

An approach to program representation of the R & F systolic LU-factorization algorithm

M.P. Bekakos

*Department of Informatics, Athens University of
Economics and Business, Athens, Greece*

ABSTRACT

In this work, we view systolic algorithms as programs, each consisting of a multiple assignment statement that describes the algorithmic operation and the synchronous data movement between the nodes of the systolic architecture. We carry out the development of the LU-factorization algorithm for band matrices using the R&F technique (Bekakos and Evans [9], [2]) and applying for its design a traditional program development technique based on an invariant.

INTRODUCTION

One of the most significant fields in the area of parallel processing is that of systolic algorithms, which are usually illustrated by snapshots of nodes and lines, descriptions of processing at each node in the snapshot and data movement between nodes. A pictorial representation of an algorithm does not lead itself readily to a proof of correctness, although it suggests that it can be implemented on a VLSI chip.

From all the above it becomes apparent that we do not attempt to propose a VLSI design methodology for systolic algorithms, because we do not consider many of the limitations that would be imposed by any real system.

Although such a sort of proposal concerns a later stage in the design, through the use of a traditional program development technique, for the R&F method, a design methodology seems that could be derived mechanically.

The primary contribution of this work is the representation of the R&F systolic LU-factorization algorithm by a program derived from an invariant.

368 Applications of Supercomputers in Engineering

SYSTOLIC ALGORITHM REPRESENTATION

Multiple Assignment Statements

Our program will make use of multiple assignment statements. A multiple assignment statement of the form,

$$x, y := f(x', y'), g(x', y'),$$

assigns $f(x', y')$ and $g(x', y')$ to x, y , respectively, where x', y' are the values of x, y prior to the execution of the statement. We shall allow the right sides of assignments to be conditional expressions, e.g.

$$x := \begin{cases} 0, & \text{if } a > 0 \\ 1, & \text{if } a \leq 0 \end{cases}$$

by,

$$x := 0 \text{ if } a > 0 \quad \sim \quad 1 \text{ if } a \leq 0.$$

The program will consist of declarations of its variables and their initial values, and one multiple assignment statement. The program execution implies the execution of this statement repeatedly for an infinite number of times. Such a non-terminating execution is convenient for reasoning, although the program may be stopped when the left and right sides of the statement are equal in value, because no further change in variable values will then be possible. Program structuring with one multiple assignment may seem too restrictive; such a program, however, proves to be quite adequate for representing systolic algorithms. Such an assignment may be thought of as a synchronous computation, thus capturing the essence of systolic computations.

Notational Program Development

We shall introduce a variable t , denoting the step number (t is initially *zero* and is increased by *one* in each execution of the statement) and state an invariant relating various data items and t . We shall be guided by the limited fan-in, fan-out and linearity requirements in postulating an invariant, as in Chandy and Mistra [5].

The invariant simplifies the effort in deriving initial and boundary conditions and the effect of statement execution is the invariant preservation when t is increased by *one*.

Finally, we shall use the symbol $||$ (cf. Chandy and Mistra [5]) to break up a multiple assignment statement into its component assignments for further convenience in reading.

THE LU-FACTORIZATION OF A BAND MATRIX

Given a band matrix A , its LU-factorization results in a pair of matrices L (Lower diagonal with 1's on the main diagonal) and U (Upper diagonal), which satisfy the following equations

$$A^0[i, j] = A[i, j]$$

$$A^{r+1}[i, j] = A^r[i, j] - L[i, r] \cdot U[r, j], \quad r \geq 0$$

$$L[i, r] = \begin{cases} 0, & \text{if } i < r \\ 1, & \text{if } i = r \\ A^r[i, r]/U[r, r], & \text{if } i > r \end{cases}$$

$$U[r, j] = \begin{cases} 0, & \text{if } r > j \\ A^r[r, j], & \text{if } r \leq j, \end{cases}$$

$$(A^r[i, j] = A[i, j], \text{ for } r \geq 0).$$

When we apply the rotate and fold concept, i.e., by proceeding from the top and bottom of the matrix simultaneously, we obtain *two* LU-factorization streams functioning concurrently in opposite directions, one from the top downwards and the other vice versa. Certainly, the two factorization streams should confront each other in the centre of the matrix and the degree of difficulty in handling the factorizing procedures in this part is directly dependent upon the size and the semi-bandwidth of the matrix.

We can distinguish two cases for which the corresponding mathematical formulae, which produce matrices L , U using the R&F technique, are given in Bekakos and Evans [2].

Description of the Systolic Network Operation

The architecture of the systolic array, with which the R&F algorithm for the LU-factorization will be matched, must be now determined.

We shall use the notation q , p , for the bottom and top diagonals of A , respectively, which are defined as $p \equiv p' - 1$, $q \equiv q' + 1$, (where p' , q' are the semi-bandwidths of matrix A).

Hence (cf. Bekakos [4]),

$$A[i, r] = 0, \text{ unless } q \geq i - r \geq p.$$

From the above, for band matrices, we can define

370 Applications of Supercomputers in Engineering

$$A^{r+1}[i, j] = \begin{cases} A[i, j], & \text{if } (<i-r> > p), \text{ or } (<r-j> < q) \\ A^r[i, j] - L[i, r] \cdot U[r, j], & \text{otherwise.} \end{cases}$$

The invariant is

$$t = i + r + j \Rightarrow$$

$$[(r \geq 0, i \geq r, j > r \Rightarrow X[<i-r>, <r-k>] = L[i, r])$$

and

$$(r \geq 0, i > r, j \geq r \Rightarrow Y[<i-r>, <r-k>] = U[r, j])$$

and

$$(i \geq 0, j \geq 0, i \geq r, j \geq r \Rightarrow Z[<i-r>, <r-j>] = A^r[i, j]) .$$

Next, we shall examine the initial conditions required to satisfy the invariant and the way that it can be preserved when t is increased by *one*.

The first two conditions, when $t=0$, are satisfied because there are no i, r, j satisfying these conditions. The last condition, for all $i \geq 0, j \geq 0$, can be satisfied by considering $r = -(i + j)$; hence, we have

$$Z[<2i+j>, <-i-2j>] = A^r[i, j] = A[i, j] \quad \{\text{since } j < 0\}.$$

Let us define $u = <i - r>$ and $w = <r - j>$. We shall focus on such cases where $u \geq 0$ and $w \leq 0$, because matrices L, U must be computed, since they are not available initially.

The input in each node is :

$$X[u, w] = L[i, r].$$

For all i, r, t , where $i \geq r \geq 0$ and $t > <1+2r>$, there exist some nodes (u, w) , $u \geq 0, w \geq 0$, such that

$$X[u, w] = L[i, j].$$

We distinguish two cases:

i) When $t > <i+2r>$, then $L[i, r]$ has already been computed and has to be assigned as input to the proper node.

ii) When $t = <i+2r>$, then $L[i, r]$ is to be computed and then assigned as input to the proper node.

Applications of Supercomputers in Engineering 371

Consequently, for the first case {equivalently, $w < 0$ },

$$X[u, w] := L[i, r];$$

hence, we have a shift for input data and the invariant is preserved by: $X[u, w-1] := X[u, w]$.

While for the second case {equivalently, $w = 0$ }, the invariant is preserved by computing $L[i, r]$ according to its defining equation and then assigning it to the proper variable. At the following step, i.e., the $(t+1)^{\text{th}}$ step, the only j satisfying $t+1 = i+r+j$ is $j = r+1$.

At this step we must have

$$X[u, -1] = L[i, r] \text{ and substituting for } L[i, r]$$

$$X[u, -1] := \{L[i, r] = \}$$

$$A^r[i, r] / U[r, r] \text{ if } i > r \sim 1 \text{ if } i = r.$$

From the invariant, at $t = \langle i+2r \rangle$,

$$A^r[i, r] = Z[\langle i-j \rangle, 0],$$

$$\text{Also, at } t = \langle i+2r \rangle \{i > r, j = r\}, U[r, r] = Y[\langle i-r \rangle, 0] .$$

Hence, we have the following assignment (we can rewrite $i < r$, as $u > 0$),

$$X[u, -1] := Z[u, 0] / Y[u, 0] \text{ if } u > 0 \sim 1 \text{ if } u = 0.$$

$$\textbf{Input } Y[u, w] = U[r, j].$$

Similarly, we distinguish two cases:

$$\text{i) } t > \langle 2r+j \rangle \text{ \{equivalently, } u > 0\}, \\ Y[u+1, w] := Y[u, w] \text{ (data shift);}$$

$$\text{ii) } t = \langle 2r+j \rangle \text{ \{equivalently, } u = 0\}, \\ Y[1, w] := A^r[r, j].$$

At step $t = \langle 2r+j \rangle$, $A^r[r, j] = Z[0, \langle r-j \rangle]$; hence,

$$Y[1, w] := Z[0, w].$$

$$\textbf{Input } Z[u, w] = A^r[i, r].$$

From the equations

372 Applications of Supercomputers in Engineering

$$A^{r+1}[i, j] = \begin{cases} A[i, j], & \text{if } (<i-r> > p), \text{ or } (<r-j> < q) \\ A^r[i, j] - L[i, r] \cdot U[r, j], & \text{otherwise.} \end{cases}$$

The invariant is preserved by using similar arguments to derive the assignment:

$$\begin{aligned} Z[u-1, w+1] &:= Z[u, w] \text{ if } u > p \text{ or } w < q \quad \sim \\ Z[u, w] - X[u, w] \cdot Y[u, w] &\text{ if } u \leq p \text{ and } w \geq q. \end{aligned}$$

Program LU {LU-factorization of a band matrix}

initially : $t = 0$

$\quad$ < for all $i \geq 0, j \geq 0 : Z[\langle 2i+j \rangle, \langle -i-2j \rangle] = A[i, j] >$

assign : <for all $u, w ::$

$\quad (\mid \mid X[u, w-1] := X[u, w] \text{ if } w < 0 \quad \sim \quad Z[u, 0] / Y[u, 0] \text{ if } w = 0$

$\quad \text{and } u > 0 \quad \sim \quad 1 \text{ if } w = 0 \text{ and } u = 0$

$\mid \mid Y[u+1, w] := Y[u, w] \text{ if } u > 0 \quad \sim \quad Z[0, w] \text{ if } u = 0$

$\mid \mid Z[u-1, w+1] := Z[u, w] - X[u, w] \times Y[u, w] \text{ if } u \leq p \text{ and } w \geq q \quad \sim$
 $\quad Z[u, w] \text{ if } u > p \text{ or } w < q)$

$\text{if } t = (u-w) \bmod 3 >$

$\mid \mid t := t+1$

end{LU}.

THE R&F LU-FACTORIZATION ALGORITHM MATCHES THE SYSTOLIC ARCHITECTURE

We shall determine the size of the array and show how the R&F algorithm fits the systolic architecture which was previously described. We separate the whole procedure in *odd* and *even* steps.

In case of the *odd* steps, we note that, the element of the main diagonal belongs to the top stream, while the elements of the super- and subdiagonal belong to the bottom stream; this interchanging between the top and bottom stream elements continues in the remaining diagonals, as the bandwidth of the rotated and folded matrix increases, depending upon the element of the main diagonal. The opposite applies for the case of the *even* steps. We shall make use of a static scheme for the representation of each of the factorization streams, which can be conveniently integrated into one scheme.

If we assume the LU-factorization of a matrix A , diagrammatically shown in Fig. 1, we shall specify a three number set, in such an order as element $\{A, L, U\}$, to describe the elements flow in the systolic array as shown in Fig. 2. Note that, the bold and normal typed elements in this Figure belong to the bottom

Applications of Supercomputers in Engineering 373

and top data stream, respectively.

Next, we must determine when we shall have results; in other words, we wish to find T and u, w such that,

$$(t=T) \Rightarrow (Z[u,w] = A[i,j]).$$

We must specify k such as,

$$A^{r+1}[i,j] = A^r[i,j].$$

This holds when

$$\langle i-r \rangle > p \text{ or } \langle r-j \rangle < q,$$

or when

$$L[i,r] = 0 \text{ or } U[r,j] = 0.$$

Hence, the minimum value of r , for which it holds, is given by

$$r^* = \min(\langle i-r \rangle, \langle r-j \rangle) \Rightarrow r^* = \min(\langle i-p \rangle, \langle j+q \rangle) + 1,$$

and from the invariant at $T=i+r^*+j$,

$$Z[\langle i-r^* \rangle, \langle r^*-j \rangle] = A[i,j] \text{ \{this node will give us results\}}.$$

Note that, in case of $q = -p \Rightarrow r^* = \min(i,j) + p + 1$. Hence, the systolic array has a regular diamond structure. To the contrary, for arbitrary p, q the structure is not regular.

CONCLUSIVE REMARKS

The primary goal of this work is to apply a traditional programming technique to develop an efficient systolic algorithm implementing the R&F method. The two factorization streams, however, will confront each other in the centre of the matrix and the degree of difficulty in handling the factorizing procedures in this part is directly dependent upon the size and the semi-bandwidths of the matrix.

It has been assumed that the matrix A has the property that its LU-factorization can be done by Gaussian elimination without pivoting. This is true, for example, when A is a symmetric positive - definite, or an irreducible, diagonally dominant matrix. Usually this condition is not a problem, since most

374 Applications of Supercomputers in Engineering

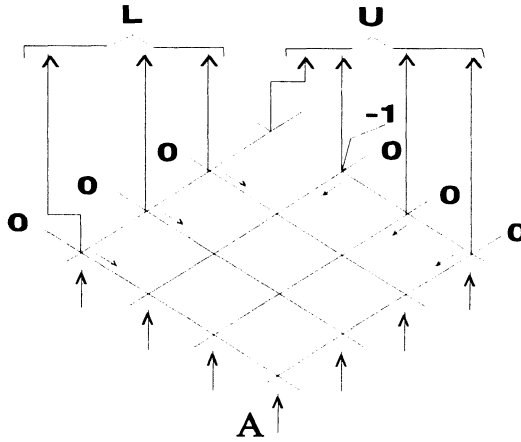


Fig. 1: The Systolic Architecture for the LU-Factorization of a Matrix A (for $p = 3$, $q = -3$).

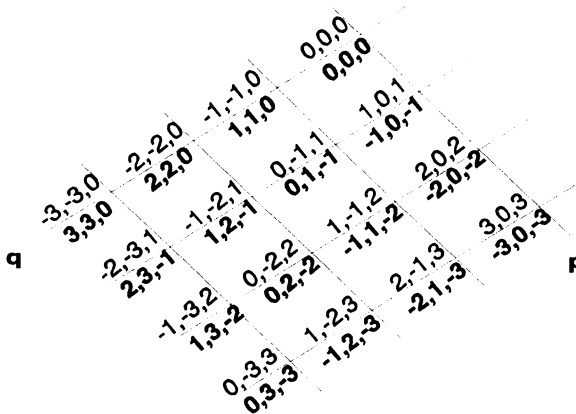


Fig. 2: The Flow of the Elements for the R&F LU-Factorization of a Matrix A (for $p = 3$, $q = -3$).



of the systems encountered in practice are diagonally dominant; on the other hand, however, there is currently no efficient way of incorporating a pivot strategy into existing vector or parallel algorithms.

Once the L and U factor matrices are computed, it is relatively easy to solve the resulting triangular linear systems applying again the R&F technique (cf. Bekakos and Evans [2]).

REFERENCES

1. Papers in journals:

1. Bekakos, M.P., and Evans, D.J., "A 'Rotating' and 'Folding' Algorithm Using a Two-Dimensional 'Systolic' Communication Geometry", *Parallel Computing*, 4, pp. 221-228, Elsevier Science Pub., B.V., North-Holland, 1987.
2. Bekakos, M.P., and Evans, D.J., "Systolic LU-Factorization 'Dequeues' for Tridiagonal Systems", *Int. Journal of Computer Maths.*, Vol. 25, pp. 299-320, Gordon and Breach Science Pub., Inc., U.K., 1988.
3. Bekakos, M.P., and Evans, D.J., "Concurrent Systolization of Quindagonal Linear Systems", *Int. Journal of Computer Maths.*, Vol. 46, No. 3&4, pp. 163-181, Gordon and Breach Science Pub., Inc., U.K., 1993.
4. Bekakos, M.P., "Program Representation of a R&F Systolic Algorithm", Submitted to *Int. Journal of Computer Maths.*, May, 1993.
5. Chandy, K.M., and Mistra, J., "Systolic Algorithms as Programs", *Distributed Computing*, Springer-Verlag, 1, pp. 177-183, 1986.
6. Li, G.J., and Wah, B.W., "The Design of Optimal Systolic Arrays", *IEEE Trans. Comput.*, C-34, pp. 66-77, 1985.

2. Chapter in a book:

7. Kung, H.T., and Leiserson, C.E., "Algorithms for VLSI Processor Arrays", Section 8.3, in: eds. Mead, C., and Conway, L., *Introduction to VLSI Systems*, Addison-Wesley, 1980.

3. Edited book:

8. Kung, H.T., and Lam, M., "An Approach to Automatic Generation of Linear Systolic Array Programs", in *Parallel Processing for Computer Vision and Display*, eds. Dew, P.M., Earnshaw, R.A., Heywood, T.R., Addison-Wesley Pub. Co. Inc., U.K., 1989.



4. Paper in Conference Proceedings:

9. Bekakos, M.P., and Evans, D.J., "The Exposure and Exploitation of Parallelism on Fifth Generation Computer Systems", *Proceedings of the Int. Conf. Parallel Computing '85*, eds. Feilmeier, M., Joubert, G., and Schendel, U., pp. 425-442, Elsevier Science Pub. B.V., North-Holland, 1986.