SYSTOLIC IMPLEMENTATION OF SAMPLE-BY-SAMPLE CONJUGATE GRADIENT ALGORITHM

Ramin Baghaie
Helsinki University of Technolog
Laboratory of Telecommunications Technolog
P.O. Box 3000
02015 HUT, Finland

ABSTRACT

In this paper, we consider an efficient Conjugate Gradient (CG) algorithm that can be used when non-block processing or sample-by-sample update is of interest and compare its computational complexity with the conventional CG algorithm. We then review the VLSI implementation of the CG algorithms. Furthermore, for the hardware implementation of the sample-by-sample CG algorithm, a novel systolic array is proposed. With the aid of this systolic architecture, the time complexity of the algorithm will be reduced to $O(N)$.

I. INTRODUCTION

Amongst the adaptive algorithm, the Least Mean Squares (LMS) algorithm is the most widely used algorithm. Since its introduction, due to its simplicity and low complexity the LMS algorithm has been widely used [1]. However, the convergence rate of the LMS algorithm is rather slow and it depends on the eigenvalue spread of the autocorrelation matrix of the inputs. In order to combat this problem, recursive least-squares (RLS) algorithm can be used. This is only possible at the expense of heavy computational complexity and matrix manipulations. In order to reduce the computational complexity of the RLS algorithm, fast RLS algorithms have been proposed [1]. Unfortunately, fast RLS algorithms have a tendency to be numerically unstable. In order to improve the convergence rate of the LMS algorithm and yet avoid matrix inversions other algorithms such as the CG algorithm should be utilized.

Conjugate gradient methods were first developed in the early 1950s for iterative solutions of the finite linear equations [2]. Recently, they have found their way in many applications such as multiuser detection in Wideband Code Division Multiple Access (WCDMA) [3,4] and mobile user tracking systems [5,6].

In this paper, we review some of the techniques that have been utilized for the implementation of the CG algorithm. We the concentrate on the VLSI implementation of the algorithm. Special attention will be given to the systolic implementation of the CG algorithm. This paper is organized as follows. In Section II, we study the Sample-by-Sample CG algorithm and in Section III we compare its computational complexity with the conventional CG algorithm. In Section IV, the VLSI implementation of the algorithm is discussed. Concluding remarks are provided in Section V.

II. CONJUGATE GRADIENT ALGORITHM

Consider the linear system

$$\mathbf{Rw} = \mathbf{b} \tag{1}$$

where $\mathbf{R}$ is a N-by-N known symmetric a positive definite matrix, $\mathbf{b}$ is a known N-by-1 vector, and $\mathbf{w}$ is a N-by-1 unknown vector. Direct solution of (1), i.e., $\mathbf{w} = \mathbf{R}^{-1}\mathbf{b}$ requires matrix inversio $\mathbf{R}^{-1}$, which is computationally expensive. In many applications, when the dimension of the matrix $\mathbf{R}$ is large, or when $\mathbf{R}^{-1}$ should be calculated periodically, the problem becomes even more severe.

In order to solve (1) and yet avoid matrix inversions iterative methods should be utilized. This could be achieved b

minimizing the cost function $f(\mathbf{w})$ of the following equation.

$$f(\mathbf{w}) = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{R}\mathbf{w} - \mathbf{w}^{\mathrm{T}}\mathbf{b} \qquad (2)$$

For the minimization of (2) different methods such as the CG algorithm exist [7]. The Conjugate Gradient method combats this problem by using a set of linearly independent vectors $\mathbf{p}(n)$ that are conjugate gradient with respect to $\mathbf{R}$. Although different variety of the CG algorithm exist [7], they are not suitable for the non-block processing or sample-by-sample update.

In [8], a Conjugate Gradient algorithm was presented that is suitable for such applications. This algorithm also known as the Modified CG (MCG) is presented in Table I.

TABLE I
SAMPLE-BY-SAMPLE CG ALGORITHM

Set initial conditions:

$\mathbf{w}(0) = \mathbf{0}$, $\mathbf{g}(0) = \mathbf{b}(0)$, $\mathbf{p}(0) = \mathbf{g}(0)$

for $n = 1, 2, \ldots$

$$\mathbf{v}(n) = \mathbf{R}(n-1)\mathbf{p}(n-1)$$

$$\alpha(n) = \eta\frac{\mathbf{p}^{\mathrm{T}}(n-1)\mathbf{g}(n-1)}{\mathbf{p}^{\mathrm{T}}(n-1)\mathbf{v}(n)}$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \alpha(n)\mathbf{p}(n-1)$$

$$\mathbf{g}(n) = \lambda\mathbf{g}(n-1) - \alpha(n)\mathbf{v}(n) + \mathbf{x}(n)(d(n) - \mathbf{x}^{\mathrm{T}}(n)\mathbf{w}(n-1))$$

$$\beta(n) = \frac{(\mathbf{g}(n) - \mathbf{g}(n-1))^{\mathrm{T}}\mathbf{g}(n)}{\mathbf{g}^{\mathrm{T}}(n-1)\mathbf{g}(n-1)}$$

$$\mathbf{p}(n) = \mathbf{g}(n) + \beta(n)\mathbf{p}(n\text{-}1)$$

In this algorithm $\alpha(n)$ is the step size which minimizes the cost function along the search direction $\mathbf{p}(n)$ and it is used in the update of the weight vector $\mathbf{w}(n)$, $\mathbf{g}(n)$ is the residual of the function and points to the direction of the steepest descent. The new weight vector $\mathbf{w}(n)$ is computed as a linear combination of the previous weight vector and the search direction. Factor $\beta(n)$ ensures that the R-orthogonality is preserved between the new search directions.

III. COMPUTATIONAL COMPLEXITY OF CG

As compared to the conventional CG algorithm, in the MCG algorithm, computations of the residual vector $\mathbf{g}(n)$ and factor $\beta(n)$ are more complex and require a higher number of vector inner products Therefore, in this section the computational complexities of the MCG algorithm and the conventional CG also referred to as the Block Conjugate Gradient (BCG) [7] algorithms are studied and compared. In these calculations, one division has the same complexity as one multiplication. Note that for estimating the complexities, we have only considered the number of multiplications. This is due to the fact that multiplications are more complex than additions [9]. The results are shown in Table II.

TABLE II
COMPARISON OF COMPUTATIONAL COMPLEXITIES OF DIFFERENT CG ALGORITHMS

Algorithm	Number of multiplications
BCG	$K(N^2 + 5N + 2) - 2N - 1$
MCG	$N^2 + 10N + 3$

K: Maximum number of iterations for a block

When calculating the computational complexity of the BCG, one should note that in the last iteration for updating the filter coefficients the only necessary computation required is the calculation of the step size α. It is clear that the computational complexity of the BCG depends on K, the number of iterations. Thus, for large N the computational complexity of the BCG is K times of the complexity of the sample-by-sample CG algorithm.

IV. SYSTOLIC IMPLEMENTATION OF THE ALGORITHM

In this section, we discuss the implementation of the MCG algorithm and focus on developing an efficient very large scale integration (VLSI) array processor that is suitable for real time applications. For this purpose, we design a systolic array that targets

the most computationally intensive block of the MCG algorithm

A. Review of the Implementation Techniques

In many applications, in order to meet the demand for high computing rates and to achieve acceptable execution speed the conventional serial implementation methods are not sufficient. Thus, parallel architectures should be utilized. For matrix computations needed in the MCG algorithm of Table I, several classes of parallel architectures such as multiprocessors, systolic-type arrays, vector computers and array computers have been proposed [10].

Although many of these parallel architectures have demonstrated their effectiveness for executing matrix-vector computations, they may not be suitable for VLSI due to the broadcasting or complex interconnection network. These drawbacks led to the introduction of application-specific architectures and in particular systolic arrays, which are natural for matrix operations. They match the fine granularity of parallelis available in the computations and have very low overhead in communication and synchronization. In addition, the regular nature of systolic-type arrays meets the requirements for effective use of VLSI [10,11].

In the CG algorithm, the most computationa intensive operations are matrix-vector and inner vector products. Although, for these operations a variety of systolic architectures exists, the main problem is to map the entire algorithm onto a suitable and practical VLSI architecture. In the MCG however, due to the serial nature of the algorithm, there is a very low degree of parallelism. This is also the case when dealing with the BCG algorithm.

In [12], systolic implementation of the Preconditioned Block Conjugate Gradient algorithm (PCG) was discussed. In their work, they utilized the result in [13] and therefore, proposed a two-dimensional systolic architecture for the PCG algorithm. The PCG algorithm of [13] has a more parallel nature. However, due to the rounding errors the algorithm of [13] is unstable. Therefore, in practice the systolic architecture in [12] can not be reliable and may not feasible for practical applications. Furthermore, the above mentioned architecture is not suitable for the MCG algorithm of Table I.

B. Systolic Implementation

In this section, we design a novel systolic architecture that reduces the time complexity of the MCG algorithm to $O(M)$. As discussed in the previous section, due to the serial nature of the algorithm, there is a very low degree of parallelism in the algorithm. Furthermore, due to the iterative nature of the algorithm and different resetting schemes required for β, direct mapping of the MCG algorithm to ASIC is not trivial. Our systolic architecture targets the matrix-vector and vector-vector products needed in the calculation of the step size α and β. Consider the calculation of the step size α:

$$\alpha(n) = \eta \frac{\mathbf{p}^{\mathrm{T}}(n-1)\mathbf{g}(n-1)}{\mathbf{p}^{\mathrm{T}}(n-1)\mathbf{R}(n-1)\mathbf{p}(n-1)} \tag{3}$$

For simplicity, we introduce the new variable $\mathbf{v}(n)$ as follows:

$$\mathbf{v}(n) = \mathbf{R}(n)\mathbf{p}(n) \tag{4}$$

Due to the sample-by-sample update scheme in the MCG algorithm, the correlation matrix $\mathbf{R}(n)$ varies in every sample. However, in many applications such as user tracking [5] when calculating the weight vectors for M individual sources, $\mathbf{R}(n)$ remains the same. Therefore in such cases, for M consecutive iterations the same $\mathbf{R}(n)$ will be utilized [6]. As a result, for the systolic implementation of (3) a 2D array implementation is adopted. The elements of the $\mathbf{R}(n)= r_{ij}(n)$ $(i,j = 1, \ldots, N)$ are therefore preloaded into this array processor.

Now, consider the following vector-vector multiplications that are needed in the MCG algorithm.

$$\mathbf{pg}(n-1) = \mathbf{p}^{\mathrm{T}}(n-1)\mathbf{g}(n-1) \tag{5}$$

$$\mathbf{pv}(n) = \mathbf{p}^{\mathrm{T}}(n-1)\mathbf{v}(n) \tag{6}$$

For the realization of (5) and (6), a linear array is selected. For synchronization purposes, the linear array is placed below the 2D array. Fig. 1 illustrates the proposed

systolic array when $N = 4$. In Fig. 1b, the cell function of each PE is illustrated.

Furthermore, this architecture utilizes the availability of the residual vector $\mathbf{g}(n)$ and performs the following vector inner product needed in the calculation of β.

$$\mathbf{gg}(n-1) = \mathbf{g}^{\mathrm{T}}(n-1)\mathbf{g}(n-1) \qquad (7)$$

From Table I, it can be seen that for the calculation of the residual vector $\mathbf{g}(n)$, the vector $\mathbf{v}(n)$ of (4) is needed. This can be achieved in two ways. One way is that keep the elements of $\mathbf{v}(n)$ and after every 2N step transfer them sequentially from the PE2. Another way is to slightly modify the PE2 by adding an out port. Fig. 2 illustrates an alternative scheme for the PE2.

The total number of PEs required in this systolic architecture is N^2+N. In order to calculate the throughput of the systolic array, we assume that one time step of the global clock corresponds to the processing time required for each PE. For the initialization of PE1s, N time steps are needed. Thus, the total computation time required by the array is $3N$ steps. Fig. 3 illustrates the flow of data in the proposed systolic architecture for different time steps.

V. Conclusions

In this paper, we studied an efficient Conjugate Gradient algorithm that can be utilized when non-block processing or sample-by-sample update is of interest. We then compared the computational complexity of this algorithm with the conventional CG algorithm also known as BCG. Furthermore, for the hardware implementation of the sample-by-sample CG algorithm, a novel systolic array was proposed. With the aid of this systolic architecture, the time complexity of the CG algorithm will be reduced to $O(N)$. Future research should be directed towards mapping the system into a fixed number of processors when N is large.

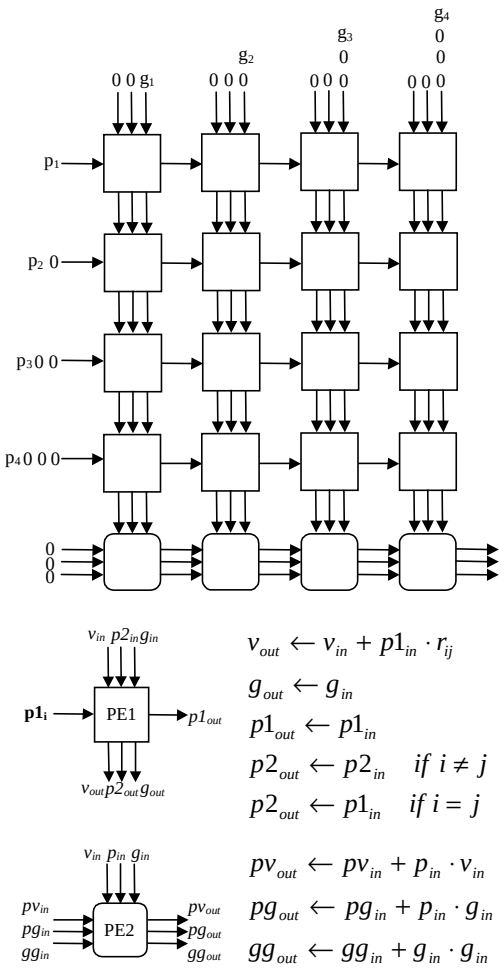

Fig. 1. a) Systolic architecture when N=4
b) Input-output ports and cell functions

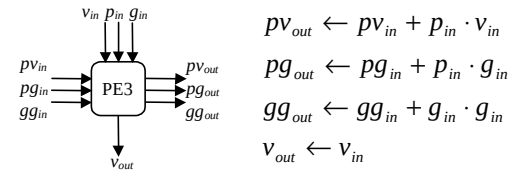

Fig. 2. Input-output ports and cell functions of the modified PE2

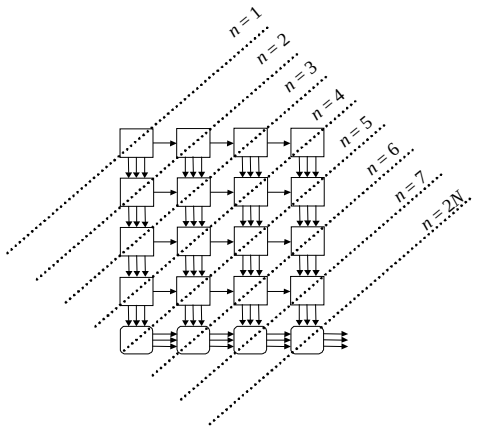

Fig. 3. Illustration of the data movement for each computation step

ACKNOWLEDGEMENTS

This work is part of a research project of the Institute of Radio Communication (IRC) funded by Technology Development Center (TEKES), NOKIA Research Center, Sonera Ltd. and the Helsinki Telephone Company.

REFERENCES

[1] P. Diniz, Adaptive Filtering, *Algorithms and Practical Implementation*. Kluwer Academic Publishers, 1997.

[2] M.R. Hestenes and E. Stiefel, "Method of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409-436, Dec. 1952.

[3] M. Juntti, B. Aazhang, and J. Lilleberg, "Iterative implementation of linear multiuser detection for dynamic asynchronous CDMA systems," *IEEE Transactions on Communications*, vol. 46, no. 4, pp. 503-508, Apr. 1998.

[4] S. Das, J.R. Cavallaro, and B. Aazhang, "Computationally efficient multiuser detectors," in *Proceedings IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC'97*, Helsinki, Finland, vol. 1, pp. 62-67, September 1997.

[5] P. Karttunen and R. Baghaie, "Conjugate Gradient Based Signal Subspace Mobile User Tracking," in Proceedings *IEEE Vehicular Technology Conference*, VTC'99 Spring, Houston, Texas, USA, May 1999.

[6] Z. Fu and E. Dowling, "Conjugate gradient projection subspace tracking, *IEEE Transactions on Signal Processing*, vol. 45, no. 6, pp. 1664-1668, June 1997.

[7] G.H. Golub and C.F. Van Loan, *Matrix computations*. The Johns Hopkins University Press, 1989.

[8] S.P. Chang and A.W. Willson, "Adaptive filtering using modified conjugate gradient," in *Proceedings 38th Midwest Symposium on Circuits and Systems*, Rio de Janeiro, Brazil, pp. 243-246, August 1995.

[9] T. Callaway and E. Swartzlander, "Optimizing arithmetic for signal processing," in *Proceedings IEEE Workshop on VLSI Signal Processing, V*, Napa Valley, California, pp. 91-100, October 1992.

[10] J.H. Moreno and T. Lang, "Matrix computations on systolic-type meshes, *IEEE Computer*, pp. 32-51, Apr. 1993.

[11] S.Y. Kung, *VLSI Array Processors*. Englewood Cliffs, New Jersey: Prentice Hall, 1988.

[12] J. Tasic, M. Gusev, and D.J. Evans, "Systolic implementation of preconditioned conjugate gradient method in adaptive transversal filters, *Parallel Computing*, vol. 18, no. 9, pp. 1053-1065, Sept. 1992.

[13] Y. Saad, "Practical use of polynomial preconditionings for the conjugate gradient method," *SIAM Journal of Scientific Statistical computing*, vol. 6, no. 4, pp. 865-881, Oct. 1985.