# A Floating-Point Processor for Fast and Accurate Sine/Cosine Evaluation

Vassilis Paliouras, *Member, IEEE,* Konstantina Karagianni, *Member, IEEE,* and Thanos Stouraitis, *Senior Member, IEEE*

***Abstract*—A VLSI architecture for fast and accurate floating-point sine/cosine evaluation is presented, combining floating-point and simple fixed-point arithmetic. The algorithm implemented by the architecture is based on second-order polynomial interpolation within an approximation interval which is partitioned into regions of unequal length. The exploitation of certain properties of the trigonometric functions and of specific bit patterns that appear in the involved computations, has led to reduced memory size and low overall hardware complexity. In fact, a 40% memory size reduction is achieved by the introduced simplified memory interleaving scheme, when compared to a traditional interleaved memory architecture. The proposed architecture has been designed and simulated in a 0.7-$\mu$m CMOS process technology, to prove its amenability for VLSI implementation. The time required to evaluate a sine is less than the time required for three single-precision floating-point multiply-accumulate (MAC) operations, while the computed values are guaranteed to be accurate to half a unit in last position (ulp). To prove the accuracy of the algorithm, an error analysis for the computation of second-order Hörner polynomial is provided, based on novel formulae which have been recently introduced in the literature by the authors for roundoff error bounds in floating-point addition and multiplication.**

***Index Terms*—Arithmetic, digital arithmetic, floating-point arithmetic, roundoff errors, very-large-scale integration.**

## I. INTRODUCTION

HARDWARE support for the evaluation of sine/cosine functions has become an essential demand in many applications, as these functions appear frequently in scientific computations, signal processing, and computer graphics. A wide variety of hardware algorithms for the evaluation of sine and cosine functions has been reported in the literature. Among them, CORDIC-based techniques, linear approximation algorithms, and general polynomial or even rational approximations are the most commonly used [1]–[5]. In [5], the reader may find a survey on well-known techniques for computing, among other functions, the sine/cosine function.

The architectures for the computation of trigonometric functions can be distinguished into two classes, depending on whether they include a multiplier or not. Concerning multiplierless architectures, they may lead to inefficient solutions due to their large memory requirements, their need for multi-operand addition hardware [2], or their slow linear convergence. Moreover, such architectures do not utilize the floating-point multiplier which does very often exist in modern VLSI architectures, to cover computational needs besides the evaluation of the sine/cosine. It stands to reason that the existence of this multiplier should be exploited. Hence, architectures that, with moderate hardware overhead, allow the computation of sine and cosine functions on floating-point multipliers/adders, provide a good solution: they achieve high utilization of existing resources, while the introduction of under-utilized hardware, yet expensive in terms of area occupation, is avoided. Interpolation-based techniques, which come under this category, are particularly interesting due to their accuracy, speed of convergence, and suitability for VLSI implementation. Compared to stored-coefficient polynomial approximation, interpolation is found to require less memory space.

In this paper, an algorithm and a corresponding VLSI architecture for the fast and accurate computation of single-precision floating-point sine and cosine are introduced. The algorithm is based on second-order polynomial interpolation with partitioning of the interpolation interval into regions of unequal lengths, while properties of the trigonometric functions and of particular bit patterns that appear in the computations, are exploited to reduce memory requirements and computational complexity. The computations are performed by combining floating-point and simple fixed-point arithmetic. In particular, the interpolating polynomial coefficients are computed by bit manipulations and simple fixed-point operations, while the interpolating polynomial is computed using an IEEE 754-compliant floating-point multiplier and a floating-point adder of extended precision. It is shown that, due to the nature of the particular computations, the extended precision does not impose significant hardware overhead.

Furthermore, by utilizing novel formulae for roundoff error bounds on floating-point multiplication and addition, which have been recently introduced by the authors in the literature, it has been shown that the introduced hardware algorithm achieves accuracy better than half a unit in last position (ulp). The new error bounds lead to simplified hardware by allowing the use of reduced-wordlength intermediate results, compared to wordlengths dictated by more loose error bounds for obtaining identical accuracy. In the particular application, the error bounds are utilized to determine the maximum tolerable error in the computation of the interpolating polynomial coefficients, as the contribution of floating-point operations' roundoff is subtracted from the target overall error, the difference being the requirement for coefficient accuracy. The minimal wordlengths of intermediate results in coefficient evaluation, which does

Manuscript received June 1998; revised January 2000. This paper was recommended by Associate Editor M. Bayoumi.

The authors are with the Electrical and Computer Engineering Department, University of Patras, 26500 Patras, Greece.

Publisher Item Identifier S 1057-7130(00)04207-5.

not contain floating-point operations, have been identified by exhaustive simulation, and are feasible due to the small number of coefficients.

A main advantage of the proposed architecture is that even though it employs a memory interleaved scheme, it requires 40% less memory space than a direct interleaved memory-based architecture. The memory reduction is achieved by employing trigonometric identities to reduce the number of successive values that need to be simultaneously fetched from the memory to two, instead of three, which are normally required for second-order polynomial interpolation [6].

To prove the amenability of the proposed algorithm and architecture for VLSI implementation, layout organization issues are also studied. A layout organization, the main benefit of which is the minimization of routing complexity, is derived. The particular layout which does not include a floating-point MAC, fits into approximately 26-mm$^2\mu$m in a 0.7-$\mu$m double-metal CMOS process technology, and the corresponding delay equals the delay of a MAC; thus the time required for a sine evaluation equals three MAC's. As the proposed architecture can be implemented in a variety of ways in terms of the micro-architecture of the constituent blocks, a variety of area/time specifications can be met. For example, the delay can be halved, at the corresponding increase of hardware complexity. Nevertheless, the general layout organization is applicable independently of the adopted design style (standard-cell, handcraft design) or the particular architecture implementation.

The remainder of this paper is organized as follows. In Section II, the proposed algorithm is presented. In Section III, a VLSI architecture for its implementation is described. VLSI layout organization issues and performance comparisons are discussed in Section IV, and conclusions are presented in Section V.

## II. The Proposed Algorithm for Sine/Cosine Computation

The proposed algorithm computes the sine/cosine of a floating-point number in the range of $-(\pi/2)$ to $\pi/2$. Since it holds that $\sin(-x) = -\sin x$ and $\cos(-x) = \cos x$, the approximation interval may be reduced to $[0, \pi/2]$. To achieve the half-an-ulp accuracy constraint while minimizing memory requirements, a flexible memory partitioning scheme has been applied: the whole approximation interval is partitioned into regions $[x'_j, x'_{j+1})$, divided into subintervals by the points $x_{i,j}$ that have a distance $h_j$ between them, i.e., $x_{i,j} \in [x'_j, x'_{j+1})$, with $x_{i+1,j} = x_{i,j} + h_j$ and $x_{i-1,j} = x_{i,j} - h_j$. The length $h_j$ in each region $j$, is selected to be the maximum power of two that permits the meeting of the error constraint in the particular region. A constant $h_j$ independent of $j$, throughout the entire approximation interval would have led to excessive memory requirements. The fact that $h_j$ is not constant over the range of interest does not affect the applicability of the interpolation algorithm.

The second-order interpolation polynomial $\hat{f}(u)$ that approximates the sine $f(x) = \sin x$ of an argument $x$, is

$$\hat{f}(u) = a + u \times (b + u \times c) \tag{1}$$

where

$$a = f(x_{i,j}) \tag{2}$$

$$b = \frac{1}{2}\left(f(x_{i+1,j}) - f(x_{i-1,j})\right) \tag{3}$$

$$c = \frac{1}{2}\left((x_{i+1,j}) - 2f(x_{i,j}) + f(x_{i-1,j})\right) \tag{4}$$

and

$$x_{i,j} = \left\lfloor \frac{x}{h_j} \right\rfloor h_j. \tag{5}$$

The argument of the polynomial is $u = (x/h_j) - \lfloor x_{i,j}/h_j \rfloor \in [0, 1)$. Notice that if $x \in [x_{0,j}, x_{1,j})$, then (3) and (4) require the function value at the point $x_{-1,j}$ defined as $x_{-1,j} = x_{0,j} - h_j$.

The approximation error $\varepsilon$ of the second-order polynomial interpolation in the interval $[x_{i,j}, x_{i+1,j})$ is bounded by $\varepsilon_{\text{approx}}(x_{i,j}, h_j)$ [7], with

$$\varepsilon < \varepsilon_{\text{approx}}(x_{i,j}, h_j) = \frac{h_j^3}{3!}\frac{2\sqrt{3}}{9}\left|\cos x_{i,j}\right|. \tag{6}$$

In case $x < 2^{-12}$, it is $\sin x \simeq x$ [2], so $\sin x$ is approximated by

$$\hat{f}(x) = x$$

while for $\cos x$, it holds that $\cos x \simeq 1 - (x^2/2)$.

An outline of the proposed algorithm is shown in Fig. 1, where $u_{FXP}$ denotes the denormalized fixed-point representation of $u$. The computation of $u_{FXP}$ in Line 16, consists of selecting the $35 - |\log_2 h_j|$ least significant bits of $x$. In Line 17, $u_{FXP}$ is normalized to adhere to the IEEE 754 single-precision format, by

$$\frac{u_{FXP}}{2^{\lfloor \log_2 u_{FXP} \rfloor}} 2^{\lfloor \log_2 u_{FXP} \rfloor}.$$

To compute $\sin x$ by the algorithm of Fig. 1, the following input parameters are determined:

1) the $x'_j$ points that partition the entire approximation interval into regions of subintervals of length $h_j$;
2) the subinterval lengths $h_j$;
3) the mantissa word lengths $t_a$, $t_b$, and $t$ of the polynomial coefficients $a$, $b$, and $c$, respectively.

These parameters are predetermined and are not computed for each argument $x$.

An algorithm for determining the approximation interval partitioning, i.e., the sequences $x'_j$ and $h_j$, is shown in Fig. 2. The total computational error bound utilized in the algorithm of Fig. 2 consists of three components: the error bound due to polynomial approximation $AppErr(x, h) = (h^3/3!)2(\sqrt{3}/9)\cos(x)$, the bound of the error in coefficient computation, and the the error bound due to floating-point operation roundoff.

The triplet $\{t_a, t_b, t\}$ of the word lengths of the coefficients defines the quantization component of the complete computational error through the introduced error bounds for FLP opera-

```
0.  Predefined data:  interval lengths $h_j$,
partitioning sequence $x'_j$, lookup table entries $\sin x_{i,j}$
1.  Input data:  argument $x$
2.  IF $\sin x$ THEN
3.  IF $x < 2^{-12}$ THEN RETURN $x$ ENDIF
4.  ELSE CALL SINE_CALCULATION
5.  ENDIF
6.  IF $\cos x$ THEN
7.  IF $x < 2^{-12}$ THEN RETURN $1 - \frac{x^2}{2}$ ENDIF
8.  ELSE $x \leftarrow \frac{\pi}{2} - x$
9.  CALL SINE_CALCULATION
10.  ENDIF
11.  END.
12.  PROCEDURE SINE_CALCULATION
13.  BEGIN
14.  Select $x'_j : \max_j\{x'_j\} \leq x$
15.  Determine the corresponding $h_j$
16.  Compute $u_{FXP}$
17.  Compute FLP argument $u$
18.  Compute coefficients $a, b, c$
19.  RETURN $(c \cdot u + b) \cdot u + a$
20.  END PROCEDURE.
```

Fig. 1. Outline of the proposed algorithm.

```
$i := 0$ ; $j := 0$ ; $t := 23$ ; $t_a := 26$ ; $t_b := 22$ ;
$x''_0 := 2^{-12}$ ; $h_0 := 2^{-15}$ ; $x_{0,0} := x''_0$ ; $low = x_{0,0}$;
WHILE $\left(low \leq \frac{\pi}{2}\right)$ DO
        BEGIN
        $testh := 2h_j$ ;
        $a = \sin(low)$ ;
        $b = \frac{1}{2}\left(\sin(low + testh) - \sin(low - testh)\right)$ ;
        $c = \frac{1}{2}\left|\sin(low + testh) - 2\sin(low) + \sin(low - testh)\right|$ ;
        $er1 = 2^{-t-1}2^{Ex(c)} + 2^{-t_b-1}2^{Ex(b)} + 2^{-t-1}2^{Ex(b)} + 2^{-t_a-1}2^{Ex(a)}$;
        $er2 = 2^{-t_a-1}2^{Ex(a)} + 2^{-t_b-1}2^{Ex(b)} + 2^{-t-1}2^{Ex(c)}$;
        $toterr = er1 + er2 + AppErr(low, testh)$ ;
        IF $\frac{toterr}{2^{-23}2^{\lfloor \log_2 \sin(low) \rfloor}} \leq \frac{1}{2}$ THEN
            $j := j + 1$ ;
            $h_j := testh$ ;
            $x''_j := x_{i,j}$ ;
            $i := 0$ ;
        ELSE
            $i := i + 1$ ;
            $x_{i,j} := x_{i-1,j} + h_j$;
            $low = low + h_j$ ;
        END IF;
END WHILE;
RETURN $\left(\{x''_j\}, \{h_j\}\right)$;
```

Fig. 2. Algorithm for the determination of $x''_j$ and $h_j$. The function $Ex(x) = \lfloor \log_2 x \rfloor$ returns the exponent of the quantity $x$, when expressed in IEEE-754 compliant format.

tions. The requirement of half-an-ulp accuracy and the quantization component dictate a maximum acceptable value for the approximation error, a bound on which is given by (6). Hence, the sequences $x'_j$ and $h_j$ that define the partitioning of the entire approximation interval should impose an approximation error not larger than the previously derived acceptable maximum value. A second constraint is obtained by the need for minimal memory requirements, an estimate of which is

$$M = \sum_j \frac{x'_{j+1} - x'_j}{h_j}.$$

Therefore, a procedure for the definition of the predefined data of the algorithm in Fig. 1, consists of identifying the triplet $\{t_a, t_b, t\}$ which dictates the sequences $x'_j$ and $h_j$ that impose minimal memory requirements. Running the Algorithm of Fig. 2 for various triplets, it has been found that a triplet which minimizes the memory and meets the error specification is $\{t_a, t_b, t\} = \{26, 22, 23\}$.

To guarantee the correctness of the result and to simplify the operations for the argument reduction, the sequence of $x'_j$ occurs from $x''_j$ computed in the algorithm of Fig. 2, as dictated by

$$x'_j = \left\lceil \frac{x''_j}{h_j} \right\rceil h_j \qquad (7)$$

which returns the smallest integral multiple of $h_j$ that is greater than $x'_j$. The derived regions are shown in Table I.

For simplicity, several successive regions have been merged into one, to form a region which consists of subintervals, the length of each of which equals to the minimum of the lengths of the subintervals of the original regions, shown in Table I. In this way, the complexity of address generation is reduced at the cost of a memory size increase of a few words. However, in the particular case, more words do not necessarily require more chip area to be stored, due to ROM generator restrictions. Specifically, the available ROM generators (cf. [8]) impose restrictions on the minimal size and on the aspect ratio of the memory array,

TABLE I
$x'_j$ AND $h_j$ SEQUENCES

| $j$ | Region | Subinterval Length $h_j$ | | Merged Region | $j$ |
|---|---|---|---|---|---|
| | $\left[x'_j, x'_{j+1}\right)$ | single region | merged region | $\left[\hat{x}'_j, \hat{x}'_{j+1}\right)$ | |
| 0 | $\left[2^{-12}, \frac{5}{16384}\right)$ | $2^{-15}$ | $2^{-15}$ | $\left[2^{-12}, \frac{5}{8192}\right)$ | 0 |
| 1 | $\left[\frac{5}{16384}, \frac{5}{8192}\right)$ | $2^{-14}$ | | | |
| 2 | $\left[\frac{5}{8192}, \frac{5}{4096}\right)$ | $2^{-13}$ | $2^{-13}$ | $\left[\frac{5}{8192}, \frac{5}{2048}\right)$ | 1 |
| 3 | $\left[\frac{5}{4096}, \frac{5}{2048}\right)$ | $2^{-12}$ | | | |
| 4 | $\left[\frac{5}{2048}, \frac{9}{1024}\right)$ | $2^{-11}$ | $2^{-11}$ | $\left[\frac{5}{2048}, \frac{17}{512}\right)$ | 2 |
| 5 | $\left[\frac{9}{1024}, \frac{17}{512}\right)$ | $2^{-10}$ | | | |
| 6 | $\left[\frac{17}{512}, \frac{33}{256}\right)$ | $2^{-9}$ | $2^{-9}$ | $\left[\frac{17}{512}, \frac{33}{256}\right)$ | 3 |
| 7 | $\left[\frac{33}{256}, \frac{13}{16}\right)$ | $2^{-8}$ | $2^{-8}$ | $\left[\frac{33}{256}, \frac{13}{16}\right)$ | 4 |
| 8 | $\left[\frac{13}{16}, \frac{3}{2}\right)$ | $2^{-7}$ | $2^{-7}$ | $\left[\frac{13}{16}, \frac{\pi}{2}\right)$ | 5 |
| 9 | $\left[\frac{3}{2}, \frac{\pi}{2}\right)$ | $2^{-6}$ | | | |

TABLE II
BINARY VALUES OF $\cos(h_j)$ FOR $h_j = 2^{-i}$

| $i$ | $\cos(2^{-i})$ |
|---|---|
| 15 | 0.11111111111111111111111111111110000000000000000000000000000000001 |
| 13 | 0.11111111111111111111111111100000000000000000000000000000101010101 |
| 12 | 0.11111111111111111111111110000000000000000000000000001010101010101 |
| 11 | 0.11111111111111111111111000000000000000000000000010101010101010101 |
| 10 | 0.11111111111111111111100000000000000000000000101010101010101010101 |
| 9 | 0.11111111111111111110000000000000000000001010101010101010101010010 |
| 8 | 0.11111111111111111000000000000000000010101010101010101010010011111 |
| 7 | 0.11111111111111100000000000000000101010101010101010010011111010011 |
| 6 | 0.11111111111110000000000000001010101010101010010011111010010100011 |

as it is required to have a number of columns, which is a particular integral multiple of the number of rows. Finally, to avoid increasing the maximum ROM access delay, merging should be performed so that the size of the largest memory bank remains unchanged.

## A. *Evaluation of the Interpolation Polynomial Coefficients*

The evaluation of the coefficients $a$, $b$, and $c$ of the polynomial (1) is discussed below. The computation of the coefficient $a$ requires fetching the value $f(x_{i,j})$ from the memory. From

$$\begin{aligned} f(x_{i+1,j}) + f(x_{i-1,j}) &= \sin(x_{i,j} + h_j) + \sin(x_{i,j} - h_j) \\ &= 2f(x_{i,j}) \cos h_j \end{aligned}$$

we obtain

$$f(x_{i+1,j}) = 2f(x_{i,j}) \cos h_j - f(x_{i-1,j}). \tag{8}$$

From (3) and (8), it is obtained that $b$ can be computed through

$$b = \sin x_{i,j} \cos h_j - \sin x_{i-1,j} \tag{9}$$

where $x_{i,j}$ is defined by (5). The use of (9) greatly simplifies the interleaved ROM organization, as there is no need to simultaneously fetch the three subsequent words $f(x_{i-1,j})$, $f(x_{i,j})$, and $f(x_{i+1,j})$, which appear in (3), as required in the method by Lewis [6], but rather only two: $f(x_{i,j})$ and $f(x_{i-1,j})$.

From the entries of Table II, for the particular $h_j = 2^{-i}$ values, it is obtained that

$$\cos 2^{-i} \simeq 0.\underbrace{11\cdots1}_{2i+1}\underbrace{00\cdots0}_{2i+3}101\,010\,101. \tag{10}$$

The trailing bit pattern 101 010 101 is the shortest one that guarantees the meeting of the error constraint on $b$. From (10), it follows that

$$\cos h_j \simeq 1 - \frac{h_j^2}{2} + \frac{h_j^4}{2^4}\frac{341}{2^9} = 1 - \frac{h_j^2}{2} + \frac{341}{8192}h_j^4.$$

The error $e_{\cos}(h_j)$ due to the approximate evaluation of $\cos h_j$, such that

$$\cos h_j = 1 - \frac{h_j^2}{2} + \frac{341}{8192}h_j^4 + e_{\cos}(h_j) \tag{11}$$

introduces an error term in $b$, as follows:

$$\begin{aligned} b &= \sin x_{i,j}\left(1 - \frac{h_j^2}{2} + \frac{341}{8192}h_j^4 + e_{\cos}(h_j)\right) - \sin x_{i-1,j} \\ &= \left(\sin x_{i,j} + \frac{341}{8192}h_j^4 \sin x_{i,j}\right) \\ &\quad - \left(\sin x_{i-1,j} + \frac{h_j^2}{2}\sin x_{i,j}\right) + e_{\cos}(h_j)\sin x_{i,j}. \end{aligned} \tag{12}$$

Hence, the error term in $b$, due to approximate evaluation of $\cos h_j$, is

$$\begin{aligned} \varepsilon_b(h_j, x_{i,j}) &= e_{\cos}(h_j) \sin x_{i,j} \\ &= \left(\cos h_j - 1 + \frac{h_j^2}{2} + \frac{341}{8192}h_j^4\right) \sin x_{i,j}. \end{aligned} \tag{13}$$

As the values $\sin x_{i,j}(1 + (341/8192)h_j^4)$ and $\sin x_{i-1,j} + (h_j^2/2)\sin x_{i,j}$ are close, the subtraction in (12) may lead to a dramatic loss of precision. To obtain $b$ with half-ulp accuracy, the particular values should be available with increased length of significand. Exhaustive computer simulation, which is feasible due to the small number of possible coefficients, has shown that 38 bits of fixed-point input, word length of stored sine values, suffice for accurate computation of the coefficient $b$.

The computational complexity of (4) can be reduced when it is restated as

$$c = \frac{1}{2}\left(\sin(x_{i,j} + h_j) - 2\sin x_{i,j} + \sin(x_{i,j} - h_j)\right) \tag{14}$$

$$= -2\sin x_{i,j} \sin^2\left(\frac{h_j}{2}\right) \tag{15}$$

$$-2f(x_{i,j}) \sin^2\left(\frac{h_j}{2}\right). \tag{16}$$

By rounding the bit pattern 1 010 101 010 101 that follows the string of 1's, to 101010101011, the entries of Table III reveal that $\sin^2(2^{-i-1})$ may be approximated by

$$\sin^2(2^{-i-1}) \simeq 1.\underbrace{11\cdots1}_{2i+1}101\,010\,101\,011 \times 2^{-2i-3}$$

TABLE III
BIT PATTERNS FOR THE COMPUTATION OF $\sin^2(h_j/2)$ WHEN $h_j = 2^{-i}$

| $i$ | $\sin^2(2^{-i-1})$ |
|---|---|
| 15 | $1.11111111111111111111111111111111111110101010101010101010101010101 \times 2^{-33}$ |
| 13 | $1.11111111111111111111111111111110101010101010101010101010101101100 \times 2^{-29}$ |
| 12 | $1.11111111111111111111111111111010101010101010101010101010111011000001 \times 2^{-27}$ |
| 11 | $1.11111111111111111111111111101010101010101010101010101101100000010111 \times 2^{-25}$ |
| 10 | $1.1111111111111111111111110101010101010101010101010110110000001011011100 \times 2^{-23}$ |
| 9 | $1.11111111111111111111110101010101010101010101011011000000101101 1000001 \times 2^{-21}$ |
| 8 | $1.1111111111111111111101010101010101010101011011000000101101100000010000 \times 2^{-19}$ |
| 7 | $1.11111111111111111101010101010101010101101100000010110101111111001100 \times 2^{-17}$ |
| 6 | $1.1111111111111111010101010101010101101100000010110101110101110101111011 \times 2^{-15}$ |

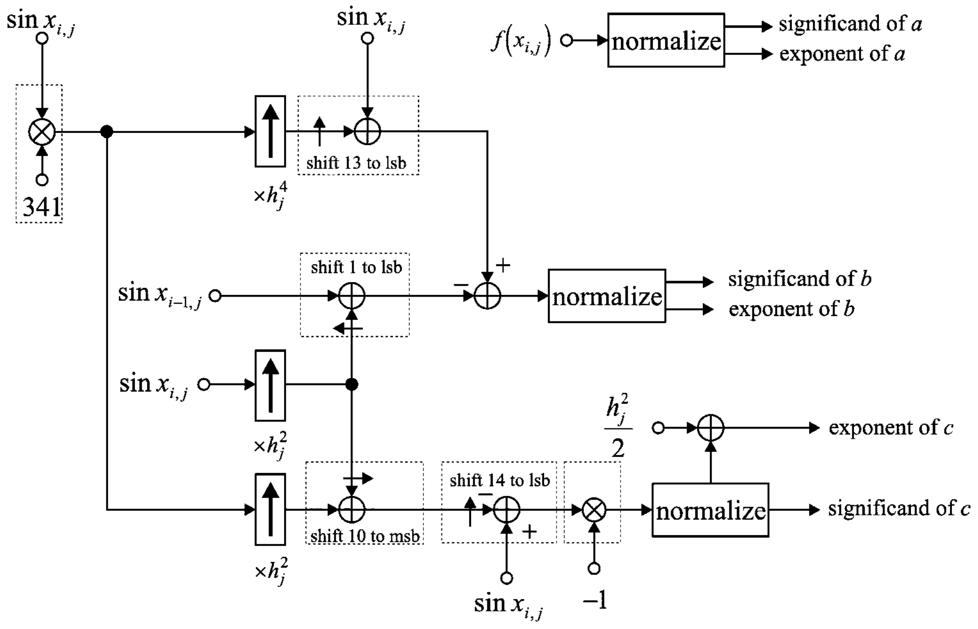


Fig. 3. Determination of interpolating polynomial coefficients.

the right-hand side of which, can be written as

$$\begin{aligned}\left(2-2^{-2i-1}+2^{-2i-13}2731\right)2^{-2i-3}\\ &= \frac{2^{-2i}}{4}-2^{-4i}\left(\frac{1}{16}-\frac{2731}{2^{16}}\right)\\ &= \frac{2^{-2i}}{4}-\frac{1365}{65\,536}2^{-4i}.\end{aligned}$$

Following an approach similar to the one followed for the estimation of the error term in the evaluation of $b$, it is found that the error term in $c$ is

$$\begin{aligned}\varepsilon_c(h_j, x_{i,j}) &= -2e_{\sin}(h_j)\sin x_{i,j}\\ &= -2\left(\sin^2\left(\frac{h_j}{2}\right)-\frac{h_j^2}{4}+\frac{1365}{65\,536}h_j^4\right)\sin x_{i,j}.\end{aligned}$$

It can be noticed that the constant 1365 can be decomposed as $1365 = 2^{10}+341$. Therefore a single multiplication by the constant 341 is needed for the computation of both the coefficients $b$ and $c$, instead of two, one with 341 and another with 1365. This property is exploited in the architecture of Fig. 3, which will be detailed in Section III. The word lengths of the intermediate results have been determined through exhaustive simulation and they are the minimal ones that guarantee the accuracy required for the coefficients.

## *B. Roundoff in Second-Order Polynomial Computation*

The second-order Hörner polynomial computation roundoff error is evaluated, based on the error bounds

$$\varepsilon_p = \epsilon_m 2^{E_p} \tag{17}$$

for floating-point multiplication and

$$\varepsilon_a = \epsilon_m 2^{E_a} \tag{18}$$

for floating-point addition, which were proposed in [9]. In (17) and (18), $\epsilon_m$ is the machine epsilon, while $E_p$ and $E_a$ are the exponents of the bounds on roundoff errors in floating-point multiplication and floating-point addition, respectively.

For the $\sin x$ computation, the computed polynomial $\tilde{y}$, which includes operations' roundoff error terms, is given as

$$\begin{aligned}\tilde{y} &= (cu + \varepsilon_{p_1} + b + \varepsilon_{a_1})u + \varepsilon_{p_2} + a + \varepsilon_{a_2} \\ &= cu^2 + bu + a + (\varepsilon_{p_1} + \varepsilon_{a_1})u + \varepsilon_{p_2} + \varepsilon_{a_2} \qquad (19)\end{aligned}$$

where error sources have been inserted to model roundoff at each operation. The total roundoff error term $|\Delta y| = |\tilde{y} - y|$, contained in (19), is bounded as follows:

$$\begin{aligned}|\Delta y| \le &\left(2^{-t-1}2^{E_c+E_u+1} + 2^{-t_b-1}2^{E_{a_1}}\right)u + 2^{-t-1}2^{E_{p_2}} \\ &+ 2^{-t_a-1}2^{E_{a_2}} \qquad (20)\end{aligned}$$

$$\begin{aligned}&= 2^{-t}u2^{E_c+E_u} + 2^{-t_b-1}u2^{E_{a_1}} + 2^{-t-1}2^{E_{p_2}} \\ &\quad + 2^{-t_a-1}2^{E_{a_2}}. \qquad (21)\end{aligned}$$

The maximum exponent of $(cu + b)u$, denoted in (20) by $E_{p_2}$ is $E_{p_2} = E_b$, because $\max_{0 \le u < 1}\{|b + cu| \cdot |u|\} < b$, since $\max_u\{|b + cu|\} = b$ and $0 \le u < 1$. The exponent of the inner summation is $E_{a_1} = E_b$, because $\max_{0 \le u < 1}\{|b + cu|\} = b$. Hence, (21) can be reduced into

$$|\Delta y| \le 2^{-t}u2^{E_c+E_u} + 2^{-t_b-1}2^{E_b}u + 2^{-t-1}2^{E_b} + 2^{-t_a-1}2^{E_a}. \qquad (22)$$

Since $u \in [0, 1)$, then $u = m_u 2^{E_u}$, with $E_u = \lfloor \log_2 u \rfloor < 0$. The product $u2^{E_u}$ in (22) can be written as

$$u2^{E_u} = u2^{\lfloor \log_2 u \rfloor} \qquad (23)$$

and it is maximized as $u$ approaches 1. Therefore, the maximum value that can be assumed by $u2^{E_u}$ is the limit $\lambda$, expressed using a dummy variable $\delta$, as

$$\lambda = \lim_{\delta \to 0}\left((1-\delta)2^{\lfloor \log_2(1-\delta) \rfloor}\right) = \frac{1}{2} \qquad (24)$$

because due to the floor function, $\lfloor \log_2(1-\delta) \rfloor = -1$ for small $\delta$. From (23) and (24), it is obtained that

$$0 \le u2^{\lfloor \log_2 u \rfloor} < \frac{1}{2} \qquad (25)$$

when $u \in [0, 1)$. Hence, the total operations' roundoff error is bounded by

$$\varepsilon_{\max} = 2^{-t}\,\frac{1}{2}\,2^{E_c} + 2^{-t_b-1}2^{E_b} + 2^{-t-1}2^{E_b} + 2^{-t_a-1}2^{E_a}. \qquad (26)$$

Finally, the contribution of coefficient quantization to the total error is described by

$$\varepsilon_{\text{coeff}} = |\Delta a| + |\Delta b|u + |\Delta c|u^2. \qquad (27)$$

The error bounds (26) and (27) are added to the bound (6) of polynomial approximation error to form the total error bound

$$\varepsilon_{\text{tot}} = \varepsilon_{\max} + \varepsilon_{\text{coeff}} + \varepsilon_{\text{approx}} \qquad (28)$$

which is exploited by the algorithm of Fig. 2 for the derivation of the sequences $x'_j$ and $h_j$. In the particular algorithm, $er1$ denotes the error $\varepsilon_{\max}$, $er2$ is an upper bound to $\varepsilon_{\text{coeff}}$, and $AppErr$ is the approximation error $\varepsilon_{\text{approx}}$.

The total error in the interval $[x_{i,j}, x_{i,j+1})$ depends on the interval length $h_j$ and the lower limit $x_{i,j}$, which dictate the polynomial coefficients $a$, $b$, and $c$ as implied by (2)–(4).

Utilizing the absolute error bound in $[x_{i,j}, x_{i,j+1})$, offered by (28), it is obtained that the relative error for $x \in [x_{i,j}, x_{i,j+1})$ is

$$\varepsilon_r(x) = \frac{\varepsilon_{\text{tot}}(x_{i,j})}{\sin x} \le \frac{\varepsilon_{\text{tot}}(x_{i,j})}{\sin x_{i,j}} \qquad (29)$$

because $\sin x_{i,j} \le \sin x$, when $0 < x_{i,j} \le x < x_{i,j+1} < (\pi/2)$.

To guarantee accuracy within half an ulp, the absolute error should satisfy

$$\varepsilon_{\text{tot}}(x_{i,j}) \le \frac{1}{2}\,2^{-23}2^{\lfloor \log_2 \sin x_{i,j} \rfloor} \qquad (30)$$

as $\lfloor \log_2 \sin x_{i,j} \rfloor$ is the minimum exponent of $\sin x$ in $[x_{i,j}, x_{i,j+1})$.

The algorithm of Fig. 2 derives the sequence $h_j$ so that the resulting partition defined by $x'_j$ and $h_j$ leads to $x_{i,j}$ values that satisfy (30), and thus, it is guaranteed that the half-ulp error constraint is met throughout the entire interval $[0, (\pi/2))$.

It is now proven that increased accuracy of the coefficients $a$ and $b$, and of the addition, increases the accuracy of the complete polynomial evaluation. Initially it is shown that in each of the two floating-point multiplication–addition operations, the product is aligned to the added coefficient. It can be shown that $b > |c|u$, when

$$x < \tan^{-1}\frac{\sin h_l}{1 - \cos h_l} \qquad (31)$$

where $h_l$ denotes the largest $h_j$ used. For the last interval of Table I, where $h_l = 2^{-7}$, it is found that $x < 1.566\,89$; hence, the last interval does not satisfy the condition (31). However, since the maximum value of $u$ in the last interval, is $u_{\max} = (\pi/2h_l) - \lfloor \pi/2h_l \rfloor = 0.0619$, it holds that $|c|u_{\max} = 1.89 \times 10^{-6} < b$. Therefore $cu$ is always aligned to $b$. Also, $a > b + cu > (b + cu)u$, when $x > \tan^{-1}\sin h_0$. In case $h_0 = 2^{-15}$, $x > \tan^{-1}\sin 2^{-15} = 0.000\,03$, which falls outside the range of interest, $2^{-12} \le x < (\pi/2)$. Therefore, in the floating-point adder, it is the product that has to be aligned to the coefficients $b$ and $a$.

The alignment consists of down shifting the significand of the product while introducing zeros at the most significant bit positions, until the exponents of the product and the coefficient are identical. Therefore, bits of the denormalized product, beyond the 24th position, assume correct values and, if the coefficients $a$ and $b$ are of appropriate word length, no accuracy is lost. In particular, if the coefficients $a$ and $b$ are available with more than 23 fractional bits, the 27-bit fixed-point adder, which is the basis of the floating-point adder, can be exploited to deliver higher accuracy, only at the cost of replacing the three least significant 1-bit half adders with full adders. The hardware cost of extending the precision in this way is practically negligible.

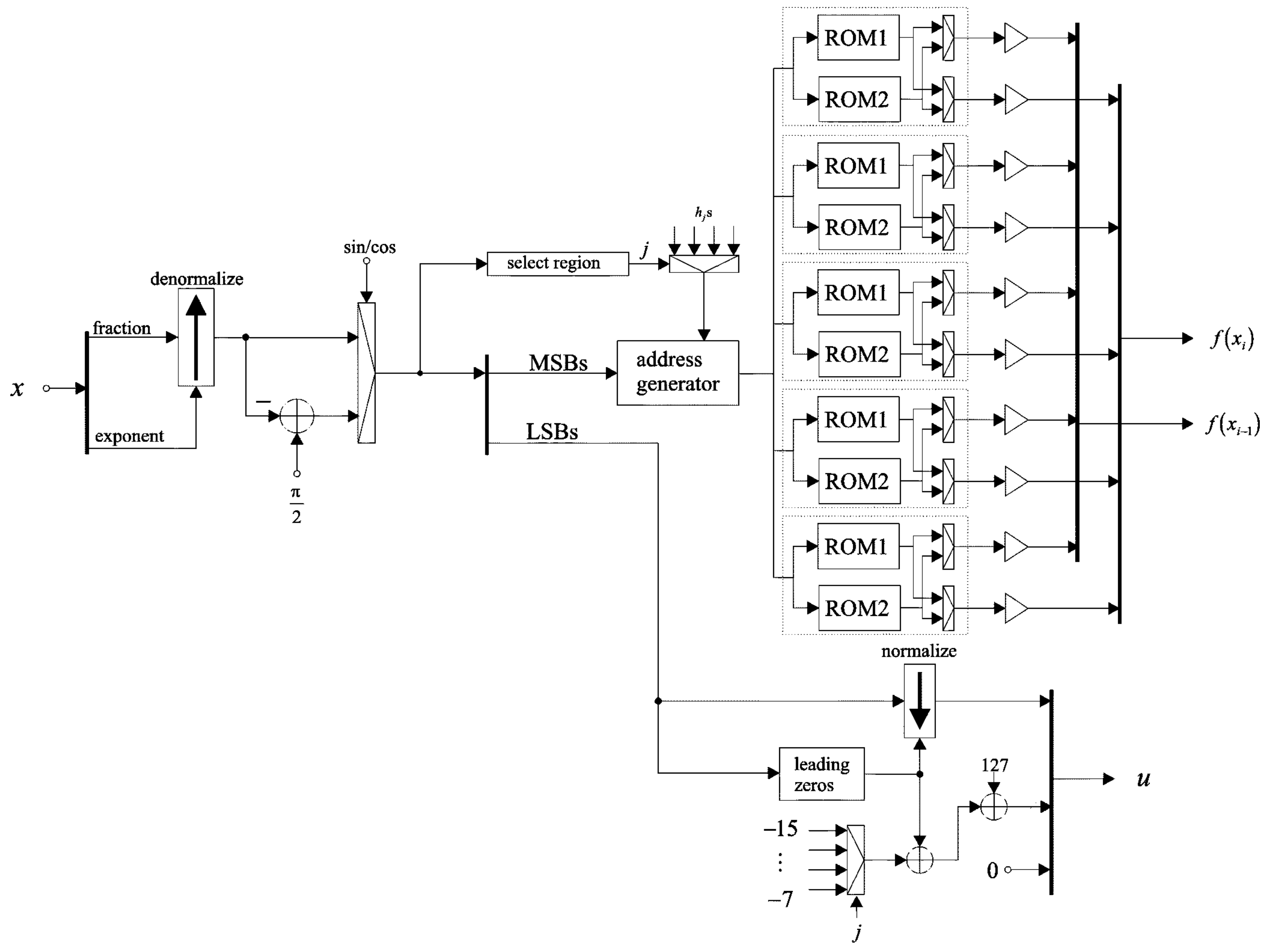


Fig. 4. Input decoding, polynomial argument generation, and memory organization.

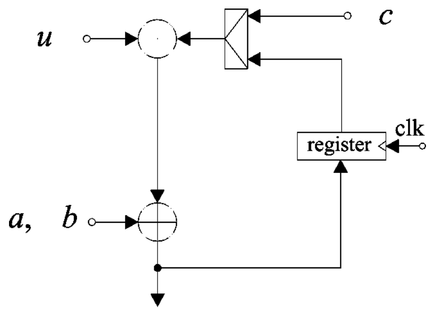


Fig. 5. Computation of the second-order polynomial.

## III. The Proposed Architecture

The floating-point sine/cosine architecture consists of the following building blocks:

1) the input decoding and table lookup unit, shown in Fig. 4;
2) the coefficient evaluation unit, shown in Fig. 3;
3) the floating-point polynomial evaluation unit of Fig. 5.

The organization of the sine/cosine polynomial coefficient computation unit is discussed below. The argument $x$ assumed to be in the interval $[2^{-12}, \pi/2]$, is received by the input decoder. The argument is converted from its IEEE 754-compatible format to a 36-bit fixed-point number. The input denormalizer down shifts the significand $x_{\text{frac}}$ of $x$ as many positions as dictated by the exponent of $x$, $E_x$. Subsequently, if cosine is to be computed, the derived number is subtracted from the quantity $\pi/2$. Let $x'$ denote the actual argument of which the sine is to be computed and which occurs from the denormalization and, if necessary, the subtraction from $\pi/2$. The address generator receives the 16 most significant bits of $x'$ and it computes in parallel the differences

$$d_j = x' - \hat{x}'_j \tag{32}$$

where $j = 0, 1, \cdots, 5$ and $\hat{x}'_j$'s are the lower bounds of the intervals obtained by the merging procedure as shown in Table I. The sign bits of the differences $d_j$ can be decoded as shown in Table IV to assign the most significant part of the appropriate difference to the address $A$. The decoded sign bits also select the appropriate $h_j$.

The $35 - |\log_2 h_j|$ least-significant bits are normalized and the exponent is corrected by the corresponding value to form the polynomial argument $u$, in floating-point format. While a multiplication without hidden-bit encoding is possible, it is preferred to encode $u$ into a legitimate IEEE 754 format. This choice imposes extra hardware to re-encode the mantissa of $u$ and appropriately correct its exponent, but it removes a more complex nor-

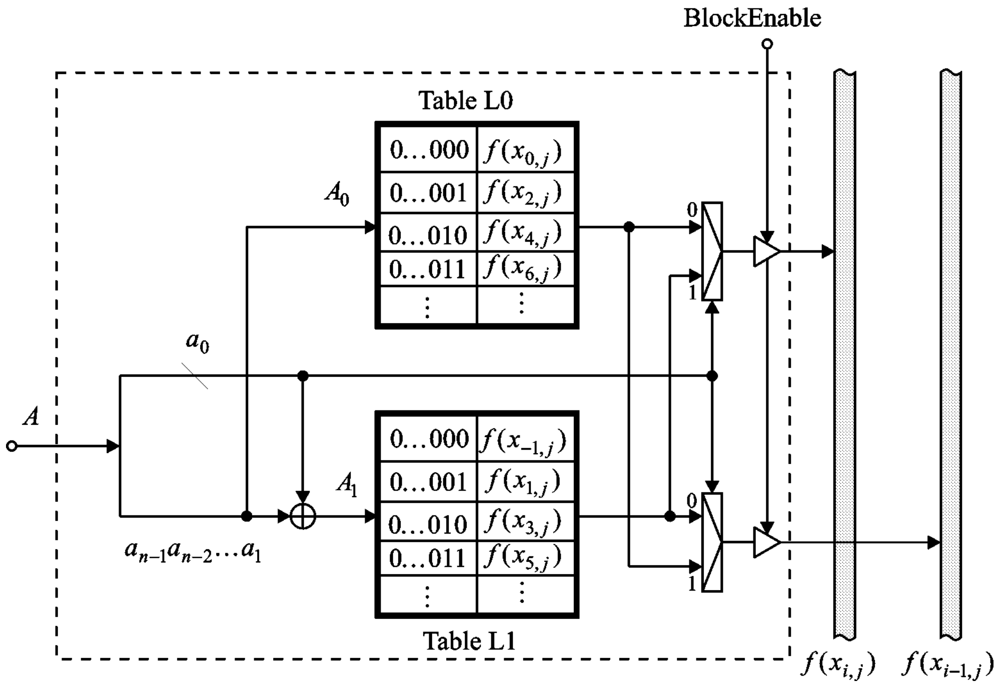


Fig. 6. Organization of an LUT unit composed of two tables L0, L1, a reduced adder, and multiplexing logic. Notice that $f(x_{-1,j}), x_{-1,j} = x_{0,j} - h_j$, is also stored.

TABLE IV
DECODING OF SIGN BITS TO GENERAGE THE ADDRESS $A$

| sign bits | $d_j$ assigned to $A$ |
|---|---|
| 00000 | $d_5$ |
| 00001 | $d_4$ |
| 00011 | $d_3$ |
| 00111 | $d_2$ |
| 01111 | $d_1$ |
| 11111 | $d_0$ |

malization circuitry from the IEEE 754-compliant multiplier. In this way, delay is removed from the critical path, usually defined by the multiplier.

The memory banks, which contain the stored sine values, are organized into six look-up table (LUT) units. Each LUT unit corresponds to a region $[\hat{x}'_j, \hat{x}'_{j+1})$. Regions of few samples have been merged to reduce address generation complexity at a small memory size increase. Each of the LUT units can be organized as shown in Fig. 6, so that duplication of stored data is avoided. Each LUT unit is composed of two physically separated ROM tables, denoted as L0 and L1, which are simultaneously accessed. The table L0 contains the interpolated function samples $f(x_{0,j}),\ f(x_{2,j}),\ \cdots,\ f(x_{2k,j})$, while table L1 contains the function samples $f(x_{-1,j}), f(x_{1,j}),\ \cdots,\ f(x_{2k-1,j})$. The addresses $A0$ and $A1$, which correspond to L0 and L1, respectively, are derived from the address $A = a_{n-1}a_{n-2}\cdots a_0$ generated by the preprocessor as follows. The least significant bit of $A$, $a_0$, is added to the remainder of the address word $a_{n-2}a_{n-1}\cdots a_1$ to generate $A1$, i.e.,

$$A1 = a_{n-1}a_{n-2}\cdots a_1 + \underbrace{0\cdots 0}_{n-2}a_0.$$

The address $A0$ consists of the most significant part of $A$, i.e., $A0 = a_{n-1}a_{n-2}\cdots a_1$. The relation of the addresses $A0$ and $A1$ to $A$ is shown in Table V. The least significant bit of the address word, $a_0$, is also used to select which word will be assigned to $f(x_{i,j})$ and which one to $f(x_{i-1,j})$. In particular, when $a_0 = 0$, the sample from L0 is assigned to $f(x_{i,j})$, and the sample from L1 is assigned to $f(x_{i-1,j})$. The assignments are reversed when $a_0 = 1$. The described LUT unit architecture permits fetching in parallel two consecutive data words without the need of storing duplicated data.

After the function samples have been fetched, the floating-point polynomial coefficients are computed, using simple fixed-point arithmetic, as shown in Fig. 3.

The coefficient evaluation unit operates as follows. Initially, the two words fetched from the memory banks are denormalized to form 38-bit fixed-point quantities. The remainder of the operations are performed on these fixed-point numbers. A multiplier with the constant 341 produces the 10-bit value $341 \sin x_{i,j}$ organized into nine integer bits and a fractional one. This product is utilized in both $b$ and $c$ computation. To compute $b$, the product is multiplied by $h_j^4$, i.e., down shifted by $|\log_2 h_j^4| = 4|\log_2 h_j|$ bit positions. The resulting value is represented with 26 fractional bits. It is then down shifted by 13 positions and added to the 38-bit fixed-point value of $\sin x_{i,j}$. The final result $b$ occurs as the difference of the derived sum from a value computed in parallel as follows. In parallel, $\sin x_{i,j}$ is down shifted $2|\log_2 h_j| + 1$ positions to be added to $\sin x_{i-1,j}$. The difference is then converted to the floating-point representation $m_b 2^{E_b}$.

To compute $c$, the seven most significant bits of the product $341 \sin x_{i,j}$ are down shifted $2|\log_2 h_j|$ positions and the 16 fractional bits of this partial result are added to the 29-bit quantity $\sin x_{i,j} h_j^2$, which has been up shifted ten positions. The

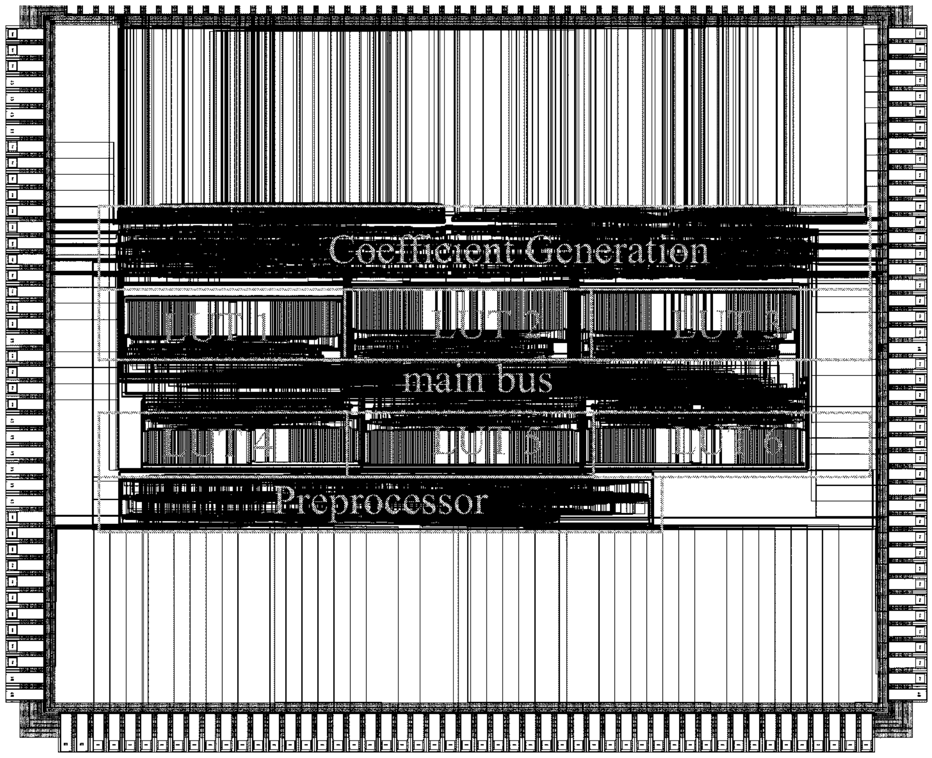


Fig. 7. Organization of the proposed sine/cosine processor layout. Notice that each LUT unit contains two ROM macrocells.

TABLE V
RELATION OF THE ADDRESSES $A0$, $A1$ OF THE TABLES L0 AND L1 TO THE ADDRESS $A$ GENERATED AT THE PREPROCESSOR

| Address $A$ | $A0$ | $A1$ |
|---|---|---|
| 000 | 00 | 00 |
| 001 | 00 | 01 |
| 010 | 01 | 01 |
| 011 | 01 | 10 |
| 100 | 10 | 10 |
| ⋮ | ⋮ | ⋮ |

obtained 29-bit sum is down shifted 14 positions and subtracted from $\sin x_{i,j}$. A normalizer produces the mantissa and an intermediate exponent. The final exponent occurs by reducing the intermediate one by the quantity $2|\log_2 h_j|+1$. Coefficient $c$ is obtained in the floating-point format $m_c 2^{E_c}$, after altering the sign of the mantissa, because $c < 0$. The word lengths of the various intermediate results have been minimized through exhaustive simulation, which is feasible due to the limited number of possible coefficients.

The sine value is evaluated in the floating-point polynomial evaluation unit, shown in Fig. 5, in two clock cycles. During the first cycle the computation $P = c \times u + b$ is performed, and in a second clock cycle, the operation $P \times u + a$ is executed.

The coefficients $b$ and $a$ are added to the corresponding products by a floating-point adder, in an extended precision mode. In this mode, a 27-bit mantissa addition is performed to guarantee that the sine and cosine are computed with sufficient accuracy. For compatibility with the IEEE 754 standard, the floating-point adder should also offer a single-precision mode.

## IV. VLSI IMPLEMENTATION, PERFORMANCE EVALUATION, AND COMPARISONS

The proposed architecture comprises the input denormalizer, a 36-bit subtractor from a constant, the address generator, an assortment of LUT units offering a total memory space of approximately 16 kbits, and logic for the computation of $a$, $b$, and $c$, while the function evaluation is performed in a floating-point MAC unit. The layout of the proposed sine/cosine processor can be efficiently organized as shown in Fig. 7. The particular implementation does not contain a floating-point MAC and it can fit into a bounding box of area less than 26 mm$^2$ in a double-metal 0.7-$\mu$m CMOS technology. The design is build using a Europractice standard-cell library and the corresponding ROM macro-cells [8] by following a design methodology based on VHDL and synthesis tools. The lower part of the layout comprises the preprocessor, i.e., the input decoder, $\sin x/\cos x$ selection, the polynomial argument $u$ generator, and the address generator. The assortment of the LUT's are placed at both sides of a main bus composed of two data busses [for $f(x_{i-1,j})$ and $f(x_{i,j})$] and an address bus. The upper part of the layout contains the coefficient generation circuitry. The presented layout is characterized by the bus-based organization, which is preferred as it keeps the routing area cost relatively low. In fact, the particular approach requires approximately 30% less area, when compared to a multiplexed architecture of local interconnections. The particular layout contains approximately 8500 equivalent gates and a total of 16 kbit of ROM, where an equivalent gate occupies area equal to that of a 2-input NAND gate in the particular library. The particular implementation of the processor exhibits a delay of 182 ns, measured by means of the static timing analyzer QuickPath (from

Mentor Graphics). By altering the micro-architecture of the various adders, subtractors, and shifters of which the processor is composed, a variety of area/time performance targets may be met. For example, by applying a timing optimization procedure, delays of less than 90 ns have been achieved, at the cost of increasing the hardware complexity to approximately 10 500 eq. gates. To facilitate delay comparisons, it is stated that the delay of a 8500-gate implementation equals that of a IEEE-754 single-precision multiply-add unit, which has been designed in this technology for comparison purpose and with the same design methodology (VHDL/synthesis). Further performance improvement in an area/time sense is anticipated, if a handcraft design approach is adopted. However, the benefits of the layout organization in Fig. 7 do not depend on the implementation style (standard-cell or handcraft layout); hence, it can be followed even if a standard-cell design style is not adopted. The delay of the polynomial coefficient evaluation equals to the delay of a MAC; hence the sine evaluation is completed within the delay of $3t_{\mathrm{MAC}}$, where $t_{\mathrm{MAC}}$ is the delay of a single-precision floating-point MAC. The achieved accuracy is always better than half an ulp.

The performance of the proposed architecture is compared to the performance of a few of the methods and architectures met in the literature, and that refer to the evaluation of single-precision floating-point $\sin x$. Das *et al.* propose a scheme for evaluating four commonly used functions with the help of an LNS processor [10]. The sine function is computed in $82t_{\mathrm{fpa}}$, where $t_{\mathrm{fpa}}$ is the delay of a fixed-point addition. Assuming that $t_{\mathrm{fpa}}$ is $10t_g$, where $t_g$ is the gate delay, the overall delay for the evaluation of the sine is approximately $5t_{\mathrm{MAC}}$, excluding the time required for the conversions between the floating-point and the LNS representations. The accuracy obtained is not guaranteed to be within half an ulp. Wong and Goto have introduced the Add-Table lookup-Add (ATA) method for single-precision floating-point evaluation of elementary functions [2]. In this case 2.5 Mbits of memory space and 3000 gates of additional logic are required for the evaluation of sine. The delay reported is $26.6t_g$, assuming a memory access of $6t_g$. However, the access delay for the required lookup tables when implemented in the adopted technology becomes approximately $100t_g$; therefore, in the ATA method a sine is computed in less than $1t_{\mathrm{MAC}}$. Elguibaly [11] has proposed an adaptive CORDIC algorithm ($\alpha$-CORDIC) which exploits the existence of a multiplier within a processor. The subject of the hardware requirements of an ASIC for the implementation of the algorithm is not discussed in [11]; however, besides the multiplier, a set of LUT's is required. Even though the utility of $\alpha$-CORDIC, as that of any CORDIC, lies in its generality and flexibility the time required for the calculation of a trigonometric function is much longer than the time achieved by the processor proposed in this paper.

Hence, compared to [10], the proposed architecture is much faster, achieves better accuracy and does not impose a conversion overhead. Compared to [2] the proposed architecture is advantageous when utilized in a processor which already contains a floating-point MAC, as it requires approximately 160 times less memory space. Finally, compared to [11], the algorithm implemented by the proposed architecture is very much faster.

## V. Conclusion

In this paper, a computational technique and its VLSI architecture that allows the evaluation of sine and cosine by an ordinary floating-point multiply–accumulate datapath with an incremental hardware overhead has been presented. By exploiting properties of the trigonometric functions, the computational complexity of determining the polynomial coefficients has been reduced. Furthermore, the partitioning of the approximation interval into regions of unequal lengths reduces the memory space required to achieve the half an ulp accuracy. The proposed algorithm exhibits high utilization of hardware resources existing in a processor, i.e., the floating-point multiplier and the adder. The accuracy, low memory requirements, and high speed offered by the proposed technique suggest that it may be useful in applications that rely on floating-point arithmetic and dedicated hardware support, such as 3-D computer graphics.

## References

[1] I. Koren and O. Zinaty, "Evaluating elementary functions in a numerical coprocessor based on rational approximations," *IEEE Trans. Comput.*, vol. 39, pp. 1030–1037, Aug. 1990.

[2] W. F. Wong and E. Goto, "Fast evaluation of the elementary functions in single precision," *IEEE Trans. Comput.*, vol. 44, pp. 453–457, Mar. 1995.

[3] P. T. P. Tang, "Table-lookup algorithms for elementary functions and their error analysis," in *Proc. 10th Symp. Computer Arithmetic*, 1991, pp. 232–236.

[4] P. W. Baker, "Suggestion for a fast binary sine/cosine generator," *IEEE Trans. Comput.*, vol. 25, pp. 1134–1136, Nov. 1976.

[5] V. Kantabutra, "On hardware for computing exponential and trigonometric functions," *IEEE Trans. Comput.*, vol. 45, pp. 328–339, Mar. 1996.

[6] D. M. Lewis, "Interleaved memory function interpolators with application to an accurate LNS arithmetic unit," *IEEE Trans. Comput.*, vol. 43, pp. 974–982, Aug. 1994.

[7] J. H. Mathews, *Numerical Methods for Mathematics, Science, and Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[8] *European Silicon Structures (Library Manual)*, ECPD07 CMOS Digital Library, 1992.

[9] V. Paliouras, K. Karagianni, and T. Stouraitis, "Error bounds for floating-point polynomial interpolators," *Electron. Lett.*, vol. 35, pp. 195–197, Feb. 1998.

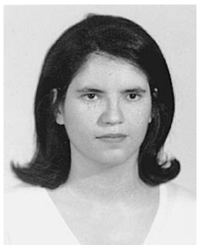
[10] D. Das, K. Mukhopadhyaya, and B. P. Sinha, "Implementation of four common functions on an LNS co-processor," *IEEE Trans. Comput.*, vol. 44, pp. 155–161, Jan. 1995.

[11] F. Elguibaly, "$\alpha$-CORDIC: An adaptive CORDIC algorithm," *Can. J. Elect. Comput. Eng.*, vol. 23, no. 3, pp. 133–138, 1998.

**Vassilis Paliouras** (S'90–M'92) received the Diploma degree in electrical engineering (*magna cum laude*) in 1992, and the Ph.D. degree in 1999, both from the University of Patras, Greece.

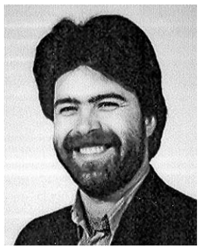
His research interests include computer arithmetic, DSP architectures, and VLSI design methodologies. He has published more than 20 articles. He currently serves in the Greek Army.

In 1997, Dr. Paliouras received the MEDCHIP Award on VLSI design. He is a Member of the Technical Chamber of Greece.



**Konstantina Karagianni** (S'93–M'98) was born in Kitimat, B.C., Canada, in 1968. She received the Diploma in electrical engineering from the University of Patras, Greece, in 1992. She is currently working toward the Ph.D. degree in the Electrical and Computer Engineering Department, University of Patras, Greece.

Her research interests include VLSI architectures for DSP and control applications, methodologies for mapping algorithms onto parallel processors, development of CAD tools, and computer arithmetic. She is a Member of the Technical Chamber of Greece.



**Thanos Stouraitis** (S'82–M'86–SM'97) received the B.S. degree in physics and the M.S. degree in electronic automation from the University of Athens, Greece, in 1979 and 1981, respectively, the M.S. degree in electrical engineering from the University of Cincinnati, OH, in 1983, and the Ph.D. degree from the University of Florida, Gainesville, in 1986.

He is a currently a Professor with the Electrical and Computer Engineering Department, University of Patras, Greece. He has served on the faculty of the University of Florida and The Ohio State University. He has published two books, several book chapters, and about 30 journal and 70 conference papers in the areas of computer architecture, computer arithmetic, VLSI signal and image processing, and low-power processing.

Dr. Stouraitis is currently the Chair of the IEEE Circuits and Systems (CAS) Society's Technical Committee on VLSI Systems and Applications, in addition to serving on the Digital Signal Processing and the Multimedia Systems Committees. He has served as the General Chair of the IEEE Third International Conference on Electronics, Circuits, and Systems (ICECS'96), Co-Program Chair for EUSIPCO'98, and regularly serves on the Program Committee of various CAS and Signal Processing conferences. He was awarded the Outstanding Ph.D. Dissertation Award of the University of Florida and a Certificate of Appreciation by the IEEE CAS Society in 1997.