A 231-MHz, 2.18-mW 32-bit Logarithmic Arithmetic Unit for Fixed-Point 3-D Graphics System

Hyejung Kim, *Student Member, IEEE*, Byeong-Gyu Nam, Ju-Ho Sohn, *Student Member, IEEE*, Jeong-Ho Woo, *Student Member, IEEE*, and Hoi-Jun Yoo, *Senior Member, IEEE*

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

***Index Terms*—ALU, antilogarithm, logarithmic number system (LNS), 3-D graphics.**

I. Introduction

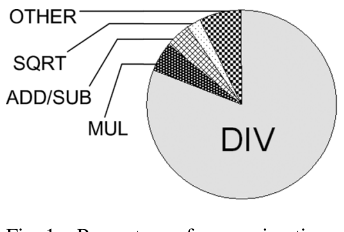
NOWADAYS, the real-time three-dimensional (3-D) graphics are one of the attractive applications for mobile systems [1]. As 3-D graphics are becoming more and more familiar to people, there have been increasing demands for high-quality graphics for new applications such as avatar, advertisement, and games. Many 3-D graphics processors have been studied [1], [2] to meet these requests. The mobile system has low resolution and a smaller screen size than a PC system. However, the mobile system also has limited resources in terms of CPU performance, memory capacity, and battery energy. Most 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].

The 3-D graphic processors require heavy arithmetic calculations like division, reciprocal, square-root, square, and powering operations in contrast to general processors. Fig. 1 shows the percentage of processing time of 3-D graphics rendering pipeline, and the heavy arithmetic functions take 83% of the total processing time [3]. Of course, these functions consume most of the computing power because they use most of the clock cycles in the real time 3-D graphics systems. Also, for the low power consumption, the clock cycles of these complex functions should be reduced as much as possible.

Manuscript received February 17, 2006; revised June 23, 2006.

The authors are with the Division of Electrical Engineering Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Korea (e-mail: seeseah@ee-info.kaist.ac.kr; bgnam@eeinfo.kaist.ac.kr; sohnjuho@eeinfo.kaist.ac.kr; denber@eeinfo.kaist.ac.kr; hjyoo@ee.kaist.ac.kr).

Digital Object Identifier 10.1109/JSSC.2006.882887



░	DIV	80.95%
▦	MUL	5.11%
▨	ADD/SUB	4.18%
▤	SQRT	2.65%
▩	OTHER	7.11%

Fig. 1. Percentage of processing time of 3-D graphics rendering pipeline [3].

The logarithmic number system (LNS) has been studied to simplify arithmetic computations for lower computation complexity, high computation speed, and small gate counts [5]–[13]. Although there is conversion overhead from integer to logarithm or logarithmic to integer, the overhead is much smaller than that of 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 a 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 error in hardware implementation [7]–[11]. In the approximation methods, typically sizable errors occur during logarithmic and antilogarithmic converting. However, their operation errors have not been carefully examined in relation to the real applications. Although many approximation methods have been proposed for error reduction, their errors are still too large to be used for 3-D graphics systems. 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 3-D 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 that performs complex operations in LNS and simple addition/subtraction operations in fixed-point number systems 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.

The organization of this paper is as follows. The proposed 32-bit logarithmic arithmetic unit will be discussed in Section II. In addition, in this section, the small error logarithmic and antilogarithmic conversion algorithms will be

TABLE I
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**	$z = x - y$	$X + \log_2(1-2^{Y-X})$

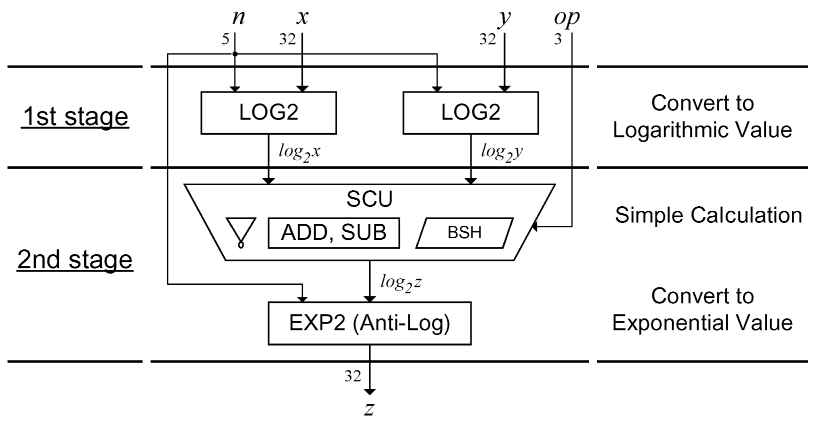


Fig. 2. Top architecture of LAU.

described. The evaluation results will be shown in Section III, and implementation results and measurement results will be explained in Section IV. Finally, the conclusion of our work will be summarized in Section V.

II. 32-bit LOGARITHMIC ARITHMETIC UNIT (LAU)

A. 32-bit LAU

The LAU computes the complex functions such as multiplication, division, and square-root operations by using only simple addition, subtraction, and shift operations. When $X = \log_2 x$ and $Y = \log_2 y$, Table I summarizes the normal operations and their equivalent logarithmic operations. Contrary to complex functions, simple addition and subtraction operations need more complex nonlinear computations in LNS. In this study, we combine two different number systems into a unified arithmetic unit for better performance: simple addition/subtraction in a fixed-point number system and complex functions in LNS.

The top architecture of the proposed LAU is shown in Fig. 2. The unit is composed of two binary logarithmic converters (LOG2s) in the first stage, a simple calculation unit (SCU), and a binary antilogarithmic converter (EXP2) in the second stage. The SCU is composed of an inverter, an adder/subtracter (ADD/SUB), and a barrel shifter (BSH). 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 a high clock frequency. Since the logarithmic converter takes a 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. Thus, the LAU performs the functions given in Table I in two cycles except for the powering operation x^y. Further explanation on the powering operation will be given in Section III. 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 a 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, the fixed-point system can operate at higher clock frequencies. Each stage of the 3-D 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. Fig. 3(a) shows the details 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. However, the internal number range of LAU is fixed to Q6.26. as shown in Fig. 3(b), to reduce the calculation complexity and latency. After LAU calculation, the output number range is modified to Qm.n. which

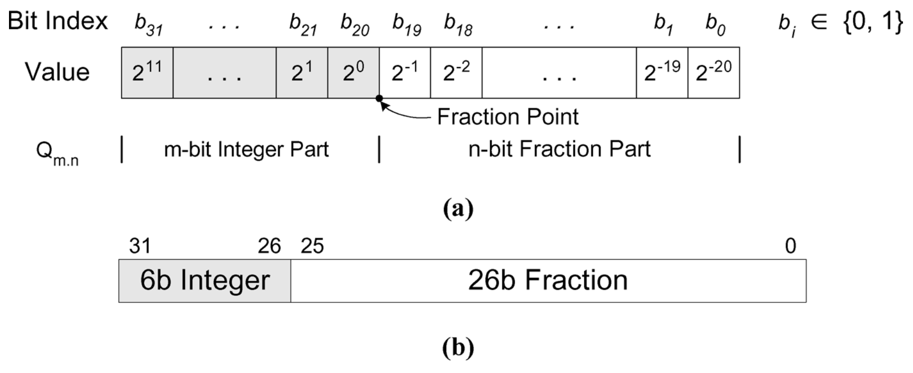


Fig. 3. Fixed-point number format. (a) Number format (example: Q12.20 Format). (b) Internal fixed number range Q6.26 of LAU.

TABLE II
COMPARISION OF LOGARITHMIC APPROXIMATION ERROR

	Mitchell [7]	SanGregory [8]	Combet [9]	Hall [10]	Abed [11]	This work
Max. +error	5.361	0.431	0.185	0.126	0.154	0.044
Max. -error	0	-1.540	-0.267	-0.718	-0.153	-0.050
Error Range (%)	5.361	1.191	0.452	0.844	0.307	0.094

is suitable to the 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 $-32.0 < \log_2 x < 32.0$, 6 bit are needed to represent the 2's complement signed integer value. The six most significant digits of $\log_2 x$ represent the integer part and the remaining 26 bits represent the fractional part.

B. Logarithmic Converter Block

In general, the piecewise interpolation methods were used in the binary logarithm conversion algorithms [7]–[11]. They usually used 2–6-region piecewise-linear methods. Their error ranges are shown in Table II, and the minimum error with number range of Q32.0 is 0.31% [7]–[11]. In this study, we divide the fraction part into eight regions to further reduce its error rate and use the straight linear interpolation in each region. Fig. 4 shows the proposed algorithm which tracks the exact value much better than Mitchell's [7] and Hall's [10] methods do.

The binary logarithmic conversion algorithm was 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

$$x = 2^k(1 + f) \tag{1}$$

where 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 by the new value of k', which is equal to $k - n$. f is the fraction parts in the range [0, 1] located on the right-hand side of the leading bit. Taking the binary logarithm for both sides of (1), the following equations can be obtained:

$$\log_2 x = k' + n + \log_2(1+f), \quad \text{where } k' = k - n \tag{2}$$

$$\log_2(1+f) \cong \alpha_i \cdot f + \beta_i, \quad \text{where } i = 0, 1, \ldots, 7 \tag{3}$$

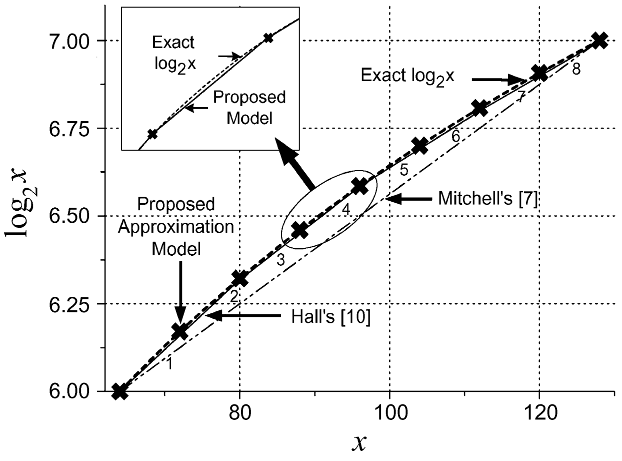


Fig. 4. Comparison results of approximation value.

where $\log_2(1 + f)$ is the fractional part after the binary logarithmic conversion. This term is calculated by piecewise interpolation represented in (3). α_i and β_i are the coefficients which have 7- 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 (3) 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 two as their denominators, they can be calculated by only shifters and adders. When defining the coefficients, the error range should be considered as well. Table III shows the optimized values of α_i and β_i.

Fig. 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. The logarithmic converter block is composed of 32-bit count leading zero (CLZ),

TABLE III
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

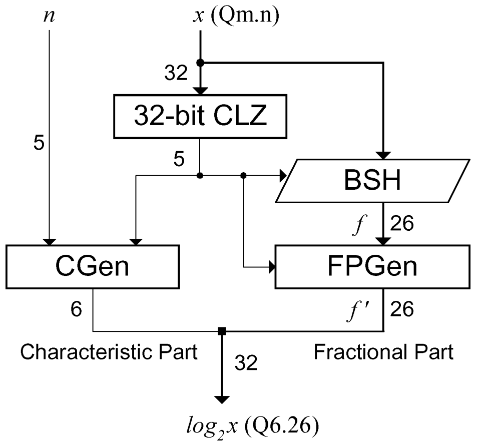


Fig. 5. Architecture of the logarithmic converter.

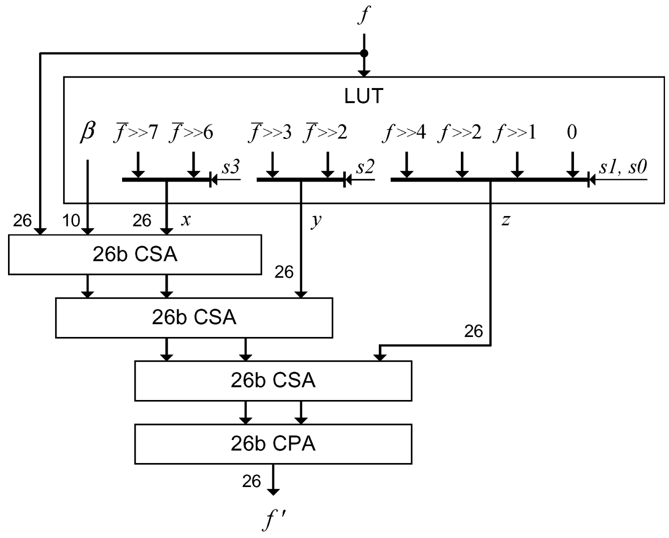


Fig. 6. Architecture of the FPGen block.

BSH, a characteristic generator (CGen), and fractional part generation block (FPGen). The variable input number range Qm.n. is modified to the fixed number range of Q6.26. at the end of the logarithmic converter. The CLZ block calculates the number of the leading zero bits of the input. The five 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 the FPGen block and the characteristic part is generated by the 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 Fig. 6. FPGen generates the approximated fractional value of (3) and Table III. It is implemented by a hardwired shifter, MUX, carry save adder (CSA), and carry propagating adder (CPA). f is the original fractional part value and f' is the modified value. β is the approximation coefficient and $s0-s3$ are selection bits made by the three most significant bits of f. x, y, and z are compensation values 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 the 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.

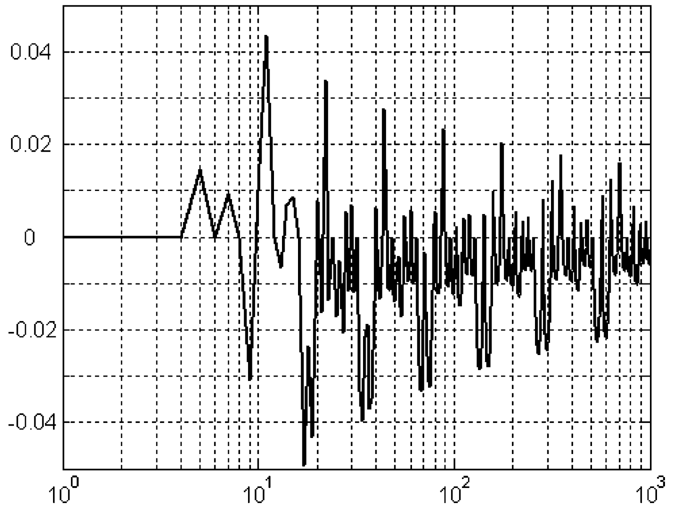


Fig. 7. Percent error of the proposed logarithmic converter.

The error of the logarithmic converting algorithm $\mathrm{error}_{\log}$ is given as

$$\begin{aligned}\mathrm{error}_{\log} &= \log_2 x - (\log_2 x)' \\ &= \log_2(1+f) - (\alpha \cdot f + \beta). \qquad (4)\end{aligned}$$

The maximum percent error range is $-0.050 < \mathrm{error}_{\log} < 0.044$, and Fig. 7 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. The proposed logarithmic converter operates at a maximum frequency of 260 MHz 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

TABLE IV
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

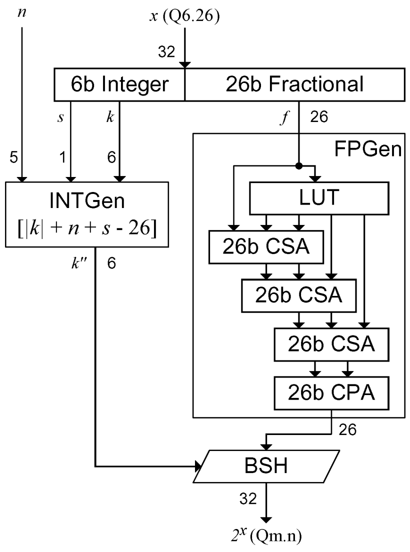


Fig. 8. Architecture of the antilogarithmic converter.

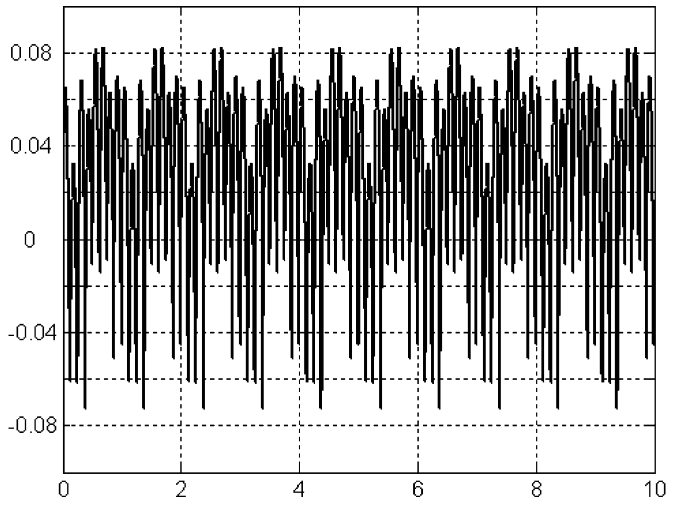


Fig. 9. Percent error of the proposed antilogarithmic converter.

55 MHz in 0.6-μm technology which is equivalent to 61FO4. The critical path and the maximum error of the logarithmic converter are reduced by 31% and 69.4%, respectively.

C. Antilogarithmic Converter Block

In general, the piecewise interpolation method is also used for the antilogarithmic converting algorithm [12]. The fixed-point representation of the antilogarithm is composed of the six most significant bits representing the integer part 2^k and the remaining 26 bits of the fractional part representing 2^f as given in (5). This value is divided into two parts, according to the positive or the negative input x:

$$\begin{aligned}2^x &= 2^{k+f} = \left| \begin{array}{ll} 2^k \cdot 2^f, & x \geq 0 \\ 2^{k-1} \cdot 2^{1-|f|}, & x < 0 \end{array} \right. \\ &= 2^{k'+f'} \\ &= 2^{k'} \cdot 2^{f'} \end{aligned} \tag{5}$$

where k and f values are replaced when x is a negative value in order to make the fractional part a positive value. Thus, k and f are modified to k' and f' as given by the following first two equations:

$$k' = \left| \begin{array}{ll} k, & x \geq 0 \\ k-1, & x < 0 \end{array} \right. \tag{6}$$

$$f' = \left| \begin{array}{ll} f, & x \geq 0 \\ 1-|f|, & x < 0 \end{array} \right. \quad \text{where } 0 \leq f' < 1 \tag{7}$$

$$k'' = k' + n - 26, \tag{8}$$

$$2^{f'} \cong \alpha_i \cdot f + \beta_i, \quad \text{where } i = 0, 1, \ldots, 7. \tag{9}$$

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 (9). 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 IV.

Fig. 8 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. The antilogarithmic converter is composed of the integer part generation block (INTGen), FPGen, and BSH. The computation time is shorter than the logarithmic converter because the integer part and the fractional part are calculated separately. CGen calculates the integer part by using simple addition of x, sign bits, and n. FPGen is composed of a lookup table (LUT), carry save adders (CSAs), and a CPA similar to the logarithmic converter's. The final BSH shifts the fractional part value by the result value of the CGen 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 $\text{error}_{\text{antilog}}$ is equal to

$$\begin{aligned}\text{error}_{\text{antilog}} &= 2^x - (2^x)' \\ &= 2^k \left[2^f - (\alpha \cdot f + \beta)\right].\end{aligned} \tag{10}$$

The maximum percent error range is $-0.070 < \text{error}_{\text{antilog}} < 0.082$ and Fig. 9 shows its error graph. The proposed eight-region interpolation algorithm is compared with the previous works of the antilogarithm converters [12]

TABLE V
COMPARISON OF THE 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

TABLE VI
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

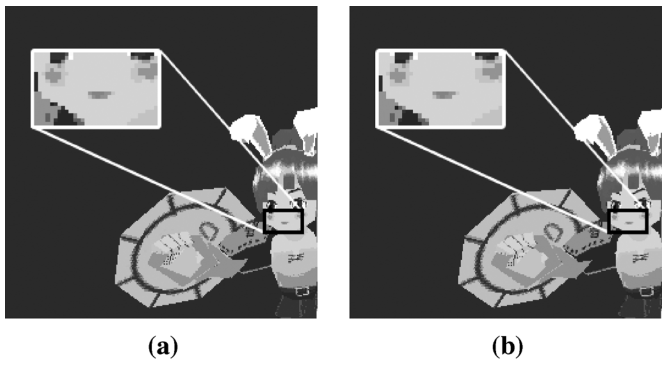


Fig. 10. Comparison of 3-D graphics result. (a) Normal fixed-point calculation. (b) LAU calculation.

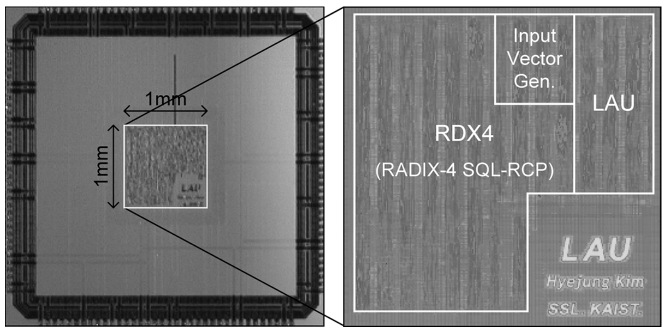


Fig. 11. Chip photograph.

in Table V. The maximum error range of the antilogarithmic converter is reduced by over ten times.

III. EVALUATION RESULTS

The proposed LAU is verified in the 3-D graphics processing software environment before its chip is implemented. Fig. 10 is the test scene and the in-box shows a zoomed image for the accuracy comparison. The test model consists of 1700 polygons with lighting and texture mapping. The screen resolution is 512 $\times$ 512 and the texture size is 256 $\times$ 256. The variable number range Qm.n is used to evaluate the scene. Fig. 10(a) shows the results of the normal fixed-point calculation, and Fig. 10(b) shows the picture of LAU calculation. All 3-D graphics operations—vertex matrix transformation, vertex lighting, rendering, and texture mapping—of the second figure are performed by the LAU, except for addition and subtraction. No noticeable difference is found by naked eyes between the two images. Table VI shows the maximum error ranges of the LAU when calculating the test model, of which maximum error is less than 0.21%. Since the 0.21% error occurs when the value is $2^{32}-1$, the error range should be much smaller if the value is less than $2^{32}-1$. Its small error range is within a tolerable range for the small screen size images of the mobile system.

The specular lighting is one of the most time-consuming parts in the 3-D graphics lighting algorithm because of its powering computation [13]. In general, x^y is computed by long iterations or using a large LUT [14]. A unique solution to x^y is developed for LAU. x^y or $2^{y\cdot\log_2 x}$ can be computed in only four steps by using six different operations as shown as follows.

Step 1)

$$X = \log_2 x. \tag{11}$$

Step 2)

$$\begin{aligned} X' &= \log_2 X \\ Y &= \log_2 y. \end{aligned} \tag{12}$$

Step 3)

$$\begin{aligned} Z &= X' + Y \\ z &= 2^Z. \end{aligned} \tag{13}$$

Step 4)

$$2^z = 2^{y\cdot\log_2 x} = x^y. \tag{14}$$

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

IV. CHIP IMPLEMENTATION AND MEASUREMENTS RESULTS

The proposed LAU is implemented into a chip by using 1-poly 6-metal 0.18-μm CMOS technology to test its efficiency. A chip photograph is shown in Fig. 11. Radix-4 reciprocal-square-root (RDX4) is also implemented for the performance comparison [2]. The gate counts of LAU and RDX4 are 9 k and 44 k, respectively. The core size is 1.0 mm $\times$ 1.0 mm and the LAU size is 240 μm $\times$ 600 μm. The fabricated LAU operates at the frequency of 231 MHz with 1.8-V supply voltage and its Shmoo plot is given in Fig. 12. The maximum

TABLE VII
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 VIII
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

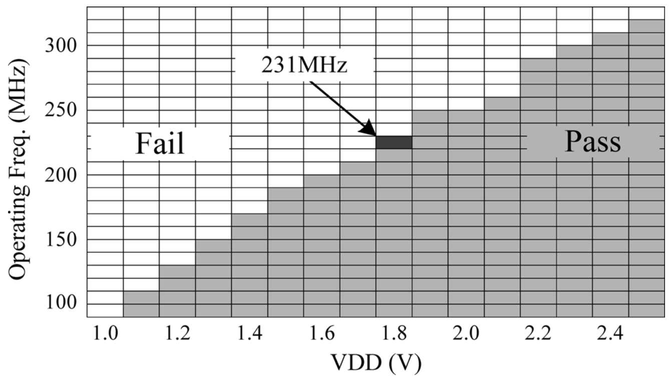


Fig. 12. Shmoo plot of LAU.

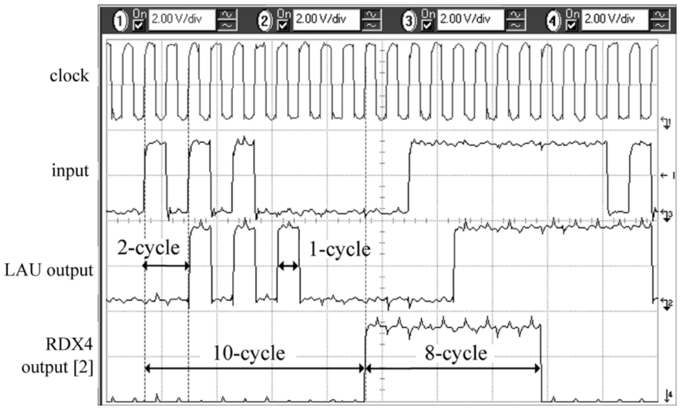


Fig. 13. Measurement result of test chip.

operating frequency of RDX4 is measured as 60 MHz. The measured waveforms of the critical path are shown in Fig. 13. The latency and throughput of the LAU is measured to be 2-cycle and 1-cycle, respectively, while those of RDX4 are measured as 10-cycle and 8-cycle, respectively. By using the LAU in fixed-point arithmetic, the performance is improved by five times compared with the complex RDX4 method. The power consumption of the LAU is 2.18 mW for one-operand computation and 3.07 mW for two-operand computation. The power consumption of RDX4 is 4.29 mW, which is 1.97 times that of LAU's. Table VII compares the performance of the LAU with that of RDX4, and Table VIII summarizes the characteristics of the fabricated LAU chip.

The LAU is applied to the real 3-D graphics rendering processor for texture mapping unit [15]. 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.

V. CONCLUSION

A 32-bit LAU is proposed for mobile 3-D graphics system. The arithmetic unit consists of a binary logarithmic converter, an adder, a shifter, and a binary exponential converter. It uses eight-region piecewise-linear interpolation approximation algorithms and supports a variable number range to compute complex functions fast and accurately. LAU is implemented with 0.18-μm CMOS technology and it takes 9-k gates. The fabricated LAU runs at 231 MHz and performs multiplication, division, reciprocal, reciprocal-square-root, and square operations in only two cycles and powering operation in four cycles. The errors of computations are within 0.2%, which is tolerable in the case of small screen size, 512 $\times$ 512, of a mobile 3-D graphics system. It consumes 2.18 mW for one-operand computation and 3.07 mW for two-operand computation. It is integrated and successfully operated in a mobile 3-D graphics system [15].

Appendix A

The follow equations describe the definition of LUT for logarithmic FPGen block:

$$\begin{aligned} f' &= f + (f \gg 1) + (\bar{f} \gg 3) + (\bar{f} \gg 7), \\ &\quad \text{for} \quad f \in [0, 1/8) \\ f' &= f + (f \gg 1) + (\bar{f} \gg 2) + (\bar{f} \gg 6) + (15/1024), \\ &\quad \text{for} \quad f \in [1/8, 2/8) \\ f' &= f + (f \gg 2) + (\bar{f} \gg 3) + (\bar{f} \gg 6) + (46/1024), \\ &\quad \text{for} \quad f \in [2/8, 3/8) \\ f' &= f + (\bar{f} \gg 7) + (91/1024), \quad \text{for} \quad f \in [3/8, 4/8) \\ f' &= f + (\bar{f} \gg 3) + (f \gg 4) + (\bar{f} \gg 7) + (123/1024), \\ &\quad \text{for} \quad f \in [4/8, 5/8) \\ f' &= f + (\bar{f} \gg 3) + (\bar{f} \gg 6) + (167/1024), \\ &\quad \text{for} \quad f \in [5/8, 6/8) \\ f' &= f + (\bar{f} \gg 2) + (f \gg 4) + (\bar{f} \gg 6) + (215/1024), \\ &\quad \text{for} \quad f \in [6/8, 7/8) \\ f' &= f + (\bar{f} \gg 2) + (\bar{f} \gg 7) + (264/1024), \\ &\quad \text{for} \quad f \in [7/8, 1). \end{aligned}$$

The follow equations describe the definition of LUT for antilogarithmic FPGen block:

$$\begin{aligned} f' &= f + (\bar{f} \gg 2) + (\bar{f} \gg 5) + (1024/1024), \\ &\quad \text{for} \quad f \in [0, 1/8) \\ f' &= f + (\bar{f} \gg 2) + (f \gg 5) + (f \gg 7) + (1015/1024), \\ &\quad \text{for} \quad f \in [1/8, 2/8) \\ f' &= f + (\bar{f} \gg 3) + (\bar{f} \gg 7) + (995/1024), \\ &\quad \text{for} \quad f \in [2/8, 3/8) \\ f' &= f + (\bar{f} \gg 4) + (f \gg 7) + (964/1024), \\ &\quad \text{for} \quad f \in [3/8, 4/8) \\ f' &= f + (f \gg 5) + (\bar{f} \gg 7) + (924/1024), \\ &\quad \text{for} \quad f \in [4/8, 5/8) \\ f' &= f + (f \gg 3) + (\bar{f} \gg 7) + (864/1024), \\ &\quad \text{for} \quad f \in [5/8, 6/8) \\ f' &= f + (f \gg 2) + (\bar{f} \gg 5) + (\bar{f} \gg 7) + (792/1024), \\ &\quad \text{for} \quad f \in [6/8, 7/8) \\ f' &= f + (f \gg 2) + (f \gg 4) + (f \gg 7) + (695/1024), \\ &\quad \text{for} \quad f \in [7/8, 1). \end{aligned}$$

References

[1] R. Woo, S. Choi, J.-H. Sohn, S.-J. Song, and H.-J. Yoo, "A 210-mW graphics LSI implementing full 3-D pipeline with 264 Mtexels/s textureing for mobile multimedia applications," *IEEE J. Solid-State Circuits*, vol. 39, no. 2, pp. 358–367, Feb. 2004.

[2] J.-H. Sohn, R. Woo, and H.-J. Yoo, "A programmable vertex shader with fixed-point SIMD datapath for low power wireless applications," in *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware*, 2004, vol. 1, pp. 107–114.

[3] K. Yosida, T. Sakamoto, and T. Hase, "A 3D graphics library for 32-bit mocroprocessors for embedded systems," *IEEE Trans. Consum. Electron.*, vol. 44, no. 4, pp. 1107–1114, Aug. 1998.

[4] J.-A. Pineiro *et al.*, "High-speed double-precision computation of reciprocal, division, square root, and inverse square root," *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1377–1388, Dec. 2002.

[5] E. I. Chester and J. N. Coleman, "Matrix engine for signal processing applications using the logarithmic number system," in *Proc. IEEE Application-Specific Syst., Architectures Processors*, Jul. 2002, pp. 315–324.

[6] J. N. Coleman *et al.*, "Arithmetic on the European logarithmic microprocessor," *IEEE Trans. Comput.*, vol. 49, no. 7, pp. 702–715, Jul. 2000.

[7] J. N. Mitchell, Jr., "Computer multiplication and division using binary logarithms," *IRE Trans. Electron. Comput.*, 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 Syst.*, pp. 388–391, Aug. 1999.

[9] M. Combet, H. Zonneveld, and L. Verbeek, "Computation of the base two logarithm of binary numbers," *IEEE Trans. Electron. Comput.*, 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. Comput.*, vol. C-19, no. 2, pp. 97–105, Feb. 1970.

[11] K. H. Abed, "CMOS VLSI implementation of a low-power logarithmic converter," *IEEE Trans. Comput.*, vol. 52, no. 11, pp. 1421–1433, Nov. 2003.

[12] K. H. Abed and R. E. Siferd, "VLSI implementation of a low-power antialgorithmic converter," *IEEE Trans. Comput.*, vol. 52, no. 9, pp. 1221–1228, Sep. 2003.

[13] M. Segal and K. Akeley, The OpenGL Graphics System: A Specification Version 1.2, Silicon Graphics Inc., pp. 44–50, Apr. 1999.

[14] B.-G. Nam, M.-W. Lee, and H.-J. Yoo, "Development of a 3-D graphics rendering engine with lighting acceleration for handheld multimedia systems," *IEEE Trans. Consum. Electron.*, vol. 51, no. 3, pp. 1020–1027, Aug. 2005.

[15] J.-H. Woo, M.-W. Lee, H. Kim, J.-H. Sohn, and H.-J. Yoo, "A 1.2 Mpixels/s/mW 3-D rendering processor for portable multimedia application," in *Proc. IEEE Asian Solid State Circuits Conf.*, Nov. 2005, pp. 297–300.

Hyejung Kim (S'04) received the B.S. degree in electrical engineering and computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2004, where she is currently working toward the M.S. degree.

Her research interests include low-power arithmetic and processor design for mobile applications.

Byeong-Gyu Nam received the B.S. degree (*summa cum laude*) in computer engineering from Kyungpook National University, Daegu, Korea, in 1999, and the M.S. degree in computer science from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2001. He is currently working toward the Ph.D. degree in electrical engineering at KAIST.

From 2001 to 2002, he was a Research Engineer with the Parallel System Research Laboratory, Electronics and Telecommunication Research Institute, Daejeon, Korea. His research interests include low-power design and implementation of 3-D graphics processors and microprocessors for handheld systems.

Ju-Ho Sohn (S'01) received the B.S. (*summa cum laude*) and M.S degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2001 and 2003, respectively. He is currently working toward the Ph.D. degree in the same department at KAIST.

His research interests include low-power high-performance circuits and multimedia system design with specific interest in 3-D computer graphics architecture and its implementation for mobile applications.

Jeong-Ho Woo (S'02) received the B.S. and M.S. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2002 and 2004, respectively. He is currently working toward the Ph.D. degree in electrical engineering at KAIST.

His research includes low-power design of mobile multimedia systems with 3-D computer graphics and its implementation. In particular, he is currently involved with low-power programmable rendering processors for mobile devices.

Hoi-Jun Yoo (SM'04) graduated from the Electronic Department of Seoul National University, Seoul, Korea, in 1983, and received the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 1985 and 1988, respectively. His Ph.D. work concerned the fabrication process for GaAs vertical optoelectronic integrated circuits.

From 1988 to 1990, he was with Bell Communications Research, Red Bank, NJ, where he invented the two-dimensional phase-locked VCSEL array, the front-surface-emitting laser, and the high-speed lateral HBT. In 1991, he became Manager of a DRAM design group with Hyundai Electronics and designed a family of fast-1 MDRAMs and synchronous DRAMs, including 256-M SDRAM. From 1995 to 1997, he was a faculty member with Kangwon National University. In 1998, he joined the faculty of the Department of Electrical Engineering, KAIST. In 2001, he founded a national research center, System Integration and IP Authoring Research Center (SIPAC), funded by the Korean government to promote worldwide IP authoring and its SoC application. From 2003 to 2005, he was the Project Manager for SoC with the Korea Ministry of Information and Communication. His current interests are SoC design, IP authoring, high-speed and low-power memory circuits and architectures, design of embedded memory logic, optoelectronic integrated circuits, and novel devices and circuits. He is the author of the books *DRAM Design* (Hongleung, 1996; in Korean) and *High Performance DRAM* (Sigma, 1999; in Korean).

Dr. Yoo was the recipient of the Electronic Industrial Association of Korea Award for his contribution to DRAM technology in 1994 and the Korea Semiconductor Industry Association Award in 2002.