AMSE JOURNALS-AMSE IIETA publication-2017-Series: Advances A; Vol. 54; N°2; pp 197-202
Submitted Apr. 04, 2017; Revised Sept. 25, 2017, Accepted Sept. 30, 2017
https://doi.org/10.18280/ama_a.540205

FPGA Implementation of Arcsine Function Using CORDIC Algorithm

Anurup Saha, Archisman Ghosh, K. Gaurav Kumar

ADES Lab, Dept. of E.T.C.E, Jadavpur University, Kolkata 700032, India
(sahaanurup24@gmail.com, archismanghosh12@gmail.com, kgauravkumar35@gmail.com)

Abstract

This paper presents finite state machine based hardware to calculate arcsine function using CORDIC algorithm. CORDIC algorithms provide an effective methodology to compute a large range of transcendental functions, since it only requires addition, shift and subtraction operations. The design has been done using verilog HDL and it has been tested in SPARTAN-3 FPGA. Since different applications may require different accuracy, the focus has been on how an FPGA implementation of arcsine function can be easily reconfigured when higher precision is required.

Key words

CORDIC, FPGA, Computer arithmetic, Arcsine, ASM.

1. Introduction

CORDIC, abbreviated from Coordinate Rotation Digital Computer, was introduced by Jack. E. Volder [1]. CORDIC algorithms can be used to evaluate elementary functions like trigonometric functions, hyperbolic functions etc. [2] [3]

CORDIC algorithms are iterative. So number of iterations is chosen according to required precision.

There are 2 other approaches to compute such functions- (a) Look-Up Table (LUT) based implementation and (b) Polynomial Approximation. [4] [5] LUT-based designs can be error-

prone, unless the size of LUT is sufficiently large. But, storing large LUT is a cost-inefficient solution to the problem.

Polynomial approximation, derived from Taylor series, has the disadvantage that it requires multipliers. On the other hand, CORDIC algorithms can be implemented using adders, subtractors and shifters only.

Due to its simplicity, CORDIC algorithms are used in a wide range of applications like the HP-35 calculator, 8087 mathcoprocessor [6], radar signal processors [7] and robotics.

2. Cordic Algorithm

In 2-D plane, when a vector P(x,y) is rotated by an angle α as shown in figure-1, new coordinates of the vector P'(x',y') can be obtained by ,

$$x' = x\cos\alpha - y\sin\alpha = \cos\alpha\,(x - y\tan\alpha) \quad (1)$$

$$y' = y\cos\alpha + x\sin\alpha = \cos\alpha\,(y + x\tan\alpha) \quad (2)$$

If $\tan\alpha = 2^{-i}$, the expression within bracket can be calculated using shift, subtraction and add operation. By choosing a proper sequence of rotations $\alpha_0, \alpha_1, \ldots \alpha_{n-1}$ necessary trigonometric functions can be evaluated.

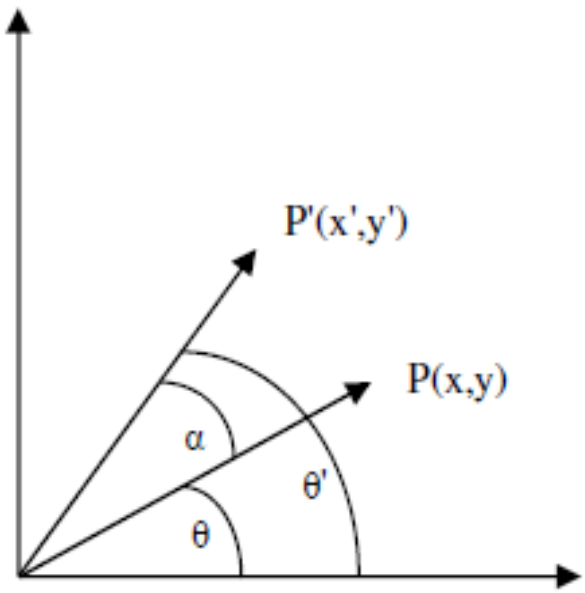

Fig.1. Rotating Vector on 2-D Plane

3. Steps to Compute Arcsine

By starting from the X-axis with a vector of pre-computed length, arcsine function is determined. To compute the arcsine of a given number z, i.e $\sin^{-1}(z)$, the following equations are used:

$$x_{i+1} = x_i - m * {}^{y_i}/_{2^i} \tag{3}$$

$$y_{i+1} = y_i + m * {}^{x_i}/_{2^i} \tag{4}$$

$$\theta_{i+1} = \theta_i + m * \tan^{-1}({}^{1}/_{2^i}) \tag{5}$$

where, m= +1 if, y_i<z-1 otherwise.

Based on these equations, an Algorithmic State Machine with Datapath(ASMD) chart [8] can be developed as shown below:

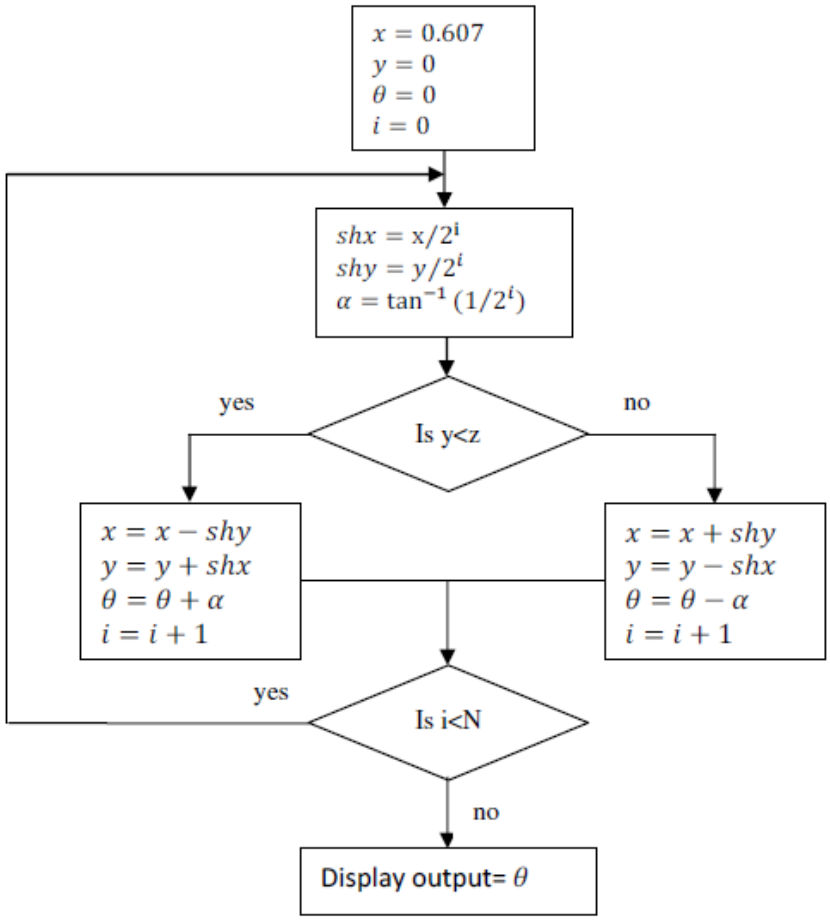

Fig.2. ASMD Chart for Evaluation of Arcsine Function

Here N represents the number of iterations. As N increases, accuracy of the system improves. The hardware architecture is as follows:

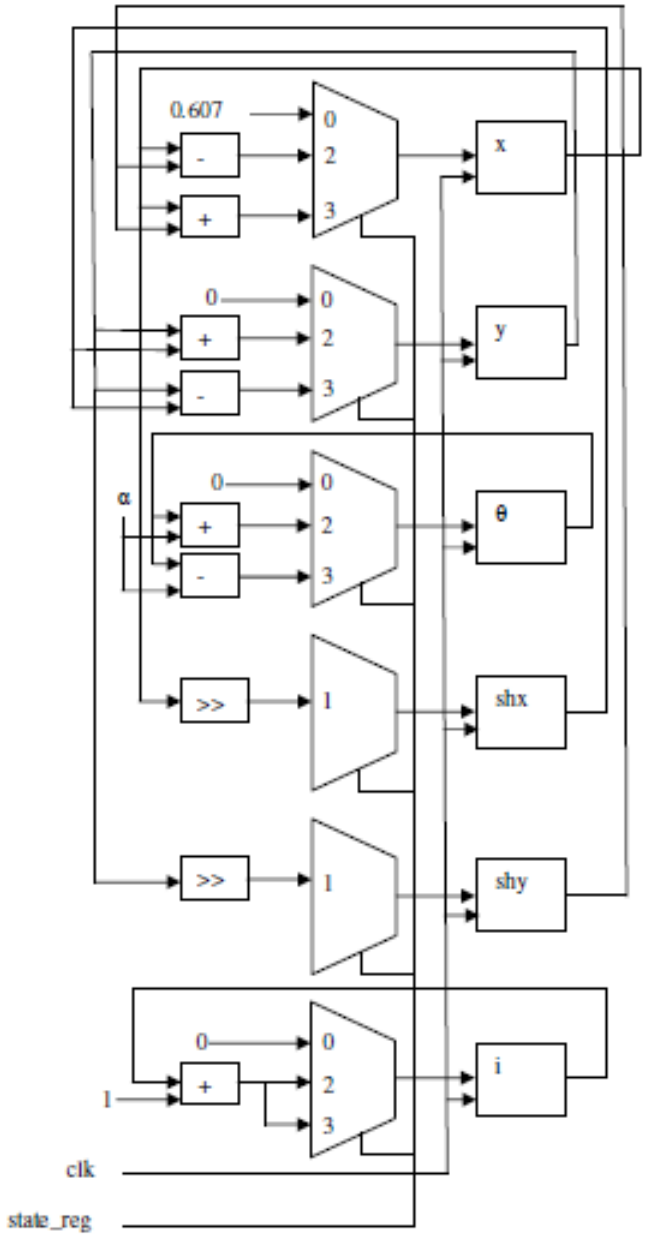

Fig.3. Proposed Architecture for Evaluation of Arcsine Function

Here state_reg represents the present state of the finite state machine. A global clock has been used for synchronus operations.

4. Results and System Performance

Verilog hardware description language has been used for implementation. The design was tested in SPARTAN-3 FPGA (XC3S50). Hardware resources required for the design can be summarized as:

Tab.1. FPGA Resource Utilization

Test Condition	Used	Utilization
Number of slice flip-flops	90 of 1536	5%
Number of occupied slices	101 of 768	13%
Number of LUTs	173 of 1536	11%
Number IOBs	25 of 124	20%

Computation time and accuracy depends on N. If larger N is chosen, it results in higher precision and more computation time. For example, $\sin^{-1}(0.80) = 53.13°$. Percentage of error, and number of clock cycles required for computation have been shown as a function of N in Table.2.

Tab.2. Number of Clock Cycles, Error Vs. N

N	Number of clock cycles	Error (%)
8	26	0.51%
9	29	0.13%
10	32	0.06%

It is to be noted that the amount of computation in each iteration is constant. As a result number of clock cycles required is a linear function of N.

Conclusions

In this paper, a novel FSM-based hardware to evaluate arcsine function has been proposed using CORDIC algorithm. Albeit CORDIC algorithms are relatively slower, simplicity of the algorithm ensures minimum requirement of hardware resources. As a result, these algorithms are being used in various applications. Also, depending on the necessity of the application, the architecture can be easily modified to operate under higher accuracy.

The concept can be used to determine arccosine and arctangent functions by choosing appropriate direction of rotations.

Acknowledgment

The authors would like to thank Prof. M. K. Naskar, Asim Maiti for their constant help, guidance, and support.

References

1. J.E. Volder, The CORDIC trigonometric computing technique, 1959, IRE Transactions on Electronic Computers, vol. EC-8, pp. 330-334.
2. J.S. Walther, A unified algorithm for elementary functions, 1971, Spring Joint Computer Conf., pp. 379-385.
3. B. Parhami, Computer arithmetic, Oxford University Press, pp. 361-371.
4. W.E. Ferguson, T. Brightman, Accurate and monotone approximations of some transcendental functions, 1991, Proceedings 10th Symposium on Computer Arithmetic, pp.237-244.
5. P.T.P. Tang, Table-lookup algorithms for elementary functions and their error analysis, 1991, Proceedings 10th IEEE Symposium on Computer Arithmetic, pp. 232-236.
6. J. Duprat, J.M. Muller, The CORDIC algorithm:new results for fast VLSI implementation, 1993, IEEE Transactions on Computers, vol. 42, pp. 168-178.
7. R.J. Andraka, Building a high performance bitserial processor in an FPGA, 1996, Proceedings of Design SuperCon '96, pp. 5.1-5.21.
8. P.P. Chu, FPGA prototyping by verilog examples, John Wiley & Sons, pp 139-141.