4A-26

An Efficient Fixed-point Arithmetic Processor Using A Hybrid CORDIC Algorithm

Hong-Thu Nguyen, Xuan-Thuan Nguyen, and Cong-Kha Pham
The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo, Japan
{hongthu,xuanthuan}@vlsilab.ee.uec.ac.jp
phamck@uec.ac.jp

Related work

i	θ
0	45.000000000
1	26.565051177
2	14.036243468
3	7.125016349
4	3.576334375
5	1.789910608
6	0.895173710
7	0.447614171
8	0.223810500
9	0.111905677
10	0.055952892
11	0.027976453
12	0.013988227
13	0.006994114
14	0.003497057
15	0.001748528

Fig. 1: Set of angle constants

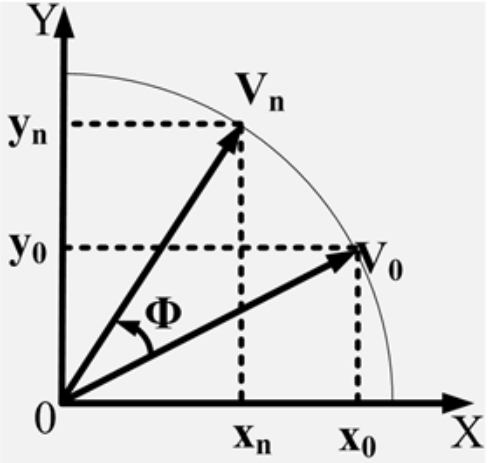

Fig. 2: A 2 dimension plane

- *CORIDC:*
  - Calculate several common mathematical functions.
  - The input value will be rotated by the set of predefined angle constants to return the final values.
- *Conventional CORIDC (Conv. CORDIC):*
  - Utilize all angles in set of predefined angle constant.
  - Require one scale factor value for any input angle.
- *Angle Recording CORIDC (ARD CORDIC):*
  - Utilize only several angle of constant.
  - Each input angle require one scale factor.
- *Scaling free CORIDC (SCFE-CORDIC):*
  - Utilize only small angle constant.
  - Don't require the scale factor.

Proposed Hardware Architecture

- *Previous method's review:*

Method	Advantages	Drawbacks
Conv. CORDIC	- Low resource architecture.	- High iteration → high latency (N)
ARD CORDIC	- Low latency ($\frac{N}{2}$).	- High complexity.
SCFE CORDIC	- Low resource architecture.	- High iteration. - Low range of input angle.

- *Proposed Architecture's idea:*

Algorithm 1: The pseudocode of ARD-SCFE CORDIC scheme.

$z_0 = \Phi$;
$x_0 = 1$;
$y_0 = 0$;
while $|z_i| > threshold$ **do**
  Choose angle constant θ_i by ARD CORDIC algorithm
  if $i < \frac{(N-2.585)}{3}$ **then**
    Update x, y by Eq. of ARD CORDIC
    $z_{i+1} = z_i - \theta_i$
    Store the position i
  else
    Update x, y by Eq. of SCFE-CORDIC
    $z_{i+1} = z_i - \theta_i$
  end
end
$sin\Phi = K \times y_{N-1}$
$cos\Phi = K \times x_{N-1}$

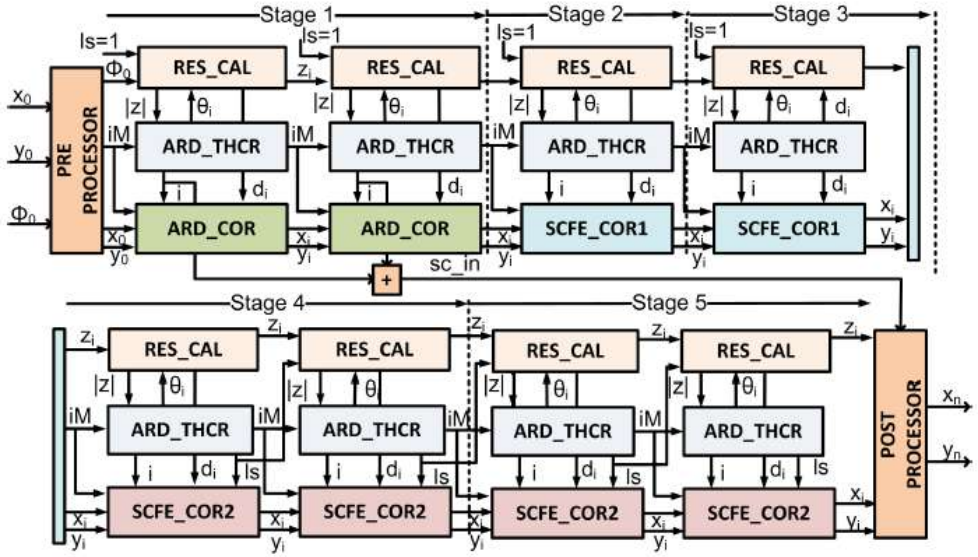

Fig. 3: The proposed architecture.

Results and Discussion

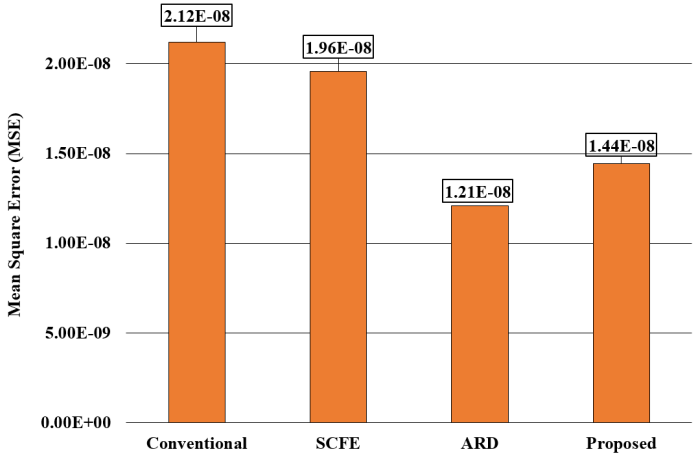

Fig. 4: The Mean Square Error (MSE) of each method.

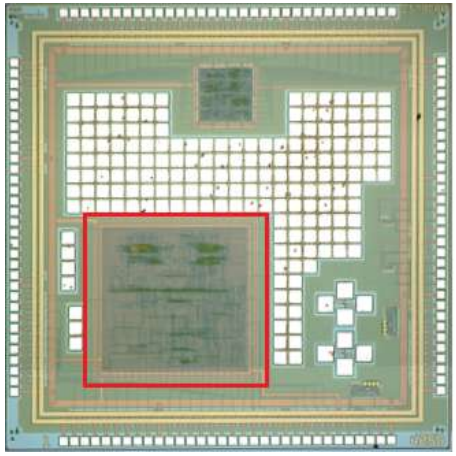

Fig. 5: The die photo of ARD-SCFE on 180 nm ROHM process.

Table 1: The comparison of algorithm complexity.

Scheme	Stages	Adders	Mul.s	Execution time
Conventional	16	48	1	16 Adders.
Angle recoding	8	24	9	8 Multipliers.
Scaling-free [7]	12	48	1	24 Adders.
RRMC [8]	7	78	0	26 Adders.
SFB [9]	9	36	1	18 Adders.
SFB4C [9]	7	32	1	14 Adders.
Adaptive recoding [10]	7	32	1	14 Adders.
Proposed	5	28	1	10 Adders.

➔Good trade-off between hardware complexity, execution time, and signal error.

Table 2: The comparison of ARD-SCFR arithmetic processor with previous design.

	[3]	[7]	Proposed
Technology (nm)	250 Bi CMOS	180 TSMC	180 ROHM
Algorithm	Scaling-free	Double rotation	ARD-SCFE
Function	Sine, cosine	Sine, cosine complex multiplier	Sine, cosine sinh, cosh, multiply
Output Data	16-bit Fixed	16-bit Fixed	18-bit Fixed
Supply Voltage (V)	2.5	–	1.8
Frequency (MHz)	20	67.1	100
Number of gates	12,350	174,138	10,720
Power (mW)	7	22.35	12.96
Energy (nJ/cycle)	0.35	0.33	0.13
Delay time (μs)	0.60	0.21	0.10
Throughput (Gbps)	0.64	2.15	3.6

➔High throughput and low-energy.

[3] T. Y. Sung et al., IET Compt. Dig. Tech., pp. 581 - 589, 2007.
[7] K. Maharatna et al., IEEE TCAS. for Video Tech., Vol. 15, pp. 1463- 1474, 2005.
[8] S. Aggarwal et al., IEEE Trans. VLSI Syst. Vol. 24, pp. 1588-1592, 2016.
[9] F. J. Jaime et al., IEEE TCAS I, Vol. 57, pp. 1654- 2010.
[10] J. Zhang et al., IEICE ELEX., Vol. 9, pp. 765- 2012.

THANK YOU

Contact information:
{hongthu,xuanthuan}@vlsilab.ee.uec.ac.jp
phamck@uec.ac.jp