Hardware Implementation of High-Throughput 3-D Rotation for Graphic Engine Using Double Rotation CORDIC Algorithm

宋志雲
Tze-Yun Sung
Department of Microelectronics Engineering

陳志炘
Chih-Sin Chen
Institute of Engineering Science

Chung Hua University
Hsinchu, Taiwan 300-12
bobsung@chu.edu.tw

Abstract

High performance architectures can be design for data intensive and latency tolerant applications by maximizing the parallelism and pipelining at the algorithm. The hardware primitives for 3-D rotation for high throughput 3-D graphics and animation are presented in this paper. The primitives are based on the 2-D CORDIC algorithm, in contrast to conventional hardware for graphic engine. The accelerated architecture of the 3-D rotation based on double rotation CORDIC algorithm is also presented in this paper. The throughput is improved by more than 30%, but the additional hardware is required by less than 40%. The 3-D central perspective method for graphic engine is performed by double rotation CORDIC processors. The throughput is also improved by more than 30%.

Keywords: 3-D rotation, double rotation CORDIC algorithm, graphic engine, 3-D perspective method, high-throughput.

1. Introduction

Three dimensional rotation (3-D) is utilized in 3-D graphics, animation, and virtual reality applications [1] [2]. The rotations are applied to large number of points, which need quiet time consuming, but can be effectively parallel and pipelined. Moreover, 3-D computer hardware has been receiving great attention recently. The conventional hardware for 3-D rotation consists mainly of multipliers and accumulators.

The CORDIC algorithm [3] [4] is widely recognized as well-suited for hardware implementation and is applied to many signal processing tasks, such as sine and cosine generation, vector rotation, coordinate transformation and linear system solver. This algorithm is especially suitable for implementation of 3-D rotation. The CORDIC requires only shifters and adders, its realization on reconfigurable hardware platforms, especially on FPGA [5]. Thus, the 3-D rotation algorithm required in 3-D graphics can be realized with vector rotation, the CORDIC could be mainly used in this function block [6].

In this paper, the architecture of 3-D rotation with CORDIC algorithm is proposed, the proposed architecture is very suitable for VLSI implementation, and the computation complexity is also evaluated. The introduction of the new concept, double rotation CORDIC algorithm, improves throughput in the 3-D rotation, by up to 30% without any noticeable error occurrence. The view of observer in 2-D display system is performed by the 3-D central perspective method [7], the architecture of that is performed by 2-D CORDIC processors.

The remainder of the paper is organized as follows. Section 2 reviews the 2-D CORDIC algorithm; section 3 presents the algorithm of CORDIC rotation in 3-D space, section 4 presents the double rotation CORDIC algorithm, the 3-D rotation with double CORDIC rotation algorithm is proposed in section 5, the 3-D central perspective method performed by CORDIC algorithm is proposed in section 6, VLSI architectures of 3-D rotation and perspective are described in section 7, The impact of new algorithms and architectures is presented and analyzed in section 8, and finally, the conclusion is given in section 9.

2. The 2-D CORDIC Algorithm

CORDIC (COordinate Rotation DIgital Computer) is an algorithm for performing a sequence of iteration computations using coordinate rotation [3] [4]. This algorithm can generate some powerful elementary functions realized only by a simple set of adders and shifters. The basic CORDIC iteration equations are

$$x_{i+1} = x_i - m\sigma_i 2^{-s(m,i)} y_i \quad (1)$$

$$y_{i+1} = y_i + \sigma_i 2^{-s(m,i)} x_i \quad (2)$$

$$z_{i+1} = z_i - \sigma_i \alpha_{m,i} \quad (3)$$

where m identifies circular (m=1), linear (m=0), and hyperbolic (m=-1) coordinate systems, i=0, 1,2,…,n-1,

$$s(m,i) = \begin{cases} 0,1,2,3,4,5,...., & m=1 \\ 1,2,3,4,5,6,...., & m=0 \\ 1,2,3,4,4,5,...., & m=-1 \end{cases}$$

$$\alpha_{m,i} = m^{-1/2} \tan^{-1}[\sqrt{m} 2^{-s(m,i)}] \quad (4)$$

the rotation σ_i for rotation mode $(z_n \to 0)$ is $\sigma_i = sign(z_i)$, while for vectoring mode $(y_n \to 0)$, it is $\sigma_i = -sign(x_i) \cdot sign(y_i)$.

Table 1 shows the elementary functions that can be evaluated by the CORDIC algorithm. For the i-th iteration, a scale factor becomes $k_{m,i} = \sqrt{1 + m\sigma_i^2 2^{-2s(m,i)}}$. After n iterations, the product of all the scale factors is

$$K_m = \prod_{i=0}^{n} k_{m,i} = \prod_{i=0}^{n} \sqrt{1 + m\sigma_i^2 2^{-2s(m,i)}} = \prod_{i=0}^{n} \sqrt{1 + m 2^{-2s(m,i)}} \quad (5)$$

where the rotation directions are defined to $\sigma_i = \{-1,+1\}$.

3. CORDIC Rotation in Three-Dimensional Space

A vector R in three dimensional space is shown in Fig. 1. It has Cartesian coordinates (X_i, Y_i, Z_i) and spherical coordinates (R_i, θ_i, ϕ_i). The vector R can be rotated to become a new vector S which has cartesian coordinates $(X_{i+1}, Y_{i+1}, Z_{i+1})$ and spherical coordinates $(R_i, \theta_i + \alpha_i, \phi_i + \beta_i)$ [8]. The relationship between the Cartesian coordinates and spherical coordinates of R and S are derived as follows:

$$X_i = R_i \cos\theta_i \sin\phi_i \quad (6)$$

$$Y_i = R_i \sin\theta_i \sin\phi_i \quad (7)$$

$$Z_i = R_i \cos\phi_i \quad (8)$$

$$X_{i+1} = R_i \cos(\theta_i + \alpha_i)\sin(\phi_i + \beta_i) \quad (9)$$

$$Y_{i+1} = R_i \sin(\theta_i + \alpha_i)\sin(\phi_i + \beta_i) \quad (10)$$

$$Z_{i+1} = R_i \cos(\phi_i + \beta_i) \quad (11)$$

The eqs. (9), (10) and (11) are expanded, we can get

$$\begin{aligned} X_{i+1} &= R_i(\cos\theta_i \cos\alpha_i - \sin\theta_i \sin\alpha_i) (\sin\phi_i \cos\beta_i + \cos\phi_i \sin\beta_i) \\ &= R_i \cos\theta_i \sin\phi_i \cos\alpha_i \cos\beta_i + R_i \cos\theta_i \cos\phi_i \cos\alpha_i \sin\beta_i \\ &\quad - R_i \sin\theta_i \sin\phi_i \sin\alpha_i \cos\beta_i - R_i \sin\theta_i \cos\phi_i \sin\alpha_i \sin\beta_i \\ &= X_i \cos\alpha_i \cos\beta_i + U_i \cos\alpha_i \sin\beta_i - Y_i \sin\alpha_i \cos\beta_i - V_i \sin\alpha_i \sin\beta_i \end{aligned} \quad (12)$$

$$Y_{i+1} = Y_i \cos\alpha_i \cos\beta_i + V_i \cos\alpha_i \sin\beta_i + X_i \sin\alpha_i \cos\beta_i + U_i \sin\alpha_i \sin\beta_i \quad (13)$$

$$Z_{i+1} = Z_i \cos\beta_i - W_i \sin\beta_i \quad (14)$$

where the U_i, V_i and W_i are defined as follows:

$$U_i = R_i \cos\theta_i \cos\phi_i \quad (15)$$

$$V_i = R_i \sin\theta_i \cos\phi_i \quad (16)$$

$$W_i = R_i \sin\phi_i \quad (17)$$

Similarly, the U_{i+1}, V_{i+1} and W_{i+1} are derived as follows:

$$U_{i+1} = U_i \cos\alpha_i \cos\beta_i - X_i \cos\alpha_i \sin\beta_i - V_i \sin\alpha_i \cos\beta_i + Y_i \sin\alpha_i \sin\beta_i \quad (18)$$

$$V_{i+1} = V_i \cos\alpha_i \cos\beta_i - Y_i \cos\alpha_i \sin\beta_i + U_i \sin\alpha_i \cos\beta_i - X_i \sin\alpha_i \sin\beta_i \quad (19)$$

$$W_{i+1} = W_i \cos\beta_i + Z_i \sin\beta_i \quad (20)$$

According to eqs. (6), (7) and (8) of the CORDIC algorithm, the eqs. (12), (13), (14), (18), (19) and (20) can be split into a set of CORDIC rotations and become as follows:

$$U_{i+1}=\frac{1}{k_i^2}(U_i-X_i\rho_i 2^{-i}-V_i\delta_i 2^{-i}+Y_i\delta_i\rho_i 2^{-2i}) \quad (21)$$

$$V_{i+1}=\frac{1}{k_i^2}(V_i-Y_i\rho_i 2^{-i}+U_i\delta_i 2^{-i}-X_i\delta_i\rho_i 2^{-2i}) \quad (22)$$

$$W_{i+1}=\frac{1}{k_i}(W_i+Z_i\rho_i 2^{-i}) \quad (23)$$

$$X_{i+1}=\frac{1}{k_i^2}(X_i+U_i\rho_i 2^{-i}-Y_i\delta_i 2^{-i}-V_i\delta_i\rho_i 2^{-2i}) \quad (24)$$

$$Y_{i+1}=\frac{1}{k_i^2}(Y_i+V_i\rho_i 2^{-i}+X_i\delta_i 2^{-i}+U_i\delta_i\rho_i 2^{-2i}) \quad (25)$$

$$Z_{i+1}=\frac{1}{k_i}(Z_i-W_i\rho_i 2^{-i}) \quad (26)$$

where

$$\cos\alpha_i=\frac{1}{\sqrt{1+2^{-2i}}} \quad (27)$$

$$\sin\alpha_i=\frac{\delta_i 2^{-i}}{\sqrt{1+2^{-2i}}} \quad (28)$$

$$\cos\beta_i=\frac{1}{\sqrt{1+2^{-2i}}} \quad (29)$$

$$\sin\beta_i=\frac{\rho_i 2^{-i}}{\sqrt{1+2^{-2i}}} \quad (30)$$

$$k_i=\sqrt{1+2^{-2i}} \quad (31)$$

In the two-dimensional CORDIC algorithm, we choose $\alpha_i=\delta_i\tan^{-1}2^{-i}$ and $\beta_i=\rho_i\tan^{-1}2^{-i}$, where δ_i and ρ_i are $\in\{-1,1\}$.

The eqs. (21) and (22) can be written in the form of matrix multiplications as follows:

$$\begin{bmatrix}U_{i+1}\\V_{i+1}\end{bmatrix}=\frac{1}{k_i^2}\left\{\begin{aligned}&\begin{bmatrix}1 & -\delta_i 2^{-i}\\ \delta_i 2^{-i} & 1\end{bmatrix}\begin{bmatrix}U_i\\V_i\end{bmatrix}\\ &-\rho_i 2^{-i}\cdot\begin{bmatrix}1 & -\delta_i 2^{-i}\\ \delta_i 2^{-i} & 1\end{bmatrix}\begin{bmatrix}X_i\\Y_i\end{bmatrix}\end{aligned}\right\} \quad (32)$$

Similarly, the eqs. (24) and (25) can be written in the form of matrix multiplications as follows:

$$\begin{bmatrix}X_{i+1}\\Y_{i+1}\end{bmatrix}=\frac{1}{k_i^2}\left\{\begin{aligned}&\begin{bmatrix}1 & -\delta_i 2^{-i}\\ \delta_i 2^{-i} & 1\end{bmatrix}\begin{bmatrix}X_i\\Y_i\end{bmatrix}\\ &+\rho_i 2^{-i}\cdot\begin{bmatrix}1 & -\delta_i 2^{-i}\\ \delta_i 2^{-i} & 1\end{bmatrix}\begin{bmatrix}U_i\\V_i\end{bmatrix}\end{aligned}\right\} \quad (33)$$

According to eqs. (32) and (33), we find that there are four 2-dimensional CORDIC rotations in the 3-dimensional rotation. Nevertheless, the scale factor of Z_{i+1} and W_{i+1} is different from that of U_{i+1}, V_{i+1}, X_{i+1} and Y_{i+1}, we can prescale the inputs or post scale the outputs by the constant scale factor K for Z_{i+1} and W_{i+1}, and K^2 for U_{i+1}, V_{i+1}, X_{i+1} and Y_{i+1}, where

$$K=\prod_{i=0}^{n-1}k_i \quad (34)$$

$$K^2=\prod_{i=0}^{n-1}k_i^2 \quad (35)$$

4. Double Rotation 2-D CORDIC Algorithm and Architecture

The basic concept of the accelerated CORDIC algorithm is to reduce the iterations. The double rotation CORDIC algorithm is developed to reduce the iterations or computation time [9] [10]. The double rotation CORDIC iteration equations should be derived and the computation complexity should be also evaluated.

The CORDIC iteration equations in a circular coordinate system are also written in the form of matrix multiplications.

$$\begin{bmatrix}x_{2i+1}\\y_{2i+1}\end{bmatrix}=\begin{bmatrix}1 & -\sigma_{2i}2^{-2i}\\ \sigma_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}x_{2i}\\y_{2i}\end{bmatrix} \quad (36)$$

$$\begin{bmatrix}x_{2i+2}\\y_{2i+2}\end{bmatrix}=\begin{bmatrix}1 & -\sigma_{2i+1}2^{-(2i+1)}\\ \sigma_{2i+1}2^{-(2i+1)} & 1\end{bmatrix}\begin{bmatrix}x_{2i+1}\\y_{2i+!}\end{bmatrix} \quad (37)$$

According to eqs. (6) and (7), we obtain

$$x_{2i+2}=(1-\sigma_{2i}\sigma_{2i+1}2^{-(4i+1)})x_{2i}-(\sigma_{2i}2^{-2i}+\sigma_{2i+1}2^{-(2i+1)})y_{2i} \quad (38)$$

$$y_{2i+2}=(\sigma_{2i}2^{-2i}+\sigma_{2i+1}2^{-(2i+1)})x_{2i}+(1-\sigma_{2i}\sigma_{2i+1}2^{-(4i+1)})y_{2i} \quad (39)$$

$$z_{2i+2}=z_{2i}-\sigma_{2i}\tan^{-1}2^{-2i}-\sigma_{2i+1}\tan^{-1}2^{-(2i+1)} \quad (40)$$

The additional computation complexity of parallel processing for eqs. (38) and (39) is two carry-save additions ((3,2)CSAs) and one shift for each iteration. In n-bit operand system, while $i\geq\frac{n}{4}-1$, eqs. (38) and (39) becomes

$$x_{2i+2}=x_{2i}-(\sigma_{2i}2^{-2i}+\sigma_{2i+1}2^{-(2i+1)})y_{2i} \quad (41)$$

$$y_{2i+2}=(\sigma_{2i}2^{-2i}+\sigma_{2i+1}2^{-(2i+1)})x_{2i}+y_{2i} \quad (42)$$

Thus, the additional computation complexity of parallel processing is one (3,2)CSA and one shift for each iteration.

The basic intention to realize the double rotation CORDIC algorithm is to generate more σ values in each step. Now, the proposed architecture requires two σ values in each step. The σ -value prediction algorithm is described as below:

σ_{2i} is determined by sign of $z(2i)$, and three equations for determining z(2i+2) are defined as

$$z_1(2i+2) = z(2i) - \sigma_{2i}(\tan^{-1} 2^{-2i} + \tan^{-1} 2^{-(2i+1)}) \quad (43)$$

$$z_2(2i+2) = z(2i) - \sigma_{2i}(\tan^{-1} 2^{-2i} - \tan^{-1} 2^{-(2i+1)}) \quad (44)$$

$$z_3(2i+2) = z(2i) - \sigma_{2i} \tan^{-1} 2^{-2i} \quad (45)$$

The flowchart for the σ_{2i+1}-prediction and $z(2i+2)$ determination algorithm is illustrated in Fig. 2, detailed flowcharts for specific cases are illustrated in Fig. 3 and 4, respectively. Now, the σ_{2i+1}-prediction and $z(2i+2)$ determination algorithm is analyzed and developed, this algorithm is simple and easy to implement on hardware. Thus, the algorithm is very suited to VLSI implementation. The determination circuit of σ_{2i+1} and z(2i+2) is shown in Fig. 5. The series constants of $(\tan^{-1} 2^{-2i} + \tan^{-1} 2^{-(2i+1)})$, $(\tan^{-1} 2^{-2i} - \tan^{-1} 2^{-(2i+1)})$ and $(\tan^{-1} 2^{-2i})$ are stored in ROM and the size of ROM is $\frac{3}{2}n$ words. The accelerated CORDIC architecture with the rotation mode in the circular coordinate system is shown in Fig. 6. In this architecture, the (4,2) carry-save adder (CSA) and carry-propagation adder (CPA) consists of two three-input, and two-output (3,2) carry-save adders/subtractors and one carry-look-ahead adder (CLA).

5. Accelerated 3-D Rotation Using the Double Rotation 2-D CORDIC Algorithm

The basic concept of the accelerated 3-D rotation is to reduce the iterations. The double rotation CORDIC algorithm [10] is applied to reduce the iterations or computation time. The 3-D double rotation iteration equations are derived and the computation complexity is also evaluated.

The 3-D rotation equations are also written in the form of matrix multiplications as follows:

$$\begin{bmatrix} U_{2i+1} \\ V_{2i+1} \end{bmatrix} = \frac{1}{k_{2i}^2} \left\{ \begin{array}{l} \begin{bmatrix} 1 & -\delta_{2i} 2^{-2i} \\ \delta_{2i} 2^{-2i} & 1 \end{bmatrix} \begin{bmatrix} U_{2i} \\ V_{2i} \end{bmatrix} \\ -\rho_{2i} 2^{-2i} \cdot \begin{bmatrix} 1 & -\delta_{2i} 2^{-2i} \\ \delta_{2i} 2^{-2i} & 1 \end{bmatrix} \begin{bmatrix} X_{2i} \\ Y_{2i} \end{bmatrix} \end{array} \right\} \quad (46)$$

$$W_{2i+1} = \frac{1}{k_{2i}} (W_{2i} + \rho_{2i} 2^{-2i} Z_{2i}) \quad (47)$$

$$\begin{bmatrix} X_{2i+1} \\ Y_{2i+1} \end{bmatrix} = \frac{1}{k_{2i}^2} \left\{ \begin{array}{l} \begin{bmatrix} 1 & -\delta_{2i} 2^{-2i} \\ \delta_{2i} 2^{-2i} & 1 \end{bmatrix} \begin{bmatrix} X_{2i} \\ Y_{2i} \end{bmatrix} \\ +\rho_{2i} 2^{-2i} \cdot \begin{bmatrix} 1 & -\delta_{2i} 2^{-2i} \\ \delta_{2i} 2^{-2i} & 1 \end{bmatrix} \begin{bmatrix} U_{2i} \\ V_{2i} \end{bmatrix} \end{array} \right\} \quad (48)$$

$$Z_{2i+1} = \frac{1}{k_{2i}} (Z_{2i} - \rho_{2i} 2^{-2i} W_{2i}) \quad (49)$$

According to eqs. (46), (47), (48) and (49), we obtain

$$\begin{bmatrix} U_{2i+2} \\ V_{2i+2} \end{bmatrix} = \frac{1}{k_{2i+1}^2} \cdot \begin{bmatrix} 1 & -\delta_{2i+1} 2^{-2i-1} \\ \delta_{2i+1} 2^{-2i-1} & 1 \end{bmatrix} \begin{bmatrix} U_{2i+1} \\ V_{2i+1} \end{bmatrix}$$
$$- \frac{1}{k_{2i+1}^2} \cdot \rho_{2i+1} 2^{-2i-1} \cdot \begin{bmatrix} 1 & -\delta_{2i+1} 2^{-2i-1} \\ \delta_{2i+1} 2^{-2i-1} & 1 \end{bmatrix} \begin{bmatrix} X_{2i+1} \\ Y_{2i+1} \end{bmatrix} \quad (50)$$

$$W_{2i+2} = \frac{1}{k_{2i+1}} (W_{2i+1} + \rho_{2i+1} 2^{-2i-1} Z_{2i+1}) \quad (51)$$

$$\begin{bmatrix} X_{2i+2} \\ Y_{2i+2} \end{bmatrix} = \frac{1}{k_{2i+1}^2} \cdot \begin{bmatrix} 1 & -\delta_{2i+1} 2^{-2i-1} \\ \delta_{2i+1} 2^{-2i-1} & 1 \end{bmatrix} \begin{bmatrix} X_{2i+1} \\ Y_{2i+1} \end{bmatrix}$$
$$+ \frac{1}{k_{2i+1}^2} \cdot \rho_{2i+1} 2^{-2i-1} \cdot \begin{bmatrix} 1 & -\delta_{2i+1} 2^{-2i-1} \\ \delta_{2i+1} 2^{-2i-1} & 1 \end{bmatrix} \begin{bmatrix} U_{2i+1} \\ V_{2i+1} \end{bmatrix} \quad (52)$$

$$Z_{2i+2} = \frac{1}{k_{2i+1}} (Z_{2i+1} - \rho_{2i+1} 2^{-2i-1} W_{2i+1}) \quad (53)$$

where eqs. (52) and (53) are iteration equations of the 3-D double rotation algorithm.

Thus, the 3-D double rotation equations is modified as shown below

$$\begin{bmatrix} U_{2i+2} \\ V_{2i+2} \end{bmatrix} = \frac{1}{k_{2i+1}^2 k_{2i}^2} \cdot \begin{bmatrix} 1 & -\delta_{2i+1} 2^{-2i-1} \\ \delta_{2i+1} 2^{-2i-1} & 1 \end{bmatrix}$$
$$\begin{bmatrix} 1 & -\delta_{2i} 2^{-2i} \\ \delta_{2i} 2^{-2i} & 1 \end{bmatrix} \begin{bmatrix} U_{2i} \\ V_{2i} \end{bmatrix}$$
$$- \frac{\rho_{2i} \rho_{2i+1} \cdot 2^{-4i-1}}{k_{2i+1}^2 k_{2i}^2} \cdot \begin{bmatrix} 1 & -\delta_{2i+1} 2^{-2i-1} \\ \delta_{2i+1} 2^{-2i-1} & 1 \end{bmatrix}$$
$$\begin{bmatrix} 1 & -\delta_{2i} 2^{-2i} \\ \delta_{2i} 2^{-2i} & 1 \end{bmatrix} \begin{bmatrix} U_{2i} \\ V_{2i} \end{bmatrix}$$
$$- \frac{\rho_{2i} \cdot 2^{-2i}}{k_{2i+1}^2 k_{2i}^2} \cdot \begin{bmatrix} 1 & -\delta_{2i+1} 2^{-2i-1} \\ \delta_{2i+1} 2^{-2i-1} & 1 \end{bmatrix}$$
$$\begin{bmatrix} 1 & -\delta_{2i} 2^{-2i} \\ \delta_{2i} 2^{-2i} & 1 \end{bmatrix} \begin{bmatrix} X_{2i} \\ Y_{2i} \end{bmatrix}$$

$$-\frac{\rho_{2i+1}\cdot 2^{-2i-1}}{k_{2i+1}^2 k_{2i}^2}\cdot\begin{bmatrix}1 & -\delta_{2i+1}2^{-2i-1}\\ \delta_{2i+1}2^{-2i-1} & 1\end{bmatrix}\begin{bmatrix}1 & -\delta_{2i}2^{-2i}\\ \delta_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}X_{2i}\\ Y_{2i}\end{bmatrix} \qquad (54)$$

$$W_{2i+2}=\frac{1}{k_{2i}k_{2i+1}}(W_{2i}-\rho_{2i}\rho_{2i+1}2^{-4i-1}W_{2i}+\rho_{2i}2^{-2i}Z_{2i}+\rho_{2i+1}2^{-2i-1}Z_{2i}) \qquad (55)$$

$$\begin{aligned}\begin{bmatrix}X_{2i+2}\\ Y_{2i+2}\end{bmatrix}=&\frac{1}{k_{2i+1}^2 k_{2i}^2}\cdot\begin{bmatrix}1 & -\delta_{2i+1}2^{-2i-1}\\ \delta_{2i+1}2^{-2i-1} & 1\end{bmatrix}\begin{bmatrix}1 & -\delta_{2i}2^{-2i}\\ \delta_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}X_{2i}\\ Y_{2i}\end{bmatrix}\\ &-\frac{\rho_{2i}\rho_{2i+1}\cdot 2^{-4i-1}}{k_{2i+1}^2 k_{2i}^2}\cdot\begin{bmatrix}1 & -\delta_{2i+1}2^{-2i-1}\\ \delta_{2i+1}2^{-2i-1} & 1\end{bmatrix}\begin{bmatrix}1 & -\delta_{2i}2^{-2i}\\ \delta_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}X_{2i}\\ Y_{2i}\end{bmatrix}\\ &+\frac{\rho_{2i}\cdot 2^{-2i}}{k_{2i+1}^2 k_{2i}^2}\cdot\begin{bmatrix}1 & -\delta_{2i+1}2^{-2i-1}\\ \delta_{2i+1}2^{-2i-1} & 1\end{bmatrix}\begin{bmatrix}1 & -\delta_{2i}2^{-2i}\\ \delta_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}U_{2i}\\ V_{2i}\end{bmatrix}\\ &+\frac{\rho_{2i+1}\cdot 2^{-2i-1}}{k_{2i+1}^2 k_{2i}^2}\cdot\begin{bmatrix}1 & -\delta_{2i+1}2^{-2i-1}\\ \delta_{2i+1}2^{-2i-1} & 1\end{bmatrix}\begin{bmatrix}1 & -\delta_{2i}2^{-2i}\\ \delta_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}U_{2i}\\ V_{2i}\end{bmatrix}\end{aligned} \qquad (56)$$

$$Z_{2i+2}=\frac{1}{k_{2i}k_{2i+1}}\cdot(Z_{2i}-\rho_{2i}\rho_{2i+1}2^{-4i-1}Z_{2i}-\rho_{2i}2^{-2i}W_{2i}-\rho_{2i+1}2^{-2i-1}W_{2i}) \qquad (57)$$

The additional computation complexity of parallel processing for eqs. (54), (55), (56) and (57) is three additions, one double rotation CORDIC computation and one shit for each iteration. In the *n*-bit operand system, when $i \geq \frac{n}{4}-1$, eqs. (54), (55), (56) and (57) become

$$\begin{aligned}\begin{bmatrix}U_{2i+2}\\ V_{2i+2}\end{bmatrix}=&\frac{1}{k_{2i+1}^2 k_{2i}^2}\cdot\begin{bmatrix}1 & -\delta_{2i+1}2^{-2i-1}\\ \delta_{2i+1}2^{-2i-1} & 1\end{bmatrix}\begin{bmatrix}1 & -\delta_{2i}2^{-2i}\\ \delta_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}U_{2i}\\ V_{2i}\end{bmatrix}\\ &-\frac{\rho_{2i}\cdot 2^{-2i}}{k_{2i+1}^2 k_{2i}^2}\cdot\begin{bmatrix}1 & -\delta_{2i+1}2^{-2i-1}\\ \delta_{2i+1}2^{-2i-1} & 1\end{bmatrix}\begin{bmatrix}1 & -\delta_{2i}2^{-2i}\\ \delta_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}X_{2i}\\ Y_{2i}\end{bmatrix}\\ &-\frac{\rho_{2i+1}\cdot 2^{-2i-1}}{k_{2i+1}^2 k_{2i}^2}\cdot\begin{bmatrix}1 & -\delta_{2i+1}2^{-2i-1}\\ \delta_{2i+1}2^{-2i-1} & 1\end{bmatrix}\begin{bmatrix}1 & -\delta_{2i}2^{-2i}\\ \delta_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}X_{2i}\\ Y_{2i}\end{bmatrix}\end{aligned} \qquad (58)$$

$$W_{2i+2}=\frac{1}{k_{2i}k_{2i+1}}(W_{2i}+\rho_{2i}2^{-2i}Z_{2i}+\rho_{2i+1}2^{-2i-1}Z_{2i}) \qquad (59)$$

$$\begin{aligned}\begin{bmatrix}X_{2i+2}\\ Y_{2i+2}\end{bmatrix}=&\frac{1}{k_{2i+1}^2 k_{2i}^2}\cdot\begin{bmatrix}1 & -\delta_{2i+1}2^{-2i-1}\\ \delta_{2i+1}2^{-2i-1} & 1\end{bmatrix}\begin{bmatrix}1 & -\delta_{2i}2^{-2i}\\ \delta_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}X_{2i}\\ Y_{2i}\end{bmatrix}\\ &+\frac{\rho_{2i}\cdot 2^{-2i}}{k_{2i+1}^2 k_{2i}^2}\cdot\begin{bmatrix}1 & -\delta_{2i+1}2^{-2i-1}\\ \delta_{2i+1}2^{-2i-1} & 1\end{bmatrix}\begin{bmatrix}1 & -\delta_{2i}2^{-2i}\\ \delta_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}U_{2i}\\ V_{2i}\end{bmatrix}\\ &+\frac{\rho_{2i+1}\cdot 2^{-2i-1}}{k_{2i+1}^2 k_{2i}^2}\cdot\begin{bmatrix}1 & -\delta_{2i+1}2^{-2i-1}\\ \delta_{2i+1}2^{-2i-1} & 1\end{bmatrix}\begin{bmatrix}1 & -\delta_{2i}2^{-2i}\\ \delta_{2i}2^{-2i} & 1\end{bmatrix}\begin{bmatrix}U_{2i}\\ V_{2i}\end{bmatrix}\end{aligned} \qquad (60)$$

$$Z_{2i+2}=\frac{1}{k_{2i}k_{2i+1}}(Z_{2i}-\rho_{2i}2^{-2i}W_{2i}-\rho_{2i+1}2^{-2i-1}W_{2i}) \qquad (61)$$

Thus, the additional computation complexity of parallel processing is two additions, one double rotation CORDIC computation and one shift for each iteration. The computation time of the double rotation CORDIC algorithm is also reduced [10]. The 3-D rotation with conventional CORDIC algorithm versus the 3-D rotation with double rotation CORDIC algorithm is shown in Fig. 7.

6. 3-D Central Perspective Method Using CORDIC Algorithm

The 3-D central perspective method is shown in Fig. 8 [7]. The graphic is rotated in 3-D space and mapped onto Y'-Z' plane perspectively. We obtain the coordinate $(0, y'', z'')$ in Y'-Z' plane as follows:

$$x''=0 \qquad (62)$$

$$y''=\frac{D}{D-x'}\cdot y' \qquad (63)$$

$$z''=\frac{D}{D-x'}\cdot z' \qquad (64)$$

where $D=\sqrt{x_0^2+y_0^2+z_0^2}$, (x_0, y_0, z_0) is the coordinate of observer, and $(x^{'}, y^{'}, z^{'})$ is the rotated coordinate.

7. VLSI Architectures for 3-D Rotation and Perspective with CORDIC Algorithm

7.1 The Architecture of 3-D Rotation with Conventional CORDIC Algorithm

Fig. 9 shows the architecture of the 3-D rotation with the rotation mode in a CORDIC circular coordinate system. In this architecture, the (U_{i+1}, V_{i+1}) and (X_{i+1}, Y_{i+1}) generator each consists of two 2-D CORDIC processors, two hardwire shifts and two adders/subtrators. The W_{i+1} and Z_{i+1} generator each consists of a half of 2-D CORDIC Processor.

7.2 The Architecture of 3-D Rotation with Double Rotation CORDIC Algorithm

The architecture of the 3-D rotation with double rotation CORDIC algorithm is shown in Fig. 10. In this architecture, the (U_{i+1}, V_{i+1}) and (X_{i+1}, Y_{i+1}) generator each consists of two 2-D CORDIC processors, six hardwire shifts and three adders/subtrators. The W_{i+1} and Z_{i+1} generator each consists of a half of 2-D double rotation CORDIC Processor. The 3-D rotation with double rotation CORDIC algorithm can improve the latency time by more than thirty percent [10].

7.3 The Architecture of 3-D perspective Method with CORDIC Algorithm

The proposed architecture of 3-D perspective method consists of five 2-D CORDIC processors and one subtractor. Two CORDIC processors operate in the circular coordinate system for computing $\sqrt{x_0^2 + y_0^2 + z_0^2}$, and three CORDIC processors operate in the linear coordinate system for computing $x^{''}$ and $y^{''}$. The architecture of 3-D central perspective method is shown in Fig. 11.

The hardware codes of both that with CORDIC algorithm and double rotation algorithm are written in Verilog-hardware description Language (HDL) [11] running on SUN Blade 1000 workstation under ModelSim simulation tool [12]. Both of two architectures were synthesized by Xilinx FPGA express tools [13] and emulated on the Xilinx XC2V4000 FPGA platform [14]. In the 32-bit accelerated architecture of 3-D rotation, compared with the conventional CORDIC-based architecture of 3-D rotation, the accelerated design improves the latency by more than 30%. The timing diagram for the conventional CORDIC-based architecture and the accelerated architecture of 3-D rotation is shown in Fig. 12. It is designed to evaluate the hardware and to provide an intellectual property (IP) for 3-D graphic engine.

8. Impact of New Architectures and Algorithms

The Euler angle method consists of sequence of three rotations [2] [6], each rotates one of three orthogonal axes. This method is represented by Euler angles correspond to the sequence of rotations about the coordinate axes. The 3-D rotation method is implemented by cascading two 2-D CORDIC processors [2] [6]. Lang and Antelo proposed a method that replaces two 2-D CORDIC processors by one 3-D CORDIC processor [6]. The sequence of rotations consists of one 2-D CORDIC rotation and one 3-D CORDIC rotation. Both of them require more than two 2-D CORDIC computations. According to the proposed 3-D rotation algorithm, the architecture with conventional CORDIC processors requires one 2-D CORDIC computation in parallelism to perform 3-D rotation, and the architecture with double rotation 2-D CORDIC processors requires less than one 2-D CORDIC computation in parallelism to perform 3-D rotation.

The 3-D central perspective method requires four 2-D CORDIC computations in parallelism; this method with CORDIC algorithm saves multipliers and square-root, and the implementation of this architecture is required by CORDIC processors only.

9. Conclusions

We have presented two high-throughput 3-D rotation algorithms and architectures both of them are based on 2-D CORDIC algorithm and 2-D double rotation CORDIC algorithm. It is required one or less 2-D CORDIC computation to perform 3-D rotation; and the central perspective method is also performed by 2-D CORDIC algorithm, the architecture of the central perspective method saves hardware and achieves high-performance.

The proposed architectures are implemented by 2-D CORDIC processors; the architectures are simple and regular, and suitable for VLSI implementation. The graphic engine should be improved by the proposed algorithms and architectures.

10. References

[1] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, R. Huebner, “Level of Detail for 3-D Graphics,” Morgan Kaufmann Pub., 2003.

[2] D. H. Eberly, “3-D Game Engine Design-A Practical Approach to Real-Time Computer Graphics,” Morgan Kaufmann Pub., 2001.

[3] J. E. Volder, “The CORDIC Trigonometric Computing Technique,” *IRE Transactions on Electronic Computers*, Vol. EC-8, 1959, pp. 330-334.

[4] J. S. Walther, “A Unified Algorithm for Elementary Functions,” *Spring Joint Computer Conference Proceedings*, Vol.38, 1971, pp.379-385.

[5] O. Mencer, L. Semeria, M. Morf, J. Delosme, “Application of Reconfigurable CORDIC Architecture,” *The Journal of VLSI Signal Processing, Special Issue on Reconfigurable Computing*, March 2000.

[6] T. Lang, E. Antelo, “High-Throughput 3-D Rotations and Normalizations,” *Thirty-Fifth Asilomar Conference on Signal, Systems and Computers*, 2001, pp.846-851.

[7] T. Y. Sung, “Survey of 3-D Perspective Methods for Graphic Engine,” Technical Report (SV-041117), SoC and VLSI Signal Processing Lab., Department of Microelectronics Engineering, Chung Hua University, Hsinchu, Taiwan, 2004.

[8] J. Euh, J. Chittamuru, W. Burson, “CORDIC Based Interporator for 3-D Graphics,” *IEEE Workshop on Signal Processing Systems*, 2002, pp.240-245.

[9] S. Wang, E. E. Swartzlander Jr., “Merged CORDIC Algorithm,” *Proc. Int’l Symp. Circuits and Systems*, 1995, pp.1988-1991.

[10] T. Y. Sung, C. S. Chen, M. C. Shih, “The Double Rotation CORDIC Algorithm: New Results for VLSI Implementation of Fast Sine/Cosine Generation,” *2004 International Computer Symposium (ICS-2004)*, Taipei, Taiwain, Dec. 15-17, 2004.

[11] D. E. Thomas, P. H. Moorby, The Verilog Hardware Description Language, Fifth Edition, Kluwer Academic Pub. 2002.

[12] Model ModelSim Products: *http://www.model.com/products.*

[13] Synopsys FPGA Express, *http://www.synopsys.com/products.*

[14] Xilinx FPGA products, *http://www.xilinx.com/products.*

Table 1 Functions of CORDIC arithmetic

Coordinate System	Rotation Mode $z(n) \to 0$	Vectoring Mode $y(n) \to 0$
Linear m=0	$x(n) = x(0)$ $y(n) = y(0) + x(0) \cdot z(0)$	$x(n) = x(0)$ $z(n) = z + \frac{y(0)}{x(0)}$
Circular m=1	$x(n) = \frac{1}{K_1}(x(0)\cos z(0) - y(0)\sin z(0))$ $y(n) = \frac{1}{K_1}(y(0)\cos z(0) + x(0)\sin z(0))$	$x(n) = \frac{1}{K_1}(x(0)^2 + y(0)^2)$ $z(n) = z(0) - \tan^{-1}(\frac{y(0)}{x(0)})$
Hyperbolic m=-1	$x(n) = \frac{1}{K_{-1}}(x(0)\cosh z(0) + y(0)\sinh z(0))$ $y(n) = \frac{1}{K_{-1}}(y(0)\cosh z(0) + x(0)\sinh z(0))$	$x(n) = \frac{1}{K_{-1}}(x(0)^2 - y(0)^2)$ $z(n) = z(0) + \tanh^{-1}(\frac{y(0)}{x(0)})$

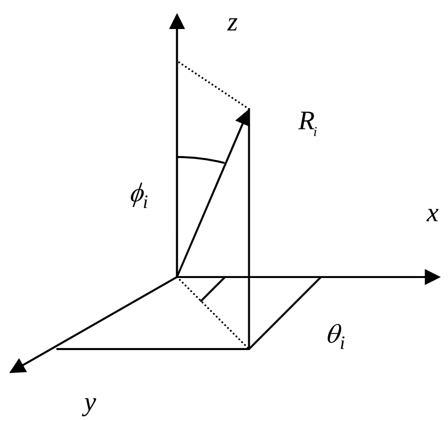


Fig.1. A vector R in three dimensional space

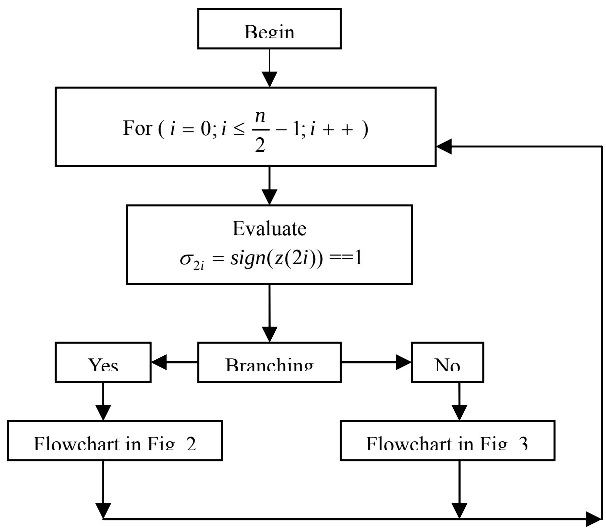


Fig. 2. Flowchart for the σ_{2i+1}-prediction and $z(2i+2)$ determination algorithm. Detailed flowcharts for specific cases when sign($z(2i)$) evaluation returns +1 , -1, and when the algorithm is in a branching are

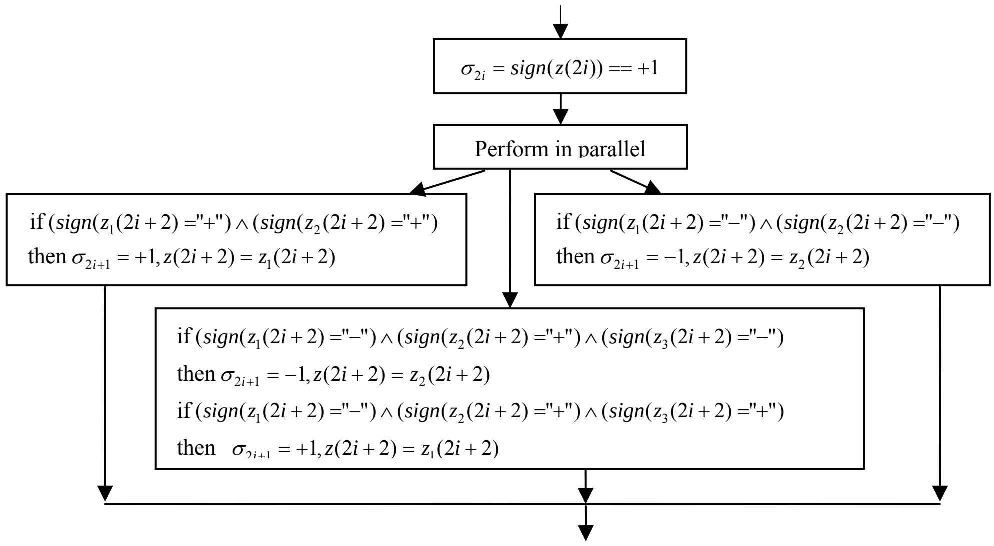


Fig. 3. Flowchart for *i*-iteration for the case when $\sigma_{2i} = sign(z(2i))$ evaluation returns +1

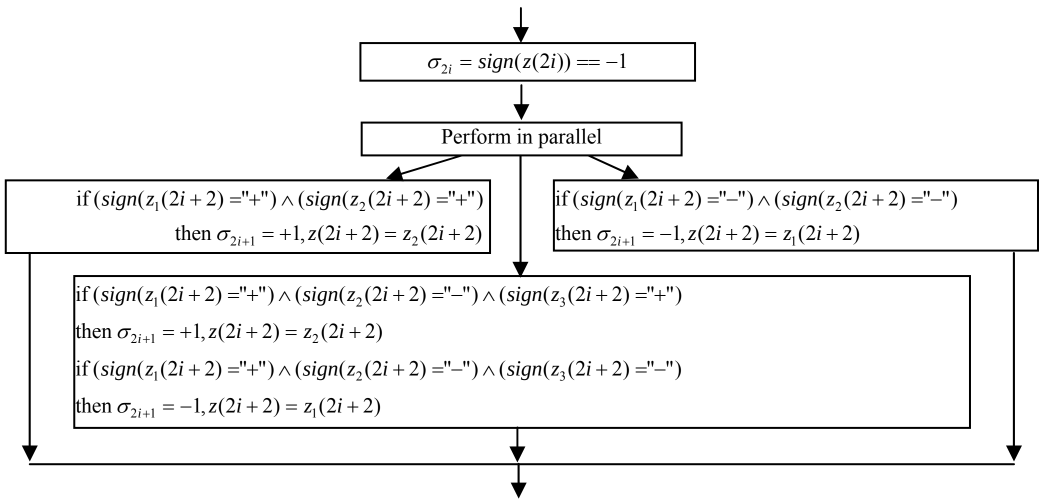
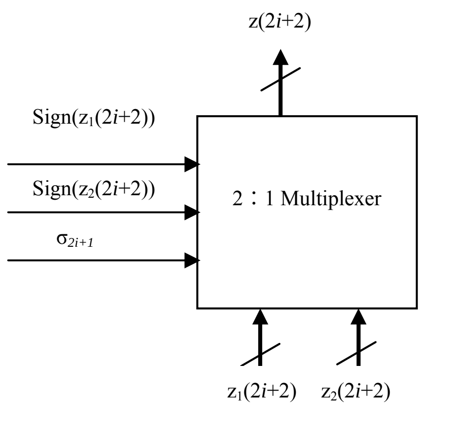
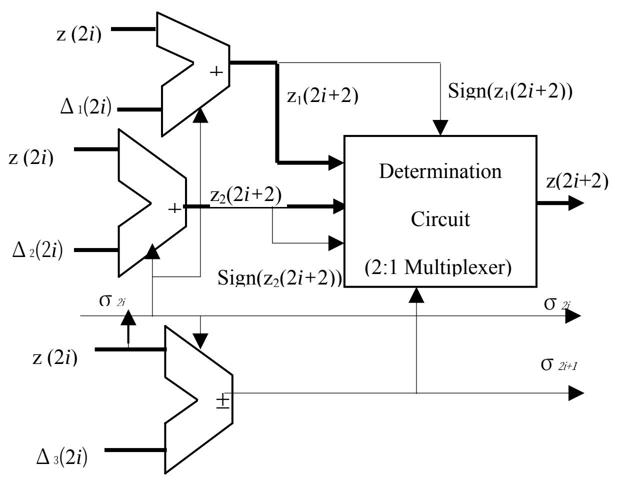


Fig. 4. Flowchart for *i*-iteration for the case when $\sigma_{2i} = sign(z(2i))$ evaluation returns -1

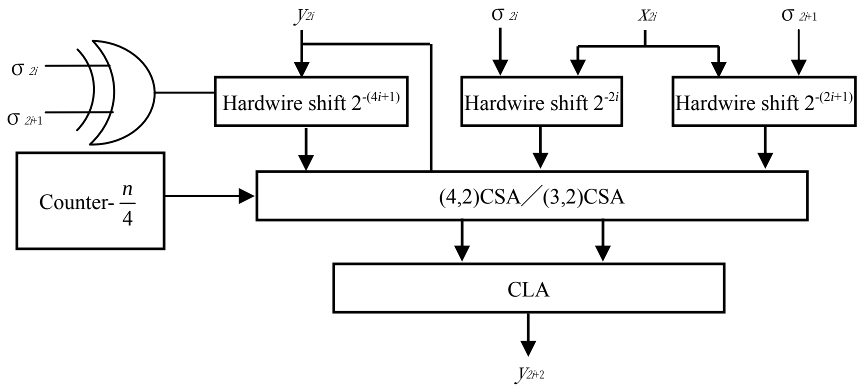


(a) Determination circuit of $z(2i+2)$

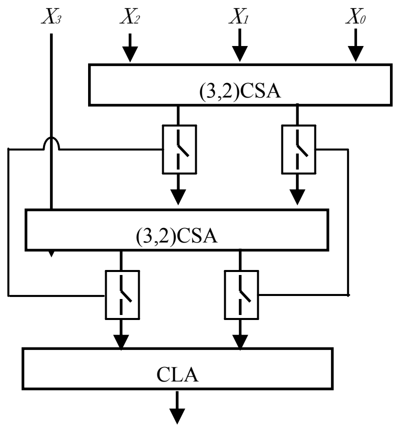


(b) $\sigma_{2i}, \sigma_{2i+1}$ and $z(2i+2)$ generator

Fig. 5. The determination circuit of σ_{2i} , σ_{2i+1} and $z(2i+2)$



(a) (3,2)CSA/(4,2)CSA



(b) vector $[x_{2i+2}\ y_{2i+2}]$ generator

Fig. 6. The accelerated CORDIC architecture with the rotation mode in the circular coordinate system.

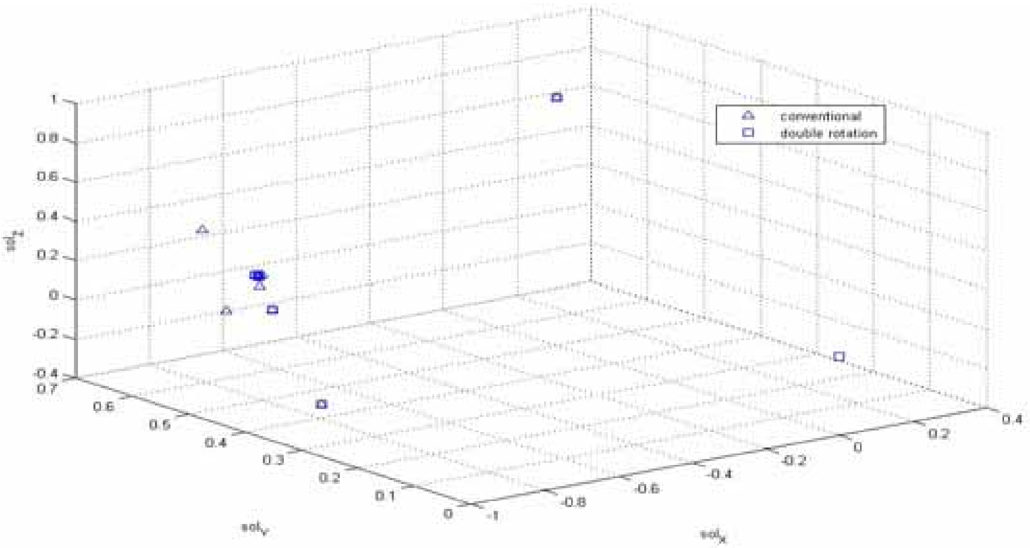


Fig. 7 3-D rotation with conventional CORDIC algorithm versus 3-D rotation with double rotation CORDIC algorithm ($R_0 = 1, \alpha_0 = \frac{\pi}{3}, \beta_0 = \frac{\pi}{4}, \theta_0 = \frac{\pi}{2}, \phi_0 = \frac{\pi}{3}$)

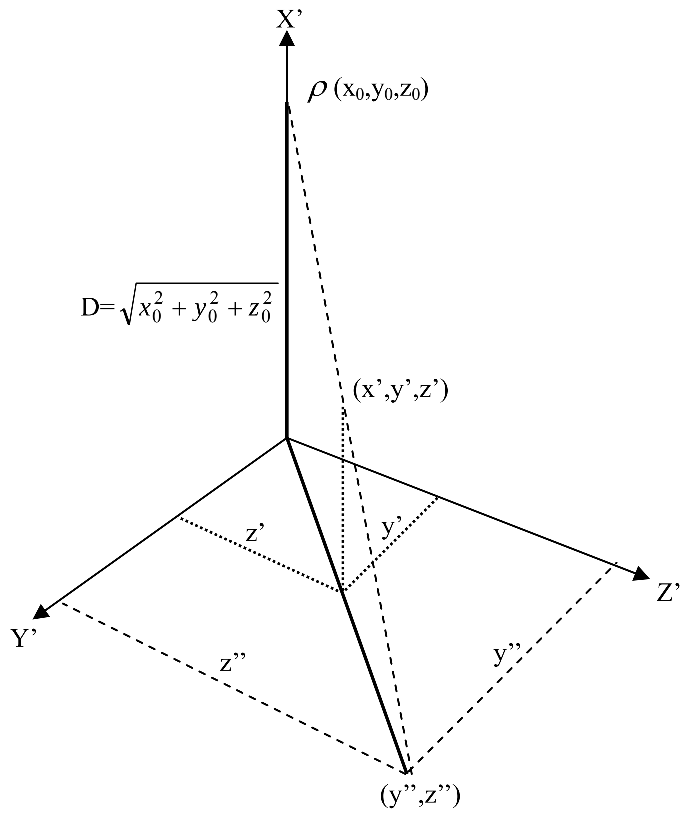
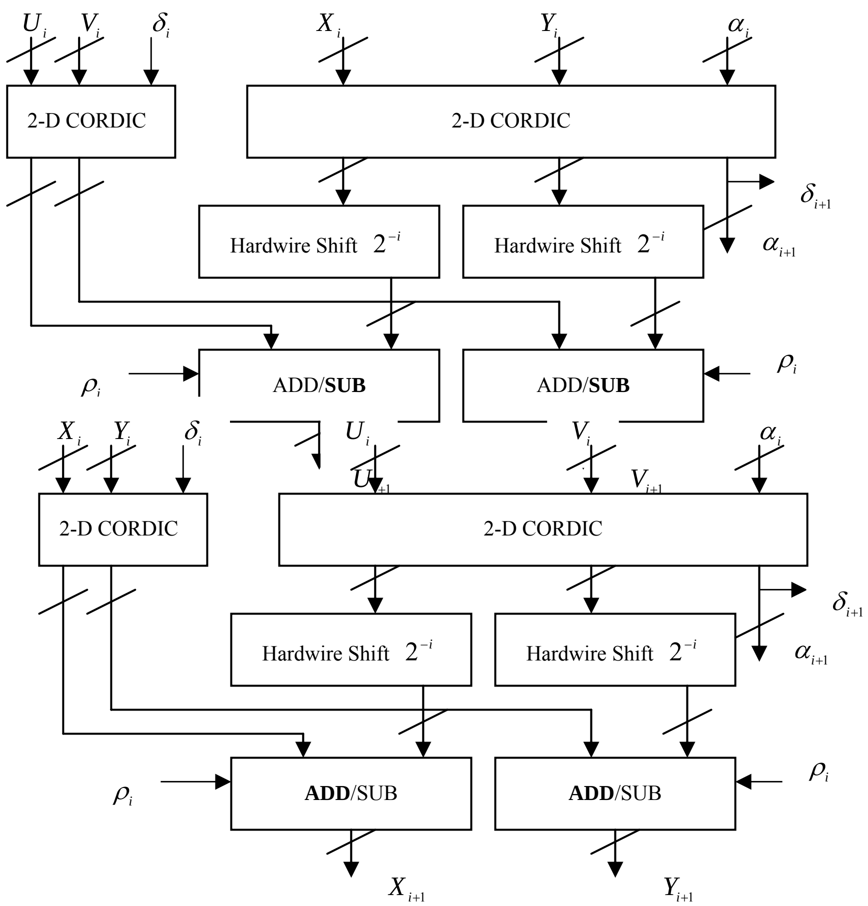


Fig. 8. The 3-D central perspective method



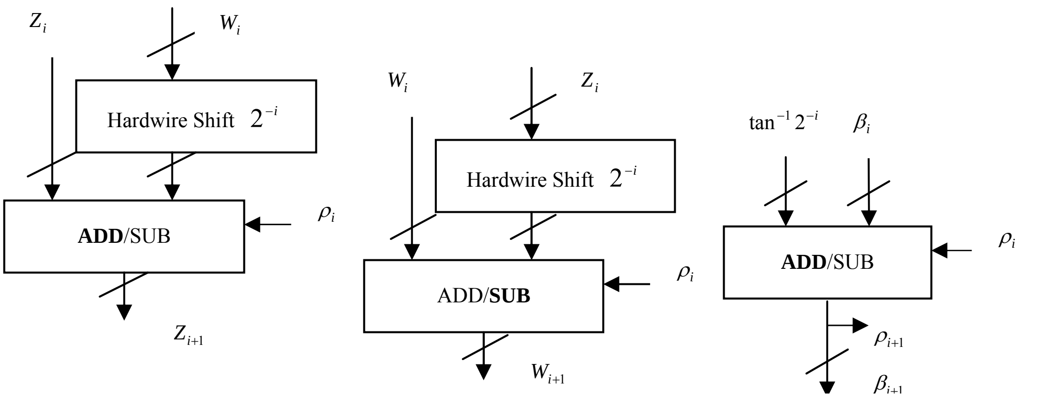
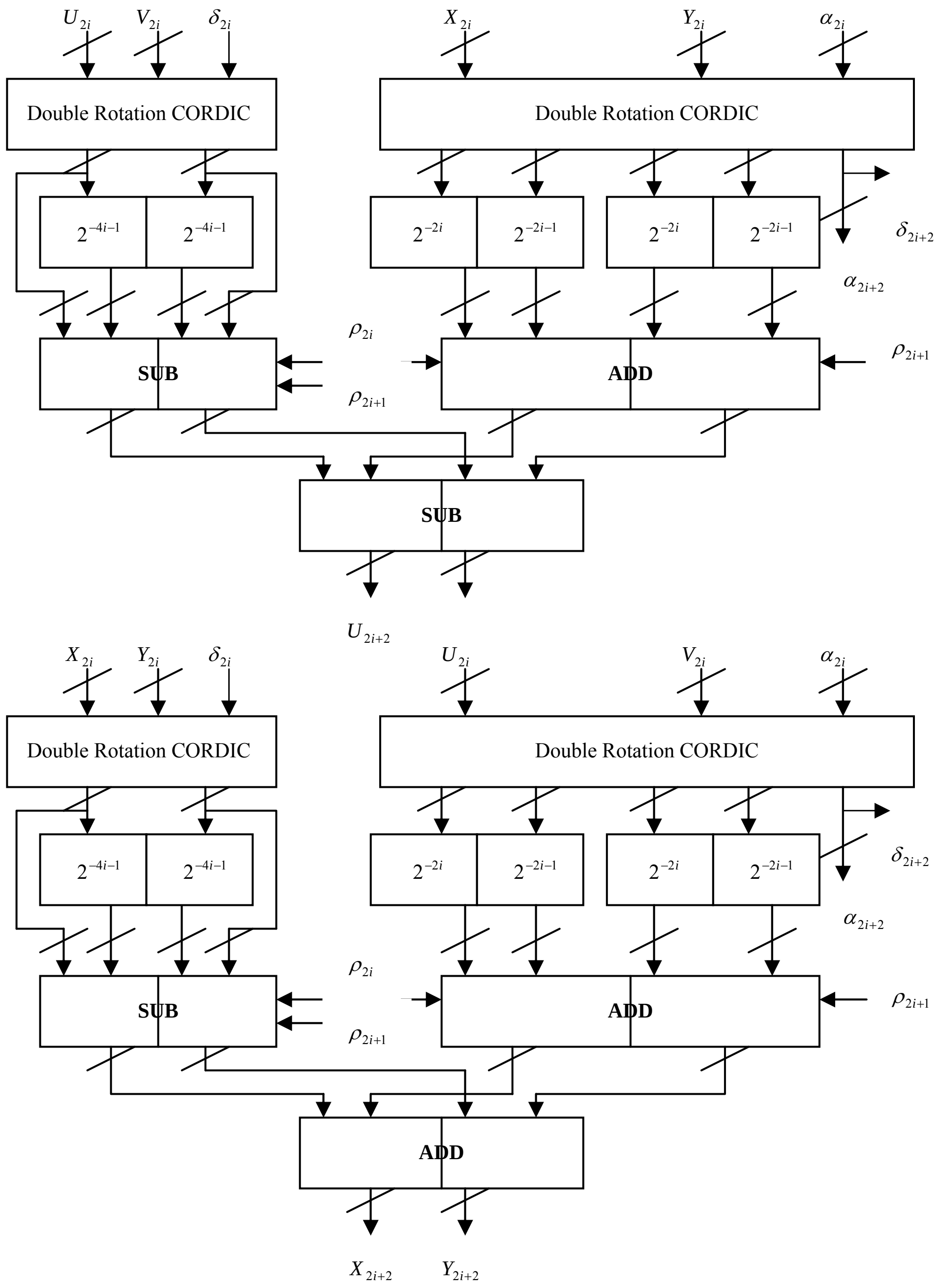


Fig. 9. The architecture of the 3-D Rotation with 2-D CORDIC algorithm



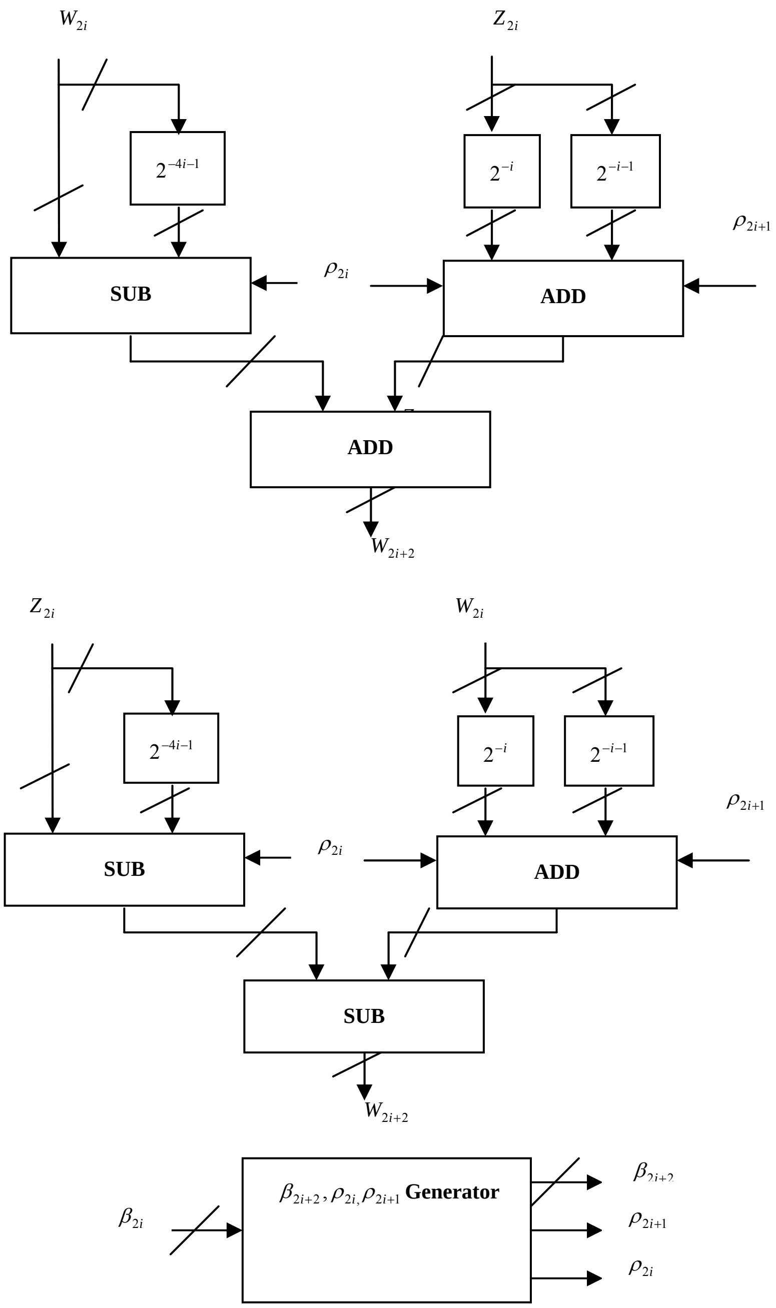


Fig. 10. The architecture of the 3-D Rotation with Double Rotation CORDIC algorithm

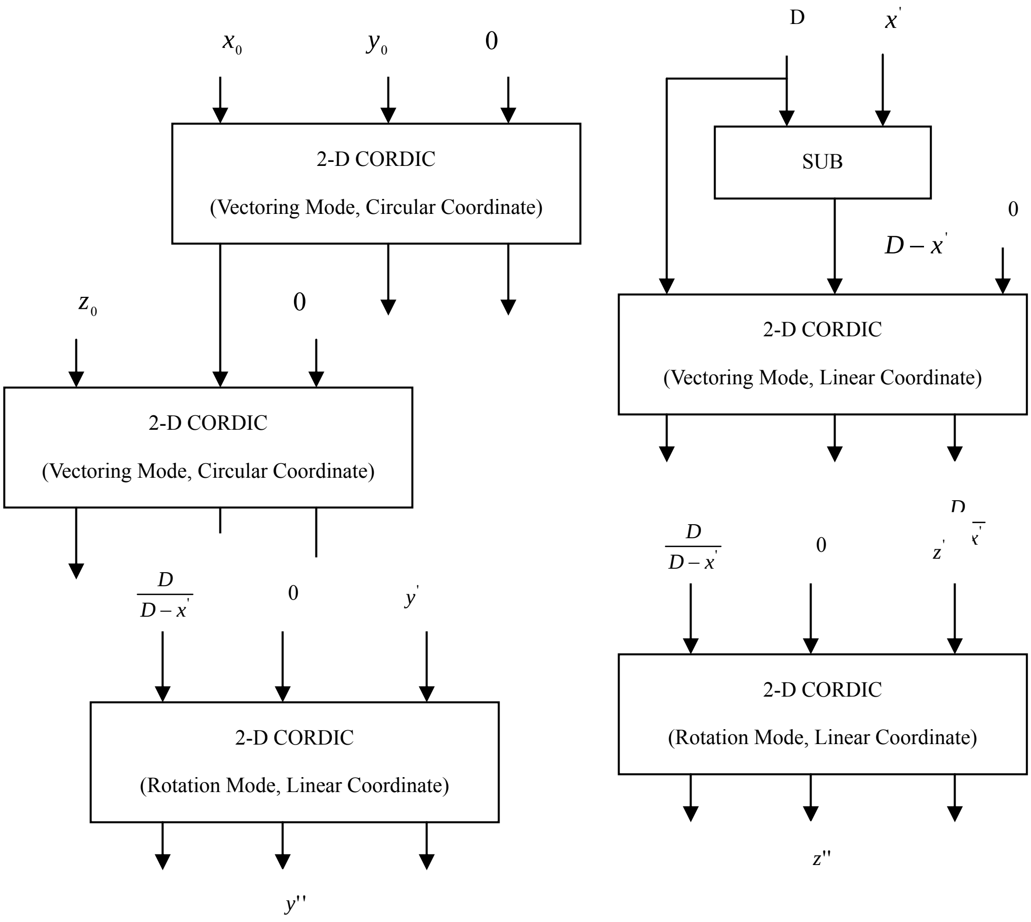


Fig. 11. The architecture of 3-D central perspective method

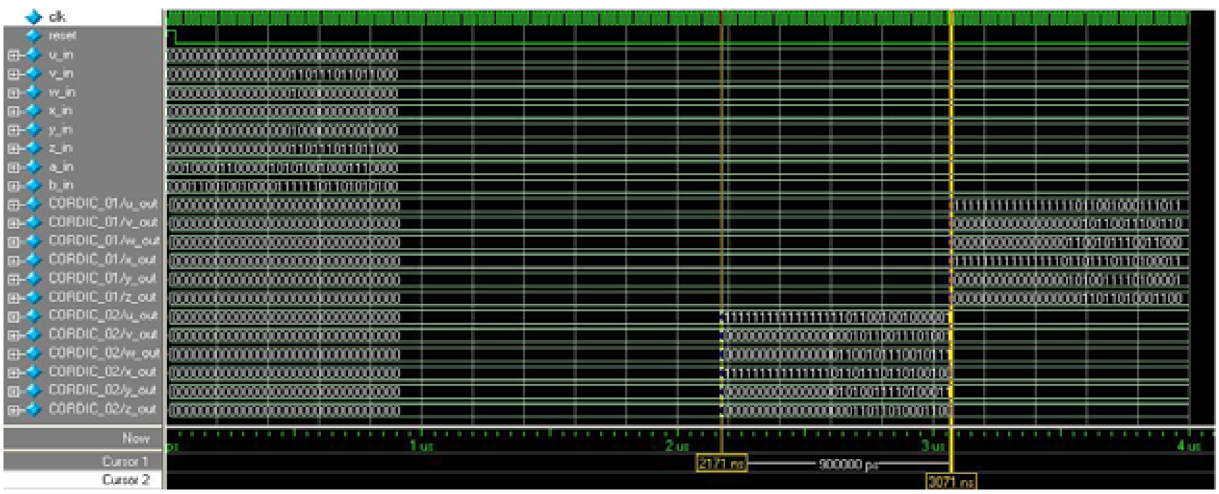


Fig. 12. The timing diagram for the conventional CORDIC-based architecture and the accelerated architecture of 3-D rotation (CORDIC_01: Conventional CORDIC, CORDIC_02: Double rotation CORDIC)