FPGA Implementation of Logarithm of A Number To Base 2

RAJAGOPAL A.
Dept. of E & C,
Dayananda Sagar College of Engineering, VTU
Bengaluru, India.
gopiluck@gmail.com

KARIBASAPPA K.
Dept. of E & C,
Dayananda Sagar Academy of Technology and Management, VTU
Bengaluru, India.
karibasappably@gmail.com

VASUNDARA PATEL K. S.
Dept. of E & C,
BMS College of Engineering, VTU
Bengaluru, India.
vasu.ece@bmsce.ac.in

***Abstract* — The miraculous powers of modern day calculations rests on three important discoveries namely Arabic notation, decimal fractions and logarithm. The concept of logarithm has been used for more than three hundred years and it has simplified lot of calculations but in today's modern systems such as FPGAs which are employed in Software Defined Radios there is a need to efficiently implement logarithm on these programmable devices. But logarithms are implemented either with a look-up table or CORDIC processing system where each has their own drawbacks. In this paper logarithm to the base of any number has been implemented using simple operations such as addition and shift operations with high degree of accuracy. The implementation is done by writing a HDL code for a Artix-7 series FPGA device.**

Keywords — Field programmable Gate Array (FPGA), Hardware description Language (HDL)

I. INTRODUCTION

Logarithm is an elementary function which is widely used in many scientific computations since many decades. As a fundamental function, logarithms can be effectively used in hardware calculations, handling multiplications, divisions, roots and powers etc. Realization of logarithmic function in hardware plays an important role because of its significant applications in wide variety of fields such as digital signal processing, communication, information theory, artificial neural networks and other multimedia applications. There are three generally used bases for logarithms, namely, Logarithms with base-10 called as Common logarithms, Logarithms with base-e called as Natural logarithms and Logarithms with base-2 called as Binary logarithms. It is evident that logarithm to base-2 in binary form would be best suitable for the digital implementation. Due to its versatile applications it becomes imperative to study and effectively implement these logarithmic techniques on programmable devices like FPGA.

The existing effective methods to compute logarithmic values of data using digital circuits can be divided into two main types : One is Look-up table technique. The other is Iterative method. The first method is faster but only used in case of low-precision requirement. The second method is slower, but best suited for high precision applications. In this paper, an efficient algorithm is presented based on binary logarithm computation which can be successfully implemented on FPGA circuit.

It was reported in [5] that the method of approximating the log to the base 2 described in [3] [4] based on linear interpolation between the powers of two improved on accuracy but the complexity increased. The accuracy reported in [5] was much better than [3] but in [6] normalization was done to improve accuracy even further. In [7] the logarithm values were stored in a look-up table avoiding all the multiplications and divisions that are associated with logarithm and this resulted in better accuracy than [3]. Available methods for implementations of logarithm rely mainly on look-up table technique which takes up lot of memory in FPGA resulting in wastage of precious memory. In this paper an efficient algorithm [2] is used which improves the accuracy and it replaces effectively the look-up table technique as it consumes expensive memory.

The paper presented here is organized as follows: Section II describes the algorithm, Section III briefly explains the comparison, Section **I**V shows the FPGA results and Section V presents the conclusion.

II. ALGORITHM

In this paper an algorithm to compute the Logarithm of a number to base 2 in binary form which can be implemented on FPGA circuit is described here below. The algorithm **[2]** is as follows:

Logarithm Algorithm:

Step 1 : Input the value X
Step 2 : counter i =1:7
Step 3 : $X_i = X$
Step 4 : If $X_i < 1$, then $X_i = X_i * 2$ and characteristics ch = ch-1
Else if $X_i > 2$, then $X_i = X_i / 2$ and ch = ch+1
Step 5 : If $1 < X_i < 2$
then, if $X_i^2 < 2$, then $b_i = 0$ and $X_{i+1} = X_i$
else if $X_i^2 > 2$, then $b_i = 1$ and $X_{i+1} = X_i / 2$
Step 6 : Increment the counter i
Step 7 : Repeat steps from 4 to 6 until i greater than 7

The above algorithm can be extended to any base by using following formula,
$\log_b x = \log_2 x / \log_2 b$. The early values of b_i are not significantly affected by introducing the round-off errors as the precise values of Xi are not needed.

The Pseudo HDL code is described below:

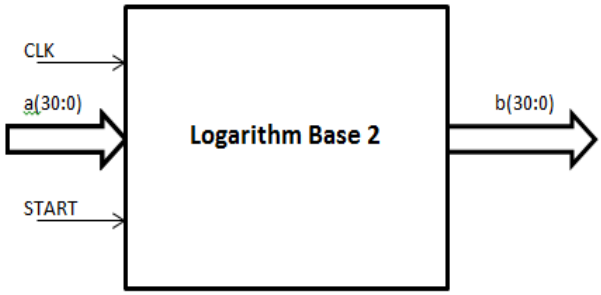

Fig. 1. Logarithm base 2 System Block diagram

The code is implemented using Fixed point arithmetic, Here the most significant bit represents the Sign :

1. Declare inputs and outputs
   Input clk, start, and a(30:0) represents 8 bit before binary points and 23 bits after the binary points.
2. Declare component mantissa
3. Process block execution-
   On posedge of clk, check if state==0 then, atemp<= a else check if atemp is between 1 and 2 i.e. 1< atemp < 2, then st <=1 to start mantissa calculation (fig 2).
   When state = 1, check if counter cnt initialised with value of 23 (as there are 23 bits after binary) has become 0.
   If cn = 0, then state <= 0 and done <= 1 (to end the mantissa calculation) else decrement the counter cnt by 1. Now check whether the squared number mant_temp is greater than 2. If so, bit value is assigned 1 and update mant_temp by right shifting 1 bit else bit value is assigned 0.
   Else if atemp < 1 or atemp > 2, left shift or right shift respectively and update the characteristic and sign variable.
4. Port map the values to mantissa as parameters
5. When done = 1, prefix sign and characteristic variable to output

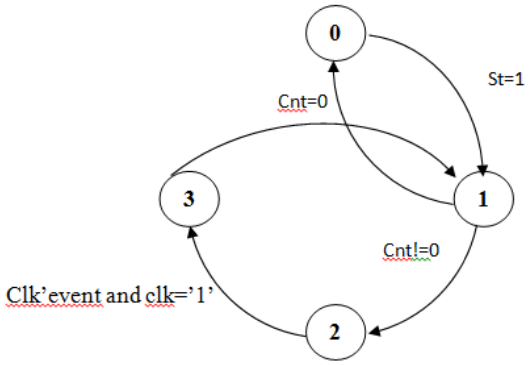

Fig. 2. State diagram for Mantissa Calculation

III. COMPARISION

The algorithm was verified with lookup table technique and percentage of error was plotted. The lookup table values were stored with a difference of 0.1 and 0.05. The error was measured using Matlab. Figures (3), (4) and (5) below show the percentage of error plot as a function of input data with difference of 0.1 and 0.05 in look-up table.

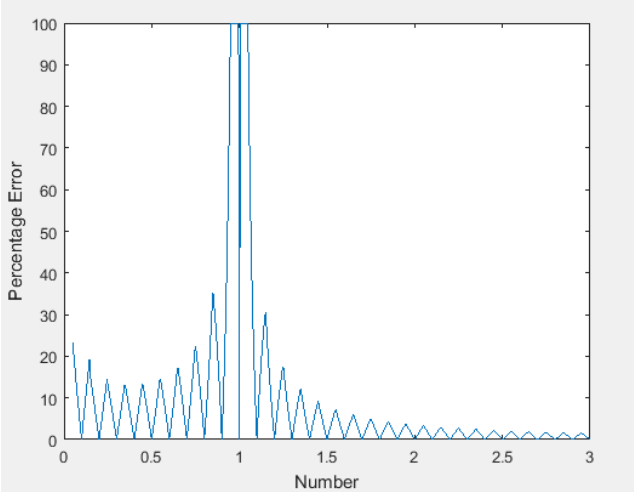

Fig. 3. Percentage of error as a function of Input number (with 0.1 difference in Look-up table)

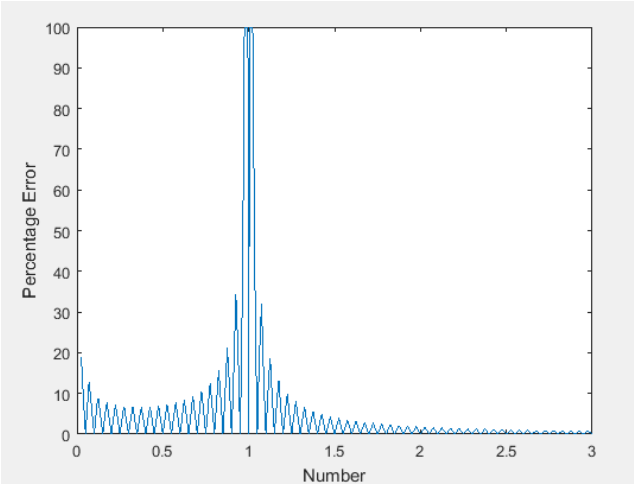

Fig 4. Percentage of error as a function of Input number (with 0.05 difference in Look-up table)

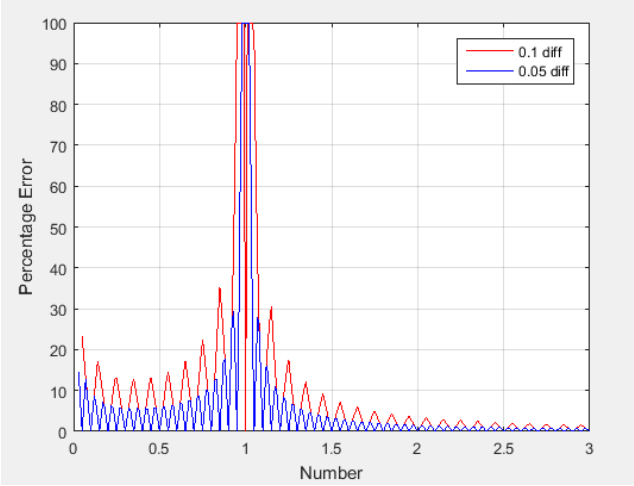

Fig. 5. Percentage of error as a function of Input number (with 0.05 and 0.1 difference in Look-up table)

IV. FPGA RESULTS

The code was successfully implemented on Nexys 4 Artix7 FPGA kit(XC7A100T-1CSG324C). The figures 6 and 7 show the result for input less than 1 and greater than 1. Here the algorithm is taking 23 binary numbers and can get accuracy up to 21 binary decimal places and verification is done for same, which indicates a discrepancy of at most 0.0000005. The accuracy of the code increases as you increase the number of bits after binary points.

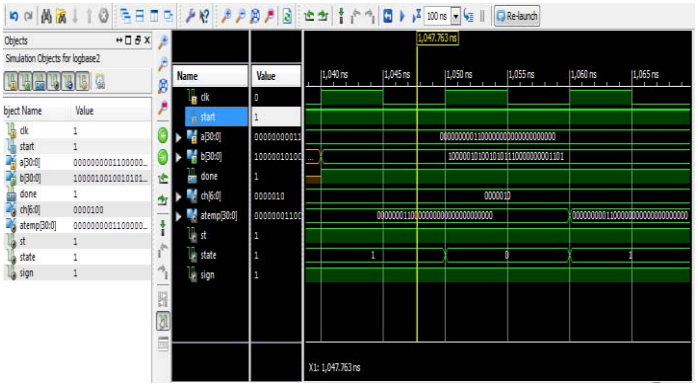

Fig. 6. Simulation results for input less than 1

Input is 0.375
$Log_2(0.375) = -1.41503$
a[30:0]=0000 0000 0110 0000 0000 0000 0000 000
b[30:0]=1000 0010 1001 0101 1100 0000 0001 101
exact value=1000 0010 1001 0101 1100 0000 0001 101(using caluculator)
answer is -2+.5849625 in binary= -1.4150374,Since the most significant bit is negative.

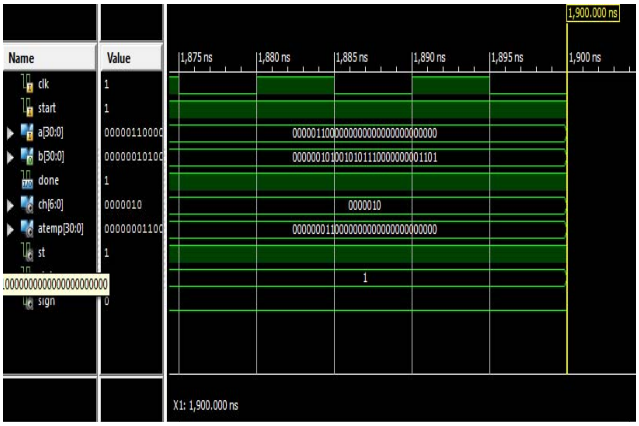

Fig. 7. Simulation results for input greater than 2.

Input is 6
$Log_2(6)=2.5849625$
a[30:0]=0000 0110 0000 0000 0000 0000 0000 000
b[30:0]=0000 0010 1001 0101 1100 0000 0001 101
exact value=0000 0010 1001 0101 1100 0000 0001 101(using caluculator)
answer is 2.5849625 in binary, most significant bit is Positive.

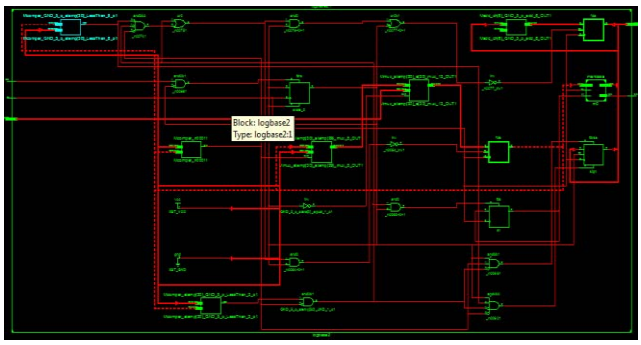

Fig. 8. RTL Schematic of Log base 2.

Table 1. shows the FPGA resource Utilization Summary (Device utilization summary) of the logarithm base 2. Table 2 shows the Timing summary of the same.

TABLE I.

FPGA Resource Utilization on Artix-7of Logarithm Base 2

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
No. of Slice registers	123	126800	1%
No. of Slice LUT's	180	63400	1%
No. of fully used Slice LUT-FF pairs	106	192	55%
No. of Bonded IOB's	65	210	30%

TABLE 2.

Timing Summary:	
Speed Grade	-3
Minimum period	6.334ns(MaximumFrequency:157.888MHz)
Minimum input arrival time before clock	1.006ns
Maximum output required time after clock	0.673ns

V. CONCLUSION

When compared to existing implementations of logarithms on a hardware requiring look-up table and interpolation techniques or other situations by using CORDIC algorithm, the technique described above is efficient and also very accurate. The above algorithm is successfully implemented on FPGA using Shift and Add method contributing higher accuracy accommodating a considerable delay. This delay induced can be overcome by pipelining technique. This algorithm finds wider application in signal and image processing where an accuracy is of prime importance. When compared to exsisting methods, this technique yields significantly lower memory utilization. Moreover the above algorithm can be used to calculate the logarithm of a number to any base.

REFERENCES

[1] Eli Maor, "e The story of a number", Princeton University press, 1994

[2] P.H Philo ,"An Algorithm to evaluate the logarithm of a number to base 2" Presented at a Conference on 'Electronic Switching and Logic Circuit Design' organized by the College of Technology, Letchworth with the support of the I.E.R.E. and held at Letchworth on 24th October 1968.

[3] J. N. Mitchell, "Computer multiplication and division using binary logarithms," *IRE Trans. Electron. Computers*, vol. 11, pp. 512–517, Aug. 1962.

[4] M.combet, H.Zonneveld, and L.Verbeek "Computation of the base two logarithm of binary numbers" IEEE Trans Electronic Computers,pp. 863-867, December 1965.

[5] S. L. SanGregory, C. Brothers, D. Gallagher, and R. E. Siferd, "A fast low-power logarithm approximation with CMOS VLSI implementation," in *Proc. IEEE Midw. Symp. Circuits Syst.*, Aug. 1999, vol. 1, pp. 388–391.

[6] S. E. Tropea, "FPGA Implementation of base-N logarithm," in Proc. 3rd Southern Conference on Programmable Logic, Feb. 2007, pp. 27-32.

[7] S. Paul, N. Jayakumar, and S. P. Khatri, "A fast hardware approach for approximate, efficient logarithm and antilogarithm computations," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 17, no. 2, pp. 269–277, Feb. 2009

[8] K. H. Abed and R. E. Siferd, "CMOS VLSI implementation of a low power logarithmic converter," *IEEE Trans. Computers*, vol. 52, no. 11,pp. 1421–1433, Nov. 2003.

[9] J.-A. Pineiro, "Algorithm and architecture for logarithm, exponential, and powering computation," *IEEE Trans. Computers* vol. 53, no. 9, pp. 1085–1096, Sep. 2004

[10] D. K. Kostopoulos, "An algorithm for the computation of binary logarithms," *IEEE Trans. Computers*, vol. 40, no. 11, pp. 1267–1270, Nov.1991 .

[11] Frangakis, G. P., "Fast binary logarithm computing circuit for binary numbers," Electronics Letters, pp. 574-575, July 1980

[12] V. Paliouras and T. Stouraitis, "Low-power properties of the logarithmic number system," in *Proc. 15th IEEE Symp. Comput. Arith. (ARITH)*, Washington, DC, 2001, p. 229