (for p=3,4,...,k)	m+1			
3_3 (q=3)	$\hat{A}_{33}$	10m+3	2m+1			1
	$\hat{A}_{33} = L_{33} \cdot U_{33}$	12m+4	2m	1		
	L_{33}^{-1}, U_{33}^{-1}	12m+3	2m		2	
	$\hat{A}_{p3}$, $\hat{A}_{3p}$	12m+4	2m+1			2(k-3)
	L_{p3}, U_{3p}	14m+4	m+1			
⋮	⋮	⋮	⋮			⋮
3_{k-1} (q=k-1)	$\hat{A}_{k-1,k-1}$	(5m+2)(k-2)-1	(k-2)m+1			1
	$L_{k-1,k-1}$; $U_{k-1,k-1}$	(6m+2)(k-2)	2m	1		
	$L_{k-1,k-1}^{-1}$; $U_{k-1,k-1}^{-1}$	(6m+2)(k-2)+2m	2m		2	
	$\hat{A}_{k,k-1}$; $\hat{A}_{k-1,k}$	(5m+2)·(k-2) +(4m-1)	(k-2)m+1			2·1
	$L_{k,k-1}$; $U_{k-1,k}$	(6m+2)(k-2)+4m	m+1			
4	$\hat{A}_{kk}$	(5m+2)(k-1)-1	(k-1)m+1			1
	$\hat{A}_{kk} = L_{kk} \cdot U_{kk}$	(6m+2)(k-1)	2m	1		
Total Time Delay		$T_1 = 6n + \frac{2n}{m} - (4m+2) = 6n$ for $n >> m >> 1$				
Total Module Counts for $n >> m >> 1$				1	2	$\frac{1}{11}(n/m)^2$

Note: q is the looping index used in Algorithm 1 and k = n/m.

Table 4. Complexity of Partitioned Matrix Algorithms

Matrix Algorithm	Time Delay	VLSI Module Count
1. L-U Decomposition	6n	$\frac{1}{11}(n/m)^2$
2. Matrix Inversion	2n	$\frac{1}{6}(n/m)^2$
3. Matrix Multiplication	n	$(n/m)^2$
4. Solution of Triangular System	2n	$\frac{1}{2}(n/m)$

Table 2.
Time/Hardware Complexity of the Partition Matrix Inversion (Algorithm 2).

Step	Submatrix Computations	Start Time	Delay	VLSI Modules Type II	VLSI Modules Type III
1	$V_{pp} = U_{pp}^{-1}$	0 (for p=1,2,...,k)	2m	k	
2_1	$U_{p,p+1} \cdot U_{p+1,p+1}^{-1} = W_{p,p+1}$	2m	m+1		
(q=1)	$V_{p,p+1} = -U_{pp}^{-1} \cdot W_{p,p+1}$	for p=1,2,...,k-1	m+1		k-1
2_2	$\sum_{r=1}^{2} U_{p,p+r} \cdot V_{p+r,p+2} = W_{p,p+2}$	3m+1	2m+1		
(q=2)	$V_{p,p+2} = -U_{pp}^{-1} \cdot W_{p,p+2}$	for p=1,2,...,k-2	m+1		k-2
⋮	⋮	⋮	⋮		⋮
2_q	$\sum_{r=1}^{q} U_{p,p+r} \cdot V_{p+r,p+q} = W_{p,p+q}$	(q+1)m+2(q-1)	q·m+1		
	$V_{p,p+q} = -U_{pp}^{-1} \cdot W_{p,p+q}$	for p=1,2,...,k-q	m+1		k-q
⋮	⋮	⋮	⋮		⋮
2_{k-1}	$\sum_{r=1}^{k-1} U_{1,1+r} \cdot V_{1+r,k} = W_{1,k}$	k(m+2)-4	(k-1)m+1		
(q=k-1) p=1	$V_{1k} = -U_{11}^{-1} \cdot W_{1,k}$	2n+2(n/m)-(m+3)	m+1		1
Total Delay Time		$T_2 = 2n+2(n/m)-2 = 2n$ for $n >> m >> 1$			
Total VLSI module counts for $n >> m >> 1$				(n/m)	$\frac{1}{6}(n/m)^2$

Note: q is the looping index in Algorithm 2 and k = n/m.

Table 3.
Time/Hardware Complexity of the Partitioned VLSI Solution of Triangular Linear Algebraic System (Algorithm 4).

Step	Submatrix Computations	Start Time	Delay	VLSI Modules Type II	VLSI Modules Type III
p=k	U_{kk}^{-1}; $(\hat{d}_k = d_k)$	0	2m	1	
	$x_k = U_{kk}^{-1} \cdot \hat{d}_k$	2m	m+1		1
p=k-1	$U_{k-1,k-1}^{-1}$	2m+2	2m	1	
	$\hat{d}_{k-1} = d_{k-1} - U_{k-1,k} \cdot x_k$	3m+1	m+1		1
	$x_{k-1} = U_{k-1,k-1}^{-1} \cdot \hat{d}_{k-1}$	4m+2	m+1		1
⋮	⋮	⋮	⋮		⋮
p=2	U_{22}^{-1}	2(k-2)(m+1)	2m	1	
	$\hat{d}_2 = d_2 - \sum_{q=3}^{k} U_{2q} \cdot x_q$	(k-2)(m+2)+(2m-1)	(k-2)m+1		1
	$x_2 = U_{22}^{-1} \cdot \hat{d}_2$	2(k-2)(m+1)+2m	m+1		1
p=1	U_{11}^{-1}	2(k-1)(m+1)	2m	1	
	$\hat{d}_1 = d_1 - \sum_{q=2}^{k} U_{1q} \cdot x_q$	(k-1)(m+2)+(2m-1)	(k-1)m+1		1
	$x_1 = U_{11}^{-1} \cdot \hat{d}_1$	2(k-1)(m+1)+2m	m+1		1
Total Time Delay		$T_4 = 2n+2(n/m) + (m-1) = 2n$ for $n >> m >> 1$			
VLSI Module Counts for $n >> m >> 1$				1	$\frac{1}{2}(n/m)$

Note: Type-III module used here are slightly modified (reduced) from the regular Type-III in Fig. 3.