Microprocessors and Microsystems 34 (2010) 96–101

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Architectural design and FPGA implementation of radix-4 CORDIC processor

Kaushik Bhattacharyya *, Rakesh Biswas, Anindya Sundar Dhar, Swapna Banerjee

Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721 302, WB, India

ARTICLE INFO

Article history:
Available online 20 January 2010

Keywords:
Architectural design
CORDIC
FPGA implementation
Latency
Radix-4
Speed
Throughput

ABSTRACT

A new scaled radix-4 CORDIC architecture that incorporates pipelining and parallelism is presented. The latency of the architecture is $n/2$ clock cycles and throughput rate is one valid result per $n/2$ clocks for n bit precision. A 16 bit radix-4 CORDIC architecture is implemented on the available FPGA platform. The corresponding latency of the architecture is eight clock cycles and throughput rate is one valid result per eight clock cycles. The entire scaled architecture operates at 56.96 MHz of clock rate with a power consumption of 380 mW. The speed can be enhanced with the upgraded version of FPGA device. A speed-area optimized processor is obtained through this architecture and is suitable for real time applications.

1. Introduction

Co-Ordinate Rotation Digital Computer (CORDIC) algorithm was originally proposed by Volder [1] in 1959 and later generalized by Walther [2] in 1971 for computation of logarithms, exponentials and square-root functions along with the trigonometric functions like sine, cosine and arctangent. It is an iterative algorithm for the calculation of the rotation of a two-dimensional vector in linear, circular and hyperbolic coordinate systems. The rotation is carried out by means of a sequence of iterations in each of which one rotation over a prefixed elementary angle (micro rotation) is evaluated by means of addition and shift operations. The rotated vector is scaled by a constant factor that must be compensated [3]. The number of iteration of conventional radix-2 CORDIC architecture limits its architecture to use in high-speed application. So the development of high-radix CORDIC algorithm is essential for reducing the number of iterations i.e., the reduction of latency time of CORDIC calculation with a reasonable increment of hardware complexity. The proper control of the tradeoff between latency time and the hardware complexity enable a designer to use higher radix CORDIC unit [4] in place of the conventional radix-2 CORDIC unit in high-speed application purposes [5]. In 1996, Antelo et al. [6] proposed a high-radix CORDIC in an effort to reduce the number of iterations. With the increment of radix, the number of iterations for a given precision is reduced, resulting in a potentially faster execution. However, two problems appear: the complexity of the selection function and the compensation of a variable scale factor. Although the full radix-4 algorithm is efficient compared to the standard radix-2 version, the scale factor overhead causes its improvement to be limited. The minimization of the computation overhead of the scale factor compensation has been attempted by different researchers [7–13].

In this paper, an existing radix-4 CORDIC algorithm is taken into account for the hardware implementation. Here the rotation is being completed within five clock cycles for 16 bit precision. The Scale factor computation is carried out in parallel to the rotation. The architecture possesses some specialties. The entire architecture is implemented in FPGA platform. Finally the compensated output with 32 bit output bit precision is available after eight clock cycle.

The radix-4 CORDIC processor design has been detailed in the subsequent sections of this paper. Section 2 is dedicated for the recapitulation of the radix-4 CORDIC algorithm in rotation mode along with parallel scale factor computation to minimize the latency time. Section 3 provides a detail of the proposed architecture of the compensated radix-4 CORDIC unit. Section 4 is basically for the implementation and discussion of the designed CORDIC processor in FPGA environment. Section 5 deals with the performance evaluation of the implemented architecture with respect to the published references. Finally some concluding remarks are provided in Section 6.

2. Radix-4 CORDIC algorithm

When a radix-4 is used instead of radix-2, the total number of iterations of the CORDIC algorithm is halved. The iterative equations of the CORDIC algorithm are extended to radix-4 [6] as follows:

* Corresponding author. Tel.: +91 9434252606.
E-mail address: kaushik@ece.iitkgp.ernet.in (K. Bhattacharyya).

0141-9331/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.micpro.2010.01.002

The radix-4 CORDIC equations are given below:

$$x_{i+1} = x_i - \sigma_i 4^{-i} y_i,\ y_{i+1} = y_i + \sigma_i 4^{-i} x_i,\ z_{i+1} = z_i - \alpha_i[\sigma_i] \tag{1}$$

where $\sigma_i \in \{-2,-1,0,1,2\}$, $\alpha_i[\sigma_i] = \tan^{-1}(\sigma_i r^{-i})$. Here r is the radix of the CORDIC. x_0 and y_0 are the coordinates of the initial vector and z_0 is the rotation angle. Consequently, x_{i+1}, y_{i+1} are the coordinates of the vector resulting from applying $i+1$ micro rotation and z_{i+1} is the angle remaining to be rotated. The final coordinates are scaled by

$$K^{-1} = \prod_{i \geqslant 0} k_i = \prod_{i \geqslant 0} \left(1 + \sigma_i^2 4^{-2i}\right)^{-1/2} \tag{2}$$

The scale factor is not constant as it depends on the sequence of σ_i. In the case of the radix-4 algorithm, this value ranges from $K = 1.0$ to $K = 2.52$ [9].

Now, we may write a variable w_i as,

$$w_i = 4^i z_i \tag{3}$$

The Eq. (3) may be needed to prove that the variable z is bounded in each of the rotations.

The selection interval for different values of iteration (1) can be obtained [6].

For $i = 0$,

$$\sigma_0 = \begin{cases} +2 & \text{if} \quad 5/8 \leqslant \widehat{W}_0 \\ +1 & \text{if} \quad 3/8 \leqslant \widehat{W}_0 < 5/8 \\ 0 & \text{if} \quad -1/2 \leqslant \widehat{W}_0 < 3/8 \\ -1 & \text{if} \quad -7/8 \leqslant \widehat{W}_0 < -1/2 \\ -2 & \text{if} \quad \widehat{W}_0 < -7/8 \end{cases} \tag{4}$$

and for $i > 0$,

$$\sigma_i = \begin{cases} +2 & \text{if} \quad \widehat{W}_i \geqslant 3/2 \\ +1 & \text{if} \quad 1/2 \leqslant \widehat{W}_i < 3/2 \\ 0 & \text{if} \quad -1/2 \leqslant \widehat{W}_i < 1/2 \\ -1 & \text{if} \quad -3/2 \leqslant \widehat{W}_i < -1/2 \\ -2 & \text{if} \quad \widehat{W}_i < -3/2 \end{cases} \tag{5}$$

The intervals verify the continuity condition. Here, $\widehat{W}_i$ is the estimate of w_i with three most significant bits. The selection function described is valid both for carry-save redundant arithmetic and non redundant arithmetic [14].

2.1. Scale factor

In radix-4 algorithm, the traditional scaling iteration technique cannot be used as in radix-2 CORDIC processors, where it is to be done by a direct multiplication. The procedure to calculate the scale factor is as follows. The computation of the scale factor is carried out in parallel and will be completed by the first $(n/4+1)$ iterations. If the iterative steps following the calculation of the scale factor are observed, then a possibility for parallel scale factor compensation can be explored.

In this paper, the inverse of the scale factor (K^{-1}) is computed in parallel for $(n/4+1)$ iterations. After $(n/4+1)$ iteration, it is multiplied with the unscaled X and Y coordinates to achieve the scaled radix-4 CORDIC output. Through the concurrent scale factor compensation procedure adopted in place of compensation in linear mode [6], it has been possible to reduce the latency time. The architectural design for implementing the algorithm is described in the next section.

3. Proposed architecture [15]

This section presents the proposed architecture of the iterative radix-4 CORDIC processor. As the widths of the data paths and registers in the majority of computer architectures happen to be multiples of 16, so the proposed scheme is designed throughout for 16 bit precision. Although the unscaled rotation and compensation part presented in this architecture is valid for 16 bit precision, it can be translated into the 32 bit iterative architecture by simply extending the word length from 16 to 32.

The proposed compensated architecture of the radix-4 CORDIC processor is comprised of the 'Unscaled CORDIC architecture' and the 'K^{-1}computation architecture'. Basically in this architecture there is a circular mode of operations for eight iterations. The scale factor computation part is also operating concurrently to generate the reciprocal of the scale factor, K^{-1} within five cycles (i.e., $n/4+1$). After five micro rotations, the two unscaled coordinate values are compensated by the already computed inverse of the scale factor.

3.1. Unscaled CORDIC architecture

The Unscaled CORDIC architecture (Fig. 2) consists of a pre-processing unit where an initial $\pi/2$ rotation can be carried out in order to obtain a convergence range of $[-\pi,\pi]$. Basically it consists of three processing paths for the x, y, w coordinates and a selection function path. During cycle 0, the pre-processing unit carries out a rotation of $\pi/2$ and by means of MUXes 1, 2 and 3, the 16 bit values of BX, BY, Z are registered. There it consumes an extra clock cycle to load the three values from out side of the CORDIC processor. The KCKT4 block is the in charge of generating extra delay of one clock period at the 0th count of the MOD 8 counter. The output values of the registers are bx1, by1 and bz1.There are two hard right shifters viz. Rshifter1 and Rshifter2. The count value at each clock edge of the MOD 8 counter is fed to the KCKT3 block which is generating a 4 bit value (count_out) at each clock edge. This 4 bit count_out will work as the shift controller of all hard shifters. The barrel shifters carry out a right shift of $d(d = 2 \times \text{iteration})$ of the registered outputs, bx2 and by2 which are 0 in this case (factor of 4° in Eq. (1)), and those will be incremented in every cycle filling the vacant left hand side with sign value. On the other hand the Z coordinate is left shifted by the 16 bit left shifter (L shifter) by an amount of $e(e = 2 \times \text{iteration})$ which is also 0 in the 0th iteration

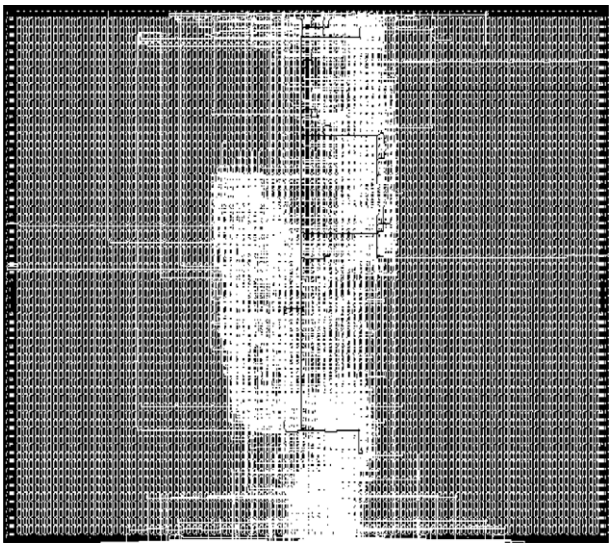

Fig. 1. Layout of the scaled radix-4 CORDIC in XCV1000BG560 FPGA device.