Parallel, Pipelined CORDICs for Reconfigurable Computing

Oskar Mencer, Martin Morf

Computer Systems Laboratory, Department of Electrical Engineering
Stanford, CA 94305, USA
email: {oskar,morf}@umunhum.stanford.edu

Abstract

Reconfigurable computing has shown impressive successes with data intensive and latency tolerant applications. Pipelined and parallel implementations of CORDICs can achieve very high throughput for rotation, and various other functions such as multiplication, division, as well as hyperbolic and other higher order functions. Reconfiguration allows us to adapt the implementation of CORDICs and related architectures to the specific needs and properties of individual applications or specific sets of applications; hence creating application specific CORDIC implementations. Therefore it is becoming evident that CORDICs are very well suited to reconfigurable computing and custom computing machines.

1. Introduction and Motivation

CORDIC algorithms were formally introduced by Volder in [1] and unified to compute elementary functions by Walther in [2]. The fundamental mathematical principles behind CORDIC algorithms can be found in their scalar form in the work of T.C. Chen [6], as pointed out by Ahmed in his thesis [19]. Ahmed showed that if T.C. Chen's convergence computation technique is applied instead of real numbers (as assumed by Chen) to complex numbers one obtains the class of CORDIC algorithms. The method of "replacing" real by complex numbers can be extended to other algebras, such as quaternions [18, 26, 27], or more generally using group theoretic methods [18] to compute higher order functions like Bessel functions [M(2) group] or Legendre polynomials [SU(2), related to quaternions or hypercomplex numbers]. As a historic note, one of the earliest methods dates back three centuries ago when Briggs generated his logarithm tables using decimal digits. Another very old mathematical concept, quaternions, has generally been avoided, since *ordinary* matrix theory can be used instead – as has been argued since the inception of quaternions. A similar debate was waged by electrical engineers in the early days of circuit theory about the use of complex numbers for representing impedances as functions of R,L,C's. Matrix methods are beginning to run their course on some advanced problems involving Maxwell's equations (e.g in advanced antenna theory and photonics.) The use of more advanced concepts such as quaternions will increase in the future. Field-Programmable Gate Arrays (FPGAs) which will be described later in more detail, enable effective implementations of quaternion CORDICs in harware (firmware).

The actual successful usage of CORDIC algorithms in hardware has a very checkered history. Some of the touted examples carry their own ironies as well. Given the fact that CORDICs compute vector (or matrix) functions – typically three arguments, and two to three results – it is surprising that sometimes only one result is used, e.g. the HP35 calculator used CORDICs, but only used one result. The HP45 calculator, done by a different design group, computed coordinate transformations with two arguments and two results – but did not use CORDICs! Similarly the Motorolla 68k floating-point coprocessor used CORDICs but saved only one result – the marketing research asked only what *scalar* function customers wanted !

The real power of CORDICs and related algorithms can only be exploited if one matches the CORDIC functions, especially the higher order functions, to applications. Matching algorithms to sets of applications is a computer architecture or system design issue. There are many ways of implementing functions and many criteria to judge such implementations.

Among the criteria we consider "the 5P's": performance, power, persistence, programability, price. Any one of these criteria could make CORDICs not to be the chosen method, unless the application and systems context is considered.

Performance calls for parallelism and maximum pipelining, unless latency is an issue. CORDICs are usually associated with "shift & add" hardware primitives. We believe that this is a too narrow view! Ahmed showed in his thesis [19] that *hybrid CORDICs* with larger lookup tables and specific multipliers instead of "shift & add" with certain multipliers, can improve the implementation according to the P metrics mentioned above. Similarly, combining the partial product arrays (PPA) techniques, described by

This research is supported by DARPA Grant Nr. DABT63-96-C-0106

E. Schwarz [20] in his thesis, with multiple equations used in CORDICs can again improve the CORDICs competitiveness over conventional high performance algorithms.

For a system consisting of hardware and software resources, *persistence* of a task corresponds to the percentage of execution time spend on this particular task. For task with very high persistence, we can then justify the extensive use of highly optimized libraries and/or hand optimization by a human expert (”brain-ware”). Such optimization seems to be even more important for CORDICs, especially if higher order functions are involved.

Custom design of CORDIC units for individual applications appears to be a complex task not suited for the average time constrained programmer. Low-level design tools and symbolic computing tools that support a domain expert are one way to create high performance CORDIC designs. Sophisticated, say web based tools, that can support a typical programmer will eventually become available; however, a perusal of the literature, especially the references (by Morf, Lee, Ahmed, Ang, Delosme) below, indicate that it will take years to build such tools ([24]). In the mean-time domain experts will have to use todays tools to create winning designs using these ideas in advanced applications.

For example, designing a high performance CORDIC processor for solving for the roots of a fourth degree polynomial can be done by first using a symbolic tool, say Mathematica, to find the four symbolic expressions for the roots – 800 kBytes. Using CORDIC functions as primitive transformations combined with common sub-experession elimination can be used to reduce these (initial) symbolic solutions from 800kB to 16 CORDIC operations, which can be organized as four pipelined stages, each with 4 parallel CORDIC operations. Such a maximally pipelined architecture with one result per cycle (and 4 cycles latency) is a very unexpected solution, given the classical solution of two cascaded third order solutions.

The (CORDIC) transformation principle can also be applied at other levels of abstraction:

At the gate-level the concept of rotations translates to conditional permutations, or Conservative Invertible functions (e.g. the Fredkin gate, or the conditional exchange - X-gate). A major application that is based on these functions is Lattice Gas (see thesis of F.F. Lee [14] and his ALGE architecture for solving fluid flow problems.) Conditional exchange (X-gates) and conditional rotation (R-gates) have been developed and exploited in photonics [29] and unitary logic in quantum computing.

Regular Arrays of CORDIC processors are related to systolic arrays. They can compute large rotations, generalized rotations (unitary transforms) or matrix decompositions (eigen-decompositions). Such regular architectures can be viewed as natural generalizations of CORDICs. By combining elementary CORDIC rotations we can achieve arbitrary size rotations. Hence, CORDICs appear as array node processors. CORDICs can also be extended to higher-order functions by using group theory.

Mathematically, generalized CORDIC functions are possibly large transformations that preserve some form of invariants or metrics. Any function can be imbedded into a larger function subject to a metric, the subject of homotopies in mathematics. These larger functions, transformations or rotations, that naturally decompose into elementary transformations or rotations. Therefore CORDICs are natural candidates for elementary operations, or primitives, that can be supported in optimized hardware, firmware, or library functions.

2. Reconfigurable Computing

Reconfigurable computing has been an active area of research over the past decade. A summary of the current state of the field is given in [9]. While it has been shown that FPGAs can achieve an improvement in performance and power over general purpose processors, competitive FPGA designs have been created mostly on a very low, structural level.

The first custom computing machines, the PAM (see section 2) and the Splash-2 [5], were build shortly after the introduction of FPGAs by Xilinx in 1985. Both projects investigated the feasibility of FPGAs as computing platforms.

Conventional general purpose processors consist of a fixed, general datapath, and programmable control (instructions) for that datapath. A few general purpose arithmetic units are highly optimized for low latency i.e.[13].

On custom computing machines datapath and control are fully programmable, allowing the designer to tailor the architecture of the computer to the structure of the algorithm. Flexibility or reconfigurability comes at the expense of latency (i.e. longer cycle time) and logic density on the chip.

Given todays technology, custom computing machines can compete with general purpose processors on latency tolerant applications that require a relatively small amount of logic. We have shown an implementation[10] of the IDEA encryption algorithm that is up to 6 times more performance / power efficient than current leading DSP microprocessors.

Due to large reconfiguration times of todays devices single FPGAs do not scale well to large problem sizes. Multiple FPGAs can be used to compute larger problems. The major drawback is the very high complexity of partitioning a design onto multiple FPGAs given a limited amount of pins. Overcoming the pin-limitation in software – with design tools – is investigated in the Virtual Wires[8] project. Elliminating the pin limitation with multichip modules of FPGAs is explored in the Teramac project[15].

Applications that execute favorably on FPGAs are therefore data intensive applications which can be executed in

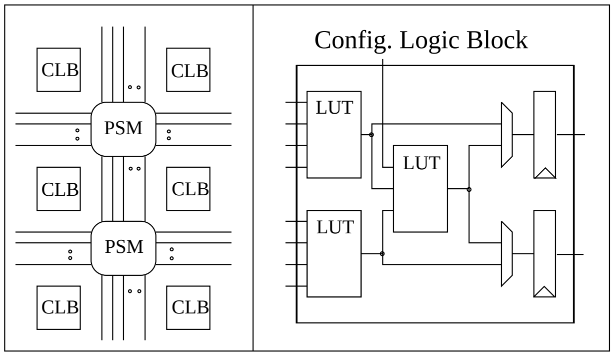

Figure 1: The figure shows the architecture of a Xilinx XC 4000 Configurable Logic Block (CLB). There are two 4-bit Lookup-Tables (LUTs) and up to two registers.

very deep pipelines (e.g. encryption, pattern matching ,etc.) and applications with a huge amount of fine grain parallelism such as lattice gas simulation[14] mentioned above.

For datapaths, hand designs are typically more efficient than compiled behavioral descriptions. In order to exploit the efficiency of hand design while simplifying the design process, we propose a bottom up approach to compilation for custom computing machines. By creating a powerful and highly optimized parameterizable library, PAM-Blox [11], we add a level of abstraction that preserves optimal area and performance while simplifying the design process.

3. The PCI Pamette Custom Computing Machine

We use the PCI Pamette board developed by Mark Shand[17] as the platform for investigating reconfigurable computing. The PCI Pamette consists of 5 Xilinx XC4000 series FPGAs.

The PCI bus with the Pamette board is mapped into main memory. Communication speed between the host CPU and the FPGAs is set by the clock speed of the PCI bus. PCI Pamette supports 33 and 66MHz with PCI bus widths of 32 and 64 bits.

The four user programmable FPGAs can be reconfigured individually or in parallel at runtime, allowing the designer to explore dynamic reconfiguration within the limits of Xilinx XC4000 FPGAs[1].

The interconnect on the PCI Pamette board is a 2D mesh connecting the four user programmable FPGAs. The signal delay from one FPGA to another is therefore very small, compared to more general switch based interconnects.

Area requirement, which is directly proportional to power consumption ([10]) is given in Configurable Logic Blocks (CLBs). Figure 1 shows the architecture of a Xilinx XC4000 CLB.

[1]For Xilinx XC4010 FPGAs, reconfiguration takes about 100 ms.

4. CORDICs on the PCI Pamette

A more recent survey of CORDIC algorithms on FPGAs is given in [16]. We implement a simple coordinate rotation unit to demonstrate the power and performance benefits from CORDIC algorithms.

The following three equations are the core of a coordinate rotation unit using the CORDIC algorithm.

$$x_i = x_{i-1} - sign \cdot y_{i-1} \cdot 2^{M_i} \quad (1)$$

$$y_i = y_{i-1} + sign \cdot x_{i-1} \cdot 2^{M_i} \quad (2)$$

$$z_i = z_{i-1} - sign \cdot tan^{-1}(2^{M_i}) \quad (3)$$

Given a point (x_0, y_0) and an angle z, the CORDIC algorithm converges towards the rotated point (x, y) while $z \to 0$. This is achieved by partitioning the angle z into powers of two. The $sign$ for each round is set by the sign of z at the previous iteration, forcing z to zero.

The series of numbers M_i are usually continuous integers i.e. $[0 \ldots (N-1)]$ which have to be chosen according to the desired convergence behavior. We have confirmed the results for the convergence behavior described in [19] by simulation of the algorithm in software.

We are currently working on FPGA implementations of various CORDIC algorithms [2].

5. Acknowledgments

We are highly indebted to Prof. M.J. Flynn for his encouragements and support as the head of the Computer Architecture and Arithmetic Group at Stanford, where this work was carried out as part of the reconfigurable architecture and adaptive arithmetic effort.

6. References

[1] J.E. Volder, *The CORDIC Trigonometric Computing Technique,* IRE Trans. on Electronic Computers, Vol. EC-8, No. 3, Sept. 1959.

[2] J.S. Walther, *A Unified Algorithm for Elementary functions,* Proc. of the 1971 Spring Joint Computer Conference.

[3] M. Morf, *Fast Algorithms for Multivariable Systems,* Ph.D Thesis, E.E. Dept., Stanford, CA, Aug. 1974.

[4] P. Bertin, D. Roncin, J. Vuillemin, *Programmable Active Memories: A Performance Assessment,* ACM FPGA, February 1992.

[2]An extended version of this paper, including detailed results of the CORDIC implementations on the PCI Pamette board, will be available from the authors by the time of the workshop.

[5] Duncan A. Buell, Jeffrey M. Arnold, Walter J. Kleinfelder, *Splash-2, FPGAs in a Custom Computing Machine,* IEEE Computer Society Press, 1996

[6] T.C. Chen, *Automatic Computation of Exponentials, Logarithms, Ratios and Square Roots,* IBM Journal of Research and Development, July 1972.

[7] H.M. Ahmed, M. Morf, D.T.L. Lee and P.H. Ang, *A VLSI Speech Analysis Chip Set Based on Square-Root Normalized Ladder Forms,* Proc. of ICASSP, Atlanta, GA, March 1981.

[8] Jonathan Babb, Russell Tessier, Anant Agarwal, *Virtual Wires: Overcoming pin limitations in FPGA-based logic emulators,* Proc. IEEE Workshop on FPGAs for Custom Computing Machines, pages 142-151, Napa, CA, 1993.

[9] W.H. Mangione-Smith, B. Hutchings, D. Andrews, A. DeHon, C. Ebeling, R. Hartenstein, O. Mencer, J. Morris, K. Palem, V. Prasanna, H. Spaanenburg, *Configurable Computing,* IEEE Computer Magazine, December 1997.

[10] O. Mencer, M. Morf, M. Flynn, *Hardware Software Tri-Design of Encryption for Mobile Communication Units,* (submitted to ASSP '98).

[11] Oskar Mencer, Martin Morf, Michael J. Flynn, *PAM-Blox: High Performance FPGA Design for Adaptive Computing,* IEEE Symposium on FPGAs for Custom Computing Machines, Napa Valley, 1998. (submitted for publication)

[12] F. F. Lee with M. Flynn *A Scalable Computer Architecture for Lattice Gas Simulation,* PhD Thesis, Stanford, June 1993

[13] S. Oberman with M. Flynn, *Design Issues in High Performance Floating Point Arithmetic Units,* PhD Thesis, E.E. Dept., Stanford, Jan. 1997.

[14] F. F. Lee with M. Flynn, *A Scalable Computer Architecture For Lattice Gas Simulations,* PhD Thesis, E.E. Dept., Stanford, June 1993.

[15] W.B. Culbertson, R. Amerson, R.J. Carter, P. Kuekes, G. Snider, *Defect Tolerance on the Teramac Custom Computer,* IEEE Symposium Field-Programmable Custom Computing Machines, Napa Valley, CA, April 1997.

[16] Ray Andraka, *A Survey of CORDIC algorithms for FPGA based computers,* Sixth International Symposium on Field Programmable Gate Arrays, Monterey, CA, 1998

[17] Mark Shand, *The PCI Pamette FPGA board at DEC Systems Research Center,* http://www.research.digital.com/SRC/pamette/

[18] D.T.L. Lee and M. Morf, *Generalized Cordic for Digital Signal Processing,* Proceedings 1982 International Conference on Acoustics, Speech and Signal Processing (ICASSP), Paris, France, May 3-5, 1982, pp. 1748-1751.

[19] H.M. Ahmed with M. Morf, *Signal Processing Algorithms and Architectures,* PhD Thesis, E.E. Dept., Stanford, June 1982.

[20] E.M. Schwarz with M. Flynn, *High-Radix Algorithms for High-Order Arithmetic Operations,* PhD Thesis, E.E. Dept., Stanford, Jan. 1993.

[21] H.M. Ahmed, J.-M. Delosme and M. Morf, *Highly Concurrent Computing Structures for Matrix Arithmetic and Signal Processing,* IEEE Computer, pp. 65-82, Jan. 1982.

[22] M. Morf and J.-M. Delosme, *Matrix Decomposition and Inversions via Elementary Signature-Orthogonal Transformations,* International Symposium on Mini- and Microcomputers, San Francisco, CA, May 1981.

[23] J.-M. Delosme and M. Morf, *Algorithms and Architectures for Modern Signal Processing,* Trends and Perspectives in Signal Processing, Vol. 3, No. 2, pp. 7-10, June 1983.

[24] M. Morf, P.H. Ang and J.-M. Delosme, *Development of Signal Processing Algorithms via Functional Programming,* Proc. 1983 Int. Conf. on ASSP, Boston, MA, pp. 1176-1179, April 1983.

[25] S.-F. Hsiao and J.-M. Delosme, *CORDIC Householder Algorithms,* 10th IEEE Symposium on Computer Arithmetic, Grenoble, France, pp. 256-263, June 1991.

[26] J.-M. Delosme, *Bit-level Systolic Algorithm for the Symmetric Eigenvalue Problem,* Proc. Int. Conf. on Application Specific Array Processors, Princeton, NJ, pp. 770-781, Sept. 1990.

[27] J.-M. Delosme, *Parallel Implementations of the SVD Using Implicit CORDIC Arithmetic,* SVD and Signal Processing II: Algorithms, Analysis and Applications, R. Vaccaro Editor, Elsevier Science Publishers, pp. 33-56, 1991.

[28] J.-M. Delosme, *CORDIC Algorithms: Theory and Extensions,* Advanced Algorithms and Architectures for Signal Processing IV, Proc. SPIE 1152, pp. 131-145, Aug. 1989.

[29] J.S. Powell, J.A. Trezza, M. Morf, and J.S. Harris, Jr., *Vertical Cavity X-Modulators for Reconfigurable Optical Interconnection and Routing,* International Conference on Massively Parallel Processing Using Optical Interconnections 1996, Maui, Hawaii, October 1996.