A unified algorithm for elementary functions

by J. S. WALTHER

Hewlett-Packard Company
Palo Alto, California

SUMMARY

This paper describes a single unified algorithm for the calculation of elementary functions including multiplication, division, sin, cos, tan, arctan, sinh, cosh, tanh, arctanh, ln, exp and square-root. The basis for the algorithm is coordinate rotation in a linear, circular, or hyperbolic coordinate system depending on which function is to be calculated. The only operations required are shifting, adding, subtracting and the recall of prestored constants. The limited domain of convergence of the algorithm is calculated, leading to a discussion of the modifications required to extend the domain for floating point calculations.

A hardware floating point processor using the algorithm was built at Hewlett-Packard Laboratories. The block diagram of the processor, the microprogram control used for the algorithm, and measures of actual performance are shown.

INTRODUCTION

The use of coordinate rotation to calculate elementary functions is not new. In 1956 Volder developed a class of algorithms for the calculation of trigonometric and hyperbolic functions, including exponential and logarithm. In 1959 he described a COordinate Rotation DIgital Computer (CORDIC) for the calculation of trigonometric functions, multiplication, division, and conversion between binary and mixed radix number systems. Daggett in 1959 discussed the use of the CORDIC for decimal-binary conversions. In 1968 Liccardo did a master's thesis on the class of CORDIC algorithms.

It is not generally realized that many of these algorithms can be merged into one unified algorithm.

COORDINATE SYSTEMS

Let us consider coordinate systems parameterized by m in which the radius R and angle A of the vector $P=(x, y)$ shown in Figure 1 are defined as

$$R=[x^2+my^2]^{1/2}$$

$$A=m^{-1/2}\tan^{-1}[m^{1/2}y/x]$$

It can be shown that R is the distance from the origin to the intersection of the curve of constant radius with the x axis, while A is twice the area enclosed by the vector, the x axis, and the curve of constant radius, divided by the radius squared. The curves of constant radius for the circular $(m=1)$, linear $(m=0)$, and hyperbolic $(m=-1)$ coordinate systems are shown in Figure 1.

ITERATION EQUATIONS

Let a new vector $P_{i+1}=(x_{i+1}, y_{i+1})$ be obtained from $P_i=(x_i, y_i)$ according to

$$x_{i+1}=x_i+my_i\delta_i \quad (3)$$

$$y_{i+1}=y_i-x_i\delta_i \quad (4)$$

where m is the parameter for the coordinate system, and δ_i is an arbitrary value. The angle and radius of the new vector in terms of the old are given by

$$A_{i+1}=A_i-\alpha_i \quad (5)$$

$$R_{i+1}=R_i{}^*K_i \quad (6)$$

where

$$\alpha_i=m^{-1/2}\tan^{-1}[m^{1/2}\delta_i] \quad (7)$$

$$K_i=[1+m\delta_i{}^2]^{1/2} \quad (8)$$

The angle and radius are modified by quantities which are independent of the coordinate values. Table I gives the equations for α_i and K_i after applying identities $A2$ and $A5$ from the appendix.

For n iterations we find

$$A_n=A_0-\alpha \quad (9)$$

$$R_n=R_0{}^*K \quad (10)$$

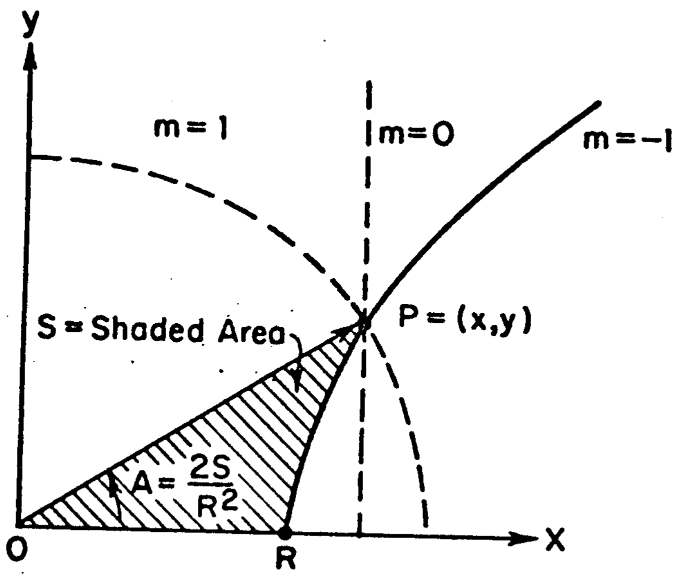

Figure 1—Angle A and Radius R of the vector $P=(x, y)$

where

$$\alpha=\sum_{i=0}^{n-1}\alpha_i \tag{11}$$

$$K=\prod_{i=0}^{n-1}K_i \tag{12}$$

The total change in angle is just the sum of the incremental changes while the total change in radius is the product of the incremental changes.

If a third variable z is provided for the accumulation of the angle variations

$$z_{i+1}=z_i+\alpha_i \tag{13}$$

and the set of difference equations (3), (4), and (13) is solved for n iterations, we find,

$$x_n=K\{x_0\cos(\alpha m^{1/2})+y_0 m^{1/2}\sin(\alpha m^{1/2})\} \tag{14}$$

$$y_n=K\{y_0\cos(\alpha m^{1/2})-x_0 m^{-1/2}\sin(\alpha m^{1/2})\} \tag{15}$$

$$z_n=z_0+\alpha \tag{16}$$

where α and K are as in equations (11) and (12).

TABLE I—Angles and Radius Factors

Coordinate System m	Angle α_i	Radius Factor K_i
1	$\tan^{-1}\delta_i$	$(1+\delta_i^2)^{1/2}$
0	δ_i	1
−1	$\tanh^{-1}\delta_i$	$(1-\delta_i^2)^{1/2}$

These relations are summarized in Figure 2 for $m=1$, $m=0$ and $m=-1$ for the following special cases.

1. A is forced to zero: $y_n=0$.
2. z is forced to zero: $z_n=0$.

The initial values x_0, y_0, z_0 are shown on the left of each block in the figure while the final values x_n, y_n, z_n are shown on the right. The identities given in the appendix were used to simplify these results. By the proper choice of the initial values the functions $x\,z$, y/x, $\sin z$, $\cos z$, $\tan^{-1} y$, $\sinh z$, $\cosh z$, and $\tanh^{-1} y$ may be obtained. In addition the following functions may be generated.

$$\tan z=\sin z/\cos z \tag{17}$$

$$\tanh z=\sinh z/\cosh z \tag{18}$$

$$\exp z=\sinh z+\cosh z \tag{19}$$

$$\ln w=2\tanh^{-1}[y/x] \text{ where } x=w+1 \text{ and } y=w-1 \tag{20}$$

$$(w)^{1/2}=(x^2-y^2)^{1/2} \text{ where } x=w+\tfrac{1}{4} \text{ and } y=w-\tfrac{1}{4} \tag{21}$$

CONVERGENCE SCHEME

The angle A of the vector P may be forced to zero by a converging sequence of rotations α_i which at each step brings the vector closer to the positive x axis.

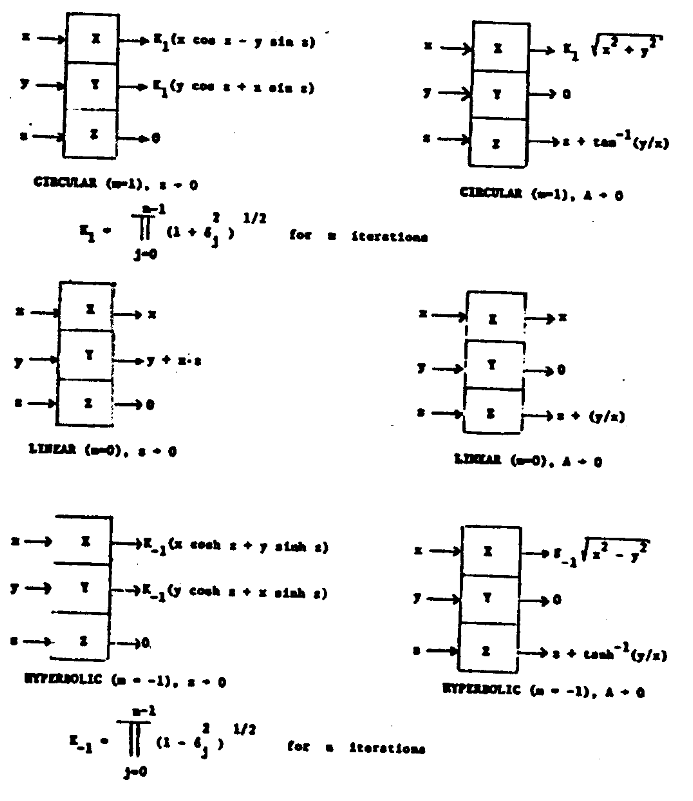

Figure 2—Input-output functions for CORDIC modes

The magnitude of each element of the sequence may be predetermined, but the direction of rotation must be determined at each step such that

$$|A_{i+1}| = ||A_i| - \alpha_i| \tag{22}$$

The sum of the remaining rotations must at each step be sufficient to bring the angle to at least within α_{n-1} of zero, even in the extreme case where $A_i = 0$, $|A_{i+1}| = \alpha_i$. Thus,

$$\alpha_i - \sum_{j=i+1}^{n-1} \alpha_j < \alpha_{n-1} \tag{23}$$

The domain of convergence is limited by the sum of the rotations.

$$|A_0| - \sum_{j=0}^{n-1} \alpha_j < \alpha_{n-1} \tag{24}$$

$$\max |A_0| = \alpha_{n-1} + \sum_{j=0}^{n-1} \alpha_j \tag{25}$$

To show that A converges to within α_{n-1} of zero within n steps we first prove the following theorem.

Theorem

$$|A_i| < \alpha_{n-1} + \sum_{j=i}^{n-1} \alpha_j \tag{26}$$

holds for $i \geq 0$.

Proof

We proceed by induction on i. The hypothesis (26) holds for $i=0$ by (24). We now show that if the hypothesis is true for i then it is also true for $i+1$. Subtracting α_i from (26) and applying (23) at the left side yields

$$-\left[\alpha_{n-1} + \sum_{j=i+1}^{n-1} \alpha_j\right] < -\alpha_i < |A_i|$$

$$-\alpha_i < \left[\alpha_{n-1} + \sum_{j=i+1}^{n-1} \alpha_j\right] \tag{27}$$

Application of (22) then yields

$$|A_{i+1}| < \alpha_{n-1} + \sum_{j=i+1}^{n-1} \alpha_j \tag{28}$$

as was to be shown. Therefore, by induction, the hypothesis holds for all $i \geq 0$.
In particular, the theorem is true for $i=n$ so that

$$|A_n| < \alpha_{n-1}. \tag{29}$$

The same scheme may be used to force the angle in z to zero. The proof of convergence proceeds exactly as before except that A is replaced by z in equations (22) through (29). By equation (25) z has the same domain of convergence as A.

$$\max |z_0| = \max |A_0|. \tag{30}$$

Note that since K is a function of δ_i^2, where $\delta_i = m^{-1/2} \tan[m^{1/2}\alpha_i]$, K is independent of the sequence of signs chosen for the α_i. Thus, for a fixed sequence of α_i magnitudes the constant $1/K$ may be used as an initial value to counteract the factor K present in the final values.

TABLE II—Shift Sequences for a binary code

radix ρ	coordinate system m	shift sequence $F_{m,i}; i \geq 0$	domain of convergence $\max\|A_0\|$	radius factor K
2	1	0, 1, 2, 3, 4, i, ...	~1.74	~1.65
2	0	1, 2, 3, 4, 5, $i+1$, ...	1.0	1.0
2	−1	1, 2, 3, 4, 4, 5, ...*	~1.13	~0.80

* for $m = -1$ the following integers are repeated: {4, 13, 40, 121, ..., k, $3k+1$, ...}

USE OF SHIFTERS

The practical use of the algorithm is based on the use of shifters to effect the multiplication by δ_i. If ρ is the radix of the number system and F_i is an array of integers, where $i \geq 0$, then a multiplication of x by

$$\delta_i = \rho^{-F_i} \tag{31}$$

is simply a shift of x by F_i places to the right. The integers F_i must be chosen such that the angles

$$\alpha_{m,F_i}| = m^{-1/2} \tan^{-1}(m^{1/2}\rho^{-F_i}) \tag{32}$$

satisfy the convergence criterion (23). The domain of convergence is then given by (25).

Table II shows some F sequences, convergence domains, and radius factors for a binary code.

The hyperbolic mode ($m=-1$) is somewhat complicated by the fact that for $\alpha_i = \tanh^{-1}(2^{-i})$ the convergence criterion (23) is not satisfied. However, it can be shown that

$$\alpha_i - \left(\sum_{j=i+1}^{n-1} \alpha_j\right) - \alpha_{3i+1} < \alpha_{n-1} \tag{33}$$

and that therefore if the integers {4, 13, 40, 121, ..., k, $3k+1$, ...} in the F_i sequence are repeated then (23) becomes true.

4

TABLE III—Prescaling Identities

Identity	Domain	Domain of Convergence
$\sin\left(Q\frac{\pi}{2}+D\right)=\begin{cases}\sin D \text{ if } Q \bmod 4=0\\ \cos D \text{ if } Q \bmod 4=1\\ -\sin D \text{ if } Q \bmod 4=2\\ -\cos D \text{ if } Q \bmod 4=3\end{cases}$	$\|D\|<\frac{\pi}{2}=1.57$	1.74
$\cos\left(Q\frac{\pi}{2}+D\right)=\begin{cases}\cos D \text{ if } Q \bmod 4=0\\ -\sin D \text{ if } Q \bmod 4=1\\ -\cos D \text{ if } Q \bmod 4=2\\ \sin D \text{ if } Q \bmod 4=3\end{cases}$	$\|D\|<\frac{\pi}{2}=1.57$	1.74
$\tan\left(Q\frac{\pi}{2}+D\right)=\sin\left(Q\frac{\pi}{2}+D\right)\Big/\cos\left(Q\frac{\pi}{2}+D\right)$	$\|D\|<\frac{\pi}{2}=1.57$	1.74
$\tan^{-1}\left(\frac{1}{y}\right)=\frac{\pi}{2}-\tan^{-1}(y)$	$\|y\|<1.0$	∞
$\sinh(Q\log_e 2+D)=\frac{2^Q}{2}[\cosh D+\sinh D-2^{-2Q}(\cosh D-\sinh D)]$	$\|D\|<\log_e 2=0.69$	1.13
$\cosh(Q\log_e 2+D)=\frac{2^Q}{2}[\cosh D+\sinh D+2^{-2Q}(\cosh D-\sinh D)]$	$\|D\|<\log_e 2=0.69$	1.13
$\tanh(Q\log_e 2+D)=\sinh(Q\log_e 2+D)/\cosh(Q\log_e 2+D)$	$\|D\|<\log_e 2=0.69$	1.13
$\tanh^{-1}(1-M2^{-E})=\tanh^{-1}(T)+(E/2)\log_e 2$	$0.17<T<0.75$	$(-0.81, 0.81)$
where $T=(2-M-M2^{-E})/(2+M-M2^{-E})$	for $0.5\leq M<1, E\geq 1$	
$\exp(Q\log_e 2+D)=2^Q(\cosh D+\sinh D)$	$\|D\|<\log_e 2=0.69$	1.13
$\log_e(M2^E)=\log_e M+E\log_e 2$	$0.5\leq M<1.0$	$(0.10, 9.58)$
$\text{sqrt}(M2^E)=\begin{cases}2^{E/2}\,\text{sqrt}(M) & \text{if } E \bmod 2=0\\ 2^{(E+1)/2}\,\text{sqrt}(M/2) & \text{if } E \bmod 2=1\end{cases}$	$\begin{cases}0.5\leq M<1.0\\ 0.25\leq M/2<0.5\end{cases}$	$(0.03, 2.42)$
$(M_x 2^{E_x})(M_y 2^{E_y})=(M_x M_y)2^{E_x+E_y}$	$0.5\leq\|M_x\|<1.0$	$(-1.0, 1.0)$
$(M_y 2^{E_y})/(M_x 2^{E_x})=(M_y/2M_x)2^{E_y-E_x+1}$	$0.25\leq\|M_y/2M_x\|<1.0$	$(-1.0, 1.0)$

EXTENDING THE DOMAIN

The limited domain imposed by the convergence criterion (25) may be extended by means of the prescaling identities shown in Table III. For example, to calculate the sine of a large argument, we first divide the argument by $\pi/2$ obtaining a quotient Q and a remainder D where $|D|<\pi/2$. The table shows that only sin D or cos D need be calculated and that $\pi/2$ is within the domain of convergence. Note that the sine and cosine can be generated simultaneously by the CORDIC algorithm and that the answer may then be chosen as plus or minus one of these according to Q mod 4. As a second example, to calculate the logarithm of a large argument we first shift the argument's binary point E places until it is just to the left of the most significant non-zero bit. The fraction M then satisfies $0.5\leq M<1.0$ and as shown in the table therefore falls within the domain of convergence. The answer is calculated as $\log_e M+E\log_e 2$.

ACCURACY

The accuracy at the nth step is determined in theory by the size of the last of the converging sequence of rotations α_i, and for large n is approximately equal in digits to F_{n-1}. The accuracy in digits may conveniently be made equal to L, the length of storage used for each variable, by choosing n such that $F_{n-1}=L$.

In practice the accuracy is limited by the finite length of storage. The truncation of input arguments performed to make them fit within the storage length gives rise to unavoidable error, the size of which depends on the sensitivity of the calculated function to small changes in the input argument. In a binary code, the truncation of intermediate results after each of L iterations gives rise to a total of at most $\log_2 L$ bits of error. This latter error can be rendered harmless by using $L+\log_2 L$ bits for the storage of intermediate results.

In a normalized floating point number system it is desirable that all L bits of the result be accurate, independent of the absolute size of the argument. To accomplish this for very small arguments it is necessary to keep each storage register in a normalized form; i.e., in a form where there are no leading zeros. It is possible to do this by transforming the iteration equations (3), (4), (13) to a normalized form according to the following substitutions.

$$x \text{ becomes } x' \tag{34}$$

$$y \text{ becomes } y'\, 2^{-E} \tag{35}$$

$$z \text{ becomes } z'\, 2^{-E} \tag{36}$$

$$\alpha_F \text{ becomes } \alpha_F'\, 2^{-F} \tag{37}$$

where E, a positive integer, is chosen such that the initial argument, placed into either the y or z register, is normalized.

The result of the substitutions is

$$x' \leftarrow x' + my'2^{-(F+E)} \tag{38}$$

$$y' \leftarrow y' - x'2^{-(F-E)} \tag{39}$$

$$z' \leftarrow z' + \alpha_F'2^{-(F-E)} \tag{40}$$

For simplicity the subscripts i and $i+1$ have been dropped. Instead, α has been expressed as a function of F as in equation (32), and the replacement operator ($\leftarrow$) has been used. i may be initialized to a value such that $F_i=E$:

$$i_{\text{initial}} \leftarrow \{i \mid F_i = E\}, \tag{41}$$

and n may be chosen such that L significant bits are obtained:

$$n \leftarrow \{n \mid F_{n-1} - E = L\}. \tag{42}$$

Note that $n - i_{\text{initial}} \approx L$ and that therefore providing $L+\log_2 L$ bits for the storage of intermediate results is still adequate.

The radius factor K is now a function of $i = i_{\text{initial}}$ as well as m.

$$K_{m,i} = \prod_{j=i}^{n-1} (1+m2^{-2F_j})^{1/2} \tag{43}$$

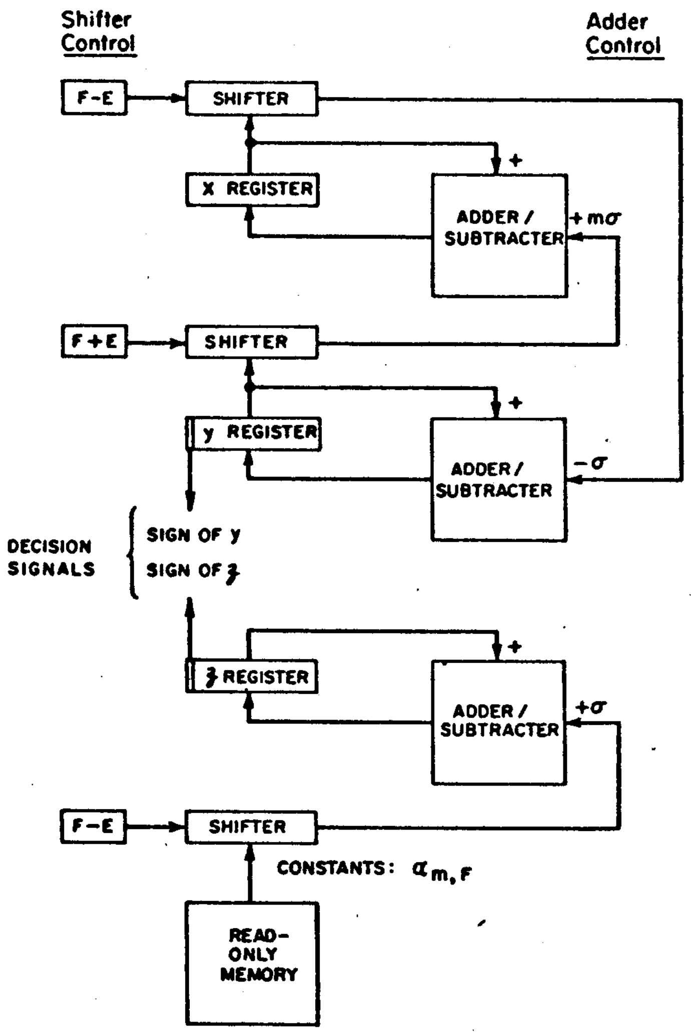

Figure 3—Hardware block diagram

Fortunately, not all the reciprocal constants $1/K_{m,i}$ need to be stored since for large values of i

$$\frac{1}{K_{m,i}} \approx 1 - m(2/3)2^{-2i}, \tag{44}$$

and therefore all the constants having $i > L/2$ are identical to within L significant bits. Therefore, only $L/2$ constants need to be stored for $m=+1$ and also for $m=-1$. For $m=0$ no constants need to be stored since $K_{0,i}=1$ for $i \geq 1$.

A similar savings in storage can be made for the angle constants $\alpha_{m,F}$ since for large values of F

$$\alpha'_{m,F} \equiv \alpha_{m,F}\, 2^F \approx 1 - m(1/3)2^{-2F}, \tag{45}$$

and thus, as for the K constants, only $L/2$ constants need to be stored for $m=+1$ and also for $m=-1$. For $m=0$ no constants need to be stored since $\alpha'_{0,F}=1$ for $F \geq 1$.

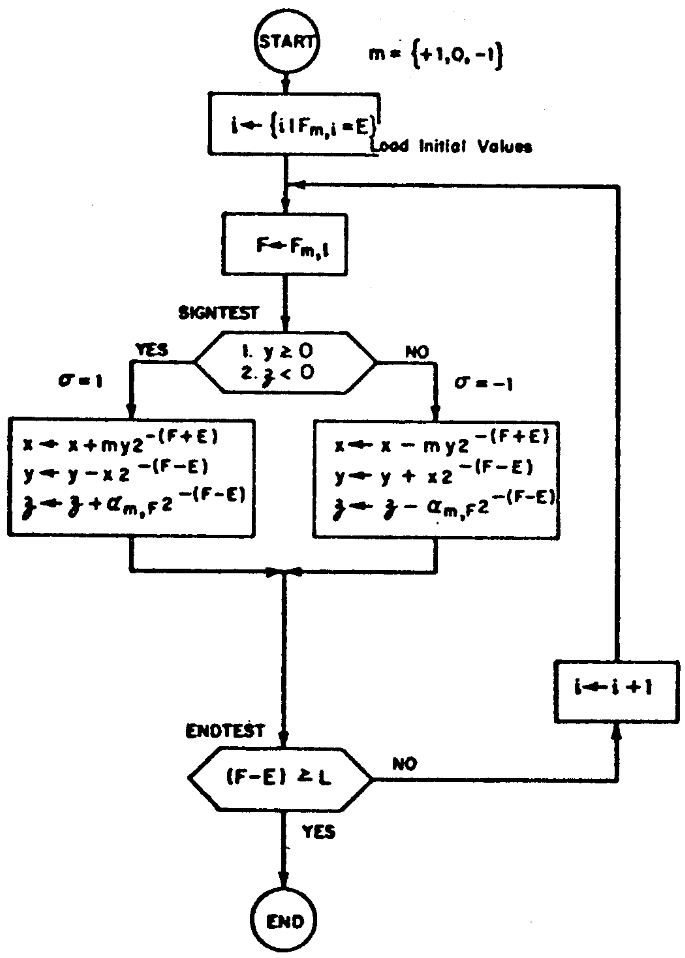

Figure 4—Flowchart of the microprogram control

HARDWARE IMPLEMENTATION

A hardware floating point processor based on the CORDIC algorithm has been built at Hewlett-Packard Laboratories. Figure 3 shows a block diagram of the processor which consists of three identical arithmetic units operated in parallel. Each arithmetic unit contains a 64-bit register, an 8-bit parallel adder/subtracter, and an 8-out-of-48 multiplex shifter. The assembly of arithmetic units is controlled by a microprogram stored in a read-only memory (ROM), which also contains the angle and radius-correction constants. The ROM contains 512 words of 48 bits each and operates on a cycle time of 200 nanoseconds.

The processor accepts three data types: 48-bit floating point, 32-bit floating point, and 32-bit integer. All the functions are calculated to 40 bits of precision (approximately 12 decimal digits), and the accuracy is limited only by the truncation of input arguments.

The essential aspects of the microprogram used to execute the CORDIC algorithm are shown in Figure 4. The initial argument and correction constants are loaded into the three registers and m is set to one of the three values 1, 0, -1. If the initial argument is small, it is normalized and E is set to minus the binary exponent of the result, otherwise, E is set to zero. Next, i is initialized to a value such that $F_{m,i}=E$. A loop is then entered and is repeated until $F_{m,i}-E=L$. In this loop the direction of rotation necessary to force either of the angles A or z to zero is chosen; the binary variable σ, used to control the three adder/subtracters, is set to either $+1$ or -1; and the iteration equations are executed.

Table IV gives a breakdown of the maximum execution times for the most important functions. The figures in the column marked "data transfers from computer" are the times for operand and operation code transfers between the processor and an HP-2116 computer.

The processor retains the result of each executed function. Thus, add, subtract, multiply and divide require only one additional operand to be supplied, and the one operand functions do not require any operand transfers. The first operand is loaded via the LOAD instruction, and the final result is retrieved via the STORE instruction.

TABLE IV—Maximum Execution Times

ROUTINE	CORDIC EXECUTION μsec	PRESCALE, NORMALIZE, MISC. μsec	DATA TRANSFERS FROM COMPUTER μsec	TOTAL μsec
LOAD	0	5	25	30
STORE	0	0	15	15
ADD	0	15	25	40
SUBTRACT	0	25	25	50
MULTIPLY	60	15	25	100
DIVIDE	60	15	25	100
SIN	70	85	5	160
COS	70	85	5	160
TAN	130	85	5	220
ATAN	70	15	5	90
SINH	70	55	5	130
COSH	70	55	5	130
TANH	130	55	5	190
ATANH	70	45	5	120
EXPONENTIAL	70	55	5	130
LOGARITHM	70	45	5	120
SQUARE-ROOT	70	25	5	100

7

CONCLUSION

The unified CORDIC algorithm is attractive for the calculation of elementary functions because of its simplicity, its accuracy, and its capability for high speed execution via parallel processing. Its applications include desktop calculators, as in the HP-9100 series; air navigation computers, as described in Volder's original work; and floating point processors, as illustrated in this paper.

ACKNOWLEDGMENTS

The author wishes to thank the many people at Hewlett-Packard Laboratories and Cupertino Division for their contributions and support.

REFERENCES

1 D H DAGGETT
Decimal-binary conversion in Cordic
IRE Transactions on Electronic Computers Vol EC-8 No 3 pp 335-339 September 1959

2 M A LICCARDO
An interconnect processor with emphasis on Cordic mode operation
Masters Thesis EE Dept University of California at Berkeley September 1968

3 J E VOLDER
Binary computation algorithms for coordinate rotation and function generation
Convair Report IAR-1 148 Aeroelectronics Group June 1956

4 J E VOLDER
The Cordic trigonometric computing technique
IRE Transactions on Electronic Computers Vol EC-8 No 3 pp 330-334 September 1959

APPENDIX

Mathematical identities

Let $i = (-1)^{1/2}$

$$z \equiv \lim_{m \to 0} m^{-1/2} \sin(zm^{1/2}) \tag{A1}$$

$$z \equiv \lim_{m \to 0} m^{-1/2} \tan^{-1}(zm^{1/2}) \tag{A2}$$

$$\sinh z \equiv -i \sin(iz) \tag{A3}$$

$$\cosh z \equiv \cos(iz) \tag{A4}$$

$$\tanh^{-1} z \equiv -i \tan^{-1}(iz) \tag{A5}$$