(M, p, k)-Friendly Points: A Table-based Method to Evaluate Trigonometric Function

Dong Wang, Jean-Michel Muller, Nicolas Brisebarre, Milos Ercegovac

(M, p, k)-Friendly Points: A Table-based Method to Evaluate Trigonometric Function

Dong Wang, Jean-Michel Muller, Nicolas Brisebarre, and Miloš D. Ercegovac

***Abstract*—Linear (order-one) function evaluation schemes, such as bipartite and multipartite tables, are usually effective for low precision approximations. For high output precision, the lookup table size is often too large for practical use. This study investigates the so-called (M, p, k) scheme that reduces the range of input argument to a very small interval so that trigonometric functions can be approximated with very small lookup tables and a few additions/subtractions. An optimized hardware architecture is presented and implemented in both FPGA device and standard cell based technology. Experimental results show that the proposed scheme achieves more than 50% reduction in total chip area compared with the best linear approach for 24-bit evaluation.**

***Index Terms*—Bipartite table, trigonometric function evaluation, field-programable gate array (FPGA).**

I. Introduction

Trigonometric functions are extensively used in computer graphics, digital signal processing, communication systems, and robotics, to name a few fields of application. Hardware function generators are usually desirable because of their major advantages in speed over software solutions and their potential to save power by avoidance of the use of hundreds of general-purpose instructions.

In 1984, Sunderland et al. [3] considered approximating a 12-bit sine function in hardware with a input argument x less than $\pi/2$ with the use of tables. They proposed to evenly split x (in binary form) into three 4-bit sub-words, i.e., $x = x_0 + x_1 + x_2$, where $x_0 < \pi/2$, $x_1 < 2^{-4}\pi/2$ and $x_2 < 2^{-8}\pi/2$, and use the following equation

$$\sin(x_0 + x_1 + x_2) \approx \sin(x_0 + x_1) + \cos(x_0)\sin(x_2) \quad (1)$$

By doing so, instead of using one large table with 12 address bits, two small tables, that each has 8 address bits are needed: one for $\sin(x_0+x_1)$ and one for $\cos(x_0)\sin(x_2)$. In 1995, Das Sarma and Matula [4] introduced a new scheme to evaluate reciprocal functions. Two tables are addressed in parallel by the partitioned input argument. The outputs are then summed and rounded to the desired accuracy. This scheme is called the *bipartite method*. Later, Schulte and Stine [5] generalized such a scheme to other elementary functions and termed it the symmetric bipartite table method (SBTM), which also generalizes the scheme of Sunderland et al. Compared with straightforward tabulation, the bipartite approach uses only two tables with $2q$ address bits instead of one table with $3q$ address bits (assuming that the input argument is split into three q-bit sub-words).

An enhanced scheme was then proposed by Dinechin and Tisserand [6], who divided the input argument into more than three sub-words, so that multiple small tables can be built. The scheme is thereby named as multipartite table-lookup method (MTM). Both SBTM and MTM are linear (order-one) approximations that possess an inherent limitation: to evaluate functions with high precision, the input argument needs to be reduced to very small values [2]. Very recently, Brisebarre, Ercegovac and Muller [1] proposed a new scheme called (M, p, k)-friendly points to approximate trigonometric functions by using two small bipartite tables and only a few additions. However, no analysis of the error bound or practical hardware implementations were conducted to evaluate the effectiveness of this scheme. Low and Jong [2] revisited the tables-and-additions strategy and successfully proposed an integrated add-table lookup-add (iATA) approach, which could save 20% to 60% of memory compared with MTM. Other approaches, such as LUT cascades [7] and order-two piecewise polynomial approximation and interpolation approaches [8] are also studied in the literature.

In this study, we investigate and develop implementations of the (M, p, k) scheme. Section II introduces the proposed algorithm, rectify prior incorrect definitions and performs an analysis of the error bound. Section III describes an optimized hardware architecture that efficiently implements the proposed algorithm. Section IV provides the FPGA and ASIC based implementation results. Section IV concludes the paper.

II. The (M, p, k) Scheme

A. (M, p, k)-Friendly Points and Angles

Given an n-bit input angle x in the range of $[0, \pi/2)$, we aim to evaluate the trigonometric functions $\sin(x)$ and $\cos(x)$ with p-bit output precision. Assuming that $\hat{x}$ approximates x (we will see later how $\hat{x}$ is obtained), we define $\theta = x - \hat{x}$. When θ is much smaller than $\pi/2$, $\sin(\theta)$ and $\cos(\theta)$ can be efficiently approximated with high accuracy with the use of the aforementioned table-and-addition-based schemes. We can then obtain $\sin(x)$ and $\cos(x)$ in the form of

$$\begin{aligned} \sin(x) &= \sin(\hat{x}) \cdot \cos(\theta) + \cos(\hat{x}) \cdot \sin(\theta) \\ \cos(x) &= \cos(\hat{x}) \cdot \cos(\theta) - \sin(\hat{x}) \cdot \sin(\theta) \end{aligned} \quad (2)$$

Instead of directly implementing (2) in hardware with the use of four multipliers, the (M, p, k) scheme uses a different

Manuscript received ?, 2013; revised ?, 2013.

D. Wang is with the Institute of Information Science, Beijing Jiaotong University, Beijing, China, 100044. e-mail: wangdong@bjtu.edu.cn.

M. D. Ercegovac is with the Computer Science Department, University of California at Los Angeles, Los Angeles, USA, CA 90095.

N. Brisebarre and J.-M. Muller are with the CNRS-Laboratoire LIP, Ecole Normale Supérieure de Lyon, 69364 Lyon Cedex 07, France.

This work was partially supported by NNSF of China Grant No. 61106022.

TABLE I
THE FIRST AND LAST ENTRIES OF TABLE T0 FOR $n = p = 24$, $m = 8$, $k = 7$ AND $r = 7$.

Index	$x_0.x_1 \cdots x_7$	a	b	$\hat{x}_T$	$\lvert\hat{x}_T - x_0.x_1 \cdots x_7 1\rvert$	z	e
0	00000000	256	1	3.90623e-003	1.98680e-008	$1000000000000000\bar{1}000000000000000010$	8
1	00000001	256	3	1.17182e-002	5.36398e-007	$100000000000001\bar{1}00\bar{1}000000001000\bar{1}0010$	8
2	00000010	468	9	1.92284e-002	3.02851e-004	$10010\bar{1}000000001000100000000010010000$	9
...	...	...	...	...	...	...	
200	11001000	2	481	1.56664	2.32097e-004	$1000100001000000\bar{1}00100010\bar{1}0000000\bar{1}$	9
201	11001001	0	1	1.57080	3.42242e-003	10000000000000000000000000000000000	0

way to perform these computations. The scheme seeks special pairs of numbers a, b, and z, which satisfy $\cos(\hat{x}) = a \cdot z$ and $\sin(\hat{x}) = b \cdot z$. If a and b are bounded by a integer M and $z = 1/\sqrt{a^2 + b^2}$ has no more than k nonzero bits when recoded into the canonical form, which contains no adjacent nonzero digits [10], one can implement (2) at the cost of a small number of additions/subtractions when M and k are small. We refer to such numbers (a, b) as (M, p, k)-friendly points and the corresponding $\hat{x}$ as an (M, p, k)-friendly angle. More precisely,

Definition-I: A pair of integers (a, b) is an (M, p, k)-friendly point if the two following conditions are met:

1) $0 \le a < M = 2^m$ and $0 \le b < M = 2^m$;
2) The number

$$z = rnd\left(1/\sqrt{a^2 + b^2}, p + m + 2\right) \tag{3}$$

can be written in its canonical form

$$2^{-e} \cdot 1.z_1 z_2 \cdots z_p z_{p+1} \cdots z_{p+m+1} z_{p+m+2} = \sum z_i 2^{-e-i}$$

where function $rnd(x, n)$ rounds variable x to its n-th fractional bit; m and e are integers, $z_i \in \{-1, 0, 1\}$, and the number of nonzero z_i's is less than or equal to k for $i = 1, 2, \cdots, p + m + 2$. Note that the definition of z is more accurate than that of [1].

Definition-II: The angle $\hat{x}$ in the range of $0 \le \hat{x} < \pi/2$ is an (M, p, k)-friendly angle if either $\hat{x} = 0$ or

$$\hat{x} = \arctan(b/a)$$

where (a, b) is an (M, p, k)-friendly point.

B. Tabulating the Desired Points and Angles

To implement (2), one needs to tabulate the appropriate a, b, z and $\hat{x}$ in a lookup table. The search for the desired points is a trial-and-error process: First, the input $x = x_0.x_1x_2 \cdots x_{n-1}$ is split into two sub-words, i.e., the higher $(r + 1)$-bit part (as the table address):

$$x_T = x_0.x_1x_2x_3 \cdots x_{r-1}x_r$$

and the lower part $2^{-r} \cdot x_{r+1}x_{r+2} \cdots x_{n-1}$, where r is an integer. The angle x_T divides the input range $[0, \pi/2)$ into smaller regions. Second, we select one (M, p, k)-friendly point (for given M and k) whose angle is closest to $x_0.x_1x_2x_3 \cdots x_{r-1}x_r1$ in each region and denote it as $\hat{x}_T$. The largest distance between $\hat{x}_T$ and $x_0.x_1x_2x_3 \cdots x_{r-1}x_r1$ is denoted as D_r . If D_r is smaller than 2^{-r-1}, then all the x within that region will be at a distance

$$D_r + 2^{-r-1} \le 2^{-r} \tag{4}$$

from $\hat{x}_T$. Consequently, the value of $\theta = x - \hat{x}_T$ has an absolute value less than 2^{-r}. If there are no such $\hat{x}_T$ found, M and k will be enlarged and another round of search is performed. As long as r is sufficiently large, we can build small bipartite tables to generate $\sin(\theta)$ and $\cos(\theta)$ with high output precision. Table I provides a numerical example of the selected (M, p, k)-friendly points and angles for $n = p = 24$, $m = 8$, $k = 7$ and $r = 7$. The search process, performed off-line, has a time complexity of $O(M^2)$ for given r.

C. Computation Steps of the Algorithm

Assuming that the correct (M, p, k)-friendly points (a, b) and angles $\hat{x}_T$ are stored in a lookup table T0, the following computation steps are used to implement (2):

1) Subtraction of $\theta = x - \hat{x_T}$ is performed, and the values of $\sin(\theta)$ and $\cos(\theta)$ are looked up in two bipartite tables;
2) The following is computed:

$$\begin{aligned} S &= b \cdot \cos(\theta) + a \cdot \sin(\theta) \\ C &= a \cdot \cos(\theta) - b \cdot \sin(\theta) \end{aligned} \tag{5}$$

Assuming that a and b are stored in radix-4 with a digit set {-2,-1,0,1,2}, each multiplication can be implemented in $\lceil m/2 \rceil$ additions/subtractions;
3) $S \cdot z$ and $C \cdot z$ are performed to implement (2). Because z is in the canonical representation and the number of nonzero bits is bounded by k, the two multiplications can also be simplified to $2(k + 1)$ additions/subtractions.

We note that the efficiency of the proposed scheme relies on how small θ can be for not-too-large values of parameters M and k. Generally, for a given n and p, multiple pairs of parameters can be selected. For instance, we list several possible values of the parameters for $n = 24$ in Table II. The optimal choice of the parameters is determined by the hardware cost and performance. A scheme is proposed in Section III-B to address this design issue.

D. Error Bound

By denoting $\tilde{z}$, $\widetilde{\sin}(\theta)$ and $\widetilde{\cos}(\theta)$ as the exact values of $1/\sqrt{a^2 + b^2}$, $\sin(\theta)$ and $\cos(\theta)$, respectively, three truncation errors can be expressed in $e_z = (z - \tilde{z})$, $e_s = [\sin(\theta) - \widetilde{\sin}(\theta)]$

TABLE II
VALUES OF PARAMETERS k, M AND r FOR $n = 24$, $p = 24$.

M \ r	5	6	7	8	9	10	11
128	7	8	9	na	na	na	na
256	7	7	8	9	na	na	na
512	6	6	7	8	9	na	na
1024	5	6	7	7	8	9	na
2048	4	5	6	7	7	8	9

and $e_c = [\cos(\theta) - \widetilde{\cos}(\theta)]$. Then, the total approximation error entailed by the proposed algorithm is bounded by

$$\begin{aligned} |\widetilde{\cos}(x) - \cos(x)| &= \Big|(\tilde{z} - z)[a\widetilde{\cos}(\theta) - b\widetilde{\sin}(\theta)] \\ &\quad + [az(\widetilde{\cos}(\theta) - \cos(\theta)) - bz(\widetilde{\sin}(\theta) - \sin(\theta))]\Big| \\ &= \big|\sqrt{a^2 + b^2} \cdot \widetilde{\cos}(\theta + \hat{x})e_z + az \cdot e_c - az \cdot e_s\big| \\ &< 2^m\sqrt{2} \cdot |e_z| + |e_c| + |e_s| \end{aligned} \tag{6}$$

where $\widetilde{\cos}(x)$ is the exact value of the final result obtained by infinite precision computations. From (3), we already know that $|e_z| < 2^{-(p+m+3)}$. If both outputs of the sine and cosine bipartite tables have a maximum absolute error of $2^{-(p+2)}$, the absolute value of the total error is bounded by 2^{-p}. Therefore, the precision of the results is only determined by the parameters n and p.

III. HARDWARE ARCHITECTURE

A. Proposed Architecture

Fig. 1 shows the top-level hardware architecture that implements the proposed algorithm. Table T0 stores the precomputed (M, p, k)-friendly points (a, b), angles $\hat{x}_T$, and associated z. SBT0 and SBT1 are two small bipartite lookup tables that generate $\sin(\theta)$ and $\cos(\theta)$.

In Table T0, the m-bit integers a and b are recoded in radix-4 with digit set $\{-2, -1, 0, 1, 2\}$. To reduce the area of shifters used, we introduce another constraint on z, and only those $\hat{x}_T$'s whose associated z satisfies this extra condition are tabulated. Fig. 2 shows that z is divided into the lower $\lfloor (p + m + 2)/2 \rfloor$-bit and higher $\lceil (p + m + 2)/2 \rceil$-bit sub-words, and the new constraint requires that the number of nonzero z_i's in the two sub-words is not larger than $\lfloor k/2 \rfloor$ and $\lceil k/2 \rceil$, respectively. Usually, a new round of search for $\hat{x}_T$ should be performed. The value of m might need to be enlarged to find a sufficient number of $\hat{x}_T$'s.

After recoding, z has $(k+1)$ fields. The first field (Field-0) has $log_2(e)$ bits and is reserved for the leading "1". The other fields either have $\lceil \log_2 \frac{(p+m+2)}{2} \rceil + 1$ bits or $\lfloor \log_2 \frac{(p+m+2)}{2} \rfloor + 1$ bits, in which one sign bit is added at the most significant bit. Those fields that have no corresponding nonzero z_i are filled with all "1", which represents multiplication by zero.

The multiple generation module (MGEN) generates $\lceil m/2 \rceil$ "multiples" by implementing a simple logic as described in [11]. The multi-operand adder (MOPADD) has a compressor tree structure based on [3:2] carry-save adders when standard cell based implementations are targeted. For FPGA implementation, an [N:3] redundant adder structure is proposed in this study to efficiently utilize the carry-propagate adders (CPAs) and ternary adder structures on FPGAs with dedicated carry-chains that can provide carry propagation by more than one order of magnitude faster compared with the use of general logic resources [14]. Fig. 3 shows an example of a [9:3] compressor for the proposed redundant adder structure. The compressor consists of two diagonally deployed linear arrays (only one is shown in Fig. 3) of CPAs. The structure can be efficiently mapped into two arrays of CPAs, in which the fast carry-chain (blue lines) is efficiently utilized. The proposed redundant adder significantly reduces the critical path delay of the proposed architecture.

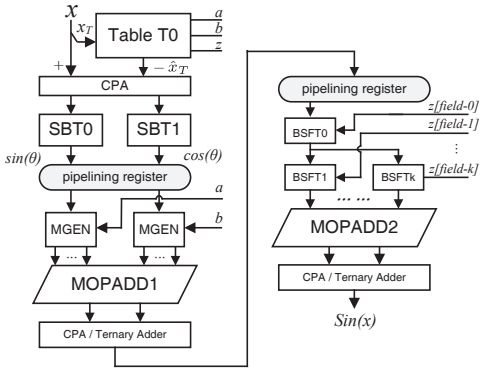

Fig. 1. The proposed hardware architecture for table-based trigonometric function evaluation circuit utilizing (M, p, k)-friendly angles and points.

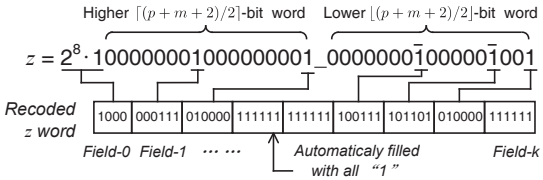

Fig. 2. The proposed recoding scheme for variable z.

The right-shift barrel shifter (BSFT) has an internal structure that merges two consecutive levels of 2-to-1 shift operations into a single stage [15]. Efficiently mapping each stage into 4-to-1 multiplexors can reduce critical path delay without increasing the shifter area. Sign extension is performed after shifting.

B. Selecting the Optimal Parameters

From Section III-A, we know that the total size of table T0 and two bipartite tables can be estimated by

$$S_{T0} = 2^{r+1} \cdot \Big[\lceil m/2 \rceil \cdot 6 + p + \lceil \log_2(p + m + 2) \rceil \cdot k\Big]$$

$$S_{SBT} = 2^{\lfloor 2(n-1-r)/3 \rfloor + 1} \cdot [(p + 4) + p/4]$$

where S_{T0} and S_{SBT} denote the memory cost in bits. Applying the precomputed parameter values in Table II, we draw the diagram of Fig. 4 to illustrate the variation trend of the lookup table size with parameter r. An optimal value for r to minimize the memory usage can be observed.

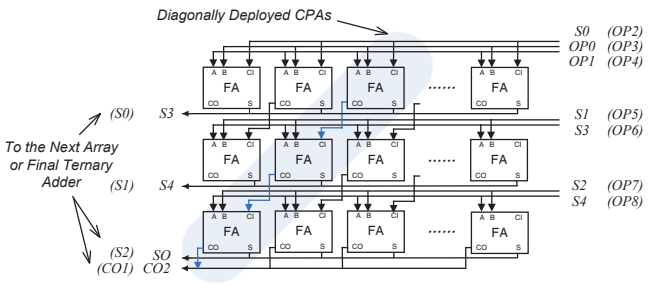

Fig. 3. The proposed parallel linear array structure for a [9:3] compressor. The output signals SO, $CO1$ and $CO2$ are fed to a ternary adder, which can support 3-input addition (A+B+C=D) with the same amount of resource overhead and similar speed as a simple 2-input adder. FA: full-adder.

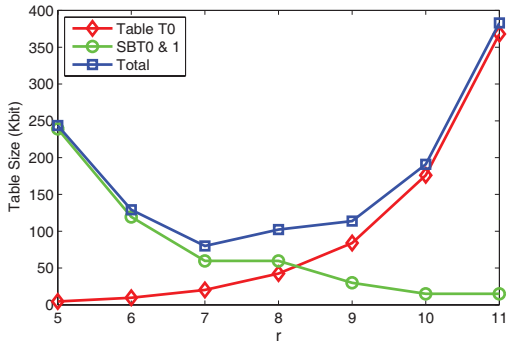

Fig. 4. The estimated lookup table sizes for different values of r when $n = 24$, $p = 24$. For $r = 11$, the total lookup size is 384 Kbits, which is 4.5 times larger than the lookup size for $r = 7$ (85.7 Kbits).

Similarly, one can estimate the area of the multi-operand adders and shifters by

$$S_{ADD} = \left[(2\lceil m/2\rceil - 2) + 2(k-2)\right] \cdot (p+m+2) \cdot NR$$

$$S_{SFT} = \lceil m/2\rceil \cdot (p+2) \cdot 2 + \lceil \log_2(e)\rceil \cdot (p+m+2) + \lceil \log_2(p+m+2)\rceil \cdot (p+m+2) \cdot k$$

where S_{ADD} is calculated in numbers of FA used, whereas S_{SFT} is counted in numbers of 2-to-1 multiplexors. NR denotes the ratio of the area of one 2-to-1 multiplexor to that of one FA. In this study, we assume that $NR = 1.6$ for ASIC and $NR = 1$ for FPGA implementation. After the optimal r is selected, another diagram can be drawn to find the optimal m. Usually, multiple local optimal values can be considered. For the case of $n = 24$, they are $m = 7, 9, 11$. This diagram is not presented because of limited space.

The critical path delay of the second pipeline stage is substantially affected by the value of m, whereas that of the third stage is determined by k. Table II shows that the value of k gradually increases as m decreases. Therefore, in this study, we select the pair of m and k that minimize the value of $m + k$ (i.e., $m = 9$, $k = 7$ for the presented instance).

IV. Implementation Results

The proposed hardware architecture was written in RTL-level System Verilog. Functional simulations were performed with Cadence NC-sim 5.4. The verified designs were implemented in two technologies: 1) Virtex-II XC2V1000-FG456-5 FPGA device synthesized and fitted with the ISE 8.1i tool[1] and 2) TSMC 65nm CLN65G standard cell library synthesized by Synopsys Design Compiler C-2010.03-SP1, placed-and-routed by SoC Encounter 10.1.

Table III first compares the estimated hardware costs of the proposed design with three different schemes, namely, the multipartite table (MTM [6]), piecewise table lookup approximation (Sasao [7]) and order-two interpolation (Lee [8]). The memory resource is counted in bit, while other supporting arithmetic/logic units are measured in the numbers of FA. The proposed design is observed to achieve 79% and 73%

[1]The old device and compilation tool were selected for a fair comparison with the reference designs [2], [9]. Optimization goal and effort were set as "Speed" and "High", respectively.

TABLE III
Estimated Resource Requirements of Different Schemes for the Evaluation of $\sin(x)$ 24-Bit Precisions.

Resource	MTM [6] (table-add.)	Sasao [7] (approxim.)	Lee [8] (interp.)	Proposed
Logic (FA)	260	672	20,796	2,392
Memory (bit)	419,840	327,680	3,773[a]	87,885

[a]The range of x is $[0, \pi/4)$.

reduction in memory resource compared with [6] and [7] for 24-bit evaluations, respectively. The increase in logic resources are $8.2\times$ and $1.5\times$. Because of the degree-2 interpolation algorithm adopted, the scheme of [8] can further compress the lookup tables used. However, due to the large multipliers used, the growth in logic resources is significant. More accurate comparison of the total circuit area are made after the designs are mapped on a specific technology.

Two design instances with precisions $n = 16$ and $n = 24$ were finally implemented. Tables IV and V report the implementation results. To enable direct comparison architectures that do not utilize lookup tables, all ROMs were synthesized into pure combinational logic cells (ASIC) or distributed ROM structures (FPGA) as suggested by [8]. The MTM designs were generated in VHDL by using the program provided by [12] and implemented in the same technologies. A balanced two-stage pipeline structure is implemented such that each stage has a similar delay to the proposed design.

Both tables show that the total area of the proposed design is almost two times larger than that of MTM for 16-bit evaluation. It is revealed that, when using the proposed algorithm to perform low precision evaluations, reducing the lookup table size does not satisfactorily compensate for the hardware cost introduced by the extra arithmetic operations. For 24-bit evaluation, the advantage of the proposed architecture is obvious: it achieves more than 63% reduction in the total circuit area over MTM for both pipelined and non-pipelined FPGA based implementations. It is also observed that the CPAs in the proposed architecture introduces a 42% longer critical latency. However, a simple pipelining scheme enables both scheme working at a similar frequency at the cost of a few registers. For ASIC designs, the area reduction is 51%, and the

TABLE IV
COMPARISON OF FPGA BASED IMPLEMENTATION RESULTS.

Scheme	Precision (bit)	Lookup Table (Slice)	Total Area (Slice)	Critical Path Delay (ns)	Latency (ns)
Proposed	16	159 (15%)	1,078 (100%)	11.5	34.5
(3-stage pipelined)	24	1,144 (39%)	2,861 (100%)	16.7	50.1
MTM [6]	16	426 (80%)	532 (100%)	10.8	21.7
(2-stage pipelined)	24	7,735 (98%)	7,910 (100%)	18.3	36.7
Proposed	16	159	988	29.1	29.1
(Non-pipelined)	24	1,130	2,642	38.2	38.2
MTM [6]	16	425	511	18.6	18.6
(Non-pipelined)	24	7,697	7,881	26.8	26.8
Xilinx CORDIC IPcore [13]	16	na	595	91.6	91.6
(Non-pipelined)	24	na	1,248	146.5	146.5
Radix-4 CORDIC [9] (Non-pipelined)	16	na	1,184	37.0	37.0
iATA [2] (Non-pipelined)	24	na	4,963	30.9	30.9

TABLE V
COMPARISON OF STANDARD CELL-BASED IMPLEMENTATION RESULTS.

Scheme	Precision (bit)	Lookup Table (Kgate)	Total Area (Kgate)	Critical Path Delay (ns)	Latency (ns)
Proposed	16	993 (12%)	8,093 (100%)	4.6	13.8
(3-stage pipelined)	24	4,223 (29%)	14,256 (100%)	6.9	20.7
MTM [6]	16	2,205 (72%)	3,061 (100%)	2.9	5.8
(2-stage pipelined)	24	28,526 (97%)	29,292 (100%)	8.5	17.1

latency gap also decreases to 17%. Our approach consumes a 42% lesser chip area than the newly reported iATA [2] scheme, but their critical path delays are close.

CORDIC-based designs are compared with the proposed scheme in Table IV. Standard parallel CORDIC structures are implemented with the Xilinx IPcore generator [13]. For both 16- and 24-bit evaluations, the proposed design achieves more than 70% reduction in the critical path delay. The main contribution comes from the optimized multi-operand redundant adder structure presented in this paper. Compared with the radix-4 CORDIC design [9], our proposed architecture is both superior in critical path delay and total circuit area. Because of the limited information, however, we can not directly compare with the schemes presented in [7] and [8].

V. CONCLUSION

This brief has presented a new scheme to compute sine and cosine functions. The main contribution of our work includes rectifying prior incorrect definitions, conducting an analysis on the error bound and developing an optimal hardware architecture that efficiently implements the proposed algorithm. The 16- and 24-bit design instances are coded in Verilog HDL and mapped on Xilinx FPGA and 65nm standard cell based technology. Comparison of our proposed work with multipartite and CORDIC-based designs shows that a considerable reduction in chip area and critical path delay is achieved for 24-bit evaluations.

REFERENCES

[1] N. Brisebarre, M.D. Ercegovac, and J.-M. Muller, "(M, p, k)-friendly points: a table-based method for trigonometric function evaluation," *in Proc. IEEE 23rd Int'l Conf. Application-Specific Systems, Architectures and Processors*, 2012.

[2] J. Y. L. Low and C. C. Jong, "A memory-efficient tables-and-additions method for accurate computation of elementary functions," *IEEE Trans. Comput.*, vol.62, no.5, pp.858-872, May 2013.

[3] D. A. Sunderland, R. A. Strauch, S. W. Wharfield, H. T. Peterson, and C. R. Cole, "CMOS/SOS frequency synthesizer LSI circuit for spread spectrum communications," *IEEE Journal of Solid State Circuits,* vol.SC-19, no.4, pp.497-506, 1984.

[4] D. D. Sarma, D. W. Matula, "Faithful bipartite ROM reciprocal tables," *in Proc. 12th Symp. Comput. Arith.,* pp.17-28, Jul. 1995.

[5] M. Schulte and J. Stine, "Approximating elementary functions with symmetric bipartite tables," *IEEE Trans. Comput.*, vol.48, no.8, pp. 842-847, Aug. 1999.

[6] F. Dinechin and A. Tisserand, "Multipartite Table Method," *IEEE Trans. Comput.*, vol.54, no.3, pp.319-330, March 2005.

[7] T. Sasao, S. Nagayama, J. T. Butler, "Numerical function generators using LUT cascades," *IEEE Trans. Comput.*, vol.56, no.6, pp.826-838, June 2007.

[8] D. Lee, R. C. C. Cheung, W. Luk and J. D. Villasenor, "Hardware implementation trade-offs of polynomial approximations and interpolations," *IEEE Trans. Comput.*, vol.57, no.5, pp.686-701, May 2008.

[9] B. Lakshmi and A. S. Dhar, "VLSI architecture for parallel radix-4 CORDIC," *Microprocessors and Microsystems,* vol. 37, no. 1, pp.79-86, Feb. 2013.

[10] C. K. Koc and S. Johnson, "Multiplication of signed-digit numbers," *Electronics Letters,* vol.30, no.11, pp.840-841, May 1994.

[11] M. D. Ercegovac and T. Lang, *Digital Arithmetic,* Morgan Kaufmann Publishers, Elsevier Science Ltd., 2004.

[12] F. Dinechin, "The multipartite method for function evaluation," www.ens-lyon.fr/LIP/Arenaire/Ware/Multipartite, 2009.

[13] Xilinx Corp., "LogiCORE IP CORDIC v3.0, DS249," www.xilinx.com/support/documentation, 2004.

[14] J. Hormigo, J. Villalba and E. L. Zapata, "Multioperand redundant adders on FPGAs," *IEEE Trans. Comput.*, vol.62, no.10, pp.2013-2025, Oct. 2013.

[15] S. Das and S. P. Khatri, "A timing-driven approach to synthesize fast barrel shifters," *IEEE Trans. Circuits Sys. II, Exp. Briefs,* vol.55, no.1, pp.31-35, Jan. 2008.