International Conference on
Computer System and Education, IISc, 1994

Design of Systolic arrays for QR Decomposition

Manoj Misra
Department of Computer Science and Engg.
H.B.T.I. Kanpur
Rajat Moona
Department of Computer Science and Engg.
IIT Kanpur.

Abstract

In this paper we propose design of three systolic arrays to perform QR decomposition of a square matrix. Our first design is based on Given's rotation method [1]. In contrast to the earlier designs [1], [2] based on this method, our design uses $n(n-1)/2$ homogeneous PEs. Our next design is based on Householder method [4]. This method performs only $(n-1)$ square root operations in contrast to $n(n-1)$ square root operations performed in Given's method [6]. Our third design first reduces the given matrix to Hessenberg form [5] and then applies Q-R decomposition to it.

1 Introduction

An important matrix problem that arises in many applications like signal processing [8], image processing, solution of differential equations etc. is that of solving a set of simultaneous linear eqations. The usual numerical method adopted to solve this problem is to triangularize the coefficient matrix and then use back substitution. There are several methods to triangularize matrix, however a numerically attractive method is the QR decomposition [7] that can be defined as follows.

Given an $N \times N$ matrix A, there exists an orthogonal matrix Q and an upper triangular matrix R where A = QR.

In this paper we propose three designs of systolic array for QR decomposition. Our first design is based on Given's rotation [1] method. S.Y. Kung proposed two systolic arrays [1], [2] for doing QR decomposition with hardware complexity $O(n^2)$ and time complexity $O(n)$ based on Given's method. Here we propose yet another design based on the Given's rotation method with similar hardware and time complexity. However, in our design the PE utilization is better compared to the other designs. Moreover in comparison with designs proposed by S.Y. Kung, our design uses homogeneous PEs and is therfore simpler to implement. The second de-

sign is based on the Householder transformation method [4]. This design uses a triangular array. The Householder transformation method performs less number of square root operations in comparison to Given's method. Our third design takes the advantage of the fact that although the factorization is possible for any matrix, the numerical procedure is very much faster for a Hessenberg matrix [5]. So we first propose a systolic array that reduces the original matrix into Hessenberg form and then applies the QR decomposition to it. In contrast to the design proposed by N. Torratba and J.J. Navarro [3] our design reduces the matrix to Hessenberg form using Householder reduction. This array can be used in applications such as finding eigenvalues, where QR decomposition is performed several times.

2 A Systolic array based on Given's Rotation

In the Given's algorithm the subdiagonal elements of the first column are nullified first, followed by the subdiagonal elements of the second column and so forth until an upper triangular form is eventually reached. The complete procedure can be described as below.

For an invertible matrix A, the upper triangular matrix R is obtained as follows.

$$\begin{aligned} Q^T A &= R \\ Q^T &= Q_{N-1} Q_{N-2} \dots Q_1 \\ \text{and } Q_P &= Q^{(P,P)} Q^{(P+1,P)} \dots Q^{(N-1,P)} \end{aligned}$$

where $Q^{q,p}$ is the Given's rotation operator used to annihilate the matrix element located at row $(q+1)$ and column p and has the following form.

$$\begin{bmatrix} 1 & 0 & . & . & . & . & 0 & 0 \\ 0 & 1 & . & . & . & . & 0 & 0 \\ 0 & 0 & . & . & . & & 0 & 0 \\ . & . & . & \cos\theta & \sin\theta & . & . & . \\ . & . & . & -\sin\theta & \cos\theta & . & . & . \\ 0 & 0 & . & . & . & . & 0 & 0 \\ 0 & 0 & . & . & . & . & 1 & 0 \\ 0 & 0 & . & . & . & . & 0 & 1 \end{bmatrix} \begin{matrix} \\ \\ \\ \text{row } q \\ \text{row } q+1 \\ \\ \\ \\ \end{matrix}$$

where $\theta = \tan^{-1}[a_{q-1,p}/a_{q,p}]$. The above operation of creating $\cos\theta$ and $\sin\theta$ is named Given's generation (GG).

The matrix product $A' = Q^{q,p} A$ is then.

$$\begin{aligned} a'_{q,k} &= a_{q,k}\cos\theta + a_{q+1,k}\sin\theta \\ a'_{q+1,k} &= -a_{q,k}\sin\theta + a_{q+1,k}\cos\theta \\ a'_{jk} &= a_{jk} \ \forall j <> q \text{ and } q+1; \\ &\quad \text{and } \forall k = 1..N \end{aligned}$$

The operations in first two eqauations above form the Given's rotation (GR).

The sequential program for QR decomposition can be written as follows.

For k from 1 to $N-1$
  For i from $N-1$ to k
    $\theta = \tan^{-1}(a(i+1,k)/a(i,k))$
    For j from k to N
      $\text{temp1} = a(i,j)\cos\theta + a(i+1,j)\sin\theta$
      $\text{temp2} = a(i,j)\sin\theta + a(i+1,j)\cos\theta$
      $a(ij) = \text{temp1}$
      $a(i+1,j) = \text{temp2}$

This sequential program is of $O(N^3)$, where N is the problem size. The single assignment form of this program and the corresponding dependence graph (DG) (fig. 1) as given by S.Y. Kung in his book [1] are as follows.

For k from 1 to $N-1$
  For i from $N-1$ to k
    For j from k to N
      $ox(i,j,k) = xy(i-1,j,k-1)$
      if $i = N-1$

$oy(i,j,k) = ny(i,j,k-1)$
else
$oy(i,j,k) = nx(i+1,j,k)$
if $j = k$
$\theta(i,j,k) = tan^{-1}\frac{\text{oy}(i,j,k)}{\text{ox}(i,j,k)}$
else
$\theta(i,j,k) = \theta(i,j-1,k)$
$nx(i,j,k) =$
$ox(i,j,k)\cos(\theta(i,j,k))+$
$oy(i,j,k)\sin(\theta(i,j,k))$
$ny(i,j,k) =$
$-ox(i,j,k)\sin(\theta(i,j,k))+$
$oy(i,j,k)\cos(\theta(i,j,k))$

At each node (i,j,k), $ox(i,j,k)$ and $oy(i,j,k)$ denote the old values of two elements in a rotation. New values of these two elements are represented by $nx(i,j,k)$ and $ny(i,j,k)$. The initial conditions are $nx(i,j,0) = a_{ij}$ and $ny(N-1,j,0) = a(N,j)$. At the end of the computation, $nx(i,j,i)$ will contain row i of the resultant matrix R ($i \leq j < N$). Row N of matrix R will be available in $ny(N-1,j,N-1)$.

There are only one kind of arcs between K-planes of the DG, $(i,j,K-1)$ to (i,j,k). Kung suggested two designs of systolic arrays [1], [2] by taking the projection direction [0 −1 1] and [1 0 0]. The array with projection direction [1 0 0] saves half of the hardware, but both arrays make use of two different type of PEs; one at the boundary which performs Given's generation (GR) and the second which performs the Given's generation (GG). (The actual implementation makes use of three different types of cells [3]) Here we suggest a systolic array by taking the projection direction [0 1 0]. This array is also a triangular array but all the PEs are of same type and so PE utilization is more in this case as clear by the space-time diagram (fig. 4). Moreover the number of PEs required is less $\frac{(n-1)n}{2}$ as opposed to $\frac{n(n+1)}{2}$ in this case for same matrix size. A signal flow graph (SFG) (fig. 2) obtained by choos-

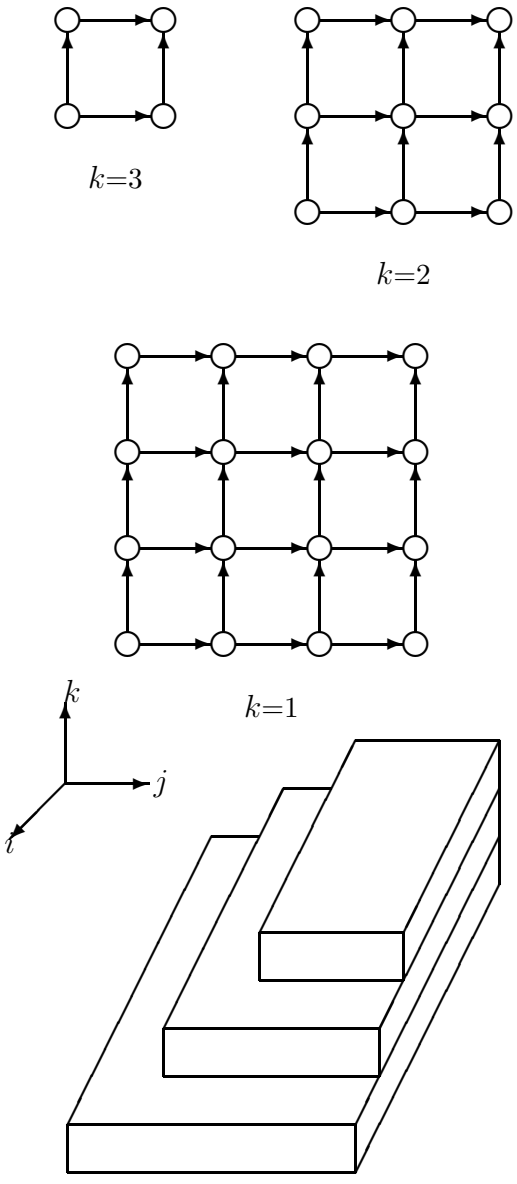


Figure 1: Dependence Graph (DG) based on Given's Rotation

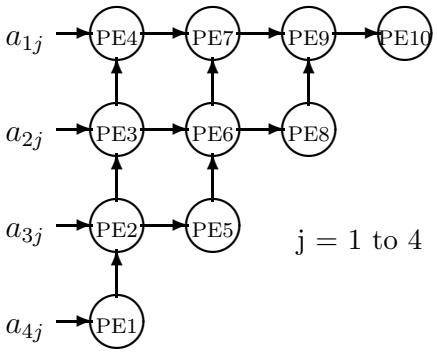


Figure 2: Signal Flow Graph (SFG) based on Given's Rotation

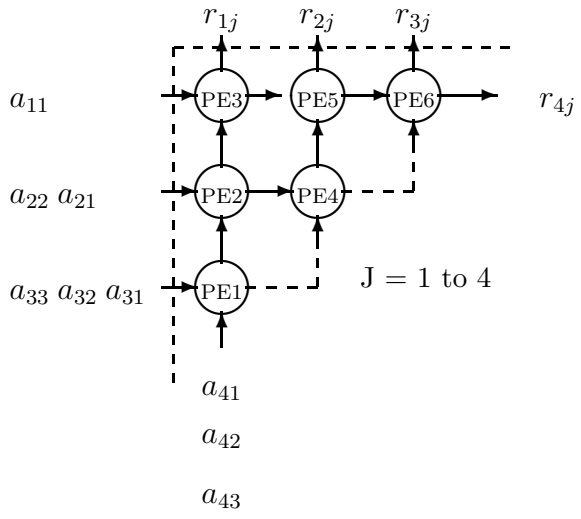
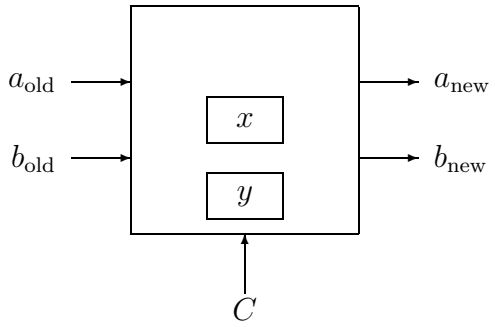


Figure 3: Modified Signal Flow Graph based on Given's Rotation

ing the projection direction [0 1 0] for a 4×4 matrix is shown.

As for any recursion index k, i varies from $N-1$ to k and we are calculating θ from $A_{i+1,k}$ and a_{ik} we can remove PE1, PE5, PE8 and PE10 in this SFG and we can input row $i+1$ and i directly to a PE which calculates θ_i. The modified array will appear as shown in fig. 3.

The design of a single PE is given below.



when C = 0

$$\begin{aligned} x &= a_{\text{old}}/\text{SQR}(a_{\text{old}}^2 + b_{\text{old}}^2) \\ y &= a_{\text{old}}/\text{SQR}(a_{\text{old}}^2 + b_{\text{old}}^2) \\ a_{\text{new}} &= b_{\text{old}} * y + a_{\text{old}} * x \\ b_{\text{new}} &= -b_{\text{old}} * x + a_{\text{old}} * y \end{aligned}$$

when C = 1

$$\begin{aligned} a_{\text{new}} &= b_{\text{old}} * y + a_{\text{old}} * x \\ b_{\text{new}} &= -b_{\text{old}} * x + a_{\text{old}} * y \end{aligned}$$

The time-space diagram given below shows the activities in different PEs in successive clock cycles (fig. 4).

The time complexity of this design is $O(n)$ and number of PEs (hardware complexity) is $O(n^2)$. But as it is clear from the time-space diagram we are getting the same results (for a 4×4 matrix) now only with 6 PEs in contrast to Kung's design which has 10 PEs. Moreover PE utilization is more in this case. Further, if we consider that the time required to compute square root is generally far more than the time required for addition and multiplication, then pipelining of different instances of the problem will be more effective in the design proposed here.

3 Systolic array based on Householder transformation

Householder transformation is used as an elementary step in a number of basic trans-

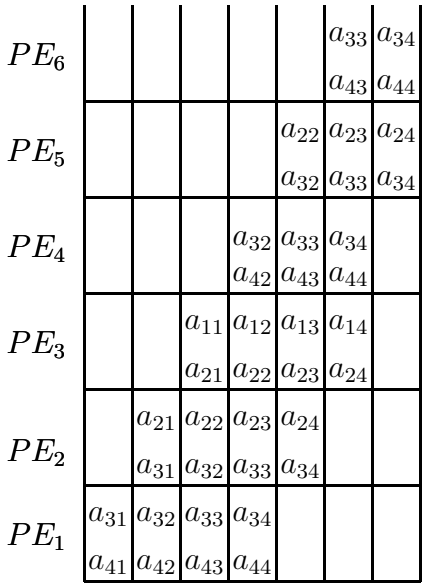


Figure 4: Time space diagram for the first array

formations e.g. for decomposing a matrix A into an orthogonal matrix Q and an upper triangular matrix R (QR decomposition algorithm). The transformation is implemented by first applying the following decomposition.

$$Q(1)A = \begin{bmatrix} x & x & x & \cdots & x \\ 0 & & & & \\ 0 & & & A_s & \\ \vdots & & & & \\ 0 & & & & \end{bmatrix}$$

where $Q(1) = I - 2\frac{uu^T}{u^Tu}$, $u = A_1 + \|A_1\|z$ and A_1 is the first column vector of A, and $Z = [100\ldots0]^T$, then repeatedly applying similar decompositions, namely $Q(i)$,$\forall i = 2, 3, 4 \ldots N-1$,

$$Q = Q(N-1)Q(N-2)\ldots Q(1)$$

The sequential program for the above procedure can be given as below.

For $k = 1$ to $N-1$
  $S(k-1) = 0$
  For $i = k$ to N
    $S(i) = S(i-1) + a(i,k) * a(i,k)$
  $P = \mathrm{SQR}(S(N))$
  $u(k) = P + a(k,k)$
  For $i = k+1$ to N
    $u(i) = a(i,k)$
  $R = 0$
  For $i = k$ to N
    $R = R + u(i) * u(i)$
  For $j = k$ to N
    $v(j) = 0$
    For $i = k$ to N
      $v(j) = v(j) + u(i) * A(i,j)$
  For $i = k$ to N
    For $j = k$ to N
      $A(i,j) = A(i,j) -$
        $(2/R) * u(i) * v(j)$

The time complexity of this sequential program is $O(n^3)$. The single assignment form of this sequential program can be written as follows.

For $k = 1$ to $N-1$
  $S(k, n+1) = 0$
  For $i = N$ to k
    $S(k,i) = S(k,i+1) +$
      $a(k-1,i,k) * a(k-1,i,k)$
  $P(k) = \mathrm{SQR}(S(k,k))$
  $u(k,k) = P(k) + a(k-1,k,k)$
  For $i = k+1$ to N
    $u(k,i) = a(k-1,i,k)$
  $Q(k,k-1) = 0$
  For $i = k$ to N
    $Q(k,i) = Q(k,i-1) +$
      $u(k,i) * u(k,i)$
  For $j = k$ to N
    $v(k,k-1,j) = 0$
    For $i = k$ to N
      $v(k,i,j) = v(k,i-1,j) +$
        $a(k-1,i,j) * u(k,i)$
      $a(k,i,j) = a(k-1,i,j) -$
        $\frac{2}{Q(k,n)} * u(k,i) * v(k,j)$

The single assignment program can be further modified to a form which can be directly

mapped on a DG.

For $k = 1$ to $N - 1$
  $S(k, N + 1, k) = 0$
  For $j = k$ to N
    $v(k, N + 1, j) = 0$
    For $i = N$ to k
    if $(j = k)$ & $(i \neq k)$
      $S(k, i, j) = S(k, i + 1, j)+$
        $a(k - 1, i, j)^2$
      $u(k, i, j) = a(k - 1, i, j)$
    if $(j = k)$ & $(i = k)$
      $P(k, i, j) = \mathrm{SQR}(S(k, i + 1, j)+$
        $a(k - 1, i, j)^2)$
      $u(k, i, j) = P(k, i, j) + a(k - 1, i, j)$
      $Q(k, i, j) = S(k, i + 1, j) + u(k, i, j)^2$
    if $(j \neq k)$
      $u(k, i, j) = u(k, i, j - 1)$
      $Q(k, k, j) = Q(k, k, j - 1)$
      $v(k, i, j) = v(k, i + 1, j)+$
        $a(k - 1, i, j) * u(k, i, j)$
  For $i = k$ to N
    For $j = k$ to N
      $Q(k, i, j) = Q(k, i - 1, j)$
      $v(k, i, j) = v(k, i - 1, j)$
      $a(k, i, j) = a(k - 1, i, j)-$
        $(2/Q(k, i, j)) * u(k, i, j) * v(k, i, j)$

A DG for this single assignment form has the edges from $k - 1$ to k, $j - 1$ to j and bidirectional edges in direction i i.e. edge from $i + 1$ to i, and from $i - 1$ to i. First u, $v(1)$ and Q are calculated in the first column. Then u and Q are propagated in direction j and $v(2) \ldots v(n)$ are calculated in respective columns. Processing proceeds downwards and new elements of array A are calculated and sent to the layer corresponding to recursion index 2. This continues for all layers. The DG for this is given in fig. 5.

We choose the projection direction [1 0 0] and map the activities of all the nodes in same column onto a single PE. The array so obtained is a triangular array with each row corresponding to a recursion index (fig. 6).

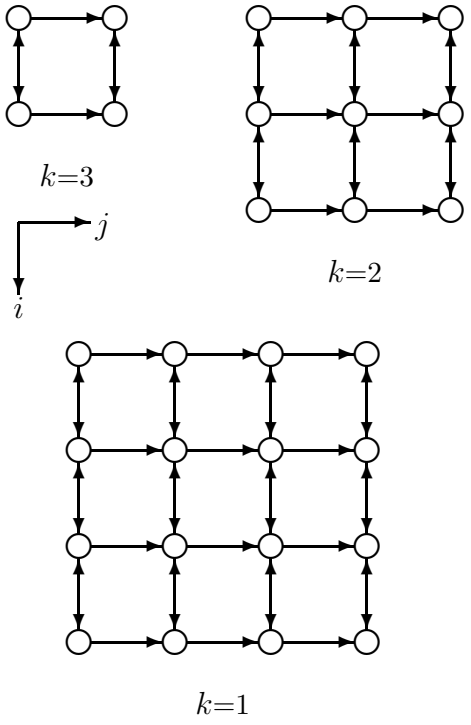


Figure 5: Dependence Graph for the second array

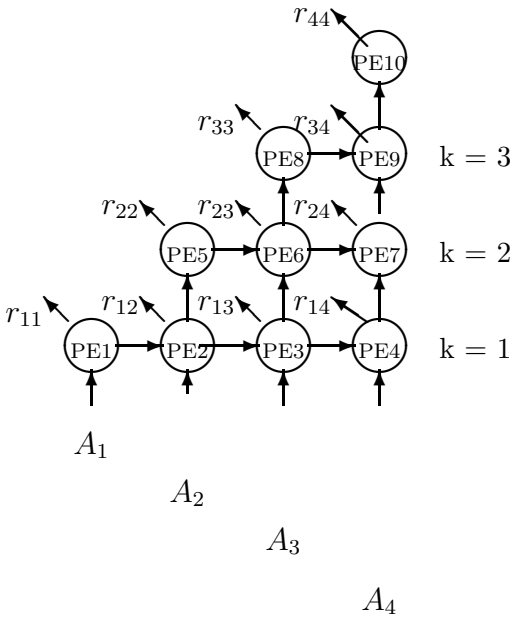


Figure 6: Space Flow Graph for the second array

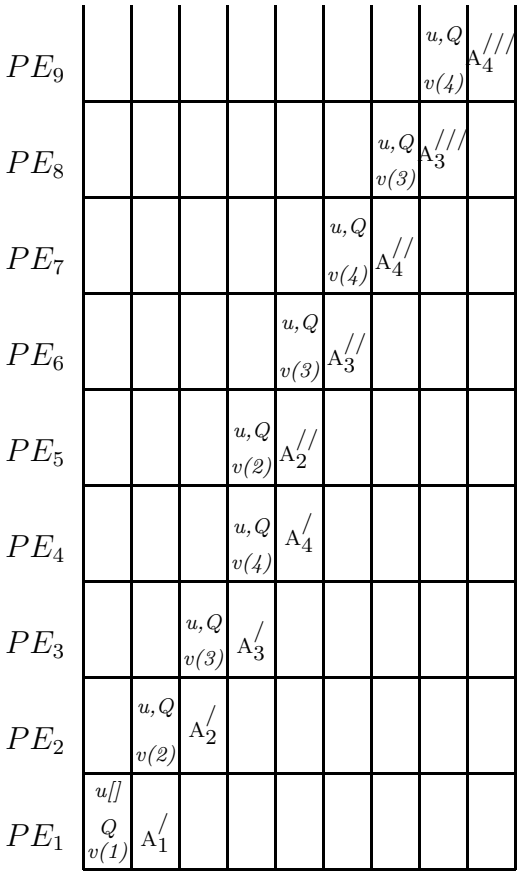
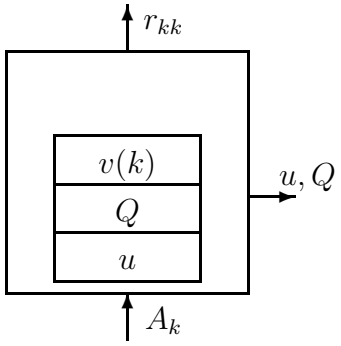


Figure 7: Time space diagram for the second array

Let A_i represent the column i of matrix A. then the activities taking place in each PE can be visualized by the time-space diagram given in fig. 7.

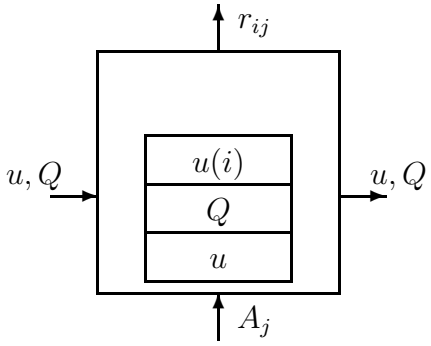
As it is clear from this time-space diagram, next instance of the problem can be sent into the pipeline after two clock cycles and after that all the PEs will be busy all the time (100% PE utilization). For one instance of the problem the activities taking place within a PE are $O(n)$, but they can be further pipelined within a PE. The array starts outputting the next row of the matrix after every three clock cycles from the first result. The hardware complexity in terms of number of PEs is $O(n^2)$. Each PE performs the following operations.

PEs at the diagonal (PE1, PE5, PE8) (fig. 8) performs the following operations.



k is the recursion index
A_k represents column k of matrix A

Figure 8: Diagonal PEs for the second array



A_k represents column k of matrix A

Figure 9: Non-diagonal PEs for the second array

$$
\begin{aligned}
S &= S + a_{ik}^2 \; i = k..N \\
P &= SQR(S) \\
u(k) &= P + a_{kk} \\
u(i) &= a_{ik} \; i = k + 1..N \\
R &= R + u(i) * u(i) \; i = k..N \\
v(k) &= v(k) + u(i) * A(k,i) \; i = k..N \\
r(k,k) &= A(k.k) - (2/R) * u(k) * v(k)
\end{aligned}
$$

k is the recursion index

Non-diagonal PEs (fig. 9) performs the following operations.

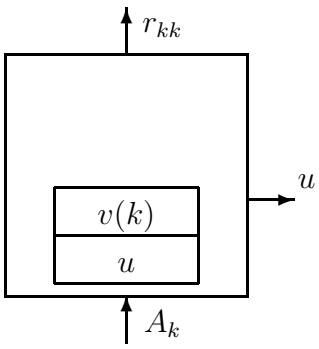
$$
\begin{aligned}
v(j) &= v(j) + u(i) * A(i,j)\ i = K..N \\
A(i,j) &= A(i,j) - (2/R) * u(i) * v(j) \\
&\quad i = k..N
\end{aligned}
$$

4 Householder Reduction

Our third design first reduces the given matrix into hessenberg form and then applies the QR decomposition to it. In many applications the task of QR factorization has to be done many times i.e. finding out eigenvalues of the matrix. Usually, the QR iteration is computationally expensive $O(n^3)$. But if we first reduce the matrix into hessenberg form, the reduction to hessenberg form has complexity $O(n^3)$, and further the complexity of each iteration is $O(n^2)$ for the sequential program. The sequential program to reduce the given matrix into hessenberg form is shown.

For $k = 1$ to $N - 2$
  $u(k) = 0; P = 0$
For $I = k + 1$ to N
  $P = P + A(I,k) * A(I,k)$
  $S = SQR(P)$
  $u(k+1) = SQR((1 + A(k+1,k)/S)/2)$
For $I = k + 2$ to N
  $u(I) = ((A(I,k)/2)/S)/u(k+1)$
For $J = 1$ to N
For $I = k + 1$ to N
  $v(J) = v(J) + u(I) * a(I,J)$
For $I = 1$ to N
For $J = K$ to N
  $A(I,J) = A(I,J) - 2 * u(I) * v(J)$

The systolic array for this sequential program can be obtained in the same way as in the previous design based on the householder transformation. The operations performed by the non-diagonal elements remain same as in the previous design. The operations performed by the diagonal elements (fig 10) are changed as follows.



k is the recursion index

Figure 10: Diagonal PEs for the third array

$$
\begin{aligned}
P &= p + A(I,K) * A(I,K) \\
S &= SQR(P) \\
u(k+1) &= SQR(1 + A(k+1,k)/S) \\
u(I) &= A(I,k)/2/S/u(k+1) \\
&\quad I = k + 2..N \\
v(k) &= v(k) + u(I) * A(I,k) \\
&\quad I = k + 1..N \\
A(k,k) &= A(k,k) - 2 * u(k) * v(k)
\end{aligned}
$$

5 Conclusion

In this paper we proposed three systolic arrays for decomposition of a matrix in its Q-R form. The first array is based on well known Given's rotation method. In contrast to $n(n+1)/2$ PEs required by two existing systolic arrays based on Given's rotation, this array requires only $n(n-1)/2$ PEs and all the PEs are homogeneous making it simpler to implement. Our next design is based on Householder transformation method. As the number of square root operations performed by this array are only $(n-1)$ in comparison to $n(n-1)/2$ in first design we believe that the implementation of this array will be faster than the first one. Moreover if

many instances of the problem are initiated into this array the PE utilization of this array is 100%. As explained our third design first reduces the given matrix into hessenberg form and then applies the QR iteration to it. This array is useful for the applications where QR decomposition has to be performed many times, as the process of finding QR decomposition for a hessenberg matrix is faster.

References

[1] S.Y.Kung; VLSI array processors; Prentice Hall, New Jersey (1988).

[2] Systolic Signal processing Systems; Edited by Earl E. Swartzlander, Marcel Dekker, New York (1987).

[3] Systolic array processors; Edited by J. McCanny, J. McWhirter, Earl E. Swartzlander, Jr., Academic Press, London, (1987)

[4] D. Kahaner, C. Moler and S.Nash; Numerical Metods and Software; Printce Hall, Englewood Cliffs (1989).

[5] R.E.Scraton; Further Numerical Methods in Basic; Adward Arnold, London (1987).

[6] A.Ralston; A First course in Numerical Analysis; McGraw Hill, NewYork (1978).

[7] R.L. Johnston; Numerical Methods - A Software approach; John Wiley, NewYork (1982).

[8] M. Moonen and J. Vandewalle; A Systolic array for Recursive least square computations; IEEE Transactions on signal processing, Vol 41 No 2, Feb. 1993