A very efficient double pipe systolic array for result reusable matrix multiplication, matrix polynomial and powers of a matrix.

G. Isern¹ *
V. Matamoros²

Universidad Central de Venezuela
Escuela de Computación
Apartado 49125, Caracas
Venezuela
E-mail: isern@dino.conicit.ve
E-mail: iftr12%cpvop3@intopl.intevep.pdv.com
Fax: 58 2 6627233
Fax: 58 2 7081833

ABSTRACT

A systolic array for matrix multiplication is presented. It has the property that the results of one multiplication can reenter the array and be used in the next multiplication without memorizing intermediate results and without the addition of delays thanks to the use of multicast lines for reentering the previous result. This systolic design also achieves a high performance by the use of a second level of pipelining by allowing the processing elements themselves to be pipelined. This systolic architecture shows not only high performance but also optimizes input output ports and execution time. We show how this array can be used for the calculation of a matrix polynomial and for matrix powers, problems those, that look alike with the result reusable matrix multiplication.

Keywords: systolic algorithms, matrix multiplication, reusable multiplication, matrix polynomial, matrix powers.

¹ Profesor Investigador Asistente, MSc UCV, Coordinador Investigación y Extensión (Informática) UCV.

² Ingeniero de Telecomunicaciones y Redes, CORPOVEN S.A., Filial de Petróleos de Venezuela.
1. Introduction.

Matrix multiplication is one of the earliest algorithms for which a systolic implementation was found (Kung and Leiserson, 1978 [1]). It received considerable attention because of its practical importance in signal processing as well as numerical analysis. Optimizing computation time and I/O bandwidth have been two challenging issues for any architect working in the domain of systolic arrays, especially for the matrix multiplication problem.

Concerning these points of view, several schemes to solve the problem have been proposed by Hwang and Cheng [2], Isern [3,4,5] and Montagne, Surós, Isern [6]. To our knowledge, only two systolic arrays for matrix multiplication allow the results to be used immediately for the next step. Such designs will be said to be result reusable and were described by Culik [7] in 1982 and by Quinton [8] in 1986. Quinton's systolic array allows the results to be used immediately but with only one pipelining level and the need of slowing down by a factor of two the rate of input (delay).

This paper describes a simple and the fastest result reusable systolic array for matrix multiplication, based on doublepipe and multicast scheme. The original algorithm without double pipe [8] was discovered by looking at the dependency mapping procedure of successive matrix multiplication steps of the form \( x(n) = x(n-1) \). Section 2 gives us a simple informal explanation of the original algorithm for matrix multiplication within and without result reusability. In section 3 a double pipe plus multicast bus model is shown. Then in section 4, applications of this new design for the matrix polynomial calculation and to the computation of matrix powers are presented. Finally, the concluding remarks are given in section 5.

2. Result reusable matrix multiplication systolic array.

This algorithm uses \( m \times k \) cell processors arranged in a mesh configuration to multiply an \( m \times n \) matrix \( A \) by an \( n \times k \) matrix \( B \). Mesh rows are numbered \( 1, \ldots, m \) and \( 1, \ldots, k \). Matrices \( A \) and \( B \) are fed into the boundary processors in column 1 and row 1, respectively, as shown in figure 1 for \( n = 4 \), \( m = 5 \) and \( k = 3 \). Note that row \( i \) of matrix \( A \) lags one time unit behind row \( i-1 \) for \( 2 \leq j \leq k \). This ensures that \( A_{ij} \) meets \( B_{kj} \) in processor \( P(i,j) \) at the right time. At the end of the algorithm, element \( C_{ij} \) of the product matrix \( C \) resides in cell processor \( P(i,j) \). Initially \( C_{ij} \) is zero. Subsequently, when \( P(i,j) \) receive two inputs \( a \) and \( b \) it:

1. Multiplies them,
2. Adds the result to \( C_{ij} \)
3. Sends \( a \) to \( P(i,j+1) \) unless \( j = k \), and
4. Sends \( b \) to \( P(i+1,j) \) unless \( i = m \)

Elements \( A_{m1} \) and \( B_{1k} \) take \( m + k + n - 2 \) steps from the beginning of the computation to reach \( P(m,k) \). Since \( P(m,k) \) is the last processor to terminate, this many steps are required to compute the product. Assuming \( A \) and \( B \) \( n \times n \) matrices the time taken will be \( 3n - 2 \) i.e. \( t(n) = O(n) \).
Systolic Array for matrix multiplication

Figure 1
Doppler effect in systolic Arrays

Figure 2
Quinton's result reusable matrix multiplication systolic array

\[
\begin{align*}
\mathbf{a} & = \begin{bmatrix}
\mathbf{a}_{11} & \mathbf{a}_{12} & \mathbf{a}_{13} \\
\mathbf{a}_{21} & \mathbf{a}_{22} & \mathbf{a}_{23} \\
\mathbf{a}_{31} & \mathbf{a}_{32} & \mathbf{a}_{33}
\end{bmatrix} \\
\mathbf{b} & = \begin{bmatrix}
\mathbf{b}_{11} & \mathbf{b}_{12} & \mathbf{b}_{13} \\
\mathbf{b}_{21} & \mathbf{b}_{22} & \mathbf{b}_{23} \\
\mathbf{b}_{31} & \mathbf{b}_{32} & \mathbf{b}_{33}
\end{bmatrix}
\end{align*}
\]

Figure 3
However this implementation suffers for the drawback that the results do not move and consequently a systolic output scheme has to be designed in order to recover them.

Suppose now, that we want the results $C_{ij}$ to be reused for the next multiplication step, i.e., they should enter by the left column of the array. Consider row $i$ of the array of figure 2 and assume that a new right to left link is provided between the cells. The final value $C_{ij}$ is obtained when the last element $A_{in}$ visits cell $i,j$. When this occurs, we can then send the final result to the left, and $j$ steps later, the result reaches the left cell of the array. However, the rate at which the $C_{ij}$'s leaves the array is only half the rate at which the $A_{ij}$'s enter (also known in Physics as Doppler effect).

A very simple solution [8] consists in slowing down by a factor two the input rate of $A$ and $B$, without any other change (see figure 3) or without delay, with a high speed time multiplexed shared bus (multicast lines) as will be explained in section 3. Figure 3 shows the design that is obtained. Each cell of the left column is provided with a feedback loop controller with a multiplexer which makes it possible to select either the input of a new matrix $A$ or to feedback the result of the previous multiplication. The total time now will be $(3n(nm - 1) - 2)/2$, where $nm$ is the number of matrices to be multiplied.

3. - Double pipe and multicast result reusable matrix multiplication systolic array.

The two techniques to be described here will accelerate the matrix product and the result reusable matrix multiplication.

The multicast (type of broadcast) consists in eliminate the slowing down synchronization procedure by the use of special multiple bus that permits immediately transmit the results of cell $C_{ij}$ to the multiplexer at the left boundary of the systolic array (see figure 4), this divide by two the total time of the algorithm.

The second method uses a second level of pipelining inside each cell processor, consisting of a two four stages floating point operators, one for floating point addition and the other for floating point multiplication (see figure 5), therefore the real pipe length is multiplied by 8, if each stage needs $1/8$ of the cell processing time, we can speedup the input rate of $A$ and $B$ by a factor of 8, therefore we divide by 8 the total time of the operation. This changes generates some minor modifications to the cell processors owing to the operands synchronization of the add operator with the multiplication result (figure 5). In order to synchronize it is necessary to attach a four stages shift register (Y input in the figure). The time required for this systolic network is then:

$$t(n) = (3n(nm - 1) - 2)/8$$
Figure 4

Multicast line, result reusable matrix multiplication systolic array
Two four stages floating point operators second level of pipelining

Figure 5
If \( nm = 2 \), this is only one product so,

\[
t(n) = 0.37n - 0.25
\]

plus the time for output the results \( n \). This is the fastest algorithm known for this problem (see table 1).

<table>
<thead>
<tr>
<th>Author</th>
<th>( T ) (time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cheng-Sahni</td>
<td>3</td>
</tr>
<tr>
<td>Jagadish-Kailath</td>
<td>2,5</td>
</tr>
<tr>
<td>Kak</td>
<td>2</td>
</tr>
<tr>
<td>Kung-Lehman</td>
<td>4</td>
</tr>
<tr>
<td>Kung-Leiserson</td>
<td>4</td>
</tr>
<tr>
<td>Li-Wah</td>
<td>2</td>
</tr>
<tr>
<td>Melkemi-Tchuente 1</td>
<td>3</td>
</tr>
<tr>
<td>Melkemi-Tchuente 2</td>
<td>2</td>
</tr>
<tr>
<td>Porter-Aravena</td>
<td>2</td>
</tr>
<tr>
<td>Preparata-Vuillemin</td>
<td>3</td>
</tr>
<tr>
<td>Quinton-Joinnault-Gachet</td>
<td>4</td>
</tr>
<tr>
<td>Weiser-Davis</td>
<td>2</td>
</tr>
<tr>
<td>Benaini-Robert</td>
<td>1,5</td>
</tr>
<tr>
<td>This paper (Isern)</td>
<td>0,37</td>
</tr>
</tbody>
</table>

Comparison of existing arrays (complexity order)

Table 1

4.- Applications of the systolic array design.

This section describes some applications of the result reusable matrix multiplication double pipe multicast systolic array.

4.1.- Matrix Polynomial

A simple modification [8] allows us to perform multiply-and-add step of the form \( x(n) \leftarrow x(n-1) \cdot A + B \). The accumulation of \( B \) can be done when the values \( x(n-1) \cdot A \) reach the left row of the array, as depicted by figure 6. Using Horner's rule for the calculation of the matrix polynomial

\[
P = \sum_{K=0}^{M} B_k \cdot A^k, \quad P \text{ can be carried on using the following recurrence:}
\]

\[
x(0) = B_M
\]

\[
x(k) = x(k-1) \cdot A + B_{M-k}
\]

\[
P = x(n)
\]

Therefore \( P \) can be computed in the new design on:
Original systolic array for matrix polynomial calculation, without double pipe and multicast

Figure 6
4.2. - Powers of a matrix.

A well known efficient algorithm for the computation of matrix powers \( P = A^N \) was described by Knuth in 1969 [9]. The method is quite ancient; it appeared before 200 B.C. in Pingala's Hindu Classic Chandah-Sutra; however, there seem to be no other references to this method outside of India during the next 2000 years. It does not require storage and consists in using the binary representation of \( N \) to control a sequence of square or multiply by \( A \) steps. We take the binary representation of \( N \) and we recorded in a new sequence, \( C_q \) is \( S \) if \( N_k = 1 \) and \( S \) if \( N_k = 0 \), each \( S \) means "square the current matrix" and \( X \) "multiply the current matrix by \( A \)". The following recurrences provides the basic calculation \( A^N \):

\[
\begin{align*}
x(0) &= A \\
x(n) &= (x(n-1))^2 & \text{if } C_{q-n} = S \\
P &= x^N & \text{else } x(n-1) \cdot A
\end{align*}
\]

The modification here [8] is, route the result also the result to the upper row of the array and depending of \( C_q \), \( A \) or \( x^{(n-1)} \) will be from top to bottom (see figure 7).

As the binary representation of \( N \) has \( \log_2 N+1 \) the time on the new design is:

\[
t(n) = (3N(\log_2 N+1)-2)/8
\]

5. - Concluding remarks.

It has been described a very simple and the fastest systolic array for matrix multiplication (table 1). It also has the advantage that the results can be reused, without any storage also this architectural design can be used for the calculation of matrix polynomials and Powers of matrices with excelents times, by the use in all the cases of a double pipe and a type of multicast scheme.
Original systolic array for the power of matrix calculation, without double pipe and multicast

Figure 7
REFERENCES.


