Consiglio Nazionale delle Ricerche

ISTITUTO DI ELABORAZIONE DELLA INFORMAZIONE

PISA

VLSI COMPUTATION OF THE INVERSE AND THE DETERMINANT OF A MATRIX

B. Codenotti, F. Romani

Nota interna B84-12

Settembre 1984

VLSI COMPUTATION OF THE INVERSE AND THE DETERMINANT OF A MATRIX

B. CODENOTTI[1] and F. ROMANI[1]

[1]Istituto di Elaborazione dell'Informazione - CNR, via S.Maria 46, PISA, ITALY.

ABSTRACT

In this paper, recursive VLSI designs for the inversion of nonsingular matrices and determinant computation are presented.

The VLSI network for the inversion requires a time $T = O(n \log n)$ and an area $A = O(n^2 \log^3 n)$, matching the area x (time)^2 lower bound up to logarithmic factors. The same complexity is attained by the network for the computation of the determinant.

Index Terms - VLSI Design, Area-Time Complexity, Matrix Inversion, Determinant computation.

I. INTRODUCTION AND PRELIMINARIES

In this paper, the VLSI inversion of a nonsingular matrix and the computation of the determinant of a given matrix are studied. While several optimal VLSI designs for matrix multiplication are known [PRE], seems to be more difficult to obtain area-time upper bounds close to the $\Omega(n^4)$ lower bound for matrix inversion [SAV]. A recursive formula for matrix inversion and determinant computation leads here to VLSI designs with operation time

$$T = O(n \log n) \quad \text{and area} \quad A = O(n^2 \log^3 n).$$

Let us recall some results about the structure of the inverse of a nonsingular matrix.

PROPOSITION 1.1

Let A be an nxn matrix partitioned as $A = \begin{bmatrix} U & V \\ W & Z \end{bmatrix}$, where U and Z are square matrices of size pxp and (n-p)x(n-p) respectively, with p<n. Moreover, assume A, U, Z to be nonsingular. Then the following relations hold

$$A^{-1} = \begin{bmatrix} C^{-1} & X \\ Y & D^{-1} \end{bmatrix}, \text{ where}$$

$$C = U - V Z^{-1} W, \qquad X = -U^{-1} V D^{-1}$$

(1.1)

$$D = Z - W U^{-1} V, \qquad Y = -Z^{-1} W C^{-1}.$$

Moreover

(1.2) $\quad \mathrm{Det}(A) = \mathrm{Det}(U)\ \mathrm{Det}(D).$

The proof follows by a straightforward application of LDR block factorization and Woodbury formula [HOU pp.123-127].

■

These results can be recursively applied with $p=\lceil n/2 \rceil$ to compute the determinant and the inverse of a nonsingular matrix A, provided that the minors U and Z are nonsingular at any level of recursion. Note that if A is positive definite, then U, Z, A^{-1} and hence C^{-1}, D^{-1} and C, D are positive definite and the algorithm can be applied without needing row or column permutations.

In general, one can use the relations:

$$A^{-1} = (A^T A)^{-1} A^T, \qquad \mathrm{Det}(A) = (\mathrm{Det}(A^T A))^{1/2},$$

in order to deal with the positive definite matrix $A^T A$.

For the sake of simplicity, assume A to be an nxn nonsingular positive definite matrix, with n power of two; equations (1.1) lead to the recursive design of fig.1, where modules denoted by I implement the inversion of size n/2, modules denoted by * and - perform respectively matrix multiplication and subtraction of size n/2.

The well known synchronous VLSI model of computation (see [THO]) is used; elementary scalar multiplication and addition are assumed to require the same fixed amount of time t/2.

The area-time complexity of this design depends on the

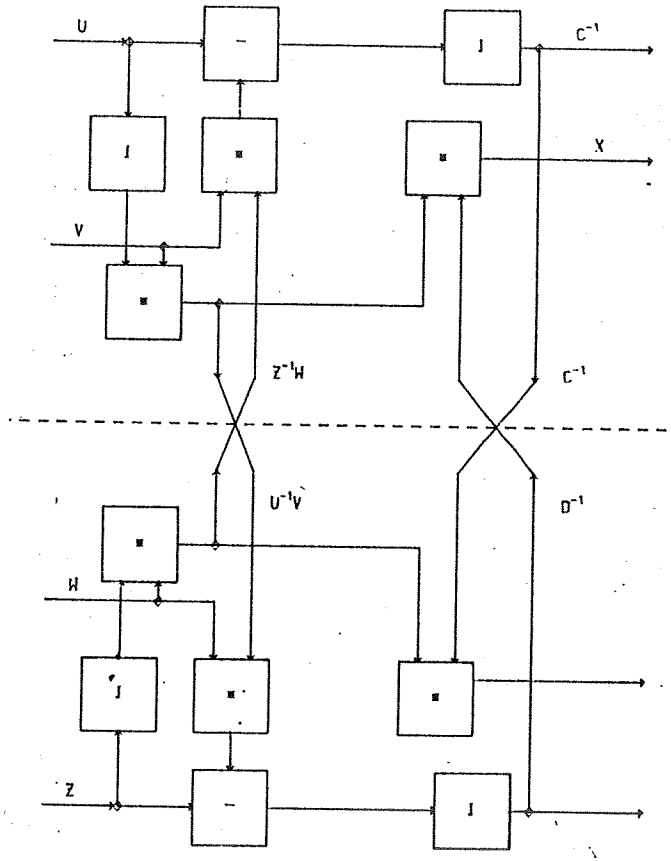

Fig.1 Recursive module structure.

choice of matrix multipliers and the corresponding layouts. A straightforward implementation uses n^2 wires for trasmitting a matrix and the mesh of tree structure [LEI] for matrix multiplication. In this case it is easy to show that the complexity is $T=O(n)$ and $A=O(n^4)$. A better result can be obtained by using smaller multipliers and a more complicated way of routing the elements of the matrices.

The following types of elementary processors are used (fig. 2).

a) Delay unit: introducing a delay of time t.

b) Buffer node: duplicating its input into the outputs.

c) Switch node: routing the input into one output, according to an internal counter.

d) Adding element: performing the addition of its inputs.

e) Arithmetic processor: performing a scalar multiplication with accumulation. The node has also switching capabilities.

Each elementary processor can be arranged in an $O(\log n)$ area [BREKUN] and processes one input in time t, which in the following will be considered the unitary step. With these elements a synchronous matrix multiplier requiring $A=O(n^2 \log n)$ and $T=O(n)$ can be derived.

II. A SYNCHRONOUS MATRIX MULTIPLIER

The matrix multiplier uses the structure presented in [PREP], choosing r=n to get $T=O(n)$. The problem is to obtain

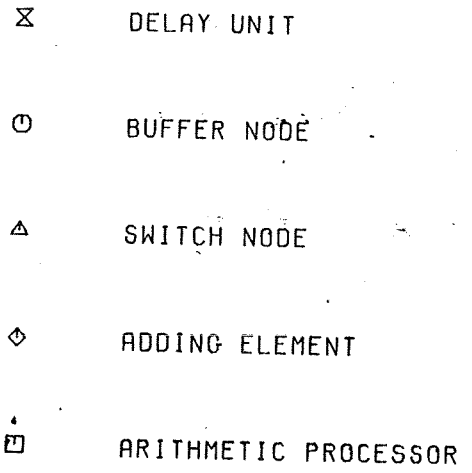

Fig.2. Elementary processors.

compatibility between inputs and outputs of the multiplier to allow cascading of modules. In our design an nxn matrix is routed sequentially by columns, along a path consisting of n wires, beginning with the n-th column. Fig.3 shows the multiplier structure. Matrix A enters in the network by columns and the elements of the i-th row are delayed by n+i-1 time units. Matrix B enters in the network by columns and it is transposed by an array of switch nodes containing a modulo n counter. The elements of the j-th column are delayed by n+j-1 time units. An array of arithmetic processors performs the matrix multiplication, according to the algorithm [PREP]. The resulting matrix C is accumulated in the array and it is again trasmitted by columns subsequently. The matrix multiplier requires an $O(n^2 \log n)$ area and $O(n)$ time.

In fig.4 a simple design of a matrix adder compatible with the previous module and introducing an unitary delay is presented. The adder requires an $O(n^2)$ area and $O(n)$ time.

III. RECURSIVE VLSI DESIGNS

A more accurate design for matrix inversion is presented in fig.5. Delay lines have been inserted to synchronize the inputs of all the modules. The whole network has two inputs and two outputs each consisting of n/2 wires. The modules contained in the dashed rectangles have the following functions. In input, the matrices V,U and Z,W, respectively,

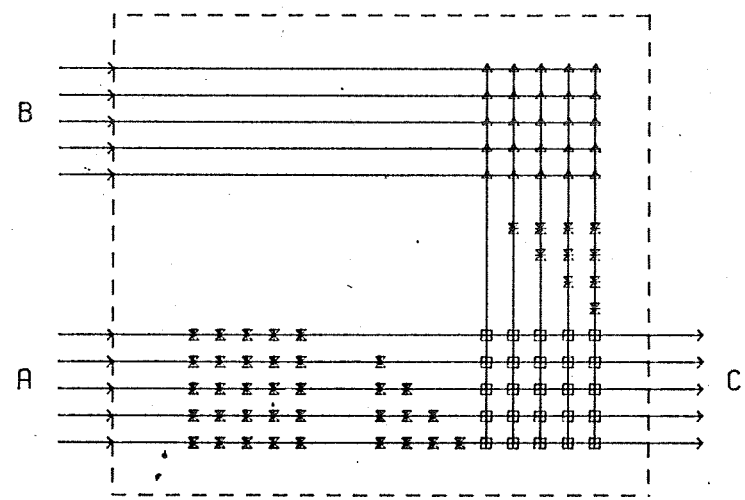

Fig.3. VLSI module performing matrix multiplication.

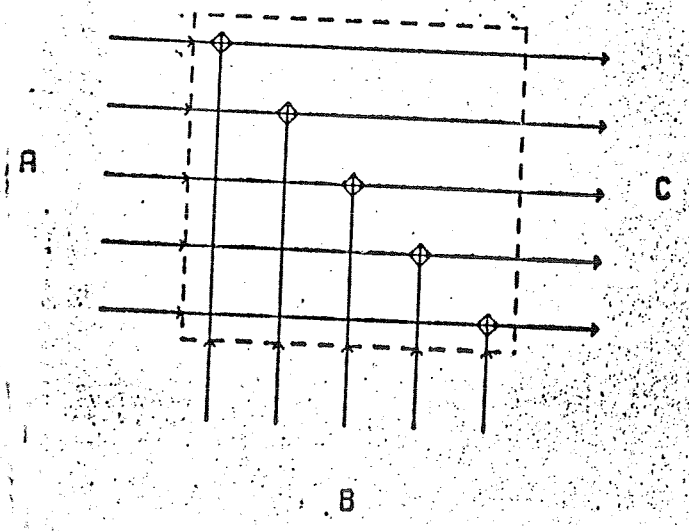

Fig.4. VLSI module performing matrix addition.

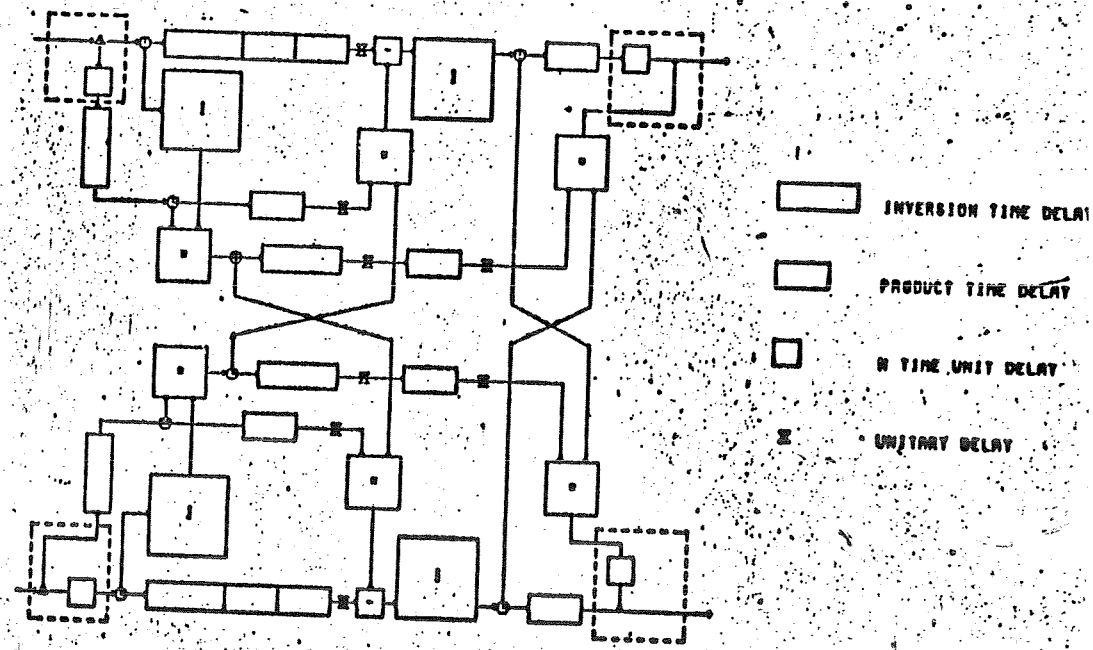

Fig.5. Recursive VLSI circuit performing matrix inversion.

are separated and routed on the proper paths; n-step delays ensure synchronization. In output, the matrices X, C^{-1} and D^{-1}, Y, respectively, are routed sequentially on the same output path.

The size of this layout can be estimated by simple considerations. Let H and L be respectively the height and the lenght of the layout. Then it is easy to see that

$$H(n) = 2\,H(n/2) + O(n \log^{1/2} n) = O(n \log^{3/2} n), \text{ and}$$

$$L(n) = 2\,L(n/2) + O(n \log^{1/2} n) = O(n \log^{3/2} n).$$

Hence, we obtain an area of order $O(n^2 \log n^3)$.

The time needed by this circuit can be readily estimated, considering all the steps in the layout between input and output together with the delay units. We obtain

$$T(n) = 2\,T(n/2) + O(n) = O(n \log n),$$

that yields the bound

$$AT^2 = O(n^4 \log^5 n).$$

Analogously, formula (1.2) leads to a recursive VLSI design for the computation of the determinant of a given matrix (see fig.6). From the figure, it is trivial to see that

$T(n) = O(n \log n)$ and, disregarding the problems due to numerical instability, the area has the same order than the one required by matrix inversion.

Hence the computation of the determinant and the inversion of a matrix have the same area-time complexity.

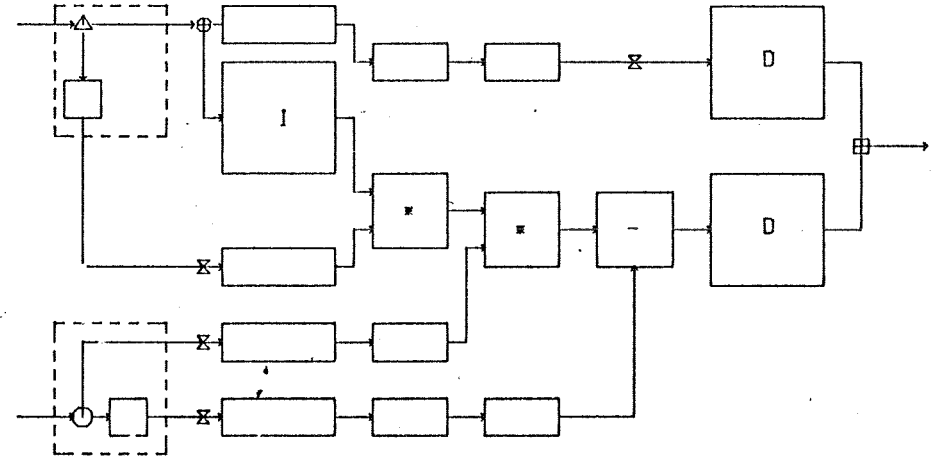

Fig.6. Recursive VLSI circuit performing determinant computation.

REFERENCES

A.BOJANCZYK, R.P.BRENT and H.T.KUNG, Numerically Stable Solution of Dense Systems of Linear Equations Using Mesh-Connected Processors. SIAM J. Sc. Stat. Comp. 5 (1984) 95-104.

R.P.BRENT and H.T.KUNG, A Regular Layout for Parallel Adders. IEEE Trans. Comput. C-31 (1982) 260-264.

B. CODENOTTI, G.LOTTI and F. ROMANI, VLSI Implementation of Fast Solvers For Band Linear Systems With Constant Coefficient Matrix. Submitted for publication.

A.S.HOUSEHOLDER, The Theory of Matrices in Numerical Analysis. Blaisdell, New York (1964).

H.T.KUNG and C.E.LEISERSON, Algorithms of VLSI Processor Arrays. Symposium on Sparse Matrix Computations, Knoxville, TN (1978).

F.T.LEIGHTON, New Lower Bound Tecniques for VLSI. Proceedings of the 22nd Annual IEEE Symposium on Foundations of Computer Science (1981), 1-12.

C.E.LEISERSON, Area-efficient graph layouts (for VLSI). Proceedings of the 21st Annual IEEE Symposium on Foundations of Computer Science (1980), 270-281.

F.P.PREPARATA and J.E.VUILLEMIN, Area-Time Optimal VLSI Networks for Multiplying Matrices. Inf. Proc. Lett. 11 (1980), 77-81.

J.E.SAVAGE, Area-Time Tradeoffs for Matrix Multiplication and related problems in VLSI models. Journal of Computer and System Sciences 22 (1981), 230-242.

C.D.THOMPSON, Area-Time Complexity for VLSI. Proc. 11th Annual ACM Symp. on the Theory of Computing (SIGACT) (1979) 81-88.