A linear algebra processor using Monte Carlo methods

A linear algebra processor using Monte Carlo methods

T. P. Plaks*, G. M. Megson**, O. Cadenas** and V. N. Alexandrov**
* London South Bank University
** The University of Reading, UK

Introduction

Many problems in control engineering, scientific computations, military and space applications can be reduced to the solution of basic Linear Algebra problems, for solving systems of linear equations: $AX = B$. For example many problems arising from partial differential equations where local effects or ordering of finite difference/finite elements produce recurrences with limited spatial and temporal extension. Because the problems are often large and involve many zero elements there is great need to speed up computations or exploit structure of the matrices [13, 8]. In the former case to allow larger problems to be solved or more iterations of a problem with varying parameters to be solved in the same time. In the sparse case both computation time and hardware involved can be optimized by removing redundancies from standard dense matrix approaches to the problem.

There are many ways to solve the system, $AX = B$ [1, 7, 13], among which is to find first the inversion A^{-1} and then calculate $X = A^{-1}B$. The inversion of matrices with Banded structure [1, 7] is the classic problem in Linear Algebra. In this paper we consider of hardware implementation of these algorithms and, therefore, we will use the systolic or regular array approach for implementation of the proposed methods. The best known arrays for matrix inversion are based on the so-called Algebraic Path problem [6, 11, 12] and formulation of the Gauss Jordan method [3]. These arrays generally require $4n$ or $5n$ steps using n^2 processors and do not exploit any pivoting. Unfortunately the arrays cannot be easily adapted to the banded or sparse cases and require the same run-time and processors as for dense systems even for triangular matrices.

In this paper we introduce a Linear Algebra Processor based on Monte Carlo method for matrix inversion [4, 5, 14]. The processor is based on a regular array approach and is implemented on an FPGA platform. The most important part of the processor is the Matrix inversion regular processor array based on Monte Carlo method. The Monte Carlo array uses a modification of the technique proposed in [9] to exploit the zero structure in band systems [10]. This analysis results both in decreases in run-time and array latency but also in drastically reduced hardware compared with previous Monte-Carlo array for dense matrices and compared to previous solvers. In particular we consider arrays with $\mathcal{O}(TNw)$ to $\mathcal{O}(nTNw)$ elements where $T < \sqrt{n}$ is the length of a Markov chain, N is the number of chains, n is the order of the matrix and w is the band width. The run-time ranges from $\mathcal{O}(n^2 + T + N + w)$ to $\mathcal{O}(n + T + N + w)$ when the design is laid out in higher dimensions. Consequently we produce arrays with computation time $t < 3n + w$ steps where a step is the cost of a inner product.

A part of the Monte-Carlo based matrix inversion regular array has been implemented in Virtex-II to invert a matrix of size 64×64. The FPGA circuit implementation runs at a frequency of over 200 MHz. It is observed that a well structured and regular architecture can balance the computational resources of modern FPGA while maintaining high performance. The architecture is not limited by FPGA computational resources to invert matrices of moderate size, but the practical implementation of the architecture for matrices of large sizes is limited by the number of pin-out count of the design [2].

Monte Carlo method

Dense matrices

To find the inverse $C = A^{-1}$ of a dense matrix B of order n we compute $D = I - A$ (I is the identity matrix), and then use the Monte Carlo method described below [9]:

$$c_{rr'} \approx \frac{1}{N} \sum_{s=1}^{N} \left[\sum_{(j|k_j=r')} W_j \right]_s, \quad W_j = W_{j-1} \frac{d_{k_{j-1}k_j}}{p_{k_{j-1}k_j}}, \tag{1}$$

where $c_{rr'}$ is the element of C. The sequence of indices k_j, $0 \leq j \leq T$, form the states of a Markov chain with transition probabilities $p_{k_{j-1}k_j}$ from state k_{j-1} to k_j. Transitions will be denoted as tuples $< k_{j-1}, k_j >$. Thus, a transition determines the element of D, d_{k_{j-1},k_j}. For each j-th state the W_j is defined recurrently. A Markov chain starts with a row index, i.e. for computing an element $c_{rr'}$, the chain starts at $k_0 = r$, and only these W_j are included into sum of $c_{rr'}$ for which $k_j = r'$. To compute one row of inverse, N Markov chains (trials) are used over which the averages are computed to get the result. A Markov chain is denoted by index s, $1 \leq s \leq N$. Notice, that N and T are the parameters of a stochastic process: the mathematical expectations of the number and length of Markov chains respectively. Thus, N and T do not depend on the matrix size, but depend only on the required precision of computation and the norm of matrix [9]. In the following, we introduce matrix E such that $e_{p,q} = d_{p,q}/p_{p,q}$.

Banded and sparse matrices

The *Monte Carlo method*, because of the probabilistic nature, produces the possibility to cut off the zero parts of computations. Observe that the Monte Carlo methods operates with rows and columns of the matrix not as objects with fixed ordered structures but as sets of elements which are selected for computations in an arbitrary order. Thus, the number of elements in the sets does not affect the computation structure and zero elements can be easily excluded from the set [10]. Suppose, the size of matrix is 1024×1024, and there are 1% non-zero elements. Thus, there are about 10000 elements that can be rearranged as a new, dense matrix of size 100×100.

Conclusion

In this paper we present a linear algebra processor that is based on a novel, Monte Carlo method approach for matrix inversion. This approach allows us to achieve significant improvement in performance and to reduce the amount of required hardware. Also, the same approach is easily used for banded and sparse matrices reducing significantly the time and hardware resources.

References

[1] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation. Numerical Methods.* Prentice/Hall, Int., 1989.

[2] O. Cadenas, G. Megson, and T. Plaks. FPGA circuits for a Monte-Carlo based matrix inversion architecture. In T. P. Plaks and P. M. Athanas, editors, *ERSA'03. The International Conference on Engineering of Reconfigurable Systems and Algorithms, June 23–26, 2003, Las Vegas, USA*, page . CSREA Press, 2003.

[3] M. Cosnard. Designing parallel algorithms for linearly connected processors and systolic arrays. *Advances in Parallel Computing*, 1:273–317, 1990.

[4] J. H. Curtiss. Monte Carlo methods for the iteration of linear operators. *J. Math. Phys*, 32(4):209–232, 1954.

[5] J. H. Curtiss. A theoretical comparision of the efficiencies of two classical methods and Monte Carlo method for computing one component of the solution of a set of linear algebraic equations. In *Proc. Symp. on Monte Carlo Methods*, pages 191–233. John Wiley and Sons, 1956.

[6] J. M. Delosme. A parallel algorithm for the algebraic path problem. In M. C. et al., editor, *Parallel and Distributed Algorithms*, pages 67–78. Bonas, France, 1988.

[7] T. L. Freeman. *Parallel Numerical Algorithms*. Series in Computer Science. Prentice/Hall, Int., 1992.

[8] G. H. Golub and P. van Dooren, editors. *Numerical Linear Algebra, Digital Signal Processing and Parallel Algorithms*. NATO ASI Series F: Computer and Systems Sciences Vol. 70. Springer-Verlag, 1991.

[9] G. M. Megson, V. N. Aleksandrov, and I. T. Dimov. Systolic matrix inversion using a Monte Carlo method. *Parallel Algorithms and Applications*, 3: 311–330, 1994.

[10] T. P. Plaks, G. M. Megson, and V. N. Alexandrov. A family of efficient regular arrays for band and sparse matrix inversion using Monte Carlo methods. In M. Deville and R. Owens, editors, *Proc. of the 16th IMACS World Congress 2000 on Scientific Computation, Applied Mathematics and Simulation, Lausanne, Switzerland, August 21-25, 2000. (CD-ROM), 6 pages*. IMACS, USA, 2000.

[11] S. Rajopadhye. An improved systolic algorithm for the algebraic path problem. *INTEGRATION*, pages 279–296, 1993.

[12] T. Risset and Y. Robert. Synthesis of processor arrays for the algebraic path problem unifying old results and deriving new architectures. *Parallel Processing Letters*, 1(1):19–28, 1991.

[13] H. A. van der Vorst and P. van Dooren, editors. *Parallel Algorithms for Numerical Linear Algebra*. Advances in Parallel Computing, 1. North-Holland, Amsterdam, 1990.

[14] J. R. Westlake. *A Handbook of Numerical Matrix Inversion and Solution of Linear Equations*. John Wiley and Sons, New York, 1968.