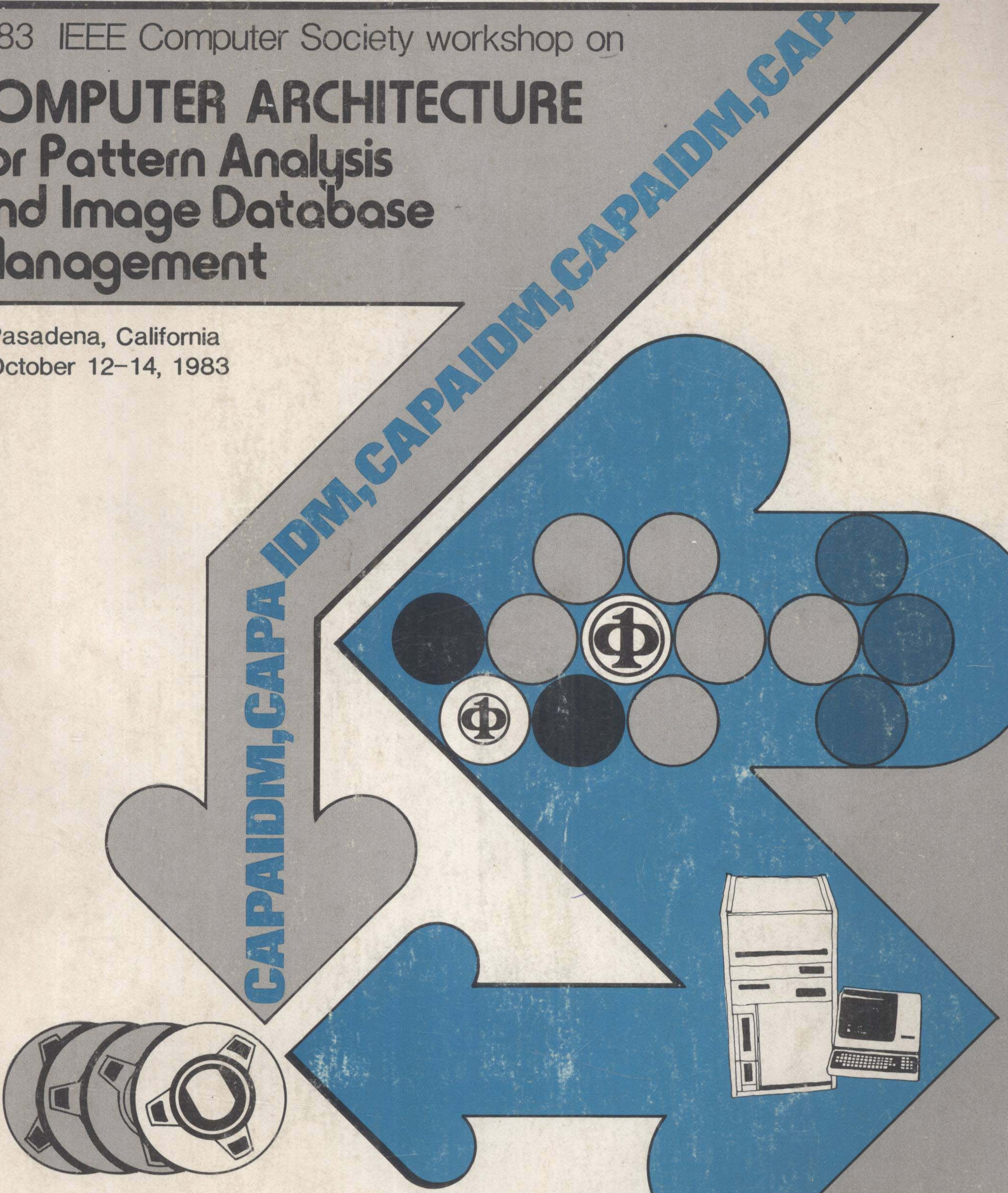
1983 IEEE Computer Society workshop on

COMPUTER ARCHITECTURE for Pattern Analysis and Image Database Management

Pasadena, California
October 12–14, 1983



ISBN 0-8186-0498-0
IEEE CATALOG NO 83CH1929-9
LIBRARY OF CONGRESS NO 83-81620
IEEE COMPUTER SOCIETY ORDER NO 498

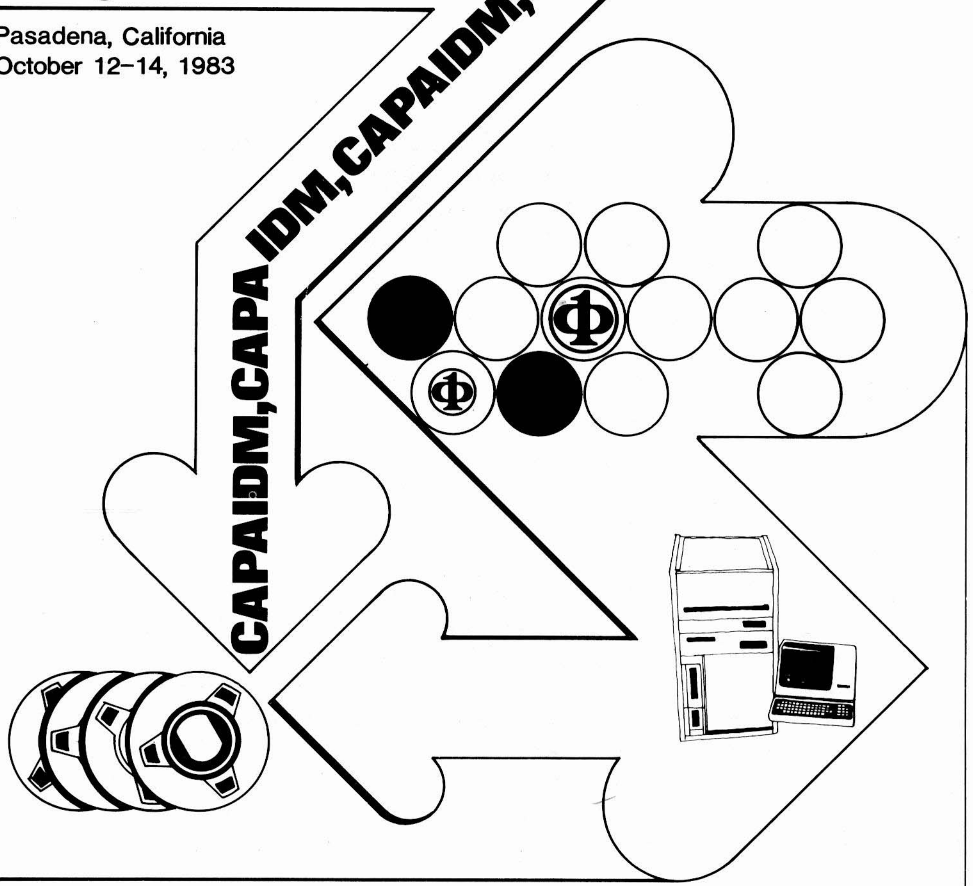
IEEE COMPUTER SOCIETY PRESS

1884 · 1984 THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

1983 IEEE Computer Society workshop on

COMPUTER ARCHITECTURE for Pattern Analysis and Image Database Management

Pasadena, California
October 12–14, 1983



ISBN 0-8186-0498-0
IEEE CATALOG NO 83CH1929-9
LIBRARY OF CONGRESS NO 83-81620
IEEE COMPUTER SOCIETY ORDER NO 498

IEEE COMPUTER SOCIETY PRESS

1884 · 1984 A CENTURY OF ELECTRICAL PROGRESS THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

Published by IEEE Computer Society Press
1109 Spring Street
Suite 300
Silver Spring, MD 20910

ISBN 0-8186-0498-0 (paper)
ISBN 0-8186-4498-2 (microfiche)
ISBN 0-8186-8498-4 (casebound)
Library of Congress No. 83-81620
IEEE Catalog No. 83CH1929-9
IEEE Computer Society Order No. 498

Order from: IEEE Computer Society
Post Office Box 80452
Worldway Postal Center
Los Angeles, CA 90080

IEEE Service Center
445 Hoes Lane
Piscataway, NJ 08854

1884 · 1984 A CENTURY OF ELECTRICAL PROGRESS The Institute of Electrical and Electronics Engineers, Inc.

Preface

This proceedings, containing some 45 papers, represents the written record of the Second (1983) Workshop on Computer Architecture for Pattern Analysis and Image Database Management (CAPAIDM). The workshop is sponsored by IEEE Computer Society, Machine Intelligence and Pattern Analysis Technical Committee. The interdisciplinary area of CAPAIDM cuts across at least three separate disciplines, namely, computer architecture, pattern analysis, and database management. The technical program of the workshop consists of eleven sessions of invited and contributed papers and two panel sessions. The quality of the papers submitted and the strong interest in participating in the workshop attest to the increasing importance of the subject matter, both as an active research area and as a field for promising applications.

The success of this workshop comes primarily from the dedicated efforts of a relatively small number of people. These are the volunteers who organize the workshop, plan its program and proceedings, and look after the many little details involved in the local arrangements. On behalf of the Workshop Committee, I would like to express my sincere thanks to the members of the Program Committee, Local Arrangements, Publication, Secretary-Treasurer, as well as session organizers and referees. We would also like to acknowledge all the help from the staff of IEEE Computer Society and the travel support from NSF.

R.J. Wall
Program Chairman

Workshop Committee

Co-Chairmen

K.S. Fu
Purdue University

Allen Klinger
University of California at Los Angeles

Program Committee

R. Bernstein, International Business Machines
K. Castleman, Jet Propulsion Laboratory
S.K. Chang, Illinois Institute of Technology
B. Chern, National Science Foundation
J. Cutts, Jet Propulsion Laboratory
A. Habibi, Aerospace Corporation
K. Hwang, Purdue University
R. Larsen, National Aeronautics and Space Administration
T.N. Mudge, University of Michigan
R. Nathan, Jet Propulsion Laboratory
A. Oosterlinck, Center for Human Genetics, Belgium
H.J. Siegel, Purdue University
L.J. Siegel, Purdue University
L. Uhr, University of Wisconsin

Local Arrangements Chairman

A. Zobrist
Jet Propulsion Laboratory

Publication Chairman

R.J. Wall
Jet Propulsion Laboratory

Secretary/Treasurer

M. Naraghi
Jet Propulsion Laboratory

Table of Contents

Preface iii

Workshop Committee. iv

Session I: VLSI Architectures I (J.A. Cutts, Chairman)

VLSI Arrays with Limited I/O Bandwidth for
Pattern Analysis. 2
P.S. Liu and T.Y. Young
An EPL-Based VLSI Architecture. 10
J. Barat
Large Array VLSI Filter 15
R. Nathan

Panel Session II: VLSI Architectures for Image Processing
(K. Hwang, Moderator; R. Kuhn, R. Lau, G. Nudd,
S.Y. Kung, Z. Stroll, and R. Wood: Panelists)

Session III: Processing Architectures (K.R. Castleman, Chairman)

Quadrant Architecture for Fast In-Place Algorithms. 26
Ph. W. Besslich and J.O. Kurowski
The Architecture of a 1024^2 Acquisition, Processing,
and Display System for Radiological Images. 34
N.J. Mankovich and H.K. Huang
The Conversion via Software of a SIMD Processor
into a MIMD Processor 37
A. Guzman, M. Gerzso, K.B. Norkin,
and S.Y. Vilenkin
The TRO Cellular Logic Computer 47
K. Preston, Jr. and R. Ragusa
A Pipeline Multiprocessor Architecture for High
Speed Cell Image Anaysis. 52
K.R. Castleman, K.H. Price, R. Eskenazi,
M.M. Ovadya, and M.A. Navon

**Session IV: Algorithms and Architectures for
Computer Vision** (T.N. Mudge, Chairman)

Case Study of a Program for the Recognition of
Occluded Parts. 56
T.N. Mudge and T.S. Abdel-Rahman
Architecture for a Computer Vision System 61
R. Jain
A Class of Localized, Non-Linear Image Filters
and an Architecture for Implementation. 62
R.M. Lougheed
A Multicluster Approach to Computer Vision. 68
A.P. Reeves and C.H. Jeon
Algorithms for Real-Time Motion Estimation. 75
P.H. Eichel and E.J. Delp

Session V: Multicomputer Processing (H.J. Siegel and L.J. Siegel, Co-Chairmen)

Feasible Multi-Computer Architectures, Given 3-Dimensional Stacked Wafers. 80
L. Uhr

Exploitation of Parallelism and Ultraspeed Integrated Circuits in the Next Generation of Distributed Super Signal Processors 87
B.K. Gilbert, B.A. Naused, S.M. Hartley, W.K. VanNurden, and R. Deming

Parallel Algorithms for Computer Vision 93
T.A. Rice and L.J. Siegel

A Distributed Memory Management System for PASM. 101
J.T. Kuehn, H.J. Siegel, and M. Grosz

Session VI: VLSI Architectures II (R. Nathan, Chairman)

A VLSI Systolic Architecture for Pattern Clustering. 110
L.M. Ni and A.K. Jain

Reconfigurable VLSI Arrays for Pattern Analysis and Image Processing 118
T.Y. Young, P.S. Liu, and Y. Gao

VLSI Architectures for Designing VLSI Chips. 125
T.E. Mangir

Architectural Principles for the Design of "Wide Band" Image Analysis Systems. 126
U. Bruning, W.K. Giloi, and C.E. Liedtke

Session VII: Algorithms and Applications I (A. Habibi, Chairman)

Optimal Design of Systolic Arrays for Image Processing. 134
G.-J. Li and B.W. Wah

Performance of Neighbor and Overlap Methods for Image Refinement. 142
A. Klinger

A Non-Left-to-Right Parser for Syntactic Shape Recognition 143
A. Giordana and L. Saitta

Session VIII: Hierarchical Algorithms and Architectures for Pattern Analysis (L. Uhr, Chairman)

A Boolean Matching Operator for Hierarchical Cellular Logic 152
S.L. Tanimoto

Region Matching in Pyramids 153
W.I. Grosky and R. Jain

Pyramid Machines: Some Advantages 159
A. Rosenfeld
Augmenting Pyramids and Arrays by
Compounding Them with Networks. 162
L. Uhr

Session IX: Multiprocessor Concurrent Processing
(R. Larsen, Chairman)

Architecture of the MPP 170
K.E. Batcher
Image Processing Applications on the MPP. 175
S.C. Cox
ZMOB--A Multicomputer System for Study
of Parallelism in Computing 176
A.K. Agrawala
Interprocessor Communication Requirements
for Parallel Image Processing 177
T.R. Kushner and A. Rosenfeld
Ultra High Performance Computer
System Issues 184
K.R. Wallgren

Session X: Image Languages and Data Bases
(S.-K. Chang, Chairman)

A User Oriented Language for Image Processing
and Object Recognition. 186
G. Cividini, G. Gini, and M. Gini
Performance Evaluation of Database Machines
Based on the Data Relationships 187
L.L. Miller and A.R. Hurson
Image Navigation and Editing Techniques 193
S.-K. Chang
Automatic Input of Water Depth Map. 194
Y. Tozawa, M. Ioka, and M. Sato

Panel Session XI: Future Trends in Image Processing and Pattern Analysis (R. Bernstein, Moderator; A. Sawchuck and A. Habibi: Panelists)

Session XII: Algorithms and Applications II
(Chairman to be determined)

Texture Analysis Using Two-Dimensional
Quadrature Filters. 206
H. Knutsson and G.H. Granlund
A New Pipelined/Parallel Architecture for
Two Dimensional Fast Fourier Transform. 214
W. Liu
A Parallel Earley's Parser and Its Application
to Syntactic Image Analysis 220
Y.P. Chiang and K.S. Fu

VLSI Architectures for the Linear Discriminant
Function Classifier 227
R.B. Urquhart
New Data Representations of an Octree
for an Octree Processor 233
K. Yamaguchi and T.L. Kunii

Session XIII: Industrial Systems (A. Oosterlinck, Chairman)

A Reconfigurable Multiprocessor Structure
for Image Processing. 236
L. Van Eycken
Industrial Inspection by Morphological
Virtual Gauging 237
S.R. Sternberg and E.S. Sternberg
Systems Architecture of a Remote Sensing
Expert System 248
W.C. Chiou, Sr.

Late Paper

A Boolean Matching Operator for
Hierarchical Cellular Logic 253
S.L. Tanimoto

Author Index. 257

Session I
VLSI Architectures

Chairman

J.A. Cutts
Jet Propulsion Laboratory

VLSI ARRAYS WITH LIMITED I/O BANDWIDTH FOR PATTERN ANALYSIS

Philip S. Liu*
Center 4424
American Bell
Denver, Colorado 80234

Tzay Y. Young
Dept. Elec. and Comp. Eng.
University of Miami
Coral Gables, Florida 33124

Matrix multiplication and covariance matrix inversion are needed in many pattern analysis algorithms. For the VLSI multiplication array, three configurations of multiplexing loading, processor row loading, and processor group loading are used to minimize the effect of limited I/O bandwidth. Under different adverse situations, a properly chosen configuration can significantly reduce the computation time.

Two new orthogonally-connected triangular arrays for L-U decomposition of a matrix using Crout's and Cholesky's methods respectively are proposed. To minimize the limited I/O bandwidth or limited I/O pins problem, reconfiguration techniques are used to restructuring a computing array, so that it can carry out in succession several distinct functional computation of matrix inversion without the data leaving the array. Computation time for pattern analysis is reduced, since the limited I/O bandwidth affects only the first and last phases of the necessary computations.

Introduction

The significance of VLSI architecture is well documented[1-2], and several systolic type VLSI array structures have been proposed[3-11]. Such arrays may be attached or interfaced to a host system or systems as special purpose devices. As many processing elements as possible can be fabricated into a VLSI chip to form an entire or part of a computing array.

The computation speed of a VLSI array may be limited by the bandwidth of the host system or the VLSI array whichever is smaller. The bandwidth is directly related to the data rate and number of the host system's I/O lines or the number of the computing array's I/O pins. With limited I/O bandwidth or capacity, it may not be possible in some cases to pipe sufficient amount of data into the computing array to sustain continuously the activities of all the processing elements that can be possibly fabricated into the VLSI computing array. Under this situation, the design of the computihg array should aim at minimizing the adverse effect of limited I/O capacity and preserving the large data storage and parallel computation advantages offered by VLSI computing arrays.

Statistical pattern classification can be divided into two phases, the learning phase and the classification phase. It is assumed that classification is based on linear or quadratic discriminant function.[12] During the learning phase, it is necessary to compute covariance matrix from sample vectors. During the classification phase, the computations involved are matrix multiplication, matrix-vector multiplication, vector dot product, and covariance matrix inversion. Thus, in this paper, we concentrate on the effect of limited I/O bandwidth on matrix operations, since the same effect will limit the computation speed of VLSI arrays designed for pattern analysis. Specific examples of pattern analysis arrays using matrix operations can be found in the literature[13-16].

Hwang and Su[14] considered VLSI arrays for the computation of Fisher's linear discriminant, using a partitioned matrix approach. Liu and Fu[15] discussed VLSI architecture for minimum distance classifiers, an important subclass of linear classifiers. In a companion paper[15], the authors discussed VLSI arrays for computing linear and quadratic discriminant functions.

For the VLSI matrix multiplication array, three configurations of interfacing and control are studied, with emphasis on the adverse effect of limited I/O bandwidth. Two new algorithms are proposed for Crout's and Cholesky's methods of L-U decomposition of a matrix, using orthogonally-connnected triangular arrays. Reconfiguration techniques are used to restructure a computing array, so that it can carry out in succession the several distinct functional computations of matrix inversion without the data leaving the array. Computation time is reduced, since the limited I/O bandwidth affects only the first and last steps of functional computations.

Effect of I/O Bandwidth on Matrix Multiplication

Consider the multiplication of two nxn matrices,

$$\underline{Z} = \underline{A}\,\underline{B} \qquad (1)$$

*This work was done while P.S. Liu was with the University of Miami.

CH1929-9/83/0000/0002$01.00 © 1983 IEEE

Assuming the matrix A is already inside the array, B can be piped in to interact with A to produce the multiplication result as shown in Fig. 1. During each computation cycle, all the cells in the array perform the same multiplication step: Each processing element (PE) takes the sum of partial products from its left neighbor and add it to its partial product of a x b, and then passes the sum to its right neighbor for the next multiplication step. For each multiplication step, the data stream of B is piped one row deeper into the array. Since loading of A and piping of B can use the same set of array input pins, the pin requirement of the multiplication array is reduced.

Without I/O limitation, the computation time is (4n - 2) units, including n units of matrix A loading time and (3n - 2) units of multiplication time. A PE, once activated, performs the necessary multiplication at every cycle time, until there is no more data entering the PE. At the peak of computation, about 75 percent of the n^2 PE's compute simultaneously.

In the following, we assume that the matrices A and B are provided by two separated sources, e.g., one by the host system and the other by a sub-system, and each system has g I/O bus lines leading to the multiplication array. The array has 2f I/O pins. The maximum data rates of each I/O bus lines and array I/O pins are assumed identical. For convenience purposes and without loss of generality, f and g are normalized so that in the ideal case, f = g = n and the multiplication-and-loading time of (4n - 2) units is maintained.

In reality, the effective computation rate is reduced because of I/O bandwidth limitations. If f or g is less than n, clearly the effective computation rate is reduced by a factor of min(f,g)/n, and hence the multiplication time under I/O constraint is bounded by

$$E_o = \frac{n(3n - 2)}{\min(f,g)} \tag{2}$$

Scheme 1: Multiplexing Loading

The two matrices A and B use the same set of I/O pins of the array. As shown in Fig. 2, a data selector first loads matrix A into the array, and then pipes in matrix B for the multiplication. The computation time is

$$E_1 = \frac{n^2}{g} + \frac{n(3n - 2)}{g}, \text{ if } f \geq g \tag{3}$$

$$E_1 = \frac{n^2}{g} + \frac{n(3n - 2)}{f}, \text{ if } g > f \tag{4}$$

The first term in each E_1 is the matrix A loading time. The second term is the actual multiplication time. It is noted that $E_1 \simeq 4E_0/3$. For f > g, part of the array's I/O bandwidth is completely wasted, which could be put into good use as shown in scheme 2.

Scheme 2: Processor Row Loading

The two matrices are gated into the array using two separate sets of array input pins assuming f > g. In this scheme, loading of A may partially overlap with the matrix multiplication steps, so that overall computation time may be reduced. For illustration purpose, let n = 4, g = 2, and f ≥ 3. Since the total number of array I/O pins are 2f ≥ 6, then one-third of them or 2f/3 can be assigned to handle the input data stream of A, another one-third to handle the input data stream of B, with the remaining one-third to handle the output stream of Z.

The array control and interface structure is shown in Fig. 3. The basic processing cell in this scheme can perform under control either a shift operation or multiplication operation. Initially, the rows of A are loaded and stored into the array starting from the bottom row of the array with each row delayed by one matrix element loading time. It is observed that multiplication operation of matrix elements can start immediately for all cells in a row as soon as the row is completely loaded. In Fig. 3, we use a double buffering scheme with pin 1 and pin 2 connected to the even and odd rows, respectively. Input pin 3 is connected to buffer B1 which in turn is connected to buffer B2. Transfer of data between input pin 3 and B1 and that between B1 and B2 are carried out concurrently.

It should be noted that in this example, multiplication steps are performed at every other time intervals, and loading of the top two processor rows overlaps with the loading of the B buffers and multiplications carried out at the bottom two processor rows. Piping the result out of the VLSI array can also be handled in a similar fashion as piping in the elements of B.

With the assumption that 2f/3≥g, the computation time E_2 is

$$E_2 = \frac{n}{g}((g-1) + (3n-2)), 2f/3 \geq g . \tag{5}$$

In (5), n(g-1)/g is the effective loading time of matrix A. 2f/3≥g means that the array has enough input pins to accommodate the number of output lines g of either the host system or subsystem. Two separate data streams can be piped into the array with overlap, and the total execution time is reduced. If g>2f/3 the effective bandwidth of the host system or subsystem is reduced and the execution time increases accordingly. Thus,

$$E_2 = \frac{n}{2f/3}((2f/3-1) + (3n - 2)), g>2f/3 \tag{6}$$

It is easy to show from (3) - (6) that E_2 is smaller than E_1 when 2f/3≥g and E_1 is smaller than E_2 when f>g. Thus, E_1 is identical to E_2 somewhere inside the range f>g>2f/3. By letting $E_1 = E_2$, we obtain, after simplifications,

$$g = \frac{4f(2n-1)}{2f+9(n-1)} \quad . \qquad (7)$$

Hence, the execution time of scheme 2 is shorter if g is less than the value in (7).

Let us consider the ratio of E_2/E_1, assuming $2f/3 \geq g$. If n is a large number and n>>g, we have from (3) and (5), $E_2/E_1 \simeq 3/4$ and $E_2/E_0 \simeq 1$, in other words, the second scheme's execution time is only 75% of that in the first interfacing scheme and it approaches the lower bound, assuming that the host system and the subsystem have the same bandwidth. If the effective bandwidths of the subsystem and the host are different due to different number of output lines and/or different data rates of the lines, the effective matrix $\underline{A}$ loading time and the effective multiplication time in E_1 and E_2 must be adjusted accordingly, and similar analysis can be carried out to decide which scheme should be adopted.

Scheme 3: Processor Column Group Loading

This scheme pipes in data for a processor row and data for the B buffers simultaneously. Using data from the previous scheme, the structure for doing this is shown in Fig. 4. During T_0 and T_1, C1 and C3 issue 2 shift commands simultaneously to pipe in elements for the bottom row and the buffers. These two shift commands and subsequent shift commands on C1 are delayed by the bottom row for 2 unit times before they are passed on to the next upper row, which shifts in its own data during T_2 and T_3 and repeats the same delay procedure for its upper row. During T_2, control line C2 issues a multiplication command to the bottom processor row to initiate the first multiplication step. This multiplication command and subsequent ones issued at T_4, T_6, T_8 and so on will be delayed for 2 unit times before they are passed on to the next upper row, which executes the commands at T_4, T_6, T_8,... and delays each of them for 2 unit times before passing them on. This procedure applies to subsequent processor rows. During T_2, C3 issues a shift command to shift in data from buffers B1 and B3, and during T_3, it issues a shift command to complete the updating of the B buffers. From here on, we repeat the commands issued during T_2 and T_3 until all results are obtained. Data selectors must be inserted between adjacent groups of processor columns. During data loading time, a processor in the leading column of a processor group, under the control of C1, would select data from the external data line. After that, it inputs data from a corresponding processor in the trailing column of the prceding processor group, so that multiplication steps can be carried out along the corresponding processor row.

Compared withe the second scheme, this scheme requires (g-1)n extra data selectors which allow the array to be reconfigurable to some degree. Since loading time of matrix $\underline{A}$ is completely eliminated, the required computation time E_3 in this scheme is

$$E_3 = \frac{n(3n-2)}{g} \ , \quad 2f/3 \geq g \ , \qquad (8)$$

which equals E_0. Using the same analysis as discussed in scheme 2, we conclude that if $g > 4f(2n-1)/3(3n-2)$, the third scheme is preferred to the multiplexing loading scheme.

Triangular Arrays for L-U Decomposition

The first step to compute the inverse of a non-singular covariance matrix $\underline{R} = [r_{ij}]$ is to decompose it into an upper triangular matrix $\underline{U} = [u_{ij}]$ and a lower triangular matrix $\underline{L} = [\ell_{ij}]$. In this section, we discuss two algorithms for L-U decomposition using orthogonally-connected triangular arrays. With n(n+1)/2 PE's in the arrays, the space-time products are about the same as other known decomposition arrays[3-9]. The major advantages of the triangular arrays are: (i) simple PE's, (ii) simple, orthogonally-connected communication structure, and (iii) two-dimensional data flow.

Crout's Algorithm

The covariance matrix $\underline{R}$ can be decomposed using the following recurrence procedure[17]:

$$c_{ij}^{(1)} = r_{ij} \quad ,$$

$$c_{ij}^{(k+1)} = c_{ij}^{(k)} - \ell_{ik} u_{kj} \quad ,$$

$$\ell_{ik} = \begin{cases} 0 & \text{if } i < k \ , \\ 1 & \text{if } i = k \ , \\ c_{ik}^{(k)} \ u_{kk}^{-1} & \text{if } i > k \ , \end{cases} \qquad (9)$$

$$u_{kj} = \begin{cases} 0 & \text{if } k > j \ , \\ c_{kj}^{(k)} & \text{if } k \leq j \ . \end{cases}$$

The operations required of each processing element and data movement of the array are shown in Fig. 5, assuming a 4x4 covariance matrix. The PE has 2 inputs c_{in} and u_{in} and 2 outputs c_{out} and u_{out} which are buffered or latched internally, and it has an internal storage register ℓ . Each PE operates in two modes, the multiplicatoin mode and the division mode. It is assumed that each operation takes one unit of time to complete. The multiplication mode allows the PE's in the array to implement the recurrence $c_{ij}^{(k+1)} = c_{ij}^{(k)} - \ell_{ik} u_{kj}$ in the decomposition and at the same time move u_{kj} upward. The division mode allows a PE to calculate $\ell_{ik} = c_{ik}^{(k)} u_{kk}^{-1}$ when necessary

The skewed columns of the matrix $\underline{R}$ are piped into the array in parallel as inputs.

The input data streams are piped one column deeper into the array every two units of time. The array goes through alternate division and multiplication operations. For each multiplication operation, all PE's are activated. For each division operation, however, only some PE's with the proper input data at that time will be allowed to do division and store the result into internal register ℓ. In general, and starting at the bottom PE_{11}, the ℓ_{ik} elements are computed during odd time intervals. Table 1 shows the time and place at wich each ℓ_{ik} is computed:

Table 1 Computation sequence of ℓ_{ik}

Time interval	ℓ_{ik} computed	PE involved
T_1	ℓ_{11}	PE_{11}
T_3	ℓ_{21}	PE_{21}
T_5	ℓ_{31}	PE_{31}
T_7	ℓ_{22}, ℓ_{41}	PE_{22}, PE_{41}
T_9	ℓ_{32}	PE_{32}
T_{11}	ℓ_{42}	PE_{42}
T_{13}	ℓ_{33}	PE_{33}
T_{15}	ℓ_{43}	PE_{43}
T_{19}	ℓ_{44}	PE_{44}

The computation pattern for ℓ_{ik} can be generalized. Each computation of ℓ_{ik} is delayed successively by 2 units of time for the PE's within each column of the array and by 4 units of time for the PE's within each row of the array. A computed ℓ_{ik} is stored inside its PE and used to compute subsequent $c_{ij}^{(k)}$'s and u_{kj}'s. Control signals could be piped and distributed with proper delay through the processing elements themselves, or the control timing could be customized and built into each PE individually. From the computation sequence of ℓ_{ik} as shown, it can be deduced that decomposition of any nxn covariance matrix takes (6n-3) units of time. With n(n+1)/2 PE's used in the array, the space-time product is approximately $3n^3$.

Cholesky's Algorithm

Covariance matrices are symmetric, and Cholesky's method exploits the symmetry of a matrix. We now show that our basic triangular array structure can be modified for Cholesky's decomposition method. The computation is much simpler than the hyperbolic Cholesky's method[9]. With n(n+1)/2 PE's and (3n-2) units of computation time, the space-time product is approximately $3n^3/2$; the unit cycle time is limited by the square-root operation.

A symmetric matrix $\underline{R}$ can be decomposed into $\underline{U}^T \underline{U}$. Cholesky's method may be expressed as [17]

$$c_{ij}^{(1)} = r_{ij} , \quad i \le j ,$$

$$c_{ij}^{(k+1)} = c_{ij}^{(k)} - u_{ki} u_{kj} , \quad k<i\le j ,$$

$$u_{ii} = \sqrt{c_{ii}^{(i)}} , \qquad (10)$$

$$u_{ij} = c_{ij}^{(i)} u_{ii}^{-1} , \quad i < j .$$

The triangular array is shown in Fig. 6, which consists of two types of processing elements. The square cell performs $c_{out} \leftarrow c_{in} - uu_{in}$ and passes u_{in} upward. The circular cell performs the square root operation when it is first activated, and it performs only division in the remaining cycles. It is noted from (10) and the figure that u_{ij} is computed by PE_{ii} and it is then moved upward and stored in PE_{ji} at the appropriate cycle. In Fig. 6, we have just completed the computation of u_{23} by PE_{22} and stored it in PE_{32}. The computation sequence is shown in Table 2, including the time that u_{ij} is stored in PE_{ji}.

Table 2 Computation sequence of u_{ij}

Time interval	u_{ij} computed at PE_{ii}	u_{ij} stored in PE_{ji}
T_1	u_{11}	PE_{11}
T_2	u_{12}	PE_{21}
T_3	u_{13}	-
T_4	u_{14}, u_{22}	PE_{31}, PE_{22}
T_5	u_{23}	PE_{32}
T_6	u_{24}	PE_{41}
T_7	u_{33}	PE_{33}, PE_{42}
T_8	u_{34}	PE_{43}
T_{10}	u_{44}	PE_{44}

A Reconfigurable Array for Matrix Inversion

A reconfigurable, orthogonally-connected square array may be used for computing the inverse of the covariance matrix $\underline{R}$, and the block diagram of this array is shown in Fig. 7. Internally, the outputs from the top are fed back to the bottom of the same corresponding columns. The square array can be reconfigurated into a triangular array by modifying the data paths leading into the PE's along the diagonal of the array.

Covariance Matrix Inversion

Crout's method is chosen for L-U decomposition, which does not take into consideration the symmetry of the covariance matrix. We choose this method because with this approach, the unit cycle time is limited by multiplication or division, and because the same type of PE's can be used for other functional computations. If the symmetric Cholesky's method is used, the number of time units required can be reduced by half; however, the cycle time is longer, since it is limited by the computation of the square root. Fig. 8 shows the reconfiguration of a square array into the triangular L-U decomposition array.

After L-U decomposition, the required steps for covariance matrix inversion are the inversion of the two triangular matrices $\underline{L}$ and $\underline{U}$, and the multiplication of the inversed triangular matrices. Thus in step 2, we calculate $\underline{V} = \underline{U}^{-1}$ using a reconfigurated lower triangular array. The processing operations of the PE's in this step is very similar to that in L-U decomposition with different data. During odd clock cycles, we have $v \leftarrow c_{in}/u_{in}$ or no operation. During even clock cycles, we have $c_{out} \leftarrow c_{in} - vu_{in}$ and $u_{out} \leftarrow u_{in}$. As soon as u_{11} and subsequent u_{ij}'s leave the top of the array, they are piped back without delay to the bottom of the array for the computation of $\underline{V}$. The array is gradually reconfigurated into a lower triangular array, and after the computation, the matrix $\underline{V}$ remains in the array.

Step 3 is the computation of the inverse of $\underline{L}$, which is denoted by $\underline{M} = [m_{ij}]$, and since $\underline{R}$ is symmetric $m_{ji} = v_{ij}u_{jj}$. In this step, duplicated values of v_{ij} in each processor column are rotated together four times along each column of the square array. As a v_{ij} passes through PE_{ij}, $m_{ji} = v_{ij}u_{jj}$ is calculated and shifted upward as replacement for v_{ij}. At the end, both v_{ij} and m_{ji} reside in the same PE_{ij}.

In the final step from within the array, the matrix $\underline{M}$ is piped to the top and then into the array again through the bottom. Piping of each processor column is delayed by one unit time. $\underline{M}$ interacts with $\underline{V}$ still inside the array, and multiplication steps are carried out as defined in the multiplication array. The output is therefore $\underline{U}^{-1}\underline{L}^{-1}$ which is the inverse of $\underline{R}$.

Performance Analysis

Let us first consider the case that the four steps of matrix inversion are performed for an nxn matrix $\underline{R}$ using four separate special purpose chips. With sufficiently large I/O bandwidth, the execution time of step 1 is (6n-3) units, which is also the time required for step 2. It can be seen from above discussions that overlap I/O and computations exist between the two arrays; thus with overlaps accounted for, the execution time of step 1 and step 2 is (8n-2). The time required for step 3 and step 4 are n and (4n-2) units respectively. It is noted that step 3 cannot start until all results from step 2 become available, and step 4 cannot start until step 3 is completed. The total execution time is, therefore, (13n-4) units.

With limited I/O bandwidth, we assume that only f pins are available for either input or output purposes for every array, and $f<n$. Then, the total execution time is n(13n-4)/f units.

Using the reconfigurable array, the limited I/O bandwidth affects the first and last steps only. Thus, the execution time of step 1 and step 2 is, under the limitation of f input pins, n(4n-2)/f + 4n. Step 4 requires n(4n-2)/f time units, since it is limited by the f output pins. The total execution time for the reconfigurable array is n(8n-4)/f + 5n units. Under severe pin limitations, n/f becomes a large number. The reconfigurable array takes approximately $8n^2/f$ units of time to compute, compared to $13n^2/f$ units for separate arrays. Thus, we have a reduction of 38 percent of required computation time.

References

(1) Mead, C. and Conway, L., Introduction to VLSI Systems, Addison-Wesley, Reading, Mass., 1980.

(2) Snyder, L., "Overview of the CHIP Computer" in VLSI 81-Very Large Scale Integration, John P. Gray ed. Academic Press, 1981.

(3) Kung, H.T. and Leiserson, C.E., "Systolic Arrays (for VLSI)", Proc. Symp. Sparse Matrix Computations and Their Applications, Nov. 2-3, 1978, pp. 256-282.

(4) Kung, H.T. and Leiserson, C.E., "Algorithms for VLSI Processor Arrays", in Introduction to VLSI Systems, Mead C.A. and Conway L., Addison-Wesley, Reading, Mass., 1980, pp. 271-292.

(5) Johnsson, L., "Computational Arrays for Band Matrix Equations", Technical Report, Caltech, Computer Science Dept., May 1981.

(6) Johnsson, L.,"VLSI Algorithms for Doolittle's Crout's and Cholesky's Methods", Proc. IEEE International Conf. on Circuits and Computers Sept. 1982, pp 372-376.

(7) Kung, S.Y., "VLSI Array Processor for Signal Processing", Proc. MIT Conf. on Advanced Research in Integrated Circuit, Jan. 1980, pp. 28-30.

(8) Kung, S.Y., Arun, K.S., Gal-Ezer, R.J. and Bhaskar Rao, D.V., "Wavefront Array Processor: Language, Architecture and Applications", IEEE Trans. Comput., vol. C31, Nov. 1982, pp. 1054-1066.

(9) Ahmed, H.M., Delsome, J.M., and Morf, M., Highly Concurrent Computing Structure for Matrix Arithematic and Signal Processing", Computer, Jan. 1982, pp. 65-82.

(10) Hwang, K. and Cheng, Y.-H., "Partitioned Matrix Algorithm for VLSI Arithematic Systems", IEEE Trans. Comput., vol. C-31, Dec. 1982, pp. 1215-1224.

(11) Liu, P.S. and Young, T.Y., "VLSI Array Design Under Constraint of Limited I/O Bandwidth", IEEE Trans. Comput., in press.

(12) Young, T.Y. and Calvert, T.W., Classification, Estimation and Pattern Recognition, Elsevier, New York, 1974.

(13) Young, T.Y. and Liu, P.S., "Impact of VLSI on Pattern Recognition and Image Processing", in VLSI Electronics: Microstructure Science, vol. 4, N.G. Einspruch, Ed., Academic, New York, 1982, pp. 319-360.

(14) Hwang, K. and Su, S.P., "A Partitioned Matrix Approach to VLSI Pattern Classification", Proc. Workshop on Comp. Arch. for Pattern Analysis and Image Database Management, Nov. 1981, pp. 168-171.

(15) Liu, H.H. and Fu, K.S., "VLSI Algorithm for Mimimum Distance Classification", Proc. IEEE Int. Conf. on Computer Design: VLSI in Computers, Oct. 1983.

(16) Young, T.Y., Liu, P.S. and Gao, Y., "Reconfigurable VLSI Arrays for Pattern Analysis and Image Processing", Proc. Workshop on Comp. Arch. for Pattern Analysis and Image Database Management, Oct. 1983.

(17) Westlake, J.R., A Handbook of Numerical Matrix Inversion and Solution of Linear Equations, Wiley, New York, 1968.

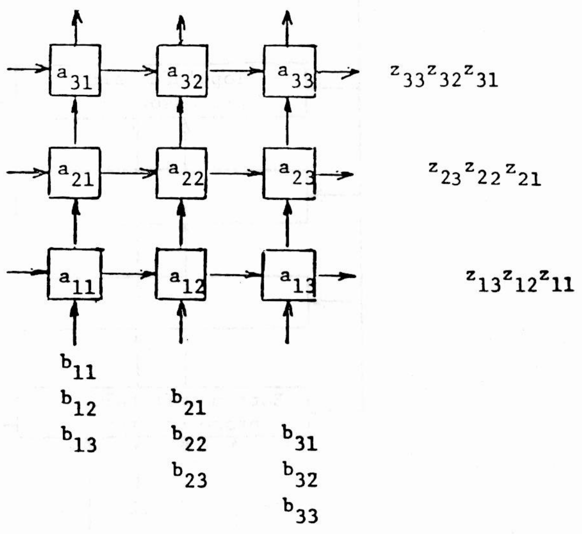


Fig. 1 - Data flow of multiplication array

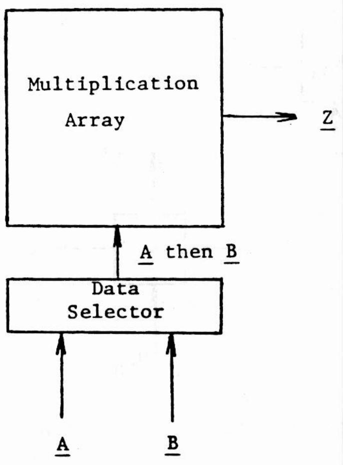


Fig. 2 - Block diagram of multiplication array with separate source matrices

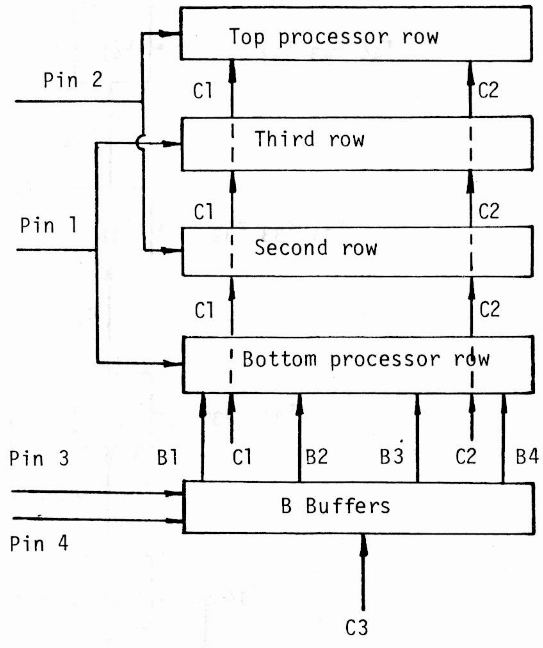
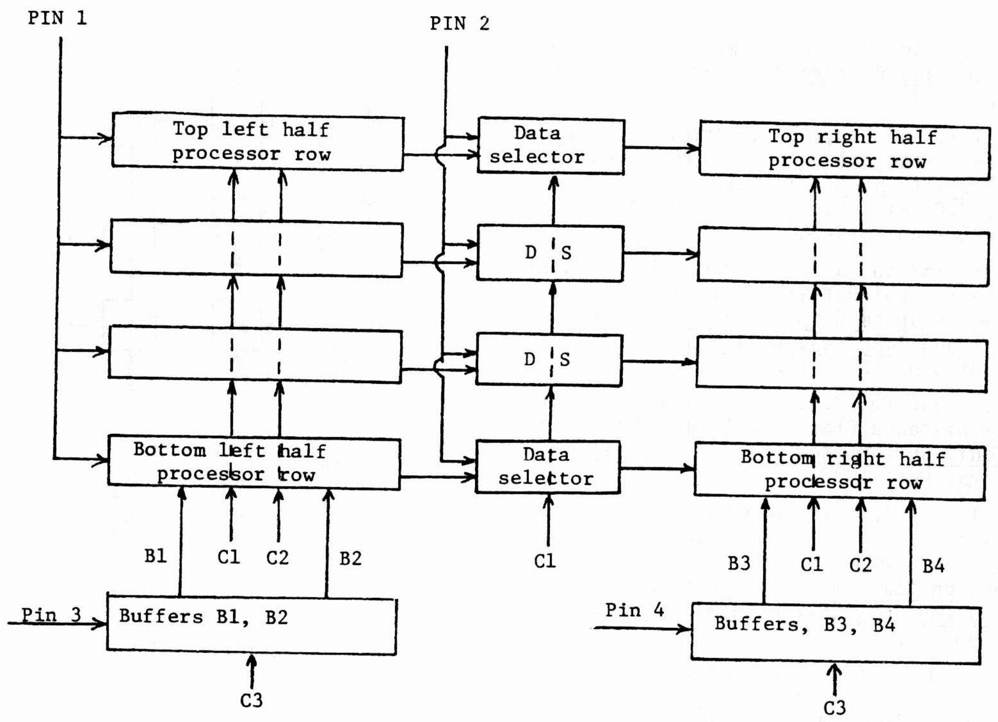


Fig 3 - Block diagram of processor row loading scheme



PIN 1
PIN 2
Top left half processor row
Data selector
Top right half processor row
D S
D S
Bottom left half processor row
Data selector
Bottom right half processor row
B1
C1
C2
B2
C1
B3
C1
C2
B4
Pin 3
Buffers B1, B2
Pin 4
Buffers, B3, B4
C3
C3

Fig. 4 - Block diagram of processor column group loading scheme

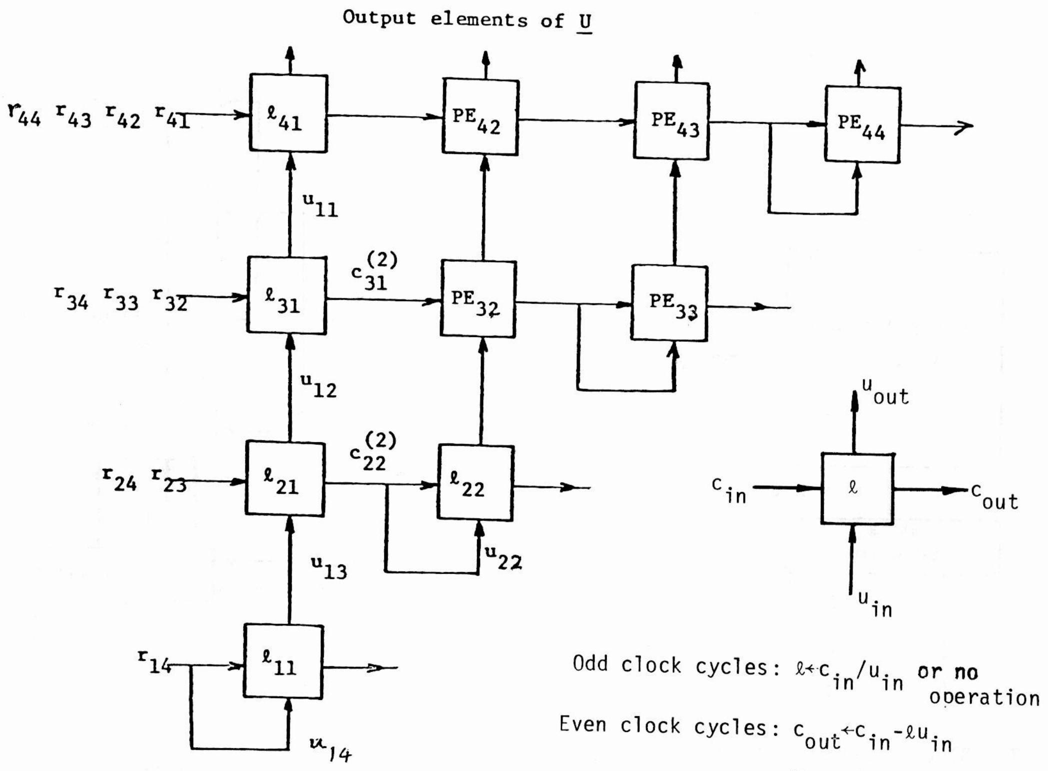


Fig. 5 - Triangular array for decomposition of covariance matrix