A FIXED-POINT IMPLEMENTATION OF THE EXPANDED HYPERBOLIC CORDIC ALGORITHM

D. R. LLAMOCCA-OBREGÓN† and C. P. AGURTO-RÍOS†

† *Grupo de Procesamiento Digital de Señales, Pontificia Universidad Católica del Perú, Av. Universitaria cdra 18, Lima 32- Perú. llamocca.dr@pucp.edu.pe; agurto.cp@pucp.edu.pe*

***Abstract*— The original hyperbolic CORDIC (Coordinate Rotation Digital Computer) algorithm (Walther, 1971) imposes a limitation to the inputs' domain which renders the algorithm useless for certain applications in which a greater range of the function is needed. To address this problem, Hu *et al.* (1991) have proposed an interesting scheme which increments the iterations of the original hyperbolic CORDIC algorithm and allows an efficient mapping of the algorithm onto hardware. A fixed-point implementation of the hyperbolic CORDIC algorithm with the expansion scheme proposed by Hu *et al.* (1991) is presented. Three architectures are proposed: a low cost iterative version, a fully pipelined version, and a bit serial iterative version. The architectures were described in VHDL, and to test the architecture, it was targeted to a Stratix FPGA. Various standard numerical formats for the inputs are analyzed for each hyperbolic function directly obtained: *Sinh, Cosh, Tanh*$^{-1}$ *and exp*. For each numerical format and for each hyperbolic function an error analysis is performed.**

I. INTRODUCTION

The hyperbolic CORDIC algorithm as originally proposed by Walther (1971) allows the computation of hyperbolic functions in an efficient fashion. However, the domain of the inputs is limited in order to guarantee that outputs converge and yield correct values, and this limitation will not satisfy the applications in which nearly the full range of the hyperbolic functions is needed.

Various strategies have been proposed to address the problem of limited convergence of the hyperbolic CORDIC algorithm. One strategy is to use mathematical identities to preprocess the CORDIC input quantities (Walther, 1971). While such mathematical identities work, there is no single identity that will remove or reduce the limitations of all the functions in the hyperbolic mode. In addition, the mathematical identities are cumbersome to use in hardware applications because their implementation requires a significant increase in processing time and hardware (Hu *et al*. 1991). Another approach, proposed by Hu *et al*. (1991), involves a modification to the basic CORDIC algorithm (inclusion of additional iterations) that can be readily implemented in a VLSI architecture or in a FPGA without excessively increasing the processing time.

Three architectures for the fixed-point implementation of the hyperbolic CORDIC algorithm with the expansion scheme proposed by Hu *et al*. (1991) are presented: a low cost iterative version, a fully pipelined version, and a bit serial iterative version. Results in terms of resource count and speed were obtained by targeting the architectures, described in VHDL, to a Stratix FPGA of ALTERA®.

Four different numerical formats are proposed for the inputs. For each hyperbolic function, an analysis of each numerical format is performed and the optimal number of iterations along with the optimal format for the angle are obtained. Finally, an error analysis is performed for each hyperbolic function with each numerical format. The data obtained with the fixed-point architectures are contrasted with the ideal values obtained with MATLAB®.

II. EXPANSION SCHEME FOR THE HYPERBOLIC CORDIC ALGORITHM

A. Original Hyperbolic CORDIC algorithm

The original hyperbolic CORDIC algorithm, first described by Walther (1971), states the following iterative equations:

$$\begin{aligned} X_{i+1} &= X_i + \delta_i Y_i 2^{-i} \\ Y_{i+1} &= Y_i + \delta_i X_i 2^{-i} \\ Z_{i+1} &= Z_i - \delta_i \theta_i \end{aligned} \tag{1}$$

$$\text{Where: } \theta_i = Tanh^{-1}\left(2^{-i}\right) \tag{2}$$

And i is the index of the iteration (i=1,2,3,…N). The following iterations must be repeated in order to guarantee the convergence: *4, 13, 40,… k, 3k + 1*. The value of δ_i is either +1 or –1 depending on the mode of operation:

$$\begin{aligned} &Rotation: \quad \delta_i = -1 \quad if\ z_i < 0, +1, otherwise \\ &Vectoring: \quad \delta_i = -1 \quad if\ x_i y_i \geq 0, +1, otherwise \end{aligned} \tag{3}$$

In the rotation mode, the quantities X, Y and Z tend to the following results, for sufficiently large N:

$$\begin{aligned} X_n &\leftarrow A_n\left[X_0 CoshZ_0 + Y_0 SinhZ_0\right] \\ Y_n &\leftarrow A_n\left[Y_0 CoshZ_0 + X_0 SinhZ_0\right] \\ Z_n &\leftarrow 0 \end{aligned} \tag{4}$$

And, in the vectoring mode, the quantities X, Y and Z tend to the following results, for sufficiently large N:

$$\begin{aligned} X_n &\leftarrow A_n\sqrt{X_0{}^2 - Y_0{}^2} \\ Y_n &\leftarrow 0 \\ Z_n &\leftarrow Z_0 + Tanh^{-1}\left(Y_0 / X_0\right) \end{aligned} \tag{5}$$

$$\text{Where '}A_n\text{' is: } A_n \leftarrow \prod_{i=1}^{N}\sqrt{1 - 2^{-2i}} \tag{6}$$

With a proper choice of the initial values X_0, Y_0, Z_0 and the operation mode, the following functions can be