Journal of Korea Multimedia Society Vol. 11, No. 6, JUNE 2008(pp. 816-827)

Design of Pipelined Floating-Point Arithmetic Unit for Mobile 3D Graphics Applications

Byeong-Yoon Choi[†], Chang-Soo Ha[††], Jong-Hyoung Lee[†††], Zoran Salcic[††††], Duck-Myung Lee[†††††]

ABSTRACT

In this paper, two-stage pipelined floating-point arithmetic unit (FP-AU) is designed. The FP-AU processor supports seventeen operations to apply 3D graphics processor and has area-efficient and low-latency architecture that makes use of modified dual-path computation scheme, new normalization circuit, and modified compound adder based on flagged prefix adder. The FP-AU has about 4-ns delay time at logic synthesis condition using 0.18㎛ CMOS standard cell library and consists of about 5,930 gates. Because it has 250 MFLOPS execution rate and supports saturated arithmetic including a number of graphics-oriented operations, it is applicable to mobile 3D graphics accelerator efficiently.

Key words: floating-point arithmetic, 3D graphics, mobile graphics, OpenGL-ES, JSR 184

1. INTRODUCTION

As the demand of mobile electronics market is growing, mobile devices, such as cellular phone, pocket sized PC or personal digital assistants (PDA) and navigator are becoming more popular. These mobile devices need 3-dimensional (3D) graphics accelerator to support 3D games and multimedia applications. The architectural paradigm of 3D graphics accelerator used in the mobile applications such as OpenGL ES and JSR184 is changing from fixed pipelined processor to programmable shader processor to support flexible and optimized rendering[1]. The shader processor adopts SIMD (single instruction multiple data stream) or multi-threaded architecture. The core components of the shader processor are floating-point arithmetic unit (FP-AU), floating-point multiplication unit (FMUL) and special function unit (SFU). To meet the requirements of small area and low power consumption simultaneously, the graphics-oriented architecture must be applied to the mobile device. While the conventional high-performance floating-point unit used in conventional microprocessor must meet IEEE 754 floating point standard, the FP-AU embedded in the graphics accelerator must satisfy the real time requirement and meet the IEEE 754 standard optionally.

In this paper, we designed area efficient and low latency floating point arithmetic unit optimized for mobile graphics applications. The paper is organized as follows. In section 2, 3D graphics accelerator based on programmable shader approach is shown. In section 3, hardware design of pipelined floating point arithmetic unit to support seventeen

※ Corresponding Author : Byeong-Yoon Choi, Address : (614-714) Department of Computer Engineering, Dongeui University, Busan, Korea, TEL : +82-51-890-1706, FAX : +82-51-890-2629, E-mail : bychoi@deu.ac.kr
Receipt date : Oct. 31, 2007, Approval date : May 15, 2008
[†] Department of Computer Engineering, Dongeui University
[††] Department of Computer Engineering, Dongeui University
(E-mail : systemonchip@daum.net)
[†††] Department of Electronics Engineering, Dongeui University
(E-mail : jonghlee@deu.ac.kr)
[††††] Department of ECE, University of Auckland, NZ
(E-mail : z.salcic@auckland.ac.nz)
[†††††] Nexuschips Co. Seoul
(E-mail : dmlee@nexuschips.com)
※ This work was supported by Ministry of Information and Communication(MIC), Korea and IDEC software.

operations is described. In section 4, performance evaluation results for the designed FP-AU are discussed, and conclusions follow in section 5.

2. GRAPHICS ACCELERATOR IN MOBILE APPLICATION

The purpose of 3D graphics pipeline is to take the description of a scene in three-dimensional space and to map it into a two-dimensional projection on the view surface [2]. The pipeline consists of geometry and rendering stage. The simplified view for the conventional 3D graphics pipeline is shown in figure 1.

Recently, to provide 3D graphics accelerator with the programmability, 3D graphics processors based on shader architecture shown in figure 2 are widely adopted from desktop PCs to mobile devices. The processor consists of vertex shader, rasterizer and fragment shader [3].

We designed the vertex and fragment shaders which can be embedded in the 3D mobile graphics accelerator. The designed vertex shader (VS) is shown in figure 3. Instructions of vertex shader processor are executed via six-stage pipeline as shown in figure 4. The instruction pipeline consists of IF(instruction fetch), AG(address generation and pre-instruction decode), OF(operand fetch and instruction decode), EX1(execute 1), EX2(execute 2), and WB(write-back) stages. Because the VS processor doesn't have data memory access instructions such as load and store instructions, it does not have MEM(memory access) pipeline stages, which is differently from conventional

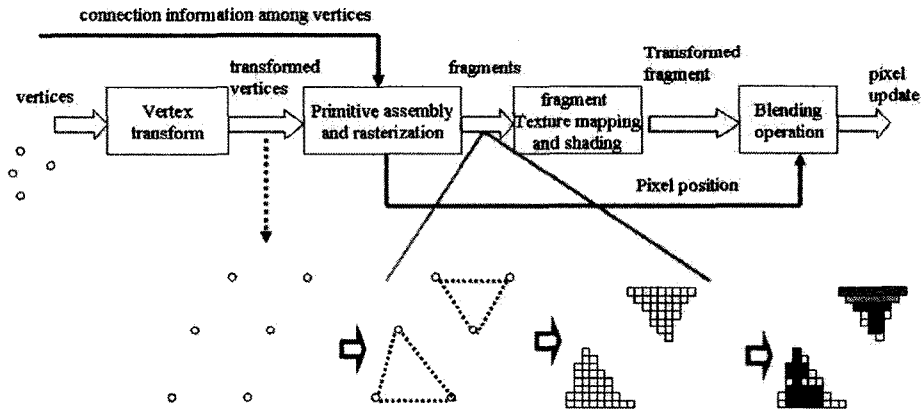


Fig. 1. Simplified view for 3D graphics pipeline

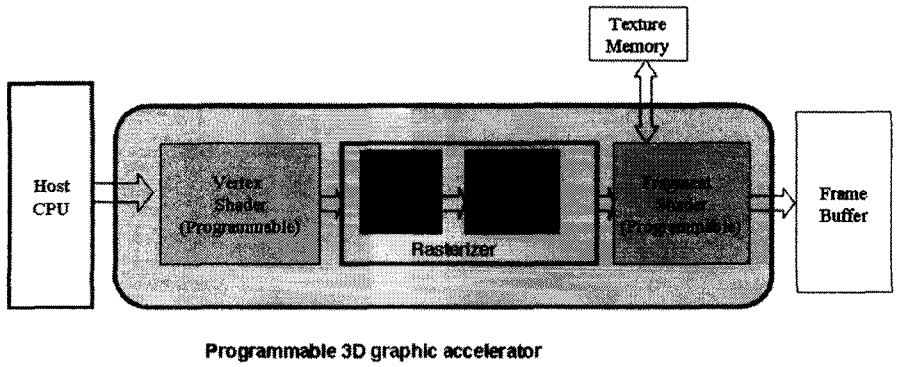


Fig. 2. Programmable graphics processor based on shader architecture

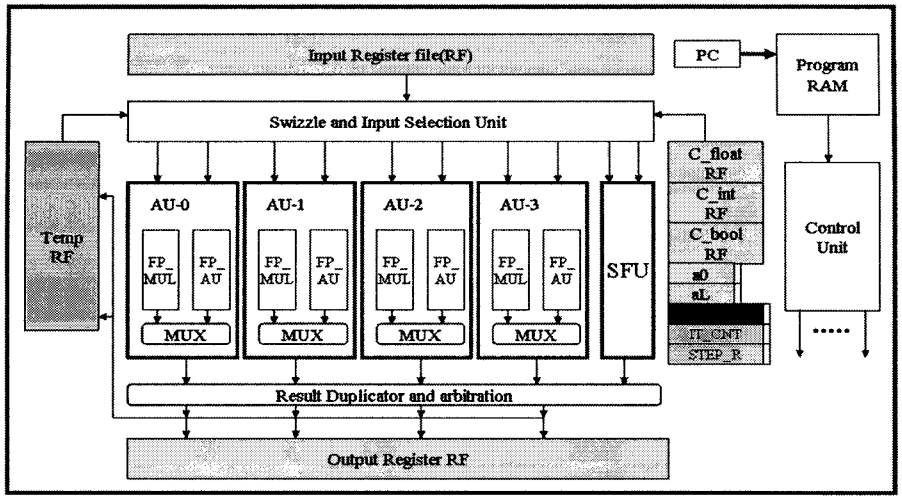


Fig. 3. Block diagram of vertex shader processor

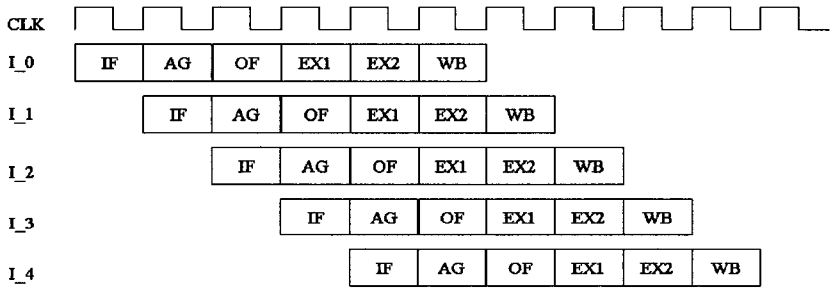


Fig. 4. Instruction pipeline adopted in the vertex shader processor

RISC processor pipeline. From timing analyses for the instruction pipeline, we observe that two pipeline stages (EX1, EX2) can be assigned to FP-AU operation. Although more pipeline stages can make FP-AU to operate at a higher clock rate, datapath must handle complex data dependency and also has long latency problems. Therefore, the pipeline depth of FP-AU is limited to two.

3. Hardware Design of Pipelined Floating-Point Arithmetic Unit

The seventeen graphics-oriented operations are chosen for FP-AU from the detailed analyses for OpenGL-ES and DirectX shader specifications. The FP-AU consists of sign unit, exponent unit, fraction unit and data-stationary pipelined controller. The operations supported in the FP-AU are summarized in table 1. The floating point and the integer data format adopted in the FP-AU are IEEE single precision format (32-bit) and 24-bit integer format, respectively. In the case of integer format, 32-bit word consists of lower 24-bit valid data and upper 8-bit sign extended data. The default rounding mode is round-to-nearest even (RNE) mode. But FLOOR and ITOF operations use RTZ(round to zero) mode rather than RNE mode. Although conventional ITOF operation needs RNE rounding mode, the ITOF instruction of this FP-AU can adopt RTZ mode because of 24-bit integer format. Unlike the IEEE 754 floating point standard, FP-AU supports saturated arithmetic which converts the final result to maximum or minimum values of destination register in case of

Table 1. Operations supported in FP-AU

Mnemonics	descriptions	operation
NOP	No operation	
ABS Rd, Rs	Absolute	Rd ← abs(Rs)
NEG Rd, Rs	Negate	Rd ← -Rs
MOV Rd, Rs	Move	Rd ← Rs
ADD Rd, Rs, Rt	Add	Rd ← Rs + Rt
SUB Rd, Rs, Rt	Subtract	Rd ← Rs - Rt
MAX Rd, Rs, Rt	Maximum	Rd ← max{Rs, Rt}
MIN Rd, Rs, Rt	Minimum	Rd ← min{Rs, Rt}
ITOF Rd, Rs	Integer to Float	Rd ← float(Rs)
FLOOR Rd, Rs	floor	Rd ← ⌊Rs⌋
FTOI Rd, Rs	Float to integer	Rd ← Integer(Rs)
SEQ Rd, Rs, Rt	Set if equal	Rd←1, if Rs == Rt Rd ←0, else
SGE Rd, Rs, Rt	Set if greater or equal	Rd ← 1, if Rs ¤ Rt Rd ← 0, else
SLT Rd, Rs, Rt	Set if less than	Rd ← 1, if Rs < Rt Rd ← 0, else
SGN Rd, Rs	sign	Rd ← -1, if Rs < 0 Rd ← 0, if Rs ==0 Rd ← 1, if Rs > 0
CLAMP Rd, Rs	Clamp to MAXpower	If (Rs < - MAXpower) Rd ← -MAXPower else if (Rs > MAXpower) Rd ← MAXPower else Rd ← Rs

CMP Rd, Rs, Rt	Compare and select one operand using LT flag of the previous SLT operation	If (LT_flag ==1)
		Rd ← Rt
		else
		Rd ← Rs

overflow or underflow. To realize area efficient and short latency architecture, three techniques, namely, dual-path computation scheme, new normalization scheme and compound adder based on modified flagged prefix adder are applied to FP-AU. And other operations except ADD and SUB operations are implemented using the dual-path architecture of floating-point addition and some dedicated hardware.

3.1 Modified dual-path computation scheme

The floating-point addition and subtraction operations can be divided into dual paths, that is, near path and far path according to effective operation and exponent difference. The far path of conventional dual-path scheme [4-5] is adopted in case of addition and subtraction with exponent difference larger than one. On the contrary the near path is adopted in subtraction of which exponent difference is zero or one. The conventional dual-path approach requires complex rounding operation in both paths. To eliminate rounding step in the near path, the modified dual-path approach suggested in the SUN SPARC RISC [6] is adopted. By considering the MSB of the mantissa embedded in the larger operand, it eliminates rounding step in the near path. In the new scheme, the subtraction with exponent difference of 1 and mantissa of larger

than or equal to 1.5 is classified into far path, which is different from the conventional scheme. The modified dual-path scheme is shown in figure 5. The near path and the far path are executed through two pipelined stages and each stage consists of adder or long length shifter to balance the delay between two pipeline stages. The NOP, ABS, NEG, MOV, ITOF, SEQ, SGE, SGN, and SLT operations are embedded into the near path structure. And FLOOR and FTOI operations were embedded into the far path structure. MAX and MIN operations use both the near and far paths. CMP and CLAMP are implemented by use of the near path and some dedicated hardware.

In the far path shown in figure 5, add/sub operation, rounding and normalization steps are reduced into one concurrent step using novel technique based on compound adder and efficient rounding scheme. Because the normalization step requires 0-bit, 1-bit right/left operations, possible output formats can be classified into a few cases. Considering rounding, normalization and post normalization, the final result selection operation can be implemented by using a simple shifter such as a multi-input multiplexer. The concurrent add/round/normalization step needs an efficient compound adder which generates both "A+B" and "A+B+1" outputs, gin generator and round logic. The valid results are selected by control signal generated in the rounding and result selection control logic. The selection control signals are dependent on fraction overflow and underflow conditions, two LSB bits of compound adder (bit [LSB], bit [LSB+1]) and rounding control bits such as guard (g), round (r) and sticky bit (s). Figure 6 represents the block diagram of the far path fractional part.

The gin generator generates gin bit which is the least significant bit(LSB) of the final result after rounding operation in case of fraction underflow. The compound adder generates two sum signals X+Y, X+Y+1 and some flag signals. C^1_{out} and C^0_{out}

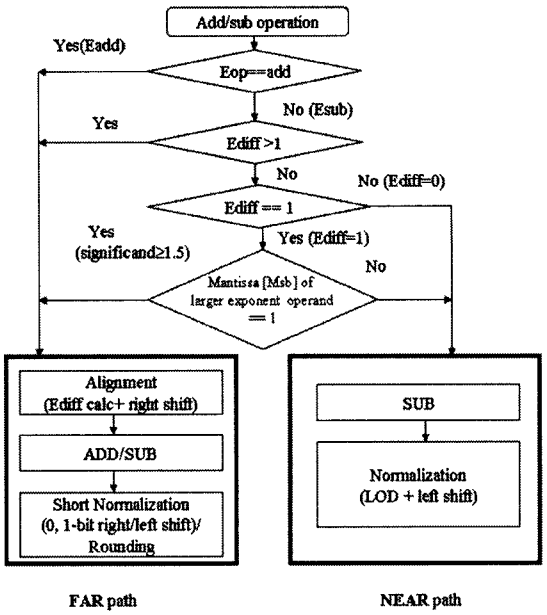


Fig. 5. Modified two-path arithmetic scheme without rounding operation in the near path

signals represent carry-out signals of X+Y, and X+Y+1, respectively and are used to detect fraction overflow generated before and after rounding operation for addition result. $S1[msb]$ and $S0[msb]$ signals means MSB signals of X+Y, and X+Y+1, respectively and are used to detect fraction underflow generated before rounding for subtraction result. The $bit[LSB]$ and $bit[LSB+1]$ signals represent bit[0] and bit[1] output signals of X+Y, respectively and are used as control signal of RNE rounding operation. The sticky bit used in the rounding operation can be described by the equation (1) using the number of trailing zeros in the smaller operand (LEN) and exponent difference (d). The value of sticky bit is one, if exponent difference is greater than or equal to the number of trailing zeros in the smaller operand plus constant "three".

$$sticky\_bit = d \geq (LEN + 3) \quad (1)$$

Figure 7 represents the sticky bit generator. The three-operand adder is implemented using one CSA (carry save adder) and one CPA (carry propagate adder) instead of two CPAs to reduce delay.

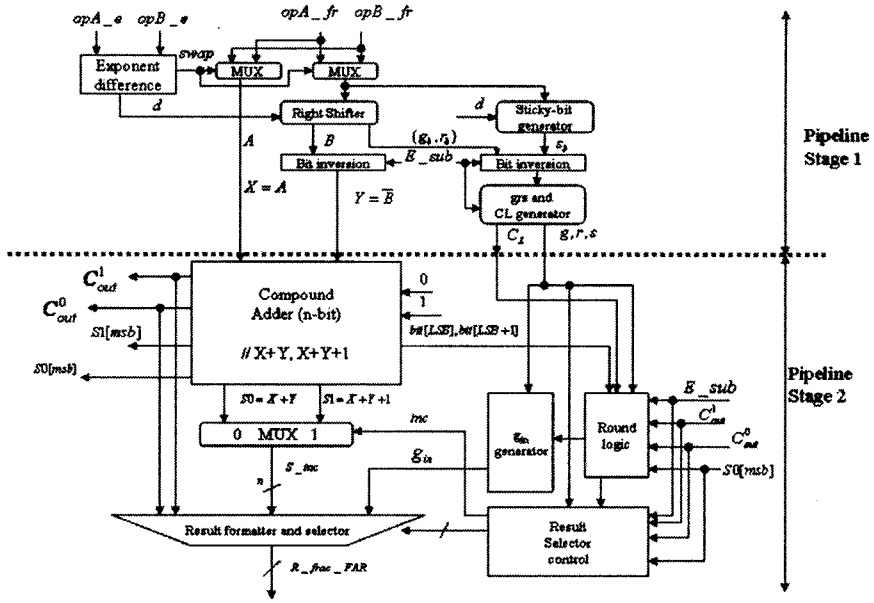


Fig. 6. The block diagram of far path fractional part

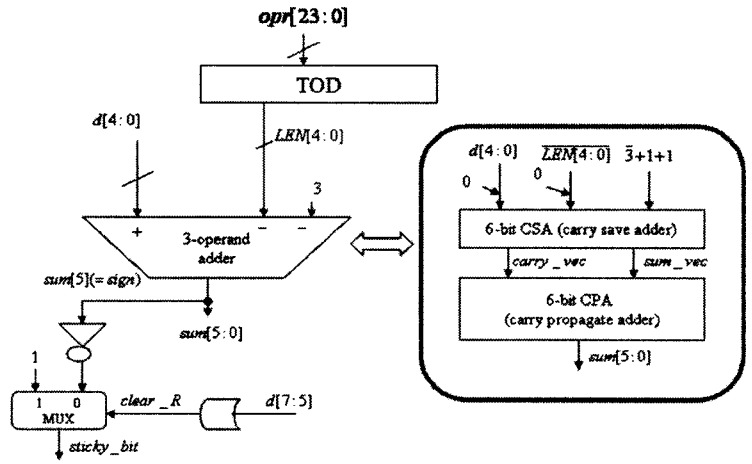


Fig. 7. Sticky bit generator

The near path consists of exponent estimator, compound adder, bit converter, LOD (leading one detector)[7], and normalization circuit. Because the result of subtraction can be negative in case exponent difference is zero, negative result must be converted into sign-magnitude if $\overline{C_{out}^{f}}$ is detected. The conventional conversion operation requires addition to implement two's complement operation. Because the near path of FP-AU has compound adder, the conversion operation can be efficiently implemented by equation (2) using one's complement operation for result $X+\overline{Y}$ instead of addition.

$$-(X-Y)=-(X-Y-1)-1=Y-X=\overline{X+\overline{Y}} \quad (2)$$

The near path fractional part is shown in figure 8. The near path fractional path doesn't' have rounding hardware because we adopt the modified dual-path scheme and uses LOD (leading one de-

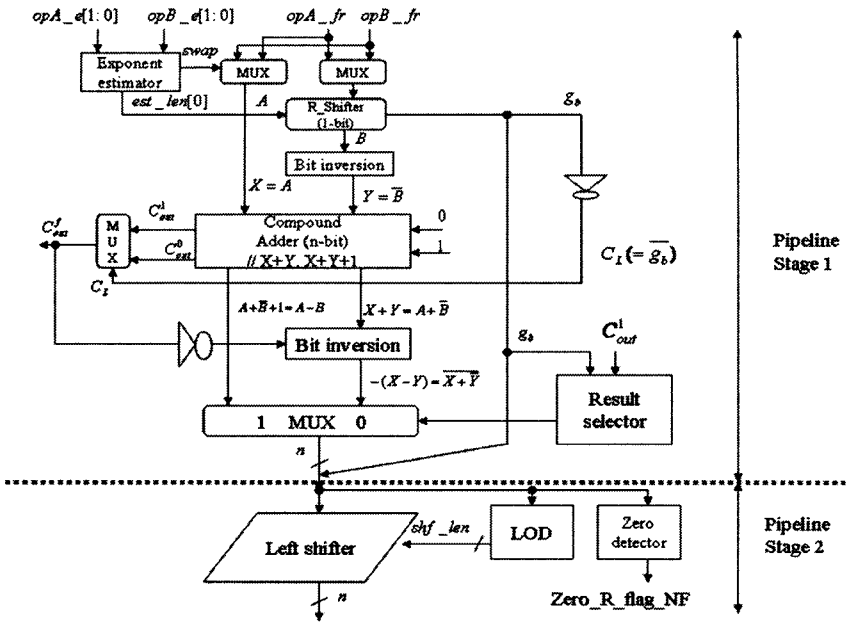


Fig. 8. The block diagram of near path fractional part

tector) to normalize the subtraction result.

3.2 compound adder using modified flagged prefix adder

The conventional compound adder consists of two adders or modified carry select adders which generate two outputs, such as X+Y, X+Y+1. The flagged prefix adder [8] is a modified parallel prefix adder with modified prefix tree and output logic. By controlling output logic using "inc" control signal, "X+Y" or "X+Y+1" can be easily generated. Therefore, the flagged prefix adder can be efficiently used as compound adder. But the compound adder used in the far path fractional part of figure 6 must generate the additional flag signals, such as C^1_{out}, C^0_{out}, $S1[msb]$, $S0[msb]$, $bit[LSB]$ and $bit[LSB+1]$. Because the original flagged prefix adder doesn't generate the above signals, it must be modified. Because all output nodes of the flagged prefix tree have group carry generate GG^0_i and group carry propagate signals GP^0_i in addition to bitwise carry propagate p_i and carry generate g_i signals unlike the original prefix adder, the above flag signals can be easily generated using combination of GG^0_i, GP^0_i, p_i and g_i. The figure 9 represents the new compound adder based on the modified flagged prefix adder. The flagged prefix adder is widely used in various parts of FP-AU, such as compound adder of near path and exponent difference unit, because the flagged prefix adder can generate X-Y, Y-X in addition to X+Y, X+Y+1 by simple modification of output cell and additional control signal.

3.3 new normalization circuit of the near path

The conventional normalization schemes in the near path are classified into two types namely, LOP(leading one predictor)-based scheme[9-10] and LOD(leading one detector) based scheme. Although the LOP scheme shown in figure 10(b) has desirable characteristics that shift control signals can be generated using input operands of sub unit concurrently subtraction operation, the shift control signals may have 1-bit error. Therefore, after coarse shift operation using shift control signals, additional fine shift operation is needed to compensate 1-bit error. On the contrary, LOD scheme

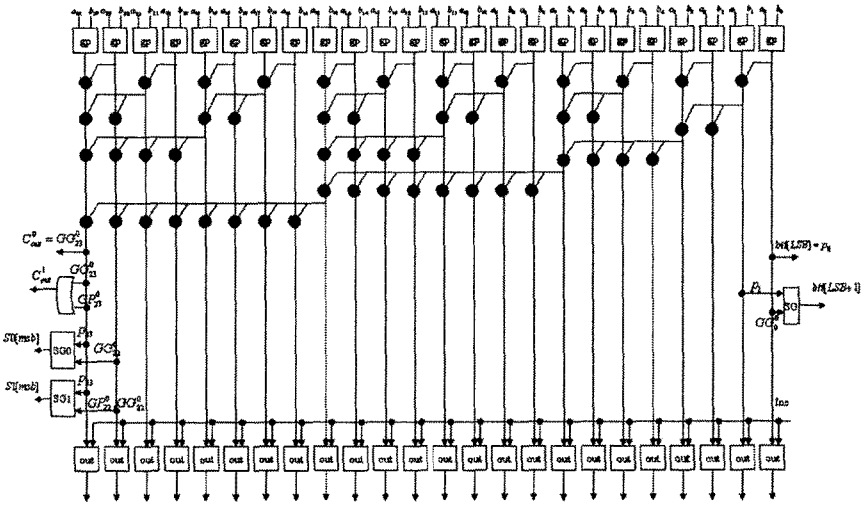


Fig. 9. Compound adder using the modified flagged prefix adder

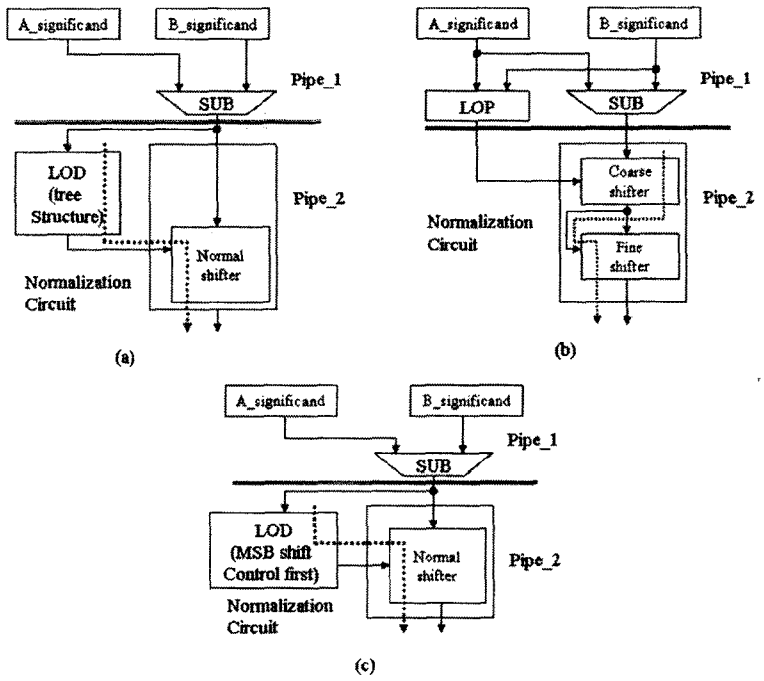


Fig. 10. Normalization scheme of near path fractional part : (a) Tree-based LOD scheme, (b) LOD scheme (c) MSB-first scheme of near path fractional part

shown in figure 10(a) generates shift control signals using the result of subtraction operation and control signals for normalization, which makes normalization shifter can be implemented without a fine shifter. While the LOP scheme has better delay characteristics than LOD scheme because LOP scheme can work concurrently with subtraction operation, the LOD scheme has smaller area. Because the conventional LOD circuit has tree-structure and generates all shift outputs si-

multaneously, it implements LOD operation and normalization sequentially. Recently, the new LOD circuit shown in figure 11 which generates shift control signals from MSB(most significant bit) to LSB(least significant bit) sequentially was proposed by Antelo[11]. By using normalization scheme shown in figure 10(c) that combines MSB-first LOD circuit with LOD scheme of figure 10(a), normalization operation can begin immediately after multi-input NOR gate delay of MSB-first LOD. Thus new scheme has delay characteristics comparable to LOP scheme of figure 10(b) with area merit of conventional LOD scheme.

3.4 Implementation scheme of data format conversion instructions

According to table 1, FP-AU has three format conversion instructions, ITOF, FLOOR and FTOI which convert data format between 24-bit integer data and 32-bit floating point data. The ITOF(integer-to-float) instruction is implemented using the following steps.

[step 1] "0 - B" or "0+B" operations are selectively executed according to sign bit of integer input data B. If integer data is negative, "0-B" operation is executed. Because the length of integer is 24-bit, rounding operation is not required.

[step 2] normalization length, norm_length is calculated using LOD circuit.

[step 3] final results are generated as follows..

R[frac] = normalized_result of "0-B" or "0+B"

R[exp] = (bias+23)-norm_length

R[sign] = sign bit of integer

The ITOF instruction is implemented using the modified near path including addition operation. The FTOI(float-to-integer) instruction includes the following steps.

[step 1] the right shift length, shift_len for floating-point data B is calculated as follows.

Shift_len = (bias + 23) - exp(B)

[step 2] if shift_len is greater than or equal to zero, right shift operation for B is executed and then rounding control bit, g, r, sticky bit are derived. If shift_len is negative, overflow condition is generated.

[step 3] According to sign bit of floating-point data, the different operations are executed.

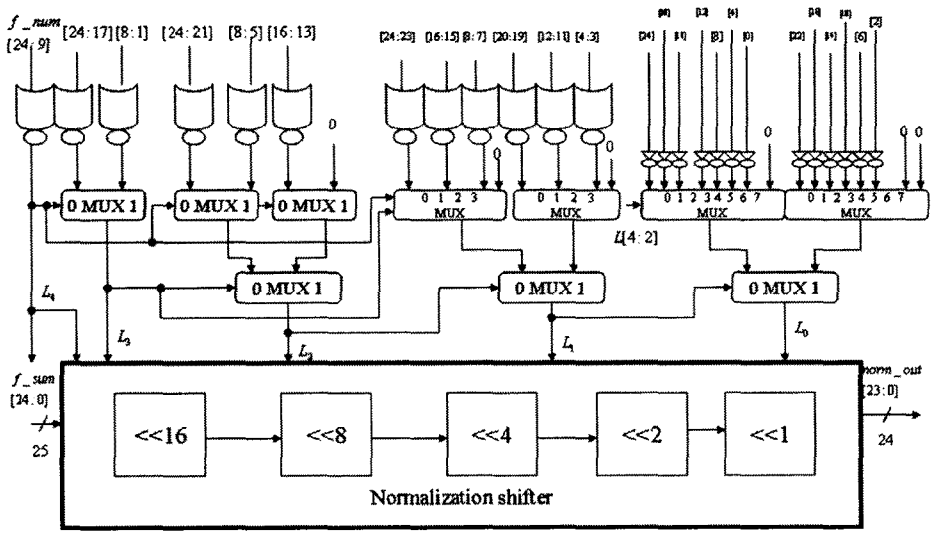


Fig. 11. Normalization circuit using new LOD circuit that generates MSB-first shift control signal

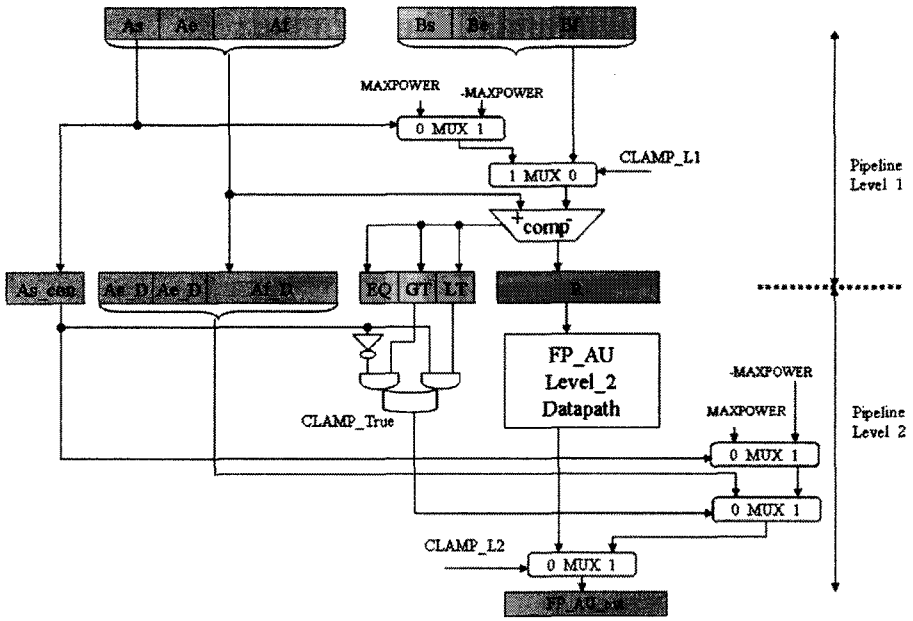


Fig. 12. Operational flow of CLAMP operation

If sign bit is zero, "0+ righ_shift (B)" operation is executed concurrently with the rounding operation. If sign bit one, "0-right_shifter (B)" is executed concurrently with the rounding operation.

The FTOI operation is implemented using the modified far path. The floor operation is similar to FTOI instruction, except the rounding operation. The floor operation must be implemented with round-to-zero(RTZ) rounding mode rather than round-to-nearest even (RNE) rounding mode.

3.5 Implementation scheme of CLAMP instruction

The CLAMP instruction is adopted to implement LIT (light coefficient) operation of DirectX and OpenGL specification. CLAMP operation is implemented using the near path and dedicated hardware. While the near path is used as comparator that generates flag signals, such as EQ(equal), GT(greater than) and LT(less than) flags, the dedicated hardware is used to generate and select hardwired constant values, ±MAXpower. The operational flow of CLAMP operations is shown in figure 12.

4. PERFORMANCE EVALUATIONS

To verify the floating point arithmetic unit, the functional model to generate test vectors for FP-AU is developed using C language before actual hardware design process using Verilog HDL. The figure 13 represents Modelsim simulation waveform for FLOOR instruction. Through comparison between two results from C-model and HDL model, the correct operation of FP-AU circuit is verified. The electrical characteristics of FP-AU are shown in table 2. The logic synthesis is done using Synopsys Design Compiler and 0.18μm CMOS standard cell library. The FP-AU consists of about 5,930 gates and has 250 MHz operating frequency. Because it has 250 MFLOPS execution rate and supports saturated arithmetic and a number of graphics-oriented operations, it can be applicable to mobile 3D graphics accelerator efficiently. Also it can be used as hardware IP(intellectual property) of DSP and 3D graphics processor.

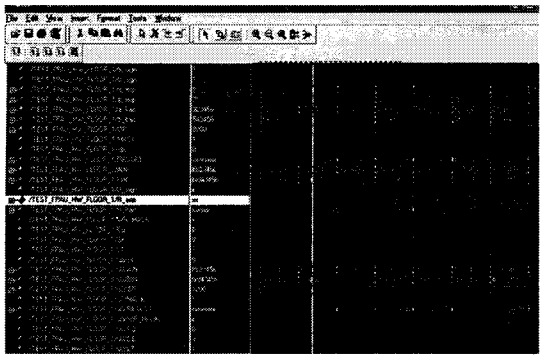


Fig. 13 Modelsim waveform for FLOOR operation

Table 2. Electrical characteristics

Computational scheme	Modified dual path scheme (near and far path)
Rounding mode	Round-to-nearest even / Round-to-zero (Floor operation)
Number of pipeline stage	2
Supported data format	IEEE SP floating point data and 24-bit integer
Maximum operating frequency	250MHz @0.18μm CMOS technology
Number of gates	5,930
Number of operations	17

5. CONCLUSIONS

In this paper, we describe the designed floating-point unit for mobile 3D graphics applications. The FP-AU can execute seventeen operations for DirectX3D and OpenGL-ES 3D graphics API. It also has two-stage pipelined architecture and adopts novel arithmetic schemes, such as saturated arithmetic, compound adder based on modified flagged prefix adder, dual-path arithmetic structure with no rounding near path and new normalization circuit using MSB-first control signal LOD circuit. The FP-AU consists of about 5,930 gates at logic synthesis condition using 0.18μm CMOS standard cell library and has maximum clock frequency of about 250 Mhz. Because it has 250 MFLOPS execution rate and supports saturated arithmetic and a number of graphics-oriented operations, it can be applicable to mobile 3D graphics accelerator.

REFERENCES

[1] D. Astle and D. Durnil, *OpenGL ES Game Development*, Thomson Course Technology, 2004.

[2] Ju-ho Sohn, *Design and Optimization of Geometry Acceleration for Portable 3D graphics*, KAIST M.S. Thesis, Dec. 2002.

[3] K. Gray, *Microsoft DirectX9 Programmable Pipeline*, Microsoft Press 2004.

[4] J.B. Brugure and T.L. Lang, *Rounding in Floating Point Addition using a compound adder*, Internal Report, Dept. of Electronics and Computer Engineering, University of Santiago de Compostela, Spain, July, 2000.

[5] N.T. Quach and M.J. Flynn, *An Improved Algorithm for High-Speed Floating Point Addition*, Technical Report CSL-TR-9-442, Stanford University 1990.

[6] A. Nami, and A. Dhablania, "1-GHz HAL SPARC64 dual floating-point unit with RAS features," in Proc., *15th IEEE Symposium on Computer Arithmetic*, pp. 173-183. 2001.

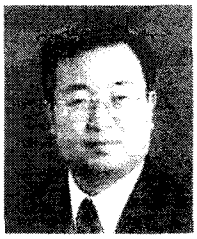
[7] V.G. Oklobdjija, "An algorithm and novel design of leading zero detector: comparison with logic synthesis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol.2, No.1, pp. 124-128, Mar. 1994.

[8] N. Burgess, "The Flagged Prefix Adder and its applications in Integer Arithmetic," *Journal of VLSI Signal Processing*, Vol.31, No.3, pp. 263-271, 2002.

[9] E. Hokenek and R.K. Montoye, "Leading-zero anticipator(LZA) in the IBM RISC system/6000 floating-point execution unit," *IBM Journal of Research and Development*, Vol. 34, No.1, pp. 71-77, Jan. 1990.

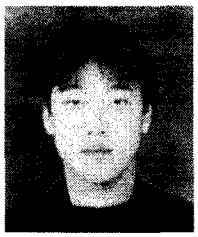
[10] J. D. Bruguera, "Leading-One Prediction with Concurrent Position Correction," *IEEE Transaction on Computers*, Vol.48, No.10, pp.1083-1097, Oct. 1999.

[11] E. Antelo, M. Boo, J.D.Brugera, and E.L. Zapata, "A Novel Design of a two Operand Normalization Circuit," *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, Vol.6, No.1, pp. 173-176, Mar. 1998.



Byeong-Yoon Choi

Byeong-Yoon Choi received the B.S., M.S., and Ph.D. degrees in Electronic Engineering from Yonsei University, Seoul, Korea in 1985, 1987, and 1992, respectively. Since March 1993, he has been with the department of computer engineering, Dong-Eui University, Busan, Korea. He is now a professor. He has been visiting research fellow in University of Auckland, NZ from January 2006 to December 2006. His research interests include System on a chip design of 3-dimensional graphics processor, RISC microprocessor, network processor, cryptographic processor, and embedded system.

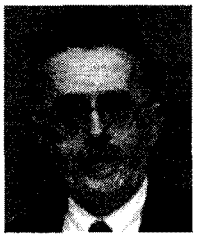


Chang-Soo Ha

Chang-Soo Ha received the B.S. and M.S. degrees in computer engineering from Dongeui University, Busan, Korea in 2003 and 2006 respectively. Since March 2006, he has been a Ph.D. degree student of Dong-Eui University. His research interests include design of 3 dimensional graphics processor, mobile computing, and embedded system design.

Jong-Hyoung Lee

Jong-Hyoung Lee received the BS and MS degrees in Electronic Engineering from Yonsei University, Seoul, Korea and Ph.D from Virginia Tech, USA in 2000. From January 1990 to June 1994, he worked with Daewoo Telecommunications in Seoul, Korea. After one year serving at the Sprint Advanced Technology Lab., Burlingame, CA, as a principal R&D Engineer, he joined Opthos, San Carlos, CA in May 2001 as a system engineer. Since March 2002, he has been with the department of electronic engineering, Dong-Eui University, Busan, Korea. He is now an assistant professor and his research interests include the analysis of transmission effects in optical networks, the design of all-optical metropolitan networks, and the design of low-power CMOS ICs.



Zoran Salcic

Zoran Salcic is a professor of University of Auckland, New Zealand. His research interests include embedded system design, complex digital systems design and prototyping of FPGAs/FPLDs, hardware description languages, reactive embedded systems, custom and reconfigurable computing, and mobile computing.

Duck-Myung Lee

Duck-Myung Lee received the BS and MS degrees in Electronic Engineering from Yonsei University, Seoul, Korea in 1990, 1992 respectively. From March 1998 to July 2000, he worked with Hyundai Electronics in Ichon, Korea as senior ASIC Design Engineer. Since June 2002, he has been with the NexusChips Inc., Seoul, Korea as technical director. His research interests include the design of hardware accelerated graphic chipset, 3D graphics accelerator and mobile computing.