Design of 32 bit Logarithmic Arithmetic Logic Unit

A thesis report submitted for BTP phase I

by

Raushan Singh (160102054)
Piyush Raj (160102049)

under the guidance of

Dr. Gaurav Trivedi

to the

DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
November 2019

Abstract

The aim of our project is to design a simple model that provides logarithmic and anti-logarithmic operations in arithmetic logic unit that uses the principle of Linear piece-wise approximation. The linear approximation method reduces the latency factor, consumes less power and area as compared to the conventional methods that use multipliers and Look-Up Tables. The logarithmic processing unit receive the input in form of floating point number that has sign, an exponent, and mantissa, and output is a fixed point number which has an integer and a fraction. The anti-logarithmic unit has input as fixed point number and output as floating point number. The log/anti-log unit can reduce the computation time of operations of multiplication, division and other complex calculations.

Contents

List of Figures **ii**

1 Introduction **1**

1.1 Linear Approximation Motivation 1

1.2 Linear approximation Application 2

2 Linear piecewise Approximation **4**

2.1 Background 4

2.2 Conventional Methods 4

2.3 Linear Approximation Method 5

3 Result and Simulation **11**

3.1 Observations 11

4 Conclusion and Future Work **14**

References **15**

List of Figures

1.1 Multiplication Hardware 2

1.2 Floating Point 3

2.1 Logarithmic Unit Flow Chart 6

2.2 Logarithmic Unit 7

2.3 Anti-logarithmic Flow Chart 9

3.1 Linear Piecewise Logarithmic Approximation Comparison-I 12

3.2 Linear Piecewise Anti-logarithmic Approximation Comparison-I 12

3.3 Linear Piecewise Logarithmic Approximation Comparison-II 13

3.4 Linear Piecewise Anti-logarithmic Approximation Comparison-II 13

Chapter 1

Introduction

Arithmetic Logic Unit is actually the main part of any central processing unit in a computer system. It can also be referred as the heart of digital computers. It enables to do all the arithmetical and logical operations in the system. Day by Day, the world is seeing an exponential growth in the field of technology and computers. The processing unit are becoming faster and efficient. This report provides the background of our simple approach to reduce the processing speed of an ALU. The first chapter deals with the motivation of our approach to use logarithmic/ anti-logarithmic units. The second chapter deals with our linear approximation method implementation. The third chapter deals with the result and simulation of the proposed approach and the fourth chapter is the conclusion and future works.

1.1 Linear Approximation Motivation

Arithmetic operations like multiplication and division take more computation time than other operations. Following figure explains the conventional way of computing the multiplication. Here is a 32-bit multiplier register which requires a 64-bit multiplicand register. This is because we need to shift 32-bit multiplicand that will move 32 bit left, over 32 steps.

Hence 64-bit multiplicand register is required with initialization as zero. This register is then shifted left 1 bit each step to align the multiplicand with the sum being accumulated in the 64-bit Product register.

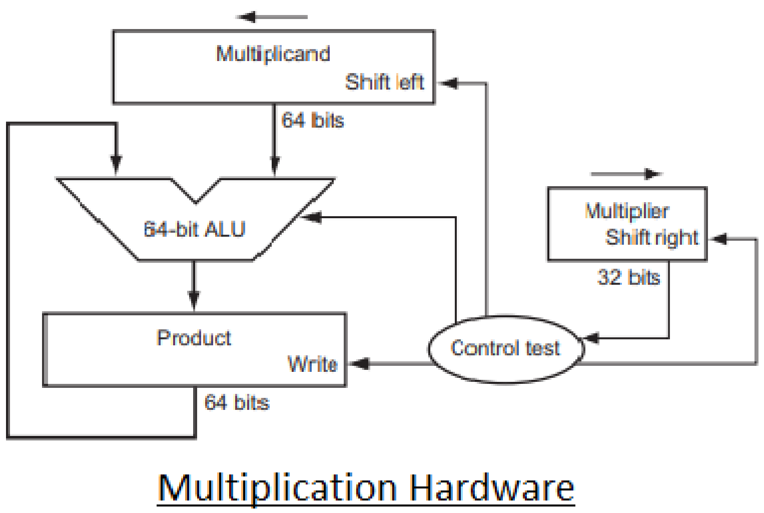

Fig. 1.1 Multiplication Hardware

To overcome this problem we perform the multication and division operation using logarithmic and anti-logarithmic unit, which is develop by linear approximation method. Here logarithm and anti-logarithm of a floating type number is computed using just linear addition and shifting which would be discussed in the coming chapters. Suppose we have to calculate multiplication of two numbers "A" and "B". If we take log of "A" and "B", and add them followed by anti-log of the whole thing will give the product "AB". Since our approach uses linear approximation, the computational time reduces many folds and hence this method becomes more efficient than the conventional method of computing multiplication.

1.2 Linear approximation Application

Many technical fields such as scientific computing, Digital Signal Processing, Computer graphics, Artificial neural networks, logarithmic number applications, and other media applications use logarithmic and anti-logarithmic calculations extensively for data processing. Logarithm is actually the exponent to which another base number is raised to make that number. It is the inverse process of exponentiation. The logarithm to the base 2 is referred to as common logarithm. Antilogarithm is the inverse operation of logarithm.

For linear approximation technique we are using, requires the input to be in floating number.

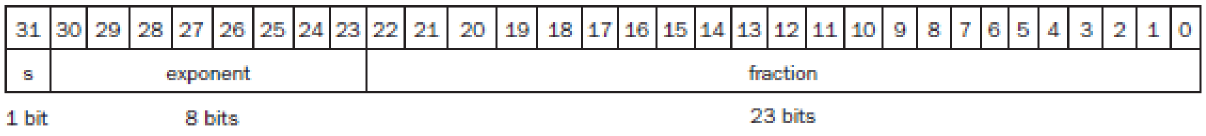

Fig. 1.2 Floating Point

Floating point number in hardware design consists of three components, i.e. Sign(S), exponent and fraction(mantissa). S is of 1-bit, it denotes the sign of the number ('0' denotes non-negative and '1' denotes negative). Exponent is contained as 8-bits in single precision and 11-bits in double precision. Fraction is the fractional part of the number, it lies between 0 and 1. It is of 23-bits in single precision and 52-bits in double precision. In general Floating point number is represented as shown below:

$$(-1)^S * (1 + F) * 2^E$$

Chapter 2

Linear piecewise Approximation

2.1 Background

Logarithm is a number that represents the power to which a fixed number which is called the base is raised to produce a given number. It can also be seen as a reverse operation of exponential. The logarithm whose base is 2, is said to be common logarithm. Anti-logarithm is actually the reverse operation of this. There are a lot of applications where we use logarithmic and anti-logarithmic computations very extensively such as in DSP (Digital Signal Processing), Artificial neural networks, computer graphics and other scientific computations.

2.2 Conventional Methods

The conventional methods of computing logarithm and anti-logarithm requited the use of Look-Ups Table also know as LUT. These methods used to take 5 to 8 clock cycles, depending upon the hardware implementation. In the conventional method, the parabolic curves of the log/anti-log is divided into fixed divisions and curve fitting process is used to derive the co-efficient and store it into the memory.Suppose one has to perform operation such as in equation $px^2 + qx + r$, then the values of p, q and r are fetched from the Look-Up Table and calculated using the adders and multipliers.These multipliers and LUT which is used in the conventional method, take more area on the hardware and takes 5 to 8 clock cycles to perform the operation.

2.3 Linear Approximation Method

This chapter provides a simple and efficient approach to approximate the logarithm a floating point number. The floating point number is taken log to the base 2 and converted to a fixed point number using a log component. We want to integrate our processor with piece-wise linear log/anti-log approximation components.

In our first part, we design piecewise linear log model that computes logarithmic operation, which is a inverse operation of exponentiation. Logarithm of a number is actually exponent of a number called base number which is raised to produce that number. In our design the input is in the form of a floating point number. Floating- point number actually has three part in its 32 bits. The first part is sign bit(S), second one is exponent(E) and mantissa (M). The output we get is fixed point number (also called 'log number') which has two parts :- integer (Z) and fraction(n). This approximation converts the floating point number to fixed point number.

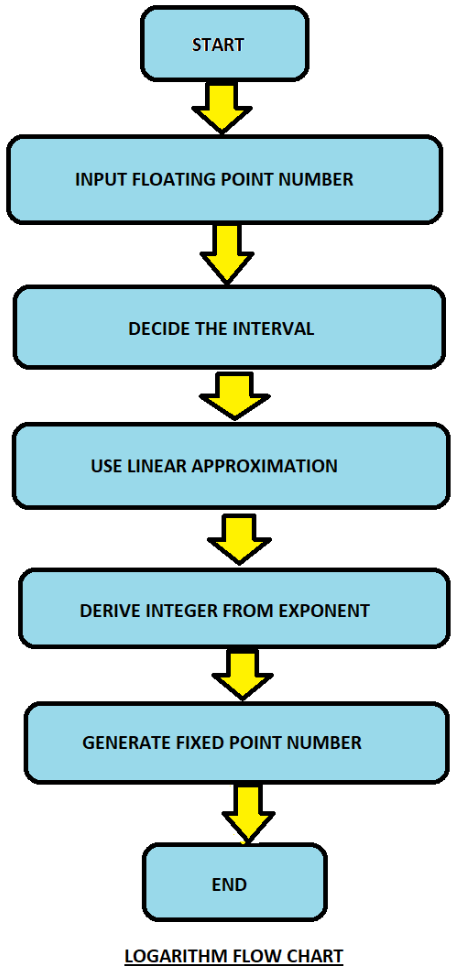
START
INPUT FLOATING POINT NUMBER
DECIDE THE INTERVAL
USE LINEAR APPROXIMATION
DERIVE INTEGER FROM EXPONENT
GENERATE FIXED POINT NUMBER
END
LOGARITHM FLOW CHART

Fig. 2.1 Logarithmic Unit Flow Chart

$$Log(1+n) = n + \frac{1}{4}n + \frac{1}{8}n \qquad 0 < n < 0.125 \tag{2.1}$$

$$n + \frac{1}{4}n + \frac{1}{64} \qquad 0.125 < n < 0.25 \tag{2.2}$$

$$n + \frac{1}{16}n + \frac{1}{16} \qquad 0.25 < n < 0.5 \tag{2.3}$$

$$n - \frac{1}{8}n + \frac{1}{8} + \frac{1}{32} \qquad 0.5 < n < 0.75 \tag{2.4}$$

$$n - \frac{1}{4}n + \frac{1}{4} \qquad 0.75 < n < 1 \tag{2.5}$$

The second part is the anti-logarithmic component that perform anti-log operation in

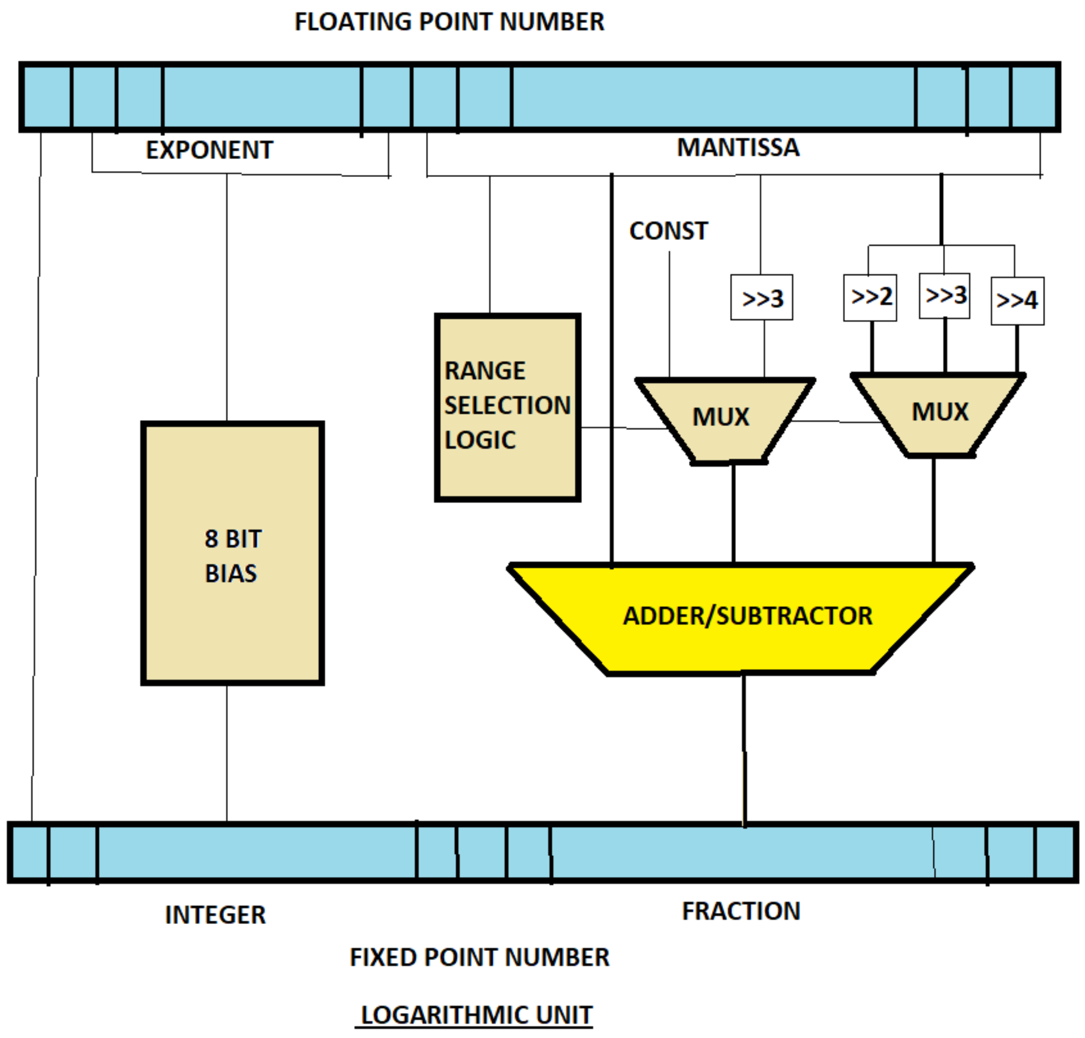

Fig. 2.2 Logarithmic Unit

a similar way. This is shown in figure 2.3.It transforms the fixed point number which was the output of log unit, into floating point output. The equations are shown on page no. 9(equations no. 2.6-2.9).

The following equations are the detailed linear piecewise approximation of logarithmic unit.The piecewise interval taken here is .0625.

$$\mathrm{Log}_2(1+r) = \begin{cases} r + \frac{r}{4} + \frac{r}{8} + \frac{r}{64} & 0 \leq r < 0.0625 \\ r + \frac{r}{4} + \frac{r}{16} + \frac{1}{256} + \frac{1}{1024} & 0.0625 \leq r < 0.125 \\ r + \frac{r}{4} + \frac{1}{64} - \frac{1}{512} & 0.125 \leq r < 0.1875 \\ r + \frac{r}{8} + \frac{r}{16} + \frac{1}{64} + \frac{1}{128} & 0.1875 \leq r < 0.25 \\ r + \frac{r}{8} + \frac{1}{32} + \frac{1}{128} & 0.25 \leq r < 0.3125 \\ r + \frac{r}{16} + \frac{r}{128} + \frac{1}{16} & 0.3125 \leq r < 0.375 \\ r + \frac{r}{64} + \frac{r}{128} + \frac{1}{16} + \frac{1}{128} & 0.375 \leq r < 0.4375 \\ r - \frac{r}{64} + \frac{1}{16} + \frac{1}{32} & 0.4375 \leq r < 0.5 \\ r - \frac{r}{16} + \frac{r}{128} + \frac{1}{8} & 0.5 \leq r < 0.5625 \\ r - \frac{r}{8} + \frac{r}{32} + \frac{1}{8} + \frac{1}{128} & 0.5625 \leq r < 0.625 \\ r - \frac{r}{8} + \frac{1}{8} + \frac{1}{32} & 0.625 \leq r < 0.6875 \\ r - \frac{r}{8} - \frac{r}{32} + \frac{1}{8} + \frac{1}{16} & 0.6875 \leq r < 0.75 \\ r - \frac{r}{4} + \frac{r}{32} + \frac{1}{4} & 0.75 \leq r < 0.8125 \\ r + \frac{r}{8} + \frac{r}{4} & 0.8125 \leq r < 0.875 \\ r - \frac{r}{4} + \frac{1}{4} & 0.875 \leq r < 0.9375 \\ r - \frac{r}{4} - \frac{r}{64} + \frac{1}{4} + \frac{1}{64} & 0.9375 \leq r < 1 \end{cases}$$

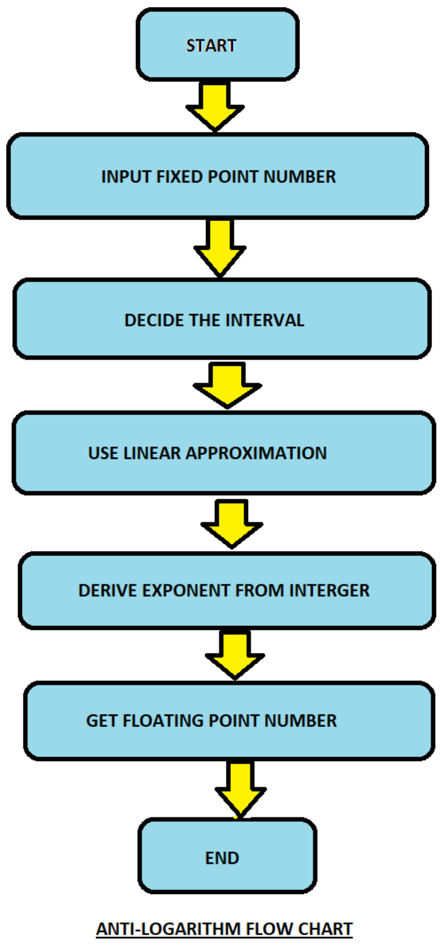
START
INPUT FIXED POINT NUMBER
DECIDE THE INTERVAL
USE LINEAR APPROXIMATION
DERIVE EXPONENT FROM INTERGER
GET FLOATING POINT NUMBER
END
ANTI-LOGARITHM FLOW CHART

Fig. 2.3 Anti-logarithmic Flow Chart

$$2^n = 1 + n - \frac{1}{4}n \qquad 0 < n < 0.25 \tag{2.6}$$

$$1 + n - \frac{1}{8}n - \frac{1}{32} \qquad 0.25 < n < 0.5 \tag{2.7}$$

$$1 + n + \frac{1}{16}n - \frac{1}{8} \qquad 0.5 < n < 0.75 \tag{2.8}$$

$$1 + n + \frac{1}{4}n - \frac{1}{16}n - \frac{1}{4} - \frac{1}{16} \qquad 0.75 < n < 1.0 \tag{2.9}$$

In the above logarithmic anti-linear approximation, we are taking interval gap of 0.25 unit. For increasing accuracy we are taking interval that is the square of previous interval i.e. 0.625 unit.Equations given below are the detailed linear piecewise approximation.

$$2^r = \begin{cases} r - \frac{r}{4} - \frac{r}{32} + 1 & 0 \leq r < 0.0625 \\ r - \frac{r}{4} + \frac{r}{64} + 1 - \frac{1}{512} & 0.0625 \leq r < 0.125 \\ r - \frac{r}{4} + \frac{r}{32} + 1 - \frac{1}{128} & 0.125 \leq r < 0.1875 \\ r - \frac{r}{8} - \frac{r}{16} + 1 - \frac{1}{64} & 0.1875 \leq r < 0.25 \\ r - \frac{r}{8} - \frac{r}{32} + 1 - \frac{1}{32} + \frac{1}{128} & 0.25 \leq r < 0.3125 \\ r - \frac{r}{8} + 1 - \frac{1}{32} & 0.3125 \leq r < 0.375 \\ r - \frac{r}{16} - \frac{r}{64} + 1 - \frac{1}{32} - \frac{1}{64} & 0.375 \leq r < 0.4375 \\ r - \frac{r}{32} - \frac{r}{64} + 1 - \frac{1}{16} & 0.4375 \leq r < 0.5 \\ r + \frac{r}{512} + 1 - \frac{1}{8} + \frac{1}{32} & 0.5 \leq r < 0.5625 \\ r + \frac{r}{16} + \frac{r}{32} + 1 - \frac{1}{8} + \frac{1}{128} & 0.5625 \leq r < 0.625 \\ r + \frac{r}{16} + \frac{n}{32} + 1 - \frac{1}{8} - \frac{1}{64} & 0.625 \leq r < 0.6875 \\ r + \frac{r}{8} + \frac{r}{64} + 1 - \frac{1}{4} + \frac{1}{16} & 0.6875 \leq r < 0.75 \\ r + \frac{r}{8} + \frac{r}{16} + 1 - \frac{1}{4} + \frac{1}{32} & 0.75 \leq r < 0.8125 \\ r + \frac{r}{4} - \frac{r}{128} + 1 - \frac{1}{4} - \frac{1}{128} & 0.8125 \leq r < 0.875 \\ r + \frac{r}{4} + \frac{r}{32} + 1 - \frac{1}{4} - \frac{1}{16} & 0.875 \leq r < 0.9375 \\ r + \frac{r}{4} + \frac{r}{8} + 1 - \frac{1}{4} - \frac{1}{8} & 0.9375 \leq r < 1 \end{cases}$$

Chapter 3

Result and Simulation

3.1 Observations

In this chapter, we have shown the comparison of the ideal logarithmic and anti-logarithmic values with the values calculation from the linear piecewise approximation method of computing log/anti-log. The removal of the LUT and multiplier from the hardware circuits reduces the time latency from 5-8 clock cycles to around one clock cycle.

The **Red** curves are the graphs of **Linear Piece-wise Approximation Method**, while the **Blue** curves are the **ideal logarithm**.

Fig 3.1 and Fig 3.2 displays the comparison of simple linear piece-wise approximation (Pg.No.6 and Pg.No.9)with the ideal graphs. And Fig 3.3 and Fig 3.4 displays the comparison of detailed linear piece-wise approximation(Pg.No.8 and Pg.No.10) with the ideal graphs.

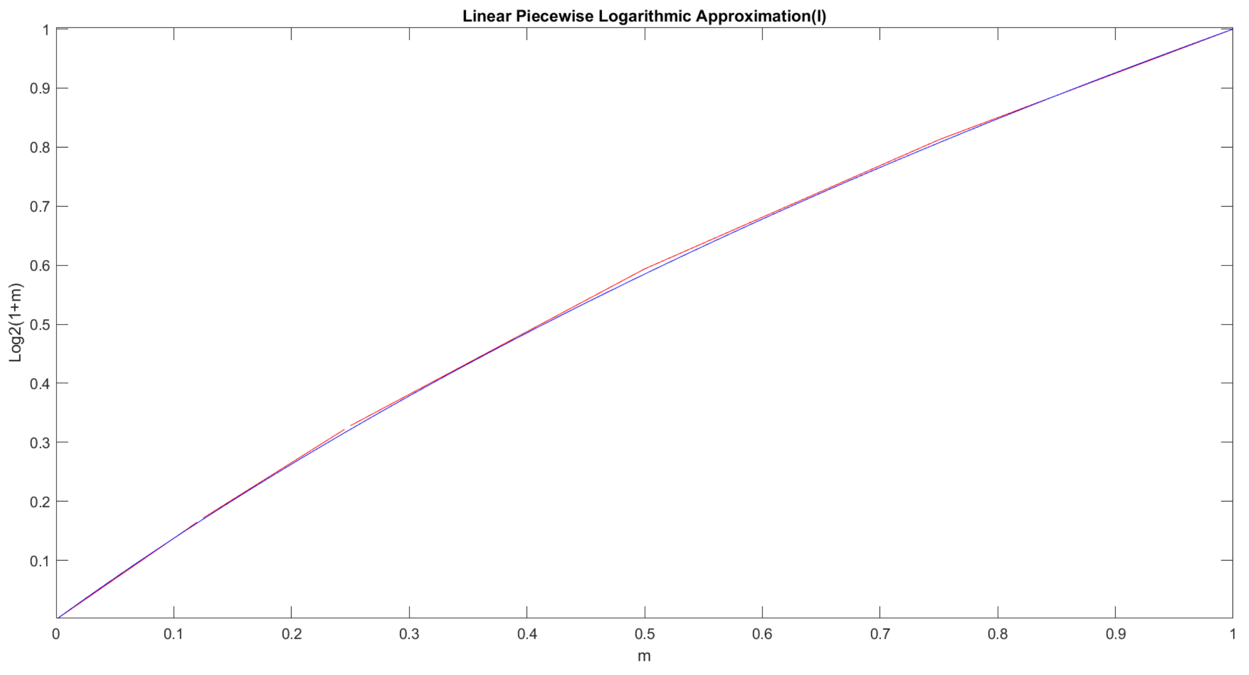

Fig. 3.1 Linear Piecewise Logarithmic Approximation Comparison-I

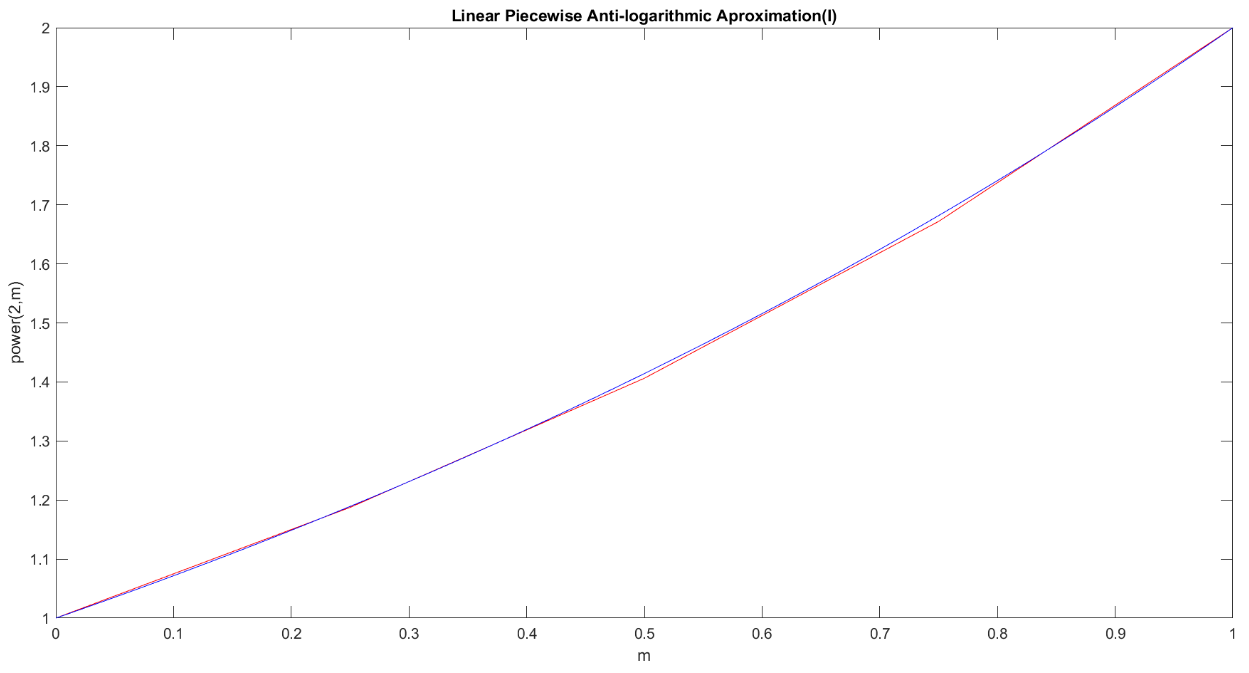

Fig. 3.2 Linear Piecewise Anti-logarithmic Approximation Comparison-I

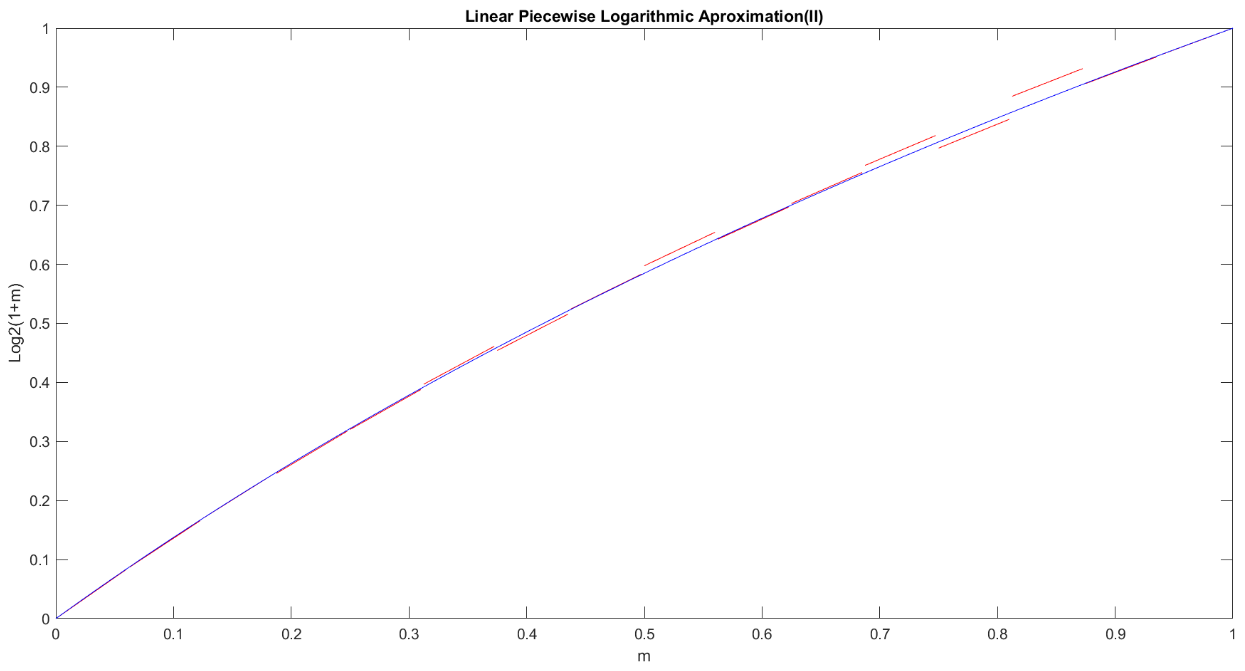

Fig. 3.3 Linear Piecewise Logarithmic Approximation Comparison-II

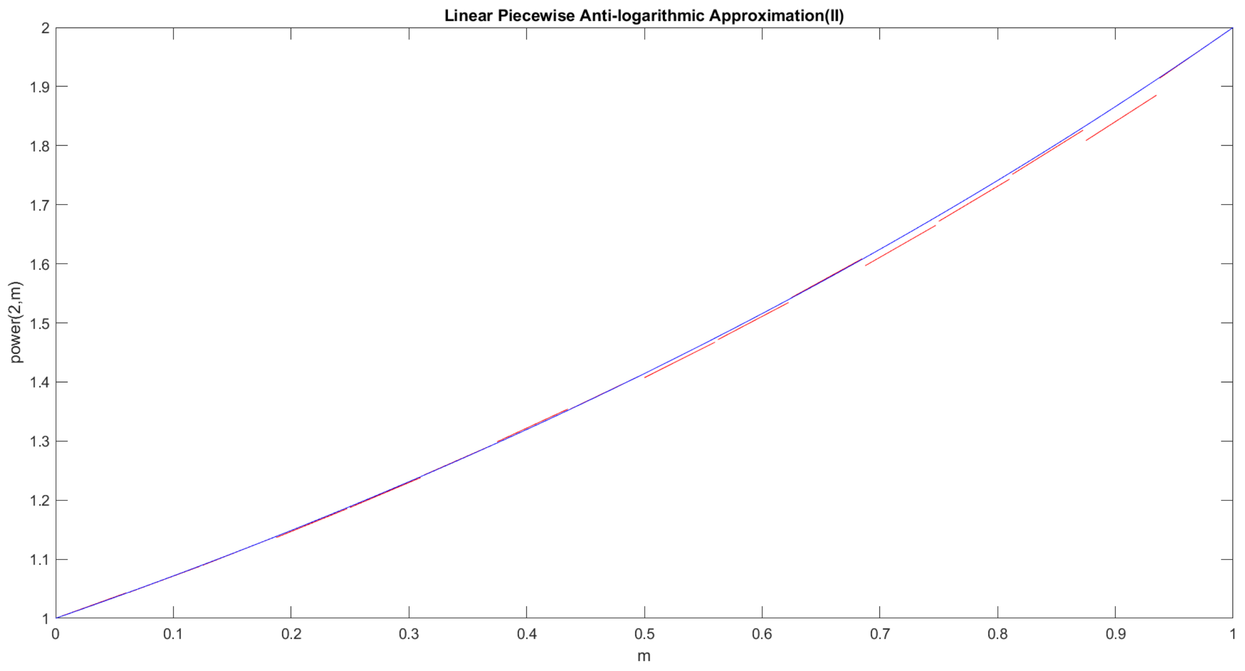

Fig. 3.4 Linear Piecewise Anti-logarithmic Approximation Comparison-II

Chapter 4

Conclusion and Future Work

In our new approach, since we are not using multipliers and Look-Up Tables in our designs, an increase in efficiency along with saving of the area and power is being observed. This is a good advantage over our conventional methods. Since in our approach we are using linear piecewise approximation method, the computation time comes down to around single clock cycle to calculate the log/anti-log operations. Hence we say that the proposed design implementation is faster than the traditional LUT approach , in many aspects like less area, less energy and single clock cycle latency as compared to the conventional method.

The Logarithmic unit integrated with the processing unit will reduce a lot of complex computation and dealing with data in the fields like digital signal processing, artificial intelligence, scientific computing and many others.

References

[1] "Computer Organisation and Design: The Hardware/Software interface" *David A. Patterson, John L. Hennessy* ,Fifth Edition, May. 2016.

[2] Kamlesh R. Pillai and Gurpreet S. Kalsi, "Implementation of Logarithm and Anti-Logarithmic operations based on piecewise linear approximation", *United States Patent Application Publication*, Aug 9, 2018.

[3] "Verilog HDL : A Guide to Digital Design and Synthesis" *Samir Palnitkar* ,Second Edition, Feb. 2003.