IJIRST –International Journal for Innovative Research in Science & Technology| Volume 1 | Issue 3 | August 2014
ISSN(online) : 2349-6010

Implementation of Optimized CORDIC Designs

Bibinu Jacob
M Tech Student
Department of Electronics & Communication Engineering
Mangalam College Of Engineering, Ettumanoor

Reenesh Zacharia
Assistant Professor
Department of Electronics & Communication Engineering
Mangalam College Of Engineering, Ettumanoor

Abstract

CORDIC stands for COordinate Rotation DIgital Computer, which is likewise known as Volder's algorithm and digit-by-digit method. It is a simple and effective method to calculate many functions like trigonometric, hyperbolic, logarithmic etc. It performs complex multiplications using simple shifts and additions. It finds applications in robotics, digital signal processing, graphics, games, and animation. But, there isn't any optimized CORDIC designs for rotating vectors through specific angles. In this paper, optimization schemes and CORDIC circuits for fixed and known rotations are proposed. Finally an argument reduction technique to map larger angle to an angle less than 45 is discussed. The proposed CORDIC cells are simulated by Xilinx ISE Design Suite and shown that the proposed designs offer less latency and better device utilization than the reference CORDIC design for fixed and known angles of rotation.

Keywords: Coordinate rotation digital computer (CORDIC), digital signal processing (DSP) chip, very large scale integration (VLSI)

I. Introduction

The advances in the very large scale integration (VLSI) technology and the advent of electronic design automation (EDA) tools have been directing the current research in the areas of digital signal processing (DSP), communications, etc. in terms of the design of high speed VLSI architectures for real-time algorithms and systems which have applications in the above mentioned areas. The development rate of VLSI technology was predicted by Gordon Moore and since 1965 newer technologies have been developed by the industry fitting his predicted curve, which was introduced as the so called Moore's law. These advances have provided momentum to the designers for transforming algorithm into architecture. Many DSP algorithms use elementary functions like logarithmic, trigonometric, exponential, division and multiplication. Two of the ways of implementing these functions are by using table lookup method and through polynomial expansions. The above mentioned methods require large number of multiplications/divisions and additions/subtractions. Compared to the other approaches CORDIC is a clear winner when Hardware Multiplier is unavailable (eg microcontroller) and when need to save the gates required to implement (eg. FPGA)

Coordinate Rotation Digital Computer (CORDIC), a special purpose computer to compute many non-linear and transcendental functions, was proposed by Volder in 1959 [2]. The functions that can be computed using a CORDIC computer include trigonometric, logarithmic, exponential, hyperbolic, multiplication, division, square root, etc. [3]. Though it initially served the purpose of navigation systems, it later became a popular tool to implement several digital systems especially in the areas of digital signal processing, communications, computer graphics, etc. [3]. The simplicity of CORDIC is that it can compute any of the above mentioned functions using shifts and additions . The operating mode and the coordinate system chosen are two key factors to compute the desired functions in the CORDIC. Many signal processing and communication systems operate CORDIC in circular coordinate system and in either of rotation or vectoring modes.

Latency of computation is the major issue with the implementation of CORDIC algorithm due to its linear-rate convergence [4]. It requires (n+1) iterations to have n-bit precision of the output. Overall latency of computation increases linearly with the product of the word-length and the CORDIC iteration period. The speed of CORDIC operations is, therefore, constrained either by the precision requirement (iteration count) or the duration of the clock period. The angle recoding (AR) schemes could be applied for reducing the iteration count for CORDIC implementation of constant complex multiplications by encoding the angle of rotation as a linear combination of a set of selected elementary angles of micro-rotations.

In the conventional CORDIC, any given rotation angle is expressed as a linear combination of values of elementary angles that belong to the set to obtain an n bit value of

$$\{(\sigma \cdot \arctan(2^{-r}\ )\ ) : \sigma\ \in [\{-1,1\}, r \in\ \{1,2, \ldots . , n\}] \sum_{0}^{n-1} [\sigma_i\ \cdot \arctan(\ 2^{-i}\ )]\}.$$

However, in AR methods, this constraint is relaxed by adding zero into the linear combination to obtain the desired angle using relatively fewer terms of the form $\{(\sigma \cdot \arctan(2^{-r}\ )\ )$for $\sigma\ \in [\{-1,0,1\}$.

The contributions of this paper are as follows.

(1) Reference CORDIC cell is designed for comparison.
(2) Optimized set of micro-rotations are derived for the implementation of fixed-angle vector-rotation.

(3) A novel hardware pre-shifting scheme is suggested for reduction of barrel-shifter complexity.
(4) Bi-rotation CORDIC circuits are designed and used for high speed fixed-angle vector rotations.
(5) Shift-add operations for corresponding scaling circuits are derived.
(6) Modified Bi-Rotation CORDIC circuit to reduce hardware complexity are designed.
(7) Argument reduction technique to map any angle to an angle less than 45 degree angle.

II. REFERENCE CORDIC DESIGN

The CORDIC algorithm can be employed in two different modes, known as the “rotation” mode and the "vectoring" mode. In the rotation mode, the co-ordinate components of a vector and an angle of rotation are given and the co-ordinate components of the original vector, after rotation through a given angle, are computed. In the vectoring mode, the co-ordinate components of a vector are given and the magnitude and angular argument of the original vector are computed. Rotation-mode CORDIC algorithm to rotate a vector (U_x , U_y) through an angle to obtain a rotated vector (V_x , V_y) is given by the equation,

$$(U_x)_{i+1} = \mathbf{K}\ [(U_x)_i - \sigma_i (U_y)_i * 2^{-i}] \qquad (1)$$

$$(U_y)_{i+1} = \mathbf{K}\ \mathbf{[}(U_y)_i + \sigma_i (U_x)_i * 2^{-i}] \qquad (2)$$

$$\Phi_{i+1} = \Phi_i - \sigma_i \tan^{-1}(2^{-i}) \qquad (3)$$

Where

σ_i = -1 for $\Phi_i < 0$ else
= 1.
i = Iterations required to get the desired accuracy
K = Scale Factor

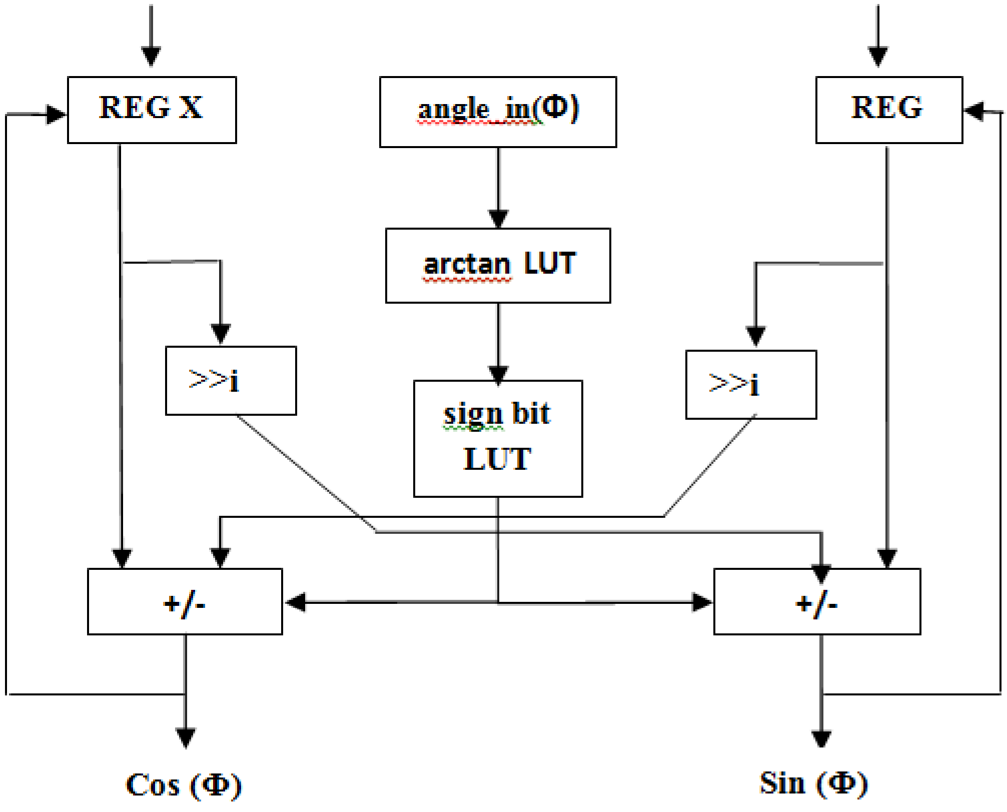


Fig.1. Reference CORDIC Design

Scaling factor K is given by

$$\mathbf{K} = \prod_{i=0}^{n-1} 1\Big/\sqrt{[1+2^{-2i}]} \approx \mathbf{0.607} \qquad (4)$$

K can be ignored in the iterative process and then applied afterward with a scaling factor: For fixed rotation, Φ_i could be pre-computed and the sign-bits corresponding to $\boldsymbol{\sigma_i}$ could be stored in CORDIC circuit. The CORDIC circuit therefore need not compute the remaining angle during the CORDIC iterations.

A reference CORDIC circuit for fixed rotations according to " (1)" and "(2)" is shown in Fig 1. Ux and Uy are fed as input to the pair of input registers and the successive feedback values at each iteration are fed in parallel to the input registers. Note that conventionally we feed the pair of input registers with the initial values and as well as the feedback values are fed through a pair of multiplexers. The sign bit to determine whether addition or subtraction needs to happen in adder/subtractor module is stored in the sign bit LUT.

III. PROPOSED CORDIC DESIGNS

A. *Optimization Of Elementary Angle Set*

For rotation of a vector through a known and fixed angle of rotation using a rotation-mode CORDIC circuit, we can find a set of a small number of pre-determined elementary angles $\{\alpha_{i\,,}$ for $0 \leq i \leq m-1\}$, where $\alpha_i = \arctan(2^{-k(i)})$ is the elementary angle to be used for the i^th^ micro-rotation in the CORDIC algorithm , and m is the minimum necessary number of micro-rotations. Meanwhile, it is well known that the rotation through any angle $\{0 < \theta \leq 2\pi\}$ can be mapped into a positive rotation through without any extra arithmetic operations [5].

Hence, as a first step of optimization, we perform the rotation mapping so that the rotation angle lies in the range of $\left\{0 < \emptyset \leq \frac{\pi}{4}\right\}$ without any extra arithmetic operations. Therefore the CORDIC algorithm can be modified as

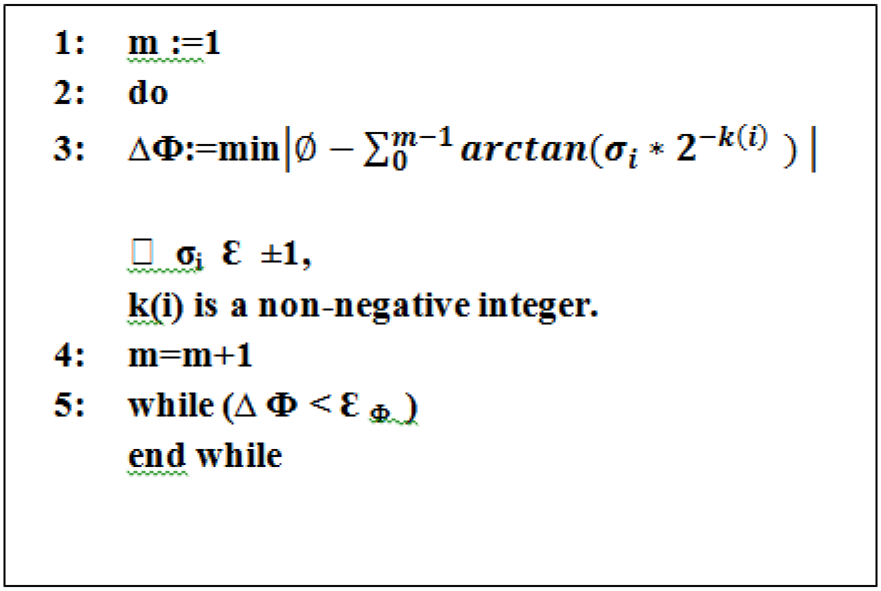
$(U_x)_{i+1} = (U_x)_i - \sigma_i (U_y)_i *2^{-k(i)}$ (5)

$(U_y)_{i+1} = (U_y)_i + \sigma_i (U_x)_i *2^{-k(i)}$ (6)

The scale-factor now depends on the $\{\alpha_i\}$ set. The accuracy of CORDIC algorithm depends on how closely the resultant rotation $\emptyset_A$ due to all the micro-rotations in "(1)" approximates to the desired rotation angle $\emptyset$, which in turn determines the deviation of actual rotation vector from the estimated value. We show here that only a few elementary angles are sufficient to have a CORDIC rotation in the range$[0,\ \pi/4]$and different sets of elementary angles can be chosen according to the accuracy requirement.

The simple pseudo code to optimize a set of micro-rotations is described in **Algorithm 1**. If the maximum accuracy $\in_\emptyset$ which is defined as the maximum tolerable error between desired angle and approximated angle is given as an input, the optimization algorithm searches the parameters k(i) and σ_i that can minimize an objective function ΔΦ. The algorithm starts with the single micro-rotation, i.e.=1 , then if the micro-rotation that has smaller angle of deviation than $\in_\emptyset$ cannot be found, the number of micro-rotations is increased by one and the optimization algorithm is run again. Exhaustive search is employed in the optimization algorithm to search the entire parameter space for all the combinations of k(i) and σ_i. Based on the obtained micro-rotations, the parameters for scaling operation can be searched with the different objective function.

ALGORITHM 1 to obtain optimal micro rotations



```
m :=1
do
$\Delta\Phi$:=min$\left|\emptyset - \sum_0^{m-1} arctan(\sigma_i * 2^{-k(i)})\right|$

□ $\sigma_i$ $\varepsilon$ ±1,
k(i) is a non-negative integer.
m=m+1
while ($\Delta\Phi < \varepsilon_{\Phi}$)
end while
```

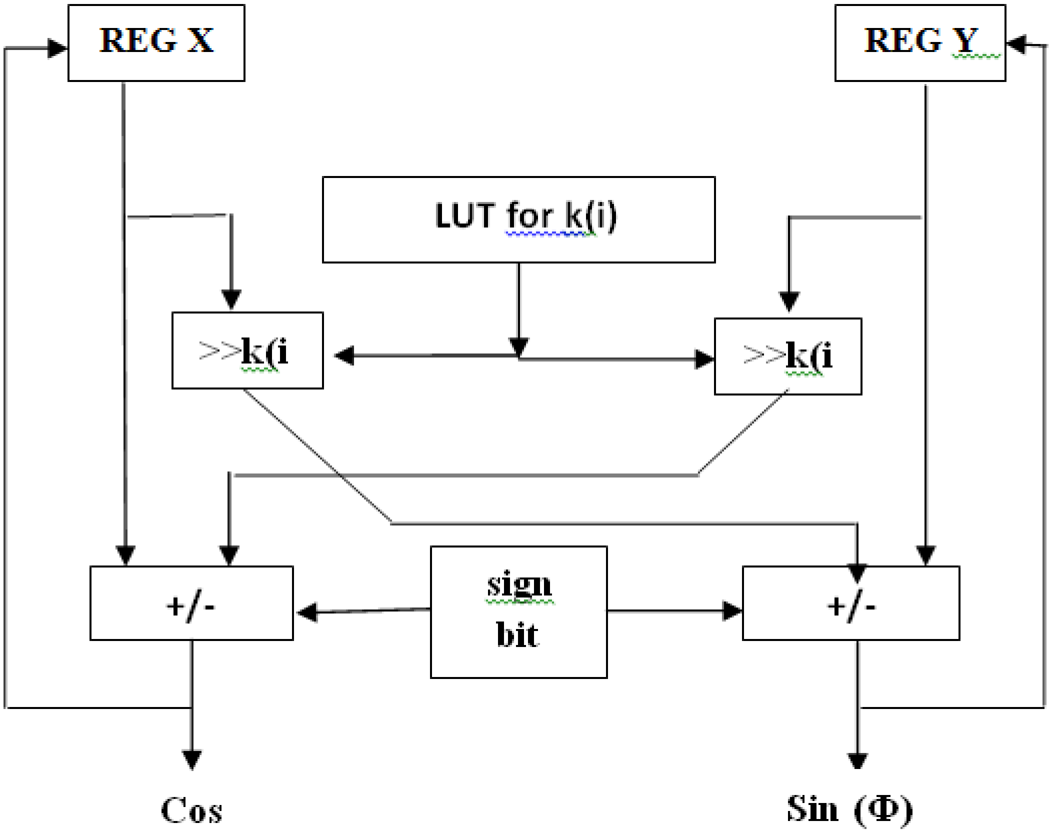


Fig. 2. CORDIC cell to obtain optimal micro rotations

Φ – desired rotation angle
$\mathcal{E}_{\Phi}$ - maximum tolerable error - 0.04^0
k(i) – scaling parameter for each iteration for a particular angle.
m – minimum necessary no of micro rotations

In the experiment with the maximum input angular deviation $\mathcal{E}_{\Phi} = 0.04^0$, we found that a set of four selected micro-rotations is enough. In Table I, it is shown that rotations through any angle in the range $0 < \emptyset \le \frac{\pi}{4}$ could be achieved with maximum angular deviation $\mathcal{E}_{\Phi}=0.037^0$ radian, where $\Delta\emptyset = |\emptyset - \emptyset_A|$.

TABLE 1. Optimization Of Full Rotation With Four Micro-Rotations

Φ^0	k(0), s_0	k(1), s_1	k(2), s_2	k(3), s_3
45	0,1	0,1	0,1	0,1
44	0,1	5,0	8,0	8,0
43	0,1	5,0	8,0	8,0
42	0,1	4,0	7,0	7,0
41	0,1	4,0	7,0	7,0
40	0,1	3,0	6,1	8,1
39	0,1	3,0	6,1	8,1
38	0,1	3,0	6,0	6,0
37	0,1	3,0	6,0	6,0
36	2,1	2,1	2,1	3,0
35	2,1	2,1	2,1	3,0
34	1,1	3,1	7,0	8,0
33	1,1	3,1	7,0	8,0
32	1,1	4,1	6,1	6,1
31	1,1	4,1	6,1	6,1
30	2,1	2,1	6,1	6,1
29	2,1	2,1	6,1	6,1
28	1,1	7,1	7,1	7,1
27	1,1	7,1	7,1	7,1
26	1,1	5,0	8,1	8,1
25	1,1	5,0	8,1	8,1
24	1,1	4,0	4,0	4,0
23	1,1	4,0	4,0	4,0

22	*2,1*	*2,1*	*3,0*	*10,1*
21	*2,1*	*2,1*	*3,0*	*10,1*
20	*1,1*	*3,0*	*7,0*	*7,0*
19	*1,1*	*3,0*	*7,0*	*7,0*
18	*1,1*	*2,0*	*4,1*	*4,1*
17	*1,1*	*2,0*	*4,1*	*4,1*
16	*2,1*	*6,1*	*10,1*	*10,1*
15	*2,1*	*6,1*	*10,1*	*10,1*
14	*1,1*	*2,0*	*7,1*	*7,1*
13	*1,1*	*2,0*	*7,1*	*7,1*
12	*3,1*	*4,1*	*8,1*	*10,1*
11	*3,1*	*4,1*	*8,1*	*10,1*
10	*3,1*	*9,0*	*9,0*	*9,0*
9	*3,1*	*9,0*	*9,0*	*9,0*
8	*3,1*	*9,0*	*9,0*	*9,0*
7	*3,1*	*9,0*	*9,0*	*9,0*
6	*3,1*	*5,0*	*7,0*	*9,1*
5	*3,1*	*5,0*	*7,0*	*9,1*
4	*4,1*	*7,0*	*9,0*	*9,0*
3	*4,1*	*7,0*	*9,0*	*9,0*
2	*6,1*	*9,1*	*9,1*	*9,1*
1	*6,1*	*9,1*	*9,1*	*9,1*

B. Implementation Of Micro-Rotations

Since the elementary angles and direction of micro-rotations are predetermined for the given angle of rotation, the angle estimation data-path is not required in the CORDIC circuit for fixed and known rotations. Moreover, because only a few elementary angles are involved in this where is the maximum number of shifts in the set of selected micro-rotations. The output of the barrel-shifters are loaded as the LSBs to the add/subtract units, and the MSBs of the corresponding operand of add/subtract unit are hardwired to 0.

A CORDIC circuit for complex constant multiplications is shown in Fig. 2. The LUT contains the control-bits for the number of shifts corresponding the micro-rotations to be implemented by the barrel-shifter and the directions of micro-rotations are stored in the sign-bit LUT. The major contributors to the hardware-complexity in the implementation of a CORDIC circuit are the barrel-shifters and the adders. There are several options for the implementation of adders [6], from which a designer can always choose depending on the constraints and requirements of the application. But, we have some scope to develop techniques for reducing the complexity of barrel-shifters over the conventional designs as discussed in the followings.

1) *Minimization of Barrel-Shifter Complexity by Hardwired Pre-Shifting:*

A barrel-shifter for maximum of S shifts for word-length L is implemented by $\log_2(S+1)$ -stages of demultiplexors, where each stage requires number of 1:2 line MUXes. The hardware-complexity of barrel-shifter, therefore increases linearly with the word-length and logarithmically with the maximum number of shifts. We can reduce the effective word-length in the MUXes of the barrel-shifters, and so also the number of stages of MUXes by simple hardwired pre-shifting as shown in Figs. 3. If is the minimum number of shifts in the set of selected micro-rotations, we can load only the more-significant bits (MSBs) of an input word from the registers to the barrel-shifters, since the less significant bits (LSBs) would get truncated during shifting. The barrel-shifter, therefore, needs to implement a maximum of shifts only, Therefore, the hardware-complexity of a barrel-shifter could be reduced by the hardwired pre-shifting approach. The time involved in a barrel-shifter could also be reduced by hardwired pre-shifting, since the delay of the barrel-shifter is proportional to the number of stages of MUXes, and it also be possible to reduce the number stages by hardwired pre-shifting.

In Table I, we find that the minimum number of shifts is greater than one in more than 75% of the cases. Using hardwired pre-shifting, it would therefore be possible to considerably reduce the total number of shifts to be implemented by barrel-shifters, so as to substantially reduce the hardware-complexity and delay of the barrel-shifters. A conventional barrel-shifter for maximum of shifts is implemented by S-stages of 2:1 MUXes. But, when the number of shifts is known a priori, one can design the barrel-shifter to include the specific shifts. For implementing four discrete shifts (see Table I) irrespective of the maximum number of shifts, the barrel-shifter would require three stages of 2:1 MUXes by hardwiring the shifts.

2) *Bi-Rotation CORDIC Cell:*

It is found that using only two micro-rotations, it is possible to get an accuracy up to 0.033 radian. Although the accuracy achieved by two micro-rotations is inadequate in many situations, but can be used for some applications where the outputs are quantized, e.g., in case of speech and image compression, etc., Besides, the rotations with four and six micro-rotations can also be implemented successively by two and three pairs of micro-rotations, respectively. Therefore, we design an efficient CORDIC circuit to implement a pair of micro-rotations, and named as bi-rotation CORDIC.

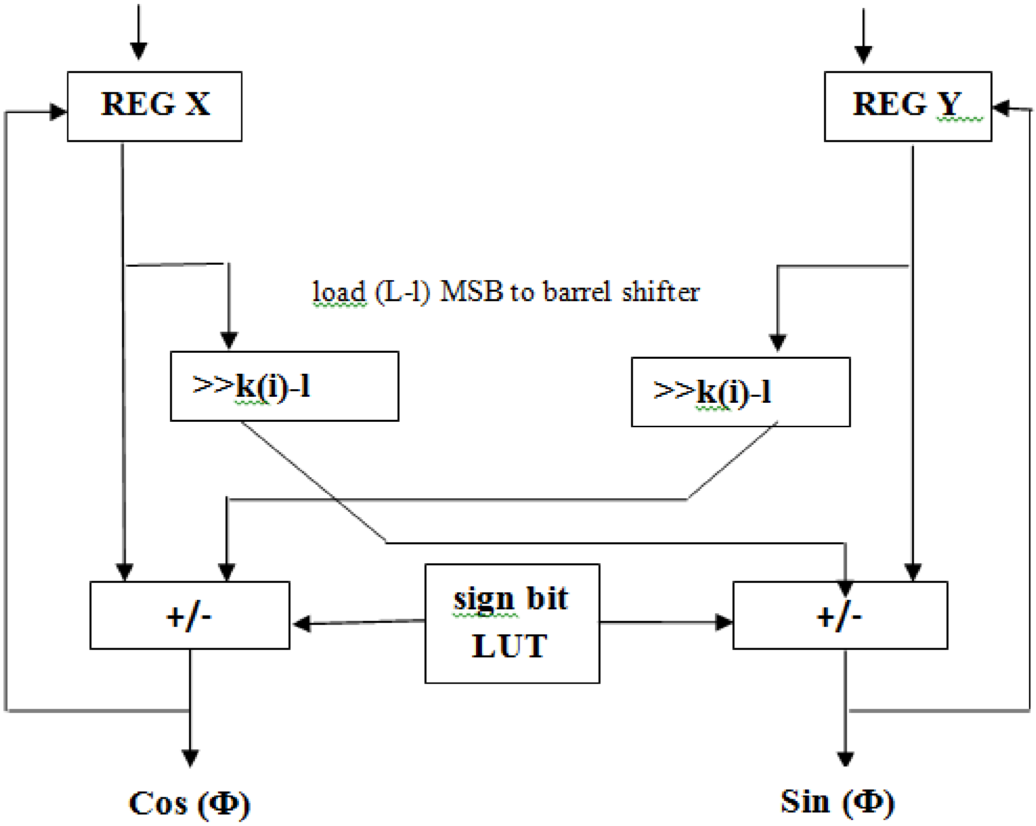


Fig. 3. CORDIC cell with hardware preshifting

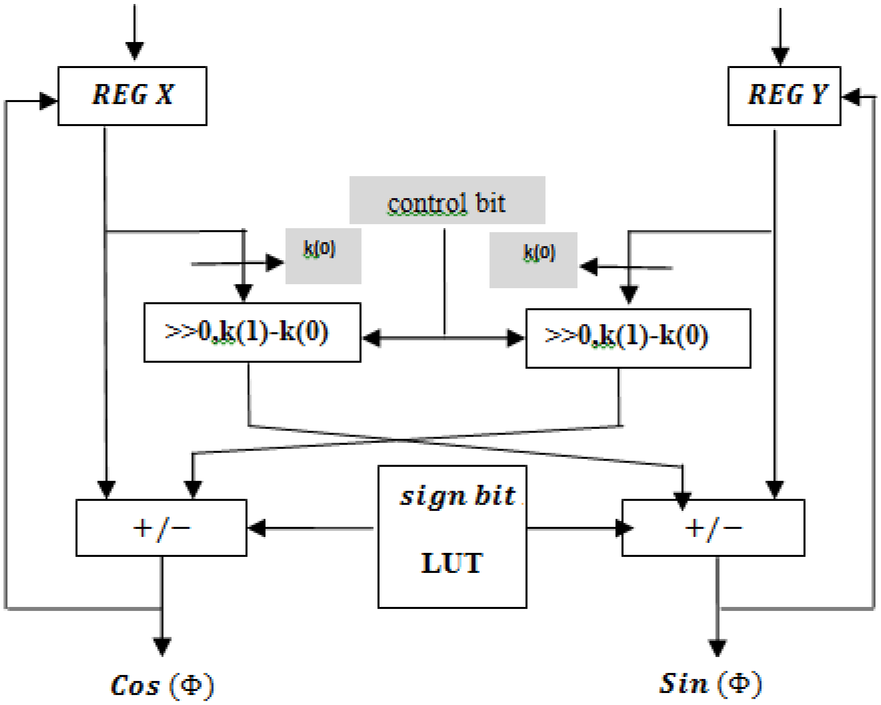


Fig. 4. Bi-Rotations CORDIC cell with hardware preshifting

The proposed circuit for bi-rotation CORDIC is shown in Fig. 4. It consists of an adder-module, two 2:1 multiplexers and a sign-bit LUT of two bit size. The adder-module consists of a pair of adders/subtractor. The adders/subtractor perform additions or subtractions according to the sign-bit available from the LUT. The components of the input vector are loaded to the input-registers .The output of the registers are sent in two lines where the content of the register is fed to one of the adders/subtractor directly while that in the other line is loaded to the barrel-shifter pre-shifted by bit-locations to right by hardwired pre-shifting technique. The output of the adders are loaded back to the input registers for the second CORDIC iteration. The bi-rotation

CORDIC involves only a pair of barrel-shifters consisting of only one stage of 2:1 MUXes. The control-bit for the barrel-shifters is 0 for the first micro-rotation (no shift) and 1 for the second micro-rotation (shif through).

3) *Modified CORDIC Cell*

A CORDIC Cell consist of an adder-module and a pair of barrel shifters . The adder-module consists of a pair of adders/subtractor and a sign bit LUT which determines whether to perform addition or subtraction. This can be further modified to reduce the device utilization .Each input has a barrel shifter to shift them with the specified value. In the modified CORDIC, instead of using separate barrel shifters for inputs Ux and Uy, a single barrel shifter is used. Inputs are selected with the help of a 2:1 multiplexer. When the select line to the multiplexer is 1,Ux will be given to barrel shifter an when select line is 0,Uy will be given to barrel shifter. Similarly, instead of using separate adder/subtractor to generate sine and cosine value, a single adder/subtractor is used. Input to the adder is selected using another 2:1 multiplexer. The modified CORDIC cell is shown in fig 5.This modification can be applied to Bi-Rotation CORDIC cell. as shown in fig 6.

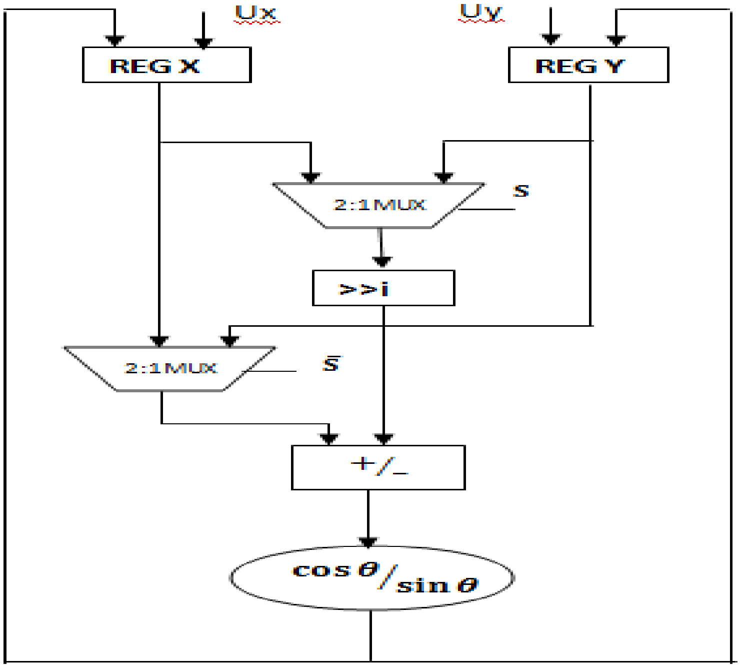


Fig.5. Modified CORDIC cell

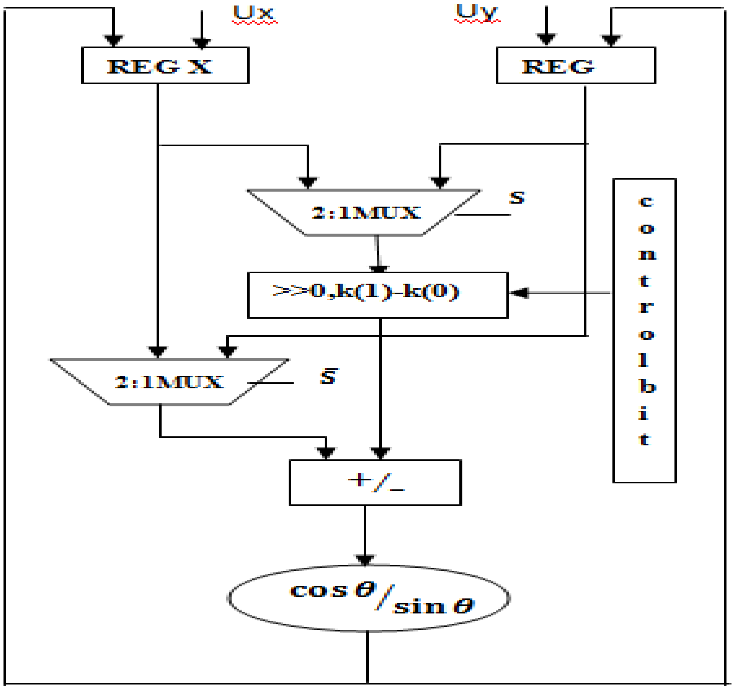


Fig. 6. Modified Bi-CORDIC cell

IV. SCALING OPTIMIZATION AND IMPLEMENATION

In order to match with the optimized set of elementary angles for the micro-rotations scaling optimization is done.

A. Scaling Approximation for Fixed Rotations

The generalized expression for the scale-factor can be expressed explicitly for the selected set of m1 micro rotations as

$$\mathrm{K} := \prod_{i=0}^{m_1-1} \frac{1}{[1+2^{-2k(i)}]} \qquad (7)$$

where k(i) for $0 \leq i < m_1$ is the number of shifts in the i^{th} micro rotation. Except for k(i) is 0 (ie.rotation by 45) can be expressed as

$$1 - \frac{x}{2} + \frac{3x^2}{8} - \frac{5x^3}{16} + \frac{35x^4}{128} - \frac{63x^5}{256} + \cdots \qquad (8)$$

where $x = 2^{-2i}$, i being the number of shifts in a micro-rotation, and can be expressed alternatively in terms of as

$$1 - \frac{1}{2^{2i+1}} + \frac{3}{2^{4i+3}} - \frac{5}{2^{6i+4}} + \frac{35}{2^{8i+7}} - \frac{63}{2^{10i+8}} + \cdots \quad (9)$$

Replacing each term in (4) by the expression of (6), we can obtain an approximate scale-factor as a product of shift-add terms of form

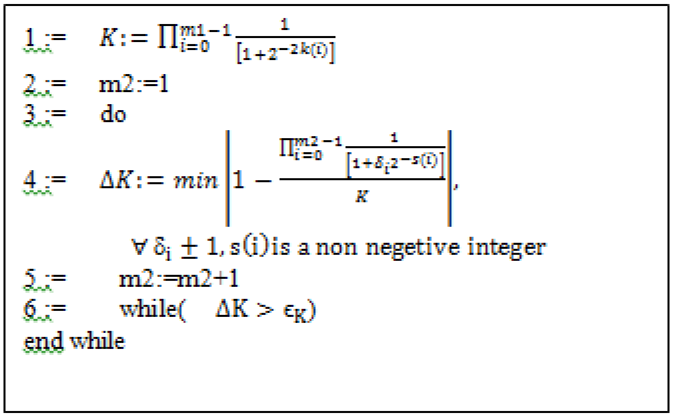
$$\mathrm{K_A} := \prod_{i=0}^{m_2-1} [1 + \delta_i 2^{-s(i)}\ ] \qquad (10)$$

where s(i) is the number of shifts performed for the i^{th} iteration of scaling and $\delta_i = \pm 1$ and m_2 is maximum number of scaling iterations required for the approximation.

The number of terms of (6), those are required to be accounted for to obtain the approximate scale-factor K_A [given by (7)] can be estimated according to value of and the desired output accuracy which is limited by the number of micro-rotations used for the pseudo rotation. The number of shifts-add/subtract terms in the expression of (7) is therefore minimized separately for the CORDIC implementations by four micro-rotations and six micro-rotations for different angles of rotation. It can be found that for four micro-rotation CORDIC implementation, where the error in $\emptyset$ is $\sim 0.04°$, only the first two terms in (6) contribute for $0 \leq i \leq 4$, while up to the third and the fifth terms contribute for $(0 \leq i \leq 2$ and $0 \leq i \leq 1)$ respectively. Similarly, for six micro-rotation CORDIC implementation, where the error in $\emptyset$ is $\sim 0.0005°$ the first two terms in (6) contribute for , while up to the third, fourth and fifth terms contribute for $(0 \leq i \leq 2\ ), (0 \leq i \leq 3)$ and $0 \leq i \leq 1)$respec-tively. Accordingly, we have obtained the recursive shift-add expressions of scale-factor K_A in the form of (7).

Algorithm 2 describes the optimization scheme to search the parameters k(i) and σ_i . Once the set of micro-rotations is obtained by **Algorithm 1**, the ideal scaling factor K can be calculated using (4). The objective function ΔK is defined as deviation of ${}^{K_A}/_{K}$from 1, i.e., . $\Delta K = \left|1 - {}^{K_A}/_{K}\right|$. The algorithm starts with the single term of scaling, then the number of scaling terms is increased by one until ΔK is smaller than the given maximum deviation $\in_k$ which needs to be set as the same value as $\in_\emptyset$ in the **Algorithm 1** since ΔK and $\Delta\emptyset$ contribute equally to the overall approximation error.

Algorithm 2 for scaling



```
1 :=   $K := \prod_{i=0}^{m1-1} \frac{1}{[1+2^{-2k(i)}]}$
2 :=   m2:=1
3 :=   do
4 :=   $\Delta K := min \left|1 - \frac{\prod_{i=0}^{m2-1} \frac{1}{[1+\delta_i 2^{-s(i)}]}}{K}\right|,$
           $\forall\ \delta_i \pm 1, s(i)$is a non negetive integer
5 :=      m2:=m2+1
6 :=      while(   $\Delta K > \epsilon_K$)
end while
```

TABLE 2 Optimized Shifts obtained from algorithm 2

Φ^0	s(0), t_0	s(1), t_1	s(2), t_2	K	K_A
45	1,0	1,0	1,0	0.7056	0.607
44	1,0	1,0	1,0	0.7056	0.607
43	2,0	4,0	9,1	0.7057	0.7018
42	2,0	4,0	9,1	0.7057	0.7018

41	2,0	4,0	8,1	0.7057	0.7059
40	2,0	4,0	8,1	0.7057	0.7059
39	2,0	4,0	9,0	0.7016	0.7018
38	2,0	4,0	9,0	0.7016	0.7018
37	2,0	4,0	9,0	0.7016	0.7018
36	2,0	4,0	9,0	0.7016	0.7018
35	3,0	5,1	8,1	0.9060	0.9059
34	3,0	5,1	8,1	0.9060	0.9059
33	3,0	6,1	10,0	0.8875	0.8878
32	3,0	6,1	10,0	0.8875	0.8878
31	4,0	4,0	6,1	0.8926	0.8926
30	4,0	4,0	6,1	0.8926	0.8926
29	4,0	8,1	8,1	0.9411	0.9412
28	4,0	8,1	8,1	0.9411	0.9412
27	4,0	5,0	6,0	0.8944	0.8940
26	4,0	5,0	6,0	0.8944	0.8940
25	4,0	5,0	6,0	0.8940	0.8940
24	4,0	5,0	6,0	0.8940	0.8940
23	4,0	4,0	6,1	0.8927	0.8926
22	4,0	4,0	6,1	0.8927	0.8926
21	4,0	8,0	8,0	0.9339	0.9338
20	4,0	8,0	8,0	0.9339	0.9338
19	3,0	6,1	10,0	0.8875	0.8878
19	3,0	6,1	10,0	0.8875	0.8878
17	3,0	7,0	9,0	0.8659	0.8665
16	3,0	7,0	9,0	0.8659	0.8665
15	5,0	10,1	10,1	0.9700	0.9697
14	5,0	10,1	10,1	0.9700	0.9697
13	3,0	7,0	7,0	0.8677	0.8682
12	3,0	7,0	7,0	0.8677	0.8682
11	7,0	9,0	9,0	0.9903	0.9903
10	7,0	9,0	9,0	0.9903	0.9903
s9	7,0	7,0	7,0	0.9918	0.9922
8	7,0	7,0	7,0	0.9918	0.9922
7	7,0	7,0	7,0	0.9923	0.9922
6	7,0	7,0	7,0	0.9923	0.9922
5	7,0	7,0	7,0	0.9918	0.9922
4	7,0	7,0	7,0	0.9918	0.9922
3	9,0	9,0	9,0	0.9980	0.9980
2	9,0	9,0	9,0	0.9980	0.9980
1	13,0	13,0	13,0	0.9999	0.9999

A. Implementation Of Scaling

Scaling and micro-rotations could be implemented either in the same circuit in interleaved manner or in two separate stages. The implementation of scaling as well as the micro-rotation would however depend on the level of desired accuracy, and the implementation of scaling also depends on the implementation of micro-rotations. Therefore, realization of the scaling circuits is discussed further.

4) *Generalized Implementation of Scaling:*

The generalized CORDIC circuit for fixed rotation to perform the micro-rotation and the scaling in interleaved manner in alternate cycles is shown in Fig. 7. It involves an additional line-changer circuit to change the path of unshifted (direct) input. The structure and function of line-changer is shown in Fig. 8. The line-changer is placed on the unshifted input data line to keep the critical path the same as that of Fig. 2.

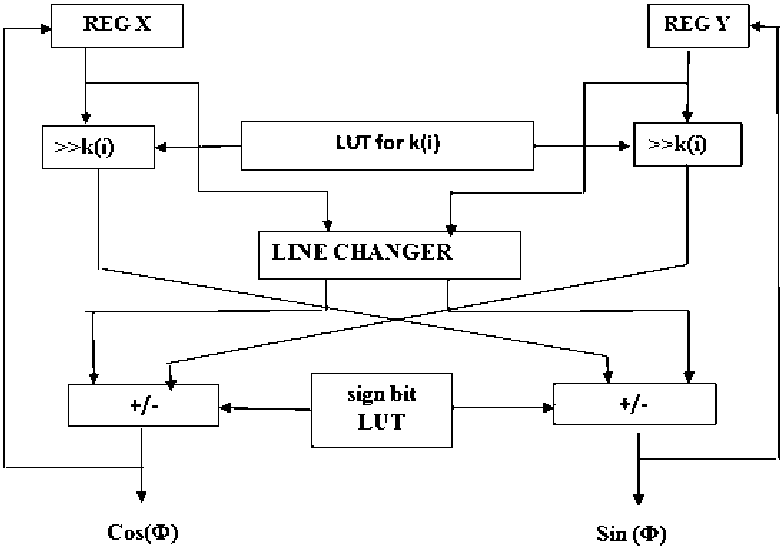


Fig. 7. Interleaved Scaling

If control bit =1 then
Y1 = X1 and Y2 = X2;
else
Y2 =X1 and Y1 = X2.

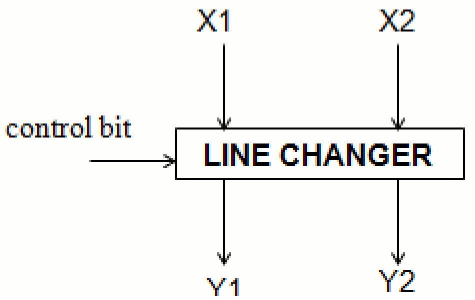


Fig. 8. Line Changer block diagram

V. ARGUMENT REDUCTION TECHNIQUE

So far using algorithm 1 and algorithm 2, the scaling parameter values are derived for angle in less than 45. Since both these algorithm perform exhaustive search to find the scaling parameters, an argument reduction technique. The main objective of the argument reduction technique is to uniquely map the results of a CORDIC rotation with a large target angle(for angle >45°) to the results of a CORDIC rotation with a relatively small target angle(ie. <45). To do this, we consider the four quadrants of the coordinate system. We first examine the CORDIC rotation of an input vector with target angle θ lying in the first quadrant(we need to consider only angle between 45 and 90). Thus angle θ can be mapped to an angle less than 45 degree by the equation Sine(θ) = Cos(90- θ). The angle falling in second, third and fourth quadrant can be either mapped to another angle Φ by using the following trigonometric equations respectively.

For Second quadrant (90 > $\theta \leq$ 180)

Φ = (180 - θ) (11)

Similarly for Third (180 > $\theta \leq$ 270) and Fourth quadrant (270 > $\theta \leq$ 360)

Φ = (270 - θ) and (12)

Φ = (360- θ) respectively. (13)

The resultant angle Φ always lies in the first quadrant.

Table 3 Comparison Table

Designs	NOS(%)	NOL(%)	NOA	NOM	Max Combinational path delay(ns)	max. Combinational frequency(MHz)
Reference Design (8 iterations)	38	35	49	836	7.701	117.81
Proposed Designs						
Separate scaling	27	25	16	368	11.923	82.311

(4 iterations)						
Interleaved Scaling (4 iterations)	30	27	19	554	9.937	62.850
Hardware Preshifting (4 iterations)	22	21	21	235	11.093	63.737
Bi-Rotation CORDIC (2 iterations)	11	10	10	71	12.724	75.546
Modified Bi-Rotation CORDIC (2 iterations)	7	7	6	67	9.937	62.850

NOS- No of Slice , NOL - No of LUT , NOA - No of adders, NOM - No of Multiplexers

VI. Results and discussion

Hardware and timing constraints of the optimized designs are discussed. A reference architecture is desired for straight forward implementation of fixed rotations. The complexity and performance of the proposed designs are compared with a reference design for a given accuracy.

The proposed design of CORDIC cell with separate scaling and with interleaved scaling offers better device utilization than the reference design. But the maximum combinational path delay is slightly higher for CORDIC cell with separate scaling and interleaved scaling. The proposed design to reduce the hardware complexity of barrel shifters offer lesser number of slices, LUT's and multiplexers when compared to the other proposed CORDIC cells with 4 iterations. The two proposed Bi-Rotation CORDIC cell offers even better device utilization when compared with CORDIC cell with 4 iterations. When comparing the two Bi-Rotation cells ,the modified Bi-Rotation CORDIC offers less number of adders and multiplexers than the other..Device Utilization and timing constraints are shown in Table3.

The reference design and the proposed design are simulated using Xilinx ISE Design suite 14.7 and implemented on Spartan 3E FPGA.

VII. CONCLUSION

CORDIC algorithm, is an iterative method to perform rotation and there for to compute sine values. It may not be the fastest method to calculate Sine functions, but when compared to the other approaches CORDIC is a clear winner when Hardware Multiplier is unavailable and when need to save the gates required to implement .

In this paper , a few optimized CORDIC designs are proposed .A reference design is also proposed for comparison. FPGA provides an ideal environment for design and implementation of digital functions. These designs are simulated using ISE Design suite 14.7 to verify the correctness and implemented on SPARTEN 3E FPGA. The proposed designs offer better device utilization and timing constraints than the reference design.

REFERENCES

[1] Pramod Kumar Meher and Sang Yoon Park, “CORDIC Designs for Fixed Angle Rotations”, *IEEE Trans. VLSI Systems*, vol. 21, no. 2, pp. 217, February. 2013.

[2] J. E. Volder, “The CORDIC trigonometric computing technique”, *IRE Trans. Electronic Computers*, vol. 8, no. 3, pp. 330{334, Sept. 1959.

[3] J.K Hu, “CORDIC based VLSI architecture for Digital Signal processing”, *IEEE Signal Process Mag,* vol. 9, no. 3, page 26-35, July. 1992.

[4] J. R. Cavallaro and F. T. Luk, “CORDIC arithmetic for a SVD processor,”*J. for Parallel Distrib. Comput.*, vol. 5, pp. 271–290, 1988.

[5] K. Maharatna, S. Banerjee, E. Grass, M. Krstic, and A. Troya, “Modified virtually scaling free adaptive CORDIC rotator algorithm and architecture,”*IEEE Trans. Circuits Syst. for Video Technol.*, vol. 15, no.11, pp. 1463–1474, Nov. 2005.

[6] A. R. Omondi, *Computer Arithmetic Systems: Algorithms, Architectures,and Implementations*. New York: Prentice-Hall, 1994.