World Academy of Science, Engineering and Technology
International Journal of Electrical and Computer Engineering
Vol:1, No:9, 2007

Evaluating Sinusoidal Functions by a Low Complexity Cubic Spline Interpolator with Error Optimization

Abhijit Mitra and Harpreet Singh Dhillon

***Abstract*—**We present a novel scheme to evaluate sinusoidal functions with low complexity and high precision using cubic spline interpolation. To this end, two different approaches are proposed to find the interpolating polynomial of $\sin(x)$ within the range $[-\pi,\ \pi]$. The first one deals with only a single data point while the other with two to keep the realization cost as low as possible. An approximation error optimization technique for cubic spline interpolation is introduced next and is shown to increase the interpolator accuracy without increasing complexity of the associated hardware. The architectures for the proposed approaches are also developed, which exhibit flexibility of implementation with low power requirement.

***Keywords*—**Arithmetic, spline interpolator, hardware design, error analysis, optimization methods.

I. Introduction

Evaluating basic functions like sinusoids is an important part of many applications such as scientific computations, signal processing and computer graphics. Although software routines are available for the computation of these functions using standard floating point instructions, application-specific hardware/firmware combination is preferred to that for minimizing the cost and power of these implementations. Therefore, the hardware support for the evaluation of these functions is always an integral part of the computation. Depending upon the nature of the application, e.g., fast evaluation of these functions, low power architecture or cost constraint on the architecture, a wide variety of hardware algorithms have been developed. The most commonly used among them are COordinate Rotation DIgital Computer (CORDIC) based techniques [1], [2], linear approximation algorithms [3], and general polynomial [4] or even rational approximations [5]. A variety of look-up table methods [6] have also been applied to develop efficient architectures for approximating elementary functions. Among these, two parallel table look-ups to obtain a carry save function approximation [7], using small multipliers with small tables [8] and combining table look-up with an enhanced minimax quadratic approximation [9] are different important techniques. Apart from these, the popular memory interleaving schemes [10] are also employed for the fast evaluation of these where polynomial interpolation [11] is used to approximate the value of the function. In [12], one can find a survey of the well-known techniques for computing, among other trigonometric and exponential functions, the sinusoidal function. However, many present day applications require low power digital sinusoidal generators [13], which, in turn, call for hardware algorithms leading to optimizing the memory required by reducing the size of look-up tables for these sinusoidal generators. In this paper one such low power and low complexity scheme for generation of sinusoidal functions is presented with two different approaches, both with only a few number of memory elements. The proposed scheme is mainly based on cubic spline interpolation [14] and one of our earlier work [15] where a prototype of sinusoidal evaluation is given. However, the work [15] lacks to enhance the system precision along with flexibility of implementation issues. We properly address all these issues here. The main virtue of the present scheme is to propose a low power scheme with respect to the number of stored interpolator coefficients as well as to introduce a simplified hardware structure with high precision that arises from an error optimization technique.

The proposed approach especially computes the sine of a number (in radian) in the range $[-\pi, \pi]$. Since $\sin(\pi - x) = \sin(x)$ and $\sin(\pi + x) = -\sin(x)$, the interval range can be shortened to $[-\pi/2, \pi/2]$. Further, as $\sin(-x) = -\sin(x)$, the approximation interval may effectively be reduced to $[0,\ \pi/2]$ [16]. The task is then to find an interpolating polynomial for $\sin(x)$ in the interval $[0,\ \pi/2]$ with certain data points in this range. In order to have a low power scheme, the number of these data points should be kept as low as possible so that the interval $[0,\ \pi/2]$ is sub-divided into a small number of intervals to store only a few coefficients of the interpolating polynomial. It is shown that by using a cubic spline interpolation, storing the value of sine only at one point in between the interval $(0,\ \pi/2)$ is enough to have a precise approximation of the function. This leads to an efficient low power, low complexity hardware architecture. To this end, firstly, equal intervals are chosen for interpolation with the chosen data point as $x = \pi/4$ which comes out to be a sub-optimal choice. An error optimization technique is then taken up to find the proper data point such that the approximation error is nearly optimized to extend the accuracy. With this, atleast 1-bit increase in precision is achieved without any additional increment in the hardware complexity. A similar investigation is also carried out by taking two data points in the interval $(0,\ \pi/2)$ (i.e., considering three equal intervals for interpolation). This increases the accuracy, but hardware complexity is also increased as compared to the first case. Here too, the introduction of the error optimization is shown to yield a 1-bit increase in accuracy without any further increase

Manuscript received September 9, 2007.

The authors are with the Department of Electronics and Communication Engineering, Indian Institute of Technology (IIT) Guwahati 781 039, India. E-Mail: (a.mitra, harpreet) @iitg.ernet.in.

in hardware complexity.

The paper is organized as follows. In Section 2, fundamentals of cubic spline interpolation are explained briefly. Section 3 introduces the proposed schemes for both two and three interval interpolations. The error analysis and corresponding high-precision hardware architectures are given in Sections 4 and 5 respectively. The paper is concluded in Section 6 by summarizing the main concepts introduced herein.

II. Fundamentals of Cubic Spline Interpolation

Cubic spline function is basically a cubic polynomial that interpolates the data in several intervals with properly chosen coefficients such that the continuity of the function is not broken. In other words, it can be written as a picewise function of the form:

$$S(x) = \begin{cases} s_1(x) & \text{if} \quad x_1 \le x < x_2 \\ s_2(x) & \text{if} \quad x_2 \le x < x_3 \\ & \cdots \\ s_{n-1}(x) & \text{if} \quad x_{n-1} \le x < x_n \end{cases} \tag{1}$$

where any $s_i(x)$, $i = 1, 2, ..., n-1$, is a third degree polynomial defined by

$$s_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \tag{2}$$

with a_i, b_i, c_i and d_i being the coefficients of the cubic polynomial in the interval $[x_i, x_{i+1})$. The first and second derivatives of these $n-1$ equations are important in this process and they are

$$s_i'(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i, \tag{3}$$

$$s_i''(x) = 6a_i(x - x_i) + 2b_i \tag{4}$$

for $i = 1, 2, ..., n-1$. The basic properties of such cubic spline functions along with their suitability in curve fitting technique is given below in brief.

A. Properties of spline functions

$S(x)$ is defined to be a *spline function* of order $m \ge 1$ if it satisfies the following two properties:

1) $S(x)$ is a polynomial of degree $< m$ on each interval $[x_{i-1},\ x_i]$;
2) The r^{th} derivative, $S'^{(r)}(x)$, is continuous in the whole range $[x_1,\ x_n]$, for $0 \le r \le m-2$.

The derivative of a spline of order m is a spline of order $m-1$. From the above definition, cubic spline comes out to be a spline function of order $m = 4$. Therefore, it should conform to the following properties:

1) The piecewise function $S(x)$ should interpolate all the data points;
2) $S(x)$, $S'(x)$, $S''(x)$ should be continuous in the interval $[x_1,\ x_n]$.

With the help of these two properties, cubic splines are efficiently used for curve fitting. However, it can be shown that this system is under-determined. In order to generate a unique cubic spline, two other conditions must be imposed on the system [14]. According to these conditions, there can be many different types of splines, some of which are: natural spline, parabolic runout spline, cubic runout spline, periodic spline, clamped spline etc. Intersted readers can get a good survey of these in [17].

Unique cubic spline interpolation technique is always preferred for curve fitting purpose because it can be shown that data interpolated by a spline behaves more or less like the original function. The main virtue of the spline as a data correlation tool is its consistency and efficiency which we demonstrate in our proposed scheme.

III. The Proposed Scheme

The proposed algorithm computes the sine of a number in the range $[0,\ \pi/2]$. To have a low power architecture, number of the fixed data points in the interval should be as small as possible. It is shown that by using cubic spline interpolation we can get a precise approximation of sine by using only one data point (say $x = k$) in the interval $(0,\ \pi/2)$. To this end, equal intervals are taken for interpolation (i.e., the data point is chosen to be $k = \pi/4$) at first [15]. An error optimization method is then introduced to find a proper k which minimizes the approximation error involved. This data point comes out to be $k = 0.877$ radian. For more precise approximation, we consider another case with three-interval interpolation where two data points (say $x = k$ and $x = l$) are taken in the interval $(0,\ \pi/2)$. Firstly, equal intervals are chosen for interpolation. The data points in this case are $k = \pi/6$ and $l = \pi/3$. The approximation error is optimized next and the data points are calculated to be $k = 0.634$ and $l = 1.115$ radian, which clearly depicts that optimization leads to unequal interval interpolation.

A. Two-interval interpolation

In this case, the interpolating function takes the form:

$$S(x) = \begin{cases} s_1(x) & \text{if} \quad 0 \le x < k \\ s_2(x) & \text{if} \quad k \le x \le \pi/2 \end{cases} \tag{5}$$

where $s_i(x)$ is a third degree polynomial defined by (2) for $i = 1$ and 2, i.e., we get two cubic polynomials, one for each interval. To determine this cubic spline uniquely, we need to find all the eight coefficients. By using the properties of the cubic spline [14], we get the following equations:

$$s_1(0) = 0,\ s_1(k) = \sin(k),\ s_2(k) = \sin(k),\ s_2(\pi/2) = 1; \tag{6}$$

$$s_1'(k) = s_2'(k),\ s_1''(k) = s_2''(k). \tag{7}$$

With these six known values, the system becomes under-determined as number of unknowns is eight. To determine the cubic spline uniquely, two more conditions are required that will basically define the behaviour of the interpolating function at the end points. These conditions can be derived from the nature of the sine curve. There can be many possible choices, but the conditions used in this paper are:

$$s_1'(0) = 1,\ s_2'(\pi/2) = 0 \tag{8}$$

as they provide a simple form of implementation in terms of hardware.

With the help of (6)-(8), we get a unique interpolating spline function. Taking the data point as $k = \pi/4$ this function is shown in Fig. 1.

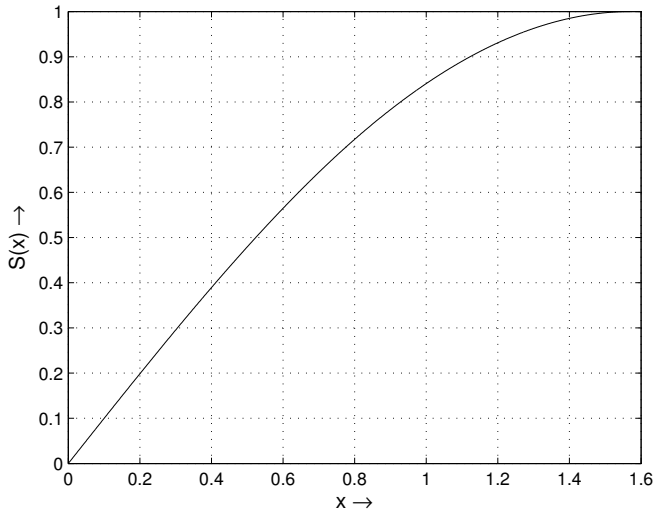

Fig. 1. Plot of $S(x)$ versus x in case of two-interval interpolation.

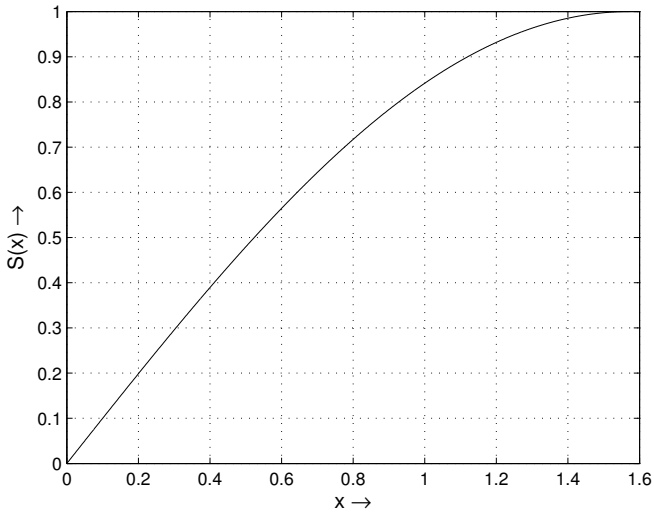

Fig. 2. Plot of $S(x)$ versus x in case of three-interval interpolation.

B. Three-interval interpolation

In this case, the interval $[0,\ \pi/2]$ is sub-divided into three intervals by two data points $x = k$ and $x = l$. Hence the interpolating spline function takes the form:

$$S(x) = \begin{cases} s_1(x) & \text{if} \quad 0 \le x < k \\ s_2(x) & \text{if} \quad k \le x < l \\ s_3(x) & \text{if} \quad l \le x \le \pi/2 \end{cases} \tag{9}$$

where any $s_i(x)$ is a third degree polynomial defined by (2) for $i = 1$, 2 and 3, i.e., we get three cubic polynomials, one for each interval. To determine this cubic spline uniquely, we need to find all the twelve coefficients. By using the properties of the cubic spline, we get the following equations:

$$s_1(0) = 0,\ s_1(k) = \sin(k),\ s_2(k) = \sin(k),$$
$$s_2(l) = \sin(l),\ s_3(l) = \sin(l),\ s_3(\pi/2) = 1; \tag{10}$$
$$s_1'(k) = s_2'(k),\ s_1''(k) = s_2''(k),\ s_2'(l) = s_3'(l),\ s_2''(l) = s_3''(l). \tag{11}$$

As we get ten known values while the number of unknowns is twelve, it is necessary to find two more conditions. Like the previous case, here also we exploit the nature of sine function to find these equations as follows

$$s_1'(0) = 1,\ s_3'(\pi/2) = 0. \tag{12}$$

With the help of (10)-(12), we calculate all the unknowns and hence get a unique interpolating cubic spline function. Taking the data points to be $k = \pi/6$ and $l = \pi/3$, this cubic spline is shown in Fig. 2. Although this leads to a fairly precise interpolation scheme of $\sin(x)$, the associated precision can be further increased by introducing an optimization method for the approximation error, which is taken up next.

IV. Optimization of Approximation Error

The absolute approximation error for any general case of n-interval interpolation can be defined as:

$$\delta(x) = |S(x) - \sin(x)| \tag{13}$$

with $\delta(x)$ being a piecewise function that can be written as

$$\delta(x) = \begin{cases} |s_1(x) - \sin(x)| & \text{if} \quad x_1 \le x < x_2 \\ |s_2(x) - \sin(x)| & \text{if} \quad x_2 \le x < x_3 \\ & \ldots \\ |s_n(x) - \sin(x)| & \text{if} \quad x_{n-1} \le x < x_n \end{cases} \tag{14}$$

where each data point x_i, $i = 2, 3, ..., n-1$, is associated with two intervals, namely, $[x_{i-1}, x_i)$ and $[x_i, x_{i+1})$. For our case, x_1 and x_n are taken to be two fixed points as 0 and $\pi/2$ respectively. In order to optimize the approximation error $\delta_i(x)$, where $\delta_i(x) = |a_i(x-x_i)^3 + b_i(x-x_i)^2 + c_i(x-x_i) + d_i - \sin(x)|$, within the interval $[x_i, x_{i+1})$, let us assume that it has a global maxima $\delta_i(x_M)$ at $x = x_M$. Shifting the data point x_i by a small amount ϵ (> 0) towards x_{i+1} makes the intervals uneven and the error associated with $[x_i + \epsilon, x_{i+1})$ can then be written as

$$\widetilde{\delta}_i(x) = |\widetilde{a}_i(x - x_i - \epsilon)^3 + \widetilde{b}_i(x - x_i - \epsilon)^2 + \widetilde{c}_i(x - x_i - \epsilon) + \widetilde{d}_i - \sin(x)| \tag{15}$$

which will again have a global maxima $\widetilde{\delta}_i(x_N)$ at some $x = x_N$. Note that the values of coefficients normally change while shifting the end point and thus are denoted by '~' on their heads. Our aim is to find out a definite relation between $\delta_i(x_M)$ and $\widetilde{\delta}_i(x_N)$ which is given through a few remarks and corollaries in the following.

Remark 1: It can be shown that neglecting the effect of change of the coefficients (a_i, b_i, c_i, d_i) for a small ϵ (> 0), the approximation error decreases with decreasing the interval length, when $s_i(x) > \sin(x)$.

Corollary 1: It directly follows from *Remark 1* that $\delta_i(x_M) > \widetilde{\delta}_i(x_N)$. Similarly, it can be shown that for the intervals $[x_i, x_{i+1})$ and $[x_i, x_{i+1} + \epsilon)$, if $\delta_i(x_M)$ and $\widetilde{\delta}_i(x_Q)$ denote the global maxima within these intervals respectively, then $\widetilde{\delta}_i(x_Q) \ge \delta_i(x_M)$.

Corollary 2: In general, it can be said that increasing the interval length increases the maximum approximation error

while the error is reduced by decreasing the interval length, for $s_i(x) > \sin(x)$ in general.

With the help of *Corollary 2*, we develop an error optimization technique for the proposed methods for shifting the storage point(s) optimally which is discussed in the following.

A. Two-interval interpolation

Considering the same expression of approximation error $\delta(x)$ as given in (13), where $S(x)$ is an interpolating polynomial defined by (5), we get a maximum associated error as 2^{-10} ($\approx 10^{-3}$) in the case of equal interval interpolation (i.e., $k = \pi/4$). This, in turn, indicates the results to be accurate to around 10 bits. The plot of $\delta(x)$ in this case is shown in Fig. 3 (with dash-dotted line). Adopting an error optimization technique as discussed in this section, a data point is found in the interval $(0, \pi/2)$ which leads to equal error in both the interpolation intervals, i.e.,

$$max(|s_1(x) - \sin(x)|) = max(|s_2(x) - \sin(x)|) \qquad (16)$$

where $max(f(x))$ gives the maximum value of function $f(x)$ in the specified interval. The data point in this case comes out to be $k = 0.877$ radian with the corresponding maximum error as 2^{-11} ($\approx 6 * 10^{-4}$) which is plotted in Fig. 3 (with solid line). This clearly shows to have achieved a 1-bit increase in accuracy without any increase in the complexity of the hardware scheme.

B. Three-interval interpolation

Here also, the approximation error is defined as (13) with $S(x)$ being an interpolating polynomial defined by (9). Firstly, equal intervals are chosen for interpolation (i.e., $k = \pi/6$ and $l = \pi/3$). Maximum error in this case comes out to be approximately 2^{-13} ($\approx 2{*}10^{-4}$) which is shown in the plot of $\delta(x)$ in Fig. 4 (with dash-dotted line). This ensures accuracy of approximately 13 bits. Approximation error is then optimized in the analogous lines of *Corollary 2* to find such data points which leads to equal error in all the interpolation intervals, i.e.,

$$max(|s_1(x) - \sin(x)|) = max(|s_2(x) - \sin(x)|) =$$

$$max(|s_3(x) - \sin(x)|) \qquad (17)$$

where $max(f(x))$ carries its usual meaning as given earlier. After this optimization, the data points are found to be $k = 0.634$ and $l = 1.115$ radian. The maximum error in this case is 2^{-14} ($\approx 10^{-4}$) which is shown in Fig. 4 (with solid line). Clearly, in this case also there is 1-bit increase in accuracy without any increase in the complexity of the circuit.

The proposed interpolation scheme can be considered a consistent one for its adaptive convergence to original function with only one/two data point(s), which, from storage view point, is nominal to implement in hardware. We provide a brief account of the hardware implementation of both the approaches next.

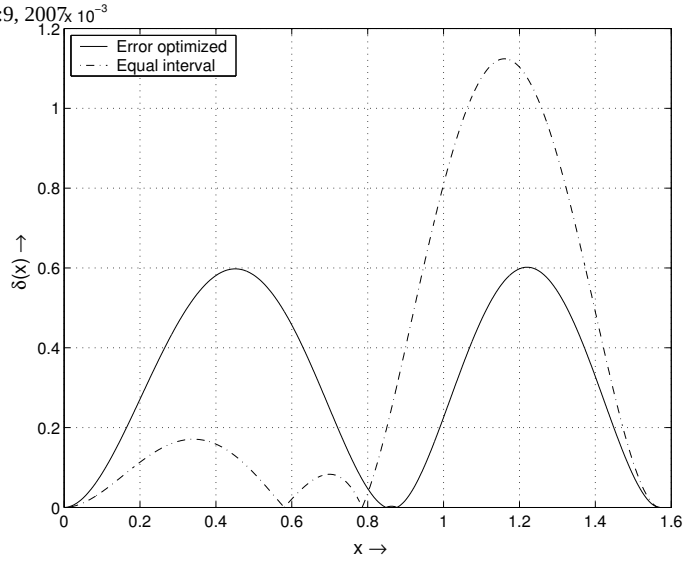

Fig. 3. Plots of $\delta(x)$ versus x for two-interval interpolation schemes.

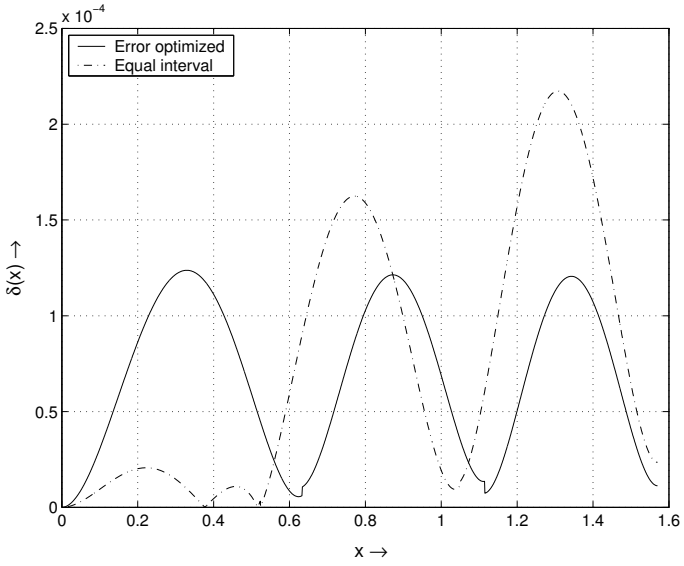

Fig. 4. Plots of $\delta(x)$ versus x for three-interval interpolation schemes.

V. Hardware Implementation of the Proposed Scheme

The main advantage of the proposed scheme stems from the fact that it can be easily realized in hardware, leading to a low complexity and low power architecture. Below, we provide the hardware realizations considering both two-interval and three-interval approaches. The structures of both the cases are more or less similar except a few extra addition or comparison operations.

A. Two-interval interpolation

The detailed hardware architecture for implementing two-interval interpolation scheme is shown in Fig. 5. This architecture can be mainly divided into three modules: preprocessor, argument generator and spline approximator.

1) Preprocessor (Module 1): The input to this module is $x \in [-\pi, \pi]$. As $\sin(-x) = -\sin(x)$, it passes $sign(x)$ to consequent modules to indicate whether $x \in [-\pi, 0)$ or $[0, \pi]$. $Abs(x)$ is then assigned to x which effectively reduces the interval of interest to $[0, \pi]$. Since $\sin(x) = \sin(\pi - x)$, the module thus further examines whether $x \in [0, \pi/2)$ or $[\pi/2, \pi]$ through a comparison operation taking the reference input to be $\pi/2$. If $x \in [0, \pi/2)$, it directly passes the value of x. Otherwise, $\pi - x$ is assigned to x thereby reducing the interval to $[0, \pi/2]$ in which the cubic spline function is defined. This module further examines whether x is a predefined data or not. The value of $\sin(x)$ is directly displayed if x is predefined, otherwise x is right shifted by one bit to get $[x] = x/2$.
2) Argument Generator (Module 2): This module finds the interval in which x lies by comparing the value of $[x]$ with the reference point $k/2$. As the data point associated with the interval in which x lies is either 0 or k, it finally yields the parameter $h = x - x_i$ to be used as the input argument of the spline approximator.
3) Spline Approximator (Module 3): In this module, the value of the cubic polynomial is found out by using the stored coefficients and the input argument $h = x - x_i$ according to the interval in which x lies as said earlier.

The algorithm is summarized in Table 1.

B. Three-interval interpolation

This scheme also leads to an efficient low power hardware architecture which is shown in Fig. 6. The architecture in this approach can also be divided into three modules as stated in the earlier case. However, the complexity in this case increases slightly for storage and computational purposes due to the increase in number of coefficients and arithmetic (addition and comparison) operations. We summarize this in Table 2.

A comparison for both the proposed schemes with respect to accuracy and hardware complexity is provided in Table 3.

VI. Conclusion

In this paper, a low complexity cubic spline interpolation technique has been introduced to approximate the sine function in the interval $[0, \pi/2]$ with two different number of data points, leading to two different cases. As the number of fixed data points are small in number within the specified range, both the cases lead to efficient low power and low complexity hardware architectures for sinusoidal generation. The usage of one/two data point(s) reduces the number of coefficients of the interpolator to eight/twelve only, respectively, which further reduces the power requirement of both the schemes from storage element point of view. To increase the hardware precision of both the proposed schemes, an error optimization scheme has been considered to shift the fixed data point(s) optimally, without any additional increment in hardware. Further, the first case leads to a fairly low complexity system whereas the second one gives rise to a relatively complex architecture with more accuracy and any one of these can thus be used depending upon the application.

TABLE I
Summary of the Two-Interval Interpolation Algorithm

Predefined: $x = 0, k, \pi/2$ and values of sine function at these points; stored a_i, b_i, c_i, d_i for $i = 1$ and 2.
1. Input data: argument $x \in [-\pi, \pi]$
2. Pass $sign(x)$ to consequent modules.
3. IF $|x| \leq \pi/2$
4. THEN $x = |x|$; ELSE $x = \pi - |x|$ ENDIF
5. IF $x = 0,\ k$ *or* $\pi/2$ THEN
6. Return the corresponding $\sin(x)$ or $-\sin(x)$ value depending upon the value of $sign(x)$. ENDIF
7. ELSE give one bit right shift to the input argument x and call this $[x]$.
8. Compare $[x]$ with $k/2$.
9. IF $[x] > k/2$ THEN
10. $[h] = [x] - k/2$. ENDIF
11. ELSE $[h] = [x]$.
12. Give one bit left shift to $[h]$ to get h.
13. Give this h as input to multiplier to recursively generate h, h^2, h^3.
14. Choose polynomial coefficients according to the interval in which x lies.
15. THEN compute $S(x) = a_i h^3 + b_i h^2 + c_i h + d_i$
16. Return $S(x)$ or $-S(x)$ depending upon the value of $sign(x)$.
17. END PROCEDURE.

TABLE II
Summary of the Three-Interval Interpolation Algorithm

Predefined: $x = 0, k, l, \pi/2$ and values of sine function at these points; stored a_i, b_i, c_i, d_i for $i = 1, 2$ and 3.
1. Input data: argument $x \in [-\pi, \pi]$
2. Pass $sign(x)$ to consequent modules.
3. IF $|x| \leq \pi/2$
4. THEN $x = |x|$; ELSE $x = \pi - |x|$ ENDIF
5. IF $x = 0,\ k\ ,\ l$ *or* $\pi/2$ THEN
6. Return the corresponding $\sin(x)$ or $-\sin(x)$ value depending upon the value of $sign(x)$. ENDIF
7. ELSE Give one bit right shift to the input argument x and call this as $[x]$.
8. Compare $[x]$ with $k/2$ and $l/2$.
9. IF $[x] < k/2$ THEN
10. $[h] = [x]$.
11. ELSEIF $k/2 < [x] < l/2$ THEN
12. $[h] = [x] - k/2$ ENDIF
13. ELSE $[h] = [x] - l/2$
14. Give one bit left shift to $[h]$ to get h.
15. Give this h as input to multiplier to recursively generate h, h^2, h^3.
16. According to the region in which x lies, choose polynomial coefficients.
17. THEN compute $S(x) = a_i h^3 + b_i h^2 + c_i h + d_i$
18. Return $S(x)$ or $-S(x)$ depending upon the value of $sign(x)$.
19. END PROCEDURE.

TABLE III
A Comparison of the Two Proposed Approaches with Error Optimization with Respect to Accuracy and Hardware Complexity in Terms of Number of Operations per Evaluation

Scheme	Accuracy (bits)	Hardware Complexity		
		Mult.	Add.	Comparisons
Two-Interval	11	6	2	3
Three-Interval	14	6	3	4

References

[1] J. E. Volder, "The CORDIC Trigonometric Computing Technique", *IRE Trans. Elect. Comput.*, vol. EC-8, pp. 330–334, Sep. 1959.
[2] J. C. Bajard, S. Kla and J. M. Muller, "BKM: A New Hardware Algorithm for Complex Elementary Functions," *IEEE Trans. Comput.*, vol. 43, no. 8, pp. 955–963, Aug. 1994.

[3] P. J. Davis, *Interpolation and Approximation*. New York: Dover Publications, 1990.
[4] J. A. Pineiro and M. D. Ercegovac, "High-Speed Double-Precision Computation of Reciprocal, Division, Square Root, and Inverse Square Root," *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1377–1388, 2002.
[5] 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.
[6] P. T. P. Tang, "Table-lookup Algorithms for Elementary Functions and their Error Analysis", in *Proc. 10th Symposium on Computer Arithmetic*, 1991, pp. 232–236.
[7] M. J. Schulte and J. E. Stine, "Approximating Elementary Functions with Symmetric Bipartite Tables," *IEEE Trans. Comput.*, vol. 48, no. 8, pp. 842–847, Aug. 1999.
[8] M. D. Ercegovac, T. Lang, J. M. Muller and A. Tisserand, "Reciprocation, Square Root, Inverse Square Root, and Some Elementary Functions using Small Multipliers," *IEEE Trans Comput.*, vol. 49, no. 7, pp. 628–637, July 2000.
[9] J. A. Pineiro, S. F. Oberman, J. M. Muller and J. D. Bruguera, "High-Speed Function Approximation using a Minimax Quadratic Interpolator," *IEEE Trans. Comput.*, vol. 54, no. 3, pp. 304–318, March 2005.
[10] V. Paliouras, K. Konstantina and S. Thanos, "A Floating-point Processor for Fast and Accurate Sine/Cosine Evaluation", *IEEE Trans. Circuit. Syst. II : Analaog and Digital Signal Processing*, vol. 47, no. 5, 1991, pp. 441–451, May 2000.
[11] 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.
[12] V. Kantabutra, "On Hardware for Computing Exponential and Trigonometric Functions", *IEEE Trans. Comput.*, vol. 45, no. 3, pp. 328–339, Mar. 1996.
[13] H. Ting, B. Liu and S. Chang, "An On-Chip Concurrent High Frequency Analog and Digital Sinusoidal Signal Generator", in *Proc. IEEE Asia-Pacific Conference on Circuits and Systems*, Tainan, Dec. 2004, pp.173-176.
[14] K. E. Atkinson, *An Introduction to Numerical Analysis*. New York: John Wiley & Sons, 1989.
[15] H. S. Dhillon and A. Mitra, "A Low Power Architecture of Digital Sinusoid Generator using Cubic Spline Interpolation", *IETE J. Edu.*, vol. 47, no. 3, pp. 129–136, July–Sept. 2006.
[16] I. Koren, *Computer Arithmetic Algorithms*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[17] S. McKinley and M. Levine. Cubic Spline Interpolation [Online]. Available: http://www-lmpa.univ-littoral.fr/ bouhamid/dossier _fichiers/cubicspline.PDF.
[18] M. J. Schulte and J. E. Stine, "Approximating Elementary Functions with Symmetric Bipartite Tables", *IEEE Trans. Comput.*, vol. 48, no. 8, pp. 842–847, August 1999.
[19] J. Cao, B. Wei and J. Cheng, "High-performance Architectures for Elementary Function Generation", in *Proc. 15th IEEE Symposium on Computer Arithmetic*, 2001, pp. 136–144.

Abhijit Mitra was born in Serampore, India, in 1975. He received the B.E.(Honors) degree from the Regional Engineering College, Durgapur, India, in 1997, the M.E.Tel.E. degree from Jadavpur University, India, in 1999 and the Ph.D. degree from the Indian Institute of Technology, Kharagpur, India, in 2004, all in electronics and communication engineering.

Since 2004, he has been with the Department of Electronics and Communication Engineering at the Indian Institute of Technology, Guwahati, India, as an Assistant Professor. He visited Indian Statistical Institute (ISI), Kolkata, as a Visiting Scientist during June-July 2007. His research interests include adaptive signal processing and signal processing applications in wireless communications with the primary emphasis on low complexity system realizations.

Dr. Mitra has been a member of IEEE since 2003 and presently serves as a reviewer of IEEE Transactions on Signal Processing, IEEE Transactions on Audio, Speech and Language Processing, and IEEE Signal Processing Letters. He is also a member of Indian Science Congress Association and Institution of Electronics and Telecommunication Engineers, India. Presently, he serves as a member of the editorial board of *Recent Patents on Electrical Engineering* (Bentham Science, USA), *International Journal of Signal Processing* and *International Journal of Information Technology* (Enformatika, Europe).

Harpreet Singh Dhillon was born in Amritsar, India, in 1986. He is currently a B.Tech. final year student in the Department of Electronics and Communication Engineering at the Indian Institute of Technology, Guwahati, India. The work reported in this paper was taken up by him, under the guidance of Dr. Abhijit Mitra, as a part of his undergraduate study.

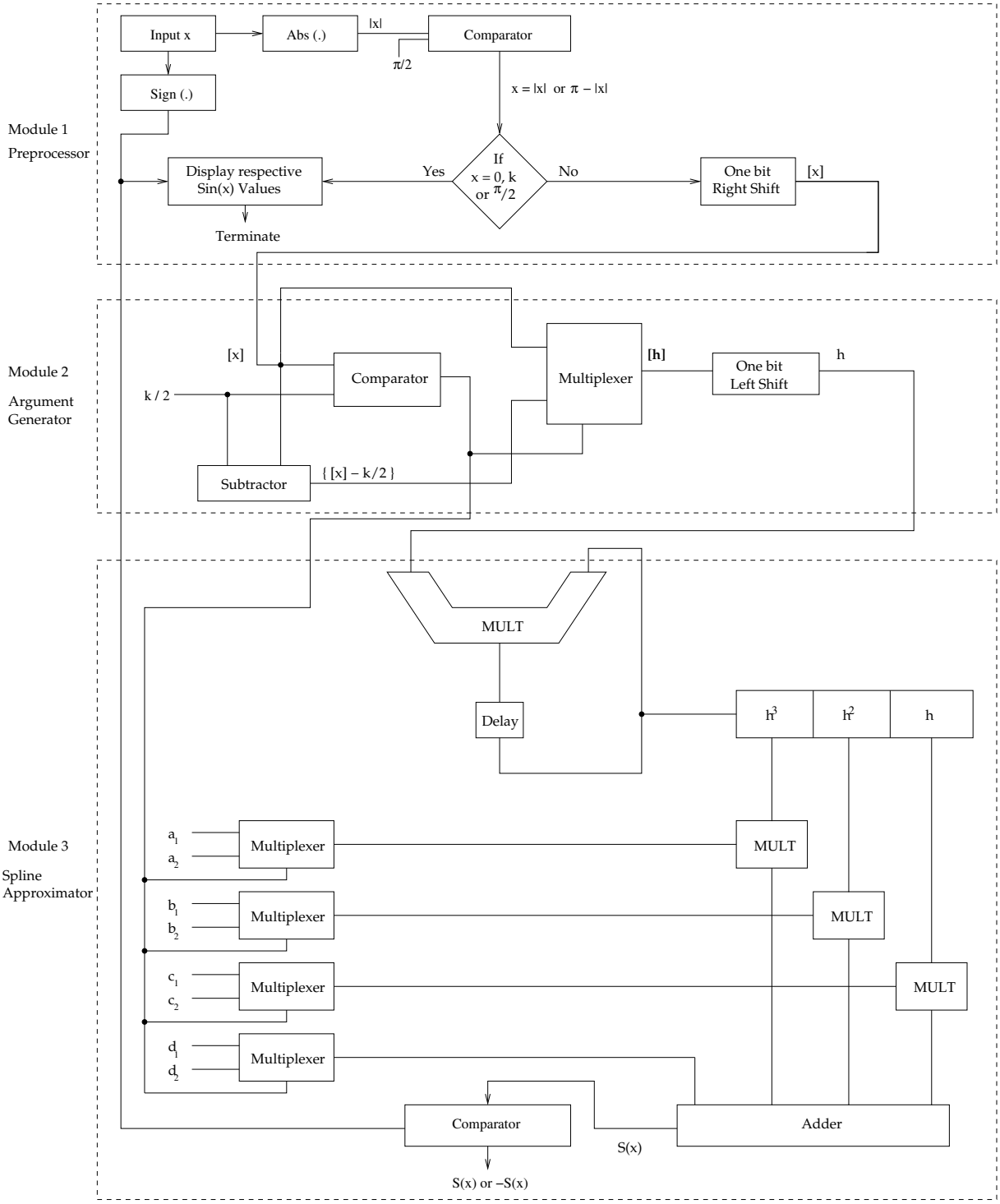

Fig. 5. Implementation of sinusoidal generator with the proposed two-interval scheme (here $h = x - x_i$).

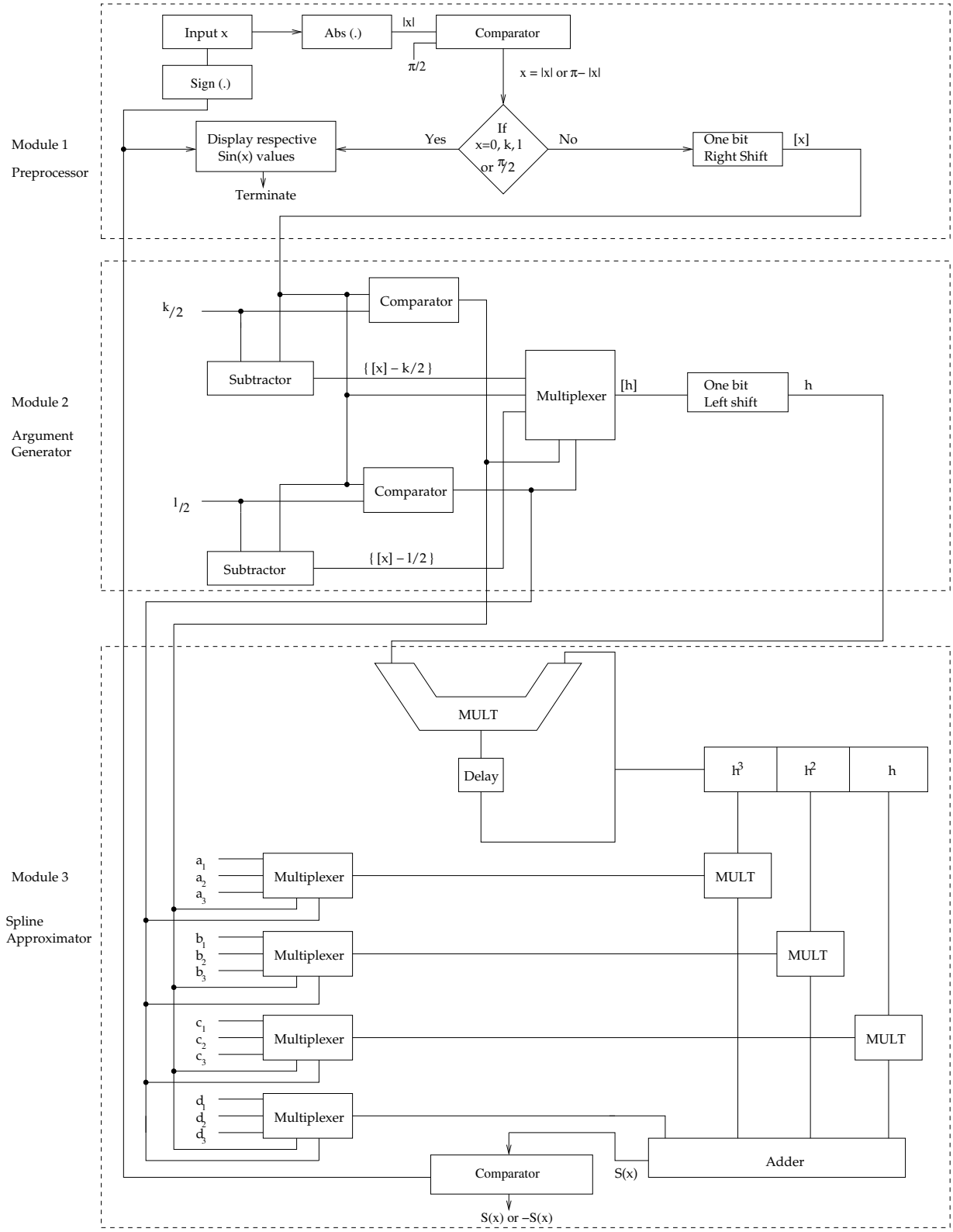

Fig. 6. Implementation of sinusoidal generator with the proposed three-interval scheme (here $h = x - x_i$).