International Journal of Scientific Engineering and Technology ISSN :2277-1581
Volume No.4 Issue No.7, pp: 387-391 01 July 2015

Efficient Logarithmic Function Approximation

Amjad F. Hajjar, Mohammad H. Awedh
Department of Electrical and Computer Engineering,King Abdulaziz University, Jeddah, Saudi Arabia
ahajjar@kau.edu.sa, mhawedh@kau.edu.sa

Abstract:A criteria is developed for the approximations of a logarithmic function to piecewise straight lines at multiple segments such that the maximum absolute error is minimized. The optimum cutting points for segmentation are estimated numerically with an exhaustive search. Our hardware implementation is restricted to integer operations (addition and shifting).

Keywords— logarithmic function, approximation error, absolute error, linear approximation.

I. INTRODUCTION

The trade-off of accuracy for speed has its significant performance improvements at expenses of some quality degradation. Approximate computation of logarithm function is an example of such trade-off. It has its use in applications (e.g. signal and image processing, telecommunication and biomedical systems) that require fast computation with acceptable amount of errors in the result of computation. Another use of approximate computation of logarithmic function is when implementing it in hardware for fast computation with low costs of size [1, 2, 3]. The use of the polynomial approximation method was first proposed by [4]. In this paper, any integer number N is expressed in binary representation as:

$$N = \sum_{i=0}^{k} 2^i z_i \tag{1}$$

where, z_i is the i^{th} binary digit, and k is the number of bits to represent N. Now if the most significant bit, z_k, is of a binary value '1', (1) can be rewritten as:

$$N = 2^k \times \left(1 + \sum_{i=0}^{k-1} 2^{i-k} z_i\right) = 2^k (1 + x) \tag{2}$$

where, x is the summation of the fraction part which is in the range $0 \le x < 1$. Taking the binary logarithm for both sides of (2) yields:

$$\log_2 N = k + \log_2 (1 + x) \tag{3}$$

Therefore, the logarithmic value of N can be obtained by detecting the position of the most significant nonzero bit of N and computing the approximate value of $\log_2(1 + x)$.

Multiple approaches can be followed in approximating the logarithm function in (3) to a line. Mitchell [4] approximated the logarithm curve by a piecewise linear curve in powers of two intervals. For $0 \le x < 1$, he uses the following approximation:

$$\log_2 (1 + x) \approx x \tag{4}$$

Mitchell's approximation, however, yields a rather large error (the maximum approximation error is as high as 0.08639). Several methods have been developed in the literature to improve the accuracy of Mitchell's approximation by adding error correction techniques. These techniques are implemented using piecewise linear interpolation or LUT-based methods. In [5], Mitchell's approximation error is stored in a LUT, whose values are interpolated and added to its approximation.

The error in approximating the logarithmic function in (4) can be reduced by dividing the range of x in (3) into regions, r, and apply a low-degree polynomial to approximate logarithmic curve in each region. For simple hardware implementation, r is chosen to be a power of two, and the range of x is subdivided into equal length sub regions. The minimization of the approximation error using piecewise linear approximations with different values of r is presented in [1, 3, 6, 7]. [6] and [8] divide the rang of x into four regions and select the coefficients to minimize the absolute error or the mean square error, respectively. In [3], the Mitchell's error is approximated by a four-region piecewise linear interpolation and a LUT-based correction is used to correct the piecewise interpolation error. [9] divides the logarithmic curve into two symmetric regions obtaining an error of 0.045.

The main point to obtain an efficient logarithmic approximation (minimizing approximation error) using piecewise linear approximations lies in finding the optimized coefficients. In [10], the authors presented a technique that minimizes the maximum relative approximation error while using a reduced number of nonzero bits for the coefficients. This technique is based on mixed-integer linear programming. In [11], eight-region piecewise linear approximation is used. The approximation coefficients are stored in the eight locations of an 18-bit ROM.

II. LINEAR APPROXIMATIONS

Multiple approaches can be followed in approximating a logarithm function to a line. First let's state our target function:

$$f(x) = \log_2 (1 + x) \quad 0 \le x < 1 \tag{5}$$

This function can be approximated to a single line or a piecewise linear function. Hence, let's consider the following target:

$$f(x) = \log_2 (1 + x) \quad \alpha \le x < \beta \tag{6}$$

where $0 \le \alpha < \beta < 1$. Let $r(x)$ be the residual function between $f(x)$ and its approximating line in a selected range such that:

$$r(x) = \log_2 (1 + x) - ax - b \quad \alpha \le x < \beta \tag{7}$$

International Journal of Scientific Engineering and Technology
Volume No.4 Issue No.7, pp: 387-391
ISSN :2277-1581
01 July 2015

In minimizing this error function, multiple norm functions could be used to find the optimum line parameters. The Euclidean norm (L_2-*norm*) is mostly used to minimize the area under the residuals (known as the least square error minimization); however, it does not guarantee the maximum error of all points to be bounded under a certain value. Utilizing the Manhattan norm (L_1-*norm*) on the other hand ensures that the absolute error values are bounded, though L_1-*norm* has discontinuities in its derivatives and hence its mathematical derivation is so complex [12, 13].

$$\begin{aligned} L_1-norm: &\quad \min_{a,b}\int_{\alpha}^{\beta}|r(x)|\cdot dx \\ L_2-norm: &\quad \min_{a,b}\int_{\alpha}^{\beta}r^2(x)\cdot dx \end{aligned} \tag{8}$$

The L_1-*norm* minimization can be carried out numerically. However, the integral of the absolute residuals is not our goal; rather we want to make sure that the highest residual is below a certain level. This can be accomplished by minimizing the distance between the minimum and maximum residual points with respect to the line slope (a) and then setting the bias constant (b) to divide this length in half to guarantee the least error level.

In studying the residual function $r(x)$, we find that it takes one of four shapes shown in Figure (1) controlled solely by the line slope (a) value.

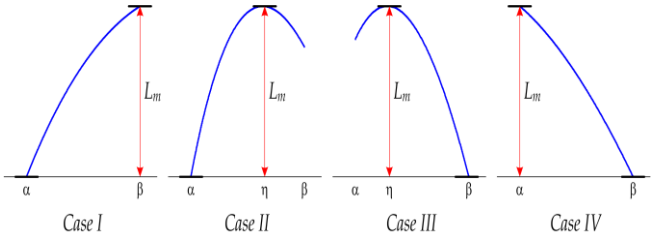

Figure (1): Cases of Maximum Residual Lengths

The first and second derivatives of the residual function are:

$$\begin{aligned} \frac{\partial}{\partial x}r(x) &= \frac{1}{\ln(2)}\frac{1}{(1+x)}-a \\ \frac{\partial^2}{\partial x^2}r(x) &= \frac{-1}{\ln(2)}\frac{1}{(1+x)^2} \end{aligned} \tag{9}$$

Notice that the second derivative of the residual function is always negative indicating that the function concaves downwards over all its range. Solving the first derivative for x gives the local maximum point η as:

$$\eta = \frac{1}{a\ln(2)}-1 \tag{10}$$

Now based on η being inside or outside the range $\alpha \le x < \beta$, we classify the residual function into four cases presented to find the maximum residual length:

$$L_m=\begin{cases} r(\beta)-r(\alpha) &= +\log_2\left(\frac{1+\beta}{1+\alpha}\right)-a(\beta-\alpha) & \eta\ge\beta \\ r(\eta)-r(\alpha) &= -\log_2[a(1+\alpha)]+a(1+\alpha)-C & \alpha\le\eta\le\beta, r(\alpha)\le r(\beta) \\ r(\eta)-r(\beta) &= -\log_2[a(1+\beta)]+a(1+\beta)-C & \alpha\le\eta\le\beta, r(\beta)\le r(\alpha) \\ r(\alpha)-r(\beta) &= -\log_2\left(\frac{1+\beta}{1+\alpha}\right)+a(\beta-\alpha) & \eta\le\alpha \end{cases} \tag{11}$$

where,

$$C=\frac{1+\ln(\ln(2))}{\ln(2)}\approx 0.91 \tag{12}$$

The residual lengths of the four cases are linear functions of the slope a. In minimizing L_m at the different cases while satisfying the range conditions, we found that the optimum solutions for each case are as follows:

$$\begin{aligned} \text{CaseI}: &\quad \hat{a} = \frac{1}{\ln 2(1+\beta)} \\ \text{CaseII,III}: &\quad \hat{a} = \frac{1}{\beta-\alpha}\log_2\left(\frac{1+\beta}{1+\alpha}\right) \\ \text{CaseIV}: &\quad \hat{a} = \frac{1}{\ln 2(1+\alpha)} \end{aligned} \tag{13}$$

Cases I and IV do not have their optimum solutions in the range specified, hence the residual lengths are not optimum. The least residual length however occurs at cases II and III at points α and β where the residuals are equal, giving the optimum solution as:

$$\hat{a}(\alpha,\beta)=\frac{1}{\beta-\alpha}\log_2\left(\frac{1+\beta}{1+\alpha}\right) \tag{14}$$

with a maximum residual length of:

$$\begin{aligned} L_m(\alpha,\beta) &= \frac{\log_2(1+\beta)^{1+\alpha}-\log_2(1+\alpha)^{1+\beta}}{\beta-\alpha} \\ &\quad -\log_2\hat{a}-C \end{aligned} \tag{15}$$

The optimum b value is such that the residual line is centred at zero, thus:

$$\begin{aligned} \hat{b}(\alpha,\beta) &= r(\alpha)+\frac{1}{2}L_m \\ &= \frac{1}{2}\frac{\log_2(1+\beta)^{1-\alpha}-\log_2(1+\alpha)^{1-\beta}}{\beta-\alpha} \\ &\quad -\frac{1}{2}\log_2\hat{a}-\frac{1}{2}C \end{aligned} \tag{16}$$

A. Single Line Case

The case where the logarithmic function is approximated to only one line is when α=0 and β=1, yielding the optimum approximating line of slope $\hat{a}=1$ and $\hat{b}=(1-C)/2\approx 0.043$ with a maximum residual length of L_m = $1-C \approx 0.086$, and a maximum absolute error of half of L_m, i.e. 0.043 or $2^{-4.54}$. Figure (2) shows the approximating function error. If the least square method is used, the optimum line

ISSN :2277-1581
01 July 2015

parameters would have been a = 0.984, b = 0.065, giving a maximum error at x = 0 of –0.065 or $2^{-3.94}$.

B. Piecewise Linear Approximation

When choosing to divide the original logarithm function into multiple lines, the question is where would be the optimum points to slice the function at. The solution can be estimated numerically with an exhaustive search. For example, suppose that we choose to divide the function into two segments. Let c_1 be the break point of the division. Then we have two approximating lines to the function. Each line has a maximum residual length of L_m based on its range. If the sum of both residuals is then minimized with respect to the cutting point c_1, then we have the optimum approximation. For this case, the total error function to be optimized is:

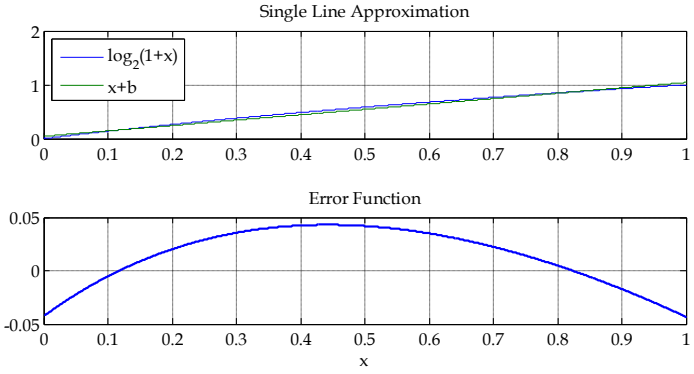

Figure (2): Signle Line Approximation

$$TRL_1 = L_m(0, c_1) + L_m(c_1, 1) \tag{17}$$

where, TRL_1 is the sum of the residual lengths to be minimized for one break point. The optimum solution for c_1 is derived numerically in this case by differentiating the TRL_1 function with respect to c_1 and equating it to zero. In general, if the logarithmic function is to be approximated into (n+1) segments with n break points: c_1 to c_n, then the sum of the residual lengths is:

$$TRL_n = L_m(0, c_1) + \sum_{i=2}^{n} L_m(c_{i-1}, c_i) + L_m(c_n, 1) \tag{18}$$

The optimum solution will be obtained by differentiating the TRL_n function with respect to each break point c_i and equating them to zeros. This derivation is carried out for a number of chosen segments; Figure (3) shows the location of the break points for the different number of segments. The figure is plotted in a logarithmic scale in both axes in order to exhibit the rhythmic shape of the solution set.

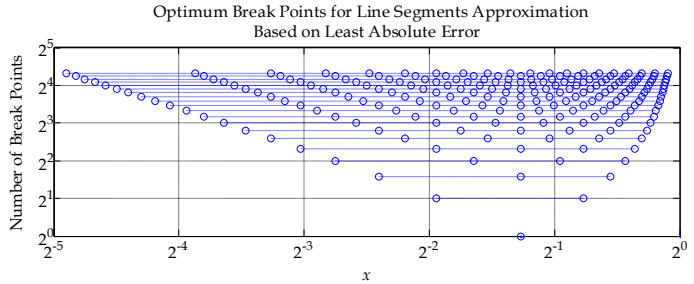

Figure (3): Optimum Break Points for Line Segments Approximation

Table (1) lists the optimum break points for different number of segments along with the maximum absolute error of the approximation noted in powers of 2. Figure (4) shows this error for up to 2^6 break points.

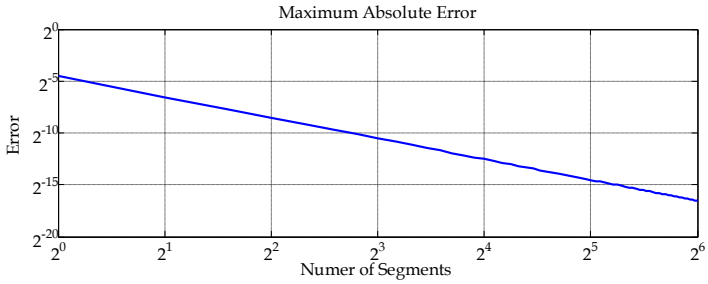

Figure (4): Maximum Absolute Error for Different Number of Segmentations

Table (1): Optimum Break Points at a Number of Segments

N_{seg}	*Location of Optimum Break Points*	*AME*
1	-	$2^{-4.54}$
2	0.414	$2^{-6.53}$
3	0.260 0.587	$2^{-7.70}$
4	0.189 0.414 0.682	$2^{-8.53}$
5	0.149 0.320 0.516 0.741	$2^{-9.17}$
6	0.122 0.260 0.414 0.587 0.782	$2^{-9.70}$
7	0.104 0.219 0.346 0.486 0.641 0.811	$2^{-10.14}$
8	0.091 0.189 0.297 0.414 0.542 0.682 0.834	$2^{-10.53}$
9	0.080 0.167 0.260 0.361 0.470 0.587 0.714 0.852	$2^{-10.87}$
10	0.072 0.149 0.231 0.320 0.414 0.516 0.625 0.741 0.866	$2^{-11.17}$

AME: absolute maximum error

C. Error Sensitivity

In the previous sections, the logarithmic function was approximated to piecewise straight lines at multiple segments such that the maximum absolute error is minimized. When implementing the approximation in hardware with binary numbers, we face a problem of dealing with the real continuous numbers of the slopes, shifts, and break points. First however we discuss the sensitivity of the maximum absolute error due to the changes in any of these parameters. The sensitivity of the error function E with respect to a parameter p is generally defined as:

$$S_E^p \equiv \frac{\partial E/E}{\partial p/p} \tag{19}$$

Figure (5) shows how sensitive the maximum absolute error is with respect to either the line slope or shift when the logarithmic function is approximated to a single line over the range $0 \le x \le 1$. At the worst case, the error is ±100% sensitive to the change in either the line slope or shift. For example, if the slope is increased by 5%, the maximum absolute error will be increased by 5% of its nominal value too. Note that when changing the slope of the shift from its nominal value, the other parameter is set fixed without change. This will be considered as a potential improvement when discretizing the parameter in the next section.

International Journal of Scientific Engineering and Technology
Volume No.4 Issue No.7, pp: 387-391
ISSN :2277-1581
01 July 2015

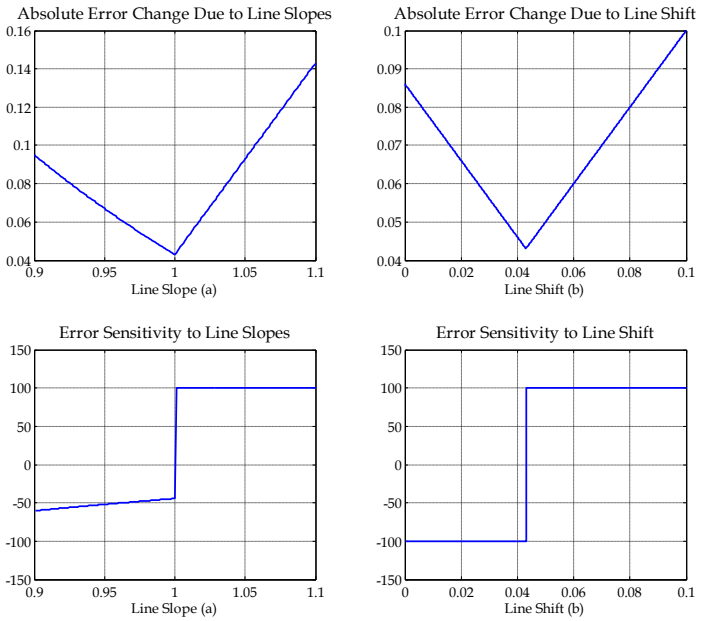

Figure (5): Maximum Absolute Error Sensitivity Due to Line Slope and Shift Changes

D. Discretizing the Solution

The main goal of this paper is to find a simple mathematical expression that is easy to implement yet fast enough for real time computation. Hence our hardware implementation is restricted to integer additions and performing multiplications and divisions by powers of two, 2^n.

Mixtures of the above simple operations would yield more complicated operations. For example, a multiplication by 3 can be split into two simple steps: a multiplication by 2 and an addition. A multiplication by 1.4 can be either approximated by a multiplication of 1.5: two steps, a division by 2 and an addition; or a multiplication by 1.375: three steps, an addition, a division by 4 and a division by 8. Figure (6) shows the error of this digitization process in two or three steps to approximate a real number in the range from 0 to 2. At most, the percentage errors are 14.3% for the 2-integer steps and 6.7% for the 3-integer steps.

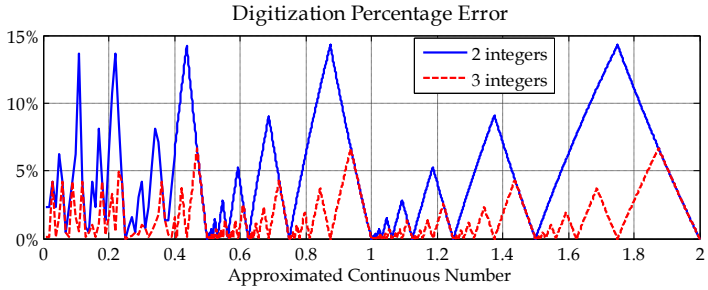

Figure (6): Digitization Percentage Error of a Continuous Number

III. DISCRETE PIECEWISE LINEAR FUNCTION

Finally, the logarithmic function of interest can be approximated into binary operations as follows:

1. a comparator to decide on the line segment
2. the line slope is mapped into a number of integer operations of divisions by powers of 2 and additions
3. the line shift is simply an addition

Table (1) lists up to 10 segments piecewise linear approximation in the continuous case as well as the discrete case of 2- and 3-integer steps to approximate the line slopes. Errors of the approximation are listed in the Table and shown in Figure (7) for comparison. For instant, segmenting the range into 3 segments yields the errors of $2^{-6.32}$, $2^{-7.56}$, and $2^{-7.70}$ for 2-integer discrete case, 3-integer discrete case, and the continuous case, respectively.

Figure (7): Error of Approximation

Table (1): Approximation errors in the contiouous case, 2-integer, and 3-integer discrete cases

N_{seg}	*Continuous Case*			*2 Powers Discrete*			*3 Powers Discrete*		
	a	***b***	***AE***	***a***	***b***	***AE***	***a***	***b***	***AE***
1	1	0.430360	$2^{-4.54}$	1	0.430360	$2^{-4.54}$	1	0.430360	$2^{-4.54}$
2	1.2071000	0.0108120	$2^{-6.53}$	1.25	-0.001812	$2^{-5.02}$	1.1875	0.012822	$2^{-6.12}$
	0.8535500	0.1572600		0.75	0.220220		0.875	0.139360	
3	1.2824000	0.0048100	$2^{-7.70}$	1.25	0.007072	$2^{-6.32}$	1.2813	0.004885	$2^{-7.56}$
	1.0179000	0.0735760		1.0156	0.074340		1.0176	0.073677	
	0.8078900	0.1969200		0.75	0.238610		0.8125	0.192820	
4	1.3213000	0.0027065	$2^{-8.53}$	1.25	0.007072	$2^{-6.81}$	1.3125	0.003127	$2^{-6.95}$
	1.1111000	0.0424820		1.125	0.037578		1.125	0.037578	
	0.9343000	0.1157100		1	0.077139		0.875	0.145640	
	0.7856500	0.2170500		0.75	0.244880		0.78125	0.220410	
5	1.3450000	0.0017324	$2^{-9.17}$	1.25	0.007064	$2^{-6.30}$	1.375	-0.001410	$2^{-8.36}$
	1.1709000	0.0276220		1.125	0.036930		1.1563	0.030468	
	1.0193000	0.0760510		1.0156	0.077414		1.0195	0.075954	
	0.8873700	0.1441000		1	0.071591		0.875	0.151230	
	0.7725000	0.2292300		0.75	0.247640		0.76563	0.234790	
6	1.3610000	0.0012032	$2^{-9.70}$	1.25	0.006795	$2^{-6.58}$	1.375	-0.000039	$2^{-8.20}$
	1.2125000	0.0193870		1.25	0.011288		1.1875	0.023443	
	1.0802000	0.0537700		1.0625	0.059138		1.0781	0.054389	
	0.9623500	0.1025900		1	0.082668		1	0.082668	
	0.8573500	0.1642600		0.75	0.236550		0.875	0.151490	
	0.7638200	0.2373900		0.75	0.249050		0.76563	0.235680	
7	1.3724000	0.0008840	$2^{-10.14}$	1.25	0.006373	$2^{-7.08}$	1.375	0.000687	$2^{-8.06}$
	1.2431000	0.0143520		1.25	0.013045		1.25	0.013045	
	1.1259000	0.0400180		1.125	0.040235		1.126	0.039984	
	1.0197000	0.0767330		1.0156	0.078298		1.0195	0.076806	
	0.9235900	0.1234500		1	0.079525		0.875	0.149940	
	0.8365100	0.1792400		0.75	0.241170		0.8125	0.195940	
	0.7576500	0.2432300		0.75	0.249830		0.75781	0.243080	
8	1.3811000	0.0006768	$2^{-10.53}$	1.5	-0.005381	$2^{-7.10}$	1.375	0.000820	$2^{-9.25}$
	1.2665000	0.0110510		1.25	0.013004		1.2656	0.011149	
	1.1614000	0.0309390		1.125	0.039144		1.1563	0.032049	
	1.0650000	0.0595510		1.0625	0.060358		1.0645	0.059720	
	0.9765800	0.0961630		1	0.084430		1	0.084430	
	0.8955300	0.1401100		1	0.075498		0.875	0.152140	
	0.8212100	0.1907800		0.75	0.244070		0.8125	0.197090	
	0.7530500	0.2476300		0.75	0.250300		0.75391	0.246810	
9	1.3879000	0.0005348	$2^{-10.87}$	1.5	-0.004489	$2^{-6.98}$	1.375	0.000820	$2^{-8.25}$
	1.2850000	0.0087709		1.25	0.012590		1.2813	0.009152	

ISSN :2277-1581
01 July 2015

	1.1897000	0.0246330		1.25	0.011246	1.1875	0.025056	
	1.1015000	0.0475550		1.125	0.039851	1.0938	0.049792	
	1.0199000	0.0770140		1.0156	0.078673	1.0195	0.077152	
	0.9442900	0.1125300		1	0.082543	1	0.082543	
	0.8742900	0.1536400		0.75	0.234010	0.875	0.153160	
	0.8094900	0.1999500		0.75	0.245990	0.8125	0.197490	
	0.7494800	0.2510500		0.75	0.250550	0.75	0.250550	
10	1.3933000	0.0004332	$2^{-11.17}$	1.5	-0.003830	$2^{-7.32}$	1.375	0.000820	$2^{-8.70}$
	1.3000000	0.0071299		1.25	0.012205	1.3125	0.005544	
	1.2129000	0.0200750		1.25	0.012606	1.1875	0.024532	
	1.1317000	0.0388500		1.125	0.040554	1.1328	0.038516	
	1.0559000	0.0630640		1.0625	0.060503	1.0625	0.060503	
	0.9851900	0.0923530		1	0.085177	1	0.085177	
	0.9192200	0.1263800		1	0.079889	0.875	0.151150	
	0.8576600	0.1648200		0.75	0.237900	0.875	0.152630	
	0.8002200	0.2073900		0.75	0.247310	0.8125	0.197230	
	0.7466400	0.2538000		0.75	0.250550	0.75	0.250550	

IV. CONCLUSION

Approximate computation of logarithm function finds its application in a multitude of applications. These applications require fast computation with acceptable amount of errors. In this paper, we develop a criteria for the approximations of logarithmic function to piecewise straight lines at multiple segments such that the maximum absolute error is minimized. For easy and fast hardware implementation of real time computation, our implementation is limited to integer additions and subtractions, and multiplications and divisions by powers of two. This yields to an error of 6.7% for the 3-integer steps approximation of real numbers.

V. REFERENCES

i *K. Abed, R. Siferd, Cmos vlsi implementation of a low-power logarithmic converter, Computers, IEEE Transactions on 52 (11) (2003) 1421–1433.*

ii *S. Paul, N. Jayakumar, S. Khatri, A fast hardware approach for approximate, efficient logarithm and antilogarithm computations, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 17 (2) (2009) 269–277.*

iii *R. Gutierrez, J. Valls, Low cost hardware implementation of logarithm approximation, Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 19 (12) (2011) 2326–2330. doi:10.1109/TVLSI.2010.2081387.*

iv *J. N. Mitchell, Computer multiplication and division using binary logarithms, Electronic Computers, IRE Transactions on EC-11 (4) (1962) 512– 517.*

v *G. Kmetz, Floating point/logarithmic conversion system, US Patent 4583180 (Apr. 15 1986), google.com/patents/US4583180*

vi *M. Combet, H. Van Zonneveld, L. Verbeek, Computation of the base two logarithm of binary numbers, Electronic Computers, IEEE Transactions on EC-14 (6) (1965) 863–867.*

vii *S. SanGregory, C. Brothers, D. Gallagher, R. Siferd, A fast, low-power logarithm approximation with cmos vlsi implementation, in: Circuits and Systems, 1999. 42nd Midwest Symposium on, Vol. 1, 1999, pp. 388–391*

viii *E. L. Hall, D. Lynch, I. Dwyer, S.J., Generation of products and quotients using approximate binary logarithms for digital filtering applications, Computers, IEEE Transactions on C-19 (2) (1970) 97–105. doi:10.1109/TC.1970.222874.*

ix *T.-B. Juang, S.-H. Chen, H.-J. Cheng, A lower error and rom-free logarithmic converter for digital signal processing applications, Circuits and Systems II: Express Briefs, IEEE Transactions on 56 (12) (2009) 931–935. doi:10.1109/TCSII.2009.2035270.*

x *D. De Caro, N. Petra, A. Strollo, Efficient logarithmic converters for digital signal processing applications, Circuits and Systems II: Express Briefs, IEEE Transactions on 58 (10) (2011) 667–671.*

xi *J. Pandey, A. Karmakar, C. Shekhar, S. Gurunarayanan, An fpga-based fixed-point architecture for binary logarithmic computation, in: Image Information Processing (ICIIP), 2013 IEEE Second International Conference on, 2013, pp. 383–388*

xii *Mark Schmid, Least Squares Optimization with L1-Norm Regularization, 2005.*

xiii *T. Hastie, R. Tibshirani, and J. H. Friedman. The Elements of Statistical Learning. Springer, August 2001.*