ACT-23 APRIL, 1980

CSL COORDINATED SCIENCE LABORATORY

APPLIED COMPUTATION THEORY GROUP

AREA-TIME OPTIMAL VLSI NETWORKS FOR MATRIX MULTIPLICATION AND INVERSION OF TRIANGULAR MATRICES

F. P. PREPARATA
J. E. VUILLEMIN

APPROVED FOR PUBLIC RELEASE. DISTRIBUTION UNLIMITED.

REPORT R-879 UILU-ENG 80-2211

UNIVERSITY OF ILLINOIS - URBANA, ILLINOIS

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
AREA-TIME OPTIMAL VLSI NETWORKS FOR MATRIX MULTIPLICATION AND INVERSION OF TRIANGULAR MATRICES

5. TYPE OF REPORT & PERIOD COVERED
Technical Report

6. PERFORMING ORG. REPORT NUMBER
R-879 (ACT-23);UILU-ENG80-2211

7. AUTHOR(s)
F. P. Preparata
J. E. Vuillemin

8. CONTRACT OR GRANT NUMBER(s)
MCS-78-13642; N00014-79-C-0424.

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
National Science Foundation
Joint Services Electronics Program Contract

12. REPORT DATE
April, 1980

13. NUMBER OF PAGES
23

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)
VLSI, matrix multiplication, triangular matrices, matrix inversion, optimal networks, area-time complexity, pipeline computation.

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

This report consists of two papers describing networks for parallel matrix operations. In the first paper, "Area-time optimal VLSI networks for multiplying matrices," we describe a class of VLSI networks having chip area A for multiplying two $n \times n$ matrices in time T, with an area $\times$ time^2 product $A \cdot T^2 = 0(n^4)$. These networks achieve Savage's [1] lower bound to this complexity measure for any T such that $\log n \leq T \leq n$. The second paper, "Optimal integrated-circuit implementation of triangular matrix inversion," describes a class of VLSI implementations of algorithms for inverting an $n \times n$ triangular matrix. These networks have area A and time T, with an area × time^2 product $AT^2 = O(n^4)$ for all values of T such that $O(\log^2 n) \leq T \leq O(n)$. Since there is a simple reduction of matrix multiplication to inversion of a triangular matrix, due to Savage's result, the presented networks are optimal in the VLSI model.

AREA-TIME OPTIMAL VLSI NETWORKS FOR MATRIX MULTIPLICATION AND INVERSION OF TRIANGULAR MATRICES

by

F. P. Preparata and Jean Vuillemin

This work was supported in part by the National Science Foundation under Grant MCS 78-13642 and Joint Services Electronics Program (U.S. Army, U.S. Navy, and U.S. Air Force) under Contract N00014-79-C-0424.

This report is issued simultaneously by the Coordinated Science Laboratory and by the Institut National de Recherche d'Informatique et d'Automatique, 78150 Rocquencourt, France.

AREA-TIME OPTIMAL VLSI NETWORKS FOR MATRIX MULTIPLICATION AND INVERSION OF TRIANGULAR MATRICES[†]

F. P. Preparata[*] and Jean Vuillemin[**]

Abstract

This report consists of two papers describing networks for parallel matrix operations. In the first paper, "Area-time optimal VLSI networks for multiplying matrices," we describe a class of VLSI networks having chip area A for multiplying two n × n matrices in time T, with an area × time^2 product $A \cdot T^2 = O(n^4)$. These networks achieve Savage's lower bound to this complexity measure for any T such that $\log n \leq T \leq n$. The second paper, "Optimal integrated-circuit implementation of triangular matrix inversion," describes a class of VLSI implementations of algorithms for inverting an n × n triangular matrix. These networks have area A and time T, with an area × time^2 product $AT^2 = O(n^4)$ for all values of T such that $O(\log^2 n) \leq T \leq O(n)$. Since there is a simple reduction of matrix multiplication to inversion of a triangular matrix, due to Savage's result the presented networks are optimal in the VLSI model.

Keywords: VLSI, matrix multiplication, triangular matrices, matrix inversion, optimal networks, area-time complexity, pipeline computation.

[†] This work was partially supported by National Science Foundation Grant MCS-78-13642, and by the Joint Services Electronics Program Contract N00014-79-C-0424, and by ERA 452 "Al Khowarizmi", Centre National de la Recherche Scientifique, France.

[*] Coordinated Science Laboratory, University of Illinois, Urbana, Illinois 61801.

[**] Laboratoire de Recherche en Informatique, Bât. 490, Université de Paris-Sud, 91405 Orsay.

AREA-TIME OPTIMAL VLSI NETWORKS FOR MULTIPLYING MATRICES[†]

Franco P. Preparata[*] and Jean E. Vuillemin[**]

Abstract: We describe a class of VLSI networks having chip area A for multiplying two n × n matrices in time T, with an area × time^2 product $A \cdot T^2 = O(n^4)$. These networks achieve Savage's [1] lower bound to this complexity measure for any T such that $\log n \leq T \leq n$.

Keywords: VLSI, matrix multiplication, optimal networks, area-time complexity, pipeline computation.

[†] This work was partially supported by National Science Foundation Grant MCS-78-13642, and by the Joint Services Electronics Program Contract N00014-79-C-0424, and by ERA 452 "Al Khowarizmi", Centre National de la Recherche Scientifique, France.

[*] Coordinated Science Laboratory, University of Illinois, Urbana, Illinois 61801.

[**] Laboratoire de Recherche en Informatique, Bât. 490, Université de Paris-Sud, 91405 Orsay.

1. Introduction

Savage [1] has shown that, under reasonable assumptions reflecting current VLSI technology [2,3,4], the designer of any circuit for multiplying two n × n matrices is confronted with a tradeoff between chip area A and computation time T expressed by $A \cdot T^2 = \Omega(n^4)$. Designs of Kung-Leiserson [5] meet that bound for $T = O(n)$ and minimal area $A = O(n^2)$.

The purpose of this note is to illustrate a general design scheme of VLSI networks for multiplying two n × n matrices. According to this scheme, a network with optimal value of the statistics AT^2 can be designed for any value of T[1] in the range $[\log n, n]$.[2] The scheme makes use of a recursively defined matrix multiplier - to be described next - and implements pipelining in an efficient way.

2. A Straightforward Matrix Multiplier

Let $U = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ and $V = \begin{pmatrix} e & g \\ f & h \end{pmatrix}$ be two s × s matrices. We present a network for computing the matrix product U × V in a recursive fashion, assuming that we know how to construct circuits for multiplying two $\frac{s}{2} \times \frac{s}{2}$ matrices, such as a, b, c, d, e, f, g, h.

The layout for such a rectangular network in figure 1, where one finds 8 recursively defined multipliers; lines drawn represent bundles of $s^2/4$ wires, corresponding to the parallel transmission of full $\frac{s}{2} \times \frac{s}{2}$ submatrices; the network comprises 4 matrix adders, which are placed in the $\frac{s^2}{4} \times \frac{s^2}{4}$ area occupied by the intersection of two bundles of wires as shown in figure 2.

[1] Computation time T is expressed in units, which also reflect the current state of technology.

[2] All logarithms in this paper are to the base of 2.

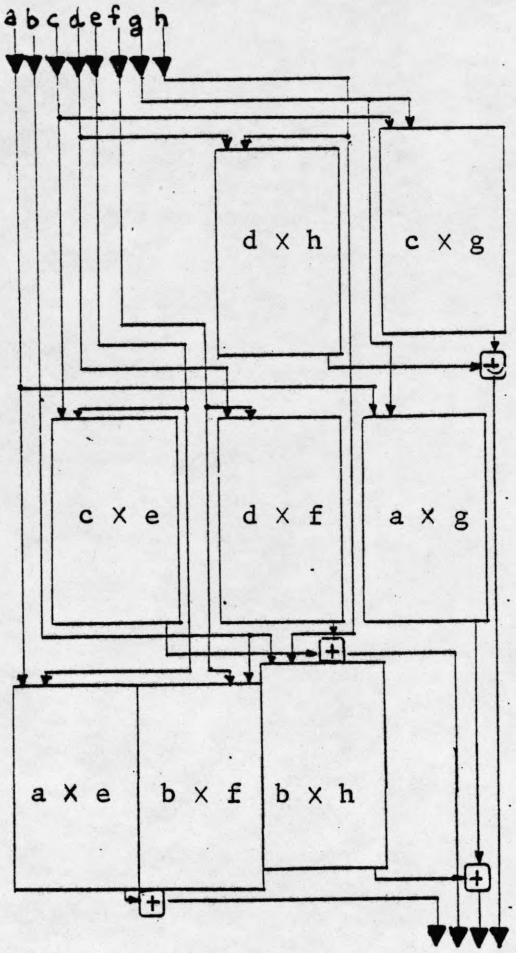


Figure 1. Layout of the recursive matrix multiplier.

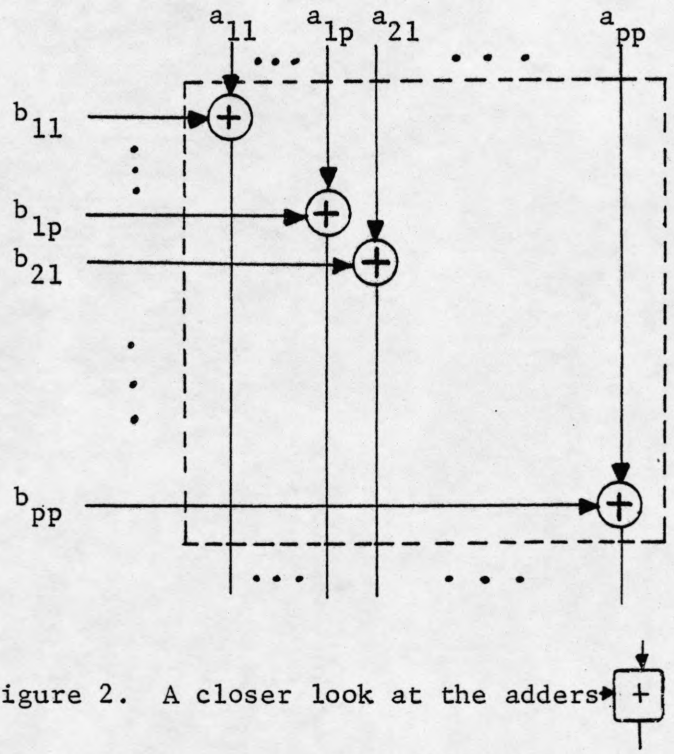


Figure 2. A closer look at the adders $\boxed{+}$ of figure 1 (p = s/2).

For specificity, we assume the matrix elements to be drawn from a finite ring, so that an elementary finite chip can be used for multiplying and adding elements in that ring in constant time and area. Our recursive definition stops at $s = 1$, where we use the elementary circuit for ring multiplication.

The width $w(s)$ of our rectangular network satisfies the recurrence

$$w(s) = 6\,\frac{s^2}{4}\,\lambda + 3w(\tfrac{s}{2}) \;, \qquad w(1) = w_1 \;,$$

where λ is the wire width as in [2] or [4], and w_1 the width of the elementary multiplier. (We assume elementary adders to have width and height λ.) The solution to this recurrence is (where we have set $s = 2^P$):

$$w(2^P) = 6(4^P - 3^P)\lambda + 3^P w_1, \text{ thus } w(s) = 0(s^2)\cdot\lambda + 0\left(s^{\log 3}\right)\cdot w_1 \;.$$

Similarly, the height $h(s)$ of our circuit satisfies $h(s) = 11\,\frac{s^2}{4}\cdot\lambda + 3h(\frac{s^2}{4})$, $h(1) = h_1$, where h_1 is the height of the elementary multiplier. The exact solution is given by $h(2^P) = 11\,(4^P - 3^P)\lambda + 3^P h_1$; thus $h(s) = 0(s^2)$ and the total chip area $A = h \times w = 0(s^4)$.

Notice that the network consists of $2\lceil\log s\rceil + 1$ levels, so subdivided: the first $\lceil\log s\rceil$ levels are buffer-drivers, whose only purpose is to construct two copies of their input; next, there is a single level of elementary multipliers, followed by $\lceil\log s\rceil$ levels of adders. Therefore the computation time of the network just described is $T(s) = \lceil\log s\rceil\cdot t_c + t_m + \lceil\log s\rceil\, t_a$, where t_c, t_m, and t_a are respectively the times for copying, multiplying, and adding, for a total of $T(s) = 0(\log s)$. Furthermore, it is important to point out that at any time before the end of the computation all intermediate results are <u>on one and the same level</u> of the matrix multiplier, which is therefore ideally suited for pipelined operation. This straightforward scheme, when used for multiplying $n \times n$

matrices, has area $A = O(n^4)$ and time $T = O(\log n)$, and thus does not yet meet the $AT^2 = \Omega(n^4)$ bound.

3. Pipelining Strategies

Consider now two n × n matrices A and B, and decompose each of them into r^2 blocks of size $\frac{n}{r} \times \frac{n}{r}$. Specifically for A we obtain

$$A = \begin{bmatrix} A_{11} & A_{12} & \dots & A_{1r} \\ A_{21} & A_{22} & \dots & A_{2r} \\ \vdots & & & \\ A_{r1} & A_{r2} & \dots & A_{rr} \end{bmatrix}, \quad A_{ij} \text{ an } \frac{n}{r} \times \frac{n}{r} \text{ matrix}$$

and similarly for B.

If we define the n × n matrices

$$C_j = \begin{bmatrix} A_{1j} \\ A_{2j} \\ \vdots \\ A_{rj} \end{bmatrix} \begin{bmatrix} B_{j1} & B_{j2} & \dots & B_{jr} \end{bmatrix} \qquad (j = 1,2,\dots,r),$$

obviously $C = A \times B = C_1 + C_2 + \dots + C_r$. Thus the product of A and B can be obtained by "accumulating" the matrices $C_1,\dots,C_r$. We now show how the calculation of these matrices and their accumulation can be pipelined.

The scheme is a classical one-way pipelining application (see [6], ch. 9). Consider the r × r square array of modules shown in figure 3. For the time being we assume that each module has a register c and receives two operands a and b (a on its "west" input and b on its "north" input); it performs the operation $c \leftarrow a \times b$, and passes on a to the "east" and b to the "south". Here a, b, and c are $(n/r) \times (n/r)$ matrices.

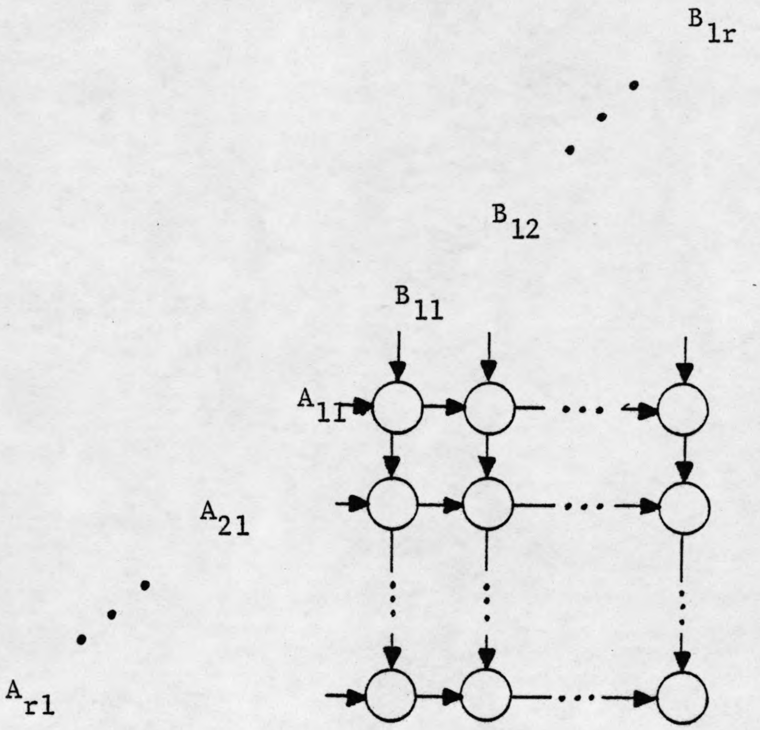
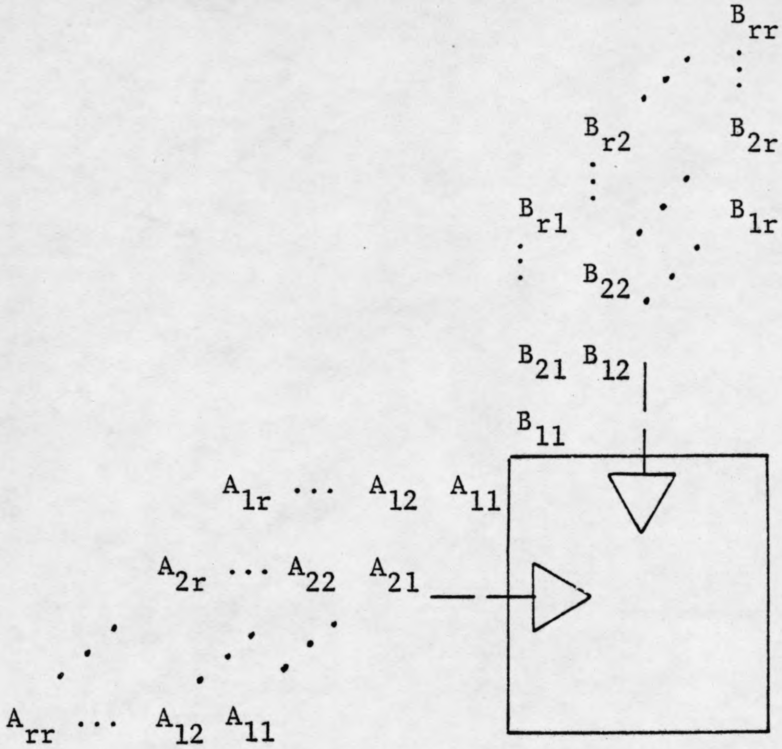


Figure 3. General layout of the network.

To compute C_1, we feed $[A_{11}\ A_{21}\ \cdots\ A_n]^T$ and $[B_{11}\ B_{12}\ \cdots\ B_{1r}]$ with the timing as shown in figure 3 at uniform speed. At each step the front of the moving data lies on a secondary diagonal of the array; the modules on this diagonal compute a product and store it, and it is obvious that after (2r-1) such steps the matrix C_1 is stored in the module registers. Since at each step only one diagonal of the array is active, the pipelining can be naturally implemented. Specifically, A is pipelined from left to right and B from top to bottom; each module is now to be viewed as an inner product processor, which executes $c \leftarrow c + ab$, thereby accumulating the matrix $C = A \times B$ (see figure 4).



B_{rr}
B_{r2}
B_{2r}
B_{r1}
B_{1r}
B_{22}
B_{21} B_{12}
B_{11}
A_{1r} ... A_{12} A_{11}
A_{2r} ... A_{22} A_{21}
A_{rr} ... A_{12} A_{11}

Figure 4. Pipelining scheme.

The structure of the inner product module is shown in figure 5. It is easily realized that the module, which incorporates an s × s multiplier of the type described in Section 2, with $s = n/r$, has width and height both $O(n^2/r^2)$.

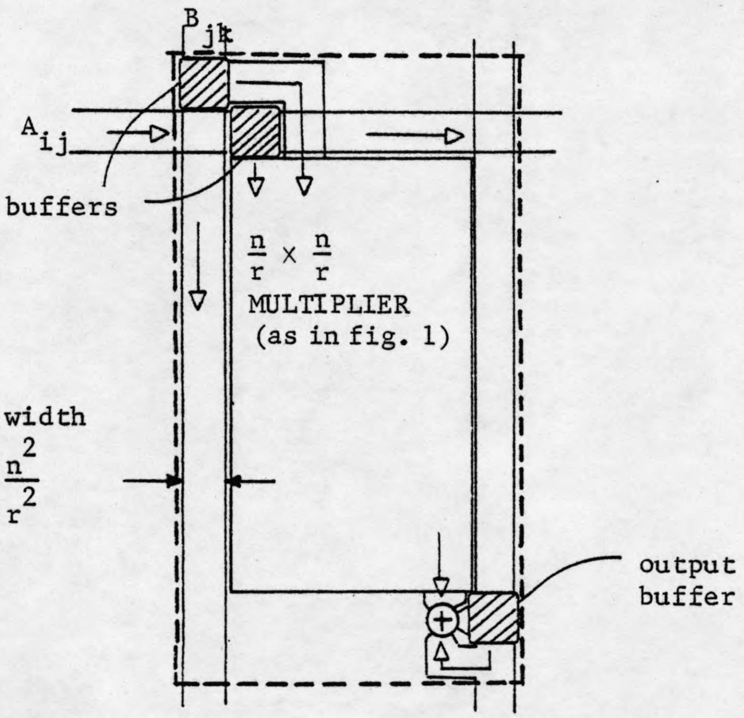


Figure 5. Structure of the inner product module.

The computation of the product matrix C is effected by a serial pipelining both through the array and the multipliers; since the former has r levels and the latter has $O(\log(n/r))$ levels, the matrix C is available after time $T = O(r+\log(n/r))$, and can be shifted out in either row-or column-major order (shifting of the output actually may begin even before the last block of C has been computed).

To evaluate the performance of the scheme, we note that the array consists of a compact mesh of $r \times r$ modules, each of which has area $O(n^4/r^4)$. Thus the network area is

$$A = O\left(\frac{n^4}{r^2}\right)$$

As for the time T, we have just shown that

$$T = O(r+\log \frac{n}{r}).$$

Combining these two results we obtain

$$AT^2 = O\left(\frac{n^4}{r^2}\right) \times r^2 \times O((1 + \frac{1}{r} \log \frac{n}{r})^2)$$

For all values of T in the range [logn,n], the term $O((1 + \frac{1}{r} \log \frac{n}{r})^2)$ is bounded by a constant, whence

$$AT^2 = O(n^4)$$

thereby substantiating our earlier claim.

Remark. There is another pipelining strategy - apparently unrelated to the preceding one - which also allows us to meet Savage's bound, but for a smaller interval of computation times T. This strategy is worth reporting.

Again, regard an $n \times n$ matrix as an $n/r \times n/r$ matrix whose elements are $r \times r$ blocks. Using a single $n/r \times n/r$ matrix multiplier of the type displayed in figure 1, we adopt the following pipelining scheme:

1) the r^2 elements of each block for both matrices A and B are fed serially on one input line (there are $(n/r)^2$ such lines);
2) the elementary multiplier (see Section 2) must now have the capability of performing an $r \times r$ matrix multiplication. Such elementary multiplier could be, for example, a hexagonal mesh of the Kung-Leiserson [5] type, with area $O(r^2)$ and time $O(r)$. The operands arrive serially, must be stored, then processed, and the result finally must be released serially; obviously the arrival and release pipelining times $O(r^2)$ predominate over the multiplication time $O(r)$.
3) The result is $n/r \times n/r$ matrix, whose elements are $r \times r$ blocks which appear serially on each output line.

We evaluate the performance of the scheme. As long as $r^2 \geq O(\log n)$ we have for the computation time that $T = O(r^2)$. As to the area, we note that the width w^* of the network is given by

$$w^* = O\left(\frac{n^2}{r^2}\right) + O\left(\left(\frac{n}{r}\right)^{\log 3}\right) \cdot w_{elem}$$

where w_{elem}, the width of the elementary multiplier is now $O(r)$. It follows that, as long as $r \leq O(n^{(2-\log 3)/(3-\log 3)})$ we have $w^* = O(n^2/r^2)$; a similar results holds for the height h^* of the network, whence $A = O(n^4/r^4)$. We conclude that

$$AT^2 = O(n^4)$$

(i.e., it is optimal) for all values of T such that $O(\log n) \leq T \leq O(n^{0.58})$.

References

1. J. E. Savage, "Area-time tradeoffs for matrix multiplication and transitive closure in the VLSI model," Proc. of the 17th Annual Allerton Conference on Communications, Control and Computing, Oct. 1979.

2. C. Mead, L. Conway, Introduction to VLSI Systems. Addison-Wesley, Reading, Mass. 1980.

3. R. P. Brent, H. T. Kung, "The area-time complexity of binary multiplication," Research Report CMU-CS-79-136, Carnegie-Mellon University, Pittsburgh, Penn., Sept. 1979.

4. C. D. Thompson, "A complexity theory for VLSI," Ph.D. Thesis, Dept. of Computer Science, Carnegie-Mellon University, Pittsburgh, Penn., Sept. 1979.

5. H. T. Kung, C. E. Leiserson, "Algorithms of VLSI processor arrays," Symposium on Sparse Matrix Computations, Knoxville, Tenn., Nov. 1978.

6. T. C. Chen, Overlap and Pipeline Processing, pp. 375-431 in Introduction to Computer Architecture, (H. S. Stone, editor), Science Research Associates, 1975.

OPTIMAL INTEGRATED-CIRCUIT IMPLEMENTATION OF TRIANGULAR MATRIX INVERSION[†]

Franco P. Preparata[*] and Jean Vuillemin[**]

Abstract

We describe a class of integrated-circuit implementations of algorithms for inverting an n × n triangular matrix. These networks have area A and time T, with an area × time^2 product $AT^2 = O(n^4)$ for all values of T such that $O(\log^2 n) \leq T \leq O(n)$. Since there is a simple reduction of matrix multiplication to inversion of a triangular matrix, and Savage [6] has given an $AT^2 = \Omega(n^4)$ lower-bound for n × n matrix multiplication, the presented networks are optimal in the VLSI model.

Keywords: VLSI, matrix inversion, triangular matrices, area-time complexity, pipeline computation, optimal networks

[†] This work was partially supported by National Science Foundation Grant MCS-78-13642, and by the Joint Services Electronics Program Contract N00014-79-C-0424, and by ERA 452 "Al Khowarizmi", Centre National de la Recherche Scientifique, France.

[*] Coordinated Science Laboratory, University of Illinois, Urbana, Illinois 61801.

[**] Laboratoire de Recherche en Informatique, Bât. 490, Université de Paris-Sud, 91405 Orsay.

1. Introduction

Increasing attention has been paid recently to the design of networks for the direct implementation of several interesting algorithms using the integrated-circuit technology (VLSI); particularly, combinatorial and numerical problems have been the target of these investigations [1-4]. Among numerical problems, several workers have directed their attention to matrix computations [1,2,5], and, as regards the design of networks, have found that the mesh interconnection of computing modules is particularly attuned to this class of problems, leading to optimal realizations [5,6] in the VLSI model [7,8].

In this paper we consider the problem of designing VLSI networks for inverting a nonsingular triangular matrix. The design complies with specifications of the VLSI model of computation recently proposed by Mead, Conway, and Thompson [7,8]. In this model, the network is a computation graph consisting of nodes (processing modules) and wires. Wires have unit width and are partitionable into two orthogonal sheaves. A data item takes a unit of time to propagate along a wire from node to node (processing time is thus absorbed into propagation time). The usual complexity metric is the area × time^2 product (AT^2), which embodies a trade-off between production cost (chip area A) and incremental cost (time T).

Within this model, Savage [6] has recently proved the following interesting result: any VLSI design for the multiplication of two $n \times n$ matrices, with chip area A and computation time T, must satisfy the lower bound $AT^2 \geq C\, n^4$, for some constant C. In [5] the authors demonstrate the existence of VLSI networks for multiplying $n \times n$ matrices with $AT^2 = O(n^4)$ for any computation time T within the bounds $\log n \leq T \leq n$. Note that an $AT^2 \geq Cn$ bound also holds for the problem of inverting a nonsingular $n \times n$ triangular matrix, since matrix multiplication is reducible to it; the straightforward reduction is based on the fact that the inverse of the $3n \times 3n$ triangular matrix

$$\begin{bmatrix} I & A & 0 \\ 0 & I & B \\ 0 & 0 & I \end{bmatrix} \qquad \text{is} \qquad \begin{bmatrix} I & -A & AB \\ 0 & I & -B \\ 0 & 0 & I \end{bmatrix}$$

i.e., it contains an $n \times n$ block equal to the product AB.

This paper is organized as follows: In Section 2 we review the general scheme for inverting an $n \times n$ triangular matrix, and evaluate two network implementations, corresponding respectively to block-partitioning the matrix and choosing extreme values for the block size in the allowable range. These two inverters are referred to as "recursive and "systolic" respectively; with respect to the AT^2 statistics only the latter is optimal for $T = O(n)$. In Section 3 we show that the recursive and systolic inverters can be combined to build networks, called "mixed" inverters, which realize the $AT^2 = \Omega(n^4)$ lower bound for all values of T such that $O(\log^2 n) \leq T \leq O(n)$.

2. The general scheme for inverting a nonsingular triangular matrix.

Let A be a nonsingular n × n triangular matrix,[1] to be thought of as an n/s × n/s matrix whose elements are s × s blocks of the original entries (s is a parameter in the range [1, n/2]); let A_{ij} be the (i,j) block of A(i,j = 1,2,...,n/s) and let $A_{ij}^{(-1)}$ be the corresponding block of A^{-1}. It is well known — and also straightforward to verify — that

$$A_{ij}^{(-1)} = -[A_{ii}^{(-1)}, A_{i,i+1}^{(-1)}, \ldots, A_{i,j-1}^{(-1)}] \cdot \begin{bmatrix} A_{ij} \\ A_{i+1,j} \\ \cdot \\ \cdot \\ \cdot \\ A_{j-1,j} \end{bmatrix} \cdot A_{jj}^{(-1)} . \qquad (1)$$

This general formula will now be specialized to two interesting cases.

2.1 Recursive inversion

The standard scheme for the parallel inversion of a triangular matrix [9,10] corresponds to specializing the general scheme to s = n/2. In this case the inverse of

$$\begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \quad \text{is} \quad \begin{bmatrix} A_{11}^{-1} & -A_{11}^{-1}A_{12}A_{22}^{-1} \\ 0 & A_{22}^{-1} \end{bmatrix} . \qquad (2)$$

This immediately suggests a recursively defined network, containing two inverters of n/2 × n/2 triangular matrices (to be used to compute A_{11}^{-1}

[1] The entries of all matrices considered in this paper are assumed to be drawn from a finite ring, so that an elementary finite chip can be used for multiplying and adding entries in constant area and time.

and A_{22}^{-1} in parallel) and a network for the parallel multiplication of two $n/2 \times n/2$ matrices (to be used to compute $(A_{11}^{-1}A_{12})A_{22}^{-1}$ in the order shown by the parenthesization). In figure 1 we show a possible layout for such a network. Each line shown carries $n^2/4$ operands in parallel

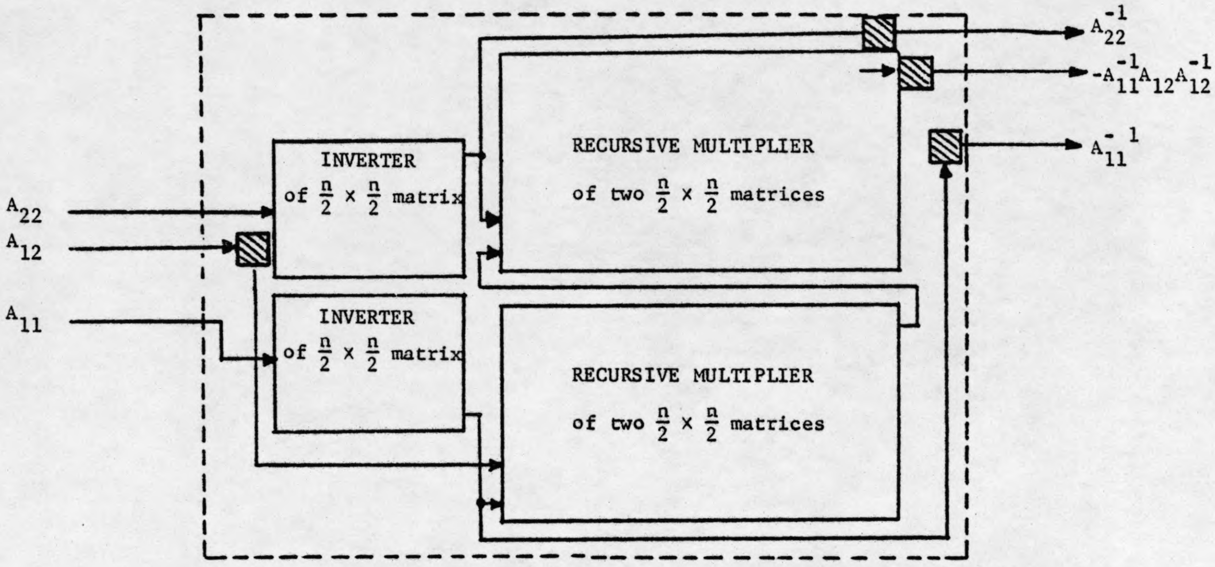

A_{22}^{-1}
$-A_{11}^{-1}A_{12}A_{12}^{-1}$
A_{11}^{-1}
INVERTER
of $\frac{n}{2} \times \frac{n}{2}$ matrix
RECURSIVE MULTIPLIER
of two $\frac{n}{2} \times \frac{n}{2}$ matrices
A_{22}
A_{12}
A_{11}
INVERTER
of $\frac{n}{2} \times \frac{n}{2}$ matrix
RECURSIVE MULTIPLIER
of two $\frac{n}{2} \times \frac{n}{2}$ matrices

Figure 1. Layout of the recursive matrix inverter; Shaded boxes are data buffers.

and the shaded surfaces are buffers of capacity $(n^2/4)$; the core of the circuit are two multipliers of two $(n/2) \times (n/2)$ matrices, of a type described in [5], and called <u>recursive multipliers</u>. Each of these multipliers has height and width bounded respectively by $(6/4)n^2$ and $(11/4)n^2$ and computes a matrix product in 2logn-1 time units. Due to the recursive definition of the inverter, a simple argument shows that its height and width are respectively bounded by $(15/4)n^2$ and $(15/3)n^2$; also, the computation time is $0((\log n)^2)$. Notice therefore that for the matrix inverter being described — called <u>recursive inverter</u> — we have the following properties (referred to are $n \times n$ matrix):

Type	A	T	AT^2
Recursive inverter	$O(n^4)$	$O(\log^2 n)$	$O(n^4 \log^4 n)$

(3)

Note that AT^2 is short of the optimal value $\Omega(n^4)$.

2.1 Systolic inversion

The next scheme to be described corresponds to the choice $s=1$ in the general method. The resulting network is a mesh of processors, each of which feeds data in and out, each time performing some computation, keeping a regular flow in the network. Such networks have been called systolic by Kung and Leiserson [1].

With our choice of s, block A_{hk} in (1) becomes entry a_{hk} (and similarly $A_{hk}^{(-1)}$ becomes $a_{hk}^{(-1)}$). The form of (1) suggests a computation method on an $n \times n$ square mesh (figure 2). Only the upper-triangular positions in this mesh need contain processing modules (i.e., denoting by M_{ij} the module in position (i,j), M_{ij} is deployed only for $j \geq i$). Modules are of two types with different computational capabilities: D-modules and M-modules, placed respectively in diagonal and off-diagonal positions.

Entry $a_{ij}^{(-1)}$ of A^{-1} will be computed in place in M_{ij}. For i = j

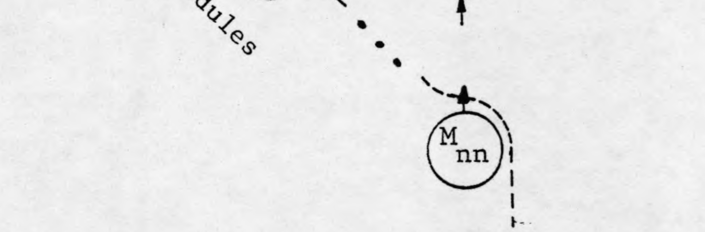


Figure 2. General structure of the systolic matrix inverter (triangular mesh)

(diagonal entry), the corresponding D-module must invert entry a_{ii}, i.e., $a_{ii}^{(-1)} = 1/a_{ii}$; for $j > i$, the process is more complex and is to be analyzed.

Each operation — inversion of an entry or multiplication of two entries — is conventionally assumed to take one "step". Assume inductively that for fixed t and $2k-h < t$, the computation of $a_{hk}^{(-1)}$ be completed in M_{hk} at the end of the (2k-h)-th step; the basis for this induction is easily established for $h = k$, in which case we just activate M_{hh} at the h-th step. We now extend the induction by showing that a_{ij} with $2j-i = t$, can be computed at the end of the t-th step. Indeed, $a_{ip}^{(-1)}$ $(p < j)$ is computed, by the inductive hypothesis, at the end of the (2p-i)-th step; after this computation is completed, $a_{ip}^{(-1)}$ is shifted to the right along the i-th row of the mesh (figure 2) (one position per step), so that $a_{ip}^{(-1)}$ resides in M_{ij} at the end of the $[(2p-i)+(j-p)]$-th step. Similarly entry a_{pj} $(p < j)$ is shifted upwards along the j-th column of the mesh of figure 1 (one position per time step) starting at step $(j+1)$, so that a_{pj} resides in M_{ij} at step $(p-i+j)$, simultaneously with $a_{ip}^{(-1)}$. Thus, the product $a_{ip}^{(-1)} \cdot a_{pj}$ can be computed in the step $(p-i+j)$, and accummulated in M_{ij}. This shows that the inner product $[a_{ii}^{(-1)},\ldots,a_{i,j-1}^{(-1)}] \cdot [a_{ij},\ldots,a_{j-1,j}]$ resides in M_{ij} at the end of step $(j-1)-i+j = 2j-i-1$. Now, recall that,by the inductive hypothesis, $a_{jj}^{(-1)}$ is computed in M_{jj} at step j; therefore it can be shifted upwards in the mesh and after (j-i) steps (i.e., at step $t = 2j-i$) it arrives in M_{ij},where the computation of $a_{ij}^{(-1)}$ is completed at the end of the t-th step, as was originally claimed.

For clarity, in figure 3(a) we illustrate the timing of the computations: Each module is labelled with an integer which denotes the step at which computation in that module is completed. Also, in figure 3(b and c) we present snapshots of the data participating in the horizontal and vertical flow, respectively, at step 7. Clearly the calculation of A^{-1} is completed in 2n-1 steps.

As regards implementation details,

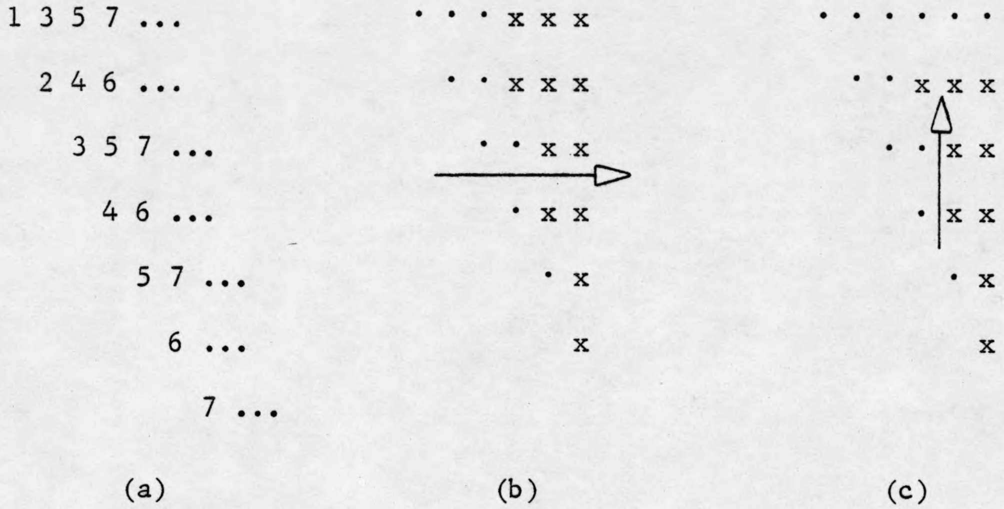


Figure 3 (a): timing of completion of computation up to step 7.
(b): data (x) participating in horizontal flow at step 7.
(c): data (x) participating in vertical flow at step 7.

initially module M_{ij} is loaded with a_{ij}. While M_{ii} $(i = 1,\dots,n)$ must simply compute $1/a_{ii}$, an M-module M_{ij} $(i \neq j)$ can be designed with one operand register R and two buffers H and V; there are two operand input lines W and S, and two operand output lines, E and N (see figure 4). Buffers H and V constantly feed output lines E and N, respectively and the module must be capable of executing the following instructions:

1. $R \leftarrow R + W \cdot S$, $H \leftarrow W$, $V \leftarrow S$, for the general step.

2. $R \leftarrow -R \cdot S$, $H \leftarrow -R \cdot S$, $V \leftarrow S$, for the final step.

It is readily realized that this is all is needed to implement the described algorithm.

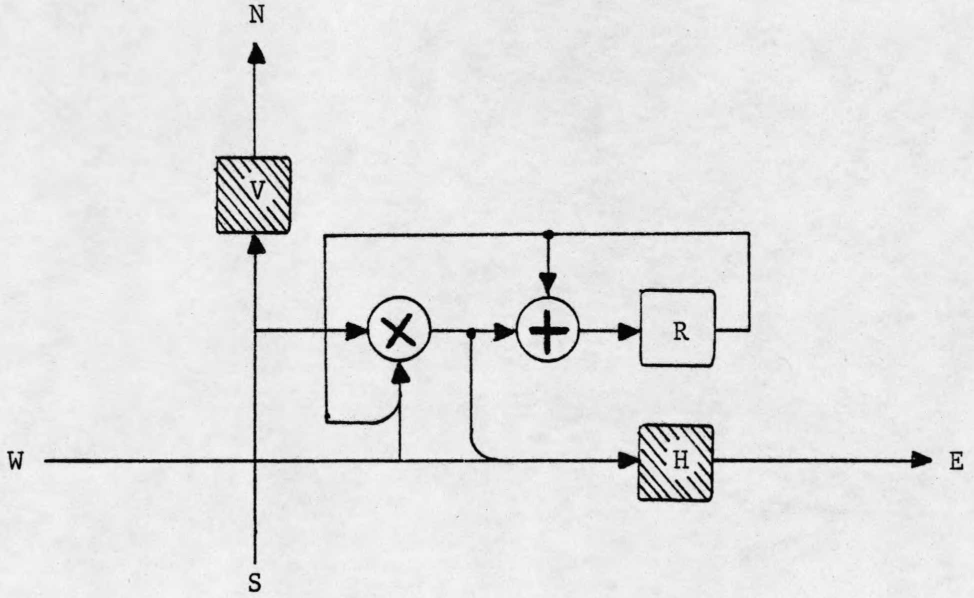


Figure 4. Structure of an M-module; H and V are buffers.

According to our original assumption that both the area of the processing modules and the time needed to execute any of the prescribed operations be bounded by a constant, we have the following:

Type	A	T	AT^2
1st-order systolic inverter	$O(n^2)$	$O(n)$	$O(n^4)$

, (4)

i.e., the network is optimal for the AT^2 measure. The optimal behavior, however, is achieved only for $T = O(n)$. An interesting question is whether it can be extended to a wider range of processing times. This question is addressed in the next section.

3. Mixed networks.

We now describe how to combine the recursive and systolic inverters described in the preceding section in order to improve the AT^2 measure for a wide range of the time parameter T.

The resulting networks - to be called mixed - have the following general structure. A mixed network is a systolic scheme, as the one described in 2.2, where the "operands", rather than being elementary entries, are blocks of s × s such entries. In the corresponding n/s × n/s triangular mesh (see figure 2), the modules must now be designed to process s × s blocks. The layout of mixed networks is chosen as in figure 3, where the modules themselves have been conveniently assumed to have a rectangular shape on the chip (else, we consider the smallest rectangle with sides parallel to the coordinate axes which contains the module). From figure 3, it is clear that while the dimensions (width and height) of the M-module determine one dimension of the network - say, its width -, the other dimension - say, its height - is determined by the larger of the corresponding values for the D- and M-modules.

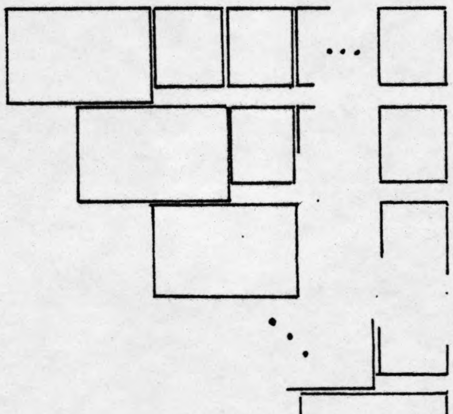


Figure 3. - General layout of mixed networks. Each line shown carries in parallel s^2 operands.

The first kind of mixed networks to be considered is one for which the following selections are made (Type-1 mixed inverter):

(1) D-modules are recursive inverters, as described in Section 2.1; they have width $5s^2$, height $(15/4)s^2$, and computation time $O(\log^2 s)$.

(2) M-modules are recursive matrix multipliers of the Preparata-Vuillemin type [5], as already used to build the recursive inverter. They can be placed on the chip so that their width and height are $2s^2$ and $(15/4)s^2$ (see [5] for details); their computation time is $O(\log s)$.

Since the heights of the D- and M-modules are of the same order (in this case, they are identical) the height of the network is $O(\frac{n}{s} \times s^2) = O(ns)$; since also the widths of the two modules are $O(s^2)$, the same holds for the width of the network. Thus, $A = O(n^2 s^2)$, and the smallest containing rectangle is nearly a square with both sides $O(ns)$. As regards computation time, the blocks A_{ii}^{-1} $(i = 1,\ldots,A/s)$ are all computed in time $O(\log^2 s)$, and after this, the mesh computation begins. We have observed in Section 2.2 that the systolic-network completes its computation in $O(n/s)$ steps, whence the total computation time is $T = O(\log^2 s + \frac{n}{s} \log s)$. If we bound the parameter s by $s \leq n/\log n$ we obtain $T = O(\frac{n}{s} \log s)$, and the performance of Type-1 networks is summarized as follows:

Type	A	T	AT^2
Type-1 mixed inverter const. $\leq s \leq n/\log n$	$O(n^2 s^2)$	$O(\frac{n}{s}\log s)$	$O(n^4 \log^2 s)$

(5)

The second kind of mixed networks (Type-2 mixed inverter) is constructed as follows:

(1) D-modules are type-1 mixed inverters (for s × s matrices). According to the preceding discussion, for any value of a parameter $r \leq s/\log s$, these modules have height and width both $O(sr)$ and computation time $O(\log^2 r + \frac{s}{r} \log r)$. Choosing $r = s/\log s$ we obtain height $O(s^2/\log s)$, width $O(s^2/\log s)$, and time $O(\log^2 s)$.

(2) M-modules are pipelined matrix multiplier, as introduced by Preparata and Vuillemin in [5]. It is shown in [5] that one such multiplier can be designed with height and width both $O(s^2/\log s)$ and computation time $O(\log s)$.

Again, the dimensions of both D-modules and M-modules are $O(s^2/\log s)$, whence:

$$A = O\left(\left(\frac{n}{s} \cdot \frac{s^2}{\log s}\right)^2\right) = O\left(\frac{n^2 s^2}{\log^2 s}\right) .$$

With respect to computation time, we obtain the same conclusions as for type-1 mixed inverters, i.e.,

$$T = \quad O(\log^2 s + \frac{n}{s} \log s) .$$

Therefore we obtain

$$AT^2 = O\left(\frac{n^2 s^2}{\log^2 s} \cdot (\log^2 s + \frac{n}{s} \log s)^2\right) \;\; = O\left(\frac{n^2 s^2}{\log^2 s} \cdot \frac{n^2}{s^2} \log^2 s (1 + \frac{s}{n} \log s)^2\right)$$

$$= O\left(n^4 \cdot (1 + \frac{s}{n} \log s)^2\right) \;.$$

Obviously, if $s \leq n/\log n$ we have $s\log s < n$, whence $AT^2 = O(n^4)$, and the performance of the Type-2 mixed inverter is so summarized:

Type	A	T	AT^2
Type-2 mixed inverter const. $\leq s < n/\log n$	$O \frac{n^2s^2}{\log^2 s}$	$O(\frac{n}{s} \log s)$	$O(n^4)$

Since as s varies from a small constant value to $n/\log n$ the computation time T varies from $O(n)$ to $O(\log^2 n)$, we say that the above network meets the $AT^2 = O(n^4)$ optimal bound for all T such that $O(\log^2 n) \leq T < O(n)$. Incidentally, even in totally unrestricted models of computation - as the shared-memory-machine [see, for example [10]] - $O(\log^2 n)$ is the smallest known running time for inverting a triangular matrix.

REFERENCES

1. H. T. Kung and C. E. Leiserson, "Algorithms for VLSI processor arrays," Symposium on Sparse Matrix Computations, Knoxville, Tenn., Nov. 1978.

2. L. J. Guibas, H. T. Kung, and C. D. Thompson, "Direct VLSI implementation of combinatorial algorithms," Proc. Conference on VLSI Architecture, Design, Fabrication, Calif. Inst. of Techn., January 1979.

3. R. P. Brent and H. T. Kung, "The area-time complexity of binary multiplication," Research Report, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Penn., July 1979.

4. C. D. Thompson, "Area-time complexity for VLSI," Proc. of the 11th Annual ACM Symposium on the Theory of Computing (SIGACT), pp. 81-88, May 1979.

5. F. P. Preparata and J. Vuillemin, "Area-time optimal VLSI networks for multiplying matrices", 14th Princeton Conference on Information Sciences and Systems, March 1980.

6. J. E. Savage, "Area-time tradeoffs for matrix multiplication and transitive closure in the VLSI model," Proc. of the 17th Annual Allerton Conference on Communications, Control, and Computing, October 1979.

7. C. Mead and L. Conway, Introduction to VLSI Systems, Addison-Wesley, Reading, Mass. 1979.

8. C. D. Thompson, "A complexity theory for VLSI", Ph.D. Thesis, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Penn., September 1979.

9. D. Heller, "A Survey of parallel algorithms in numerical linear algebra," Dept. of Comp. Sci. Carnegie-Mellon University, Pittsburgh, Pa., Feb. 1976.

10. L. Csanky, "Fast parallel matrix inversion algorithms, SIAM J. Comptng. 5, 1976, 618-623.