Pipelined Cartesian-to-Polar Coordinate Conversion Based on SRT Division

Sung-Won Lee, Ki-Seok Kwon, *Student Member, IEEE*, and In-Cheol Park, *Senior Member, IEEE*

***Abstract*—This brief proposes a new Cartesian-to-polar coordinate conversion technique based on the radix-4 SRT division. The coarse quotient is used to derive the magnitude and the coarse phase by referring to tables, while the fine quotient is applied to linearly interpolate the fine phase to be added to the coarse phase. Compared to the CORDIC-based techniques, the proposed conversion requires less internal word-length and provides parallelism between internal stages, resulting in reduced computation latency and small chip area. A prototype chip designed using 0.25-μm CMOS technology occupies 0.203 mm^2, and post-layout simulations show maximum frequency of 400 MHz and power consumption of 170 mW at 2.5 V.**

***Index Terms*—Cartesian coordinate, computer arithmetic, coordinate conversion, polar coordinate.**

I. Introduction

CARTESIAN-TO-POLAR coordinate conversion is widely used in many digital communication applications such as Mary phase-shift keying (PSK) receivers, down conversion in CDMA and UMTS, general wideband PM/FM demodulators [1], and digital beam positioning [2]. Given a Cartesian input (X, Y), the magnitude and phase of the polar coordinate are represented as follows:

$$\rho = \sqrt{X^2 + Y^2} \tag{1}$$

$$\phi = \text{atan}(Y/X). \tag{2}$$

Rapid phase extraction is reported to be crucial for clock and carrier synchronization, particularly if a latency-sensitive feedback loop is employed in receivers [3], [4]. In many applications, the phase calculation is more time-consuming and requires more accuracy than the magnitude calculation.

There have been two methods to convert Cartesian coordinate to polar coordinate. The first method is to use a look-up table, which records a polar coordinate value for every possible input. Though this method is simple and provides the lowest latency, it has a drawback that the size of memories required to store the polar outputs is tremendous if the input bit-width is not small. For example, 2^{2n} entries are needed when both of the two inputs are represented in n bits. The second is to calculate the polar coordinate values based on the CORDIC algorithm [5], [6]. The CORDIC approach can be implemented without employing multipliers, but the computational latency and internal word-length are proportional to the output precision requirement. As those factors are directly related to conversion latency and chip area, implementing a high-precision high-speed CORDIC hardware leads to a considerable problem. Some speed-up techniques [7]–[10] have been proposed to reduce the latency at the expense of additional hardware resources, leading to increased chip area.

This brief proposes an area- and power-efficient coordinate conversion method that is based on the SRT division algorithm [11]. The proposed conversion requires less internal word-length and provides parallelism between internal stages, resulting in reduced computation latency and small hardware size compared to the CORDIC approach. Approximate division approaches such as division-by-convergence [12] and Newton-Raphson approximation [13] are not employed as they have to be associated with large-sized multipliers and it is hard to process them in parallel with other steps.

II. Related Works

As the conventional CORDIC vectoring mode takes iteration cycles as many as the required precision, it is not suitable for applications that need short latency. There have been two approaches for reducing the latency.

First, the truncation technique [7] replaces the last half of iteration cycles with a single rotation that approximates the remaining iterations. This relies on the fact that the early iterations in the original CORDIC algorithm give more significant contribution than the late iterations, because the rotation angle decreases as the iteration proceeds. This truncation reduces the latency at the cost of additional hardware resources required for the single rotation. The accuracy analysis of the remaining rotation angles yields the minimum number of iterations satisfying the given precision. On the other hand, this algorithm still suffers from the fact that the internal bit resolution must be large enough to prevent the accumulation of round-off errors in the early half iterations.

The second approach is based on the angle rotation with reciprocal calculation [8]–[10]. This is a lot similar to the first approach. The main difference is that the earlier CORDIC iterations are replaced with a sequence of reciprocal table lookup, multiplication and pseudo-angle rotation. Compared with the truncated CORDIC algorithm, this approach calls for many multipliers expensive in terms of chip area. In addition, the resulting magnitude is not as accurate as that of the truncated CORDIC algorithm.

Manuscript received January 4, 2007; revised March 24, 2007. This work was supported in part by Institute of Information Technology Assessment through the ITRC. This brief was recommended by Associate Editor S. Tsukiyama.

S.-W. Lee is with the Samsung Advanced Institute of Technology Yongin 449-712, Korea.

K.-S. Kwon and I.-C. Park are with the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Korea (e-mail: icpark@ee.kaist.ac.kr).

Digital Object Identifier 10.1109/TCSII.2007.898897

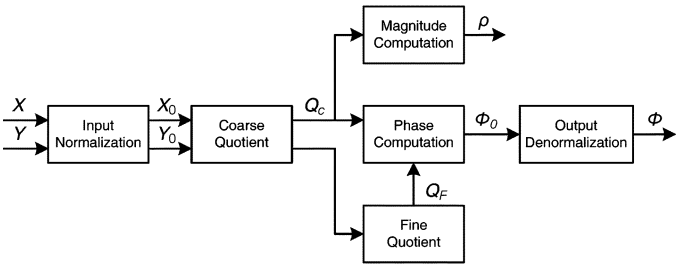

Fig. 1. Flow diagram of the proposed coordinate conversion.

III. Proposed Architecture

In the proposed method, the magnitude ρ is computed based on the following equation instead of (1), while the computation of phase ϕ is based on (2).

$$\rho = X \times \frac{\sqrt{X^2+Y^2}}{X} = X \times \sec\left(\operatorname{atan}\left(\frac{Y}{X}\right)\right). \quad (3)$$

To evaluate the equations, we first compute the quotient of (Y/X). The computed quotient Q is used to lookup a table containing the values of $\sec(\operatorname{atan}(Q))$, and then the lookup value is multiplied by X to compute the magnitude. At the same time, Q is also used to lookup a table containing $\operatorname{atan}(Q)$ to generate the phase. Unlike the CORDIC rotation, the division algorithm needs less bit resolution in the quotient calculation steps. Therefore, the coordinate conversion is accomplished with a division, a multiplication and two table lookups. Although this approach is simple and straightforward, there is a problem to be solved in practical implementations. The table size grows exponentially according to the bit-length of Q, leading to an impractical implementation if the bit-length of Q is large.

To reduce the table size, the accuracy of magnitude and phase is analyzed for communication applications. As practical applications require 12-bit phase accuracy and 6-bit magnitude accuracy under 15-bit output precision, the magnitude can be computed with only the upper part of Q, while the phase requires a more accurate quotient. Hence, the quotient is calculated in two steps. A coarse quotient Q_C is computed in the first step and a fine quotient Q_F is computed in the second step. The coarse quotient is used to generate the magnitude, and the fine quotient is used only for the phase computation. Using Q_C, we generate a coarse phase by referring to a coarse-grained arctangent table, and the final phase is computed by linearly interpolating the coarse phase with considering Q_F. In this approach, (2) and (3) are reformulated as follows:

$$\rho \approx X \times \sec\left(\operatorname{atan}(Q_C)\right) \quad (4)$$

$$\phi \approx \operatorname{atan}(Q_C) + \alpha(Q_C) \cdot Q_F \quad (5)$$

where $\alpha(Q_C)$ is a linear slope required to compute the piecewise-linear approximation (PWLA).

The division of Y/X is performed using the radix-4 SRT division algorithm in order to prevent the inherently iterative division from degrading the overall performance severely. As shown in Fig. 1, the overall flow of the proposed coordinate conversion is mainly composed of two stages, excluding the input normalization and output denormalization parts to be explained later. The first stage calculates Q_C, which is fed to two lookup tables, one for magnitude computation and the other for phase computation. Each entry of the phase table contains the slope and the coarse phase of Q_C, and each entry of the magnitude table contains the value of $\sec(\operatorname{atan}(Q_C))$ to be multiplied by X. The second stage continues the SRT division to obtain Q_F to be multiplied by the slope. This is a PWLA of the arctangent function and involves a modified radix-4 Booth's multiplication. In order to reduce processing latency, the multiplication is decomposed into a series of partial multiplications to compute the fine quotient and the phase in parallel, leading to shorter processing latency. Note that the previous works cannot support parallelism among the internal calculation steps.

If a specific application mandates high precision magnitude, we can replace the secant function with sine and cosine functions to calculate ρ by

$$\rho \approx X \times \cos\left(\operatorname{atan}(Q_C)\right) + Y \times \sin\left(\operatorname{atan}(Q_C)\right). \quad (6)$$

Though this equation needs two multipliers and one adder, its result is as accurate as the arctangent. Note that the magnitude computation algorithms described in [8], [10] do not make the same level of precision, even if interpolation algorithms are used.

A. Input Normalization and Output Denormalization

As the polar coordinate is $\pi/4$ symmetric, we relocate the input (X, Y) to the lower half of the first quadrant in order to reduce the resulting hardware complexity. Table I shows how to relocate the input to the region of [0, $\pi/4$]. In addition to this relocation, the input is scaled to meet the ranges of the partial remainder and the divisor allowable in the radix-4 SRT division where each quotient digit is restricted to $\{-2, -1, 0, 1, 2\}$. In the SRT division, the partial remainder should not be greater than two third of the divisor, and the divisor should be in the range of [0.5, 1]. If the input is relocated to (X_r, Y_r), X_r and Y_r are both positive and X_r is not less than Y_r. Therefore, X_r and Y_r are scaled by shifting both of them until the shifted X_r is in [0.5, 1], and then Y_r is shifted right by one bit to make it less than a half of the shifted X_r. This scaling has an additional effect of increasing the accuracy of results when X and Y have small values. The normalized input (X_0, Y_0) resulting from the relocation and scaling is applied to the radix-4 SRT division. To compensate the additional one-bit shifting of Y_r, we interpret the resulting quotient as being shifted right by one bit. Therefore, the quotient should be multiplied by 2 to compensate the additional shifting, which can be achieved by shifting left the resulting quotient by one bit. After the coarse quotient is computed, the magnitude is obtained through the following:

$$\rho \approx X_r \times \sec\left(\operatorname{atan}(Q_C)\right). \quad (7)$$

When the magnitude and phase are computed for the normalized input, the phase must be modified to compensate the relocation effect. There is no need to take into account the effect of scaling, because scaling by the same factor does not change the quotient value. The phase is relocated in reverse through the following transformation:

$$\phi = \delta(X, Y) + \gamma(X, Y) \cdot \phi_0(X_0, Y_0) \quad (8)$$

where δ and γ are correction factors determined by the given input (X, Y) as summarized in Table I.

TABLE I
INPUT NORMALIZATION AND PHASE DENORMALIZATION

Input Conditions			X_r	Y_r	δ	γ
$X \geq 0$	$Y \geq 0$	$\lvert X\rvert < \lvert Y\rvert$	Y	X	$\pi/2$	-1
$X \geq 0$	$Y \geq 0$	$\lvert X\rvert \geq \lvert Y\rvert$	X	Y	0	+1
$X < 0$	$Y \geq 0$	$\lvert X\rvert < \lvert Y\rvert$	Y	$-X$	$\pi/2$	+1
$X < 0$	$Y \geq 0$	$\lvert X\rvert \geq \lvert Y\rvert$	$-X$	Y	π	-1
$X < 0$	$Y < 0$	$\lvert X\rvert < \lvert Y\rvert$	$-Y$	$-X$	$-\pi/2$	-1
$X < 0$	$Y < 0$	$\lvert X\rvert \geq \lvert Y\rvert$	$-X$	$-Y$	$-\pi$	+1
$X \geq 0$	$Y < 0$	$\lvert X\rvert < \lvert Y\rvert$	$-Y$	X	$-\pi/2$	+1
$X \geq 0$	$Y < 0$	$\lvert X\rvert \geq \lvert Y\rvert$	X	$-Y$	0	-1

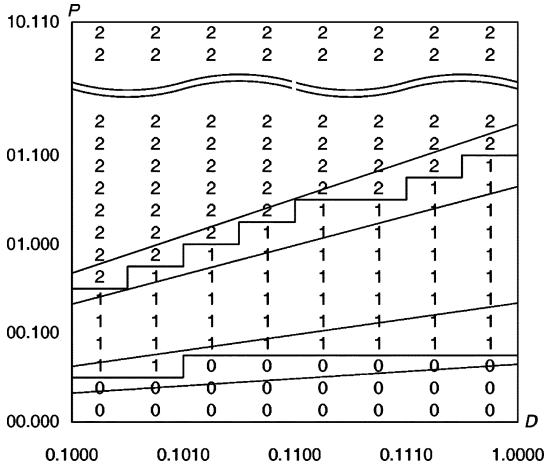

Fig. 2. $P - D$ plot for q_i selection.

B. Quotient Computation

The given input is normalized such that $0.5 \leq X_0 < 1$ and $Y_0 \leq X_0$ in the input normalization stage, which guarantees the highest accuracy during the radix-4 SRT division process. Each digit of the quotient $Q = 0.q_1 \cdots q_n$ is serially computed one by one, and a quotient digit q_i is selected from $\{-2, -1, 0, 1, 2\}$ in step i of the division process by comparing the partial remainder to the divisor. This comparison is the most complicated operation in the division process [13]. After k iteration steps, the quotient can be calculated by summing up all the quotient digits as follows:

$$Q_k = \sum_{i=1}^{k} q_i \cdot 4^{-i}. \tag{9}$$

As a quotient digit corresponds to 2 bits effectively, we need to iterate $n/2$ steps to generate an n-bit quotient Q, that is, $Q = Q_{n/2}$. Thus, the coarse quotient resulting from k iterations, where $k < n/2$, can be regarded as a coarse quotient that approximates the precise quotient of Y/X.

In each division step, the value of q_i can be selected by referring to the $P - D$ plot, where P and D stand for the partial remainder and the divisor, respectively. The divisor corresponds to X_0, and the initial remainder is set to Y_0. The partial remainder used in step i is 4 times the remainder computed in step $i - 1$. Fig. 2 illustrates how q_i is assigned when the partial remainder is positive. If the partial remainder is negative, we can use the same $P - D$ plot with negating the partial remainder and the quotient digit. Note that 4 most significant bits of the divisor and 5 most significant bits of the partial remainder are sufficient to determine the quotient digit correctly.

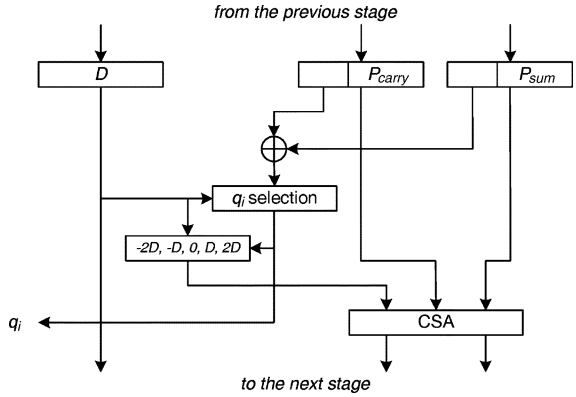

Fig. 3. General structure of the radix-4 SRT division step.

In order to speed up the division process, the partial remainder is represented in a redundant form consisting of sum and carry values. Since a truncated partial remainder is sufficient to select the next quotient digit, only 6 most significant bits of carry and sum values are actually taken to compute the truncated partial remainder. There is no need to complete the calculation of the partial remainder at any intermediate division step. Each division step has a block structure similar to Fig. 3.

The divisor X_0 derived from a given input (X, Y) is not changing throughout all the division steps. Thus, the quotient selection hardware does not raise serious hardware complexity. Fig. 4 shows the quotient selection circuit used in the proposed architecture. Before starting the division iterations, we read a column of the table shown in Fig. 4(a), which is a simplified version of the $P - D$ plot shown in Fig. 2. Several rows that are constant in Fig. 2 are dropped out, and each element is represented with one bit, as there are only two possible values in every non-constant row. If $P < 0$, the 1's complement of P is used as an input and the resulting quotient is negated, as shown in Fig. 4(b) where 5 most significant bits excluding the sign bit are inverted if the sign bit is negative and the output is saturated to take into account the upper part dropped in Fig. 4(a). Using the truncated partial remainder and the column values read from the simplified table, the quotient digit q_i can be determined simply by selecting one of the column values according to the truncated partial remainder, as depicted in Fig. 4(c). The required circuits are so simple that the critical path delay is also very short. Compared to the conventional radix-4 quotient selection circuitry requiring a programmable logic array (PLA) whose input and output bit-widths are 8 and 3, respectively, the proposed quotient selection logic is much more compact.

C. Fine Phase Computation

A coarse quotient value addresses one pair of a coarse phase and its slope. Then the fine quotient multiplied by the slope is added to the coarse phase to produce the final phase. To improve the latency, a parallel computation is introduced in multiplying the fine quotient by the slope.

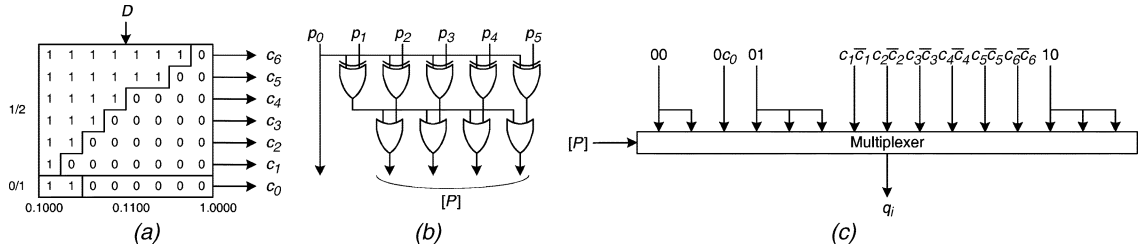

Fig. 4. Proposed quotient digit selection. (a) Table entry, (b) truncated partial remainder, and (c) q_i selection.

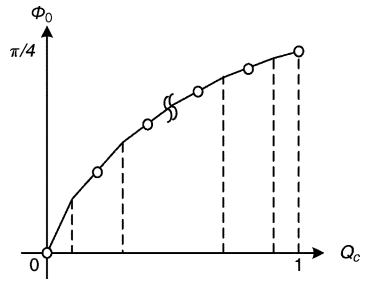

Fig. 5. Proposed PWLA of the arctangent function.

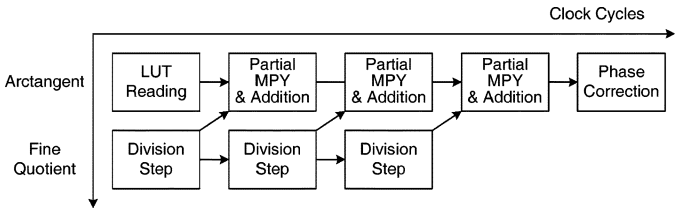

Fig. 6. Timing diagram of the fine phase compustation.

If we consider each division step performed to generate the fine quotient, (5) can be expanded as follows:

$$\phi_0 = \text{atan}(Q_C) + \sum_{i=c+1}^{n/2} \alpha(Q_C) \cdot q_i \cdot 4^{-i}$$
$$= \text{atan}(Q_C) + \alpha(Q_C) q_{c+1} 4^{-c-1} + \cdots + \alpha(Q_C) q_{n/2} 4^{-n/2}. \quad (10)$$

Instead of waiting for the completion of the fine quotient computation, the slope is multiplied as soon as a new quotient digit is determined in each division step. These intermediate products are serially summed up to achieve the precise phase in the range of $0 \le \phi_0 \le \pi/4$. As the quotient digit can be either positive or negative, we use the linear approximation scheme depicted in Fig. 5. Unlike the conventional PLA method, a coarse phase indicates the midpoint of a section except for the cases of $Q_C = 0$ and $Q_C = 1$.

The phase computation stage starts right after the coarse quotient has been computed. The fine quotient computation and the phase computation take place concurrently as shown in Fig. 6. Conventional linear interpolation algorithms guarantee n-bit accuracy, if $2^{n/2}$ sample points and interpolation terms (slope values) are available. In the sense of PWLA, the proposed approach is similar to the conventional method. However, a single quotient digit of the fine phase is encoded just like the modified Booth encoding as soon as it is generated in the division process. The encoded quotient digit multiplied by the slope acquired using the coarse quotient is serially added, which leads to a structure intermingling the division and the multiplication. In addition to making the latency short, the proposed intermingled structure yields higher clock frequency as it makes each stage simple.

TABLE II
OPTIMIZED BIT-WIDTH OF EACH DIVISION STEP NEEDED TO GUARANTEE THE ACCURACY

Bit-width of each division stage						Error (LSB)
1	2	3	4	5	6	
14	13	13	11	9	6	1

IV. IMPLEMENTATION DETAILS AND RESULTS

The division algorithm is modified to optimize performance for coordinate converter applications. First of all, the bit-widths of the partial remainder and divider in each division step are optimized to reduce the hardware complexity needed for carry-save adders and pipeline latches, as the final remainder is not necessary in the coordinate converter. Assuming 14-bit inputs and 6 division steps, a number of simulations are conducted to derive the minimal bit-width of each division step under the error constraint of less than 1 LSB. Table II summarizes the simulation result, where we can see that the bit width can be gradually reduced as the division step goes on.

In this configuration, a few latter division steps do not need the redundant partial remainder represented with sum and carry values, because the required bit-width is not large. In addition, if the number of bits is smaller than or equal to that required for the calculation of the truncated partial remainder, the redundant representation of the partial remainder is no longer necessary. In the latter division steps, the partial remainder can be quickly computed using a fast carry-propagation adder. From this observation, the last two division steps are merged into one step by duplicating the quotient selection circuit that is fast and compact. Merging the two steps can save some amount of hardware as well as one cycle of latency. Reducing the latency is important in communication applications [3].

Based on the proposed approach of coordinate conversion, a prototype chip is designed using 0.25-μm CMOS technology. The overall block diagram is shown in Fig. 7. The design is described in Verilog HDL and synthesized from the description. To achieve high throughput, a pipelined processing with 9 stages is employed in the design. The first two stages are dedicated to the input normalization that relocates and scales a given input. After three division steps are completed, we obtain the coarse quotient to be applied to the calculation of the magnitude and coarse phase. Additional three division steps are performed in

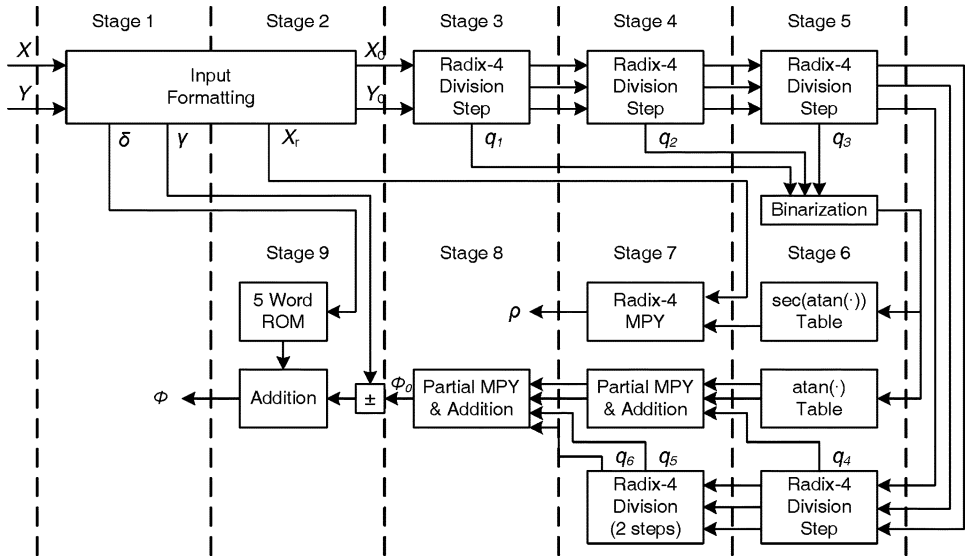

Fig. 7. Overall block diagram of the proposed coordinate converter.

TABLE III
PERFORMANCE COMPARISON

	Hwang [10]	This Work
Technology	0.25-μm, 5LM, 2.5V	0.25-μm, 5LM, 2.5V
Core area	0.484 mm^2 (including BIST)	0.203 mm^2
Input word-length	14 bit	14 bit
Output word-length	15 bit	15 bit
Accuracy of ϕ	12 bit	12 bit
Accuracy of ρ	6 bit	7 bit
Latency	19 cycles	9 cycles
Maximum frequency	406 MHz	400 MHz (post-layout)
Power consumption	470 mW	170 mW (post-layout)

two cycles to generate the fine phase in three cycles. The final cycle is to denormalize the calculated phase as expressed in (8).

The chip occupies 0.205 mm^2 and its equivalent gate count is 8500 approximately. The maximum phase error is 0.00024 or 12 fractional bits. The magnitude accuracy is 7 fractional bits. The maximum operating frequency is as high as 400 MHz and the power consumption of the core is less than 170 mW. The critical path is caused by Stage 7 shown in Fig. 7, performing the magnitude multiplication. Though the maximum frequency can be slightly improved by breaking the multiplication into two stages, this approach is not employed in the chip design because the resulting speedup is not noticeable and additional pipeline registers are required. Table III summarizes the performance obtained by post-layout simulation. Compared to the previous implementation reported as the most efficient one [10], the proposed approach reduces latency, area and power consumption.

V. CONCLUSION

A new coordinate conversion technique has been presented to overcome the drawbacks of the previous approaches such as long latency and large word length required in the CORDIC approach and large hardware complexity of the modified angle rotation technique. The proposed coordinate conversion approach is to directly use the radix-4 SRT division to reduce the latency and hardware complexity. The magnitude is calculated using only the coarse quotient, while the phase is computed by adding the coarse phase resulting from the coarse quotient to the linearly interpolated fine phase obtained by multiplying the slope by the fine quotient. Targeting the practical specification of 12-bit phase accuracy, the prototype design confirms that the proposed coordinate conversion approach is effective in reducing latency and power consumption as well as hardware cost.

REFERENCES

[1] B. S. Song and I. S. Lee, "A digital FM demodulator for FM, TV, and wireless," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 42, no. 12, pp. 821–825, Dec. 1995.

[2] C. Neri, G. Baccarelli, S. Bertazzoni, F. Pollastrone, and M. Salmeri, "Parallel hardware implementation of RADAR electronics equipment for a LASER inspection system," *IEEE Trans. Nucl. Sci.*, vol. 52, no. 6, pp. 2741–2748, Dec. 2005.

[3] M. P. Flitz and W. C. Lindsey, "Decision-directed burst-mode carrier synchronization techniques," *IEEE Trans. Commun.*, vol. 40, pp. 1644–1653, Oct. 1992.

[4] M. Andronico, S. Casale, A. La Corte, and C. Pinnisi, "A new algorithm for fast synchronization in a burst mode PSK demodulator," in *Proc. IEEE Int. Conf. Commun.*, 1995, pp. 1641–1646.

[5] J. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. 8, pp. 330–334, Sep. 1959.

[6] J. S. Walther, "A unified algorithms for elementary functions," in *Proc. Spring Joint Computer Conf.*, 1971, pp. 379–385.

[7] D. Timmermann, H. Hahn, and B. Hosticka, "Modified CORDIC algorithm with reduced iterations," *Electron. Lett.*, vol. 25, no. 15, pp. 950–951, Jul. 1989.

[8] D. Fu and A. N. Willson, Jr., "A high-speed processor for rectangular-to-polar conversion with applications in digital communications," in *Proc. IEEE Globecom*, Dec. 1999, vol. 4, pp. 2172–2176.

[9] D. Fu and A. N. Willson, Jr., "A two-stage angle-rotation architecture and its error analysis for efficient digital mixer implementation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 3, pp. 604–614, Mar. 2006.

[10] D. D. Hwang, D. Fu, and A. N. Willson, Jr., "A 400-MHz processor for the conversion of rectangular to polar coordinates in 0.25-μm CMOS," *IEEE J. Solid-State Circuits*, vol. 38, no. 10, pp. 1771–1775, Oct. 2003.

[11] J. E. Robertson, "A new class of digital division methods," *IRE Trans. Electron. Comput.*, vol. 7, pp. 218–222, Sep. 1958.

[12] S. F. Anderson, J. G. Earle, R. E. Goldschmidt, and D. M. Powers, "The IBM/360 model 91: Floating point execution unit," *IBM J. Res. Dev.*, vol. 11, pp. 34–53, Jan. 1967.

[13] I. Koren, *Computer Arithmetic Algorithms*. Natick, MA: Peters, 2002.