A UNIFIED VIEW OF CORDIC PROCESSOR DESIGN

Shaoyun Wang [1] , *Vincenzo Piuri* [2] , *and Earl E. Swartzlander, Jr.* [3]

[1] Crystal Semiconductor Corporation, 4210 S. Industrial Dr., Austin, TX 78744, USA.
swang@crystal.cirrus.com

[2] Department of Electronics and Information, Politecnico di Milano, 20133 Milano, Italy.
piuri@elet.polimi.it

[3] Department of Electrical and Computer Engineering, University of Texas at Austin, Austin, TX 78712, USA.
e.swartzlander@compmail.com

ABSTRACT

CORDIC processors are powerful computing systems for applications involving large amount of rotation operations or for evaluating several elementary funcitons. In this paper, a unified view of the different architectures proposed in literatures is presented. The goal is to provide a wide spectrum of architectures, a coordinated and comprehensive design methodology, and the main figures of merit characterizing architectures' performance and complexity. This provides the basic guidelines to the designer for an optimal choice of the architectureal apporach with respect the specific requirements and constraints of the specific application.

1. INTRODUCTION

The CORDIC algorithm is a well-known and widely-studied iterative technique[1, 2] for evaluating many basic arithmetic operations and elementary functions. These functions are frequently used in applications expressed in terms of basic plane rotations.

Many papers have been published on the design of advanced architectural approaches either to enhance the performance or to reduce the circuit complexity in order to match the requirements and the constraints of the specific application[e.g. 3-5]. Several solutions were also implemented in commercial products[6-8]. The set of these structures are, however, quite differentiated and presented in a non-homogeneous way. The designers have to design and evaluate each technique separately in order to choose the most suitable one.

In this paper, a comprehensive and coordinated view of CORDIC processors is proposed. The goal is to provide a continuous and homogeneous spectrum of solutions to the designer with the related figures of merit as guidelines for choosing the architectural structure fitting application constraints and requirements. All the known and possible solutions are placed in a reference space. Two basic transformation rules on mapping the CORDIC algorithm onto a hardware dedicated architecture are identified: namely, *combining* and *unrolling*. The first one merges CORDIC steps into only one operational clock cycle. The later re-maps the operations performed by an iterative CORDIC architecture implemented as a sequential machine onto a combinatoric circuit. In the reference space, each of these transformations is associated to one dimension. The values characterizing their application are used as measures in the corresponding dimension.

This set of architectures includes not only the known architectures, but also a number of new intermediate structures that could better match application constraints and requirements on circuit complexity, latency, and throughput.

2. COMBINED ARCHITECTURES

The original CORDIC algorithm[1, 2] is a bit-recursive implementation of the forward and backward Givens rotations, also called *rotation* and *vectoring*, respectively. The i-th CORDIC iteration equations are:

$$\begin{cases} x_{i+1} = x_i - m\sigma_i 2^{-S(m,i)} y_i \\ y_{i+1} = y_i + \sigma_i 2^{-S(m,i)} x_i \\ z_{i+1} = z_i - \sigma_i \alpha_{m,i} \end{cases} \qquad i = 0,1,\ldots,N \quad (1)$$

where:

$-m \in \{-1, 0, 1\}$ is the coordinate parameter, that specifies whether the rotation is in a hyperbolic, linear, or circular coordinate system, respectively; $\sigma_i \in \{-1,+1\}$ is the rotation direction, given by $\sigma_i = sign(z_i)$ for rotation, or by $\sigma_i = -sign(x_i y_i)$ for vectoring; $2^{-S(m,i)}$ is the rotation coefficient which define the modifications of the vector coordinates induced by the current rotation iteration; $S(m,i)$ is the shift sequence that define the current value of the rotation coefficient (For example, $S(m,i)=i$ for linear coordinate systems.); $\alpha_{m,i}$ is the rotation angle given by

0-7803-3636-4/97 $10.00 © 1997 IEEE

$$\alpha_{m,i} = m^{-1/2} \arctan(m^{1/2} 2^{-S(m,i)}).$$

Due to the choice of the rotation coefficient, the iteration scaling factor $k_{m,i}$ must be introduced to restore the modulus of the vector at each iteration; it is given by $k_{m,i} = (1+m\sigma_i^2 2^{-2S(m,i)})^{1/2}$. Being the rotation directions restricted to {-1, 1}, the iteration scale factors does not depend on σ_i; therefore, normalization can be applied only once on the final results as a global scaling factor $K_m = \Pi_{i=0..N} (1+m 2^{-2S(m,i)})^{1/2}$.

The first approach to design CORDIC processors was based on direct mapping of the iterative operations of the CORDIC algorithm onto a sequential digital machine. The architecture emulated exactly the sequencing of the algorithm steps. This allowed for realizing very compact structures, but the latency required to complete one run was high since operations are strictly serial. The operation of the parallel adder/subtractors is controlled by the rotation direction. The adder/subtractor is implemented by using a traditional two-input adder for two's-complement integers: the first input is the first addend, while the second input may be either the nominal value of the second operand or its one's complement, according with the current value of the rotation direction. For the generation of the coordinates x and y, N-bit N-position shifters are required to prepare the second operands of the adder/subtractors, controlled by the number of the current iteration directly. For the generation of the coordinate z, the angle corresponding to the current iteration must be provided by using a look-up table. A dedicated control circuit (σ-generator) is used to compute the rotation direction for the current iteration, according to the current values of the accumulators X, Y, and Z, and the CORDIC operation mode.

The analysis of the characteristic figures of circuit complexity, latency, and throughput must take into account the specific structural choices for the basic building blocks composing the processor architecture, namely the adders, the shifters, and the σ-generator. The structure of this last circuit is quite simple and is dependent from the operations performed in the processor: there are basically no major alternative design strategies affecting its complexity and its performances. Shifters may be implemented by using barrel shifters or switches: since the interconnection flexibility of both of these approaches is identical, but barrel shifters have a smaller circuit complexity and a lower latency, we consider only the barrel shifters. In fact, for the same topology, they have a smaller circuit complexity and a lower latency than a network of switches. As in [3], adders may be implemented by using: ripple-carry adders, carry-save adders, carry-look-ahead adders, conditional-sum adders, an array of limited-size carry-look-ahead adders between which the carry is propagated in a ripple way, or other structures. As examples, in this paper, the processor internal structure is built by using only ripple-carry adders or the array of carry-look-ahead adders having ripple-carry propagation between the adder blocks.

Combining is a transformation rule that can be applied to the nominal CORDIC algorithm to reorganize its operations onto a dedicated hardware structure. This rule merges more steps into the same computational cycle in order to save some latency by removing some storing operation of intermediate results. We define the *order* of a combined architecture as the number of CORDIC iterations that are fused in the same clock cycle. In combined architectures, even if the clock cycle time is increased, the time per CORDIC iteration is reduced. As a consequence, the latency is also decreased and the throughput enhanced, by increasing the circuit complexity. Different kind of techniques for merging are considered and evaluated in the paper as accuracy is concerned.

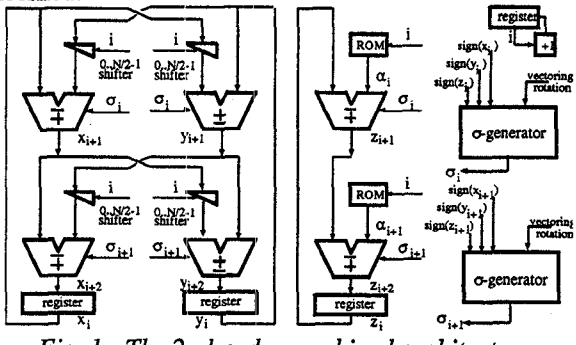

Fig. 1 - The 2nd-order combined architecture.

The resulting architecture for the 2nd-order case is given in Fig. 1. In the maximum-order case, i.e., the one having order equal to N, all CORDIC iterations are completely cascaded and mapped onto separate circuits; this is practically the architecture presented in [4]. There are as many adders for each coordinate and σ-generators as the number N of CORDIC iterations. The system becomes therefore, a pure combinatoric circuit. No accumulator is necessary. Besides, no shifter is required since each shifter should move the operand into only one fixed position, i.e., it can be hardwired. The increase of circuit complexity and the clock cycle are maximum or quite-maximum (due to possible savings mentioned above); latency is minimum, while throughput becomes maximum among all combined architectures.

3. PIPELINED ARCHITECTURES

Unrolling is a transformation rule that can be applied to each sequential architecture described in Section 2 to

reorganize its operations onto a combinatoric structure. This rule maps each operation performed by the considered combined architecture onto a dedicated unit, so that no reusing of the same unit is performed during execution of the complete CORDIC algorithm as in the combined case. Since each unrolled structure contains a register where the combined counterpart applies a storing operation into the accumulator, it is automatically a pipelined architecture. We define the *order* of an unrolled architecture as the number of CORDIC iterations that are fused in the same pipeline stage.

The circuit complexity increases rapidly since no time-multiplexing of components is exploited. Conversely, the throughput is highly increased since it becomes always equal to the inverse of the clock cycle time.

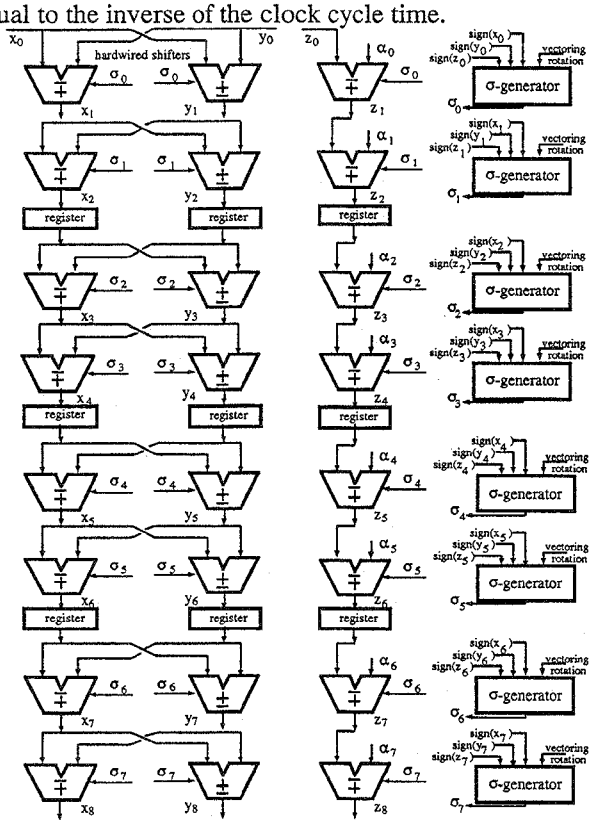

Fig. 2 - The 2nd-order pipelined architecture.

By unrolling the 1st-order combined structure, we obtain the well-known pipeline structure available in the literature. The 2nd-order pipelined architecture is given in Fig. 2. The extreme conditions are achieved when the order becomes maximum, i.e., equal to N. In this case, all CORDIC iterations are executed within the same pipeline stage: the pipeline granularity is minimum. Operations are completely cascaded and performed by different hardware components, no pipeline register is contained in the structure to separate groups of iterations: this architecture coincides exactly with the maximum-order combined structure.

4. EVALUATIONS AND DESIGN GUIDELINES

The summary of the analysis performed in the previous sections is graphically shown in Fig. 3, Fig. 4, Fig. 5, and Fig. 6 for the case of 16-bit operands and, as a consequence, of $N=16$ CORDIC iterations; similar results can be achieved for different values of N. As the circuit complexity is concerned (Fig. 3), the combined architectures have an increasing complexity as the order increases; conversely, the size of the pipelined ones progressively decreases till coinciding at the maximum order with the combined case. The clock cycle time and the latency (Figs. 4 and 5, respectively) are identical since the pipelined case is simply the unrolled version of the combined one, being the latency of the σ-generator usually higher than the other components working in parallel with it. For both the architectural approaches, the clock cycle time increases as the order increases since more CORDIC iterations must be accommodated in the same clock cycle. The latency has a non-monotonic behavior with respect to the order since additional void CORDIC iteration must be introduced in the case N is not a multiple of the order.

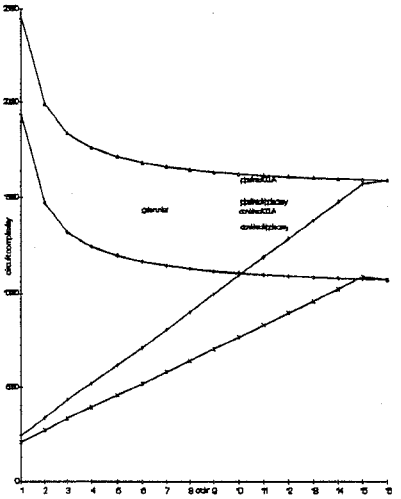

Fig. 3 - Circuit Complexity

In the paper, circuit complexity is evaluated for each architecture and for each different internal structure of the basic blocks. As a first approximation, we adopted the traditional gate count in order to give an idea of the complexity independently from the specific realization technology, in order to allows for an architectural-level selection of the most suited approach for the specific constraints. To have a good estimation of the transistor count and, as a consequence, to have a rough relative evaluation of the silicon area occupied by the circuits, we used only two-input gates in designing and evaluating the prototype structures. By using the same evaluation technique based on the (two-input) gate count, the clock cycle time as well as the latency and the throughput are derived.

Detailed comparison and discussion of all these figures of merit are also given to support the design choice s at the architectural level. The unified view of the architectural solutions presented in the paper as a continuous wide spectrum of possible alternatives allows for an optimum choice of the architecture by taking into account the application constraints and the requirements on precision, circuit complexity, latency, and throughput, contemporaneously.

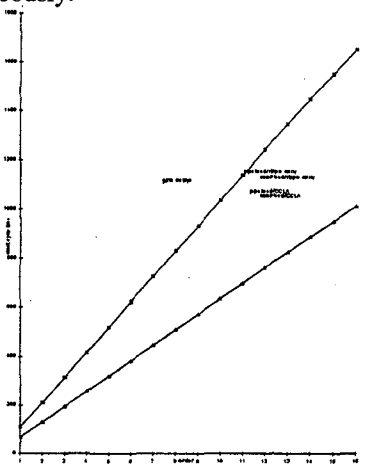

Fig. 4 - Clock Cycle Time

For the given application, the designer can derive the minimum throughput which is sufficient to deliver the results for subsequent operations, in particular when massive-computing applications are envisioned. He can also evaluate the maximum latency, which is relevant in several control and robotics applications.

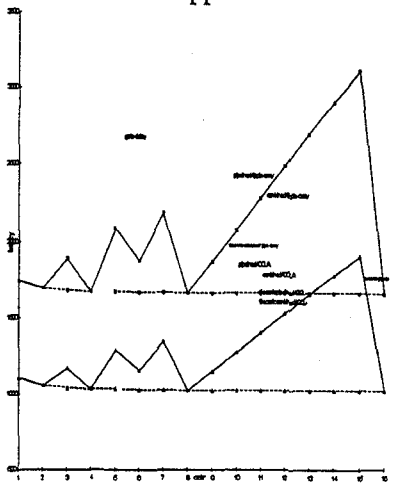

Fig. 5 - Latency

When integration of the architecture in a VLSI/ULSI/WSI device is constrained by the size of the processor or by the power consumed, the maximum circuit complexity may be identified as a high-level indicator of the circuit silicon area or of the power consumption. The clock cycle time may be lower bounded by the specific integration technology, according to the characteristic behavior of transmission delays.

With the actual values of these constraints, in the three-dimensional architectures' space introduced in this paper, the designer can identify the solutions satisfying the constraints. Among these solutions (if existing), he can chose the one which optimizes the most relevant figure of merit for the specific application, or the one that best balances two or more of these figures.

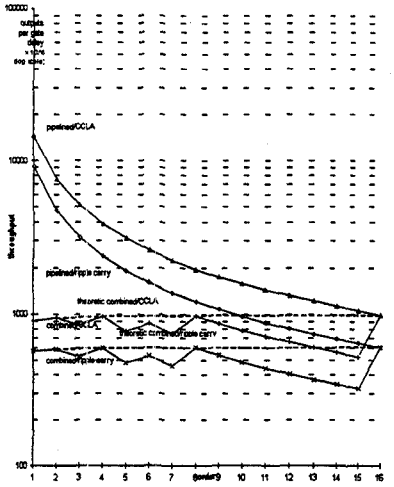

Fig. 6 - Throughput

When a constraint on the maximum latency is given, there are solutions with the same order both in the combined part and in the pipelined one, since practically both of these classes have the same latency. Similarly for the clock cycle time.

REFERENCES

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

[2] J. S. Walther, "A Unified Algorithm for Elementary Functions", *Proc. 1971 Spring Joint Computer Conference*, vol. 38, pp. 379-385, 1971.

[3] A. A. J. de Lange and E. F. Deprettere, "Design and Implementation of a Floating-Point Quasi-Systolic General Purpose CORDIC Rotator for High-Rate Parallel Data and Signal Processing", *10th Symp. on Comput. Arith.*, pp. 272-281, 1991.

[4] D. Timmermann, H. Hahn, and B. J. Hosticka, "Low Latency Time CORDIC Algorithms", *IEEE Trans. on Comput.*, Vol. 41, pp.1010-1015, 1992.

[5] S. Wang and E. E. Swartzlander, Jr., "Merged CORDIC Algorithm", *Proc. of ISCAS 95*, pp. 1988-1991, 1995.

[6] D. S. Cochran, "Algorithms and Accuracy in the HP-35", *Hewlett-Packard Journal*, Vol. 10, No. 11, pp. 10-11, 1972.

[7] G. L. Haviland and A. A. Tuszynski, "A CORDIC Arithmetic Processor Chip", *IEEE Trans. on Comput.*, Vol. 29, pp. 68-79, 1980.

[8] F. Williams, "The CORDIC Algorithm - Cast in Silicon", *Electronic Engineering*, Vol. 61, pp. 47-50, 1989.