Low Power Implementations for Adaptive Filters

Marius Vollmer Stephan Klauke Jürgen Götze

University of Dortmund, Information Processing Lab,
http://www-dt.e-technik.uni-dortmund.de,
{marius.vollmer, stephan.klauke, juergen.goetze}@udo.edu

Camera-ready Copy for

Kleinheubacher Berichte

Manuscript-No. (none)

Low Power Implementations for Adaptive Filters

Marius Vollmer Stephan Klauke Jürgen Götze

University of Dortmund, Information Processing Lab,
http://www-dt.e-technik.uni-dortmund.de,
{marius.vollmer, stephan.klauke, juergen.goetze}@udo.edu

Abstract. In this text we present an approach for reducing switching activity in adaptive filters. We show how to introduce substantial approximations into the basic QR algorithm that underlies many applications of adaptive filters, such as beamforming, echo cancellation, channel equalization, or system identification. These approximations work within the CORDIC devices that form the basic elements of the array formulation of an updating QR algorithm. They directly lead to a reduced hardware demand of the filter structure. We demonstrate the feasibility of this approach by presenting simulation results of an adaptive channel equalizer.

1 Introduction

In recent years, power consumption has become a critical design concern for ICs/ASICs. This is not only a result of the preceeding success of portable consumer electronics but also due to the increasing costs for packaging and cooling of ICs with high power consumption. In contrast, the main design constraint in many applications (e. g. mobile communication) is given by the real-time requirements of the implemented signal processing tasks. Concurrent processing techniques had to be introduced early to gain the desired data rates needed for this kind of applications.

About two decades ago, systolic architectures were first introduced as an efficient way for the implementation of many algorithms in different fields of applications. Systolic arrays show a high degree of concurrency, they have a regular and modular structure and thus a regular and local interconnection scheme. These properties make them an ideal solution for applications with high throughput and large processing bandwidth (Fortes and Wah, 1987). As the systolic architecture itself is highly optimized in terms of parallelism and speed, the common architecture level low power strategy “trading area/speed for power” (Mehra et al., 1996; Chandrakasan and Brodersen, 1995; Landman, 1994) fails to gain a significant power reduction.

1.1 Power reduction at the algorithmic level

Since it is well known that significant power reductions can be obtained on the algorithmic level (Liu et al., 1998; Mehra et al., 1996; Luschi et al., 2000), we investigate possible power savings in parallel algorithms by introducing approximations at a level that is above that of individual gates, adders, or even multipliers. It should be noted that the regularity of the implementation and the data flow is not changed by our modifications so that the existing experience in designing parallel structures can still be applied.

In the case of adaptive filters, the underlying algorithm is often the updating QR algorithm, which in turn can be implemented with a parallel processor array. Each processor in this array performs an orthogonal rotation which represents a special kind of matrix transformation. This special matrix transformation can be implemented by a CORDIC device and it is the matrix transformation as a whole that is approximated so that a more efficient implementation of the corresponding hardware unit results.

2 The Updating QR Decomposition

A well known method for solving a optimization problem that employs a least squares criterion is the *QR decomposition* of the defining matrix.

In the linear model

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{1}$$

the matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ and the vector $\boldsymbol{b} \in \mathbb{R}^n$ are known and describe the problem. The vector $\boldsymbol{x} \in \mathbb{R}^m$ is unknown and the goal is to determine an estimation $\hat{\boldsymbol{x}}$ of $\boldsymbol{x}$ that minimizes the *least square error*

$$e = \|\boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{b}\|^2. \tag{2}$$

For channel equalization, $\boldsymbol{A}$ would contain the channel estimates in a suitable arrangement, $\boldsymbol{b}$ would contain the received signal, and $\hat{\boldsymbol{x}}$ is the estimate of the transmitted signal. For system identification (for example channel estimation), $\boldsymbol{A}$ would contain the stimulus signal in a suitable arrangement (the training sequence), $\boldsymbol{b}$ the output of the unknown system (the distorted training sequence), and $\hat{\boldsymbol{x}}$ would be the estimates of the system parameters (channel tap coefficients).

The solution $\hat{\boldsymbol{x}}$ that minimizes e from (2) can be computed from the QR decomposition of $\boldsymbol{A}$: Given

$$\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}, \tag{3}$$

where $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ is an orthogonal matrix, i.e. $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$, and $\boldsymbol{R} \in \mathbb{R}^{n\times m}$ is upper triangular, it can be shown that $\hat{\boldsymbol{x}}$ can be determined by solving

$$\boldsymbol{R}\hat{\boldsymbol{x}} = \boldsymbol{y} + \boldsymbol{e} \quad \text{with} \quad \boldsymbol{y} = \boldsymbol{Q}^T\boldsymbol{b}, \quad e_i = 0 \text{ for } i \le m \tag{4}$$

for $\hat{\boldsymbol{x}}$ and $\boldsymbol{e}$ via backsubtitution on $\boldsymbol{R}$. The elements of $\boldsymbol{R}$ that are known to be zero do not need to be stored and considered in this process, of course.

The way the QR decomposition is actually computed is to apply a sequence of *elementary orthogonal rotations* to $\boldsymbol{A}$ that each introduce one zero into the transformed matrix until $\boldsymbol{R}$ results (Golub and van Loan, 1996). When these transformations are also applied to $\boldsymbol{b}$ in addition to $\boldsymbol{A}$, the vector will be transformed into $\boldsymbol{Q}^T\boldsymbol{b} = \boldsymbol{y}$, just as $\boldsymbol{A}$ is transformed into $\boldsymbol{Q}^T\boldsymbol{A} = \boldsymbol{R}$. It is therefore possible to compute the desired right hand-side $\boldsymbol{y}$ at the same time as computing $\boldsymbol{R}$. Thus, the matrix $\boldsymbol{Q}$ does not need to be computed or stored explicitely.

In an adaptive system, it is often desirable to compute an *update* to an existing QR decomposition when new data arrives (Haykin, 1996). Usually, the new data yields a new row $\boldsymbol{a}^T$ and a new element c that are appended to the previous data to form

$$\boldsymbol{A}' = \begin{bmatrix}\boldsymbol{A}\\ \boldsymbol{a}^T\end{bmatrix} \in \mathbb{R}^{(n+1)\times m}, \quad \boldsymbol{b}' = \begin{bmatrix}\boldsymbol{b}\\ c\end{bmatrix} \in \mathbb{R}^{(n+1)}. \tag{5}$$

It is easy to see that the new QR decomposition $\boldsymbol{A}' = \boldsymbol{Q}'\boldsymbol{R}'$ can be computed as

$$\boldsymbol{Q}' = \begin{bmatrix}\boldsymbol{Q} & \\ & 1\end{bmatrix}\boldsymbol{\Theta}, \quad \boldsymbol{R}' = \boldsymbol{\Theta}^T\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{a}^T\end{bmatrix}, \tag{6}$$

where $\boldsymbol{\Theta} \in \mathbb{R}^{(n+1)\times(n+1)}$ is the orthogonal matrix defined by the QR decomposition

$$\begin{bmatrix}\boldsymbol{R}\\ \boldsymbol{a}^T\end{bmatrix} = \boldsymbol{\Theta}\boldsymbol{R}'. \tag{7}$$

Additionally, the right hand-side is updated according to

$$\boldsymbol{y}' = \boldsymbol{\Theta}^T\begin{bmatrix}\boldsymbol{y}\\ c\end{bmatrix}. \tag{8}$$

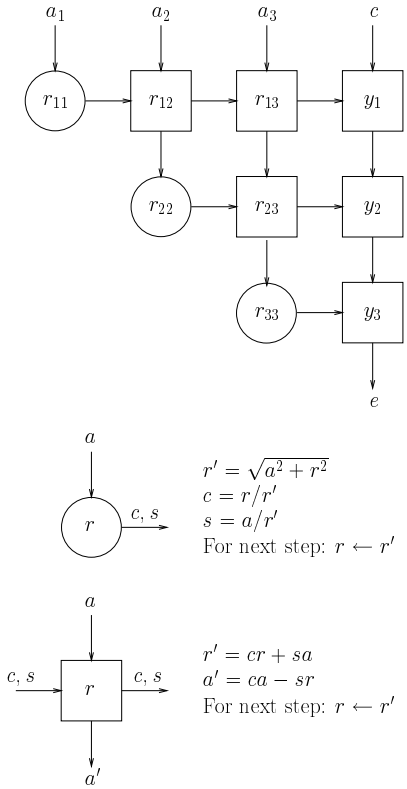

Figure 1. Hardware structure for updating QR decomposition.

Stated differently, the task of updating $\boldsymbol{R}$ to $\boldsymbol{R}'$ consists of tacking $\boldsymbol{a}^T$ onto the bottom of $\boldsymbol{R}$ and finding the sequence of elementary orthogonal rotations that annihilates these new elements. Although conceptually the triangular matrix of the QR decomposition grows in each updating step, the number and positions of the elements that are not know to be zero remain constant so that no additional resources are consumed for each step. The right hand-side *does* grow from step to step, however, and the new element must be treated appropriately. In some applications, the new element characterizes

the error of the LS approximation; in others, it characterizes the new element of the desired output signal. In any case, it needs to be removed in order to keep the updating process run within constant resources.

Figure 1 shows a data flow structure that can compute one updating step. When the triangular array has been filled with the non-zero elements of $\boldsymbol{R}$ and the interesting elements of $\boldsymbol{y}$, it will compute the corresponding elements of $\boldsymbol{R}'$ and $\boldsymbol{y}'$ when the elements of $\boldsymbol{a}^T$ are applied to its inputs. The output e yields the new element of the right-hand side.

The operations performed by the circular and square cells are also detailed in figure 1. In essence, the circular cells compute the orthogonal transformation needed to annihilate the element of $\boldsymbol{a}$ from their column; the square cells to the right of them apply this transformation to the rest of the row.

3 Approximate CORDIC

While the operations listed in figure 1 look daunting at first, they can be efficiently implemented in hardware by using *CORDIC* devices (Volder, 1959; Hu, 1992). The idea is to compose a complete elementary rotation from several *micro rotations* that are easy to implement.

An elementary rotation can be expressed as a 2 by 2 matrix

$$\boldsymbol{G} = \begin{bmatrix} \cos(\phi) & \sin(\phi) \\ -\sin(\phi) & \cos(\phi) \end{bmatrix} = \begin{bmatrix} c & s \\ -s & c \end{bmatrix} \tag{9}$$

where ϕ is the rotation angle (clockwise). The circular cells in figures 1 need to find the parameters c and s of $\boldsymbol{G}$ such that a vanishes, while the square cells will simply compute a matrix vector product with $\boldsymbol{G}$.

To facilitate implementing these two tasks in hardware, the complete $\boldsymbol{G}$ is approximated by a sequence of micro rotations of the form

$$\boldsymbol{M}_i = K_i \begin{bmatrix} 1 & \mu_i 2^{-i} \\ -\mu_i 2^{-i} & 1 \end{bmatrix}, \quad \mu_i = \pm 1, \quad K_i = (1 + 2^{-2i})^{-\frac{1}{2}}. \tag{10}$$

The parameter μ_i determines the direction of rotation and i determines the angle. The hardware operations corresponding to this matrix (except for the multiplication by K_i) are additions and shifts. Figure 2 shows a schematic for implementing a micro-rotation. The scaling factor K_i is independent of μ_i and can be accumulated over the sequence of micro rotations.

Let the exact rotation be approximated by N micro-rotations:

$$\boldsymbol{G} \approx \prod_{i=0}^{N-1} \boldsymbol{M}_i, \qquad K = \prod_{i=0}^{N-1} K_i. \tag{11}$$

For any fixed N, the cumulative scaling factor K is a constant and can be implemented by a hardwired multiplier, for example.

The parameter N then is an indicator of the amount of hardware resources required to implement the CORDIC device, and of the accuracy attained. The goal is to keep N as low as possible while still achieving useful results from the adaptive filtering process. It should be noted that varying N is different from varying the word length of the number representation, for example.

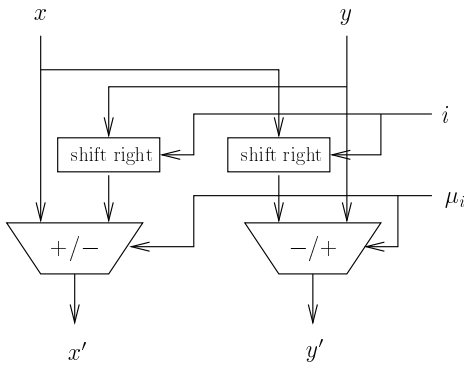

Figure 2. Hardware for computing a unscaled micro-rotation.

4 Simulation Results

To verify the effectiveness of the described approximation method, simulations of an adaptive filter for channel equalization were performed. Figure 3 depicts the simulation setup. The data source produces a stream of symbols from the set $\{-1, 1\}$. These symbols are distorted by a channel that is modelled as a three tap FIR filter with coefficients

$$h_i = \frac{1}{2} + \frac{1}{2}\cos\Big(\frac{2\pi}{W}(i-2)\Big), \quad i = 1, 2, 3. \tag{12}$$

Here, W determines the amount of distortion introduced by the channel. After the channel, white noise is added to the signal to

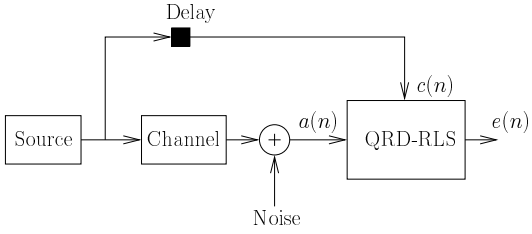

Figure 3. Simulation setup.

form the input $a(n)$ to the adaptive filter block. The source signal is delayed to match the delay introduced by the channel and the adaptive filter itself, and fed into the filter as the reference signal $c(n)$.

For every time step, the adaptive filter computes one QR updating process. The row $\boldsymbol{a}^T$ is formed from the new input sample $a(n)$ and from $M-1$ previous samples to be

$$\boldsymbol{a}^T = \begin{bmatrix} a(n) & a(n-1) & \cdots & a(n-M+1) \end{bmatrix}. \tag{13}$$

The parameter M determines the number of taps of the adaptive filter. The new element for the right hand side is $c = c(n)$. The output of the QR array in figure 1 yields the error for this time step: $e(n) = e$. This error indicates how well the filter has adapted itself to the channel distortions.

Figure 4 shows actual simulation results for a SNR of 10dB, the parameters $W = 2.9$, and $M = 11$ and four values for N, the number of micro-rotations.

As can be expected, the residual error floor improves with a growing number of micro-rotations, N. Already $N = 8$ micro-rotations match the performance of the QRD-RLS algorithm with exact rotations. Here, "exact" refers to an implementation with IEEE double precision floating point numbers. For the CORDIC algorithm, more than 50 micro-rotations would be required to achieve the numerical accuracy of double precision floating point operations. Hence, a reduction to $N = 6$ or $N = 8$ represents a significant win.

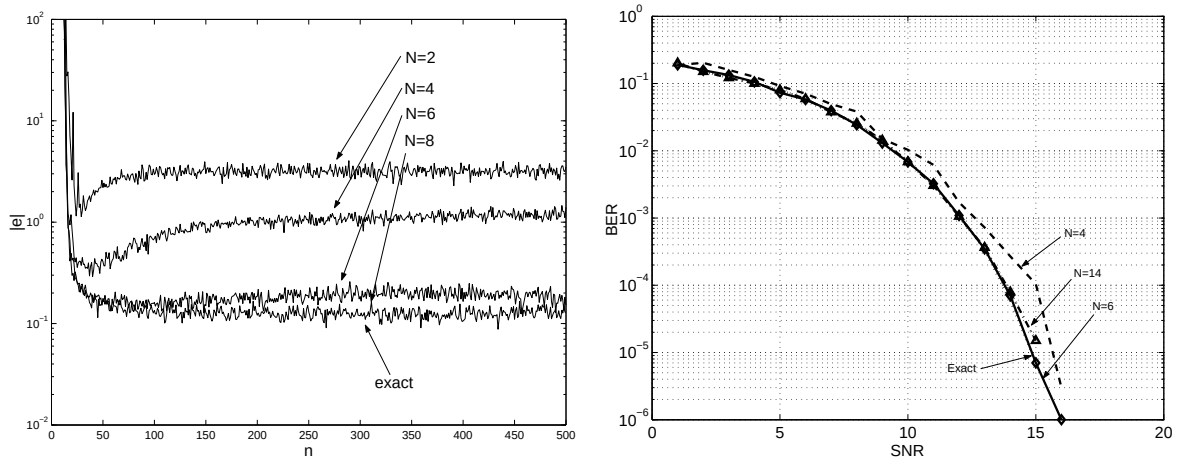

Figure 4. Simulation results: Convergence for exact rotations, $N = 8$, $N = 6$, $N = 4$, and $N = 2$; Bit error ratio for exact rotations, $N = 4$, $N = 6$, and $N = 14$.

Figure 4 also presents bit error ratios that have been obtained by letting the equalization filter adapt for $n = 500$ symbols and then using the resulting FIR coefficients to filter the remaining symbols until either 1 million symbols have been received or 1000 errors have been detected.

It can be seen that beyond $N = 6$ no improvements are visible in terms of bit error ratio. However, no significant variations in the dependencies upon the SNR can be observed. Thus, a dynamic selection of the parameter N dependend on the noise level does not seem promising. However, it is also possible to perform the adaptation iteratively, as explained in (Götze, 1994).

References

Chandrakasan, A. and Brodersen, R., Minimizing Power Consumption in Digital CMOS Circuits, *Proceedings of the IEEE*, *83*, 1995.

Fortes, J. and Wah, B., Special issue: Systolic arrays – from concept to implementation, *Computer*, *20*, 12–17, 1987.

Golub, G. and van Loan, C., *Matrix Computations*, The John Hopkins University Press, third edn., 1996.

Götze, J., An Iterative Version of the QRD for Adaptive RLS Filtering, in *SPIE Conference on "Advanced Signal Processing: Algorithms, Architectures and Implementations"*, pp. 438–450, San Diego, USA, 1994.

Haykin, S., *Adaptive Filter Theory*, Prentice Hall, third edn., 1996.

Hu, Y. H., CORDIC-Based VLSI Architectures for Digital Signal Processing, *IEEE Signal Processing Magazine*, 1992.

Landman, P. E., *Low-Power Architectural Design Methodologies*, Ph.D. thesis, University of California at Berkeley, 1994.

Liu, K., Wu, A.-Y., Raghupathy, A., and Chen, J., Algorithm-based low-power and high-performance multimedia signal processing., in *Proceedings of the IEEE*, vol. 86, pp. 1155–1202, 1998.

Luschi, C., Sandell, M., Strauch, P., Wu, J.-J., Ila, C., Ong, P.-W., Baeriswyk, R., Battaglia, F., Karageorgis, S., and Yan, R.-H., Advanced Signal-Processing Algorithms for Energy-Efficient Wireless Communications, in *Proc. IEEE*, vol. 88, pp. 1633–1650, 2000.

Mehra, R., Lidsky, D., Abnous, A., Landman, P., and Rabaey, J., Algorithm and architectural level methodologies for low power, *Low Power Design Methodologies*, *Kluwer Academic Publishers*, 1996.

Volder, J. E., The CORDIC Trigonmetric Computing Technique, *IRE Transactions on Electronic Computers*, *EC*, 330–334, 1959.