Improved Matrix Product Computation using Double-Pipeline Systolic Arrays

A volume-efficient retimed hexagonal array for computing matrix product is described. The new array requires the same time as the well-known hex array of Kung and Leiserson but uses only half the hardware. The design is separated out on to two planar layers which communicate with each other only at the array boundaries, and requires the same number of input–output connections as the traditional array. Individual cells compute with an efficiency of $e = 2/3$.

Received October 1987

1. Introduction

In this short paper we examine the extension of double-pipe splittings for the derivation of a fast volume-efficient systolic matrix product array. The concept of double pipes for improving linear systolic arrays was introduced by the authors.[1] By extensions in Ref. 2, the general D^z pipe was shown to be a variant of the well-known parallel processor tree forms for matrix vector and matrix product operations which yield $O(\log_2 n)$ and $O(n \log_2 n)$ time respectively. We conclude that the linear systolic arrays of Kung and Leiserson,[3] which have attracted a great deal of attention with the rapid advances in VLSI technology, were simply special cases of this processor tree which traded off processors against speed and efficiency. The subject of this paper is to demonstrate that the double-pipe splitting of Ref. 1 extends in a straightforward manner to the matrix product computations and the hexagonal array of Ref. 3. In addition, and by way of illustration, we also present a three-dimensional design for the matrix product calculation. Representation of systolic arrays in space rather than the plane has received little attention to date, although Rosenburg[4] presents a case study of 3-D design which indicates that the extra dimension for wire routing should yield significant savings in space. Caulfield *et al.*[5] have also presented a method for implementing systolic array processing using optical computing and free-space transmission of data. In contrast to these designs our array is limited to just two planar layers (or laminars), which only require communication between laminars at the edge of the design. In addition we can improve cell efficiency and reduce the total cell count while leaving the array bandwidth unchanged.

2. Additive splittings

In the case of a general additive splitting the matrix product can be written in the form

$$C = AB = \sum_{i=1}^{m} A_i B_i \quad j = 1(1)m \qquad (1.1)$$

for two $n*n$ band matrices A and B of bandwidths w_1 and w_2 respectively. The double-pipe splitting has $m = 2$ and the form,

$$AB = (A_1 + A_2)(B_1 + B_2), \qquad (1.2)$$

where A_i and B_j for $n = 6$ have the forms,

$$A_1 = \begin{bmatrix} a_{11} & 0 & a_{13} & 0 & a_{15} & 0 \\ 0 & a_{22} & 0 & a_{24} & 0 & a_{26} \\ a_{31} & 0 & a_{33} & 0 & a_{35} & 0 \\ 0 & a_{42} & 0 & a_{44} & 0 & a_{46} \\ a_{51} & 0 & a_{53} & 0 & a_{55} & 0 \\ 0 & a_{62} & 0 & a_{64} & 0 & a_{66} \end{bmatrix} \quad A_2 = \begin{bmatrix} 0 & a_{12} & 0 & a_{14} & 0 & a_{16} \\ a_{21} & 0 & a_{23} & 0 & a_{25} & 0 \\ 0 & a_{32} & 0 & a_{34} & 0 & a_{36} \\ a_{41} & 0 & a_{43} & 0 & a_{45} & 0 \\ 0 & a_{52} & 0 & a_{54} & 0 & a_{56} \\ a_{61} & 0 & a_{63} & 0 & a_{65} & 0 \end{bmatrix}$$

and

$$B_1 = \begin{bmatrix} b_{11} & 0 & b_{13} & 0 & b_{15} & 0 \\ 0 & b_{22} & 0 & b_{24} & 0 & b_{26} \\ b_{31} & 0 & b_{33} & 0 & b_{35} & 0 \\ 0 & b_{42} & 0 & b_{44} & 0 & b_{46} \\ b_{51} & 0 & b_{53} & 0 & b_{55} & 0 \\ 0 & b_{62} & 0 & b_{64} & 0 & b_{66} \end{bmatrix} \quad B_2 = \begin{bmatrix} 0 & b_{12} & 0 & b_{14} & 0 & b_{16} \\ b_{21} & 0 & b_{23} & 0 & b_{25} & 0 \\ 0 & b_{32} & 0 & b_{34} & 0 & b_{36} \\ b_{41} & 0 & b_{43} & 0 & b_{45} & 0 \\ 0 & b_{52} & 0 & b_{54} & 0 & b_{56} \\ b_{66} & 0 & b_{63} & 0 & b_{65} & 0 \end{bmatrix}$$

and

$$C = C_1 + C_2 + C_3 + C_4$$

where

$$(a)\ C_1 = A_1 B_1, \quad (b)\ C_2 = A_1 B_2, \quad (c)\ C_3 = A_2 B_1, \quad (d)\ C_4 = A_2 B_2. \qquad (1.3)$$

Now consider that the result of the product (1.3*a*) has the form

$$C_1 = \begin{bmatrix} c_{11} & 0 & c_{13} & 0 & c_{15} & 0 \\ 0 & c_{22} & 0 & c_{24} & 0 & c_{26} \\ c_{31} & 0 & c_{33} & 0 & c_{35} & 0 \\ 0 & c_{42} & 0 & c_{44} & 0 & c_{46} \\ c_{51} & 0 & c_{53} & 0 & c_{55} & 0 \\ 0 & c_{62} & 0 & c_{64} & 0 & c_{66} \end{bmatrix}$$

and taking advantage of the zero patterns of A_1, B_1 and C_1 produces a special hex array with re-timed dataflow,[6] and re-placed cells[7] as shown in Fig. 1.

This array requires at most $T = 3n/2 + \min(\lceil w_1/2 \rceil, \lceil w_2/2 \rceil)$ ips cycles, and follows from the facts that only a single neutral element is associated with every two genuine input data elements, and the effective bandwidths of A_1 and B_1 are bounded by $\lceil w_1/2 \rceil$ and $\lceil w_2/2 \rceil$ respectively. As a result the hex array for producing C_1 requires approximately $w_1 w_2/4$ cells. This is one-quarter that of a normal hex for matrix product and improves efficiency from $e = 0.5$ to $e = 0.66$ and computes twice as quickly. These advantages are retained for the remaining partial products (1.3*b–d*) by converting them all to a normal form with the same matrix structure as (1.3*a*) by simple computation preserving row and column interchanges.

For (1.3*b*) C_2 and B_2, interchanges are performed by swapping columns i and $i+1$ for $i = 1(2)n-1$ producing matrices $\bar{C}_2$ and $\bar{B}_2$. In (1.3*c*) C_3 and A_2 produce $\bar{C}_3$ and $\bar{A}_2$ by interchanging row i and $i+1$ for $i = 1(2)n-1$, while for (1.3*d*), A_2, B_2 and C_4 are permuted to produce $\hat{A}_2$, $\hat{B}_2$ and $\hat{C}_4$ by a two-step process, first interchanging rows i and $i+1$ of A_2 and C_4, and secondly swapping columns i and $i+1$ of B_2 and C_4 for $i = 1(2)n-1$. The resulting normal form of (1.3) is

$$(a)\ C_1 = A_1 B_1, \quad (b)\ \bar{C}_2 = A_1 \bar{B}_2, \quad (c)\ \bar{C}_3 = \bar{A}_2 B_1, \quad (d)\ \hat{C}_4 = \hat{A}_2 \hat{B}_2, \qquad (1.4)$$

and can be solved on the same array as (1.3*a*) using four passes, and requires a time $T = 4(3n/2) + 4 \min(\lceil w_1/2 \rceil, \lceil w_2/2 \rceil) = 2(3n + \min(w_1, w_2))$; that is, twice the time of the original hex scheme but a quarter of the cells. It also has the interesting property of preserving the bound AT^2,[8] as $(A/4)(2T)^2 = AT^2$, where A and T are the area and time requirements of the original hex.

3. Double-pipe hexagonal schemes

A D^1-pipe hex design using two layers is now easily constructed by placing these smaller hexes on separate layers and noticing that C_1 and C_4 have the same structure, as do C_2 and C_3. Hence two passes form the full matrix product by computing C_1 and C_4 on separate layers on the first pass and C_2, C_3 on different layers on the second pass.

Like the double pipe for matrix vector in Ref. 1, the final results $C_1 + C_4$ and $C_3 + C_2$ can be overlapped with hex operations by introducing an upper boundary of $\lceil w_1/2 \rceil + \lceil w_2/2 \rceil - 1$ adders to each hexagonal array. Because the arrays actually compute using (1.4) the order of the matrix elements must be restored before the addition takes place. Fortunately the localised permutations used to derive (1.4) keep the recovery simple, as $\bar{C}_2$, $\bar{C}_3$ and $\hat{C}_4$ have the forms

$$\bar{C}_2 = \begin{bmatrix} c_{12} & 0 & c_{14} & 0 & c_{16} & 0 \\ 0 & c_{21} & 0 & c_{23} & 0 & c_{25} \\ c_{32} & 0 & c_{34} & 0 & c_{36} & 0 \\ 0 & c_{41} & 0 & c_{43} & 0 & c_{45} \\ c_{52} & 0 & c_{54} & 0 & c_{56} & 0 \\ 0 & c_{61} & 0 & c_{63} & 0 & c_{65} \end{bmatrix} \quad \bar{C}_3 = \begin{bmatrix} c_{21} & 0 & c_{23} & 0 & c_{25} & 0 \\ 0 & c_{12} & 0 & c_{14} & 0 & c_{16} \\ c_{41} & 0 & c_{43} & 0 & c_{45} & 0 \\ 0 & c_{32} & 0 & c_{34} & 0 & c_{36} \\ c_{61} & 0 & c_{63} & 0 & c_{65} & 0 \\ 0 & c_{52} & 0 & c_{54} & 0 & c_{56} \end{bmatrix}$$

and

$$\hat{C}_4 = \begin{bmatrix} c_{22} & 0 & c_{24} & 0 & c_{26} & 0 \\ 0 & c_{11} & 0 & c_{13} & 0 & c_{15} \\ c_{42} & 0 & c_{44} & 0 & c_{46} & 0 \\ 0 & c_{31} & 0 & c_{33} & 0 & c_{35} \\ c_{62} & 0 & c_{64} & 0 & c_{66} & 0 \\ 0 & c_{51} & 0 & c_{53} & 0 & c_{55} \end{bmatrix}$$

Example: $AB = C$ with A and B matrices from splitting of form (1.2)

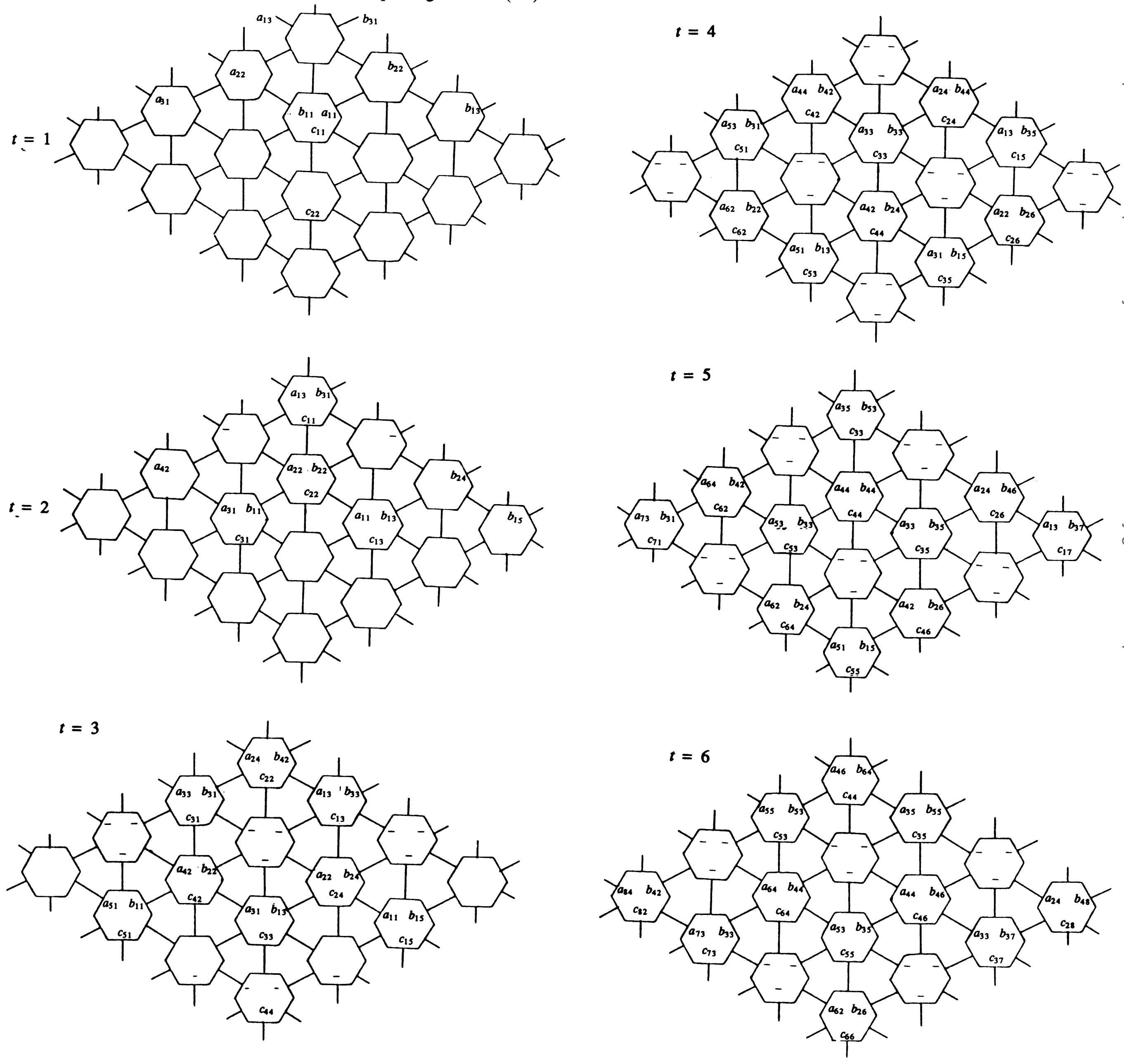
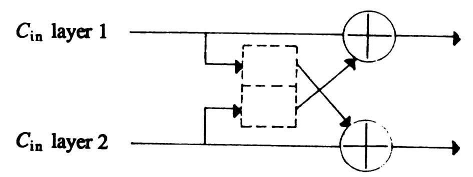
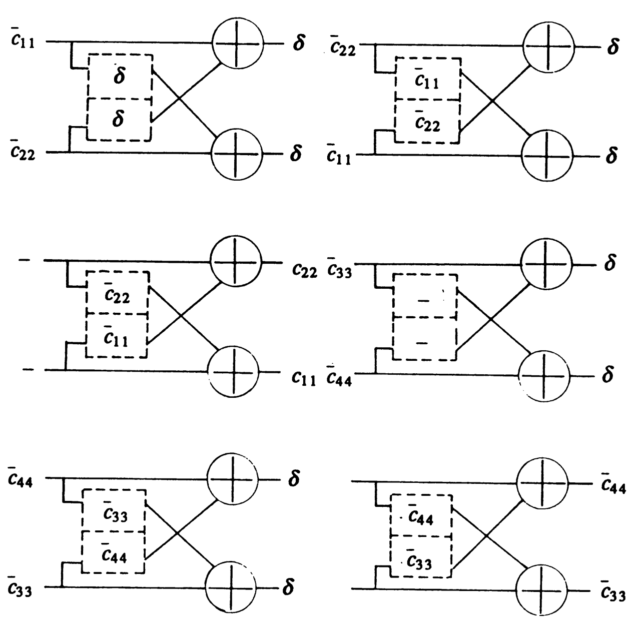


Figure 1. Normal form of single layer of double-pipe hex.



(*a*) Adder delay arrangement.



(*b*) Production of full result for main diagonal column.

Figure 2. Snapshots of double-pipe adder/delay arrangement.

Simply swapping the diagonal elements of each diagonal $2*2$ block in $\bar{C}_4$ and $\hat{C}_4$ and adding C_1 and $\bar{C}_2$ (or vice versa) produces the correct result, and can be achieved using only a single delay with each adder (see Fig. 2).

4. Conclusion

Our main result can be stated simply as the following theorem.

Theorem

The matrix product of two $n*n$ matrices A and B of bandwidths w_1 and w_2 can be computed on a D^1 pipe hex in at most $T = 3n + \min(w_1, w_2) + 2$ cycles using $2(\lceil w_1/2 \rceil \lceil w_2/2 \rceil)$ inner product cells and $2(\lceil w_1/2 \rceil + \lceil w_2/2 \rceil - 1)$ adders and delay cells.

That is, we can compute the matrix product in the same time complexity as the 2-D (planar) scheme of Kung and Leiserson[3] with only half the hardware and improved cell efficiency using two planes (or layers) connected only at the edges. Furthermore, the input and output bandwidth – although modified – requires the same number of connections. In addition, by using the ordinary data positioning of Kung and Leiserson on this new array, problems of smaller bandwidth can be solved separately in parallel using different layers.

In a 3-D setting connections at the edge of layers are highly desirable from the viewpoint of manufacturing multi-layer designs. We also point out that as a true multiple-layer technology (even one restricted to a couple of layers) is some way off the design can be incorporated into fast circuits at the board level.

Finally we observe that Robert[9] and Robert and Tchuente[10] independently applied the double-pipe technique to improve the solution of linear systems by linear back-substitution arrays. The extension of double pipes to the hexagonal LU-factorisation of Kung and Leiserson is not as straightforward, because the LU array is undecomposable,[11] preventing the use of the technique used here. Robert improved LU-factorisation by using $2*2$ block partitioning to form an implicit double pipe. However, in Ref. 2 it is shown that higher block sizes will not improve on Robert's scheme. This indirectly demonstrates that double pipes will not benefit systolic arrays involving explicit feedback of complete results.

G. M. MEGSON* and D. J. EVANS
Loughborough University of Technology, Loughborough, Leicestershire, LE11 3TU.

* Dr Megson is now Joint Atlas research fellow, Oriel College, Oxford University; correspondence about this paper should be addressed to him at Rutherford Appleton Laboratory, Chilton, Didcot, Oxon.

References

1. G. M. Megson and D. J. Evans, Soft-systolic pipelined matrix algorithms. In *Parallel Computing 85*, edited M. Feilmeier, G. Joubert and U. Schendel. (Report of the international conference on Parallel Computing, 1985.)
2. G. M. Megson, Novel algorithms for the soft-systolic paradigm. *PhD Thesis*, Loughborough University of Technology, U.K. (1987).
3. H. T. Kung and C. E. Leiserson, Systolic arrays for VLSI. In *Sparse matrix Proceedings 1978*, edited I. S. Duff and G. W. Stewart. *SIAM*, pp. 256–282 (1979).
4. A. L. Rosenburg, 3-D VLSI: a case study. J. ACM, vol. 30, pp. 397–416 (1983).
5. H. J. Caulfield, W. T. Rhodes, M. J. Foster and S. Horvitz, Optical implementation of systolic arrays processing. *Optic Communications* **40** (2), 86–90 (1981).
6. C. E. Leiserson and J. B. Saxe, Optimising synchronous systems. *J. VLSI and Computer Systems* **1** (1), 41–67 (1983).
7. G. Rothe, On the connection between hexagonal and unidirectional rectangular systolic arrays. *Aegean Workshop on Computing (AWOC 86) VLSI Algorithms and Architectures, Proceedings of the 2nd International Workshop on Parallel Computing and VLSI, 1986*. Heidelberg: Springer.
8. J. E. Savage, Area-time tradeoffs for matrix multiplication and related problems in VLSI models. *Journal of Computer and System Sciences* **22**, 230–242 (1981).
9. Y. Robert, Block LU decompositions of a band matrix on a systolic array. *International Journal of Computer Mathematics* **17**, 295–315 (1985).
10. Y. Robert and M. Tchuente, Parallel solution of band triangular systems on VLSI arrays with limited fanout. In *International Workshop on Modelling and Performance Evaluation of Parallel Systems*, edited M. Becker. Grenoble, North Holland. pp. 209–229 (1984).
11. R. Schreiber, On the systolic arrays of Brent, Luk, and Van Loan. *Proceedings of SPIE symposium*, vol. 431. real time signal processing VI Society of Photo-optical Instrumentation Engineers (1983).