North-Holland
Microprocessing and Microprogramming 11 (1983) 221–225

Implementation of Transcendental Functions on a Numerics Processor

Rafi Nave
Intel Israel Ltd., M.T.M. Scientific Industries Center, P.O. Box 1659, Haifa 31 015, Israel

The Intel 8087 is a numerical processor that delivers unprecendented functionality and performance for a single VLSI chip. One of its key features is the performance of the transcendental (logarithmic and trigonometric) instructions that compute with very high precision and fast execution times.

The above was achieved by making three major engineering decisions:

(a) The selection of an instruction set which is minimal enough to be implemented in firmware, and still covers the primitives for all desired transcendentals;

(b) The selection of the CORDIC algorithms for implementation of those primitives;

(c) The selection of minimal hardware extensions to the architecture that enable efficient execution.

Keywords: Numerical processor, transcendental instructions, performance.

1. Introduction

The implementation of transcendental functions usually involves the selection of the instruction set, algorithms and firmware. In the case of the Intel 8087 [1, 2], the constraints of code size, and existing architecture accompanied by ambitious performance and precision objectives implied the need for a unique solution.

1.1. Transcendental Instructions Objectives

The instruction set had to include the basic instructions that will enable, with minimal software effort, to calculate all logarithmic, exponential, hyperbolic and trigonometric functions.

A very high precision, as measured by a relative error smaller than 3.2^{-64} for results that have 64 bits of precision should be obtained.

The execution time of those basic instructions had to be less than 200 micro seconds.

1.2. Design Constraints

The micro code size was limited to about 500 lines for the whole transcendentals set.

The floating point data path and control were optimized for the basic instruction set (ADD, SUBTRACT, MULTIPLY, etc...). Only minimal, essential, extensions could be tolerated.

Only minor modifications to the existing micro instructions were permitted.

1.3. Implementation Strategy

1.3.1. Selection of Instruction Set

It was determined that 4 instructions will do the job: LOGARITHM, EXPONENT, TANGENT and ARCTANGENT. It can easily be shown that all the desired transcendental functions can be computed with this set. The domain of trigonometric operands was selected so that the application of the REMAINDER operation can easily reduce any operand to the proper range; $(0, \pi/4)$.

1.3.2. Selection of Algorithm

The computation of a transcendental function can be in one of several techniques: Polynomial (like TAYLOR series), continued fractions, rational approximations, or coordinate rotation – CORDIC [3–7]. The domain of operands, performance requirements, and given architecture led us to the conclusion that CORDIC will give the most effective results. This will be illustrated throughout this paper by the example of the TANGENT function.

2. The CORDIC Technique

2.1. The Motivation for CORDIC

The polynomial or rational approximations usually incur a computation error which is proportional to some power of the operand. For example, if one takes 5 elements of the series:

$$Y=\ln(1-X)$$

$$=X+\frac{X^2}{2}+\frac{X^3}{3}+\frac{X^4}{4}+\frac{X^5}{5}+\frac{X^6}{6}+\ldots, \qquad (1)$$

the error, for small enough X's is roughly: $X^6/6$. If we could evaluate the function Y by using an arbitrarily small X, we could obtain the desirable small enough computation error.

The CORDIC algorithms provide such a controlled approach to reversible arbitrary reduction of the operand's size, evaluation of the function, and a build up of the result for the original operand from the result computed for the reduced operand.

2.2. The TANGENT Example

The coordinates rotation is illustrated in Fig. 1. The coordinates system is rotated by angle β and the following relations hold:

$$\mathrm{tg}(\alpha_i)=\frac{Y_i}{X_i},\ \mathrm{tg}(\alpha_{i+1})=\frac{Y_{i+1}}{X_{i+1}}, \qquad (2)$$

$$\alpha_i=\alpha_{i+1}+\beta,$$

$$\mathrm{tg}(\alpha_i)=\frac{\mathrm{tg}(\alpha_{i+1})+\mathrm{tg}(\beta)}{1-\mathrm{tg}(\alpha_{i+1})\cdot\mathrm{tg}(\beta)}. \qquad (3)$$

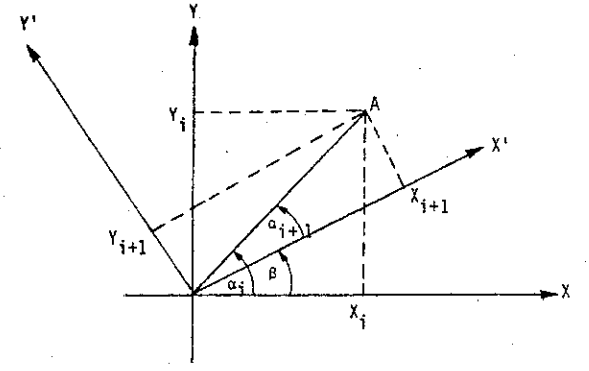

Fig. 1. Rotation of Coordinates.

If we substitute (2) in (3), we get:

$$\frac{Y_i}{X_i}=\frac{Y_{i+1}+\mathrm{tg}(\beta)\cdot X_{i+1}}{X_{i+1}-Y_{i+1}\cdot\mathrm{tg}(\beta)}. \qquad (4)$$

If we select a very "special" β, such that $\mathrm{tg}(\beta)=2^{-i}$ [i.e., $\beta=\mathrm{Arctg}(2^{-i})$], we get:

$$\frac{Y_i}{X_i}=\frac{Y_{i+1}+2^{-i}\cdot X_{i+1}}{X_{i+1}-2^{-i}\cdot Y_{i+1}}. \qquad (5)$$

In other words, if we reduce the operand's size from α_i to α_{i+1} and compute $\mathrm{Tg}(\alpha_{i+1})=Y_{i+1}/X_{i+1}$, we can compute $\mathrm{Tg}(\alpha_i)=Y_i/X_i$ through the relations:

$$\begin{aligned} Y_i&=Y_{i+1}+2^{-i}\cdot X_{i+1},\\ X_i&=X_{i+1}-2^{-i}\cdot Y_{i+1}. \end{aligned} \qquad (6)$$

It is important to note that a multiplication by 2^{-i} is simply a shift right by i locations. This coordinate rotation can be applied as many times as one wishes. It can be proven that the representation:

$$\alpha=\sum_{i=1}^{N} q_i\cdot\beta_i+\alpha_{N+1} \text{ where } \alpha<\pi/4 \qquad (7)$$

$$\beta_i=\mathrm{arctg}(2^{-i}),\quad q_i=\{{}^{0}_{1}\}$$

is unique and $\alpha_{N+1}<2^{-N}$.

Similar techniques (although less illustrative) can be developed for other functions that converge for small X's to 0 such as:

$$Y=\ln(1-X),\quad Y=2^X-1,$$

or $Y=\mathrm{arctg}(X)$.

2.3. Computational flow

All CORDIC algorithms consist of three major steps:

1. The PSEUDO DIVIDE where the operand is reduced to a desired small value, while a "record" of the reduction process is stored in the form of a PSEUDO QUOTIENT. The PSEUDO QUOTIENT's i-th bit is set to 1 for every iteration i, after which the residue is still positive, otherwise it is set to 0 and no reduction takes place.

2. The RATIONAL APPROXIMATION computing the function for the last remainder with a guaranteed precision.

3. The PSEUDO MULTIPLY where the result

for the original operand is built up iteratively, using the result of the rational approximation as a starting value, and the PSEUDO QUOTIENT as the computation control. Only rounding errors affect the accuracy of this step.

3. Implementation Example – TANGENT

3.1. Polynomial Approximation

The TANGENT function can be approximated by:

$$\mathrm{tg}(Z) = Z + \tfrac{1}{3}Z^3 + \tfrac{2}{15}Z^5 + \tfrac{7}{315}Z^7 + \ldots \quad (8)$$

If we compute only N elements of the series the error is roughly the $(N+1)$'s element. Since Z could be as big as $\frac{\pi}{4}$, the convergence of this series is relatively slow, and it may take over 20 elements to compute to our desired precision! This will yield an unacceptably high execution time, far higher than the objectives we set.

3.2. The CORDIC Implementation

The CORDIC technique described in Section 2 is applied. The TANGENT computation steps are:

(a) A PSEUDO DIVIDE step during which the operand Z is decomposed in the following way:

$$Z = \sum_{i=1}^{N} q_i \cdot \tau_i + Z_{N+1}$$

where

$$q_i = \{{}^{0}_{1}\} \text{ and } \tau_i = \mathrm{arctg}(2^{-i}) \quad (9)$$

It can be shown that $Z_{N+1} < 2^{-n}$, and that this representation is unique.

(b) The RATIONAL APPROXIMATION step computing the function:

$$\mathrm{tg}(Z_{N+1}) = \frac{3 \cdot Z_{N+1}}{3 - (Z_{N+1})^2}. \quad (10)$$

It can be shown that the relative error in this approximation is $E_r = \frac{1}{45}(Z_{N+1})^4$ and thus, if $Z_{N+1} < 2^{-16}$ we can guarantees $E_r < 2^{-65}$.

(c) The PSEUDO MULTIPLY sequence is now applied. Let $Y_{N+1} := \mathrm{tg}(Z_{N+1})$ and $X_{N+1} := 1$. In an iterative process for $i = N$ through 1, one can compute.

$$Y_i = Y_{i+1} + q_i \cdot X_{i+1} \cdot 2^{-i}$$

and

$$X_i = X_{i+1} - q_i \cdot Y_{i+1} \cdot 2^{-i}. \quad (11)$$

At each step $\mathrm{tg}(Z_i) = Y_i / X_i$ and when N times applied, the desired final result is obtained: $\mathrm{tg}(Z) = Y_1 / X_1$.

It is important to notice that this PSEUDO MULTIPLY process is as accurate as the precision of the constants τ_i, and the rounding of the machine. In our case the wide data path (67 bits) guarantees that the cumulative relative error will not exceed 2^{-63}.

3.3. Hardware Requirements

The given floating point machine already had the facilities for multiplication and division, an elaborate barrel shifter and micro code looping support, as well as enough working registers. Thus, the only additions to the hardware were:

1. Constant ROMs: for the constants $\tau_i = \mathrm{arctg}(2^{-i})$, as well as for a similar set of constants for the logarithm functions.
2. A 16-bit shift register for the PSEUDO QUOTIENT storage and testing.
3. A capability to set the shift span, and to address the constant ROM's, based on the contents of the loop counter. This is essential for the fast execution and code compactness during the PSEUDO DIVIDE and PSEUDO MULTIPLY sequences.

With the above hardware extensions, the micro code implementation managed to meet all the design objectives.

4. Numerical Results

The above algorithms were used in the implementation of the Intel 8087 Transcendental Instructions with the following results.

4.1. Precision

The precision of all instructions was better than a relative error of 3.2^{-64} for floating point numbers with 64 bits of precision.

4.2. Execution Time

The execution speed of the primitives, as well as the operands' domains are listed in Table 1.

A full support for high level elementary functions is provided by a software library utilizing the primitive transcendental functions. The performance of some of those functions for a 5MHz and 8MHz 8087 & 8086 chip set is listed in Table 2.

Table 1. Primitive Transcendental Instructions

Instruction	Function	Operands Domain	Clock Count	Execution Time (microseconds) 5MHz	8MHz
FPTAN	$Y/X = \mathrm{Tg}(Z)$	$0 \le Z \le \frac{\pi}{4}$	450	90	57
FP ATAN	$Z = \mathrm{arctg}(Y/X)$	$0 \le Y \le X < \infty$	650	130	82
FL2X	$Z = Y \cdot \log_2 X$	$-\infty < Y < +\infty$ $0 < X < +\infty$	950	190	120
FL2XP1	$Z = Y\log_2(1+X)$	$-\infty < Y < +\infty$ $\frac{\sqrt{2}}{2} - 1 < X < 1 - \frac{\sqrt{2}}{2}$	850	170	107
F2XM1	$Z = 2^X - 1$	$0 \le X \le \frac{1}{2}$	500	100	63

Table 2. Elementary Functions Library

Function	8086 + 8087 Execution Time (Microseconds) 5 MHz	8 MHz
SIN	486	304
COS	489	306
TAN	338	211
ASIN	456	285
ATAN	303	189
ACOS	476	298
TANH	542	339
COSH	528	330
SINH	535	334
EXP	451	282
Y**X	624	390
LOG	298	186
LOGIO	298	186

5. Conclusion

It has been shown that for a micro coded numerical execution machine that has wide data paths, a high precision and low execution time can be obtained by using the CORDIC technique. The careful implementation can yield compact code size and only minimal hardware extensions without compromising on either precision or speed.

Acknowledgement

I would like to thank Prof. W. Kahan for his technical guidance and outline of the theory behind the specific algorithms that were chosen. J.F. Palmer for the ongoing numerics expertise and dialogue, C. Wymore and A. Kornhauser for their support in the implementation of the various algorithms in hardware and firmware, Y. Talgam for proposing valuable improvements to this paper, and the whole Intel Israel design team for their share in the 8087 effort.

References

[1] J. Palmer, R. Nave, C. Wymore, R. Koehler and C. McMinn, Making Mainframe Mathematics Accessible to Mini Computers, Electronics, Vol. 23 (8 May 1980).

[2] R. Nave, J. Palmer, A Numeric Data Processor, Digest of Technical Papers, ISSCC (Feb. 1980), 108–109.

[3] J.E. Volder, The Cordic Trigonometric Computing Technique, IRE Trans. Electron. Computers, Vol. EC-8 (Sept. 1959), 330–334.

[4] W.H. Specker, A Class of Algorithms for LNX, EXPX, SINX, COSX, $\mathrm{TAN}^{-1}X$, and $\mathrm{COT}^{-1}X$, IEEE Trans. Electron. Computers, Vol. EC-14 (1965), 85–86.

[5] J.S. Walther, A Unified Algorithm for Elementary Functions, AFIPS 1971 Proc., Spring Joint Computer Conf. (1971), 379–385.

[6] T.C. Chen, Automatic Computation of Exponentials, Logarithms, Ratios and Square Roots, IBM J. Res. Develop., Vol. 16 (July 1972), 380–388.

[7] M. Andrews, D.E. Eggerding, A Pipelined Computer Architecture for Unified Elementary Function Evaluation, Comput. and Electron. Engineering, Vol. 5 (1978), 189–202.

Appendix

Basic Definitions

BARREL SHIFTER Hardware module that is able to shift, in one clock, a series of bits by a specified number of locations.

CORDIC "Coordinate Rotation Digital Computer" – an algorithm that computes a function by structured rotation of coordinates.

PRIMITIVE Subset of an intricate instruction. It is the kernel, hardest to implement, or performance limiting portion which can be extended to the full instruction, with simple software.

PSEUDO DIVIDE Reduction of an operand's magnitude resulting in a PSEUDO QUOTIENT, and a remainder that is known to be smaller than a certain bound.

PSEUDO MULTIPLY build up of the result using the PSEUDO QUOTIENT, and function of the remainder.

TRANSCENDENTAL FUNCTIONS Mathematical functions that cannot be represented by a ratio of 2 finite polynomials, and thus have no rational representation.

Rafi Nave was born in Tel Aviv, in 1949. He served in the Israeli Army as a communication technician from 1967 to 1970. He graduated from the Electrical Engineering Faculty at the Technion, Haifa, in 1974, and received his M.Sc degree in 1979. He joined Intel Israel, a wholly owned subsidiary of Intel Corporation, in 1974. He was responsible for the development of the INTEL 8279, he supervised the development of the INTEL 8259A and 8080A, and managed the development of the iNTEL 8087. He is currently the General Manager of Intel Israel.