STABLE PARALLEL ALGORITHMS FOR COMPUTING AND UPDATING THE QR DECOMPOSITION

E. J. Kontoghiorghes and M. R. B. Clarke
Dept. of Computer Science, QMW College,
Mile End Road, London E1 4NS, U.K

ABSTRACT

In this paper we propose new stable parallel algorithms based on Householder transformations and compound Given's rotations to compute the QR decomposition of a rectangular matrix. The predicted execution time of all algorithms on the massively parallel SIMD array processor AMT DAP 510, have been obtained and analyzed. Modified versions of these algorithms are also considered for updating the QR decomposition, when rows are inserted to the data matrix.

INTRODUCTION

A number of algorithms have been proposed and analyzed to compute and update the QR decomposition[3, 4, 7, 5, 6, 11]. Here we consider fast stable algorithms based on Householder transformations and compound Givens rotations to compute and update the QR factorization on a massively parallel SIMD array processor.

The (complete) QR factorization of an m x n ($m > n$) matrix A is divided into two stages. In stage 1 the matrix A is reduced to lower trapezoidal form and in stage 2 the lower trapezoid is triangularized. Let $Q \in \Re^{m \times m}$, $P \in \Re^{n \times n}$ be orthogonal, $\Pi \in \Re^{n \times n}$ be a permutation matrix and the rank of A be k ($k < n$). The orthogonal decompositions of stages 1 and 2 are then

$$Q^T A \Pi = \begin{bmatrix} 0 & 0 \\ L_1 & L_2 \end{bmatrix} \quad \textbf{(stage 1)} \tag{1}$$

and

$$\begin{bmatrix} L_1 & L_2 \end{bmatrix} P = \begin{bmatrix} 0 & L \end{bmatrix} \textbf{ (stage 2)} \tag{2}$$

where $L_2 \in \Re^{k \times k}$ and $L \in \Re^{k \times k}$ are both lower triangular.

For the updating of the QR decomposition we will consider the case where A is of full column rank and $\Pi = I_n$. That is, under these assumptions (1) can be writen as

$$Q^T A = \begin{bmatrix} 0 \\ L \end{bmatrix} \tag{3}$$

where $L \in \Re^{n \times n}$ is lower triangular non singular. If the rows inserted to the data matrix A after we computed (3) are denoted by $B \in \Re^{k \times n}$, then we need to compute the orthogonal decomposition

$$\hat{Q}^T \begin{bmatrix} L \\ B \end{bmatrix} = \begin{bmatrix} \hat{L} \\ 0 \end{bmatrix} \tag{4}$$

where now $\hat{Q}$ is an $(n + k)$ x $(n + k)$ orthogonal matrix and $\hat{L} \in \Re^{n \times n}$ is lower triangular.

1. Notation

First we describe a general notation used throughout the paper. An m x n real matrix A with elements $a_{i,j}$ ($1 \le i \le m, 1 \le j \le n$) will be denoted by $A = [a_{i,j}] \in \Re^{m \times n}$ and similarly $V = [v_i] \in \Re^m$ will denote a vector with real elements v_i ($1 \le i \le m$). The k^{th} column and row of A are given by a_{*k} and a_{k*} respectively and $A[i, j : k, s]$ is a k x s sub–matrix of A starting from element $a_{i,j}$. A zero dimension will denote a null matrix or vector and all vectors are considered to be column vectors unless transposed, i.e a_{k*} is a column vector and a_{k*}^T is a row vector.

The application of arithmetic operations and functions on an array will be equivalent to applying the corresponding serial arithmetic operations and functions to the elements of this array. When an operation is between an array and a scalar, the scalar is expanded to the mode of the array.

2. The AMT DAP 510 series

The AMT DAP 500 series, is a bit organized (Distributed) Array Processor, having 1024 Processing Elements (PE) arranged in a two dimensional 32 x 32 array[12, 13]. Each PE comprises a number of one–bit registers, an arithmetic/ logical unit and data paths, including connections to a section of memory that is local to each PE. Although each PE can only access data residing in its local memory, there are facilities for shifting data across the array in any one of four directions. In addition to that, there are Row and Column highways which connect the PEs of the array. These row and column highways provide rapid data broadcast between the master control unit and the array processors (*fig.*(1)).

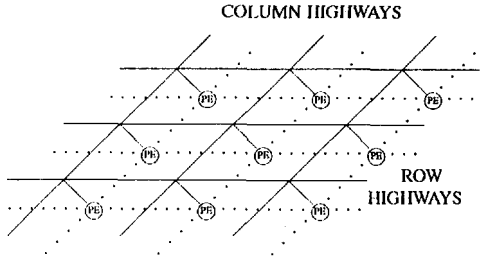

Fig. 1: PE interconnections

Data items in a vector or matrix are distributed over the DAP processors, each being processed by a single processor. Operations are (conditionally) performed by all PEs at once. If the size of a vector exceeds the number of DAP PEs, the system organizes the vector so that each PE operates on more than one element. With matrices, this will happen if either one of the sides of the matrix exceeds the Edge Size (ES) of the DAP.

Fortran–Plus is the main language provided for programming the DAP[1]. It has features that allow the programmer to manipulate whole arrays (matrices, vectors) as single objects. Operations on these objects have, (a) the ability to operate on the complete object with no loop constructors, (b) selection operators that conditionally act on parts of these objects and (c) functions performing special operations.

All the algorithms in this paper were implemented on the AMT DAP 510 which is abbreviated to DAP and all times are in msec unless otherwise stated. For the data of our problems we have used normally distributed random values generated by the DAP routine *AMT_GS_RAND_MC_NORM*[2].

COMPUTING THE QR DECOMPOSITION

1. Computing stage 1

In this section we consider the computation of the orthogonal decomposition (1) using Householder transformations with column pivoting. This method is also used when A is of full column rank but ill conditioned. Let $I_n^{(i,\mu)}$ be the matrix I_n with columns $n-i+1$ and μ interchanged (i.e $I_n^{(i,\mu)}$ is an elementary permutation matrix) and

$$Q_i^T = I_m - \frac{h^{(i)} h^{(i)^T}}{b_i} \tag{5}$$

be an m x m Householder matrix which annihilates the first $m-i$ elements of $a_{*\,n-i+1}$ (pivot column), when it is premultiplied to A. If $V = [v_i] \in \Re^n$ is defined as $v_i = \|a_{*,i}\|^2$ and for $i = 1, \cdots, k$

$$\left.\begin{aligned} &\mu_i = \max_{\varrho}(v_\varrho) \quad (1 \le \varrho \le n-i+1) \\ &A = Q_i^T (A\, I_n^{(i,\mu_i)}) \\ &\quad = \begin{bmatrix} L_{11}^{(i)} & 0 \\ L_{21}^{(i)} & L_{22}^{(i)} \end{bmatrix} \begin{matrix} m-i \\ i \end{matrix} \\ &\qquad\quad n-i \quad i \\ &V = V\, I_n^{(i,\mu_i)} \\ &v_j = v_j - a_{m-i+1\,j}^2 \ (j = 1, \ldots, n-i) \end{aligned}\right\} \tag{6}$$

then the matrices in (1) are equivalent to $[\,L_1 \;\; L_2\,] = [\,L_{21}^{(k)} \;\; L_{22}^{(k)}\,]$, $Q^T = \prod_{i=1}^{k} Q_{k-i+1}^T$ and $\Pi = \prod_{i=1}^{k} I^{(i,\mu_i)}$.

The criterion used to declare rank$(A) = k$ is $v_{\mu_{k+1}}^{(k)} < \tau$, where τ is an absolute tolerance parameter and its value depends on the scaling of A[6, 11]. The value of τ is assumed to be given. Note that v_p in (1) is the square Euclidean norm of the p^{th} column of $L_{11}^{(i)}$.

The permutation $A\, I_n^{(i,\mu_i)}$ interchanges two columns of A based on the following strategy. From $L_{11}^{(i-1)}$ select the column with the maximum Euclidean norm, which in (6) is given by $\mu_i = \max_{\varrho}(v_\varrho)$ where $1 \le p \le n-i+1$. The colurhns μ_i and $n-i+1$ of A are interchanged, that is, $a_{*\,\mu_i}$ becomes the pivot column of the i^{th} Householder transformation. This interchange results in the diagonal of the lower triangular matrix $L_{22}^{(i)}$ to be increasing in magnitude.

The permutation matrix Π can be stored and computed using one of the two vectors $\xi, \zeta \in \Re^n$, where $\Pi = [\,e_{\zeta_1} \; \cdots \; e_{\zeta_n}\,] = [\,e_{\xi_1} \; \cdots \; e_{\xi_n}\,]^T$. With initial values $\zeta_i = \xi_i = i\ (i = 1, \ldots, n)$, a permutation $I_n^{(i,\mu_i)}$ is equivalent to swapping first the elements ζ_{n-i+1} and ζ_{μ_i} of ξ and then swapping the elements $n-i+1$ and μ_i of ζ.

For the implementation of (6) on the DAP, let $n = N$ ES, $m = M$ ES and $1 < M \le$ ES. The time required to construct and apply the i^{th} Householder transformation is given by

$$\begin{aligned} T_1(M,N) &= a_0 + a_1 \lceil \frac{m}{ES} \rceil + a_2 \lceil \frac{m}{ES} \rceil \lceil \frac{n}{ES} \rceil \\ &= a_0 + a_1 M + a_2 M N \end{aligned} \tag{7}$$

where $a_0 = 2.43$, $a_1 = 0.014$ and $a_2 = 1.683$. Since $Q_i^T A$ affects only the top $(m-i+1)$ x $(n-i+1)$ sub–matrix of A, we divide the Householder transformations into the K sets $S^{(1)}, \ldots, S^{(K)}$ with $S_j^{(i)}$ equivalent to $Q_{(i-1)ES+j}^T$, where $K = \lceil \frac{k}{ES} \rceil$, $\delta = k - (K-1)$ES, $1 \le j \le t_i$ and

$$t_i = \begin{cases} \delta & \text{if } i = K \\ \text{ES} & \text{otherwise} \end{cases}$$

For $m_i = (M-i+1)$ES and $n_i = (N-i+1)$ES, $S_j^{(i)}$ is applied only to the top m_i x n_i sub–matrix of A which is denoted by $A^{(i)}$. In *fig.*(2) we show the state of the matrix $A \in \Re^{16\,x\,12}$ when $i = 2$, $j = 3$ and ES=4. The Householder transformation corresponding to $S_3^{(2)}$, is applied to $A^{(2)} \in \Re^{12\,x\,8}$ (dash box).

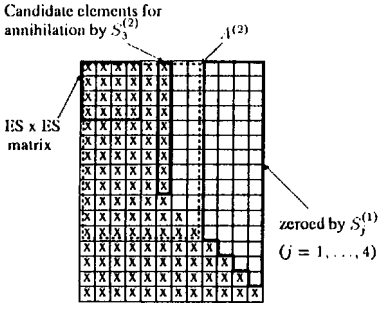

Fig. 2: Applying $S_3^{(2)}$ $(m = 16, n = 12, \text{ES} = 4)$

For A having full column rank, that is $k = n$ or otherwise $K = N$ and $\delta =$ ES, the total time spent in applying the Householder transformations is given by

$$\begin{aligned} T_3(M,N) &= ES\frac{N}{6}(\,6a_1 + 3a_2(2M - N + 1) + \\ &\qquad a_3(3MN + 3M - N^2 + 1)\,) \end{aligned}$$

and in this case, the estimated execution time of implementing (6) on the DAP, is found to be

$$\begin{aligned} T_{h_1}(M,N) &= N(\,105.9 + 31.8M + 5.9N + \\ &\qquad 27.8MN - 9.2N^2\,) \end{aligned}$$

From table(1) we observe that the time spent in applying the Householder transformations is approximately 90% of the total time[8].

M	N	exec. time	$T_{h_1}(M,N)$	$T_3(M,N)$
2	1	0.22	0.22	0.19
4	1	0.34	0.34	0.30
8	1	0.58	0.56	0.51
8	2	1.56	1.56	1.40
8	7	10.56	10.56	9.61
9	1	0.64	0.64	0.57
9	4	5.08	5.08	4.63
9	8	14.83	14.83	13.57
10	1	0.70	0.70	0.62
10	3	3.58	3.58	3.26
10	9	20.12	20.12	18.50
15	1	0.99	1.00	0.89
15	3	5.31	5.31	4.89
15	9	32.82	32.82	30.64
15	13	58.90	58.90	54.98

Table 1: Time in seconds for computing (1)

2. Computing stage 2

The computation of (2) using Householder transformations is similar to the computation of (1). For $\tilde{n} = n - k$, $L_1 = [\tilde{l}_{i,j}]$ and $L_2 = [l_{i,j}]$, let P_i be the n x n Householder matrix

$$P_i = I_n - \frac{1}{c_i} \begin{bmatrix} \tilde{l}_{i,*} \\ \gamma_i e_i \end{bmatrix} \begin{bmatrix} \tilde{l}_{i,*}^T & \gamma_i e_i^T \end{bmatrix} \qquad (8)$$

where $\gamma_i = l_{i,i} \pm s$, $s^2 = \|\tilde{l}_{i,*}\|^2 + l_{i,i}^2$, $c_i = \gamma_i s$ and $I_k = [e_1 \dots e_k]$. The application of P_i from the right of $[L_1 \;\; L_2]$ will annihilate the i^{th} row of L_1 by affecting the whole matrix L_1 and only the i^{th} column of L_2. After we have applied $P_1, \dots, P_i$ $(1 \le i \le k)$ from the right of $[L_1 \;\; L_2]$, the first i rows of L_1 are zero and L_2 remains lower triangular. The orthogonal matrix P in (2) is then defined as $\prod_{i=1}^{k} P_i$.

The evaluation of the i^{th} Householder transformation gives

$$\begin{bmatrix} L_1 & L_2 \end{bmatrix} P_i = \begin{bmatrix} L_1 - x\,\tilde{l}_{i,*}^T & L_2 - \gamma_i\, x\, e_i^T \end{bmatrix}$$

where $x = (L_1 \tilde{l}_{i,*} + \gamma_i l_{*,i})/c_i$. Using the same notation as in the previous section, $S_j^{(i)}$ $(1 \le i \le K,\ 1 \le j \le t_i)$ now corresponds to P_ρ, where $\rho = j + \sum_{q=1}^{i-1} t_q$ and

$$t_i = \begin{cases} \delta, & i = 1 \\ \mathrm{ES}, & 1 < i \le K \end{cases}$$

The transformation $S_j^{(i)}$ is applied only to the affected bottom $(k - \sum_{q=1}^{i-1} t_q)$ x $(\tilde{n}+1)$ sub–matrix of $[L_1 \;\; l_{*,i}]$, with execution time

$$T(k^{(i)}, \eta) = a_0 + a_1\eta + a_2\lceil\frac{k^{(i)}}{ES}\rceil + a_3\eta\lceil\frac{k^{(i)}}{ES}\rceil$$

where $\eta = \lceil\frac{\tilde{n}}{ES}\rceil$, $a_0 = 1.57$, $a_1 = 0.06$, $a_2 = 0.22$ and $a_3 = 0.54$. Thus, the total time spent in applying all the Householder transformations to compute (2) is given by

$$T_{h_2}(k,\eta) = \delta T(k,\eta) + ES \sum_{i=2}^{K} T(k^{(i)}, \eta)$$

For the computation of (2) we consider now a new Given's sequence which we call ngs. The ngs requires a total of $k + \tilde{n} - 1$ compound disjoint Given's rotations (cdgr) to annihilate the elements of L_1. The application of the cdgr does not create any non zero elements above the main diagonal of L_2 and previously created zeroes of L_1 are preserved.

Let $G_{i,j} \in \Re^{n \times n}$ be the Givens rotation matrix which annihilates $\tilde{l}_{i,j}$ when it is applied from the right of $[L_1 \;\; L_2]$, by effecting the row columns $\tilde{l}_{*,j}$ and $l_{*,i}$. At most $\min(\tilde{n}, k)$ disjoint Givens rotations can be applied simultaneously and we assume that $\tilde{n} \le k$. All the elements annihilated from the application of the i^{th} cdgr, say $G^{(i)}$, lie on a diagonal and $d_{i,j}$ will denote the diagonal of L_1 starting from $\tilde{l}_{i,j}$. If e_i is the number of elements annihilated from the application of $G^{(i)}$, then

$$e_i = \begin{cases} i, & \text{if } 1 \le i < \tilde{n} \\ \tilde{n}, & \text{if } \tilde{n} \le i \le k \\ \tilde{n} + k - i, & \text{if } k < i < \tilde{n} + k \end{cases}$$

In *fig.*(3 a) a number i $(1 \le i \le 27)$ corresponds to the elements annihilated from the application of $G^{(i)}$, when $k = 20$ and $\tilde{n} = 8$.

For the implementation of the ngs on the DAP, let $\tilde{n} = \eta$ES and $k = K$ES. The application of the cdgr is divided in three phases with phase Φ_p applying the cdgr in $S^{(p)}$ $(p = 1, 2, 3)$, where

$$\begin{aligned} S_i^{(1)} &= G^{(i)} && \text{for } i = 1, \dots, \tilde{n} - 1 \\ S_i^{(2)} &= G^{(\tilde{n}+i-1)} && \text{for } i = 1, \dots, k - \tilde{n} \\ S_i^{(3)} &= G^{(k+i-1)} && \text{for } i = 1, \dots, \tilde{n} \end{aligned}$$

Phase Φ_2 is divided into the $K - \eta$ sub–phases, $\varphi_1, \dots, \varphi_{K-\eta}$, where at φ_i the diagonals $d_{11}, d_{21}, \dots, d_{ES1}$ of the sub–matrix $L_1[(i-1)\mathrm{ES}+1, 1 : k-(i-1)\mathrm{ES}, \tilde{n}]$ are annihilated. In phases Φ_2 and Φ_3, top zero ES x ES matrices of the affected sub–matrices of L_1 and L_2, are excluded from the computations. In *fig*(3 b) we show the phases and the sub–phases of ngs, when ES = 4.

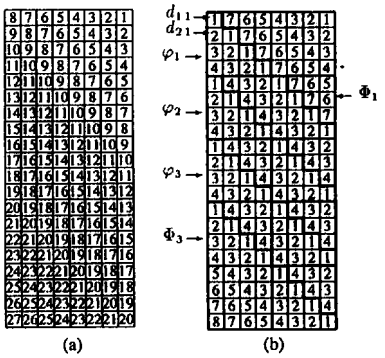

Fig. 3: ngs and its phases (ES = 4)

The estimated execution time of constructing and applying the cdgr $G^{(i)}$ is

$$T(K, e_i) = 1.5 + K\lceil\frac{e_i}{ES}\rceil(0.887 + 0.004K - 0.006\lceil\frac{e_i}{ES}\rceil)$$

and thus, the time required to apply all the cdgr of the ngs, is given by

$$T_{ngs}(K,\eta) = ES\sum_{i=1}^{\eta} T(K,i) - T(K,\eta) + ES\sum_{i=1}^{K-\eta} T(K+1-i,\eta) + ES\sum_{i=1}^{\eta} T(\eta+1-i,\eta+1-i)$$

The total execution time for computing (2) using ngs, is found to be

$$T_g(K,\eta) = K(a_0 + a_1\eta + a_2 K + a_3\eta K + a_4\eta^2) + \eta(a_5 - a_6\eta^2)$$

where $a_0 = 35.7$, $a_1 = 38.8$, $a_2 = 11.0$, $a_3 = 17.8$, $a_4 = 15.1$, $a_5 = 65.9$ and $a_6 = 2.9$. From *fig.*(4) where $T_g(K,\eta) = 0$ for $K < \eta$, we clearly observe the efficiency in performance when computing (2) using Householder reflections rather than Given's rotations, on the DAP.

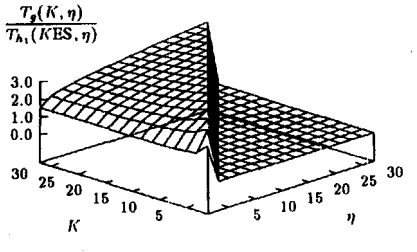

Fig. 4: Ratio $T_g(K,\eta)/T_{h_1}(KES,\eta)$

UPDATING THE QR DECOMPOSITION

Modified versions of the methods used to compute (2) can be employed to solve the updated problem (4). For $L = [l_{i,j}]$ and $B = [b_{i,j}]$, we define the Householder matrix

$$H^{(i)} = I_{n+k} - \frac{1}{c_i}\begin{bmatrix} \eta_i\, e_i \\ b_{*,i} \end{bmatrix}\begin{bmatrix} \eta_i\, e_i^T & b_{*,i}^T \end{bmatrix} \qquad (9)$$

where $\eta_i = l_{i,i} + s$, $c_i = s\,\eta_i$, $s = sign(l_{i,i})(l_{i,i}^2 + \|b_{*,i}\|^2)^{1/2}$ and e_i be the i^{th} column of I_n. The Householder transformation

$$\begin{bmatrix} L \\ B \end{bmatrix} = H^{(i)}\begin{bmatrix} L \\ B \end{bmatrix}$$

which annihilates the elements of $b_{*,i}$, is equivalent to

$$l_{i,*} = l_{i,*} - \eta_i\, x$$

and

$$B = B - b_{*,i}\, x^T$$

where

$$x^T = (\eta_i\, l_{i,*}^T + b_{*,i}^T\, B)\,/\,c_i$$

The matrix $\tilde{Q}^T$ in (4) is then equivalent to $\prod_{i=1}^{n} H^{(i)}$.

The ngs (*fig.*(3 a)) considered previously, can also be used to annihilate the elements of B by preserving the lower triangular form of L in (4). The annihilation of the element $b_{i,j}$ is achieved by applying a Given's plane rotation between the rows $b_{i,*}$ and $l_{j,*}$. That is, if $G_{i,j} \in \Re^{(n+k)\times(n+k)}$ is this Given's rotation, then

$$G_{i,j} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & & s & & \\ & & & 1 & & & \\ & & -s & & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{bmatrix} \begin{matrix} \\ \\ \leftarrow j \\ \\ \leftarrow n+i \\ \\ \\ \end{matrix}$$

$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \uparrow \qquad \uparrow$$
$$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j \qquad n+i$$

where $c = l_{j,j}/t$, $s = b_{i,j}/t$ and $t^2 = l_{j,j}^2 + b_{i,j}^2$. The implementation of these and other methods[9, 10] for computing (4) on the DAP, are similar to the ones described earlier.

DISCUSSION

The aim has been to present stable parallel algorithms for computing and updating the QR decomposition. The implementation of these fairly easy to understand algorithms, on the massively parallel AMT DAP 510 was considered. The estimated execution time of all algorithms has been obtained and analyzed. A comparison between the predicted times for computing (2) has shown the superiority in performance of Householder transformations on the DAP (*fig.*(4)).

References

1. Active Memory Technology (AMT) Ltd. *Fortran – Plus enhanced*, 1990.
2. Active Memory Technology (AMT) Ltd. *AMT General Support Library*, 1992.
3. P. Businger and G.H. Golub. Linear least squares solutions by Housholder transformations. *Numerische Mathematik*, 7, 1965.
4. J.H. Chambers. *Computational methods for data analysis*. John Wiley and Sons, Inc., 1977.
5. P.E. Gill, G.H. Golub, W. Murray, and M.A. Saunders. Methods for modifying matrix factorizations. *Mathematics of Computation*, 28(126), 1974.
6. G.H. Golub and C.F. Van Loan. *Matrix computations*. North Oxford Academic, 1983.
7. J.W.Daniel, W.B. Gragg, L. Kaufman, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the Gram–Schmidt QR factorization. *Mathematics of Computation*, 30(136), 1976.
8. E.J. Kontoghiorghes and M.R.B. Clarke. Algorithms for the solution of the OLM on the DAP. Technical Report 602, QMW College, Dept. of Computer Science, 1992.
9. E.J. Kontoghiorghes and M.R.B. Clarke. Solving the modified OLM using a distributed array processor. Technical Report 604, QMW College, Dept. of Computer Science, 1992.
10. E.J. Kontoghiorghes and M.R.B. Clarke. Computing the complete orthogonal decomposition using a SIMD array processor. In Springer-Verlag, editor, *PARLE' 93*, LNCS, 1993. (in press).
11. C.L. Lawson and R.J. Hanson. *Solving Least Squares Problems*. Prentice–Hall Englewood Cliff, 1974.
12. D. Parkinson. The distributed array processor (DAP). *Computer physics communications*, 28, 1983.
13. D. Parkinson, D.J. Hunt, and K.S. MacQueen. The AMT DAP 500. In *33rd IEEE Computer Society International Conference*, San Francisco, 1988.