# IMPLEMENTATION OF MOBILE USER TRACKING SYSTEM

Ramin Baghaie and Petri Karttunen
Laboratory of Telecommunications Technology
Helsinki University of Technology
P.O. BOX 3000, FIN-02015 HUT, Finland
e-mail: {ramin.baghaie, petri.karttunen}@hut.fi

## Abstract

This paper considers the implementation of two mobile user tracking systems that are based on different step-by-step adaptation schemes. We first estimate the computational complexities of different units of the tracking systems. Based on these estimations, we partition the implementation task into two parts: software and hardware. For the hardware implementation of the tracking unit a systolic architecture is proposed. With the aid of the proposed systolic array the time complexity of the tracking unit is reduced to $O(M)$.

## 1 Introduction

Currently, in many applications such as beamforming based communication for mobile users, there is a large demand for tracking the location of the mobile user. Mobile users can be located when Direction-of-Arrival (DOA) estimates are established at the base stations. The user tracking problem can be solved by the methods of the trigonometric geometry from the intersection of the two or more Lines-of-Bearing (LOB).

There are numerous techniques that can be applied for such tracking problems. In the signal subspace based user tracking system of [1], for a step-by-step update an efficient conjugate gradient based algorithm was developed. In [2], an adaptive high-resolution noise subspace approach was presented.

In this paper, we consider implementation of the tracking systems in [1,2], and focus on developing efficient VLSI architectures that are suitable for real-time applications. This paper is organized as follows. Section 2 briefly presents the structure of the mobile user tracking systems in [1,2]. In Section 3, computational complexities of these systems are evaluated. In Section 4, implementation issues of the tracking systems are discussed. Furthermore, for the tracking unit of the signal subspace method a novel systolic architecture is designed that reduces the required computational time by an order of magnitude. Concluding remarks are provided in Section 5.

## 2 Mobile User Tracking System

Generally, tracking systems can be classified into two different groups depending on whether spatial structure or eigenstructure of the correlation matrix has been utilized. In the case of the *spatially structured method*, we establish a structured basis for the *signal subspace* that is spanned by the steering vectors of mobile users. However, in the case of the *eigenstructure based method* the $M$ orthogonal eigenvectors and eigenvalues are computed. The eigenvectors corresponding to the $N$ largest eigenvalues establish the orthogonal basis for the signal subspace. In the similar way, $M$-$N$ eigenvectors corresponding to noise eigenvalues establish the *noise subspace*, which is the complement of the signal subspace.

In the user tracking system of [1], for directly updating the signal subspace related array response vector, a step-by-step update scheme of the conjugate gradient based algorithm was developed. In [2], for tracking an eigenvector corresponding to the minimum eigenvalue of the sample correlation matrix, an adaptive high-resolution noise subspace approach was presented. Figure 1 illustrates the overall system model of the tracking systems in [1,2]. In the following subsections, we briefly describe the function of each unit.

### 2.1 Tracking Unit

The signal subspace tracking problem is formulated as the quadratic cost function for which a step-by-step update scheme has been implemented [1]. For the step-by-step update scheme the modified CG (MCG) algorithm has been utilized [3].

In the MCG algorithm of Table I, $\alpha(n)$ is the step size that minimizes the cost function through the line search procedure along the search direction $\mathbf{p}_{n-1}$. The residual vector $\mathbf{g}(n)$ points to the direction of the steepest descent. $\lambda_f$ is the forgetting factor and the auxiliary step size parameter $\eta$ should be $(\lambda_f - 0.5) \le \eta \le \lambda_f$ [3]. Factor $\beta(n)$ ensures that the $R$-orthogonality is preserved between search directions.

TABLE I
SAMPLE-BY-SAMPLE CG (MCG) 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, …

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

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

$$\mathbf{g}(n) = \lambda_f \mathbf{g}(n-1) - \alpha(n)\mathbf{R}(n-1)\mathbf{p}(n-1) + \mathbf{x}(n)(d(n) - \mathbf{x}^{\mathrm{H}}(n)\mathbf{w}(n-1))$$

$$\beta(n) = \max\left\{\frac{(\mathbf{g}(n) - \mathbf{g}(n-1))^{\mathrm{H}}\mathbf{g}(n)}{\mathbf{g}^{\mathrm{H}}(n-1)\mathbf{g}(n-1)}, 0\right\}$$

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

end

In the system derivation, for the sake of simplicity initially unknown antenna response vector $\mathbf{a}(\theta_k)$ has been replaced with the weight vector notation $\mathbf{w}(n)$. Vector $\mathbf{w}(n)$ as provided by the tracking unit converges to a steering vector which correlates best the desired user signal $d(n)$.

In the noise subspace method [2], for the step-by-step update scheme the following equations have been applied:

$$\lambda(n) = \frac{\mathbf{w}^{\mathrm{H}}(n)\mathbf{R}(n)\mathbf{w}(n)}{\mathbf{w}^{\mathrm{H}}(n)\mathbf{w}(n)} \quad (1)$$

$$\mathbf{g}(n) = -\mathbf{R}(n)\mathbf{w}(n) + \lambda(n)\mathbf{w}(n) \quad (2)$$

$$\alpha(n) = \frac{\mathbf{g}^{\mathrm{H}}(n)\mathbf{g}(n)}{\mathbf{g}^{\mathrm{H}}(n)\mathbf{R}(n)\mathbf{g}(n)} \quad (3)$$

$$\mathbf{w}(n) = \mathbf{w}(n-1) + \alpha(n)\mathbf{g}(n-1) \quad (4)$$

### 2.2 DOA Extraction Unit

In the DOA unit of the signal subspace approach, the new tracking angle estimates $\theta_k(n)$, ($k$=1, …, $N$) are computed through the Least Square (LS) fitting criterion which is based on the small deviations in the array manifold [1]. Thus, the LS criterion can be expressed as:

$$\hat{\theta}_k^{(\mathrm{LS})}(n) = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}\mathbf{z}_k(n) \quad (5)$$

where $\mathbf{z}_k(n)$ consists of array samples of $\theta_k(n)$, and $\mathbf{H}$ is the $M\times1$ observation vector.

On the other hand, the DOA unit needed in the noise subspace method is indeed more complex [2]. In this approach, the following zero-tracking method is utilized:

$$z_k(n+1) = z_k(n) + \Delta\mathbf{w}^{\mathrm{T}}(n)\frac{\partial\mathbf{W}(z_k(n))}{\partial z_k(n)} \quad (6)$$

where $\Delta\mathbf{w}(n) = \alpha(n)\mathbf{g}(n)$ and $\partial\mathbf{W}(z_k(n))/\partial z_k(n)$ can be expressed as:

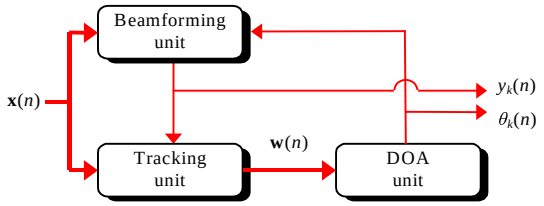



Figure 1: Overall mobile user tracking system for $M$ users [1,2]

$$\frac{\partial\mathbf{W}(z_k(n))}{\partial z_k(n)} = \frac{1}{w_1}\left[\frac{z_1^{-(M-1)}}{\prod_{m=2}^{M}(z_m - z_1)} \quad \cdots \quad \frac{z_M^0}{\prod_{m=1}^{M-1}(z_m - z_M)}\right]^{\mathrm{T}} \quad (7)$$

Thus, the angle estimate $\theta_k(n)$ can be calculated as:

$$\theta_k(n) = -\arcsin(\mathrm{imag}(\log(z_k(n)))/\pi) \quad (8)$$

### 2.3 Beamforming Unit

In this unit, for both methods the following conventional beamforming method was utilized [1].

$$y_k(n) = \left(\mathbf{w}^{\mathrm{H}}(n)\mathbf{w}(n)\right)^{-1}\mathbf{w}^{\mathrm{H}}(n)\mathbf{x}(n) \quad (9)$$

## 3 Complexity of the System

In Table II for both signal and noise subspace methods, the order of computational complexity of different units of the tracking systems are estimated and compared.

TABLE II
COMPARISON OF COMPUTATIONAL COMPLEXITIES

| Unit | Order of complexity | |
|---|---|---|
| | Signal Subspace Method | Noise Subspace Method |
| *Tracking* | $O(NM^2)$ | $O(2M^2)$ |
| *DOA* | $O(NM)$ | $O(NM^2)$ |
| *Beamforming* | $O(NM)$ | $O(NM)$ |

*M: Number of antennas*
*N: Number of sources*

When estimating the computational complexity of the DOA unit, complexity of elementary functions such as log($x$), exp($x$), or sin($x$) was not considered. This is due to the fact that for the implementation of such functions different techniques such as table oriented methods, iterative methods, and polynomial approximations can be utilized. In fact in [4] it was shown that the aforementioned functions can be evaluated to $n$ significant bits in $O(Mu(n)\log(n))$ steps, where $Mu(n)$ is units of time required to multiply $n$-bit numbers.

Although the performance of the noise subspace method is slightly better in a stationary signal scenario as compared to the signal subspace method, however, in a non-stationary channel it has a similar tracking performance [2].

Nevertheless, tracking mobile users in a non-stationary channel is much more demanding as compared to the parameter estimation schemes in a stationary channel.

From the implementation point of view, the complexity of the DOA unit in the noise subspace method is an order of magnitude larger than the equivalent unit in the signal subspace method. Furthermore, in the noise subspace method, orthogonalization procedures such as the Gram-Schmidt method may be needed. Therefore, by comparing both the performances and complexities of the proposed tracking systems in [1,2], we conclude that the signal subspace method is a more attractive scheme. Consequently, in Section 4 we will concentrate on the implementation of the tracking system in [1].

As can be seen from Table II, in the signal subspace method the tracking unit has the highest order of complexity. The core of this unit is the sample-by-sample CG algorithm of Table I. As compared to the conventional CG algorithm also referred to as Block Conjugate Gradient (BCG), in the MCG algorithm the computation of the residual vector $\mathbf{g}(n)$ and the factor $\beta(n)$ are more complex and require a higher number of vector inner products. Thus, we study and compare the computational complexities of the sample-by-sample CG and the BCG algorithms. The results are shown in Table III.

It is clear that the computational complexity of the BCG depends on the number of iterations $I$ and for a large $M$, the BCG is $I$ times more complex than the sample-by-sample CG algorithm.

TABLE III
COMPARISON OF COMPUTATIONAL COMPLEXITIES OF TWO CG ALGORITHMS

| *Algorithm* | *Number of complex multiplications* |
|---|---|
| BCG | $I(M^2+5M+2)-2M-1$ |
| MCG | $M^2+10M+3$ |

*I: Maximum number of iterations for a block*

## 4 Implementation of the System

For real-time applications, in order to meet the demand of high sampling rates the conventional DSP-based implementation methods are not sufficient. Consequently, for the implementation of units with high computational complexity, application-specific integrated circuits (ASIC) should be utilized.

As can be seen from Table II, in the signal subspace method the most computationally intensive block is the tracking unit. In this unit, the order of complexity for $N$ sources is $O(NM^2)$. In Figure 2, the hardware (HW)/software (SW) partitioning of this system is illustrated.

In this section, we discuss the implementation of the HW partition that is needed for the MCG algorithm and focus on developing an efficient 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.

### 4.1 Review of the Implementation Techniques

As can be seen from Table I, in the tracking unit of [1] the most computationally intensive operations are the matrix computations. Furthermore, in order to meet the demand for high sampling rates and to achieve acceptable execution speed the conventional serial implementation methods are not sufficient. Thus, parallel architectures should be utilized.

For the matrix-vector 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 [5].

Although many of these parallel architectures have demonstrated their effectiveness for executing matrix-vector computations, due to their broadcasting or complex interconnection network they may not be suitable for VLSI implementation. 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 parallelism 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 [5,6].

Although, for the matrix operations needed in the MCG algorithm, 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 algorithm, due to the serial nature of the algorithm, there is a very low degree of parallelism and therefore, parallelization of the algorithm is not trivial. Similarly, this is the case when implementation of the BCG algorithm is of interest [7,8].

### 4.2 Systolic Implementation

In this section, we design a 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 the requirement for different resetting schemes for $\beta$ [3], direct mapping of the MCG algorithm to ASIC is not practical.

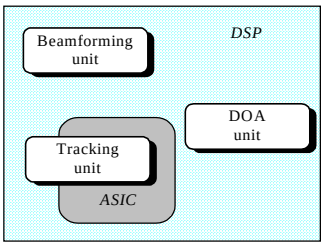



Figure 2: Partitioning of the overall system into HW/SW

Therefore, our systolic architecture targets the matrix-vector and vector-vector products needed in the calculation of the step size $\alpha$ and the factor $\beta$ of Table I.

Consider the calculation of the step size $\alpha$:

$$\alpha(n) = \eta \frac{\mathbf{p}^{\mathrm{H}}(n-1)\mathbf{g}(n-1)}{\mathbf{p}^{\mathrm{H}}(n-1)\mathbf{R}(n-1)\mathbf{p}(n-1)} \qquad (10)$$

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

$$\mathbf{v}(n) = \mathbf{R}(n)\mathbf{p}(n) \qquad (11)$$

Due to the sample-by-sample update scheme in the MCG algorithm, the correlation matrix $\mathbf{R}(n)$ varies in every sample. However, when calculating the weight vectors for $N$ individual sources, $\mathbf{R}(n)$ remains the same and therefore, for $N$ iterations the same $\mathbf{R}(n)$ is used. As a result, for the systolic architecture a 2D array implementation is adopted. The elements of the $\mathbf{R}(n)=r_{ij}(n)$ ($i,j$ = 1, …, $M$) are preloaded into this array processor and remain constant for $N$ iterations.

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

$$\mathbf{pg}(n-1) = \mathbf{p}^{\mathrm{H}}(n-1)\mathbf{g}(n-1) \qquad (12)$$

$$\mathbf{pv}(n) = \mathbf{p}^{\mathrm{H}}(n-1)\mathbf{v}(n) \qquad (13)$$

For the realization of (12) and (13), a linear array is selected. For synchronization purposes, the linear array is placed below the 2D array. Figure 3 illustrates the proposed systolic array when $M$=4. In Figure 3b, the cell function of each Processor Element (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 $\beta$.

$$\mathbf{gg}(n-1) = \mathbf{g}^{\mathrm{H}}(n-1)\mathbf{g}(n-1) \qquad (14)$$

As can be seen from Table I, for the calculation of the residual vector $\mathbf{g}(n)$, matrix-vector multiplication of (11), i.e. vector $\mathbf{v}(n)$, is required. For utilizing $\mathbf{v}(n)$ two methods can be exercised. One method is to keep the elements of $\mathbf{v}(n)$ by allocating a local memory to each PE2 and then sequentially transfer them to the host from the last PE2 of the linear array. The second method is to slightly modify the PE2s. This can be achieved by adding an extra output port to the PE2s as it is illustrated in Fig. 4. The total number of PEs required in this systolic architecture is $M^2+M$.

For the implementation of the complex multipliers needed in the PEs, Strength Reduction (SR) transformation technique has been utilized [9]. By utilizing the SR transformation the total number of real multiplications needed in a complex multiplier is reduced to only three.

To clarify this further, consider the complex multiplications required in PE1, i.e. $pr = (pr_R + jpr_I) = p1 \cdot r$. By utilizing the SR technique we have:

$$pr_R = (p1_R - p1_I)r_I + p1_R(r_R - r_I) \qquad (15a)$$

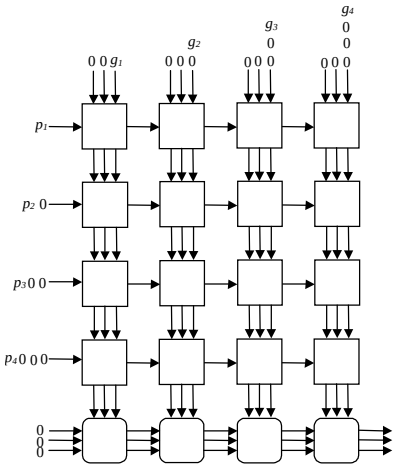
$$pr_I = (p1_R - p1_I)r_I + p1_I(r_R + r_I) \qquad (15b)$$

As can be seen from (15) and Figure 5, by utilizing the SR transformation the total number of real multiplications needed in a complex multiplication is reduced to only three. This is at the expense of having three additional adders. However, it is well known that multiplications are more complex than additions and consume much more power as well. In fact, for a single complex multiplication power reductions of up to 25% can be achieved [9]. Thus, the SR transformation can result in remarkable savings in consumed power and silicon area.

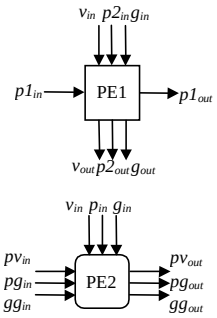
In order to calculate the throughput of the systolic array, we assume that one time step of the global clock corresponds to the processing time required for each PE. For the initialization of PE1s, $M$ time steps are needed. Thus, the total computation time required by the array is $3M$ steps. Figure 6 illustrates the flow of data in the proposed systolic architecture for different time steps.

## 5 Conclusions

In this paper, implementation of two different mobile user tracking systems were discussed. It was shown that from both performance and complexity point of views, the signal subspace method is a better choice and its implementation should be considered. Thus, for a more realistic implementation, the signal subspace based tracking system was partitioned into two parts: HW and SW. For the HW implementation of the tracking unit, a systolic architecture was proposed. With the aid of this array, the time complexity of the most computationally intensive unit was reduced to $O(M)$. Furthermore, for the implementation of the complex multipliers needed in the PEs, the SR transformation technique was utilized. As a result, remarkable savings in consumed power and silicon area were achieved. Future research should be directed towards mapping the system into a fixed number of processors when the number of antennas $M$ is large.



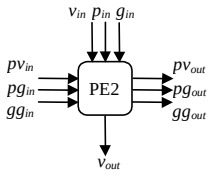
(a)



$$v_{out} \leftarrow v_{in} + p1_{in} \cdot r_{ij}$$
$$g_{out} \leftarrow g_{in}$$
$$p1_{out} \leftarrow p1_{in}$$
$$p2_{out} \leftarrow p2_{in} \quad if\ i \neq j$$
$$p2_{out} \leftarrow p1_{in} \quad if\ i = j$$
$$pv_{out} \leftarrow pv_{in} + p^*_{in} \cdot v_{in}$$
$$pg_{out} \leftarrow pg_{in} + p^*_{in} \cdot g_{in}$$
$$gg_{out} \leftarrow gg_{in} + g^*_{in} \cdot g_{in}$$

(b)

Figure 3: a) Systolic architecture for $M$=4
b) Input-output ports and cell functions



$$pv_{out} \leftarrow pv_{in} + p^*_{in} \cdot v_{in}$$
$$pg_{out} \leftarrow pg_{in} + p^*_{in} \cdot g_{in}$$
$$gg_{out} \leftarrow gg_{in} + g^*_{in} \cdot g_{in}$$
$$v_{out} \leftarrow v_{in}$$

Figure 4: I/O ports and cell functions of the modified PE2

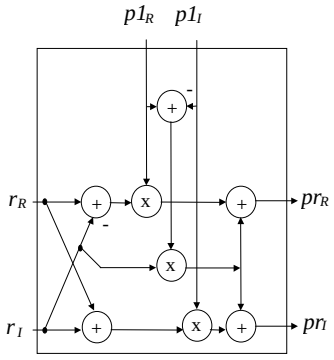


Figure 5: Implementation of a complex multiplier using the SR technique

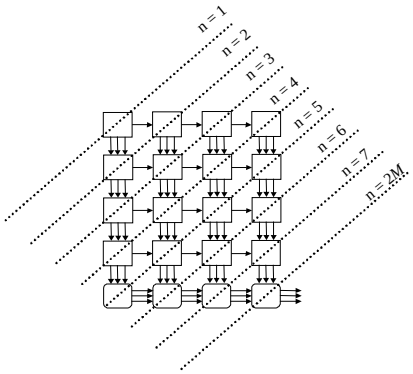


Figure 6: Illustration of the data movement for each computation step


## Acknowledgment

The authors would like to thank Mr. Kimmo Kettunen for his valuable comments. This work is part of a research project of the Institute of Radio Communication (IRC) funded by the National Technology Agency (TEKES), NOKIA Research Center, Sonera Ltd., and the Helsinki Telephone Company.


## References


[1] P. Karttunen and R. Baghaie, "Conjugate gradient based signal subspace mobile user tracking," in *Proc. IEEE Vehicular Tech. Conf.*, Houston, Texas, USA, vol. 2, pp. 1172-1176, May 1999.

[2] P. Karttunen, "An algorithm for noise subspace based mobile user tracking," in *Proc. Int. Symposium on Personal, Indoor and Mobile Radio Communications*, Osaka, Japan, September 1999.

[3] 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.

[4] R. P. Brent, "Fast multiple-precision evaluation of elementary function," *Journal of the Ass. for Comput. Machinery*, vol. 23, pp. 242-251, April 1976.

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

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

[7] 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.

[8] 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.

[9] N. Shanbhag and M. Goel, "Low-Power adaptive filter architectures and their applications to 51.84 Mb/s ATM-LAN," *IEEE Trans. on Signal Processing*, vol. 45, pp. 1276-1290, May 1997.