Elementary Function Generators for Neural-Network Emulators

Stamatis Vassiliadis, *Fellow, IEEE*, Ming Zhang, *Member, IEEE*, and José G. Delgado-Frias, *Senior Member, IEEE*

***Abstract*—Piece-wise first- and second-order approximations are employed to design commonly used elementary function generators for neural-network emulators. Three novel schemes are proposed for the first-order approximations. The first scheme requires one multiplication, one addition, and a 28-byte lookup table. The second scheme requires one addition, a 14-byte lookup table, and no multiplication. The third scheme needs a 14-byte lookup table, no multiplication, and no addition. A second-order approximation approach provides better function precision; it requires more hardware and involves the computation of one multiplication and two additions and access to a 28-byte lookup table. We consider bit serial implementations of the schemes to reduce the hardware cost. The maximum delay for the four schemes ranges from 24- to 32-bit serial machine cycles; the second-order approximation approach has the largest delay. The proposed approach can be applied to compute other elementary function with proper considerations.**

***Index Terms*—Elementary function generators, hardwired neuroemulators, neural-network functions, piecewise approximation, square root implementation, trigonometric functions.**

I. Introduction

INEXPENSIVE high-performance hardwired emulators with acceptable function precision are highly desirable [1]. Even though precision may have important consequences in the neural paradigm it is only recently that such a question has been under investigation [1]–[2]. In the absence of a general guideline regarding the notion of "acceptable precision" in neural computations, we assume the precision achieved by other high-performance low-cost designs proposed for sigmoid generators [3]–[5], and propose schemes that improve both speed and precision. We consider a number of widely used elementary functions which include: sigmoid function[1] $\mathrm{sigm}(u)$, sigmoid function derivative $\mathrm{sigm}'(u)$, logarithm function $\ln(u)$, exponential function e^{-u}, trigonometric functions $\sin(u)$ and $\cos(u)$, hyperbolic tangent function $\tanh(u)$, square root function $\sqrt{u}$, and inverse and inverse square functions $1/u$ and $1/u^2$. Most of these elementary functions relate to the neural activation and the neural-network learning and they are required to compute the network parameters.

The elementary functions can be implemented using lookup tables [8]–[10], direct computations, e.g., using polynomial power series evaluations [11], hybrid approaches [12], [13], iterative approaches [14] and piece-wise approximations [3]–[6]. A lookup table is inexpensive to build and it may not introduce additional errors except the error incurred due to the input value representation. When the function precision requirement increases, the requirement in the table size escalates exponentially. A direct computation requires a large number of multiplications and additions, it is used when high precision is required. In the iterative approaches one bit is generated in each computation cycle [14]. The computation delay for this scheme is higher than other schemes. The hybrid approaches provide a tradeoff for the requirement of lookup table and numerical computations. It can be used for the computation of high precision sigmoid function [12], [13].

In this study we primarily investigate inexpensive high-performance hardwired implementations of elementary functions with an average error in the order of 10^{-3} and a maximum error in the order of 10^{-2}. A very important aspect of our proposal is the following: all of the schemes we present here can accommodate an arbitrary number of functions with no hardware changes. The only requirement is the storing in the memory of appropriate values for the functions to be computed with the computational hardware and the memory unchanged. Furthermore we note that precision can be improved with the use of more than eight segments for the approximation with the additional expenses of memory.

This paper is organized as follows. In Section II, we define the number system used for the computation of both the first- and second-order approximation. In Section III, we study three first-order approximation schemes with the hardware requirement from one multiplication and one addition to no multiplication and no addition. In Section IV, a second-order approximation scheme is proposed which requires one multiplication and two additions. The performance for a bit serial implementation of the schemes is discussed in Section V and an evaluation of a sigmoid generator using the proposed schemes is presented in Section VI. We conclude this paper with some remarks in Section VII.

II. Internal Number Sysytem and Notations

The representation used for the computations to generate the elementary function employs a fixed point fractional number system, denoted as the internal number system, with four integer bits and ten fraction bits as shown below:

$$\boxed{x_3\, x_2\, x_1\, x_0 \,|\, x_{-1}\, x_{-2}\, x_{-3}\, x_{-4}\, x_{-5}\, x_{-6}\, x_{-7}\, x_{-8}\, x_{-9}\, x_{-10}}$$

$\uparrow$radix point.

Manuscript received June 9, 1999; revised April 10, 2000.

S. Vassiliadis is with the Department of Electrical Engineering, Delft University of Technology, 2638 CD Delft, The Netherlands.

M. Zhang is with AT&T Wireless, Kirkland, WA 98033 USA.

J. G. Delgado-Frias is with the Department of Electrical Engineering, University of Virginia, Charlottesville, VA 22904-4743 USA.

Publisher Item Identifier S 1045-9227(00)09852-0.

[1]Sigmoid functions: $\mathrm{sigm}(u) = 1/(1 + e^{-u})$ and $\mathrm{sigm}'(u) = e^{-u}/(1 + e^{-u})^2$.

The most significant bit x_3 of the internal number system is a sign bit; "0" and "1" represent a positive and negative number, respectively. The choice of the length of the representation relates to an evaluation we conducted on ideal computations considering only the method error. Our evaluation indicates that the method error alone is in the order of 10^{-3} to 10^{-4} suggesting an internal representation limited to ten fractional bits which produces a representation error 2^{-10} which is in the order of 10^{-3}. A larger number of bits in the internal number system will not reduce substantially the error since the method error is the dominant error. Our scheme is applicable to both sign magnitude and two=s complement notations. The maximum value to be represented by the internal number system is $2^3 - 2^{-10} = 7.999$ for both sign magnitude and two's complement notations.

In the internal number system, in a number of occasions, a number α can be expressed as a set of known binary valued bits followed by a number of bits having unknown values denoted as the changing bits. For example: $\alpha = 0000.101x\,xxxx\,xx$. The first seven bits of α have the fixed binary value 0000.101 while the remaining bits are changing bits; "x" is used to denote a changing bit. Furthermore we denote by "&" the bit by bit "and" of two internal numbers. For example, if $\alpha = 0000.1111\,1100\,00$ and $\beta = 0000.1000\,0000\,11$ then $\alpha\&\beta = 0000.1000\,0000\,00$. We also denote mask$(n, m)$ as an internal number whose nth to mth fraction bits are "1" and all other bits are "0." For example mask$(-1, -4) = 0000.1111\ 0000\,00$ and mask$(3, -10) = 1111.1111\,1111\,11$. With the notation of mask(n, m), the truncation of an internal number α to nth fraction bit can be easily be expressed as: α&mask$(3, -n)$.

For an elementary function $F(u)$ and $u \in [\lambda_0, \lambda_1)$, if $F(u)$ is approximated by $H(u)$, then the average and maximum errors of this estimation are defined to be the average and maximum values of $|H(u) - F(u)|$ for u uniformly sampled on 10^6 points in the domain of $[\lambda_0, \lambda_1)$

Let $u_i = \lambda_0 + i * \Delta$ where $\Delta = (\lambda_1 - \lambda_0)/10^6$. Then

$$\text{Average Error} = \frac{\sum_{i=0}^{10^6-1} |H(u_i) - F(u_i)|}{10^6} \tag{2.1}$$

$$\text{Maximum Error} = \max_{\lambda_0 < u_i < \lambda_1} |H(u_i) - F(u_i)|. \tag{2.2}$$

We will use the average and maximum errors to evaluate the precision of the elementary function generators.

III. First-Order Approximations

A. First-Order Approximation: Scheme-1

In a segment, denoted by $[\alpha, \beta]$, a linear approximation function $H(u)$ can be used to compute the elementary function using the following equation: $H(u) = A + C * u$. To compute this equation one multiplication and one addition is required. The two linear parameters A and C are computed using the least squares method. For $u \in [\alpha, \beta]$, the ideal function output $F(u)$ and its linear approximations $H(u)$ are computed over the N inputs $u_i (0 \le i < N)$, the error of the linear approximation is $|H(u_i) - F(u_i)|$. The squares of these errors, $ER(A, C)$, over N inputs are defined to be

$$ER(A, C) = \sum_{i=0}^{N-1} (H(u_i) - F(u_i))^2$$

where

$$\begin{cases} D = (b-a)/N \\ u_i = a + i * D \end{cases} \qquad i = 0, 1, \cdots, N-1.$$

$ER(A, C)$ is a function of A and C. The minimum value of $ER(A, C)$ can be found by setting the derivatives of $ER(A, C)$ with respect to A and C to be zero. Consequently, we have that

$$\frac{\partial ER}{\partial A} = 2 \sum_{i=0}^{N-1} (H(u_i) - F(u_i))$$
$$\frac{\partial ER}{\partial C} = 2 \sum_{i=0}^{N-1} (H(u_i) - F(u_i)) u_i$$

for $H(u_i) = A + C * u_i$ we have that

$$N * A + N * C * \overline{X} - N * \overline{Y} = 0$$
$$N * A\overline{X} + N * C * \overline{X^2} - N * \overline{XY} = 0$$

where

$$\overline{X} = \frac{\sum_{i=0}^{N-1} u_i}{N} \quad \overline{X^2} = \frac{\sum_{i=0}^{N-1} u_i^2}{N} \quad \overline{Y} = \frac{\sum_{i=0}^{N-1} F(u_i)}{N}$$
$$\overline{XY} = \frac{\sum_{i=0}^{N-1} u_i * F(u_i)}{N}.$$

Thus

$$C = \frac{\overline{X} * \overline{Y} - \overline{XY}}{\overline{X}^2 - \overline{X^2}}$$

and

$$A = \overline{Y} - C * \overline{X} = \frac{\overline{XY} * \overline{X} - \overline{X^2} * \overline{Y}}{\overline{X}^2 - \overline{X^2}}. \tag{3.1}$$

Clearly, A and C are function of N, and the segment parameters α and β. The larger N is, the more precise the A and C are estimated. In this paper N is chosen to be 10^5.

B. First-Order Approximation: Scheme-2

In this scheme the multiplication requirement is substituted with a power of two multiplication. The equation for this linear approximation is given by the following:

$$H(u) = A + C * u \quad \text{where } |C| = 2^{-n} \text{ and } n \text{ is an integer.} \tag{3.2}$$

The square errors of this first-order approximation denoted as $ER(A, C)$ are defined to be

$$ER(A, C) = \sum_{i=0}^{N-1} (H(u_i) - F(u_i))^2 \text{ where } |C| = 2^{-n}. \tag{3.3}$$

When C is fixed, the parameter A can be determined using the least squares method by setting $\partial ER/\partial A$ to zero, implying that

$$\frac{\partial ER}{\partial A} = 2\sum_{i=0}^{N-1}(H(u_i) - F(u_i)) = 0. \tag{3.4}$$

Consequently

$$\sum_{i=0}^{N-1} A + C * u_i - F(u_i) = 0 \quad A = \overline{Y} - C * \overline{X}$$

where

$$|C| = 2^{-n}. \tag{3.5}$$

The parameter C is not known, and it should be chosen in such a way that $|C| = 2^{-n}$ and $ER(A, C)$ is minimum. The following steps summarize the procedure to determine the parameters A and C for the segment $[\alpha, \beta]$.

1) Set $n = 0$, $C_{opt} = 1$, $ER_{\min} = 1000$, $N = 10^5$, $\Delta = (b - a)/N$ where C_{opt} is the optimal value of C such that $ER(A, C_{\text{opt}})$ takes minimum value.
2) Let $C = \pm 2^{-n}$ and $A = \overline{Y} - C * \overline{X}$, compute the square sum of the error over the segment

$$ER(A, C) = \sum_{i=0}^{N-1}(A + C * u_i - F(u_i))^2$$

where

$$\begin{cases} D = (b - a)/N \\ u_i = a + i * D \end{cases} \qquad i = 0, 1, \cdots, N-1.$$

3) If $ER(A, C) < ER_{\min}$ then $ER_{\min} = ER(A, C)$, $C_{opt} = C$.
4) $n = n + 1$.
5) if $n < 20$ go to Step 2).
6) For $C = C_{opt}$ compute $A = \overline{Y} - C_{opt} * \overline{X}$.

C. First-Order Approximation: Scheme-3

In this scheme we proceed further and eliminate also the addition. This can be achieved by rounding the parameters of A' in scheme-2 to A in such a way that all the changing bits of $(2^{-n} * u)$ do not overlap with the nonzero bits of A. The equation we consider has the form of $A + C * u$ where $|C| = 2^{-n}$ and the changing bits of $(2^{-n} * u)$ do not overlap with the nonzero bits of A.

The easiest way to obtain the parameter A is using scheme-2 to find an optimum solution for A' and $C(|C| = 2^{-n})$ from the procedures given in the last section. We find the first changing bit of u in the segment. If the changing bits of u begin from the mth fraction bit then the changing bits of $2^{-n}u$ initiate from the $n + m$th fraction bit. Consequently if we round A' to $m + n - 1$th fraction bits in the following way: $A = (A' + 2^{-(n+m-1)})\&\text{mask}(3, -(n + m - 1))$ then A and $2 - nC$ do not have overlapping changing bits.

The first changing bit of A' is determined according to the segment input u belongs to. A and $C = \pm 2^{-n}$ values which give the minimum average error of function estimation in the segment can be obtained using a heuristic search. The heuristic search operates on the A and C values which are located in the vicinity of the initial estimate A'. For a segment $[\alpha, \beta)$ the "optimal" solutions for A and C are denoted as A_{opt} and C_{opt} by the following.

1) Determine the first changing bit m for the segment $[\alpha, \beta)$. Set $n = 0$, $C_{\text{opt}} = 1$, $ER_{\min} = 1000$, $N = 10^5$, $\Delta = (b - a)/N$ where C_{opt} is the optimal value of C such that $ER(A, C)$ is minimum.
2) Let $C = \pm 2^{-n}$, $A = \overline{Y} - C * \overline{X}$ and $\Delta A = 2^{-(n+m-1)}$.
3) Let $A = A\&\text{mask}(3, -(n + m - 1)) - 10 * \Delta A$ and count $= 1$.
4) $A = A + \Delta A$, $ER(A, C) = \sum_{i=0}^{N-1}(A + C * u_i - F(u_i))^2$ If $ER(A, C) < ER_{\min}$ then $ER_{\min} = ER(A, C)$, $A_{opt} = A$, $C_{opt} = C$, count = count +1.
5) if (count <11) go to Step 4).
6) $n = n + 1$.
7) if $n < 20$ go to Step 2.

Here we chose for this presentation the number of sample points N to be 10^5.

D. Implementations of the Linear Approximation Schemes

Since the function inputs may have a very wide dynamic range, many segments may need to be divided in order to have a good function approximation. Parameters A and C (used in Scheme-1) can be stored in the memory (e.g., ROM) for all segments. The more segments we divide the input domain, the larger ROM size is required. In order to reduce the large ROM size requirement, we perform a transformation of the function input so that the function input domain is significantly reduced.

For example, we express the function input u for the logarithm function as $u = 2^m\omega$ where $\omega \in [1, 2)$ then we have that $\ln(u) = \ln(2^m\omega) = m * \ln(2.0) + \ln(\omega)$. The multiplication of $m * \ln(2.0)$ can be computed either by one multiplication or by lookup table with the number of word entries to be m. For all other elementary functions Table I shows the transformations for the function inputs. In Table I, the symbol "Int" is used to express the integer portion of input u for the exponential function e^{-u}. As it can be observed in Table I, the domain of the sigmoid function for the linear approximation is reduced to interval (−4, 4). For inputs beyond this range, an output of "0" or "1" is generated depending on the sign bit of the inputs. If the input is less than −4 then the sigmoid function output is forced to zero, if the input is greater than four then the function output is forced to one. Using the linear approximation described previously denoted as scheme-1, and assuming that the number of sample points N used to estimate the linear parameters is 10^5, the parameters A and C can be obtained for each segment.

For simplicity of discussion, we consider the u domain for the elementary function $F(u)$ to be the transformed ω domain given by the third column in Table I, denoted as $[\lambda_0, \lambda_1]$. The choice of the number of segments depends on the implemented function. Since the width of the w for the trigonometric functions $\sin(u)$ and $\cos(u)$ is 3.14 which is not a number in power of two, we chose the total number of segments to be seven for the segment width to be 0.5. The width of a segment is $2^{-3} * \Delta$ where $\Delta = (\lambda_1 - \lambda_0)$ is the width of the u domain. For the Scheme-1 the estimated parameters A and C using 10^5 uniformly sampled data in each segment are shown in Table II.

In Table II, the first line in each box is parameter A and the second line is parameter C. It should also be pointed out that Table II describes the general rule for the closure of the segment. There are some special cases, for example

TABLE I
TRANSFORMS OF FUNCTION INPUT

Function	Transforms	ω domain	Internal representation of ω
sigm(u)	N/A	(-4,4)	xxxx. xxxx xxxx xx
sigm'(u)	ω=|u|	[0,8)	0xxx. xxxx xxxx xx
sin(u)	ω=u+kπ	[0, 3.14)	00xx. xxxx xxxx xx
cos(u)	ω=u+kπ	[0, 3.14)	00xx. xxxx xxxx xx
ln(u)	$\omega=2^m u$	[1,2)	0001. xxxx xxxx xx
e^{-u}	ω=u-Int	[0,1)	0000. xxxx xxxx xx
tanh(u)	ω=|u|	[0,8)	0xxx. xxxx xxxx xx
1/u	$\omega=2^m u$	[1,2)	0001. xxxx xxxx xx
$\sqrt{u}$	$\omega=2^m u$	[0,1)	0000. xxxx xxxx xx
$1/u^2$	$\omega=2^m u$	[1,2)	0000. xxxx xxxx xx

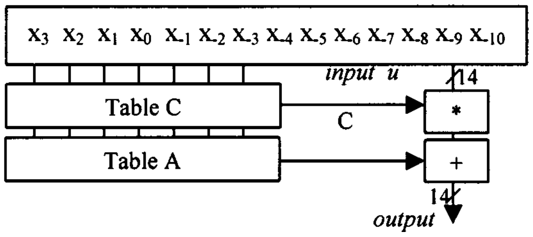
TABLE II
PARAMETERS A AND C FOR SCHEME-1

Function	u interval	Δ	Segments							
			$[\lambda_0, \lambda_0+\Delta)$	$[\lambda_0+\Delta, \lambda_0+2\Delta)$	$[\lambda_0+2\Delta, \lambda_0+3\Delta)$	$[\lambda_0+3\Delta, \lambda_0+4\Delta)$	$[\lambda_0+4\Delta, \lambda_0+5\Delta)$	$[\lambda_0+5\Delta, \lambda_0+6\Delta)$	$[\lambda_0+6\Delta, \lambda_0+7\Delta)$	$[\lambda_0+7\Delta, \lambda_0+8\Delta)$
sigm(u)	(-4,4)	1	0.1321 0.0290	0.2561 0.0711	0.4106 0.1495	0.4962 0.2326	0.5038 0.2326	0.5894 0.1495	0.7439 0.0711	0.8679 0.0290
sigm'(u)	[0,8)	1	0.2586 -0.0550	0.2890 -0.0929	0.2210 -0.0597	0.1247 -0.02722	0.0602 -0.0109	0.0269 -0.0041	0.0115 -0.0015	0.0048 -0.0006
sin(u)	[0,3.14)	0.5	0.0041 0.9629	0.1292 0.7271	0.5474 0.3134	1.2838 -0.1771	2.1746 -0.6243	2.9037 -0.9185	3.1323 -0.9970	
cos(u)	[0,3.14)	0.5	1.0203 -0.2459	1.2321 -0.6774	1.4909 -0.9431	1.5348 -0.9779	1.1181 -0.7732	0.1283 -0.3793	-0.7796 -0.0707	
ln(u)	[1,2)	0.125	-0.9407 0.9418	-0.8292 0.8426	-0.7289 0.7623	-0.6378 0.6959	-0.5543 0.6402	-0.4773 0.5928	-0.4057 0.5519	-0.3390 0.5162
e^{-u}	[0,1)	0.125	0.9988 -0.9398	0.9851 -0.8294	0.9608 -0.7319	0.9287 -0.6459	0.8908 -0.5700	0.8490 -0.5030	0.8047 -0.4439	0.7591 -0.3918
tanh(u)	(0,8)	1.0	0.0479 0.7717	0.6005 0.1938	0.9113 0.0292	0.9838 0.0040	0.9973 0.0005	0.9996 0.0001	0.9999 0.0000	1.0000 0.0000
1/u	[1,2)	0.125	1.8854 -0.8877	1.6864 -0.7103	1.5254 -0.5813	1.3925 -0.4845	1.2810 -0.4100	1.1860 -0.3515	1.1041 -0.3046	1.0328 -0.2666
$\sqrt{u}$	[0,1)	0.125	0.0943 2.2628	0.2126 1.1647	0.2777 0.8972	0.3296 0.7571	0.3743 0.6673	0.4140 0.6034	0.4503 0.5550	0.4838 0.5166
$1/u^2$	[1,2)	0.125	2.6679 -1.6744	2.1341 -1.1983	1.7460 -0.8870	1.4549 -0.6748	1.2310 -0.5253	1.0551 -0.4169	0.9144 -0.3364	0.8001 -0.2753

the sigmoid function, the inputs may be negative. The definitions for segments residing on the negative axis with sign magnitude notation is different from the one with two's complement notations in order to express the input in each segment with the same number of changing bits for both number systems. We note that a difference between the sign magnitude and two's complement notation is on the closures of the intervals. The sign magnitude procedures operate as wanted if the closure of the interval is on the right-hand side, e.g., $(-4, -3]$, $(-3, -2]$, $(-2, -1]$, $(-1, 0)$ for the negative number. To the contrary the two's complement notation requires the closure to be on the left-hand side, e.g., $(-4, -3)$, $[-3, -2)$, $[-2, -1)$ and $[-1, 0)$. Even though Table II can be used for both notations, for ease of description the interval notation is the two's complement notation as we use for the description of Table II only $[\lambda_0, \lambda_0 + \Delta)$ rather than both notations.

To implement the scheme-1 in hardware, tables implemented by ROM can be used to store the two linear parameters A and C, the address of the table is generated directly from the function inputs. For the function input expressed in the internal number system as $u = x_3x_2x_1x_0.x_{-1}x_{-2}x_{-3}x_{-4}x_{-5}x_{-6}x_{-7}x_{-8}x_{-9}x_{-10}$, the address bits are $x_3x_2x_1x_0x_{-1}x_{-2}x_{-3}$ for all the elementary function. Each elementary function uses only part of these bits to decode the table entries. The number of table entries for A and C are both seven for the trigonometric functions, and eight for the other elementary functions. Consequently, the total table size for A and C is $7 * 2 * 14 = 196$ bits ≈25 bytes for the trigonometric functions and $8 * 2 * 14 = 224$ bits $= 28$ bytes for the other elementary functions. The block diagram of implementation of scheme-1 is shown in Fig. 1.

For the first-order approximation scheme-2 where no multiplication is required, the parameters A and $C(|C| = 2^{-n})$



x_3 x_2 x_1 x_0 x_{-1} x_{-2} x_{-3} x_{-4} x_{-5} x_{-6} x_{-7} x_{-8} x_{-9} x_{-10}
input u
14
Table C
C
*
Table A
+
14
output

Fig. 1. Block diagram for Scheme-1.

are needed for each segment. For the segments we defined in scheme-1, the parameters A and C can be determined as shown in Table III following the procedures given in Section III-B. In Table III, the first line in each box is parameter A and the second line in the box is parameter C. As can be observed from Table III, $|C| = 2^{-n}$ which satisfies the constrain we impose in equation (3.3).

For the Scheme-2, only the parameter A need to be stored in the ROM, the shifter control parameter can be generated by combinational logic which can be decoded from the address bits. The address bits for accessing the table for the scheme-2 are the same as in scheme-1 for the elementary functions we have discussed. The number of table entries for storing A is seven for trigonometric functions and eight for the other elementary functions. The total table size requirement is $7 * 14 = 98$ bits ≈ 13 bytes for the trigonometric functions and $8 * 14 = 112$ bits $= 14$ bytes for the other elementary functions. The block diagram of the implementation of scheme-2 is shown in Fig. 2. As shown in Table III, the value of $C(|C| = 2^{-n})$ may be negative, the designed "Shifter" should be able to change the sign bit for sign magnitude notation and provide the proper inversion for two's complement notation of its output depending on the input segment.

For the first-order approximation using scheme-3, the parameters A and $C(|C| = 2^{-n})$ are determined by the steps given in Section III-C, these values are shown in Table IV.

We express the parameter A in Table IV in hexadecimal notation. In performing $A \pm 2^{-n} * u$ in sign magnitude notation the operation can be either effective addition or effective subtraction [15]. As can be seen from Table IV, the values A are always positive in all the segments we consider. For the case allowed by the values of A, C and the input (u) there are two distinct operations (effective addition and effective subtraction) that result for the sign magnitude operations [15].

The procedure used to perform the approximation can be easily explained using an example of effective addition. For sigmoid function $\mathrm{sigm}(u)$ with $u \in (-4, -3]$, u can be expressed in the internal number system as $u = 1011.xxxx\,xxxx\,xx$. From Table IV it is observed that $A = 0000.0000\ 0000\,00$ and $C = -2^{-7}$, thus

$$\begin{aligned} H(u) &= A + 2^{-7}u \\ &= 0000.0000\,0000\,00 + 0000.0000\,011x\,xx \\ &= 0000.0000\,011x\,xx. \end{aligned} \quad (3.6)$$

An example covering and explaining the second case can be found in [22].

The final outcome is that for the example the approximation $H(u)$ can be accomplished using bit inversion of the changing bits of the input and merging, thus eliminating the need of addition/subtraction. This procedure can be applied to all entries with proper consideration.

One example to design a sign magnitude logarithm generator from Table IV is given in Table V. The first column of Table V is the data format of the function inputs in sign magnitude notation for the eight segment we divide the input domain [1, 2). The second column of Table V is the function output data format in sign magnitude notation. It can be observed from Table V that each bit of the function output can be obtained either directly from the function input bit or its bit inverse or a constant zero or one depending on the segment the input belongs to.

For the internal number system assumed to be two=s complement notation, the input and the output of the functions can be negative. Two of the four cases, namely A, C and u all positive, and A and u are positive and C is negative, are covered by sign magnitude procedure. The other two cases, A and C are positive and u is negative, and A is positive and C and u are negative, are fully presented in [22] where it is shown that similar to the sign magnitude notation, the final outcome for the two's complement notation is that the approximation $H(u)$ can be accomplished using fixed valued bits followed by changing bits or by the bit inversion of the changing bits of the input. Using the same method for the above examples, the data output formats for the sigmoid are obtained and shown in Table VI; two's complement notation is used.

A general block diagram of the Scheme-3 is shown in Fig. 3. In Fig. 3, the bit inverse control logic determines if the input bits should be inverted. For example, the input bits are inverted for input in segments 1, 2, 3, 5, 7, 8 to compute the logarithm function. Referring to Fig. 3, the numbers in Table I and Table II determine if the output bits after the shifter are set to one or zero. Table I and Table II store two internal numbers. For a given bit position if the set bit is one and the reset bit is zero, then the corresponding output bit is one, otherwise if the set bit is zero and the reset bit is one the output bit is reset to zero. If the set and the reset bits are both zero, then the output bits of the flip flop are the same as the corresponding input bits to the shifter. The set and reset bits cannot be one at the same time, this can be achieved by designing the table without two ones in the same bit position. The number of entries for the two words in the tables is the number of segments we divide the input. For elementary function to be the trigonometric function $\sin(\omega)$ and $\cos(\omega)$, the number of segments is seven, then the number of table entries is seven as well. Each word stores 14 bits, thus the total ROM size is $7 * 2 * 14 = 196$ bits $= 25$ bytes. For the other elementary function, the number of word entries is eight and the total table size is $8 * 2 * 14 = 224$ bits $= 28$ bytes. The following example shows the method to determine the numbers in Table I and Table II. Additional examples are found in [22].

For the logarithm with input in the segment of [1, 1.125), the output format is 0000.0000 111, then two numbers in Table I and Table II are: Table I = 0000.0000 1110 00 and Table II = 1111.1111 0000 00. What is performed is forcing changing bits of output format to zero for both Table I and Table II entries, then passing and inverting the nonchanging bits to the Table I and Table II entries, respectively.

TABLE III
PARAMETERS A AND $C(C = 2^{-n})$ FOR SCHEME-2

Function	u interval	Δ	Segments							
			$[\lambda_0, \lambda_0+\Delta)$	$[\lambda_0+\Delta, \lambda_0+2\Delta)$	$[\lambda_0+2\Delta, \lambda_0+3\Delta)$	$[\lambda_0+3\Delta, \lambda_0+4\Delta)$	$[\lambda_0+4\Delta, \lambda_0+5\Delta)$	$[\lambda_0+5\Delta, \lambda_0+6\Delta)$	$[\lambda_0+6\Delta, \lambda_0+7\Delta)$	$[\lambda_0+7\Delta, \lambda_0+8\Delta)$
sigm(u)	(-4,4)	1	0.1398 2^{-5}	0.2346 2^{-4}	0.3738 2^{-3}	0.5049 2^{-2}	0.4951 2^{-2}	0.6262 2^{-3}	0.7654 2^{-4}	0.8602 2^{-5}
sigm (u)	[0,8)	1	0.2623 -2^{-4}	0.2435 -2^{-4}	0.2280 -2^{-4}	0.1388 -2^{-5}	0.0465 -2^{-7}	0.0257 -2^{-8}	0.0143 -2^{-9}	0.0079 -2^{-10}
sin(u)	[0,3.14)	0.5	-0.0052 2^{0}	0.2996 2^{-1}	0.6266 2^{-2}	1.1925 -2^{-3}	1.8950 -2^{-1}	3.1277 -2^{0}	3.1415 -2^{0}	
cos(u)	[0,3.14)	0.5	1.0214 -2^{-2}	1.0991 -2^{-1}	1.5621 -2^{0}	1.5736 -2^{0}	1.6284 -2^{0}	0.4603 -2^{-1}	-0.8047 -2^{-4}	
ln(u)	[1,2)	0.125	-1.0025 2^{0}	-1.0161 2^{0}	-1.0409 2^{0}	-0.3562 2^{-1}	-0.3352 2^{-1}	-0.3207 2^{-1}	-0.3117 2^{-1}	-0.3075 2^{-1}
e^{-u}	[0,1)	0.125	1.0025 -2^{0}	1.0171 -2^{0}	0.8883 -2^{-1}	0.8648 -2^{-1}	0.8514 -2^{-1}	0.8469 -2^{-1}	0.8503 -2^{-1}	0.8606 -2^{-1}
tanh(u)	(0,8)	1.0	-0.0662 2^{0}	0.5162 2^{-2}	0.9062 2^{-5}	0.9842 2^{-8}	0.9953 2^{-10}	0.9946 2^{-10}	0.9937 2^{-10}	0.9927 2^{-10}
$1/u$	[1,2)	0.125	2.0048 -2^{0}	1.4366 -2^{-1}	1.4187 -2^{-1}	1.4148 -2^{-1}	1.4216 -2^{-1}	1.0147 -2^{-2}	1.0051 -2^{-2}	1.0007 -2^{-2}
$\sqrt{u}$	[0,1)	0.125	0.1107 2^{1}	0.2435 2^{0}	0.2456 2^{0}	0.2234 2^{0}	0.4684 2^{-1}	0.4851 2^{-1}	0.4949 2^{-1}	0.4993 2^{-1}
$1/u^2$	[1,2)	0.125	3.0139 -2^{1}	1.8986 -2^{0}	1.8943 -2^{0}	1.2036 -2^{-1}	1.1915 -2^{-1}	1.1954 -2^{-1}	0.7579 -2^{-2}	0.7510 -2^{-2}

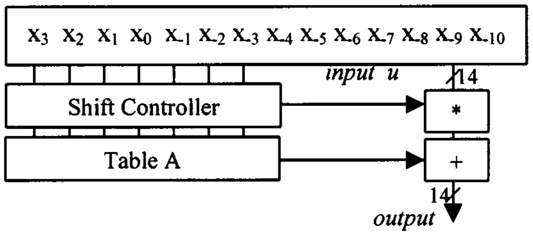


Fig. 2. Block diagram for Scheme-2.

We note here that one of the two tables is not necessary for an implementation at the expense of control and inverting logic, this reduces the table size by half.

The errors of the elementary function generators have three sources: method error, representation error and what we denote as a "partitioning" error. The method error is the error generated from the first-/second-order polynomial series evaluations. The representation error is generated due to the finite word length of the internal number representation. The multiplication and the bit shifting may generate representation errors due to the truncation. The "partitioning" error relates to the elementary functions $\mathrm{sigm}(u)$, $\mathrm{sigm}'(u)$ and $\tanh(u)$ when the function output is force to one or zero when the input is beyond the piecewise approximation range. To evaluate the performance of the three linear approximation schemes the average and maximum error for each scheme is computed for all the elementary function we have discussed. For the elementary functions $\mathrm{sigm}(u)$, $\mathrm{sigm}'(u)$ and $\tanh(u)$, the "partitioning" errors must be considered for the function generator precision. The average error denoted as AVE-ERR and the maximum error denoted as MAX-ERR are computed using (2.1) and (2.2) with consideration of finite length word length of the internal number system based on the 10^6 outputs for input uniformly sampled in the domain of $(-8, 8)$.

For the other elementary functions considered in this paper, the AVE-ERR and the MAX-ERR are computed for the input u in the domain given by the second column of Table I for the 10^6 outputs for inputs uniformly sampled in the u domain. The average and maximum errors are found for the elementary functions using the three schemes as shown in Table VII using computer simulations. The table size requirements for the three schemes for the elementary functions are also shown in the table (note that the computation for scheme-3 assume a single table implementation).

IV. SECOND-ORDER APPROXIMATION: SCHEME-4

The function inputs are divided into segments and a second-order approximation is used to compute the elementary functions in each segment. For a segment $[\alpha, \beta]$, if an input $u \in [\alpha, \beta]$, a second-order approximation for the sigmoid function can be computed as $c_0 + c_1 * u + c_2 * u^2 = c_0 + u * (c_1 + c_2 * u)$. It requires two multiplications and two additions for the computation. In order to reduce computing delay and complexity, we propose an approximation method that requires computations of only one multiplication and two additions

$$H(u) = A + C * (u + B)^2 \qquad \text{where } |C| = 2^{-n}. \tag{4.1}$$

Since we constrain C to be a value with a power of two, multiplication with C can be computed by bit shifting consequently only one multiplication is required to compute (4.1).

We compute the ideal function output $F(u_i)$ and its second-order approximation $H(u_i)$ on N uniformly spaced

TABLE IV
PARAMETERS A AND $C(|C| = 2^{-n})$ FOR SCHEME-3

Function	u interval	Δ	Segments							
			$[\lambda_0, \lambda_0+\Delta)$	$[\lambda_0+\Delta, \lambda_0+2\Delta)$	$[\lambda_0+2\Delta, \lambda_0+3\Delta)$	$[\lambda_0+3\Delta, \lambda_0+4\Delta)$	$[\lambda_0+4\Delta, \lambda_0+5\Delta)$	$[\lambda_0+5\Delta, \lambda_0+6\Delta)$	$[\lambda_0+6\Delta, \lambda_0+7\Delta)$	$[\lambda_0+7\Delta, \lambda_0+8\Delta)$
sigm(u)	(-4,4)	1	0.000 -2^{-7}	0.400 2^{-4}	0.600 2^{-3}	0.800 2^{-2}	0.800 2^{-2}	0.a00 2^{-3}	0.c00 2^{-4}	0.fe0 -2^{-7}
sigm'(u)	[0,8)	1	0.400 -2^{-4}	0.400 -2^{-4}	0.280 -2^{-5}	0.400 -2^{-4}	0.0c0 -2^{-7}	0.0c0 -2^{-7}	0.020 -2^{-10}	0.020 -2^{-10}
sin(u)	[0,3.14)	0.5	0.000 2^{0}	0.400 2^{-1}	0.a00 2^{-2}	1.300 -2^{-3}	3.000 -2^{0}	1.c00 -2^{-1}	0.e00 -2^{-2}	
cos(u)	[0,3.14)	0.5	1.000 -2^{-2}	1.800 -2^{0}	1.800 -2^{0}	1.800 -2^{0}	0.800 -2^{-1}	0.800 -2^{-1}	0.800 -2^{-1}	
ln(u)	[1,2)	0.125	0.200 -2^{-4}	0.c00 -2^{-1}	0.980 -2^{-2}	0.000 2^{-2}	0.a40 -2^{-3}	0.180 2^{-2}	0.9f0 -2^{-6}	0.c80 -2^{-4}
e^{-u}	[0,1)	0.125	1.000 -2^{0}	1.000 -2^{0}	0.e00 -2^{-1}	0.e00 -2^{-1}	1.200 -2^{0}	0.b00 -2^{-2}	0.a80 -2^{-2}	0.a00 -2^{-2}
tanh(u)	(0,8)	1.0	0.000 2^{0}	0.800 2^{-2}	0.e80 2^{-5}	0.fc0 2^{-8}	0.fec 2^{-10}	1.014 -2^{-10}	1.018 -2^{-10}	1.01c -2^{-10}
1/u	[1,2)	0.125	1.700 -2^{-1}	1.700 -2^{-1}	1.700 -2^{-1}	1.100 -2^{-2}	1.080 -2^{-2}	1.700 -2^{-1}	1.000 -2^{-2}	1.000 -2^{-2}
$\sqrt{u}$	[0,1)	0.125	0.300 -2^{-2}	0.400 2^{0}	0.400 2^{0}	0.700 2^{-1}	0.e00 -2^{-2}	0.800 2^{-1}	0.800 2^{-1}	0.800 2^{-1}
$1/u^2$	[1,2)	0.125	3.000 -2^{1}	1.e00 -2^{0}	1.e00 -2^{0}	1.300 -2^{-1}	1.300 -2^{-1}	1.300 -2^{-1}	0.c00 -2^{-2}	0.c00 -2^{-2}

TABLE V
A LOGARITHM GENERATOR USING SCHEME-3

input u	logarithm output
0001.000x xxxx xx	0000.0000 111$\overline{\text{x}}$ $\overline{\text{xx}}$
0001.001x xxxx xx	0000.0010 $\overline{\text{xxxx}}$ $\overline{\text{xx}}$
0001.010x xxxx xx	0000.0100 0$\overline{\text{xxx}}$ $\overline{\text{xx}}$
0001.011x xxxx xx	0000.0101 1xxx xx
0001.100x xxxx xx	0000.0111 00$\overline{\text{xx}}$ $\overline{\text{xx}}$
0001.101x xxxx xx	0000.1000 0xxx xx
0001.110x xxxx xx	0000.1001 0111 1$\overline{\text{x}}$
0001.111x xxxx xx	0000.1010 100$\overline{\text{x}}$ $\overline{\text{xx}}$

inputs $u_i (0 \le i < N)$ in the segment $[\alpha, \beta]$, where u_i is defined to be

$$\begin{cases} D = (b-a)/N \\ u_i = a + i * D \end{cases} \qquad i = 0, 1, \cdots, N-1. \tag{4.2}$$

The error between the ideal and the estimated function is $|H(u_i) - F(u_i)|$ and the square sum of the second-order approximation error over N sample inputs in segment $[\alpha, \beta]$ is

$$ER(A, B, C) = \sum_{i=0}^{N-1} \left(H(u_i) - F(u_i)\right)^2. \tag{4.3}$$

The least squares method can also be used to find the parameters A, B and C so that $ER(A, B, C)$ is minimum. When C is given, the minimum value of $ER(A, B, C)$ can be found by setting the derivatives of $ER(A, B, C)$ with respect to A and B to be zero. Then we have that

$$\frac{\partial ER}{\partial A} = 2 \sum_{i=0}^{N-1} H(u_i) - F(u_i) = 0$$

and

$$\frac{\partial ER}{\partial B} = 4C \sum_{i=0}^{N-1} \left(H(u_i) - F(u_i)\right)(u_i + B) = 0 \tag{4.4}$$

by substituting (Definition 4.1), $H(u_i) = A + C * (u_i + B)^2$ (Definition 4.1) and computations (the explicit derivation can be found in [23]). It can be concluded that

$$\begin{aligned} &N * A + N * C * \overline{X^2} + 2N * B * C * \overline{X} \\ &\quad + N * C * B^2 - N * \overline{Y} = 0 \qquad (4.5.1) \\ &N * A\overline{X} + N * C * \overline{X^3} + 2N * B * C * \overline{X^2} \\ &\quad - N * C * B^2 * \overline{X} - N * \overline{XY} = 0 \qquad (4.5.2) \end{aligned}$$

where

$$\overline{X} = \frac{\sum_{i=o}^{N-1} u_i}{N} \quad \overline{X^2} = \frac{\sum_{i=o}^{N-1} u_i^2}{N} \quad \overline{X^3} = \frac{\sum_{i=o}^{N-1} u_i^3}{N}$$

$$\overline{Y} = \frac{\sum_{i=o}^{N-1} F(u_i)}{N} \quad \overline{XY} = \frac{\sum_{i=o}^{N-1} u_i * F(u_i)}{N}.$$

Solving (4.5), A and B can be determined to be

$$B = \frac{\overline{X} * \overline{Y} - \overline{XY} + C * \left(\overline{X^3} - \overline{X^2} * \overline{X}\right)}{2 * C * \left(\overline{X}^2 - \overline{X^2}\right)}$$

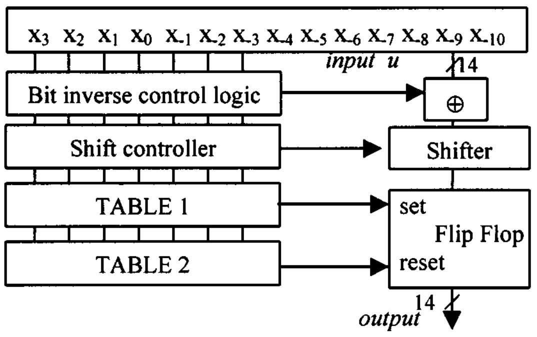
and

$$A = \overline{Y} - C\overline{X^2} - 2B * C * \overline{X} - C * B^2. \tag{4.6}$$

For $|C| = 2^{-n}$ with n being an integer, the larger the number of samples N is, the more precise the A and B are estimated. In this investigation, we chose N to be 10^5. A set of A, B, C and $ER(A, B, C)$ are obtained for n changing from zero to 20. For n greater than 20, since $2^{-n} < 2^{-20} \ll 1$, the second-order

TABLE VI
OUTPUT FORMAT FOR THE SIGMOID AND ITS DERIVATIVE IN 8 SEGMENTS

Segments	input format	sigm(u)	sigm'(u)
[-4,-3)	1100.xxxx xxxx xx	$\mathtt{0000.0000\ 011\overline{x}\ \overline{xx}}$	$\mathtt{0000.0000\ 011\overline{x}\ \overline{xx}}$
[-3,-2)	1101.xxxx xxxx xx	0000.0001 xxxx xx	0000.0001 0xxx xx
[-2,-1)	1110.xxxx xxxx xx	0000.001x xxxx xx	0000.0010 xxxx xx
[-1,0)	1111.xxxx xxxx xx	0000.01xx xxxx xx	0000.0011 xxxx xx
[0,1)	0000.xxxx xxxx xx	0000.10xx xxxx xx	$\mathtt{0000.0011\ \overline{xxxx}\ \overline{xx}}$
[1,2)	0001.xxxx xxxx xx	0000.110x xxxx xx	$\mathtt{0000.0010\ \overline{xxxx}\ \overline{xx}}$
[2,3)	0010.xxxx xxxx xx	0000.1110 xxxx xx	$\mathtt{0000.0001\ 0\overline{xxx}\ \overline{xx}}$
[3,4)	0011.xxxx xxxx xx	$\mathtt{0000.1111\ 011\overline{x}\ \overline{xx}}$	$\mathtt{0000.0000\ \overline{xxxx}\ \overline{xx}}$



$x_3\ x_2\ x_1\ x_0\ x_{-1}\ x_{-2}\ x_{-3}\ x_{-4}\ x_{-5}\ x_{-6}\ x_{-7}\ x_{-8}\ x_{-9}\ x_{-10}$
input u
14
Bit inverse control logic
⊕
Shift controller
Shifter
TABLE 1
set
Flip Flop
TABLE 2
reset
14
output

Fig. 3. Block dagram of Scheme-3

term $2^{-n} * (u + B)^2$ can be neglected compared to the representation error of our internal number system. We exhaustively search the minimal value of $ER(A, B, C)$ for $C = \pm 2^{-n}$ with n varying from zero to 20 to identify the number $C_{\rm opt}$ such that $ER(A, B, C_{\rm opt})$ is minimum.

The procedure to determine A, B and $C_{\rm opt}$ in segment $(\alpha, \beta]$ is given by the following five steps:

1) Let $n = 0$, $C_{opt} = 1$, $ER_{\min} = 1000$, $N = 10^5$, $\Delta = (b - a)/N$.
2) Let $C = \pm 2^{-n}$ compute A, B according to (4.6.1) and (4.6.2) and the square sum of the error $ER(A, B, C)$ over the segment: $ER(A, B, C) = \sum_{i=0}^{N-1} (A + C * (u_i + B)^2 - F(u_i))^2$.
3) If $ER(A, B, C) < ER_{\min}$ then $ER_{\min} = ER(A, B, C)$ and $C_{opt} = C$.
4) $n = n + 1$, if $n < 20$ go to Step 2).
5) For C_{opt} compute the optimum values of A and B according to (4.6.1) and (4.6.2)

The number of segments chosen to implement this second-order approximation is the same as the one used for the linear approximation in scheme-1, scheme-2 and scheme-3.

For the second-order approximation, the parameters A, B, and C are computed using 10^5 sampled data for inputs uniformly spaced in each segment. The parameters A, B, and C are shown in Table VIII. In Table VIII, the first line in each box is parameter A, the second line is parameter B and the third line is parameter C whose absolute value is a power of two which can be seen from the Table VIII.

For some of the elementary functions, for example, $\mathrm{sigm}(u)$, $\mathrm{sigm}'(u)$, $\ln(u)$, $\cdots$, the internal representation of $(u + B)^2$ may be greater than the maximum value of the our internal number system, $2^3 - 2^{-10} \approx 7.999$. This represents an overflow in the internal representation. This difficulty of generating a nonrepresentable number when the number of integer bits in the internal number system is not large enough can be overcome by modifying (4.1) to the following form:

$$H(u) = A \pm 2^{-M} \left(2^{-K} u + D\right)^2 \tag{4.7}$$

where $M = n \bmod 2$, $K = \lfloor n/2 \rfloor$, and $D = B * 2^{-k}$.

The function $\lfloor \cdot \rfloor$ used above is a floor function it truncates the fraction bits and retain only the integer bits. It is verified for each segment by estimating the maximum intermediate number in computation that all the intermediate numbers in the computation of (4.7) are representable in our internal number system for all the elementary functions. An example of using modified second-order approximation equation (4.7) is to compute a sigmoid function for inputs in the segment $[\lambda_0 + 4\Delta, \lambda + 5\Delta] = [0, 1)$, the parameters A, B, C are given in Table VIII to be 1.0556, -4.2220 and -2^{-5}, respectively. The parameters for the modified equation (4.7) are $A = 1.0556$, $M = (5 \bmod 2) = 1$, $K = \lfloor 5/2 \rfloor = 2$, and $D = B * 2^{-2} = -1.0555$ and the modified equation for sigmoid generator for input in the segment [0, 1) becomes: $H(u) = 1.0556 - 2^{-1}(2^{-2}u - 1.0555)^2$.

It can be verified easily that $(2^{-2}u - 1.0555)^2$ does not overflow in the fixed point representation with four integer bits we used for $u \in [0, 1]$. The block diagram of the implementation of the modified second-order approximation given in (4.7) is shown in Fig. 4; parameters A and D are stored in two tables.

From computer simulations with 10^6 sampled function outputs assuming an input uniformly spaced in the input domain of ω, we have examined the elementary functions we considered and computed the average error denoted as AVE-ERR and maximum error denoted as MAX-ERR using the definitions of (2.1) and (2.2). The computed average and maximum errors are shown in Table IX. The last row of the table shows the required table size in unit of bytes.

TABLE VII
AVERAGE AND MAXIMUM ERROR OF FIRST–ORDER APPROXIMATION SCHEMES

	Scheme-1			Scheme-2			Scheme-3		
	Scheme-1			Scheme-2			Scheme-3		
Function	AVE-ERR	MAX-ERR	Table size	AVE-ERR	MAX-ERR	Table size	AVE-ERR	MAX-ERR	Table size
sigm(u)	3.5X10^{-3}	1.8 X10^{-2}	28	4.2 X10^{-3}	2.0 X10^{-2}	14	6.9 X10^{-3}	2.4 X10^{-2}	14
sigm'(u)	2.6X10^{-3}	8.8X10^{-3}	28	2.1X10^{-3}	1.6X10^{-2}	14	5.4X10^{-3}	2.1X10^{-2}	14
sin(u)	5.1X10^{-3}	2.2X10^{-2}	25	1.2X10^{-2}	7.0X10^{-2}	13	3.2X10^{-2}	1.1X10^{-1}	13
cos(u)	4.7X10^{-3}	2.1X10^{-2}	25	1.2X10^{-2}	7.1X10^{-2}	13	4.5X10^{-2}	1.2X10^{-1}	13
ln(u)	1.3X10^{-3}	3.1X10^{-3}	28	3.8X10^{-3}	1.5X10^{-2}	14	2.4X10^{-2}	9.7X10^{-2}	14
e^{-u}	7.1X10^{-3}	1.9X10^{-3}	28	3.4X10^{-3}	1.5X10^{-2}	14	1.1X10^{-2}	3.5X10^{-2}	14
tanh(u)	5.0X10^{-3}	5.7X10^{-2}	28	1.0X10^{-2}	1.7X10^{-1}	14	1.2X10^{-2}	2.4X10^{-1}	14
1/u	1.5X10^{-3}	2.4X10^{-3}	28	2.8X10^{-3}	1.5X10^{-2}	14	1.1X10^{-2}	6.3X10^{-2}	14
$\sqrt{u}$	2.6X10^{-3}	9.5X10^{-2}	28	5.1X10^{-3}	1.1X10^{-1}	14	2.1X10^{-2}	2.0X10^{-1}	14
$1/u^2$	1.7X10^{-3}	5.9X10^{-3}	28	4.2X10^{-3}	2.6X10^{-2}	14	1.1X10^{-2}	4.0X10^{-2}	14

TABLE VIII
PARAMETERS A, B, AND C

Function	u interval	Δ	**Segments**							
			$[\lambda_0, \lambda_0+\Delta)$	$[\lambda_0+\Delta, \lambda_0+2\Delta)$	$[\lambda_0+2\Delta, \lambda_0+3\Delta)$	$[\lambda_0+3\Delta, \lambda_0+4\Delta)$	$[\lambda_0+4\Delta, \lambda_0+5\Delta)$	$[\lambda_0+5\Delta, \lambda_0+6\Delta)$	$[\lambda_0+6\Delta, \lambda_0+7\Delta)$	$[\lambda_0+7\Delta, \lambda_0+8\Delta)$
sigm(u)	(-4,4)	1	0.0156 4.4294 2^{-6}	0.0353 3.6378 2^{-5}	0.0049 3.8922 2^{-5}	-0.0556 4.2220 2^{-5}	1.0556 -4.2220 -2^{-5}	0.9951 -3.8922 -2^{-5}	0.9647 -3.6378 -2^{-5}	0.9844 -4.4294 -2^{-6}
sigm'(u)	[0,8)	1	0.2484 -0.0598 -2^{-4}	-0.1269 -7.4432 2^{-7}	0.0135 -4.4100 2^{-6}	0.0163 -4.3712 2^{-6}	0.0034 -5.8900 2^{-8}	0.0019 -6.5539 2^{-9}	0.0009 -7.2841 2^{-10}	0.0004 -7.7896 2^{-10}
sin(u)	[0,3.14)	0.5	2.1017 -4.1015 -2^{-3}	1.2085 -2.2043 -2^{-2}	0.9986 -1.5634 -2^{-1}	0.9999 -1.5729 -2^{-1}	0.9753 -1.6257 -2^{-1}	1.2267 -0.9129 -2^{-2}	-1.9174 -7.0587 2^{-3}	
cos(u)	[0,3.14)	0.5	0.9995 -0.0041 -2^{-1}	1.1882 0.6048 -2^{-2}	2.0934 2.5223 -2^{-3}	-4.0025 -9.5728 2^{-4}	-1.2247 -3.7964 2^{-2}	-0.9971 -3.1293 2^{-1}	-1.0000 -3.1415 2^{-1}	
ln(u)	[1,2)	0.125	0.5042 -2.0043 -2^{-1}	0.8816 -2.8726 -2^{-2}	0.8529 -2.8370 -2^{-2}	0.8472 -2.8293 -2^{-2}	0.8562 -2.8429 -2^{-2}	1.2259 -4.0585 -2^{-3}	1.2038 -4.0199 -2^{-3}	1.1944 -4.0025 -2^{-3}
e^{-u}	[0,1)	0.125	0.4978 -1.0023 2^{-1}	0.4850 -1.0169 2^{-1}	0.1961 -1.7763 2^{-2}	0.2286 -1.7293 2^{-2}	0.2449 -1.7025 2^{-2}	0.2498 -1.6936 2^{-2}	0.2467 -1.7003 2^{-2}	0.2381 -1.7210 2^{-2}
tanh(u)	(0,8)	1.0	1.0502 -2.0435 -2^{-2}	0.9768 -2.2752 -2^{-3}	0.9938 -2.9676 -2^{-5}	0.9992 -4.0140 -2^{-8}	0.9999 -4.7788 -2^{-10}	1.0000 -5.5377 -2^{-10}	1.0001 -6.5051 -2^{-10}	1.0001 -7.5007 -2^{-10}
1/u	[1,2)	0.125	0.7440 -1.5063 2^{0}	0.5900 -1.8978 2^{-1}	0.5929 -1.8938 2^{-1}	0.4610 -2.4065 2^{-2}	0.4719 -2.3825 2^{-2}	0.4690 -2.3904 2^{-2}	0.3662 -3.0310 2^{-3}	0.3740 -3.0037 2^{-3}
$\sqrt{u}$	[0,1)	0.125	0.3365 -0.1332 -2^{4}	0.6031 -0.4787 -2^{1}	0.9612 -1.2097 -2^{-1}	0.9481 -1.1946 -2^{-1}	1.1952 -1.8971 -2^{-2}	1.1933 -1.8943 -2^{-2}	1.5173 -3.0323 -2^{-3}	1.5019 -3.0038 -2^{-3}
$1/u^2$	[1,2)	0.125	0.5359 -1.4811 2^{1}	0.5290 -1.4871 2^{1}	0.3838 -1.7560 2^{0}	0.2565 -2.1123 2^{-1}	0.2716 -2.0878 2^{-1}	0.1775 -2.5213 2^{-1}	0.1913 -2.4852 2^{-2}	0.1905 -2.4882 2^{-2}

V. BIT SERIAL IMPLEMENTATION OF PIECE-WISE APPROXIMATION SCHEMES

In neural computing, the number of neurons involved in the computation can be large, and a parallel computing unit could be very expensive. Due to the requirement of large neurons in an implementation [9], [10], [16]–[21], a pipelined bit-serial computation provides a cost effective solution for neural-network computing. In bit serial computation and because our internal number system has ten fraction bits and four integer bits, an addition requires 14 machine cycles to compute and a multiplication requires 28 machine cycles. In the bit serial computation data are computed in a pipelined fashion. The worst case delay of the computing is less than the total delay of each computing

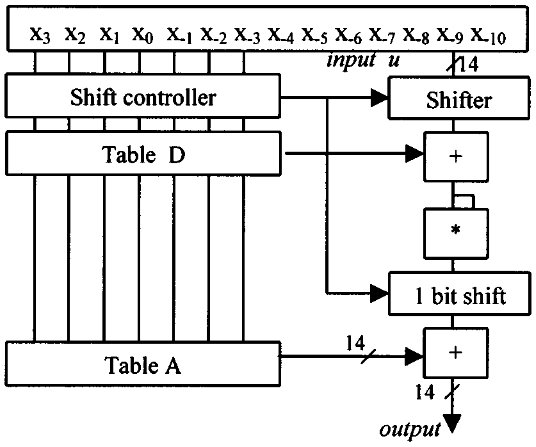

x_3 x_2 x_1 x_0 x_{-1} x_{-2} x_{-3} x_{-4} x_{-5} x_{-6} x_{-7} x_{-8} x_{-9} x_{-10}
input u
14
Shift controller
Shifter
Table D
+
*
1 bit shift
Table A
14
+
14
output

Fig. 4. Block diagram for Scheme-4.

unit because the multiplier/adder can initiate the computation without waiting for the all the input bits to be available.

In Scheme-1 the delay to generate the function output include: the delay of memory access to generate parameters A and C which can be restricted to one machine cycle; the delay of multiplication $C*u$; and the delay for addition. Since we consider to use pipelined bit serial computations, the total delay for the function generator is less than the summation of the delays of memory access, multiplication and addition because the bit serial adder can start to compute without waiting for the multiplier to provide all the input bits. The bit serial adder can start computing as long as the tenth fraction bit of $(C * u)$ is available which takes 11 machine cycles in our internal number system. To generate the 14-bit output for the bit serial adder, 14 more machine cycles are needed. Consequently the total delay to generate the function output is: 1(memory access) $+$ 11(multiplication) $+$ 14(addition) $=$ 26 machine cycles.

In scheme-2, the computation of delay is similar to the previous scheme. It is estimated that the maximum delay for scheme-2 is 24 machine cycles. In scheme-3, the computation delay includes the delays for the exclusive-or (XOR) logic, shifter, and set–reset logic. The total delay for the scheme-3 has been estimated to be 25 machine cycles.

The delay for the second-order approximation involves the delay of shifter representing the power of two multiplication of 2^{-K}. The integer K is defined in(4.7) as $K = \lfloor n/2 \rfloor$. From Table VIII, the maximum value of n is ten thus the maximum value of K is five. Consequently the maximum delay for the shifter to implement the power of two multiplication is five machine cycles. The worst case delay for scheme-4 is estimated to be 32 machine cycles. The maximum computation delays for the four proposed schemes are summarized as follows (the computation units are designed in a pipelined bit serial fashion): Scheme-1(first order) $=$ 26, Scheme-2(first order) $=$ 24, Scheme-3(first order) $=$ 25, and Scheme-4(second order) $=$ 32.

VI. Evaluation

For our evaluations we consider two quantities: the average error and maximum error.[2] We consider the implementation of the first-order approximation scheme proposed in [5], [6] and implemented by Murtagh/Tsoi [4] and the second-/third-order approximations proposed in [3]. For convenience of comparisons, we denote the first-order approximation scheme-1 used for sigmoid generations as S1, and we denote S2, S3 and S4 for the sigmoid generators using the first-order approximations scheme-2, scheme-3 and the second-order approximation scheme-4, respectively. We also denote K2 and K3 as the second– and third-order approaches, respectively, which are proposed in [3], and we denote PS16 and PS8 for the existing first-order approximation approach implemented in [7] for the sigmoid generators. The estimated hardware requirements (including multipliers and adders) and precision are shown in Table X further discussion regarding PS16, PS8, K2 and K3 can be found in [22].

Based on the estimations of Table X, we derived Table XI reporting the ratios of performance and hardware requirement for our proposed sigmoid generators (denoted as schemes S1, S2, S3 and S4) and the existing sigmoid generators (denoted as schemes PS16, PS8, K2 and K3) in terms of average error, maximum error, lookup table size and the worst case delay. In Table XI, AVE-ERR is the average error, MAX-ERR is the maximum error, "Table" is the lookup table size and the delay is the worst case computation delay. If the ratio is less than 1.00, then the proposed scheme outperforms the existing scheme for the corresponding parameter given on the left column of the Table XI.

From the Table XI, it is noticed that all of our proposals for the sigmoid generator have smaller average error than the existing schemes. In particular, the average error of the proposal S1 is as small as 4% of the scheme denoted as K3 and as much as 41% of the scheme PS16 as shown in Table XI. The maximum error and the worst case delay of the proposal S1 is less than the existing schemes with a speed up that varies between 1.23 and 1.81. The proposal S1 requires larger lookup table size than the schemes K2 and K3, however compared to the gain of the implemented function average error (5% of the scheme K2 and 4% of the scheme K3), the cost of lookup tables is worthwhile. The proposal S2 requires the least table size in our four proposals and outperforms the existing schemes with a speedup up to 1.96. The proposal S3 has the best performance and outperforms the existing schemes with a speedup from 1.28 to 1.88. The proposal S4 have the least average and maximum errors among all the proposals. The four proposals S1, S2, S3, and S4 provide different tradeoff between the average and maximum errors, table size, and the worst case delay and outperform the existing schemes.

VII. Concluding Remarks

We have presented four schemes for the generation of elementary functions using first-order and second-order approximations. An important aspect of the proposed schemes is the capability to accommodate the functions we propose and possibly an arbitrary number of additional functions with a common design to all functions. By loading the values that are unique to individual functions, high performance and very small cost can be achieved, making the proposed schemes suitable for a large number of hardwired neural application and extremely valuable for the design

[2]Additional evaluations are reported in [23].

TABLE IX
AVERAGE AND MAXIMUM ERRORS OF SECOND-ORDER APPROXIMATION SCHEME-4

Parameter	*sigm(u)*	*sigm'(u)*	*sin(u)*	*cos(u)*	*ln(u)*	e^{-u}	*tanh(u)*	*1/u*	$\sqrt{u}$	$1/u^2$
AVE-ERR	2.6×10^{-3}	5.0×10^{-4}	1.0×10^{-3}	9.1×10^{-4}	5.1×10^{-4}	3.4×10^{-4}	1.6×10^{-3}	5.8×10^{-4}	1.2×10^{-3}	4.8×10^{-4}
MAX-ERR	1.8×10^{-2}	4.6×10^{-3}	5.5×10^{-3}	5.5×10^{-3}	1.4×10^{-3}	1.2×10^{-3}	1.6×10^{-2}	1.3×10^{-3}	5.3×10^{-2}	2.0×10^{-3}
Table size	28	28	25	25	28	28	28	28	28	28

TABLE X
COMPARISONS OF SIGMOID GENERATORS

Scheme	Average Error	Maximum Error	Table Size	Delay	Mult/Add
PS16	8.6×10^{-3}	1.9×10^{-2}	28	32	0 / 0
PS8	2.5×10^{-2}	4.9×10^{-2}	14	32	0 / 0
K2	6.4×10^{-2}	1.6×10^{-1}	4	33	1 / 2
K3	7.8×10^{-2}	2.0×10^{-1}	4	47	2 / 2
S1	3.5×10^{-3}	1.8×10^{-2}	28	26	1 / 1
S2	4.2×10^{-3}	2.0×10^{-2}	14	24	0 / 1
S3	6.9×10^{-3}	2.4×10^{-2}	14	25	0 / 0
S4	2.6×10^{-3}	1.8×10^{-2}	28	32	1 / 2

TABLE XI
RATIOS FOR S1, S2, S3, AND S4 OVER SCHEMES PS16, PS8, K2, AND K3

	S1				S2				S3				S4			
	PS16	PS8	K2	K3	PS16	PS8	K2	K2	PS16	PS8	K2	K2	PS16	PS8	K2	K2
ave_err	0.41	0.14	0.05	0.04	0.49	0.17	0.07	0.05	0.80	0.28	0.11	0.09	0.30	0.10	0.04	0.03
max_err	0.95	0.37	0.11	0.09	1.05	0.41	0.13	0.10	1.26	0.49	0.15	0.12	0.95	0.37	0.11	0.09
Table	1.00	2.00	7.00	7.00	0.50	1.00	3.50	3.50	0.50	1.00	3.50	3.50	1.00	2.00	7.00	7.00
Delay	0.81	0.81	0.79	0.55	0.78	0.78	0.76	0.53	0.75	0.75	0.73	0.51	1.00	1.00	0.97	0.68

TABLE XII
HARDWARE REQUIREMENT OF THE SCHEMES

Hardware	Scheme-1	Scheme-2	Scheme-3	Scheme-4
multiplication	1	0	0	1
addition	1	1	0	2
shifter	0	1	1	1
exclusive-or	0	0	1	0
flip-flop	0	0	1	0
lookup table	25-28 bytes	13-14 bytes	13-14 bytes	25-28 bytes

of neural emulators advancing them to a more general purpose environment. We have established that under the prespecified criteria of having an average error not exceeding 10^{-3} (i.e., a representation error of around 2^{-10}) and a maximum error of 10^{-2}, two schemes for all functions can satisfy the requirement. The proposed schemes provide wide tradeoff selections between the hardware requirement and the achieved function precision. In addition to registers that may be needed to hold input–output values, the required hardware for the four schemes includes: multiplication, addition, bit-shifter, exclusive-or, flip-flop, and lookup tables as shown in Table XII.

The overall conclusion of the investigation is that the second-order approximation and one of the first-order schemes have the potential to be used as a general purpose devices to accommodate potentially a larger number of functions other than the ones we consider. The third first-order approximation scheme can be used successfully for sigmoid generators and the second scheme can be used for more than half of the functions we considered. Finally we note that the eight segment design was chosen to facilitate the presentation. The proposed schemes are general and more segments can be used to improve the precision, with proper consideration, at the expense of additional memory.

REFERENCES

[1] J. L. Holt and J. N. Hwang, "Finite precision error analysis of neural network hardware implementations," *IEEE Trans. Comput.*, vol. 42, pp. 281–290, Mar. 1993.

[2] W. Lin, J. G. Delgado-Frias, S. Vassiliadis, and G. G. Pechanek, "Machine and bit precision influence on the Hopfield-Tank model for the TSP," in *Proc. Int. Joint Conf. Neural Networks*, Japan, Oct. 1993.

[3] H. K. Kwan, "Digital implementation methods for nonlinear functions in neural networks," IEEE Trans. Neural Networks, to be published.

[4] A. Murtagh and A. C. Tsoi, "Implementation issues of sigmoid function and its derivative for VLSI digital neural networks," in *Proc. Inst. Elect. Eng.*, vol. 139, Pt. E, May 1992, pp. 207–214.

[5] C. Alippi and G. Storti-Gajani, "Simple approximation of sigmoid functions: Realistic design of digital VLSI neural networks," in *Int. Symp. Circuits Syst.*, 1991, pp. 1505–1508.

[6] D. J. Myers and G. Storti-Gajani, "Efficient implementation of piecewise linear activation function for digital VLSI neural networks," *Electron. Lett.*, vol. 25, no. 24, pp. 1662–1663, Nov. 1989.

[7] W. Byrne, "Alternating minimization and Boltzmann machine learning," *IEEE Trans. Neural Networks*, vol. 3, pp. 612–620, July 1992.

[8] T. A. Brubaker and J. C. Becker, "Multiplication using logarithms implemented with read-only memory," *IEEE Trans. Comput.*, vol. 24, Aug. 1975.

[9] P. Treleaven, M. Pacheco, and M. Vellasco, "VLSI architectures for neural networks," *IEEE Micro.*, pp. 8–27, Dec. 1989.

[10] P. Treleaven and M. Vellasco, "Neural networks on silicon," in *Wafer Scale Integration, III*, M. Sami and F. Distante, Eds. Amsterdam, The Netherlands: Elsevier, 1990.

[11] M. Zhang, J. G. Delgado-Frias, S. Vassiliadis, and G. G. Pechanek, "Hardwired sigmoid generator," IBM Endicott, NY, IBM Tech. Rep. TR 01.C492, Dec. 1992.

[12] P. T. Tang, "Table-driven implementation of the exponential function for IEEE floating-point arithmetic," *ACM Trans. Math. Software*, vol. 15, no. 2, pp. 144–157, June 1989.

[13] S. Gal and B. Bachelis, "An accurate elementary mathematical library for the IEEE floating point standard," *ACM Trans. Math. Software*, vol. 17, no. 1, pp. 26–45, March 1991.

[14] T. C. Chen, "Automatic computations of exp, log, ratios and square," *IBM J. Res. Dev.*, p. 380, July 1972.

[15] S. Vassiliadis, D. S. Lemon, and M. Putrino, "AS/370 sign-magnitude floating point adder," *IEEE J. Solid-State Circuits*, vol. 24, pp. 1062–1070, Aug. 1989.

[16] D. Hammerstrom, "A VLSI architecture for high-performance, low-cost, on-chip learning," in *IEEE Int. Joint Conf. Neural Networks*, vol. II, June 1990, pp. 537–544.

[17] U. Ramacher, M. Wesseling, and K. Goser, "WSI architecture of a neurocomputer module," in *IEEE Int. Conf. Wafer Scale Integration*, 1990, pp. 124–130.

[18] F. Blayo and P. Hurat, "A VLSI systolic array dedicated to Hopfield neural network," in *VLSI Artificial Intell.*, J. Delgado-Frias and W. Moore, Eds. Boston, MA: Kluwer, 1989, pp. 255–264.

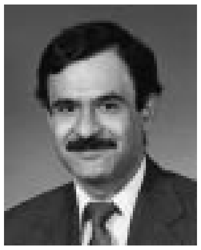
[19] S. Vassiliadis, G. Pechanek, and J. Delgado-Frias, "SPIN: A sequential pipelined neurocomputer," in *Proc. IEEE Int. Conf. Tools Artificial Intell.*, San Jose, CA, Nov. 1991.

[20] G. G. Pechanek, S. Vassiliadis, and J. G. Delgado-Frias, "Digital neural emulators using tree accumulation and communication structures," *IEEE Trans. Neural Networks*, vol. 3, pp. 934–950, Nov. 1992.

[21] J. G. Delgado-Frias, S. Vassiliadis, G. G. Pechanek, W. Lin, S. Barber, and H. Ding, "A VLSI pipelined neuroemulator," in *VLSI for Neural Networks and Artificial Intelligence*, J. Delgado-Frias and W. Moore, Eds. New York: Plenum, 1993.

[22] S. Vassiliadis, J. G. Delgado-Frias, and M. Zhang, "First-order piece-wise sigmoid generator," *Artificial Neural Networks in Engineering*, Nov. 1993.

[23] S. Vassiliadis, M. Zhang, and J. G. Delgado-Frias, "Elementary function generators for neural network emulators," IBM, Austin, TX, IBM Tech. Rep. TR 51.0802, Dec. 1993.



Stamatis Vassiliadis (M'86–SM'92–F'97) received the Dr.Eng. degree in electronic engineering from Politecnico di Milano, Milan, Italy, and the Ph.D. degree in computer science from the University of Namur, Namur, Belgium.

He is a Professor in the Electrical Engineering Department of Delft University of Technology, Delft, The Netherlands. He has also served in the faculties of Cornell University, Ithaca, NY, and the State University of New York, Binghamton. He worked for a decade with IBM in the Advanced Workstations and Systemslaboratory in Austin, TX, the Mid-Hudson Valley Laboratory, Poughkeepsie, NY, and the Glendale laboratory in Endicott, NY. At IBM he was involved in a number of projects regarding computer design, organizations, and architectures and the leadership to advanced research projects. He has been involved in the design and implementation of several computer systems including the IBM 9370 model 60. A number of his inventions have been implemented in commercially available systems and processors, including the IBM Power II, the IBM AS/400 Models 400, 500, and 510, Server Models 40S and 50S, and the IBM AS/400 Advanced 36. His research interests include computer architecture, embedded systems, hardware design and functional testing of computer systems, parallel processors, computer arithmetic, EDFI for hardware implementations, and neural networks.

Dr. Vassiliadis received numerous awards for his work, including 23 levels of Publication Achievement Awards, 15 levels of Invention Achievement Awards and an Outstanding Innovation Award for Engineering/Scientific Hardware Design in 1989. Six of his 67 patents have been rated with the highest patent ranking in IBM; in 1990 he was awarded the highest number of patents in IBM.

Ming Zhang (S'88–M'93) was born in Heilongjiang province, China, on May 19, 1963. He received the B.S. degree in electrical engineering from the University of Science and Technology of China, Hefei, in 1982, the M.S. degree in Circuits and Systems from the Chinese Academy of Sciences, Beijing, in 1985, and the Ph.D. degree in advanced technology from the State University of New York, Binghamton, in 1993.

Since September 1993, he has been with AT&T Wireless Services as a Senior Member of the technical team. His current research interests include neural network applications in wireless system modeling and simulation, voice over IP technology, and third-generation wireless technologies.

José G. Delgado-Frias (S'81–M'81–SM'90) received the B.S. degree from the National Autonomous University of Mexico, the M.S. degree from the National Institute for Astrophysics, Optics, and Electronics, Mexico, and the Ph.D. degree from Texas A&M University, College Station, all in electrical engineering.

Since Fall 2000, he has been a faculty member in the Electrical Engineering Department at The University of Virginia, Charlottesville. Prior to this appointment, he was a faculty member in the Electrical Engineering Department, State University of New York, Binghamton, from 1989 to 2000. He was a Postdoctoral Research Fellow with the University of Oxford, U.K. His research interests include computer architecture/organization, interconnection networks, VLSI design, neural networks, and optimization using genetic algorithms. He has coauthored more than 80 technical papers and coedited three books. He has been granted more than 25 U.S., European, and World patents.

Dr. Delgado-Frias is a member of the Association for Computing Machinery, the American Society for Engineering Education, and Sigma Xi. In 1994, he received the State University of New York System Chancellor's Award for Excellence in Teaching.