EFFICIENT SIMULTANEOUS ROUNDING METHOD REMOVING STICKY-BIT FROM CRITICAL PATH FOR FLOATING POINT ADDITION

Woo-Chan Park*, Tack-Don Han*, and Shin-Dug Kim*
*Dept. of Computer Science, Yonsei University, Seoul 120-749,Korea
***E-mail**: {chan, hantack, sdkim}@kurene.yonsei.ac.kr

Abstract

Processing flow of the conventional floating point addition /subtraction operation consists of several steps, i.e., alignment, addition/subtraction, normalization, and rounding stages *in this order. In [11], a floating adder/subtractor performing addition/subtraction and IEEE rounding in parallel was presented, where any additional execution time nor any high speed adder for rounding operation was not required. But, the Sticky-bit, which is generated at the alignment stage, is included in the critical path delay. In this research, a technique to remove the Sticky-bit generation from the critical path is proposed. Its hardware model and correctness proofs are provided and evaluated. Proposed floating point adder provides effectiveness in the points of chip area and its execution time.*

I. INTRODUCTION

FPU (Floating Point Unit) is the principle component in graphics accelerators, DSP (Digital Signal Processors), and high performance computer systems. As the chip integration density increases due to advances in semiconductor technology, it has become possible for the FPU to be placed on a single chip together with the CPU (Central Processing Unit)[1,2,3,4,5]. In this case, a compromise between the chip area available and the FPU functions included should be considered. In general, only some primary arithmetic units such as an adder/subtractor and a multiplier are integrated on a chip and additional software handling is required for further complete floating point operations. Therefore, the overall floating point operations are greatly affected by how the floating point multiplier and the adder/subtractor are designed.

In general, processing flow of the conventional floating point addition/subtraction operation consists of alignment, addition/subtraction, normalization, and rounding stages[5,6,7]. To process the rounding operation, either a high speed incrementor or adder is required. Furthermore, renormalization might occur due to an overflow from the rounding operation. To overcome these disadvantages, several methods have been provided.

First, [8] proposed that the floating point addition/subtraction operation can be performed by the two paths according to the absolute value of the difference of the two exponent values. If that absolute value is less than or equal to one, a high performance shifter for alignment stage can be replaced with a simple multiplexer. Otherwise, a high performance shifter for normalization stage can be changed with simple multiplexer. Thus, a critical path of the floating point addition/subtraction operation is either alignment, addition/subtraction, and rounding operations or addition/subtraction, normalization, and rounding operations. By using this technique, a reasonable amount work required for a shift operation for either alignment or normalization can be reduced with total floating point addition/subtraction operation. This method is adapted in [9,10]. But, a rounding stage is still required and additional hardware components are needed to support the two-path method. And, because controlling the two dataflows is much more complex than the conventional one, it requires additional complex logics and complex routings to be difficult to implement. Therefore, it causes to waste the area on the chip for the additional logics and the routing area with wire delays.

Second, a structure for performing rounding and addition/subtraction operation in parallel was provided in [11]. By analyzing the operational flow of floating point addition/subtraction operation, performing rounding and addition/subtraction in parallel can be achieved with area efficiency. The floating point adder/subtractor presented in [11] does not require any additional execution time nor any high speed adder for rounding operation. In addition, because the rounding step can be performed prior to the normalization operation, the renormalization step is not required. Thus, performance improvement and cost-effective design can be achieved by this approach. But, the Sticky-bit (Sy-bit), which is generated at the alignment stage, is included in the critical path delay. Thus, the generation of the Sy-bit affects the overall performance of floating point addition/subtraction operation.

In this paper, the technique for removing Sy-bit from the critical path delay of [11] is provided. Its hardware model and correctness proofs are provided and evaluated. Hence, the proposed floating point adder can provide effectiveness in the points of chip area and its execution time. Next section presents a brief overview of IEEE rounding methods and basic notations and definitions. A hardware model which can execute rounding and addition/subtraction in parallel is provided in Section 3. Section 4 illustrates the proposed method for performing rounding and addition in parallel. Comparison and conclusion are made in Section 5.

II. BACKGROUND

In this section, IEEE rounding method[12] and basic notations and definitions are provided. A normalized floating point number according to the IEEE's standard is expressed as $A=(-1)^s \times 1.f \times 2^{e-bias}$, where s denotes the sign bit for a fraction, f denotes the fraction expressed in the form of absolute values, and e denotes the biased exponent. Note that $1.f$ in the above equation is called the significand.

The alignment stage shifts the significand of the smaller exponent to the right by the difference of the two exponents. The information corresponding to the data loss of the significand should be provided in the alignment stage for proper rounding operation based on the IEEE standard. For

0-7803-6470-8/00/ $10.00

the sake of rounding, three types of bits are defined: Guard bit G, Round bit R, and Sticky bit Sy[7]. The G becomes the MSB of bits that are lost, R becomes the second MSB, and the Sy is the boolean ORed value of the rest of the bits lost. Therefore, many boolean OR operations are required to calculate Sy-bit value.

IEEE standard 754 stipulates four rounding modes, which are Round to Nearest, Round to Zero, Round to Positive Infinity, and Round to Negative Infinity. These four rounding modes can be classified mainly into Round to Nearest, Round to Zero, and Round to Infinity because Round to Positive Infinity and Round to Negative Infinity can be divided into Round to Zero and Round to Infinity according to the sign of a number. From now on, the following equation denotes the result of rounding operation according to each of rounding modes. Return 0 means truncation, and return 1 means incrementation as the result of any rounding operation.

$$Round_{mode}(LSB, G, R, Sy) \quad (1)$$

Suppose that two significands A and B, which are absolute value types and are of length n bits, have values $A = a_{n-1}a_{n-2}...a_0$ and $B = b_{n-1}b_{n-2}...b_0$. At alignment stage, two significands are aligned by comparing two exponents. The significand of the *smaller* or the equal exponent is denoted as A, and significand of the *larger* or the equal exponent is represented as B. New aligned values for A and B are denoted by A^a and B^a respectively. Then, the result of alignment for A can be formed by shifting k bit positions, as the difference between two exponents. Furthermore, G, R, and Sy bits are generated in the alignment stage with respect to A, resulting in the aligned value $A^a = 0...0a_{n-1}a_{n-2}...a_kGRSy$, where $0 \le k \le n$. B is not changed after alignment and still represented as $B = b_{n-1}b_{n-2}...b_0$. The result of the addition of A^a and B^a is $F = A^a + B^a = C_F f_{n-1}f_{n-2}...f_0GRSy$, where C_F is the overflow bit of the result of $A^a + B^a$.

To simplify the notation, the binary point is to be located between LSB and G bit positions. Then, G, R, and Sy bit positions become a fraction portion and significand bits above them, which are the most significant n+1 bits, are an integer portion. The integer portion is represented by using the subscript I and the fractional portion by using the subscript T. Thus, most significant n+1 bits of F are the integer portion F_I, while G, R, and Sy of F are the fractional portion F_T.

From now on, '$\wedge$' denotes boolean AND, '$\vee$' denotes boolean OR, and '$\oplus$' denotes boolean exclusive-or. 'X' is used to denote the don't care state.

III. HARDWARE MODEL

A hardware model capable of performing rounding and addition/subtraction in parallel is shown in Fig. 1[11]. In Fig. 1, because the alignment and normalization stages of the presented floating point adder/subtractor are identical to those of the conventional one, they are omitted. Above hardware model is obtained from the algebraical analysis about performing rounding and addition/subtraction in parallel. In [11], detailed analysis and logical gate level implementation are discussed according to each rounding mode.

Exclusive-or unit either passes the input value to the output or acts as an inverter according to the input signal. The half adder produces the carry part and the sum part. An empty slot is reserved at the LSB of the carry part by the half adder. The predictor bit is inserted into this empty slot. The predictor bit is activated in the case of addition and Round to Infinity mode, but in other cases predictor should be zero. Adder0 and adder1, which are represented as the dotted box, are implemented by a single CSA (Carry Select Adder) and C_{out} is an overflow signal of adder0. The selector signal selects one of the result values, such as two inputs i0 and i1, after executing addition/subtraction and rounding. If selector = 0 then i0 is selected, and if selector = 1 then i1 is selected as the output value of the multiplexer. The selector signal in Fig. 1 is selected by the current rounding mode among the $selector_{mode}$ of each rounding mode.

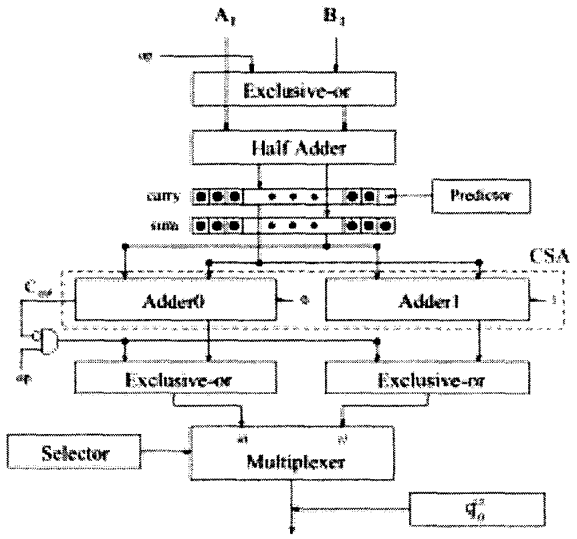

Fig. 1 Hardware model for performing IEEE rounding and addition/subtraction in parallel.

This floating point adder/subtractor does not need any additional hardware for rounding and renormalization, but it is constructed by using an additional n-bit half adder, a predictor, a selector, and a logic for q_0^{LS}. However, any high speed adder for rounding and any additional hardware for renormalization, which are both required in the conventional floating point adder/subtractor, may accompany much longer execution time and use a large amount of chip area than the additional hardware designed in this approach. Therefore, this floating point adder/subtractor provides effectiveness in the point of chip area and improved execution time. But, in the case of addition and Round to Infinity mode, the Sy-bit calculation is included in the critical path delay because Sy-bit is inserted into the predictor logic.

IV. NEW RARALLEL ROUNDING ALGORITHM

In this section, a method to remove the Sy-bit, which is generated at the alignment step, from the critical path is discussed. The same hardware model shown in Fig. 1 is used in this approach.

As shown in Fig. 1, $F = A^a + B^a = C_F f_{n-1} f_{n-2} \cdots f_0 GRSy$. In the case of an addition, the shifting operation for normalization is performed depending on the value of C_F. The shifting operation is not required in the case of $C_F = 0$, while shifting one bit to the right should be performed in the case of $C_F = 1$. The former case is denoted as NS (no shift) and the latter case is denoted as RS (right shift).

The rounding result without shifting for rounding position (NS or RS) is denoted as $Q = q_n q_{n-1} \cdots q_0$. In the case of NS, Q is represented by Q^{NS}. Then, Q^{NS} can be obtained as follows.

$$\begin{aligned} Q^{NS} &= (f_{n-1} f_{n-2} \cdots f_0) + Round_{mode}(f_0, G, R, Sy) \\ &= F_I + Round_{mode}(f_0, G, R, Sy) \end{aligned} \quad (2)$$

In the case of RS, Q is represented Q^{RS} by and Q^{RS} is as follows.

$$\begin{aligned} Q^{NS} &= ((C_F f_{n-1} \cdots f_1) + \\ &Round_{mode}(f_1, f_0, G, R \vee Sy)) \times 2 \\ &= F_I + 2 \times Round_{mode}(f_1, f_0, G, R \vee Sy) \end{aligned} \quad (3)$$

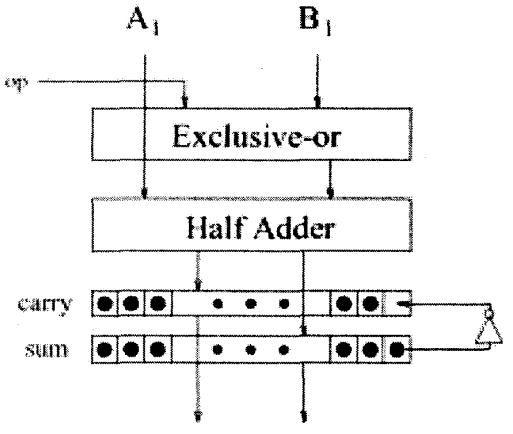

Fig. 2 The logic for predictor.

According to (2) and (3), one of three possible cases, i.e., F_I, $F_I + 1$, and $F_I + 2$, needs to be generated to perform rounding and addition in parallel. Therefore, if predictor and selector are properly selected, then the result value after performing addition and rounding in parallel, that is Q, can be generated.

In [11], because Sy-bit is inserted into predictor logic, a Sy-bit calculation is included in the critical path. To solve this problem, predictor logic must be organized independent of Sy-bit. Thus, in this approach, as shown in Fig. 2, predictor is assigned as the inverted value of the LSB of the sum part. The LSB of the sum part is denoted as sum_{LSB}. Then, four cases should be considered according to the value of sum_{LSB} and the condition of either NS or RS.

In the case of RS and $sum_{LSB} = 0$, predictor becomes one. And, because the two input values of i0 and i1, which are generated by carry select adder in Fig. 2, are F_I+predictor and F_I +predictor+1, they result in F_I +1 and F_I +2 respectively. Then, in the case of RS, most significant n bits are valid for final result. Because sum_{LSB} equals to zero and the most significant n-bits of $C_F f_{n-1} \cdots f_1 f_0$ +1 are equivalent to $C_F f_{n-1} \cdots f_1$+0, F_I+1 can be represented as $C_F f_{n-1} \cdots f_1$. And, because the most significant n-bits of $C_F f_{n-1} \cdots f_1 f_0$ +2 are equivalent to $C_F f_{n-1} \cdots f_1$+1, F_I+2 can be represented as $C_F f_{n-1} \cdots f_1$+1. Thus, in this case, if the rounding result is increment, then the result value after addition and rounding results in i1. Otherwise, i0 becomes the result value after addition and rounding. In the case of RS and $sum_{LSB} = 1$, predictor becomes zero. Then, the inputted values into i0 and i1 result in F_I+0 and F_I+1 respectively. In this case, F_I+0 equals to $C_F f_{n-1} \cdots f_1$+0. And, because sum_{LSB} equals to one, F_I +1 can be represented as $C_F f_{n-1} \cdots f_1$+1. Therefore, if the rounding result is increment then i1 should be selected, otherwise i0 is selected as a result value after addition and rounding operations. This condition equals to the above first case. Thus, considering above two RS cases, selector can be represented as $Round_{mode}(f_1, f_0, G, R \vee Sy)$ which is equivalent to the result of rounding operation in RS case.

According to (2), if the rounding result is truncation then F_I+0 should be selected, otherwise F_I+1 selected. In the case of sum_{LSB}=0, the input values into i0 and i1 result in F_I+1 and F_I+2 respectively. Thus, if the rounding result is increment, then i0 input value becomes a result value after addition and rounding operations. While the rounding result is truncation, F_I+0 should be selected. But, F_I+0 cannot be generated in this hardware model. Because sum_{LSB} equals to zero, most significant n-1 bits of F_I+0 and F_I+1 are equivalent to each other. Thus, when the rounding result is truncation, i0 should be selected as a result value after addition and rounding operations, and LSB value of the i0 should be changed into zero. Therefore, considering above

two NS cases, The selector signal becomes zero regardless of the result of rounding operation, in the case of sum_{LSB}=0. And, LSB value of the i0 must be zero when the rounding result is truncation. In the case of sum_{LSB}=1, input values into i0 and i1 result in F_l+0 and F_l+1 respectively. Therefore, if the rounding result is increment then i1 should be selected, otherwise i0 is selected as a result value after addition and rounding operations.

V. COMPARISON AND CONCLUSIONS

In this paper, a technique for removing Sy-bit from the critical path delay of [11] is provided. Its hardware model and correctness proofs are provided and evaluated. To compare the performance between conventional floating point adder and the proposed one, synthesis and simulation using COMPASS design automation tool are performed. Comparing with [11], proposed one can provide performance improvement about 15%. After careful full-custom layout, more speed gain may be achieved by 2%. Thus, removing Sy-bit from the critical path delay turns out to attain reasonable performance improvement from the simulation. Therefore, this floating point adder provides effectiveness in the points of chip area and its execution time.

V. ACKNOWLEDGMENTS

This work is supported by National Research Laboratory Projects from Ministry of Science & Technology of Republic of Korea.

REFERENCES

[1] T. Horel and G. Lauterbach, "UltraSPARC-III: Designing third-generation 64-bit performance," *IEEE Micro,* vol. 19, no. 3, pp.73-85, June 1999.

[2] K. C. Yeager, "The Mips R10000 superscalar microprocessor," *IEEE Micro*, vol. 16, no. 2, pp. 28-40, Apr. 1996.

[3] R. E. Kessler, "The Alpha 21264 Microprocessor," *IEEE Micro*, vol. 19, no. 2, pp. 24-36. Apr. 1999.

[4] S. P. Song, M. Denman and J. Chang, "The PowerPC604 RISC microprocessor," *IEEE Micro*, vol. 14, no. 5, pp. 8-17, Oct. 1994.

[5] M. C. Becker, M. S. Allen, C. R. Moore, J. S. Muhich, D. P. Tuttle, "The PowerPC 601 microprocessor," *IEEE Micro*, vol. 13, no. 5, pp. 54-68, Oct. 1993.

[6] L. Kohn and N. Margulis, "Introducing the Inter i840 64-bit microprocessor," *IEEE Micro*, vol. 9, no. 4, pp. 15-30, Aug. 1989.

[7] D. Goldberg, "Computer arithmetic," Appendix A of J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach, Morgan Kaufmann Publishers Inc, 1996.

[8] M. P. Farmwald, "*On the design of high performance digital arithmetic units*", PhD thesis, Stanford University, Aug. 1981.

[9] M. Birman, A. Samuels, G. Chu, T. Chuk, L. Hu, J. McLeod, and J. Barnes, "Developing the WTL3170/3171 sparc floating-point coprocessors," *IEEE Micro*, vol. 10, no. 1, pp. 55-64, Feb. 1990.

[10] B. J. Benschneider, W. J. Bowhill, E. M. Cooper, M. N. Gavrielov, P. E. Gronowski, V. K. Maheshwari, V. Peng, J. D. Pickholtz, and S. Samudrala, " A pipelined 50-Mhz CMOS 64-bit floating-point arithmetic processor," *IEEE Jounal of Solid-state Circuit*, vol. 24, no. 5, pp. 1317-1323, Oct. 1989.

[11] W. C. Park, S. W. Lee, O. Y. Kown, T. D. Han, and S. D. Kim, "Floating point adder/subtractor performing IEEE rounding and addition/subtraction in parallel," *IEICE Trans. Information and Systems*, vol. e79-d, no. 4, pp. 297-305, April 1996.

[12] IEEE Std 754-1985, "IEEE standard for binary floating-point arithmetic," *IEEE*, 1985.