Adaptive beamforming using pipelined transform domain filters

GEORGE-OTHON GLENTIS

Technological Education Institute of Crete, Branch at Chania, Department of Electronics, 3, Romanou Str, Chalepa, 73133 Chania, Crete, GREECE.

Abstract: In this paper a pipelined realization of a Transform Domain adaptive beamformer, is presented. The sliding Discrete Fourier Transform is utilized for the data transformation. The adaptation process is performed by the Delayed Least Mean Squared, Generalize Sidelobe Canceller. By proper retiming of the introduced delays, fully pipelined architectures are derived, suitable for parallel implementation on a general purpose parallel machine or on dedicated VLSI hardware, using a systolic or a wavefront array processors.

Key-Words: Adaptive Antennas, Generalized Sidelobe Canceller, Transform Domain LMS, pipelined filters.

1 Introduction

Adaptive antenna arrays are important components of modern radar and telecommunication systems, [1]-[5]. An antenna array consists of a set of antenna elements that are spatially arranged at known locations. By tuning the amplitude and phase of the wavefronts at each antenna element, it is possible to electronically steer the antenna to a desired direction and to place nulls in other directions. An adaptive array continuously modify its antenna beampattern, in a desired way, by means of an adaptive optimization algorithm. The antenna beampattern is optimized so that maximum gain is offered in specific directions corresponding to the desired signal, while maximum attenuation is placed in specific directions that correspond to the undesired interference signal or jammer. Adaptive antennas can be used for high sensitivity reception, spatial filtering for interference reduction, adaptive beamforming, sidelobe cancellation, direction of arrival detection, wireless communication systems, etc.

Adaptive beamformers are tuned by adaptive optimization algorithms that operate continuously, on the basis of the incoming information (data). A broadband adaptive beamformer consists of a multi-input single output linear combiner and an adaptive algorithm that adjusts the weights is some optimal way, [1]-[8]. The constrained Least Mean Squared (C-LMS) algorithm is perhaps one of the most popular ways of adapting the antenna arrays weights. The C-LMS algorithm minimizes the noise power at the array output, in the direction of interest. An efficient implementation of the C-LMS method has been proposed by Griffiths and Jim, known as the Generalized Sidelobe Canceller (GSC), where, the constrained LMS problem is transformed to an unconstrained one, by means of a properly chosen blocking matrix, [6].

LMS like algorithms are popular due to the low computational complexity and the simplicity in the hardware realization of the underline algorithmic structure. However, the convergence rate of the LMS based Generalized Sidelobe Canceller heavily depends on the eigenvalue spread of the correlation matrix of the input data. In an attempt to improve the convergence rate of the original LMS-GSC scheme, Discrete Unitary Transforms, such as the Discrete Fourier Transform, have been utilized in order to decorrelate the input data, [7]-[8]. The Transformed Domain LMS-GSC algorithms, may have increased convergence rate for some classes on input signals, yet the computational complexity remains similar to that of the original LMS based scheme.

In many beamforming applications, very high
sample rates, and/or multi-input linear of large size, are required. The resulting computational burden of the adaptation mechanism of the GSC beamformer becomes extremely high. This means that for a real-time implementation, pipelined and/or parallel computational schemes should be developed. Pipelining and parallelism in the computation mechanism can be used to design low power implementations, which are necessary for some power constrained applications.

In this paper, an efficient pipelined architecture for the implementation of the Transform Domain LMS GSC adaptive beamformed is presented. The unitary transform utilized is the Discrete Fourier Transform (DFT) and it is implemented by means of a sliding window DFT which allows for full pipelining. An amount of time delay is subsequently introduced into the original adaptive scheme, resulting to the Delayed TD-LMS-GSC adaptive algorithm. Proper retiming of the existing delays result to a fully pipelined architecture, suitable for parallel implementation on a general purpose parallel machine or on dedicated VLSI hardware, using a systolic or a wavefront array processors, [12].

2 The TD LMS-GSC algorithm

Let us consider a linear antenna array, which consists of $K$ equally spaced antenna elements. Let $v_i(n)$, $i = 1,2 \ldots K$, be the signals obtained by passing the antenna array output through delay elements, needed to steer the array in the desired look direction. These input signals are transformed by a vector $b$ and a matrix $B$ into a main channel $x_1(n)$ and $K-1$ auxiliary channels $x_i(n)$, $i = 2, \ldots K$. Constants $b$ and $B$ are designed such that the target signal is prevented from passing through to the auxiliary channels, while it is allowed to pass unimpeded to the main channel. A possible choice is to set all elements of $b$ equal to 1, while setting $B$ to have elements form the Walsh-order Walsh function, [6].

The desired response signal, $d(n)$, is obtained by passing the primary signal $x_1(n)$ through a fixed target signal filter, that is used to control the frequency response of the beamformer in the look direction. The auxiliary signals are fed to a set of tapped delay lines, each with $L-1$ unit delay elements. The output signal $y(n)$ is obtained by applying a Transform Domain multichannel linear combiner, as

$$y(n) = C^*U(n)$$

$C$ is a vector that carries the coefficients of the multichannel linear combiner, as

$^{1}a^*$ is the complex conjugate of $a$. $^T$ denotes the vector transpose. $^H$ denotes for the Hermitian operator (conjugate and transpose).
The presence of a time delay in the error feedback loop, resulting to the Delayed TD-LMS GSC algorithm, i.e., eqs. (9)-(11), prohibits the full pipelining and/or parallelism of the algorithm. A remedy to this bottleneck is the introduction of an adaptation delay, \( \mathbf{U}_i(n) \) is the transformed data that correspond to the \( i \)-th auxiliary channel, i.e.,

\[
\mathbf{U}_i(n) = \mathbf{S}\mathbf{x}_i(n), i = 2, \ldots K
\]

(3)

\( \mathbf{x}_i(n) \) corresponds to the time-domain vector of the \( i \)-th tap delayed line, i.e.,

\[
\mathbf{x}_i(n) = [x_i(n) \ x_i(n-1) \ldots x_i(n-L+1)]^T
\]

(4)

\( \mathbf{S} \) is a unitary transformation matrix of dimensions \( L \times L \), i.e. \( \mathbf{S}^H\mathbf{S} = \mathbf{I} \). The antenna array coefficients vector \( \mathbf{C} \) is organized in a similar way.

The weight vector \( \mathbf{C} \) is adaptively estimated on the basis of the incoming data, minimizing the MSE of the error signal, between the conventional beamformer output, \( \mathbf{d}(n) \) and the output of the sidelobe canceller, \( y(n) \), i.e.

\[
e(n) = \mathbf{d}(n) - y(n)
\]

(5)

The Transform Domain LMS algorithm can be applied for the adaptive minimization of the MSE, [7], [8]. It is described by the following set of equations

\[
f_i(n) = \mathbf{S}\mathbf{x}_i(n)
\]

(6)

\[
p_i(n) = \lambda p_{i}(n) + (1-\lambda) \text{diag}([f_{1,i}(n)]^2 \ldots [f_{L,i}(n)]^2)
\]

(7)

\[
\mathbf{F}_i(n) = \mu p_i^{-1}(n-1)f_i(n)
\]

(8)

\[
y_i(n) = \mathbf{C}_i^H(n-1)f_i(n)
\]

(9)

\[
e(n) = y(n) - \sum_{i=2}^{K} y_i(n)
\]

(10)

\[
\mathbf{C}_i(n) = \mathbf{C}_i(n-1) + \mathbf{F}_i(n)e^*(n)
\]

(11)

\( \mu \) is a positive constant that controls the convergence speed of the algorithm.

The inner product computations involved in the error feedback loop of the TD-LMS GSC algorithm, i.e., eqs. (9)-(11), prohibits the full pipelining and/or parallelism of the algorithm. A remedy to this bottleneck is the introduction of an adaptation delay, [9], [11], resulting to the Delayed TD-LMS GSC scheme. The presence of a time delay in the error feedback loop, allows for the development of high throughput pipelineable and/or parallel schemes for the implementation of the D-TD-LMS algorithm on ASIC VLSI systolic or wavefront array processors. The D-TD-LMS GSC algorithm introduce a certain amount of delay in the adaptation mechanism, as

\[
\mathbf{C}_i(n) = \mathbf{C}_i(n-1) + \mathbf{F}_i(n-\rho)e^*(n-\rho)
\]

(12)

When the DFT is used as the unitary transformation, \( \mathbf{f}_i(n) \) is the sliding window Fourier transform of the input data \( \mathbf{x}_i(n) \), and each element of \( \mathbf{C}_i(n) \) is associated with a specific frequency band. The sliding window DFT algorithm can be efficiently implemented either using a sliding FFT algorithm, or a frequency-sampling filter structure, [10]. In both cases, the computational complexity is \( L \) complex multiplications per iteration period. However, the later case is more suitable for the VLSI implementation, since it has a regular structure. It is implemented using a set of first order recursive equations of the form, \( m = 0, \ldots L - 1 \)

\[
f_{i,m+1}(n) = p e^{-j\frac{2\pi m}{L}} f_{i,m+1}(n-1)+x_i(n)\rho^m x_i(n-L)
\]

\( \rho \in (0,1) \) is a stabilization factor that is used to compensate for the marginal stability of the original realization, [10].

A detailed signal flow diagram of the \( i \)-th processing column of the Delayed TD-LMS GSC adaptive beamformer is depicted in Figure 3. Six types
of processing elements are utilized, namely PE-1 up to PE-6, each performing elementary complex operations. The computational task of each processing element is described in Table 1. PE’s P-1 to P-4 involve feedforward interconnections. Thus, pipelining of these PE’s can be achieved by placing delay latches in between. On the other hand, P-5 and P-6 are connected via a long feedback loop, and as a result, some extra effort is required for the pipelining of these elements. By retiming the delays existing in the error feedback loop, three pipelined implementations of the D-TD-LMS GSC algorithm are developed.

3 Pipelined Delayed TD LMS GSC architectures

A pipelined architecture of the D-TD-LMS GSC can readily be derived by replacing the serial inner product estimation of the error signal, by a binary tree adder scheme. The presence of the adaptation delay in the error feedback loop of the original data-flow graph (Figure 3), can be used for the efficient pipelining of the binary tree adder that estimates the output signals \( y_i(n) \). Once \( y_i(n) \) have been estimated, the output signal \( y(n) \) is obtained by means of a pipelined binary tree adder. The amount of adaptation delay that is required for full pipelining of the D-TD-LMS GSC is \( D_{A1} = ([log_2(K)] + 1) + ([log_2(L)] + 1) \). The data-flow graph of the i-th column of the pipelined D-TD-LMS GSC algorithm is shown in Figure 4. Pipeline latches introduced between the stages of the binary tree adder are depicted by small black rectangular boxes. Further improvement can be achieved by pipelining PE’s P1 to P3, by introducing pipelining latches in between, and by delaying the input sequence \( y(n) \) by the same amount of time. These latches are depicted by small gray rectangular elements. The output latency of this pipelined scheme is \( D_{O1} = D_{A1} + 3 \). Notice that, PE’s P5 and P6 can work on parallel. Thus, the critical path is now determined to be \( T_C = \max\{\tau_d, \tau_m + \tau_a\} \) where, \( \tau_d \) is the time required for the division, and \( \tau_m, \tau_a \) is the time required for the multiplication and the addition operation, respectively. This pipelined scheme requires the smallest amount of adaptation delay. However, data broadcasting is required. Signals \( e(n) \) and \( u_i(n) \) should drive all PE-5’ and PE-2’, simultaneously.

An alternative pipelined architecture for the D-TD-LMS GSC is possible, that avoids the use of a binary tree adder, allowing, thus, for a systolic implementation. The data-flow graph that is depicted in Figure 3 consists of \( L \) identical columns of PE’s. The adaptation delay units that appears in the error feedback loop are retimed by proper vertical cut sets. \( D_{A2} = L-1+([log_2(K)]+1) \) such delays are required for the full pipelining of the D-TD-LMS GSC algorithm. The second term in the previous expression is due to the fact that the output signal \( y(n) \) is obtained from the partial estimates \( y_i(n) \), by means of a pipelined binary tree adder. The resulting architecture is shown in Figure 5. The critical path in this case is easily shown to be equal to the former architecture, i.e., \( T_C = \max\{\tau_d, \tau_m + \tau_a\}. \) The out-

<table>
<thead>
<tr>
<th>P.E.</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>P-1</td>
<td>( u(n) = x(n) - p^*x(n-L) )</td>
</tr>
<tr>
<td>P-2</td>
<td>( f(n) = pe^{-j\frac{2\pi m}{n}} f(n-1) + u(n) )</td>
</tr>
<tr>
<td>P-3</td>
<td>( p(n) = \lambda p(n-1) + (1 - \lambda)</td>
</tr>
<tr>
<td>P-4</td>
<td>( F(n) = \mu f(n)/p(n) )</td>
</tr>
<tr>
<td>P-5</td>
<td>( C(n) = C(n-1) + F(n)e^*(n) )</td>
</tr>
<tr>
<td>P-6</td>
<td>( e_i(n) = e_{i-1}(n) - C^*(n-1)f(n) )</td>
</tr>
</tbody>
</table>

Figure 4: A pipelined architecture for the D-TD-LMS GSC using a binary tree adder.

Table 1: Computational task of the Processing Elements
put latency is $D_{O2} = D_{A2} + 3$. The fully pipelined D-TD-LMS GSC has a modular structure and requires local data communication. It can be easily transformed into a locally recursive algorithm using the canonical mapping methodology and hence into an efficient VLSI array processor implementation either in systolic or in wavefront architecture [12]. However, it requires the maximum amount of adaptation delay.

In both cases, some extra time delays should be introduced to pipeline the preprocessing part of the GSC, which contributes to an extra amount of $\log_2(K)$ time delays, provided that pipelined binary tree adders have been utilized.

4 Simulation

The performance of the proposed Delayed TD LMS GSC algorithm is illustrated by a typical computer simulation. The simulation scenario is similar to that of [9]. A stationary narrowband target signal is mitigated by three stationary narrowband jammers with different directions of arrival. The background noise is a zero mean Gaussian white noise signal. Specific values of the simulation parameters are given below

<table>
<thead>
<tr>
<th>Signal</th>
<th>f</th>
<th>$\theta$</th>
<th>SNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>0.1</td>
<td>0°</td>
<td>10 db</td>
</tr>
<tr>
<td>Jammer 1</td>
<td>0.3</td>
<td>34°</td>
<td>20 db</td>
</tr>
<tr>
<td>Jammer 2</td>
<td>0.4</td>
<td>−49°</td>
<td>40 db</td>
</tr>
<tr>
<td>Jammer 3</td>
<td>0.25</td>
<td>−24°</td>
<td>30 db</td>
</tr>
</tbody>
</table>

Parameters $f$ and $\theta$ denote the normalized frequency and the incident angle (relative to the broadside) of the plane wave signals, respectively. The adaptive beamformer consists of $K = 17$ linear array elements, equally spaced at half wavelength distance at the maximum frequency of interest, $f_{max}$, and it is steered in the direction of the target signal. $L = 8$ delay elements are associated to each array elements.

The learning curve of the standard Griffiths-Jim GSC LMS algorithm and the proposed Delayed TD-LMS GSC method is depicted in Figure 5. It is evidence that the proposed method outperforms the standard approach. The beampatterns of the proposed method (after convergence), at different frequencies of interest, are depicted in Figure 6.

5 Conclusions

In this paper a pipelined realization of a Transform Domain adaptive beamformer, has been presented. The sliding Discrete Fourier Transform is utilized for the data transformation. The adaptation process is performed by the Delayed Least Mean Squared, Generalize Sidelobe Canceller.

The pipelined operation of the algorithm have
been achieved introducing a proper amount of adaptation delay to the original algorithm, resulting to a Delayed TD-LMS GSC scheme. By retiming the adaptation delay that has been introduced in the error feedback loop, three pipelined architectures have been proposed that allows for full pipelining of the algorithm. The proposed architectures are suitable for parallel implementation on a general purpose parallel machine or on dedicated hardware, integrated on ASIC or ASIP VLSI processors.

References:


