WIKIPEDIA

CORDIC

CORDIC (for **CO**ordinate **R**otation **DI**gital **C**omputer), also known as **Volder's algorithm**, or: **Digit-by-digit method Circular CORDIC** (Jack E. Volder),[1][2] **Linear CORDIC**, **Hyperbolic CORDIC** (John Stephen Walther),[3][4] and **Generalized Hyperbolic CORDIC** (**GH CORDIC**) (Yuanyong Luo et al.),[5][6] is a simple and efficient algorithm to calculate trigonometric functions, hyperbolic functions, square roots, multiplications, divisions, and exponentials and logarithms with arbitrary base, typically converging with one digit (or bit) per iteration. CORDIC is therefore also an example of digit-by-digit algorithms. CORDIC and closely related methods known as **pseudo-multiplication** and **pseudo-division** or **factor combining** are commonly used when no hardware multiplier is available (e.g. in simple microcontrollers and FPGAs), as the only operations it requires are additions, subtractions, bitshift and lookup tables. As such, they all belong to the class of shift-and-add algorithms. In computer science, CORDIC is often used to implement floating-point arithmetic when the target platform lacks hardware multiply for cost or space reasons.

Contents

History

Similar mathematical techniques were published by Henry Briggs as early as 1624[7][8] and Robert Flower in 1771,[9] but CORDIC is better optimized for low-complexity finite-state CPUs.

CORDIC was conceived in 1956[10][11] by Jack E. Volder at the aeroelectronics department of Convair out of necessity to replace the analog resolver in the B-58 bomber's navigation computer with a more accurate and faster real-time digital solution.[11] Therefore, CORDIC is sometimes referred to as a digital resolver.[12][13]

In his research Volder was inspired by a formula in the 1946 edition of the *CRC Handbook of Chemistry and Physics*:[11]

$$\begin{aligned} K_n R\sin(\theta \pm \varphi) &= R\sin(\theta) \pm 2^{-n} R\cos(\theta), \\ K_n R\cos(\theta \pm \varphi) &= R\cos(\theta) \mp 2^{-n} R\sin(\theta), \end{aligned}$$

with $K_n = \sqrt{1+2^{-2n}}$, $\tan(\varphi) = 2^{-n}$.

His research led to an internal technical report proposing the CORDIC algorithm to solve sine and cosine functions and a prototypical computer implementing it.[10][11] The report also discussed the possibility to compute hyperbolic coordinate rotation, logarithms and exponential functions with modified CORDIC algorithms.[10][11] Utilizing CORDIC for multiplication and division was also conceived at this time.[11] Based on the CORDIC principle, Dan H. Daggett, a colleague of Volder at Convair, developed conversion algorithms between binary and binary-coded decimal (BCD).[11][14]

In 1958, Convair finally started to build a demonstration system to solve radar fix-taking problems named *CORDIC I*, completed in 1960 without Volder, who had left the company already.[1][11] More universal *CORDIC II* models *A* (stationary) and *B* (airborne) were built and tested by Daggett and Harry Schuss in 1962.[11][15]

Volder's CORDIC algorithm was first described in public in 1959,[1][2][11][13][16] which caused it to be incorporated into navigation computers by companies including Martin-Orlando, Computer Control, Litton, Kearfott, Lear-Siegler, Sperry, Raytheon, and Collins Radio.[11]

Volder teamed up with Malcolm McMillan to build *Athena*, a fixed-point desktop calculator utilizing his binary CORDIC algorithm.[17] The design was introduced to Hewlett-Packard in June 1965, but not accepted.[17] Still, McMillan introduced David S. Cochran (HP) to Volder's algorithm and when Cochran later met Volder he referred him to a similar approach John E. Meggitt (IBM[18]) had proposed as *pseudo-multiplication* and *pseudo-division* in 1961.[18][19] Meggitt's method was also suggesting the use of base 10[18] rather than base 2, as used by Volder's CORDIC so far. These efforts led to the ROMable logic implementation of a decimal CORDIC prototype machine inside of Hewlett-Packard in 1966,[20][19] build by and conceptually derived from Thomas E. Osborne's prototypical *Green Machine*, a four-function, floating-point desktop calculator he had completed in DTL logic[17] in December 1964.[21] This project resulted in the public demonstration of Hewlett-Packard's first desktop calculator with scientific functions, the hp 9100A in March 1968, with series production starting later that year.[17][21][22][23]

When Wang Laboratories found that the hp 9100A used an approach similar to the *factor combining* method in their earlier LOCI-1[24] (September 1964) and LOCI-2 (January 1965)[25][26] *Logarithmic Computing Instrument* desktop calculators,[27] they unsuccessfully accused Hewlett-Packard of infringement of one of An Wang's patents in 1968.[19][28][29][30]

John Stephen Walther at Hewlett-Packard generalized the algorithm into the *Unified CORDIC* algorithm in 1971, allowing it to calculate hyperbolic functions, natural exponentials, natural logarithms, multiplications, divisions, and square roots.[31][3][4][32] The CORDIC subroutines for trigonometric and hyperbolic functions could share most of their code.[28] This development resulted in the first scientific handheld calculator, the HP-35 in 1972.[28][33][34][35][36][37] Based on hyperbolic CORDIC, Yuanyong Luo et al. further proposed a Generalized Hyperbolic CORDIC (GH CORDIC) to directly compute logarithms and exponentials with an arbitrary fixed base in 2019.[5][6][38][39][40] Theoretically, Hyperbolic CORDIC is a special case of GH CORDIC.[5]

Originally, CORDIC was implemented only using the binary numeral system and despite Meggitt suggesting the use of the decimal system for his pseudo-multiplication approach, decimal CORDIC continued to remain mostly unheard of for several more years, so that Hermann Schmid and Anthony Bogacki still suggested it as a novelty as late as 1973[16][13][41][42][43] and it was found only later that Hewlett-Packard had implemented it in 1966 already.[11][13][20][28]

Decimal CORDIC became widely used in pocket calculators,[13] most of which operate in binary-coded decimal (BCD) rather than binary. This change in the input and output format did not alter CORDIC's core calculation algorithms. CORDIC is particularly well-suited for handheld calculators, in which low cost – and thus low chip gate count – is much more important than speed.

CORDIC has been implemented in the ARM-based STM32G4, Intel 8087,[43][44][45][46][47] 80287,[47][48] 80387[47][48] up to the 80486[43] coprocessor series as well as in the Motorola 68881[43][44] and 68882 for some kinds of floating-point instructions, mainly as a way to reduce the gate counts (and complexity) of the FPU sub-system.

Applications

CORDIC uses simple shift-add operations for several computing tasks such as the calculation of trigonometric, hyperbolic and logarithmic functions, real and complex multiplications, division, square-root calculation, solution of linear systems, eigenvalue estimation, singular value decomposition, QR factorization and many others. As a consequence, CORDIC has been used for applications in diverse areas such as signal and image processing, communication systems, robotics and 3D graphics apart from general scientific and technical computation.[49][50]

Hardware

The algorithm was used in the navigational system of the Apollo program's Lunar Roving Vehicle to compute bearing and range, or distance from the Lunar module.[51]:14[52]:17 CORDIC was used to implement the Intel 8087 math coprocessor in 1980, avoiding the need to implement hardware multiplication.[53]

CORDIC is generally faster than other approaches when a hardware multiplier is not available (e.g., a microcontroller), or when the number of gates required to implement the functions it supports should be minimized (e.g., in an FPGA or ASIC). In fact, CORDIC is a standard drop-in IP in FPGA development applications such as Vivado for Xilinx, while a power series implementation is not due to the specificity of such an IP, i.e. CORDIC can compute many different functions (general purpose) while a hardware multiplier configured to execute power series implementations can only compute the function it was designed for.

On the other hand, when a hardware multiplier is available (*e.g.*, in a DSP microprocessor), table-lookup methods and power series are generally faster than CORDIC. In recent years, the CORDIC algorithm has been used extensively for various biomedical applications, especially in FPGA implementations.

The STM32G4 series and certain STM32H7 series of MCUs implement a CORDIC module to accelerate computations in various mixed signal applications such as graphics for Human Machine Interface and field oriented control of motors. While not as fast as a power series approximation, CORDIC is indeed faster than interpolating table based implementations such as the ones provided by the ARM CMSIS and C standard libraries.[54] Though the results may be slightly less accurate as the CORDIC modules provided only achieve 20 bits of precision in the result. For example, most of the performance difference compared to the ARM implementation is due to the overhead of the interpolation algorithm, which achieves full floating point precision (24 bits) and can likely achieve relative error to that precision.[55] Another benefit is that the CORDIC module is a coprocessor and can be run in parallel with other CPU tasks.

The issue with using power series is that while they do provide small absolute error, they do not exhibit well behaved relative error.[56]

Software

Many older systems with integer-only CPUs have implemented CORDIC to varying extents as part of their IEEE floating-point libraries. As most modern general-purpose CPUs have floating-point registers with common operations such as add, subtract, multiply, divide, sine, cosine, square root, $\log_{10}$, natural log, the need to implement CORDIC in them with software is nearly non-existent. Only microcontroller or special safety and time-constrained software applications would need to consider using CORDIC.

Modes of operation

Rotation mode

CORDIC can be used to calculate a number of different functions. This explanation shows how to use CORDIC in *rotation mode* to calculate the sine and cosine of an angle, assuming that the desired angle is given in radians and represented in a fixed-point format. To determine the sine or cosine for an angle β, the y or x coordinate of a point on the unit circle corresponding to the desired angle must be found. Using CORDIC, one would start with the vector v_0:

$$v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

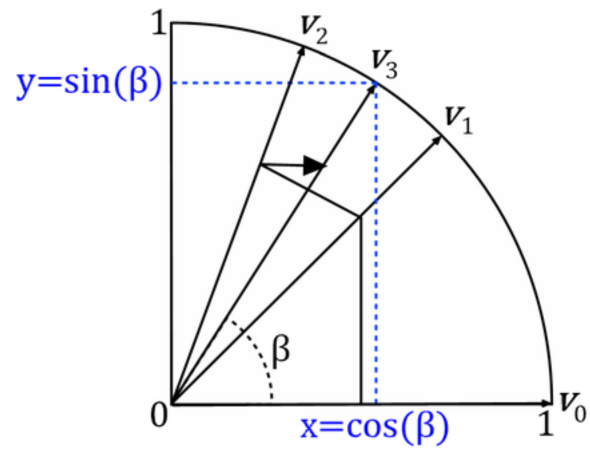
In the first iteration, this vector is rotated 45° counterclockwise to get the vector v_1. Successive iterations rotate the vector in one or the other direction by size-decreasing steps, until the desired angle has been achieved. Step i size is $\arctan\left(2^{-i}\right)$ for $i = 0, 1, 2, \ldots$.

More formally, every iteration calculates a rotation, which is performed by multiplying the vector v_i with the rotation matrix R_i:

$$v_{i+1} = R_i v_i.$$

The rotation matrix is given by

$$R_i = \begin{bmatrix} \cos(\gamma_i) & -\sin(\gamma_i) \\ \sin(\gamma_i) & \cos(\gamma_i) \end{bmatrix}.$$



An illustration of the CORDIC algorithm in progress

Using the following two trigonometric identities:

$$\cos(\gamma_i) = \frac{1}{\sqrt{1+\tan^2(\gamma_i)}},$$
$$\sin(\gamma_i) = \frac{\tan(\gamma_i)}{\sqrt{1+\tan^2(\gamma_i)}},$$

the rotation matrix becomes

$$R_i = \frac{1}{\sqrt{1+\tan^2(\gamma_i)}} \begin{bmatrix} 1 & -\tan(\gamma_i) \\ \tan(\gamma_i) & 1 \end{bmatrix}.$$

The expression for the rotated vector $v_{i+1} = R_i v_i$ then becomes

$$v_{i+1} = \frac{1}{\sqrt{1+\tan^2(\gamma_i)}} \begin{bmatrix} 1 & -\tan(\gamma_i) \\ \tan(\gamma_i) & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix},$$

where x_i and y_i are the components of v_i. Restricting the angles γ_i such that $\tan(\gamma_i) = \pm 2^{-i}$, the multiplication with the tangent can be replaced by a division by a power of two, which is efficiently done in digital computer hardware using a bit shift. The expression then becomes

$$v_{i+1} = K_i \begin{bmatrix} 1 & -\sigma_i 2^{-i} \\ \sigma_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \end{bmatrix},$$

where

$$K_i = \frac{1}{\sqrt{1+2^{-2i}}},$$

and σ_i is used to determine the direction of the rotation: if the angle γ_i is positive, then σ_i is +1, otherwise it is −1.

K_i can be ignored in the iterative process and then applied afterward with a scaling factor

$$K(n) = \prod_{i=0}^{n-1} K_i = \prod_{i=0}^{n-1} \frac{1}{\sqrt{1+2^{-2i}}},$$

which is calculated in advance and stored in a table or as a single constant, if the number of iterations is fixed. This correction could also be made in advance, by scaling v_0 and hence saving a multiplication. Additionally, it can be noted that[43]

$$K = \lim_{n \to \infty} K(n) \approx 0.6072529350088812561694$$

to allow further reduction of the algorithm's complexity. Some applications may avoid correcting for K altogether, resulting in a processing gain A:[57]

$$A = \frac{1}{K} = \lim_{n \to \infty} \prod_{i=0}^{n-1} \sqrt{1 + 2^{-2i}} \approx 1.64676025812107.$$

After a sufficient number of iterations, the vector's angle will be close to the wanted angle β. For most ordinary purposes, 40 iterations (n = 40) is sufficient to obtain the correct result to the 10th decimal place.

The only task left is to determine whether the rotation should be clockwise or counterclockwise at each iteration (choosing the value of σ). This is done by keeping track of how much the angle was rotated at each iteration and subtracting that from the wanted angle; then in order to get closer to the wanted angle β, if β_{n+1} is positive, the rotation is clockwise, otherwise it is negative and the rotation is counterclockwise:

$$\beta_0 = \beta$$

$$\beta_{i+1} = \beta_i - \sigma_i \gamma_i, \quad \gamma_i = \arctan(2^{-i}).$$

The values of γ_n must also be precomputed and stored. But for small angles, $\arctan(\gamma_n) = \gamma_n$ in fixed-point representation, reducing table size.

As can be seen in the illustration above, the sine of the angle β is the y coordinate of the final vector v_n, while the x coordinate is the cosine value.

Vectoring mode

The rotation-mode algorithm described above can rotate any vector (not only a unit vector aligned along the x axis) by an angle between −90° and +90°. Decisions on the direction of the rotation depend on β_i being positive or negative.

The vectoring-mode of operation requires a slight modification of the algorithm. It starts with a vector the x coordinate of which is positive and the y coordinate is arbitrary. Successive rotations have the goal of rotating the vector to the x axis (and therefore reducing the y coordinate to zero). At each step, the value of y determines the direction of the rotation. The final value of β_i contains the total angle of rotation. The final value of x will be the magnitude of the original vector scaled by K. So, an obvious use of the vectoring mode is the transformation from rectangular to polar coordinates.

Implementation

Software example

The following is a MATLAB/GNU Octave implementation of CORDIC that does not rely on any transcendental functions except in the precomputation of tables. If the number of iterations n is predetermined, then the second table can be replaced by a single constant. With MATLAB's standard double-precision arithmetic and "format long" printout, the results increase in accuracy for n up to about 48.

```
function v = cordic(beta,n)
% This function computes v = [cos(beta), sin(beta)] (beta in radians)
% using n iterations. Increasing n will increase the precision.

if beta < -pi/2 || beta > pi/2
    if beta < 0
        v = cordic(beta + pi, n);
    else
        v = cordic(beta - pi, n);
    end
    v = -v; % flip the sign for second or third quadrant
    return
end

% Initialization of tables of constants used by CORDIC
% need a table of arctangents of negative powers of two, in radians:
% angles = atan(2.^-(0:27));
angles =  [  ...
    0.78539816339745   0.46364760900081   0.24497866312686   0.12435499454676 ...
    0.06241880999596   0.03123983343027   0.01562372862048   0.00781234106010 ...
    0.00390623013197   0.00195312251648   0.00097656218956   0.00048828121119 ...
    0.00024414062015   0.00012207031189   0.00006103515617   0.00003051757812 ...
    0.00001525878906   0.00000762939453   0.00000381469727   0.00000190734863 ...
    0.00000095367432   0.00000047683716   0.00000023841858   0.00000011920929 ...
    0.00000005960464   0.00000002980232   0.00000001490116   0.00000000745058 ];
% and a table of products of reciprocal lengths of vectors [1, 2^-2j]:
% Kvalues = cumprod(1./abs(1 + 1j*2.^(-(0:23))))
Kvalues = [ ...
    0.70710678118655   0.63245553203368   0.61357199107790   0.60883391251775 ...
    0.60764825625617   0.60735177014130   0.60727764409353   0.60725911229889 ...
    0.60725447933256   0.60725332108988   0.60725303152913   0.60725295913894 ...
    0.60725294104140   0.60725293651701   0.60725293538591   0.60725293510314 ...
    0.60725293503245   0.60725293501477   0.60725293501035   0.60725293500925 ...
    0.60725293500897   0.60725293500890   0.60725293500889   0.60725293500888 ];
Kn = Kvalues(min(n, length(Kvalues)));

% Initialize loop variables:
v = [1;0]; % start with 2-vector cosine and sine of zero
poweroftwo = 1;
angle = angles(1);

% Iterations
for j = 0:n-1;
    if beta < 0
        sigma = -1;
    else
        sigma = 1;
    end
    factor = sigma * poweroftwo;
    % Note the matrix multiplication can be done using scaling by powers of two and addition subtraction
    R = [1, -factor; factor, 1];
    v = R * v; % 2-by-2 matrix multiply
    beta = beta - sigma * angle; % update the remaining angle
    poweroftwo = poweroftwo / 2;
    % update the angle from table, or eventually by just dividing by two
    if j+2 > length(angles)
        angle = angle / 2;
    else
        angle = angles(j+2);
    end
end

% Adjust length of output vector to be [cos(beta), sin(beta)]:
v = v * Kn;
```

```
return

endfunction
```

The two-by-two matrix multiplication can be carried out by a pair of simple shifts and adds.

```
    x = v[0] - sigma * (v[1] * 2^(-j));
    y = sigma * (v[0] * 2^(-j)) + v[1];
    v = [x; y];
```

In Java the Math class has a `scalb(double x,int scale)` method to perform such a shift,[58] C has the ldexp function,[59] and the x86 class of processors have the `fscale` floating point operation.[60]

Hardware example

The number of logic gates for the implementation of a CORDIC is roughly comparable to the number required for a multiplier as both require combinations of shifts and additions. The choice for a multiplier-based or CORDIC-based implementation will depend on the context. The multiplication of two complex numbers represented by their real and imaginary components (rectangular coordinates), for example, requires 4 multiplications, but could be realized by a single CORDIC operating on complex numbers represented by their polar coordinates, especially if the magnitude of the numbers is not relevant (multiplying a complex vector with a vector on the unit circle actually amounts to a rotation). CORDICs are often used in circuits for telecommunications such as digital down converters.

Double iterations CORDIC

In the publications: http://baykov.de/CORDIC1972.htm and http://baykov.de/CORDIC1975.htm it was proposed to use the **double iterations** method for the implementation of the functions: arcsinX, arccosX, lnX, expX, as well as for calculation of the hyperbolic functions. Double iterations method consists in the fact that unlike the classical CORDIC method, where the iteration step value changes EVERY time, i.e. on each iteration, in the double iteration method, the iteration step value is repeated twice and changes only through one iteration. Hence the designation for the degree indicator for double iterations appeared: i = 1,1,2,2,3,3... Whereas with ordinary iterations: i = 1,2,3... The double iteration method guarantees the convergence of the method throughout the valid range of argument changes.

The generalization of the CORDIC convergence problems for the arbitrary positional number system http://baykov.de/CORDIC1985.htm with Radix R showed that for the functions sin, cos, arctg, it is enough to perform (R-1) iterations for each value of i (i = 0 or 1 to n, where n is the number of digits), i.e. for each digit of the result. For the functions ln, exp, sh, ch, arth, R iterations should be performed for each value i. For the functions arcsin and arccos 2 (R-1) iterations should be performed for each number digit, i.e. for each value of i. For arsh, arch functions, the number of iterations will be 2R for each i, that is, for each result digit.

Related algorithms

CORDIC is part of the class of "shift-and-add" algorithms, as are the logarithm and exponential algorithms derived from Henry Briggs' work. Another shift-and-add algorithm which can be used for computing many elementary functions is the BKM algorithm, which is a generalization of the logarithm and exponential algorithms to the complex plane. For instance, BKM can be used to compute the sine and cosine of a real angle x (in radians) by computing the exponential of $0 + ix$, which is $\operatorname{cis}(x) = \cos(x) + i\sin(x)$. The BKM algorithm is slightly more complex than CORDIC, but has the advantage that it does not need a scaling factor (K).

See also

- Methods of computing square roots
- IEEE 754
- Floating-point units
- Digital Circuits/CORDIC in Wikibooks

References

1. Volder, Jack E. (1959-03-03). "The CORDIC Computing Technique" (http://www.computer.org/csdl/proceedings/afips/1959/5054/00/50540257.pdf) (PDF). *Proceedings of the Western Joint Computer Conference (WJCC)* (presentation). San Francisco, California, USA: National Joint Computer Committee (NJCC): 257–261. Retrieved 2016-01-02.
2. Volder, Jack E. (1959-05-25). "The CORDIC Trigonometric Computing Technique" (http://home.citycable.ch/pierrefleur/Jacques-Laporte/Volder_CORDIC.pdf) (PDF). *IRE Transactions on Electronic Computers*. The Institute of Radio Engineers, Inc. (IRE) (published September 1959). **8** (3): 330–334 (reprint: 226–230). EC-8(3):330–334. Retrieved 2016-01-01.
3. Walther, John Stephen (May 1971). Written at Palo Alto, California, USA. "A unified algorithm for elementary functions" (http://home.citycable.ch/pierrefleur/Jacques-Laporte/Welther-Unified%20Algorithm.pdf) (PDF). *Proceedings of the Spring Joint Computer Conference*. Atlantic City, New Jersey, USA: Hewlett-Packard Company. **38**: 379–385 – via American Federation of Information Processing Societies (AFIPS).
4. Walther, John Stephen (June 2000). "The Story of Unified CORDIC" (https://dl.acm.org/citation.cfm?id=2812970). *The Journal of VLSI Signal Processing*. Hingham, MA, USA: Kluwer Academic Publishers. **25** (2 (Special issue on CORDIC)): 107–112. doi:10.1023/A:1008162721424 (https://doi.org/10.1023%2FA%3A1008162721424). ISSN 0922-5773 (https://www.worldcat.org/issn/0922-5773). S2CID 26922158 (https://api.semanticscholar.org/CorpusID:26922158).
5. Luo, Yuanyong; Wang, Yuxuan; Ha, Yajun; Wang, Zhongfeng; Chen, Siyuan; Pan, Hongbing (September 2019). "Generalized Hyperbolic CORDIC and Its Logarithmic and Exponential Computation With Arbitrary Fixed Base". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. **27** (9): 2156–2169. doi:10.1109/TVLSI.2019.2919557 (https://doi.org/10.1109%2FTVLSI.2019.2919557). S2CID 196171166 (https://api.semanticscholar.org/CorpusID:196171166).
6. Luo, Yuanyong; Wang, Yuxuan; Ha, Yajun; Wang, Zhongfeng; Chen, Siyuan; Pan, Hongbing (September 2019). "Corrections to "Generalized Hyperbolic CORDIC and Its Logarithmic and Exponential Computation With Arbitrary Fixed Base" ". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. **27** (9): 2222. doi:10.1109/TVLSI.2019.2932174 (https://doi.org/10.1109%2FTVLSI.2019.2932174).
7. Briggs, Henry (1624). *Arithmetica Logarithmica*. London. (Translation: [1] (http://www-gap.dcs.st-and.ac.uk/~history/Miscellaneous/Briggs/index.html) Archived (https://web.archive.org/web/20160304192134/http://www-gap.dcs.st-and.ac.uk/~history/Miscellaneous/Briggs/index.html) 4 March 2016 at the Wayback Machine)

8. Laporte, Jacques (2014) [2005]. "Henry Briggs and the HP 35" (https://web.archive.org/web/20150309055201/http://www.jacques-laporte.org/Briggs%20and%20the%20HP35.htm). Paris, France. Archived from the original (http://www.jacques-laporte.org/Briggs%20and%20the%20HP35.htm) on 2015-03-09. Retrieved 2016-01-02. [2] (http://home.citycable.ch/pierrefleur/Jacques-Laporte/Briggs%20and%20the%20HP35.htm)
9. Flower, Robert (1771). *The Radix. A new way of making logarithms* (https://books.google.com/books?id=mYpaAAAAcAAJ). London: J. Beecroft. Retrieved 2016-01-02.
10. Volder, Jack E. (1956-06-15), *Binary Computation Algorithms for Coordinate Rotation and Function Generation* (internal report), Convair, Aeroelectronics group, IAR-1.148
11. Volder, Jack E. (June 2000). "The Birth of CORDIC" (https://web.archive.org/web/20160304064804/http://late-dpedago.urv.cat/site_media/papers/fulltext_2.pdf) (PDF). *Journal of VLSI Signal Processing*. Hingham, MA, USA: Kluwer Academic Publishers. **25** (2 (Special issue on CORDIC)): 101–105. doi:10.1023/A:1008110704586 (https://doi.org/10.1023%2FA%3A1008110704586). ISSN 0922-5773 (https://www.worldcat.org/issn/0922-5773). S2CID 112881 (https://api.semanticscholar.org/CorpusID:112881). Archived from the original (http://late-dpedago.urv.cat/site_media/papers/fulltext_2.pdf) (PDF) on 2016-03-04. Retrieved 2016-01-02.
12. Perle, Michael D. (June 1971), "CORDIC Technique Reduces Trigonometric Function Look-Up", *Computer Design*, Boston, MA, USA: Computer Design Publishing Corp.: 72–78 (NB. Some sources erroneously refer to this as by *P. Z. Perle* or in *Component Design*.)
13. Schmid, Hermann (1983) [1974]. *Decimal Computation* (https://books.google.com/books?id=uEYZAQAAIAAJ) (1 (reprint) ed.). Malabar, Florida, USA: Robert E. Krieger Publishing Company. pp. 162, 165–176, 181–193. ISBN 0-89874-318-4. Retrieved 2016-01-03. (NB. At least some batches of this reprint edition were misprints with defective pages 115–146.)
14. Daggett, Dan H. (September 1959). "Decimal-Binary Conversions in CORDIC" (https://www.researchgate.net/researcher/74881302_D_H_Daggett). *IRE Transactions on Electronic Computers*. The Institute of Radio Engineers, Inc. (IRE). **8** (3): 335–339. doi:10.1109/TEC.1959.5222694 (https://doi.org/10.1109%2FTEC.1959.5222694). ISSN 0367-9950 (https://www.worldcat.org/issn/0367-9950). EC-8(3):335–339. Retrieved 2016-01-02.
15. Advanced Systems Group (1962-08-06), *Technical Description of Fix-taking Tie-in Equipment* (report), Fort Worth, Texas, USA: General Dynamics, FZE-052
16. Schmid, Hermann (1974). *Decimal Computation* (https://archive.org/details/decimalcomputati0000schm) (1 ed.). Binghamton, New York, USA: John Wiley & Sons, Inc. pp. 162 (https://archive.org/details/decimalcomputati0000schm/page/162), 165–176, 181–193. ISBN 0-471-76180-X. Retrieved 2016-01-03. "So far CORDIC has been known to be implemented only in binary form. But, as will be demonstrated here, the algorithm can be easily modified for a decimal system.* [...] *In the meantime it has been learned that Hewlett Packard and other calculator manufacturers employ the decimal CORDIC techniques in their scientific calculators."
17. Leibson, Steven (2010). "The HP 9100 Project: An Exothermic Reaction" (http://www.hp9825.com/html/the_9100_part_2.html). Retrieved 2016-01-02.
18. Meggitt, John E. (1961-08-29). "Pseudo Division and Pseudo Multiplication Processes" (http://home.citycable.ch/pierrefleur/Jacques-Laporte/Meggitt_62.pdf) (PDF). *IBM Journal of Research and Development*. Riverton, New Jersey, USA: IBM Corporation (published April 1962). **6** (2): 210–226, 287. doi:10.1147/rd.62.0210 (https://doi.org/10.1147%2Frd.62.0210). Retrieved 2016-01-09. "John E. Meggitt B.A., 1953; PhD, 1958, Cambridge University. Awarded the First Smith Prize at Cambridge in 1955 and elected a Research Fellowship at Emmanuel College. [...] Joined IBM British Laboratory at Hursley, Winchester in 1958. Interests include error-correcting codes and small microprogrammed computers." ([3] (http://ieeexplore.ieee.org/stamp/stamp.jsp?reload=true&tp=&arnumber=5392370), [4] (http://ieeexplore.ieee.org/stamp/stamp.jsp?reload=true&tp=&arnumber=5392370))

19. Cochran, David S. (2010-11-19). "A Quarter Century at HP" (http://www.hpmemoryproject.org/timeline/dave_cochran/a_quarter_century_at_hp_00.htm#chapter_07) (interview typescript). Computer History Museum / HP Memories. 7: Scientific Calculators, circa 1966. CHM X5992.2011. Retrieved 2016-01-02. "I even flew down to Southern California to talk with Jack Volder who had implemented the transcendental functions in the *Athena* machine and talked to him for about an hour. He referred me to the original papers by Meggitt where he'd gotten the pseudo division, pseudo multiplication generalized functions. [...] I did quite a bit of literary research leading to some very interesting discoveries. [...] I found a treatise from 1624 by Henry Briggs discussing the calculation of common logarithms, interestingly used the same pseudo-division/pseudo-multiplication method that MacMillan and Volder used in *Athena*. [...] We had purchased a LOCI-2 from Wang Labs and recognized that Wang Labs LOCI II used the same algorithm to do square root as well as log and exponential. After the introduction of the 9100 our legal department got a letter from Wang saying that we had infringed on their patent. And I just sent a note back with the Briggs reference in Latin and it said, "It looks like prior art to me." We never heard another word." ([5] (http://www.hpmemoryproject.org/an/pdf/A_Quarter_Century_at_HP110829.pdf))
20. Cochran, David S. (1966-03-14). "About utilizing CORDIC for computing transcendental functions in BCD" (private communication with Jack E. Volder).
21. Osborne, Thomas E. (2010) [1994]. "Tom Osborne's Story in His Own Words" (http://www.hp9825.com/html/osborne_s_story.html). Retrieved 2016-01-01.
22. Leibson, Steven (2010). "The HP 9100: The Initial Journey" (http://www.hp9825.com/html/the_9100_project.html). Retrieved 2016-01-02.
23. Cochran, David S. (September 1968). "Internal Programming of the 9100A Calculator" (http://www.hpmemoryproject.org/timeline/dave_cochran/hpj_sep68.htm). *Hewlett-Packard Journal*. Palo Alto, California, USA: Hewlett-Packard: 14–16. Retrieved 2016-01-02. ([6] (http://www.hparchive.com/Journals/HPJ-1968-09.pdf))
24. *Extend your Personal Computing Power with the new LOCI-1 Logarithmic Computing Instrument* (http://www.oldcalculatormuseum.com/a-loci1br-23.html), Wang Laboratories, Inc., 1964, pp. 2–3, retrieved 2016-01-03
25. Bensene, Rick (2013-08-31) [1997]. "Wang LOCI-2" (http://www.oldcalculatormuseum.com/wangloci.html). *Old Calculator Web Museum*. Beavercreek, Oregon City, Oregon, USA. Retrieved 2016-01-03.
26. "Wang LOCI Service Manual" (http://bitsavers.informatik.uni-stuttgart.de/pdf/wang/loci/Wang_LOCI_Service_Manual.pdf) (PDF). Wang Laboratories, Inc. 1967. L55-67. Retrieved 2018-09-14.
27. Bensene, Rick (2004-10-23) [1997]. "Wang Model 360SE Calculator System" (http://www.oldcalculatormuseum.com/wang360.html). *Old Calculator Web Museum*. Beavercreek, Oregon City, Oregon, USA. Retrieved 2016-01-03.

28. Cochran, David S. (June 2010). "The HP-35 Design, A Case Study in Innovation" (http://www.hpmemoryproject.org/wb_pages/d_cochran_01.htm). HP Memory Project. Retrieved 2016-01-02. "During the development of the desktop HP 9100 calculator I was responsible for developing the algorithms to fit the architecture suggested by Tom Osborne. Although the suggested methodology for the algorithms came from Malcolm McMillan I did considerable amount of reading to understand the core calculations [...] Although Wang Laboratories had used similar methods of calculation, my study found prior art dated 1624 that read on their patents. [...] This research enabled the adaption of the transcendental functions through the use of the algorithms to match the needs of the customer within the constraints of the hardware. This proved invaluable during the development of the HP-35, [...] Power series, polynomial expansions, continued fractions, and Chebyshev polynomials were all considered for the transcendental functions. All were too slow because of the number of multiplications and divisions required. The generalized algorithm that best suited the requirements of speed and programming efficiency for the HP-35 was an iterative pseudo-division and pseudo-multiplication method first described in 1624 by Henry Briggs in 'Arithmetica Logarithmica' and later by Volder and Meggitt. This is the same type of algorithm that was used in previous HP desktop calculators. [...] The complexity of the algorithms made multilevel programming a necessity. This meant the calculator had to have subroutine capability, [...] To generate a transcendental function such as Arc-Hyperbolic-Tan required several levels of subroutines. [...] Chris Clare later documented this as *Algorithmic State Machine* (ASM) methodology. Even the simple Sine or Cosine used the Tangent routine, and then calculated the Sine from trigonometric identities. These arduous manipulations were necessary to minimize the number of unique programs and program steps [...] The arithmetic instruction set was designed specifically for a decimal transcendental-function calculator. The basic arithmetic operations are performed by a 10's complement adder-subtractor which has data paths to three of the registers that are used as working storage."
29. US patent 3402285A (https://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=US3402285A), Wang, An, "Calculating apparatus", published 1968-09-17, issued 1968-09-17, assigned to Wang Laboratories ([7] (http://www.freepatentsonline.com/3402285.html), [8] (http://www.google.com/patents/US3402285))
30. DE patent 1499281B1 (https://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=DE1499281B1), Wang, An, "Rechenmaschine fuer logarithmische Rechnungen", published 1970-05-06, issued 1970-05-06, assigned to Wang Laboratories ([9] (http://www.google.com/patents/DE1499281B1))
31. Swartzlander, Jr., Earl E. (1990). *Computer Arithmetic* (https://books.google.com/books?id=egIpAQAAMAAJ). **1** (2 ed.). Los Alamitos: IEEE Computer Society Press. ISBN 9780818689314. 0818689315. Retrieved 2016-01-02.
32. Petrocelli, Orlando R., ed. (1972), *The Best Computer Papers of 1971* (https://books.google.com/books?id=f6ezAAAAIAAJ), Auerbach Publishers, p. 71, ISBN 0877691274, retrieved 2016-01-02
33. Cochran, David S. (June 1972). "Algorithms and Accuracy in the HP-35" (http://www.hpl.hp.com/hpjournal/72jun/jun72a2.pdf) (PDF). *Hewlett-Packard Journal*. **23** (10): 10–11.
34. Laporte, Jacques (2005-12-06). "HP35 trigonometric algorithm" (https://web.archive.org/web/20150309055210/http://www.jacques-laporte.org/Trigonometry.htm). Paris, France. Archived from the original (http://www.jacques-laporte.org//Trigonometry.htm) on 2015-03-09. Retrieved 2016-01-02. [10] (http://home.citycable.ch/pierrefleur/Jacques-Laporte/Trigonometry.htm)
35. Laporte, Jacques (February 2005) [1981]. "The secret of the algorithms" (https://web.archive.org/web/20160818122704/http://www.jacques-laporte.org/TheSecretOfTheAlgorithms.htm). *L'Ordinateur Individuel*. Paris, France (24). Archived from the original (http://www.jacques-laporte.org/TheSecretOfTheAlgorithms.htm) on 2016-08-18. Retrieved 2016-01-02. [11] (http://home.citycable.ch/pierrefleur/Jacques-Laporte/TheSecretOfTheAlgorithms.htm)

36. Laporte, Jacques (February 2012) [2006]. "Digit by digit methods" (https://web.archive.org/web/20160818121038/http://www.jacques-laporte.org/digit_by_digit.htm). Paris, France. Archived from the original (http://www.jacques-laporte.org/digit_by_digit.htm) on 2016-08-18. Retrieved 2016-01-02. [12] (http://home.citycable.ch/pierrefleur/Jacques-Laporte/digit_by_digit.htm)
37. Laporte, Jacques (February 2012) [2007]. "HP 35 Logarithm Algorithm" (https://web.archive.org/web/20160818120118/http://www.jacques-laporte.org/Logarithm_1.htm). Paris, France. Archived from the original (http://www.jacques-laporte.org/Logarithm_1.htm) on 2016-08-18. Retrieved 2016-01-07. [13] (http://home.citycable.ch/pierrefleur/Jacques-Laporte/Logarithm_1.htm)
38. Wang, Yuxuan; Luo, Yuanyong; Wang, Zhongfeng; Shen, Qinghong; Pan, Hongbing (January 2020). "GH CORDIC-Based Architecture for Computing Nth Root of Single-Precision Floating-Point Number". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. **28** (4): 864–875. doi:10.1109/TVLSI.2019.2959847 (https://doi.org/10.1109%2FTVLSI.2019.2959847). S2CID 212975618 (https://api.semanticscholar.org/CorpusID:212975618).
39. Mopuri, Suresh; Acharyya, Amit (September 2019). "Low Complexity Generic VLSI Architecture Design Methodology for Nth Root and Nth Power Computations". *IEEE Transactions on Circuits and Systems I: Regular Papers*. **66** (12): 4673–4686. doi:10.1109/TCSI.2019.2939720 (https://doi.org/10.1109%2FTCSI.2019.2939720). S2CID 203992880 (https://api.semanticscholar.org/CorpusID:203992880).
40. Vachhani, Leena (November 2019). "CORDIC as a Switched Nonlinear System". *Circuits, Systems, and Signal Processing*. **39** (6): 3234–3249. doi:10.1007/s00034-019-01295-8 (https://doi.org/10.1007%2Fs00034-019-01295-8). S2CID 209904108 (https://api.semanticscholar.org/CorpusID:209904108).
41. Schmid, Hermann; Bogacki, Anthony (1973-02-20). "Use Decimal CORDIC for Generation of Many Transcendental Functions". *EDN*: 64–73.
42. Franke, Richard (1973-05-08). *An Analysis of Algorithms for Hardware Evaluation of Elementary Functions* (http://calhoun.nps.edu/bitstream/handle/10945/29706/analysisofalgori00fran.pdf) (PDF). Monterey, California, USA: Department of the Navy, Naval Postgraduate School. NPS-53FE73051A. Retrieved 2016-01-03.
43. Muller, Jean-Michel (2006). *Elementary Functions: Algorithms and Implementation* (http://perso.ens-lyon.fr/jean-michel.muller/SecondEdition.html) (2 ed.). Boston: Birkhäuser. p. 134. ISBN 978-0-8176-4372-0. LCCN 2005048094 (https://lccn.loc.gov/2005048094). Retrieved 2015-12-01.
44. Nave, Rafi (March 1983). "Implementation of Transcendental Functions on a Numerics Processor". *Microprocessing and Microprogramming*. **11** (3–4): 221–225. doi:10.1016/0165-6074(83)90151-5 (https://doi.org/10.1016%2F0165-6074%2883%2990151-5).
45. Palmer, John F.; Morse, Stephen Paul (1984). *The 8087 Primer* (https://archive.org/details/8087primer00palm) (1 ed.). John Wiley & Sons Australia, Limited. ISBN 0471875694. 9780471875697. Retrieved 2016-01-02.
46. Glass, L. Brent (January 1990). "Math Coprocessors: A look at what they do, and how they do it". *Byte*. **15** (1): 337–348. ISSN 0360-5280 (https://www.worldcat.org/issn/0360-5280).
47. Jarvis, Pitts (1990-10-01). "Implementing CORDIC algorithms – A single compact routine for computing transcendental functions" (https://web.archive.org/web/20160304085613/http://www.drdobbs.com/database/implementing-cordic-algorithms/184408428). *Dr. Dobb's Journal*: 152–156. Archived from the original (http://www.drdobbs.com/database/implementing-cordic-algorithms/184408428) on 2016-03-04. Retrieved 2016-01-02.
48. Yuen, A. K. (1988). "Intel's Floating-Point Processors". *Electro/88 Conference Record*: 48/5/1–7.

49. Meher, Pramod Kumar; Valls, Javier; Juang, Tso-Bing; Sridharan, K.; Maharatna, Koushik (2008-08-22). "50 Years of CORDIC: Algorithms, Architectures and Applications" (https://eprints.soton.ac.uk/267873/1/tcas1_cordic_review.pdf) (PDF). *IEEE Transactions on Circuits and Systems I: Regular Papers* (published 2009-09-09). **56** (9): 1893–1907. doi:10.1109/TCSI.2009.2025803 (https://doi.org/10.1109%2FTCSI.2009.2025803). S2CID 5465045 (https://api.semanticscholar.org/CorpusID:5465045).
50. Meher, Pramod Kumar; Park, Sang Yoon (February 2013). "Low Complexity Generic VLSI Architecture Design Methodology for Nth Root and Nth Power Computations". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. **21** (2): 217–228. doi:10.1109/TVLSI.2012.2187080 (https://doi.org/10.1109%2FTVLSI.2012.2187080). S2CID 7059383 (https://api.semanticscholar.org/CorpusID:7059383).
51. Heffron, W.G.; LaPiana, F. (1970-12-11). "Technical Memorandum 70-2014-8: The Navigation System of the Lunar Roving Vehicle" (https://www.hq.nasa.gov/alsj/19790072520_1979072520.pdf) (PDF). *NASA*. Washington, D.C.: Bellcomm.
52. Smith, Earnest C.; Mastin, William C. (November 1973). "Technical Note D-7469: Lunar Roving Vehicle Navigation System Performance Review" (https://www.hq.nasa.gov/alsj/19740003321_1974003321.pdf) (PDF). *NASA*. Huntsville, Alabama: Marshall Space Flight Center.
53. Shirriff, Ken (May 2020). "Extracting ROM constants from the 8087 math coprocessor's die" (http://www.righto.com/2020/05/extracting-rom-constants-from-8087-math.html). *righto.com*. Self-published by Ken Shirriff. Retrieved 2020-09-03. "The ROM contains 16 arctangent values, the arctans of 2^{-n}. It also contains 14 log values, the base-2 logs of $(1+2^{-n})$. These may seem like unusual values, but they are used in an efficient algorithm called CORDIC, which was invented in 1958."
54. "Getting started with the CORDIC accelerator using STM32CubeG4 MCU Package" (https://www.st.com/resource/en/application_note/dm00614795-getting-started-with-the-cordic-accelerator-using-stm32cubeg4-mcu-package-stmicroelectronics.pdf) (PDF). STMircoelectronics. Retrieved 2021-01-01.
55. "CMSIS/CMSIS/DSP_Lib/Source/ControllerFunctions/arm_sin_cos_f32.c" (https://github.com/ARM-software/CMSIS/blob/master/CMSIS/DSP_Lib/Source/ControllerFunctions/arm_sin_cos_f32.c). *Github*. ARM. Retrieved 2021-01-01.
56. "Error bounds of Taylor Expansion for Sine" (https://math.stackexchange.com/questions/2464759/error-bounds-of-taylor-expansion-for-sine). *Math Stack Exchange*. Retrieved 2021-01-01.
57. Andraka, Ray (1998). "A survey of CORDIC algorithms for FPGA based computers" (http://www.andraka.com/files/crdcsrvy.pdf) (PDF). *ACM*. North Kingstown, RI, USA: Andraka Consulting Group, Inc. 0-89791-978-5/98/01. Retrieved 2016-05-08.
58. "Class Math" (https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html#scalb-double-int-). *Java Platform Standard* (8 ed.). Oracle Corporation. 2018 [1993]. Archived (https://web.archive.org/web/20180806221131/https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html) from the original on 2018-08-06. Retrieved 2018-08-06.
59. "ldexp, ldexpf, ldexpl" (http://en.cppreference.com/w/c/numeric/math/ldexp). *cppreference.com*. 2015-06-11. Archived (https://web.archive.org/web/20180806130141/https://en.cppreference.com/w/c/numeric/math/ldexp) from the original on 2018-08-06. Retrieved 2018-08-06.
60. "Section 8.3.9 Logarithmic, Exponential, and Scale". *Intel 64 and IA-32 Architectures Software Developer's Manual Volume 1: Basic Architecture* (http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-1-manual.pdf) (PDF). Intel Corporation. September 2016. pp. 8–22.

Further reading

- Parini, Joseph A. (1966-09-05). "DIVIC Gives Answer to Complex Navigation Questions". *Electronics*: 105–111. ISSN 0013-5070 (https://www.worldcat.org/issn/0013-5070). (NB. *DIVIC* stands for *DIgital Variable Increments Computer*. Some sources erroneously refer to this as by *J. M. Parini*.)
- Anderson, Stanley F.; Earle, John G.; Goldschmidt, Robert Elliott; Powers, Don M. (1965-11-01). "The IBM System/360 Model 91: Floating-Point Execution Unit" (http://home.citycable.ch/pierrefleur/Jacques-Laporte/ibmrd1101E.pdf) (PDF). *IBM Journal of Research and Development*. Riverton, New Jersey, USA (published January 1967). **11** (1): 34–53. doi:10.1147/rd.111.0034 (https://doi.org/10.1147%2Frd.111.0034). Retrieved 2016-01-02.
- Liccardo, Michael A. (September 1968). *An Interconnect Processor with Emphasis on CORDIC Mode Operation* (MSc thesis). Berkeley, CA, USA: University of California, Berkeley, Department of Electrical Engineering. OCLC 500565168 (https://www.worldcat.org/oclc/500565168).
- US patent 3576983A (https://worldwide.espacenet.com/textdoc?DB=EPODOC&IDX=US3576983A), Cochran, David S., "Digital calculator system for computing square roots", published 1971-05-04, issued 1971-05-04, assigned to Hewlett-Packard Co. ([14] (http://www.google.com/patents/US3576983))
- Chen, Tien Chi (July 1972). "Automatic Computation of Exponentials, Logarithms, Ratios, and Square Roots" (http://home.citycable.ch/pierrefleur/Jacques-Laporte/chen.pdf) (PDF). *IBM Journal of Research and Development*. **16** (4): 380–388. doi:10.1147/rd.164.0380 (https://doi.org/10.1147%2Frd.164.0380). ISSN 0018-8646 (https://www.worldcat.org/issn/0018-8646). Retrieved 2016-01-02.
- Egbert, William E. (May 1977). "Personal Calculator Algorithms I: Square Roots" (http://www.hpl.hp.com/hpjournal/pdfs/IssuePDFs/1977-05.pdf) (PDF). *Hewlett-Packard Journal*. Palo Alto, California, USA: Hewlett-Packard. **28** (9): 22–24. Retrieved 2016-01-02. ([15] (http://www.hparchive.com/Journals/HPJ-1977-05.pdf))
- Egbert, William E. (June 1977). "Personal Calculator Algorithms II: Trigonometric Functions" (http://www.hpl.hp.com/hpjournal/pdfs/IssuePDFs/1977-06.pdf) (PDF). *Hewlett-Packard Journal*. Palo Alto, California, USA: Hewlett-Packard. **28** (10): 17–20. Retrieved 2016-01-02. ([16] (http://www.hparchive.com/Journals/HPJ-1977-06.pdf))
- Egbert, William E. (November 1977). "Personal Calculator Algorithms III: Inverse Trigonometric Functions" (http://www.hpl.hp.com/hpjournal/pdfs/IssuePDFs/1977-11.pdf) (PDF). *Hewlett-Packard Journal*. Palo Alto, California, USA: Hewlett-Packard. **29** (3): 22–23. Retrieved 2016-01-02. ([17] (http://www.hparchive.com/Journals/HPJ-1977-11.pdf))
- Egbert, William E. (April 1978). "Personal Calculator Algorithms IV: Logarithmic Functions" (http://www.hpl.hp.com/hpjournal/pdfs/IssuePDFs/1978-04.pdf) (PDF). *Hewlett-Packard Journal*. Palo Alto, California, USA: Hewlett-Packard. **29** (8): 29–32. Retrieved 2016-01-02. ([18] (http://www.hparchive.com/Journals/HPJ-1978-04.pdf))
- Senzig, Don (1975). "Calculator Algorithms". *IEEE Compcon Reader Digest*. IEEE: 139–141. IEEE Catalog No. 75 CH 0920-9C.
- Baykov, Vladimir D. (1972), Вопросы исследования вычисления элементарных функций по методу «цифра за цифрой» (http://baykov.de/cordic1972.htm) [*Problems of elementary functions evaluation based on digit by digit (CORDIC) technique*] (PhD thesis) (in Russian), Leningrad State University of Electrical Engineering
- Baykov, Vladimir D.; Smolov, Vladimir B. (1975). *Apparaturnaja realizatsija elementarnikh funktsij v CVM* Аппаратурная реализация элементарных функций в ЦВМ (http://baykov.de/cordic1975.htm) [*Hardware implementation of elementary functions in computers*] (in Russian). Leningrad State University. Archived (https://web.archive.org/web/20190302110929/http://baykov.de/Cordic1975.htm) from the original on 2019-03-02. Retrieved 2019-03-02.
- Baykov, Vladimir D.; Seljutin, S. A. (1982). Вычисление элементарных функций в ЭКВМ [*Elementary functions evaluation in microcalculators*] (in Russian). Moscow: Radio i svjaz (Радио и связь).

- Baykov, Vladimir D.; Smolov, Vladimir B. (1985). Специализированные процессоры: итерационные алгоритмы и структуры (http://baykov.de/cordic1985.htm) [*Special-purpose processors: iterative algorithms and structures*] (in Russian). Moscow: Radio i svjaz (Радио и связь).
- Coppens, Thomas, ed. (January 1980). "CORDIC constants in TI 58/59 ROM". *Texas Instruments Software Exchange Newsletter*. Kapellen, Belgium: TISOFT. **2** (2).
- Coppens, Thomas, ed. (April–June 1980). "Natural logarithm computation scheme / e^x computing scheme / 1/x computing scheme". *Texas Instruments Software Exchange Newsletter*. Kapellen, Belgium: TISOFT. **2** (3). (about CORDIC in TI-58/TI-59)
- TI Graphic Products Team (1995) [1993]. "Transcendental function algorithms" (https://perso.uclouvain.be/alphonse.magnus/num1a/cordic.txt). Dallas, Texas, USA: Texas Instruments, Consumer Products. Archived (https://web.archive.org/web/20160317004110/https://perso.uclouvain.be/alphonse.magnus/num1a/cordic.txt) from the original on 2016-03-17. Retrieved 2019-03-02.
- Jorke, Günter; Lampe, Bernhard; Wengel, Norbert (1989). *Arithmetische Algorithmen der Mikrorechentechnik* (https://books.google.com/books?id=DqYWAQAAMAAJ) (in German) (1 ed.). Berlin, Germany: VEB Verlag Technik. pp. 219, 261, 271–296. ISBN 3341005153. EAN 9783341005156 (https://eandata.com/lookup/9783341005156). MPN 5539165. License 201.370/4/89. Retrieved 2015-12-01.
- M. Zechmeister *Solving Kepler's equation with CORDIC double iterations* Institut für Astrophysik, Georg-August-Universität, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany, Preprint 10 August 2020, p. 1-10. (http://arxiv.org/pdf/2008.02894.pdf)
- Frerking, Marvin E. (1994). *Digital Signal Processing in Communication Systems* (1 ed.).
- Kantabutra, Vitit (1996). "On hardware for computing exponential and trigonometric functions". *IEEE Transactions on Computers*. **45** (3): 328–339. doi:10.1109/12.485571 (https://doi.org/10.1109%2F12.485571).
- Johansson, Kenny (2008). "6.5 Sine and Cosine Functions". *Low Power and Low Complexity Shift-and-Add Based Computations* (https://www.diva-portal.org/smash/get/diva2:1733/FULLTEXT02.pdf) (PDF) (Dissertation thesis). Linköping Studies in Science and Technology (1 ed.). Linköping, Sweden: Department of Electrical Engineering, Linköping University. pp. 244–250. ISBN 978-91-7393-836-5. ISSN 0345-7524 (https://www.worldcat.org/issn/0345-7524). No. 1201. Archived (https://web.archive.org/web/20170813200504/http://www.diva-portal.org/smash/get/diva2:1733/FULLTEXT02.pdf) (PDF) from the original on 2017-08-13. Retrieved 2021-08-23. (x+268 pages)
- Banerjee, Ayan (2001). "FPGA realization of a CORDIC based FFT processor for biomedical signal processing". *Microprocessors and Microsystems*. Kharagpur, West Bengal, India. **25** (3): 131–142. doi:10.1016/S0141-9331(01)00106-5 (https://doi.org/10.1016%2FS0141-9331%2801%2900106-5).
- Kahan, William Morton (2002-05-20). "Pseudo-Division Algorithms for Floating-Point Logarithms and Exponentials" (https://web.archive.org/web/20151225080205/http://cims.nyu.edu/%7Edbindel/class/cs279/logexp.pdf) (PDF). Berkeley, CA, USA: University of California. Archived from the original (http://www.cims.nyu.edu/~dbindel/class/cs279/logexp.pdf) (PDF) on 2015-12-25. Retrieved 2016-01-15.
- Cockrum, Chris K. (Fall 2008). "Implementation of a CORDIC Algorithm in a Digital Down-Converter" (http://cockrum.net/Cockrum_Fall_2008_Final_Paper.pdf) (PDF).
- Lakshmi, Boppana; Dhar, Anindya Sundar (2009-10-06). "CORDIC Architectures: A Survey" (https://doi.org/10.1155%2F2010%2F794891). *VLSI Design*. Kharagpur, West Bengal, India: Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology (published 2010-10-10). **2010**: 1–19. doi:10.1155/2010/794891 (https://doi.org/10.1155%2F2010%2F794891). 794891.
- Savard, John J. G. (2018) [2006]. "Advanced Arithmetic Techniques" (http://www.quadibloc.com/comp/cp0202.htm). *quadibloc*. Archived (https://web.archive.org/web/20180703001722/http://www.

quadibloc.com/comp/cp0202.htm) from the original on 2018-07-03. Retrieved 2018-07-16.

External links

- Wang, Shaoyun (July 2011), *CORDIC Bibliography Site* (http://cordic-bibliography.blogspot.com/2011/07/cordic-bibliography-site-revive.html)
- Soft CORDIC IP (verilog HDL code) (https://github.com/srohit0/CORDIC)
- *CORDIC Bibliography Site* (https://web.archive.org/web/20001017173921/http://devil.ece.utexas.edu/)
- BASIC Stamp, CORDIC math implementation (http://www.emesystems.com/BS2mathC.htm)
- CORDIC implementation in verilog (http://srohit.googlepages.com)
- CORDIC Vectoring with Arbitrary Target Value (http://portal.acm.org/citation.cfm?id=626526.627179)
- PicBasic Pro, Pic18 CORDIC math implementation (http://www.picbasic.co.uk/forum/showthread.php?p=70269#post70269)
- Python CORDIC implementation (http://code.activestate.com/recipes/576792)
- Simple C code for fixed-point CORDIC (http://www.dcs.gla.ac.uk/~jhw/cordic/)
- Tutorial and MATLAB Implementation – Using CORDIC to Estimate Phase of a Complex Number (http://luminouslogic.com/dsp-simple-phase-estimation-approximation-cordic-matlab.htm)
- Descriptions of hardware CORDICs in Arx with testbenches in C++ and VHDL (http://bibix.nl/index.php?menu1=arx_ip)
- An Introduction to the CORDIC algorithm (https://www.allaboutcircuits.com/technical-articles/an-introduction-to-the-cordic-algorithm/)
- Implementation of the CORDIC Algorithm in a Digital Down-Converter (https://cockrum.net/files/Cockrum_Fall_2008_Final_Paper.pdf)
- 50-th Anniversary of the CORDIC Algorithm (https://groups.google.com/forum/#!msg/comp.arch.fpga/NxZxkmUoE54/zp8MvF_uVf0J)
- Implementation of the CORDIC Algorithm: fixed point C code for trigonometric and hyperbolic functions (https://www.st.com/content/ccc/resource/technical/document/design_tip/group0/9c/20/c6/67/50/10/4e/9d/DM00441302/files/DM00441302.pdf/jcr:content/translations/en.DM00441302.pdf), C code for test and performance verification (https://www.st.com/content/ccc/resource/technical/document/design_tip/group0/ec/b8/82/cc/a0/e5/49/0d/DM00446487/files/DM00446487.pdf/jcr:content/translations/en.DM00446487.pdf)