Parallel Computing 4 (1987) 239–258
North-Holland

VLSI systems for band matrix multiplication *

Kam Hoi CHENG ** and Sartaj SAHNI

Computer Science Department, University of Minnesota, Minneapolis, MN 55455, U.S.A.

Received August 1985
Revised January 1986

Abstract. We examine several VLSI architectures and compare these for their suitability for various forms of the band matrix multiplication problem. The following architectures are considered: chain, broadcast chain, mesh, broadcast mesh and hexagonally connected. The forms of the matrix multiplication problem that are considered are: band matrix × vector and band matrix × band matrix. Metrics to measure the utilization of resources (bandwidth and processors) are also proposed. An important feature of this paper is the inclusion of correctness proofs. These proofs are provided for selected designs and illustrate how VLSI designs may be proved correct using traditional mathematical tools.

Keywords. VLSI systems, systolic systems, band matrix multiplication.

1. Introduction

Several authors have considered VLSI architectures for band matrix multiplication. Kung [7] has proposed chain and hexagonal processor array architectures for band matrix multiplication. He spawned considerable further work in the development of VLSI systems for specific problems. A bibliography of over 150 research papers dealing with this subject appears in [8]. [3,4,12,14] are references that deal specifically with some form of the band matrix multiplication problem.

In this paper, we reconsider the following forms of the band matrix multiplication problem: band matrix × vector and band matrix × band matrix. The first of these forms finds significant application in the solution of elliptic partial differential equations by iterative finite difference or finite element methods [2]. The second form has become a classical problem for systolic system design. It was the first problem studied for the hexagonal architecture [9].

For each of the above forms of the band matrix multiplication problem, we consider how the multiplication problem can be solved in VLSI. For this, we consider various VLSI architectures. The major architectures considered are:

(a) *Chain*: This architecture consists of some number of processing elements (PEs) connected together as in Fig. 1(a). PEs may directly communicate with neighbouring PEs only. Input/output may be performed through some subset of the PEs.

(b) *Broadcast chain*: A broadcast line is drawn as a continuous line through all the PEs that are to receive broadcast data. Figure 1(b) shows a broadcast chain with one broadcast line. Data put onto the broadcast line is assumed to reach all PEs on the line in one time unit.

(c) *Mesh*: A mesh is a two-dimensional arrangement of PEs with nearest neighbor connec-

* This research was supported in part by the National Science Foundation under grant MCS-83-05567.
** Current address: Computer Science Department, University of Houston, Houston, Texas, U.S.A.

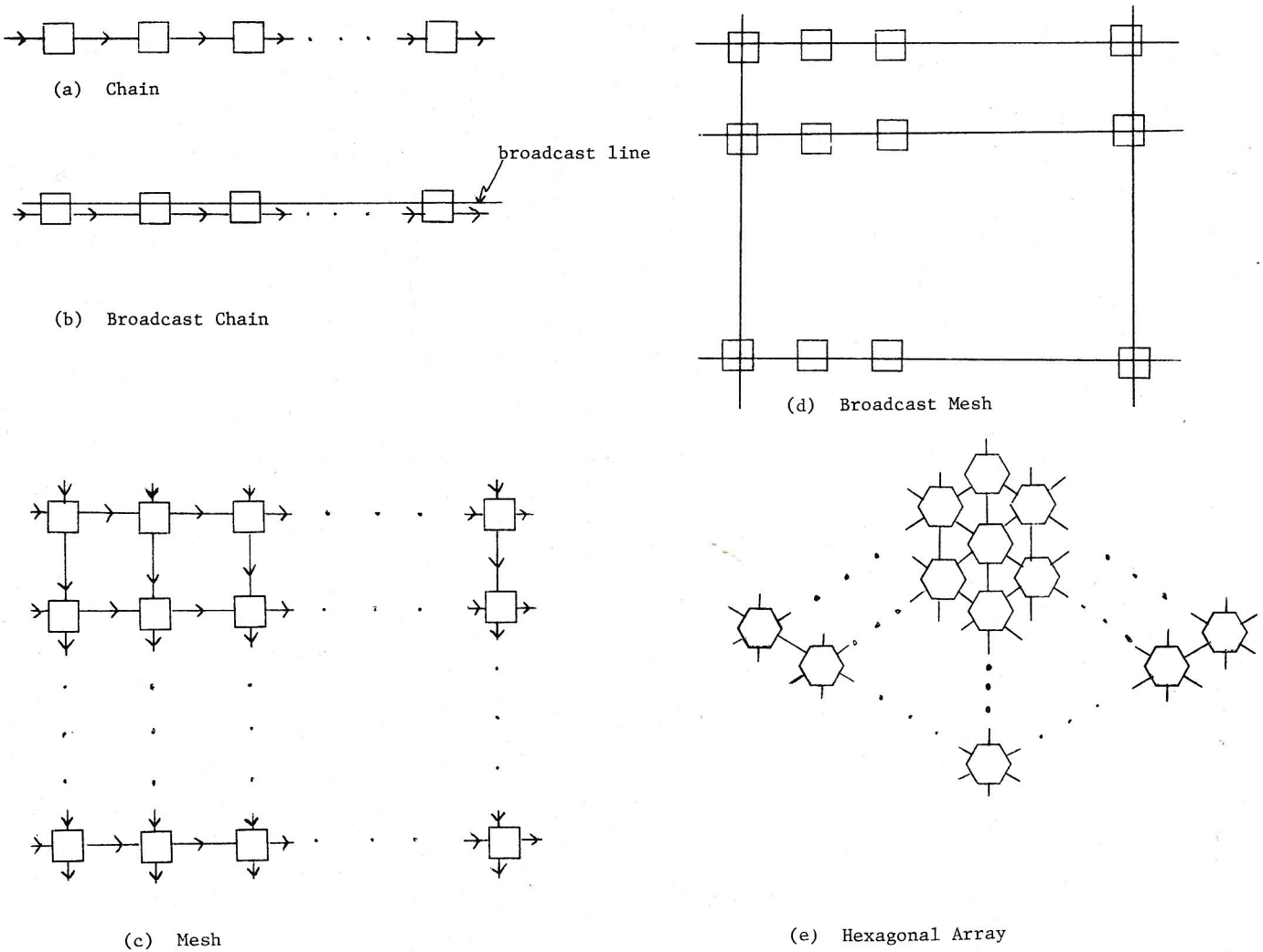
(a) Chain
broadcast line
(b) Broadcast Chain
(d) Broadcast Mesh
(c) Mesh
(e) Hexagonal Array

Fig. 1.

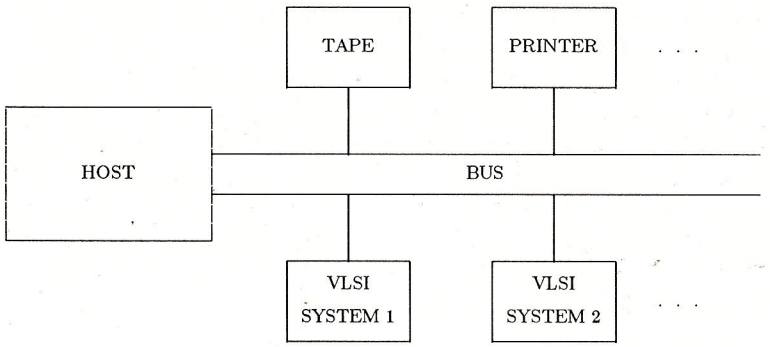

Fig. 2.

tions (see Fig. 1(c)). Input/output can be performed only through some subset of the boundary PEs.

(d) *Broadcast mesh*: This is a mesh with broadcast lines added. The broadcast lines may span rows or columns as in Fig. 1(d).

(e) *Hexagonal processor array*: This is a two-dimensional PE arrangement in which each PE has 6 nearest neighbors and connections to each (Fig. 1(e)). Once again, input/output can be performed only via the PEs at the boundary.

In addition to these six architectures, variations of some are also considered. We make the assumption that the VLSI system for band matrix multiplication will be connected to the bus of the host computer as a peripheral (Fig. 2).

The design of a VLSI system should take the following into account:

(1) *Processors*: how many processors are used in the VLSI system? This figure is denoted by P.

(2) *Bus bandwidth*: the maximum amount of data to be transmitted between the host and the VLSI system in any cycle. This figure is denoted by W.

(3) *Speed*: how much time does the VLSI system need to complete its task? This time may be decomposed into the times τ_C (time for computations) and τ_D (time for data transmissions both within the VLSI system and between the host and the VLSI system). The total time taken by the VLSI system will range from $\max\{\tau_C, \tau_D\}$ (corresponding to the case of total overlap between computations and data transmission) to $\tau_C + \tau_D$ (corresponding to the case of no overlap). In our analysis, we shall measure only the number, T_C, of computation steps and the number, T_D, of data movement steps. Note that $\tau_C = \alpha T_C$ and $\tau_D = \beta T_D$ where α and β are technology dependent.

One may expect that by using a very high bandwidth B and a large number of processors P, we can make T_C and T_D quite small. So, T_C and T_D are not in themselves very good measures of the effectiveness with which the resources B and P have been used.

Let O denote the time spent for arithmetic operations (i.e., computation) by a single processor algorithm. The ratio

$$R_C = P * T_C/O$$

measures the effectiveness of processor utilization. We see that $R_C \geqslant 1$ for every VLSI design that is based on this single processor algorithm. Three conditions are necessary to guarantee full utilization of the processors, namely:

(1) all processors are used all the time,

(2) all computations are useful in contributing to the final result,

(3) no computation is performed more than once.

If during any time cycle that a computation is performed, all the above three conditions are satisfied, then $P * T_C$ will be exactly equal to O and $R_C = 1$. In all other cases, we see that $R_C > 1$. As an example, consider the problem of multiplying two $n \times n$ matrices X and Y to get Z. Each element of Z is the sum of n products. We shall count one multiplication and addition as one arithmetic (or computation) step. Hence, $O = n^3$. (We shall consider VLSI algorithms based on the classical $O(n^3)$ matrix multiplication algorithm only.) If $P = n$, then $T_C \geqslant n^2$.

Let D denote the total amount of data that needs to be transmitted between the host and VLSI system for a particular problem. The ratio

$$R_D = W * T_D / D$$

measures the effectiveness with which the bandwidth W has been used. Clearly, $R_D \geqslant 1$ for every VLSI design. As an example, consider the multiplication of two $n \times n$ matrices. The host needs to send $2n^2$ elements to the VLSI system and receive n^2 elements back. So, $D = 3n^2$. With a bandwidth of n, T_D must be at least $3n$. T_D will exceed $3n$ if the bandwidth is not used to capacity at all times.

In evaluating a VLSI design, we shall be concerned with T_C and T_D and also with R_C and R_D. We would like R_C and R_D to be close to 1. Finally, we may combine the two efficiency ratios R_C and R_D into the single ratio $R = R_C * R_D$. A design that makes effective use of the available bandwidth and processors will have R close to 1.

Huang and Abraham [4] have proposed an efficiency measure that is identical to R when $T_D = T_C$. They do not, however, decompose their measure into the constituent components R_C and R_D. For the algorithms they study, this decomposition is unnecessary as $T_C = T_D$. More specifically, their algorithms and those of [7–11] have the property that each cycle of the algorithm is comprised of some computation and data movement. Some of the algorithms we study in this paper violate this property. These algorithms contain several cycles of data preload and output during which no computation is performed. In many technologies these data movement cycles will be significantly faster than the computation and data move cycles. So, it is not reasonable to simply add together both types of cycles. However, in the case of these algorithms too, a good design will have R_C and R_D close to 1.

For each of the designs considered in this paper, we compute R_C, R_D and R. In several cases, our designs have improved efficiency ratios than all earlier designs using the same model. In comparing different architectures for the same problem, one must be wary about over emphasizing the importance of R_C, R_D and R. Clearly, using $P = 1$ and $W = 1$, we can get $R_C = R_D = R = 1$ and no speedup at all. So, we are really interested in minimizing T_C and T_D while keeping R close to 1.

We wish to caution the reader concerning the significance of R. R is only a measure of the effectiveness with which the resources: bandwidth and processors are used. If our objective is simply to minimize R, then a single processor design with $W = 1$ does this. This design, however, gives us no speedup. Our true objective is to get a high throughput while using the available resources effectively. If we have a VLSI design with a high R value, then this is a clue to us that the design can probably be improved. So, we may use R to measure how good a job we have done in exploiting the given architecture. Another point to keep in mind is that our metric, R, does not include a measure of the complexity of the individual processors and the interprocessor connections. It appears to be quite difficult to include these in any realistic manner.

An important feature of this paper is the inclusion of correctness proofs. These proofs are provided for selected designs and illustrate how VLSI designs may be proved correct. Remaining proofs are easily obtained. We use neither the s-transform notation developed in [5,6,10,15]

Table 1
Symbols used

Symbol	Meaning	Symbol	Meaning
A	input $n \times n$ matrix, $c = Ab$ or $C = AB$	T_C	computation time
b	input vector, $c = Ab$	T_D	data transmission time
B	input $n \times n$ band matrix, $C = AB$	R_P	processor utilization efficiency
c	product vector, $c = Ab$	R_W	bandwidth utilization efficiency
C	output $n \times n$ product matrix, $C = AB$	R	$R_P * R_W$
D	amount of data to be transmitted	S	$\min\{n, w_A + w_B - 1\}$
m	$\min\{w_A, w_B\}$	w	bandwidth of matrix A or $w_A + w_B$
M	$\max\{w_A, w_B\}$	w_A	bandwidth of A
n	A is an $n \times n$ matrix	w_B	bandwidth of B
O	computation time by a one processor algorithm	W	input/output bandwidth
P	number of processors		

nor the formal model developed in [3]. It is our contention that VLSI designs of the type developed here can be proved correct using traditional mathematical methods.

Table 1 collects together all the symbols used in subsequent sections and provides a brief description of each.

2. Band matrix × vector multiplication

2.1. The problem

Input: An $n \times n$ matrix A with bandwidth w ($w \ll n$) and a column vector b.
Output: A column vector c such that $c = Ab$.
Parameters: $O \sim wn$, $D \sim (w+2)n$.

2.2. Chain with O(n) bandwidth

The architecture and input structure are shown in Fig. 3. PE i computes c_i. w_1 and w_2 are, respectively, the number of diagonals below and above the main diagonal. As can be seen, the a's are input by diagonals starting with the lowermost one. Algorithm 2.1. (see Fig. 4) describes the working of the PEs.

Correctness. The correctness of Fig. 3 may be established in the following way. Let $a(i, t)$ and $b(i, t)$ denote the tth a and b values to enter PE i. Then from Fig. 3, we see that $a(i, t) = a_{i,i-w_1+t-1}$ and $b(i, t) = b_{i-w_1+t-1}$, $1 \leqslant i \leqslant n$, $1 \leqslant t \leqslant w$ (the defined a_{ij}'s and b_j's are extended so that $a_{ij} = b_j = 0$ whenever $j \leqslant 0$ or $j > n$). As t ranges from 1 to w, the w diagonals of A enter the VLSI system (from lowermost to uppermost). The computed c_i is $\sum_{t=1}^{w} a_{i,i-w_1+t-1} * b_{i-w_1+t-1}$, which is readily seen as being correct.

Performance. We see that $P = n$, $W = n + 1$, $T_C = w$, $T_D = w + 2$, $R_P \sim 1$, $R_W \sim 1 + 1/n \sim 1$, $R = 1 + 1/n \sim 1$ (when n is large).

2.3. Chain with O(1) bandwidth

The architecture is similar to that of Fig. 3 except that input and output can be performed

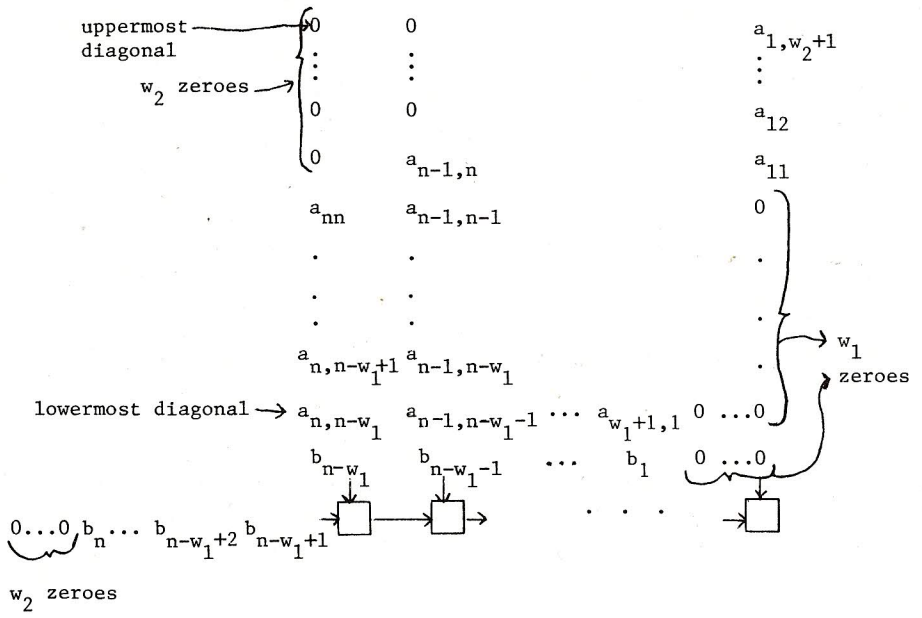

Fig. 3.

through the leftmost PE only. The data is input in the order needed by the design of Section 2.2. The row of b's and the lowermost diagonal of A are entered first. Next, b_{n-w_1+1} and the next diagonal of A are entered, and so on.

Performance. For this new scheme, $p = n$, $W = 1$, $T_C = w$, $T_D = (w+2)n$, $R_P \sim 1$, $R_W \sim 1$, $R \sim 1$.

2.4. Bidirectional chain with O(w) bandwidth

Kung [7] and Leiserson [12] have proposed the bidirectional chain architecture of Fig. 5. The performance figures for this design are $P = w$, $W = (w/2+1)$, $T_C \sim 2n + w$, $T_D \sim 2n + w$, $R_P \sim 2$, $R_W \sim 1$ and $R \sim 2$. Leiserson [12] points out that pairs of processors can be combined into 1 as only half the processors are active at any time instance. When this is done, R_P and R become 1.

---

input the row of b's from the top;
input the first row of a's from the top;
$c \leftarrow a * b$;
for $i \leftarrow 2$ **to** n **do**
  input next row of a's from top;
  input a b from the left shifting all other b's right by one;
  $c \leftarrow c + ab$;
end
output the n c's from the top;

Fig. 4. Algorithm 2.1.

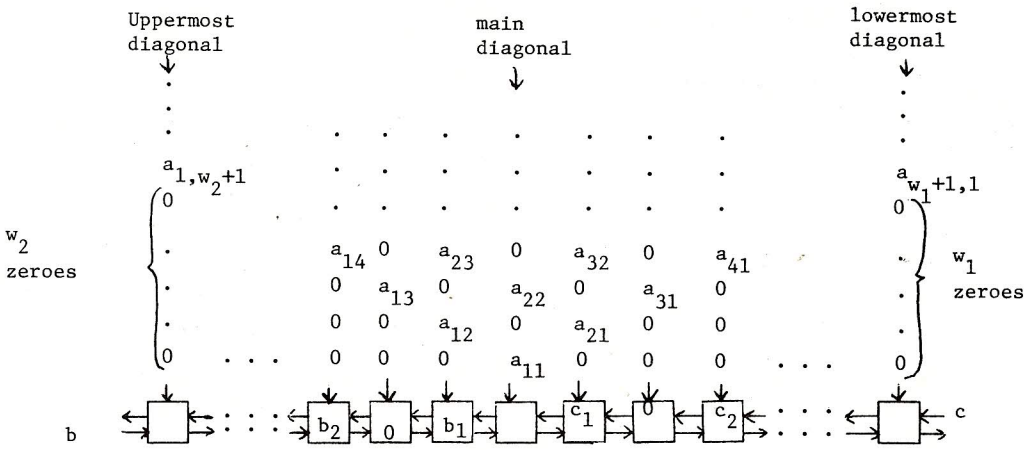

Fig. 5.

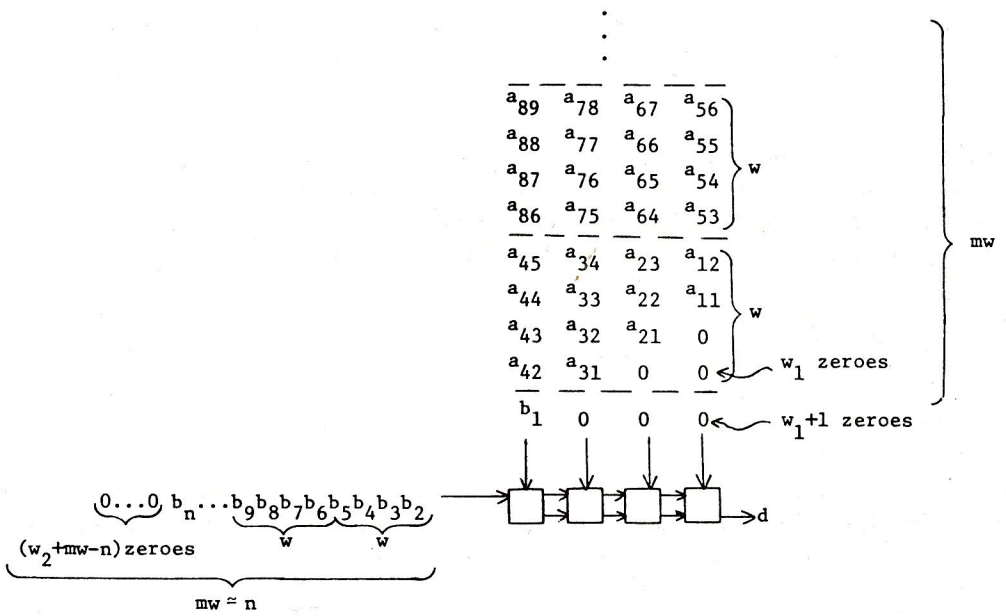

Fig. 6.

```
input row of b's from above;
for q ← 1 to m do
  d ← c;
  c ← 0;
  for k ← 1 to w do
    input row of a's from top;
    input a b from left and shift b's right;
    output and shift d's right;
    c ← c + ab;
  end
end
output the last w c's from top;
```

Fig. 7. Algorithm 2.2.

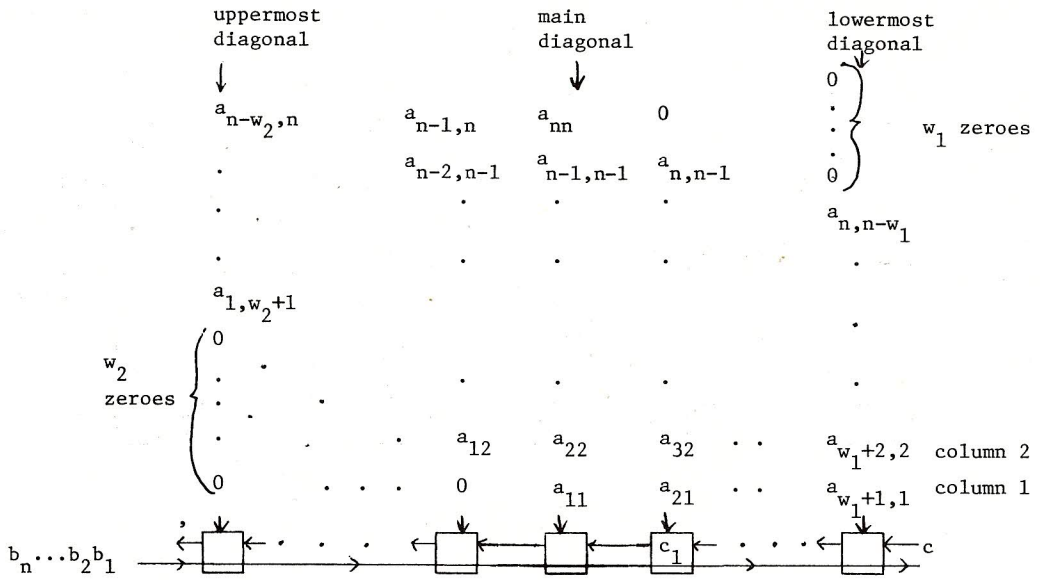

Fig. 8.

It is interesting to note that similar performance can be obtained by a unidirectional chain with O(w) bandwidth. The system computes c in $m = \lceil n/w \rceil$ cycles. In cycle q, PE(i) computes $c_{i+(q-1)w}$, $1 \leqslant q \leqslant m$. Example input for the case $w = 4$, $w_1 = 2$, $w_2 = 1$ is shown in Fig. 6. Algorithm 2.2 (see Fig. 7) describes how the system works.

Performance. For the unidirectional chain, we have $P = w$, $W = w + 2$, $T_C = mw \sim n$, $T_D = mw + 1 \sim n + 1$, $R_P \sim 1$, $R_W \sim 1$, and $R \sim 1$.

2.5. Broadcast chain with O(w) bandwidth

Huang and Abraham [4] have proposed the broadcast chain architecture of Fig. 8. The number of PEs used is w, the a's are entered by columns and the b's broadcast in the order b_1, b_2, ... Each processor computes $c \leftarrow c + ab$ and then sends the c value to the left. Clearly,

Table 2

Performance	Architecture				
	Chain with O(n) bandwidth	Chain with O(1) bandwidth	Bidirectional chain with O(w) bandwidth [12]	Chain with O(w) bandwidth	Broadcast chain with O(w) bandwidth [4]
P	n	n	$w/2$	w	w
W	$n+1$	1	$w/2+1$	$w+2$	$w+1$
T_C	w	w	$2n+w$	n	n
T_D	$w+2$	$(w+2)n$	$2n+w$	$n+1$	$n+w$
R_P	1	1	1	1	1
R_W	1	1	1	1	1
R	1	1	1	1	1

$O \sim wn$, $D \sim (w+2)n$.

$P = w$, $W = w + 1$, $T_C = n$, $T_D \sim n + w$, $R_P \sim 1$, $R_W \sim 1 + w/n \sim 1$, $R \sim 1 + w/n \sim 1$. In light of our unidirectional chain design of Section 2.4, there is little advantage to having a broadcast capability for chains for the band matrix × vector problem.

2.6. Summary

The performance of the VLSI architectures for the band matrix × vector product problem are summarized in Table 2. This appears to be the first time that O(n) and O(1) bandwidth chain designs have been developed for this problem. Earlier work has been restricted to O(w) bandwidth chains. In one case, a bidirectional chain was used and, in the other, a broadcast capability. We have shown that neither of these capabilities is needed. Equally good performance is obtainable by a unidirectional chain with O(w) bandwidth.

3. Band matrix multiplication

3.1. The problem

Input: Two $n \times n$ band matrices A and B. The bandwidth of A is w_A and that of B is w_B.

Output: An $n \times n$ band matrix C that is the product of A and B. The bandwidth of C is at most $w_A + w_B - 1$.

Parameters: $m = \min\{w_A, w_B\}$, $M = \max\{w_A, w_B\}$, $S = \min\{n, w_A + w_B - 1\}$, $w = w_A + w_B$, $O \sim w_A w_B n$, $D \sim 2(w_A + w_B)n$.

3.2. Chain with O(w) bandwidth

The product matrix can be computed one column at a time using the architecture of Fig. 3. Only S processors are needed as there are at most this many non-zero terms in a column. Computing each column of C requires us to compute the product of an $S \times w_B$ band matrix with bandwidth w_A and a $w_B \times 1$ vector. This is done n times to get all the columns of C.

Performance. We see that $P = S$, $W = w$, $T_C = w_A n$, $T_D = (w_A + 2)n$, $R_P \sim 1 + w_A/w_B$, $R_W \sim 1 + w_A/2$, $R \sim (1 + w_A/w_B)(1 + w_A/2)$. If $w_A > w_B$, the computation may be rearranged so that $R_P \sim 1 + w_B/w_A$, $R_W \sim 1 + w_B/2$, $R \sim (1 + w_B/w_A)(1 + w_B/2)$. So, $T_C = mn$, $T_D = (m + 2)n$, $R_P \sim 1 + m/M \sim 2$, $R_W \sim 1 + m/2$, $R \sim (1 + m/M)(1 + m/2) \sim 2 + m$.

When each PE has $2w_A + 2w_B - 1$ words of memory, system performance may be improved by using the PEs in the following manner. A chain with n PEs is employed. When $w_A \leqslant w_B$,

---

for $q \leftarrow 1$ **to** $w_A + w_B - 1$ **do**
  PE(i) computes the element (if any) in column i of the qth diagonal of the band of C (Diagonals are numbered from the top);
  if $q < w_A + w_B - 1$ **then**
    [each PE except the rightmost shifts its row of A to the right and the leftmost PE inputs row $n + q - \min\{u_1 + u_2, n - 1\}$ of the band of A (if this is $> n$, then zeros are input)];
end

Fig. 9. Algorithm 3.1.

PE(i) is used to compute column i of C. When $w_A > w_B$, PE(i) computes row i of C. We describe the process only for the case $w_A \leqslant w_B$.

The band of B is input in column major order and that of A by rows. Following this, PE(i) contains column i of the band of B and row $i - \min\{u_1 + u_2,\ n-1\}$ of the band of A. If

$$\begin{bmatrix} a_{11} & a_{12} & & & \\ a_{21} & a_{22} & a_{23} & & \\ a_{31} & a_{32} & a_{33} & a_{34} & \\ & a_{42} & a_{43} & a_{44} & a_{45} \\ & & a_{53} & a_{54} & a_{55} \end{bmatrix} \quad \begin{bmatrix} b_{11} & b_{12} & & & \\ b_{21} & b_{22} & b_{23} & & \\ & b_{32} & b_{33} & b_{34} & \\ & & b_{43} & b_{44} & b_{45} \\ & & & b_{54} & b_{55} \end{bmatrix} = \begin{matrix} 0\ 0\ 0 \\ 0\ 0 \\ 0 \end{matrix} \begin{bmatrix} c_{11} & c_{12} & c_{13} & & \\ c_{21} & c_{22} & c_{23} & c_{24} & \\ c_{31} & c_{32} & c_{33} & c_{34} & c_{35} \\ c_{41} & c_{42} & c_{43} & c_{44} & c_{45} \\ & c_{52} & c_{53} & c_{54} & c_{55} \end{bmatrix} \begin{matrix} \\ \\ \\ 0 \\ 0\ 0 \end{matrix}$$

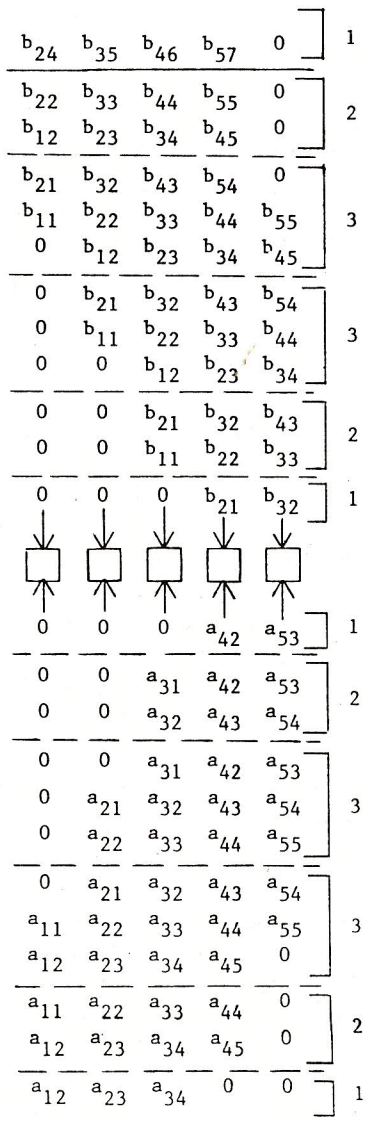

Fig. 10.

$i \leqslant \min\{u_1 + u_2, n-1\}$, then the A elements are zero. u_1 and u_2 are, respectively, the upper bandwidths of A and B. Each PE is now ready to compute the uppermost diagonal of C. The computation of the band of C proceeds as in Algorithm 3.1 (see Fig. 9).

Performance. For this design, we have $P = n$, $W = 1$, $T_C = w_A w_B$, $T_D \sim (2n + m)w$, $R_P \sim 1$, $R_W \sim 1 + w_A/(2n) \sim 1$ and $R \sim 1$.

3.3. Chain with O(n) bandwidth

The n processors are used to compute the $w_A + w_B - 1$ diagonals of C in $w_A + w_B - 1$ steps. In the ith phase, the jth processor from the left computes the jth element in the ith diagonal from the bottom. The position of a diagonal element is determined by extending the band of C in the natural way so that each diagonal has exactly n elements in it (see Fig. 10). The ith element in a diagonal is on row i. The appropriate A and B data needed for the computation of the diagonal elements are input into the system for each C element to be computed. As the processors do not share data, no interconnection between the PEs is needed. So, they might as well be independent PEs. Let l_1 and u_1 be, respectively, the number of diagonals below and above the main diagonal of A. Let l_2 and u_2 be the corresponding figures for B. An example input data for the case $n = 5$, $l_1 = 2$, $u_1 = 1$, $l_2 = 1$, $u_2 = 1$ is shown in Fig. 10.

Performance. As can be seen, $P = n$, $W = 2n$, $T_C = w_A w_B$, $T_D = 2 + \sum_{i=1}^{m} i + [w_A + w_B - 2m - 1]m + w_A + w_B - 1 = w_A w_B + w_A + w_B - 1$, $R_P \sim 1$, $R_W \sim 1 + (w_A w_B)/w$ and $R = 1 + (w_A w_B)/w$.

3.4. Chain with O(1) bandwidth

Two algorithms can be devised from the multiple input chain. The first algorithm works like the chain with $O(w)$ bandwidth except that all input enters from the leftmost PE and the results are extracted from the right. For this algorithm, $P = S$, $W = 2$, $T_C = mn$, $T_D = ((m + 1)n + 1)S$, $R_P \sim 1 + m/M \sim 2$, $R_W \sim 1 + m$ and $R \sim 2(1 + m)$.

The second algorithm works essentially as the n independent PE computation described in Section 3.3. However, this time the PE to PE interconnection is used. The performance figures for this design are $P = n$, $W = 2$, $T_C = w_A w_B$, $T_D = n(w_A w_B + w - 1)$, $R_P \sim 1$, $R_W \sim 1 + (w_A w_B)/w$ and $R \sim 1 + (w_A w_B)/w$.

3.5. Mesh with O(n) bandwidth

The result matrix consists of $w_A + w_B - 1$ diagonals. The mesh architecture contains one PE for each element on these diagonals. Hence, the configuration depends on the precise $w_A + w_B - 1$ diagonals of the product matrix C that contain non-zero elements. Figure 11 shows the architecture for the same case as Fig. 10, i.e. when the bandwidth of C is 6 and there are two diagonals above and three below the main diagonal. The PE at position (i, j) computes c_{ij}.

The PEs are initially loaded so that they contain multipliable pairs. I.e., PE(i, j) contains a_{ik} and b_{kj} for some k, or one or both of the A and B registers are zero. In addition, the b s in each column and the a's in each row repeat with a period M. The configuration for an example A and B is shown in Fig. 11. S data moves are required to obtain this configuration. Now a multiplication takes place. This is followed by $M - 1$ cycles in which the a's move right, the b's down and a multiply-add is performed. Finally, the results are extracted in S units of time.

More specifically, let $A(i, j)$ and $B(i, j)$ be respectively the A and B values in the PE at position (i, j) (we extend the PE configuration by $w_A - 1$ PEs to the left of each row and

$$\begin{bmatrix} a_{11} & a_{12} & & & \\ a_{21} & a_{22} & a_{23} & & \\ a_{31} & a_{32} & a_{33} & a_{34} & \\ & a_{42} & a_{43} & a_{44} & a_{45} \\ & & a_{53} & a_{54} & a_{55} \end{bmatrix} \quad \begin{bmatrix} b_{11} & b_{12} & & & \\ b_{21} & b_{22} & b_{23} & & \\ & b_{32} & b_{33} & b_{34} & \\ & & b_{43} & b_{44} & b_{45} \\ & & & b_{54} & b_{55} \end{bmatrix}$$

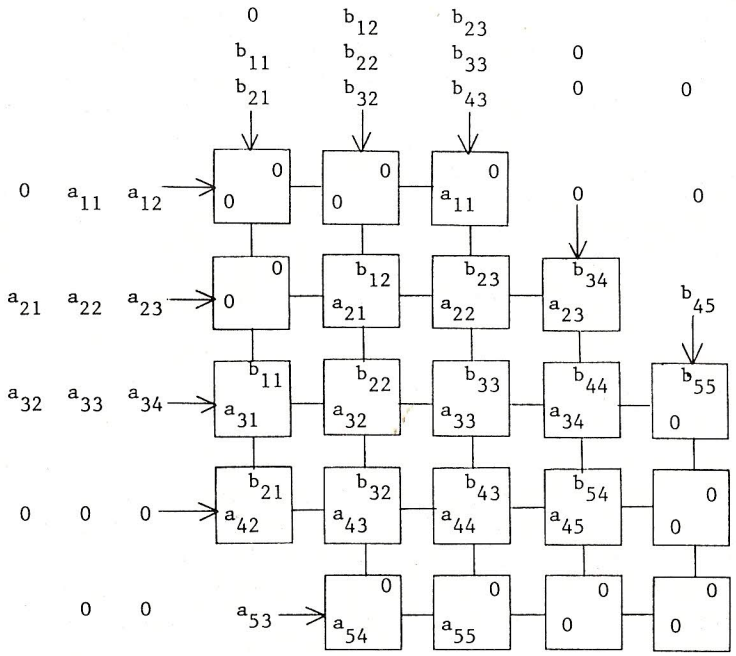

Fig. 11.

above each column from those pictured in Fig. 11). Let l_1 and u_1 be, respectively, the number of diagonals below and above the main diagonal of A. Let l_2 and u_2 be the corresponding figures for B. We consider the case $w_A \geq w_B$ only. The case $w_A < w_B$ is similar. First define the A and B values to be: (for $k = 0, 1, 2, \ldots$)

$$A(i, j) = \begin{cases} a_{i,\, i+j+w_A k-n}, & j \in [n - w_A k - l_1,\ n - w_A k], \\ a_{i,\, i+j+w_A(k+1)-n}, & j \in [n - w_A(k+1) + 1,\ n - w_A k - l_1), \end{cases}$$

$$B(i, j) = \begin{cases} b_{i+j+w_A k-n,\, j}, & i \in [n - w_A k - u_2,\ n - w_A k], \\ 0, & i \in (n - w_A(k+1) + l_2,\ n - w_A k - u_2), \\ b_{i+j+w_A(k+1)-n,\, j}, & i \in (n - w_A(k+1),\ n - w_A(k+1) + l_2]. \end{cases}$$

This gives us a preliminary data pattern with period w_A. If an index above exceeds n or is less than 1, the corresponding value is set to 0.

Now, some of the A's and B's defined above need to be changed to zero. In the computation of c_{ij}, only the terms in the sum $\sum_{q=\max\{i-l_1, j-u_2\}}^{\min\{i+u_1, j+l_2\}} a_{iq}\, b_{qj}$ are needed. Suppose that a_{is} and b_{rj} are the A and B values assigned to the PE at position (i, j). For a_{is} to be useful, s must be in the range

$$\left[\min_{0\leqslant t<w_A}\{\max\{i-l_1, j-u_2+t\}\}, \max_{0\leqslant t<w_A}\{\min\{i+u_1, j+l_2+t\}\}\right]$$
$$= [\max\{i-l_1, j-u_2\}, \min\{i+u_1, j+l_2+w_A-1\}].$$

If s is not in this range, then $A(i, j)$ is set to zero. Similarly, for b_{rj} to be useful, r must be in the range:

$$\left[\min_{0\leqslant t<w_A}\{\max\{i-l_1+t, j-u_2\}\}, \max_{0\leqslant t<w_A}\{\min\{i+u_1+t, j+l_2\}\}\right]$$
$$= [\max\{i-l_1, j-u_2\}, \min\{i+u_1+w_A-1, j+l_2\}].$$

If r is not in this range, then we may set $B(i, j)$ to zero. We now have the final data pattern to be used.

Correctness. We shall show that the data pattern defined above provides each PE with a multipliable pair initially. From the data pattern definition and the flow of data, it follows that each PE always contains a multipliable pair and so the correct results are computed.

PE(i, j) is present in the systolic system iff

$$-(u_1+u_2) \leqslant i-j \leqslant l_1+l_2, \tag{1}$$

First consider the case $j \in [n-w_Ak-l_1, n-w_Ak]$ for some k. For this case, we get from (1):

$$-(u_1+u_2)+j \leqslant i \leqslant l_1+l_2+j$$

or

$$-(u_1+u_2)+n-w_Ak-l_1 \leqslant i \leqslant n-w_Ak+l_1+l_2$$

or

$$n-w_A(k+1)-u_2+1 \leqslant i \leqslant n-w_Ak+l_1+l_2.$$

This interval for i may be divided into the eight intervals:

1. $i \in [n-w_A(k+1)-u_2+1,\ n-w_A(k+1)+l_2]$,
2. $i \in (n-w_A(k+1)+l_2,\ n-w_Ak-u_2)$,
3. $i \in [n-w_Ak-u_2,\ n-w_Ak]$,
4. $i \in (n-w_Ak,\ n-w_Ak+l_2]$,
5. $i \in (n-w_Ak+l_2,\ n-w_A(k-1)-u_2)$
6. $i \in [n-w_A(k-1)-u_2,\ n-w_A(k-1)]$,
7. $i \in (n-w_A(k-1),\ n-w_A(k-1)+l_2]$,
8. $i \in (n-w_A(k-1)+l_2,\ n-w_Ak+l_1+l_2]$.

For each of these intervals, we show that PE(i, j) contains a multipliable pair.

Interval 1. Let $A(i, j)=a_{is}$ be the value assigned to this PE in the initial pattern. From the definition of the initial pattern and our assumption on j, it follows that $s=i+j+w_Ak-n$. From the definition of interval 1, we get

$$s \in [j-w_A-u_2+1, j-w_A+l_2].$$

Since $w_A \geq w_B = l_2 + u_2 + 1$ (by assumption), $j - w_A + l_2 < j - u_2$ and s is not in the useful range. Hence $A(i, j)$ was modified to 0 in the data pattern.

Intervals 2, 5 and 8. In this case, $B(i, j) = 0$ in the initial pattern.

Intervals 3 and 4. Now, $A(i, j) = a_{i,i+j+w_A k-n}$ and $B(i, j) = b_{i+j+w_A k-n,j}$.

Intervals 6 and 7. In these intervals, $B(i, j) = b_{rj} = b_{i+j+w_A(k-1)-n,j}$ and $A(i, j) = a_{i,i+j+w_A k-n}$ in the initial data pattern. However, r is in the range

$$[i + n - w_A k - l_1 + w_A(k-1) - n,\ i + n - w_A k + w_A(k-1) - n]$$
$$= [i - w_A - l_1,\ i - w_A].$$

So, $B(i, j)$ is set to zero in the latter modification of the initial data pattern.

The second and final case to consider is $j \in [n - w_A(k+1) + 1,\ n - w_A k - l_1)$ for some k. For (i, j) in the band of C, we have

$$-(u_1 + u_2) + j \leqslant i \leqslant l_1 + l_2 + j$$

or

$$-(u_1 + u_2) + n - w_A(k+1) + 1 \leqslant i < l_1 + l_2 + n - w_A k - l_1$$

or

$$n - w_A(k+1) - u_1 - u_2 + 1 \leqslant i < n - w_A k + l_2.$$

This time we consider the seven intervals:

1. $i \in (n - w_A(k+1) - u_1 - u_2,\ n - w_A(k+2) + l_2]$,
2. $i \in (n - w_A(k+2) + l_2,\ n - w_A(k+1) - u_2)$,
3. $i \in [n - w_A(k+1) - u_2,\ n - w_A(k+1)]$,
4. $i \in (n - w_A(k+1),\ n - w_A(k+1) + l_2]$,
5. $i \in (n - w_A(k+1) + l_2,\ n - w_A k - u_2)$,
6. $i \in [n - w_A k - u_2,\ n - w_A k]$,
7. $i \in (n - w_A k,\ n - w_A k + l_2)$.

For each of these intervals, we show that PE(i, j) contains a multipliable pair.

Interval 1. Since $A(i, j) = a_{is} = a_{i,i+j+w_A(k+1)-n}$ in the initial data pattern, s is in the range $(j - u_1 - u_2,\ j - w_A + l_2]$. However, as $w_A \geq w_B$, $j - w_A + l_2 < j - u_2$. So, $A(i, j)$ was modified to 0 in the data pattern.

Intervals 2 and 5. $B(i, j) = 0$ in these intervals.

Intervals 3 and 4. Now, $A(i, j) = a_{i,i+j+w_A(k+1)-n}$ and $B(i, j) = b_{i+j+w_A(k+1)-n,j}$. So the PE has a multipliable pair.

Intervals 6 and 7. In these intervals, $B(i, j) = b_{rj} = b_{i+j+w_A k-n,j}$. Since $j \in [n - w_A(k+1) + 1,\ n - w_A k - l_1)$, r is in the range

$$(i + n - w_A(k+1) + w_A k - n,\ i + n - w_A k - l_1 + w_A k - n) = (i - w_A,\ i - l_1).$$

So, $B(i, j)$ is modified to zero.

Performance. For this scheme, we have $P \sim Sn$, $W = 2n$, $T_C = M$, $T_D = 2S + M - 1$, $R_P \sim M$, $R_W \sim 3$ and $R \sim 3M$.

3.6. Mesh with O(1) bandwidth

This is very similar to the mesh with O(n) bandwidth. An interconnect along the lowermost and uppermost diagonals is added (Fig. 12) to facilitate easy input and output to and from the system.

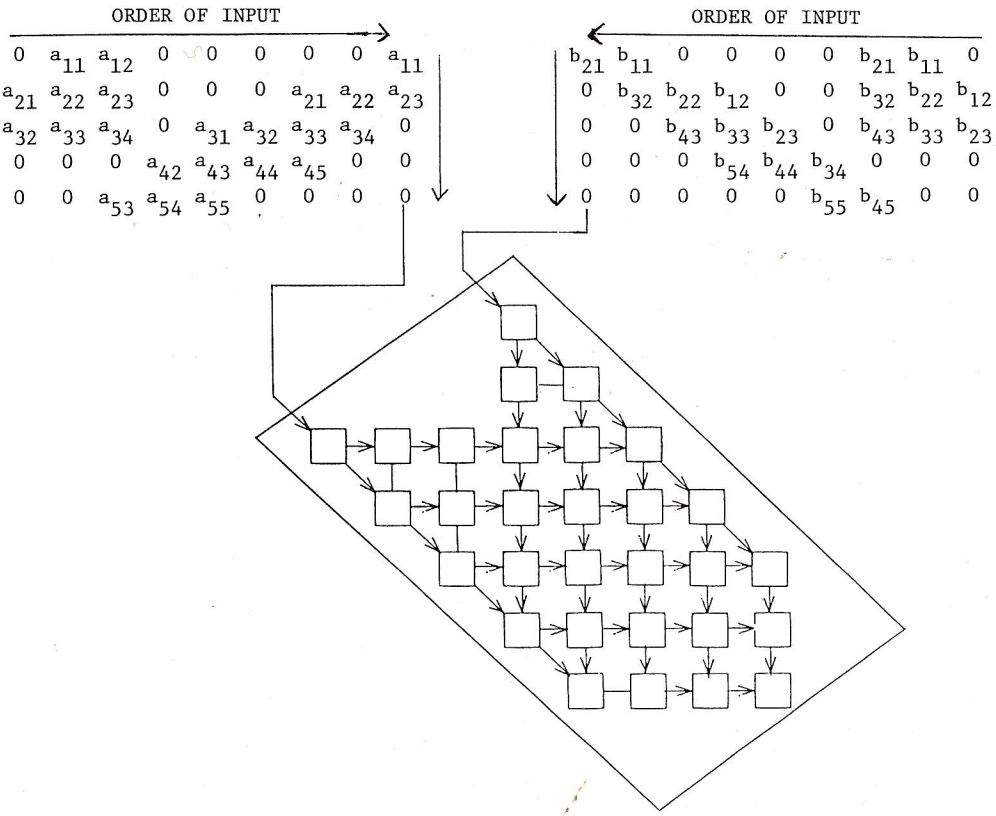

Fig. 12.

Performance. For this architecture, we have $P = mn$, $W = 2$, $T_C = M$, $T_D = 2mn + M - 1$, $R_P \sim M$, $R_W \sim 3$ and $R \sim 3M$.

3.7. Broadcast mesh with O(n) bandwidth

A broadcast mesh architecture suitable for the multiplication of band matrices is shown in Figs. 13–15. Here, we assume that $w_A + w_B - 1 \leqslant n$ and $w_B \leqslant w_A$ (the case when $w_A < w_B$ is symmetric). The broadcast capability is used only for matrix B. The b's are broadcast into the systolic array by diagonals while the a's are input as exactly $\min\{n + u_2,\ w_A + w_B - 1\}$ columns. Row i of the systolic array inputs row i of matrix A (i.e. the part of this row in the band with a suitable shift; the elements enter right to left). Figure 13 shows the the input pattern for the matrices of Fig. 10. Figure 14 shows the input pattern for two 6×6 matrices A and B with $w_A = w_B = 2$. A has one diagonal below the main diagonal while B has one above. Figure 15 shows the input sequence for the case where $w_A = 3$, $w_B = 2$ and neither have any diagonals below the main diagonal. The architecture and input pattern for $n \times n$ matrices with bandwidth w_A and w_B are shown in Fig. 16. Let l_1, u_1, l_2 and u_2 be as in Section 3.5.

The system begins by initializing the A, B and C registers of all PEs to zero. Then, $\min\{n - u_1,\ w_A\} - 1$ columns of A input are accepted, shifting each to the right one unit. This is followed by w_B cycles of input and shift A's right, broadcast B's, multiply and add to C. Finally, S left shifts suffice to extract the results. The correctness of the systolic scheme described above follows from the observation that the elements on the uppermost diagonal of C

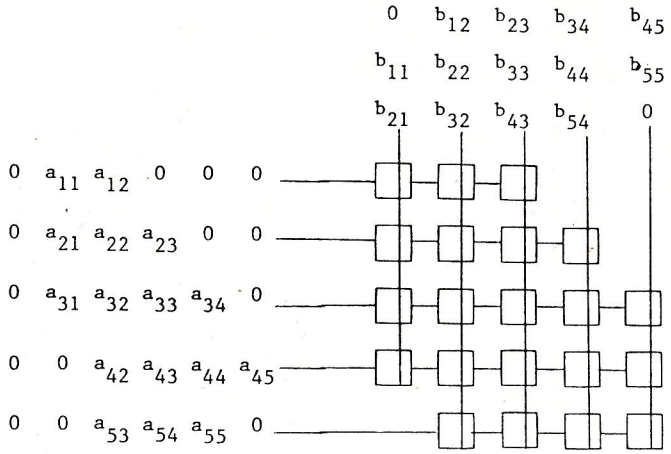

Fig. 13.

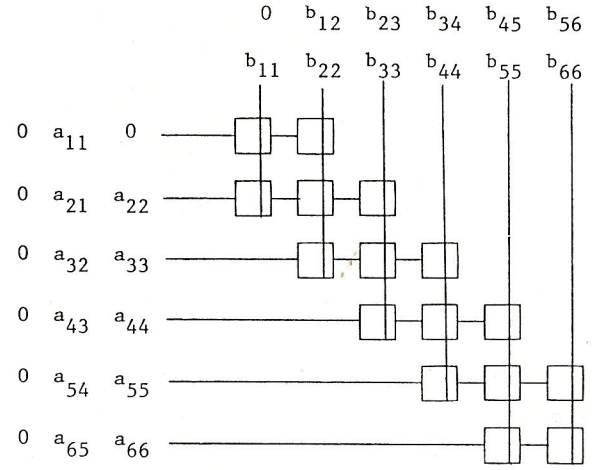

Fig. 14.

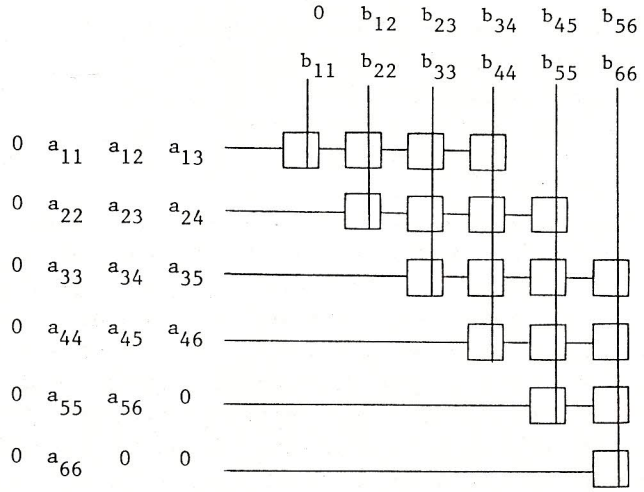

Fig. 15.

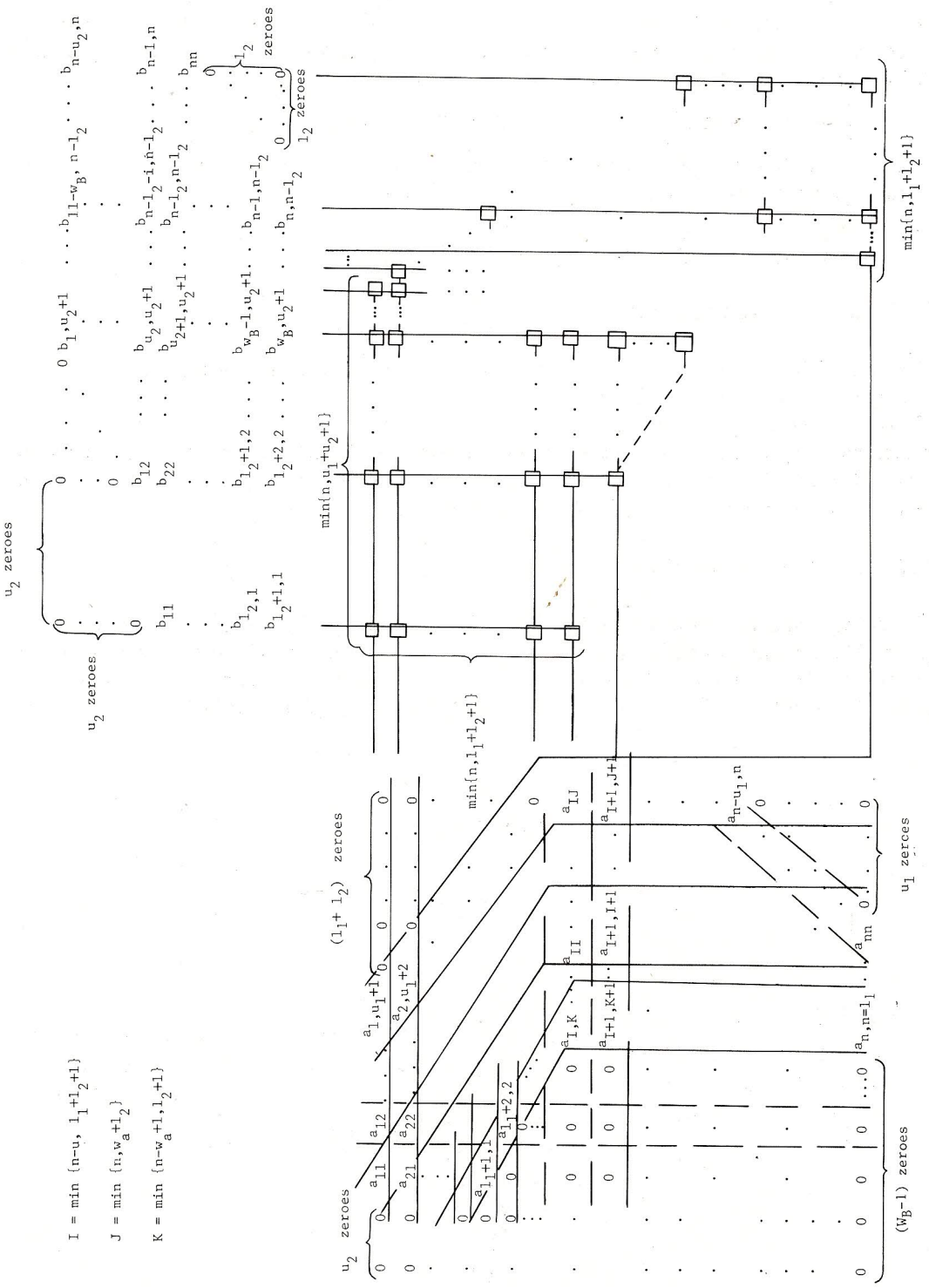
I = min {n-u, l1+l2+1}
J = min {n,wa+l2}
K = min {n-wa+1, l2+1}
u2 zeroes
u2 zeroes
l2 zeroes
l2 zeroes
min{n,u1+u2+1}
min{n,l1+l2+1}
min{n,l1+l2+1}
(l1+ l2) zeroes
u2 zeroes
u1 zeroes
(WB-1) zeroes

Fig. 16.

have just one $a_{ik}b_{kj}$ term, those on the next have 2, and so on. So, it is sufficient to load only the first $\min\{n-u_1,\ w_A\}$ diagonals of C before the first product is performed. The input patterns for A and B ensure that PE$(i,\ j)$ computes only the terms in the product of c_{ij}.

Correctness. To establish the correctness of Fig. 16 ($w_B \leqslant w_A$), let $a(i,\ j,\ t)$, $b(i,\ j,\ t)$ and $c(i,\ j,\ t)$ be, respectively, the a, b and c values in PE$(i,\ j)$ just after the tth multiplication and addition is performed. It is easy to see that

$$a(i,\ j,\ t) = a_{i,j+l_2-t+1}, \qquad b(i,\ j,\ t) = b_{j+l_2-t+1,j}.$$

(The defined a_{ij}'s and b_{ij}'s are extended so that $a_{ij} = b_{ij} = 0$ whenever $j \leqslant 0$ or $j > n$.) Now, the value c_{ij} can be seen to be

$$C(i,\ j,\ w_B) = \sum_{t=1}^{w_B} a(i,\ j,\ t) * b(i,\ j,\ t) = \sum_{t=1}^{w_B} a_{i,j+l_2-t+1} * b_{j+l_2-t+1,j}.$$

This is readily seen as being correct.

Performance. The working of the algorithm can be divided into the following two cases:

(1) $w_A \leqslant n-u_1,\ w_A + w_B - 1 \leqslant n$,

(2) $w_A > n-u_1,\ w_A + w_B - 1 \leqslant n$.

For each case, T_D can be shown to be $\leqslant 2(w_A + w_B - 1)$. Hence, the performance figures are $P \sim Sn$, $W = 2n$, $T_C = m$, $T_D \leqslant 2(w_A + w_B - 1)$, $R_P \sim Smn/(nw_Aw_B)$. If $w_A + w_B - 1 \leqslant n$, $R_P \sim (w_A + w_B - 1)m/(w_Aw_B) \leqslant 2$, $R_W \sim 2$ and $R \leqslant 4$.

3.8. Hexagonally connected processor array

The hexagonal connection was originally employed by Kung [7] for the band matrix product problem. His original design has an R value of 3. This has been improved to 1 by Huang and Abraham [4]. The design of [4] is shown in Fig. 17 with the input pattern for the matrices of Fig. 10. The essential difference between this and the design of [7] is that in Kung's design, the

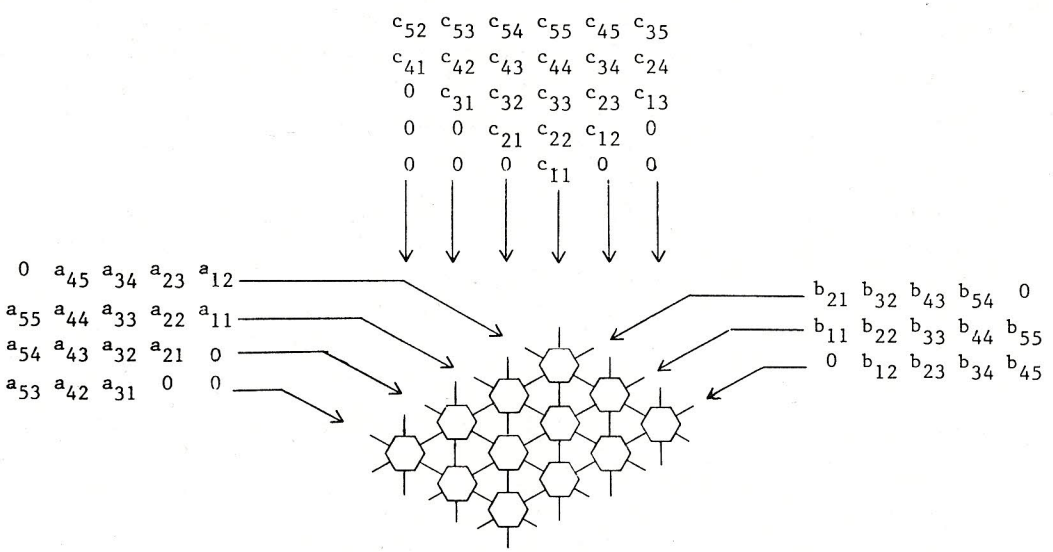

Fig. 17.

Table 3

Performance	Architecture							
	Chain with O(w) bandwidth	Chain with O(w) memory	Chain with O(n) bandwidth	Chain with O(1) bandwidth	Mesh with O(n) bandwidth	Mesh with O(1) bandwidth	Broadcast mesh with O(n) bandwidth	Hexagonal with O(w) bandwidth [4]
P	S	n	n	n	Sn	Sn	Sn	$w_A w_B$
W	w	1	$2n$	2	$2n$	2	$2n$	$2w$
T_C	mn	$w_A w_B$	$w_A w_B$	$w_A w_B$	M	M	m	$n+m-1$
T_D	$(m+2)n$	$(2n+m)w$	$w_A w_B + w$	$n(w_A w_B + w)$	$2S+M$	$2nS+M$	$2w$	$n+m$
R_P	2	1	1	1	M	M	2	1
R_W	$1+m/2$	1	$1+(w_A w_B)/w$	$1+(w_A w_B)/w$	3	3	2	1
R	$2+m$	1	$1+(w_A w_B)/w$	$1+(w_A w_B)/w$	$3M$	$3M$	4	1

$m = \min\{w_A, w_B\}$, $M = \max\{w_A, w_B\}$, $S = \min\{n, w_A + w_B - 1\}$, $w = w_A + w_B$, $O \sim w_A w_B n$, $D \sim 2(w_A + w_B)n$.

C's flow from bottom to top rather than from top to bottom. This difference in the direction of flow requires a separation in time between successive elements of A, B and C.

Performance. From Fig. 17, we see that $P = w_A w_B$, $W = 2w$, $T_C = n + m - 1$, $T_D = n + m$, $R_W = R_P \sim 1$ and $R \sim 1$ (when $n \gg m$).

3.9. Summary

The performance of the VLSI architectures for the band matrix multiplication problem is summarized in Table 3. Despite the complexity of several of our designs, we have been unable to achieve R values close to 1 except for some instances. It remains to be seen whether an R of 1 can be achieved.

4. Conclusions

This paper has examined VLSI systems for the band matrix product problem. Most of the commonly proposed VLSI architectures have been considered. In many cases, we have been able to improve upon earlier work. We have shown that the correctness of even complex VLSI designs for the band matrix product problem can be established using traditional mathematical tools. While in some instances, the proof obtained is itself quite intricate, it appears unlikely that the design or proof could have been obtained any easier (if at all) using the s-transform method.

Unlike earlier designs, many of our designs use a single PE to compute an individual element of the product vector c or product matrix C. Consequently, error diagnosis is easier than for designs in which each c value results from computations in several PEs.

References

[1] E. Dekel, D. Nassimi and S. Sahni, Parallel matrix and graph algorithms, *SICOMP* **10** (4) (1981) 657–675.
[2] L. Hageman and D. Young, *Applied Iterative Methods* (Academic Press, New York, 1981).
[3] E. Horowitz, VLSI architectures for matrix computations, *Proc. IEEE International Conference on Parallel Processing* (1979) 124–127.
[4] K.H. Huang and J.A. Abraham, Efficient parallel algorithms for processor arrays, *Proc. IEEE International Conference on Parallel Processing* (1982) 271–279.

[5] L. Johnsson and D. Cohen, A mathematical approach to modeling the flow of data and control in computational networks, in: H.T. Kung et al., eds., *VLSI Systems and Computations* (Computer Science Press, Rockville, MD, 1981) 213–225.
[6] L. Johnsson, D. Cohen, U. Weiser and A. Davis, Towards a formal treatment of VLSI arrays, *Proc. Caltech Conference on VLSI* (1981) 375–398.
[7] H.T. Kung, Let's design algorithms for VLSI systems, *Proc. Caltech Conference on VLSI* (1979) 65-90.
[8] H.T. Kung, A listing of systolic papers, Department of Computer Science, Carnegie-Mellon University, 1984.
[9] H.T. Kung and C.E. Leiserson, Systolic arrays for VLSI, Department of Computer Science, Carnegie-Mellon University, 1978.
[10] H.T. Kung and W.T. Lin, An algebra for VLSI algorithm design, Carnegie-Mellon University, Technical Report, 1983.
[11] H.T. Kung, A. Nowatzyk and M. Ravisharkar, The universal host: Architectures and system configuration, Carnegie-Mellon University, Technical Report, 1983.
[12] C.E. Leiserson, *Area-Efficient VLSI Computation* (MIT Press, Cambridge, MA, 1983).
[13] R. Melhem and W. Rheinboldt, A mathematical model for the verification of systolic networks, *SIAM J. Comput* **13** (3) (1984) 541-565.
[14] R.W. Priester, H.J. Whitehouse, K. Bromley and J.B. Clary, Signal processing with systolic arrays, *Proc. IEEE International Conference on Parallel Processing* (1984) 207–215.
[15] U. Weiser and A. Davis, A wavefront notation tool for VLSI array design, in: H.T. Kung et al., eds., *VLSI Systems and Computations* (Computer Science Press, Rockville, MD, 1981) 226–234.