HARDWARE IMPLEMENTATION OF THE BASE TWO LOGARITHMIC NUMBER SYSTEM

by

Steve Chih Hsiung

B.Ed., National Kaohsiung Teachers' College (Taiwan), 1981

M.S., University of North Dakota, 1986

A MASTER'S REPORT

Submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Electrical and Computer Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1988

Approved by:

D.V. Satish Chandra

Major Professor

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLE

List of Symbols

Symbol	Meaning
T	Fixed-Point or Floating-Point Number
M	Mantissa of an Integer Number
E	Exponent of an Integer Number
$\log_2 T$	The Base Two Logarithmic Number
N	Decimal Number
X	Binary Fraction
E	Error
A	Decimal Number
B	Decimal Number
A'	Base Two Logarithmic Number
B'	Base Two Logarithmic Number
K	The Most Significant "1" Bit Position in A Decimal Number
P	Product of Two Decimal Numbers

List of Symbols --- Continued

Symbol	Meaning
$\log_2 P'$	Approximation of Two Logarithmic Numbers Product
E_m	Maximum Error of Product
Q	Quotient of Two Decimal Numbers
$\log_2 Q'$	Approximation of Two Logarithmic Numbers Division
E_d	Maximum Error of Division
$\log A_2(1+X)$	Piece-Wise Linear Approximation
E_{max}	Maximum Error
E_{min}	Minimun Error
S	Sign Bit
τ	Scaling Factor ($\tau > 0$)
J	Finite Precision of Rounding a Base Two Logarithmic Number
$\oplus$	Exclusive OR Operation
$\beta(X)$	$\log_2(1+2^X)$
$\gamma(X)$	$\log_2(1-2^X)$

List of Symbols --- Continued

Symbol	Meaning
Mul	Multiplication (Product)
Div	Division
Sum	Addition (Summation)
Sub	Subtraction
σ_E^2	Variance of Errors
n_1	Register for Storing Integer Part of N, before Convertion
n_2	Register for Storing Fraction Part of N, before Convertion
n_1'	Register for Storing Integer Part of N, after Convertion
n_2'	Register for Storing Fraction Part of N, after Convertion
Q_n	Bits in Register n_1 and n_2, before Convertion
P_n	Bits in Priority Units
W	Detection Gate W=1 When $0 < N < 1$

List of Symbols --- Continued

Symbol	Meaning
R_n	Bits in Register n_1 and n_2, after Conversion
I_n	Bits Output from Decoder
ADC	Analog to Digital Converter
BLNC	Binary Logarithmic Number Converter
ABLNC	Anti-Binary Logarithmic Number Converter
ALU	Arithmetic Logic Unit
msb	Most Significant Bit
lsb	Least Significant Bit

CHAPTER I

INTRODUCTION

Many number systems have been used to implement computer arithmetic units. Most of the implementations use binary fractions and binary integers in either fixed-point or floating-point arithmetic. These systems have the problem of slow speed or high circuit complexity. The residue number system is attractive for its high speed. However, division, overflow detection, and magnitude comparison have effectively prevented the widespread use of this number system in general purpose computers.

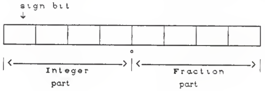
This report deals with the logarithmic representation of numbers, which offers a considerable increase in the dynamic range of digital computer arithmetic operations. The arithmetic operations discussed in this report are addition, subtraction, multiplication, and division. A brief review of the different number system representations is given below.

1.1 Fixed-Point and Floating-Point Representation:

An n-bit binary word representing a fixed-point number T is:

$$T = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + \ldots\ldots + a_2 2^2 + a_1 2^1 + a_0 2^0 +$$
$$a_{-1}2^{-1} + a_{-2}2^{-2} + \ldots\ldots + a_{-n}2^{-n}$$

Negative numbers may be represented by assigning the first bit of the binary word as a sign bit, or alternatively by using the two's complement algorithm.

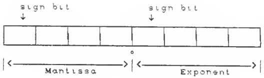


An n-bit binary word can also be represented as a floating-point number.

$$T = M * 2^E$$

Where M is the mantissa and E is the exponent of the integer number T. M is usually scaled to be a fraction whose decimal value lies in the range of $1/2 \leq M < 1$. [12]

The exponent E represents how many places the binary point should be shifted to the right ($E > 0$) or the left ($E < 0$).



$$\text{If } T = .1101 * 2^{11} \qquad \text{binary}$$

$$= [1/2 + 1/4 + 1/16] * 2^{3} \qquad \text{decimal}$$

$$= 6.5 \qquad \text{decimal}$$

1.2 Logarithmic Representation:

Fixed-point numbers are simple and easy to use, but they are limited to the range the number can be represented, and overflow will cause inaccuracy in the computation. Floating-point numbers are more flexible than the fixed-point numbers and are the dominating choice of system designers when a large dynamic range and high precision are required simultaneously. Floating-point multiplication and division require a complex series of additions, subtractions, shifts, and iterations, which are time consuming.

If we take a look at the characteristics of the logarithmic numbers, the multiplication and division operations are changed to addition and subtraction operations. The computation time in addition and subtraction operations are much shorter than multiplication and division operations. Because all the signals in the digital computer are in the binary format, the binary logarithmic numbers are best suited for use in digital computers. The binary logarithms may be determined approximately from the number itself by simple shifting and counting. The logarithmic number system supports high speed and high precision arithmetic.

Let N be a nonzero binary number with finite length.

$$N = \sum_{i=j}^{K} 2^{i} Z_{i} \text{ (decimal)}$$

Here i, j, K are integer numbers ($K \geq j$) and $Z_i = 0$ or 1. Z_i is the ith order bit of the binary number N. Z_K is the most significant bit (msb) and Z_j is the least significant bit (lsb) of N. If Z_K is "1",

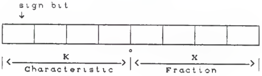
$$\text{then } N = 2^{K} \left[1 + \sum_{i=j}^{K-1} 2^{i-K} Z_i\right]$$

$$\text{Let } X = \sum_{i=j}^{K-1} 2^{i-K} Z_i, \qquad 0 \leq X < 1,\ K \geq j,$$

$$N = 2^{K}(1 + X)$$

Assume $\log_2(1 + X) \simeq X$

Then $\log_2 N = K + X$



Logarithmic arithmetic has been used in the implementation of digital filters [8,9,11,13,26], fast Fourier transforms [22], and other digital signal rocessing algorithms. Logarithmic arithmetic algorithms have accuracy with speed, while floating-point arithmetic provides accuracy at the expense of speed.

CHAPTER II

ANALYSIS OF THE BASE TWO LOGARITHMIC NUMBER SYSTEM

A number of approximation techniques have been proposed for the fast computation of the binary logarithmic numbers, such as "Focus Number System" proposed by Edgar [5], [15], which is similar to the "Sign/Logarithm Number System", and "Binary Logarithm" proposed by Lo [17] which used the same simple shifting and counting techniques but added a fixed number to reduce the transformation errors. Here, we choose the three representative techniques to analyze the characteristics of the binary logarithmic numbers.

2.1 Simple Shifting and Counting:

The first proposal of binary logarithmic number system was made by Mitchell [20]. This approximation to binary logarithmic number is easy to generate just by simple shifting and counting. To find the binary logarithm of a binary number, use the most significant "1" bit position to determine the characteristic, and interpret the remaining bits as a binary fraction.

For example, consider $13_{10} = 1101_2$ and $\log_2 13 = 3.700439718_{10}$. The most significant "1" bit is in the 2^3 position, and the characteristic is 3. Considering the bits to the right of the most significant "1" as a binary fraction there results 0.101 which equivalent to 0.625 in decimal. The approximation is $\log_2 13 \simeq 3.625_{10} \simeq 11.101_2$.

In logarithmic arithmetic the multiplication and division operations are reduced to simple addition and subtraction operations respectively. Consider a binary number N:

$$N = \sum_{i=j}^{K} 2^i Z_i \text{ (decimal)}$$

Where $N = Z_K \cdots\cdots Z_3 Z_2 Z_1 Z_0 . Z_{-1} Z_{-2} Z_{-3} \cdots\cdots Z_j$ (binary)

If Z_K is the most significant "1" bit

Then $N = 2^K \left[1 + \sum_{i=j}^{K-1} 2^{i-K} Z_i\right]$

Let $X = \sum_{i=j}^{K-1} 2^{i-K} Z_i$, $\quad 0 \leq X < 1$,

X is interpreted as a binary fraction

$$\therefore N = 2^K(1 + X)$$

$\log_2 N = K + \log_2(1 + X)$

We assume $\log_2(1 + X) \simeq X$

So the error $E = \log_2(1 + X) - X$

$$\frac{dE}{dX} = \frac{1}{(1 + X)\ln 2} - 1 = 0$$

$$\Longrightarrow \quad X = \frac{1}{\ln 2} - 1 = 0.44269$$

$$0 \le E \le \log_2(1 + X) - X$$

$$\Longrightarrow \quad 0 \le E \le \log_2(1.44269) - 0.4426$$

$$\Longrightarrow \quad 0 \le E \le 0.08639 \text{ is the maximun}$$

error in the absolute value.

Multiplication:

$$\text{Let } A' = \log_2 A = K_1 + \log_2(1 + X_1)$$

$$B' = \log_2 B = K_2 + \log_2(1 + X_2)$$

$$P = AB = 2^{A'} 2^{B'} = 2^{K1+K2}(1 + X_1)(1 + X_2)$$

$$\log_2 P' \simeq K_1 + K_2 + X_1 + X_2$$

$$\log_2(1 + X) \simeq X$$

Without carry: $\log_2 P' = K_1 + K_2 + (X_1 + X_2), \; X_1 + X_2 < 1$

With carry: $\log_2 P' = (1 + K_1 + K_2) + (X_1 + X_2 - 1),$

$$X_1 + X_2 \ge 1$$

Take the antilogarithm:

$$P' = 2^{K1+K2}(1 + X_1 + X_2), \; X_1 + X_2 < 1$$

$$P' = 2^{K1+K2+1}(X_1 + X_2), \quad X_1 + X_2 \ge 1$$

$$\text{The error } E_m = \frac{P' - P}{P} = \frac{P'}{P} - 1$$

The maximum $E_m = -11.1$ % at $X_1 = X_2 = 1/2$

Division:

$$Q = A / B = 2^{A'} / 2^{B'} = 2^{K1-K2}\left(\frac{1 + X_1}{1 + X_2}\right)$$

$$\log_2 Q' \simeq K_1 + X_1 - K_2 - X_2$$

Without borrow: $\log_2 Q' = (K_1 - K_2) + (X_1 - X_2)$,

$$X_1 - X_2 \geq 0$$

With borrow: $\log_2 Q' = (K_1 - K_2 - 1) + (1 + X_1 - X_2)$,

$$X_1 - X_2 < 0$$

Take the antilogarithm:

$$Q' = 2^{K1-K2}(1 + X_1 - X_2), \quad X_1 - X_2 \geq 0$$

$$Q' = 2^{K1-K2-1}(2 + X_1 - X_2), \quad X_1 - X_2 < 0$$

The error $E_d = \dfrac{Q' - Q}{Q} = \dfrac{Q'}{Q} - 1$

The maximum $E_d = 12.5\%$ at $X_1 = 1$, $X_2 = 1/2$ without borrow or at $X_1 = 0$, $X_2 = 1/2$ with borrow.

2.2 Piece-Wise Linear Approximation:

The approximation $\log_2(1 + X) \simeq X$ is to substitute the base two logarithmic curve by straight lines connecting the points of the curve where $\log_2 N$ has an integral value. The characteristic of $\log_2 N$ is equal to the number of bits between the leftmost "1" bit and the binary point of N.

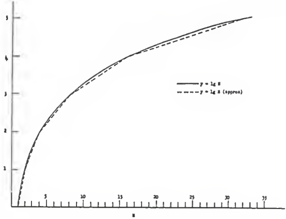


Figure 1. Logarithmic Curve and Straight-Line Approximation.

Using the piece-wise linear approximation, as proposed by Combet [3], we can have a reduction in the conversion error. The general form in each interval is $\log A_2(1 + X) = X + af(X) + b$.

$$\text{Where } f(x) = \begin{cases} X \text{ if slope} \geq 1 \\ 1 - X \text{ if slope} < 1, \text{ we could take} \\ \qquad f(X) = X \end{cases}$$

We can use the four segments for the approximation $\log A_2(1 + X)$ in each interval:

$$\begin{cases} \log A_2(1+X) = X + (5/16)X, & 0 \le X < 1/4 \\ \log A_2(1+X) = X + 5/64, & 1/4 \le X < 1/2 \\ \log A_2(1+X) = X + (1/8)\overline{X} + 3/128, & 1/2 \le X < 3/4 \\ \log A_2(1+X) = X + (1/4)\overline{X}, & 3/4 \le X < 1 \end{cases}$$

For example, let's consider the same number 13_{10} = 1101_2, and $\log_2 13 = 3.700439718_{10}$.

$13_{10} = 1101_2$

$= 2^3(1 + 0.101)$

For $1/2 \le X = 0.101_2 = 0.625_{10} < 3/4$

$\therefore \log A_2(1 + X) = 0.625 + (1/8)0.25 + 3/128$

$= 0.6796875_{10}$

$\therefore \log_2 13 \simeq 3.6796875_{10}$

The piece-wise linear approximation involves not only shifting and counting operations to find the characteristic and the approximated mantissa as the straight line approximation but also binary decision for the determination of the type of correction and addition of the binary numbers.

The results of errors are $E = \log_2(1+X) - \log A_2(1+X)$, maximum positive error $E_{max} = 0.008$ at $X = 0.44$, maximum negative error $E_{max} = -0.006$ at $X = 0.25$, and error range $0.008 + 0.006 = 0.014$.

In addition to the binary logarithm approximation error, errors are also introduced by the finite length registers in which the binary logarithmic numbers are stored.

2.3 Table Look-Up Method:

In the sign/logarithm number system proposed by Swartzlander [21], a number is represented by a sign bit and the logarithm of the absolute value of the number (scaled to avoid negative logarithms). Any real number A is represented by its sign S_A, and the binary logarithm of its magnitude A'.

$$S_A = 1 \quad \text{if } A \le 0$$

$$S_A = 0 \quad \text{if } A \ge 0$$

$$S_A = 0 \text{ or } 1 \quad \text{if } A = 0$$

$$A' = \log_2(|\tau A|), \quad \text{if } A > 1/\tau$$

$$A' = 0, \qquad \text{if } A \le 1/\tau$$

$$A = (1 - 2S_A)(1/\tau)2^{A'}$$

A is scaled by a constant factor τ to ensure that $A' \ge 0$.

J_A is the finite precision binary logarithmic number formed by rounding A'.

$$\therefore J_A = [1/2 + \log_2|\tau A|2^{\eta-1}] * 2^{1-\eta}, \text{ if } |A| > 1/\tau$$

$$J_A = 0, \text{ if } |A| \le 1/\tau$$

Hhere [Y] denotes the largest integer that is not larger than Y. The constant 1/2 causes round off to occur in the formation of J_A instead of simple truncation, thus unbiasing the error and reducing error accumulation.

Choose $\tau = 2^{\eta}$, where the η-1 bits represent the fractional part of J_A.

$$J_A = (J_n J_{n-1} \ldots J_{\eta} \circ J_{\eta-1} \ldots J_1) = \sum_{\iota=1}^{n} J_{\iota} 2^{\iota-\eta}$$

sign bit
↓

	$J_n J_{n-1} \ldots J_{\eta} \circ J_{\eta-1} \ldots J_1$

For example, let's consider the number $13_{10} = 1011_2$ again. Now $\log_2 13 = 3.700439718_{10}$. If we choose eight bits in both integer and fraction part of binary logarithm format.

Then $\eta = 8$ and $\tau = 2^8 = 256$

For $A = 13_{10}$

$$A' = \log_2(13 * 256) = 11.70043972_{10}$$

$$\therefore J_A = [1/2 + \log_2|\tau A|2^{8-1}] * 2^{1-8}$$

$$= [1/2 + 1497.656284] * 2^{-7}$$

$$= 1498 * 2^{-7} = 11.703125_{10}$$

The approximate value of $\log_2 13$, after scaling back is:

$$\log_2 13 \simeq 11.703125 - 8 = 3.703125_{10}$$

Multiplication:

$$Mul = A' + B' = \log_2(\tau A) + \log_2(\tau B) = \log_2(\tau\tau AB)$$

$$\therefore J_{Mul} = J_A + J_B - J_\tau$$

Where $J_\tau = [1/2 + \log_2(\tau)2^{\eta-1}]2^{1-\eta}$

$$S_{Mul} = S_A \oplus S_B$$

Division:

$$Div = A' - B' = \log_2(\tau A) - \log_2(\tau B) = \log_2(AB)$$

$$\therefore J_{Div} = J_A - J_B + J_\tau$$

Where $J_\tau = [1/2 + \log_2(\tau)2^{\eta-1}]2^{1-\eta}$

$$S_{Div} = S_A \oplus S_B$$

Addition:

Sum = A + B ====> Sum = A(1 + B/A)

or Sum = B(1 + A/B)

If $J_A \geq J_B$

$$S_{Sum} = S_A$$

$$J_{Sum} = J_A + \beta(J_B - J_A)$$

Where $\beta(X) = \log_2(1 + 2^X)$

If $J_A < J_B$

$$S_{Sum} = S_B$$

$$\therefore J_{Sum} = J_B + \beta(J_A - J_B)$$

Where $\beta(X) = \log_2(1 + 2^X)$

But $\beta(X)$ is rounded off as: $\beta(X) = 2^{1-\eta}[1/2 + 2^{\eta-1}\log_2(1+2^X)]$

Subtraction:

$$Sub = A - B \Longrightarrow Sub = A(1 - B/A) \text{ or}$$

$$Sub = B - A \Longrightarrow Sub = B(1 - A/B)$$

If $J_A \geq J_B$

$$S_{Sub} = S_A$$

$$J_{Sub} = J_A + \gamma(J_B - J_A)$$

Where $\gamma(X) = \log_2(1 - 2^X)$

If $J_A < J_B$

$$S_{Sub} = S_B$$

$$J_{Sub} = J_B + \gamma(J_A - J_B)$$

Where $\gamma(X) = \log_2(1 - 2^X)$

But $\gamma(X)$ is rounded off as: $\gamma(X) = 2^{1-\eta}[1/2 + 2^{\eta-1}\log_2(1-2^X)]$

The values of $\beta(X)$ and $\gamma(X)$ are obtained from the look-up table in the ROM memory. The function $\beta(X)$ or $\gamma(X)$ introduces an error term which could be expressed by:

$$E = \beta(J_B - J_A) - \log_2(1 + 2^{JB-JA})$$

If $J_B - J_A = X$

$$\text{Then } E = \{2^{1-\eta}[1/2 + 2^{\eta-1}\log_2(1 + 2^X)]\} - \log_2(1 + 2^X)$$

$$\text{Since} \quad -2^{n-\eta+1} < X < 2^{n-\eta+1}$$

$$-2^{-\eta} < E \leq 2^{-\eta}$$

$$\sigma_E^2 = 2^{-2\eta}/3 \quad [22]$$

Table I

Error Comparison of Binary Logarithmic Number System

	E_{max}	E_{min}
1. Simple Shifting and Counting	0.0866	0
2. Piece-Wise Linear	0.014	0
3. A Table Look-Up	$2^{-\eta}$ (Where $\eta-1$ is the bits represent the fractional part of the register, in the 8 bit example $\eta-1 = 8$)	$-2^{-\eta}$

CHAPTER III

HARDWARE IMPLEMENTATION AND DESCRIPTION

3.1 Binary Logarithmic Number Conversion:

In formating the base two logarithm numbers ($N \longrightarrow \log_2 N$), only positive numbers greater than 1 ($N \geq 1$) are considered. The procedure can be extended to numbers in the range $0 < N < 1$. A non-zero binary number N with finite length can be written as:

$$N = \sum_{\iota=j}^{K} 2^{\iota} z_{\iota}$$

$$= 2^{K} + \sum_{\iota=j}^{K-1} 2^{\iota} z_{\iota}$$

$$= 2^{K}(1 + X)$$

Where $X = \sum_{\iota=j}^{k-1} 2^{\iota-K} z_{\iota}$ represents the binary fraction [6] which is that part of the number to the right of the most significant "1".

For example: (n1 and n2 are two registers for storing N)

When $N \geq 1$

$$N = \underset{h}{0}\,\underbrace{\overbrace{1\,\overbrace{011001}^{m}}^{n1}.\overbrace{1011000}^{n2}}_{x} \text{ (binary)}$$

where h is the number of bits on the left of the most significant "1" written in register n_1 and n_2, and m is the number of bits between this "1" and binary part.

$$N' = \overbrace{\underbrace{0}_{h}1\underbrace{0110011011000}_{X}}^{n_1 + n_2}. = 2^{n_2} N$$

$$\therefore N' = N\ 2^{n_2} \Longrightarrow N = \frac{N'}{2^{n_2}} = \frac{(1+X)\,2^{n_1+n_2-h-1}}{2^{n_2}}$$

$$= 2^{n_1-h-1}(1+X)$$

$$\log_2 N \simeq n_1 - h - (1 - X) \qquad (\text{Assume: } \log_2(1+X) \simeq X)$$

$$\simeq m + X \qquad \text{where } m = n_1 - h - 1$$

When $0 < N < 1$

$$N = \underbrace{\overbrace{00000000}^{n_1}}_{h}.\overbrace{\underbrace{000}_{m}1\underbrace{0110}_{X}}^{n_2}$$

$$\log_2 N \simeq n_1 - h - (1-X) = n_1 - (n_1+m) - (1-X)$$

$$= -m - (1-X)$$

Where $(1-X)$ is the two's complement of X $(0 \leq X < 1)$ also $\log_2 N \simeq -(m + \bar{X})$, $\bar{X}$ is one's complement of X, when $[n_2 - (m + 1)] \gg 1$.

The direct transfromation from binary numbers to binary logarithmic numbers is implemented using the hardware design proposed by Frangakis [6]. This hardware logic does not require any shifting and counting thus resulting in faster computations.

The binary logarithm conversion procedure is indicated by the following block diagrams.

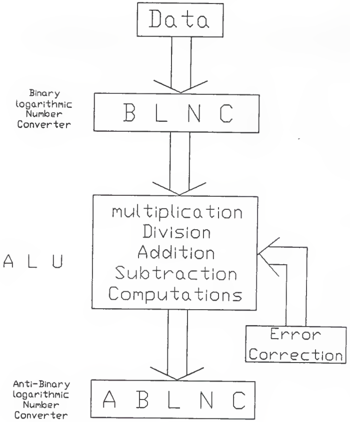


Figure 2. Block Diagram of the Binary Logarithmic Number System.

3.2 Binary Logarithmic Number Converter:

In transforming the binary numbers to the binary logarithm numbers, we choose eight bits for each register n_1 and register n_2. The BLNC (Binary Logarithmic Number Converter) can be represented as shown below:

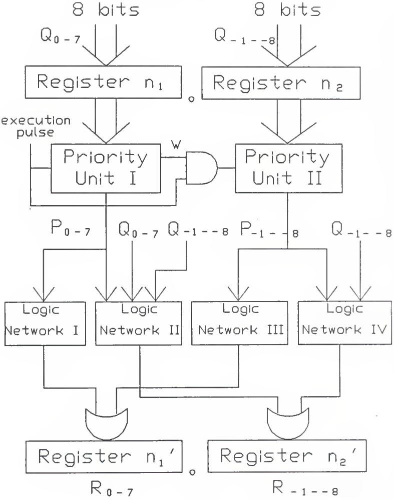


Figure 3. Block Diagram of the BLNC.

Priority Unit I: It is used to detect the most significant "1" bit written in register M1.

$$P_0 = Q_0\bar{Q}_1\bar{Q}_2\bar{Q}_3\bar{Q}_4\bar{Q}_5\bar{Q}_6\bar{Q}_7$$

$$P_1 = Q_1\bar{Q}_2\bar{Q}_3\bar{Q}_4\bar{Q}_5\bar{Q}_6\bar{Q}_7$$

$$P_2 = Q_2\bar{Q}_3\bar{Q}_4\bar{Q}_5\bar{Q}_6\bar{Q}_7$$

$$P_3 = Q_3\bar{Q}_4\bar{Q}_5\bar{Q}_6\bar{Q}_7$$

$$P_4 = Q_4\bar{Q}_5\bar{Q}_6\bar{Q}_7$$

$$P_5 = Q_5\bar{Q}_6\bar{Q}_7$$

$$P_6 = Q_6\bar{Q}_7$$

$$P_7 = Q_7$$

$$W = \bar{Q}_0\bar{Q}_1\bar{Q}_2\bar{Q}_3\bar{Q}_4\bar{Q}_5\bar{Q}_6\bar{Q}_7$$

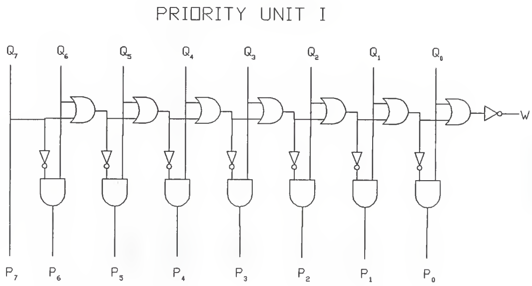


Figure 4. The Priority Unit I.

Priority Unit II: It is used to detect the most significant "1" bit written in register n2, when $0 < N < 1$ and $W = 1$.

$$P_{-1} = Q_{-1}$$

$$P_{-2} = Q_{-2}\bar{Q}_{-1}$$

$$P_{-3} = Q_{-3}\bar{Q}_{-2}\bar{Q}_{-1}$$

$$P_{-4} = Q_{-4}\bar{Q}_{-3}\bar{Q}_{-2}\bar{Q}_{-1}$$

$$P_{-5} = Q_{-5}\bar{Q}_{-4}\bar{Q}_{-3}\bar{Q}_{-2}\bar{Q}_{-1}$$

$$P_{-6} = Q_{-6}\bar{Q}_{-5}\bar{Q}_{-4}\bar{Q}_{-3}\bar{Q}_{-2}\bar{Q}_{-1}$$

$$P_{-7} = Q_{-7}\bar{Q}_{-6}\bar{Q}_{-5}\bar{Q}_{-4}\bar{Q}_{-3}\bar{Q}_{-2}\bar{Q}_{-1}$$

$$P_{-8} = Q_{-8}\bar{Q}_{-7}\bar{Q}_{-6}\bar{Q}_{-5}\bar{Q}_{-4}\bar{Q}_{-3}\bar{Q}_{-2}\bar{Q}_{-1}$$

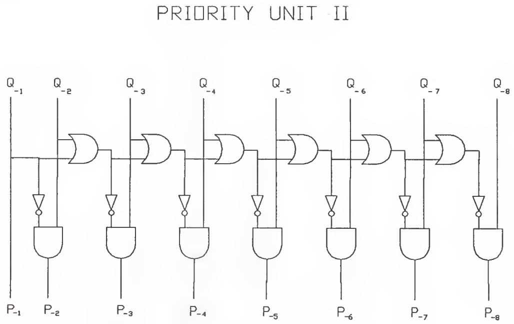


Figure 5. The Priority Unit II.

Logic Network I (for $N > 1$): It is used to determine which flip-flop ($R_0 \rightarrow R_2$) in register n1 to be set to "1".

$P_0 \xrightarrow{set} 0 \longrightarrow 0.00 \cdots$

$P_1 \xrightarrow{set} R_0 \longrightarrow 1.00 \cdots$

$P_2 \xrightarrow{set} R_1 \longrightarrow 10.00 \cdots$

$P_3 \xrightarrow{set} R_0, R_1 \longrightarrow 11.00 \cdots$

$P_4 \xrightarrow{set} R_2 \longrightarrow 100.00 \cdots$

$P_5 \xrightarrow{set} R_0, R_2 \longrightarrow 101.00 \cdots$

$P_6 \xrightarrow{set} R_1, R_2 \longrightarrow 110.00 \cdots$

$P_7 \xrightarrow{set} R_0, R_1, R_2 \longrightarrow 111.00 \cdots$

Logic Network III (for $0 < N < 1$): It is used to determine which flip-flop ($R_0 \rightarrow R_7$) in register n1 to be set to "1".

$P_{-1} \xrightarrow{set} R_0, R_1, R_2, \ldots, R_7 \longrightarrow 11111111. \cdots$

$P_{-2} \xrightarrow{set} R_1, R_2, R_3, \ldots, R_7 \longrightarrow 11111110. \cdots$

$P_{-3} \xrightarrow{set} R_0, R_2, R_3, \ldots, R_7 \longrightarrow 11111101. \cdots$

$P_{-4} \xrightarrow{set} R_2, R_3, R_4, \ldots, R_7 \longrightarrow 11111100. \cdots$

$P_{-5} \xrightarrow{set} R_0, R_1, R_3, \ldots, R_7 \longrightarrow 11111011. \cdots$

$P_{-6} \xrightarrow{set} R_1, R_3, R_4, \ldots, R_7 \longrightarrow 11111010. \cdots$

$P_{-7} \xrightarrow{set} R_0, R_3, R_4, \ldots, R_7 \longrightarrow 11111001. \cdots$

$P_{-8} \xrightarrow{set} R_3, R_4, R_5, \ldots, R_7 \longrightarrow 11111000. \cdots$

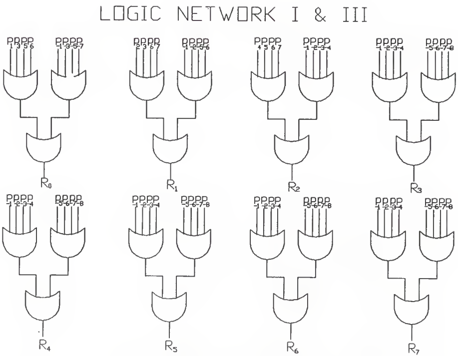


Figure 6. The Logic Network I & III.

Logic Network II (for N > 1): It is used to determine which flip-flop ($R_{-1} \rightarrow R_{-8}$) in register nz to be set to "1".

$$R_{-1} = P_0Q_{-1}+P_1Q_0+P_2Q_1+P_3Q_2+P_4Q_3+P_5Q_4+P_6Q_5+P_7Q_6$$

$$R_{-2} = P_0Q_{-2}+P_1Q_{-1}+P_2Q_0+P_3Q_1+P_4Q_2+P_5Q_3+P_6Q_4+P_7Q_5$$

$$R_{-3} = P_0Q_{-3}+P_1Q_{-2}+P_2Q_{-1}+P_3Q_0+P_4Q_1+P_5Q_2+P_6Q_3+P_7Q_4$$

$$R_{-4} = P_0Q_{-4}+P_1Q_{-3}+P_2Q_{-2}+P_3Q_{-1}+P_4Q_0+P_5Q_1+P_6Q_2+P_7Q_3$$

$$R_{-5} = P_0Q_{-5}+P_1Q_{-4}+P_2Q_{-3}+P_3Q_{-2}+P_4Q_{-1}+P_5Q_0+P_6Q_1+P_7Q_2$$

$$R_{-6} = P_0Q_{-6}+P_1Q_{-5}+P_2Q_{-4}+P_3Q_{-3}+P_4Q_{-2}+P_5Q_{-1}+P_6Q_0+P_7Q_1$$

$$R_{-7} = P_0Q_{-7}+P_1Q_{-6}+P_2Q_{-5}+P_3Q_{-4}+P_4Q_{-3}+P_5Q_{-2}+P_6Q_{-1}+P_7Q_0$$

$$R_{-8} = P_0Q_{-8}+P_1Q_{-7}+P_2Q_{-6}+P_3Q_{-5}+P_4Q_{-4}+P_5Q_{-3}+P_6Q_{-2}+P_7Q_{-1}$$

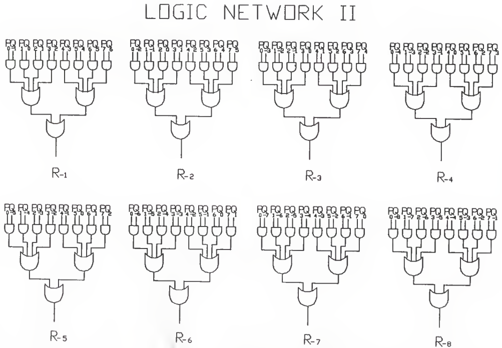


Figure 7. The Logic Network II.

Logic Network IV (for $0 < N < 1$): It is used to determine which flip-flop ($R_{-1} \to R_{-8}$) in register nz to be set to "1".

$$R_{-1} = P_{-1}Q_{-2}+P_{-2}Q_{-3}+P_{-3}Q_{-4}+P_{-4}Q_{-5}+P_{-5}Q_{-6}+P_{-6}Q_{-7}+P_{-7}Q_{-8}$$

$$R_{-2} = P_{-1}Q_{-3}+P_{-2}Q_{-4}+P_{-3}Q_{-5}+P_{-4}Q_{-6}+P_{-5}Q_{-7}+P_{-6}Q_{-8}$$

$$R_{-3} = P_{-1}Q_{-4}+P_{-2}Q_{-5}+P_{-3}Q_{-6}+P_{-4}Q_{-7}+P_{-5}Q_{-8}$$

$$R_{-4} = P_{-1}Q_{-5}+P_{-2}Q_{-6}+P_{-3}Q_{-7}+P_{-4}Q_{-8}$$

$$R_{-5} = P_{-1}Q_{-6}+P_{-2}Q_{-7}+P_{-3}Q_{-8}$$

$$R_{-6} = P_{-1}Q_{-7}+P_{-2}Q_{-8}$$

$$R_{-7} = P_{-1}Q_{-8}$$

$$R_{-8} = 0$$

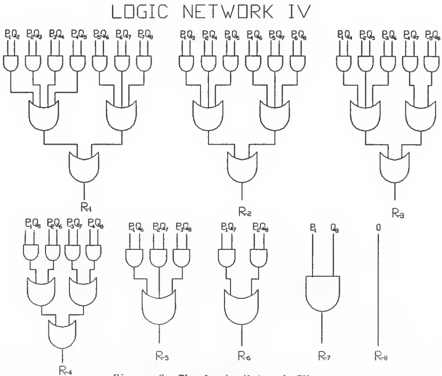


Figure 8. The Logic Network IV.

The number N ($N \geq 1$ stored in registers n_1 and n_2 will appear as the logarithmic number $\log_2 N$ in registers n_1' and n_2' after the conversion. If $0 < N < 1$, then logarithmic number in registers n_1' and n_2' will be in one's complement representation.

3.3 Anti-Binary Logarithmic Number Converter:

To transform binary logarithm numbers to binary numbers, we choose eight bits for each register n_1 and register n_2. This is the inverse procedure of taking the binary logarithmic numbers. The ABLNC (Anti-Binary Logarithmic Number Converter) can be represented as shown in Figure 9:

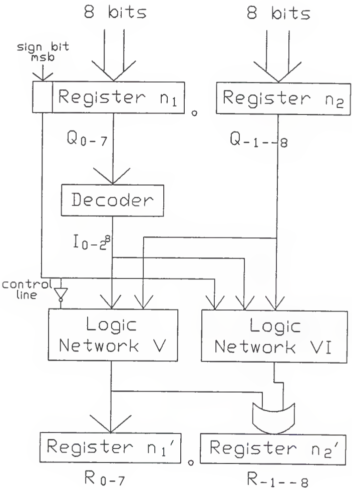


Figure 9. The Block Diagram of ABLNC.

Control line selects either logic network V or VI depending on whether the msb in register n1 is set to "1" or set to "0".

Logic Network V (for N > 0):

$$R_0 = I_0+I_1Q_{-1}+I_2Q_{-2}+I_3Q_{-3}+I_4Q_{-4}+I_5Q_{-5}+I_6Q_{-6}+I_7Q_{-7}+I_8Q_{-8}$$

$$R_1 = I_1+I_2Q_{-1}+I_3Q_{-2}+I_4Q_{-3}+I_5Q_{-4}+I_6Q_{-5}+I_7Q_{-6}+I_8Q_{-7}+I_9Q_{-8}$$

$$R_2 = I_2+I_3Q_{-1}+I_4Q_{-2}+I_5Q_{-3}+I_6Q_{-4}+I_7Q_{-5}+I_8Q_{-6}+I_9Q_{-7}+I_{10}Q_{-8}$$

$$R_3 = I_3+I_4Q_{-1}+I_5Q_{-2}+I_6Q_{-3}+I_7Q_{-4}+I_8Q_{-5}+I_9Q_{-6}+I_{10}Q_{-7}+I_{11}Q_{-8}$$

$$R_4 = I_4+I_5Q_{-1}+I_6Q_{-2}+I_7Q_{-3}+I_8Q_{-4}+I_9Q_{-5}+I_{10}Q_{-6}+I_{11}Q_{-7}+I_{12}Q_{-8}$$

$$R_5 = I_5+I_6Q_{-1}+I_7Q_{-2}+I_8Q_{-3}+I_9Q_{-4}+I_{10}Q_{-5}+I_{11}Q_{-6}+I_{12}Q_{-7}+I_{13}Q_{-8}$$

$$R_6 = I_6+I_7Q_{-1}+I_8Q_{-2}+I_9Q_{-3}+I_{10}Q_{-4}+I_{11}Q_{-5}+I_{12}Q_{-6}+I_{13}Q_{-7}+I_{14}Q_{-8}$$

$$R_7 = I_7+I_8Q_{-1}+I_9Q_{-2}+I_{10}Q_{-3}+I_{11}Q_{-4}+I_{12}Q_{-5}+I_{13}Q_{-6}+I_{14}Q_{-7}+I_{15}Q_{-8}$$

$$R_{-1} = I_0Q_{-1}+I_1Q_{-2}+I_2Q_{-3}+I_3Q_{-4}+I_4Q_{-5}+I_5Q_{-6}+I_6Q_{-7}+I_7Q_{-8}$$

$$R_{-2} = I_0Q_{-2}+I_1Q_{-3}+I_2Q_{-4}+I_3Q_{-5}+I_4Q_{-6}+I_5Q_{-7}+I_6Q_{-8}$$

$$R_{-3} = I_0Q_{-3}+I_1Q_{-4}+I_2Q_{-5}+I_3Q_{-6}+I_4Q_{-7}+I_5Q_{-8}$$

$$R_{-4} = I_0Q_{-4}+I_1Q_{-5}+I_2Q_{-6}+I_3Q_{-7}+I_4Q_{-8}$$

$$R_{-5} = I_0Q_{-5}+I_1Q_{-6}+I_2Q_{-7}+I_3Q_{-8}$$

$$R_{-6} = I_0Q_{-6}+I_1Q_{-7}+I_2Q_{-8}$$

$$R_{-7} = I_0Q_{-7}+I_1Q_{-8}$$

$$R_{-8} = I_0Q_{-8}$$

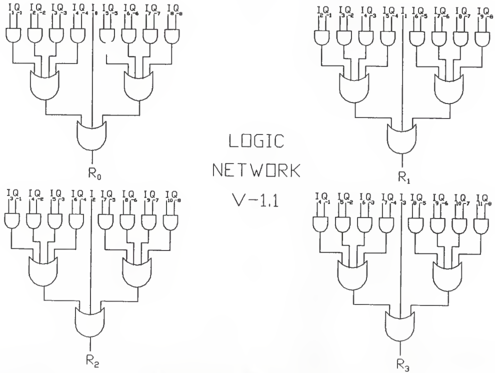


Figure 10. The Logic Network V-1.1.

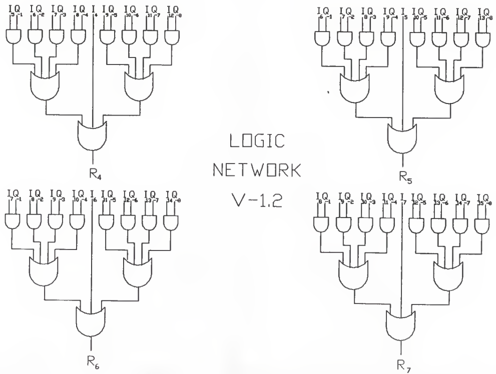


Figure 11. The Logic Network V-1.2.

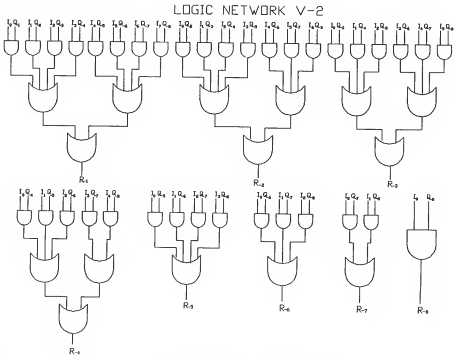


Figure 12. The Logic Network V-2.

Logic Network VI (for N < 0):

$$R_{-1} = I_{15}$$

$$R_{-2} = I_{14} + I_{15}Q_{-1}$$

$$R_{-3} = I_{13} + I_{15}Q_{-2} + I_{14}Q_{-1}$$

$$R_{-4} = I_{12} + I_{15}Q_{-3} + I_{14}Q_{-2} + I_{13}Q_{-1}$$

$$R_{-5} = I_{11} + I_{15}Q_{-4} + I_{14}Q_{-3} + I_{13}Q_{-2} + I_{12}Q_{-1}$$

$$R_{-6} = I_{10} + I_{15}Q_{-5} + I_{14}Q_{-4} + I_{13}Q_{-3} + I_{12}Q_{-2} + I_{11}Q_{-1}$$

$$R_{-7} = I_{9} + I_{15}Q_{-6} + I_{14}Q_{-5} + I_{13}Q_{-4} + I_{12}Q_{-3} + I_{11}Q_{-2} + I_{10}Q_{-1}$$

$$R_{-8} = I_{8} + I_{15}Q_{-7} + I_{14}Q_{-6} + I_{13}Q_{-5} + I_{12}Q_{-4} + I_{11}Q_{-3} + I_{10}Q_{-2} + I_{9}Q_{-1}$$

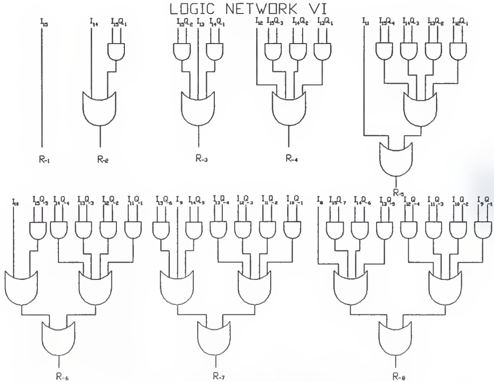


Figure 13. The Logic Network VI.

3.4 Decription of the Four Basic Arithmetic Operations:

The four basic arithmetic operations in the base two logarithmic number system are described as follows.

Multiplication and Division:

$$\left.\begin{aligned} \text{Let } A' &= \log_2 A \\ B' &= \log_2 B \end{aligned}\right\} \text{ ====> Through BLNC}$$

$$\begin{aligned} \log_2 AB &= \log_2 A + \log_2 B \\ &= A' + B' \end{aligned}$$

$\therefore$ Multiplication: $AB = 2^{A' + B'}$ ====> Through ABLNC

$$\begin{aligned} \log_2 (A/B) &= \log_2 A - \log_2 B \\ &= A' - B' \end{aligned}$$

$\therefore$ Division: $A/B = 2^{A' - B'}$ ====> Through ABLNC

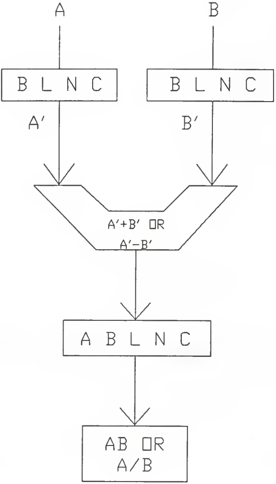


Figure 14. The Block Diagram Diagram of Multiplication And Division Operations.

Addition and Subtraction:

$$\left.\begin{aligned} \text{Let } A' &= \log_2 A \\ B' &= \log_2 B \end{aligned}\right\} \text{====> Through BLNC}$$

$$A + B = A(1 + B/A)$$

$$\log_2(A + B) = \log_2 A + \log_2(1 + B/A)$$

$$\therefore \text{ Addition: } A + B = 2^{A' + \beta(B' - A')}$$

$$\text{Where } \beta(B' - A') = \log_2(1 + B/A) = \log_2(1 + 2^{B' - A'})$$

$$\therefore \beta(X) = \log_2(1 + 2^X) \text{ ====> in ROM}$$

Subtraction: $A - B = A(1 - B/A)$

$$\log_2(A - B) = \log_2 A + \log_2(1 - B/A)$$

$$A - B = 2^{A' + \gamma(B' - A')}$$

$$\text{Where } \gamma(B' - A') = \log_2(1 - B/A)$$

$$\therefore \gamma(X) = \log_2(1 - 2^X) \text{ ====> in ROM}$$

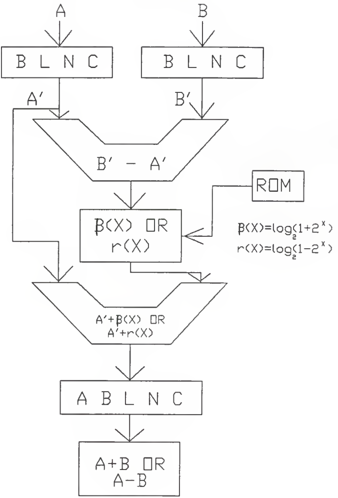


Figure 15. The Block Diagram of Addition and Subtraction Operations.

CHAPTER IV

SUMMARY AND CONCLUSIONS

The three ways of formating the binary logarithmic numbers, simple shifting and counting, piece-wise linear approximation, and table look-up method are discussed in Chapter II.

Comparing the errors in these three methods, as shown in Table I, the table look-up method has the least error, but it requires more processing time and large ROM memory. The simple shifting and counting has the largest error, but it is the fastest processing method.

In the hardware implementation discussed in Chpater III, we use direct logic gates to approximate the binary logarithmic numbers which is even more faster than the simple shifting and counting method. In the addition and subtraction operations we use the look-up table ROM to approximate $\log_2(1 + 2^x)$ and $\log_2(1 - 2^x)$.

The error produced by the hardware implementation discussed in this report is the same as that produced by simple shifting and counting technique. Other methods could be used to reduce the error but at the expense of speed. A logarithmic A/D converter may be useful for the direct processing of analog signals in the real world. Hardware implementation of floating-point to binary logarithmic number transformation needs further study.

REFERENCES

1. Brubaker, T. A. and Becker, J. C., "Multiplication Using Logarithms Implemented with Read-Only Memory", IEEE Trans. on Computers, Vol. C-24, pp. 761-765, Aug. 1975.
2. Chandra, D. V. S., "Accumulation of Coefficient Round off Error in Fast Fourier Transform Implemented with Logarithmic Number System", IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-35, pp. 1633-1636, Nov. 1987.
3. Combet, M., Zonneveld, H. Van, and Verbeck, L., "Computation of the Base Two Logarithm of Binary Numbers", IEEE Trans. on Electronic Computers, Vol. EC-14, pp. 863-867, Dec. 1965.
4. Dean, K. J. and Sc., M., "Design of Binary Logarithm Generators", IEEE Proceeding Instrumentation Electrical Engineering, Vol. 115, No. 8, pp. 1118-1120, 1968.
5. Edger, A. D. and Lee, S. C., "Focus Microcomputer Number Systems", Communications of the ACM, Vol. 22, pp. 166-177, Mar. 1979.
6. Frangakis, G. P. and Sc., M., "A New Binary Logarithm-Based Computing System", IEEE Proceedings, Vol. 130, Pt. E, No. 5, pp. 169-173, Sept. 1983.
7. Frangakis, G. P., "Fast Binary Logarithm Computing

Circuit for Binary Number Less than One", Electroinics Letters, Vol. 16, No. 15, pp. 574-575, July 1980.

8. Frey, M. L. and Taylor, F. J., "Table Reduction Technique for Logarithmically Architected Digital Filters", IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-33, pp. 718-719, June 1985.

9. Hall, E. L., Lynch, D. D., and Dwyer, S. A. III, "Generation of Products and Quotients Using Approximate Binary Logarithms for Digital Filtering Applications", IEEE Trans. on Computers, Vol. C-19, pp. 97-105, Feb. 1970.

10. Kan, E. P. F. and Aggarival, J. K., "Error Analysis of Digital Filter Employing Floating-Point Arithmetic", IEEE Trans. on Circuit Theory, Vol. CT-18, No. 16, pp. 678-685, Nov. 1971.

11. Kingsbury, N. G. and Rayner, F. J. W., "Digital Filtering Using Logarithm Arithmetic", Electronics Letters, Vol. 7, pp. 56-58, Feb. 1971.

12. Kuo, Benjamin C., Digital Control System, Holt, Rinehart and Winston Inc., pp. 12-19, 1980.

13. Kurokawa, T., Payne, J. A., and Lee, S. C., "Error Analysis of Recursive Digital Filters Implemented with Logarithmic Number Systems", IEEE Trans. on Acoustics,

Speech, and Signal Processing, Vol. ASSP-28, pp. 706-715, Dec. 1980.

14. Lang, J. H., Zukowski, C. A., Lamaire, R. O., and An, C. H., "Integrated Circuit Logic Arithmetic Units", IEEE Trans. on Computers, Vol. C-34, pp. 475-482, May 1985.

15. Lee, S. C. and Edgar, A. D., "The Focus Number System", IEEE Trans. on Computers, Vol. C-26, No. 11, pp. 1167-1170, Nov. 1977.

16. Lo, Hao-Yung, "Binary Logarithms for Computing Integral and Non-Integral Roots and Powers", International Jounal of Electronics, Vol. 40, No. 4, pp. 357-364, April 1976.

17. Lo, Hao-Yung and Aoki, Yoshinao, "Generation of a Precise Binary Logarithm with Differential Grouping Programming Logic", IEEE Trans. on Computers, Vol. C-34, pp. 681-691, Aug. 1985.

18. Majithia, J. C. and Levan, D., "A Note on Base-2 Logarithm Computations", Proceedings IEEE (Lett.), Vol. 61, pp. 1519-1520, Oct. 1973.

19. Marino, D., "New Algorithms for the Approximation Evaluation in Hardware of Binary Logarithm and Elementary Functions", IEEE Trans. on Computers, Vol. C-19, pp. 1416-1421, Dec. 1972.

20. Mitchel, J. N. Jr., "Computer Multiplication and Division Unsing Binary Logarithm", IRE Trans. on Electroinc Computers, Vol. EC-11, pp. 512-517, Aug. 1962.

21. Swartzlnader, E. E. Jr. and Alexopoulos, A. G., "Sign/ Logarithm Number System", IEEE Trans. on Computers, Vol. C-29, pp. 1238-1242, Dec. 1975.

22. Swartzlander, E. E. Jr.,Chandra, D. V. S., Nagle, H. T. Jr., and Starks, S. A., "Sign/Logarithm Arithmetic for FFT Implementation", IEEE Trans. on Computers, Vol. C-32, pp. 526-534, June 1983.

23. Taylor, F. J., Grill, R., Joseph, J., and Radke, J., "A 20 Bit Logarithmic Number System Processor", IEEE Trans. on Computers, Vol. 37, No. 2, pp. 190-200, Feb. 1988.

24. Taylor, F. J., "An Extended Precision Lograithmic Number System", IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-31, No. 1, Feb. 1983.

25. Tzafestas, S. G. and Frangakis, G. P., "Binary Logarithm-Based Computing Systems: Application to Digital Filter", Digital Techniques, in Simulation and Control, S. G. Tzafestas (Ed.), Elsevier Science Publishers B. V. (North-Holland), 1985.

26. Vainio, O. and Neuvo, Y., "Logarithmic Arithmetic in

FIR Filters", IEEE Trans. Circuits and Systems, Vol. CAS-33, pp. 826-828, Aug. 1986.

ACKNOWLEDGEMENTS

The author wishes to express his appreciation and thanks to Dr. Satish D. V. Chandra, major advisor, for his guidance and helpful suggestions and encouragement. The counsel, patience, and direction have been invaluable.

Thanks are due to author's committee members, Dr. Gary L. Johnson and Dr. Chi-Lung Huang for their constructive criticism.

Thanks are also due to author's parents and Elaine Song for thier moral support and encouragement during the prepartation of this report.

HARDWARE IMPLEMENTATION OF THE BASE TWO

LOGARITHMIC NUMBER SYSTEM

by

Steve Chih Hsiung

B.Ed., National Kaohsiung Teachers' College (Taiwan), 1981

M.S., University of North Dakota, 1986

AN ABSTRACT OF A MASTER'S REPORT

Submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical and

Computer Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1988