A New Pipelined Divider with a Small Lookup Table

Jong-Chul Jeong, Woong Jeong, Hyun-Jae Woo,
Seung-Ho Kwak, Woo-Chan Park*, Moon-Key Lee, Tak-don Han*
Dept. of Electrical and Electronics Engineering, Yonsei University, Seoul, Korea
*Dept. of Computer Science, Yonsei University, Seoul, Korea
134 Shinchon-dong, Seodaemun-gu, Seoul, 120-749, Korea
Telephone : +82-2-2123-4731
jongblue@spark.yonsei.ac.kr

ABSTRACT
Most of the existing hardware dividers execute the division operation with iterations, while recently the research on the pipeline-able division unit is tried. It is a difficult point in the known pipeline-able division unit that a large lookup table is required. In this paper, we propose the cost-effective pipeline-able divider which needs smaller lookup table than other pipeline-able divider. The latency of the proposed divider is 3 cycles and this divider reduced the area about 70% in comparison with the one of P. Hung.

Keywords
Computer Arithmetic, Division, Floating Point, Look-up Table.

1. INTRODUCTION
The division is lower used than other arithmetic in computer operation. For that reason, the divider has designed in low size and with iterative method. Though the latency of a divider is longer than one of other arithmetic, total system is not much affected by latency[1].

Because a technology of VLSI grows and an area of high performance application expands, the division without iteration is proposed. Specially, 3D graphic processing is embossed on a important application of processor, and the pipeline-able division with high quality is needed for high performance of 3D graphic acceleration[2][3].

P. Hung proposed the pipeline-able division to modify Taylor series expansion[2]. P. Hung's divider has a larger area than the divider implemented by a iterative algorithm. In this paper, we propose the new method to modify pipeline-able division algorithm of P. Hung. The latency of the proposed divider is 3 cycles and this divider reduced the area about 70% in comparison with the one of P. Hung.

This paper is consists of the following chapters. Chapter 2 explains the related works and chapter 3 explains the proposed algorithm and architecture. In chapter 4, we decide the size of look-up table and multipliers in proposed divider. We compare the proposed algorithm with other algorithms in chapter 5 and have a conclusion in chapter 6.

2. RELATED WORKS
The division algorithm falls into the two categories of multiplicative and subtractive algorithms[4]. Multiplicative methods use a hardware integrated with the floating-point multiplier and look-up table and they calculate the quotient by approximation. Newton-Raphson and series expansion algorithms are the well-known algorithms in the multiplicative methods. They calculate a quotient by multiplying a dividend by a reciprocal from a look-up table.

The disadvantage of these algorithms is that they need a large look-up table for the high-speed division. For example, the 16-bit seed Newton-Raphson or series expansion divider needs a 64KB look-up table[5]. These 16-bit seed dividers execute the division operation by two iterations at the single precision. If the 8-bit seed dividers are used, the look-up table size is 128B, which is smaller than the 16-bit seed look-up table. But the latency of this divider is longer than one of the other dividers because of three iterations. Accurate quotient approximation algorithm which uses Taylor series expansion has two or three look-up table, and it requires a 19.5KB look-up table for one-iteration-division at the single precision[6].

The algorithms proposed by A. Liddicoat and by P. Hung have the recently proposed pipeline-able division algorithms[2][3]. A. Liddicoat's divider calculates the 3rd order term of Newton-Raphson algorithm, and the latency of this divider is decreased by the parallelism. P. Hung's divider uses Taylor series expansion, and its strong points are a simple architecture and a small look-up table size.

P. Hung shows the division operation can be replaced to the follow equation by Taylor series expansion.

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

In the equation (1), $Y_h = 2^0 y_0 + 2^{-1} y_1 + 2^{-2} y_2 + \cdots + 2^{-(p+1)} y_{p+1}$ and $Y_l = Y - Y_h$. Because X and Y are the normalized fixed point number -- it means x_0 and y_0 are always 1 --, the variable in the equation (1) have the boundary conditions such as the equation (2).

$$\begin{aligned} 1 \le X, Y < 2 \\ 1 \le Y_h < 2 - 2^{-p+1} \\ 0 \le Y_l < 2^{-p+1} \end{aligned} \quad (2)$$

0-7803-7363-4/02/$17.00 ©2002 IEEE.

If we use the Taylor series 1st and 2nd order term in the equation (1), we can approximate the equation (1) such as the equation (3).

$$\frac{X}{Y} \approx \frac{X(Y_h - Y_l)}{Y_h^2} \tag{3}$$

According to the equation (3), the division operation can be executed by means of multiplying the dividend X by $Y_h - Y_l$ from the divisor, and multiplying the previous result by $1/Y_h^2$ from look-up table. It shows how to process p. Hung's division algorithm in <Figure 1>. In this block diagram, $Y_h - Y_l$ can be executed without real subtraction because it can be calculated by modifying to the Booth encoder of the multiplier which multiplies X by $Y_h - Y_l$[7].

The algorithm proposed by P. Hung approximates $1/Y_h^2$ by using a look-up table. The look-up table size can be decreased to other existing algorithm by this approximation. Nevertheless, the divider implemented by P. Hung's method has the weak point that its area is larger than a iterative divider's. P. Hung's divider needs about 13KB look-up table at the single precision and it is impossible to implement at the double precision because of its huge area. So the area of look-up table is an important issue in the pipeline-able divider.

3. PROPOSED DIVIDER ARCHITECTURE

The main goal of this paper is to reduce the look-up table size, the weakest point of pipeline-able divider. If P. Hung's algorithm is executed with a very small look-up table, the coarse quotient is calculated. The remainder can be calculated by multiplying the dividend by this coarse quotient. If P. Hung's algorithm is executed with this remainder replaced at the position of dividend and two quotients are added, we can calculate the final quotient. This process seems very complex, but it can be executed with a smaller look-up table and 4 less-precision multiplier than P. Hung's by means of eliminating the redundant operation.

3.1 Proposed Algorithm

The proposed algorithm is as follows. Coarse quotient $\tilde{Q}$ can be defined by reducing the bit-width of Y_h in the equation (3).

$$\tilde{Q} \approx \frac{X(Y_h - Y_l)}{Y_h^2} \tag{4}$$

The remainder can be calculated by the following equation.

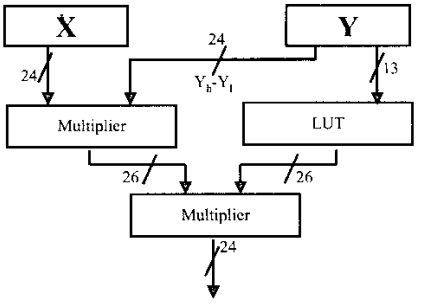
$$\tilde{X} \approx X - Y \cdot \tilde{Q}$$

Now, Let replace X with the remainder $\tilde{X}$ in the equation (4). If two quotients are added, the final quotient can be calculated.

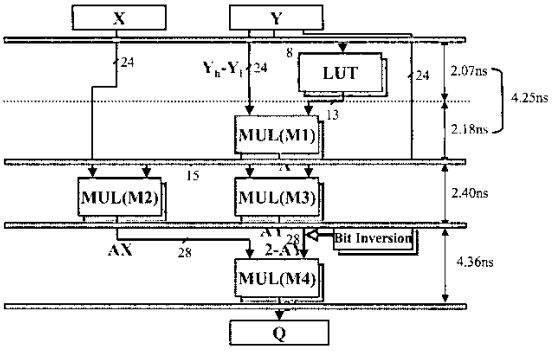
$$\tilde{\tilde{Q}} \approx \frac{\tilde{X}(Y_h - Y_l)}{Y_h^2}$$

$$\frac{X}{Y} \approx \tilde{Q} + \tilde{\tilde{Q}}$$

The above equation can be rearranged by the follow.



<Figure 1> P. Hung's division algorithm



<Figure 2> The proposed algorithm architecture

$$\begin{aligned}\frac{X}{Y} &\approx \tilde{Q} + \tilde{\tilde{Q}} \\ &= \frac{(X + \tilde{X})(Y_h - Y_l)}{Y_h^2} \\ &= (X + \tilde{X})A \\ &= (2X - Y\tilde{Q})A \\ &= (2X - AYX)A \\ &= (2 - AY)AX\end{aligned} \tag{5}$$

Where

$$A = \frac{(Y_h - Y_l)}{Y_h^2}$$

3.2 Proposed Hardware Architecture

It shows the hardware block diagram for the proposed algorithm in <Figure 2>. Processing the equation (5) can be divided into 4 steps. First, $1/Y_h^2$ is calculated in the look-up table. Second, A is calculated to multiply $1/Y_h^2$ by $Y_h - Y_l$. Third, AX and AY are calculated through parallel multiplications. Last, the final quotient AX(2-AY) is calculated by multiplication, where 2-AY can be calculated by the bit-inversion of AY. For example, 0.11010 can be calculated by the bit-inversion of 1.00101, which is the 1 ulp little value than 0.11011, the result of 2-1.00101. If a bit-inversion is used, compared with a real subtraction, an adder is removed but the error is added. In the chapter 4, the influence of this error will be analyzed.

A look-up table and 4 multipliers are required to process the proposed algorithm. The latency of this algorithm is 1 LUT + 3

MUL because two multiplications are executed parallel. The area and latency of 3 multipliers out of 4 multipliers are somewhat less than P. Hung's because their bit-width is smaller than the bit-width of P.Hung's. In the single precision, the size of final multiplier is 28×28, two multipliers are 24×15, and the other is 24×13. The value in the look-up table is also accessed in very short time because the size of look-up table is very small. So the latency of this divider can be determined 3 cycles in the single precision. In the first cycle, it is executed to calculate A with reference of look-up table and multiplication. Two parallel multiplications are executed in the second cycle and the final multiplication is executed in the last cycle.

It shows the latency of each unit in the proposed divider based on the complied macro of Samsung 0.25um ASIC technology in <Figure 2>[8]. The entry number and bit-width of look-up table, and each bit-width of multipliers in the <Figure 2> will be determined in the chapter 4, Error Analysis.

4. ERROR ANALYSIS

The error analysis of the proposed algorithm is essential for the hardware divider design because it provides the base data to determine the look-up table size and the multiplier bit-width. Also the error analysis enables to compare with other division algorithms.

Four kinds of error are considered to analyze the error of the proposed algorithm in this paper. First error is the error caused by the entry restriction of a look-up table, which is generated by limiting the bit-width of Y_h to p. Second error is caused by the bit-width restriction of a look-up table, which is determined by the accuracy of look-up table. Third error is caused by the rounding of multipliers, which is determined by the rounding position of multipliers. Last error is caused by a bit-inversion, which is always 1 ulp.

4.1 Selection of the Entry and Bit-width of Look-up Table

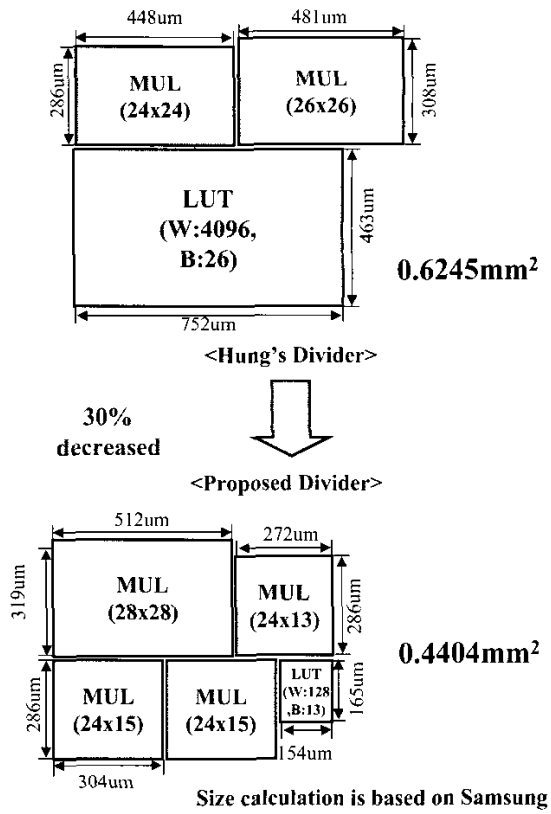
When the above four errors are considered, we can get the entry and bit-width of look-up table. It shows the entry and bit-width of look-up table by significant bit in <Table 1>. m is significant bit + 1 and q is bit-width of look-up table. The entry number of look-up table is 2^{p-1}. In the single precision (m=24), the entry number of look-up table is 128 and the bit-width of look-up table is 13-bit. In the double precision (m=53), the entry number of look-up table is 16k and the bit-width of look-up table is 28-bit.

4.2 Selection of Multipliers' Size

In the single precision, multipliers' size are 24×13, 24×15, 24×15 and 28×28 or 24×13, 24×17, 24×17 and 27×27. I will omit the detailed calculation.

5. COMPARISON WITH OTHER ALGORITHMS

In this chapter, we examine merits and demerits of the proposed algorithm compared with P. Hung's algorithm. And, characters and capacities of the proposed algorithm are checked to compare with subtractive algorithm, Newton-Raphson algorithm and series expansion algorithm.



<Figure 3> Area comparison two algorithms

<Table 1> Look-up table size of the proposed algorithm

m	*p*	# of entry	*q*	ROM size (Byte)
15	6	32	8	32
16	6	32	9	36
.	.	.	.	.
23	7	64	14	112
24	8	128	13	208
25	8	128	14	224
.	.	.	.	.
52	15	16K	27	54K
53	15	16K	28	56K

5.1 Compare with P. Hung's Algorithm

P. Hung's algorithm consists of one look-up table and two multiplier[2] (see <Figure 1>). Its latency is 1 LUT + 1 MUL or 2 MUL. As the accuracy of divider increase, the latency of multiplier increases linearly, but the latency of look-up table increases by geometrical progression. So, the total latency is 2

MUL in case of low accuracy and 1 LUT + 1 MUL in case of high accuracy. In the single precision, the total latency is 2 MUL because a latency of look-up table is shorter than one of the multipliers. In the double precision, the total latency is 1 LUT + 1 MUL because a latency of look-up table is longer.

The latency of proposed algorithm is 1 LUT + 3 MUL. But, the size of look-up table is smaller than P. Hung's and two out of three multipliers have a short latency because of a short bit-width. So, the latency is a little longer than P. Hung's in the single precision but is much shorter in the double precision.

Their algorithms are pipeline-able and are taken 1 cycle to process one pipeline step. In the single precision, the latency of P. Hung is 2 cycle and the proposed algorithm is 3 cycle. In the double precision, the latency of the proposed algorithm is 8 cycle. But, it is impossible to implement P. Hung's algorithm in the double precision because of its huge look-up table.

It shows the size of two algorithms in the single precision in <Figure 3>. It excepts the size of flip-flops in pipeline. As a result, the proposed algorithm is a 30% smaller size than P.Hung's. Two algorithms summarize in <Table 2>.

5.2 Compare with Other Algorithms

In <Table 3>, it shows a comparison the proposed algorithm with other algorithm in the single precision. It decides a latency and a size on the basis of each division algorithm[5]. The proposed algorithm is the smallest size out of algorithms which are not more than 5 cycle in latency . But, the proposed algorithm is bigger size than SRT radix-4, 8-bit seed Newton-Raphson and 8-bit seed series expansion. So, the proposed algorithm is efficient in system with a short latency and a high frequency of division.

<Table 2> Comparison between P. Hung's and the proposed

		P. Hung's	Proposed
General	Latency	2 MUL or 1 MUL + 1 LUT	1 LUT + 3 MUL
	Through-put	1 cycle	1 cycle
	LUT	1 (# of entries : $2^{m/2}$, bit-width : $m+2$)	Very smaller than Hung's
	MUL	*2*	*4*
In the single	Latency	*7.6ns*	*11.01*
	Latency	*2 cycle*	*3*
	LUT	*13KB*	*208B*
	MUL	26×26, 24×24	24×13, 24×15(2), 28×28 or 24×13, 24×17(2), 27×27
In the double	Latency	Impossible to implement	8
	LUT		56KB
	MUL		53×28, 53×28(2), 58×58

<Table 3> Comparison with other algorithm

Algorithm and implementation method		latency	Pipeline-ability	# of iterations	Area	Size of LUT
SRT Radix-4		12	×	12	1.5	
Newton-Raphson	8b seed	6	×	3	1.0	128B
	16b seed	4	×	2	22	64KB
Series expansion	8b seed	7	×	3	1.0	128B
	16b seed	5	×	2	22	64KB
Accurate quotient approximation		3	×	1	9.8	19.5KB
Hung's		2	○		6.8	13KB
Proposed		3	○		4.6	240B

6. CONCLUSION

This paper proposes the new pipeline-able division algorithm. And, the proposed algorithm is compared with other algorithm. The proposed divider consists of one look-up table and four multiplier in the single precision and is smaller size than other pipeline-able dividers. The latency of the proposed divider is 3 cycles and this divider reduced the area about 70% in comparison with P. Hung's. This divider is efficient in system with high frequency of division such as 3D graphics accelerator[11]. The proposed divider has analyzed an error in using a numerical formula and simulated in using a high-level language.

7. REFERENCES

[1] S. F. Oberman and M. J. Flynn, "Design Issues in Division and Other Floating Point Operations,", *IEEE Transactions on Computers*, Vol. 46, No. 2, pp 154-161, Feb. 1997.

[2] P. Hung, H. Fahmy, O. Mencer and M. J. Flynn, "Fast division algorithm with a small lookup table," *Conference Record of the Thirty-Third Asilomar Conference on Signals, Systems and Computers*, Vol. 2, May pp 1465-1468, 1999.

[3] A. A. Liddicoat and M. J. Flynn, "Pipeline-able Division Unit," *Technical Report No. CSL-TR-00-809*, Computer Systems Laboratory, Stanford University, pp 11-28, Sep. 2000

[4] I. Koren, *Computer Arithmetic Algorithms*, Prentice Hall, pp 153-161, 1993.

[5] S. F. Oberman and M. J. Flynn, "Division Algorithms and Inplementations," *IEEE Transactions on Computers*, Vol. 46, No. 8, pp 833-854, Aug. 1997.

[6] D. Wong and M. J. Flynn, "Fast Division Using Accurate Quotient Approximations to Reduce the Number of Iterations," *IEEE Transactions on Computers*, Vol. 41, No. 8, pp 981-985, Aug. 1992.

[7] C. N. Lyu and D. Matula, "Redundant Binary Booth Recoding," *Proceedings of IEEE Symposium on Computer Arithmetic*, pp 50-57, Jul. 1995.

[8] Samsung Electronics Co. Ltd., *MDL110 0.25um 2.5V CMOS Standard Cell Library for Pure Logic/MDL Products*, 1999.

[9] A. Kugler, "The Setup for Triangle Rasterization," *11th Eurographics Workshop on Computer Graphics Hardware*, pp 49-58, Aug. 26 1996.