Low-resource low-latency hybrid adaptive CORDIC with floating-point precision

Hong-Thu Nguyen^{1a)}, Xuan-Thuan Nguyen¹, Trong-Thuc Hoang²,
Duc-Hung Le², and Cong-Kha Pham¹

¹ The University of Electro - Communications,

1–5–1 Chofugaoka, Chofu, Tokyo 182–8585, Japan

² The University of Science, Ho Chi Minh City,

227 Nguyen Van Cu St., Dist. 5, Vietnam

a) hongthu@vlsilab.ee.uec.ac.jp

Abstract: Despite being proposed since more than 50 years ago, COordinate Rotation DIgital Computer (CORDIC) is still one of the most effective algorithms for elementary function calculation so far. Original CORDIC, however, suffers high latency due to its nature of unvarying number of rotations. As a result, a low-latency hybrid adaptive (HA) CORDIC is proposed in this paper. Firstly, adaptive angle selection decreases total iterations up to 50% with respect to higher accuracy of results. Secondly, hybrid architecture including fixed-point input and floating-point output reduces the total hardware utilization and enhances the dynamic range of final results. Lastly, parallel and pipeline processing together with resource sharing technique allow the design to operate fully at 175.7 MHz with low resource consumption—1,139 LUTs and 489 registers.

Keywords: CORDIC, low-latency, low-resource, hybrid, floating point

Classification: Electron devices, circuits, and systems

References

- [1] J. E. Volder: IRE Trans. Electron. Comput. **EC-8** (1959) 330. DOI:10.1109/TEC.1959.5222693
- [2] J. S. Walther: Proc. Spring Joint Computer Conf. (1971) 379.
- [3] P. K. Meher, J. Valls, T.-B. Juang, K. Sridharan and K. Maharatna: IEEE Trans. Circuits Syst. I, Reg. Papers **56** (2009) 1893. DOI:10.1109/TCSI.2009.2025803
- [4] A. M. Despain: IEEE Trans. Comput. **C-23** (1974) 993. DOI:10.1109/T-C.1974.223800
- [5] E. H. Wold and A. M. Despain: IEEE Trans. Comput. **C-33** (1984) 414. DOI:10.1109/TC.1984.1676458
- [6] R. Touzi, R. K. Hawkins and S. Cote: IEEE Trans. Geosci. Remote Sens. **51** (2013) 487. DOI:10.1109/TGRS.2012.2201946
- [7] D. M. Munoz, D. F. Sanchez, C. H. Llanos and M. Ayala-Rincón: SPL VI Southern Programmable Logic Conf. (2010) 55. DOI:10.1109/SPL.2010.5483002
- [8] P. Surapong and M. Glesner: FPL Int. Conf. Field Programmable Logic and Applications (2011) 382. DOI:10.1109/FPL.2011.74

- [9] N. Dhume and R. Srinivasakannan: Xilinx Application Note XAPP552 (2012) 1.
- [10] J. Zhou, Y. Dong, Y. Dou and Y. Lei: ICESS Int. Conf. Embedded Software and Systems (2008) 616. DOI:10.1109/ICISS.2008.95
- [11] Y. H. Hu and S. Naganathan: IEEE Trans. Comput. **42** (1993) 99. DOI:10.1109/12.192217
- [12] T. K. Rodrigues and E. E. Swartzlander: IEEE Trans. Comput. **59** (2010) 522. DOI:10.1109/TC.2009.190
- [13] P. K. Meher and S. Y. Park: IEEE Trans. Very Large Scale Integr. (VLSI) Syst. **21** (2013) 217. DOI:10.1109/TVLSI.2012.2187080
- [14] R. Shukla and K. C. Ray: IEEE Trans. Comput. **63** (2014) 3066. DOI:10.1109/TC.2013.173
- [15] S. Aggarwal, P. K. Meher and K. Khare: IEEE Trans. Circuits Syst. I, Reg. Papers **60** (2013) 314. DOI:10.1109/TCSI.2012.2215778
- [16] Y. Lei, Y. Dou, S. Guo and J. Zhou: LNCS Adv. Parallel Process. Technol. **6965** (2011) 127. DOI:10.1007/978-3-642-24151-2_10
- [17] Altera: Floating-Point Megafunctions User Guide (2013) 1.

1 Introduction

CORDIC, a simple and efficient iterative algorithm to compute elementary functions, was first introduced by Volder [1] in 1959 and later extended by Walther [2] in 1971. In fact, CORDIC only requires the additions, subtractions, and shift operations, thereby fitting with Very-Large-Scale Integration (VLSI) system design. For this reason, a vast amount of research in CORDIC algorithm and hardware solution still are in progress although CORDIC is more than 50 years old [3].

CORDIC-based Fast Fourier Transform (FFT) [4, 5] plays an essential role in most multimedia and wireless communication applications, where the evaluation of trigonometric functions are imperative. Fixed-point number representation is widely utilized in those systems due to its simple calculation and sufficient precision. However, some advanced applications such as Synthetic Aperture Radar (SAR) data processing [6], require not only highly precise results but also suitable format to manage wide dynamic range of numbers. In those systems, floating-point representation, instead of fixed-point, are deployed to retain real numbers resolution and accuracy effectively. Many approaches to highly efficient floating-point CORDIC, therefore, have been proposed recently. D. M. Munoz et al. [7] described a two-operation-mode floating-point CORDIC architecture that can compute the sine, cosine, or arctangent function. The design achieved an operating frequency of 86.1 MHz with elapsed time around 90 clock cycles at single-precision configuration. P. Surapong et al. [8] proposed an 8- and 16-stage pipelined floating-point CORDIC for phase and magnitude detector. As much as, 23% slice register and 38% slice lookup table (LUT) of Xilinx Virtex-5 are used for this 16-stage system, whereas the maximum frequency is only 133.8 MHz. Both Nikhil Dhume et al. [9] and Jie Zhou et al. [10] presented a hybrid approach that firstly convert floating-point input into fixed-point format. A fixed-point CORDIC, then, computes the trigonometric functions and those results, lastly, are transformed into IEEE 754 floating-point format.

In order to reduce error in the results of CORDIC system, more iterations must be performed. However, the number of iterations and clock cycles of each iteration significantly affect the latency of CORDIC algorithm. Many approaches to enhance the precision without sacrificing the latency, therefore, have been increasingly attractive. In 1993, Y. H. Hu et al. [11] proposed a method called angle recording which could reduce 50% the number of iterations. K. R. Terence et al. [12] proposed parallel angle recording method that converged to the final result in the least number of iterations. This method chooses all angle constants in one step but requires a large number of comparison logics. P. K. Meher et al. [13] used angle recording scheme for rotating a fixed angle that could be used in specific application areas such as robotics, graphics, games, and animation. R. Shukla et al. [14] proposed a new low-latency CORDIC algorithm that combining two existing algorithms. This new one can reduce the iterations to $(3n/8 + 1)$ with n is the number of bit precision. Likewise, another reducing iteration algorithm was proposed by S. Aggarwal et al. [15], which could be applied to waveform generator.

In this paper, a low-latency hybrid adaptive CORDIC with floating-point precision is proposed. Moreover, this proposed scheme can obtain low-resource consumption. The contributions of this research are described in detail as follows.

- Low-resource: Hybrid architecture includes 24-bit fixed-point input angle in degree and 32-bit IEEE 754 floating-point sine/cosine outputs. This architecture aims to balance the calculation accuracy with resource utilization. Furthermore, in floating-point arithmetic, resource sharing technique is implemented to reduce the logic utilization by around 70% in comparison with Altera library of the same function.
- Low-latency: Parallel processing is applied to fixed- and floating-point components, and pipeline processing is deployed in each of them to improve the throughput as well as latency. Moreover, adaptive technique, which reduces the number of iterations by obtaining the final results in the least number of constant angles, is an important part to reduce the latency.
- High-precision: Adaptive technique and hybrid architecture improve not only latency and resource consumption, respectively, but also the precision of the proposed system.

2 Proposed CORDIC algorithm

2.1 Overview of conventional CORDIC algorithm

CORDIC algorithm consists of two operation modes namely vectoring and rotation mode. In this paper, rotation mode is focused on to calculate sine and cosine of an input angle Φ . Initially, the initial vector of CORDIC algorithm is $V_0(x_0, y_0)$. After each micro-rotation i , this vector is adjusted by an angle constant θ_i as long as residual angle z_i approaches to zero. The equation to calculate x_{i+1} , y_{i+1} which do not contain gain factor k_i in each iteration of rotation mode is (1).

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

Gain factor, K is used to eliminate k_i multiplication at each iteration i , where N is the total predetermined angles.

$$K = \prod_{i=0}^{N-1} k_i = \prod_{i=0}^{N-1} \cos \theta_i \quad (2)$$

Trigonometric results, finally, are derived from x_{N-1} , y_{N-1} and K through (3), and sum of θ_i almost reaches input angle Φ .

$$\begin{aligned}\sin(\Phi) &= x_{N-1} * K \\ \cos(\Phi) &= y_{N-1} * K\end{aligned}\quad (3)$$

$$\Phi = \sum_{i=0}^{N-1} d_i \theta_i \quad (4)$$

2.2 Proposed hybrid adaptive (HA) CORDIC algorithm

The HA-CORDIC algorithm is proposed to reduce the number of iterations and thereby reduces the latency. In other words, only several angles in the set of N angle constants are utilized to form the closest result. This proposed optimized 4-step algorithm developed for sine/cosine calculation is shown in Fig. 1. To begin with, the input angle is converted into predefined range. Secondly, an optimum set of micro-rotations whose sum approximates the input angle are selected. The coordinate (x, y) , and K factor, then, are correspondingly updated. Finally, sine and cosine values are achieved by the product of latest x_t , y_t , and K , together with simple adjustments.

2.2.1 Angle normalization

The CORDIC algorithm will only converge across a limited range of input values. In rotation mode, convergence is guaranteed for the angles below the sum of entire N angle constants, i.e. between -99.88° and 99.88° . However, all of the trigonometric values of any angle in circular can be interpolated from trigonometric value of an angle in range $[0^\circ, 45^\circ]$. Therefore, a trigonometric identity is used to translate any outside-range angle into that within this range. By employing several simple adjustments, outside-range angles can be shortened the range $[0^\circ, 45^\circ]$.

2.2.2 Angle selection

Assume that 16 angle constants, θ , are employed in the proposed CORDIC as shown in Table I. Conventional CORDIC exploits all θ_i angles and their direction d_i for trigonometric computations, as determined in (1). For example, 30° input angle costs 16 iterations.

Fig. 1. The flow chart of HA-CORDIC algorithm.

$$(\theta_0 - \theta_1 + \theta_2 - \theta_3 + \theta_4 + \theta_5 - \theta_6 + \theta_7 - \theta_8 - \theta_9 + \theta_{10} + \theta_{11} - \theta_{12} + \theta_{13} - \theta_{14} - \theta_{15}) = 30.000834^\circ$$

In case the input angle is 30° , the residual angle of conventional CORDIC is $8.34e - 4$. The proposed method, however, only requires five iterations to achieve better residual angle, $2.45e - 4$.

$$(\theta_1 + \theta_4 - \theta_9 - \theta_{11} - \theta_{15}) = 29.999755^\circ$$

The key point of this method is that in each iteration i , angle constant θ_i is selected so that residual angle z_i converges to zero. The iteration stops upon z_i is smaller than a predefined *threshold*.

In order to choose the angle constant in each iteration, this method uses a set of parameters, C , which expresses the range of residual angles around one angle constant, as defined in (5). The details of C is described in Table I. On this basis, the pseudocode determines which angle constant is chosen summarized in Fig. 2.

$$c_i = \begin{cases} \frac{\theta_i + \theta_{i+1}}{2} & \text{if } 0 \leq i \leq (N - 2) \\ \frac{\theta_i}{2} & \text{otherwise} \end{cases} \quad (5)$$

Table I. The values of θ , C , and K .

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

```

1:  $i = j = 0$ 
2:  $threshold = c(15)$ 
3: while  $z(j) > threshold$  and  $j < (N - 1)$  do
4:   if  $z(i) > c(j)$  then
5:     select rotation angle  $\theta(j)$ 
6:     update residual  $z(i)$ 
7:      $i = i + 1$ ;
8:   end if
9:    $j = j + 1$ ;
10: end while

```

Fig. 2. The pseudocode of angle selection function.

2.2.3 Pre-calculation: coordinate, residual, and factor adaption

The adaption of coordinate (x, y) , residual z , and factor K are described in (6). Because only several angles are selected, factor 2^{-i} , θ_i , and k_i are replaced by 2^{-j} , θ_j , and k_j respectively. Both j and θ_j are obtained in Fig. 2, where k_j is listed in Table I.

$$\begin{aligned}
 x_{i+1} &= x_i - d_i y_i 2^{-j} \\
 y_{i+1} &= y_i + d_i x_i 2^{-j} \\
 z_{i+1} &= z_i - d_i \theta_j \\
 K &= K * k_j
 \end{aligned} \tag{6}$$

If the residual is smaller than the defined threshold, the latest coordinate (x_t, y_t) , and gain factor K can be achieved. Finally, X , Y value can be obtained by (7).

$$\begin{aligned} X &= x_t * K \\ Y &= y_t * K \end{aligned} \quad (7)$$

2.2.4 Post-calculation: sine/cosine recovery

Before coordinate (X, Y) is calculated, the input angle is normalized, thereby the final sine and cosine results of the input angle must be recovered from (X, Y) values.

3 Proposed hardware architecture

3.1 Overview

The proposed design is composed of four main modules namely ANGLE_SELECTION (ASEL), FIFO, PRE_CALCULATION (PREC), and POST_CALCULATION (POSC), which are illustrated in Fig. 3a. The input is 24-bit fixed-point (FIX) angle that format is 1.8.15, i.e. 1-bit sign, 8-bit magnitude, and 15-LSBs, and the outputs are two 32-bit floating-point (FLP) trigonometric results.

The processing of HA-CORDIC is separated into two parallel threads, which can be seen from Fig. 3b. Because of the difference in operating cycles, a FIFO is inserted between ASEL and PREC to ease the latency. In fact, ASEL and PREC/POSC cost one and two clock cycles for FIX and FLP operation, respectively. Besides, pipeline processing is applied to all modules to increase the throughput. Depending on the number of iterations is determined in ASEL, the execution latency of each angle is markedly different. Each module is described in more detail below.

Fig. 3. (a) The general block diagram of proposed CORDIC. (b) The mechanism of proposed CORDIC.

3.2 Angle selection (ASEL)

Module ASEL strives to obtain the precise result with the least number of iterations by reducing divergent pseudo-rotations. This module is composed of three main components, ANGLE_NORMALIZER (ANOR), SET_NEXT_ROTATION (SNR), and CHECK_LAST_ROTATION (CLR), which are depicted in Fig. 4. Beforehand, ANOR converts input angle $iData$ into normalization range of $[0^\circ, 45^\circ]$. It can be seen in Fig. 5a, if a circle is split into eight pieces, any angle from the first to the seventh piece can be transformed into equivalent angle in zeroth piece. The recovery information (Rec. info) is transferred to the post-calculation to adjust the final result. By using the Correction equation, the final sine/cosine results can be achieved.

Fig. 4. The hardware architecture of phase search circuit.

Fig. 5. (a) A description of normalization technique. (b) The hardware architecture of SET NEXT ROTATION.

After receiving the normalized angle, SNR determines the next angle in ROM_THETA by utilizing a pair of ROM_C and priority encoder, as illustrated in Fig. 5b. Module ROM_THETA includes 16 angle constants θ while ROM_C stores the range of residual angles C around one angle constant, which is defined in (5). In order to eliminate the *while* loop in pseudocode from Fig. 2, a set of comparators is deployed in parallel together with a priority encoder to search for next suitable angle θ at the speed of one cycle. The iteration completes as soon as residual angle register Z is smaller than *threshold*.

At each iteration i , CLR checks whether the current process is the last or not. If the current process is the final iteration, CLR signals ASEL to stop calculating and start the new input angle rotation in following cycle. The pseudocode of this circuit is described in Fig. 6. If input angle Φ is approximately zero, no rotation is executed (line 5 and 6). If z is within the determined range, updated z will become smaller than the threshold in next rotation (line 7 to 10). Because the addition and subtraction affect the circuit frequency, two LUTs ROM_s ($\theta_i - threshold$) and ROM_a ($\theta_i + threshold$) are implemented instead. In this hardware system, module SNR and CLR are implemented in parallel.

A collection of 4-bit *norm.info*, 2-bit *last.rotation*, 1-bit *sign*, and 4-bit *phase.addr*, lastly, is brought together and put into FIFO. At the same time, CONTROL_LOGIC gets FIFO to accept data by asserting *FF_wrrreq*. If FIFO is not available or current angle is still in progress, *oReady* will go to low level and thereby CORDIC cannot accept new input angle.


```

1:  $i = 1$ 
2:  $threshold = ROM\_C(15)$ 
3:  $last\_rotation = \{0, 0\}$ 
4: while  $i < 16$  do
5:   if  $|z| \leq threshold$  then
6:      $last\_rotation[0] = 1$ 
7:   else if  $|z| \in [\theta(0) - threshold, \theta(0)]$  then
8:      $last\_rotation[1] = 1$ 
9:   else if  $|z| \in [\theta(i) - threshold, \theta(i) + threshold]$  then
10:     $last\_rotation[1] = 1$ 
11:   end if
12: end while

```

Fig. 6. The pseudocode of angle selection function.

3.3 Pre-calculation (PREC)

PREC contains four main components: Floating-point Adder Subtractor (FADD_SUB), Floating-point Multiplier k_i (FMUL_ki), Floating-point Multiplier XYK (FMUL_XYK), and CONTROL_LOGIC, as shown in Fig. 7a.

FADD_SUB calculates X and Y due to i and $signZ$ in each iteration, as illustrated in Fig. 7b. The initial values of X and Y are set as *one* and *zero*, respectively, immediately after FADD_SUB is reset. FADD_SUB is active by asserting *start* within two clock cycles while holding both $signZ$ and i . Simultaneously, *phase* becomes the control signal for multiplexing the data path during the operation. The 4-stage *shifter* performs a right shift operation with zeros fetched into empty MSB because of the fraction parts of x and y . The sign decision checks $signZ$, *phase*, and sign of previous X and Y data to decide the operation, addition or subtraction, in Carry Look Ahead (CLA) adder. The two's complement (2's complement), then, will correct the result in case it is a negative number. Finally, all of the information will produce the sign of the result to complete the process of the module.

FMUL_ki produces the gain-factor K by multiplying each step-factor k_i in each iteration i , as shown in Fig. 7c. Initial value of register *RegK* in module FMUL_ki is set as *one* and is sequentially updated by previous K and ROM_K that is depicted in Table I. FMUL_ki also requires 2-clock-delay *start* for its pipeline computation.

FMUL_XYK calculates the products of latest X/Y and K whose circuit is illustrated in Fig. 7d. The *last* signal is set within two clock cycles while remaining the X and Y to enable the FMUL_XYK. The raw cosine and sine results are ready in third and fourth clocks, respectively. The results of sine, cosine pre-calculation are stored in registers *regX*, *regY*. This proposed hardware uses parallel and pipeline processing and resource sharing techniques. It can be seen that by using proposed scheme latency, throughput, and hardware utilization are improved significantly.

3.4 Post-calculation (POSC)

The raw sine and cosine values are combined with *rec_info* to form the final trigonometric results. The recovery information *rec_info* given in Fig. 5a is utilized to select the suitable adjusted *pre_sin* or *pre_cos*.

Fig. 7. The hardware architecture of (a) PREC circuit. (b) FADD.SUB circuit. (c) FMUL.ki circuit. (d) FMUL.XYK circuit.

4 Experimental results

The performance of the proposed CORDIC is evaluated in two aspects: algorithm and hardware design. The first assessment proves that proposed algorithm requires fewer iterations but achieves higher precision than the original CORDIC. The second assessment compares this work with the other floating-point systems in terms of latency and resource utilization.

4.1 Evaluation of algorithm

In order to assess the algorithm, 9001 angles, from $[-45^\circ, 45^\circ]$ are generated. The total number of iterations are observed at three different number of angle constants—namely N . As can be seen in Fig. 8, the proposed HA-CORDIC only requires maximum 4, 6, and 8 micro-rotations in case $N = 8, 12, 16$, respectively. In other words, in the worst case, the HA-CORDIC is still $2.7X$, $2.4X$, and $2.3X$ faster than the original one. Moreover, on average, the latency improvements are $3.8X$, $3.5X$, and $3.4X$ at $N = 8, 12, 16$, respectively.

The precision of two algorithms is measured by mean square error (MSE) of the residual angle. In fact, the more the residual angle is close to zero, the more the precision of sine/cosine can be achieve. It can be seen in Fig. 9, with the variation of N from 8 to 16, the precision of both methods increases. Nevertheless, the proposed CORDIC always delivers smaller MSE value, $2.51e-7$, than that from conventional one, $1.02e-6$.

4.2 Evaluation of hardware design

The proposed HA-CORDIC is synthesized by Altera Quartus 14.0 with Stratix IV FPGA target. The latency, resource utilization (LUT, Register, Memory) and

Fig. 8. The comparison in number of iterations in case of 8, 12, and 16 predetermined angle constants.

Fig. 9. The comparison in mean square error of residual angles.

Table II. The comparison between HA-CORDIC with other floating-point systems in terms of device family (A), latency (clocks) (B), frequency (MHz) (C), lookup table (D), register (E), memory (F), and DSP (G).

	[7]	[8]	[9]	[10]	[16]	[17]	This work
(A)	Xilinx Virtex 5	Xilinx Virtex 5	Xilinx Virtex 7	Altera Stratix II	Xilinx Virtex 6	Altera Stratix IV	Altera Stratix IV
(B)	93	–	130	–	–	36	12/20/26
(C)	86.1	133.8	280.0	195.1	253.5	258.3	175.7
(D)	3,152	26,811	6,514	6,469	13,744	5,612	1,139
(E)	–	16,274	4,725	5,372	–	4,231	498
(F)	2,832	–	4,894	–	–	3,575	11
(G)	0	2	9	–	96	32	8

operating frequency among HA-CORDIC and the other floating-point CORDIC systems are illustrated in Table II. Unlike original CORDIC methods, HA-CORDIC latency is varied due to the dynamic rotation but the latency results are

much shorter than the other results. At $N = 16$, it costs 12, 20, and 26 clock cycles in best case (zero or one rotation), typical case (five rotations), and worst case (eight rotations). Besides, HA-CORDIC resource utilization is much better than the others regarding the operating frequency of 175.7 MHz.

5 Conclusion

In this paper, a low-latency hybrid adaptive CORDIC with floating-point precision is proposed. Because of adaptive angle selection, proposed HA-CORDIC needs fewer latency while achieves more precise trigonometric results rather than the original one. Furthermore, hybrid architecture, which includes a fixed-point input angle in degree and two floating-point sine/cosine outputs not only reduces the hardware resource in total but also enhances the results precision in general. The experiments show that the design is fully operational at 175.7 MHz and costs 12 and 26 latency cycles in best and worst case, respectively, in case of $N = 16$. Besides, HA-CORDIC is likely to integrate into other advanced systems easily due to given acknowledge signals and low resource consumption — 1,139 LUTs and 489 registers only.