A Low-Power Unified Arithmetic Unit for Programmable Handheld 3-D Graphics Systems

Byeong-Gyu Nam, *Student Member, IEEE*, Hyejung Kim, *Student Member, IEEE*, and Hoi-Jun Yoo, *Senior Member, IEEE*

***Abstract*—A low-power, area-efficient four-way 32-bit multifunction arithmetic unit has been developed for programmable shaders for handheld 3-D graphics systems. It adopts the logarithmic number system (LNS) at the arithmetic core for the single-cycle throughput and the small-size low-power unification of various complicated arithmetic operations such as power, logarithm, trigonometric functions, vector-SIMD multiplication, division, square root and vector dot product. 24-region and 16-region piecewise linear logarithmic and antilogarithmic converters are proposed with 0.8% and 0.02% maximum conversion error, respectively. All the supported operations are implemented with less than 6.3% operation error and unified into a single arithmetic platform with maximum four-cycle latency and single-cycle throughput. A 93 K gate test chip is fabricated using one-poly five-metal 0.18-μm CMOS technology. It operates at 210 MHz with maximum power consumption of 15.3 mW at 1.8 V.**

***Index Terms*—Computer arithmetic, handheld systems, logarithmic number system, 3-D computer graphics, unified arithmetic unit.**

I. Introduction

MODERN graphics processing units (GPUs) for 3-D graphics systems adopt programmable processors, known as *shaders*, in 3-D graphics pipeline stages to provide more realistic images [1]–[4]. The shaders use not only addition and multiplication but also very complicated operations like division, square root and elementary functions like power, logarithm, and trigonometric functions. Fig. 1 shows the instruction cycle counts required for the standard OpenGL transformation and lighting (TnL) running on a commercial DSP [5]. However, it is a challenging issue to realize these various kinds of operations on a handheld 3-D graphics platforms, for example the RamP [1], that have small footprints and limited power supply.

There have been studies on multifunction units dealing with various elementary functions for high-end graphics systems [6], [7]. However, they did not take into account the power consumption and unification of elementary function with vector arithmetic functions, fundamental operations in shaders. Although [8] presents a matrix processor based on the logarithmic domain arithmetic for power-efficient design, its power consumption is still not optimized to the handheld systems and the operation set is not suitable for the 3-D graphics shaders. In this paper, we present a unified vector and elementary function unit for mathematical operations used in the 3-D graphics programmable shaders. It unifies elementary functions including power, logarithm to any base and trigonometric functions with vector-SIMD multiplication, division, square root and vector dot product in a single arithmetic platform.

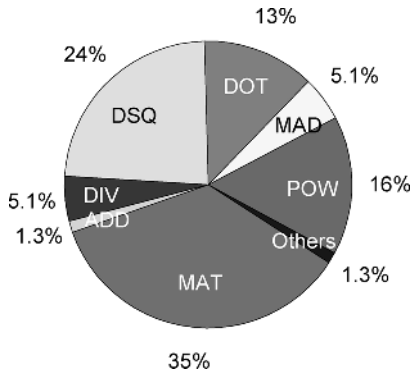


Fig. 1. Cycle count breakdown for each operation in geometry transformation and lighting.

In addition to this issue, the precision requirement is another critical issue for the arithmetic in 3-D graphics. As the screen resolution of the handheld systems gets higher, the precision requirement for the 3-D graphics APIs increases from fixed-point to 24-bit floating-point [4] and this is expected to increase to 32–bit full-precision floating-point in the future graphics APIs. However, in this work, 32-bit fixed-point precision is adopted because the precision requirement for the small screens like QVGA, widely adopted for today's cell phones, can be covered by this precision, as illustrated in [9]. Thus, the proposed arithmetic unit operates on the 32-bit fixed-point input/output values and adopts the logarithmic number system (LNS) at the arithmetic core for the small-size, low-power unification, and single-cycle throughput for all the operations.

Section II proposes the fixed-point hybrid number system (FXP-HNS) for our functional unit and logarithmic and antilogarithmic converters for this number system. Based on these, a unified functional unit is described in Section III. The simulation and implementation results for the proposed functional unit are presented in Section IV, and finally, the conclusion is summarized in Section V.

II. Fixed-Point Hybrid Number System

It is well known that the logarithmic number system (LNS) can simplify various arithmetic operations [10]. However, the

Manuscript received December 1, 2006; revised March 26, 2007.

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

Digital Object Identifier 10.1109/JSSC.2007.900243

TABLE I
OPERATIONS IN ORDINARY AND LOGARITHMIC ARITHMETIC

Operation	Ordinary Arithmetic	Logarithmic Arithmetic
Multiplication	$x \times y$	$X + Y$
Division	$x \div y$	$X - Y$
Square Root	$\sqrt{x}$	$X \gg 1$
Addition	$x + y$	$X + \log_2(1 + 2^{Y-X})$
Subtraction	$x - y$	$X + \log_2(1 - 2^{Y-X})$

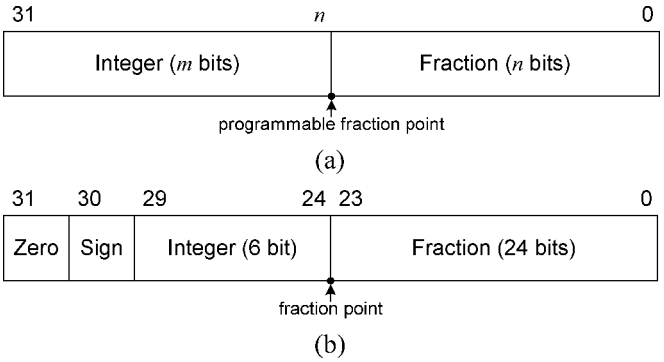


Fig. 2. Number format of FXP-HNS. (a) Fixed-point number. (b) Logarithmic number.

addition and subtraction become more complicated in LNS, where they require nonlinear function evaluation as shown in Table I. A hybrid approach of the LNS and floating-point number system (FLP) was introduced to solve this problem in [11], where the addition and subtraction are performed in FLP while other operations are done in the LNS. However, this approach still requires the complicated and power-consuming floating-point addition and subtraction, which require denormalization for exponent alignment and normalization of the final result.

In this paper, we propose the fixed-point hybrid number system (FXP-HNS) exploiting only the strong points of the fixed-point number system (FXP) and the LNS; simple addition/subtraction in FXP and other complicated operations in LNS. While most 3-D graphics systems integrate complicated power-consuming floating-point units, the 3-D graphics systems based on the FXP show improved power and area efficiency as reported in [9], [12]. Hence, our approach results in simple implementation of complicated arithmetic functions with small size and low power consumption.

The formats of fixed-point and logarithmic numbers in the FXP-HNS are shown in Fig. 2. The fixed-point number has the format of $Qm.n$, where m and n represent the number of bits for integer and fraction parts, respectively. It has programmable precision to deal with the different precision requirements of shader programs. The precision can be programmed from $Q32.0$ to $Q1.31$. The values covered by this format are $[-2^{-31} \ldots 2^{31} - 1]$.

Since the logarithms for zero and negative value are not defined, the absolute value is used for logarithmic conversion and the *zero* and *sign* are encoded into additional bits. Therefore, the format of the logarithmic number has the *sign* and *zero* bits and 6-bit integer to cover the entire range of the corresponding fixed-point number and 24-bit fraction.

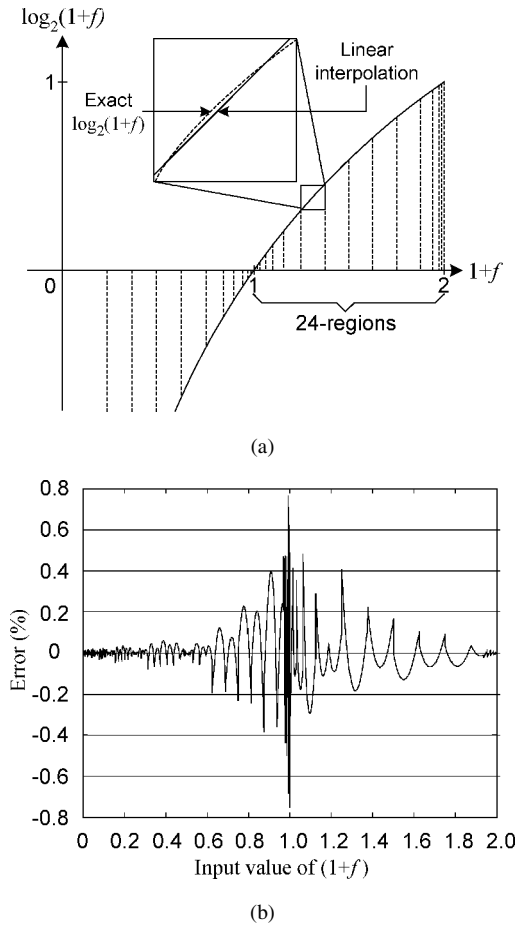


Fig. 3. Proposed logarithmic conversion scheme. (a) Logarithmic conversion scheme. (b) Error graph.

For the conversion between the fixed-point and logarithmic numbers in the FXP-HNS, the logarithmic and antilogarithmic converters are proposed. These use piecewise linear approximation for the low-power and small-area implementation.

A. Logarithmic Converter

In [13]–[16], simple piecewise linear approximation schemes using small coefficient lookup tables were proposed for the power- and area-efficient implementation. Although these reduce the conversion error for integer inputs quite well, the logarithmic converters revealed quite large relative error for real inputs around 1. Therefore, we propose an error control scheme for the logarithmic converters.

When x is a 32-bit FXP input, the x and its logarithm can be represented as (1) and (2), respectively:

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

$$\log_2 x = k + \log_2(1 + f). \tag{2}$$

The $k \in \{-31, -30, \ldots, 31\}$ is the characteristic of the logarithm and $\log_2(1 + f)$ is the fractional part in the range

TABLE II
COEFFICIENT LOOKUP TABLE FOR LOGARITHMIC CONVERTER

f	a_i	b_i	f	a_i	b_i
$[0, 256/2^{14})$	$23/2^4$	0	$[15,360/2^{14}, 15,616/2^{14})$	$49/2^6$	$1,935/2^{13}$
$[256/2^{14}, 512/2^{14})$	$45/2^5$	$7/2^{14}$	$[15,616/2^{14}, 15,872/2^{14})$	$193/2^8$	$16,191/2^{16}$
$[512/2^{14}, 1,024/2^{14})$	$351/2^8$	$7/2^{12}$	$[15,872/2^{14}, 16,128/2^{14})$	$95/2^7$	$2,115/2^{13}$
$[1,024/2^{14}, 2,048/2^{14})$	$21/2^4$	$6/2^{10}$	$[16,128/2^{14}, 16,256/2^{14})$	$95/2^7$	$67,635/2^{18}$
$[2,048/2^{14}, 3,072/2^{14})$	$159/2^7$	$31/2^{11}$	$[16,256/2^{14}, 16,320/2^{14})$	$189/2^8$	$34,315/2^{17}$
$[3,072/2^{14}, 4,096/2^{14})$	$19/2^4$	$26/2^{10}$	$[16,320/2^{14}, 16,352/2^{14})$	$189/2^8$	$137,239/2^{19}$
$[4,096/2^{14}, 6,144/2^{14})$	$35/2^5$	$51/2^{10}$	$[16,352/2^{14}, 16,368/2^{14})$	$189/2^8$	$137,227/2^{19}$
$[6,144/2^{14}, 8,192/2^{14})$	$257/2^8$	$86/2^{10}$	$[16,368/2^{14}, 16,376/2^{14})$	$189/2^8$	$68,611/2^{18}$
$[8,192/2^{14}, 10,240/2^{14})$	$119/2^7$	$123/2^{10}$	$[16,376/2^{14}, 16,380/2^{14})$	$189/2^8$	$137,219/2^{19}$
$[10,240/2^{14}, 12,288/2^{14})$	$110/2^7$	$335/2^{11}$	$[16,380/2^{14}, 16,382/2^{14})$	$189/2^8$	$548,871/2^{21}$
$[12,288/2^{14}, 14,336/2^{14})$	$51/2^6$	$215/2^{10}$	$[16,382/2^{14}, 16,383/2^{14})$	$189/2^8$	$274,433/2^{20}$
$[14,336/2^{14}, 15,360/2^{14})$	$3/2^2$	$257/2^{10}$	$[16,383/2^{14}, 1)$	$189/2^8$	$4,390,913/2^{24}$

of [0,1). In piecewise linear approximation, the nonlinear term $\log_2(1+f)$ is approximated as

$$\log_2(1+f) \cong a_i \cdot f + b_i \tag{3}$$

where a_i and b_i are the approximation coefficients defined for each approximation region i in the range of $1 \leq (1+f) < 2$. Here, we divide the input $(1+f)$ into finer subdivisions around the input of 1, since the error increases as the input value gets closer to 1. This results in 24 approximation regions, and the coefficients for each region are listed in Table II.

This logarithmic conversion scheme is shown in Fig. 3(a). Its maximum conversion error for the real numbers is less than 0.8%. Fig. 3(b) shows the error graph for logarithmic conversion. The logarithmic converter based on this scheme is shown in Fig. 4. The input x is converted into its absolute value through the ABS block. The LOD block computes the characteristic value k' by detecting the leading one bit. It takes the bits following the leading one as the fraction value f. The 14 most significant bits (MSBs) of f are used for addressing the lookup table in the APP block. The APP block approximates the nonlinear fractional part $\log_2(1+f)$ by $a_i \cdot f + b_i$ and its multiplication is implemented by summation of shift terms. Therefore, the APP carries out the summation of five terms using the CSA tree and CPA. Since this converter receives input values with variable precision of $Qm.n$, the actual characteristic value k is computed by $k' - n$. The conversion result is obtained by packing the *zero*, *sign*, characteristic value, and the approximated fraction value into 32 bits logarithmic number.

B. Antilogarithmic Converter

The computation results in the logarithmic number should be converted into the fixed-point number to make the final computation results. Therefore, a power- and area-efficient antilogarithmic converter also based on the piecewise linear approximation scheme is proposed.

TABLE III
COEFFICIENT LOOKUP TABLE FOR ANTILOGARITHMIC CONVERTER

f	a_i	b_i	f	a_i	b_i
$[0, 1/16)$	$91/2^7$	$65,524/2^{16}$	$[8/16, 9/16)$	1	$936/2^{10}$
$[1/16, 2/16)$	$47/2^6$	$16,359/2^{14}$	$[9/16, 10/16)$	$67/2^6$	$909/2^{10}$
$[2/16, 3/16)$	$99/2^7$	$2,035/2^{11}$	$[10/16, 11/16)$	$35/2^5$	$879/2^{10}$
$[3/16, 4/16)$	$13/2^4$	$8,079/2^{13}$	$[11/16, 12/16)$	$73/2^6$	$846/2^{10}$
$[4/16, 5/16)$	$27/2^5$	$8,013/2^{13}$	$[12/16, 13/16)$	$19/2^4$	$810/2^{10}$
$[5/16, 6/16)$	$113/2^7$	$989/2^{10}$	$[13/16, 14/16)$	$5/2^2$	$758/2^{10}$
$[6/16, 7/16)$	$117/2^7$	$977/2^{10}$	$[14/16, 15/16)$	$83/2^6$	$716/2^{10}$
$[7/16, 8/16)$	$123/2^7$	$956/2^{10}$	$[15/16, 1)$	$87/2^6$	$656/2^{10}$

When x is a logarithmic number, the LNS representation of the x is composed of integer part k and fractional part f. The antilogarithmic conversion of x can be represented as

$$2^x = 2^k \cdot 2^f. \tag{4}$$

In this equation, 2^k is just a shift operation and the nonlinear term 2^f can be approximated by piecewise linear interpolation as given in

$$2^f \cong a_i \cdot f + b_i \tag{5}$$

where a_i and b_i are the approximation coefficients defined for each approximation region i in the range of $0 \leq f < 1$. We divide the f into 16 even approximation regions since the error for the antilogarithmic conversion is evenly spread over the entire region. The coefficients for each approximation region are listed in Table III.

This antilogarithmic conversion scheme is shown in Fig. 5(a). Its maximum conversion error for the real numbers is less than 0.02%. Fig. 5(b) shows the error graph for the antilogarithmic conversion. The antilogarithmic converter is shown in Fig. 6. The input logarithmic number is split into the characteristic value k and fractional value f. The 4 MSBs of f are used for addressing the lookup table in the APP block. The APP block

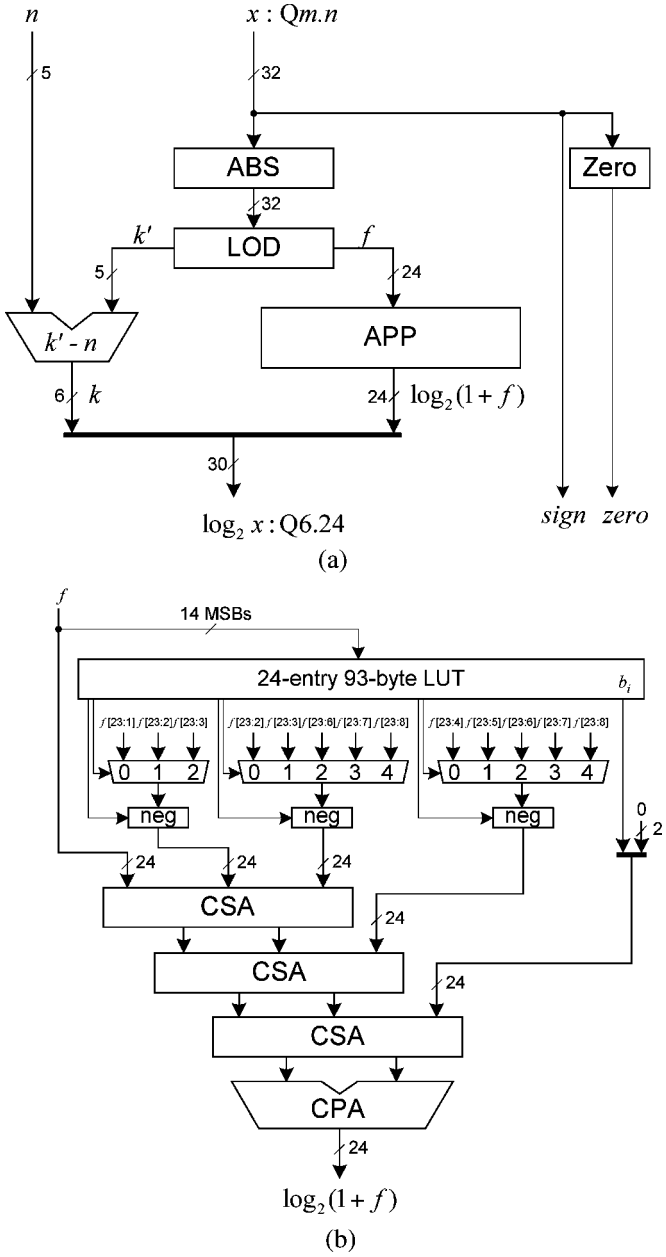


Fig. 4. Block diagram of logarithmic converter. (a) Logarithmic converter. (b) Approximation block.

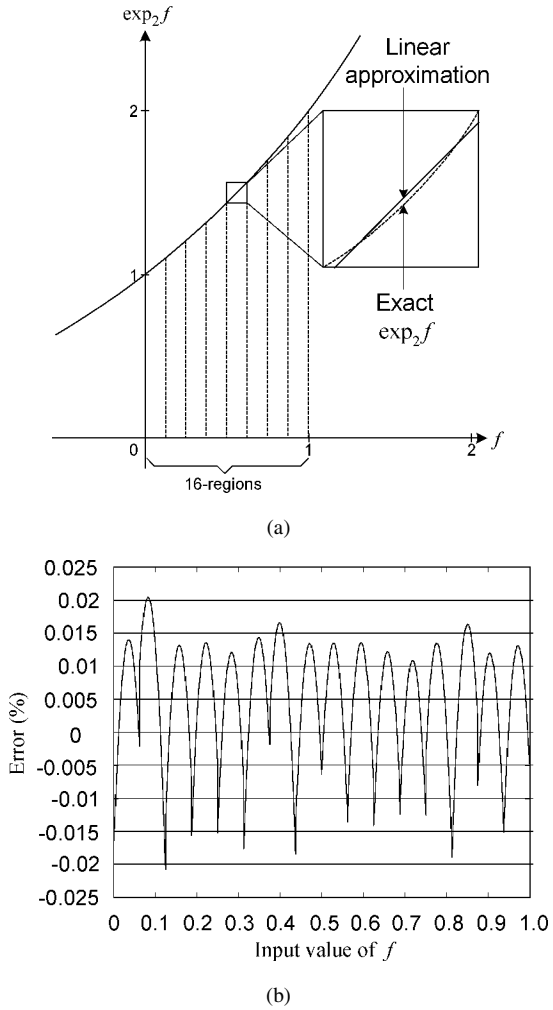


Fig. 5. Proposed antilogarithmic conversion scheme. (a) Antilogarithmic conversion scheme. (b) Error graph.

approximates the nonlinear part 2^f by $a_i \cdot f + b_i$ and its multiplication is also implemented by summation of shift terms. The summation of five terms is carried out using the CSA tree and CPA. The enabled *sign* bit negates these terms for negative output. The shift amount is computed by $n - 24 + k$ since the final result should be adjusted to the format with the input precision, i.e., $Qm.n$. Finally, the enabled *zero* bit selects the output of zero.

III. Unified Arithmetic Unit

A. Overall Architecture

A unified arithmetic unit is presented based on FXP-HNS. It unifies the vector arithmetic operations with elementary functions in a single four-way arithmetic platform, the standard arithmetic platform for modern programmable shaders [2]. It has fully pipelined architecture with four pipeline stages. In this architecture, the input operands are converted into the logarithmic domain, where complicated operations are reduced into simple operation, and the results are restored into fixed-point domain, where simple additions and subtractions are carried out.

The overall block diagram of the proposed unit is shown in Fig. 7. The eight 32-bit fixed-point input operands, x_i and y_i in each channel, are converted into 32-bit logarithmic number through logarithmic converters at the first stage (LOGC stage). The second stage (LNS stage) contains a programmable carry save adder (CSA) tree for a single 30b $\times$ 32b multiplier or four-way 30b $\times$ 8b multipliers according to target elementary functions. The final summation of the carry and partial sum from the CSA tree can be carried out with the four 38-bit carry propagation adders in the LNS stage. These are also used with four 1-bit shifters for the vector arithmetic operations. The third stage (ALOGC stage) converts the 38-bit computation results from the four adders in logarithmic number into 32-bit fixed-point number with the given precision of input operand through antilogarithmic converters. The final stage (ADD stage) consists

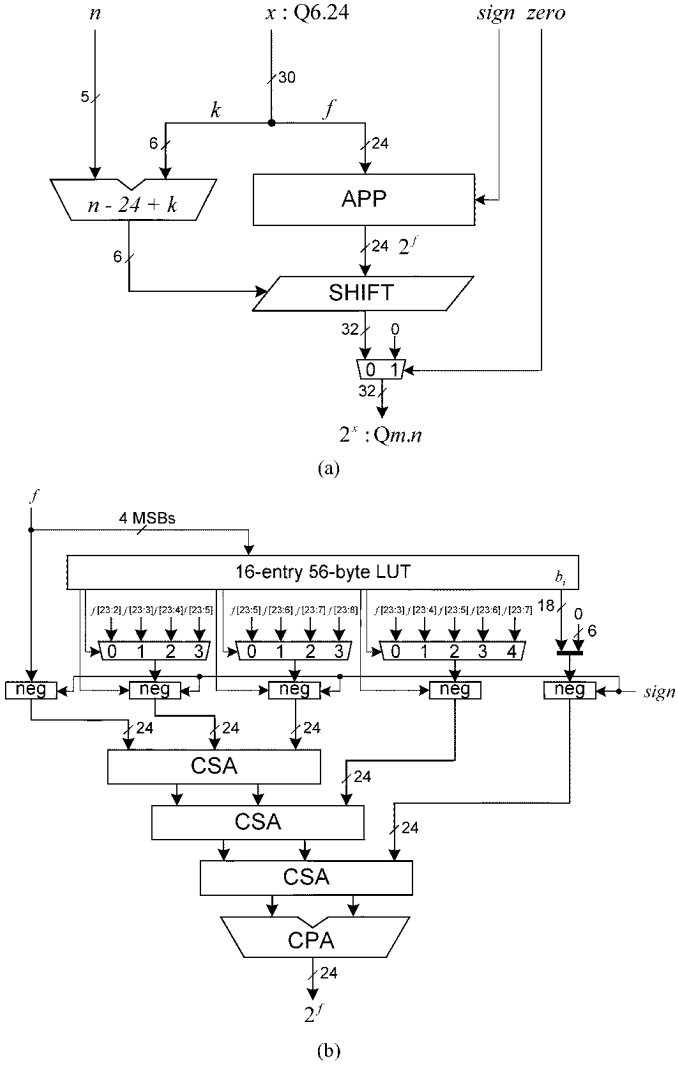


Fig. 6. Block diagram of antilogarithmic converter. (a) Antilogarithmic converter. (b) Approximation block.

of a carry-propagation adder (CPA) tree that can be programmed into a single adder tree with five 32-bit input operands or independent four-way two-input 32-bit adders according to the target operations.

B. Vector Operations

In conventional architecture, only vector-SIMD multiplication, multiply-and-add and vector dot product are implemented for the vector operations [2], [9]. In our proposed architecture, the division and square root are also defined as vector-SIMD operations since it is useful for vector-SIMD $x \div \sqrt{y}$ calculation used in vector normalizations of 3-D graphics algorithms. The multiplication, division, square root and multiply-and-add (MAD) can be represented by a single generic operation defined as

$$(x_i \otimes y_i^p \oplus z_i)_{i\in\{0,1,2,3\}} \quad \text{where } \otimes \in \{\times, \div\},\ \oplus \in \{+,-\},\ p \in \{0.5, 1\}. \tag{6}$$

For example, operations such as $x \times y$, $x \div y$, $x \times y + z$, $x \div \sqrt{y} - z, \ldots$, etc., can be represented with this generic operation. This expression can be converted into the expression in FXP-HNS as

$$\left(2^{(\log_2 x_i)\oplus(\log_2 y_i \gg q)} \oplus z_i\right)_{i\in\{0,1,2,3\}} \quad \text{where } \oplus \in \{+,-\},\ q \in \{0,1\}. \tag{7}$$

According to (7), multiplication, division, square, and square root can be converted into the addition, subtraction, left shift, and right shift in the logarithmic number domain, respectively. In this architecture, the square root can be computed in conjunction with multiplication or division without any additional cycles. The diagram for this vector-SIMD unit is also shown in Fig. 7. In this figure, the LNS stage is augmented with a carry-save adder (CSA) tree for elementary functions. The dot product in FXP-HNS of expression (8) is also implemented using the vector-SIMD unit for vector element multiplication.

$$\sum_{i=0}^{i=3} x_i \times y_i = \sum_{i=0}^{i=3} 2^{\log_2 x_i + \log_2 y_i}. \tag{8}$$

For the summation of the multiplication results, the four-way two-input 32-bit adders used for the vector-SIMD MAD operation are programmed into a single five-input 32-bit adder tree as shown in Fig. 8. In this case, the *bias* port is disabled, but it will be used for trigonometric operations.

C. Elementary Functions

The elementary functions such as power of arbitrary exponent, logarithm to arbitrary base, trigonometric, inverse trigonometric, hyperbolic, and inverse hyperbolic functions are unified with vector operations.

The use of Taylor series expansions leads to the unification of trigonometric, hyperbolic and their inverse functions with the vector operations. For the first five terms of the Taylor series computation, a new generic operation is defined as

$$c_0 x^{k_0} \oplus c_1 x^{k_1} \oplus c_2 x^{k_2} \oplus c_3 x^{k_3} \oplus c_4 x^{k_4} \tag{9}$$

where $\oplus \in \{+,-\}$ and c_i and k_i are positive real and integer constants, respectively.

For example, Taylor series like $\sin x \simeq x - x^3/3! + x^5/5! - x^7/7! + x^9/9!$ and $\cos x \simeq 1 - x^2/2! + x^4/4! - x^6/6! + x^8/8!$ can be represented by this generic operation. The computation of the powers required for this operation can be converted into multiplications in logarithmic number domain by transforming (9) into FXP-HNS expression as

$$c_0 x^{k_0} \oplus 2^{(\log_2 c_1 + k_1 \times \log_2 x)} \oplus 2^{(\log_2 c_2 + k_2 \times \log_2 x)} \oplus 2^{(\log_2 c_3 + k_3 \times \log_2 x)} \oplus 2^{(\log_2 c_4 + k_4 \times \log_2 x)} \tag{10}$$

where $\oplus \in \{+,-\}$, $\log_2 c_i$ and k_i are real and positive integer constants, respectively. Each term $2^{(\log_2 c_i + k_i \times \log_2 x)}$ in (10) requires an addition and a multiplication in logarithmic number domain with constant $\log_2 c_i$ and x converted into logarithmic number. The first term $c_0 x^{k_0}$ in (10) is not converted into the logarithmic number domain since the first term of the Taylor series expansion tends to be just a constant or the input x. For the

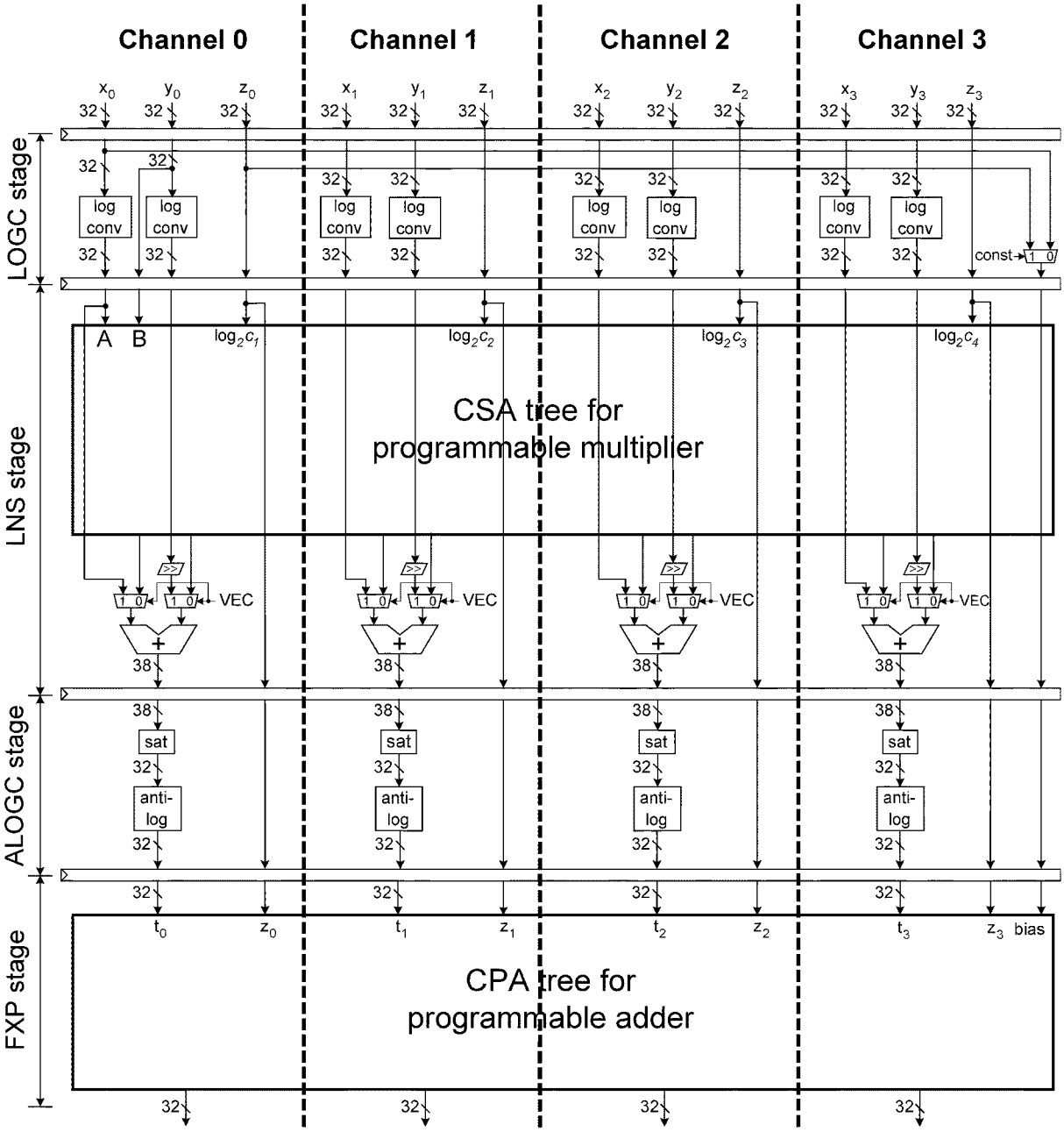


Fig. 7. Unified arithmetic unit in FXP-HNS.

other four terms, a 30b × 8b radix-4 Booth multiplier is sufficient since the k_i in each term is small integer that can be represented within 8 bits. Therefore, the multiplications required for (10) can be computed using four 30b × 8b multipliers. Fig. 9 shows that each 30b × 8b CSA tree has an extra input port to add the coefficient $\log_2 c_i$, which is presented as an input operand to each channel. The four-way 38-bit CPAs to add the carry and partial sum of $Q14.24$ from each CSA tree are shared with the four-way adders in LNS stage for the vector arithmetic operations. The 14-bit integers are saturated into 8-bit values by testing the overflow or underflow conditions. The saturated results of $Q8.24$ are converted into 32-bit fixed-point number through antilogarithm converters.

The $\oplus$ operations in (10) can be computed with the five-input adder tree shown in Fig. 8 by programming each CPA into an adder or a subtractor according to the target Taylor series. The first term of the Taylor series is directly fed into this adder tree through the *bias* port. The final result can be obtained from the channel 2 of the ADD stage.

The power function can be converted into the multiplication in logarithmic number domain according to

$$x^y = 2^{(y \times \log_2 x)}. \tag{11}$$

Therefore, a radix-4 30b × 32b multiplier is required in LNS stage. As shown in Fig. 9, this multiplier can be obtained by programming the four 30b × 8b multipliers used for the trigonometric functions into a single 30b × 32b multiplier and this

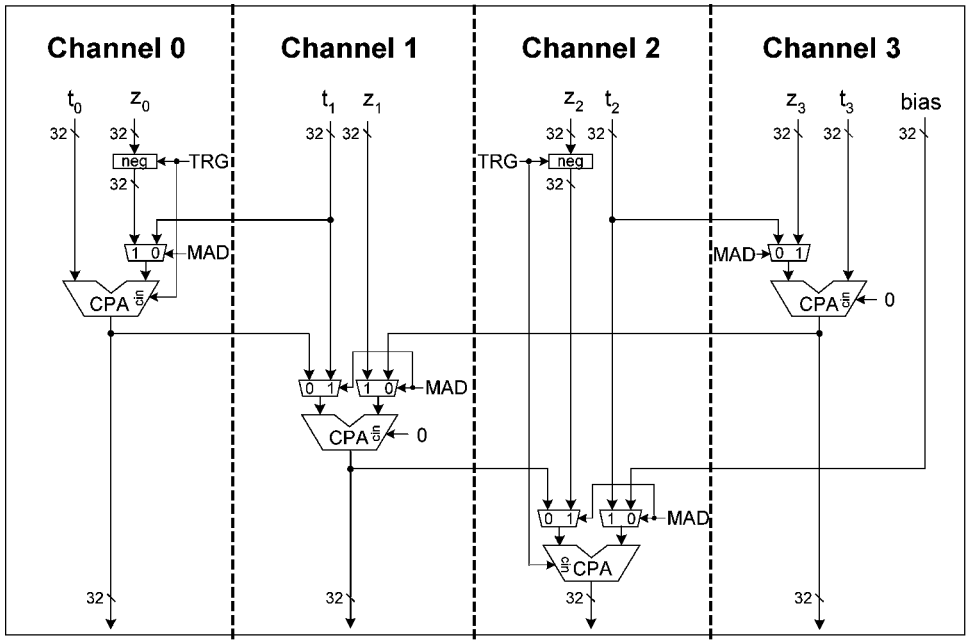


Fig. 8. CPA tree for programmable adder.

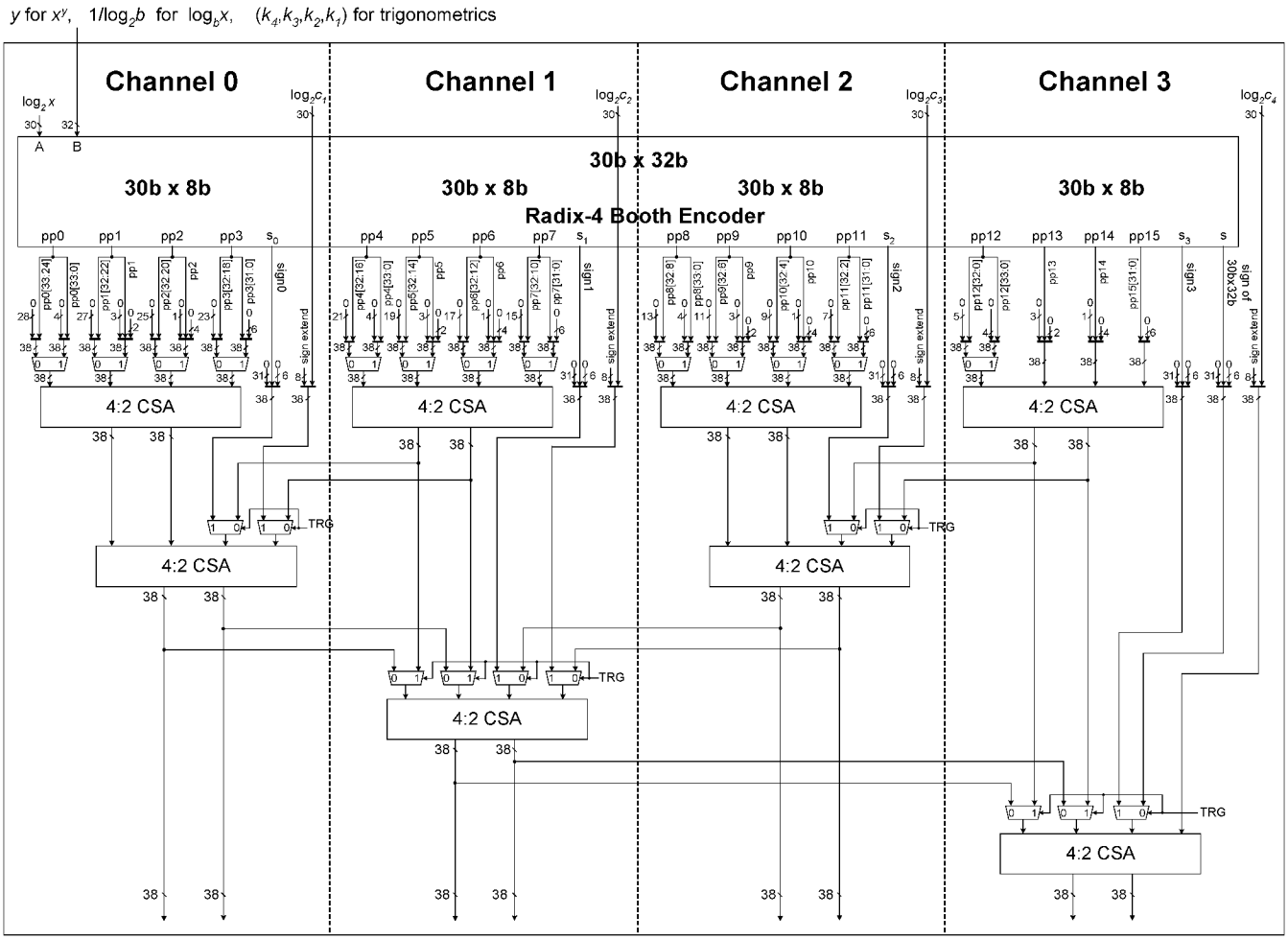


Fig. 9. CSA tree for programmable multiplier.

One - Hot Booth Encoding Table

Multiplier bits $b_{i+1}\, b_i\, b_{i-1}$	Encoded Operation
000	0
001	+ Multiplicand
010	+ Multiplicand
011	+ 2 x Multiplicand
100	- 2 x Multiplicand
101	- Multiplicand
110	- Multiplicand
111	0

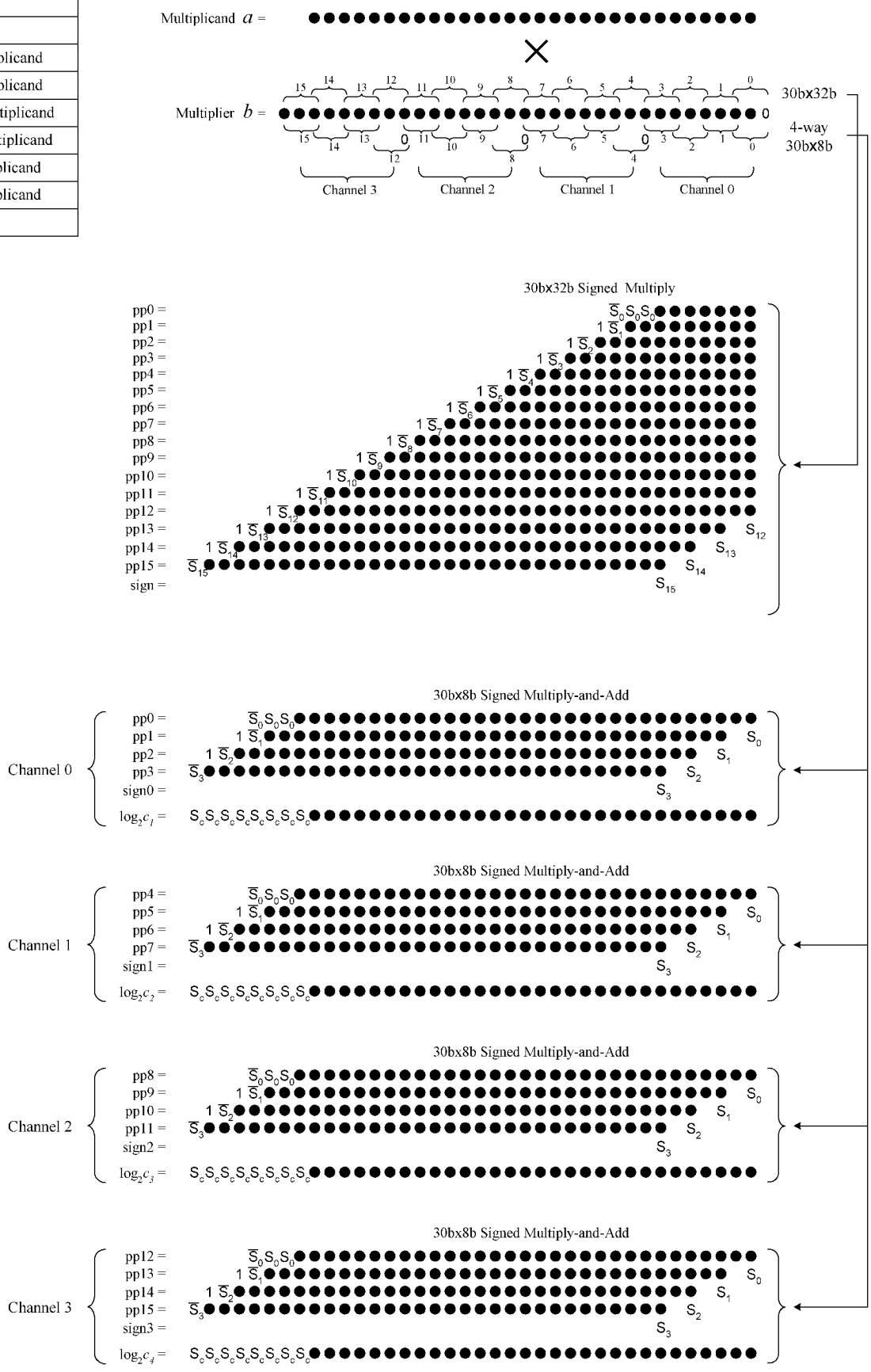
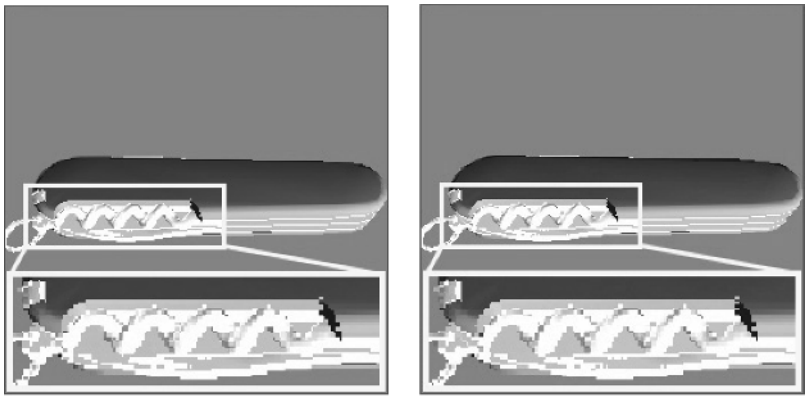


Fig. 10. Dot diagram for programmable multiplication scheme.

TABLE IV
THE ERROR OF EACH OPERATION

TYPE	OPERATIONS	MAX ERROR (%)	AVG ERROR (%)
VECTOR OPERATIONS	Vector-SIMD multiply	0.87	0.043
	Vector-SIMD multiply-with-sqrt	1.25	0.0342
	Vector-SIMD divide	0.75	0.0097
	Vector-SIMD divide-by-sqrt	1.5	0.0195
	Vector-SIMD multiply-and-add	1.75	0.017
	Vector dot product	2.3	0.043
ELEMENTARY FUNCTIONS	Power	2.9	0.088
	Logarithm	1.2	0.04
	Sine	0.3	0.012
	Cosine	2.5	0.134
	Arctangent	6.3	0.303



(a) (b)

Fig. 11. Comparison of TnL effects. (a) Scene from FXP arithmetic. (b) Scene from FXP-HNS arithmetic.

programmable multiplication scheme is shown in Fig. 10. Since the y is in the range of [0,128] in specular lighting [17], for which the power function is mainly used, it has the format of $Q8.24$. Therefore, the CSA tree for the 30b $\times$ 32b multiplier computes only 38 MSBs, truncating the 24 LSBs of fractional part, to make 24-bit fraction value for the antilogarithmic converter. These are added using the 38-bit CPA used for the vector arithmetic operations in channel 3. The multiplication result is saturated into $Q8.24$ by testing the overflow or underflow conditions. The saturated result is converted into 32-bit fixed-point format through antilogarithmic converter. The final result can be obtained from the ALOG stage of channel 3.

The logarithm to arbitrary base can be computed by multiplying a constant with the binary logarithm, according to

$$\log_b x = k \times \log_2 x \quad \text{where} \quad k = 1/\log_2 b. \tag{12}$$

The multiplication of the constant k can be done using the 30b $\times$ 32b multiplier used in the power function. The 32-bit constant k of $Q8.24$ should be given as an input operand. The final result can also be obtained from channel 3 of the ALOG stage.

IV. IMPLEMENTATION RESULTS

A. Simulations

The proposed function unit has been simulated in 3-D graphics software environment with FXP and FXP-HNS libraries. The maximum and average errors for each operation are summarized in Table IV. The high error rate for the arctangent operation is mainly from the limited number of five terms used in Taylor series expansion rather than the conversion errors from the logarithmic and antilogarithmic converters. This error can be reduced by multiple executions of arctangent operations changing the coefficients and accumulating the results to increase the number of terms for Taylor series expansion. The test 3-D graphics scenes are shown in Fig. 11 to show the reliability of the proposed arithmetic unit. The test model in Fig. 11 consists of 5878 polygons and the screen size is 512 $\times$ 512. In Fig. 11, the OpenGL TnL operation [17] is tested to show the geometry stage effects of various operations including vector dot product, vector normalization, power, and trigonometric functions. The average lighting error between the FXP and FXP-HNS arithmetic is 0.026%. This error is within the tolerable range for the small screens of the handheld systems. The in-box shows a zoomed image for the accuracy comparison. Unnoticeable error can be found with naked eyes between two images.

B. Chip Implementation

A test chip is fabricated by TSMC one-poly five-metal 0.18-μm CMOS technology. A micrograph of the 93 K gate chip is shown in Fig. 12, lined out with the regions for each pipeline stage. Its core size is 2.9 mm^2. The chip characteristics are shown in Table V. The area breakdown of the chip is shown

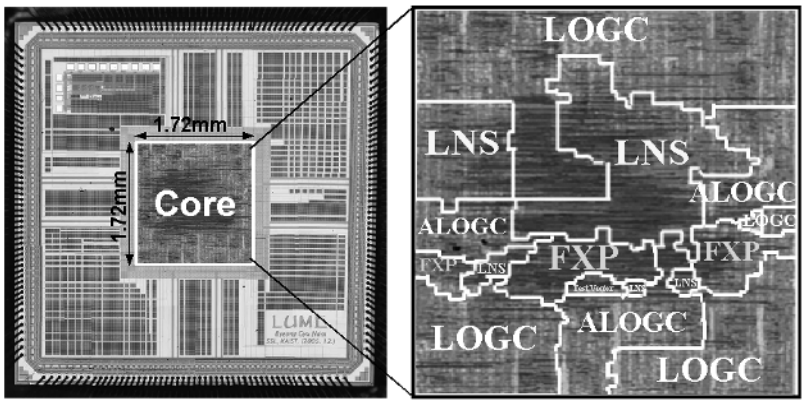


Fig. 12. Chip micrograph.

TABLE V
CHIP CHARACTERISTICS

Process Technology	1-poly 5-metal TSMC 0.18um CMOS technology
Power Supply	Core: 1.8V, I/O: 3.3V
Operating Frequency	210MHz
Power Consumption	Max. 15.3mW
Gate Count	93K
Area	Core: 1.72mm × 1.72mm Die: 5mm × 5mm
Package	208pin LQFP

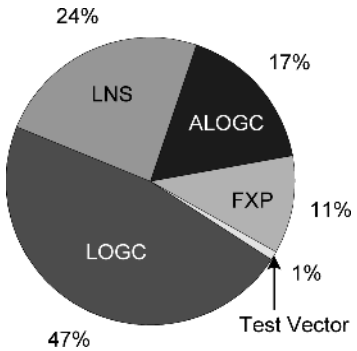


Fig. 13. Area breakdown of the chip.

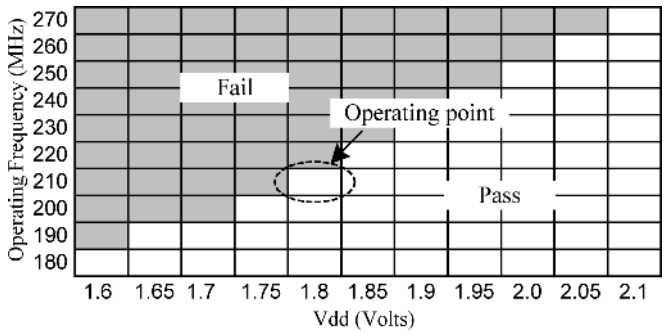


Fig. 14. Shmoo plot.

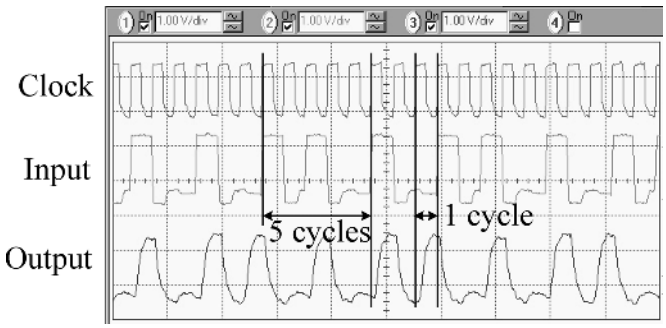


Fig. 15. Measured waveform.

in Fig. 13. As shown in this graph, about half of the chip area is taken by the logarithmic converters.

The pipeline registers are clock gated by the control of each operation to reduce switching power consumption. Table VI summarizes the power consumption and latency of each operation. The test chip was measured with a current meter when the chip operates with the embedded test vectors. The maximum power consumption is 15.3 mW at 1.8 V for arctangent operation, which maximally turns on the computing logics in entire pipeline stages including a logarithmic converter, programmable multiplier, four antilogarithmic converters and programmable adder.

The pipeline operates with single-cycle throughput at 210 MHz. The shmoo plot is given in Fig. 14. The measured waveforms of the arctangent operation are shown in Fig. 15. The five-cycle latency represents internal four-cycle latency with an additional cycle caused by output registers, inserted for the stable output test.

C. Comparison

For the comparison with other works, we selected the vector-SIMD multiply-and-add (VMAD) operation, the common operation among the compared works and useful for the 3-D geometry transformation. Table VII shows the comparison results. As the processing speed gets higher, the area and power consumption also increases. Since the area and power consumption are also important factors for handheld devices, we use the following figure of merit (FoM) for the comparison with other works.

$$\text{FoM} = \frac{\text{Performance(VMAD/s)} \times \text{Accuracy(\%)}}{\text{Power(mW)} \times \text{Area(mm}^2\text{)}}. \quad (13)$$

It takes into account the area and power consumption, and the accuracy is also taken into account to evaluate the effect of errors from our chip. According to this FoM, this work shows

TABLE VI
POWER/LATENCY OF EACH OPERATION

TYPE	OPERATIONS	POWER (mW)	LATENCY / THROUGHPUT (cycles)
VECTOR OPERATIONS	Vector-SIMD multiply	8.03	3 / 1
	Vector-SIMD multiply-with-sqrt	8.1	3 / 1
	Vector-SIMD divide	8.15	3 / 1
	Vector-SIMD divide-by-sqrt	8.3	3 / 1
	Vector-SIMD multiply-and-add	10.1	4 / 1
	Vector dot product	9.8	4 / 1
ELEMENTARY FUNCTIONS	Power	10.2	3 / 1
	Logarithm	11.8	3 / 1
	Sine	15.2	4 / 1
	Cosine	15.1	4 / 1
	Arctangent	15.3	4 / 1

TABLE VII
COMPARISON OF RELATED FIXED-POINT DESIGNS

Reference	Performance (VMAD/s)	Accuracy (%)	Power (mW)	Area (mm^2)	FoM (VMAD·% /s/mW/mm^2)	Technology (μm)
[18]	200M	100	20.1	3.15	315.88M	0.18
This work	210M	98.25	10.1	2.96	690.14M	0.18

TABLE VIII
COMPARISON WITH FLOATING-POINT DESIGNS

Reference	Performance (VMAD/s)	Accuracy (%)	Power (mW)	Area (Ktransistors)	Technology (μm)
[19]	400M	100	250	N.A.	0.13
[20]	5600M	100	1400	768	0.09
This work	210M	98.25	10.1	372	0.18

2.18 times higher value than others despite the richer operation set supported. In Table VIII, this work is also compared with floating-point based designs to show the gains from fixed-point design.

V. CONCLUSION

An area-efficient, low-power unified functional unit is proposed. It integrates vector and elementary functions required for programmable 3-D graphics shaders into a single arithmetic platform. The fixed-point hybrid number system (FXP-HNS) is proposed for the small-size, low-power unification and single-cycle throughput implementation of these operations. A proper programming of the CSA tree and CPA tree can make all these operations. New logarithmic converter and antilogarithmic converters are proposed with maximum 0.8% and 0.02% errors, respectively. Implementation results show that the FXP-HNS arithmetic is useful for handheld 3-D graphics systems even though it carries a little computation error. A test chip operates at 210 MHz with 15.3 mW power consumption at 1.8 V. All the operations are integrated with 93 K gates and achieve single-cycle throughput with maximum four-cycle latency. The comparison result shows that our work achieves 2.18 times improvement according to the proposed figure of merit.

REFERENCES

[1] J.-H. Sohn, Y.-H. Park, C.-W. Yoon, R. Woo, S.-J. Park, and H.-J. Yoo, "Low-power 3D graphics processors for mobile terminals," *IEEE Commun. Mag.*, vol. 43, no. 12, pp. 90–99, Dec. 2005.

[2] E. Lindholm, M. J. Kilgard, and H. Moreton, "A user-programmable vertex engine," in *Proc. ACM SIGGRAPH 2001*, pp. 149–158.

[3] Microsoft Corporation, "Microsoft DirectX Technology Overview." [Online]. Available: http://www.microsoft.com/windows/directx

[4] Khronos Group, OpenGL-ES 2.0. [Online]. Available: http://www.khronos.org

[5] *"ADSP-21000 Family Application Handbook,"* Analog Devices, Jul. 1994.

[6] S. F. Oberman and M. Y. Siu, "A high-performance area-efficient multifunction interpolator," in *Proc. IEEE 17th Int. Symp. Computer Arithmetic (ARITH17)*, Jun. 2005, pp. 272–279.

[7] J.-A. Pineiro, S. F. Oberman, J.-M. Muller, and J. D. Bruguera, "High-speed function approximation using a minimax quadratic interpolator," *IEEE Trans. Computers*, vol. 54, no. 3, pp. 304–318, Mar. 2005.

[8] S. Pan, Y. Ben-Arie, E. Orian, I. Barak, Y. Shapira, S. Bresticker, H. David, H. Folkman, J. Efrat, L. Tzukerman, Z. Dahan, D. Kolton, and Y. Shvager, "A 32 b 64-matrix parallel CMOS processor," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) 1999 Dig. Tech. Papers*, pp. 262–263.

[9] 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*, Aug. 2004, pp. 107–114.

[10] J. N. Mitchell, Jr., "Computer multiplication and division using binary logarithms," *IRE Trans. Electronic Computers*, vol. 11, pp. 512–517, Aug. 1962.

[11] F.-S. Lai and C.-F. E. Wu, "A hybrid number system processor with geometric and complex arithmetic capabilities," *IEEE Trans. Computers*, vol. 40, no. 8, pp. 952–962, Aug. 1991.

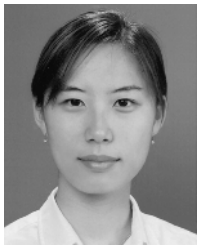
[12] G. K. Kolli, "3D graphics optimization for ARM architecture," presented at the Game Developers Conf., San Francisco, CA, 2002.

[13] 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.

[14] 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, no. 8, pp. 97–105, Feb. 1970.
[15] K. H. Abed and R. E. Siferd, "CMOS VLSI implementation of a low-power logarithmic converter," *IEEE Trans. Computers*, vol. 52, no. 11, pp. 1421–1433, Nov. 2003.
[16] K. H. Abed and R. E. Siferd, "VLSI implementation of a low-power antilogarithmic converter," *IEEE Trans. Computers*, vol. 52, no. 9, pp. 1221–1228, Sep. 2003.
[17] *OpenGL Programming Guide*, 3rd ed. Reading, MA: Addison Wesley, 1999, OpenGL ARB.
[18] J.-H. Sohn, J.-H. Woo, M.-W. Lee, H.-J. Kim, R. Woo, and H.-J. Yoo, "A fixed-point multimedia co-processor with 50 Mvertices/s programmable SIMD vertex shader for mobile applications," in *Proc. Eur. Solid-State Circuits Conf. (ESSCIRC)*, 2005, pp. 207–210.
[19] F. Arakawa, T. Yoshinaga, T. Hayashi, Y. Kiyoshige, T. Okada, M. Nishibori, T. Hiraoka, M. Ozawa, T. Kodama, T. Irita, T. Kamei, M. Ishikawa, Y. Nitta, O. Nishii, and T. Hattori, "An embedded processor core for consumer appliances with 2.8 GFLOPS and 36 Mpolygons/s," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, 2004, pp. 334–335.
[20] H.-J. Oh, S. M. Muller, C. Jacobi, K. D. Tran, S. R. Cottier, B. W. Michael, H. Nishikawa, Y. Totsuka, T. Namatame, N. Yano, T. Machida, and S. H. Dhong, "A fully pipelined single-precision floating-point unit in the synergistic processor element of a CELL processor," *IEEE J. Solid-State Circuits*, vol. 41, no. 4, pp. 759–771, Apr. 2006.

Byeong-Gyu Nam received the B.S. degree (*summa cum laude*) in computer engineering from Kyungbook National University, Daegu, Korea, in 1999, and the M.S. degree in computer science from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2001.

From 2001 to 2002, he was with the Electronics and Telecommunication Research Institute (ETRI), Daejeon, Korea. He received the Ph.D. degree in electrical engineering from KAIST in 2007. His Ph.D. research focused on low-power computer arithmetic and 3-D graphics processors for handheld applications.



Hyejung Kim received the B.S. and M.S degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2004 and 2006, respectively. She is currently working toward the Ph.D degree in the same department at KAIST.

Her research interests include low-power arithmetic unit and microprocessor design for mobile and implantable system applications.

Hoi-Jun Yoo 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 at Hyundai Electronics and designed a family of fast-1 MDRAMs and synchronous DRAMs, including 256M 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 at 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 in 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* (Seoul, Korea: Hongleung, 1996; in Korean) and *High Performance DRAM* (Seoul, Korea: Sigma, 1999; in Korean).

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