석사학위논문

Master's Thesis

휴대용 3D 그래픽 시스템을 위한 Logarithmic Arithmetic Unit의 연구

A Logarithmic Arithmetic Unit Design for Mobile 3D Graphics System

김 혜 정 (**金 惠 貞** Kim, Hyejung)

전자전산학과 전기 및 전자 공학 전공

Department of Electrical Engineering and Computer Science,

Division of Electrical Engineering

한 국 과 학 기 술 원

Korea Advanced Institute of Science and Technology

2006

휴대용 3D 그래픽 시스템을 위한 Logarithmic Arithmetic Unit의 연구

A Logarithmic Arithmetic Unit Design for Mobile 3D Graphics System

A Logarithmic Arithmetic Unit Design for Mobile 3D Graphics System

Advisor : **Professor Hoi-Jun Yoo**

by

Hyejung Kim

Department of Electrical Engineering and Computer Science
Division of Electrical Engineering
Korea Advanced Institute of Science and Technology

A thesis submitted to the faculty of the Korea Advanced Institute of Science and Technology in partial fulfillment of requirements of the degree of Master of Engineering in the Department of Electrical Engineering and Computer Science, Division of Electrical Engineering

Daejeon, Republic of Korea
2006. 6. 13
Approved by

Professor Hoi-Jun Yoo
(Major Advisor)

휴대용 3D 그래픽 시스템을 위한 Logarithmic Arithmetic Unit의 연구

김 혜 정

위 논문은 한국과학기술원 석사학위논문으로 학위논문 심사위원회에서 심사 통과하였음.

2006년 6월 13일

심사위원장 유 회 준 (인)

심사위원 박 인 철 (인)

심사위원 신 영 수 (인)

MEE

20043175

김 혜 정, Hyejung Kim. A Logarithmic Arithmetic Unit Design for Mobile 3D Graphics System. 휴대용 3D 그래픽 시스템을 위한 Logarithmic Arithmetic Unit의 연구. Department of Electrical Engineering and Computer Science, Division of Electrical Engineering. 2006. 50p. Advisor Prof. Yoo, Hoi-Jun. Text in English

Abstract

A 32-bit fixed-point logarithmic arithmetic unit is proposed for the possible application to mobile 3D graphics system. The proposed logarithmic arithmetic unit performs division, reciprocal, square-root, reciprocal-square-root and square operations in 2 clock cycles, and powering operation in 4 clock cycles. It can program its the number range for accurate computation flexibility of 3D graphics pipeline, and 8-region piecewise linear approximation model for logarithmic and anti-logarithmic conversion to reduce the operation error under 0.2%. Its test chip is implemented by 1-poly 6-metal 0.18um CMOS technology with 9k gates. It operates at the maximum frequency of 231MHz and consumes 2.18mW at 1.8V supply.

사랑하는 가족에게 바칩니다.

Table of Contents

List of Figures

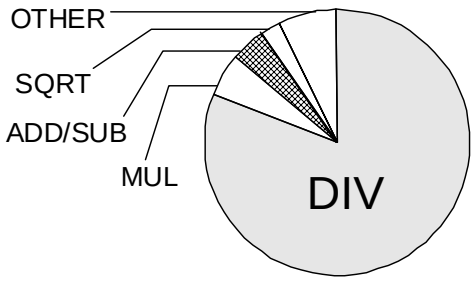
List of Tables

CHAPTER 1

INTRODUCTION

1.1 Motivation

Nowadays, the real-time 3D graphics is one of the attractive applications for mobile systems [1]. As 3D graphics is getting more and more familiar to the people, there have been increasing demands for high quality graphics for new applications such as avatar, advertisement and games. Many 3D graphics processors have been studied [1]-[2] to meet these requests. The mobile system has low resolution and small screen size than PC system. However, the mobile system also has limited resources such as CPU performance, memory capacity and battery energy. Most of mobile systems use low power 32-bit processors such as ARM or MIPS, and the fixed point arithmetic units have been used for lower power consumption since they consume less power than floating point units [2].



	DIV	80.95%
	MUL	5.11%
	ADD/SUB	4.18%
	SQRT	2.65%
	OTHER	7.11%

Figure 1.1 Percentage of Processing Time of the Operations in the 3D Graphics Rendering Pipeline [3]

The 3D graphic processors require heavy arithmetic calculations like division, reciprocal, square-root, square and powering operations in contrast to general processors. The figure 1.1 shows the percentage of processing time of 3D graphics rendering pipeline, and the heavy arithmetic functions take 83% of total processing time [3]. Of course these functions consume most of computing power because they use most of the clock cycles in the real time 3D graphics systems. And for the low power consumption the clock cycles of these complex functions should be reduced as much as possible.

The LNS (Logarithmic Number System) has been studied to simplify

arithmetic computations for lower computation complexity, high computation speed, and small gate counts [5]-[12]. Although there is the conversion overhead from integer to logarithm or logarithmic to integer, the overhead is much smaller than that of the conventional computation hardware. For this reason there were trials to use the LNS for DSP applications [5]-[6]. However, they performed the addition and subtraction operations in nonlinear function like LNS which make them more complicated and result in slow operation speed and high power consumption.

Since Mitchell introduced the binary logarithmic converting algorithm [7], the linear approximation algorithms for logarithmic arithmetic have been studied to minimize the error in the hardware implementation [7]-[12]. In the approximation methods, usually sizable errors occur during logarithmic and antilogarithmic converting. However, their operation errors have not been carefully examined in relation to its real applications. Although many approximation methods have been proposed for error reduction, their errors are still too large to be used for 3D graphics system. In this paper we propose a new algorithm to reduce the logarithmic converting error and antilogarithmic converting error. In addition, we propose a logarithmic arithmetic unit with reduced hardware complexity by controlling its number range for

fixed-point 3D graphics applications while its speed and power consumption are improved. It can reduce the area and the power consumption to be suitable for mobile application. Moreover, a hybrid method which performs complex operations in LNS and simple addition/subtraction operations in fixed-point number system, is proposed for simple implementation of arithmetic functions with low cost. The proposed 32-bit logarithmic arithmetic unit is verified by fabrication and measurements of the real chip and later its detail design and measurement results will be explained.

1.2 Thesis Organization

The organization of this paper is as follows. The brief 3D graphics pipeline will be introduced in Chapter 2. The proposed 32-bit logarithmic arithmetic unit will be discussed in Chapter 3. In addition, in this section the small error logarithmic and antilogarithmic conversion algorithms will be described. The evaluation results, the implementation results and measurement results will be shown in Chapter 4. Finally, the conclusion of our work will be summarized in Chapter 5.

CHAPTER 2

3D GRAPHICS PIPELINE

2.1 3D Graphics Pipeline

The full 3D graphics pipeline can be considered as two stages, the geometry and the rendering stage for the vertex processing and the pixel processing, respectively [13]. The figure 2.1 shows the overall 3D graphics pipeline.

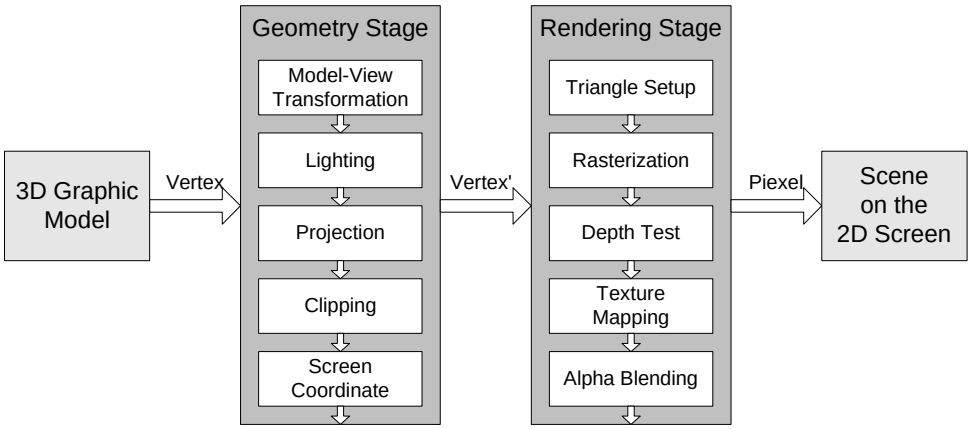


Figure 2.1 The Overall 3D Graphics Pipeline

The 3D graphic model is entered into 3D graphics pipeline by vertex format. First, the Geometry stage operates model-view transformation which incorporates of translation, rotation, scaling and shifting operations. After the transformation, the lighting operation is performed. The lighting operation is defined by the vertex material properties and the light sources. The intensity term is composed of ambient, diffuse, specular and emissive terms. The projection and the clipping operations are performed at next stage which are defined view frustum and the clipped the vertex outside the view frustum. At last, the each coordinate component is divided by the w-component to transform the vertex from the homogeneous coordinate space into Euclidean space.

The new vertexes outputs are used rendering stage inputs. The rendering stage performs triangle setup which finds the triangle point, edge and the scan line. The rasterization fills the color of the triangle by interpolation method. The depth test is performed to remove hidden pixels in a polygon and the texture mapping wrap a 3D model object with prepared 2D texture images. The alpha blending operation blends the new pixel value with the previous pixel value. The pixel values are generated after rendering stage and that pixels are rasterized on the 2D screen.

2.2 Special Operations for 3D Graphics System

The 3D graphic processors require special arithmetic calculations in contrast to general applications.

The matrix multiplications for the model view transformation operations. The multiplication of 4 by 4 matrix with 4 vectors are used for translation, scaling and rotation operations. The general matrix multiplication format is shown in equation (1).

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} C_{00} & C_{01} & C_{02} & C_{03} \\ C_{10} & C_{11} & C_{12} & C_{13} \\ C_{20} & C_{21} & C_{22} & C_{23} \\ C_{30} & C_{31} & C_{32} & C_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \qquad \text{Eq. (1)}$$

Equation (2) shows the general lighting calculation format. The squaring, reciprocal-square-root, division and the inner product operations are necessary for the calculations of lighting.

$$color = \left(\frac{(L_x, L_y, L_z)}{\sqrt{L_x^2 + L_y^2 + L_z^2}} \right) \bullet \left(\frac{(n_x, n_y, n_z)}{\sqrt{n_x^2 + n_y^2 + n_z^2}} \right) \qquad \text{Eq. (2)}$$

The powering operation like equation (3) is necessary for the specular lighting.

$$specular = x^a \qquad \text{Eq. (3)}$$

In the rendering stage, lots of division operations such as equation (4) are performed on each pixels for there location and color informations and the texture mapping. This division operation is not complex as much as lighting calculation, but is a burden because every pixel of every scene need division operations.

$$(x', y', z', w') = \frac{(x, y, z, w)}{\Delta x} \qquad \text{Eq. (4)}$$

CHAPTER 3

PROPOSED SYSTEM DESIGN

3.1. 32-bit Logarithmic Arithmetic Unit

The Logarithmic Arithmetic Unit (LAU) computes the complex functions such as multiplication, division and square-root by using only simple addition, subtraction and shift operations. When $X=\log_2 x$ and $Y=\log_2 y$, the table 3.1 summarizes the normal operations and their equivalent logarithmic operations. On the contrary to complex functions, simple addition and subtraction need more complex non linear computations in LNS. In this study, we combine two different number systems into unified arithmetic unit for better performance: Simple addition/subtraction in fixed-point number system and complex functions in LNS.

Table 3.1 Operations in Logarithmic Number System.

Operation		Normal Arithmetic	Logarithmic Arithmetic
Multiplication	**MUL**	$z = x \cdot y$	X+Y
Division	**DIV**	$z = x / y$	X-Y
Reciprocal	**RCP**	$z = 1 / x$	-X
Square Root	**SQRT**	$z = \sqrt{x}$	X >> 2
Reciprocal Square Root	**RSQ**	$z = 1 / \sqrt{x}$	-X >> 2
Square	**SQR**	$z = x^2$	X << 2
Powering	**POW**	$z = x^y$	$Y + \log_2 X$
Addition	**ADD**	$z = x + y$	$X + \log_2(1+2^{Y-X})$
Subtraction	**SUB**	$z = x - y$	$X + \log_2(1-2^{Y-X})$

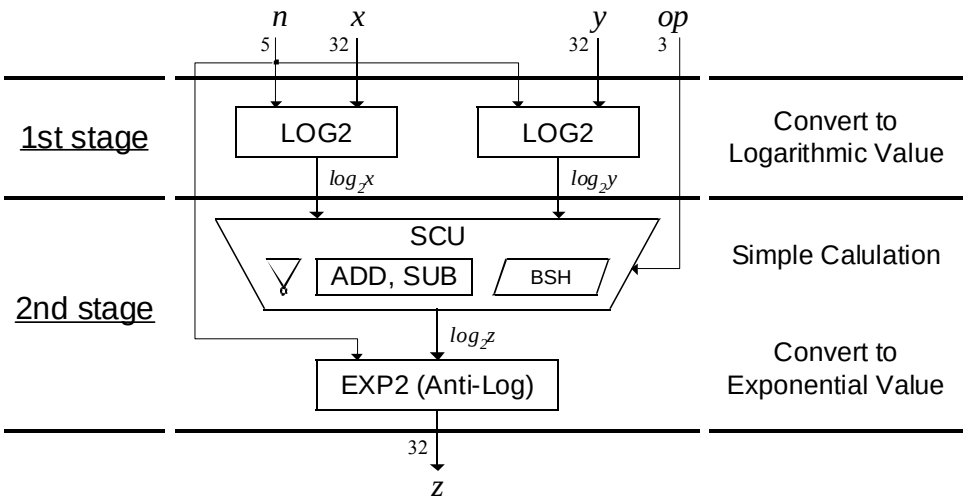


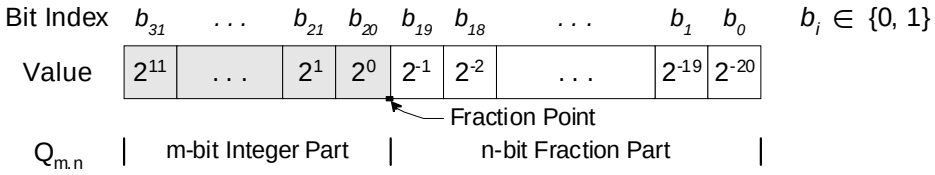
Figure 3.1 Top Architecture of the LAU

The top architecture of the proposed LAU is shown in figure 3.1. The unit is composed of two LOG2s (binary logarithmic converter) in the first stage, SCU (Simple Calculation Unit), and an EXP2 (binary antilogarithmic converter) at the second stage. SCU is composed of an inverter, an ADD/SUB (adder/ subtracter) and a BSH (barrel shifter). *x* and *y* are the operand for the LAU. *n* decides the number range of each computation, and *op* selects the required operation. LAU is pipelined for fast operation at high clock frequency. Since the logarithmic converter takes longer time than the exponential converter does, the SCU is located in the second stage to distribute the time budget to each pipeline stage evenly. So the LAU performs the functions given in table-I in 2-cycle except for the powering operation, x^y. More explanation on the powering operation will be given in Chapter 4. In addition, if single operand computations like square-root operation are performed, the second logarithmic converter is turned off by the clock gating to reduce the power consumption.

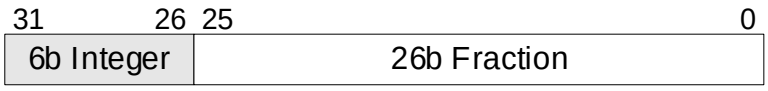
Usually there are two reasons why the fixed-point number system is used instead of floating-point number system [2]. First the hardware architecture of the fixed-point arithmetic is much simpler than that of

the floating-point arithmetic because the fixed-point arithmetic uses only integer datapath. Second, fixed-point can operate at higher clock frequency. Each stage of the 3D graphics pipeline requires its own number range for accuracy optimization. Therefore, the programmable number range which can vary the number range dynamically is necessary even if the fixed-point arithmetic is used for the flexibility of the number system hardware.

The variable number range is denoted as Qm.n, where 'm' is the number of bits representing the integer part and 'n' is the number of bits representing the fractional part. The figure 3.2(a) shows the details



(a) Number Format (Example : Q12.20 Format)



(b) Internal Fixed Number Range Q6.26 of LAU

Figure 3.2 Fixed-Point Number Format.

of the number format of Qm.n. The LAU supports the programmable number range of which input and output interfaces are varied by the external 5-bit input *n*. But the internal number range of LAU is fixed to Q6.26. as shown in figure 3.2(b) to reduce the calculation complexity and latency. After LAU calculation, the output number range is modified to Qm.n. which is suitable to programmer's convenience and the number system compatibility. Since the 32-bit fixed-point input range can be $2^{-32}-1 \leq x \leq 2^{31}-1$ which is equivalent to $-32.0 < log_2 x < 32.0$, the 6-bit is needed to represent the 2's complement signed integer value. The 6 most significant digits of $log_2 x$ represent the integer part and remaining 26 bits represent the fractional part.

3.2 Logarithmic Converter Block

In general, the piecewise interpolation methods were used in the binary logarithm conversion algorithms [7]-[11]. They usually used 2-6 regions piecewise linear methods. Their error ranges are shown in table 3.2. The minimum error with number range of Q32.0 of previous work is 0.31% [7]-[11].

Table 3.2 Comparison of Logarithmic Approximation Errors.

	Mitchell [7]	SanGreg-ory [8]	Combet [9]	Hall [10]	Abed [11]	This work
Max. +error (%)	5.361	0.431	0.185	0.126	0.154	0.015
Max. -error (%)	0	-1.540	-0.267	-0.718	-0.153	-0.049
Error Range (%)	5.361	1.191	0.452	0.844	0.307	0.064

The optimal number of piecewise regions and the coefficients are selected by simulation. Since affecting on the hardware complexity, the two terms mentioned above should be carefully selected. Figure 3.3 shows the variation of the error range according to the number of

piecewise region and the number of coefficients. The error rate are reduced by increasing the number of regions and the coefficients. However more than 3 coefficients don't affect the error rate, but increase the hardware size. Therefore the 8-region 3-coefficient are the optimal point.

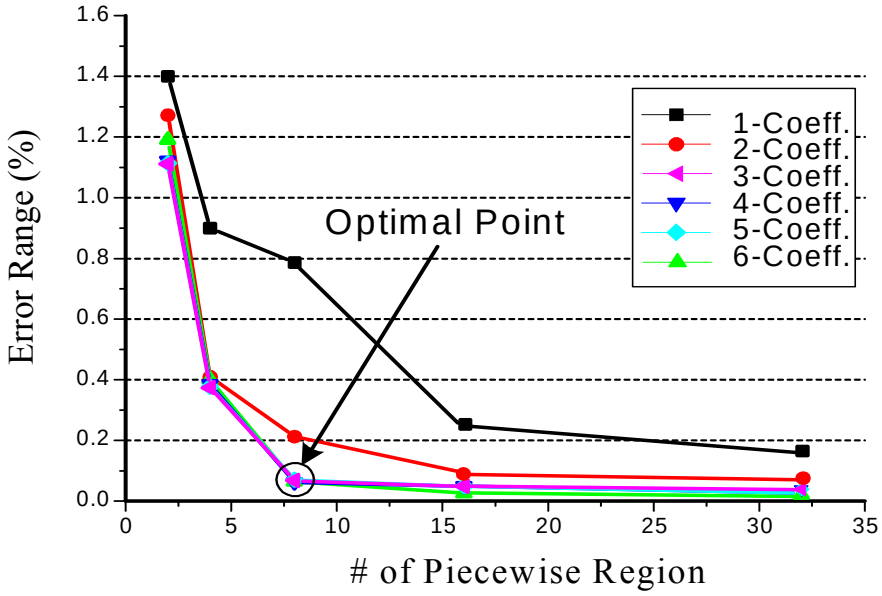


Figure 3.3 The Error Range According to The Number of Piecewise Regions and The Number of Coefficients

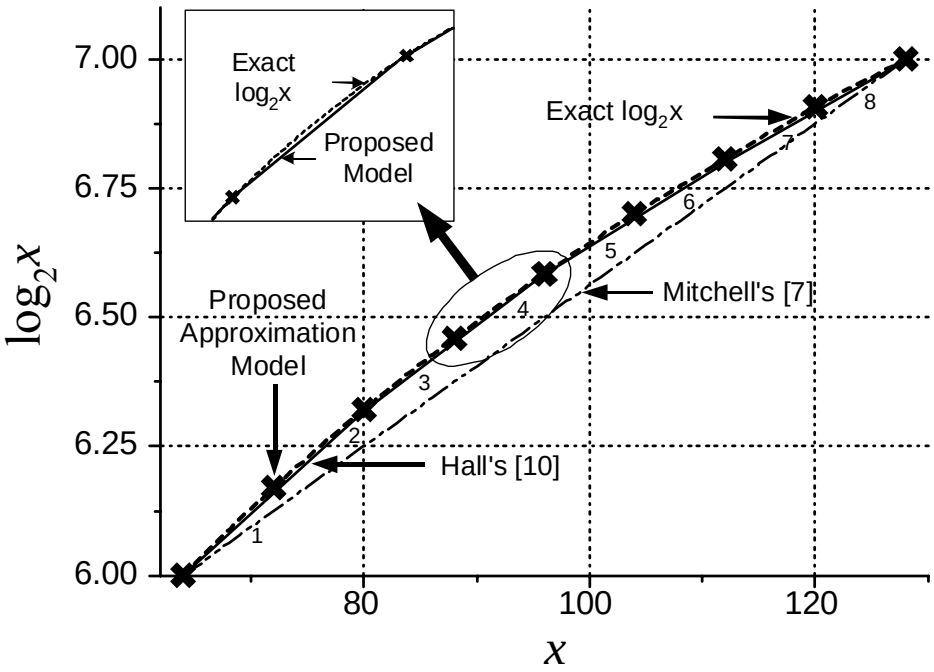


Figure 3.4 Comparison Results of Approximation Value

In this study, we divide the fraction part into 8-region to further reduce its error rate, and use the straight linear interpolation in each region. Figure 3.4 shows the proposed algorithm which tracks the exact value much better than Mitchell's [7] and Hall's [10] methods.

The binary logarithmic conversion algorithm is first presented by Mitchell [7]. The modified logarithmic conversion algorithm can be summarized as follows:

Let x be a fixed-point 32-bit input which has the variable number range of Qm.n. Its value can be written as equation (5).

$$x = 2^k(1+f) \qquad \text{Eq. (5)}$$

k is the characteristic value of the logarithm in Mitchell's equation [7] and n is the number range decision value. Executing the variable number range operation, the characteristic value k is replaced to the new value of k' which is equal to k-n. f is the fraction parts in the range [0, 1] located in the right side of the leading-one bit. Taking the binary logarithm for both sides of the equation (5), equation (6) and (7) can be obtained.

$$\log_2 x = k'+n+\log_2(1+f), \quad where\ k' = k-n \qquad \text{Eq. (6)}$$

$$\log_2(1+f) \cong \alpha_i \cdot f + \beta_i, \quad where \quad i = 0, 1, ..., 7. \qquad \text{Eq. (7)}$$

$\log_2(1+f)$ is the fractional part after the binary logarithmic conversion. This term is calculated by 8 region piecewise interpolation represented in equation (7). α_i and β_i are the coefficients which have 7-bit and

10-bit resolution, respectively, and decide the slope of the piecewise linear interpolation. Since the fractional part is divided into eight regions, eight different coefficients are necessary for eight regions. Each coefficient is obtained through fitting the equation (7) to the real value by varying α_i and β_i in order to reduce the complexity because they have a direct effect on the hardware implementation. Since the coefficients are the fraction numbers with powers of 2 as their denominators, they can be calculated by only shifters and the adders. When defining the coefficients, the error range which should be considered too. Table 3.3 shows the optimized values of α_i and β_i. Equations (8) describe the detail logarithmic piecewise expressions.

Table 3.3 Coefficient of Logarithmic Approximation Model.

f	α	β	f	α	β
[0.0, 1/8)	175/128	0	**[4/8, 5/8)**	119/128	123/1024
[1/8, 2/8)	155/128	20/1024	**[5/8, 6/8)**	110/128	167/1024
[2/8, 3/8)	142/128	46/1024	**[6/8, 7/8)**	102/128	215/1024
[3/8, 4/8)	129/128	84/1024	**[7/8, 8/8)**	95/128	264/1024

Eq. (8.a) ~ Eq. (8.h)

$$f' = f + (f >> 1) + (\bar{f} >> 3) + (\bar{f} >> 7) \qquad for\ f \in [0, 1/8)$$

$$f' = f + (f >> 1) + (\bar{f} >> 2) + (\bar{f} >> 6) + (15/1024) \qquad for\ f \in [1/8, 2/8)$$

$$f' = f + (f >> 2) + (\bar{f} >> 3) + (\bar{f} >> 6) + (46/1024) \qquad for\ f \in [2/8, 3/8)$$

$$f' = f + (\bar{f} >> 7) + (91/1024) \qquad for\ f \in [3/8, 4/8)$$

$$f' = f + (\bar{f} >> 3) + (f >> 4) + (\bar{f} >> 7) + (123/1024) \qquad for\ f \in [4/8, 5/8)$$

$$f' = f + (\bar{f} >> 3) + (\bar{f} >> 6) + (167/1024) \qquad for\ f \in [5/8, 6/8)$$

$$f' = f + (\bar{f} >> 2) + (f >> 4) + (\bar{f} >> 6) + (215/1024) \qquad for\ f \in [6/8, 7/8)$$

$$f' = f + (\bar{f} >> 2) + (\bar{f} >> 7) + (264/1024) \qquad for\ f \in [7/8, 1)$$

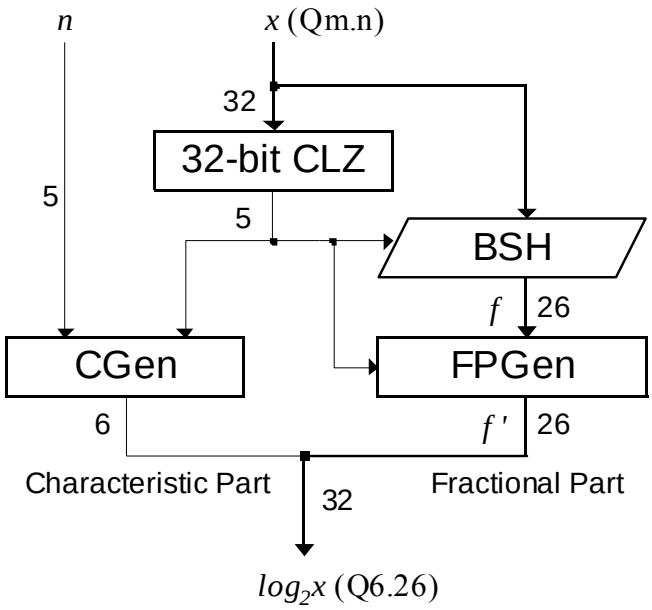


Figure 3.5 Architecture of Logarithmic Converter

Figure 3.5 shows the proposed architecture of the logarithmic converter block. *x* is an operand to be converted into the logarithmic number, and *n* is the number range selection bit. Logarithmic converter block is composed of 32-bit CLZ (Count Leading Zero), BSH (barrel shifter), CGen (Characteristic Generator) and FPGen (Fractional Part Generation block). The variable input number range Qm.n. is modified to the fixed number range of Q6.26. at the end of the logarithmic converter. CLZ block calculates the number of the leading zero bits of

the input. The 5 bits of the CLZ block output decide the characteristic value and the amount of shift value of BSH. BSH converts the input number range of Qm.n. to Q6.26. After shifting operation, the fractional part is generated by FPGen block and the characteristicpart is generated by CGen block. The above two values are combined to give the logarithmic conversion result.

Since the delay time of FPGen is the longest, it is important to optimize the FPGen to get the high operating frequency. The detailed architecture of the proposed FPGen is shown in figure 3.6. FPGen generates the approximated fractional value of the equation (7) and table 3.3. It is implemented by hardwired shifter, MUX, CSA (Carry Save Adder) and CPA (Carry Propagating Adder). f is the original fractional part value and f' is the modified value. β is the approximation coefficients and *s0-s3* are selection bits made by 3 most significant bits of *f*. *x, y* and *z* are compensation value of the slope for accuracy which are selected by the MUXes. The final result f' is calculated by adding these compensation values to the value of original fraction part *f*. Although the fractional part has very high resolution, the coefficients α and β are selected to minimize the number of

MUXes and the adders in order to reduce the latency and power consumption. To calculate the 2's complement subtraction operation the inverters and the adders are necessary. In this study, the coefficients consist of only the addition operations so that the internal overhead of the inverters and the adders can be removed.

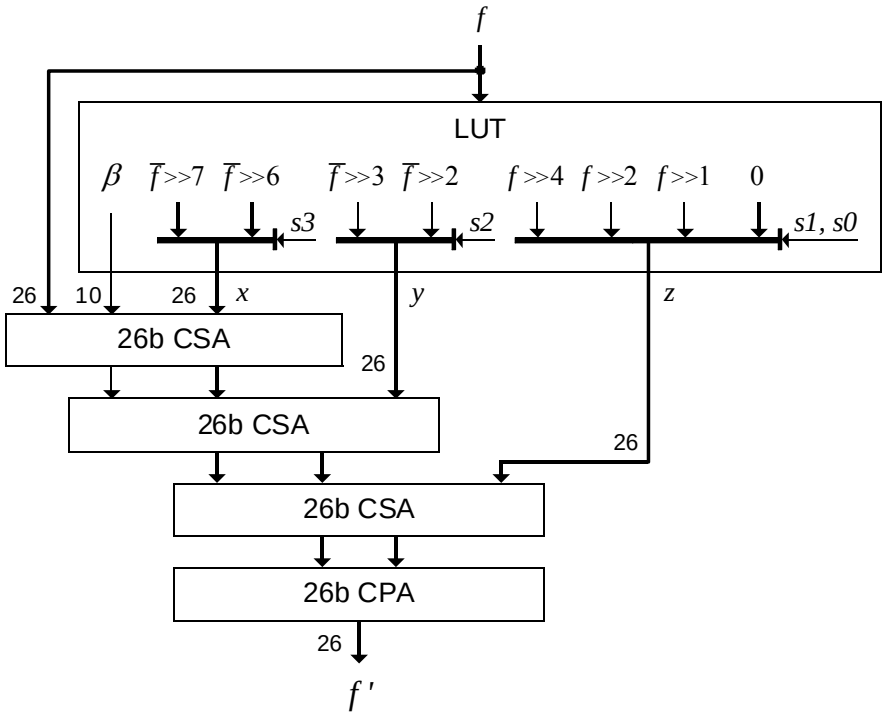


Figure 3.6 Architecture of FPGen (Fractional Part Generation) Block

The figure 3.7 show the graphs of the error range and the number of gate counts according to the number of CSAs of FPGen. The error range of logarithmic converter is significantly reduced until the 3 CSAs. On the other side, the gate counts increase as increasing the CSAs. The 3 CSAs are the optimal point on both logarithmic and anti-logarithmic converters.

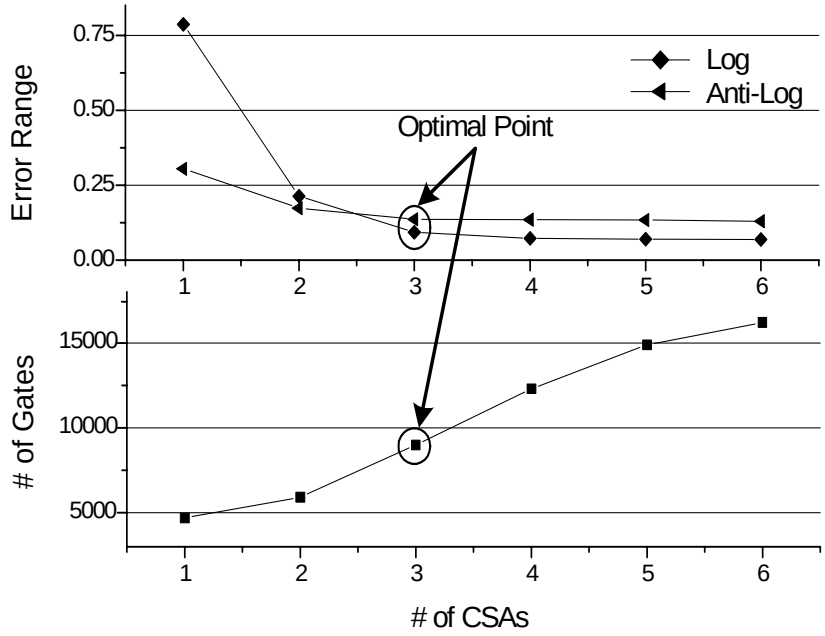


Figure 3.7 The Error Range and The Gate Counts According to the Number of CSAs.

The error of the logarithmic converting algorithm, $error_{log}$, is given as equation (9).

$$\begin{aligned} error_{\log} &= \log_2 x - (\log_2 x)' \\ &= \log_2(1+f) - (\alpha \cdot f + \beta) \end{aligned} \qquad \text{Eq. (9)}$$

The maximum percent error range is $-0.049 < error_{log} < 0.015$ and figure 3.8 shows the error graph. The error range of the proposed method is the smallest among those of any other reported methods [7]-[11] because of its finely divided range and the optimized coefficients.

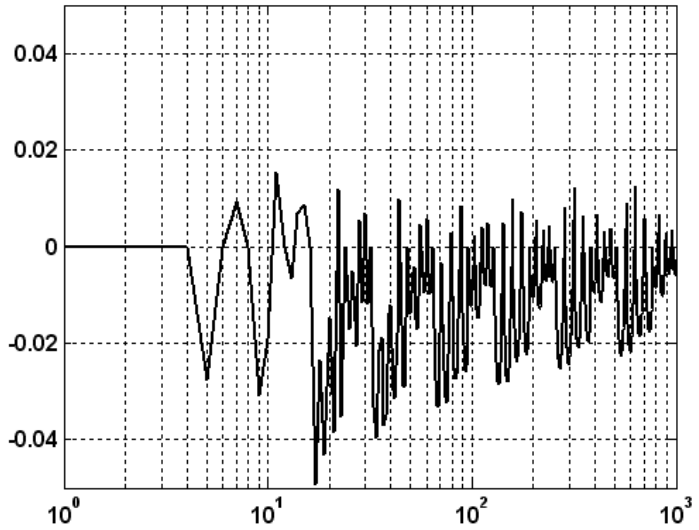


Figure 3.8 The Percent Error of Proposed Logarithmic Converter

The proposed logarithmic converter operates at the maximum frequency of 260MHz as a simulation result which is equivalent to 42FO4. It is much faster than the conventional scheme. For example, the simulation operating frequency of the conventional logarithmic converter [11] was 55MHz in 0.6um technology which is equivalent to 61FO4. The critical path delay and the maximum error of the logarithmic converter are reduced by 31% and 79.2%, respectively.

3.3 Antilogarithmic Converter Block

In general, the piecewise interpolation method is also used for antilogarithmic converting algorithm [12]. The fixed-point representation of the antilogarithm is composed of six most significant bits representing the integer part 2^k and remaining 26 bits of the fractional part representing 2^f as given in the equation (10). This value is divided into two parts, according to the positive or the negative input x.

$$2^x = 2^{k+f} = \begin{cases} 2^k \cdot 2^f & x \geq 0 \\ 2^{k-1} \cdot 2^{1-|f|} & x < 0 \end{cases}$$
$$= 2^{k'+f'} = 2^{k'} \cdot 2^{f'} \qquad \text{Eq. (10)}$$

k and f values are replaced when x is negative value in order to make fractional part positive value. So k and f are modified to k' and f' as equation (11) and (12).

$$k' = \begin{cases} k, & x \geq 0 \\ k-1, & x < 0 \end{cases} \qquad \text{Eq. (11)}$$

$$f' = \begin{cases} f, & x \geq 0 \\ 1-|f|, & x < 0 \end{cases}, \quad \textit{where } 0 \leq f' < 1 \qquad \text{Eq. (12)}$$

$$k'' = k' + n - 26, \qquad \text{Eq. (13)}$$

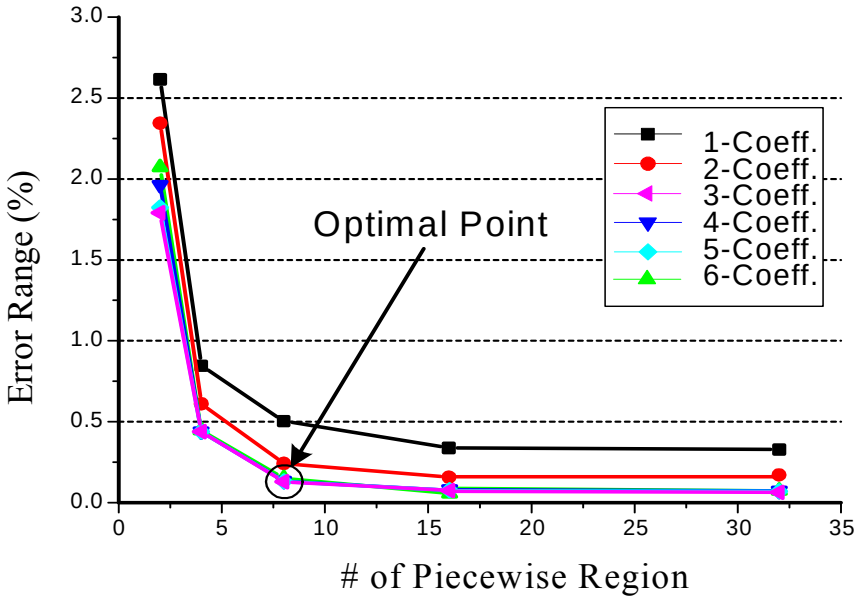


Figure 3.9 The Error Range According to The Number of Piecewise Regions and The Number of Coefficients

Figure 3.9 shows the variation of the error range according to the number of piecewise region and the number of coefficients. Increasement of the number of regions and the coefficients reduce the error range, but increase the hardware cost. The error rate is saturate from 8-region, 3-coefficient point, so that is the optimal point of antilogarithmic algorithm. The fractional part is divided by 8-region, and it is described as the equation (14)

$$2^{f'} \cong \alpha_i \cdot f + \beta_i, \quad where\ i = 0, 1, ..., 7. \qquad \text{Eq. (14)}$$

Executing the variable number range, the final integer part result is modified to *k''* which is equals to *k'+n-26* by the number range selection value *n*. The fractional part $2^{f'}$ is calculated in piecewise interpolation represented by equation (14). Because the output is produced by shifting the fractional value by integer number, it is very important to generate the accurate fractional value. The coefficients for fractional part are shown in table 3.4.

Table 3.4 Coefficient of Antilogarithmic Approximation Model

f	α	β	f	α	β
[0.0, 1/8)	92/128	1024/1024	**[4/8, 5/8)**	132/128	924/1024
[1/8, 2/8)	93/128	1015/1024	**[5/8, 6/8)**	143/128	864/1024
[2/8, 3/8)	111/128	995/1024	**[6/8, 7/8)**	155/128	792/1024
[3/8, 4/8)	121/128	964/1024	**[7/8, 8/8)**	169/128	295/1024

Equations (15) describe the detail anti-logarithmic piecewise expressions.

Eq. (15.a)-(15.h)

$$f' = f + (\bar{f} >> 2) + (\bar{f} >> 5) + (1024/1024) \quad for\ f \in [0, 1/8)$$

$$f' = f + (\bar{f} >> 2) + (f >> 5) + (f >> 7) + (1015/1024) \quad for\ f \in [1/8, 2/8)$$

$$f' = f + (\bar{f} >> 3) + (\bar{f} >> 7) + (995/1024) \quad for\ f \in [2/8, 3/8)$$

$$f' = f + (\bar{f} >> 4) + (f >> 7) + (964/1024) \quad for\ f \in [3/8, 4/8)$$

$$f' = f + (f >> 5) + (\bar{f} >> 7) + (924/1024) \quad for\ f \in [4/8, 5/8)$$

$$f' = f + (f >> 3) + (\bar{f} >> 7) + (864/1024) \quad for\ f \in [5/8, 6/8)$$

$$f' = f + (f >> 2) + (\bar{f} >> 5) + (\bar{f} >> 7) + (792/1024) \quad for\ f \in [6/8, 7/8)$$

$$f' = f + (f >> 2) + (f >> 4) + (f >> 7) + (695/1024) \quad for\ f \in [7/8, 1)$$

Figure 3.10 shows the architecture of the antilogarithmic converter block. *x* is an operand to be converted into antilogarithmic number, and *n* is the number range selection bit as it is in the logarithmic converter.

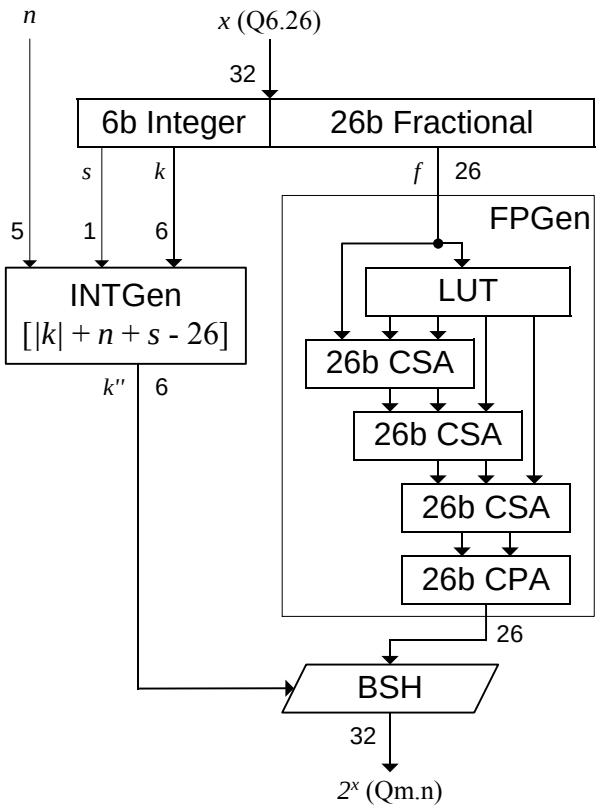


Figure 3.10 Architecture of Antilogarithmic Converter

The antilogarithmic converter is composed of INTGen (INTeger part Generation block), FPGen (Fractional Part Generation block) and BSH (Barrel Shifter). The computation time is shorter than logarithmic converter because the integer part and fractional part are calculated separately. INTGen calculates the integer part by using simple addition

of x, sign bit and n.

FPGen is composed of LUT (Look Up Table), 3 CSAs(Carry Save Adder) and a CPA (Carry Propagation Adder) similar to the logarithmic converter's. The 3 CSAs are also the optimal point of anti-logarithmic converters by figure 3.7. The final BSH shifts the fractional part value by the result value of INTGen block and then makes the output number range from Q6.26 to Qm.n as specified by the number range decision input n.

The error of the antilogarithmic converting algorithm, $error_{antilog}$, is equal to equation (16).

$$\begin{aligned} error_{anti\log} &= 2^x - (2^x)' \\ &= 2^k[2^f - (\alpha \cdot f + \beta)] \end{aligned} \tag{16}$$

The maximum percent error range is -0.070 < $error_{antilog}$ < 0.082 and figure 3.11 shows its error graph. The proposed 8-region interpolation algorithm is compared to the previous works of the antilogarithm converters [12] in table 3.5. The maximum error range of the antilogarithmic converter is reduced by over ten times.

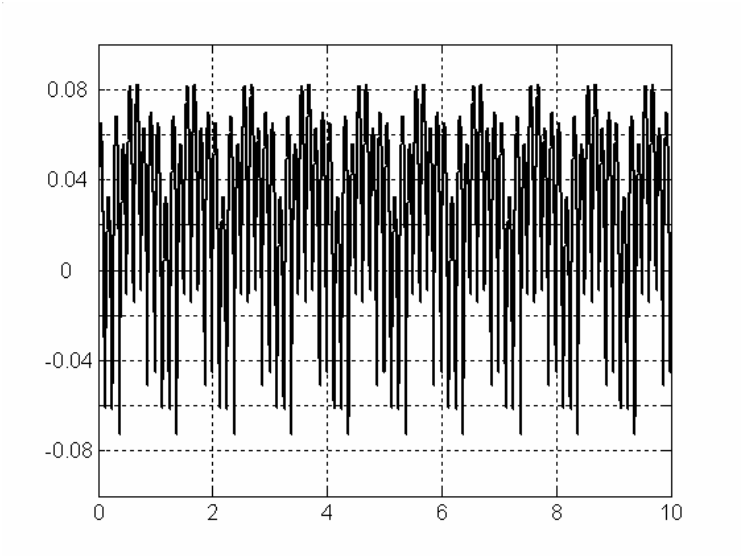


Figure 3.11 The Percent Error of Proposed Antilogarithmic Converter

Table 3.5 Comparison of Antilogarithmic Approximation Error [12].

	Mitchell's Straight-line	Hall's 4-equation	Abed's 7-region	This work
Max. +error	6.1476	0.3032	0.3477	0.082
Max. -error	0	-0.4736	-0.5786	-0.070
Error Range (%)	6.1476	0.7768	1.5358	0.152

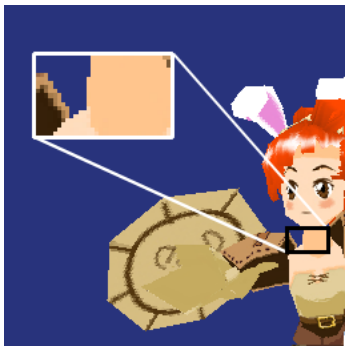
CHAPTER 4

RESULTS

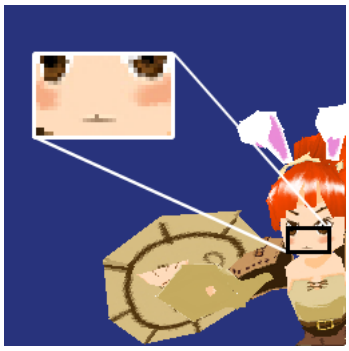
4.1 Evaluation Result

The proposed LAU is verified in 3D graphics processing software environment before its chip is implemented. Figure 4.1 is the test scene and the in-box shows a zoomed image for the accuracy comparison. The test model consists of 1,700 polygons with lighting and texture mapping. The screen resolution is 512x512 and the texture size is 256x256. The variable number range Qm.n are used to evaluate the scene. Figure 4.1 (a), (b) and (c) show the results of the 1-region, 4-region and 8-region approximation LAU calculation, respectively. Figure 4.1(d) shows the results of the normal fixed-point calculation. The first three 3D graphics operations – vertex matrix transformation, vertex lighting, rendering and texture mapping – performed by the

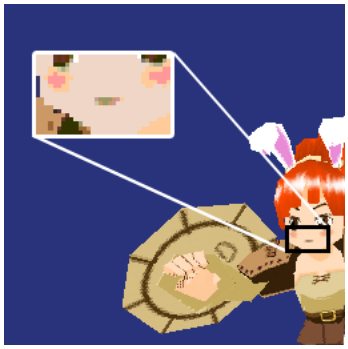
LAU, except addition and subtraction. (a) and (b) images are described with large error which means 1 and 4 region approximation is not enough for real 3D graphics application. But, unnoticeable difference is found by naked eyes between (c) and (d) images. The proposed 8-region approximation is suitable for 3D graphics application.



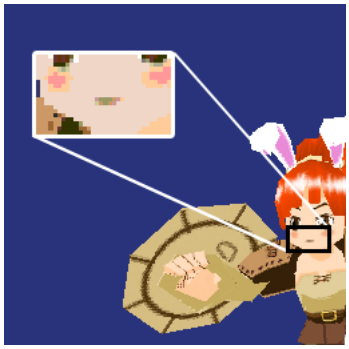
(a) LAU 1-region



(b) LAU 4-region



(c) LAU 8-region



(d) Normal fixed-point calculation

Figure 4.1 Comparison of 3D Graphics Result.

Three test model with various feature. The models have different polygon numbers, lighting calculation and texture calculation. Each features and results are shown in figure 4.2. The results show that the result images of LAU have not different looks compare to fixed point results, therefore the LAU is adopted for 3D graphics system.

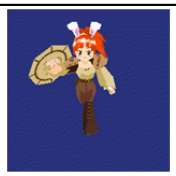
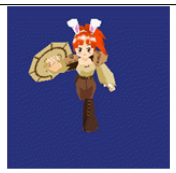
	Swiss Knife	Tank	Girl
Polygon #	6000	600	1700
Features	Lighting / No-texture	No-lighting / Texture	Lighting / Texture
FPX			
LAU			

Figure 4.2 Comparison Result of Real 3D Graphics Test Model

Table 4.1 shows the maximum error ranges of the LAU when calculating the test models, of which maximum error is less than 0.21%, and it is within tolerable range for the small screen size images of the mobile system.

Table 4.1 Maximum Percent Error Range of LAU

OP	RCP	SQRT	RSQ	SQR	MUL	DIV	POW
error (%)	0.13	0.10	0.11	0.15	0.20	0.21	0.15

The specular lighting is one of the most time-consuming parts in 3D graphics lighting algorithm because of its powering computation [14]. In general, x^y is computed by long iterations, or using large look-up table [15]. An unique solution to x^y is developed for LAU. x^y or $2^{y \cdot \log_2 x}$ can be computed in only 4 steps by using 6 different operations as shown in the following equations (17)-(20).

Step 1. $X = \log_2 x$ Eq. (17)

Step 2. $X' = \log_2 X$, $Y = \log_2 y$ Eq. (18)

Step 3. $Z = X' + Y$, $z = 2^Z$ Eq. (19)

Step 4. $2^z = 2^{y \cdot \log_2 x} = x^y$ Eq. (20)

The error induced by powering operation is less than 0.15%.

4.2 Chip Implementation Results

The proposed LAU is implemented into a chip by using 1-poly 6-metal 0.18um CMOS technology to test its efficiency. A chip photograph is shown in figure 4.3. RDX4 (Radix-4 reciprocal-square-root) is also implemented for the performance comparison [2]. The gate counts of LAU and RDX4 are 9k and 44k, respectively. The core size is 1.0*mm*x1.0*mm* and the LAU size is 240μm x 600μm.

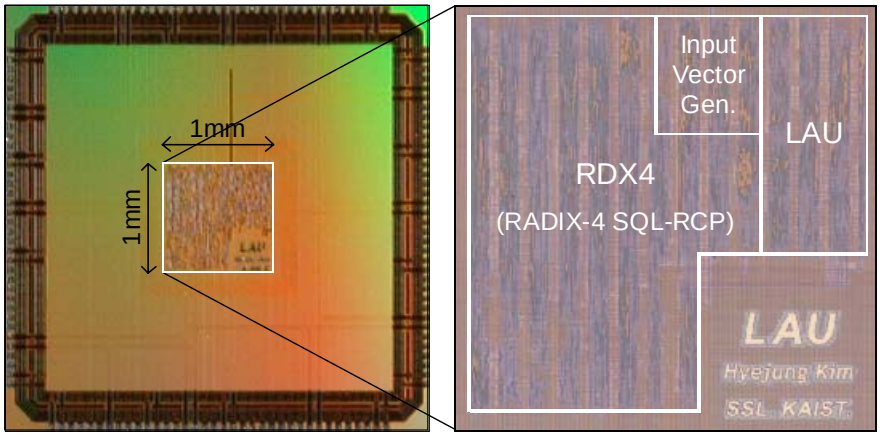


Figure 4.3 Chip Photograph

4.3 Measurement Results

The fabricated LAU operates at the frequency of 231MHz with1.8V supply voltage and its SHMOO plot is given in figure 4.4.

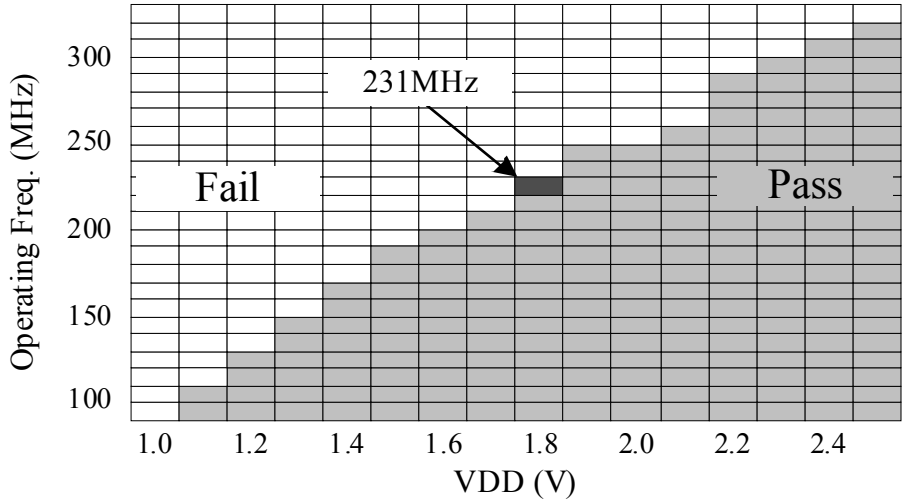


Figure 4.4 Shmoo Plot of LAU

The maximum operating frequency of RDX4 is measured as 60MHz. The measured waveforms of the critical path are shown in figure 4.5. The latency and throughput of the LAU is measured to be 2-cycle and 1-cycle, respectively, while those of RDX4's are measured as 10-cycle

and 8-cycle, respectively. By using LAU in fixed-point arithmetic, the performance is improved by 5 times compared to complex RDX4 method. The power consumption of the LAU is 2.18mW for one operand computation and 3.07mW for two operands computation. The power consumption of RDX4 is 4.29mW, which is 1.97 times of LAU's. Table 4.2 compares the performance of the LAU with that of RDX4, and table 4.3 summarizes the characteristics of the fabricated LAU chip.

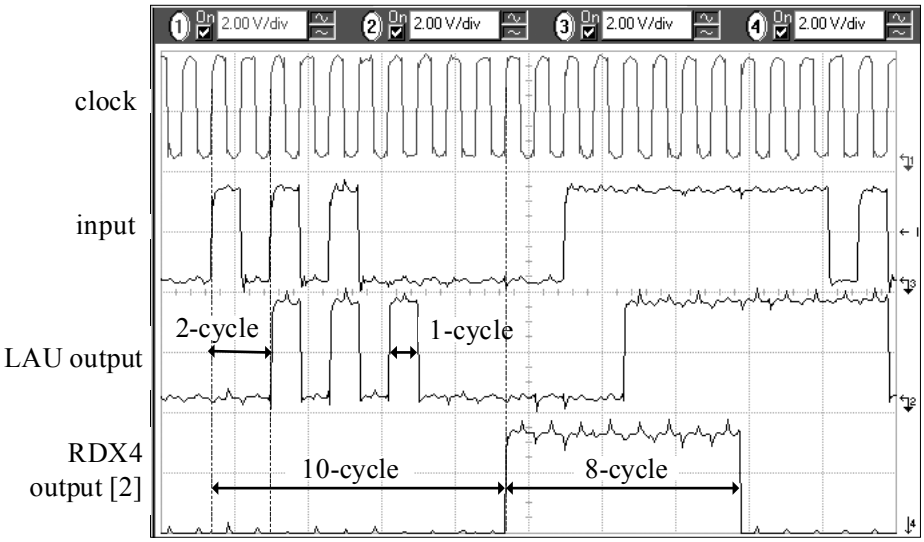


Figure 4.5 Measurement Result of Test Chip

Table 4.2 The comparison of LAU with RDX4

	RDX4 [2]	LAU
Gate count	44k	9k
Latency / Throughput	10-cycle / 8-cycle	2-cycle / 1-cycle
Max. operating frequency	60MHz	231MHz
Power consumption	4.29mW	2.18mW

Table 4.3 Characteristics of The Fabricated LAU Chip

Process Technology	1-poly 6-metal 0.18um CMOS technology
Power Supply	1.8V
Operating Frequency	231MHz
Latency / Throughput	2-cycle / 1-cycle
Power Consumption	1-operand : 2.18mW 2-operand : 3.07mW
Gate Counts	9k
Area	die : 4.0mm x 4.0mm (pad limited) core : 1.0mm x 1.0mm

The LAU is applied to the real 3D graphics rendering processor for texture mapping unit [16]. By implementing LAU instead RADIX-4 divider, the operating speed is improved by 4.2 times, and power consumption and silicon area can be reduced.

CHAPTER 5

CONCLUSIONS

5.1 Conclusions

A 32-bit Logarithmic Arithmetic Unit is proposed for mobile 3D graphics system. The arithmetic unit consists of a binary logarithmic converter, an adder, a shifter and a binary exponential converter. It uses 8-region piecewise linear interpolation approximation algorithms and supports variable number range to compute complex functions fast and accurately. LAU is implemented with 0.18μmCMOS technology and it takes 9k gates. The fabricated LAU runs at 231MHz, and performs multiplication, division, reciprocal, reciprocal-square-root and square operations in only 2-cycle, and powering operation in 4-cycle. The errors of computations are within 0.2%, which is tolerable in the case of small screen size, 512x512,of mobile 3D graphics system. It

consumes 2.18mW for 1 operand computation, 3.07mW for 2 operands computation. It is integrated and successfully operated in a mobile 3D graphics system [16].

SUMMARY

최근 들어 실시간 3D 그래픽은 휴대용 시스템에서 관심이 증가하고 있는 분야이다. 3D 그래픽이 친숙해질수록 사람들은 높은 수준의 어플리케이션을 원하게 되고, 이를 해결하기 위하여 많은 그룹들은 3D 그래픽을 위한 가속기를 만들었다. 3D 그래픽 가속기는 다른 일반적인 프로세서와는 달리 매우 복잡하고, 무거운 수학 연산기들을 많이 필요로 한다. 또한 휴대용 시스템은 CPU, 메모리나 배터리 에너지 같은 자원의 한계를 가지고 있기 때문에 적은 비용의 복잡한 수학의 연산기 디자인은 매우 중요하다.

나눗셈, 역수 계산 같은 복잡한 수학 연산들은 3D 그래픽에서 매우 중요하지만, 매우 많은 계산 시간과 파워 소모을 차지하기 때문에 최적화가 필요하다. 본 논문에서는 이러한 문제를 Logarithmic Number System을 사용한 Logarithmic 연산기를 구현하여 고속 저전력으로 연산을 실행하는 방법을 제안하였다.

제안된 Logarithmic 연산기는 실수를 Log Number로, Log Number를 실수로의 변환을 8개의 구간으로 나누어 근사하여 각각 0.064%와 0.152%의 에러를 얻어 휴대용 3D 그래픽과 같이 화면의 크기가 작은 경

우 눈으로 식별이 불가능할 정도의 결과를 제공한다.

제안된 Logarithmic 연산기를 2 Stage Pipeline으로 구성하여 0.18um CMOS 공정으로 칩을 제작하였다. 제작된 칩은 1.8V에서 231MHz로 동작하고 2 Cycle의 Latency와 1 Cycle의 Throughput을 갖는다. 또한 0.2%의 계산 에러를 가지며, Gate 개수는 약 9,000개이다.

제안된 Logarithmic 연산기는 종래의 Radix-4 연산기보다 Gate의 개수는 4.88 감소하였고, 성능은 5배 향상했으며, 소비 전력은 약 2배 감소하였다. 또한 실제의 3D 그래픽 시스템에 적용하여 성공적이고 안정적인 동작을 검증하였다.

REFERENCES

[1] Ramchan Woo, Sungdae Choi, Ju-Ho Sohn, Seong-Jun Song and Hoi-Jun Yoo, "A 210-mW Graphics LSI Implementing Full 3-D Pipeline With 264 Mtexels/s Textureing for Mobile Multimedia Applications", IEEE Journal of Solid-State Circuits, Vol.39, No.2, pp.358-367, Feb. 2004.

[2] Ju-Ho Sohn, Ramchan Wooand Hoi-Jun Yoo, "A Programmable Vertex Shader with Fixed-Point SIMD Datapath for Low Power Wireless Applications", Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware 2004, Vol.1, pp.107-114, 2004.

[3] Kanako Yosida, Tadashi Sakamoto and Tomohiro Hase, "A 3D Graphics Library For 32-bit Mocroprocessors For Embedded Systems", IEEE Trans. On Consumer Electronics, Vol.44,– 1114, Aug. 1998.

[4] J.-A.Pineiro et al. "High-Speed Double-Precision Computation of Reciprocal, Division, Square Root, and Inverse Square Root", IEEE Trans. On Computer, vol. 51, pp.1377-1388, Dec. 2002.

[5] E.I.Chester, J.N.Coleman, "Matrix Engine for Signal Processing Applications using the Logarithmic Number System", Proc. IEEE Application-Specific Systems, Architectures and Processors, pp.315-324, Jul. 2002.

[6] J.N.Coleman et al. "Arithmetic on the European Logarithmic Microprocessor", IEEE Trans. On Computer, vol.49, no.7, pp.702-715, Jul. 2000.

[7] J.N. Mitchell Jr., "Computer Multiplication and Division Using Binary Logarithms," IRE Trans. Electronic Computers, vol. 11, pp. 512-517, Aug. 1962.

[8] S.L. SanGregory, R.E. Siferd, C. Brother, and D. Gallagher, "A Fast, Low-Power Logarithm Approximation with CMOS VLSI Implementation," Proc. IEEE Midwest Symp. Circuits and Systems, pp.388-391, Aug. 1999.

[9] M. Combet, H. Zonneveld, and L. Verbeek, "Computation of the Base Two Logarithm of Binary Numbers," IEEE Trans. Electronic Computers, vol. 14, pp. 863-867, Dec. 1965.

[10] E.L. Hall, D.D. Lynch, and S.J. Dwyer III, "Generation of Products and Quotients Using Approximate Binary Logarithms for Digital Filtering Applications," IEEE Trans. Computers, vol. 19, pp. 97-105, Feb. 1970.

[11] Khalid H.Abed, "CMOS VLSI Implementation of a Low-Power Logarithmic Converter", IEEE Trans. On Computer, vol.52, No. 11, pp.1421-1433, Nov. 2003.

[12] Khalid H.Abed, Ramond E.Siferd, "VLSI Implementation of a Low-Power Antialgorithmic Converter", IEEE Trans. On Computer, vol.52, No.9, pp.1221-1228, Sep. 2003

[13] Tomas Akenine-Moller and Eric Haibes, "REAL-TIME RENDERING", Second Edition, 2002 by A K Peters, Ltd.

[14] Mark Segal, Kurt Akeley, "The OpenGL Graphics System : A Specification Version 1.2", Silicon Graphics Inc. pp.44-50, Apr. 1999.

[15] Byeong-Gyu Nam, Min-wuk Lee and Hoi-Jun Yoo, "Development of a 3-D Graphics Rendering Engine with Lighting Acceleration for Handheld Multimedia Systems", IEEE Trans. on Consumer Electronics, vol. 51, No. 3, pp.1020-1027, Aug. 2005

[16] Jeong-Ho Woo, Min-Wuk Lee, Hyejung Kim, Ju-Ho Sohn and Hoi-Jun Yoo, "A 1.2Mpixels/s/mW 3-D Rendering Processor For Portable Multimedia Application", Proc. IEEE Asian Solid State Circuits Conference, pp.297-300, Nov. 2005.

ACKNOWLEDGEMENT

반도체 시스템 연구실의 일원이 되어 2년 6개월의 시간이 흘러, 석사 과정을 무사히 마치게 되었습니다. 지금의 제가 있기 까지 도움을 주신 많은 분들에게 짧게나마 감사의 마음을 전합니다.

학부 시절부터 많이 부족한 저를 이끌어주시고, 때로는 다정하게 때로는 엄하게 용기를 북돋아 주시는 유회준 교수님께 진심으로 감사드립니다. 그리고 바쁘신 가운데도 저의 논문을 심사해 주시고 값진 조언을 해주신 박인철 교수님, 신영수 교수님께도 감사드립니다.

지난 3년간 동고동락 하였고, 앞으로도 많은 시간을 함께 보내게 될 반도체 시스템 실험실 동료들에게도 진심으로 고마운 마음을 전합니다.

최고를 위해 달려가는 모범을 보여준 람찬오빠와, 세중오빠. 가장 닮고 싶은 선배였던 멋진 강민오빠. 철 없던 신입생 시절 받은 많은 조언과 가르침은 잊지 못할 것입니다.

저의 연구에 대해서 많은 조언을 해주시는 병규오빠. 회사에서의 많은 경험을 가르쳐 주시는 교민 오빠. 학부 시절부터 저의 사수가 되어 주었던 주호오빠. 팀장으로서 책임감 넘치는 성대오빠. 연구에 대한 진지한 자세를 몸소 보여주는 성준오빠. 늘 큰오빠 같이 푸근한 정호오빠. 한결

같은 긍정적이고 성실한 모습을 배우고 싶은 동현오빠. 정신적인 버팀목이 되어주는 선영언니. 저를 위하여 많은 조언을 아끼지 않고, 실험실에 잘 적응하도록 도움을 주신 선배님들께 감사의 말씀을 전합니다.

연구를 하면서도, 실험실 생활을 하면서도 많은 의지가 되는 담오빠. 날카로운 주영이. 창의력이 돋보이는 제빈이. 듬직하면서도 엉뚱한 승진이. 부족한 저를 선배로 맞아 고생하고 있는 후배들에게도 고마움을 전합니다. 최고의 라이벌이자 최고의 파트너인, 항상 의지가 되는 석사 동기인 남준오빠와 관호에게도 고마움을 전합니다. 또한 많은 사무를 처리해 주시는 은수언니에게도 고마움을 전합니다.

멀리 떨어져 있어 맏이 역할을 하지 못하는 저를 믿고 응원해주는 가족들에게 감사의 마음을 전합니다. 언니의 빈자리를 대신해 채워주고 있는 은정이. 속 깊은 막내 호경이. 언제나 제 편이 되어 주시는 할머니. 마지막으로 큰 사랑을 주시는 어머니, 아버지께 이 모든 결실을 바칩니다.

CURRICULUM VITAE

Name Hyejung Kim (김혜정, 金惠貞)

Date of Birth October 22, 1982

Address Division of Electrical Engineering,
Department of Electrical Engineering and Computer Science,
Korea Advanced Institute of Science and Technology,
373-1, Guseong-dong, Yuseong-gu.,
Daejeon, 305-701, Republic of Korea

Education

2000.3 - 2004.2 B.S. in Div. of EE, Dept. of EECS, KAIST

2004.3 - 2006.8 M.S. in Div. of EE, Dept, of EECS, KAIST

Publication

[1] **Hyejung Kim**, Byeong-Gyu Nam, Ju-ho Sohn, and Hoi-Jun Yoo, "A 231MHz, 2.18mW 32-bit Logarithmic Arithmetic Unit for Fixed-Point 3D Graphics System," in *IEEE Proc. Int. Asian Solid-State Circuits Conference,* Nev. 2005.