SCIENTIFIC PUBLICATIONS OF THE STATE UNIVERSITY OF NOVI PAZAR
SER. A: APPL. MATH. INFORM. AND MECH. vol. 3, 1 (2011), 53-58.

Selecting Mathematical Method for Systolic Processing

M. P. Bekakos, I. Ž. Milovanović, T. I. Tokić, Ć. B. Dolićanin, E.I.Milovanović

Abstract: The most important aspect in the design of a systolic arrays is the mapping of the algorithm to the processor array. However, not all algorithms can be systolized. Only highly regular algorithms with the structure of nested loops are suitable for systolic implementation. Therefore, one has to choose an appropriate mathematical method that can be systolized. In this paper we analyze the problem of choosing the most suitable mathematicam method for systolic implementation. We illustrate this problem on the example of matrix multiplication.

Keywords: systolic arrays, matrix multiplication

1 Introduction

In the last decade, the rapid development of VLSI computing techniques has had a significant impact on the development of novel computer architectures. One class of architectures, the so-called systolic arrays, first introduced by Kung [3], has gained popularity because of its ability to exploit massive parallelism and pipelining to achieve high performance. Informally, a systolic system can be envisaged as an array of synchronized processors (or processing elements, abbreviated as PEs) which process data in parallel by passing them from PE to PE in regular rhythmic fashion. Systolic arrays have balanced, uniform, grid-like architectures of special PEs that process data like n-dimensional pipeline. In psychology, the term systolic describes the contraction (systole) of the heart, which regularly sends blood to all cells in the body. Analogously, a systolic computer performs operations in repetitive, rhythmic manner.

The fundamental concept behind a systolic architecture is that von Neumann bottleneck is greatly alleviated by repeated use of fetched data item in a physically distributed array of PEs. The regularity of these arrays leads to inexpensive and dense VLSI implementations, which imply high performance at low cost. Application specific systolic arrays fit naturally into the concept of hardware library, where functional units are in relation to the host computer as subroutines from a software library are to production code.

Manuscript received September 23 2010 ; accepted January 12 2011

M. P. Bekakos is with Democritus University of Thrace, Xanthi, Greece; I. Ž. Milovanović, T. I. Tokić, E. I. Milovanović are with the University of Niš, Faculy of Electronic Engineering, Niš, Serbia; Ć. B. Dolićanin is with the State Unersity of Novi Pazar, Novi Pazar, Serbia.

Systolic arrays have been designed for a wide variety of computationally intensive problems in signal processing, numerical problems, pattern recognition, database and dictionary machines, graph algorithms, and automata. To meet performance requirements of these applications, it is often necessary to dedicate hardware with parallel processing capabilities to these specialized operations. Systolic arrays, due to their structural regularity and consequent suitability for VLSI implementation, are frequently used for this purpose.

Three factors have contributed to the evolution of systolic arrays into leading approach for computationally intensive applications [1]:

- Technology Advances: The growth of VLSI/WSI technology. Smaller and faster gates allow a higher rate of on-chip communication as data have to travel a shorter distance. Higher data densities permit more complex PEs with higher performance and granularity. Progress in design tools and simulation techniques ensure that a systolic PE can be fully simulated before fabrication, reducing the chances it will fail to work as designed. The regular modular arrays also decrease time to design and test , as fully tested unique cells can be copied quickly and arranged into systolic array. In addition, as VLSI/WSI designs become more complex, a regular systolic array provides an efficient way to ensure fault tolerance. Any fault-tolerant mechanism built into one PE is extensible to all PEs. Field Programmable Gate Arrays (FPGA) permit a reconfigurable architecture in which the architecture of the PE can be programmed to match the computation required by the algorithm.

- Parallel processing: Efforts to add concurrency to conventional von Neumann computers have yielded a variety of techniques such as coprocessors, multiple functional units, data pipelining (and parallelism), and multiple homogenous processors. Systolic arrays combine features from all of these architectures in a massively parallel architecture that can be integrated into existing platforms without a complete redesign. A systolic array can act as a coprocessor with multiple functional units and/or processors in an n-dimensional pipeline. The I/O requirements are mitigated using data pipelining by allowing adjacent PEs to reuse the input data, but a systolic array has also incremental instruction processing or computational pipelining. Each PE computes an incremental result, and both the input data and partial results flow through the array.

- Demanding scientific applications: The technology growth in the past three decades has produced the computers that make it feasible to attack scientific applications on a larger scale. New applications requiring increased computational performance have been developed that were not possible earlier. Examples of these application include interactive language (or speech) recognition, text recognition, virtual reality, database operations, and real-time image and signal processing. These applications require massive, repetitive parallel processing, and hence, systolic computing.

A number of implementation issues determine a systolic array performance and efficiency. Designers must be able to understand the performance trade-offs early in the design cycle to quickly produce economical designs. Some of the issues are: algorithm mapping, integration into existing systems, extensibility, PE functionality, clock synchronization, reliability. The most important aspect in the design of a systolic computer is the mapping of the algorithm to the processor array. In the systolic paradigm, every algorithm requires a specialized systolic design in which communication data streams, PE definitions, and input-output patterns are customized. It is usually considered that a systolic algorithm for a given problem is already at a disposal. But, in practice this is not the case. Namely, one has first to construct the systolic algorithm. However, not all algorithms can be systolized. Only highly regular algorithms with the structure of nested loops are suitable for systolic implementation. Therefore, one has to choose an appropriate mathematical method that can be systolized. Unfortunately, there is no general rule how to find such method. Instead, the problem is solved from case to case. To cope with this problem efficiently, one must have deep understanding of the problem to be solved, systolic algorithms, systolic processing and systolic array design methodology. In this paper we will illustrate this problem on the example of finding the most suitable mathematical method for systolic computing of matrix multiplication.

Computational tasks can be conceptually classified into two families: compute-bound computations and I/O-bound computations. For example, matrix multiplication represents compute bound computation. On the other hand, adding two matrices is I/O-bound task. Speeding-up a compute-bound computations can often be accomplished in a relatively simple and inexpensive manner, that is by the systolic approach, without increasing I/O requirements. I/O-bounded tasks are not convenient for systolic implementation.

2 The choice of mathematical method

Let $A = (a_{ik})$ and $B = (b_{kj})$ be two matrices of order $n \times n$. We are interested to find out the most suitable mathematical method for systolic computing of the product $C = A \cdot B$, $C = (c_{ij})$. A lot of methods, both sequential and parallel, can be find in the literature for computing this product (see for example [2-6], [8]). Here, we pick up and analyze four of them to find out if they can be systolized.

- The method of scalar (inner) products

  The computation of product $C = A \cdot B$, can be accomplished in the following way

$$C = \left[\ \vec{A}_{1\bullet} \quad \vec{A}_{2\bullet} \quad \dots \quad \vec{A}_{n\bullet}\ \right]^T \cdot \left[\ \vec{B}_{\bullet 1} \quad \vec{B}_{\bullet 2} \quad \dots \quad \vec{B}_{\bullet n}\ \right] \tag{1}$$

  where $\vec{A}_{i\bullet}$ is the i-th row-vector of matrix A, and $B_{\bullet i}$, is the j-th column-vector of matrix B.

- The method of medium products

$$C = \left[\ A\vec{B}_{\bullet 1} \quad A\vec{B}_{\bullet 2} \quad \dots \quad A\vec{B}_{\bullet n}\ \right] \tag{2}$$

- The method of outer products

$$C = \sum_{k=1}^{n} C^{(k)}, \tag{3}$$

where

$$C^{(k)} = \vec{A}_{\bullet k} \vec{B}_{k\bullet} \tag{4}$$

- The standard method

$$c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}, \qquad 1 \le i \le n, \quad 1 \le j \le n \tag{5}$$

i.e.

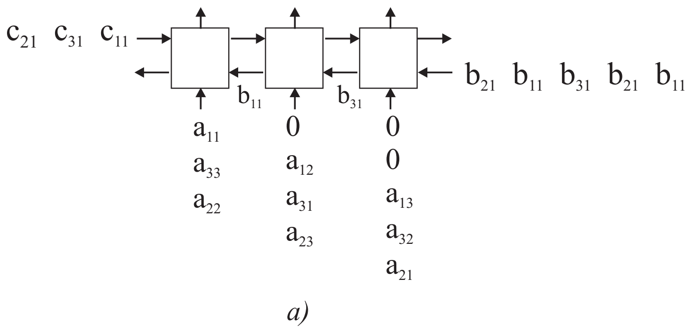
$$c_{ij}^{(k)} = c_{ij}^{(k-1)} + a_{ik} b_{kj}, \quad c_{ij}^{(0)} \equiv 0, \quad c_{ij}^{(n)} \equiv c_{ij} \tag{6}$$

for all $i = 1,2,\ldots,n$, $j = 1,2,\ldots,n$ and $k = 1,2,\ldots,n$.

The method of scalar products computes n^2 scalar products of the corresponding vectors. It possesses high degree of fine-grain parallelism. Each of the vectors in (1) takes part in n computations which implies that pipelining could be exploited. The basic operation in this case is product of two vectors. However, it falls into category of I/O-bounded tasks, and hence, it is not convenient for systolic implementation.

The method of medium products, (2), consists of n products of matrix A and corresponding column-vectors of matrix B. It possesses high degree of coarse-grain parallelism which makes it inconvenient for systolic implementation on Two-dimensional (2D) arrays. Namely, processing elements (PEs) in the systolic array are capable to perform only simple binary operations, like adding and/or multiplying two scalars. On the other hand, matrix-vector product is compute-bonded computation and it can be efficiently implemented on one-dimensional (1D) systolic arrays (see, for example, [9]). We can use this fact to design bidirectional linear systolic array (BLSA) which performs the computation of type $A \cdot \vec{B}_{\bullet j}$. The product of two matrices, $C = A \cdot B$, can be obtained by repeating the computation of type $A \cdot \vec{B}_{\bullet i}$ n-times (for $i = 1,2,\ldots,n$). Matrix product is obtained after $T_{tot} = n(3n-2)$ time units (one time unit is assumed to be time needed to perform one add-multiply operation). So, we conclude that the method of medium products is not suitable for implementation on 2D arrays but it is for the 1D arrays. Figure 1. shows data schedule at the beginning of the computation of product $A \cdot \vec{B}_{\bullet 1}$ in the BLSA when n=3.

Similarly, the method of outer products, given by (3) and (4), is not suitable for the synthesis of 2D systolic arrays. However, the expression (4) can be viewed as the product of two rectangular matrices of dimension $n \times 1$ and $1 \times n$, respectively. Since the inner dimension is equal 1, it is not difficult to conclude that during the synthesis process 2D systolic array degrades to 1D array. Thus, for example, in the case of projection direction vector $\vec{\mu} = [1\ 1\ 0]^T$, expression (4) can be used to design the bidirectional linear systolic array which in turn can be used to compute the product $C = A \cdot B$ by repeating the computation of type (4) n times. This array consists of n PEs and it can compute matrix product



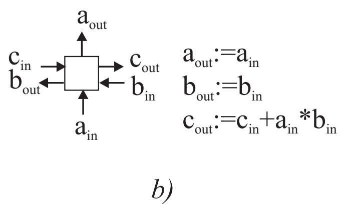


Fig. 1. a) Data flow in the BLSA during the computation of $A \cdot \vec{B}_{\bullet 1}$ for n=3; b) Functional property of the PE

for $T_{tot} = n(2n-1)$ time units. This time is better than one obtained by the BLSA synthesized according to the method of medium products. Figure 2. shows data schedule at the beginning of the computation of $C^{(1)} = \vec{A}_{\bullet 1}\vec{B}_{1\bullet}$ in the BLSA for $n = 3$.

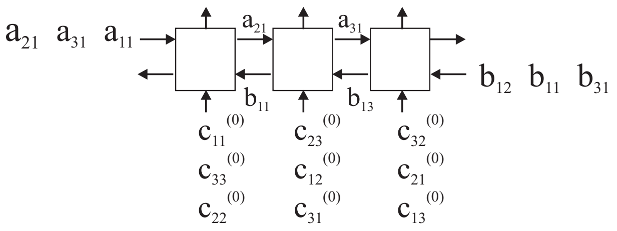


Fig. 2. Data flow in the BLSA during the computation of $C^{(1)} = \vec{A}_{\bullet 1}\vec{B}_{1\bullet}$ for n=3

According to the previous analysis it can be concluded that method defined by (1) is not suitable for systolic implementation. Methods defined by (2) and (3), (4), are suitable for the implementation on 1D systolic arrays but not for 2D arrays. The standard method, defined by (5) and (6) is convenient for implementation on both 1D and 2D systolic arrays (see, for example [2,3,4,7]). It satisfies all the constraints for efficient systolic implementation.

3 Conclusion

We have analyzed the problem of choosing the most suitable mathematical method for systolic implementation. We have illustrated this problem on the example of matrix multiplication. We have chosen four different methods and analyzed them. We have shown that one of the chosen methods is inconvenient for systolic implementation, two are partly suitable and that the standard one meets all requirements for systolic implementation. The problem of matrix multiplication was chosen only for the illustration purposes, since it is well-known problem. Similar analysis should be performed for every other problem. For successful implementation on systolic array the chosen method, and consequently the algorithm, has to satisfy the following constraints: it must be highly regular (nested loop algorithms), the basic operations should be simple, and the required communications must be of near-neighbor type.

References

[1] K. T. JOHNSON, A. R. HURSON, B. SHIRAZI, *General purpose systolic arrays*, IEEE Computer 11 (1993), 20-31.

[2] S. Y. KUNG, it VLSI array processors, Prentice Hall, New Jersey, 1988.

[3] H. T. KUNG, *Why systolic architectures?*, Computer, 15 (1982), 37-46.

[4] S. G. SEDUKHIN, it The designing and analysis of systolic algorithms and structures, Programming, 2 (1990), 20-40. (in Russian)

[5] V. V. VOEVODIN, *Mathematical models and methods in parallel processing*, Nauka, Moscow, 1986. (in Russian)

[6] I. Z. MILENTIJEVIĆ, I. Ž. MILOVANOVIĆ, E. I. MILOVANOVIĆ, M. K. STOJČEV, *The design of optimal planar systolic arrays for matrix multiplication*, Comput. Math. Appl., 33, 6, (1997), 17-35.

[7] I. Ž. MILOVANOVIĆ, T. I. TOKIĆ, E. I. MILOVANOVIĆ, M. K. STOJČEV, *Determining the number of processing elements in systolic arrays, Facta Universitatis*, Ser. Math. Inform., Vol 15, 1 (2000), 123-132.

[8] J. M. ORTEGA, *Introduction to parallel and vector solution on linear systems*, Plenum Press, 1988.

[9] I. Ž. MILOVANOVIĆ, E. I. MILOVANOVIĆ, I. Z. MILENTIJEVIĆ, M. K. STOJČEV, *Designing of processor-time optimal systolic arrays for band matrix-vector multiplication*, Comput. Math. Appl., Vol 32, 2 (1996), 21-31