Brief Contributions

Sigmoid Generators for Neural Computing Using Piecewise Approximations

Ming Zhang, Stamatis Vassiliadis, and José G. Delgado-Frias

Abstract—A piecewise second order approximation scheme is proposed for computing the sigmoid function. The scheme provides high performance with low implementation cost; thus, it is suitable for hardwired cost effective neural emulators. It is shown that an implementation of the sigmoid generator outperforms, in both precision and speed, existing schemes using a bit serial pipelined implementation. The proposed generator requires one multiplication, no look-up table and no addition. It has been estimated that the sigmoid output is generated with a maximum computation delay of 21 bit serial machine cycles representing a speedup of 1.57 to 2.23 over other proposals.

Index Terms—Nonlinear function generators, sigmoid function, piecewise approximations, neural networks, hardware for twos complement notation, error analysis.

1 Introduction

FOR a cost-effective digital neural network hardware implementation, generating the sigmoid function described by equation 1.1 [1] in low precision with high performance is important to reduce the system cost [2].

$$\text{sigm}(u) = \frac{1}{1+e^{-u}} \tag{1.1}$$

In assuming "inexpensive" hardware for the implementation of sigmoid generators, piecewise approximation approaches have been proven to provide a good choice on the trade off between the hardware requirements and the function precision [3], [4], [5], [6].

Piecewise approximations can be roughly divided into linear and higher order approximations. Using piecewise linear approximations some schemes have been proposed and implemented [3], [5]. In particular it has been shown in [3], [5], [7] that when assuming eight and 16 segments the estimated delay of the implementation is 32 pipelined bit serial machine cycles. The eight segment implementation requires an estimated table of 14 bytes of storage while producing an average error of 2.5×10^{-2} and a maximum error of 4.9×10^{-2}. The 16 segment implementation requiries 28 bytes of storage while producing an average error of 8.6×10^{-3} and a maximum error of 1.9×10^{-2}.

We have proposed piecewise linear approximation schemes [7] that have the same memory requirements when compared to the schemes described in [3], [5] with a number of improvements. First, the proposed schemes reduce the delay from 32 cycles to 24-25 cycles depending on the number of segments considered. Second, they reduce the average error to the order of 10^{-3} from the average of 10^{-2} for an eight segment implementation while producing a comparable maximum error of 2.0×10^{-2}. Third, for a 16 segment implementation our schemes [7] reduce the average error by half over the 16 segment implementation of the scheme presented in [5].

In this paper, we investigate a second order approximation scheme that substantially improves higher order piecewise approximations. In particular we propose a scheme that requires no memory as well as reduces substantially the delay and the average and maximum errors for the implementation. The average and maximum errors are comparable to the first order approximation schemes while improving the speed. Furthermore, we show that our scheme can be used for sign magnitude as well as twos complement notations which are two of the most commonly used number representations. Our scheme can be easily extended to further reduce the error when compared to first order approximations.

The paper is organized as follows. In Section 2, we define the internal number system used for second order approximation. In Section 3, the second order approximation scheme which requires one multiplication is proposed. The performance and function precision are evaluated and compared to a bit serial implementation of the scheme in Section 4. We conclude this paper with some remarks in Section 5.

2 Number System Representation

The number representation to describe our approach, denoted as the internal number system, employs a fixed point fractional number system with four integer and 10 fraction bits (shown below).

$x_3x_2x_1x_0$	$x_{-1}x_{-2}x_{-3}x_{-4}\ x_{-5}x_{-6}x_{-7}x_{-8}\ x_{-9}x_{-10}$

↑ radix point

The most significant bit x_3 of the internal number system is a sign bit. The sign bit is set to 1 if the number is negative and set to 0 if the number is positive. It should be noted that the choice of the length of the representation relates to the evaluation we conducted on ideal computations considering only the method error. Our evaluation suggested that the method error alone will be greater than 10^{-3} suggesting that the internal representation should be limited to 10 fractional bits which produces a representation error 2^{-10} that is also in the order of 10^{-3}. We note here that a larger length will not improve substantially the error because the method error will be the dominant error thus considering larger fractions may not justify the expense. Furthermore, note that the scheme we propose is applicable to both sign magnitude and twos complement notations. The choice of one versus the other is entirely dependent on implementation considerations and not discussed any further. In the presentation, when necessary, appropriate discussion is reported. The maximum value to be represented by the internal number system is $2^3 - 2^{-10} \approx 7.999$. For example, assuming sign magnitude internal number system, two numbers –2.6 and 2.6 are represented as:

$$-2.6 \approx 1010.1001\ 1001\ 10$$
$$2.6 \approx 0010.1001\ 1001\ 10$$

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

• *M. Zhang is with AT&T Wireless Services, INC., Kirkland, WA 98033.*
• *S. Vassiliadis is with the Department of Electrical Engineering, Delft University of Technology, Delft, The Netherlands.*
• *J.G. Delgado-Frias is with the Department of Electrical Engineering, State University of New York, Binghamton, NY 13902-6000. E-mail: jdf@parallel.ee.binghamton.edu.*

Manuscript received Jan. 4, 1994; revised Apr. 29, 1996.
For information on obtaining reprints of this article, please send e-mail to: transcom@computer.org, and reference IEEECS Log Number C96073.

the changing bits. For example assuming that α is represented by α = 0010.xxxx xxxx xx the first four bits are the known bits having the binary value 0010 while the remaining bits are changing bits. The symbol "x" is used to denote a changing bit.

The average and maximum errors are used to evaluate the precision of the sigmoid function generator. In this paper, if a function F(u) is approximated by function H(u) with $u \in [\lambda_0, \lambda_1)$, then the error of this approximation is $|H(u) - F(u)|$ and the average and maximum errors are obtained by uniformly sampling u on 10^6 points in the domain of (λ_0, λ_1). The average and maximum errors are expressed as:

$$\begin{cases} \text{Average Error} = \dfrac{\sum_{i=0}^{10^6-1} |H(u_i) - F(u_i)|}{10^6} & (2.1) \\ \text{Maximum Error} = \max\limits_{\lambda_0 < u_i < \lambda_1} |H(u_i) - F(u_i)| & (2.2) \end{cases}$$

3 Second Order Approximation

In this scheme the sigmoid function inputs are divided into segments and a second order approximation is used for the computation in each segment. For a segment $[\alpha, \beta]$, if the input u is in the interval $u \in [\alpha, \beta]$, in general a second order approximation implies that the sigmoid function can be computed by $c_0 + c_1 * u + c_2 * u^2 = c_0 + u * (c_1 + c_2 * u)$. This operation requires two multiplications and two additions. A second order approximation method, we proposed in [7], requires only one multiplication and two additions. This method provides the basis for the schemes proposed here. The equation for this second order sigmoid generator is given below:

$$H(u) = A + C * (u + B)^2 \qquad \text{where } |C| = 2^{-n} \tag{3.1}$$

Since C is a power of two, the multiplication with C can be computed by bit shifting. Consequently only one multiplication is required to compute (3.1), simplifying the design.

We compute the ideal function outputs $F(u_i)$ and its second order approximation $H(u_i)$ on N uniformly spaced inputs u_i $(0 \le i < N)$ in the segment $[\alpha, \beta]$, where u_i are defined to be:

$$\begin{cases} \Delta = (\beta - \alpha)/N \\ u_i = \alpha + i * \Delta \end{cases} \quad i = 0, 1, \ldots, N-1 \tag{3.2}$$

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

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

The least squares method can be used to find the parameters A, B and C. When C is given, the minimum value of ER(A, B, C) exists when $\frac{\partial ER}{\partial A} = 0$ and $\frac{\partial ER}{\partial B} = 0$. Then we have that

$$\begin{cases} \dfrac{\partial ER}{\partial A} = 2\sum_{i=0}^{N-1} H(u_i) - F(u_i) = 0 & (3.4.1) \\ \dfrac{\partial ER}{\partial B} = 4C\sum_{i=0}^{N-1} \left(H(u_i) - F(u_i)\right)(u_i + B) = 0 & (3.4.2) \end{cases}$$

From (3.4.1) it can be obtained that $4B*C*\sum_{i=0}^{N-1} H(u_i) - F(u_i) = 0$

Then substitute $H(u_i) = A + C * (u_i + B)^2$ into (3.4),

$$\begin{cases} \sum_{i=0}^{N-1} A + C*(u_i + B)^2 - F(u_i) = 0 & (3.5.1) \\ \sum_{i=0}^{N-1} \left(A + C*(u_i + B)^2 - F(u_i)\right)u_i = 0 & (3.5.2) \end{cases}$$

Hence

$$\begin{cases} \sum_{i=0}^{N-1} A + C*u_1^2 + 2C*B*u_i + C*B^2 - F(u_i) = 0 & (3.6.1) \\ \sum_{i=0}^{N-1} A*U_i + C*u_1^3 + 2C*B*u_1^2 + C*B^2 u_i - F(u_i)*u_i = 0 & (3.6.2) \end{cases}$$

and

$$\begin{cases} N*A + N*C*\overline{X^2} + 2N*B*C*\overline{X} + N*C*B^2 - N*\overline{Y} = 0 & (3.7.1) \\ N*A*\overline{X} + N*C*\overline{X^3} + 2N*B*C*\overline{X^2} + N*C*B^2\,\overline{X} - N*\overline{XY} = 0 & (3.7.2) \end{cases}$$

where

$$\begin{cases} \overline{X} = \dfrac{\sum_{i=0}^{N-1} u_i}{N} \quad \overline{X^2} = \dfrac{\sum_{i=0}^{N-1} u_i^2}{N} \quad \overline{X^3} = \dfrac{\sum_{i=0}^{N-1} u_i^3}{N} \\ \overline{Y} = \dfrac{\sum_{i=0}^{N-1} F(u_i)}{N} \quad \overline{XY} = \dfrac{\sum_{i=0}^{N-1} u_i * F(u_i)}{N} \end{cases}$$

A and B can be determined from (3.7.1) and (3.7.2) to be

$$\begin{cases} B = \dfrac{\left(\overline{X}*\overline{Y} - \overline{XY}\right) + C*\left(\overline{X^3} - \overline{X^2}*\overline{X}\right)}{2C*\left(\overline{X^2} - \overline{X}^2\right)} & (3.8.1) \\ A = \overline{Y} - C*\overline{X^2} - 2B*C*\overline{X} - C*B^2 & (3.8.2) \end{cases}$$

For $|C| = 2^{-n}$ with n being an integer, the larger the number of sample points N is, the more precise the A and B are estimated. Here we chose N to be 10^5. For n greater than 18 the second order term $2^{-n} * (x + B)^2$ will be 0 in our internal number system as a result of shifting 18 bits to compute the multiplication by 2^{-n}. Consequently we examine the square error of ER(A, B, C) for $C = \pm 2^{-n}$ with n changing from 0 to 18. We exhaustively search the minimum value of ER(A, B, C) for $C = \pm 2^{-n}$ with n varying from 0 to 18 to identify the number C_{opt} such that ER(A, B, C_{opt}) is minimum.

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

step1: Let n = 0, $C_{opt} = 1$, $ER_{min} = 1{,}000$, $N = 10^5$,

$$\Delta = \frac{(\beta - \alpha)}{N}$$

step2: Let $C = \pm 2^{-n}$ Compute A, B according to (3.8.1) and (3.8.2) and the square sum of the error ER(A, B, C) over the segment:

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

step3: If ER(A, B, C) < ER_{min} then ER_{min} = ER(A, B, C) and C_{opt} = C

step4: n = n + 1, if n < 18 go to step2

step5: For C_{opt} compute the optimum values of A and B according to (3.8.1) and (3.8.2)

Using these steps the values of the parameters A, B, and C can be estimated for the second order approximations; however, these values may require the ability to handle larger numbers than the

internal number representation we adopted. To overcome this problem, we have modified (3.1) to become:

$$y = H(u) = A \pm 2^{-M}(2^{-K}u + D)^2 \tag{3.9}$$

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

By estimating the maximum intermediate number in the computations, all the intermediate numbers in the computation of (3.9) are representable in our internal number system for the sigmoid function generator. The second order approximation can be implemented by dividing the inputs into two segments to be (–4, 0) and [0, 4). The sigmoid function output is forced to values of 0 and 1 when the input goes beyond these two segments; this in turn produces a maximum error of the same order of magnitude as the method error. For this implementation we use the modified version (3.9) which is computed as follows:

$$H(u) = \begin{cases} 0.01964 + 2^{-1}\left(2^{-2} * u + 0.96377\right)^2 & -4 < u < 0 \\ 0.98046 - 2^{-1}\left(2^{-2} * u - 0.96377\right)^2 & 0 \le u < 4 \end{cases} \tag{3.10}$$

$$\approx \begin{cases} 2^{-1}\left(1 - \left|2^{-2} * u\right|\right)^2 & -4 < u < 0 \\ 1 - 2^{-1}\left(1 - \left|2^{-2} * u\right|\right)^2 & 0 \le u < 4 \end{cases} \tag{3.11}$$

3.1 Sigmoid Function in the Sign Magnitude Notation

In the sign magnitude notation, the input u can be expressed as:

$$u = x_3 0 x_1 x_0 . x_{-1} x_{-2} x_{-3} x_{-4} x_{-5} x_{-6} x_{-7} x_{-8} x_{-9} x_{-10}$$

where $x_2 = 0$ because we assume the input $|u| < 4$ for the two segment implementation of the second order approximation. Then

$$|2^{-2} * u| = 0000.x_1 x_0 x_{-1} x_{-2} x_{-3} \ldots . x_{-8}$$

We denote

$$H_0 = 1 - |2^{-2} * u| = 0000.1111\ 1111\ 11 - 0000.x_1 x_0 x_{-1} x_{-2} x_{-3} \ldots . x_{-8} + 0000.0000\ 0000\ 01$$

$$= 0000.\ \overline{x_1}\ \overline{x_0}\ \overline{x_{-1}}\ \overline{x_{-2}}\ \overline{x_{-3}}\ \ldots .\overline{x_{-8}} + 0000.0000\ \ 0000\ \ 01$$

$$= 0000\ \overline{x_1}\ \overline{x_0}\ \overline{x_{-1}}\ \overline{x_{-2}}\ \overline{x_{-3}}\ \ldots .\overline{x_{-8}} \tag{3.12.1}$$

$$H_1 = 2^{-1} * H_0 * H_0 \tag{3.12.2}$$

$$H_2 = 1 - H_1 \tag{3.12.3}$$

Then $0 \le H_0 < 1$ and $0 \le H_1 < 0.5$. If H_1 is expressed in internal number system as

$$H_1 = 0000.h_{-1} h_{-2} h_{-3} \ldots\ldots h_{-10}$$

H_2 satisfies:

$$H_2 = 1 - H_1 = 0000.1111\ 1111\ 11 - 0.h_{-1} h_{-2} h_{-3} \ldots\ldots h_{-10} + 0000.0000\ 0000\ 01$$

$$= 0000.\ \overline{h_{-1} h_{-2} h_{-3}}\ \ldots .\overline{h_{-10}} + 0000.0000\ 0000\ 01$$

$$= 0000.\ \overline{h_{-1}}\ \overline{h_{-2}}\ \overline{h_{-3}}\ \ldots .\overline{h_{-10}}$$

Combining (3.11) and (3.12) the sigmoid function can be approximated by

$$H(u) = \begin{cases} H_1 & x_3 = 1 \\ H_2 & x_3 = 0 \end{cases}$$

$$\approx \begin{cases} 0000.\ h_{-1}\ \ h_{-2}\ \ h_{-3}\ \ldots .\ h_{-10} & x_3 = 1 \\ 0000.\ \overline{h_{-1}}\ \ \overline{h_{-2}}\ \ \overline{h_{-3}}\ \ldots .\ \overline{h_{-10}} & x_3 = 0 \end{cases} \tag{3.13}$$

The computation (3.13) can be summarized by the following:

$$H(u) = \overline{x_3} \oplus H_1 = \overline{x_3} \oplus \left(2^{-1} * \left(\overline{2^{-2} * u}\right)^2\right) \tag{3.14}$$

It should be noted that all the operations in (3.14), for instance inverse and exclusive-or, are for the fraction bits and the integer bits are always set to 0. The two segment implementation requires no lookup table, its hardware data flow diagram is shown in Fig. 1.

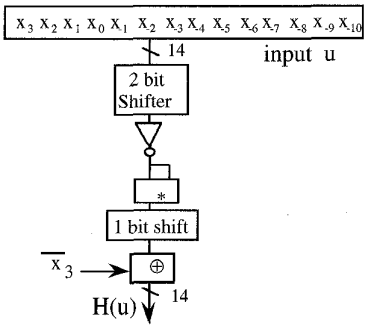

Fig. 1. Sign magnitude second order approximation in two segments.

3.2 Sigmoid Function in the Twos Complement Notation

In twos complement notation, H(u) is computed by (3.11):

$$H(u) \approx \begin{cases} 2^{-1}(1 + 2^{-2} * u)^2 & -4 < u < 0 \\ 1 - 2^{-1}(1 - 2^{-2} * u)^2 & 0 \le u < 4 \end{cases}$$

if the input u satisfies $-4 < u < 0$, then u can be expressed in twos complement notation as:

$$u = 11 x_1 x_0 . x_{-1} x_{-2} x_{-3} x_{-4} x_{-5} x_{-6} x_{-7} x_{-8} x_{-9} x_{-10}$$

then

$$1 + 2^{-2} * u = 00.x_1 x_0 x_{-1} x_{-2} x_{-3} \ldots . x_{-8}$$

if $0 \le u < 4$, then u can be expressed in twos complement notation as:

$$u = 00 x_1 x_0 . x_{-1} x_{-2} x_{-3} x_{-4} x_{-5} x_{-6} x_{-7} x_{-8} x_{-9} x_{-10}$$

then $2^{-2} * u = 0000.x_1 x_0 x_{-1} x_{-2} x_{-3} \ldots . x_{-8}$

and

$$1 - 2^{-2} * u \approx 0000.1111\ 1111\ 11 - 0000.x_1 x_0 x_{-1} x_{-2} x_{-3} \ldots . x_{-8} + 2^{-10}$$

$$\approx 00.\ \overline{x_1}\ \overline{x_0}\ \overline{x_{-1}}\ \overline{x_{-2}}\ \overline{x_{-3}}\ \ldots .\ \overline{x_{-8}}$$

Consequently

$$H(u) \approx \begin{cases} 2^{-1}(0000.X_1 X_0 X_{-1} X_{-2} X_{-3} \ldots X_{-8})^2 & -4 < u < 0 \text{ or } x_3 = 1 \\ 1 - 2^{-1}(0000.\overline{X_1}\ \overline{X_0 X_1 X_2}\ \overline{X_{-3}} \ldots \overline{X_{-8}})^2 & 0 \le u < 4 \text{ or } x_3 = 0 \end{cases} \tag{3.15}$$

We denote two numbers in twos complement notations K_0 and K to be:

$$K_0 = 00.x_1 x_0 x_{-1} x_{-2} x_{-3} \ldots . x_8 = 2^{-2} u$$

$$K = \overline{x_3} \oplus K_0$$

where the bit by bit exclusive-or $\oplus$ and the bit shift 2^{-2} is applied only on the fraction bits of K, their integer bits are always 0. Then (3.15) becomes

$$H(u) \approx \begin{cases} 2^{-1}(\overline{x_3} \oplus K_0)^2 & x_3 = 1 \\ 1 - 2^{-1}(\overline{x_3} \oplus K_0)^2 & x_3 = 0 \end{cases}$$

$$= \begin{cases} 2^{-1}(K)^2 & x_3 = 1 \\ 1 - 2^{-1}(K)^2 & x_3 = 1 \end{cases} \tag{3.16}$$

We denote

$$K_2 = 2^{-1} * K * K$$

$$K_3 = 1 - K_2$$

and assume K_2 be expressed in the internal number system as

$$K_2 = 0000.k_{-1}k_{-2}k_{-3}.....k_{-10} \tag{3.17}$$

then K_3 satisfies:

$$K3 = 1 - K2 = 0000.1111\ 1111\ 11 - 0.k_{-1}k_{-2}k_{-3}.....k_{-10} + 2^{-10}$$
$$= 0000.\overline{k_{-1}}\ \overline{k_{-2}}\ \overline{k_{-3}}\ ....\overline{k_{-10}} + 2^{-10} \approx 0000.\overline{k_{-1}}\ \overline{k_{-2}}\ \overline{k_{-3}}....\overline{k_{-10}} \tag{3.18}$$

Combining (3.16), (3.17), and (3.18) the sigmoid function can be approximated in twos complement notation by

$$H(u) = \begin{cases} K_2 & x_3 = 1 \\ K_3 & x_3 = 0 \end{cases}$$
$$\approx \begin{cases} 0000.k_{-1}k_{-2}k_{-3}.....k_{-10} & x_3 = 1 \\ 0000.\overline{k_{-1}}\ \overline{k_{-2}}\ \overline{k_{-3}}....\overline{k_{-10}} & x_3 = 0 \end{cases} \tag{3.19}$$

The computation (3.19) can be summarized by the following:

$$H(u) - \overline{x_3} \oplus K_2 = \overline{x_3} \oplus \left(2^{-1} * \left(\overline{x_3} \oplus 2^{-2}u\right)^2\right) \tag{3.20}$$

It should be noted that all the operations in (3.20), e.g., shifting, inverse and exclusive-or generate only fraction bits, the output of the integer bits for each operation are always set to 0. The two segment implementation requires no lookup table; Fig. 2 shows the hardware data flow diagram.

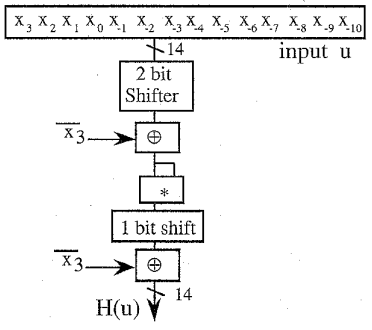

Fig. 2. Twos complement second order approximation in two segments.

4 Bit Serial Implementation and Comparison

In assuming a bit serial implementation, the proposed scheme for sign magnitude can be implemented as follows. First, we note that the bit shifting is constant. This implies that effectively the shifting is a movement of the fractional point and requires no hardware or cycle time (i.e., it can be directly hardwired). Second, in assuming that the operand is stored in a register (the sigmoid is performed after multiply-add-accumulate), the operand inversion needs no hardware or cycle time since both polarities are usually available at the registers. Third, the multiplication is effectively performed on 10 fraction bits, thus in assuming one bit multiplication per cycle results in 20 cycles for the multiplication[1].

Fourth, the multiplier output is always shifted by 1 bit; thus, it can be hardwired in the XOR that follows. This in turn implies that only the XOR delay needs be considered. Putting all together and assuming that the design is completely pipelined the overall delay for the sign magnitude implementation requires 21 machine cycles. (An extra cycle is required for the twos complement operation to be inserted for the XOR before the multiplier).

Regarding higher order piecewise approximations Kwan [6] has considered second and third order approximations. In the second order proposal the inputs are divided into two segments [–L, 0), [0, L) where L is a number in power of two. For inputs in the range of (–∞, L) and (L, ∞) the sigmoid function outputs are 0 and 1, respectively. For inputs in the range of [–L, L] the sigmoid function is computed as:

$$\text{Kwan} - 2(u) = \begin{cases} 0.5 + 2^{-1} * u * (\alpha - \beta * u) \text{ for } 0 \le u \le L \\ 0.5 - 2^{-1} * u * (\alpha - \beta * u) \text{ for } -L \le u < 0 \end{cases}$$

Where $\alpha = 2 / L$ and $\beta = 1 / L^2$

Clearly if L is chosen to be 1, forcing the output to be 1 will generate a large approximation error. The maximum value of this error is 0.27 when the input u = 1 since Kwan – 2(1) – sigm(1) = 1 – 0.73 = 0.27. There is no procedure given for the choice of L in [6]. If L is chosen to be 2^3, this implies that $\alpha = 2^{-2}$ and $\beta = 2^{-6}$ for this approximation. The multiplication of β*u can be performed by bit shifting and the total number of multiplications for Kwan's second order approach is 1. The number of addition required is 2 in this approach. Two parameters 0.5 and $\alpha = 2^{-2}$ need to be stored in Read Only Memory (ROM) which requires $2 \times 14 = 28$ bits ≈ 4 bytes in table size for a 14-bit number representation. We computed the generated function output for 10^6 input data which are uniformly spaced in the domain (–8, 8) and we found that the average and maximum errors to be 6.4×10^{-2} and 1.6×10^{-1}, respectively.

The third order approximation in Kwan's approach [6] computes the sigmoid function in the range of [–L, L] by the following equation:

$$\text{Kwan} - 3(u) = 0.5 + 2^{-1}u*(\alpha - \beta * u^2);$$
$$\text{where } \alpha = 3/(2L) \text{ and } \beta = 1/(2L^3).$$

If L is chosen to be 2^3, the parameters α and β are: $\alpha = 0.1875$ $\beta = 2^{-10}$.

It requires two multiplications and two additions to compute the sigmoid output. The table requirement is four bytes to store the parameters $\alpha = 0.1875$ and 0.5. The average and maximum errors are 7.8×10^{-2} and 2.0×10^{-1} for 10^6 generated function outputs with inputs uniformly spaced in the domain (–8, 8). We estimate that the worst case delay for Kwan's second and third order approaches are 33 and 47 cycles, respectively, using bit serial pipelined designs. Clearly the estimations for this approach suggest that the performance, in both precision and speed, of the scheme is rather poor when compared with the linear approximations described in [3], [5], [7].

In comparing the proposed scheme with Kwan's proposals (denoted here as Kwan-2 and Kwan-3 for second and third order approximations, respectively, described in Section 1 for $L=2^3$ using our internal number representation), we have generated Table 1. Analyzing Table 1, it can be concluded that:

- The hardware employed by the scheme is comparable to (if not less than) the Kwan-2 and Kwan-3 schemes. This is because our scheme requires a multiplier and a XOR while the Kwan-2 and Kwan-3 schemes require at least a multiplier, an adder and a small table.
- Our scheme outperforms Kwan-2 and Kwan-3 schemes with speedups of 1.57 and 2.23, respectively (for the sign magnitude approach), and 1.5 and 2.14 (for the twos complement approach).
- The maximum error for our scheme is 7.27 and 9.09 times smaller than the maximum error for Kwan-2 and Kwan-3, respectively.
- The average error for our scheme is approximately one order of magnitude smaller than the average error for Kwan-2 and Kwan-3 (8.31 and 10.13 times smaller, respectively).

1. We assume one bit per cycle multiplication for the comparison. Clearly other inexpensive multiplications are possible producing more than one bit per cycle. Given that this will improve the delay of the proposed scheme as well as the other schemes in [6], it is not discussed here as it will not substantially change the overall conclusions.

TABLE 1
SIGMOID EVALUATION PARAMETERS

Schemes	average error	max. error	table size	delay	multiplications	additions
Kwan-2	6.4×10^{-2}	1.6×10^{-1}	4	33	1	2
Kwan-3	7.8×10^{-2}	2.0×10^{-1}	4	47	2	2
our scheme	7.7×10^{-3}	2.2×10^{-2}	0	21* / 22†	1	0

*delay for sign magnitude
†delay for twos complement

A final consideration for the proposed scheme regards the question of comparing between the proposed first order approximation schemes and the scheme presented here. The proposed two piecewise second order approximation is clearly faster producing comparable precision to the first order schemes proposed in [3], [4], [7]. As a final question we consider the possibility of improving the accuracy at the expense of delay and hardware requirements. In particular it is of interest to consider comparable delay and hardware to the first order approximation schemes and investigate the overall accuracy. When compared with the first order approximation, we have shown in [7] that a 16-segment implementation of the scheme proposed here having the same delay and slightly more memory can significantly improve the accuracy. In particular we have shown in [7] that the error associated with the 16-segment design of the second order approximation has an average error of 5.7×10^{-4} and a maximum error of 3.6×10^{-3} which are substantially better than the first order approximation errors (which are of the order of 10^{-3} and 10^{-2} for a 16-segment design).

5 Concluding Remarks

We have presented a scheme for the generation of the sigmoid function using a second order approximations approach for both sign magnitude and twos complement notation. This scheme is one of a set of three schemes we have developed in [7]. The proposed scheme requires only one multiplication providing a performance of 21 machine cycles with an average error of 7.7×10^{-3} which is approximately one order of magnitude smaller that the other approaches for the same order of approximation. Finally all proposed schemes [7] including the one presented here have an average error on the order of 10^{-3} and inexpensive implementations, making them suitable for hardwired neural emulators.

Acknowledgments

We are especially grateful to the anonymous referees who reviewed this paper and provided us with helpful comments that improved the paper's quality.

References

[1] D.E. Rumelhart, J.L. McClelland, and the PDP Research Group, *Parallel Distributed Processing, vol. 1, Foundations*. Cambridge, Mass.: MIT Press, 1986.
[2] J.L. Holt and J.N. Hwang, "Finite Precision Error Analysis of Neural Network Hardware Implementations," *IEEE Trans. Computers*, vol. 42, no. 3, pp. 281-290, Mar. 1993.
[3] P. Murtagh and A.C. Tsoi, "Implementation Issues of Sigmoid Function and its Derivative for VLSI Digital Neural Networks," *IEE Proc.-E*, vol. 139, no.3, May 1992.
[4] D.J. Myers and G. Storti-Gajani, "Efficient Implementation of Piecewise Linear Activation Function for Digital VLSI Neural Networks," *Electronics Letter*, vol. 25, no.24, pp. 1,662-1,663, Nov., 1989.
[5] C. Alippi and G. Storti-Gajani, "Simple Approximation of Sigmoid Functions: Realistic Design of Digital VLSI Neural Networks," *Proc. IEEE Int'l Symp. Circuits and Systems*, pp. 1,505-1,508, 1991.
[6] H.K. Kwan, "Digital Implementation Methods for Nonlinear Functions in Neural Networks," unpublished manuscript, 1993.
[7] M. Zhang, S. Vassiliadis, and J.G. Delgado-Frias, "Sigmoid Generators for Neural Computing Using Piecewise Approximations," TR 1-68340-44(1994)11, Dept. of Electrical Eng., Delft Univ. of Technology, Nov. 1994.