CAMs and High Speed High Precision Data for Trigonometric Functions

N. TAHIR[1,2], S. TAGZOUT[1] , A. A. EL OUCHDI[1]

[1] Centre de Développement des Technologies Avancées, Division of Microelectronics & Nanotechnologies,
Cité du 20 Août 1956 BP.17 Baba Hassen 16303 Alger Algérie.
[2] University USTHB, Faculty of Electronic and Computer Science, LCPTS Laboratory, Algiers, Algeria.
ntahiri@cdta.dz, stagzout@cdta.dz, aelouchdi@cdta.dz

Abstract

When high performance is required, the needed hardware implementation of trigonometric functions becomes often problematic. This paper generalizes and improves a CAM based arctangent architecture that has shown an exclusive appropriateness for some critical applications compared to the Look up Table based solution, the polynomial and the rational approximations. For more illustration, detailed design specifications and different sinus function implementation results are given.

***Keywords*— Trigonometric functions, CAM, high precision, VLSI.**

1. Introduction

Computation of the trigonometric values is a most important operation in modern engineering technology. However, its difficult hardware implementation is always emphasized on the resolution of the compromise between speed, accuracy and hardware resource consumption [1]. Almost every paper presenting a trigonometric computing solution starts with a summary of the three usual trigonometric computing approaches that are high order polynomial approximations, rational approximations and Look Up Table (LUT) based methods. In [2] the CORDIC (Coordinate Rotation Digital Computer) algorithm was presented as a cost effective method for performing rotations on vectors in the 2-D plane. This algorithm was extended into rotations in circular, linear and hyperbolic coordinate systems and then, many implementations of the CORDIC have been made, both for fixed-point and floating-point with respect to the input.
As it is shown in [3], one can compute trigonometric functions using an expansion series like:

$$sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots \qquad (1)$$

In [4] a dedicated algorithm for trigonometric computing functions using the Logarithmic Number System (LNS) based on the laws of sinus and cosines is proposed, it uses a novel addressing of buffer with the middle-order bits of the LNS representation.
In [5] a Content Addressable Memory (CAM) based solution for reducing as much as possible the pipeline stages to better control the accuracy output and overcome the related drawbacks in the existing approximation implementations is proposed as well as an address decoding independent from the input tangent precision to overcome the related drawback in the usual LUT based solutions. This proposed method is used if and only if the three following requirements are satisfied [6]:

1- Input values are coded as fixed point two's complement numbers.
2- The performances of the core design should not be driven by the input signal precision. Large word input data like 32 bit words should be accepted. Then, the use of the usual LUTs based methods should be avoided.
3- The core should present the least possible pipeline stages to be used in the implementation of feedback algorithms. Hence, the use of large operators and the use of high order approximations should be avoided.

In fact, these requirements apply for algorithms containing other trigonometric function and then it can be generalized as presented in Fig.1.

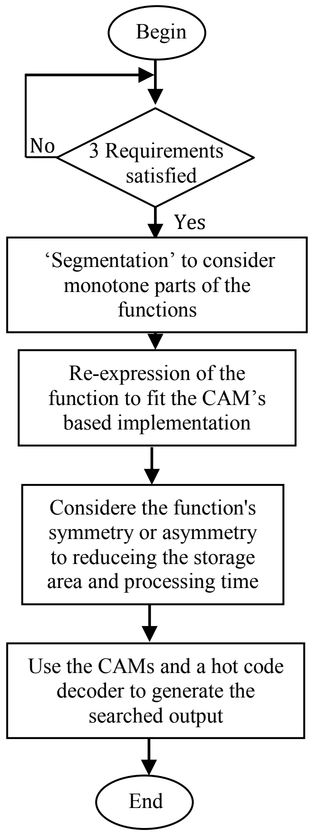
Begin
No
3 Requirements satisfied
Yes
'Segmentation' to consider monotone parts of the functions
Re-expression of the function to fit the CAM's based implementation
Considere the function's symmetry or asymmetry to reduceing the storage area and processing time
Use the CAMs and a hot code decoder to generate the searched output
End

Fig.1. Global flow to use the CAMs based method

The idea is to adapt the existing Arctan CAM solution used in [6] to any trigonometric functions by applying it to segments where the function is monotone (continually increasing or decreasing) and handle the function symmetries and asymmetries with appropriate adapters that indicate the polarity and the segment position to retrieve the searched outputs.
Fig.1. shows a global flow that may be applied for applications satisfying the three requirements listed above.
This method reduces the area, the time consumption and the power compared to the others methods e.g. when the bit width of the sinus value is 16, the angle resolution is π/180 (1°) and the range going from[0, π/2], then the memory length is reduced from 2^16=655368 words to 90 words. In this paper we are showing an adaptation and implementations results when sinus is needed. In section II we show how the segmentation of the function is done, in section III we discuss about the adaptation of the existing Arctan CAM based solution to sinus function and finally in section IV we present the implementation results.

2. Sinus function segmentation

Since the sinus function is periodic and using the principle of monotony, we have divided the period into four parts ([-π, π] in our case) as it is represented in Fig.2.
The Φ values in part I (Φ ϵ [0, π/2]) is the most important part in which sinus values are increasing and ranging between 0 and 1. The values of the Φ angle and its sinus are respectively stored in the buffer.

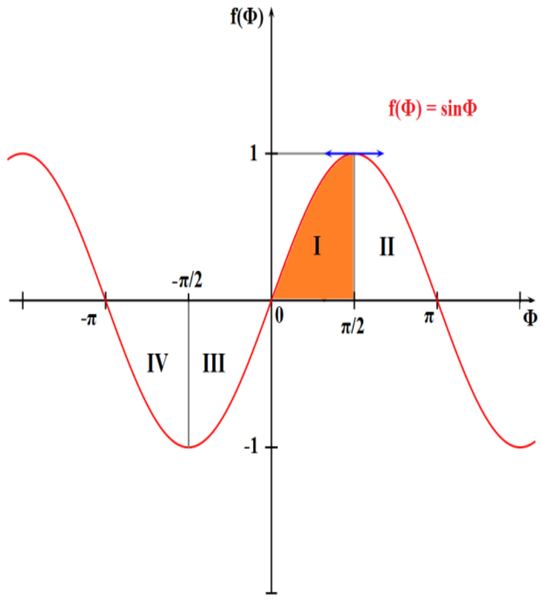

Fig.2. Sinus function segmentation

In the part II (Φ ϵ [π/2, π]), using the trigonometric formula sin (π- Φ) = sin (Φ), the sinus value is obtained from the buffer by first calculating a new value of Φ (Φ'= π- Φ) then reading the value of its function (sin (Φ')) from the ROMs.
For part III (Φ ϵ [-π/2, 0]) and part IV (Φ ϵ [-π, -π/2]), sinus values are the same as in part I and II respectively with negative values which are obtained by performing two's complement of sinus values in part I and part II respectively.
The representation of the ϕ values and their sinus are coded in fixed-point numbers [7]. The fixed-point representation of a number is given by the following equation:

$$x = s + x_{INT} + x_{FR} \qquad (2)$$

Where s is the sign bit, x_{INT} and x_{FR} represent the integer and the fraction parts. A binary number B represented in fixed point can be converted into a decimal number x by the following equation:

$$x = -B_s 2^{INT} + \sum_{1}^{i=INT-1} B_i 2^i + \sum_{i=FR}^{2} B_i 2^{-i} \qquad (3)$$

The representation of both ϕ and sin (ϕ) in 16 bits binary value is represented in the Table 1:

Table 1. Sinus values expression

	decimal value	Bit sign (S)	Integer (m)	Fraction (f)
Φ	6°.5	0	0000110	10000000
sin	0.10452846	0	0	00011010110000

3. Generalizing the CAM based solution for other trigonometric functions

The conceptual block diagram of the used circuit is composed with two essential parts that are the comparator and the decoder as it is shown in Fig.3 :

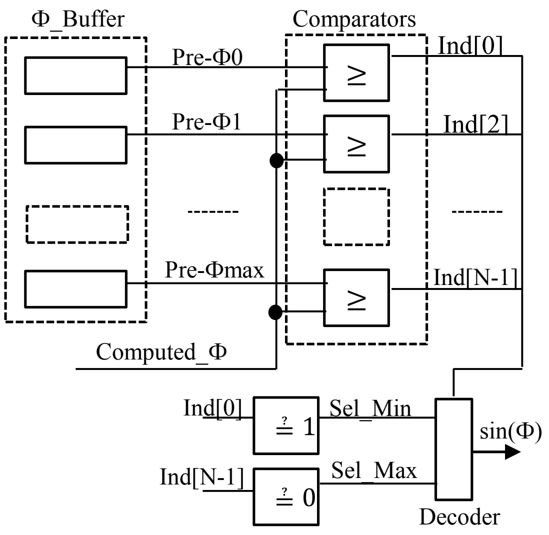

Fig.3. Sinus conceptual block diagram

Where:

- "pre_ (Φ)" is the pre-computed Φ value loaded in the angle Φ buffer.
- "(Φ)" is the computed and input angle Φ value obtained from an arithmetic unit.
- "N" as the angle's number.

- "Ind[i]" as the ith comparator's indicator. Considering that $0 \leq i < N$.
- "sin(Φ_{min})" as the smallest sinus value corresponding to $i = 0$
- "sin(Φ_{max})" as the largest sinus value corresponding to $i = N - 1$.
- "sin(Φ[i])" as the sinus value corresponding to the i_{th} angle Φ_buffer word and to the Ind[i].

During the initialization period, all the Φ and sin (Φ) values in the [0, π/2] interval are stored in the buffer. When the input angle is compared to every stored data in the buffer (pre_(Φ)). By default, each comparator's indicator (Ind[i]) is set high. When the computed (Φ) is larger or equal to the considered pre_(Φ), the comparator's output indicator (Ind[i]) is set low, then the decoder generates the searched sin(Φ) value according the three exclusive cases:

- Sin $(\Phi)_{min}$ is output when Ind [0] is kept high.
- Sin $(\Phi)_{max}$ is output when Ind [N-1] is set low.
- Sin (Φ[i]) is output when Ind[i] is different from Ind [i+1].

These selection cases are illustrated in Fig.4.

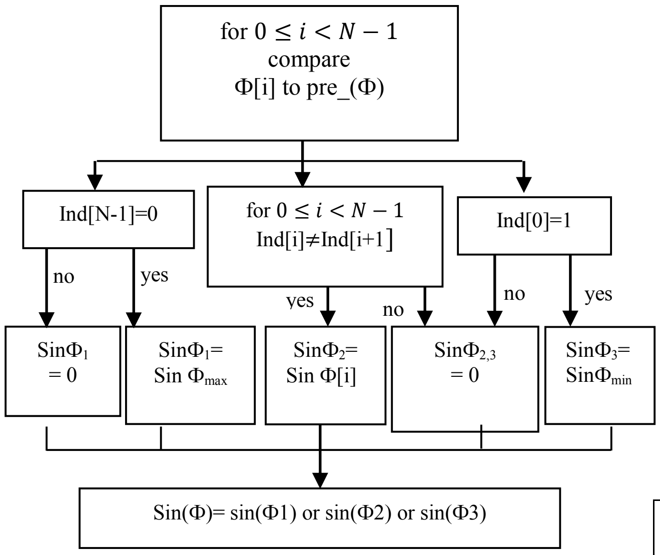
for $0 \leq i < N - 1$ compare Φ[i] to pre_(Φ)
Ind[N-1]=0
for $0 \leq i < N - 1$ Ind[i]≠Ind[i+1]
Ind[0]=1
no
yes
yes
no
no
yes
SinΦ1 = 0
SinΦ1= Sin Φmax
SinΦ2= Sin Φ[i]
SinΦ2,3 = 0
SinΦ3= SinΦmin
Sin(Φ)= sin(Φ1) or sin(Φ2) or sin(Φ3)

Fig.4. Sinus functional diagram

The general bloc diagram for recovering the result for the whole period [−π,π] is explained in section II and is illustrated in Fig.5.

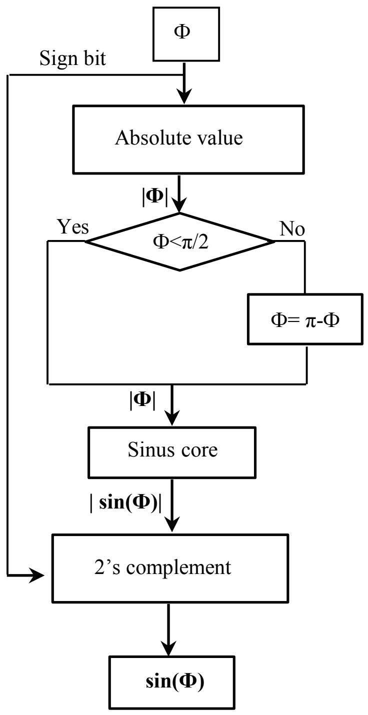

Fig.5. Sinus core extension to cover the full range $[-\pi,\pi]$

4. Implementation results

The most critical part of the CAM is the comparator implementation that depends on the Φ world length. In any kind of technology, to favor the speed, appropriate resources and algorithms need to be chosen in order to minimize the comparator's propagation delay from the Most Significant Bit to the Least Significant one. The Decoder receives the comparator indicators and outputs the searched value Φ.
The design was done by the Cadence NCLAUNCH tool, the logic synthesis was achieved using RTL COMPILER using the CMOS 0.18µ technology. Table 2 shows the synthesis results:

Table 2. The generated reports

Instance	Cells	Leakage Power(nW)	Dynamic Power(nW)
Top module	2703	1443.329	4043474.682
The comparator	1457	617.061	1484685.343
The decoder	1246	826.267	2459704.571

The logic gates area of the module core is given in Table 3.

Table 3. The areas reports

Type	Instances	Area	Area %
sequential	1009	42973.235	65.2
inverter	60	395.136	0.6
logic	1634	22535.923	34.2
total	2703	65904.294	100.0

4.1. Physical implementation

After the logical synthesis, the physical implementation was made by Encounter Digital Implementation (EDI) [8] from Cadence tool. It carries out the placement and the routing by placing the resulting Netlist of the logical synthesis into a layout while respecting the steps of the physical implementation. The obtained Layout is shown in Fig.6.

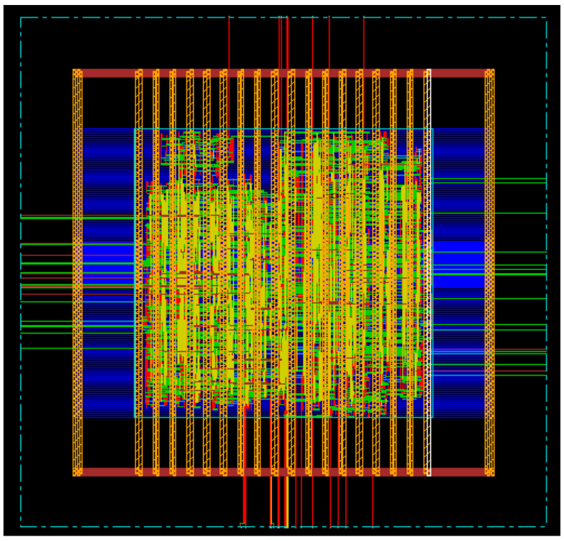

Fig.6. Layout of the Sinus core

5. Conclusion

In this work we have presented a generic approach for implementing trigonometric functions using high precision input data and a CAM based architecture. This method reduces required memory sizes compared to look up table based solutions and it gives desired output precisions while preventing from using large operators and large pipeline paths. The sinus core was successfully verified, synthetized and implemented using the CMOS 0.18µ technology and a Cadence flow.

6. References

[1] Z. ZHIHENG and W. DONGCHENG, "An Algorithm for Computing the Value of Arbitrary Angle Trigonometric Function", International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE), December 16-18, Changchun, China, 2011.

[2] C. DONG and C. HE, "Implementation of Single-Precision Floating-Point Trigonometric Functions with Small Area", International Conference on Control Engineering and Communication Technology (ICCECT), 2012.

[3] S. BALAC and F. STURM , "Algébre et analyse" cours de mathématiques page 781.2003

[4] M. G. ARNOLD, "Approximating Trigonometric Functions with the Laws of Sines and Cosines using the Logarithmic Number System", Proceedings of the 8th Euromicro conference on Digital System Design (DSD'05), Lehigh University Bethlehem, PA 18015 USA, 2005.

[5] K. PAGIAMTZIS and A. SHEIKHOLESLAMI, "Content-Addressable Memory (CAM) Circuits and Architectures: A Tutorial and Survey", IEEE JOURNAL OF SOLID-STATE CIRCUITS, VOL. 41, NO. 3, MARCH 2006.

[6] S. TAGZOUT and A. BELOUCHRANI, "Arctangent architecture for high speed and high precision data", Journal of Circuits, Systems, and Computers Vol. 20, No. 7, 1243-1259, 2011.

[7] L. Y. MING, "An Optimization Framework for Fixed-point Digital Signal Processing", Thesis for the Degree of Master of Philosophy in Computer Science and Engineering The Chinese University of Hong Kong August, 2003.

[8] "Encounter Digital Implementation System User Guide" Product Version 10.1.2 July 2011