Implementation Of Quadratic Rotation Decomposition Based Recursive Least Squares Algorithm

Manpreet Singh[1], Sandeep Singh Gill[2]
[1]University College of Engineering, Punjabi University, Patiala-India
[2]Guru Nanak Dev Engineering College, Ludhiana-India

Abstract: *Quadratic Rotation decomposition (QRD) based recursive least squares (RLS) algorithm can be used in variety of communication applications and its low complexity implementation can be of interest. In this paper we have presented FPGA implementation of QRD based RLS algorithm using Coordinate Rotation by Digital Computer (CORDIC) operator. FPGA resource estimates along with actual implementation results illustrating the weight calculation delays have also been presented presented.*

I. Introduction

Adaptive signal processing algorithms such as least mean squares (LMS), normalized LMS (NLMS), and RLS algorithms have been historically used in numerous wireless applications such as equalization, beamforming and adaptive filtering [1] and [2]. With the advent of wideband third-generation (3G) wireless systems, adaptive weight calculation algorithms are also being considered for new applications such as polynomial-based digital predistortion and MIMO antenna solutions [3]. These applications generally involve solving for an over specified set of equations 1

$$\begin{aligned} x_1(1)c_0 + x_2(1)c_1 + ...x_N(1)c_N &= y(1) + e(1) \\ x_1(2)c_0 + x_2(2)c_1 + ...x_N(2)c_N &= y(2) + e(2) \\ &\vdots \\ x_1(m)c_0 + x_2(m)c_1 + ...x_N(m)c_N &= y(m) + e(m) \end{aligned} \qquad (1)$$

where $m > N$**.**

Among the different algorithms, the recursive least squares algorithm is generally preferred for its fast convergence property. The least squares approach attempts to find the set of coefficients c_n that minimizes the sum of squares of the errors, i.e. representing $\left\{ \min \sum_m e(m)^2 \right\}$, equation 1 can be represented in matrix form as

$$Xc = y + e \qquad (2)$$

Where X is a matrix ($m \times N$, with) of noisy observations, **y** is a known training sequence, and c is the coefficient vector to be computed such that the error vector e is minimized. Direct computation of the coefficient vector **c** involves matrix inversion, which is generally undesirable for hardware implementation. Matrix decomposition-based least squares schemes such as Cholesky, lower upper (LU), singular value (SV), and QR decompositions avoid explicit matrix inversions and are more robust and well suited for hardware implementation [1]. Such schemes are being increasingly considered for high sample rate applications such as digital predistortion, beamforming and multiple-input multiple-output (MIMO) signal processing. FPGAs are the preferred hardware platform for such applications because of their capability to deliver enormous signal processing bandwidth. In recent years, FPGAs have become available with increasingly powerful embedded soft processor cores that give designers the flexibility and portability of high-level software design, while maintaining the performance benefits of parallel hardware operations in FPGAs, [4]. The rest of this paper describes the proposed implementation of QR decomposition-based RLS algorithm (QRD-RLS) on Altera's Stratix FPGA with embedded Nios soft processor technology.

Section II provides an overview of the QRD-RLS algorithm. Implementation of the QR decomposition using CORDIC (Coordinate Rotation by Digital Computer) blocks and the systolic array architecture is described in Section III. Section IV describes the implementation of the back substitution process on the Nios soft processor using custom instructions. Simulation results and resource estimates are presented in the section V.

II. QRD-RLS Algorithm

As described earlier in Figure 2 (1), the least squares algorithm attempts to solve for the coefficient vector from and. To realize this, the QR decomposition algorithm is first used to transform the matrix *X* into an upper triangular matrix R (NxN matrix) and the vector into another vector such that**.** The coefficients vector is then computed using a procedure called back substitution, which involves solving the equations shown in equation 3.

$$C_N = \frac{u_N}{R_{NN}}$$

$$C_i = \frac{1}{R_u}\left(N_i - \sum_{f=i+1}^{N} R_{if} c_f \right) \text{ for } i = N-1,\ldots 1 \qquad (3)$$

The QRD-RLS algorithm flow is depicted in Figure 1.

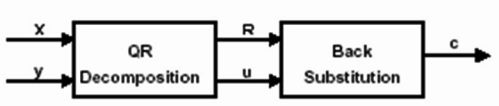

Figure 1: QR Decomposition Base Least Squares

III. QR Decomposition using Cordic Algorithm

The QR decomposition of the input matrix *X* can be performed, as illustrated in Figure 2, using the well known systolic array architecture [5]. The rows of matrix *X* are fed as inputs to the array from the top along with the corresponding element of the vector **y**. The **R** and **u** values held in each of the cells once all the inputs have been passed through the matrix are the

outputs from QR decomposition. These values are subsequently used to derive the coefficients using back substitution technique.

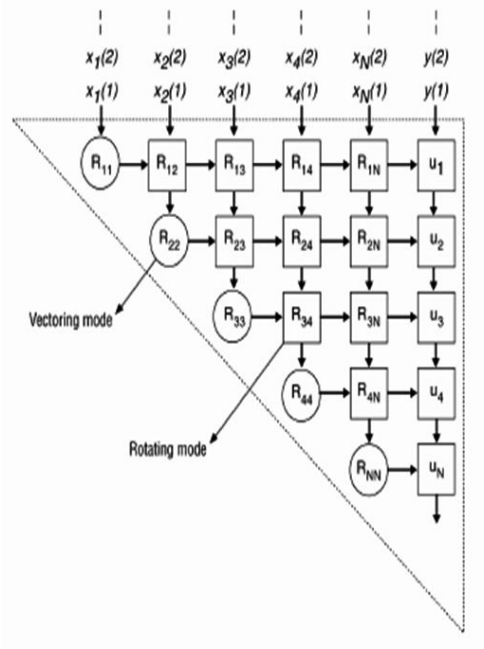

Figure 2: Systolic Array architecture for QR Decomposition

Each of the cells in the array can be implemented as a CORDIC block. CORDIC describes a method to perform a number of functions, including trigonometric, hyperbolic and logarithmic functions [6]. The algorithm is iterative, and uses only additions, subtractions and shift operations. This makes it very attractive for hardware implementations. The number of iterations depends on the precision, with more iterations being needed for more bits. Altera's CORDIC blocks have a deeply pipelined parallel architecture enabling speeds over 250MHz on Stratix FPGAs in both vectoring and rotating modes. For real inputs only one CORDIC block is required per cell. Many applications involve complex inputs and outputs to the algorithm, for which 3 CORDIC blocks are required per cell. In such cases, a single

CORDIC block can be efficiently timeshared to perform the complex operations. Table 1 illustrates the resource consumption for the CORDIC algorithm in terms of logic elements (LEs) for different input bit widths. The results obtained were achieved using Quartus® II push-button flow on Stratix devices. It can be observed that as the input bit width increases, the resource consumption also increases, whereas the decreases.

Table 1: CORDIC resource consumption

Input Width (Bits)	Number of Iterations	Logic Elements	F_{max} (MHz)
8	8	380	264
16	16	1300	219
24	24	2670	198
32	32	4600	189
40	40	7010	163

Direct mapping of the CORDIC blocks onto the systolic array shown in Figure 5 consumes significant amount of LEs and yields enormous throughput that is generally not required for many applications. The resources required to implement the array can be reduced by trading throughput for resource consumption via mixed and discrete mapping schemes.

Mixed mapping: In the mixed mapping scheme, the bottom rows in the systolic array are moved to the end of the top rows, to possibly have the same number of cells in each row. A single CORDIC block can be used to perform the operations of all the cells in a row, with the total number of CORDIC blocks required being equal to the total number of rows. Since each CORDIC block has to operate in both vectorise and rotating modes, the scheme is called mixed mapping [7].

Discrete mapping: In this scheme, at least two CORDIC blocks are required. One block is used purely for vectorise operations, while the other is used for rotate operations item [8]. This single functionality of the processors allows any gains from hardware optimization to be realizable. Further information on the different mapping schemes can be found in [7-9].

Final coefficient weight vector is derived from the outputs of the QR decomposition algorithm using a procedure called back substitution. The back substitution procedure, as seen in Figure 2, primarily involves multiplication and division operations. The Nios embedded The Nios processor is based on the revolutionary concept of embedding soft-core RISC processors within FPGAs and can operate at over 100 MHz on the Stratix FPGAs [10]. Embedded designers can create custom processor-based systems using the SOPC Builder system development tool. SOPC Builder can be used to integrate one or more configurable Nios CPUs with any number of standard peripherals, gluing the system together with the automatically generated Avalon switch fabric. Figure 3 outlines the procedure involved in computing the coefficients via back substitution. The CORDIC block performs the QR decomposition and stores the R and u values (both real and imaginary) in memory accessible to the Nios processor which then calculates the coefficient values (both real and imaginary) and stores the results back into memory.

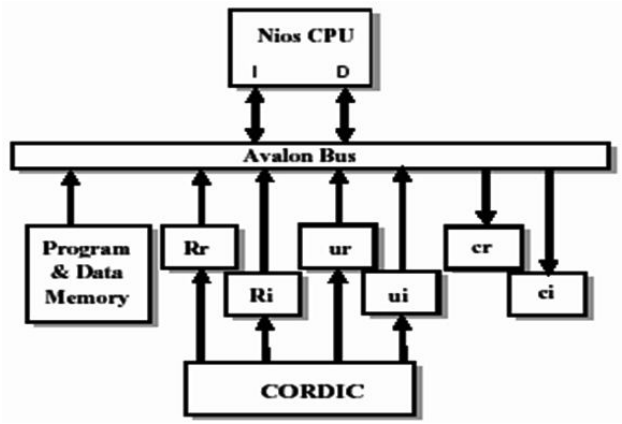

Figure 3: NIOS processor for Back substitution

A 32-bit Nios CPU can optionally be configured to include hardware 16x16 $\rightarrow$ 32 integer multiplier implemented using the digital signal processor blocks on Stratix FPGAs. The MUL instruction can then be used to complete

the multiply operation in a single clock cycle. The divide operation can be implemented as custom logic block that becomes a part of the Nios microprocessor's arithmetic logic unit (ALU). The 32-bit division can then be completed in approximately 35 clock cycles by the Nios processor operating at 100MHz. The Nios configuration wizard creates software macros in C/C++ and Assembly, providing software access to the custom logic block.

V. Resource Estimates

Table 2 gives an overview of the resource estimates for performing the CORDIC-based QR decomposition using the three different mapping schemes described in Section III.

Table 2: Resource Estimates for Different Mapping Schemes

Implementation Technique	CORDIC usage		Throughput		Cost LE/Update
	No. of Blocks	No. of LEs	Update Delay (us)	Updates/s	
Direct Mapping	54	70200	5.1	196078	0.358
Mixed Mapping	4	5200	250.85	3986	1.305
Discrete Mapping	2	2600	198.11	5047	0.515

For m=64 and N=9 with 16-bit complex inputs to the CORDIC is considered. The system clock is assumed to be at 150MHz with a single CORDIC block timesharing the three operations required for complex inputs. The estimates do not include the resources required for the scheduling of operations between the different CORDIC blocks. Update Delay is defined as the time required before all cells in systolic array are updated with their R and z values. Throughput is defined as the number of input matrices (each m x N) that are processed per second = 1/update delay. Cost is defined as the Number of LEs consumed per update. Table 2 shows that direct mapping offers the highest throughput (update/s) but at the expense of 54 CORDIC blocks requiring a huge number of LEs. In almost all applications the cost of these logic resources would make this implementation not viable. Mixed mapping reduces the resource consumption by employing only 4 CORDIC blocks with a corresponding drop in throughput. In comparison, although

the discrete mapping scheme needs 5 CORDIC blocks, it can be implemented using only 2 blocks. Since Altera's CORDIC block is iterative and pipelined, the number of blocks can be reduced from 5 to 2 with only a 3 clock cycle delay penalty. This not only further reduces the resource consumption, but provides higher throughput than the mixed mapping scheme. Thus the discrete mapping scheme best exploits the pipelining ability of Altera's CORDIC block and offers the optimum tradeoff between resource consumption and throughput.

VI. Conclusion

A low complexity FPGA implementation of QRD based RLS algorithm using CORDIC algorithm has been presented. Pipelined parallel architecture has been used to implement the CORDIC algorithm. The use of NIOS soft processor has also been presented to implement back substitution algorithm. Resource estimates have also been presented for three different mapping schemes for performing CORDIC based QRD.

References

[1] Simon Haykin, Adaptive Filter Theory, Prentice Hall, Fourth Edition.

[2] Tim Zhong Mingqian, A.S.Madhukumar, and Francois Chin, "QRD-RLS Adaptive Equalizer and its CORDIC-Based Implementation for CDMA Systems," International Journal on Wireless & Optical Communications, Volume 1, No.1 (June 2003), pages 25-39.

[3] Babak Hassibi, "An Efficient Square-Root Algorithm for BLAST," Proceedings of the 2000 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 737-740.

[4] Stratix FPGAs, http://www.altera.com

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

[6] J. Volder, "The CORDIC Trigonometric Computing Technique," IRE Trans. Electronic Computers, Vol. EC-8, pp. 330-334, 1959.

[7] C.M. Rader, "VLSI Systolic Arrays for Adaptive Nulling," IEEE Signal Processing Mag, Vol.13, No.4, pp.29-49, 1996

[8] G. Lightbody, R.L. Walke, R. Woods, J. McCanny, "Novel Mapping of a Linear QR Architecture," Proc. ICASSP, Volume IV, pp.1933-1936, 1999

[9] R.L. Walke, R.W.M. Smith, "Architectures for Adaptive Weight Calculation on ASIC and FPGA," 33rd Asilomar Conference on Signals, Systems and Computers, 1999

[10] Nios Processor, http://www.altera.com/literature/lit-nio.jsp

* * * * *