IEEE Asian Solid-State Circuits Conference
November 12-14, 2007 / Jeju, Korea

A 28.5mW 2.8GFLOPS Floating-Point Multifunction Unit for Handheld 3D Graphics Processors

Byeong-Gyu Nam and Hoi-Jun Yoo
Dept. of EECS, Korea Advanced Institute of Science and Technology (KAIST)
373-1, Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea

***Abstract*- A low-power, high-performance 4-way 32-bit floating-point multifunction unit is developed for handheld 3D graphics processors. It uses logarithmic arithmetic to unify matrix, vector, and elementary functions into a single arithmetic unit. The optimal designs of logarithmic and antilogarithmic converters are presented. An adaptive number conversion scheme is proposed and it reduces total area by 15%. With this scheme, the matrix-vector multiplication (MAT), cross-product, lerp, and logarithm ($\log_x y$ with 2 variables) are newly unified with the other operations. The unit achieves 2-cycle throughput for the MAT and single-cycle throughput for all other operations. It takes 451K transistors and achieves 2.8GFLOPS at 200MHz with 28.5mW power consumption.**

I. INTRODUCTION

Modern handheld graphics processing units (GPUs) require various operations to get realistic graphics effects [1]. In [2], a multifunction unit is proposed for this purpose. However, it was a fixed-point unit and didn't deal with the matrix-vector multiplication, required for the frequently used geometry transformation in 3D graphics. In this paper, a 4-way 32-bit floating-point (FLP) unified matrix, vector, and elementary function unit is proposed. It operates on the FLP data to meet the specification of the standard API which requires more than 24-bit FLP precision [1]. It unifies matrix-vector multiplication (MAT), vector multiplication, division, square root, multiply-add, lerp, dot-product, cross-product (CRS), and elementary functions including trigonometric functions (TRGs), power (POW) and logarithm (LOG) with 2 variables in a single 4-way arithmetic unit. The MAT, CRS, lerp, and LOG are newly unified to the previous operation set [2] with little overhead. Although it operates on the FLP data, it uses logarithmic arithmetic for the internal arithmetic. Using this, it achieves power- and area-efficient unification and single-cycle throughput with maximum 5-cycle latency for all the operations except for the MAT, for which 2-cycle throughput and 6-cycle latency is achieved.

II. NUMBER CONVERTERS

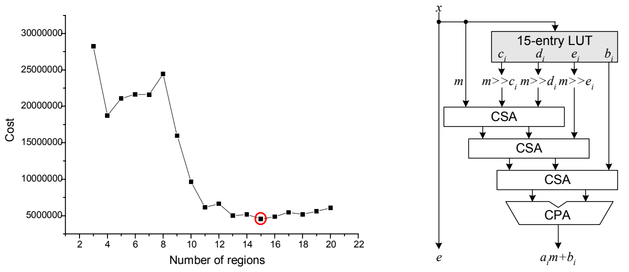
As noted in [2][3], the addition and subtraction require nonlinear function evaluations in the logarithmic domain, while other operations are reduced into simpler operations. Thus, by extending the number system in [2], our unit is designed based on the hybrid of the FLP and the logarithmic number system (LNS), where the addition and subtraction are performed in FLP while others are converted into the LNS.

A. Cost Function

The logarithmic and antilogarithmic converters between the FLP and the LNS are proposed for this hybrid number system (HNS). These use the piecewise linear approximation of the nonlinear terms to reduce hardware complexities.

The optimal number of approximation regions is determined by the tradeoff between the error rate and hardware overheads. Thus, the cost function (1) is defined as a function of power (ω), area (μ), delay (τ), and error rate (ε) for a given number of approximation regions. Here, the power term is squared to emphasis the effects of power overhead.

$$Cost(\omega,\mu,\tau,\varepsilon) = \omega^2 \cdot \mu \cdot \tau \cdot \varepsilon \qquad (1)$$

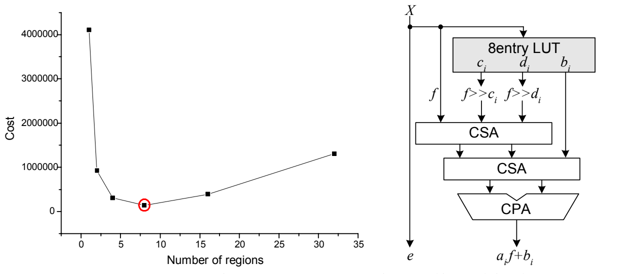


(a) Cost graph (b) Logarithmic converter

Fig. 1. Proposed logarithmic converter

B. Logarithmic Converter

For the 32-bit FLP $x = 2^e(1+m)$, its logarithmic number is represented as $\log_2 x = e + \log_2(1+m)$. The e is the integer part and $\log_2(1+m)$ is the fractional part. The $\log_2(1+m)$ is approximated by piecewise linear expressions as $\log_2(1+m) \approx a_i \cdot m + b_i$, where a_i and b_i are the approximation coefficients defined for each approximation region *i*. Fig. 1(a) shows 15 approximation regions with 0.41% maximum conversion error makes the optimal design regarding (1). Fig. 1(b) shows this logarithmic converter.



(a) Cost graph (b) Antilogarithmic converter

Fig. 2. Proposed antilogarithmic converter

C. Antilogarithmic Converter

For the logarithmic number *X*, its FLP number can be represented by $2^X = 2^{e+f} = 2^e \cdot 2^f$. The integer part *e* directly becomes the exponent of the FLP number and the nonlinear term 2^f is approximated by piecewise linear expressions like $2^f \approx a_i \cdot f + b_i$, where a_i and b_i are the approximation coefficients defined for each approximation region *i*. Fig. 2(a) shows 8 approximation regions with 0.08% maximum conversion error makes the optimal design regarding (1). Fig. 2(b) shows the antilogarithmic converter.

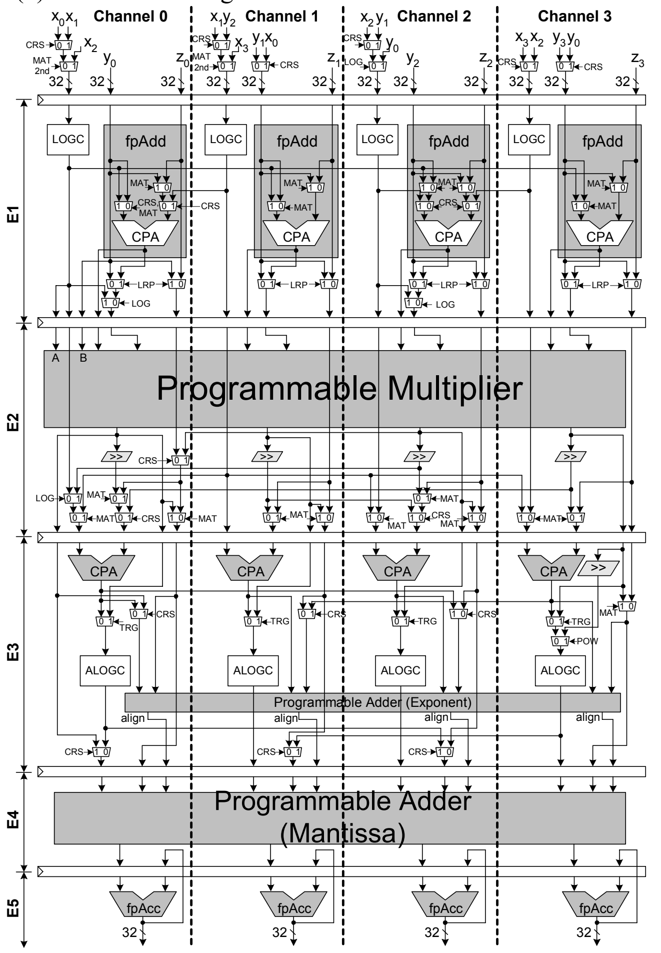


Fig. 3. Floating-point multifunction unit

III. Floating-Point Multifunction Unit

A. Overall Architecture

The overall architecture of the proposed unit is organized with 4-channels and 5 pipeline stages, as shown in Fig. 3. The 4 of 32-bit FLP operands are converted into the logarithmic numbers with 8-bit integer, 24-bit fraction, 1-bit sign, and 1-bit zero through the 4 logarithmic converters (LOGCs) in the E1. The E2 includes the programmable multiplier (PMUL), which can be used for the Booth multiplier, 4 LOGCs or 4 antilogarithmic converters (ALOGCs) according to the target operations. In the E3, 4 adders in logarithmic domain are provided for the vector operations and the resulting values are converted into the FLP numbers through the 4 ALOGCs. The FLP programmable adder (PADD) in the E4 can be programmed into a 5-input FLP adder tree or 4-way 2-input SIMD FLP adders according to target operations. The E5 provides a SIMD FLP accumulator for the final accumulation required for the MAT.

B. Adaptive Number Conversion

In [2], 8 LOGCs and a Booth multiplier (BMUL) were used in the 1st and 2nd stages of the pipeline, respectively, to implement the vector (VEC) and elementary functions (ELMs). But the ELMs used a BMUL with only 1 LOGC, while the VECs used 8 LOGCs without BMUL. Thus, we propose an adaptive number conversion (ANC), which divides the 8 LOGCs into 2 groups and places 4 in E1 and the other 4 in E2, and only converts 4 FLP operands in E1 and the other 4 are converted in E2 only if the operation is VEC. Thus the BMUL in E2 is made into a PMUL that can be programmed into the BMUL or 4 LOGCs, by just adding 15-entry 64B LOG look-up table (LUT) and sharing the CSAs and a CPA. The PMUL is also made programmable into 4 ALOGCs for the MAT by adding 8-entry 56B ALOG LUT. The PMUL is shown in Fig. 4 and it can be programmed into 4 LOGCs for VECs, a single 32b×24b BMUL for POW, 4-way 32b×6b BMUL for trigonometric functions or 4 ALOGCs for MAT.

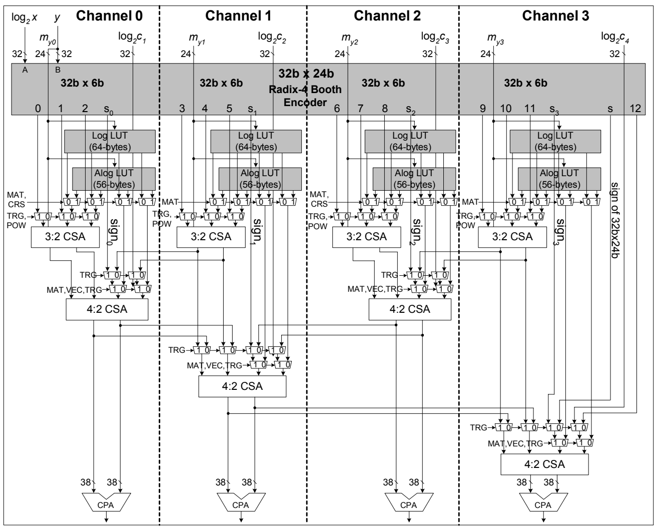


Fig. 4. Programmable multiplier (PMUL)

C. Operation Set

1) Matrix-Vector Multiplication

The geometry transformation in 3D graphics can be computed by the multiplication of 4×4-matrix with 4-element vector, which requires 16 multiplications and 12 additions. This can be converted into the HNS as in Fig. 5 requiring 20 LOGCs, 16 adders, 16 ALOGCs and 12 FLP adders. By extending the ANC, the MAT is implemented with 2-cycle throughput on the 4-way arithmetic unit, where it was implemented with 4-cycle throughput in conventional way [2]. Since the coefficients of a geometry transformation matrix are fixed during processing of a 3D object, these can be pre-converted into the logarithmic domain and be used as constants during the processing. Thus, the MAT only requires 4 LOGCs for vector element conversion, 16 adders in logarithmic domain, 16 ALOGCs and 12 FLP adders. This can be implemented in 2 phases on this 4-way arithmetic unit as illustrated in Fig. 5. In this scheme, 8 adders in logarithmic domain and 8 ALOGCs are required per phase and the 8 ALOGCs in the first phase are obtained from 4 ALOGCs in

E2 by programming the PMUL into 4 ALOGCs together with the 4 ALOGCs in E3. The CPAs in E1 and E3 are used for the 8 adders in logarithmic domain. The 4 multiplication results from the ALOGCs in E2 and the other 4 from the E3 are added in the E4 by programming the PADD into 4-way SIMD adder to get the first phase result. With the same process repeated, the accumulation with the first phase result in E5 completes the MAT.

$$\begin{bmatrix} c_{00} & c_{01} & c_{02} & c_{03} \\ c_{10} & c_{11} & c_{12} & c_{13} \\ c_{20} & c_{21} & c_{22} & c_{23} \\ c_{30} & c_{31} & c_{32} & c_{33} \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} c_{00} \\ c_{10} \\ c_{20} \\ c_{30} \end{bmatrix} x_0 + \begin{bmatrix} c_{01} \\ c_{11} \\ c_{21} \\ c_{31} \end{bmatrix} x_1 + \begin{bmatrix} c_{02} \\ c_{12} \\ c_{22} \\ c_{32} \end{bmatrix} x_2 + \begin{bmatrix} c_{03} \\ c_{13} \\ c_{23} \\ c_{33} \end{bmatrix} x_3 =$$

$$\underbrace{\boxed{\underbrace{\boxed{2^{\left(\begin{bmatrix} \log_2 c_{00} \\ \log_2 c_{10} \\ \log_2 c_{20} \\ \log_2 c_{30} \end{bmatrix} + \log_2 x_0\right)} + 2^{\left(\begin{bmatrix} \log_2 c_{01} \\ \log_2 c_{11} \\ \log_2 c_{21} \\ \log_2 c_{31} \end{bmatrix} + \log_2 x_1\right)}}}_{\textbf{MAT 1}^{\textbf{st}}\textbf{ phase}} + 2^{\left(\begin{bmatrix} \log_2 c_{02} \\ \log_2 c_{12} \\ \log_2 c_{22} \\ \log_2 c_{32} \end{bmatrix} + \log_2 x_2\right)} + 2^{\left(\begin{bmatrix} \log_2 c_{03} \\ \log_2 c_{13} \\ \log_2 c_{23} \\ \log_2 c_{33} \end{bmatrix} + \log_2 x_3\right)}}}_{\textbf{MAT 2}^{\textbf{nd}}\textbf{ phase}}$$

Fig. 5. Implementation of MAT

2) Vector-SIMD Operations

The vector-SIMD multiplication, division, square root, and multiply-add (MAD) are implemented based on [2]. The vector-SIMD linear interpolation (lerp) can be represented as (2).

$$\left(x_i\left(z_i - y_i\right) + y_i\right)_{i\in\{0,1,2,3\}} = \left(2^{\log_2 x_i + \log_2\left(z_i - y_i\right)} + y_i\right)_{i\in\{0,1,2,3\}} \qquad (2)$$

Since a LOGC is moved into E2 for the ANC, the lerp can be implemented with an FLP adder augmented in E1. This adder also reduces the latency of vector-SIMD addition and subtraction from 4 to 1 compared with [2].

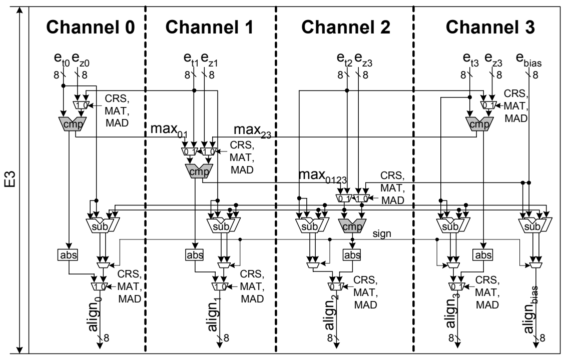
The vector-SIMD operations like (2) require 2 LOGCs for 2 operands per channel. Thus, for 4 channels, the PMUL is programmed into 4 LOGCs to make the 8 LOGCs together with the 4 LOGCs in E1.

3) Vector Products

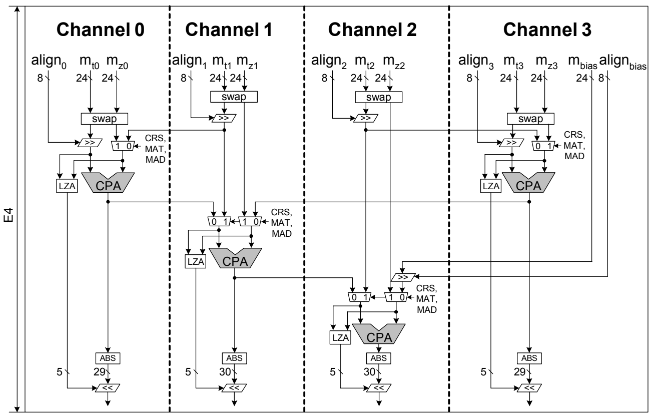
The cross product (CRS), which requires 6 multiplications, is implemented as an operation with single-cycle throughput on the 4-way arithmetic unit by extending the ANC. The CRS in HNS, represented as (3), requires 12 LOGCs, 6 adders, and 6 ALOGCs. However, it is implemented with 6 LOGCs, 6 adders in logarithmic domain, and 6 ALOGCs since only 6 different operands are used in the CRS. The required 6 LOGCs and 6 ALOGCs are obtained from the 4 LOGCs in E1 and 4 ALOGCs in E3 together with the PMUL with 2 channels programmed into 2 LOGCs and the other 2 channels into 2 ALOGCs. The CPAs in E1 and E3 are used for the 6 adders in logarithmic domain for the 6 multiplications. The final 3 FLP subtractions between the product terms can be done by programming the PADD into a 4-way SIMD adder.

$$\begin{bmatrix} x_1 y_2 - y_1 x_2 \\ x_2 y_0 - y_2 x_0 \\ x_0 y_1 - y_0 x_1 \end{bmatrix} = \begin{bmatrix} 2^{\log_2 x_1 + \log_2 y_2} - 2^{\log_2 y_1 + \log_2 x_2} \\ 2^{\log_2 x_2 + \log_2 y_0} - 2^{\log_2 y_2 + \log_2 x_0} \\ 2^{\log_2 x_0 + \log_2 y_1} - 2^{\log_2 y_0 + \log_2 x_1} \end{bmatrix} \qquad (3)$$

The dot product (DOT) is implemented with the vector element multiplication using the vector-SIMD multiplication and the final summation of product terms by programming the PADD into a single adder tree. Thus, the PADD in E3 through E4 is programmed into a 4-way SIMD adder for CRS, MAT, and MAD or a single 5-input summation tree for DOT and TRGs. This PADD is shown in Fig. 6.



(a) Exponent path



(b) Mantissa path

Fig. 6. Programmable adder (PADD)

If the mantissa path shown in Fig. 6(b) is programmed as a summation tree, the maximum exponent is determined for the mantissa alignments and all the mantissa are aligned regarding the maximum exponent to avoid repeated normalization and denormalization logics in every level of the summation tree. Since the exponent can be determined directly from the value in logarithmic domain, the exponent logic for the summation tree can be processed in parallel with the ALOGCs in the E3. In order to reduce the critical path of the exponent logic for summation tree shown in Fig. 6(a), the duplicate subtractors are used in parallel with the final comparator to compute the alignment values for both cases of the comparison results. The actual alignment values are determined when the comparison result becomes available.

4) Elementary Functions

The POW and TRGs are implemented based on [2]. In addition, the logarithm (LOG) with 2 variables ($\log_x y$) is supported rather than the logarithm with some constant bases supported in [2]. This LOG can be represented by (4).

$$\log_x y = \log_2 y / \log_2 x = 2^{\log_2\left(\log_2 y\right) - \log_2\left(\log_2 x\right)} \qquad (4)$$

The (4) requires a pair of cascaded LOGCs and these can be implemented by programming 2 channels of the PMUL into 2 LOGCs and cascading these with the 2 LOGCs in E1.

The POW can be represented as $x^y = 2^{y \times \log_2 x}$ as proposed in [2]. Since the power is converted into the multiplication in logarithmic domain, it requires a 32b×24b multiplier and the multiplier is implemented by programming the PMUL into a single 32b×24b BMUL. The TRGs are also unified with others using Taylor series as proposed in [2]. This power series requires a 4-way 32b×6b multiplier in logarithmic domain and final summation of these terms. This can be implemented by programming the PMUL into 4-way 32b×6b BMUL and the PADD into a 5-input summation tree.

IV. Implementation Results

The test 3D graphics scenes with QVGA screen are shown in Fig. 7. The transformation and lighting routine is used to test the effects of various operations including the MAT. As shown in this figure, the computation error from the LOGC and ALOGC is tolerable for this small screen.

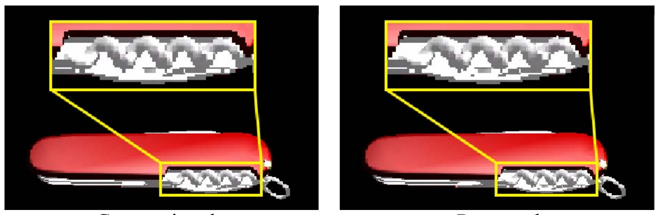
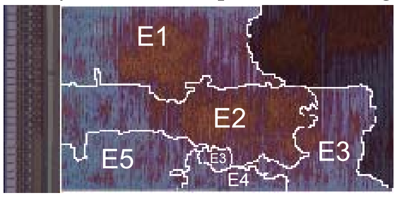


Fig. 7. Comparison of 3D graphics scenes

The proposed unit is synthesized and integrated into a vertex shader and fabricated in 0.18μm 6-metal CMOS technology [4]. The pipeline registers are fully clock-gated to disable unnecessary switching under the control of each operation. Fig. 8 shows the chip micrograph and the unit takes 451K transistors. The area distribution for the pipeline stages is shown in Fig. 9. It shows even area distribution for each stage. The E1, which took about 50% of the area in [2], takes only 25% of total area. In this design, the ANC reduces total area by 15% without performance degradation.



Process technology	1-poly 6-metal TSMC 0.18um CMOS
Transistor counts	451K
Power supply	1.8V
Operating frequency	200MHz
Power consumption	Max. 28.5mW

Fig. 8. Chip micrograph

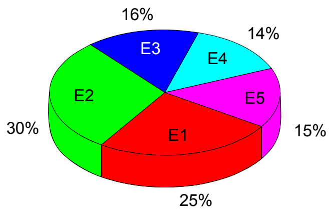


Fig. 9. Area distribution

Fig.10 shows the shmoo plot. The pipeline operates at 200MHz and its peak performance is 2.8 GFLOPS with maximum power consumption of 28.5mW for the MAT, which consists of 28 FLP operations. Fig. 11 shows the measured waveform of MAT and each input is sustained for 2 cycles due to the interlock control from the MAT instruction. Table I summarizes the error, power, latency, and throughput of selected operations. The test chip is compared with others and it shows 10.18mW/GFLOPS, the best performance among compared works. The comparison data are listed in Table II.

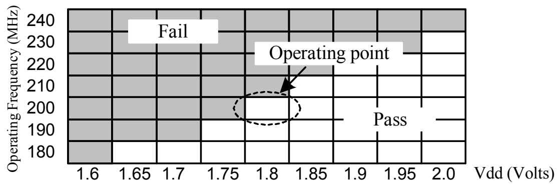


Fig. 10. Shmoo

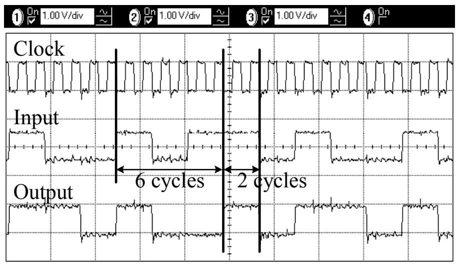


Fig. 11. Measured waveform

TABLE I
THE ERROR/POWER/LATENCY OF SELECTED OPERATIONS

OPERATIONS	MAX ERROR (%)	POWER (mW)	Latency / Throughput
Vector SIMD multiply-add	0.21	24.5	5 / 1
Vector SIMD lerp	0.4975	27.1	5 / 1
Dot product	2.434	25.8	5 / 1
Cross product	9.58	26.3	5 / 1
Matrix-vector multiply	0.0825	28.5	6 / 2
Logarithm	0.4	7.66	3 / 1
Power	2.096	9.75	3 / 1
Sine	0.1175	18.7	5 / 1

TABLE II
THE COMPARISON RESULTS

Ref	Performance (GFLOPS)	Power (mW)	Area (Ktransistors)	mW / GFLOPS	Freq. (MHz)	Process (μm)
[5]	2.8	250	N.A.	89.28	400	0.13
[6]	44.8	1400	768	43.75	5600	0.09
Proposed	2.8	28.5	451	10.18	200	0.18

V. Conclusion

A high-performance, power- and area-efficient 4-way 32-bit floating-point multifunction unit is proposed. Using logarithmic arithmetic, it unifies the matrix, vector, and elementary functions into a single arithmetic unit. The optimal designs of the 15-region LOGC and 8-region ALOGC are proposed based on the cost analysis. The ANC is proposed and reduces total area by 15%. Using the ANC, the MAT, CRS, LRP, and LOG are newly unified with the other operations. The proposed unit achieves single cycle throughput for all the operations and 2-cycle throughput for the MAT. It is fabricated with 451K transistors and shows 2.8GFLOPS at 200MHz consuming 28.5mW at 1.8V.

References

[1] Khronos Group, *OpenGL ES 2.0*, http://www.khronos.org
[2] B.-G. Nam, et al., “A 210MHz 15mW Unified Vector and Transcendental Function Unit for Programmable Handheld 3-D Graphics Systems,” in *Proc. IEEE A-SSCC*, Nov. 2006.
[3] F. Lai, and C.E. Wu, “A Hybrid Number System Processor with Geometric and Complex Arithmetic Capabilities,” *IEEE Trans. on Computers*, Vol. 40, No. 8, August 1991.
[4] B.-G. Nam, et al., “A 52.4mW 3D Graphics Processor with 141Mvertices/s Vertex Shader and 3 Power Domains of Dynamic Voltage and Frequency Scaling,” in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2007.
[5] F. Arakawa, et al., “An Embedded Processor Core for Consumer Appliances with 2.8GFLOPS and 36Mpolygons/s,” in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2004.
[6] H.-J. Oh, et al., “A Fully Pipelined Single-Precision Floating-Point Unit in the Synergistic Processor Element of a CELL Processor,” in *IEEE JSSC*, vol. 41, no. 4, pp.759-771, Apr. 2004.