HIGH-SPEED DOUBLE PRECISION COMPUTATION OF NONLINEAR FUNCTIONS

V. K. Jain **L. Lin**

Department of Electrical Engineering, University of South Florida
Tampa, Florida 33620, U.S.A
Phone: +1(813)974-4741, FAX: +1(813)974-5250

ABSTRACT

High-speed coprocessors for computing nonlinear functions are important for advanced scientific computing as well as real-time image processing. In this paper we develop an efficient interpolative approach to such coprocessors. Performed on suitable subintervals of the range of interest, our interpolation which uses third degree polynomial is adequate for many elementary functions of interest with double precision mantissas. Our method requires only one major multiplication, two minor multiplications and a few additions. The minor multiplications are for the second and third degree terms, and their significant bits are much fewer than those of the first degree term. This leads to a very fast and efficient VLSI architecture for such coprocessors. It appears that polynomial based interpolation can yield considerable benefits over previously used approaches, when execution time and silicon area are considered. Further, it is possible to combine the computation of multiple functions on a single chip, with most of the resources on the chip shared for several functions.

I. INTRODUCTION

With the recent advances in VLSI technology, special purpose VLSI chip designs have become attractive and cost-effective. *In this paper we propose a reduced cycle hardware to yield fast and accurate results for nonlinear functions with double precision mantissas.* The importance of high-speed nonlinear function generation for scientific computing as well as for signal and image processing is being recognized by the Computer Science community, as evidenced by the several papers presented on the topic at recent conferences [1]-[2] as well as in journals [3]-[4]. Our proposed algorithm is somewhat function independent so that the familiar elementary nonlinear functions--reciprocal, square-root, trigonometric/inverse trigonometric and logarithmic as well as application-specific functions (Bessel, sigmoid etc.) can be readily implemented. Such high speed coprocessors for evaluating nonlinear functions are needed for high performance scientific computations as well as for real time applications -- image processing, neural networks, and on-line solution of partial differential equations [5]-[6]. Currently used hardware realizations are based on simple iterative equations such as Newton Raphson, Cordic, etc. As pointed out by Koren and Zinaty [7], their major drawback is linear convergence. This slows down the calculation significantly, especially for high-precision operands. On the other hand, fast algorithms [8] that generate more bits per iteration, require huge ROMS and multipliers. Some approaches even require a divider; Koren's rational approximation technique is one such example.

The basis of our method arises from three factors:

(1) The range of interest is divided into a suitable number of intervals, say 2^m subintervals ($2^m + 2^{m-1}$)in case of square-root). Then the m Most Significant Bits (m+1 in

This research was supported in part by NSF under Grant No. MIP-9103286

case of square-root) are used as an index to a look-up table, wherein the coefficients of the interpolation polynomial are stored.

(2) The computation is performed on the basis of operand significance, wherein the second and third degree terms of the interpolation polynomial are computed using only the significant bits -- thereby minimizing large multiplications.

(3) The interpolation is carried out using '**Matched Interpolation Polynomials (MIP)**' discussed in the paper.

In order to generate high accuracy results in a few clock cycles, low degree polynomials are used. Admittedly, several other possibilities exist, that include Least Mean-Squares fitting and Chebyshev approximation [9]. Our study, however, shows the gains from such enhancements are suspect when actual arithmetic (involving finite word length) is performed. In this paper, therefore, we concentrate on the MIP approach. It uses a new family of polynomials discussed in the paper. Indeed, we show that the third degree MIP polynomials is adequate for a large class of elementary and trigonometric/inverse trigonometric functions with double precision mantissa. Significance based computation, as explained in the paper, can yield considerable benefits over previously used approaches, when execution time and silicon area are considered. As shown in [7], Koren and Zinaty's method requires 7 additions, 10 multiplications and, as mentioned earlier, a division. Unlike our method, their internal operands require much larger number of bits, thereby making a single addition or multiplication slower than ours. ***We estimate that the throughput using our architecture will be greater by a factor of 10 compared to Koren and Zinaty's.*** Wong's fast division algorithm [8] uses an accurate estimate of the quotient. He has suggested Taylor series approximation to compute quotients, which requires seven cycles to obtain the same accuracy that can be obtained in four cycles by our method. Further, his algorithm is specific to division only.

The organization of the paper is as follows. Section II presents the MIP interpolation formulas. Section III describes significance based computation and the corresponding hardware models. In section IV we describe the double precision (53 bit mantissa) case in detail. A study of the precision for the 53 bit mantissa is given in Section V. Comparison of MIP algorithm with two other algorithms is presented in Section VI. Concluding remarks are given in Section VII.

II. A NEW FAMILY OF INTERPOLATION POLYNOMIALS

The arithmetic in this paper is based on second and third order polynomials belonging to a new family (MIP) defined in this section. To begin the discussion, consider that the operand is N bits wide, with the minimum and maximum value spanning a semi-closed real interval $I_{domain} = [a,b)$. In the floating point case, N represents the word-length of the mantissa. The input word 'A' is segmented into two fields: the upper field consisting of M bits, and a lower field of L bits, such that M+L=N. The upper field invokes a 2^m point grid on the interval I_{domain}, where $m \leq M$; call the points on this grid as X_i, $i=0,...,2^m$. It is then reasonable to attempt to use polynomial interpolation (of appropriate degree) with the objective of estimating the value of a desired function f(A). Indeed, suppose that a given N-bit operand 'A' lies in the i-th interpolation interval $I_i = [X_i, X_{i+1})$. Then, our objective is to use a polynomial of the form

$$P_r(A) = d_{i,1} + d_{i,2}y + \ldots + d_{i,r}y^r \qquad (1)$$

which approximates f(A) over $I_i = [X_i, X_{i+1})$ according to some specified criterion. Here, y denotes the fractional location of A in the i-th interpolation interval I_i, i.e.,

$$y \triangleq \frac{(A-X_i)}{(X_{i+1}-X_i)} = \frac{(A-X_i)}{\Delta} \qquad (2)$$

Note that $\Delta = X_{i+1} - X_i$, the uniform grid interval. After investigating the least-mean-squares (LMS) and the Chebyshev criteria, versus the simpler 'Matched Interpolation Polynomials', we found that the latter is quite accurate. Hence we concentrate here on MIP due to its simplicity and intuitive appeal.

MIP Criterion

An interpolation polynomial $P_{i,r}(A)$ of degree r satisfies the MIP criterion if

(a) $P_{i,r}(X_i) = f(X_i)$,
(b) $P_{i,r}(X_{i+1}) = f(X_{i+1})$, and
(c) $P^{(j)}_{i,r}(X_i) = f^{(j)}(X_i)$, for j=1,..,r-1.

where the superscript j denotes the j-th derivate.

Note that the polynomial is parameterized by the index i of the interpolation interval. Further, note that the MIP criterion imposes *matching of the functional values at both end points of the interpolation interval, and matching of the derivatives up to degree r-1 at the left end point.*

MIP Polynomials

The interpolation polynomial of degree r which satisfies the MIP criterion is unique and is given by

$$P_r(A) = f_i + y\,(f_{i+1} - f_i) - y_c \sum_{j=2}^{r} \Delta^j y^{j-1} c_j \tag{3}$$

where f_i denotes the value of the function f at $A=X_i$ (similarly for f_{i+1}), and Δ denotes the length of the interpolation interval; recall from (2) that y denotes the fractional location of A in the i-th interpolation interval I_i, and $y_c = 1-y$ is its complement. Further, the coefficients c_j, j=2,..,r are defined as

$$c_j = \frac{(j-1)!\, c_{j-1} - f^{(j-1)}(X_i)}{(j-1)!\, \Delta} \tag{4}$$

For brevity, the proof is omitted here.

Constructive Property of MIP Polynomials
The r-th degree MIP polynomial can be built using the (r-1)-th degree polynomial by adding a new term as follows:

$$P_r(A) = P_{r-1}(A) - y_c \Delta^r y^{r-1} c_r \tag{5}$$

This property is important for two reasons. First, in simulation studies the effect of going from the (r-1)-th to the r-th degree is clearly given by the additive term on the right hand side of (5). Similarly, a VLSI architecture designed for the (r-1)-th degree interpolation can be gracefully adapted for the r-th degree interpolation. *Secondly, since the additive correction term is of lower significance, its computation can be tailored for appropriate efficiency and accuracy.*

Low Degree MIP Polynomials

$$P_2(A) = f_i + y(f_{i+1}-f_i) + y_c y \Delta \left(f_i^{(1)} - \hat{f}_i^{(1)}\right) \tag{6}$$

$$\begin{aligned} P_3(A) = {} & f_i + y(f_{i+1}-f_i) + y_c y \Delta \left(f_i^{(1)} - \hat{f}_i^{(1)}\right) \\ & + y_c y^2 \Delta^2 \left(\frac{f_i^{(2)}}{2} + \hat{f}_i^{(2)}\right) \end{aligned} \tag{7}$$

are the low degree interpolation polynomials used in this paper, where the approximations to the derivatives (signified by a hat) are given by

$$\hat{f}_i^{(1)} \triangleq \frac{(f_{i+1}-f_i)}{\Delta}, \qquad \hat{f}_i^{(2)} \triangleq \frac{(f_i^{(1)} - \hat{f}_i^{(1)})}{\Delta} \tag{8}$$

III. SIGNIFICANCE BASED COMPUTATION

A model of our nonlinear coprocessor is shown in Fig. 1. The Range Reduction Unit performs field separation (sign, exponent, and mantissa) and reduction of the input mantissa to a predetermined interval I_{domain}. Concurrently, the exponent is manipulated so that the final result computation is facilitated. For example, to perform the reciprocal operation, the normalized mantissa range is selected as [0.5, 1). The second, and in fact the major task is the computation of the function f(A) where A denotes the range reduced mantissa; we will denote the result as R. The third, and the last step is to denormalize the result R to yield R_{out}. The second of the above tasks

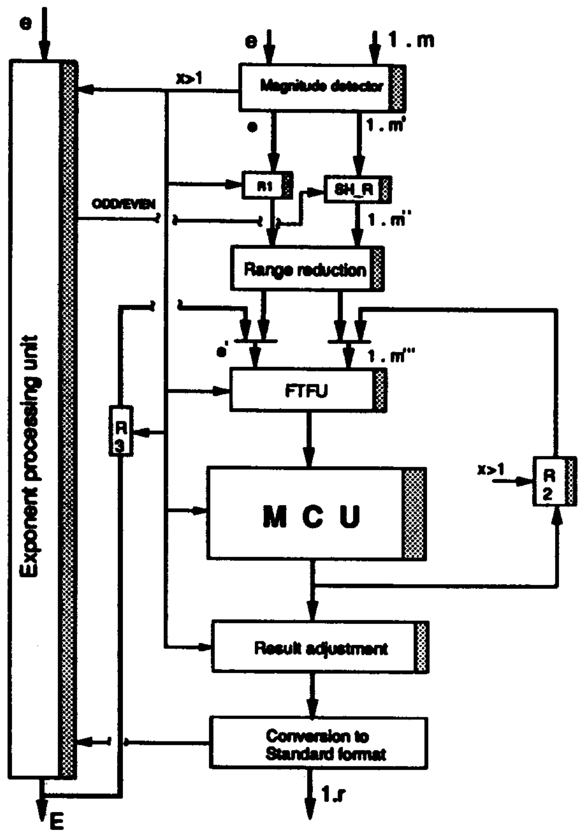

Fig. 1 Reduced cycle NONLINEAR processing unit for IEEE SPF or DPF input

FTFU: DPF to fixed-point conversion unit,
MCU: Mantissa computing unit

(Note: The shaded stripe indicates the block needs function select bits)

is, as might be expected, the most challenging and is indeed the focus of the paper. As the block diagram indicates, all steps within this task are implemented in direct hardware with simple control signal derived from the clock (hence a microcontroller is not needed). Further, the various terms in the interpolation arithmetic are computed in accordance with their significance. Specifically, the higher order terms are computed using smaller word lengths (without loss of accuracy -- practically speaking). The algorithmic explanation is given below.

Third Order Interpolation Algorithm

It is useful to rewrite equation (7) as

$$P_3(A) = f_i + y g_i + y_c y h_i + y_c y^2 q_i \tag{9}$$

where

$$g_i = (f_{i+1} - f_i),\quad h_i = \Delta\left(f_i^{(1)} - \hat{f}_i^{(1)}\right),$$
$$q_i = \Delta^2\left(\frac{f_i^{(2)}}{2} + \hat{f}_i^{(2)}\right) \tag{10}$$

The bit vector $\underline{b}_i = [\underline{f}_i\ \underline{g}_i\ \underline{h}_i\ \underline{q}_i]$ is stored in the RAM as the i-th word; here an underbar denotes the bit vector of the associated quantity. The width of the RAM clearly equals the sum of the field widths chosen for f_i, g_i, h_i, and q_i. The address i is generated from the 'M' MSBs of the input vector $\underline{A}$, i.e., the address bits are

$$\begin{aligned} \underline{X} &= [x_{M-1}\ x_{M-2} \cdots x_0] \\ &= [a_{N-1}\ a_{N-2} \cdots a_{N-M}] \end{aligned}$$

Note that the remaining bits of the input vector $\underline{A}$ can be thought of as a vector

$$\begin{aligned} \underline{Y} &= [y_{L-1}\ \ y_{L-2} \cdots\cdots y_0] \\ &= [a_{N-M-1}\ a_{N-M-2} \cdots\cdots a_0] \end{aligned}$$

Clearly, $\underline{A} = [\ \underline{X}\ \ \underline{Y}\ ]$. A little thought will indicate that the value of the bit vector $\underline{Y}$ with a binary point to the left is the quantity y used in equations (3) through (9). The upper K bits of $\underline{Y}$ are also used for a table lookup. Specifically, the bits of $[y_{L-1} \cdots y_{L-K}] \triangleq \hat{Y} = [\hat{y}_{K-1} \cdots \hat{y}_{K-1}]$ are used as an address to a ROM. Each entry of the ROM has two fields. The first contains $D_1 = \hat{Y}\hat{Y}_c$ where $\hat{Y}_c = 1 - \hat{Y}$; the other contains $D_2 = \hat{Y}^2\hat{Y}_c$.

Application

As an application of the third order interpolation, we consider a mantissa length of 53 bits (IEEE double precision format is $1.\underline{m}$, where $\underline{m}$ is 52 bits wide) and discuss the implementation of one function: the reciprocal. Through the theoretical considerations and simulations we find that the design parameters shown in Table 1 are satisfactory to ensure accuracy up to the ulp. It is useful to remark that the size of the RAM required is approximately 2 Mbits. The use of a RAM permits use of the same memory space for use with any one of several functions. A ROM could of course be used also for the case where specific nonlinear function is required, or a partitioned ROM for the multi-function case.

Table 1 53 Bit Mantissa

Reciprocal		
RAM:	Depth	8,192
	Width	58+44+31+18 = 151
ROM_1:	Depth	4,096
	Width	12
ROM_2:	Depth	256
	Width	57
ROM_3:	Depth	1024
	Width	8
Multiplier size		40x44
Adder size		60

IV. DETAILS OF 53 BIT MANTISSA ALGORITHM

We discuss in detail the *4-clock-cycle* computation of nonlinear functions for double precision mantissa (IEEE floating point standard P754). Although we have successfully investigated the applicability of the the third degree MIP interpolation polynomial of equation (9) to the reciprocal, square-root, sine, and arctangent functions attention is focused here primarily on the first one (REC).

Recall from Section II that M MSBs of the input form the index i to the interpolation interval $I_i = [X_i, X_{i+1})$, while the remaining bits (denoted by y) determine the relative position of the input value within I_i. For computing the linear term all bits of y are used, but in order to minimize the computations ***only the significant bits*** of y (denoted by $\hat{y}$) are used in the computation of second and third degree terms. Now, a trade off exists between the number of bits used for X_i and those used in $\hat{y}$ in the pursuit of desired accuracy. If 14 bits are used for X_i, then our simulations indicate that 24 MSBs of y must be used for $\hat{y}$. Although a practical sized RAM is achieved, the depth of the ROM becomes 2^{23} (since the function y(1-y) is a symmetric function). *Surprisingly, by a recursive application of significance based interpolation, the depth of the ROM can be reduced to* 2^{10}. Specifically, the quantity $\hat{y}\hat{y}_c$ can be computed using a second order interpolation as opposed to a direct table lookup; this results in a 1024 word ROM instead of a 2^{23} word ROM.

Note, that the quantity $\hat{y}\hat{y}_c$ needs to be computed for only half the range of $\hat{y}$ since $\hat{y}_c$ is defined to be *$(1-\hat{y})$*. For the other half range of $\hat{y}$, the same result is obtained by complementing the bits of $\hat{y}$. Furthermore, our simulation results indicates that only 12 MSBs of y are needed for the third degree term in (11) to guarantee the accuracy. Details on the second degree interpolation for 24 bit arguments can be found in [11].

The computation of the first degree term of (11), $y*g_i$, involves a 40×44 bit multiplication, and thus defines the duration of the clock cycle. This clock cycle duration is considerably greater than the time for the computation of $\hat{y}\hat{y}_c$ which uses only 16 bit operands for its multiplier. Therefore, these computations can be scheduled in parallel with the principal computations. The sizes of the hardware blocks used in these secondary computations are quite small compared to those of the main computations. Fig. 2 shows the flow diagram for computations for 53 bit mantissa.

Fig. 3 shows the main building blocks of the 53 bit architecture such as the input/output registers, the RAM, multipliers, the accumulators and various multiplexers that control the data flow (the oval representing second order interpolation contains the ROM). The input word is divided into X_i (14 msb's) and y (38 lsb's) of which the

X_i bits are used to address the RAM which consists of four partitions. Values of f_i, g_i, h_i and q_i are stored in these partitions respectively. In the first cycle, the values of f_i, g_i, h_i and q_i are read from the RAM. While the first degree correction term i.e., $y*g_i$ is being computed by the 40 × 44 bit multiplier, the product $\hat{y}_c\hat{y}^2*q_i$ is generated by a 12 × 18 multiplier and is added to f_i. The sum $f_i+\hat{y}\hat{y}^2*q_i$ is denoted by $S1$. Simultaneously, bits of $\hat{y}$ are used to generate the product $\hat{y}\hat{y}_c$, using the II order interpolation (as shown in the previous section). In the

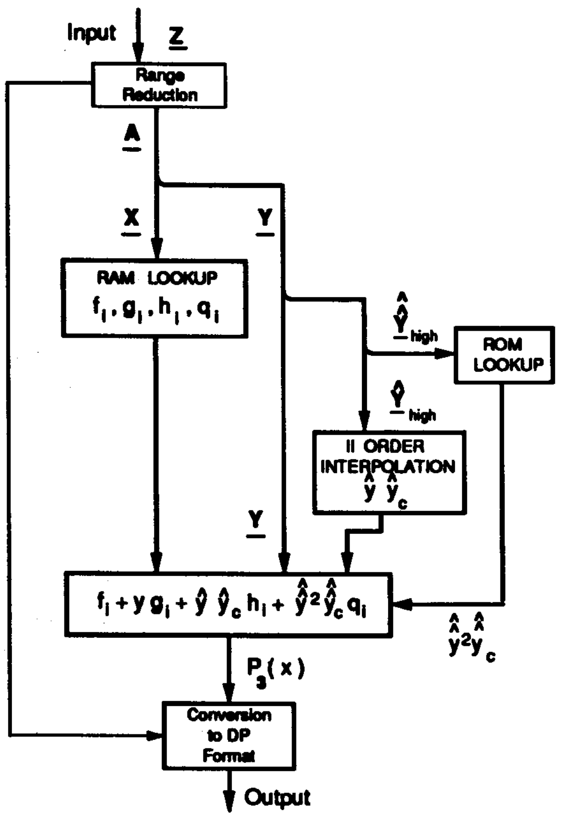

Fig. 2 Flow diagram for computation for 53 bit mantissa

second cycle, the first degree correction term is available and is added to $S1$ to yield a new sum $S2$. Also, h_i is multiplied by $\hat{y}\hat{y}_c$ using a 26x36 bit multiplier. The product $\hat{y}\hat{y}_c*h_i$ i.e., the II degree correction term is added to $S2$ to yield the final result. Details of the hardware timing are given in Table 2.

V. ARITHMETIC ERROR STUDY

To demonstrate the accuracy of our proposed scheme, we present a computer simulation study for the reciprocal function. A program was written in MACSYMA which permits 128 decimal digit precision. The following parameters were used:

Input word length	=	52 bits
Hidden '1' bit	=	1 bit
Width of f_i	=	60 bits
Width of g_i	=	43 bits
Width of h_i	=	31 bits
Width of q_i	=	17bits
Width of ACC_1	=	60 bits
Width of ACC_2	=	60 bits
Size of Multiplier 1	=	40 × 44
Size of Multiplier 2	=	26 × 36
Size of Multiplier 3	=	12 × 18

For 50,000 random vectors (inputs), the fractional ulp rate was found to be 0.095. *No 2*ulp or larger errors occurred.*

Table 2

Unit	Cycle 1	Cycle 2	Cycle 3	Cycle 4
Input	X_1	X_1	X_1	X_1
Mult_1	g_1*y_1	--	--	--
Mult_2	$q_1*\bar{y}_1$	--	--	--
ACC_1	$f_1+q_1*\bar{y}_1$ $+g_1*y_1$	$f_1+q_1*\bar{y}_1$	--	--
Mult_3	--	--	$h_1*\hat{y}\hat{y}_c$	--
ACC_2	--	--	$f_1+q_1*\bar{y}_1$-- $+g_1*y_1+h_1*\hat{y}\hat{y}c$	
Out.	--	--	--	Result1

VI. COMPARISON WITH OTHER ALGORITHMS

In this section, we compare the MIP algorithm with two other algorithms. The first of these is due to Wong and Goto. Described in [13], their method uses the Taylor series to approximate the desired function. An N bit wide operand X is decomposed as $X = X_0 + \Delta X$ where

$$\Delta X = \sum_{i=1}^{N} \Lambda_i X_i \qquad (13)$$

Here, Λ_i denotes the weight of X_i, which is in the range [0, 1). So $f(X) = f(X_0+\Delta X)$ can be approximated using

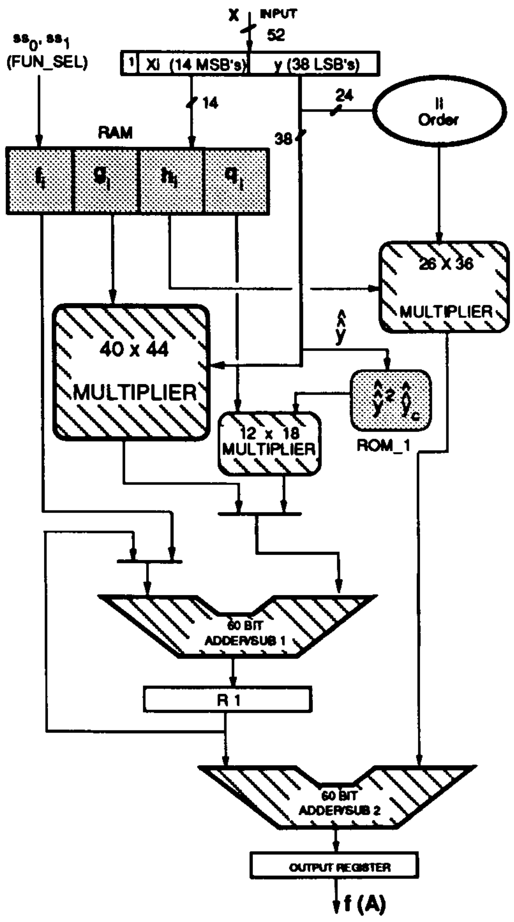

Fig. 3 Architecture for "53 Bit" nonlinear MCU

the Taylor series:

$$f(X_0 + \Delta X) = C_0 + C_1 \Delta X + \sum_{i=2} C_i (\Delta X)^i \qquad (14)$$

where $C_i = f^{(i)}(X_0) / i!$.

The cleverness of Wong and Goto consists primarily in keeping only the significant terms in the evaluation of $(\Delta X)^i$; specifically terms multiplied by Λ^k, where k is suitably chosen, are discarded to simplify computation.

A simulation study of this method was conducted for the reciprocal function under conditions identical to those for our MIP algorithm. Specifically, the number of bits for table lookup are taken to be 14 for both MIP and Wong/Goto algorithms. Thus, the table size was taken to be 16384 words for both approaches, and the interpolation itervals are identical, namely 2^{-14}. Also, both algorithms are restricted to third order polynomials. The width of the operand is 53 bits (whose MSB is set to 1 so the range of the input is [1, 2)). Written in MACSYMA language, the simulation program yields the following results:

Wong/Goto (Reciprocal)	
Probability of 1ulp errors:	0.156
Probability of 2 or more ulp errors:	0.0

With an ulp error probability of 0.156, the Wong/Goto approach clearly yields inferior results.

The second algorithm used for comparison is the third order Chebyshev polynomial approximation. We simulated this method for the reciprocal function under the same conditions as those used for the MIP and Wong/Goto algorithms. The width of operand is again 53 bits. The simulation results are given below.

Chebyshev (Reciprocal)	
Probability of one ulp errors:	0.109
Probability of 2 or more ulp errors:	0.0

Thus the probability of single ulp errors is 0.109, while two or more ulp errors do not occur.

We therefore conclude that the MIP approach (proposed in this paper) results in slightly better performance than even the Chebyshev polynomial approach. The reasons for this are so far not clearly understood. However, we conjecture that while Chebyshev polynomials are optimal in theory, their performance degrades quickly when 'finite word-length coefficients and arithmetic' are used, as one must for a VLSI architecture.

VII. CONCLUSIONS

We have developed an efficient basis for the computation of nonlinear functions in a coprocessor. It was shown that accurate results can be obtained in four clock cycles for mantissa widths of up to 53 bits (IEEE double precision format). It appears that polynomial based interpolation can yield considerable benefits over previously used approaches, when execution time and silicon area are considered. Also, it is possible to combine the computation of multiple functions on a single chip, with most of the resources on the chip shared for several functions. Current effort involves replacing the 40x44 multiplier with a 20x44 multiplier without increasing the latency of the overall hardware.

REFERENCES

[1] *Proc. 11-th Symposium on Computer Arithmetic*, pp. 80-155, IEEE Computer Society Press, June 1993.

[2] M. Taiji, J. Makino, E. Kokubo, T. Ebisuzaki, and D. Sugimoto, "HARP Chip: A 600 Mflops application-specific LSI for astrophysical N-body simulations," *Proc. Hawaii Intntl. Conf. on System Sciences*, pp. 302-311, Jan. 1994.

[3] E. P. O'Grady, and B-K Young, "A hardware oriented algorithm for floating point function generation," *IEEE Trans. on Computers*, Vol. 40, pp. 237-241, Feb. 1991.

[4] M. D. Ercegovac, and T. Lang, "Multiplication/division/square root module for massively parallel computers," *Integration, the VLSI Journal*, pp. 221-234, Dec. 1993.

[5] *Proc. IEEE International Workshop on VLSI Signal Processing*, (Ed: L. D. J. Eggermont, and E. Deprettere), 1993.

[6] *Proc. IEEE International Conference on Wafer Scale Integration*, (Ed: V. K. Jain, and P. Wyatt), 1992.

[7] I. Koren "Evaluating elementary functions in a numerical coprocessor based on rational approximations," *IEEE Trans. on Computers*, pp. 1030-1037, 1990.

[8] D. Wong, and M. Flynn, "Fast division using accurate quotient approximations to reduce the number of iterations," *IEEE Trans. on Computers*, pp. 981-995, Aug. 1992.

[9] K. Hwang et al "Evaluating elementary functions with Chebyshev polynomials on pipeline nets," *Proc. 8th Symp. Computer Arithmetic*, pp. 121-128, May 1987.

[10] "IEEE Standard for binary floating point arithmetic", *ANSI/IEEE Standard No. 754*, American National Standards Institute, Washington DC, 1985.

[11] V. K. Jain, S. A. Wadekar, and L. Lin, "Universal nonlinear component for WSI," *IEEE Trans. on Components Hybrids and Manufacturing Technology*, pp. 656-664, Nov. 1993.

[12] V. K. Jain, and L. Lin, "Nonlinear DSP coprocessor cell -- one cycle chip", *Proc. 1994 IEEE Workshop on VLSI Signal Processing, Oct. 1994.*

[13] W. F. Wong, and E. Goto, "Fast evaluation of the elementary functions in double precision," *Proc. 20th Annual Hawaii Int. Conf. on System Science*, 1994.