PAPER

Effective Fixed-Point Pipelined Divider for Mobile Rendering Processors

Yong-Jin PARK[†a)], ***Nonmember***, **Woo-Chan PARK**[††], ***Member***, **Jun-Hyun BAE**[†], **Jinhong PARK**[†], ***and*** **Tack-Don HAN**[†b)], ***Nonmembers***

SUMMARY In this paper, we proposed that an area- and speed-effective fixed-point pipelined divider be used for reducing the bit-width of a division unit to fit a mobile rendering processor. To decide the bit-width of a division unit, error analysis has been carried out in various ways. As a result, when the original bit-width was 31-bit, the proposed method reduced the bit-width to 24-bit and reduced the area by 42% with a maximum error of 0.00001%.

key words: *fixed-point divider, mobile, rendering, error analysis*

1. Introduction

In recent years, the design of high performance dividers in hardware has become increasingly important as the needs are increased by applications such as 3D graphics, multimedia, digital signal processing, etc. [1]. In particular, 3D graphics processing has become an important application of processors and, as a result, high performance dividers are required for high performance 3D graphics processing [2]–[4].

For the rasterization step of the 3D graphics pipeline, a pipelined divider for high performance is required. In a desktop PC, a floating-point format is used for the high performance rendering processor. In [5]–[7], a fixed-point format has been used for the low-cost rendering processor of a mobile device. A fixed-point pipelined divider consists of a left shifter for normalization of the dividend and the divisor, a division unit for division execution, and a right shifter for converting the result to a fixed-point format.

Many existing pipelined division unit uses a multiplicative algorithm based on the Taylor-series expansion. Hung et al. stores the first two terms of the Taylor-series in a lookup table (LUT), and executes a division by referencing the LUT in the first step and using a multiplier in the second step [8]. In [9], a cost-effective pipelined division algorithm has been proposed by modifying the Taylor-series and decreasing the LUT size. In [10], the algorithm suggested in [9] has been expanded to handle double precision floating-point numbers.

Manuscript received July 9, 2012.

[†]The authors are with the Department of Computer Science, Engineering College of Yonsei University, 134 Shinchon-Dong, Sudaemoon-Ku, Seoul 120–749, South Korea.

[††]The author is with the Department of Internet Computing, School of Computer Engineering, Sejong University, 98 Kunja-Dong, Kwangjin-Ku, Seoul 143–747, South Korea.

a) E-mail: jini@msl.yonsei.ac.kr

b) E-mail: hantack@msl.yonsei.ac.kr

DOI: 10.1587/transinf.E96.D.1443

In this paper, we propose a fixed-point division algorithm that is effective for chip area and speed by reducing the bit-width of the division unit, while allowing a very small error range. The proposed algorithm utilizes the following features. If there are leading zeros in the dividend and the divisor that are inputs of multipliers inside the division unit, the least significant bits (LSBs) of the normalization result by left shift operations equal 0 as the number of leading zeros. In addition, when a right shift calculation is carried out, the LSBs of the division unit results are discarded according to the number of right-shift calculation in order to convert the result of the division unit to a fixed-point format. By reducing the bit-width of the division unit, the proposed method results in a small error when using these two features.

We performed analytic error analyses of division operation results according to bit-width to prove the proposed method. As a result, a fixed-point divider, using a pipelined divider with a 24-bit width, reduces the area by 42%, as compared to the 31-bit fixed-point pipelined divider proposed in [9], while the maximum error is about 0.00001%. Also, the critical path has been decreased by 8%.

In the next section, we give a brief overview of the pipelined divider and its architectural features. In Sect. 3, we illustrate the new division method and its features. Error analysis, various simulation results, and performance evaluation are given in Sects. 4, 5, and 6. Conclusions are presented in Sect. 7.

2. Related Work

Hung [8] and Jeong [9] proposed pipelined division algorithms. These express division with Taylor-series expansions, and then calculate the upper two or four terms with an LUT and multipliers.

First, the Hung algorithm expands division, as in Eq. (1).

$$\frac{X}{Y} = \frac{X}{Y_h + Y_l} = \frac{X}{Y_h}\left(1 - \frac{Y_l}{Y_h} + \left(\frac{Y_l}{Y_h}\right)^2 - \cdots\right) \tag{1}$$

In Eq. (1), $Y_h = 2^0 y_0 + 2^{-1} y_1 + 2^{-2} y_2 + \cdots + 2^{-(p-1)} y_{p-1}$ and $Y_l = Y - Y_h$. If the upper two terms are used in Eq. (1), then Eq. (2) can be derived as follows:

$$\frac{X}{Y} \approx \frac{X(Y_h - Y_l)}{Y_h^2} \tag{2}$$

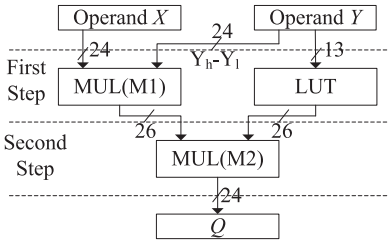


Fig. 1 Block diagram of Hung's algorithm.

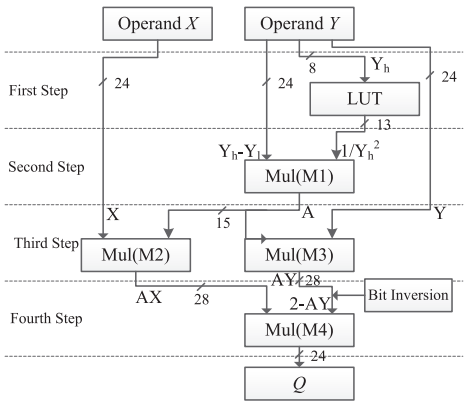


Fig. 2 Block diagram of Jeong's algorithm.

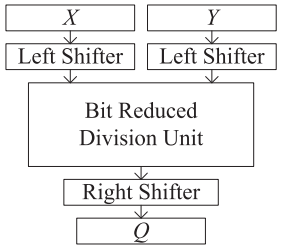


Fig. 3 Proposed fixed-point divider.

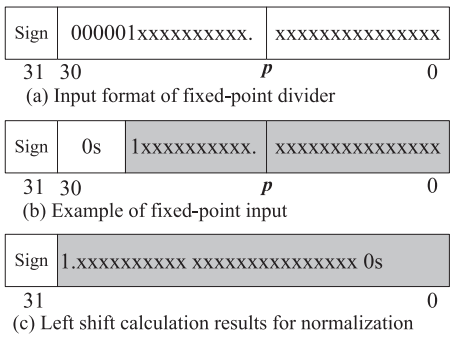


Fig. 4 Example of fixed-point division's number format.

Equation (2) requires an LUT to calculate $1/Y_h^2$, the multiplication of X and $(Y_h - Y_l)$, and the multiplication of $X(Y_h - Y_l)$ and $1/Y_h^2$. The block diagram of this process is shown in Fig. 1. To conduct a division of a single precision format with the explained procedure, approximately 13 KB LUT is required.

Jeong proposed an algorithm that derives coarse quotients ($\tilde{Q}$ and $\tilde{\tilde{Q}}$) that reduce the bit-width of Y_h, and it reduces LUT size with the addition of these two terms [9]. This method conducts division with the procedure in Eq. (3) and it reduces the area by 27%, as compared to the algorithm of [8]. Figure 2 shows the block diagram of [9].

$$\begin{aligned}\frac{X}{Y} &\approx \tilde{Q} + \tilde{\tilde{Q}} = \frac{(X+\tilde{X})(Y_h - Y_l)}{Y_h^2} = (X+\tilde{X})A \\ &= (2X - Y\tilde{Q})A \\ &= (2X - AYX)A = (2 - AY)AX, \\ \text{where} A &= \frac{(Y_h - Y_l)}{Y_h^2}\end{aligned} \tag{3}$$

3. Proposed Method

Whereas a conventional pipelined divider [9] (Fig. 2) uses the normalized value as input data, we use fixed-point value as input data. So, when we use a pipelined divider like [9], we perform a normalization operation of fixed-point input value. The procedure is conducted through a left-shift after finding leading '1' of input value on left shifter of Fig. 3.

Figure 4 shows the structure of the 32-bit fixed point (sign: 1-bit, integer: i-bit, fraction: f-bit). Figure 4 is: (a) an example of a dividend (or divisor), and (b) the value used for division calculation after the normalization process. p of the figure is the starting point of the fraction part.

LSBs of the left shift calculation results for normalization are filled with '0's, as many as leading '0's are (Fig. 4 (b) and (c)). The LSBs are the lowest bits of multiplicand and multiplier, thus the LSBs of the multiplication result values are filled with '0's, as many as leading '0's are. Meanwhile, to convert the final result of the division unit to a fixed-point format, a right-shift calculation is repeated. When the right shift calculations are carried out, the final result of fixed-point formatted division calculation does not use LSBs of division unit as many as right-shifts calculations are repeated.

The proposed method projects these features and uses upper N-bit on the normalization result of the dividend and divisor during division conduction. In this case, since the internal bit-width of the division unit is reduced, the multiplier and the LUT size are reduced, and the area and the latency of the divider can be reduced as well.

4. Boundary Conditions for Error Analysis

Figure 5 shows ε_1 and ε_2, which causes error in the division calculation of variables X' and Y' when only the N-bit of X and Y are used.

Following Fig. 4 and Fig. 5, the maximum values of X and Y occur when all digits have value '1', except the sign-bit, and can be expressed with $2^i - 2^{-f}$. The minimum value of X is '0' and Y is 2^{-f}. This is because an exceptional case of division by zero happens when Y equals 0. This can be expressed in the following formula:

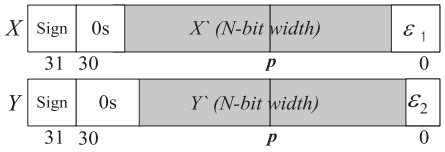


Fig. 5 Parts to be used for division (X' and Y') and loss (ε_1 and ε_2).

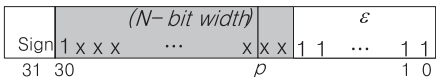


Fig. 6 When ε has the maximum value.

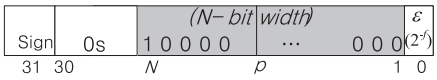


Fig. 7 Minimum value of X' when ε_1 is not zero.

$$X = X' + \varepsilon_1 \qquad Y = Y' + \varepsilon_2$$
$$0 \le X \le 2^i - 2^{-f} < 2^i$$
$$2^{-f} \le Y \le 2^i - 2^{-f} < 2^i \tag{4}$$

The minimum values of ε_1 and ε_2 arel '0' and the maximum values are the case where the most significant bit (MSB) is '1' and the lower bits after the upper N-bit are all '1', as Fig. 6 shows. It can be expressed as Eq. (5).

$$0 \le \varepsilon_1 \le 2^{i-N} - 2^{-f}$$
$$0 \le \varepsilon_2 \le 2^{i-N} - 2^{-f} \tag{5}$$

Since $X' = X - \varepsilon_1$ and $Y' = Y - \varepsilon_2$, following Eq. (4), the ranges of X' and Y' are as follows:

$$0 \le X' \le \left(2^i - 2^{-f}\right) - \left(2^{i-N} - 2^{-f}\right) = 2^i - 2^{i-N}$$
$$2^{-f} \le Y' \le \left(2^i - 2^{-f}\right) - \left(2^{i-N} - 2^{-f}\right) = 2^i - 2^{i-N} \tag{6}$$

When error occurs while division is being carried out using X' and Y' instead of X and Y, ε_1 or ε_2 are not 0. In this case, the minimum value of X' occurs when ε_1 is 2^{-f}, and the value is 2^{N-f}, as in Fig. 7. The minimum value of Y' is also 2^{N-f}. The maximum values of X' and Y' are identical to the maximum values in Eq. (6). With derivation, Eq. (6) can be converted to Eq. (7).

$$2^{N-f} \le X' \le 2^i - 2^{i-N} \quad \text{(in case of } \varepsilon_1 \ne 0\text{)}$$
$$2^{N-f} \le Y' \le 2^i - 2^{i-N} \quad \text{(in case of } \varepsilon_2 \ne 0\text{)} \tag{7}$$

If exceptional cases (overflow or underflow) of division calculation results are handled by a divider, the range of division results, without exception, will be Eq. (8). The range happens because of the expressional limitation of fixed-point representation, and the ranges of minimum and maximum values of division results are the same as the range of X in Eq. (4).

$$0 \le \left|\frac{X'}{Y'}\right| < 2^i \tag{8}$$

5. Error Analysis

In this section, the size of the error, which happens when division is carried out using X' and Y' instead of X and Y, and the portion taken by error in the results of division are analysed. Error, ε_{rr} occurring when the upper N-bit among the 31-bit are used for division, is expressed as Eq. (9):

$$|\varepsilon_{rr}| = \left|\frac{X}{Y} - \frac{X'}{Y'}\right| = \left|\frac{X' + \varepsilon_1}{Y' + \varepsilon_2} - \frac{X'}{Y'}\right|$$
$$= \left|\frac{Y'(X' + \varepsilon_1) - X'(Y' + \varepsilon_2)}{Y'(Y' + \varepsilon_2)}\right| = \left|\frac{\varepsilon_1 Y' - \varepsilon_2 X'}{Y'(Y' + \varepsilon_2)}\right| \tag{9}$$

The error ratio, ε_{rate}, which is the portion taken by error in the final division results, is expressed as Eq. (10):

$$\varepsilon_{rate} = \frac{\varepsilon_{rr}}{\frac{X'}{Y'}} * 100 = \frac{\left|\frac{\varepsilon_1 Y' - \varepsilon_2 X'}{Y'(Y' + \varepsilon_2)}\right|}{\frac{X'}{Y'}} * 100$$
$$= \left|\frac{\varepsilon_1 Y' - \varepsilon_2 X'}{X'(Y' + \varepsilon_2)}\right| * 100(\%) \tag{10}$$

The maximum error value, ε_{max}, can be analysed in three cases: a) $\varepsilon_1 = 0$, $\varepsilon_2 \ne 0$, b) $\varepsilon_2 = 0$, $\varepsilon_1 \ne 0$, and c) $\varepsilon_1 \ne 0$, $\varepsilon_2 \ne 0$. According to Eq. (4), when both ε_1 and ε_2 are '0', $X' = X$ and $Y' = Y$, so that there is no error.

5.1 Case a) $\varepsilon_1 = 0$, $\varepsilon_2 \ne 0$

When ε_1 equals 0, Eqs. (9) and (10) are as follows:

$$|\varepsilon_{rr}| = \left|\frac{0 * Y' - \varepsilon_2 X'}{Y'(Y' + \varepsilon_2)}\right| = \frac{\varepsilon_2 X'}{Y'(Y' + \varepsilon_2)} \tag{11}$$

$$\varepsilon_{rate} = \left|\frac{0 * Y' - \varepsilon_2 X'}{X'(Y' + \varepsilon_2)}\right| * 100 = \frac{\varepsilon_2}{Y' + \varepsilon_2} * 100(\%) \tag{12}$$

The maximum error occurs when X' has the maximum value and Y' has the minimum value, in accordance with Eq. (11). Since $\varepsilon_1 = 0$, $X' = X$, and $X' = 2^i - 2^{i-N}$, following Eq. (6), the minimum value of Y' is 2^{N-f}, in accordance with Eq. (7). Then, as shown in Fig. 7, ε_2 is 2^{-f}. If each of these values is inserted into Eq. (11), the following formula can be derived:

$$|\varepsilon_{max}| = \frac{\varepsilon_2 X'}{Y'(Y' + \varepsilon_2)} = \frac{2^{-f} * \left(2^{-i} - 2^{i-N}\right)}{2^{N-f} * \left(2^{N-f} + 2^{-f}\right)} \tag{13}$$

If the formula is expanded, it can be expressed as follows:

$$\frac{2^{-f} * \left(2^{-i} - 2^{i-N}\right)}{2^{N-f} * \left(2^{N-f} + 2^{-f}\right)} = \frac{2^i 2^{-f} * \left(1 - 2^{-N}\right)}{2^N 2^{-2f} * \left(2^N + 1\right)}$$
$$= \frac{1 - 2^{-N}}{2^{N-i-f} * \left(2^N + 1\right)}$$
$$= \frac{1 - 2^{-N}}{2^{N-31} * \left(2^N + 1\right)}$$

If Y' and ε_2 are inserted into Eq. (12), the maximum error ratio is as follows:

$$\varepsilon_{max\_rate} = \frac{\varepsilon_2}{Y' + \varepsilon_2} * 100 = \frac{2^{-f}}{2^{N-f} + 2^{-f}} * 100 = \frac{1}{2^N + 1} * 100(\%)$$

5.2 Case b) $\varepsilon_2 = 0, \varepsilon_1 \neq 0$

When ε_2 equals 0, Eqs. (9) and (10) become as follows:

$$|\varepsilon_{rr}| = \left|\frac{\varepsilon_1 * Y' - 0 * X'}{Y'(Y' + 0)}\right| = \frac{\varepsilon_1}{Y'} \quad (14)$$

$$\varepsilon_{rate} = \left|\frac{\varepsilon_1 * Y' - 0 * X'}{X'(Y' + 0)}\right| * 100 = \frac{\varepsilon_1 Y'}{X'Y'} * 100 = \frac{\varepsilon_1}{X'} * 100(\%) \quad (15)$$

The maximum error ε_{max} occurs when Y' has the minimum value and ε_1 has the maximum value, in accordance with Eq. (14). Also the maximum error rate, ε_{max_rate}, occurs when X' is minimum in accordance with Eq. (15). If the maximum value of ε_1 is $2^{i-N} - 2^{-f}$ following Eq. (5), then the MSB of X' is 1. The minimum value of X' is 2^{i-1}. It is expressed in Fig. 8.

Since ε_2 equals 0, the next equation can be derived when X' is inserted into Eq. (8) to calculate the minimum of Y' without overflow.

$$\frac{X'}{Y'} = \frac{2^{i-1}}{Y'} < 2^i$$

With the substitution of right 2^i and left Y', Eq. (16) can be derived. As a result of Eq. (16), the minimum of Y' should be bigger than 2^{-1}.

$$Y' > \frac{2^{i-1}}{2^i} = 2^{-1} \quad (16)$$

As Fig. 9 shows, the minimum value of Y', which is bigger than 2^{-1}, is $2^{-1} + 2^{-f}$. If the value is inserted into Eq. (14), the maximum error is expressed, as in the following formula:

$$|\varepsilon_{\max}| = \frac{2^{(i-N)} - 2^{-f}}{2^{-1} + 2^{-f}} \quad (17)$$

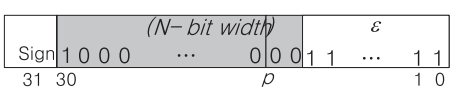


Fig. 8 Minimum of X' and maximum of ε_1 when ε_1 is the maximum.

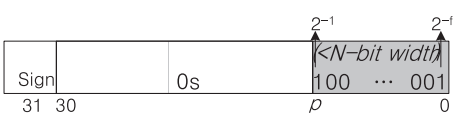


Fig. 9 Minimum value of Y' when $Y' > 2^{-1}$.

When $X' = 2^{i-1}$ and $\varepsilon_1 = 2^{i-N} - 2^{-f}$ are inserted to equation, the maximum error rate can be expressed as follows Eq. (15):

$$\varepsilon_{max\_rate} = \frac{2^{i-N} - 2^{-f}}{2^{i-1}} * 100 = \frac{2^{i-N} - 2^{-31+i}}{2^{i-1}} * 100 = (2^{-N+1} - 2^{-30}) * 100(\%)$$

5.3 Case c) $\varepsilon_1 \neq 0, \varepsilon_2 \neq 0$

Since Y' uses N-bit of leading '1' from Y and the rest of digits are ε_2, Y' and ε_2 are correlated. Since $Y' \gg \varepsilon_2$, in accordance with Eq. (9), ε_{rr} would have a maximum value when Y' has a small value rather than a big value of ε_2. Equation (18) can be derived as:

$$\varepsilon_{rr} = \left|\frac{\varepsilon_1 Y' - \varepsilon_2 X'}{Y'(Y' + \varepsilon_2)}\right| \leq \left|\frac{\varepsilon_{1MIN} Y'_{MIN} - \varepsilon_{2MAX} X'_{MAX}}{Y'_{MIN}(Y'_{MIN} + \varepsilon_{2MAX})}\right| \quad (18)$$

In accordance with Eq. (7), the maximum value of X' is $2^i - 2^{i-N}$, and the minimum value of ε_1 is 2^{-f}, following Fig. 7. In accordance with Eq. (7), the minimum value of Y' is 2^{N-f}, and ε_2 is 2^{-f}, following Fig. 7. The result of inserting these values into Eq. (18) is Eq. (19). As Eq. (19) shows, ε_{max} is always smaller than Eq. (13). So the Case c) always has a smaller error than Case a), and it does not affect the final maximum error of division result.

$$\varepsilon_{max} = \left|\frac{\varepsilon_1 Y' - \varepsilon_2 X'}{Y'(Y' + \varepsilon_2)}\right| = \frac{2^{-f} * \left(2^i - 2^{i-N}\right) - 2^{-f} * 2^{N-f}}{2^{N-f} * \left(2^{N-f} + 2^{-f}\right)} < \frac{2^{-f} * \left(2^i - 2^{i-N}\right)}{2^{N-f} * \left(2^{N-f} + 2^{-f}\right)} \quad (19)$$

6. Results of Error Analysis

In this paper, we measured the maximum error and maximum error ratio in two cases. The first case is analysis according to a number of fraction bits, which are used in the fixed-point format of OpenGL ES 2.0, which is standard for mobile devices. The second case is the analysis about the case of $i = 31$, which occurs for the largest maximum error value according to Eqs. (13) and (17).

First, because the fixed-point format of OpenGL ES 2.0 allocates 16-bit to fraction bits [11], we assume that i is 15-bit and f is 16-bit. Table 1 shows the maximum error and maximum error ratios in accordance with each N. When $N = 24$ for division, the maximum error occurs in case b). The error is 0.0038756, and the portion taken by this error as a result of division is 0.0000118%. When $N = 20$, the maximum error is 0.0624676, and the portion taken by this error as a result of division is 0.00019%.

The second case is the analysis of the maximum error that occurs, which uses i to 31-bit according to Eqs. (13) and (17). Table 2 shows the maximum error and maximum error ratios in accordance with each N. Since errors in case a) and case c) are smaller than 1, there is no error in

Table 1 Maximum error and error rate according to N in case of $i = 15$ and $f = 16$.

i	f	N		Case a)	Case b)	Case c)	Max_error	Error_rate
15	16	24	Error	$<2^{-16}$ (=0)	0.00387	$<2^{-16}$ (=0)	0.00387	0.00001 %
			Rate (%)	0	0.00001	0		
15	16	20	Error	0.00195	0.06246	0.00195	0.06246	0.000019 %
			Rate (%)	0.00009	0.00019	0.00009		
15	16	16	Error	0.49998	0.999930	0.49996	0.999930	0.00305 %
			Rate (%)	0.00152	0.00305	0.00152		

Table 2 Maximum error and error rate according to N in case of $i = 31$ and $f = 0$.

i	f	N		Case a)	Case b)	Case c)	Max_error	Error_rate
31	0	24	Error	$<2^{0}$ (=0)	127	$<2^{-16}$ (=0)	127	0.00001 %
			Rate (%)	0	0.00001	0		
31	0	20	Error	$<2^{0}$ (=0)	2047	$<2^{0}$ (=0)	2047	0.00019 %
			Rate (%)	0	0.00019	0		
31	0	16	Error	$<2^{0}$ (=0)	32767	$<2^{0}$ (=0)	32767	0.00305 %
			Rate (%)	0	0.00305	0		

Table 3 Delay and area cost comparison with Jeong's and Hung's algorithm.

		1st Step	2nd Step	3rd Step	4th Step	Total
Jeong's	Delay (τ)	576B: 5.0	31*18: 10.5	31*19: 10.5	35*35: 11.5	37.5
(N=31)	Area (fa)	576B: 158	31*18: 477	31*19*2:1,006	35*35: 1027	2,668
Jeong's	Delay (τ)	208B: 4.0	24*13: 10.0	24*15: 10.0	28*28: 10.5	34.5
(N=24)	Area (fa)	208B: 57	24*13: 259	24*15*2: 592	28*28: 636	1,544
Hung's	Delay (τ)	32*32: 10.5	34*34: 11.5	-	-	21.5
(N=31)	Area (fa)	272KB: 54,400	34*34: 976	-	-	56,209
Hung's	Delay (τ)	32*32: 833 24*24: 10.0	26*26: 10.5	-	-	20.5
(N=24)	Area (fa)	13KB: 3,640 24*24: 464	26*26: 547	-	-	4,651

the division result related to the value of N. When $N = 24$ for case b), the maximum error is 127, and the portion taken by this error as a result of division is 0.0000118%. When $N = 20$, the maximum error is 2047, and the portion taken by this error as a result of division is 0.00019%.

The comparison, in terms of the delay and area cost of the proposed scheme in relation to previous approaches, is provided in Table 3. The delay and area cost have been calculated based on the analytical method in [12]. The delays are expressed in terms of τ- the delay of a complex gate such as one full adder. The unit employed for the area cost estimation is the size of one full adder, fa.

According to Jeong's algorithm, when $N = 24$ for division, following the results in Table 3, it decreased the area by 42%, as compared to $N = 31$, and critical path delay decreases by 8%. Hung's algorithm decreased by 92%, as compared to the case of $N = 31$, and critical path delay decreases by 5%.

7. Conclusion

For the rasterization step of the 3D graphics pipeline, a divider, especially a pipelined divider for high performance, is required. Existing pipelined division algorithms use Taylor-series expansion and this uses a large LUT. In this paper, an effective fixed-point pipelined divider for area and speed is proposed for reducing the bit-width of the division unit to fit the low-cost rendering processor. To decide the bit-width for a division unit, error analysis has been carried out in various ways.

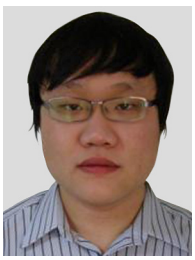
When bit-width is restricted to 24-bit from the original 31-bit, a 42% decrease of the area is possible. This is because of the reduction of the input bit-width of the multiplier and the reduction of the LUT size in the division unit. Also, analysis results of error occurrence, followed by input bit restriction, shows a maximum error of 0.00002% on the divider implemented with 24-bit. The proposed structure can be applied not only to a pipelined divider, but also to other division algorithms.

Acknowledgments

This research was supported by Samsung Advanced Institute of Technology and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2011-0027450).

References

[1] N. Ide, M. Hirano, Y. Endo, S. Yoshioka, H. Murakami, A. Kunimatsu, T. Sato, T. Kamei, T. Okada, and M. Suzuoki, "2.44-GFLOPS 300-MHz floating-point vector-processing unit for high-performance 3D graphics computing," IEEE J. Solid-State Circuit., vol.35, no.7, pp.1025–1033, 2000.
[2] A. Kugler, "The setup for triangle rasterization," Proc. 11th Eurographics Workshop Computer Graphics Hardware, pp.49–58, Aug. 1996.
[3] A.J. Thakkar and A. Ejnioui, "Design and implementation of double precision floating point division and square root on FPGAs," 2006 IEEE Aerospace Conf., Montana, United States, pp.1–7, March 2006.
[4] K. Yoshida, T. Sakamoto, and T. Hase, "A 3D graphics library for 32-bit microprocessors for embedded systems," IEEE Trans. Consum. Electron., vol.44, no.4, pp.1107–1114, 1998.
[5] 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, 2005.
[6] D. Kim and L.-S. Kim, "Area-efficient pixel rasterization and texture coordinate interpolation," Comput. Graph., vol.32, pp.669–681, 2008.
[7] W.-J. Lee, W.-C. Park, V.P. Srini, and T.-D. Han, "Simulation and development environment for mobile 3D graphics architectures," IET Comput. Digit. Tech., vol.1, pp.501–507, 2007.
[8] P. Hung, H. Fahmy, O. Mencer, and M.J. Flynn, "Fast division algorithm with a small lookup table," Conf. Record 33rd Asilomar Conf. Signals, Systems, and Computers, vol.2, pp.1465–1468, May 1999.
[9] J.-C. Jeong, W.-C. Park, W. Jeong, T.-D. Han, and M.-K. Lee, "A cost-effective pipelined divider with a small lookup table," IEEE Trans. Comput., vol.53, no.4, pp.489–495, 2004.
[10] S.B. Singh, J. Biswas, and S.K. Nandy, "A cost effective pipelined divider for double precision floating point number," Int. Conf. Application-specific Systems, Architectures and Processors, Colorado, United States, pp.132–137, Sept. 2006.
[11] A. Munshi and J. Leech, "OpenGL ES common Profile Specification," http://www.khronos.org, accessed July 1. 2012.
[12] J.A. Pineiro and J.D. Bruguera, "High-speed double-precision computation of reciprocal, division, square root, and inverse square root," IEEE Trans. Comput., vol.51, no.12, pp.1377–1388, 2002.



Yong-Jin Park received the B.S. and M.S. degrees in computer science from Yonsei University in 2003 and 2005, respectively. He is a PhD student in the Department of Computer Science at Yonsei University, Korea. He is currently researching architecture for 3D computer graphics.

Woo-Chan Park is an associate professor at Sejong University, Korea. His research interests include 3D rendering processor architecture, ray tracing accelerator, parallel rendering, high performance computer architecture, computer arithmetic, and ASIC design. He received the BS, MS, and PhD degree in computer science from Yonsei University, Seoul, Korea.

Jun-Hyun Bae received the B.S. degrees in information & communication engineering from Changwon National University in 2011. He is now M.S. degree course in computer science from Yonsei University, Seoul, Korea.

Jinhong Park works at LG Electronics. His research interests include 2D/3D graphics hardware architecture, GPGPU, and SoC platform. He received the BS, MS, and PhD degree in computer science from Yonsei University, Seoul, Korea.

Tack-Don Han is a professor in the Department of Computer Science at the Yonsei University, Korea. His research interests include high performance computer architecture, media system architecture, and wearable computing. He received a PhD in computer engineering from the University of Massachusetts.