COARSE ANGLE ROTATION MODE CORDIC BASED SINGLE PROCESSING ELEMENT QR-RLS PROCESSOR

Qiang Gao, Louise Crockett and Robert Stewart

EEE Department, University of Strathclyde,
204 George Street, Glasgow, G1 1XW, Scotland, UK
phone: + (44) 141 548 2605, fax: + (44) 141 552 2487, email: qiang.gao@eee.strath.ac.uk

ABSTRACT

Over the last 30 years Digital Signal Processing algorithm implementation has been driven by the continued progress and availability of high speed ASIC circuit technology. The classic method of CORDIC (Coordinate Rotation by Digital Computer) arithmetic has been widely implemented as part of the computational requirements of the well known QR-RLS (recursive least squares) algorithm. In this paper we propose a new modified version of the CORDIC that features a single processor element that is easily pipelinable and can be used to implement both the Givens generations and Givens rotations associated with the QR update. Using a Xilinx FPGA for implementaton results show that this proposed structure requires less resources and produces a more regular and therefore lower cost structure than other equivalent methods recently presented.

1. INTRODUCTION

Within the area of wireless communications, the increasing communications speeds required means that there is sustained demand for high speed implementations of data equalisers, adaptive beamformers, beamsteerers and MIMO (multiple input multiple output) antenna systems. The key algorithm to implement these applications is of course the least squares adaptive signal processing technique [1]. Over recent years, the favoured method of high speed (parallel) implementation has been the QR Decomposition (QRD)-RLS algorithm [1] given its fast convergence, and compared to other methods of solving the least squares equations, its excellent numerical conditioning [2]. Furthermore from an implementation point of view, the QR is very well known for its triangular systolic type array implementation [3] [4].

The structure of this paper is organised as follows. In Section 2 we review the standard QR triangular array. In Section 3, the systolic array implementation of QR-RLS with downdating is briefly reviewed. In Section 4, the conventional CORDIC algorithm based QR-RLS processing element is presented. A new *Coarse Angle* Rotation Mode CORDIC architecture is elaborated in Section 5 to replace the one conventionally used in all the Processing Elements (PEs) of the QR-RLS array. The design of an efficient channel interleaving CORDIC is presented in Section 6. Finally Section 7 proposes one new single PE QR-RLS architecture to replace the existing linear array [5][6].

2. THE CLASSIC QR TRIANGULAR ARRAY

Based on [1] and [2], the standard QR-RLS recursive algorithm is summarized as:

$$\boldsymbol{Q}(k)\begin{bmatrix}\lambda^{1/2}\boldsymbol{R}(k-1) & \lambda^{1/2}\boldsymbol{u}(k-1)\\ \boldsymbol{x}^T(k-1) & d(k-1)\end{bmatrix} = \begin{bmatrix}\boldsymbol{R}(k) & \boldsymbol{u}(k)\\ 0 & a(k)\end{bmatrix} \quad (1)$$

Where $\boldsymbol{Q}(k)$ is the orthogonal rotation matrix, which triangularises the left hand side matrix of Eq. (1), λ is the *forgetting factor* (set to a value less than 1), and $a(k)$ is the *posteriori* Least-Square residual. The $N \times N$ matrix $\boldsymbol{R}(k)$ is the well known correlation matrix, and the N element vector $\boldsymbol{x}(k)$ is the input data vector. N is the number of weights in the adaptive filter [1].

Figure 1 shows a simple signal flow graph (SFG) representation of the well known standard QR-RLS systolic array [7] aiming to find the weights of an $N = 3$ weight FIR filter (this small dimension is used to keep the SFG diagrams simple), comparing the input signal $x(k)$ with the desired signal $d(k)$ with the aim of minimising the power of the error signal $e(k) = d(k) - x(k)$ [1].

This array uses the Givens rotation to convert the input data into an upper triangular matrix $\boldsymbol{R}$ [2]. Each r_{ij} stored and updated inside of the array is one element of this upper triangular matrix, where the subscript (i,j) represents the location of the element in $\boldsymbol{R}$ matrix and $\boldsymbol{u}$ vector [4]. $\boldsymbol{x}(k)$ represents the input signal. The definitions of the Boundary Cell (BC) and Internal Cell (IC) of QR-RLS array are also described in Figure 1. s and c are used to express the Givens rotation values (often denoted Sine & Cosine) calculated with in a boundary cell. The right most column generates the product of the cosines γ. Note that this operation is normally undertaken in the BCs ([1] [3] [4] [5] [6]). However by attaching the right most column of ICs [8], it becomes possible to construct the BC and IC with very similar architectures, which can be exploited with the efficient folding of both types of cell onto a single PE (to be discussed in Section 7). The posteriori error $e(k)$ can be found from the multiplier $\otimes$ [1].

In many implementations ([5], [9]) the final coefficient weight vector is derived from the outputs of the QR decomposition algorithm using backsubstitution. To implement a systolic structure for backsubstitution, one approach is to append a linear array to the upper triangular

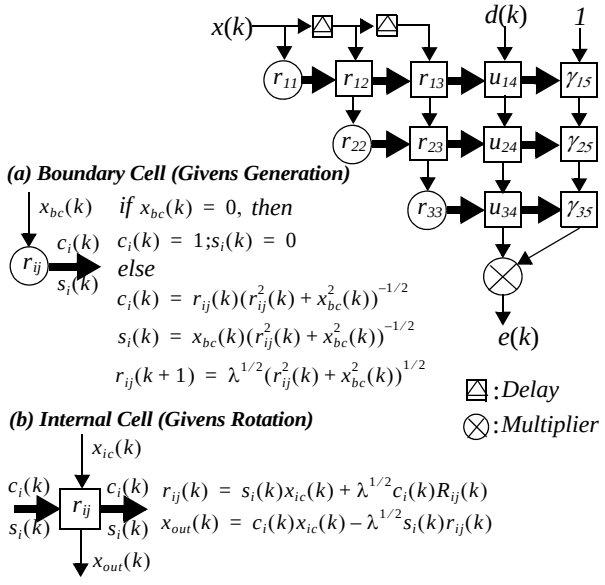


Figure 1 - Standard systolic array of QR-RLS

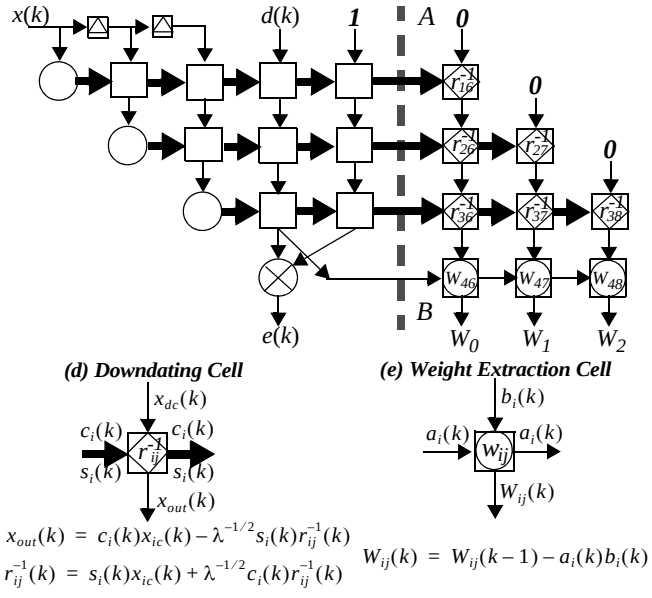


Figure 2 - Systolic array of QR-RLS with downdating

QR-RLS array [5] [9]. However it is worth noting that whereas the QR triangularisation method has excellent numerical properties and is hence the favoured method for fixed point implementation, the procedure of backsubstitution is known to be a process that is not numerically well conditioned [10].

Another issue with the regular implementation of the QR array is that it is not easily interfaced to the backsubstitution array, which is a linear array that requires the $\boldsymbol{R}$ values to be first "shifted" out from the QR array [5]. Also, the data throughput of the linear array is lower than that of the triangular array. Therefore in general the backsubstitution operation is not well suited for an FPGA dataflow array (and is usually performed on an embedded processor) [5] [9].

The extended QR-RLS (QR-RLS with downdating) algorithm derived for weight extraction could mitigate the need for the backsubstitution by adding a second lower triangular *downdating array* such that the final adaptive filter weights can then be extracted by a single multiplication and subtraction operation [10] [11].

3. QR-RLS ARRAY WITH DOWNDATING

The QR-RLS with downdating derived for weight extraction can be implemented with a CORDIC-based *double* triangular systolic array [10] [11]. The additional downdating part must satisfy the following equation [11]:

$$\boldsymbol{Q}(k)\begin{bmatrix}\lambda^{1/2}\boldsymbol{R}(k-1) & \lambda^{1/2}\boldsymbol{u}(k-1) & \lambda^{-1/2}\boldsymbol{R}^{-1}(k-1)\\ \boldsymbol{x}^T(k-1) & d(k-1) & 0\end{bmatrix} = \begin{bmatrix}\boldsymbol{R}(k) & \boldsymbol{u}(k) & \boldsymbol{R}^{-1}(k)\\ 0 & a(k) & b(k)\end{bmatrix} \quad (2)$$

An example of this QR-RLS array with downdating is shown in Figure 2. The triangular section on the left of dashed line A-B, consists of the same BCs and ICs as Figure 1. The lower triangular (downdating) section (on the right hand side of line A-B) rotates the $\boldsymbol{R}^{-1}$ matrix stored in the ◇ cells and an externally applied vector of zeros. Obviously, both the ICs in Figure 1 and the Downdating Cell (DC) in Figure 2 execute nearly the same operation. The only difference is that the *forgetting factor* in each DC is equal to $1/\lambda$. Hence the DC could also be mapped together with the IC and BC, and this folding strategy will be described in Section 7.

At this point we should point out that the use of a QR "rhomboid" array is well known to increase the arithmetic wordlength compared to the standard QR triangular array. This can be simply observed for example by noting that for initialisation, the initial values of the standard QR upper triangular array are set to very small values, say σ, and therefore equivalently the initial values in the lower triangular downdating array, must be set to $1/\sigma$. However the objective of this paper is not to review the numerical properties, but to derive a uniquely regular high speed QR array.

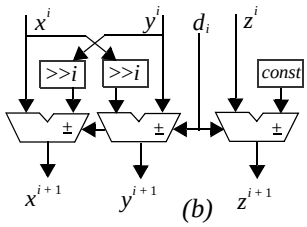
4. CORDIC QR BOUNDARY & INTERNAL CELLS

Last section mentioned that both the boundary and internal cells inside the systolic QR-RLS array are based on the Givens rotation. One low complexity technique that performs the Givens rotation is the CORDIC algorithm. It is an iterative method which is used to calculate many trigonometric and algebraic functions [12] [13]. CORDIC can be implemented using only shift and add operations, and hence results in an efficient hardware design.

For one CORDIC iterative index, i, the rotation equations are shown in Figure 3(a):

$$\begin{aligned} x^{i+1} &= x^i - d_i(2^{-i}y^i)\\ y^{i+1} &= y^i + d_i(2^{-i}x^i)\\ z^{i+1} &= z^i - d_i(\operatorname{atan}2^{-i}) \end{aligned} \quad (3)$$

(a)



(b)

Figure 3 - (a) Classic CORDIC iterative algorithm (b) The hardware implementation for single iteration element

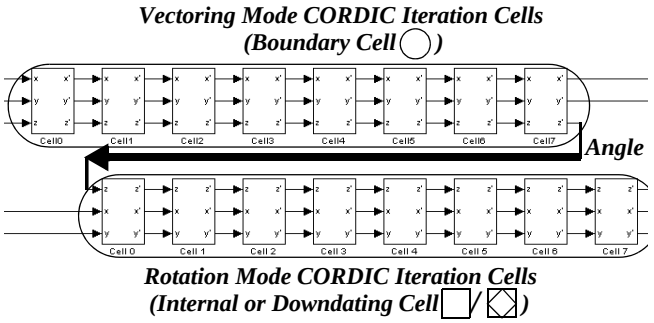


Figure 4 - Interconnection between the BC and IC/DC in [13]

Note that the number of iterations of i is a function of the desired numerical accuracy [14]. If the vector is rotated clockwise, the *decision factor* d should equal to -1, otherwise, $d = 1$. The hardware architecture of the classical single iteration element is shown in Figure 3(b).

Each boundary cell of the QR array requires the vectoring mode CORDIC where Eq. (3) is implemented with the next angle iteration chosen in order to drive $|y|$ towards zero. The internal cell uses the rotation mode CORDIC where the d (rotation direction) is chosen as a function of whether angle z is negative or positive as a result of the last iteration. As shown in Figure 4, we assume both the boundary and internal cells have 8 iteration cells, and each cell has the architecture in Figure 3(b). The number of iteration cells depends on the desired numerical accuracy [14]. The classic vectoring mode CORDIC generates the angle z, and the rotation mode performs the Givens rotation for the same angle (Figure 5).

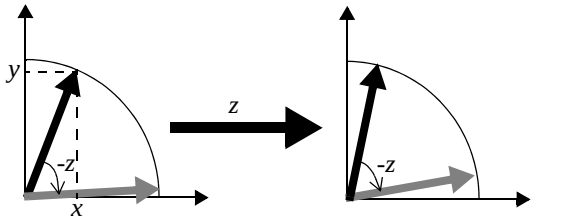


Figure 5 - Link between the classic QR-RLS BC and IC/DC

5. COARSE ANGLE ROTATION MODE CORDIC

According to the circuit implementation, based on Eq. (3), we need to build three subcircuits for x, y and z computations. One of the core outputs of this paper is to find a common structure to allow the same array to be easily shared for the various boundary and internal cells. The x and y parts are always necessary for the Givens rotation operation. In the following part of this paper, an advanced CORDIC/QR architecture is proposed in which the z component of Eq. (3) can in fact be discarded.

As an alternative to calculating the angle z for the Givens rotation, we could in fact just pass the coarse angle rotation information d, i.e a 1 or a -1; given the angle used in the boundary is that same as the angle used in the internal cells the sequence of the d directions is absolutely the same. In the next section we will demonstrate the structure of a new CORDIC based QR boundary and internal cells which are simplified based on this observation.

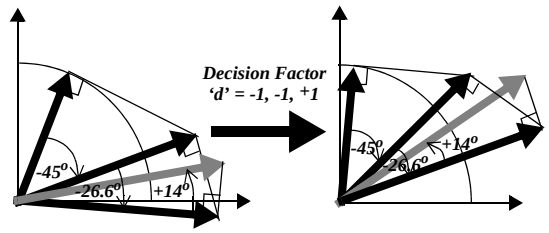


Figure 6 - Signal transfer between QR boundary and internal cells

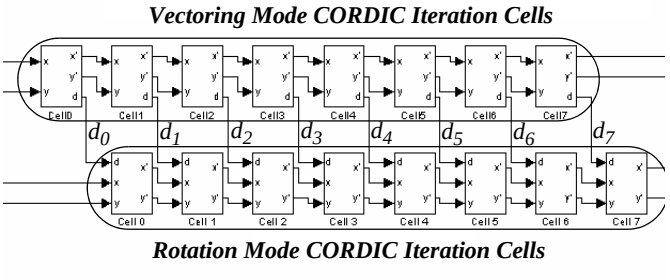


Figure 7 - Interconnection between the new BC and IC/DC

In Figure 6, during the first rotation iteration, the vector input to the boundary cell will be rotated by 45° clockwise, and the generated d is -1. When transferred to the internal cell, d is simply used to generate the required 45° rotation.

After three CORDIC iterations (and therefore three d values passed from boundary to internal) move to before the bracketed bit, we find that the initial vectors have been rotated by the required angle. For a boundary cell, the total rotated angle is 45+26.6-14=57.6°, and for internal cell, it remains the same. Clearly as with the classic CORDIC the more iterations undertaken, the more accurate the angle of rotation. Therefore it is possible to perform Givens rotations of the same angle in different cells in a QR array without explicitly computing the angle, but rather communicating the direction d of the various CORDIC coarse angle values. As will be justified and shown next this leads to hardware resource reduction and produces a more regular array.

The interconnection of new vectoring and rotation CORDICs is shown in Figure 7. The top row of the CORDIC iteration cells still represents the BC of the conventional QR-RLS array, and the bottom row of iteration cells could be considered as either one IC or DC. The link between both modes CORDIC is the *decision factor* d of each iteration.

Now the iteration cells in either boundary (BC) or internal (IC)/downdating (DC) PEs principally requires two bit-shifters and two adder/subtractors to calculate the x and y in Eq. (3), hence the BC and IC/DC proposed above can be easily mapped onto a single PE architecture by applying this folding technique.

5.1 COMPARISION WITH THE CLASSIC CORDICS

We can now compare the synthesis results from the new CORDIC (*Coarse Cordic Givens*) based 16-iteration and 18-bit wordlength BC and IC with the corresponding PEs using

Type	PE	Slices	FFs	LUTs	DSP48	BRAM
Standard Cordic Givens	Boundary	606	770	1150	0	1
	Internal	610	770	1147	0	1
Coarse Angle Cordic Givens	Boundary	409	611	789	0	1
	Internal	408	610	789	0	1
Standard Cordic Lookup	Boundary	640	780	1173	0	2

Figure 8 - Synthesis results for 3 types of CORDIC based BC/ICs (16-iteration & 18-bit wordlength)

the conventional architecture (*Standard Cordic Givens*) demonstrated in Figure 4. For this comparison, real arithmetic arrays were compared. The FPGA resource utilizations are listed in Figure 8. Xilinx ISE 10.1 is used to target a Virtex-4 xc4vsx35-10ff668 device. The new proposed Coarse Angle Rotation Mode CORDIC saves 32.5% hardware resources compared to the classic one in [13], for both the BC and ICs.

Results in Figure 8 also show that the new Coarse Angle Rotation Mode CORDIC based BC uses nearly the same amount of resources as the IC, due to their similarity.

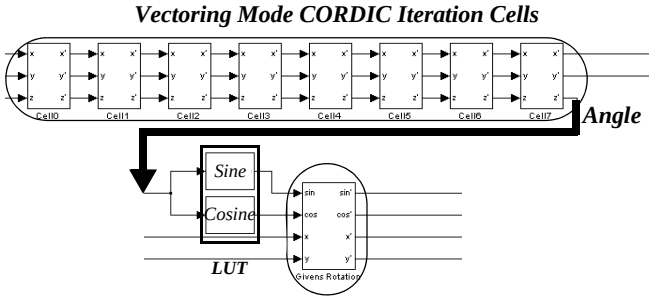


Figure 9 - Interconnection between the [5] based real BC and IC

An alternative architecture (*Standard Cordic Lookup*) adopted in [5] is shown in Figure 9. The boundary cells of the QR-RLS array in [15] use the standard CORDIC architecture which involves the angle '*z*' calculation component. One additional LookUp Table (LUT) is applied to convert each generated '*z*' to its corresponding Sine and Cosine values. With these trigonometric functions as the inputs, the IC/DC can perform a Givens rotation using only multiply and add operations. Here the wordlength and number of CORDIC iterations are still 18 bits and 16 respectively. An architecture for real Given's rotations (based on that presented in [5] for complex numbers) was constructed and compared with the architecture presented in this paper. Its synthesised resource utilization is also shown in Figure 8. The hardware cost of the boundary cell based of this structure is larger (56%) than that of our proposed boundary cell. This design also uses the Virtex-4's arithmetic elements (DSP48 slices).

The critical advantage of the coarse angle rotation mode CORDIC based BC and IC/DC is that they could be mapped onto a single processing element, which may result in the implementation of an efficient, pipelined QR-RLS processor. In the next Section, the concept of single PE QR-RLS processor will start to be introduced.

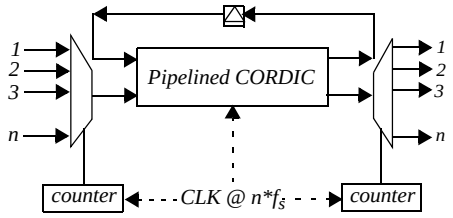


Figure 10 - Channel Interleaving

6. EFFICIENT CHANNEL INTERLEAVING CORDIC

For maximum efficiency a pipelined and parallel CORDIC architecture should be implemented. If we just assume that there are only 4 CORDIC iteration cells used in all three types of PEs in our QR-RLS array, based on the synthesis result under the same devices and wordlength in Section 5.1, the clock rate of the pipelined Coarse Angle Rotation Mode CORDIC can reach up to *200MHz*, and hence with the Channel Interleaving architecture in Figure 10, the data rate of up to *200Msamples/sec* can be supported.

The Channel Interleaving based pipelined CORDIC in Figure 10 offers a low cost hardware solution, which allows the work of several CORDICs to be done by a single cell. However, unless *n* input channels exist to fill *n* pipeline stages then there will still be some redundancy [15]. Hence the efficiency shown here is of course a special case of multichannels being available.

Recall that the critical advantage of the new BC and IC is that they could be mapped onto a single PE without a large increment of resourse ultilization. The synthesis results for this single PE architecture (which adopts the same wordlength and iteration number with that in Figure 8) is listed in Figure 11.

PE	Slices	FFs	LUTs	DSP48	BRAM
Single PE	578	795	995	0	1

Figure 11 -Synthesis result for the single PE based BC and IC sharing

Comparing above synthesis results with that in Figure 8, the cost of single PE which maps the BC and IC together is 10% less than the utilization of one BC based on [5]. Furthermore the BC and IC in [5] have totally different architectures. Hence by adopting the Coarse Angle Rotation Mode CORDIC, the Channel Interleaving based single PE architecture has the advantage of reduced cost compared to other existing techniques.

7. CHANNEL INTERLEAVING CORDIC QR-RLS

To translate the previous QR design to a single Processing Element (PE) architecture, firstly, the extended QR-RLS structure shown in Figure 12 must be transferred to the single row rolled architecture, i.e fold all the rows in the dashed rings to one linear array [5] [6]. At this point, the data rate reduces to CLK/n_1, where n_1 represents the number of rows.

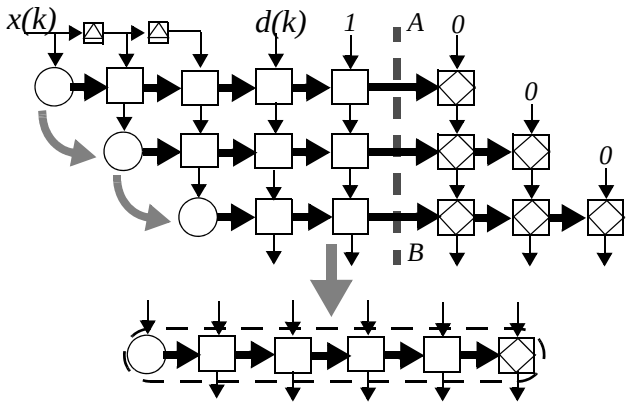


Figure 12 - Linear QR-RLS array transformation

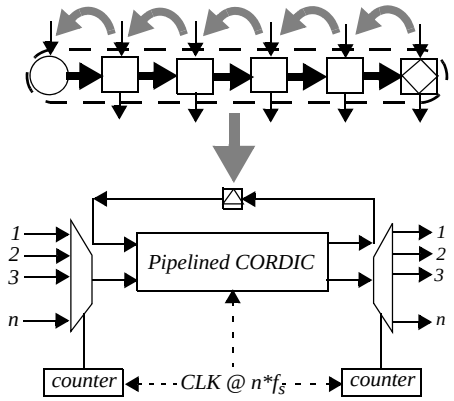


Figure 13 - Channel interleaving for multi PEs per row

Then the next step in the transformation is 'combining' all the PEs in the linear array by the space-time sharing. In Figure 13, the data streams processed by the n_2 columns of cells could be considered as n_2 separate channels, and therefore the channel interleaving scheme we discussed above could be adopted here. This would allow the row of n_2 cells to be mapped onto a single, channel interleaved CORDIC cell, as shown in Figure 13. While the work in [5] considers the efficient realisation of boundary and internal cells as discrete processing blocks, this paper focuses on generalising the architecture of the boundary and internal cells, such that they can be folded onto a single processing element. The folding operation on the internal cells without applying the channel interleaving will result in further decrease of design's data rate.

One point that needs to be mentioned is: the size of QR-RLS array depends on the iteration number of CORDIC inside each PE. When CORDIC has 16 iterations, the size of QR-RLS could not exceed 16.

8. CONCLUSIONS

In this paper we have proposed and analysed the Coarse Angle Rotation Mode CORDIC. By comparing with two types of conventional CORDIC based boundary and internal cells, it is obvious that the new CORDIC has low cost. By using the Channel Interleaving CORDIC to multiplex more than one data channels together, the pipelined architecture can be filled up efficiently. The maximum clock speed can remain the same independent of the number of multiplexed channels. This paper has proposed a method to map the standard QR-RLS array with downdating onto a single processing element architecture. The Channel Interleaving CORDIC based single PE QR-RLS processor combines low hardware consumption with the benefit of scalability, especially when implementing very large matrix sizes. The achieved data rate is reduced to $1/n_1$ of the maximum clock rate, where n_1 represents the number of rows in the original QR-RLS array with downdating.

9. ACKNOWLEDGEMENT

The authors would like to thank the EPSRC Specknet project & Xilinx XUP for support in this research work.

10. REFERENCES

[1] S Haykin. *Adaptive Filter Theory*. Prentice Hall, 3rd edition,1996. ISBN 0-13-397985-7

[2] G H Golub, C F. Van Loan, *Matrix Computations*, Edition: 3, Johns Hopkins University Press, 1996, ISBN 0801854148

[3] J G McWhirter, "Systolic array for recursive least-squares minimisation", *Electronics Letters*, Volume: 19, Issue: 18, pp: 729-730, Sept 1. 1983

[4] R Woods, J McAllister, Y Yi, G Lightbody, *FPGA-based Implementation of Signal Processing Systems*, Wiley, 1999

[5] C Dick, F Harris, M Pajic and D Vuletic,"Real-Time QRD-Based Beamforming on an FPGA Platform" *Fortieth Asilomar Conference on Signals, Systems and Computers*, pp. 1200 - 1204, Oct. 29 2006 - Nov. 1 2006

[6] QinetiQ Quixilica Floating-Point QR Processor Core Data Sheet,*http://www.eonic.co.kr/data/datasheet/transtech/FPGA/qx_qr.pdf*

[7] W M Gentleman and H T Kung, "Matrix Triangularization by Systolic Arrays," *Real-Time Signal Processing IV, Proc. SPIE*, Volume 298, 19-26.

[8] J Ma, K K. Parhi and Ed F. Deprettere, "Annihilation-Reordering Look-Ahead Pipelined CORDIC-Based RLS Adaptive Filters and Their Application to Adaptive Beamforming", IEEE Transactions on Signal Processing, Vol. 48, NO. 8, Aug 2000

[9] Altera White Paper, "Implementation of CORDIC-Based QRD-RLS Algorithm on Altera Stratix FPGA with Embedded Nios Soft Processor Technology", *http://www.altera.com/literature/wp/wp_qrd.pdf*

[10] Bin Yang, Johann F. Bohme, "Rotation-based RLS algorithm unified derivation, numerical properties, and parallel implementation," *IEEE Trans. on Signal Processing*, vol. 40, no.5, pp. 1151-1167, May 1992.

[11] M Harteneck, R W Stewart, J G McWhirter, I K Proudler, "Algorithmic Engineering Applied to the QR-RLS Adaptive Algorithm", *Proceedings of 4 th International Conference on Mathematics in Signal Processing*, 1996

[12] J. E. Volder, "The cordic trigonometric computing technique", *IRE Trans. Electronic Computers*, vol. 3, pp. 330–334, Sept. 1959.

[13] R Andraka,"A survey of CORDIC algorithms for FPGA based computers", *Proceedings of the 1998 ACM/SIGDA sixth international symposium on Field Programmable Gate Arrays*, pp. 191-200, Feb. 22-24, 1998

[14] Y. H. Hu, "The Quantization Effects of the CORDIC Algorithm", *IEEE Trans. On Signal Processing*, Vol 40, No 4, pp. 834-844, 1992

[15] K. K Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*, Wiley-Interscience, 1999