Proc. Indian Acad. Sci. (Math. Sci.), Vol. 105, No. 1, February 1995, pp. 59–71.

Symmetrizing a Hessenberg matrix: Designs for VLSI parallel processor arrays

F R K KUMAR and S K SEN

Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore 560012, India

MS received 3 November 1993; revised 19 April 1994

Abstract. A symmetrizer of a nonsymmetric matrix A is the symmetric matrix X that satisfies the equation $XA = A^tX$, where t indicates the transpose. A symmetrizer is useful in converting a nonsymmetric eigenvalue problem into a symmetric one which is relatively easy to solve and finds applications in stability problems in control theory and in the study of general matrices. Three designs based on VLSI parallel processor arrays are presented to compute a symmetrizer of a lower Hessenberg matrix. Their scope is discussed. The first one is the Leiserson systolic design while the remaining two, viz., the double pipe design and the fitted diagonal design are the derived versions of the first design with improved performance.

Keywords. Complexity; equivalent symmetric matrix; Hessenberg matrix; symmetrizer; systolic array; VLSI processor array.

1. Introduction

A symmetrizer [3, 7, 14, 16, 19, 20] of an $n \times n$ nonsymmetric matrix A is the solution X satisfying the equations $XA = A^tX$ and $X = X^t$. A symmetrizer is used in transforming a nonsymmetric matrix into an equivalent symmetric matrix [14, 20] whose eigenvalues are the same as those of the nonsymmetric matrix and is useful in many engineering problems, specifically stability problems in control theory and in the study of general matrices [14].

Let

$$B = \begin{bmatrix} b_{11} & b_{12} & 0 & \cdots & 0 \\ b_{21} & b_{22} & b_{23} & \cdots & 0 \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ b_{n-1,1} & b_{n-1,2} & b_{n-1,3} & \cdots & b_{n-1,n} \\ b_{n1} & b_{n2} & b_{n3} & \cdots & b_{nn} \end{bmatrix} \tag{1}$$

be a lower Hessenberg matrix with $b_{i,i+1} \neq 0$ for $i = 1(1)n-1$, where $i = 1(1)n-1$ denotes $i = 1, 2, \ldots, n-1$. Also, let $\mathbf{x}_i$ be the i-th row of the symmetrizer X for $i = n(-1)1$. Then, from $XB = B^tX$, we write the serial algorithm [3] as follows

STEP 1: Choose $\mathbf{x}_n \neq 0$ arbitrarily.

STEP 2: Compute $\mathbf{x}_{n-1}, \mathbf{x}_{n-2}, \ldots, \mathbf{x}_1$ recursively from

$$\mathbf{x}_i = \frac{1}{b_{i,i+1}} \left(\mathbf{x}_{i+1} * B - \sum_{p=i+1}^{n} b_{p,i+1} \mathbf{x}_p \right) \quad i = n-1(-1)1$$

As an illustration, consider

$$B = \begin{bmatrix} 3 & -4 & 0 & 0 \\ -1 & 2 & -4 & 0 \\ 2 & -1 & 6 & -2 \\ 5 & 3 & -2 & 4 \end{bmatrix}$$

Choose $\mathbf{x}_4 = [1 \quad -2 \quad 0 \quad -1]$. $\mathbf{x}_3, \mathbf{x}_2$, and then $\mathbf{x}_1$ are computed following the foregoing algorithm. Hence the symmetrizer is

$$X = \begin{bmatrix} 1{\cdot}8438 & 3{\cdot}8750 & 2{\cdot}0000 & 1{\cdot}0000 \\ 3{\cdot}8750 & 3{\cdot}2500 & 1{\cdot}5000 & -2{\cdot}0000 \\ 2{\cdot}0000 & 1{\cdot}5000 & -5{\cdot}0000 & 0{\cdot}0000 \\ 1{\cdot}0000 & -2{\cdot}0000 & 0{\cdot}0000 & -1{\cdot}0000 \end{bmatrix}$$

It can be seen that the symmetrizer is not unique because if we choose $\mathbf{x}_4 = [1 \ 1 \ 1 \ 1]$ then we get a different X.

2. Leiserson systolic design

The single assignment algorithm [10] for computing a symmetrizer of the Hessenberg matrix B in Equation (1) is as follows.

```
for i:= 1 to n do  for j:= to n do  read B[i,j];
for k:= 1 to n do  read X[n,k];
for i:= n − 1 down to 1 do
begin
  for j:= 1 to n do  Y[i,j,1]:= 0.0;
  for j:= 1 to n do
  begin
    for k:= 1 to n do  Y[i,j,k+1]:= Y[i,j,k] + X[i+1,k]*B[k,j];
    Y[i,n+i+1]:= Y[i,j,n+1];
    for p:= i+1 to n do  Y[i,j,n+p+1]:= Y[i,j,n+p] − B[p,i+1]*X[p,j];
    Y[i,j,2n+2]:= Y[i,j,2n+1]/B[i,i+1];
    X[i,j]:= Y[i,j,2n+2];
    write X[i,j];
  end
end.
```

The implementation [6, 15, 17, 18] of this single assignment code a 4×4 matrix on the Leiserson systolic array depicted in figure 1 is straightforward by using the retiming technique [10]. The allocation of the diagonals of the Hessenberg matrix to the processing cells (Type I) of the linear string of processors is as shown in figure 2. The unspecified output of PE_5 in figure 1 is ignored while its unspecified input is zero.

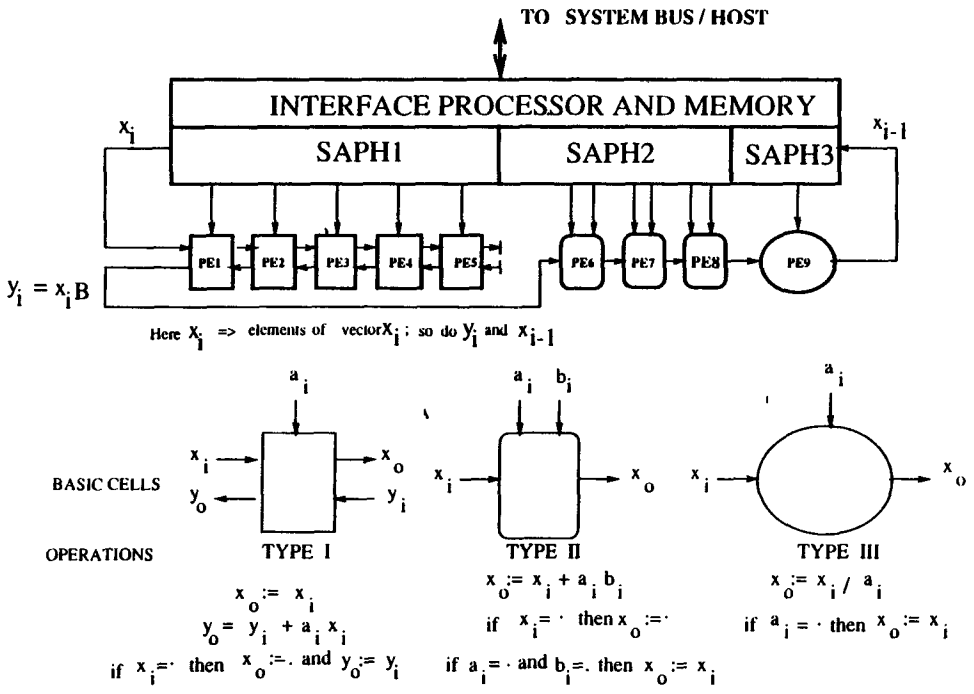


Figure 1. Systolic array cells system for a 4 × 4 matrix symmetrization.

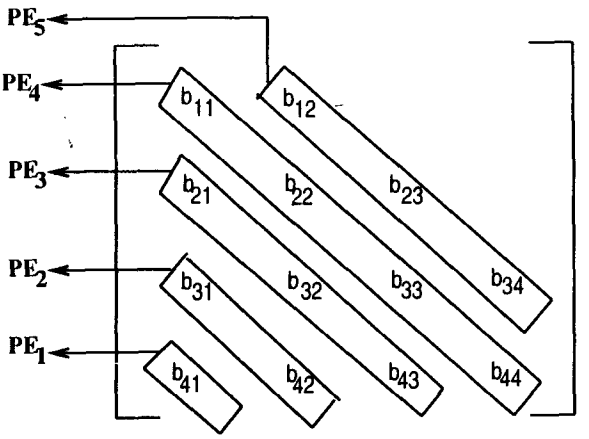
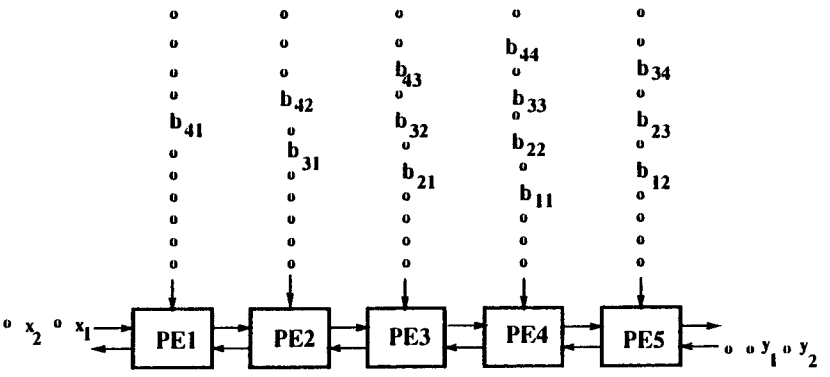


Figure 2. Systolic array cell (Type I) allocation for the diagonals of a 4 × 4 Hessenberg matrix.

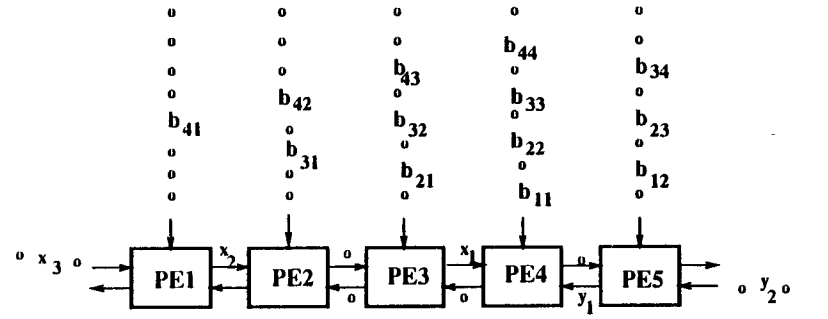
Figure 3 displays how the pumping of the row vector $\mathbf{x}_{i+1}$ and the matrix B into Type I cells is done for the matrix-vector multiplication while figure 4 demonstrates the array consisting of Types II and III cells to generate a symmetrizer row by row. The pumping will be done elementwise in Types II and III cells. The notations $\mathbf{x}_i^n, \mathbf{b}_{ij}, \mathbf{y}_i$, in figure 4, each of which has $2n - 1$ elements including tag bits are given as

$$\mathbf{x}_i^n = [x_n^i \; o \; x_{n-1}^i \; o \; \ldots \; o \; x_1^i]^t,$$

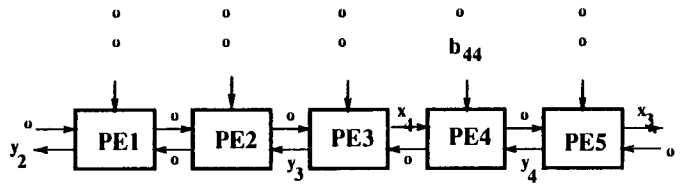
$$\mathbf{b}_{ij} = [b_{ij} \; o \; b_{ij} \; o \; \ldots \; o \; b_{ij}]^t,$$



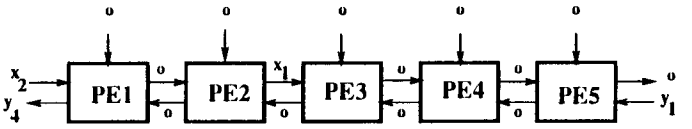
(a) Just before the first time cycle



(b) Just after the third time cycle



(c) Just after the ninth time cycle



(d) Just after thirteenth time cycle

Figure 3. 1-D systolic array for vector Hessenberg matrix multiplication.

and

$$\mathbf{y}_i = \mathbf{x}_i^n B = [y_n^i \; o \; y_{n-1}^i \; o \; \ldots \; o \; y_1^i]$$

This notation is used to conserve space.

A lower (or upper) Hessenberg matrix of order n needs $n+1$ cells of Type I. Denoting these cells $PE_1, PE_2, \ldots, PE_{n+1}$ following the same notation (and connection) as in figure 1, the diagonal consisting of only one element b_{n1} is positioned appropriately

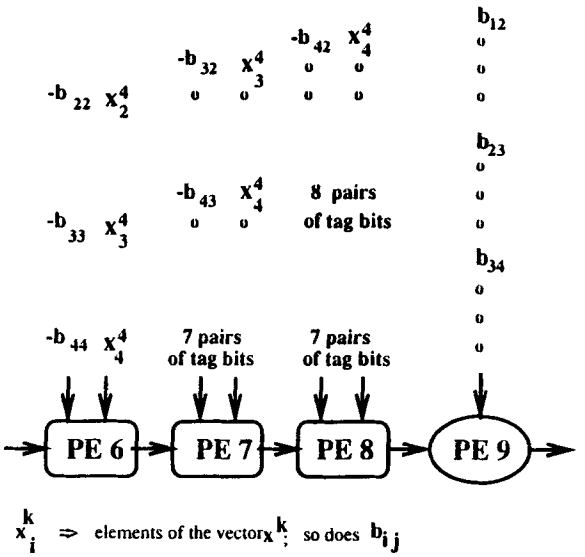


Figure 4. Systolic array for generating a 4 × 4 symmetrizer row by bow.

to be pumped into PE_1, the next diagonal (just above the foregoing diagonal) consisting of the elements $b_{n-1,1}$, b_{n2} is allocated to $PE2$. The third diagonal with elements $b_{n-2,1}, b_{n-1,2}, b_{n3}$ is assigned to PE_3 and so on. Figure 2 illustrates the allocation of the diagonals of the 4 × 4 Hessenberg (symbolic) matrix $B = [b_{ij}]$ to different cells. The generalization to an $n \times n$ matrix is immediate. However, the diagonals could have been allocated in the reverse order, i.e., the diagonal having the elements b_{12}, b_{23}, $b_{34}, \ldots, b_{n-1,n}$ could have been allocated to PE_1, the principal diagonal to PE_2, and so on. Both the allocations are functionally identical. We, however, use the former allocation.

In a one-dimensional Kung-Leiserson systolic array [4], the elements of the vector $\mathbf{x}$ flow from left to right (figure 3) for row vector-Hessenberg matrix multiplication. This array consists of Type I cells, viz., inner product step (ips) cells. The matrix elements flow into the top and the solution elements appear from the left of the cells. Here half the cells are active at any one time. It is, however, possible to orient the data flow so that the cells are all active simultaneously [8, 11]. Note that the number of cells depends on the bandwidth (number of diagonals) of B and not on the size of B. The summation with a negative sign, viz., the result

$$-\sum_{p=i+1}^{n} b_{p,i+1}\mathbf{x}_p$$

of Step 2 of the algorithm (§ 1) is computed using Type II cells as shown in figure 4. The values $\mathbf{y}_n = \mathbf{x}_n B, \mathbf{y}_{n-1} = \mathbf{x}_{n-1} B, \ldots, \mathbf{y}_1 = \mathbf{x}_1 B$ to which the results is to be added are pumped into the cells from the left while the terms of the summation are pumped into them from the top. The single division by $b_{i,i+1}$ is then carried out in Type III cell, one of which only is needed to be used irrespective of the size of B. The elements of the row-vector of the symmetrizer X which are output rhythmically one after the other by this Type III cell are then fed back as indicated in figure 1. This row-vector is then used in the computation of the remaining row-vectors of X recursively.

3. Double pipe and fitted diagonal designs

The Leiserson systolic model [6, 9, 12] needs $2n+1$ cells and $4n+1$ time cycles to obtain a row of the symmtrizer. Here we discuss two designs – one called the double pipe construction method, based on introducing a second pipe while the other, called the fitted diagonal method, on reducing the number of diagonals of the matrix B. While mapping the single assignment algorithm in § 2, the double pipe design aims at minimizing the time complexity while the fitted diagonal design the number of cells.

3.1 *Double pipe construction method*

This method [18] uses $n+1$ cells comprising two pipes – the first one consisting of odd labelled cells $PE_1, PE_3, \ldots$, while the second one the cells $PE_2, PE_4, \ldots$, where n is the order of B; in addition, it uses one adder and one delay cell (figure 5). It computes a symmetrizer B in $\left\lceil \frac{5n+3}{2} \right\rceil$ time cycles where $\lceil\ \rceil$ indicates the upper integral part. The double pipe concept increases cell efficiency and removes tag bits. It minimizes the hardware delay that exists before the start of actual computation. The data flow and the architecture of the $n+1$ cells are illustrated in figures 6 and 8, respectively.

Split the Hessenberg matrix B i.e., write $B = B1 + B2$. $B1$ contains only odd diagonals of B, where the first diagonal contains only the element b_{n1}, the second the elements $b_{n-1,1}, b_{n2}$, and so on while $B2$ the even diagonals. The remaining elements of $B1$ and $B2$ are zero. Since $\mathbf{x}_i B = \mathbf{y}_i$, we have $\mathbf{x}_i(B1 + B2) = \mathbf{y}_i$. If we allow $\mathbf{x}_i B1 = \mathbf{y}_{B1}$ and $\mathbf{x}_i B2 = \mathbf{y}_{B2}$ then $\mathbf{y}_{B1} + \mathbf{y}_{B2} = \mathbf{y}_i$. Figure 6 depicts the flow and computation of $\mathbf{y}_{B1}$ and $\mathbf{y}_{B2}$. The array needs no dummy elements, viz., the tag bits. Pipe 1 contains $\left\lceil \frac{n+1}{2} \right\rceil$ cells. Pipe 2 requires $(n+1) - \left\lceil \frac{n+1}{2} \right\rceil$ ips cells. The time complexity to obtain a row is $2n + \left\lceil \frac{n+1}{2} \right\rceil + 1$, i.e., $\left\lceil \frac{5n+3}{2} \right\rceil$.

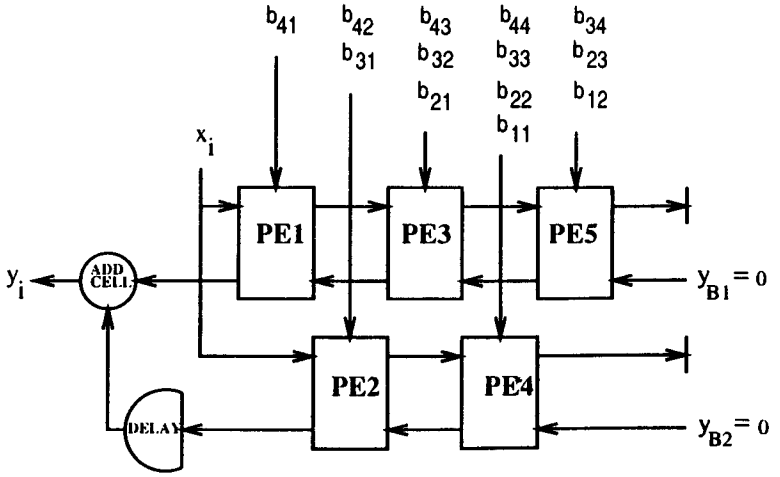


Figure 5. Double pipe array for vector Hessenberg matrix multiplication.

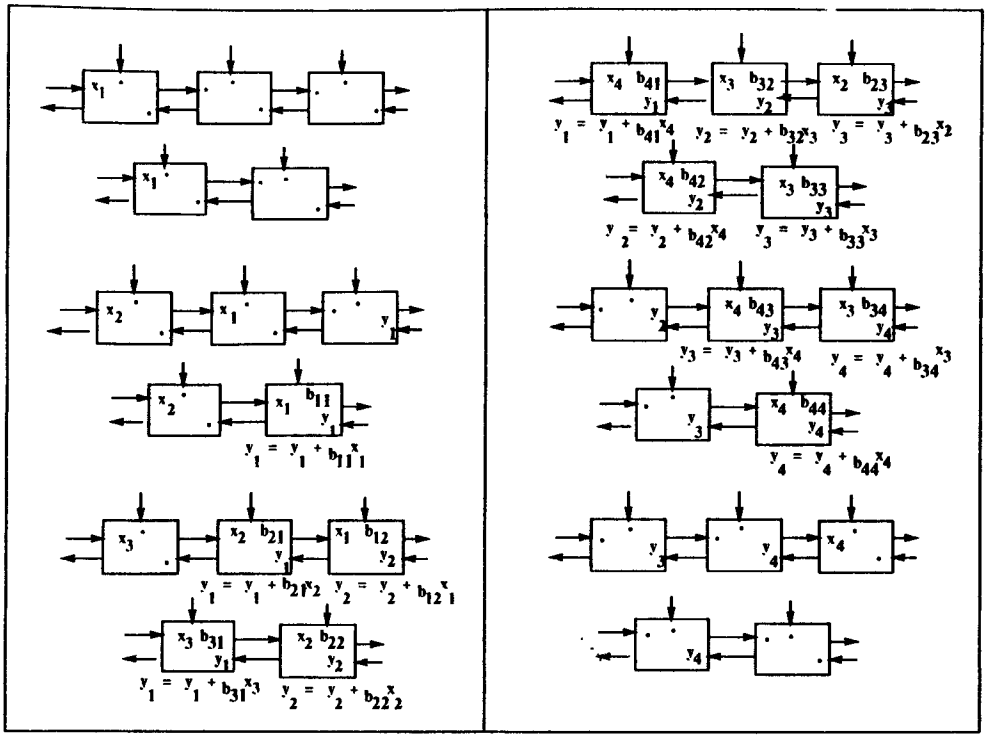


Figure 6. Data flow for double pipe method.

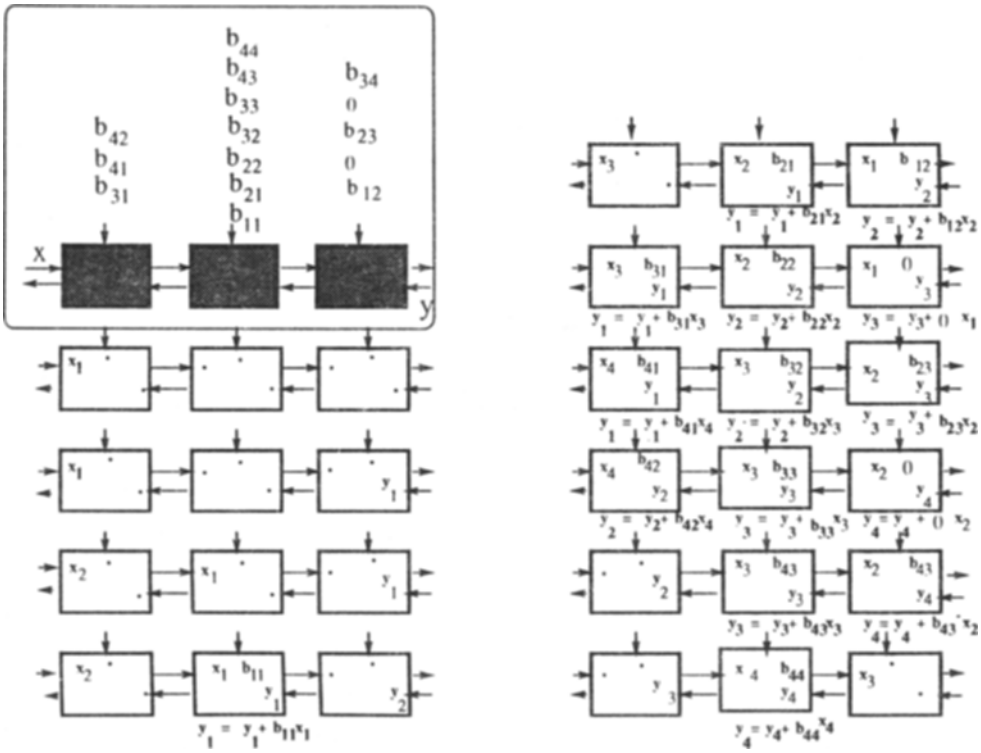


Figure 7. Fitted diagonal method and data flow.

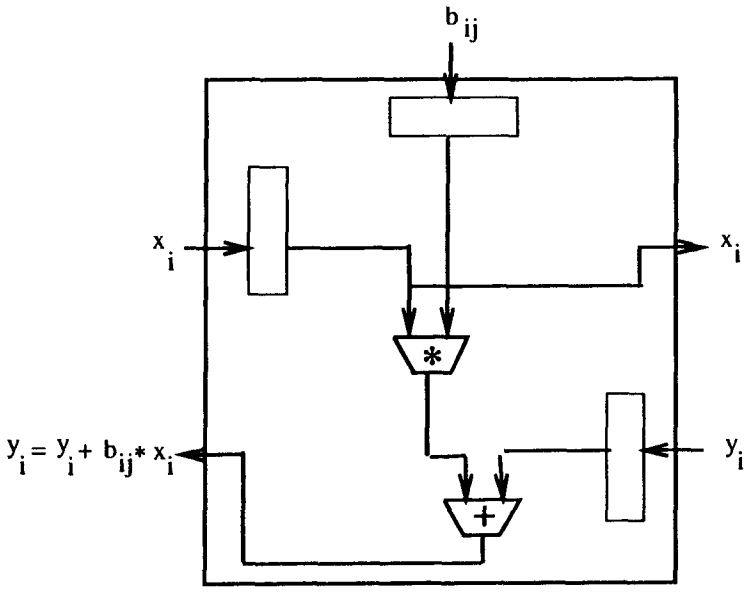


Figure 8. Architecture of ips cells for Leiserson and double pipe methods.

3.2 *Fitted diagonal method*

This method [15] consists in halving the number of diagonals, and hence the number of ips cells used is half of that required in the double pipe method. The number of diagonals can be reduced to $\frac{n+1}{2}$ by fitting two adjacent diagonals into one.

Let $\mathbf{d}_{k+1}$ and $\mathbf{d}_k$ be two adjacent diagonals, of B, of length $k+1$ and k, respectively

$$\mathbf{d}_{k+1} = (b_{n-k,1}, b_{n-k+1,2}, \cdots, b_{n,k+1}),$$
$$\mathbf{d}_k = (b_{n-k+1,1}, b_{n-k+2,2}, \cdots, b_{n,k})$$

A fitted diagonal $\mathbf{d}_f$ is defined by interleaving the elements of $\mathbf{d}_{k+1}$ and $\mathbf{d}_k$, as

$$\mathbf{d}_f = \mathbf{d}_{k+1}.\mathbf{d}_k = (b_{n-k,1}, b_{n-k+1,1}, b_{n-k+1,2}, b_{n-k+2,2}, \cdots, b_{n,k}, b_{n,k+1})$$

where $\text{length}(\mathbf{d}_f) = \text{length}(\mathbf{d}_{k+1}) + \text{length}(\mathbf{d}_k) = 2k+1$.

Therefore, if the bandwidth $n+1$ of B is even then B is transformed to fitted diagonal matrix B_F with bandwith $\frac{n+1}{2}$. For a 5×5 Hessenberg matrix $B = [b_{ij}]$.

$$B_F = \begin{bmatrix} b_{11} & b_{12} & & & & & & & \\ b_{31} & b_{21} & 0 & & & & & & \\ b_{51} & b_{41} & b_{22} & b_{23} & & & & & \\ & 0 & b_{42} & b_{32} & 0 & & & & \\ & & & b_{52} & b_{33} & b_{34} & & & \\ & & & & b_{53} & b_{43} & 0 & & \\ & & & & & & b_{44} & b_{45} & \\ & & & & & & & b_{54} & \\ & & & & & & & & b_{55} \end{bmatrix}$$

If the bandwidth is odd then B_F will have $\left\lceil \frac{n-1}{2} \right\rceil + 1$ fitted diagonals where the last diagonal is fitted with one additional diagonal of $\frac{n}{2}$ zero elements. For a 4×4 Hessenberg matrix B,

$$B_F = \begin{bmatrix} b_{11} & b_{12} & & & & & \\ b_{31} & b_{21} & 0 & & & & \\ & b_{41} & b_{22} & b_{23} & & & \\ & & b_{42} & b_{32} & 0 & & \\ & & & & b_{33} & b_{34} & \\ & & & & & b_{43} & \\ & & & & & & b_{44} \end{bmatrix}$$

Figure 7 illustrates the fitted diagonal method for a 4×4 Hessenberg matrix. A reduction in the number of PEs in this method necessitates some minor modifications to the ips cells. Each of the input vectors $\mathbf{x}_{i+1}$ and that of the output vectors $\mathbf{x}_i = \mathbf{y}$ must be kept in each of the $\left\lceil \frac{n+1}{2} \right\rceil$ PEs for two time cycles as shown in figure 9. The time complexity is the same as that for Leiserson systolic model but the number of PEs is $\left\lceil \frac{n+1}{2} \right\rceil + n$, where the last n PEs do the same job as the last n PEs in Leiserson systolic model of figure 1. This number is about 75% of those required for the conventional Leiserson systolic model. If $(n+1)$ is odd the diagonal left is fitted

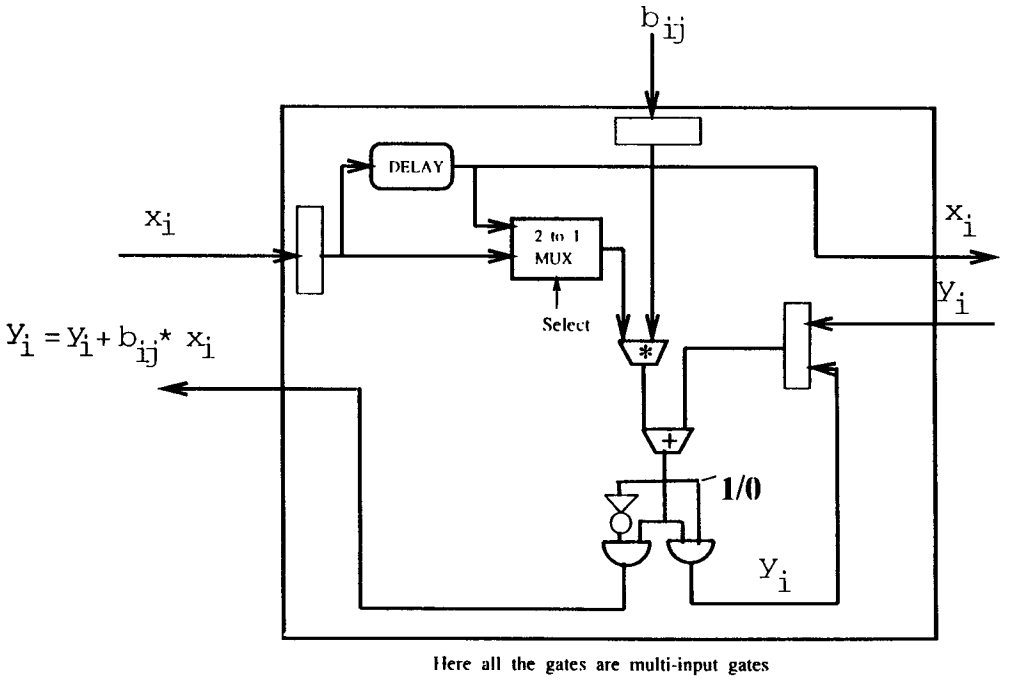


Figure 9. Architecture of ips cell for fitted diagonal method.

Table 1. Time complexity for a row and number of PEs for the designs.

Method	Number of PEs	Time complexity (for computing a row)
1. Leiserson Systolic Method	$w_1 = 2n + 1$	$w_1 + 2n$
2. Double Pipe Construction	$w_2 = 2n + 3$	$w_2 + \left\lceil \frac{n+1}{2} \right\rceil$
3. Fitted Diagonal	$w_3 = \left\lceil \frac{n+1}{2} \right\rceil + n$	$w_3 + 2n$

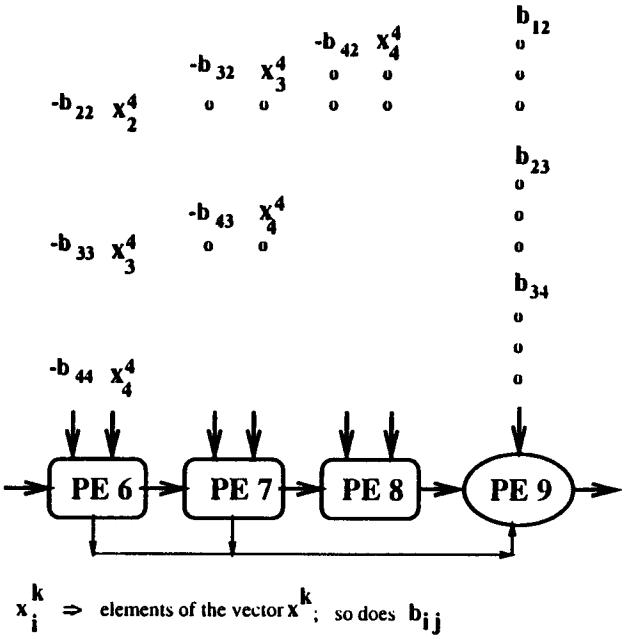


Figure 10. Modified systolic array for generating a 4 x 4 symmetrizer row by row.

with an additional diagonal of null elements. However this reduction in the number of processors needs some minor modification required for the processing elements. The first $(n + 1)$ PEs owing to the elements of vectors **x** and **y** must be kept in each processor for two clock cycles. The time complexity is the same as that of Leiserson systolic model but the number of processors is $\left\lceil \frac{n+1}{2} \right\rceil + n$ which is roughly half of that for conventional Leiserson systolic model.

We present, in table 1, a comparison of time complexity to compute a row of the symmetrizer and number of PEs for the proposed three designs.

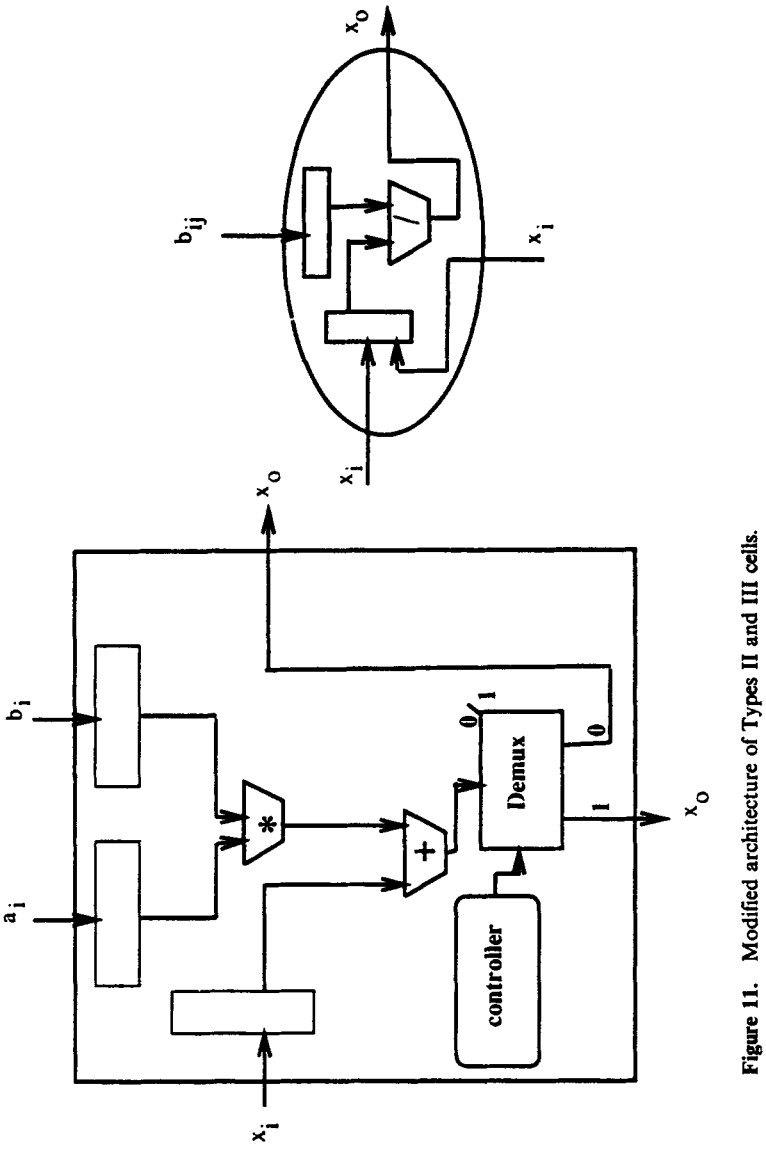


Figure 11. Modified architecture of Types II and III cells.

4. Scheduling and total time complexity

In the Leiserson systolic model, computation of a row of a symmetrizer requires $2n + w_1$ time cycles (where $w_1 = 2n + 1$). Repeating this process for all the rows independently, the total number of time cycles required is $(n-1)(2n + w_1)$. Even though this total number of time cycles is $O(n^2)$, it is still expensive. After $w_1 + n$ time cycles Type I cells (figure 1) are totally idle. A new pumping process is scheduled every $w_1 + n$ time cycles. Therefore, the total number of time cycles to obtain the symmetrizer is $T_1 = (n-1)(n + w_1) + n$ which reduces the number of time cycles by $n^2 - 2n$.

The same number of time cycles T_1 is required in fitted diagonal method even though, in this case, number of PEs is reduced by $\frac{n+1}{2}$, compared to that in Leiserson systolic model. Similarly, in the double pipe construction method, the symmetrizer is computed with the number of time cycles $T_2 = (n-1)\left(\left\lceil \frac{n+1}{2} \right\rceil + (n+3)\right)$.

If we use programmable systolic chip, then Types II and III cells are modified as in figure 10 and the cells architecture is as depicted in figure 11; Type II cells (except the last) have two output gates. The switch value is always assigned zero. The controller sets one for particular clock counter values, e.g., for Leiserson model of 4×4 matrix symmetrization, the processor PE6 controller sets the switch value one from 6th time cycle to 12th time cycle so that the data is pumped to the division cell directly. Type III cell gets input from any one of the gates. This modification reduces the tag bits in Type II and III cells. It also reduces the time complexity by $(n-2)(n-3)/2$.

5. Conclusions

The systolization procedures, i.e., all the three designs can also be easily extended to the general serial algorithm [14] to compute a symmetrizer of an arbitrary square matrix. The bandwidth will, however, be more. We hope that such a systolization will enormously reduce the complexity of computing an error-free symmetrizer [19, 20]. This error free symmetrizer will produce a more accurate equivalent symmetric matrix [14, 19] than what an approximate one does. It can be seen that when a real non-symmetric matrix has one or more pairs of complex eigenvalues then the equivalent symmetric matrix will be a complex one. Jacobi-like methods [1, 2, 5, 13] have been developed for computing eigenvalues, some of which are complex, of a complex symmetric matrix. These methods obviously make use of the "symmetry" property which results in a significant reduction in computation.

Acknowledgement

The authors thank the referee for his comments which have significantly helped in revising the paper.

References

[1] Anderson P and Loizou G, On the quadratic convergence of an algorithm which diagonalizes a complex symmetric matrix, *J. Inst. Math. Its Appl.* **12** (1973) 261–271

[2] Anderson P and Loizou G, A Jacobi-type method for complex symmetric matrices (Handbook), *Numer. Math.* **25** (1976) 347–363

[3] B N Datta, An algorithm for computing a symmetrizer of a Hessenberg matrix, (unpublished)

[4] Dew P M, VLSI architectures for problems in numerical computation, (ed.) D J Paddon, Supercomputers and Parallel Computation, New Series No. 1 (ed.), The Institute of Mathematics and Its Application Series, 1984

[5] Eberlein P J, On the diagonalization of complex symmetric matrices, *J. Inst. Math. Its Appl.* **7** (1971) 377–383

[6] Evans D J, Designing efficient systolic algorithms for VLSI parallel processor arrays, Parallel Architecture and Computer Vision, 1988

[7] Krishnamurthy E V and Sen S K, Numerical algorithms: Computations in science and engineering (1993) (New Delhi: Affiliated East-West press)

[8] Kung H T, Why systolic architectures?, *IEEE Comput.* **16** (1982) 37–46

[9] Kung H T and Leiserson C E, Systolic arrays (for VLSI), (eds) I S Duffand and G W Stewart *Sparse Matrix Proceedings* 1978, 256–82; *SIAM* (1979)

[10] Kung S Y, VLSI Array processors (1988) (New Jersey: Prentice-Hall, Englewood Cliffs)

[11] Kung S Y, Arun K S, Gal-Ezer R J and Bhaskar Rao D, Wavefront array processor: language, architecture, and applications, *IEEE Trans. Comput.* **C31** (1982) 1054–1066

[12] Mead C and Conway L, Introduction to VLSI systems (1980) (Reading, Massachusetts: Addison-Wesley)

[13] Seaton J J, Diagonalization of complex symmetric matrices using a modified Jacobi method, *Comput. J.* **12** (1969) 156–157

[14] Sen S K and Venkaiah V Ch, On computing an equivalent symmetric matrix for a nonsymmetric matrix, *Int. J. Comput. Math.* **24** (1988) 169–80

[15] Suros R and Montagne E, Optimizing systolic networks by fitted diagonals, *Parallel Computing* **4** (1987) 167–174

[16] Taussky O, The role of symmetric matrices in the study of general matrices, *Linear Algebra Appl.* **5** 1(1972) 147–154

[17] Ullman D J, Computational Aspects of VLSI, (1984) (Standford Univ.: Computer Science Press)

[18] Uwe S and Lother T, Linear systolic arrays for matrix computations, *J. Parallel and Distributed Computing* **7** (1989) 28–39

[19] Venkaiah V Ch and Sen S K, Computing a matrix symmetrizer exactly using modified multiple modulus residue arithmetic, *J. Comput. Appl. Math.* **21** (1988) 27–40

[20] Venkaiah V Ch and Sen S K, Error-free symmetrizers and equivalent symmetric matrices, *Acta Applicande Mathematicae* **21** (1990) 291–313