White Paper

Implementation of CORDIC-Based QRD-RLS Algorithm on Altera Stratix FPGA with Embedded Nios Soft Processor Technology

This white paper describes the implementation of the QR decomposition-based recursive least squares (RLS) algorithm on Altera Stratix™ FPGAs. Coordinate Rotation by Digital Computer (CORDIC) operators can be efficiently time-shared to perform the QR decomposition while consuming minimal resources. Back substitution can then be performed on the embedded soft Nios® processor by utilizing custom instructions to yield the final weight vectors. Analytical resource estimates along with actual implementation results illustrating the weight calculation delays are also presented.

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, see "References," items [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, see "References," item [3]. These applications generally involve solving for an over specified set of equations, as shown in Figure 1, where m > N.

Figure 1. Equation

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

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. $\{\min \sum_m e(m)^2\}$. Representing the above set of equations in the matrix form, see Figure 2.

Figure 2. Equation: Matrix Form Representing Equation Shown in Figure 1

$$X\mathbf{c} = \mathbf{y} + \mathbf{e} \qquad (1)$$

Where *X* is a matrix (mxN, with m>N) 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), singluar value (SV), and QR decompositions avoid explicit matrix inversions and are more robust and well suited for hardware implementation, see "References," item [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, see "References," item [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 "Resource Estimates & Simulation Results" and the results are summarized in the section "Summary."

Overview of QRD-RLS Algorithm

As described earlier in Figure 2 (1), the least squares algorithm attempts to solve for the coefficient vector **c** from *X* and **y**. 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 **y** into another vector **u** such that R**c**=**u.** The coefficients vector **c** is then computed using a procedure called back substitution, which involves solving the equations shown in Figure 3.

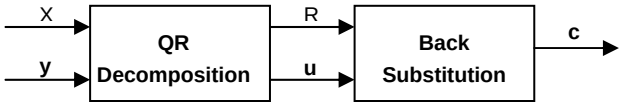
Figure 3. Equation

$$c_N = \frac{u_N}{R_{NN}} \qquad (2)$$

$$c_i = \frac{1}{R_{ii}}\left(u_i - \sum_{j=i+1}^{N} R_{ij} c_j\right) \text{for} \quad i = N-1, \ldots 1 \qquad (3)$$

The QRD-RLS algorithm flow is depicted in Figure 4.

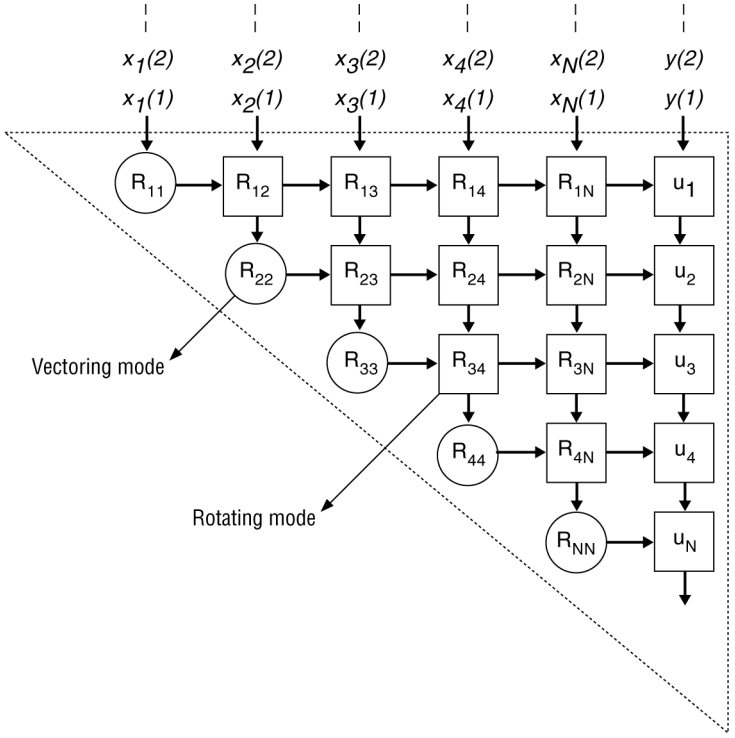
Figure 4. QR Decomposition-Based Least Squares



CORDIC-Based QR Decomposition

The QR decomposition of the input matrix *X* can be performed, as illustrated in Figure 5, using the well-known systolic array architecture, see "References," item [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 5. 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, see "References," item [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 F_{max} decreases.

Table 1. Altera 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, see "References," item [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, see "References," item [8]. This single functionality of the processors allows any gains from hardware optimisation to be realisable.

Further information on the different mapping schemes can be found in the "References," items [7], [8], and [9]. An example resource estimation of direct, mixed and discrete mapping schemes is presented in the section "Resource Estimates & Simulation Results."

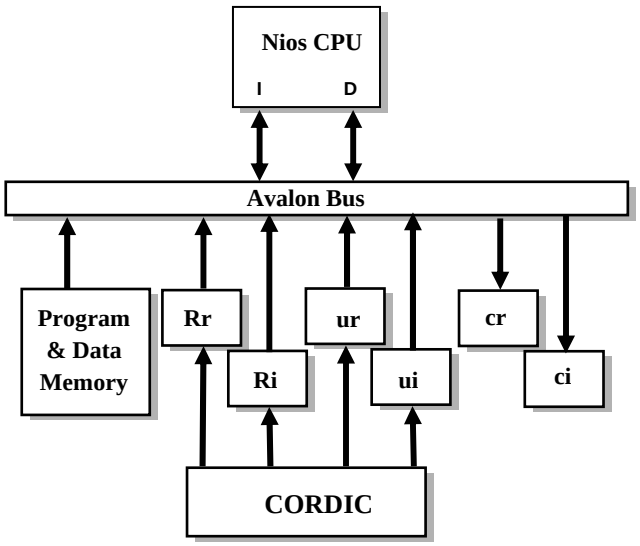
Back Substitution On Nios Soft Processor

As outlined in the section "Overview of QRD-RLS Algorithm," the 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 3, (2)-(3), primarily involves multiplication and division operations. The Nios embedded soft processor, with its custom multiply and divide instructions, is an ideal fit to implement this function.

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, see "References," item [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.

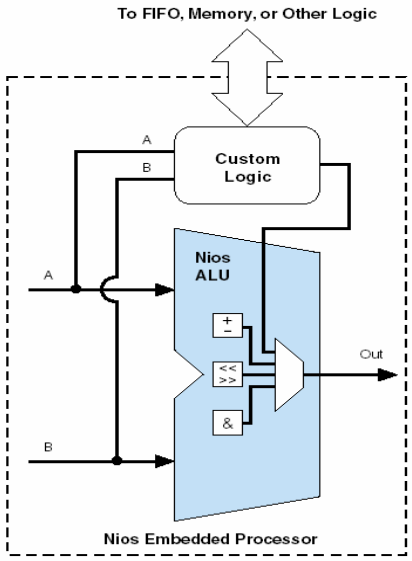
Figure 6 outlines the procedure involved in computing the coefficients via back substitution. The CORDIC block performs the QR decomposition and stores the R and u values (both real and imaginary) in memory accessible to the Nios processor which then calculates the coefficient values (both real and imaginary) and stores the results back into memory.

Figure 6. Nios Processor for Back Substitution



A 32-bit Nios CPU can optionally be configured to include a hardware 16x16 –> 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) as shown below in Figure 7. 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. The back substitution operation can thus be efficiently implemented using hardware accelerated code on the Nios processor, as observed in the simulation results in the next section.

Figure 7. Adding Custom Logic to the Nios ALU



Resource Estimates & Simulation Results

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. An example scenario of 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.

Some new terms are defined as follows:

- *Update Delay:* Time required before all cells in systolic array are updated with their R and z values
- *Throughput:* Number of input matrices (each m x N) that are processed per second = 1/update delay.
- *Cost:* Number of LEs consumed per update.

Table 2. Resource Estimates Example 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

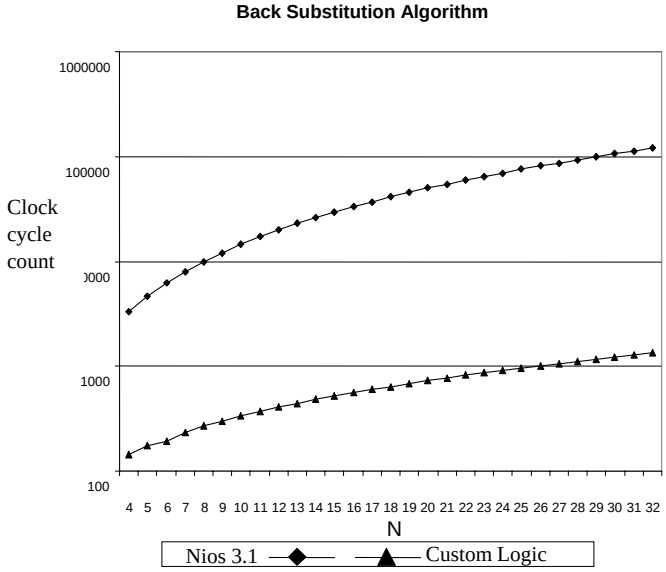
Table 2 shows that direct mapping offers the highest throughput (update/s) but at the expense of 54 CORDIC blocks requiring a huge number of LEs. In almost all applications the cost of these logic resources would make this implementation not viable. Mixed mapping reduces the resource consumption

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 (i.e., one extra clk cycle for each timeshared CORDIC operation). 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.

Figure 8 illustrates the number of clock cycles required for the Nios processor operating at 100MHz to calculate different number of complex coefficients. The back substitution subroutine written in C code calculates the coefficient array and stores into memory. The subroutine assumes that all complex arrays are stored as short (signed 16-bit precision) and all complex numbers are stored in complex conjugate form. The plot also shows the corresponding time taken to calculate the coefficients using dedicated custom logic (hardware-only approach) instead of the processor.

Figure 8. Back Substitution Results: Nios Processor vs. Hardware Only Approach



It can be observed from Figure 8 that for the previously considered example of N=9, the Nios processor takes approximately 12000 clock cycles or 120µs which is acceptable for many applications. Although slower than a pure hardware approach, the presence of the Nios processor lends flexibility to the system. Moreover, the Nios processor can be used to implement other data and control functions on the FPGA during the time it is waiting for the outputs from the CORDIC block. This facilitates a complete System on a Programmable Chip (SOPC) solution without the need for an external processor.

Summary

A novel implementation of the QRD-RLS algorithm using Altera Stratix FPGA with embedded Nios soft processor technology was proposed. The ability to exploit the deeply pipelined parallel architecture of Altera's CORDIC block to implement the QR decomposition using the Discrete mapping scheme was described and resource estimates were presented. The back substitution algorithm was implemented on the configurable Nios soft processor with custom instructions for hardware acceleration. Simulation results were also presented illustrating the efficiency of the Nios processor to facilitate an integrated SOPC solution.

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 Compute*rs, 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

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com

Copyright © 2003 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries.* All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.