Master's Thesis

Hardware Implementation of the Exponential Function Using Taylor Series and Linear Interpolation

Ateeq Ur Rahman Shaik

Department of Electrical and Information Technology,
Faculty of Engineering, LTH, Lund University, May 2014.

Master's Thesis Report

HARDWARE IMPLEMENTATION OF THE EXPONENTIAL FUNCTION USING TAYLOR SERIES AND LINEAR INTERPOLATION

By

Ateeq Ur Rahman Shaik

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Popular Scientific Essay of

”Hardware Implementation of Exponential Function using Taylor Series and Linear Interpolation”

This Master thesis is about ASIC implementation of Exponential Function using two different approaches. Exponential functions are incremental functions which are useful in areas such as Communication circuits, computer graphics, signal and image processing applications. More precisely the unary functions such as sine, cosine, exponential, logarithmic functions are applicable in signal generational circuits such as oscillators, variable gain circuits. Sometimes software implementation does not meet the requirements in some applications, in case of speed is of utmost importance it has to be implemented in hardware. The two methods used for implementation of the function such as Taylor series method and Linear Interpolation.

In Taylor series approach two different solutions as suggested. Each one has its own advantages of its own. Linear Interpolation method is memory intensive design, in which pre-calculated values are stored in an Look-up Table for future calculations of unknown values.

A total of five designs are synthesized using synopsis Design Vision STM065nm process technology using two different methods. Both the architectures are mainly focused on low power, minimizing area and better performance. A comparison is made between these architectures. The target libraries used are high and low threshold voltage. Synopsis PrimeTime is used for the estimation of dynamic power consumption.

Abstract

This thesis work is targeted towards ASIC implementation of the exponential function. Generally, unary functions like trigonometric, logarithmic and exponential functions are useful in areas such as computer graphics, signal processing and image processing for high speed applications. Sometimes software implementation does not meet the requirements in some applications, in case of speed is of utmost importance it has to be implemented in hardware. In this work the exponential function is implemented using two different architectures that are using Taylor series and Linear Interpolation. Both architectures are mainly focused on low power, minimizing area and better performance. A comparison is done between the two architectures. The designs are synthesized in synopsis Design Vision STM065nm process technology with target libraries with high and low threshold voltage. Synopsis PrimeTime is used for the estimation of dynamic power consumption.

Acknowledgments

First of all it is my immense pleasure to have the opportunity working under Prof. Peter Nilsson, sincere words are not sufficient to describe his caring nature, patience and his expertise in the field of digital design. I would like to express my heartfelt gratitude towards my supervisors Erik Hertz, Rakesh Gangarajaiah their timely availability for solving my petty issues and faults during the course of work. Again I would like to say I cannot forget the rest of my life their favor to finishing the final part of Master's degree. It was highly impossible to finish up this work without their guidance and support during this work.

Secondly I would like to thank Prof. Pietro Andreani, Prof. Henrik Sjöland, Prof. Viktor Öwall, and Associate Prof. Joachim N. Rodrigues for completing my MS successfully in the EIT department. I would like to thank my professors for teaching me all the mandatory as well as elective courses, which are part of my degree.

Special thanks to Pia Bruhn for solving all the issues related to the academic management.

I would like to thank all my friends who provided me constant support, caring, without whom the stay in Sweden would not be possible. For the fear of missing I am not naming anybody in the list.

Last but not least thanking almighty for his grace upon and giving me a chance to improve to be better human being. In the end I would like to thank my parents who made me possible to come into this world and pleasant stay till now, as well as all my family members for their support in all times.

Thank you all who supported me directly or indirectly in the process of this journey.

Contents

Abstract 5

Acknowledgments 6

1. Introduction 11

1.1. Project Specification 11

1.2. Thesis Organization 12

2. Theory 13

2.1. Taylor Series 13

2.2. Interpolation 15

2.2.1. Linear Interpolation 15

2.2.2. Polynomial Interpolation 16

2.2.3. Spline Interpolation 17

2.2.4. Piecewise Linear interpolation 18

3. Performance Metrics 19

3.1. Maximum Absolute Error: 19

3.2. Mean Error: 19

3.3. Median: 20

3.4. Standard deviation: 20

3.5. Root-mean-square (RMS) Error: 20

3.6. Error Probability Distribution 21

3.7. Decibel (dB) 21

3.7.1. Power measurement 22

3.7.2. Amplitude measurement 22

3.7.3. Error in dB scale 22

3.8. Power Analysis: 23

3.8.1. Static Power Consumption: 23

3.8.2. Short-Circuit Power Consumption: 23
3.8.3. Dynamic Power Consumption: 24
CHAPTER 4 25
4. Hardware Implementation 25
4.1. Taylor Series Architectures 27
4.1.1. Taylor Series Architecture 1 27
4.1.2. Taylor Series Architecture 2 29
4.1.3. Taylor Series Architecture 2 with pipelined stage 32
4.2. Linear Interpolation Architectures 34
4.2.1. Example of a simple LUT: 34
4.2.2. Linear interpolation Architecture with 1 LUT: 36
4.2.3. Linear interpolation Architecture with 2 LUTs with 256 Intervals: 38
4.2.4. Linear interpolation Architecture with 2 LUTs with 128 intervals: 40
5. Estimation of Error 43
5.1. Taylor series Architecture 1: 43
5.2. Taylor series Architecture 2: 45
5.3. Linear Interpolation architecture with 1 LUT: 48
5.4. Linear Interpolation with 2 LUTs and 256 intervals: 51
5.5. Linear Interpolation with 2 LUTs and 128 intervals: 54
6. Results 59
6.1. Area 59
6.2. Power 63
6.3. Timing 64
6.4. Power estimation of Taylor series architecture 2 using PrimeTime: 65

6.5. Power Estimation for Linear Interpolation with 2 LUTs using PrimeTime: 67

6.6. Comparison of Taylor series Approach and Linear Interpolation Approach 68

7. Conclusions 71

8. Future work 73

REFERENCES 75

List of Figures 77

List of Tables 79

List of Acronyms 81

CHAPTER 1

1. Introduction

Unary functions i.e. trigonometric, logarithmic and exponential functions are very useful in digital design applications. Their applications can be found in digital signal processing, communication systems, robotics, computer graphics etc. Generally for high-speed applications a simple software solution does not necessarily meet the speed requirements. So in order to achieve the desired speed, a hardware solution is investigated. In this regard optimization for speed and area are the major design challenges in a digital application specific integrated circuits (ASIC) implementation.

Researchers are currently trying to find out efficient hardware implementations of these functions to be used in signal and image processing applications. In this thesis, hardware implementations for exponential function using linear interpolation method and Taylor series have been investigated.

1.1. Project Specification

This thesis project is an investigation study and there are no fixed design specifications. The proposed design methods, which are explained in forthcoming chapters, are compared using parameters area, speed, and power consumption. In the design process minimum area and low-power consumption has primarily been taken in account. Table I shows some of the fixed design specifications.

TABLE I. SPECIFICATIONS OF THE HARDWARE IMPLEMENTATION OF EXPONENTIAL FUNCTION

Parameter	**Value**
Supply Voltage	1.2V
Accuracy	>15 bits
Cell Library	LPHVT, LPLVT
Temperature	25°C
Process	STM065nm CMOS process

1.2. Thesis Organization

This thesis report is comprised of ten chapters. Chapter 2 explains the theoretical aspects of Taylor's series and interpolation. Chapter 3 discusses some error metrics required to qualify the designs. In chapter 4 the proposed architectures are implemented on hardware level. In chapter 5 error metrics of implemented designs are demonstrated. Chapter 6 discusses and compares the results of proposed architectures behavioral models and hardware level implementations. The thesis is concluded in chapter 7 and future recommendations are provided in chapter 8.

CHAPTER 2

2. Theory

The proposed architectures in chapter 3 use Taylor series and linear interpolation for implementing the exponential function. Using Taylor series, complex functions are translated into series of low level functions (terms) which can be mapped on hardware. Whereas in interpolation a function is rebuilt using predefined data points which are stored in a memory. This chapter will explain the basics of Taylor series and the interpolation method to form a theoretical background.

2.1. Taylor Series

Taylor series is the representation of a function into an infinite summation of terms of function that are obtained by differentiating that function at any given value [1]. To represent a real function $f(x)$ at any arbitrary point $x = a$, is given by equation (2.1).

$$f(x) = f(a) + f'(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \dots + \frac{f^{(n)}(a)}{n!}(x-a)^n \tag{2.1}$$

The generalized notation of Taylor's series is shown in (2.2).

$$f(x) = f(a) + \sum_{n=0}^{\infty}\left(\frac{(f^{\,(n)}(a) * (x-a)^{\,n}}{n!} \right) \tag{2.2}$$

Taylor series expansion of most common functions such as exponential, logarithmic and trigonometric functions are expressed as follows:

The exponential function e^x is expressed in Taylor series as equation (2.3).

$$e^x = e^a\left[1 + (x-a) + \frac{1}{2}(x-a)^2 + \frac{1}{6}(x-a)^3 + \cdots\right]$$

(2.3)

The Taylor series expansion of a logarithmic function is expressed as shown in (2.4):

$$\ln(x) = \ln(a) + \frac{(x-a)}{a} - \frac{(x-a)^2}{2a^2} + \frac{(x-a)^3}{3a^3} - \cdots$$

(2.4)

Similarly trigonometric functions *sin*(*x*) and *cos*(*x*) can be expressed in Taylor series expansion as shown in equations (2.5) and (2.6).

$$\cos(x) = \cos(a) - \sin(a)(x-a) - \frac{1}{2}\cos(a)(x-a)^2 + \frac{1}{6}\sin(a)(x-a)^3 + \cdots$$

(2.5)

$$\sin(x) = \sin(a) + \cos(a)(x-a) - \frac{1}{2}\sin(a)(x-a)^2 - \frac{1}{6}\cos(a)(x-a)^3 + \cdots$$

(2.6)

2.2. Interpolation

Interpolation is a process of estimating unknown values from a set of known values provided the unknown values are within the range of known values [2]. There are a variety of interpolation techniques among them the most common are linear interpolation, polynomial interpolation, spline interpolation and piecewise linear interpolation. These interpolation techniques are explained in detail in the following subsections [2].

2.2.1. Linear Interpolation

In this Linear Interpolation method the unknown value is estimated based upon known data points provided the unknown value is in between the two known points [2]. For example, consider two known data points A(x_a, y_a) and B(x_b, y_b). A third unknown data point C(x, y) is to be calculated as shown in Fig. 1. C(x, y) can be calculated for a given value of x and y using equation (2.7) [2].

$$y = y_a + (x - x_a) * \frac{(y_b - y_a)}{(x_b - x_a)} \tag{2.7}$$

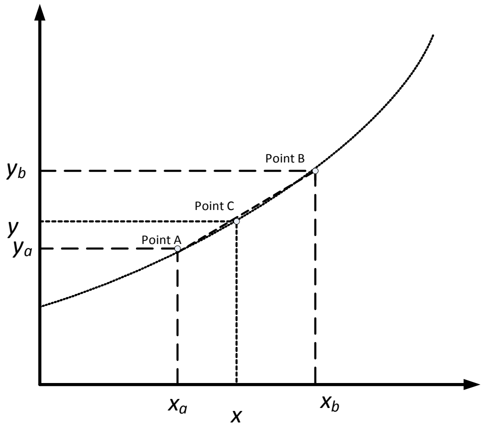

Fig. 1. The illustration of Linear Interpolation Method.

Suppose A(x_a, y_a) and B(x_b, y_b) represents A(*1, 5*) and B(*2, 8*). For x = 1.5 the value of y at point C can be calculated using equation (2.7) as shown as (2.8).

$$y = 5 + (1.5 - 1) * \frac{(8-5)}{(2-1)} \tag{2.8}$$

$$y = 6.5$$

From the equation (2.8), the result of y-coordinate at x = 1.5 is 6.5.

This method is known for its simplicity of all interpolation techniques and noted for its accuracy and speed of retrieving results. This method is most commonly used in computer graphics. In this method, the intermediate intervals are chosen such that the number is a power of 2 such as 2, 4, 8, 16, 32, 64, 128, 256, in a linear interpolation method.

The main advantages of this method are the computation speed of the operation and that it is easy to use. The gradient of difference plays an important role in evaluating the interpolating value.

2.2.2. Polynomial Interpolation

In Polynomial Interpolation method, a set of polynomial functions are used to solve the unknown interpolant value.

Consider a given set of (n+1) data points. Based upon these values there exists an interpolation polynomial function as shown in (2.9).

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + a_{n-3} x^{n-3} + \dots + a_2 x^2 + a_1 x + a_0. \tag{2.9}$$

Upon construction of (n+1) equations based on the given set of data points, in generalized form as shown in (2.10).

$$p(x_i) = y_i \quad \forall \quad i \in \{0,1,2,\dots\dots,n\} \tag{2.10}$$

Upon including all the polynomial equations into (2.10), a system of linear equations is formed. After arranging the set of (n+1) it deduces into three matrices as shown in (2.10).

$$\begin{pmatrix} {x_0}^{n} & {x_0}^{n-1} \cdots & 1 \\ {x_1}^{n} & \ddots & \vdots \\ {x_2}^{n} & {x_2}^{n-1} \cdots & 1 \end{pmatrix} * \begin{pmatrix} a_n \\ a_{n-1} \\ a_{n-2} \\ \cdots \\ \cdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \cdots \\ \cdots \\ y_n \end{pmatrix} \tag{2.10}$$

In the above matrices the left matrix is known as Vandermonde Matrix [2]. Solving the equations, there exists a unique set of solutions for the derived matrices. The solution to the polynomial equation is achieved by solving the Vandermonde matrix.

There are limitations related to the convergence of the polynomial equation. The uniform convergence cannot be guaranteed when the function is indefinitely differentiable. This condition leads to oscillatory form of solution and also complexity can be highly increased in-terms of functions where the order (n) of polynomial is large.

2.2.3. Spline Interpolation

In cases when the roots of polynomial interpolation are oscillatory in nature, then there is a solution for this equation which is known as the spline interpolation method [3]. In this interpolation method the interpolation is subdivided into various intervals or sub functions i.e., different polynomials define the nature of the function between each interval. The nature of the function is based upon the function between each interval.

There exists various forms of spline interpolation methods such as linear method, quadratic method, cubic method; in most cases spline interpolation refers to cubic method. Upon solving the coefficients of the functions leads to solution of the spline interpolation method. In this method it structures a smooth curve which is continuous in first and second derivatives of the function. This method also reduces complexity of computations in comparison with the polynomial interpolation method.

2.2.4. Piecewise Linear interpolation

Piecewise linear interpolation is a broad method of applying a simple linear interpolation to all the continuous intervals in a series of given data points [2].

Piecewise linear interpolation is generally achieved by executing the process of linear interpolation between each interpolation interval. It is one of the quickest and simplest interpolation techniques. In this method the first step is to locate the interpolant value inside the interval, then apply linear interpolation procedure. The appearance of this method resembles a set of straight lines.

Consider a given set of data points (1,20), (2,7), (3,4), (4,9), (5,7), (6,3) after interpolation the desired piecewise linear interpolations is shown in Fig. 2.

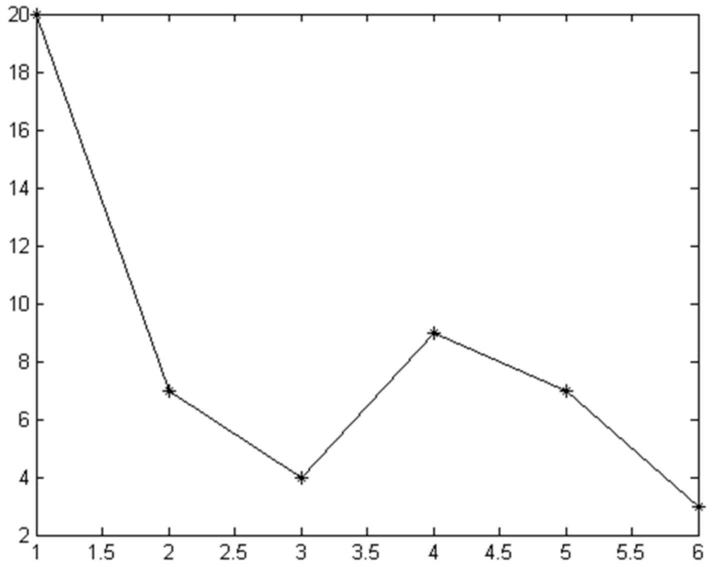

Fig. 2. Illustration of piecewise linear interpolation with given set of data points.

CHAPTER 3

3. Performance Metrics

Performance of hardware architecture can be characterized by its error behavior and its power analysis. In this regard error characterization of any design generally uses five basic metrics. These metrics are maximum absolute error, mean error, median, standard deviation, root mean square error and error probability distribution. In power analysis, static power consumption, short circuited power consumption and dynamic power consumption are considered.

Following sections will define these metrics very briefly.

3.1. Maximum Absolute Error:

The maximum Absolute Error is a metric which defines worst possible error in any design [4]. It is defined as the difference between the approximated value and actual value, as shown in (3.1).

$$\Delta x_i = |\widehat{x_i} - x_i| \tag{3.1}$$

Where, $\widehat{x_i}$ is approximation value,

x_i is actual value,

3.2. Mean Error:

Consider given n terms in a sequence of error approximating function. The Mean Error is defined as the average of the individual error of each specific quantity, as shown in (3.2) [4].

$$\bar{x} = \frac{1}{n} * \sum_{i=1}^{n} (\widehat{x_i} - x_i) \tag{3.2}$$

$\bar{x}$ is mean value

3.3. Median:

Median is defined as the mid value of any sequence of sample errors. In case there is even number of error samples, median will be the average value of the two middle values in the sequence [4].

3.4. Standard deviation:

The Standard Deviation is defined as the average of the square of the individual error term and mean error, as shown in (3.3). In simple words standard deviation represents how much error variations exist from the average value. Low standard deviation means that difference between any given value and the mean value is close to zero [4].

$$\sigma^2 = \frac{1}{n} * \sum_{i=1}^{n} \left[\widehat{x_i} - \bar{x} \right]^2 \tag{3.3}$$

3.5. Root-mean-square (RMS) Error:

The Root Mean Square value is defined as the square root of the average of the sum of the squared terms of individual errors, as shown (3.4).

$$x_{rms} = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (\widehat{x_i} - x_i)^2} \tag{3.4}$$

RMS error is very useful when the error is varying in positive and negative values.

3.6. Error Probability Distribution

Error Probability distribution is a graph interpreting the error diagram to visualize the deviation of absolute error. This diagram simplifies the interpretation of standard deviation and RMS values [4].

The following Fig.3 shows an example of Error probability distribution diagram.

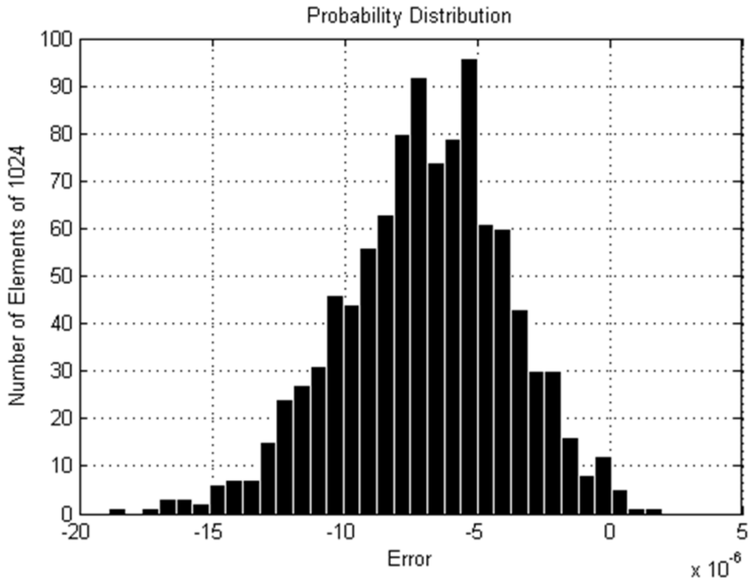

Fig. 3. Error Probability distribution diagram

3.7. Decibel (dB)

In statistical measurement quantities dB scale is mostly used in order to provide greater accuracy, high resolution and easier understanding of the metrics.

The dB-scale is generally used for power and amplitude measurements.

3.7.1. Power measurement

To analyze the *power* or *intensity* measurement of any device, in dB units [4], it is expressed as shown in (3.5)

$$P_{dB} = 10 * log_{10}\left(\frac{measured\ power(P)}{reference\ power(P_0)}\right) \tag{3.5}$$

3.7.2. Amplitude measurement

To analyze the *amplitude* measurement of characteristic such as *voltage* or *current* of any device, it is represented as shown in (3.6) [4].

$$A_{dB} = 20 * log_{10}\left(\frac{measured\ amplitude(A)}{reference\ amplitude(A_0)}\right) \tag{3.6}$$

3.7.3. Error in dB scale

Generally the Signal to Noise Ratio (SNR) is expressed in dB scale rather than linear scale to accommodate large error deviations ranging from tenth (10e-1) to billionth (10e-9) in order. The dB scale gives the freedom to express large variations in short notation. However, the error of dB scale can be further expressed in terms of number of bits as shown in (3.7).

$$Number\ of\ Bits = \frac{Error}{20 * \log_{10}(2)} \approx \frac{Error}{6.02} \tag{3.7}$$

3.8. Power Analysis:

Power analysis is an important design parameter in which the total power consumption of a system is estimated. Enormous effort is invested in optimizing power utilization of ASIC devices. In this project designs are optimized for Low-power to ultra-low power design [6] [7].

In this regard the total power consumption in any circuit is expressed as shown in (3.8):

$$P_{total} = P_{leakage} + P_{short\text{-}circuit} + P_{dynamic} \tag{3.8}$$

Where

$$P_{leakage} = V_{DD} * I_{leak}$$
$$P_{short\text{-}circuit} = I_{sc} T_{sc} V_{DD} * \alpha$$
$$P_{dynamic} = \alpha * C_{total} V_{DD}^2 f_{clk}$$

Where, $P_{leakage}$ is leakage power consumption,

$P_{short\text{-}circuit}$ is short circuit power consumption,

$P_{dynamic}$ is short dynamic power consumption,

V_{DD} is supply voltage,

I_{leak} is leakage current,

I_{sc} is short-circuit current,

T_{sc} is short-circuit duration of the circuit,

f_{clk} frequency of clock

3.8.1. Static Power Consumption:

This is the power consumption of a circuitry during powered-ON state. It is defined as shown in (3.9).

$$P_{static} = V_{DD} * (I_{substrate} + I_{oxide}) \tag{3.9}$$

3.8.2. Short-Circuit Power Consumption:

The short-circuit power consumption can be expressed in (3.10):

$$P_{short-circuit} = \alpha * I_{sc}T_{sc}V_{DD} \tag{3.10}$$

Where I_{sc} is the short circuit current from VDD to ground, T_{sc} is the short circuit duration and α is the switching factor.

3.8.3. Dynamic Power Consumption:

Power consumption of logic circuitry during the transitional phases of a digital design is known as dynamic power consumption. It is expressed in (3.11):

$$P_{dynamic} = \alpha * C_{total}{V_{DD}}^{2}f_{clk} \tag{3.11}$$

Where C_{total} is the total circuit capacitance, f_{clk} is the switching frequency and V_{DD} is the supply voltage.

CHAPTER 4

4. Hardware Implementation

ASIC implementation of a digital design involves a design flow as shown in Fig. 4. It starts with design specifications which are often defined for a project. Then a reference model is created in MATLAB according to design specifications. In the next step a behavioral model is designed in Modelsim and its results are compared to the results of the MATLAB model. The behavioral model is fine-tuned duing functional verification to meet the design specifications. Once the functional requirements of the design are met the design is synthesized on gate level net-list in Synopsis design vision. After this, timing simulations are performed on the synthesized design during post-synthesis verification. In case the design doesn't meet the timing requirements it is fine tuned in the behavioral model again and the process is followed until the timing requirements are met. When the design has acceptable performance it is ready for sign off.

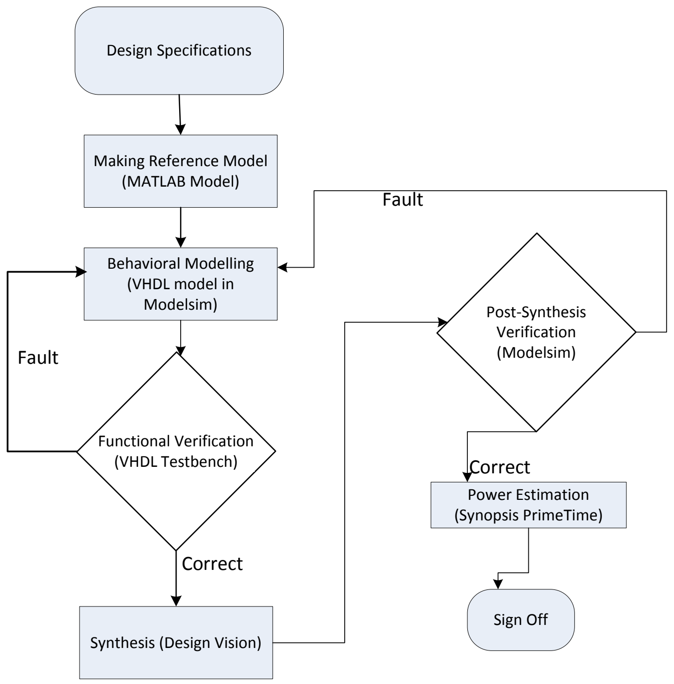
Design Specifications
Making Reference Model (MATLAB Model)
Behavioral Modelling (VHDL model in Modelsim)
Fault
Functional Verification (VHDL Testbench)
Correct
Synthesis (Design Vision)
Fault
Post-Synthesis Verification (Modelsim)
Correct
Power Estimation (Synopsis PrimeTime)
Sign Off

Fig. 4. Flow Chart of ASIC Design.

In this chapter two different techniques have been used to implement the exponential function on hardware level. The first technique uses Taylor series expansion method whereas the second technique uses the linear interpolation method. The theoretical significance of these techniques has been discussed in Chapter 2. In this chapter, different hardware implementations using these techniques will be suggested to find a better compromise between area and speed.

4.1. Taylor Series Architectures

In this section, two different architectures, architecture 1 and 2 are suggested using Taylor series expansion method. Architecture 2 is derived based on the limitations of architecture 1 and has reduced area and power consumption.

4.1.1. Taylor Series Architecture 1

To implement architecture 1 of exponential function, a Taylor series expansion is applied to exponential function e^x, substituting $f(x) = e^x$ in (2.1) reveals;

$$e^x = e^a\Big[1 + (x-a) + \frac{1}{2}(x-a)^2 + \frac{1}{6}(x-a)^3 + \frac{1}{24}(x-a)^4 + \frac{1}{120}(x-a)^5 + \frac{1}{720}(x-a)^6 + \frac{1}{5040}(x-a)^7\Big] \tag{4.1}$$

In (4.1) when factoring (*x*-*a*) out of all terms and rearranging their order we get, as shown in (4.2).

$$e^x = e^a\left[1 + (x-a)\left(1 + (x-a)\left(\frac{1}{2} + (x-a)\left(\frac{1}{6} + (x-a)\left(\frac{1}{24} + (x-a)\left(\frac{1}{120} + \frac{1}{720}(x-a)\right)\right)\right)\right)\right)\right] \tag{4.2}$$

The hardware of architecture1, having '*x'* as input and '*exp*(*x*)' as output, is generated based on equation (4.2) and is shown in Fig. 5. It has 7 multipliers and 7 adders.

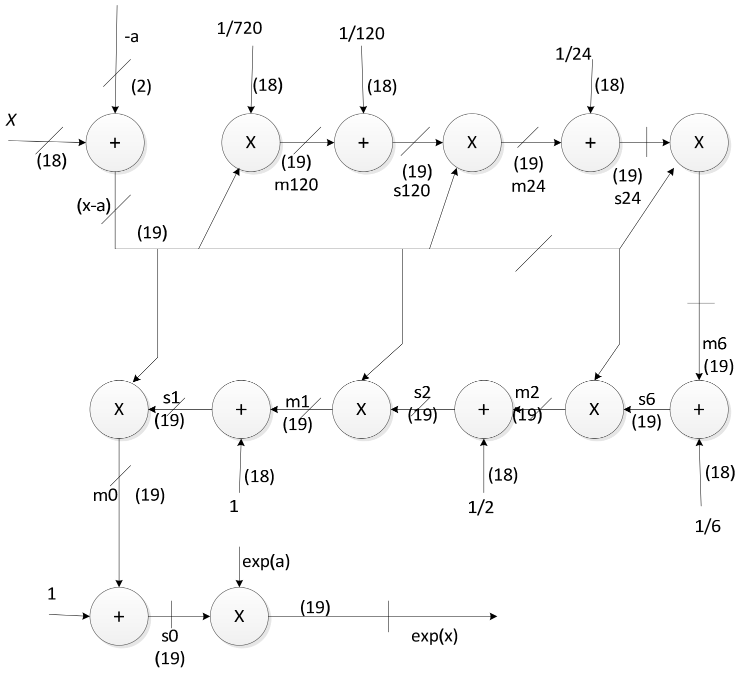

Fig. 5. Hardware implementation of Exponential function using Taylor series architecture 1.

In this architecture, the input *x* has been specified to vary between 0 and 1. After MATLAB modelling of the design the estimated wordlength is 19 bits at each stage, to achieve desired precision. This architecture uses 2's compliment fixed-point number representation for positive and negative values. The wordlength of each fixed coefficient of multipliers and adders is 18 bits. The precision achieved for architecture 1 is 15.72 bits.

4.1.2. Taylor Series Architecture 2

To derive the hardware architecture 2 using Taylor series method consider (4.2).

$$e^x = e^a \left[1 + (x-a)\left(1 \ + (x-a)\left(\frac{1}{2} + (x-a)\left(\frac{1}{6} + (x-a)\left(\frac{1}{24} \ + (x-a)\left(\frac{1}{120} + \frac{1}{720}(x-a)\right)\right)\right)\right)\right)\right] \tag{4.2}$$

Substituting the inner most part of (4.2) with k_1,

$$e^x = e^a \left[1 + (x-a)\left(1 + (x-a)\left(\frac{1}{2} + (x-a)\left(\frac{1}{6} + (x-a)\left(\frac{1}{24} + (x-a)\,k_1\right)\right)\right)\right)\right] \tag{4.3}$$

Where $k_1 = \frac{1}{120} + \frac{1}{720}(x-a) = \frac{1}{120} - \frac{a}{720} + \frac{x}{720}$

Equation (4.3) can be further reduced by substituting the inner-most factor with k_2 as shown in (4.4),

$$e^x = e^a \left[1 + (x-a)\left(1 + (x-a)\left(\frac{1}{2} + (x-a)\left(\frac{1}{6} + (x-a)\left(\frac{1}{24} + k_2\right)\right)\right)\right)\right] \tag{4.4}$$

Where $k_2 = (x-a)\left(\frac{1}{120} - \frac{a}{720} + \frac{x}{720}\right)$

$$= -\frac{a}{120} + \frac{a^2}{720} + \frac{x}{120} - \frac{2ax}{720} + \frac{x^2}{720}$$

Equation (4.4) can be re-arranged to a simpler form as shown (4.5).

$$e^x = e^a (c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 + c_5 x^5) \tag{4.5}$$

Where the constant values (C_i) are expressed in table II as follows:

TABLE II. COEFFICIENTS USED IN TAYLOR ARCHITECTURE 2

Constant	Coefficient	Numerical Value
C0	$1 - a + \frac{a^2}{2} - \frac{a^3}{6} + \frac{a^4}{24} - \frac{a^5}{120} + \frac{a^6}{720}$	0.606532118055556
C1	$1 - \frac{2a}{2} + \frac{3a^2}{6} - \frac{4a^3}{24} + \frac{31a^4}{720} - \frac{6a^5}{720}$	0.606597222222222
C2	$\frac{1}{2} - \frac{3a}{6} + \frac{6a^2}{24} - \frac{64a^3}{720} + \frac{14a^4}{720}$	0.302604166666667
C3	$\frac{1}{6} - \frac{4a}{24} + \frac{66a^2}{720} - \frac{16a^3}{720}$	0.103472222222222
C4	$\frac{1}{24} - \frac{34a}{720} + \frac{9a^2}{720}$	0.021180555555556
C5	$\frac{7}{720} - \frac{2a}{720}$	0.008333333333333

In architecture 2, all the coefficients of (C_i) can be hardwired in implementation. Equation (4.5) can be further reduced as expressed in (4.6),

$$e^x = e^a \left(c_0 + x\left(c_1 + x\left(c_2 + x\left(c_3 + x(c_4 + xc_5)\right)\right)\right)\right) \tag{4.6}$$

Now the hardware of architecture 2, having '*x*' as input and '*exp*(*x*)' as output, is generated based on equation (4.6) and is shown in Fig. 6. It has 6 multipliers and 5 adders.

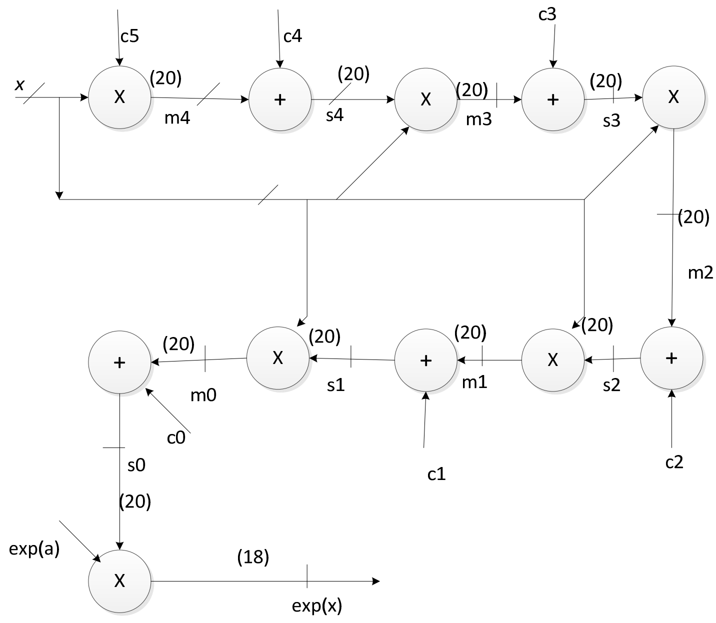

Fig. 6. Hardware implementation of Exponential function using Taylor series Architecture 2.

The input *x* has been specified in ranges from 0 to 1. All the coefficients are positive in Taylor series architecture 2. In this architecture unsigned fixed point number representation is used. Each stage has a wordlength of 20 bits, 1 bit for integer part and 19 bits fractional part. The coefficients are specified in Table II and the word length of each coefficient is 20 bits. The wordlength at the output stage is 18 bits. The precision achieved is 16.33 bits.

Fig. 7 shows the partial synthesized hardware of Taylor series architecture 2 in stm65nm process. This shows the gate level connections of all the components in architecture 2.

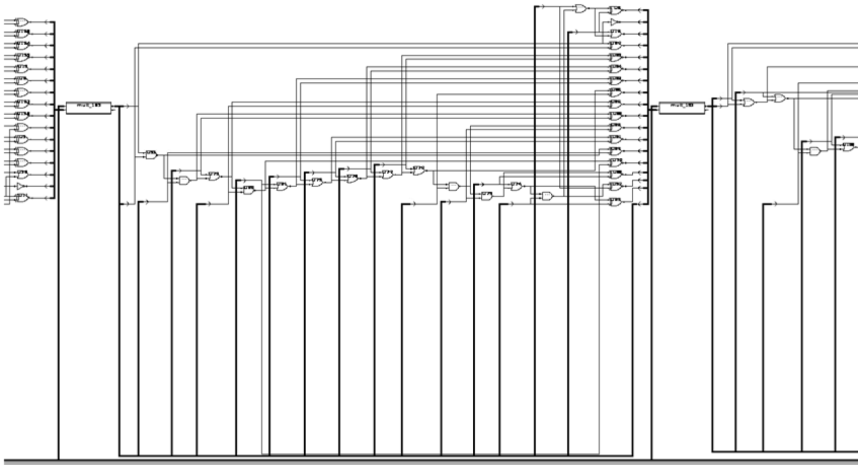

Fig. 7. Synthesized design (partial) of Taylor series architecture 2.

4.1.3. Taylor Series Architecture 2 with pipelined stage

Taylor series architecture 2 has a limitation of working up to a few MHz of frequency range. In order to calculate the dynamic power consumption of Taylor series architecture 2 up to 1 GHz it is pipelined as shown in Fig. 8.

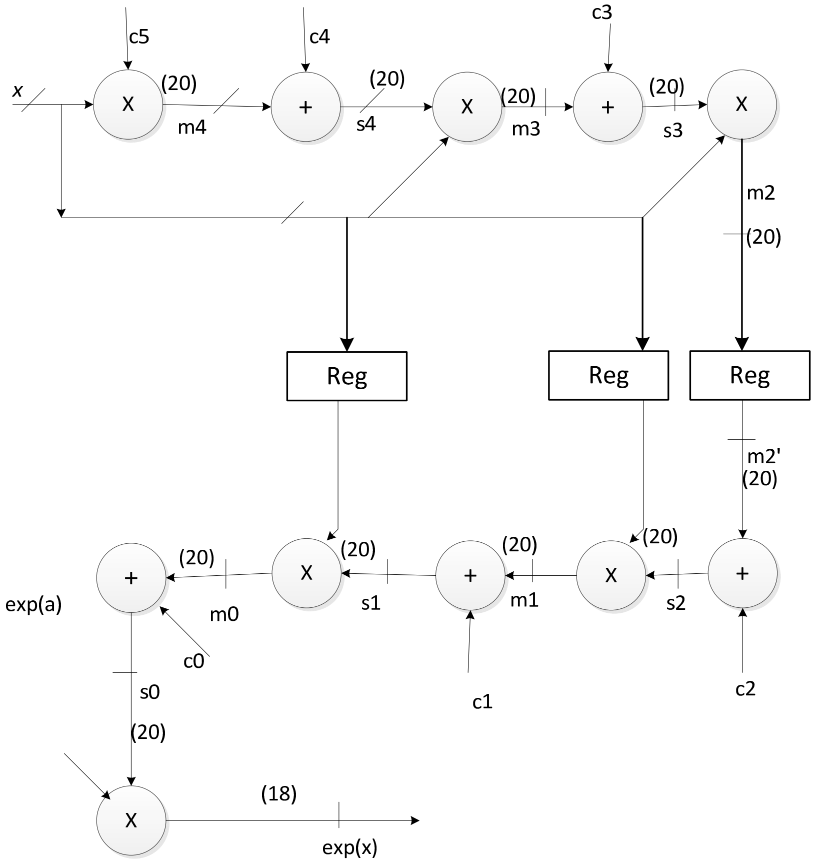

Fig. 8. Taylor Series Architecture 2 with pipelined stage

Fig. 8 shows that a pipeline stage i.e. registers, is inserted in the middle of Taylor series architecture 2 which divides the critical path into half. Taylor series architecture 2 with pipeline stage has increased operating frequency which enables the dynamic power analysis to be performed at frequency of around 1GHz.

4.2. Linear Interpolation Architectures

Besides Taylor series architectures, interpolation based architectures are implemented to compare various performance metrics. In this regard three different architectures are suggested. Implementing Linear Interpolation based architecture needs Look-up Table (LUT) to store the pre-calculated values. To understand the basic concept of using an LUT, for linear interpolation architecture, a simple example is demonstrated below.

4.2.1. Example of a simple LUT:

In this section a simple 8x8 LUT is demonstrated. Firstly an 8x8 LUT is designed in VHDL. The design is synthesized in Synopsis Design Vision to generate the hardware architecture.

The following VHDL code shows an LUT design:

```
 type t_lut_data is array (0 to 7) of
std_logic_vector(7 downto 0);
   constant data1:t_lut_data:= (

                            "01000000",
                            "01000000",
                            "01000001",
                            "01000001",
                            "01000010",
                            "01000010",
                            "01000011",
                            "01000011"
                             );
```

List1: Listing of VHDL code for a single 8x8 LUT

This design is also synthesized at gate level in Design Vision which is shown in Fig. 9.

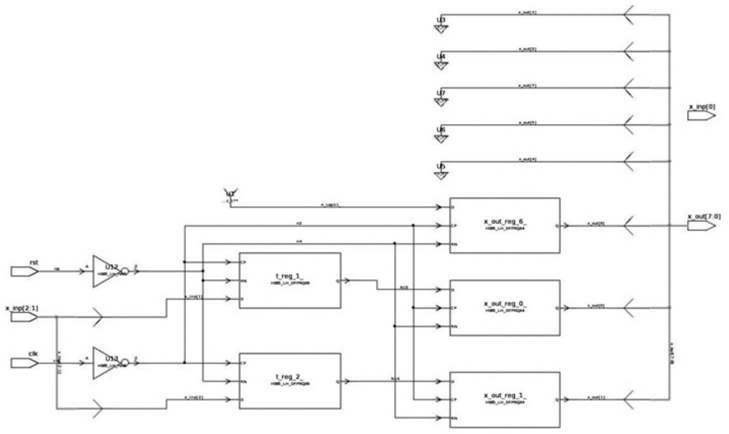

Fig. 9. An 8x8 LUT after synthesis in Design Vision.

The gate-level net-list of the LUT is generated using LPHVT technology library of stm065nm process. The following table III shows the resources (cells used) in the hardware of an 8x8 LUT.

TABLE III. CELLS USED IN HARDWARE OF AN 8X8 LUT.

Cell Attributes	Reference	Library	Area
U8	HS65_LH_IVX9	CORE65LPHVT	1.560
U10	HS65_LH_IVX9	CORE65LPHVT	1.560
t_reg[1]	HS65_LH_DFPRQX4	CORE65LPHVT	10.400
t_reg[2]	HS65_LH_DFPRQX4	CORE65LPHVT	10.400
x_out_reg[0]	HS65_LH_DFPRQX4	CORE65LPHVT	10.400
x_out_reg[1]	HS65_LH_DFPRQX4	CORE65LPHVT	10.400
x_out_reg[6]	HS65_LH_DFPRQX4	CORE65LPHVT	10.400
Total 7cells			55.120

4.2.2. Linear interpolation Architecture with 1 LUT:

In this architecture, the exponential function is implemented using linear Interpolation approach as described in section 4.3. The following mathematical equation (4.7) is a starting point of hardware architecture 1.

$$y = y_b(n) + \left(x(n) - x_b(n)\right) * \frac{(y_b(n+1) - y_b(n))}{(x_b(n+1) - x_b(n))} \tag{4.7}$$

Generally, in-terms of hardware implementation, division is a highly complex function because it demands more resources than any other logical or arithmetic function. In order to avoid direct division, there is more investigation in various techniques, to find a simple logical solution. As there are fixed number of intervals in the architecture which means that the denominator in (4.7) is always a fixed fractional number.

The equation (4.7) is reformed as (4.8) after fixing the denominator as v which is the inverse of difference of intervals.

$$y(n) = y_b(n) + v * \left(x(n) - x_b(n)\right) * \left(y_b(n+1) - y_b(n)\right) \tag{4.8}$$

Substituting $v * \left(x(n) - x_b(n)\right) = f(n)$ into (4.8).

$$y(n) = y_b(n) + f(n) * \left(y_b(n+1) - y_b(n)\right) \tag{4.9}$$

The hardware of interpolation architecture 1 using 1 LUT, with x as input and $y(n)$ as output, is generated based on equation (4.9). The architecture is shown in Fig. 10.

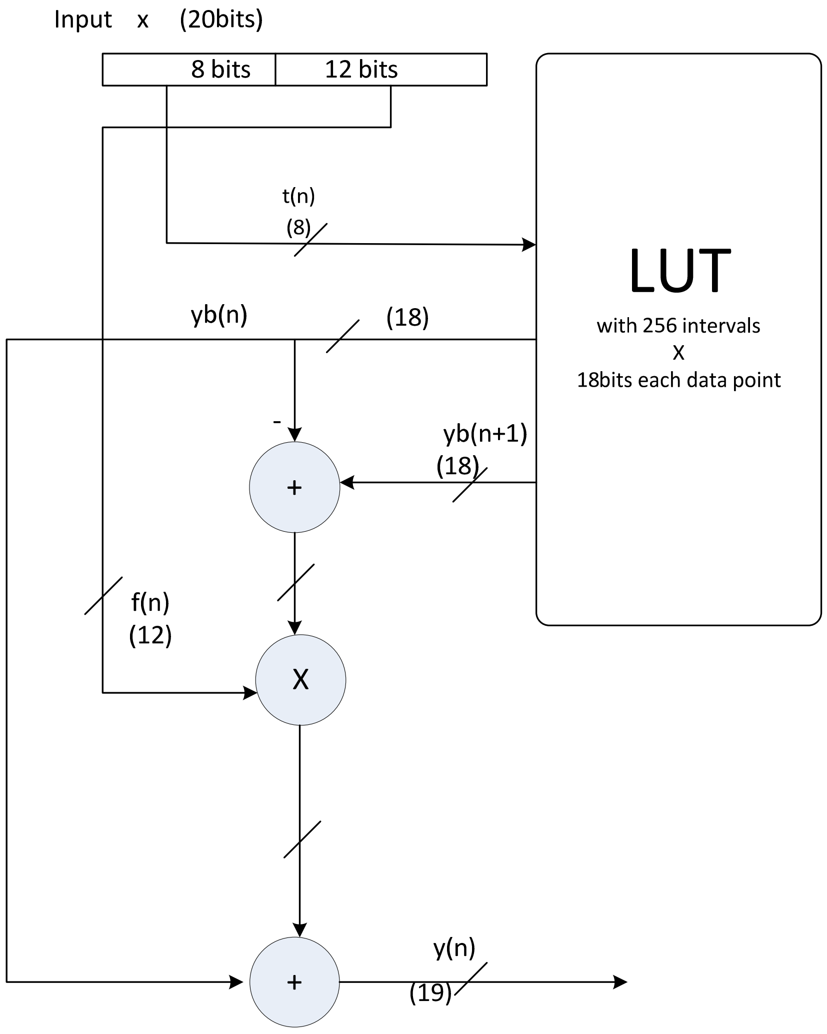

Fig. 10. Hardware implementation of exponential function using Linear Interpolation with 1 LUT.

In this architecture, a single LUT is used to store the 256 intervals of the exponential function. Each value has a precision of 18 bits. The values of intermediate intervals are generated using MATLAB. Both the base value and gradient difference are deduced from an LUT.

The designs are meant to verify 1024 intervals between 0 and 1.

From the input 'x' is of word length 20 bits, 8 Most Significant Bit (MSB) bits are used for fetching the base value and next base value. The next base value is fetched out to evaluate the gradient difference which is multiplied with the 12 Least Significant Bit (LSB) bits of input 'x'. The base value and gradient difference are evaluated using a single LUT as shown in Fig. 5. The hardware consists of an LUT, two adders and one multiplier. The precision at the output is 17.73 bits.

4.2.3. Linear interpolation Architecture with 2 LUTs with 256 Intervals:

After discussing the interpolation architecture with one LUT, a new architecture was proposed by thesis supervisor wherein the aim is to reduce the latency of the architecture. Upon investigation of the solution it is decided to use two LUTs in order to store base value and gradient difference separately.

To implement Linear Interpolation architecture with 2 LUTs consider (4.9).

$$y(n) = y_b(n) + f(n) * \left(y_b(n+1) - y_b(n)\right)$$

- ➢ In the first LUT base value is stored. Each value has a precision of 18 bits.
- ➢ In the second LUT gradient difference is stored. Each value has a precision of 12 bits.

The hardware architecture generated using this scheme is shown in the Fig. 11. It has two LUTs, one multiplier and one adder.

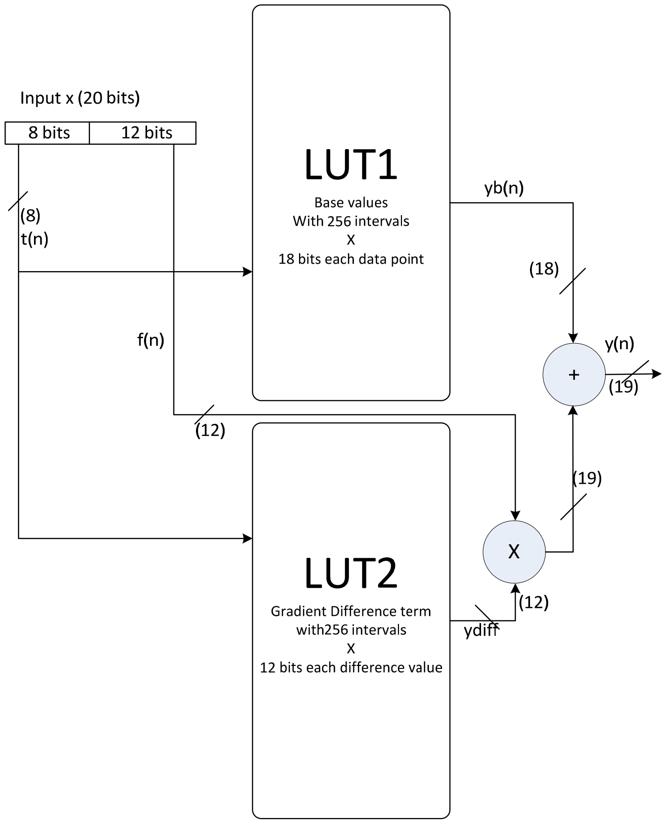

Fig. 11. Hardware implementation of exponential function using Linear Interpolation method with 2 LUTs and 256 intervals.

The input *x* has a word length of 20 bits. The first 8 MSB bits are used to address the base value and gradient coefficient from LUT1 and LUT2 respectively. The remaining 12 bits of LSB of *x* are used to multiply with the gradient coefficient to form the gradient difference which has a word length of 19 bits. The base value is added with the gradient difference to form the final result. The word length of the output is 19 bits. In this architecture, latency is reduced to large extent, however memory size is increased. The precision achieved is 17.27 bits, which is less than the precision of architecture 1 of 17.73 bits.

4.2.4. Linear interpolation Architecture with 2 LUTs with 128 intervals:

The precision achieved in the previous two architectures is more than 17 bits however the requirement of this work is 15 bits.

.

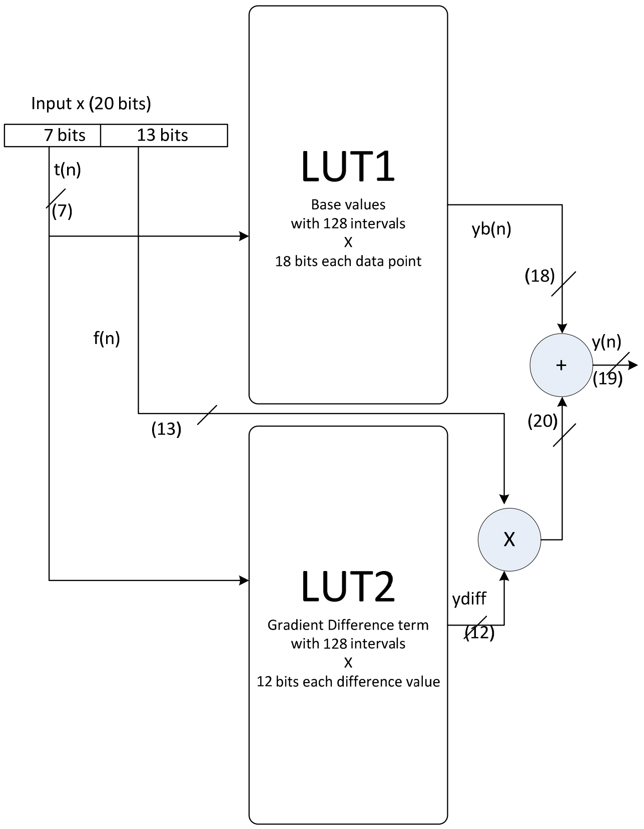

Fig. 12. Hardware implementation of exponential function using Linear Interpolation with 2 LUTs using 128 intervals.

In order to reduce the size of the memory used, the number of intervals in this architecture is reduced to 128, compared to 256 in architecture 2. Both the base value and gradient difference are calculated using two separate LUTs with 128 intervals in each LUT. The hardware of this method is shown in Fig. 12.

In this architecture two LUTs are used, each with the size of 128 intervals.

- In LUT1 base value is stored. Each value has word length of 18 bits.
- In LUT2 gradient difference coefficients are stored. Each value has word length of 12 bits.

The input *x* has a word length of 20 bits. First 7 bits MSB are used to address the base value and gradient difference coefficient in LUT1 and LUT2 respectively. The gradient difference coefficient is multiplied with the 13 bits LSB of input *x*. The wordlength of gradient difference is 20 bits. The base value and gradient difference are added to form the final result. The output wordlength is 19 bits consisting 2 bits of integer part and 17 bits of mantissa part. The hardware resources used in this architecture is 2 LUTs, one multiplier and one adder. The precision achieved using architecture 3 is 15.72 bits which is better than architecture 1 and architecture 2. Hence architecture 3 is a better compromise between size of memory and precision.

CHAPTER 5

5. Estimation of Error

In this chapter error estimation metrics, which measure the accuracy of a system, are summarized for the architectures designed in chapter 4. The error results discussed here are after post synthesis simulations of the architectures using Design Vision.

5.1. Taylor series Architecture 1:

The upper half of Fig. 12 shows the error difference in linear scale. Whereas, the equivalent number of bits of the Taylor series architecture 1 with respect to real function are shown in the lower half of Fig. 13.

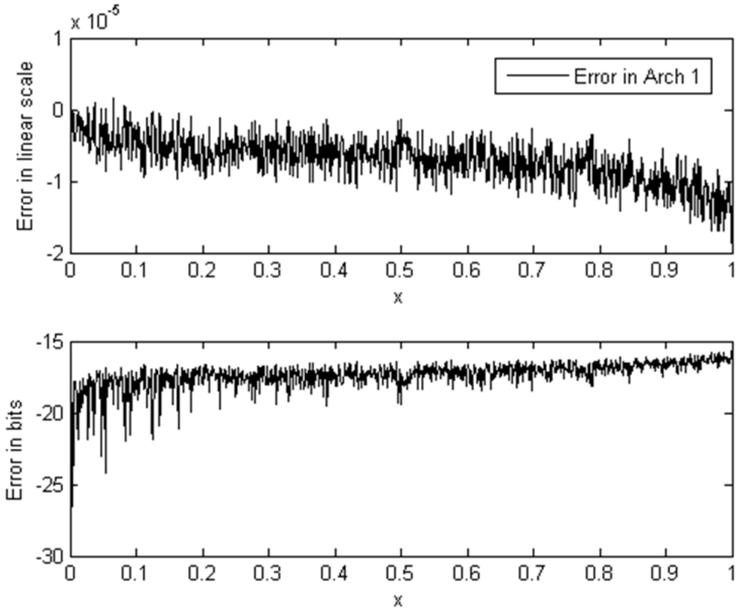

Fig. 13. Error difference of Taylor series architecture 1 in linear scale and total number of bits.

The details of error metrics are shown in Table IV.

TABLE IV. ERROR METRICS OF TAYLOR ARCHITECTURE 1.

Metric	**Quantity in Linear scale**	**Equivalent bits**
Max Error	0.00000165592	19.20
Min Error	-0.00001857604	15.71
Mean Error	-0.00000697915	17.13
Median Error	-0.00000682426	
Standard Deviation	0.00000314758	
RMS Error	0.00000765546	

System has mostly negative errors and the interesting thing is that the worst case error is equivalent to 15.71 which satisfies the system requirements.

Fig. 14 shows the histogram of probability error distribution of Taylor series architecture 1. It can also be noticed in the figure that the errors terms are negative in this architecture.

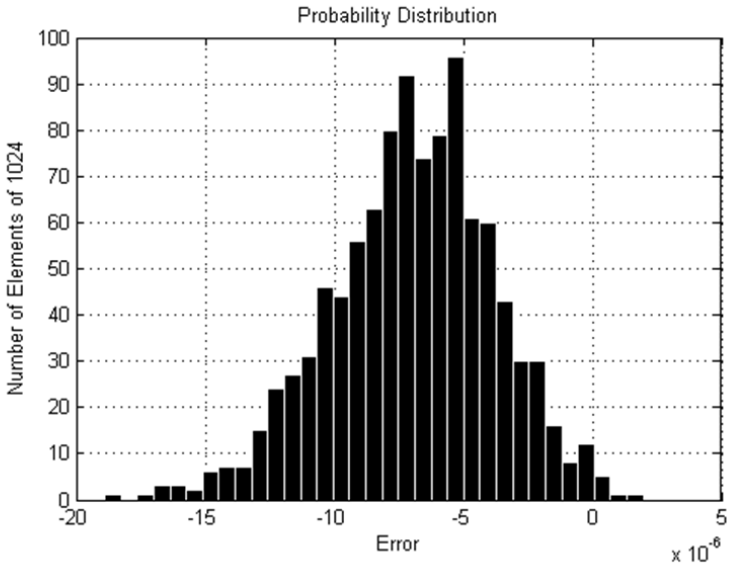

Fig. 14. Histogram of probability error distribution of Taylor Architecture 1.

5.2. Taylor series Architecture 2:

The upper half of Fig. 15 shows the error difference in linear scale. Whereas, the equivalent number of bits of the Taylor series architecture 2 with respect to real function are shown in the lower half of Fig. 15.

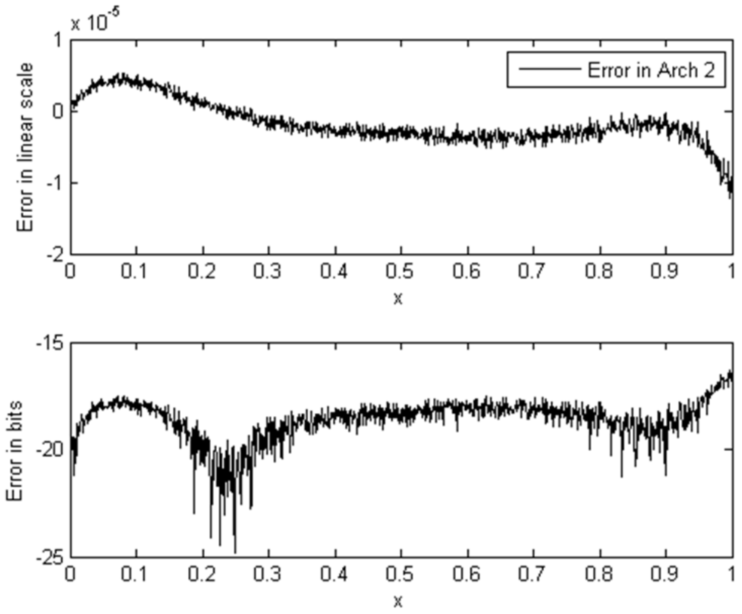

Fig. 15. Error difference of Taylor series architecture 2 in linear scale and total number of bits.

TABLE V. ERROR METRICS OF TAYLOR ARCHITECTURE 2

Metric	**Quantity in Linear scale**	**Equivalent bits**
Max Error	0.00000530197	17.53
Min Error	-0.00001213279	16.33
Mean Error	-0.00000169389	19.17
Median Error	-0.00000247776	
Standard Deviation	0.00000287994	
RMS Error	0.00000333994	

Table V shows that the equivalent bits for the worst case error is 16.33 which is better than Taylor's series architecture 1. Also there is very small difference between standard deviation and RMS error which shows the distribution to be even.

Fig. 16 illustrates the probability error distribution of Taylor series architecture 2. The distribution is on both sides of zero, with more peaks in the negative side than on the positive side.

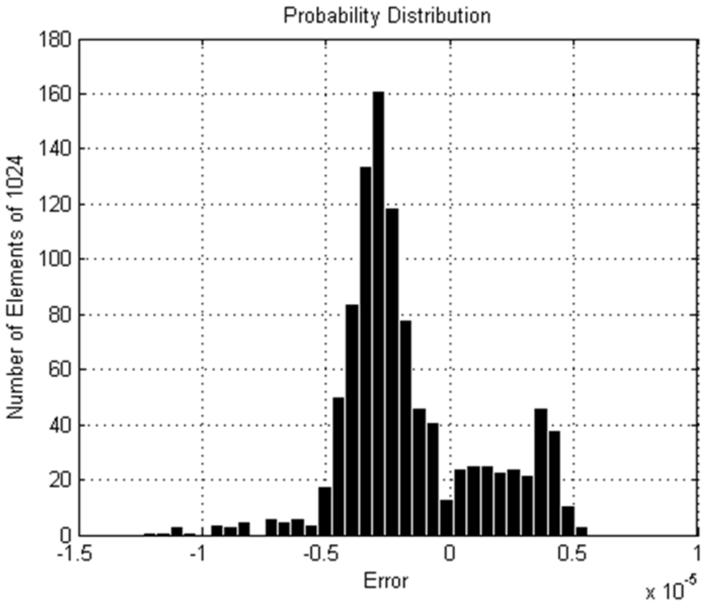

Fig. 16. Histogram of probability error distribution of Taylor Architecture 2 for exponential function implementation.

5.3. Linear Interpolation architecture with 1 LUT:

The upper half of Fig. 17 shows the error difference in linear scale. Whereas, the equivalent number of bits of the Linear Interpolation architecture 1 with respect to real function are shown in the lower half of Fig. 17.

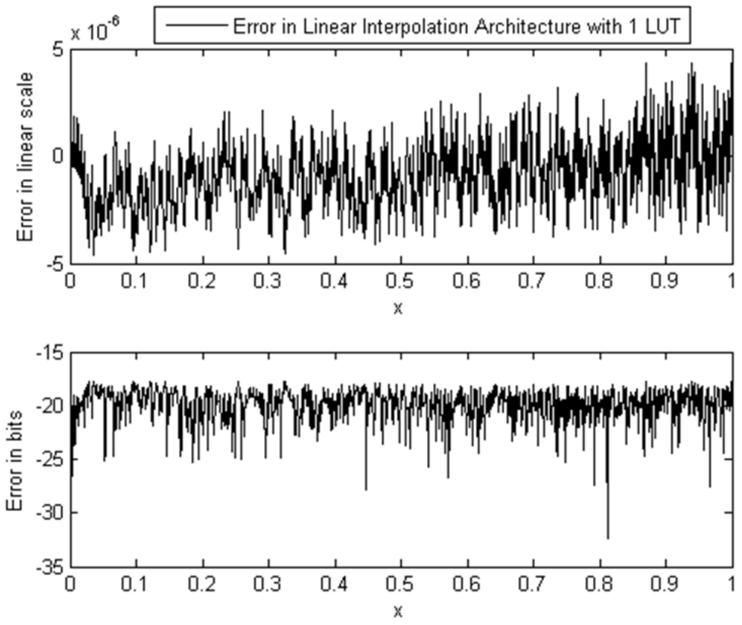

Fig. 17. Error difference of linear interpolation architecture (using 1 LUT) in linear scale and total number of bits.

TABLE VI. ERROR METRICS OF LINEAR INTERPOLATION METHOD USING 1 LUT

Metric	**Quantity in Linear scale**	**Equivalent bits**
Max Error	0.00000432	17.82
Min Error	-0.00000459	17.73
Mean Error	-0.00000085	20.16
Median Error	-0.00000091	
Standard Deviation	0.00000164	
RMS Error	0.00000184	

Table VI shows that the equivalent bits for the worst case error is 17.73 which is better than Taylor's series architectures. It can also be noticed that due to very small difference between standard deviation and RMS error, the distribution is more or less symmetric.

But from Fig. 18 of probability error distribution the peaks are more in the negative side compared to positive.

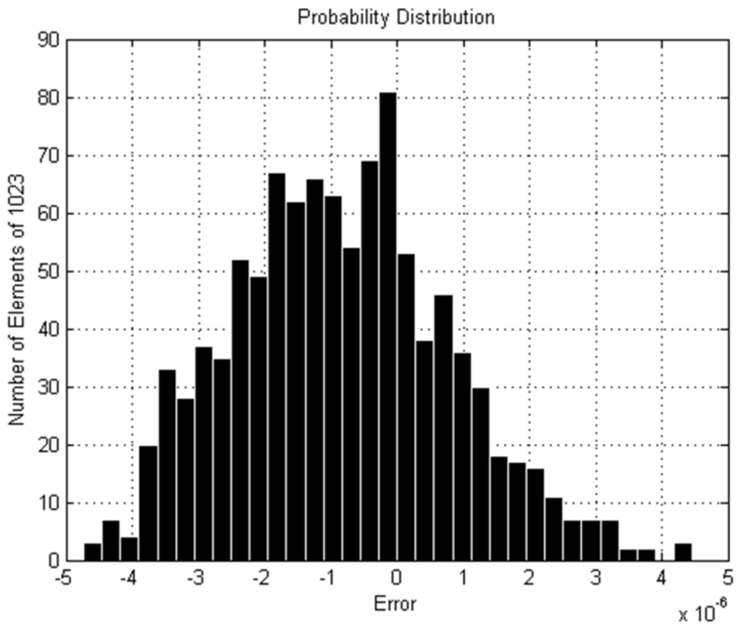

Fig. 18. Histogram showing error distribution of interpolation method for exponential function implementation using 1 LUT.

5.4. Linear Interpolation with 2 LUTs and 256 intervals:

The upper half of Fig. 19 shows the error difference in linear scale. Whereas, the equivalent number of bits of the Linear Interpolation with 2 LUTs and 256 intervals with respect to real function are shown in the lower half of Fig. 19.

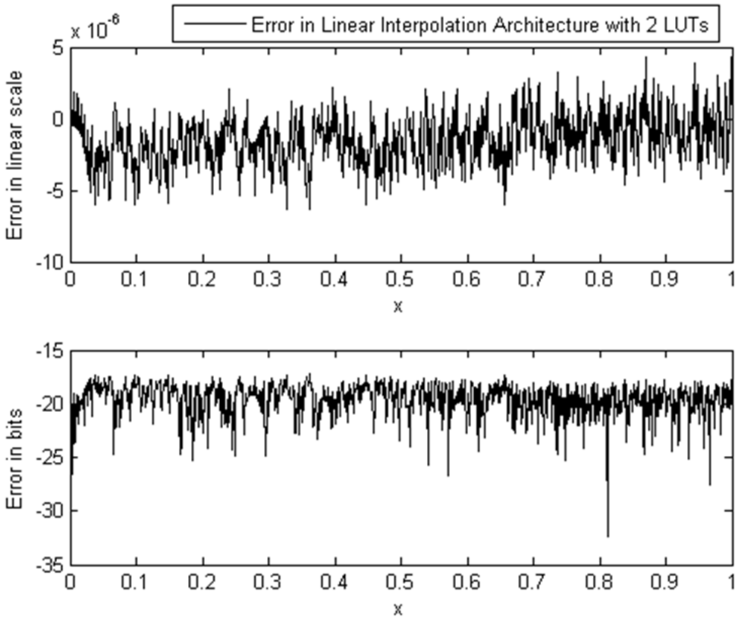

Fig. 19. Error difference of linear interpolation architecture (with 2 LUTs and 256 intervals) in linear scale and total number of bits.

TABLE VII. ERROR METRICS OF LINEAR INTERPOLATION METHOD USING 2 LUTS (256 INTERVALS)

Metric	**Quantity in Linear scale**	**Equivalent number of bits**
Max Error	0.000004323	17.82
Min Error	-0.000006351	17.26
Mean Error	-0.000001466	19.38
Median Error	-0.000001443	
Standard Deviation	0.000001775	
RMS Error	0.000002302	

Table VII shows that the equivalent bits for the worst case error is 17.26 which is better than Taylor's series architectures. The mean error is located in the negative side.

The following histogram in Fig. 20 illustrates the probability error distribution of Linear Interpolation method with two LUTs (using 256 intervals). The error distributions mean value is negative and hence the error peaks are also negative.

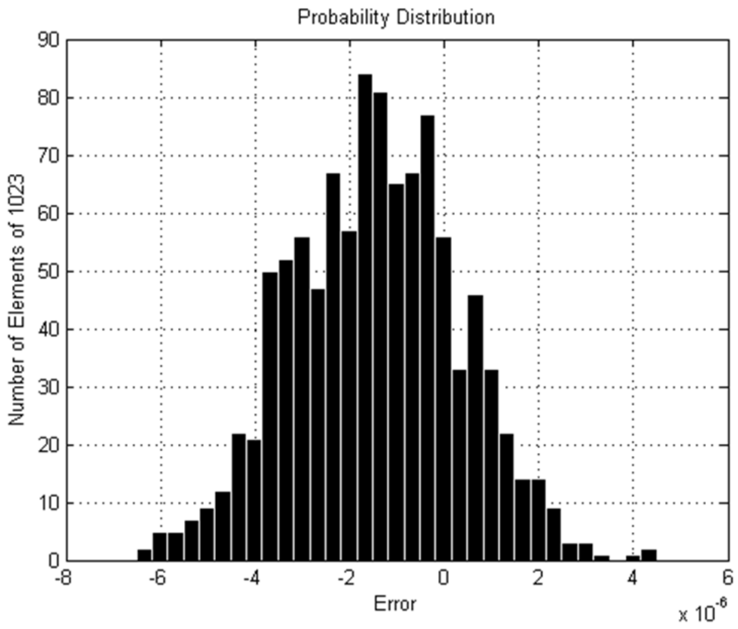

Fig. 20. Histogram of probability error distribution of interpolation method with 2 LUTs (using 256 intervals).

5.5. Linear Interpolation with 2 LUTs and 128 intervals:

The upper half of Fig. 21 shows the error difference in linear scale. Whereas, the equivalent number of bits of the Linear Interpolation with 2 LUTs and 128 intervals with respect to real function are shown in the lower half of Fig. 21.

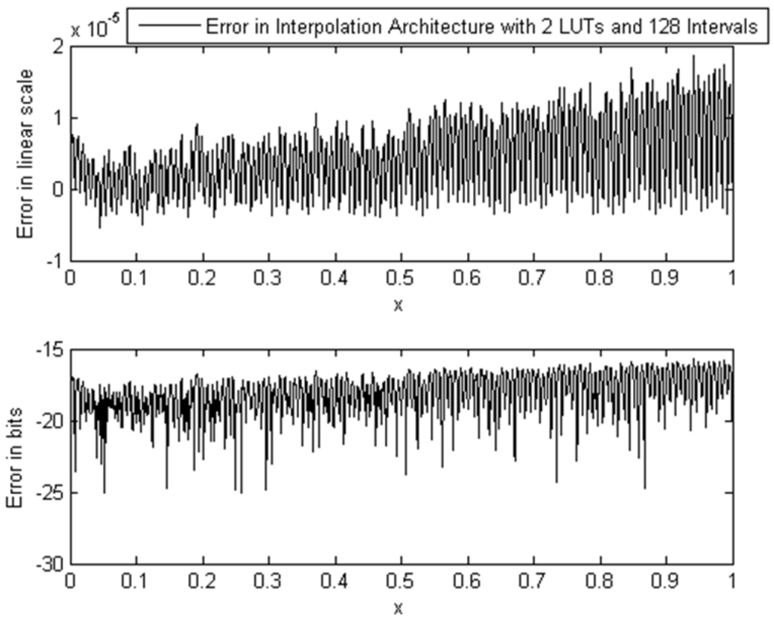

Fig. 21. Error difference of linear interpolation architecture (with 2 LUTs and 128 intervals) in linear scale and total number of bits.

TABLE VIII. ERROR METRICS OF LINEAR INTERPOLATION METHOD USING 2 LUTS USING 128 INTERVALS

Metric	**Quantity in Linear scale**	**Equivalent number of bits**
Max Error	0.000018522	17.52
Min Error	-0.000005334	15.72
Mean Error	0.000004556	17.74
Median Error	0.000004195	
Standard Deviation	0.000004827	
RMS Error	0.000006636	

Table VIII shows that the equivalent bits for the worst case error is 15.72 which satisfies the system specifications. The mean error is located in the positive side.

Fig. 22 shows the probability error distribution which has more peaks on the positive side compared to the negative side of the distribution.

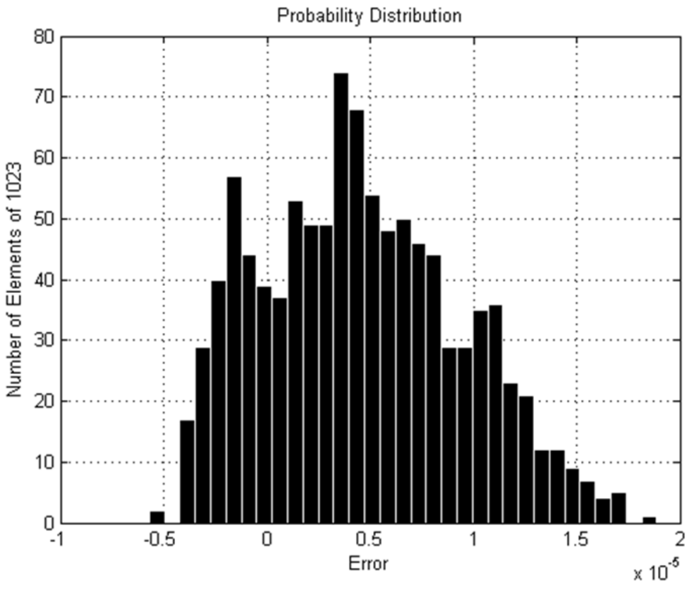

Fig. 22. Histogram showing probability error distribution of linear interpolation architecture with 2 LUTs and 128 intervals.

CHAPTER 6

6. Results

In this chapter area, timing and power results of architectures based on Taylor series and Linear Interpolation methods are presented and discussed. In this regard Design vision tool is used for area, timing and power results. Apart from Design Vision, Primetime is used for more accurate power results. These results are described in the following sections.

6.1. Area

Comparison of area between Taylor series architectures and Interpolations based architectures can be summarized as follows;

- Taylor series hardware architecture 1 consists of 7 multipliers and 7 adders.
- Taylor series hardware architecture 2 consists of 6 multipliers and 5 adders.
- Interpolation architecture with 1 LUT and 256 intervals has 1 LUT, 1 multiplier and 2 adders.
- Interpolation architecture with 2 LUTs and 256 intervals has 2 LUTs, 1 multiplier and 1 adder.
- Interpolation architecture with 2 LUTs and 128 intervals has 2 LUTs, 1 multiplier and 1 adder.

In the Tables IX and X, the area occupied by each one of architecture is presented using the technology libraries Low Power High Threshold Voltage (LPHVT) and Low Power Low Threshold Voltage (LPLVT) respectively, at a supply voltage of 1.2V at an operating frequency of 10MHz.

TABLE IX. AREA REPORT USING LIBRARY LPHVT USING 1.2V

Design	**Minimum Area Occupied (μm^2)**	**Cells**
Taylor Arch 1	22386	4949
Taylor Arch 2	20692	4531
1 LUT	6010	1566
2 LUTs	4568	1297
2 LUTs (128 intervals)	4192	1157

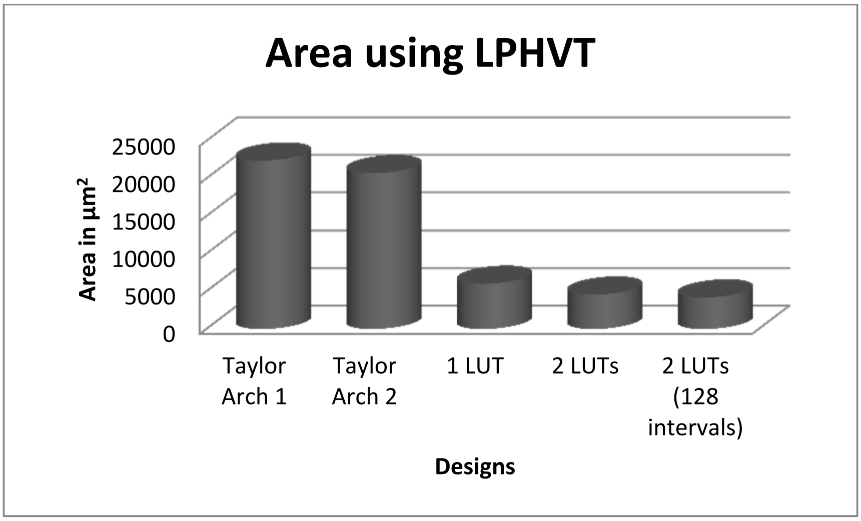

Fig. 23. Area comparison using LPHVT library

TABLE X. AREA REPORT USING LIBRARY LPLVT USING 1.2V

Design	**Minimum Area Occupied (μm^2)**	**Cells**
Taylor Arch 1	22169	4917
Taylor Arch 2	20652	4543
1 LUT	6095	1588
2 LUTs	4622	1335
2 LUTs (128 intervals)	4222	1161

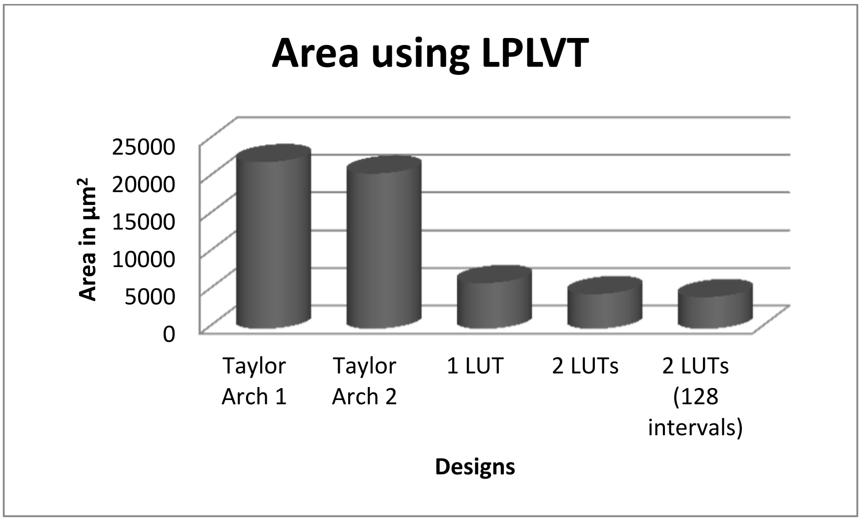

Fig. 24. Area comparison using LPLVT library

The above Fig. 23 and Fig. 24 represent the area comparison graphs of all the designs using technology libraries LPHVT and LPLVT respectively.

In the table XI and XII, it can be noticed that the size occupied by the architecture using 1 LUT is large compared to the architecture using 2 LUTs. It is due to fact that during synthesis stage flatten process is duplicating one more LUT with 18 bits word length. It can be concluded that Taylor series architecture has approximately 4 times more area than that of linear interpolation architecture.

For further information about hardware resources of Taylor series architecture 2 and Linear Interpolation, Table XI and Table XII are presented. The operating frequency is 10MHz and supply voltage of 1.2V using technology library LPHVT.

TABLE XI. RESOURCES USED FOR TAYLOR ARCHITECTURE 2 USING TECHNOLOGY LIBRARY LPHVT AT 1.2V.

Number of ports:	46
Number of nets:	4554
Number of cells:	4531
Number of combinational cells:	4508
Number of sequential cells:	23
Number of buf/inv:	500

TABLE XII. RESOURCES USED FOR LINEAR INTERPOLATION ARCHITECTURE WITH 2 LUTS AND 256 INTERVALS USING TECHNOLOGY LIBRARY LPHVT AT 1.2V.

Number of ports:	42
Number of nets:	1328
Number of cells:	1297
Number of combinational cells:	1277
Number of sequential cells:	20
Number of buf/inv:	208

6.2. Power

Low power consumption is an important design criterion in a hardware design. The power results presented in this section are approximated using Design Vision.

The power consumption of a design consist of total dynamic power, cell internal power, and cell leakage power. Table XIII shows the approximate power consumption of all designs using technology library LPHVT.

TABLE XIII. POWER REPORT USING LIBRARY LPHVT AT 10MHZ

Design with	**Total power (*uW*)**	**Cell internal power (*uW*)**	**Cell leakage power (*nW*)**
Taylor 1	117.6216	59.7387	75.4156
Taylor 2	115.9396	57.6809	66.4293
1LUT	20.5189	11.0545	22.6205
2LUTs	16.3632	9.0386	16.8428
2LUTs (128 intervals)	15.7136	8.7496	15.4545

Table XIV shows the power consumption results using technology library LPLVT.

TABLE XIV. POWER REPORT USING LIBRARY LPLVT AT 10MHZ

Design with	**Total power (uW)**	**Cell internal power (uW)**	**Cell leakage power (uW)**
Taylor 1	144.0615	75.9570	17.1698
Taylor 2	144.5094	75.0068	15.8027
1LUT	24.6240	13.3810	4.7604
2LUTs	18.4378	10.1787	3.3317
2LUTs (128 intervals)	17.7640	9.8814	2.8861

It can be deduced from Table XIII and XIV that Taylor series architecture has at-least six times more total dynamic power consumption than the linear interpolation architecture.

6.3. Timing

Timing is one of the key parameters in analyzing the performance of a design. To estimate the maximum speed of a circuit, the critical path is crucial aspect of a digital hardware. Critical path decides maximum speed of a design. Tables XV shown the critical path of all the designs using LPHVT technology library. Also Table XVI shows the critical path of all the designs using LPLVT technology library.

TABLE XV. TIMING REPORT USING LIBRARY LPHVT

Design	**Critical path (ns)**
Taylor Arch1	49.77
Taylor Arch 2	40.32
Interpolation with 1 LUT	9. 77
Interpolation with 2 LUT	7.23
Reduced LUT (128 intervals)	7.23

TABLE XVI. TIMING REPORT USING LIBRARY LPLVT

Design	**Critical path (*ns*)**
Taylor Arch1	34.83
Taylor Arch 2	24.84
Interpolation with 1 LUT	7.17
Interpolation with 2 LUT	5.17
Reduced LUT (128 intervals)	5.17

From the above tables it can be concluded that, interpolation method is approximately 5 times faster compared to Taylor series method. However, interpolation method is a memory based design. As the number of intervals are increasing, the area and complexity increases.

6.4. Power estimation of Taylor series architecture 2 using PrimeTime:

In this section, power estimations using Synopsis PrimeTime is utilized. A logarithmic scale of frequency vs power is estimated in PrimeTime for Taylor series architecture 2:

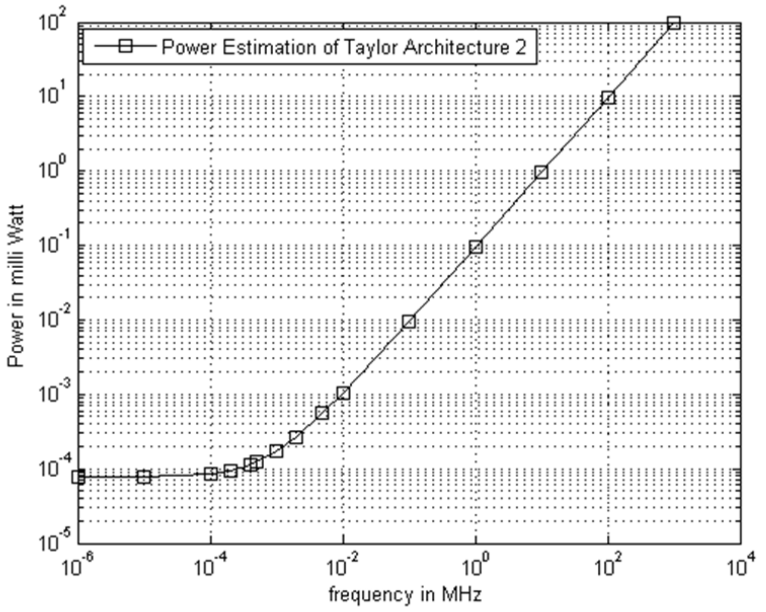

Fig. 25. A log-log graph of frequency versus power calculation for Taylor series architecture 2.

Fig. 25 shows logarithmic plot of frequency vs. power. The power calculations are estimated for cell's internal power and leakage power. At low frequencies such as near DC level the cell leakage power is of significant value compared to its internal power. At higher frequencies the leakage power is constant but the cell internal power increases exponentially. Using same set of constraints the power consumption at low frequencies is only that of static power consumption. However with the increase in frequency the dynamic power consumption comes into effect. Dynamic power consumption exponentially increases at higher frequency ranges.

6.5. Power Estimation for Linear Interpolation with 2 LUTs using PrimeTime:

Fig. 26 shows the power results of linear interpolation architecture 2 (with 2 LUTs and 256 intervals) in PrimeTime.

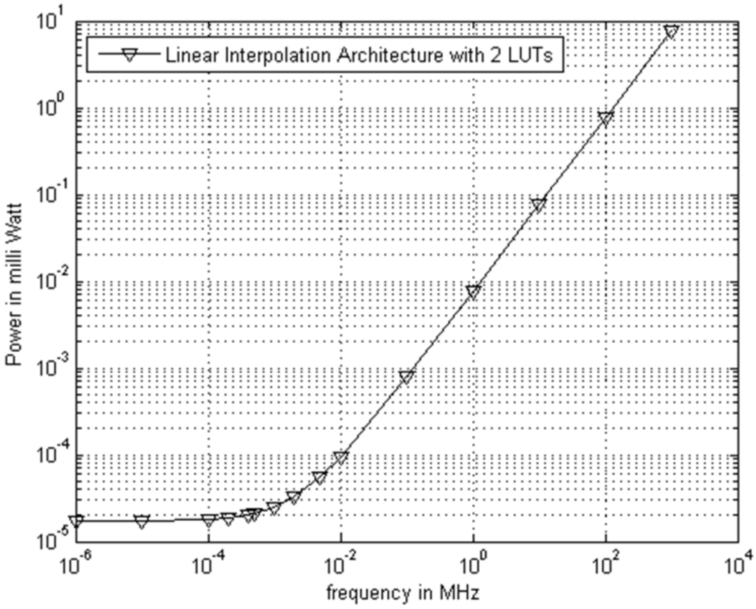

Fig. 26. A log-log graph of frequency versus power calculation for a linear interpolation architecture with 2 LUTs and 256 intervals.

In Fig. 26 a logarithmic plot of power consumption using linear interpolation architecture with 2 LUTs and 256 intervals is plotted. In this plot the power consumption for frequencies below 1 KHz is static. At higher frequencies the dynamic power consumption increases at a rate of 10 times per 10 fold increase in frequency.

6.6. Comparison of Taylor series Approach and Linear Interpolation Approach

To compare power consumption between interpolation and Taylor series method, a combined graph is plotted as shown Fig. 27.

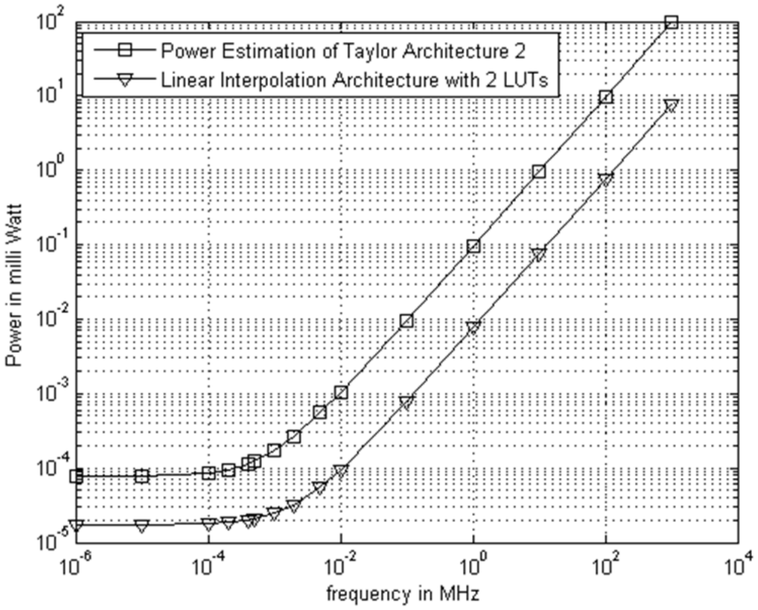

Fig. 27. Comparison of power estimation between Taylor series methodology and linear interpolation approach.

From the Fig. 27 it can be noticed that the power consumption of Taylor series architecture method is approximately ten times that of linear interpolation method, at any given frequency. Table XVII shows the dynamic power consumption results of Taylor series architecture 2 and linear interpolation method.

TABLE XVII. TABLE OF COMPARISON OF POWER CONSUMPTION OF TWO ARCHITECTURES USING PRIMETIME.

Frequency (MHz)	**Total power consumption in Taylor architecture 2 (*W*)**	**Total power consumption Interpolation architecture with 2 LUTs (*W*)**
0.000001	7.6885e-08	1.7047e-08
0.00001	7.7749e-08	1.7116e-08
0.0001	8.6383e-08	1.7809e-08
0.0002	9.5980e-08	1.8579e-08
0.0004	1.1516e-07	2.0118e-08
0.0005	1.2475e-07	2.0888e-08
0.001	1.7272e-07	2.4736e-08
0.002	2.6869e-07	3.2430e-08
0.005	5.563e-07	5.5520e-08
0.01	1.0361e-06	9.3999e-08
0.1	9.6698e-06	7.8664e-07
1	9.6007e-05	7.7130e-06
10	9.5938e-04	7.6977e-05
100	0.0096	7.6961e-04
1000	0.0959	0.0076

CHAPTER 7

7. Conclusions

Hardware implementation of exponential function had been successfully implemented using two different methods. In the first approach two architectures were implemented using Taylor series expansions;

- One with general expansion of Taylor series application.
- Second is a novel architecture with reduced number of multipliers and adders.

The second one is a Linear Interpolation method which has three different approaches;

- Architecture with one LUT and 256 intervals.
- Architecture with two LUTs and 256 intervals.
- The third architecture uses two LUTs and 128 intervals.

The emphasis is laid on minimum area, low-power design and with desired performance. The main aim of the thesis is to achieve precision of 15 bits however the designed methods are able to achieve more than 17 bits of accuracy in various approaches and its error behavior is studied. All the designs are successfully synthesized in stm65nm for minimum area, low-power and possible maximum speed. The dynamic power is recorded using Synopsis PrimeTime. Comparison of novel methodology of Taylor series is compared with the Linear Interpolation techniques, conclusions are drawn in terms of area, speed and power.

It can be concluded from this work that the Taylor series method is a less memory intensive design compared to the Linear Interpolation method. However the Taylor series architecture consumes an area five times that of Interpolation approach. The dynamic power consumption is ten times more in the Taylor series architecture than the linear interpolation architecture. The Linear Interpolation method is at-least five times faster than that of the Taylor series implementation.

CHAPTER 8

8. Future work

By using the novel methodology of Taylor series approximation, there is scope of implementing various other unary functions such as logarithmic function or trigonometric function. The LUT approach can also be substituted by other highly advanced approach like improved parabolic synthesis technique to increase the efficiency of the algorithm.

REFERENCES

[1] http://mathworld.wolfram.com/TaylorSeries.html

[2] T. W. Roberts, "*Non-oscillatory interpolation for the Semi-Lagrangian scheme*", Dissertation.

[3] G. Muntingh, "*Topics In Polynomial Interpolation Theory,*" Dissertation presented for the degree of Ph D.

[4] Erik Hertz, "*Article on Error Evaluation*", EIT, Lund, March 18, 2013.

[5] J Lai, *"Hardware Implementation of the Logarithm Function"*, Masters thesis, Lund University, Sep 2013.

[6] D. M. Lurascu, A. V. Bofill, *"Hardware Approximation of the Square Root Function"*, Masters Thesis, Lunds Universitet, Jan 2014.

[7] E. Hertz, "*Parabolic Synthesis*," Thesis for the degree of Licentiate in Engineering.

[8] www.math.smith.edu/Local/cicintro/ch10.pdf

[9] http://www.haverford.edu/physics/MathAppendices/Taylor_Series.pdf

[10] http://math.stackexchange.com/questions/218421/what-are-the-practical-applications-of-the-taylor-series.

List of Figures

Fig. 1. The illustration of Linear Interpolation Method................ 15
Fig. 2. Illustration of piecewise linear interpolation with given set of data points. 18
Fig. 3. Error Probability distribution diagram 21
Fig. 4. Flow Chart of ASIC Design. 26
Fig. 5. Hardware implementation of Exponential function using Taylor series architecture 1. 28
Fig. 6. Hardware implementation of Exponential function using Taylor series Architecture 2. 31
Fig. 7. Synthesized design (partial) of Taylor series architecture 2. 32
Fig. 8. Taylor Series Architecture 2 with pipelined stage............. 33
Fig. 9. An 8x8 LUT after synthesis in Design Vision. 35
Fig. 10. Hardware implementation of exponential function using Linear Interpolation with 1 LUT. 37
Fig. 11. Hardware implementation of exponential function using Linear Interpolation method with 2 LUTs and 256 intervals. 39
Fig. 12. Hardware implementation of exponential function using Linear Interpolation with 2 LUTs using 128 intervals. 40
Fig. 13. Error difference of Taylor series architecture 1 in linear scale and total number of bits. 43
Fig. 14. Histogram of probability error distribution of Taylor Architecture 1. 45
FIG. 15. Error difference of Taylor series architecture 2 in linear scale and total number of bits. 46
Fig. 16. Histogram of probability error distribution of Taylor Architecture 2 for exponential function implementation. 48
Fig. 17. Error difference of linear interpolation architecture (using 1 LUT) in linear scale and total number of bits. 49
Fig. 18. Histogram showing error distribution of interpolation method for exponential function implementation using 1 LUT........ 51
Fig. 19. Error difference of linear interpolation architecture (with 2 LUTs and 256 intervals) in linear scale and total number of bits. 52

Fig. 20. Histogram of probability error distribution of interpolation method with 2 LUTs (using 256 intervals). 54
Fig. 21. Error difference of linear interpolation architecture (with 2 LUTs and 128 intervals) in linear scale and total number of bits. 55
Fig. 22. Histogram showing probability error distribution of linear interpolation architecture with 2 LUTs and 128 intervals................ 57
Fig. 23. Area comparison using LPHVT library.......................... 60
Fig. 24. Area comparison using LPLVT library 61
Fig. 25. A log-log graph of frequency versus power calculation for Taylor series architecture 2. 66
Fig. 26. A log-log graph of frequency versus power calculation for a linear interpolation architecture with 2 LUTs and 256 intervals.... 67
Fig. 27. Comparison of power estimation between Taylor series methodology and linear interpolation approach. 68

List of Tables

TABLE I. specifications of the hardware implementation of exponential function 12
TABLE II. Coefficients used in Taylor architecture 2 30
TABLE III. Cells used in hardware OF an 8x8 LUT................. 35
TABLE IV. Error metrics of Taylor architecture 1.................... 44
TABLE V. Error metrics of Taylor architecture 2......................... 47
TABLE VI. Error metrics of Linear interpolation method using 1 LUt 50
TABLE VII. Error metrics of Linear interpolation method using 2 LUts (256 intervals) 53
TABLE VIII. Error metrics of Linear interpolation method using 2 LUts using 128 intervals 56
TABLE IX. Area Report using library lphvt using 1.2v 60
TABLE X. Area report using library lplvt using 1.2v 61
TABLE XI. resources used for taylor architecture 2 using technology library lphvt at 1.2v. 62
TABLE XII. resources used for linear interpolation architecture with 2 luts and 256 intervals using technology library lphvt at 1.2v. 63
TABLE XIII. power report using library lphvt at 10mhz 63
TABLE XIV. power report using library lplvt at 10mhz............ 64
TABLE XV. timing report using library lphvt............................. 64
TABLE XVI. timing report using library lplvt........................... 65
TABLE XVII. Table of comparison of power consumption of two architectures using PrimeTime... 69

List of Acronyms

ASIC	Application Specific Integrated Circuit
CMOS	Complementary Metal Oxide Semiconductor
ENOB	Effective Number of Bits
FPGA	Field Programmable Gate Array
GIDL	Gate-Induced Drain Leakage
IC	Integrated circuit
LUT	Look-up table
LPHVT	Low Power High Threshold Voltage
LPLVT	Low Power Low Threshold Voltage
NMOS	N-type Metal Oxide Semiconductor
PMOS	P-type Metal Oxide Semiconductor
RBDL	Reverse-Biased Diode Leakage
RTL	Register Transfer Level
SNR	Signal to Noise Ratio
SNDR	Signal to Noise and Distortion Ratio
VCD	Value Change Dump
VHDL	**V**HSIC **H**ardware **D**escriptive **L**anguage
VHSIC	Very High-Speed Integrated Circuit

LUND
UNIVERSITY

http://www.eit.lth.se