Shorter Table ATA Evaluation of Elementary Functions in Single Precision

Oscar N. Bria

III−LIDI . CeTAD
Facultad de Informática . Facultad de Ingeniería
Universidad Nacional de La Plata
Argentina

ABSTRACT

In this paper a slightly modification is proposed to the original Wong and Goto's ATA method for the computation of elementary functions in IEEE 754 single precision.
The identification of a trade-off leads to the proposition of a different chunk of the mantissa that in turn brings a reduction in the length of the tables.
Results are reported for usual elementary functions based on exhaustive simulations. A mixed framework including VHDL and Matlab® is used for those simulations.

Keywords: Computer Arithmetic, Single Precision Elementary Functions, Table-Based Methods, ATA Method.

1. INTRODUCTION

In the electronic mass-market, many novel application areas are arithmetic-intensive: encryption, error checking, multimedia. Therefore, is expected that computer arithmetic will expand for years to come [6].
In the computer arithmetic field, the goal is to stamp numerical theory results into real designs. That requires an in-between step of algorithmic implementations and performance evaluations. A desirable intermediate development is a design library for the implementation of embedded processors.
VHDL is a convenient language for the description of such a library [3]. VHDL is a hardware description language intended for modelling and documenting digital systems. VHDL is a high level parallel language that allows mixed level of detail descriptions and clocked simulations [1].
The so called elementary functions (inverse, square root, exponentials, logarithms, sine, arc tangent) are the most commonly used mathematical functions. Computing then accurately, quickly and inexpensively is a major goal in computer arithmetic [5].
Algorithmic implementations for elementary functions can be characterized by

the numerical representation,

the computing latency,

and the amount of affected resources.

Normally there is a trade-off between the latency and the amount of resources, and certainly there is a trade-off between those factors and the supported representation for both data and results. In most of the practical cases, simulation is required for solving that complicated optimization problem that may also include specific technological constraints.
The original goal of the studies reported here was the development of a design library in VHDL for the elementary functions using a hardware-oriented algorithm, the well known Wong and Goto's ATA [9].
This paper proposes a slightly modification to the original ATA algorithm suggested after the results of preliminary exhaustive simulations. Results from complementary exhaustive simulation are presented that confirm the utility of the change. A mixed simulation framework is used that includes VHDL and Matlab®.
The following section describes the minutia related to elementary functions that are useful for the stated purpose.
Section 3 presents a short introduction to Wong and Goto's ATA methods for computing elementary functions in single precision; while Section 4 describes the observed trade-off and the proposed modification.
Section 5 describes the simulation results that support that modification. Finally Section 6 stresses some conclusions.

2. ELEMENTARY FUNCTIONS

This section begins with a coarse description of the IEEE 754 single precision standard and continues reviewing basic concepts related to elementary function accuracy, their computation, and common tricks used for reducing the range.

IEEE 754 Single Precision

IEEE Standard 754 [4] is the most used floating-point binary number representation. This standard was developed to facilitate the portability and the development of numerically oriented programs.
The standard defines both a 32-bit and a 64-bit format [1]. The former 4 bytes format is properly called single precision (floating-point binary number) representation.
In the single precision representation 1 bit is used for coding the sign, 8 bits are reserved for the biased representation of the exponent, and 23 bits are assigned to the fractional part of the significant. A normalized number requires a 1 bit to the left of the binary point; this fixed extra bit is implied, giving an effective 24-bit significant.

Errors in the Computation of Functions

The following accuracy criterion is used for the computation of the functions in single precision:

$$\text{MARE} < 0.5 \text{ ULP} < 2^{-24} \tag{1}$$

where MARE stands for Maximum Absolute value of Relative Error which is self explanatory. The term ULP (for Unit in the Last Place) denotes the distance between two floating-point numbers. If a function is computed in floating-point arithmetic with an error less than 0.5 ULP, it means that the exactly rounded result in the round-to-nearest mode is always provided. Rounding a number X to the nearest is the machine number that is the closest to X [2].
It should be pointed out that this does not necessarily guarantee that the result is correctly rounded in the sense that the result of the computation is the nearest representable number to the infinitely accurate true result. The number of accuracy bits necessary to ensure correctly rounding for the various elementary function is in general not known [5]. Above MARE is computed using a reference 'exact' result that in any case presented here is a IEEE double precision floating-point number returned by the Matlab® intrinsic functions.
The exhaustive simulation checks are for the *absolute* errors in the approximations. To get the *relative* errors to within half ULP, the actual criterion for accuracy, we need to consider the exponent of the input. This will be discussed next.

Computing Elementary Functions

Given an accurate approximation of the functions in the ranges given in Table 1, it is fairly straightforward in most cases to handle the sign and exponent.

Function	Range	Function	Range
Reciprocal	$[1,2)$	Square Root	$[1,4)$
Exponential	$[0,1)$	Logarithm	$[0.5,2)$
Sine	$[0,\pi/2)$	Arc Tangent	$[0,1)$

Table 1: Mantissa Ranges

For most of the functions, f, we are considering, there exist functions, g and h such that

$$f(X) = h(g(s,e), \tilde{f}(m)) \tag{2}$$

where $s, e,$ and m are respectively the sign, exponent and mantissa bits of X. Since the sign and exponent bits are altogether 9 bits $G(s,e)$ can often be read from a table.

Argument Reduction Tricks
Reduction of the arguments to principal values has to be performed before computing the function (using ATA or any other method) and eventually the result have to be also manipulated. Let us explore some of those tricks.

Reciprocal:

$$1/X = (-1)^s \times 2^{-e} \times (1/m)_{\text{ATA}} \tag{3}$$

Normalization is necessary to bring the result into IEEE format. That normalization is in this case a multiplication by a factor of two, i.e., a 1 bit shift to the left for the binary fraction, then more than 25 bits are necessary in the representation of the results to maintain MARE $<$ 0.5 ULP.

Square Root:

$$\sqrt{X} = 2^{e'} \times m'$$

where

$$e' = \begin{cases} \dfrac{e}{2} & \text{if } e \text{ is even} \\ \\ \dfrac{e-1}{2} & \text{if } e \text{ is odd} \end{cases} \tag{4}$$

$$m' = \begin{cases} \left(\sqrt{m}\right)_{\text{ATA}} & \text{if } e \text{ is even} \\ \\ \left(\sqrt{2 \times m}\right)_{\text{ATA}} & \text{if } e \text{ is odd} \end{cases} \tag{5}$$

Exponential:

$$\exp(X) = \exp(p) \times (\exp(q))_{\text{ATA}} \tag{6}$$

where $X = p + q, p > 0$, i.e., de-normalize X. Note if $p > 2^7$ the result is not representable in IEEE Single Precision (overflow for those numbers) and therefore $\exp(p)$ can be read from a table of length 128.

[1]The standard also defines two implementation-dependent extended formats.
[2]With a special convention if X is exactly between two machine numbers: the chosen number is the even one.

The reduction tricks for the previously presented function are straightforward. However other elementary functions are more problematic in many senses, e.g., the sine function. The reader should refer to [9] for details about how to do radian reduction over the entire range of floating point number accurately.
There is also a serious problem with straightforward application of ATA to some elementary functions. Because ATA uses a truncated Taylor polynomial (see next section), there is a problem with the relative error when X is small as in the first argument values in the sine function. To overcome this problem the input is divided into cases and particular tables are used for each subrange. See also reference [9] for details.

3. THE ATA METHOD

In the following a short description of Wong and Goto's ATA method is given (see [8] and [9] for details). Sign and exponent precessing is ignored in any respect. Recall the following definitions: [3]

- Fraction is the aggregate of the 23-bit of the significant with explicit bit representation in the standard. The number of different fractions is 2^{23}.
- Mantissa is the normalized number representation of the significant, i.e., it includes the implicit 1 bit and the binary point. Consequently, the range of the normalized mantissa is $[1, 2)$ with a resolution of 2^{-23}.

Let X be a fixed point input representing the mantissa of an IEEE Single Precision Floating Point number which is 24 bits in length. However, due to special requirements in handing the exponent, it is sometimes possible to take values outside the $[1, 2)$ interval in which the IEEE single precision floating number's mantissa is obliged to remain. We therefore assume X to be on different ranges depending on the function as was shown in Table 1.
Divide X into the following chunks,

$$X = x_0 + \lambda x_1 + \lambda^2 x_2 + \lambda^3 x_3 \tag{7}$$

where $\lambda = 2^{-6}$, $x_i < 1$, $i = 1, 2, \text{or } 3$, are 6 bits in length. On the other hand, depending of the function, we have x_0 anywhere in $[0, 4)$. Let f be a function we wish to compute and $f^{(n)}$ be the nth derivative of f. We can write the Taylor polynomial for f as follows

$$f(X) = \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0 + \lambda x_1)}{n!}(\lambda^2 x_2 + \lambda^3 x_3)^n \tag{8}$$

The aim is to approximate $f(X)$ by neglecting terms smaller than λ^5. Expanding (8) and taking the first terms the resulting equation (9) is the ATA formula. ATA method was proposed by Wong and Goto for the approximation of elementary function for a bounded range [8]. ATA is the acronym for Add-Table lookup-Add, which are the three basic sequential operations required for computing $\tilde{f}(X)$:

1. Add (or subtract) to get the four 13 bit central differences. $x_0 + \lambda x_1 + \lambda x_2$, $x_0 + \lambda x_1 - \lambda x_2$, $x_0 + \lambda x_1 + \lambda x_3$,$x_0 + \lambda x_1 - \lambda x_3$ (the number $x_0 + \lambda x_1$ is obtained by concatenation).

2.1 Read five f values from only one table, or from parallel tables to improve latency.

2.2 Read the last term value, which is a function of x_0 and x_2, from a different table [4].

3. Add all together the six tables values.

The values stored in the tables are double precision fractional representations in the worst case, i.e., for some tables the length can be shorter than 52-bit.
Wong was able to analytically prove in [7] that, for all the functions and their ranges as given in Table 1, the maximum absolute error is no more than 0.5 ULP.
The author (as in turn Wong and Goto did) wrote a program that makes an exhaustive check of the ATA formula. This is possible because the inputs are 24 bits in length, therefore making it realistic to do so using a computer. The results are shown in Table 2 are identical to those reported in [9] except for the Sine function. The difference may

$$\begin{aligned}\tilde{f}(X) \;=\;& f(x_0 + \lambda x_1) + \frac{\lambda}{2}\{f(x_0 + \lambda x_1 + \lambda x_2) - f(x_0 + \lambda x_1 - \lambda x_2)\} + \\ & \frac{\lambda^2}{2}\{f(x_0 + \lambda x_1 + \lambda x_3) - f(x_0 + \lambda x_1 - \lambda x_3)\} + \lambda^4\left\{\frac{x_2^2}{2}f^{(2)}(x_0) - \frac{x_2^3}{6}f^{(3)}(x_0)\right\}\end{aligned} \tag{9}$$

[3]These denominations may sound inverted but they are compatible with the IEEE standard use for the word fraction.

[4]The value comes, not from the original function f but from a combination of the second and third derivative of that function.

be due to the use of Matlab® instead of Fortran.

Function	Range	Error
Reciprocal	$1 \leq X < 2$	27.3 bit
Square Root	$1 \leq X < 2$	31.6 bit
Square Root	$2 \leq X < 4$	33.3 bit
Exponential	$0 \leq X < 1$	28.3 bit
Sine	$0 \leq X < \pi/2$	29.8 bit
Natural Logarithm	$0.5 \leq X < 1$	29.1 bit
Natural Logarithm	$1 \leq X < 2$	29.1 bit
Arc Tangent	$0 \leq X < 1$	29.3 bit

Table 2: Original Maximum Absolute Error

Implementation Details

VHDL language descriptions are not presented due to lack of space. For details see [2]. Tables 3 and 4 present a summary of the relevant characteristics of the design's components for any elementary function.

Component	Characteristic	Qt.
`Adder`	13-bit adder	2
`Subtr`	13-bit subtracter	2
`Tree`	6-operand adder	1

Table 3: Components

Table	Range
`T0 and T5`	2^{12}
`T1 and T2`	$2^{12} + 2^{6} - 1$
`T3 and T4`	$2^{12} + 2^{5} - 1$

Table 4: Table Details

From Table 3 it can be observed that basically only adders are needed. From Table 4 the total tables' length is just $24,764$ [5].

4. VARYING CHUNK SIZES

Analysis of MAE

Recall that in Table 2 the Error is expressed as a bit position,

$$\text{Error} = -\log_2 \text{MAE} \tag{10}$$

The analysis of MAE from Table 2 and the consideration of the reduction tricks demonstrate that the worst case, in the sense of closeness of MARE to 1/2 ULP, corresponds to the Reciprocal. For any other function the accuracy of MAE is fairly larger than is needed. The redundancy is due to the number of bits assigned to the chunks of the mantissa to perform the required accuracy for any function.

[5]The ratio $24,764/\,2^{23}$ is improper as a performance index.

An Integer Optimization Problem

Looking at equation (9) it can be observed a trade-off between the size of the chunks and the length of the tables demanded. As some others trade-off problems, that can be expressed in terms of an optimization problem. Then, the following integer optimization problem is stated:

> Find a mantissa partition [6] that leads to the shortest tables for all functions while keeping the MARE s within 0.5 ULP.

In mathematical form, the problem is stated as

Find

$$\min_{\boldsymbol{p}} T_{\mathcal{F}}(\boldsymbol{p}) \tag{11}$$

subject to

$$E_{f_i}(\boldsymbol{p}) < \frac{\text{ULP}}{2} \qquad \forall i \tag{12}$$

where $\boldsymbol{p}$ is the unknown mantissa partition, the objective function $T_{\mathcal{F}}(\cdot)$ is the tables cumulative length function, the constraints $E_{f_i}(\cdot)$ are the maximum absolute relative error functions, and $\mathcal{F} = \{f_i\}$ is the set of all elementary function under consideration.

Furthermore partition $\boldsymbol{p}$ is constrained to four blocks or chunks as in the original ATA algorithm. In turn, $E_{f_i}(\cdot)$ also depends upon particular normalization aspects and upon the maximum absolute errors of the ATA formula.

A good approximation to function $T_{\mathcal{F}}(\cdot)$ can be computed straightforward (an example was shown in Table 4).

A definition of partition is needed. For that purpose redefine the division of X into chunks,

$$\begin{aligned} X \;=\; & x_0 + 2^{-p_1}x_1 + \\ & 2^{-(p_1+p_2)}x_2 + 2^{-(p_1+p_2+p_3)}x_3 \end{aligned} \tag{13}$$

where $x_i < 1$, $i = 1, 2, \text{or } 3$ and x_0 anywhere in $[0, 4)$. Consequently, equation (9) is easily reinterpreted.

Any partitions is now characterized by a 3-element vector, $\boldsymbol{p}_i = [\,p_1\ p_2\ p_3\,]$. The original ATA partition is $\boldsymbol{p}_0 = [\,6\ 6\ 6\,]$.

Proposed Partitions

The analytical solution of the problem is difficult. However. after common sense observations, three candidate partitions (involving shorter tables than the original) are found,

$$\boldsymbol{p}_1 = [\,5\ 6\ 6\,]\,, \quad \boldsymbol{p}_2 = [\,6\ 5\ 6\,]\,, \quad \boldsymbol{p}_3 = [\,7\ 4\ 6\,]$$

The proposed partitions are almost identical given that x_0 and x_1 are concatenated in five out

[6]The word partition is used in a loose sense. Actually, the target is an ordered partition of the mantissa into four chunks.

of six table addresses (see equation (9)). Actually that fact is a key factor used for the selection of the partitions.

The key criteria for the selection of the partitions are the following:

1. The principal way to reduce table lengths is to reduce the number of bits in the concatenation between x_0 and x_1. The proposed partitions take 11 bit for that concatenation instead of the original 12 bit.

2. The way to control the accuracy of $\hat{f}$ is varying how many bits are assigned to x_0, out of 11 bits.

From the above, it can be observed that just two elements of the partition vector remains as free variables, i.e, the third element is fix in 6.
The length of the tables for the proposed partitions are shown in Tables 5, 6 and 7. Observe that partition $\boldsymbol{p}_1$ is the most convenient in term of savings, on the contrary $\boldsymbol{p}_3$ is the less convenient in that respect.

Table	Range
`T0 and T5`	2^{11}
`T1, T2, T3, and T4`	$2^{11} + 2^6 - 1$

Table 5: Table Details for $\boldsymbol{p}_1$

Table	Range
`T0`	2^{11}
`T1, T2, T3, and T4`	$2^{11} + 2^6 - 1$
`T5`	2^{12}

Table 6: Table Details for $\boldsymbol{p}_2$

Table	Range
`T0`	2^{11}
`T1, T2, T3, and T4`	$2^{11} + 2^6 - 1$
`T5`	2^{13}

Table 7: Table Details for $\boldsymbol{p}_3$

The proposed partitions are checked in the next section.

5. SIMULATION AND DISCUSSION

Exhaustive simulation is used again to compute MAE for the new partitions. From the numerous simulations, the extracted results shown in Table 8 are summarize by the following statements:

- Reciprocal: Partition $\boldsymbol{p}_3$ is the only one that keeps MARE < 0.5 ULP.
- Other Functions: Partition $\boldsymbol{p}_1$ is enough for the rest of the elementary function considered here.

Function	Range	$\boldsymbol{p}$	Error
Reciprocal	$1 \le X < 2$	$\boldsymbol{p}_1$	23.7 bit
Reciprocal	$1 \le X < 2$	$\boldsymbol{p}_2$	24.7 bit
Reciprocal	$1 \le X < 2$	$\boldsymbol{p}_3$	25.7 bit
Square Root	$1 \le X < 2$	$\boldsymbol{p}_1$	28.2 bit
Square Root	$2 \le X < 4$	$\boldsymbol{p}_1$	31.1 bit
Exponential	$0 \le X < 1$	$\boldsymbol{p}_1$	26.6 bit
Sine	$0 \le X < \pi/2$	$\boldsymbol{p}_1$	27.2 bit
Natural Log	$1/2 \le X < 1$	$\boldsymbol{p}_1$	25.6 bit
Natural Log	$1 \le X < 2$	$\boldsymbol{p}_1$	25.6 bit
Arc Tangent	$0 \le X < 1$	$\boldsymbol{p}_1$	25.9 bit

Table 8: Maximum Absolute Error

Formally the optimization goal is reached only when partition $\boldsymbol{p}_3$ is used for all the elementary functions. Nevertheless, for particular designs, not all the functions addressed here are implemented.
As an alternative, different partitions can be used within the same design. In such a case the table length of the correction factor is different for different functions. The practical consequence is a complication of the multiplexing. Also different tables can be used for different portions of the range [2].
The reduction in the length of the tables does not necessary have a practical application. Due to technological restrictions, the length of the tables may be an integer power of two. E.g., a table of length $2^{11} + 2^6 - 1$ must be implemented as one of length 2^{12}.
The worse situation for the utility of the alternative presented here is the combination of the following four factors:

1. The table lengths are restricted to be an integer power of two.

2. A recursive design in which just a single table per function is used.

3. A rigid scheme not allowing the multiplexing of tables of different length.

4. The inclusion of the Reciprocal function.

Under those conditions there is not saving at all when comparing this method with the original ATA.
Fortunately, a bunch of other combinations arise when relaxing one or more of the above restrictions. Normally six tables per function will be used to speed up the computation, and in that situation a reduction of the length of the tables becomes more valuable.

Suppose the implementation of the Reciprocal function with six parallel tables with lengths restricted to an integer power of two. Total table length is $40,960$ for partition $\boldsymbol{p}_0$, and $26,624$ for partition $\boldsymbol{p}_3$, i.e., a 35% saving is obtained. For any other function, under similar conditions, the saving is 50%.

6. CONCLUSIONS

Wong and Goto's ATA is an arithmetic algorithm for the computation of elementary functions in single precision. ATA is an alternative method to those based not only in pure polynomial approximations but also to those based on pure table-lookup. Precisely, the distinction of ATA is the mixing of the two techniques.
This paper presents a modification to the original ATA method. That change is based on the statement of an optimization problem related to the partition of the mantissa which is central to ATA. The optimization problem is not settled after mathematical concerns but instead after a pragmatic consideration, i.e., the saving of space. That optimization problem is also not solved analytically; instead exhaustive simulations are performed. The candidate partitions for the simulations are selected after simple common sense criteria.
For many of the possible implementations, the modification allows a significant saving in space without affecting precision and latency. The simulations show a reduction of table length ranging from approx. 0 to 50% and with minor hardware additions expected.
The VHDL functional descriptions are a by-product of the mixed simulation environment used here. From them is possible to begin the hardware implementation of the components, passing through the register transfer description, all the way to the technological mapping.

REFERENCES

[1] J. Bhasker, *A Guide to VHDL Syntax*, Prentice Hall, 1995.

[2] O. N. Bria and J. O. Giacomantone, "Descripciones VHDL para Calcular la Función Recíproca en Presición Simple," *Annals VIII Congreso Argentino de Ciencias de la Computación*, Buenos Aires, Argentina, 2002.

[3] C. Hansen, O. Bringmann. W. Rosenstiel, "A VHDL reusable component model for fixed abstraction level simulation and behavioral synthesis," *Virtual Components Design and Reuse*, edited by R. Seepold, N. Martnez, Kluwer, 2001.

[4] W. Kahan, "Lecture notes on the status of standard IEEE-754, from the website `http.cs.berkeley.edu/~wkahan/`, 1996.

[5] J.-M. Muller, *Elementary Functions - Algorithms and Implementation*, Birkhäuser, 1997.

[6] B. Parhami, *Computer Arithmetic. Algorithms and Hardware Designs*, Oxford University Press, 2000.

[7] W. F. Wong, E. Goto, "Hardware for the Fast Computation of the Elementary Functions." *Dr. Eng. Sc. Thesis*, University of Tsukuba, Tsukuba, Japan,1995.

[8] W. F. Wong, E. Goto, "Fast hardware-based algorithms for elementary function computations using rectangular multipliers," *IEEE Transactions on Computer*, 1994.

[9] W. F. Wong, E. Goto, "Fast evaluation of the elmentary function in single precision," *IEEE Transactions on Computer*, 1995.