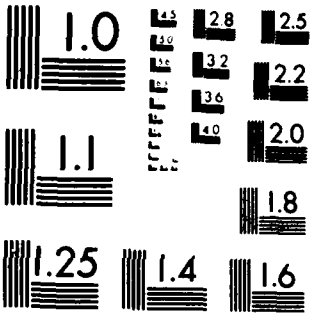
AD-A130 098 A SYSTOLIC ARCHITECTURE FOR SINGULAR VALUE DECOMPOSITION(U) STANFORD UNIV CA DEPT OF COMPUTER SCIENCE R SCHREIBER 1983 N00014-82-K-0703 1/1

UNCLASSIFIED F/G 12/1 NL

END
DATE FILMED
7 83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Presented at the 1st International Colloquium on Vector and Parallel Computing in Scientific Applications, Paris, March 198

AD A130098

Contract N00014-82-K-0703

A SYSTOLIC ARCHITECTURE FOR SINGULAR VALUE DECOMPOSITION

Robert Schreiber
Computer Science Department
Stanford University
Stanford, California 94305, USA

DTIC ELECTE MAR 16 1983 S E D

1 INTRODUCTION

Systolic arrays are highly parallel computing structures specific to particular computing tasks. They are well-suited for reliable and inexpensive implementation using many identical VLSI components. The designs consist of one and two-dimensional lattices of identical processing elements. Communication of data occurs only between neighboring cells. Control signals propagate through the array like data. These characteristics make it feasible to construct very large arrays.

Several modern methods in digital signal processing require real time solution of some of the basic problems of linear algebra [13]. Fortunately systolic arrays have been developed for many of these problems [4,10,12]. But several gaps remain. Only partially satisfying results have been obtained for the eigenvalue and singular value decompositions, for example.

This document Here we consider a systolic array for the singular value decomposition (SVD). An SVD of an m x n ($m \geq n$) matrix A is a factorization

$$A = U \Sigma V^T$$

where U is m x n with orthonormal columns, $\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_n)$ with $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$, and V is orthogonal. There are many important applications of the SVD [1,6,13].

There have been several earlier investigations of parallel SVD algorithms and arrays. First, Finn, Luk, and Pottle describe a systolic structure of $n^2/2$ processors and two algorithms that use it. But the convergence of their algorithms has not been proved and may be slow [3]. Heller and Ipsen [8] describe an array for computing the singular values of a banded matrix with bandwidth w. They use O(w) processors and $O(wn^2)$ time. Brent and Luk [2] describe an n/2 processor linear array that implements a one-sided orthogonalization method and converges reliably in O(n log n) time. Unfortunately the processors in this array are quite complex, and it is not clear that matrices with more than n columns can be efficiently accomodated.

In this paper the author discusses we discuss two topics. First, we show how an architecture for computing the eigenvalues of a symmetric matrix can be modified to compute singular values and vectors. Second, we discuss the implementation using VLSI chips of these systolic eigenvalue and SVD arrays.

The SVD is often used to regularize ill-conditioned problems. In these there are $p < n$ large singular values and n-p that are much smaller. What is needed is the pseudoinverse of the rank p matrix closest (with respect to the 2-norm) to A,

$$A_{(p)} = u_1 \sigma_1 v_1^T + \ldots + u_p \sigma_p v_p^T$$

We have recently developed a new algorithm to compute $A^+_{(p)}$ that involves nothing but a sequence of matrix-matrix products, for which systolic arrays are well-known (see, e.g., [9].) An alternate form of the algorithm can be used to compute the related orthogonal projection matrix

$$P_{(p)} = v_1 v_1^T + \ldots + v_p v_p^T$$

2 AN SVD ARCHITECTURE

Let A be a given matrix. The singular values of A will be obtained in two phases:

1. A is reduced to an upper triangular matrix B with bandwidth k+1,

   $b_{ij} = 0$ if $i > j$ or $i < j-k$,

   and $B = QAP$ where Q and P are orthogonal.

2. B is diagonalized by an iterative process equivalent to implicitly shifted QR iteration on B^TB.

With k=1 this is the standard method of Golub and Reinsch [7]. The reason for allowing k>1 is an increase in the parallelism. In phase 1, kn processors are employed; the time is O(mn/k). In phase 2, $2k^2$ processors are used; the time per iteration is 6n+O(k).

2.1 Reduction to banded form

The reduction step uses a k x n trapezoidal array that has been described in detail previously [12]. Let the m x n matrix X be partitioned as

$$X = \begin{bmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{bmatrix}$$

DTIC FILE COPY

83 03 16 065

where X_{11} is k x k. The array applies a sequence of Givens rotations to the rows of X to zero the first k columns below the main diagonal. If Q is the product of these rotations, then

$$QX = \begin{bmatrix} R_{11} & Y_{12} \\ 0 & Y_{22} \end{bmatrix}$$

where R_{11} is k x k upper triangular. R_{11}, Y_{12}, Y_{22}, and the parameters of the rotations that make up Q all flow out from the array. The time required is m. (Here and below we give "times" in units of the time required for an individual cell in the array to carry out its computation.)

Now let

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

be the given matrix. Send A through the array to produce

$$Q_1 A = \begin{bmatrix} R_{11} & C_{12} \\ 0 & C_{22} \end{bmatrix}$$

Next send $[C_{12}^T, C_{22}^T]$ through to produce

$$P_1[C_{12}^T, C_{22}^T] = [L_{12}^T, \bar{A}_{22}^T]$$

(Although the input matrix has m columns, the array can handle this factorization in time m by making $\lceil m/n \rceil$ passes over the data [12]. Now continue this process using $\bar{A}_{22}$ in place of A. After $J \ge \lceil n/k \rceil$ such steps we have produced a k+1 diagonal, upper triangular matrix B,

$$B = \begin{bmatrix} R_{1,1} & L_{1,2} & & & \\ & R_{2,2} & L_{2,3} & & \\ & & \ddots & \ddots & \\ & & & & L_{J-1,J} \\ & & & & R_{J,J} \end{bmatrix}$$

such that A = QBP where Q and P are orthogonal. The total time used is mJ = mn/k.

The transposition of data required can be done by a specialized switching device, a "systolic shifter," described earlier [12].

When singular vectors are to be computed, the rotations generated by the array may be applied to identity matrices of order m and n. This can be done by the array. These matrices accumulate the product of the rotations used, that is the orthogonal matrices Q and P above.

2.2 QR iteration

Now we consider QR iteration to get the singular values of B, hence those of A. We shall generate a sequence of matrices $\{B^{(i)}\}$ having the same structure as B and converging to a diagonal matrix. $B^{(0)} = B$ and $B^{(i+1)} = P^{(i)}B^{(i)}Q^{(i)}$ where $P^{(i)}$ and $Q^{(i)}$ are orthogonal.

First we consider QR iteration on B^TB without shifts. This can be realized by the procedure

1. Find $Q^{(i)}$ such that
   $L^{(i)} = B^{(i)}Q^{(i)}$
   is lower triangular,
2. Find $P^{(i)}$ such that
   $B^{(i+1)} = P^{(i)}L^{(i)}$
   is upper triangular.

Both steps of this procedure can be carried out by the Heller-Ipsen (HI) array [8]. This is a k x w rectangular array for QR factorization of w-diagonal matrices. In this array, plane rotations are generated at the left edge and move to the right, affecting a pair of matrix rows. Take w = k+1. $B^{(i)}$ enters the matrix at the bottom, each diagonal entering, one element at a time, into one of the processors. The array annihilates the elements of the upper triangle of $B^{(i)}$. This causes fill-in of k diagonals in the lower triangle. The resulting matrix $L^{(i)}$ emerges from the top in the same diagonal-per-processor format. It immediately enters a second array. This array annihilates the lower triangle of $L^{(i)}$ and the resulting upper triangular matrix $B^{(i+1)}$ emerges from the top (Fig. 1) The time is 2n+4k per iteration: element a_{nn} enters the bottom array at time 2n, leaves at the upper left corner at time 2n+2k, and leaves the top array at time 2n+4k.

Unshifted QR converges slowly. The rate of convergence of b_{11} to σ_1 is σ_2^2/σ_1^2. In some situations this may be adequate and the simplicity of the structure used is then a real advantage. It is also easy to pipeline the iterations. As $B^{(i+1)}$ comes out of the second array it can be sent directly into another pair of arrays to begin the (i+1)th iterations, etc. As many as n/4k iterations can be effectively pipelined; any more and the pipe length exceeds n, so that the pipe never gets full. If we choose $k = O(n^{1/2})$ and pipeline $n/4k = O(n^{1/2})$ iterations then the number of processors in both arrays is $O(n^{3/2})$ and the total time, assuming O(n) iterations of QR are required, is also $O(n^{3/2})$. These considerations also apply to the array implementation of the implicitly shifted QR algorithm that is discussed below, with one important proviso. When pipelining the iterations, some strategy for choosing several shifts in advance must be used.

2.2.1 Implicitly shifted QR iteration

To obtain adequate convergence speed we need to incorporate shifts. Following Stewart [14], suppose that one QR iteration with shift λ is performed on B^TB, and the orthogonal matrix so generated is Q. Then proceed as follows:

1. Let Q_0 be any matrix whose first k columns are the same as those of Q;
2. Using the same technique as in Section 2.1, reduce BQ_0 to upper triangular k+1 diagonal form, yielding a matrix B'.

It can be shown that $B'^{T}B'$ is the matrix that would result from one QR step with shift λ applied to $B^{T}B$.

To use the trapezoidal array as described above to carry out step 2 would be inefficient. Rather we proceed as follows. Q_0^T is composed of plane rotations that zero the first k columns of $(B^TB-\lambda)$ below the main diagonal. Applying Q_0 to B causes fill-in in the k diagonals below the main diagonal, confined to rows 2, 3, ..., 2k. See Fig. 2 for the case k=2.

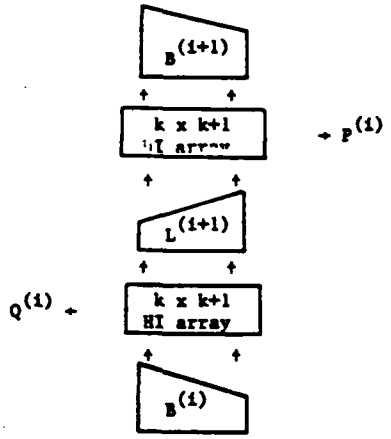


Figure 1

Unshifted QR iteration with two Heller-Ipsen arrays

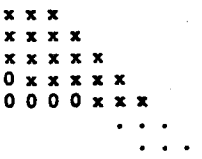


Figure 2

Structure of BQ_0, k=2

Let the first 2k rows of BQ_0 be sent into a k x 2k+1 HI array. By a sequence of plane rotations applied to the rows, the array removes the "bulge" in the lower triangle, adding a bulge of the same shape in the first 3k columns of the upper triangle. This data flows directly into another k x 2k+1 HI array that removes the elements to the right of the k^{th} superdiagonal and causes a new bulge to appear in the lower triangle, in columns k+1 through 3k-1 and extending to row 3k. (The second HI array is the mirror image of the first. Rotations are generated at its right edge and move left, affecting pairs of matrix columns. Let P_1 and Q_1 be the orthogonal matrices implicitly used by the two HI arrays. The matrix emerging from the second array is

$$B_1 = P_1 B Q_0 Q_1$$

and it has the form

$$B_1 = \begin{bmatrix} B_{11} \\ 0 \quad B_{21} \end{bmatrix}$$

where B_{11} is a k x n upper triangular, k+1 diagonal matrix, and B_{21} is an n-k x n-k matrix of the same form as BQ_0. Fig. 3 illustrates this for k=2.

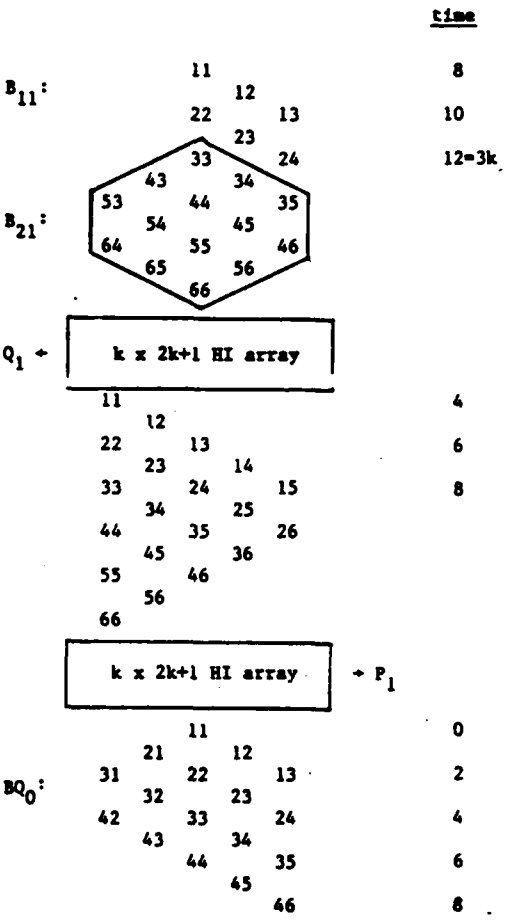


Figure 3

"Chasing the bulge" with two k x 2k+1 Heller-Ipsen arrays

Now we do exactly the same thing to B_{21}, etc. This yields matrices

$$B_j = P_j B_{j,j-1} Q_j, \quad j=2,\ldots,J$$

with

$$B_j = \begin{bmatrix} B_{j,j} & \\ 0 & B_{j+1,j} \end{bmatrix}$$

and $J = \lceil (n-1)/k \rceil$. Finally $B' = P_J \cdots P_1 B Q_0 \cdots Q_J$ is the matrix we require.

The time neede is 6n. It takes 2k steps for an HI array to start producing output. Thus, the second array starts its output at time 4k. The first element of $B_{j+1,j}$, which is the $(k+1)^{st}$ element of the main diagonal to come out of the second array, comes out at time 6k. By this time the first arrays inputs have become idle, so this element can immediately reenter. Therefore one step, from B_j to B_{j+1}, takes time 6k. There are $\lceil (n-1)/k \rceil$ such steps, hence about 6n time for the whole process.

2.3 Complex matrices

In signal processing applications, complex matrices often arise. Here we discuss the algorithms to be used for QR iteration with complex matrices. Essentially we show that the plane rotations used can be of a special form:

$$(1) \quad \begin{aligned} x' &= c^*x + \sigma y \\ y' &= -\sigma x + cy \end{aligned}$$

where x, y, and c are complex and σ is real. This saves 1/4 of the multiplications used by a fully complex plane rotation with complex s instead of σ — 12 are used instead of 16. We shall call these c,σ rotations.

It is possible to compute the SVD of a complex m x n matrix $A = A_R + iA_I$ using real arithmetic. One finds the SVD of the 2m x 2n real matrix

$$\begin{bmatrix} A_R & -A_I \\ A_I & A_R \end{bmatrix}$$

Among the 2n singular values each singular value of A occurs twice, and the singular vectors are of the form $[x_R^T, x_I^T]$ where $x = x_R + ix_I$ is a singular vector of A. But the cost is much greater. In units where the cost of doing an m x n real SVD is one, the cost for the real 2m x 2n SVD is 8 while the complex m x n approach costs 3 (not 4, since the use of the c,σ rotations saves 1/4 of the work).

We now show that the c,σ rotations suffice. To start, we note that the banded matrix B produced by the reduction phase can always be chosen to have positive real elements on its main and k^{th} superdiagonals. Indeed the reduction $B = QAP$ to k+1 diagonal, upper triangular form is not unique:

$$B = QD_1 (D_1^{-1} A D_2^{-1}) D_2 P$$

is also such a reduction for any unitary diagonal matrices D_1 and D_2. These can always be chosen to give B the stated property. In fact, the trapezoidal array can do this automatically [12]. When it generates a rotation to zero some matrix element, the second element of the pair (x,y) for instance, it chooses the parameters so that the result of the rotation is the pair

$$((|x|^2 + |y|^2)^{1/2}, 0)$$

Furthermore, the elements to be zeroed are the real elements resulting from previous rotations. The rotations to do the zeroing can, for this reason, be taken to be c,σ rotations.

Now we look at the second phase. Because of the structure of B, the main, k^{th} super and k^{th} subdiagonals of B^TB are all real. The rotations that comprise Q_0 can be taken to be c,σ rotations since they zero real elements. And by keeping track of the locations of real elements one can show that in BQ_0 all elements of the outer diagonals are real. Again because the elements to be annihilated are real, c,σ rotations can be used to eliminate the bulge. A matrix with the same structure as BQ_0 results, and the proof therefore follows by induction.

2.4 An alternate scheme

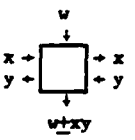
Gene Golub has pointed out that the eigenvalues of the 2n x 2n matrix

$$C = \begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix}$$

are the singular values of A taken with positive and negative sign, and if (x^T, y^T) is an eigenvector of C then x is a left singular vector of B and y is a right singular vector of B [5]. Thus we may attempt to find the eigendecomposition of C. After a symmetric interchange of rows and columns corresponding to the permutation (n+1, 2, n+2, 2, ..., 2n, n), C is a symmetric 4k-1 diagonal matrix. A 2k-1 x 4k-1 HI array can implement one step of the QR method with shifts for this matrix in n + O(k) time [10]. In the complex case, both C and the permuted C have real outermost diagonals, so c,σ rotations can be used. Thus, although twice as much hardware is used, the time per iteration is 1/6 as great as for the previous scheme.

3 VLSI IMPLEMENTATION

Now we consider how to build the cells of the HI array. The fundamental unit we use in this construction is a multiply-add cell, whose function is this:

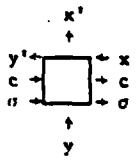


Outputs leave the cell one clock after inputs enter.

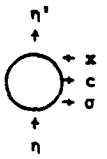
Although other primitive units (CORDIC blocks, for example) might be used, we feel that the multiply-add is a good basis for such an investigation. Currently, a floating point multiply-add is about what can be integrated on a single chip. It is almost universally useful. Indeed, the multiply-add pair is often the inner loop in numerical linear algebraic computations. Even when larger cells and pieces of arrays can be integrated into single chips, designs based on the multiply-add primitive will be useful.

We shall discuss implementation of the HI array cells for complex data. The real case was discussed earlier [11] as were the cells of the trapezoidal array [12].

The complex HI array triangularizes a banded input matrix using c,σ rotations of the form (1). The rotations are applied to a pair (x,y) of matrix elements by an internal cell



after having been generated by a boundary cell



by

$$x' = (\eta^2 + |x|^2)^{1/2}$$
$$c = x / x'$$
$$\sigma = \eta / x'$$

In the internal cell computation, 4 quantities are computed, each requiring 3 multiplies and 2 adds. Let z_R and z_I denote the real and imaginary parts of the complex quantity z. The computed values are

$$x'_R = c_R x_R + c_I x_I - \sigma y_R,$$
$$x'_I = c_R x_I - c_I x_R - \sigma y_I;$$

$$y'_R = c_R y_R - c_I y_I + \sigma x_R,$$
$$y'_I = c_R y_I + c_I y_R + \sigma x_I.$$

Using 4 multiply-add chips we can construct a compound cell that gives these results in the least possible time, 3 clocks. We assume that complex quantities are represented in "word serial" form, with the real part preceding the imaginary part on the same data path. A schedule using 4 chips that achieves the minimum latency is shown in Table 1.

Table 1. Schedule for Internal HI Cell

	Chip		Input/Output			Chip	
time	C1	C2	$c_\cdot$	y	x	C3	C4
0			c_R				
1	$c_R y_R$		c_I	y_R	x_R	$c_R x_R$	
2	$c_I y_R$	$-c_I y_I$	σ	y_I	x_I	$-c_I x_R$	$c_I x_I$
3	σx_R	$c_R y_I$	c_R			$-\sigma y_R$	$c_R x_I$
4		σx_I	c_I	y'_R	x'_R		$-\sigma y_I$
5			σ	y'_I	x'_I		

The computation at the boundary cell is this: given inputs x and η^2, compute

$$\eta'^2 = \eta^2 + x_R^2 + x_I^2$$
$$\eta' = \sqrt{\eta'^2}$$
$$c_R = x_R / \eta'$$
$$c_I = x_I / \eta'$$
$$\sigma = \eta / \eta'$$

A second primitive, for divide and square root, is needed to implement the boundary cell. We assume that a chip for computing

$$(a,b) \Longrightarrow a / b^{1/2}$$

is available. A compund cell using one multiply-add and two of these square root chips can produce results at the rate required to keep up with the internal cell. A schedule is shown in Table 2. The overall array timing is now that of the "ideal" HI array in which everything happens in a single cycle (of length 3 chip clocks). The cells are used 1/2 of the time, but two independent problems can be solved simultaneously, making full use of the hardware.

Table 2. Schedule for HI Boundary Cell

time	Chips			I/O
	mult-add	sqrt #1	sqrt #2	
1	$\rho^2 + x_R^2$			ρ^2, x_R
2	$+x_I^2$ $(=\rho'^2)$			x_I
3		$x_R[\rho'^2]^{-1/2}$		ρ
4		$x_I[\rho'^2]^{-1/2}$		ρ'^2, c_R
5		$\rho[\rho'^2]^{-1/2}$	$[\rho'^2]^{-1/2}$	c_I
6				σ, ρ'

ACKNOWLEDGEMENT

This research was partially supported by the Office of Naval Research under Contract N00014-82-K-0703 and by ESL, Incorporated, Sunnyvale, Calif.

REFERENCES

/1/ H. C. Andrews and C. L. Patterson : "Singular Value Decomposition and Digital Image Processing", IEEE Trans. Acoustics, Speech, and Signal Processing ASSP-24 (1976), pp. 26-53.

/2/ Richard P. Brent and Franklin T. Luk : "A Systolic Architecture for the Singular Value Decomposition", Report TR-CS-82-09, Department of Computer Science, The Australian National University, Canberra, 1982.

/3/ Alan M. Finn, Franklin T. Luk, and Christopher Pottle : "Systolic Array Computation of the Singular Value Decomposition", Real Time Signal Processing V, SPIE Vol. 341, Bellingham Wash., Society of Photo-optical Instrumentation Engineers, 1982.

/4/ W. M. Gentleman and H. T. Kung : "Matrix Triangularization by Systolic Array", Real Time Signal Processing IV, SPIE Vol. 298, Bellingham, Wash., Society of Photo-optical Instrumentation Engineers, 1981.

/5/ Gene Golub. Private communication.

/6/ G. H. Golub and F. T. Luk : "Singular Value Decomposition: Applications and Computations", ARO Report 77-1, Trans. of the 22nd Conf. of Army Mathematicians (1977), pp. 577-605.

/7/ G. H. Golub and C. Reinsch : "Singular Value Decomposition and Least Squares Solutions", Numer. Math. 14, (1970), pp. 403-420.

/8/ Don E. Heller and Ilse C. F. Ipsen : "Systolic Networks for Orthogonal Decompositions, with Applications", SIAM J. Scient. and Stat. Comput., to appear.

/9/ H. T. Kung and Charles Leiserson : "Systolic Arrays for (VLSI)", in Carver Mead and Lynn Conway, Introduction to VLSI Systems, Reading, Mass., Addison-Wesley, 1980.

/10/ Robert Schreiber : "Systolic Arrays for Eigenvalue Computation", Real Time Signal Processing V, SPIE Vol. 341, Bellingham, Wash., Society of Photo-optical Instrumentation Engineers, 1982

/11/ Robert Schreiber : "Systolic Arrays for Eigenvalues", Proc. of the Inter-American Workshop in Numerical Analysis, New York, Springer-Verlag, to appear.

/12/ Robert Schreiber and Philip J. Kuekes : "Systolic Linear Algebra Machines in Digital Signal Processing", in Sun-Yuan Kung, ed., Proc. of the USC Workshop on VLSI and Modern Signal Processing, Englewood Cliffs, New Jersey, Prentice-Hall, to appear.

/13/ J. M. Speiser and H. J. Whitehouse : "Architectures for Real-Time Matrix Operations", Proc. Government Microcircuit Applications Conf., held in Houston, Tex., 1980.

/14/ G. W. Stewart : Introduction to Matrix Computations, New York, Academic Press, 1973.