A low-latency parallel pipeline CORDIC

Article *in* IEICE Transactions on Electronics · April 2017

DOI: 10.1587/transele.E100.C.391

3 authors, including:

Hong-Thu Nguyen
The University of Electro-Communications
33 PUBLICATIONS **146** CITATIONS
SEE PROFILE

Xuan-Thuan Nguyen
University of Toronto
43 PUBLICATIONS **192** CITATIONS
SEE PROFILE

PAPER Special Section on Solid-State Circuit Design — Architecture, Circuit, Device and Design Methodology

A Low-Latency Parallel Pipeline CORDIC

Hong-Thu NGUYEN[†a)], **Xuan-Thuan NGUYEN**[†], *Nonmembers*, *and* **Cong-Kha PHAM**[†], *Member*

SUMMARY COordinate Rotation DIgital Computer (CORDIC) is an efficient algorithm to compute elementary arithmetic such as trigonometric, exponent, and logarithm. However, the main drawback of the conventional CORDIC is that the number of iterations is equal to the number of angle constants. Among a great deal of research to overcome this disadvantage, angle recording method is an effective method because it is capable of reducing 50% of the number of iterations. Nevertheless, the hardware architecture of this algorithm is difficult to implement in pipeline. Therefore, a low-latency parallel pipeline hybrid adaptive CORDIC (PP-CORDIC) architecture is proposed in this paper. In the design hybrid architecture was exploited together with pipeline and parallel technique to achieve low latency. This design is able to operate at 122.6 MHz frequency and costs 8, 12, and 15 clock cycles latency in the best, average, and worst case, respectively. More significantly, the latency of PP-CORDIC in the worst case is 1.1X lower than that of the Altera's commercial floating-point sine and cosine IP cores.

key words: *CORDIC, pipeline, low-latency*

1. Introduction

COordinate Rotation DIgital Computer (CORDIC) was invented by J.E. Volder [1] in 1959 and then developed by J.S. Walther [2] in 1971. Because of its strong advantages of simple and efficient methodology with low cost implementation, CORDIC has been utilized widely in various mathematical functions such as trigonometric, exponent, and logarithm. It is also employed in many applications such as signal processing, image processing, communication, robot manipulation, and graphics and animation [3].

In conventional CORDIC, there is a trade-off between latency and precision. In other words, the more the number of CORDIC iterations is, the more latency and precision the system achieves [4]. Many methods which aimed to enhance the precision along with reducing latency were proposed. Among them, Y.H. Hu et al. [5], [6], proposed angle recording method that could shrink the number of iterations to 50%. Besides, parallel angle recording CORDIC algorithm presented by K.R. Terence et al. [7] could choose all angle constants in one step. However, it required a large number of comparison circuits. T.B. Juang et al. [8] proposed a parallel CORDIC algorithm which generated the rotation direction by directly examining the input data without checking the sign of residual angle after performing the addition or subtraction. They used the binary-to-bipolar recording and micro-rotation angle recording methods to implement the parallel architecture. P.K. Meher et al. [4] utilized angle recording scheme for rotating a fixed angle that could be utilized in specific application areas such as robotics, graphics, games, and animation.

CORDIC fixed-point format architecture is attractive by its simple calculation and sufficient precision. For example, a fixed-point CORDIC-based Fast Fourier Transform (FFT) [9], [10] played an essential role in most of multimedia and wireless communication applications. However, the main drawback of fixed-point systems is the small range of input data. On the other hand, the floating-point CORDIC design satisfied high precision applications with wide range of input data. There were several floating-point implementations have been suggested. P. Surapong et al. [11] proposed a design of pipelined floating-point CORDIC architecture for high-accuracy results for the phase and magnitude detectors. N. Dhume et al. [12] proposed a three-step scheme that converted the input from floating-point format to the fixed-point format, then computed the fixed-point CORDIC trigonometric functions, and finally transformed those results into IEEE 754 floating-point format. Jie Zhou at el. [13] follows the steps in [12] and designs a CORDIC calculation with 34 pipeline stages implemented on both Altera and Xilinx FPGA.

Besides plenty of research in FPGA, there is a large amount of CORDIC implementation research in ASIC. R.K. Jain et al. [14] designed a ROM-free CORDIC architecture which used CSA (carry save adder) and CLA (carry look ahead adder) to reduce the resource utilization and power consumption. S. Kumar et al. [15] developed a hybrid architecture that also reduce the ROM utilization and required less power than that of the classical CORDIC. Another CORDIC implemented in 180 nm CMOS technology was proposed by K.C. Ray et al. [16], the power of which was found to be 350 mW at 125 MHz operating frequency.

A parallel pipeline CORDIC (PP-CORDIC) was proposed in previous paper [17]. In order to create a CORDIC design with low-latency and floating point precision results, some improvements in hardware architecture of PP-CORDIC are proposed in this paper. The techniques utilized in hardware architecture are shown below:

- Parallel architecture: The PP-CORDIC contains several CORDIC-function modules which operate in parallel. The number of CORDIC-function modules is

Manuscript received July 31, 2016.
Manuscript revised October 18, 2016.
[†]The authors are with the Faculty of Electronic Engineering, The University of Electro-Communications, Chofu-shi, 182–8585 Japan.
a) E-mail: hongthu@vlsilab.ee.uec.ac.jp
DOI: 10.1587/transele.E100.C.391

equal to the maximum iterations of input angles. For example, if the input angle requires maximum five iterations, the PP-CORDIC includes five CORDIC modules which operate in parallel and each of them receives data sequentially in turn. Because of this structure, to the best of our knowledge, PP-CORDIC could be applied for fixed and known angle which is widely utilized in robotics, graphics and animation [3], [4], [18]. Moreover, the parallel processing is employed in many significant modules of the design.

- Pipeline processing: By utilizing several CORDIC-function modules which operate in parallel, the PP-CORDIC is capable of operating in pipeline. Furthermore, pipeline processing is deployed in each small module of PP-CORDIC to improve the throughput as well as latency.

Firstly, the latency and the precision of proposed architecture was proved by MATLAB software. Secondly, the hardware structure is designed by Verilog HDL, simulated in Modelsim, and verified by Altera Quartus 13.0 with Statrix IV FPGA target. Because of the trade-off between accuracy and resource utilization, the hardware architecture contains 16 predefined input angles. There are eight configurations of PP-CORDIC which the number of CORDIC-function modules changes from one to eight. Finally, the PP-CORDIC with two CORDIC-function modules was implemented on 180 nm CMOS technology.

The remainder of this paper is organized as follows. Section 2 summarizes the conventional and angle recording CORDIC algorithm. Section 3 describes the PP-CORDIC hardware architecture. The experimental results will be shown in Sect. 4. Finally, Sect. 5 is the conclusion.

2. Algorithm

2.1 Traditional CORDIC Algorithm

Figure 1 illustrates a vector $V_0(x_0, y_0)$ in the two-dimensional plane. The coordinate value of new position when this vector rotate an angle Φ is calculated by (1).

$$\begin{aligned} x_n &= x_0 \cos\Phi - y_0 \sin\Phi \\ y_n &= y_0 \cos\Phi + x_0 \sin\Phi \end{aligned} \tag{1}$$

By using the conventional CORDIC algorithm, (1) can

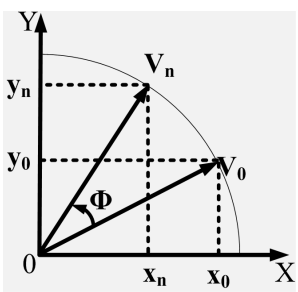


Fig. 1 A coordinate of a vector in two-dimentional plane.

be implemented by following step: 1) decompose the angle Φ into a sequence of predefined angles; 2) skip the scale factor in each micro-rotations; 3) multiply with a scale factor to get the final results.

Let call N is the number of constants in set of predefined angles. Each angle in this set is θ_i, where $i \in \{0, \ldots, N\}$, with the condition is $\tan\theta_i = 2^{-i}$. Equation (1) is rewritten using CORDIC algorithm as seen in (2).

$$\begin{aligned} x_{i+1} &= x_i - d_i y_i 2^{-i} \\ y_{i+1} &= y_i + d_i x_i 2^{-i} \\ z_{i+1} &= z_i - d_i \theta_i \\ d_i &= sign(z_i) \end{aligned} \tag{2}$$

where z_i, x_i, y_i and z_{i+1}, x_{i+1}, y_{i+1} is the residual angle and the coordinate values in the rotation i and $i+1$, respectively. Finally, the coordinate results of new position is calculated by (3).

$$\begin{aligned} K &= \prod_{i=0}^{N-1} k_i = \prod_{i=0}^{N-1} \cos\theta_i \\ x_n &= x_{N-1} \times K \\ y_n &= y_{N-1} \times K \\ \hat{\Phi} &= \sum_{i=0}^{N-1} d_i \times \theta_i \end{aligned} \tag{3}$$

where x_n, y_n is the final results of coordinate value, K is the scale factor of this algorithm and $\hat{\Phi}$ is approximate with input angle Φ.

2.2 Angle Recording Method

The angle recording algorithm is proposed to reduce the number of iterations, which, in turn, leads to the reduction of latency and the decrease of resource cost. The idea of the angle recording method is to choose not all angles constants but only several angles in the set while maintain the similar results. For decision-making in each step, the algorithm utilizes the concept of c, which express the range of residual angles around one angle constant θ_i. The value of c_i is defined by (4), and the values of the set of parameter c can be seen in Table 1. As mentioned above, a fixed set of θ_i leads to a constant scale factor k_i. That means a dynamic set of input angle Φ, as in the angle recording method, will need a dynamic set of scale factor K.

$$c_i = \begin{cases} (\theta_i + \theta_{i+1})/2 & \text{if } 0 \le i \le (N-2) \\ \theta_i/2 & \text{otherwise} \end{cases} \tag{4}$$

Algorithm 1 is the pseudo-code of the angle recording method. Firstly, the residual angle is the same with the initial input angle, as illustrated in line 1. The angle constant is search from the first to the final one using variance i, and the number of chosen angles is marked by variance j. In the beginning, the value of scale factor are set 1. While current

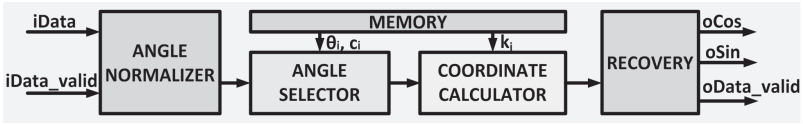


Fig. 2 The hardware architecture of CORDIC-function.

Table 1 The values of θ_i, c_i, and k_i when the number of predefined angles is 16.

i	θ	c	k
0	45.000000000	35.782525588	0.707106781
1	26.565051177	20.300647322	0.894427191
2	14.036243468	10.580629908	0.970142500
3	7.125016349	5.350675362	0.992277877
4	3.576334375	2.683122491	0.998052578
5	1.789910608	1.342542159	0.999512076
6	0.895173710	0.671393940	0.999877952
7	0.447614171	0.335712335	0.999969484
8	0.223810500	0.167858088	0.999992371
9	0.111905677	0.083929284	0.999998093
10	0.055952892	0.041964672	0.999999523
11	0.027976453	0.020982340	0.999999881
12	0.013988227	0.010491170	0.999999970
13	0.006994114	0.005245585	0.999999993
14	0.003497057	0.002622792	0.999999998
15	0.001748528	0.000874264	0.999999999

Algorithm 1: The pseudocode of angle recording method.

```
1  z_0 = Φ
2  i = j = 0
3  K = 1
4  while |z_j| > c_15 and i < N do
5      if |z_j| ∈ (c_{i+1}; c_i] then
6          z_{j+1} = z_j − sign(z_j) × θ_i
7          x_{j+1} = x_j − sign(z_j) × y_j × 2^{−i}
8          y_{j+1} = y_j + sign(z_j) × x_j × 2^{−i}
9          K = K × k_i
           j = j + 1
       end
       i = i + 1;
   end
```

residual angle z_j is still larger than the threshold c_{15}, and the order of current comparison angle i is smaller than the number of predefined angle N, the loop will be repeated, shown in line 4. If the residual angle is in range of $(c_{i+1}, c_i]$, the angle constant θ_i is chosen, depicted in line 5. Therefore the next coordinate position (x_{j+1}, y_{j+1}), the next residual angle z_{j+1}, and the scale factor K will be updated, as illustrated in line 6 to 9. The algorithm will be end until the residual angle smaller than the threshold or the order of comparison angle achieves the last one. It should be noticed that the approach of the method is to choose the angle θ_i in each iteration step j in the way that the residual angle z_j will quickly converges to zero. Subsequently, there are only several angle constants are chosen and the equation utilized to update the value of $x_{j+1}, y_{j+1}, z_{j+1}$, and K is a little different with the (2), (3).

In order to specify the difference between the conventional and angle recording CORDIC algorithm, an example of 38° input angle is described in (5) and (6). The former equation is the conventional CORDIC and the latter one is the angle recording CORDIC algorithm. In (5), the conventional CORDIC algorithm requires 16 iterations whereas the angle recording CORDIC algorithm in (6) only spends fours angle constants to obtain the approximation of 38°.

$$(\theta_0 - \theta_1 + \theta_2 + \theta_3 - \theta_4 + \theta_5 + \theta_6 - \theta_7 - \theta_8 \quad (5)$$
$$-\theta_9 + \theta_{10} + \theta_{11} - \theta_{12} + \theta_{13} + \theta_{14} - \theta_{15}) = 38.0003052^\circ$$
$$(\theta_0 - \theta_3 + \theta_9 + \theta_{12}) = 38.0008545^\circ \quad (6)$$

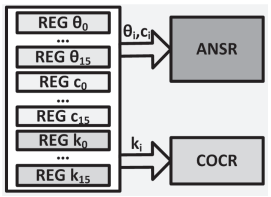


Fig. 3 The hardware architecture of MEM.

3. Implementations

Figure 2 shows the proposed architecture of CORDIC-function implementation using angle recording CORDIC algorithm. The input data is 24-bit fixed-point input angle in degree, the format of which is 1.8.15, i.e. 1-bit sign, 8-bit magnitude, and 15-bit LSBs. The outputs are two 32-bit floating-point sine and cosine results. The hardware architecture contains five significant parts: MEMORY (MEM), ANGLE NORMALIZER (ANOR), ANGLE SELECTOR (ANSR), COORDINATE CALCULATOR (COCR), and RECOVERY (REC). First of all, the predefined angles θ_i, the c_i parameters, and scaling parameter k_i are stored in MEM. The input data *iData* and valid signal *iData_valid*, firstly, are sent to ANOR for normalizing it to a predefined range $[0°, 45°]$. Secondly, ANSR determines which angle is chosen by using angle recording method. COCR then operates based on (2) with all of the information received from ANSR. Finally, REC module returns the final results of initial input data.

3.1 Memory (MEM)

MEM contains 48 registers REG θ_i, REG c_i, and REG k_i to store the values of θ_i, c_i, and k_i, respectively, where $i \in \{0, \ldots, 15\}$, as illustrated in Fig. 3. The values of θ_i and c_i are sent to ANSR to determine which angle is chosen by angle recording CORDIC. The values of k_i is transferred to COCR to calculate scale factor K. As mentioned in the algorithm part, depend on the input angle, the angle recording method

chooses the dynamic set of θ_i which leads to dynamic value of K. For example, if the input angle $\Phi = 30^\circ \simeq \theta_1 + \theta_4 - \theta_9 - \theta_{11} - \theta_{15}$, its scaling factor value is $K_{30^\circ} = k_1 \times k_4 \times k_9 \times k_{11} \times k_{15} = 0.892683553$

3.2 Angle Normalizer (ANOR)

$$\Phi_{nor} = \begin{cases} \Phi & \text{if } \Phi \in [0^\circ, 45^\circ] \\ -\Phi & \text{if } \Phi \in (-45^\circ, 0^\circ] \\ \Phi + 90^\circ & \text{if } \Phi \in [-90^\circ, -45^\circ) \\ -\Phi - 90^\circ & \text{if } \Phi \in (-90^\circ, -135^\circ] \\ \Phi + 180^\circ & \text{if } \Phi \in (-135^\circ, -180^\circ] \\ 180^\circ - \Phi & \text{if } \Phi \in (135^\circ, 180^\circ] \\ \Phi - 90^\circ & \text{if } \Phi \in (90^\circ, 135^\circ] \\ 90^\circ - \Phi & \text{if } \Phi \in (45^\circ, 90^\circ] \end{cases} \quad (7)$$

The CORDIC algorithm only converges if the input angle is in the range below the sum of entire N predefined angles. In this design, this range is between -99.88° to 99.88°. Moreover, the trigonometric values of any angle in the trigonometry circle can be estimated from that of an angle in the range $[0^\circ, 45^\circ]$. Because of two reasons mentioned above, this design normalizes all of input angles into the predefined range $[0^\circ, 45^\circ]$ by using (7). Since the range $[-180^\circ, 180^\circ]$ is the whole trigonometry circle, the input angle in this range is examined. After calculating the sine and cosine results of normalized angle, the *Recovery information* and *Recovery equation* in Table 2 are utilized to interpolate the true sine and cosine values of input angle.

The hardware architecture of ANOR is described in Fig. 4. The input signal *iData* is compared in parallel with the Range in Table 2 by CMP modules to determine the range of input angle. This parallel structure is utilized because of the low-latency purpose. When the range of *iData* is determined, the normalized angle *data_nor* will be calculated by simple adding or subtracting using ADD and SUB modules, respectively, based on (7).

3.3 Angle Selector (ANSR)

The ANSR contains three significant modules, which are SET NEXT PHASE (SNP), GET SIGN (GSN) and CHECK LAST (CLT), as illustrated in Fig. 5. The input signal is the normalized angle *data_nor* and recovery information nor_info, which are received from the ANOR, and θ_i and c_i which are transferred by MEM. Module SNP chooses which phase is the selected while module GSN determine the direction of the phase, and module CLT checks whether the rotation is the last nor not. Finally, the collection of 4-bit recovery information nor_info, 2-bit last rotation ls, 1-bit sign d, and 4-bit position of selected angle constant i are transferred to COCR to do the next step. The parallel processing of three main modules in ANSR aim to save the latency.

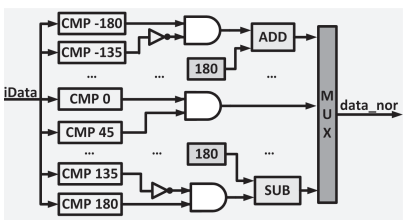


Fig. 4 The hardware architecture of ANOR.

3.4 Coordinate Calculator (COCR)

Figure 6 illustrates the architecture of COCR implementation. The input signals of this module contain the sign value d, the position of selected angle i, the last rotation signal ls, and the recovery information nor_info, which are received from ANSR module. Besides, it also contains the k_i value which is received from the MEMORY module. COCR includes five main modules which are FADD SUB X, FADD

Table 2 The recovery information and equations that used for interpolating final output results.

Range	Recovery information	Recovery equations
$[0^\circ, 45^\circ]$	00 11	$\sin\Phi = y_n$
		$\cos\Phi = x_n$
$(-45^\circ, 0^\circ]$	01 00	$\sin\Phi = -y_n$
		$\cos\Phi = x_n$
$(-90^\circ, -45^\circ]$	11 10	$\sin\Phi = -x_n$
		$\cos\Phi = -y_n$
$(-135^\circ, -90^\circ]$	11 11	$\sin\Phi = -x_n$
		$\cos\Phi = -y_n$
$(-180^\circ, -135^\circ]$	01 01	$\sin\Phi = -y_n$
		$\cos\Phi = -x_n$
$(135^\circ, 180^\circ]$	00 01	$\sin\Phi = y_n$
		$\cos\Phi = -x_n$
$(90^\circ, 135^\circ]$	10 11	$\sin\Phi = x_n$
		$\cos\Phi = -y_n$
$(45^\circ, 90^\circ]$	10 10	$\sin\Phi = x_n$
		$\cos\Phi = y_n$

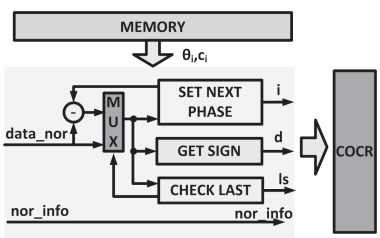


Fig. 5 The hardware architecture of ANSR.

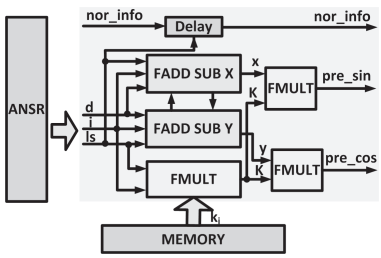


Fig. 6 The hardware architecture of COCR.

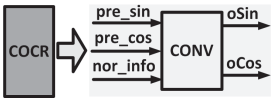


Fig. 7 The hardware architecture of REC.

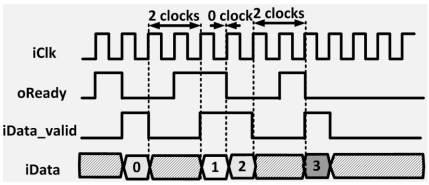


Fig. 8 The overview of CORDIC-function operation.

SUB Y, and three FMULT modules. In this architecture, FADD SUB X, FADD SUB Y and one module FMULT operate in parallel to calculate the coordinate value before scaling, called (x_n, y_n) and scale factor K, respectively. Two other FMULT modules are employed to calculate the sine and cosine results of normalized angle, which is namely *pre_sin* and *pre_cos*. Delay module is controlled by *ls* signal to make sure the value of *nor_info* are transferred to the next module in the same time with *pre_sin* and *pre_cos*. Although the architecture can use the resource sharing technique to save the resource cost, we proposed the architecture utilized the parallel technique for easing the latency.

3.5 Recovery (REC)

The *pre_sin*, *pre_cos* and *nor_info* values that are transferred by the COCR module are significant input signals of REC module, as illustrated in Fig. 7. Base on these input data, the true sine and cosine values of input angle are calculated based on the equation in Table 2, with the parameter (x_n, y_n), and $(\sin\Phi, \cos\Phi)$ are the value of (pre_sin, pre_cos) and $(oSin, oCos)$, respectively.

3.6 Parallel Structure

The angle recording method mentioned in Sect. 2 has different micro-rotations for each input angle. Therefore, the proposed hardware architecture above is not able to receive data continuously if the number of rotations of input data varies. The operation of above design is depicted in an example in Fig. 8. In the figure, the number of iterations of each input angle is different, i.e. the iterations of the input angles are two, zero, and two, respectively. Obviously, the next input angle is not capable of accepting immediately in the next clock. After finishing the process of one input angle, the CORDIC-function module will send the ready signal *oReady* to notify that it can receive the new input angle. In the next clock, the *iData_valid* signal is active, and the next input angle will be sent to CORDIC module.

In order to receive the input data in pipeline, the parallel pipeline CORDIC (PP-CORDIC) is proposed in this paper. Its architecture is illustrated in Fig. 9, which contains ANGLE NORMALIZER, MEMORY, RECOVERY

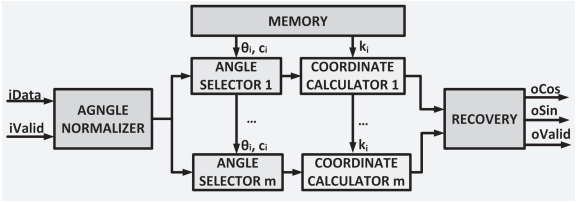


Fig. 9 The hardware architecture of parallel pipeline CORDIC.

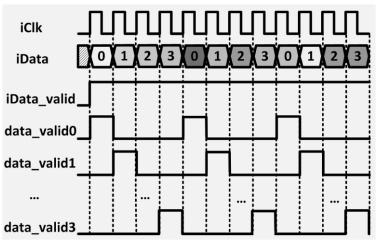


Fig. 10 The overview of PP-CORDIC operation.

and m ANGLE SELECTORs and COORDINATE CALCULATORs. In this design, m is the same with the number of iterations that input angle requires, and m ANSRs and COCRs operate in parallel. Because the angle recording CORDIC requires from zero to eight iterations in case $N = 16$, there are eight PP-CORDIC configurations with the value of m is from one to eight were proposed in this paper. In case the input angle are fixed and known, the specific PP-CORDIC configuration is chosen easily. On the other hand, if the iteration of input angle is unknown or varied, the recommendation architecture is the one which contains eight CORDIC modules. For this reason, to our best knowledge, the PP-CORDIC is proper for fixed and known angles.

Depending on the number of iterations of the input angle in the application, the architecture of the system can be utilized several CORDIC modules, i.e. if the input angles require maximum four iterations, the architecture will contain four CORDIC modules. The operation of PP-CORDIC on this example is illustrated in Fig. 10. Let call m, where $m \in \{0, 1, 2, 3\}$, is the order of each HA-CORDIC module. The figure shows that the signal *data_valid m* is turned on alternatively and repeated after each four clocks. Because after four clocks, all of the CORDIC-function is ready for a new input angle, the signal *iData_valid* is always active and *iData* is capable of putting in consecutively.

4. Experimental Results

Firstly, the angle recording CORDIC algorithm is verified by MATLAB software with $N = 16$ predefined angles. The number of iterations and the Mean Square Error (MSE) of the estimated angle are calculated and made the comparison with conventional CORDIC. The results in software mean the angle recording is a proper method for low latency architecture. Secondly, the PP-CORDIC hardware is employed in Altera Quartus 13.0 with Stratix IV FPGA target. The resource utilization, frequency and latency of proposed design is compared with other floating point CORDIC design.

Table 3 The comparison between PP-CORDIC with other floating-point systems.

	Device Family	Process (nm)	LUTs	Registers	DSPs	Frequency (MHz)	Latency (clock cycles)	Latency (ns)
[12]	Xilinx Virtex 7		6,514	4,725	9 (25x18 bit)	280.0	130	464.3
PP_CORDIC_8	Altera Stratix V		9,414	2,834	24 (27x27 bit)	154.1	15	97.3
PP_CORDIC_5	Altera Stratix V	28	6,667	2,083	15 (27x27 bit)	154.1	12	77.9
PP_CORDIC_1	Altera Stratix V		3,007	1,087	3 (27x27 bit)	154.1	8	51.9
[13]a	Xilinx Virtex 5		5,412	5,130	–	217.2	34	156.5
PP_CORDIC_8	Altera Stratix III		8,940	3,048	96 (18x18 bit)	123.8	15	121.2
PP_CORDIC_5	Altera Stratix III	65	6,173	2,219	64 (18x18 bit)	123.8	12	97.0
PP_CORDIC_1	Altera Stratix III		2,637	1,115	12 (18x18 bit)	123.8	8	64.6
[13]b	Altera Stratix II		6,469	5,372	–	195.1	34	174.3
PP_CORDIC_8	Altera Stratix II		9,055	2,639	192 (9x9 bit)	85.9	15	174.6
PP_CORDIC_5	Altera Stratix II	90	6,394	1,961	120 (9x9 bit)	85.9	12	139.7
PP_CORDIC_1	Altera Stratix II		2,838	1,057	24 (9x9 bit)	85.9	8	93.1
[19]	Altera Stratix IV		5,612	4,231	32 (18x18 bit)	258.3	36	139.4
PP_CORDIC_8	Altera Stratix IV		9,081	3,033	96 (18x18 bit)	122.6	15	122.4
PP_CORDIC_5	Altera Stratix IV	40	6,401	2,206	64 (18x18 bit)	122.6	12	98.1
PP_CORDIC_1	Altera Stratix IV		2,833	1,106	12 (18x18 bit)	122.6	8	65.4

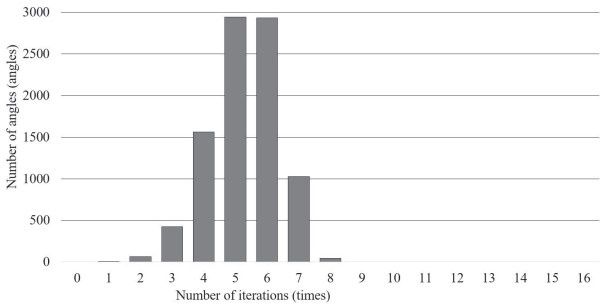


Fig. 11 The number of iterations of angle recording algorithm in case N = 16.

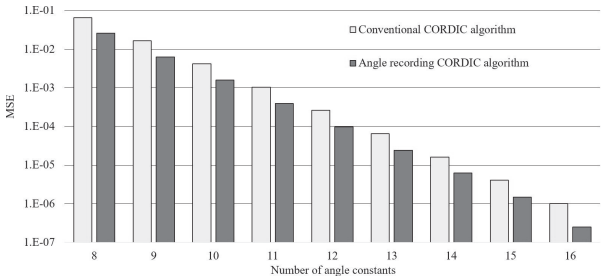


Fig. 12 The comparison in mean square error of approximate angles.

Moreover, because the results of PP-CORDIC architecture are sine and cosine of an input angle, this design is also compared with Altera floating point sine and cosine core [19].

4.1 Evaluation of Algorithm

The angle recording CORDIC was simulated in MATLAB software with 9,001 input angles from $[-45°, 45°]$ in order to calculate the number of iterations and MSE of estimated angle. In case the number of angle constants is $N = 8$, 12, and 16, the angle recording CORDIC requires 2.43, 3.77, and 5.28 micro-rotations in the average case, respectively. On the other hand, the number of iterations of conventional CORDIC is always the same as the number of predefined angles. Therefore, in the average case, the angle recording CORDIC is 3.3X, 3.2X, and 3.0X faster than conventional CORDIC in case the number of constants is 8, 12, and 16, respectively. The density distribution of the number of iterations in case of 16 angle constants is analyzed in Fig. 11. In this figure, the number of iteration spread from zero to eight, and the minimum, average and maximum iteration is 0, 5.28 and 8, respectively.

In order to compare the accuracy of angle recording CORDIC algorithm with that of the conventional CORDIC algorithm, the mean square error (MSE) of the approximate angle was calculated by (8)

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{\Phi} - \Phi)^2, \tag{8}$$

where n is the number of input angles, $\hat{\Phi}$ is the approximate angle, and Φ is the initial input angle. In fact, the more MSE value is close to zero, the more the precision of sine/cosine can be achieved. Figure 12 means that the precision of both methods increases when the number of predefined angles N varies from 8 to 16. Nevertheless, the proposed CORDIC always delivers smaller MSE value than that from conventional one.

4.2 Evaluation of Hardware Architecture

The PP-CORDIC hardware is employed in Altera Quartus 13.0 with Statrix IV FPGA target. The resource utilization, frequency and latency of proposed design is compared with other floating point CORDIC design. Moreover, because the results of PP-CORDIC architecture are sine and cosine of an input angle, this design is also compared with Altera floating point sine and cosine core.

The device family, the resource utilization (Lookup tables (LUTs), Registers, DSP), the frequency and the latency calculated in clock cycles and calculated in time (ns) of PP-CORDIC and other floating-point designs are shown in Table 3. In order to make fair comparison, the PP-CORDIC architecture was implemented on different Altera

family devices which is the same process, similar performance and resource with previous designs [20]–[24]. Suppose that *PP_CORDIC_m* indicates the PP-CORDIC architecture with m ANSRs and COCRs operate in parallel. The value of m depends on the number of iterations of angle recording CORDIC, hence its range is $\{1, 2, \ldots, 8\}$. Obviously, the resource utilization and also the latency of PP-CORDIC will be increased if the value of m is gone up. For simple table, only the results of PP_CORDIC_1, PP_CORDIC_5, and PP_CORDIC_8 are shown in Table 3.

The former latency column of Table 3 describes that the architecture in [12] spends the highest clock cycles, 130 clock cycles, to calculate the results, whereas both configurations of the design in [13] and the design in [19] spends only 34 and 36 clock cycles. Moreover, the *PP_CORDIC_m* occupies from 8 to 15 clock cycles if the value of m changes from 1 to 8, respectively. According to the results in Table 3, the proposed architecture costs the lowest latency which calculated by clock cycles in comparison with the others, i.e. its latency in the worst case still 4.5X, 2.3X, and 2.4X lower than that of the design in [12], [13], and [19], respectively.

However, the frequency column of Table 3 indicates that the PP-CORDIC architecture operates at lower frequency than that of the other designs. Subsequently, the latency calculated by time (ns), which depends on (9)

$$\text{Latency (ns)} = \frac{\text{Latency (clock cycles)}}{\text{Frequency (MHz)}} \times 1000, \qquad (9)$$

is predicted low. Nevertheless, because the number of iterations in our proposed design are low, the latency calculated by time of all configurations in the last column of the Table shows that PP-CORDIC still spends the lowest latency in comparison with the others. In the worst case, it is 4.8X, 1.3X, and 1.1X lower than that of the design in [12], [13]a, and [19], respectively, and nearly the same with that of the design in [13] implemented on Altera Stratix II.

As mentioned above, the resource utilizations of PP-CORDIC depend on the value of m. The fourth to sixth column in Table 3 show resource utilizations of all designs. The LUTs of the configurations from one to five, $m \in \{0, \ldots, 5\}$, are lower or similar to that of the previous design. Although these architectures require more the number of DSPs than that of previous one, they cost less registers than that of the others. With the designs from six to eight, they require quite more resource utilizations than that of the others. It means the PP-CORDIC architecture is the best choice for fixed and known input angles which their number of rotations is smaller than five. In case the application is not too strict on resource utilization, the configurations from six to eight are also well-chosen because of their low-latency architectures.

4.3 Evaluation of ASIC Implementation

The PP-CORDIC scheme with $m = 2$ (PP_CORDIC_2) was implemented successfully on 180 nm CMOS technology.

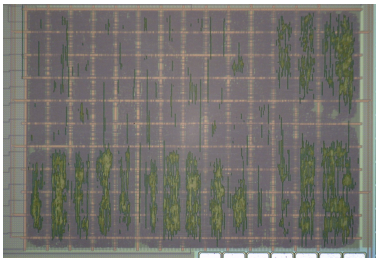


Fig. 13 The micrograph of PP_CORDIC_2 chip.

Table 4 Specification parameters of PP_CORDIC_2 chip.

Parameters	Values
Technology	CMOS 0.18 μm
Voltage	1.8 V
Size	0.85 mm × 1.2 mm
Logic cells	31,871 cells
Max. Frequency	57.14 MHz
Power	8.12 mW

Table 5 The frequency and power consumption of PP_CORDIC_2 in comparison with previous works in 180 nm CMOS technology.

Design	Frequency (MHz)	Power (mW)	Energy (nJ/cycle)
Conventional [14]	16	165	10.31
CORDIC_CLA [14]	24.5	86.7	3.54
[15]	6.47	1.23	0.19
[16]	127	350	1.97
PP_CORDIC_2	57.14	8.12	0.14

Figure 13 is the micrograph of PP_CORDIC_2 chip. Some specific parameters of PP_CORDIC_2 are shown in Table 4.

The comparison of operating frequency and power consumption of our design and previous works are shown in Table 5. All of the CORDIC designs were implemented on 180 nm CMOS technology. Because of the difference in operating frequency, it is difficult to make the comparison between the dynamic powers of these designs. Therefore, the active energy of each design calculated by Eq. (10) and showed in the final column.

$$\text{Energy (nJ/cycle)} = \frac{\text{Power (mW)}}{\text{Frequency (MHz)}}. \qquad (10)$$

The results show that, our PP_CORDIC_2 spends lowest energy in comparison with the others. Its energy is 73.6X, 25.3X, 1.4X, 14.1X times lower than that of the conventional and the proposed CORDIC in [14], [15], and [16], respectively.

5. Conclusion

In this paper, a low latency pipeline parallel CORDIC (PP-CORDIC) was proposed and verified. The parallel architecture, pipeline processing together with angle recording method are utilized in this design to achieve the low-latency and floating-point precision. Because the number of CORDIC-function modules is equal to the maximum iterations of input angle, this proposed architecture can be

applied for fixed and known angles of rotation. It can operate at 122.6 MHz frequency and achieve 8, 12, and 15 clock cycles in the best, average, and worst case, respectively. The PP-CORDIC implemntation on FPGA contains a low-latency architecture while the implementation on 180 nm CMOS technology costs low active energy.

Acknowledgments

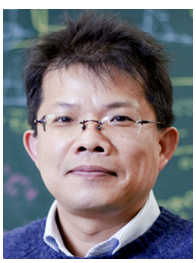
This work is supported by VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc. and Cadence Design Systems, Inc.

References

[1] J.E. Volder, "The CORDIC trigonometric computing technique," IRE Trans. Electronic Computers, vol.EC-8, pp.330–334, 1959.
[2] J.S. Walther, "A unified algorithm for elementary functions," Proc. Spring Joint Computer Conf., pp.379–385, 1971.
[3] P.K. Meher, J. Valls, T.-B. Juang, K. Sridharan, K. Maharatna, "50 Years of CORDIC: Algorithms, architectures, and applications," IEEE Trans. Circuits Syst. I, Reg. Papers, vol.56, no.9, pp.1893–1907, 2009.
[4] P.K. Meher and S.Y. Park, "CORDIC designs for fixed angle of rotation," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.21, pp.217–228, 2013.
[5] Y.H. Hu and S. Naganathan, "An angle recoding method for CORDIC algorithm implementation," IEEE Trans. Comput., vol.42, no.1, pp.99–102, 1993.
[6] Y.H. Hu and Z. Wu, "An efficient CORDIC array structure for the implementation of discrete cosine transform," IEEE Trans. Signal Process., vol.43, no.1, pp.331–336, Jan. 1995.
[7] T.K. Rodrigues and E.E. Swartzlander Jr., "Adaptive CORDIC: Using parallel angle recoding to accelerate rotations," IEEE Trans. Comput., vol.59, no.4, pp.522–531, 2010.
[8] T.-B. Juang, S.-F. Hsiao, and M.-Y. Tsai, "Para-CORDIC: Parallel CORDIC rotation algorithm," IEEE Trans. Circuits Syst., vol.51, no.8, pp.1515–1524, Aug. 2004.
[9] A.M. Despain, "Fourier transform computers using CORDIC iterations," IEEE Trans. Comput., vol.C-23, no.10, pp.993–1001, 1974.
[10] E.H. Wold and A.M. Despain, "Pipeline and parallel-pipeline FFT processors for VLSI implementations," IEEE Trans. Comput., vol.C-33, no.5, pp.414–426, 1984.
[11] P. Surapong and M. Glesner, "Pipelined floating-point architecture for a phase and magnitude detector based on CORDIC," Int. Conf. Field Programmable Logic and Applications (FPL), pp.382–384, Chania, 2011.
[12] N. Dhume and R. Srinivasakannan, Parameterizable CORDIC-Based Floating-Point Library Operations, 2012.
[13] J. Zhou, Y. Dou, Y. Lei, and Y. Dong, "Hybrid-mode floating-point FPGA CORDIC co-processor," Reconfigurable Computing: Architectures, Tools and Applications, Lecture Notes in Computer Science vol.4943, pp.256–261, Springer Berlin Heidelberg, 2008.
[14] R.K. Jain, V.K. Sharma, and K.K. Mahapatra, "A new approach for high performance and efficient design of CORDIC processor," Proc. 1st International Conference on Recent Advances in Information Technology (RAIT), pp.756–760, Dhanbad, March 2012.
[15] S. Kumar, M.A. Basiri, and N. Mahammad, "High precision and high speed handheld scientific calculator design using hardware based CORDIC algorithm," Proc. International Conference on Design and Manufacturing (IConDM2013), vol.64, pp.56–64, 2013.
[16] K.C. Ray and A.S. Dhar, "CORDIC-based unified VLSI architecture for implementing window functions for real time spectral analysis," IEE Proc. Circuits, Devices and Systems, vol.153, pp.539–544, 2006.
[17] H.-T. Nguyen, X.-T. Nguyen, C.-K. Pham, T.-T. Hoang, and D.-H. Le, "A parallel pipeline CORDIC based on adaptive angle selection," Proc. 15th International Conference on Electronics, Information, and Communication (ICEIC), pp.411–414, Vietnam, Jan. 2016.
[18] T. Lang and E. Antelo, "High-throughput CORDIC-based geometry operations for 3D computer graphics," IEEE Trans. Comput., vol.54, no.3, pp.347–361, March 2005.
[19] Altera, Floating-Point Megafunctions User Guide, 2014.
[20] Xilinx, http://www.xilinx.com/support/documentation/selection-guides/virtex7-product-table.pdf
[21] Xilinx, Virtex-7 T and XT FPGAs Data Sheet: DC and AC Switching Characteristics, 2016.
[22] Altera, Stratix V Device Overview, 2015.
[23] Altera, Stratix III FPGAs vs. Xilinx Virtex-5 Devices: Architecture and Performance Comparison, 2007.
[24] Xilinx, http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex5/v5product_table.pdf

Hong-Thu Nguyen received the B.S. and M.S degrees from the Vietnam National University of Ho Chi Minh City-University of Science, Vietnam, in Electronics and Telecommunications in 2011 and 2014, respectively. After finished the M.S. degree, in 2014, she came to Japan and became a student at The University of Electro-Communications, Tokyo, Japan. Her research interests include improving communication technique (MIMO, OFDM, ...) and designing digital systems using integrated circuits. She is a student member of IEEE.

Xuan-Thuan Nguyen received the B.S. and M.S. degree from the University of Science - Vietnam National University of Ho Chi Minh City in 2010 and 2013, respectively. Since October 2013, he is pursuing the Ph.D. degree at the University of Electro-Communications, Tokyo, Japan. His current research interests include high performance computing designs on FPGA and VLSI for big data analytics. Mr. Nguyen is a member of the IEEE.



Cong-Kha Pham received the B.S., M.S. and Ph.D. degree all from the Sophia University, Tokyo, Japan, in Electronics Engineering. From April 1992 to March 1996, he was with the Deparment of Electrical and Electronics Engineering, Sophia University, Tokyo, Japan, where he was a researach assistant. From April 1996 to March 2000, he was with the Department of Information Systems at the Tokyo University of Information Science, Chiba, Japan, where he was an Assistant Professor. Since April 2000, he has been an Associate Professor of the Department of Electronic Engineering at the University of Electro-Communications, Tokyo, Japan. His research interest include design of analog and digital systems using intergrated circuits. Dr. Pham is a member of IEEE.