AIP Conference Proceedings

Small range logarithm calculation on Intel Quartus II Verilog

Cite as: AIP Conference Proceedings **1930**, 020045 (2018); https://doi.org/10.1063/1.5022939
Published Online: 02 February 2018

Muhazam Mustapha, Anis Shahida Mokhtar and Azfar Asyrafie Ahmad

AIP Publishing

Small Range Logarithm Calculation on Intel Quartus II Verilog

Muhazam Mustapha[1, a)], Anis Shahida Mokhtar[1), b)] and Azfar Asyrafie Ahmad[2), c)]

[1]*Faculty of Engineering, Universiti Pertahanan Nasional Malaysia, 57000 Kuala Lumpur, Malaysia*
[2]*Rejimen Semboyan Diraja, Angkatan Tentera Malaysia*

[a)]Corresponding author: muhazam.mustapha@gmail.com
[b)]anis@upnm.edu.my
[c)]azfar.asyarafie@gmail.com

Abstract. Logarithm function is the inverse of exponential function. This paper implement power series of natural logarithm function using Verilog HDL in Quartus II. The mode of design used is RTL in order to decrease the number of megafunctions. The simulations were done to determine the precision and number of LEs used so that the output calculated accurately. It is found that the accuracy of the system only valid for the range of 1 to *e*.

INTRODUCTION

Logarithm is defined as the exponent to which another fixed value or its base which must be raised to produce that original number. Logarithms were first published in 1620 which were invented self-reliantly by a Scotsman, John Napier, and a Swiss, Joost Burgi. The logarithms of Napier were first introduced in 1614 whereas the logarithms from Burgi were first established in 1620 [1]. There has been a work on implementing logarithm function on hardware with CORDIC approach [2]. However, the approach in this work was based on the publications by [3-4], implemented mathematical functions' Power Series as Verilog fixed point calculations. The purpose of conducting this work was to complete the idea to implement mathematical functions library in FPGAs. This work will be conducted by comparing the values in Register Transfer Level (RTL) approach. All observed values were obtained using Quartus II. However, after methods in [3-4] were implemented on logarithm power series, the range of accuracy was only for inputs of 1 to *e*. The work was covering only 16-bit data, 14-bit fixed point format for input values. The calculation used in this work was only for 5 terms in between $1 \leq y \leq e$ range for the input.

LITERATURE REVIEW

Logarithms were greatly foreshadowed in comparison to the invention of arithmetic and geometric sequences. In a geometric sequence the successor formed a constant ratio with the previous value [5]. Mathematician John Napier invented a well-known mathematical artefact, the inventive numbering rods more quaintly known as "Napier's bones" that was famous for his device that assisted with computation and facilitated computation of complex calculations by a mechanical means [1]. A natural logarithm is a logarithm with base *e*, i.e. an irrational constant approximated to 2.718281828. It is written as $\log_e x$ or ln *x*. Power series (or Taylor or Maclaurin series for some special cases) on the other hands is defined as a summation of sequences that converges to a real number *x* in the form of $\sum_{n=0}^{\infty} a_n x^n$ [6]. The power series used in this work was as in Equation (1).

$$ln\left(\frac{1+x}{1-x}\right) = 2\left(x + \frac{x^3}{3} + \frac{x^5}{5} + \frac{x^7}{7} \ldots\right) \qquad -1 < x < 1 \tag{1}$$

International Conference on Engineering and Technology (IntCET 2017)
AIP Conf. Proc. 1930, 020045-1–020045-5; https://doi.org/10.1063/1.5022939
Published by AIP Publishing. 978-0-7354-1622-2/$30.00

where the function of x represents the n-th term in the series. For this work, the series is just considered until 5th term as it is enough for a 14-bit fraction fixed point and small values of input of $1 < y < e$. The mathematical algorithms were simulated for FPGA using RTL approach as it is one of the designs that can be used in designing a processor in terms of the flow of digital signals (data) between hardware registers and the logical operations performed. The RTL system was synchronoust. HDL used was Verilog to create high level representation of circuits, from which lower level representations and ultimately actual wiring can be derived.

Most designers specify RTL design as a possible transferred and operations performed on input and output or register data, and define the control that specifies when to transfer and operate on data [7]. At this level, RTL level, the gate level instantiation of independent flip-flops and logical operators is replaced with the registers and data-flow description of the transfers.

METHODOLOGY

The work was conducted in two phases, where first, the RTL design was derived from the algorithmic modelling of the natural logarithm function. Then this design was verified in Quartus II software and the output results were compared with the calculated values. In this paper, the scope was to calculate natural logarithm only from 1 to *e* with 16-bit data, 14 fraction fixed point – denoted by [2.14] fixed point. The value of fraction part of log repeats in every power of base. Due to this fact, the value of log 1 until log 10 can be used to calculate log of other values in powers of 10. For example, $\log 50 = \log (10 \times 5) = 1 + \log 5$. This is because, the repeated values are in the range from 1 to 10. In natural logarithm, there are the same cases of repeated values. This fact can be used to calculate logarithms of large values.

RTL Design

Before designing the RTL circuit, the algorithmic model of the digital system that was going to be designed was derived. Through the algorithm the desired level of behaviour of the proposed digital system would be known precisely. For this work, the power series used is the Equation 1. The equation was modelled FPGA implementation based on the model in [3-4]. A functional block diagram (FBD) containing a control unit (CU) and Data unit (DU) was constructed. Figure 1 shows the FBD of natural logarithm function. Unlike four terms which were used in [3-4], five terms were used in this design instead. This is because with five terms, the percentage of errors in the results were expected to be low enough. Further discussions will be done in the analysis part of this paper. This design was then implemented on Quartus II for verification and simulation.

Implementation on Quartus II

Implementation of the design in Quartus II was done using Verilog [8] referring to the FBD. Functional and timing simulations were obtained after compiling the code. In the compilation report, the number of logic element (LE) used by the design was obtained. For correctness testing, a waveform file was created and simulated. The simulated outputs were then compared to the calculated values of the term to check the accuracy of the data. Figure 2 shows the natural logarithm function calculation waveform. The value of the calculations obtained from the waveform labelled as Output Value. For speed testing of the design, the simulation mode of the waveform was set to timing. In the output waveform, the time after the start signal goes low, consisting of 5 terms clock cycles, until the stop signal goes high, is the time delay of the logarithm calculation.

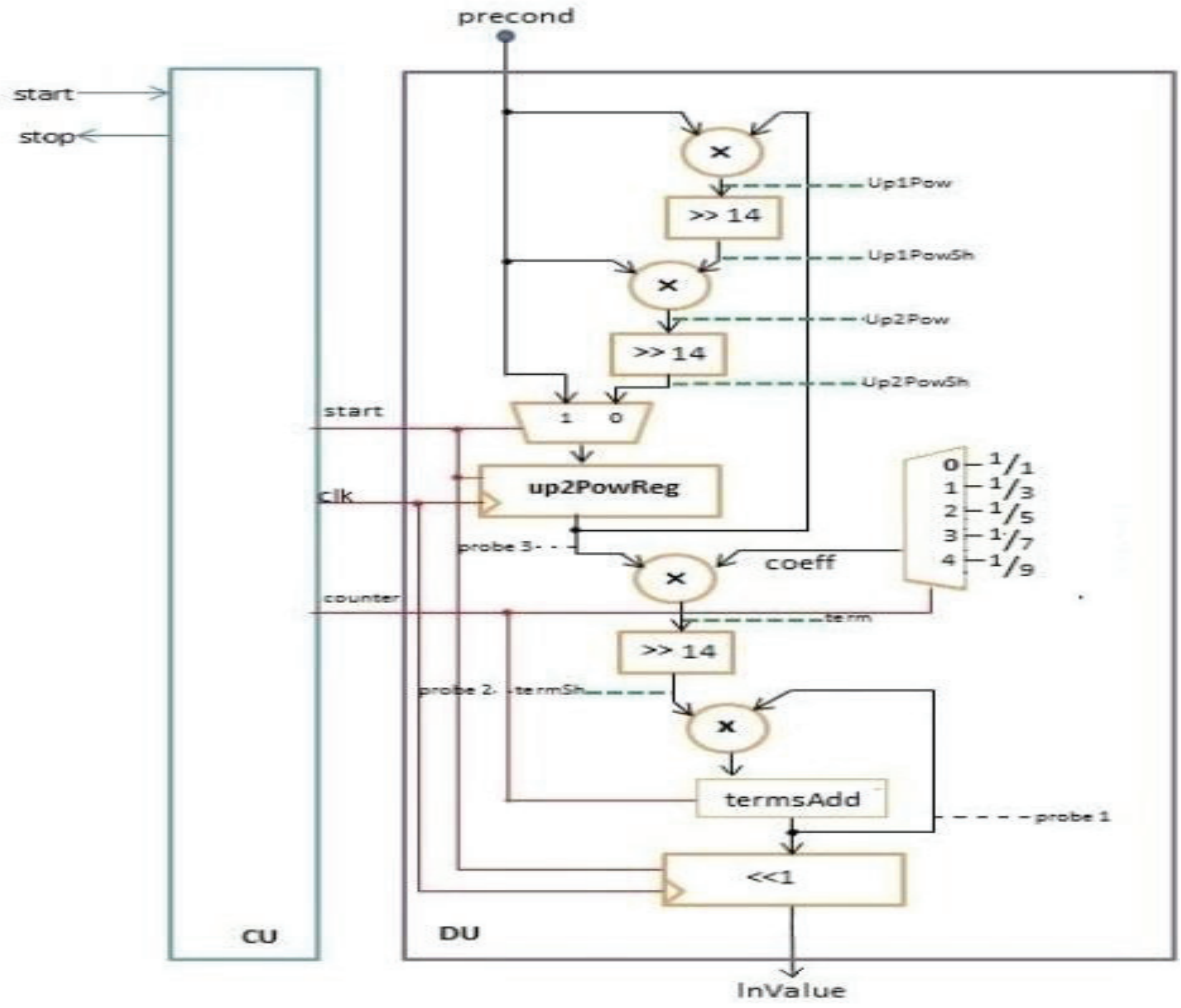

FIGURE 1. RTL design

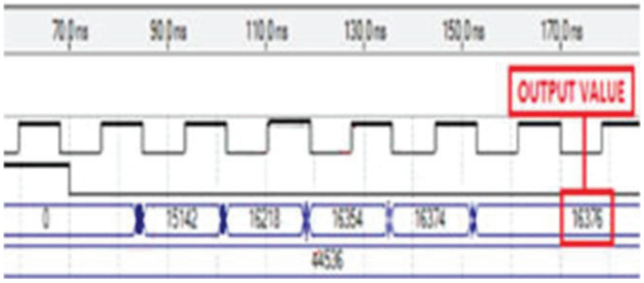

FIGURE 2. Waveform the calculation

RESULTS AND DISCUSSION

In this part, the results obtained from the simulation is discussed. The discussion done in two parts; first the verification of result; and second, to compare with different values of log natural to observe the time delay that being applied.

Verification of Result

For verification of the FBD design of the mathematical function, it is done based on the output waveform obtained through the simulations. It was then compared with the actual values. The calculated value does fit the simulated log natural value as shown in Fig. 3. This is due to the fact of the number of bits used in the design – where 14 bit is accurate enough even with error when doing multiplications of two 16-bits binary number, and while there is no overflow if the simulated and calculated values is between $1 \leq y \leq e$.

This only occurred when the value of the persecond did not exceed 16-bits. The binary values were close enough to the actual value so this action had no effect on the results. The following Table 2 proved that when the value of calculated natural logarithm is correct, the value of simulation will be slightly the same with 99.94% accuracy. Even though the values of natural logarithm can only be computed in range $1 \leq y \leq e$, the function can still be calculated for the value of larger or smaller inputs, outside the range. The repeated nature of logarithm values allows this to be done. This is kept for the next improvement of this work.

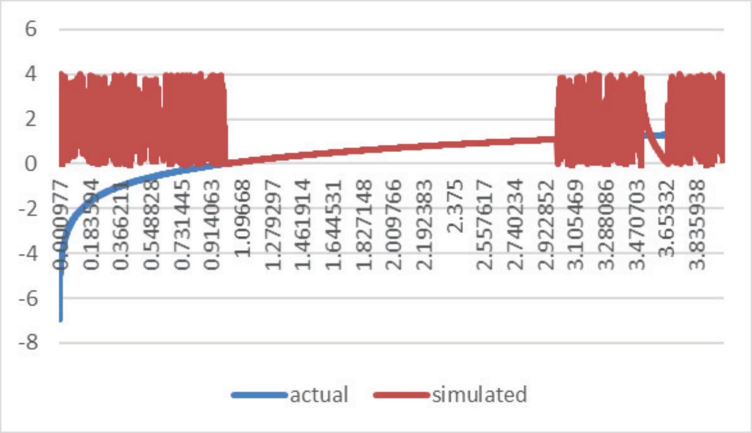

FIGURE 3. Graph of comparison between calculated and simulated values

Table 1. Comparison of calculated (actual) values and simulated values for log natural function

ln	ln*(2^{14})	Simulated	Simulated/(2^{14})	Calculated
1.34	21954	4792	0.2925	0.2927
1.68	27525	8496	0.5186	0.5188
2.02	33095	11512	0.7031	0.7031
2.36	38666	14060	0.8582	0.8587
2.72	44536	16376	0.9995	1.0000

Time Delay

Referring to Fig. 2, the fastest acceptable period for clocking was found to be 17ns whereby the simulated values are still obtained correctly. Since 5 clocks were needed, corresponding to the 5 terms computed, then the total time delay for this design is 5 × 17 = 85ns.

CONCLUSION

The result of this work showed a large room for improvements. First is to explore the bit size that is the best for calculation as opposed to the number of terms used. The 14 bits used in fraction promises a very precise calculation, but the nature of the power series, that doesn't have factorial denominators in the coefficient, prevents it to actually achieve the theoretical accuracy. Second is to equip the system so that it can handle calculation of larger or smaller values by repeated divisions or multiplications.

REFERENCES

1. K. M. Clark and C. Montelle Logarithms: The Early History of a Familiar Function - John Napier Introduces Logarithms (Mathematical Association of America, Washington DC, 2011).
2. S. Wang, Y. Shang, H. Ding, C. Wang and J. Hu, Research Journal of Applied Sciences, Engineering and Technology **6**, 119-122 (2013).
3. I. N. Mahmud, “Hardware implementation of improved mathematical functions on FPGA with RTL approach,” Final project, National Defense University of Malaysia, 2014.
4. I. N. Mahmud and M. Mustapha, “Hardware implementation of improved trigonometric functions on FPGA with RTL approach,” in 3rd International Conference on Mathematical Applications in Engineering (2014), pp. 1-8
5. P. T. P. Tang, ACM Transactions on Mathematical Software **16**, 378-400 (1990).
6. E. Kreyszig, Advanced Engineering Mathematics (Wiley, New Jersey, 2011).
7. A. Maalej and P. Y. Martinez, “Formal proof between two designs having different level of abstraction,” in 3rd International Conference on Design and Technology of Integrated Systems in Nanoscale Era (2008), pp. 1-4.
8. K. H. Yeap, H. Nisar, K. H. Chong, Z. I. Rizman, C. C. Yang, Y. T. Yong and K. C. Lai, International Journal of Electronics, Computer and Communications Technologies **3**, 10-16 (2012).