HARDWARE IMPLEMENTATION OF A SYSTOLIC ANTENNA ARRAY SIGNAL PROCESSOR BASED ON CORDIC ARITHMETIC

B. Haller, M. Streiff, U. Fleisch, and R. Zimmermann

Department of Electrical Engineering
Swiss Federal Institute of Technology (ETH)
CH–8092 Zurich, Switzerland

ABSTRACT

In this paper we present the practical hardware implementation of a systolic array for performing recursive least-squares minimisation via orthogonal matrix triangularisation. This is an extremely demanding task for high speed, real-time operation such as required in many modern adaptive antenna, radar, and sonar systems. Since the underlying Givens rotations can be efficiently computed by the CORDIC algorithm, we have implemented a dedicated CORDIC processor element (CPE) in an ASIC. All the required calculations are carried out by a network of these small and simple circuits, which are suitable for constructing a high performance systolic array, either based on MCM technology or as a macro-cell building block for a very highly integrated single chip solution. The design of an adaptive antenna signal processor is described in a top-down manner, from the proposed algorithm down to the bit-level details of the realised component.

1. INTRODUCTION

The performance of radio communication, radar, and sonar systems can be greatly improved by exploiting *spatial filtering* techniques such as *digital adaptive beamforming*. Their application has been a topic of intense military research in the past, but meanwhile has also become an area of increased interest in the field of civil mobile radio. Especially the novel concept of *space division multiple access* (SDMA) employing base-station multi-beam antenna arrays is attracting much attention lately and promises to increase the capacity of future cellular systems considerably. It is within this general context that the following work has been performed.

Adaptive arrays are essentially "smart" antenna systems which can automatically adjust their directional response, i.e. their beam pattern, to reduce interference and thus enhance the reception of wanted signals. The objective of the adaptive antenna array signal processing considered here is to minimise the total power at the output of the processor subject to certain constraints which prevent the signals of interest from being cancelled. This is achieved by selecting a set of complex (amplitude and phase) weights with which the outputs of the individual antenna elements are combined.[1] To obtain rapid weight adaptation in time-varying situations, open loop techniques are often employed. Finding the coefficients of the adaptive combiner is performed through least-squares minimisation. The corresponding set of equations can be solved using matrix QR decomposition (QRD) based on Givens rotations. An efficient parallel and pipelined systolic array architecture for carrying out the QRD in real-time was proposed by Gentleman and Kung [1]. A modified algorithm published by McWhirter [2] enables the direct extraction of the desired signal from the output of the systolic array without the need to derive the weight vector explicitly. This has several advantages in terms of hardware complexity, throughput, and numerical robustness. We have adopted the approach reported in [3] for the implementation of the presented *antenna signal processor*.

Systolic architectures have many attractive conceptual features, including the extensive use of parallel and pipelined computing as well as a regular and modular structure requiring only local communication and little control, which take maximum advantage of very large scale integration (VLSI) technology [4]. Despite these merits, their practical realisation as special-purpose processors has been hindered in the past due to a lack of suitable building blocks in the form of silicon macro-cells. The design complexity of systolic arrays can be reduced significantly by repeated use of a single specifically tailored hardware cell, providing the required functionality with a minimum of control signals and interconnection. Since many algorithms in digital signal processing (DSP) and numerical linear algebra may be cast into a framework mainly involving rotations, CORDIC (COordinate Rotation DIgital Computer) [5] processors are ideally suited as basic arithmetic units [6]. In this paper we describe a specialised *CORDIC processor element* (CPE) for rapid computation of plane rotations. Specifically, it is capable of calculating all the elementary functions required for the execution of Givens rotations. The CPE can thus support several common systolic algorithms for performing the QRD, EVD, or SVD, which are very widely employed in antenna array signal processing.

2. ADAPTIVE BEAMFORMING ALGORITHM

There exist a wide range of adaptive antenna applications each needing a slightly different hardware configuration, depending on the mechanism employed to prevent the pro-

Supported in part by the Swiss National Science Foundation (SNF project no. 21-40760). ASIC manufacturing funded by ETHZ.

[1] We restrict our attention to *narrowband* adaptive arrays.

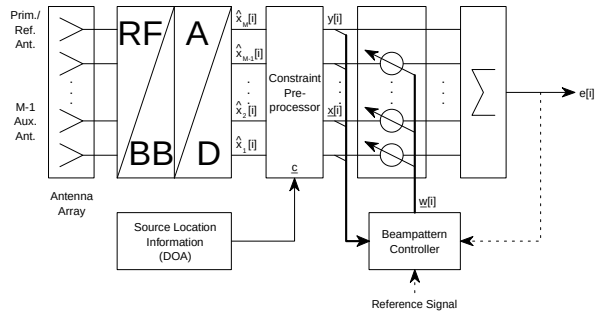

Figure 1: Generic block diagram of an adaptive antenna.

cessor from cancelling the wanted signals. This is usually achieved through some form of preprocessing which will not be discussed here due to space limitations. Rather, we will concentrate on the "heart" of the antenna signal processor that we are considering, namely the *multi-channel recursive least-squares (RLS) filter* which computes the output residual signal.

A fairly generic block diagram of an adaptive antenna is depicted in **Fig. 1**. Let us assume a configuration where one antenna in the array (= "primary" element) has a fixed weight of unity. In this case, the vector of output residuals $\mathbf{e}[i]$ from the processor can be written as:

$$\mathbf{e}[i] = \mathbf{X}[i]\mathbf{w}[i] + \mathbf{y}[i], \tag{1}$$

with the data matrix

$$\mathbf{X}[i] \triangleq \begin{pmatrix} \mathbf{x}^{\mathrm{T}}[1] \\ \vdots \\ \mathbf{x}^{\mathrm{T}}[i] \end{pmatrix} \in \mathbb{C}^{i\times(M-1)},$$

where $\mathbf{x}[i]$ is the present vector of $M-1$ auxiliary signal samples, $\mathbf{y}[i]$ stands for the vector of samples taken from the primary channel and $\mathbf{w}[i]$ represents the weight vector at the current sample time i. An estimate of the total output "power" after i samples is given by

$$E^2[i] = \mathbf{e}^{\mathrm{H}}[i]\mathbf{e}[i]. \tag{2}$$

A numerically robust technique to determine the weight vector $\mathbf{w}[i]$ which minimises $E[i] = \|\mathbf{e}[i]\|$ is by the well-known data domain method of *orthogonal triangularisation via matrix QR decomposition* (QRD) [7]. Using this algorithm a unitary matrix transformation $\mathbf{Q}[i]$ is applied to the observed data matrix $\mathbf{X}[i]$ in order to produce an upper triangular matrix $\mathbf{R}[i] \in \mathbb{C}^{(M-1)\times(M-1)}$:

$$\mathbf{Q}[i]\mathbf{e}[i] = \begin{pmatrix} \mathbf{R}[i] \\ \mathbf{0} \end{pmatrix} \mathbf{w}[i] + \begin{pmatrix} \mathbf{u}[i] \\ \mathbf{v}[i] \end{pmatrix}. \tag{3}$$

As can be seen from this equation, the minimum norm condition for the error residual $\mathbf{e}[i]$ is obtained when

$$\mathbf{R}[i]\mathbf{w}[i] + \mathbf{u}[i] = \underline{0}. \tag{4}$$

This equation defines the least-squares weight solution for the adaptive array. Since the matrix $\mathbf{R}[i]$ is upper triangular, the weight vector $\mathbf{w}[i]$ may be derived very simply by a process of back-substitution.

The triangular system of equations can be updated on a sample by sample basis according to the algorithm summarised in **Tab. 1**. The unitary update transformation

Initialisation:
$\mathbf{R}[0] = \mathbf{0} \in \mathbb{R}^{(M-1)\times(M-1)}, \quad \mathbf{u}[0] = \underline{0} \in \mathbb{R}^{M-1}$
for $i = 1, 2, \ldots$ do:
$$\hat{\mathbf{Q}}[i] \begin{pmatrix} \mathbf{R}[i-1] & \mathbf{u}[i-1] \\ \mathbf{x}^{\mathrm{T}}[i] & y[i] \end{pmatrix} = \begin{pmatrix} \mathbf{R}[i] & \mathbf{u}[i] \\ \underline{0}^{\mathrm{T}} & \tilde{e}[i] \end{pmatrix}$$
where $\hat{\mathbf{Q}}[i] = \hat{\mathbf{Q}}_{M-1}[i] \cdots \hat{\mathbf{Q}}_1[i]$
$e[i] = \tilde{e}[i] \cdot \tilde{\gamma}[i]$ with $\tilde{\gamma} = \prod_{m=1}^{M-1} \cos\theta_m[i]$

Table 1: QRD-RLS algorithm based on Givens rotations.

$\hat{\mathbf{Q}}[i]$ represents a sequence of $M-1$ *Givens rotations*, employed such that the elements of the new data "snap shot" $(\mathbf{x}^{\mathrm{T}}[i] \;\; y[i])^{\mathrm{T}}$ are eliminated, one after another, thereby reducing the system to triangular form again. Each Givens rotation operates on two rows of the matrix at a time as follows:

$$\begin{pmatrix} \cos\theta_m & \sin\theta_m \\ -\sin\theta_m & \cos\theta_m \end{pmatrix} \cdot \begin{pmatrix} \ldots & 0 & r_m & r_{m+1} & \ldots \\ \ldots & 0 & x_m & x_{m+1} & \ldots \end{pmatrix} = \begin{pmatrix} \ldots & 0 & r'_m & r'_{m+1} & \ldots \\ \ldots & 0 & 0 & x'_{m+1} & \ldots \end{pmatrix}, \tag{5}$$

where the rotation angle θ_m is chosen to cancel x_m. Repeating this process $M-1$ times zeros out all but the last element $\tilde{e}[i]$ in the bottom row of the matrix whilst updating $\mathbf{R}[i]$ and $\mathbf{u}[i]$. Obviously, the desired least-squares residual is the product of the "rotated residual" $\tilde{e}[i]$ and the series of cosine terms $\cos\theta_m$. It is calculated without explicitly computing the weight vector and applying the result to Eq. (1).

3. HARDWARE IMPLEMENTATION

McWhirter [2] has shown how the Givens rotation based algorithm described above may be implemented in a very efficient pipelined manner using a *triangular systolic array*. In what follows we present the design of such an array comprising a single type of cell, namely a *CORDIC processor element* (CPE) specifically tailored to algorithms requiring plane rotations as a basic operation, and describe the ASIC implementation of this useful component.

Systolic Array Architecture

The implementation of a systolic adaptive beamformer architecture is illustrated schematically in **Fig. 2** for the case of $M = 4$ antenna elements.[2] The cells in the basic triangular array (A-B-C) store the elements of the evolving triangular matrix $\mathbf{R}[i]$ and the ones in the right hand column (D-E) store the elements of the updated vector $\mathbf{u}[i]$. The data flow is from the top to the bottom, whilst the rotation angles θ are propagated from the left to the right of the array. In the original array [2,3], three different cell types are defined to perform the necessary calculations: (1) boundary cells to compute the rotation parameters as well

[2]For ease of presentation, we assume real input signals here. The extension to complex arithmetic is quite straightforward.

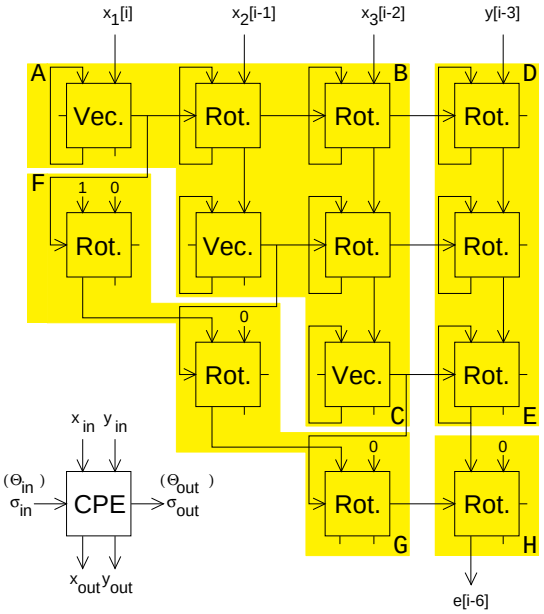

Figure 2: Systolic CPE array.

as to successively form the product of the cosine terms, (2) internal cells to carry out the rotations and (3) a final multiplier which generates the desired output residual. If these computations are done by MAC- ("multiply and accumulate") type arithmetic units, operations such as square-root, division or some trigonometric functions will most probably be required; the main disadvantage being that the determination of the rotation parameters takes many more clock/instruction cycles than it does to apply them. Most likely due to this property an asynchronous technique based on a wavefront array was chosen to build such a system in the past [8]. Our implementation of the array entirely consists of CORDIC processor elements (CPEs) which work completely synchronously driven by a single global master clock. Since all the CPEs need the same amount of time to perform their computations they can never be flooded with data. Thereby, the CPEs designated with Vec. are configured to the "vectoring" mode of operation and those labelled with Rot. operate in the "rotation" mode. Each row performs a Givens rotation, whereby the rotation angle is determined by the CPE in vectoring mode at the beginning of the row. The rotation angle is passed to the rotation CPEs to the right with one clock cycle delay, thus requiring the elements of the data vector to be applied to the array in a time staggered fashion as indicated by the indices in Fig. 2. The product of the cosine terms is successively computed by the diagonal CPEs (F–G) which receive the argument of the cosine function from the above vectoring CPEs. These cell pairs are equivalent to the boundary cells of the original array [2]. The final multiplication is also performed by a CPE (H) via a rotation with appropriate inputs.

CORDIC Processor Element

The CORDIC algorithm [5] is an iterative technique to perform two-dimensional vector rotations using only simple hardware components, essentially adders/subtractors and shifters. The basic idea behind the CORDIC scheme is to carry out vector ("macro"-)rotations by an arbitrary rotation angle θ as a series of "micro-rotations" using a fixed set of elementary rotation angles α_j

$$\theta = \sum_{j=0}^{N-1} \sigma_j \alpha_j, \quad \sigma_j \in \{-1, +1\}, \tag{6}$$

to successively approximate the final result. The CORDIC has two modes of operation called "vectoring", to compute the magnitude and phase of a vector

$$\begin{pmatrix} x_{\text{out}} \\ y_{\text{out}} \end{pmatrix} = \begin{pmatrix} \text{sgn}(x_{\text{in}}) \cdot \sqrt{x_{\text{in}}^2 + y_{\text{in}}^2} \\ 0 \end{pmatrix}, \tag{7}$$

$$\theta_{\text{out}} = -\arctan\left(\frac{y_{\text{in}}}{x_{\text{in}}}\right), \tag{8}$$

whereby the vector $(x_{\text{in}} \;\; y_{\text{in}})^{\text{T}}$ is rotated to the x-axis, and "rotation"

$$\begin{pmatrix} x_{\text{out}} \\ y_{\text{out}} \end{pmatrix} = \begin{pmatrix} \cos\theta_{\text{in}} & -\sin\theta_{\text{in}} \\ \sin\theta_{\text{in}} & \cos\theta_{\text{in}} \end{pmatrix} \cdot \begin{pmatrix} x_{\text{in}} \\ y_{\text{in}} \end{pmatrix}, \tag{9}$$

$$\theta_{\text{out}} = \theta_{\text{in}}, \tag{10}$$

in which case the vector $(x_{\text{in}} \;\; y_{\text{in}})^{\text{T}}$ is rotated by the angle θ_{in}.

The required computations, i.e. the CORDIC equations, may be implemented in a number of ways, such as bit-serial or word-parallel with the iterations being carried out in either a serial or parallel manner [9]. By choosing a recursive, pipelined, or array architecture a trade-off between area, throughput, and latency is possible, depending on the application in mind. Due to area limitations a recursive, word-parallel scheme using fixed-point arithmetic with 16 bit data words (22 bit internal precision) was implemented as an application-specific integrated circuit (ASIC). The main design target was to come up with a small circuit in terms of silicon area, thus making it feasible to integrate several CPEs on a single chip in a future project. In order to kept the throughput as high and latency as low as possible, the shift sequence proposed by Despain [10] resulting in the least number of CORDIC iterations for 16 bit precision was chosen. The implemented CORDIC algorithm is summarised in **Tab. 2**. Hereby, the rotation angles θ are represented in terms of the coefficients σ_j. Since these coefficients are passed on from one CPE to the next in the same row of the array as a continuous serial bit stream, interconnection is minimised. To achieve the greatest possible flexibility for a user friendly configuration of the array, CPE I/O has been realised by means of busses extending along the rows of the array. As mentioned earlier the elements of the current snap shot are applied to the array in a time staggered manner. Because an angle coefficient is computed in every clock cycle, each CPE operates with one cycle delay with respect to its predecessor. This allows the bus to be shared in a time-multiplex fashion delivering data to consecutive CPEs of each row during a specific CORDIC cycle.

Due to the restrictions that apply to student IC design projects at ETH, only two CPEs were integrated on a recently manufactured 1.2 μm CMOS chip. It comprises approximately 32 k transistors on a core area of 9.2 mm^2. The incorporation of several units in a future version of the

Initialisation:

$x_0 = x_{\text{in}}/2,\ y_0 = y_{\text{in}}/2$
add all-zero guard bit extension

for $j = 0 \ldots 18$ do:

$x_{j+1} = x_j - \sigma_j y_j 2^{-s_j} + \gamma_j x_j 2^{-s_j}$
$y_{j+1} = y_j + \sigma_j x_j 2^{-s_j} + \gamma_j y_j 2^{-s_j}$
where
$\sigma_j = -\text{sign}(x_j \cdot y_j) = \sigma_{\text{out}}$ in Vec. mode and
$\sigma_j = \sigma_{\text{in}}$ in Rot. mode
$s_j = \{0, 1, 2, 3, 4, 5, 6, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17\}$
$\gamma_j = \{0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1\}$

Correction cycle:

if $x_{19} < 0$ then $x_{20} = x_{19} + 1$ else $x_{20} = x_{19}$
if $y_{19} < 0$ then $y_{20} = y_{19} + 1$ else $y_{20} = y_{19}$
erase guard bits

$x_{\text{in}} = x_{20},\ x_{\text{out}} = x_{20},\ y_{\text{out}} = y_{20}$

Table 2: Implemented CORDIC algorithm based on [10].

component should be quite simple due to the highly modular and parameterised design of the CPE using VHDL. **Fig. 3** shows the data path. It achieves a typical cycle time of 25 ns with a 40 MHz system clock which is equivalent to a performance of 2×10^6 rotations per second. A detailed description of the realised device is contained in [11]. The chip provides test structures for boundary scan, pad test and core full scan via only 10 pads, allowing full testability of bare dies intended for multi-chip modules (MCMs).

4. SUMMARY

A systolic beamformer architecture has been presented, based on a single type of processing cell using CORDIC arithmetic. Subsequently, the design of an appropriate ASIC containing two CPEs was described. Approximately 40 of these small, unpackaged chips (redundant devices included) will be needed to implement a prototype antenna array signal processor in MCM technology for an $M = 5$ element array delivering complex, I and Q baseband signals at a sampling rate of up to $f_s = 2$ MHz. Preliminary simulation results confirm that our 16 bit fixed-point CPE offers sufficient precision for the envisaged applications [12].

5. REFERENCES

[1] W. M. Gentleman and H. T. Kung, "Matrix Triangularization by Systolic Arrays," in *Real-Time Signal Processing IV*, Proc. SPIE vol. 298, San Diego, CA, August 1981, pp. 19–26.

[2] J. G. McWhirter, "Recursive Least-Squares Minimization Using a Systolic Array," in *Real Time Signal Processing VI*, Proc. SPIE vol. 431, San Diego, CA, August 1983, pp. 105–112.

[3] C. R. Ward, P. J. Hargrave, and J. G. McWhirter, "A Novel Algorithm and Architecture for Adaptive Digital Beamforming," *IEEE Trans. Antennas Propagat.*, vol. 34, no. 3, pp. 338–346, March 1986.

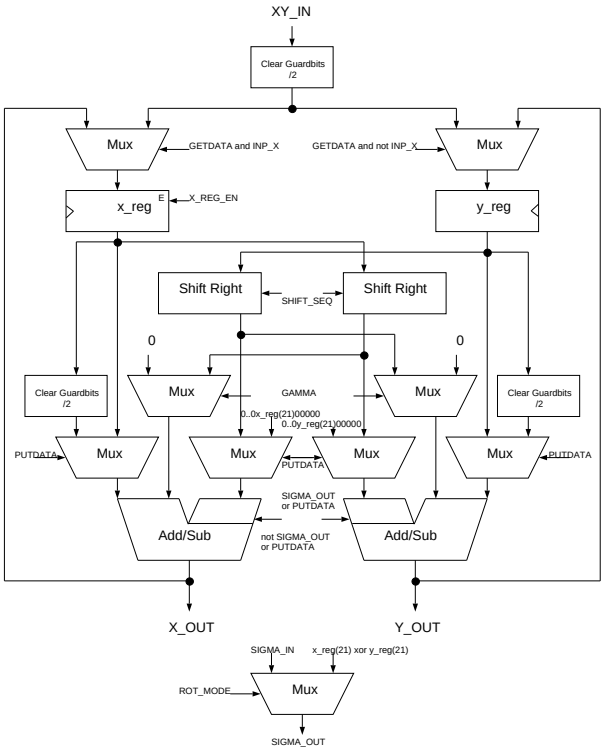

Figure 3: CPE data path.

[4] S. Y. Kung, "VLSI Array Processors," *IEEE Acoust., Speech, Signal Processing Mag.*, vol. 2, no. 3, pp. 4–22, July 1985.

[5] J. E. Volder, "The CORDIC Trigonometric Computing Technique," *IRE Trans. on Electronic Computers*, vol. 8, no. 3, pp. 330–334, September 1959.

[6] H. M. Ahmed, J.-M. Delosme, and M. Morf, "Highly Concurrent Computing Structures for Matrix Arithmetic and Signal Processing," *IEEE Computer*, vol. 15, no. 1, pp. 65–82, January 1982.

[7] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Baltimore, MD: The John Hopkins University Press, 2nd ed., 1989.

[8] J. V. McCanny and J. G. McWhirter, "Some Systolic Array Developments in the United Kingdom," *IEEE Computer*, vol. 20, no. 7, pp. 51–63, July 1987.

[9] Y. H. Hu, "CORDIC-Based VLSI Architectures for Digital Signal Processing," *IEEE Signal Processing Mag.*, vol. 9, no. 3, pp. 16–35, July 1992.

[10] A. M. Despain, "Fourier Transform Computers Using CORDIC Iterations," *IEEE Trans. Computers*, vol. 23, no. 10, pp. 993–1001, October 1974.

[11] U. Fleisch and M. Streiff, *CORDIC Processor Element for a Systolic/Wavefront Array*, Student Project Report no. IKT-NT 780 (in German), ETH Zurich, Winter Semester 1995/96.

[12] B. Haller, *RLS-Based Adaptive Antenna Array Signal Processing for Mobile Communications: Concepts, Algorithms and Dedicated Hardware Architectures*, Post-Diploma Thesis, Communication Technology Laboratory, ETH Zurich, in preparation.