PAPER

Floating-Point Multiplier with Concurrent Error Detection Capability by Partial Duplication

Nobutaka KITO[†a)], *Member*, Kazushi AKIMOTO[††], *Nonmember*, *and* Naofumi TAKAGI[††], *Senior Member*

SUMMARY A floating-point multiplier with concurrent error detection capability by partial duplication is proposed. It uses a truncated multiplier for checking of the significand (mantissa) multiplication instead of full duplication. The proposed multiplier can detect any erroneous output with error larger than one unit in the last place (1 ulp) of the significand, which may be overlooked by residue checking. Its circuit area is smaller than that of a fully duplicated one. Area overhead of a single-precision multiplier is about 78% and that of a double-precision one is about 65%.

key words: *concurrent error detection, floating-point multiplier, duplication, truncated multiplier*

1. Introduction

As aggressive scaling continues to push technology into smaller feature sizes, various design robustness concerns, such as faults by degradation [1]–[3] and soft errors due to particle hits [4], continue to arise. Circuits with concurrent error detection capability are crucial for development of reliable systems. In this paper, we propose a new floating-point multiplier which can detect any erroneous output with magnitude of error larger than one unit in the last place (1 ulp) of the significand (mantissa) by partial duplication. It works appropriately with any rounding mode defined in IEEE 754 floating-point standard [5].

For designing a reliable floating-point circuit, duplication and residue checking [6] have been used. Some processors which use full duplication for arithmetic units were reported [7], [8]. Though floating-point multipliers using full duplication achieve high capability of error detection, they have large area overhead. Reliable floating-point arithmetic units with residue checking were proposed [9] and selection of the modulus for residue checking in IBM processors was discussed in [10]. Though floating-point multipliers using residue checking can achieve high error detection ratio with smaller area overhead, they may overlook erroneous results with large magnitude of error, which is a multiple of the modulus. Furthermore, residue generation during residue checking is timing-critical because the residue needs to be generated after calculation of a result. Full duplication and residue checking are generic methods for basic arithmetic operations, and they utilize and consider little of the nature of floating-point arithmetic operation. Reliable floating-point arithmetic circuits utilizing methods other than full duplication and residue checking were proposed [11]–[14], but, to the best of our knowledge, are not actually used.

Recently, reduced precision checking is proposed for floating-point addition and floating-point multiplication in [12] and [13], respectively. The technique uses a small significand adder or a small significand multiplier for checking of the result. It mainly considers detection of very large errors, and does not treat rounding which is specific to floating-point arithmetic. Error detection ratio based on this technique reported in [13] is very low as the authors have described. The multiplier based on the method will be used for applications which need to detect only very large errors. The potential applications are different from ones of the multiplier to be proposed and ones of the traditional duplication or residue checking.

The floating-point multiplier to be proposed is based on the proven duplication method and utilizes the nature of floating-point arithmetic. The multiplier consists of a circuit for normal calculation and a circuit for checking as well as a checker which compares the outputs of the two circuits. The calculation sub-circuits of the sign bit and the exponent are duplicated. For the circuit for checking, a truncated multiplier is used for significand calculation instead of a full multiplier. The proposed floating-point multiplier is smaller than one using full duplication because a truncated multiplier sums up only about upper half of partial product bits. The area overhead of the proposed single-precision multiplier and that of the double-precision one are estimated at 78% and 65%, respectively. The floating-point multiplier detects any erroneous output with magnitude of error larger than 1 ulp, which may be overlooked by residue checking.

The proposed multiplier is suitable for applications which need reliability and tolerate small error. For examples, media processing, such as video and image processing, and calculation of a neuron's output in an neural network tolerate small error. Though the calculation of those applications tolerates small error, large error might lead to an incorrect result. Media processing and neural networks are used in various applications such as pedestrian detection from camera images [15], and recognition of the scene [16] for automotive safety or autonomous driving of cars. Reliability is crucial for automotive applications, and the proposed multiplier is useful for those applications.

This paper is organized as follows. In the next sec-

Manuscript received April 11, 2016.
Manuscript revised November 1, 2016.
Manuscript publicized December 19, 2016.

[†]The author is with the School of Engineering, Chukyo University, Toyota-shi, 470–0393 Japan.
[††]The authors are with the Graduate School of Informatics, Kyoto University, Kyoto-shi, 606–8501 Japan.
a) E-mail: nkito@sist.chukyo-u.ac.jp

DOI: 10.1587/transinf.2016EDP7158

tion, we describe floating-point multiplication and notations used in this paper. In Sect. 3, we propose a partially duplicated floating-point multiplier. In Sect. 4, we evaluate the proposed multiplier and compare it with designs using full duplication and ones using residue checking. In Sect. 5, we conclude this paper.

2. Floating-Point Multiplication

We consider IEEE 754 binary floating-point multiplication. In IEEE 754 standard, a binary floating-point number X consists of a sign bit x_s, an exponent X_e, and a fraction part of significand X_m. The bit width of the exponent is 8-bit for a single-precision number, and 11-bit for a double-precision number. The bit width of the fraction part of significand is 23-bit and 52-bit, respectively. We represent the bit width of the fraction part of significand by l. We consider multiplication of normalized numbers. A normalized floating-point number X represents $(-1)^{x_s} \times [1.X_m] \times 2^{X_e - bias}$. $bias$ is 127 for single-precision, and 1023 for double-precision.

We let the multiplicand, the multiplier, and the resultant product be X, Y, and Z, respectively. The sign bit and the exponent of Z are calculated as follows:

$$z_s = x_s \oplus y_s$$
$$Z_e = X_e + Y_e - bias + \delta$$

where δ is the increment from the significand calculation, and is either 1 or 0. Let the product of the significands $[1.X_m] \times [1.Y_m]$ be $U(= [u_{-1}u_0.u_1 \ldots u_{2l}])$. U is normalized as

$$V = \begin{cases} [1.u_0 \cdots u_{l-1}] & (\text{if } u_{-1} = 1) \\ [1.u_1 \cdots u_l] & (otherwise). \end{cases} \quad (1)$$

After the normalization, rounding is performed by adding rounding bit r to the least significant position of V, where r is either 1 or 0. We represent the result of rounding $V + r \times 2^{-l}$ as $W(= [w_{-1}w_0.w_1 \ldots w_l])$. The resultant fraction part of the significand is obtained as

$$Z_m = [.w_1 \cdots w_l].$$

The increment δ is 1 when w_{-1} or u_{-1} is 1. δ is 0, otherwise. Note that no special operation is necessary when W is $[10.0 \cdots 0]$ because all the bits other than w_{-1} are 0.

In IEEE 754 standard, four rounding modes, i.e., Round ties to even, Round toward positive, Round toward negative and Round toward zero, are defined for binary operations. For Round toward zero, rounding bit r is 0. For the other modes, it is obtained as follows.

- Round ties to even

$$r = \begin{cases} u_l \wedge (u_{l-1} \vee u_{l+1} \vee st) & (\text{if } u_{-1} = 1) \\ u_{l+1} \wedge (u_l \vee st) & (otherwise) \end{cases}$$

  Here, sticky bit st is $u_{l+2} \vee u_{l+3} \vee \cdots \vee u_{2l}$.
- Round toward positive and round toward negative

$$r = \begin{cases} pol \wedge (u_l \vee u_{l+1} \vee st) & (\text{if } u_{-1} = 1) \\ pol \wedge (u_{l+1} \vee st) & (otherwise) \end{cases}$$

  Here, pol is $\bar{z_s}$ for round toward positive or z_s for round toward negative.

3. Floating-Point Multiplier with Concurrent Error Detection Capability by Partial Duplication

We propose a floating-point multiplier which can detect any erroneous output with magnitude of error larger than 1 ulp.

3.1 Overall Structure

Figure 1 shows a block diagram of the proposed floating-point multiplier. The multiplier consists of three parts: the normal calculation circuit, the calculation circuit for checking, and the checker. In the figure, the left-half is for normal calculation and the right-half is for checking. We use duplication for the sub-circuit for the sign bit and the sub-circuit for the exponent because they are much smaller than the circuit for the significand. We use a special significand calculation circuit for checking instead of full duplication. The checker compares the normal result with the output from the circuit for checking.

3.2 Significand Calculation Circuit for Checking

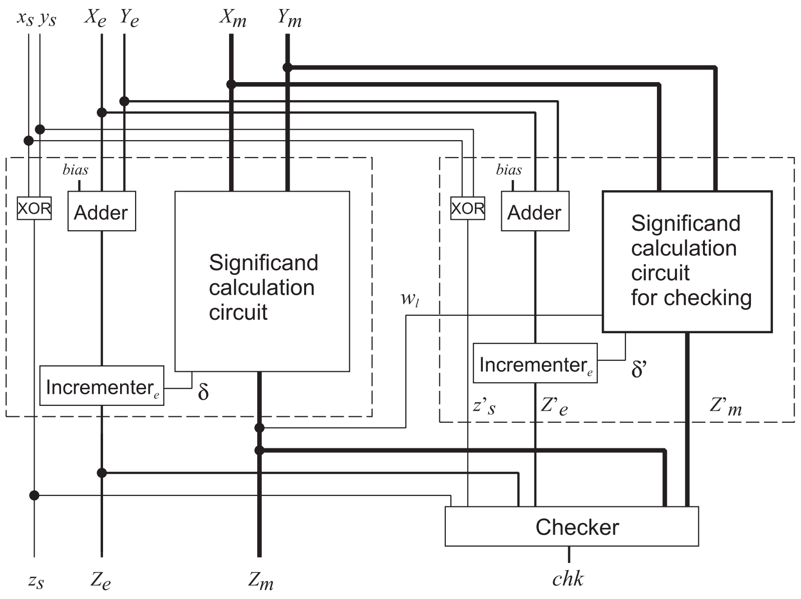
We focus on the significand calculation circuit for checking. For the significand calculation circuit for checking, we use a truncated multiplier. The significand calculation circuit for checking is smaller than the normal significand calculation circuit because a truncated multiplier sums up only upper about half of partial product bits. The output from the truncated multiplier is normalized, rounded and then, adjusted by comparing its least significant bit with that of the result from the normal significand calculation circuit.

Figure 2 shows a block diagram of the significand calculation circuit for checking. It receives X_m and Y_m, and the least significant bit of the normal significand result w_l. It outputs fraction part of significand Z'_m and increment δ' for checking. The significand calculation circuit consists of a truncated multiplier, a normalizer, a rounding bit generator, an incrementer, an adjuster, and an OR gate.

The truncated multiplier produces U' which satisfies the following inequalities.

$$U - 2^{-l} < U' \leq U \quad (2)$$

The normalizer normalizes U' and produces V' $(= [1.v'_1 \ldots v'_l])$ as formula (1). The rounding bit generator calculates r' as described in the previous section according to the rounding mode. Incrementer$_m$ sums up rounding bit r' and V', and produces W'. W' may be different from W. The adjuster generates the result $W''(= [w''_{-1}w''_0.w''_1 \ldots w''_l])$ from W' and the least significant bit of Z_m, i.e., w_l, as follows:



x_s y_s X_e Y_e X_m Y_m
bias
XOR
Adder
Significand calculation circuit
Incrementer$_e$
δ
w_l
Significand calculation circuit for checking
δ'
z'_s Z'_e Z'_m
Checker
z_s Z_e Z_m chk

Fig. 1 Floating-point multiplier with concurrent error detection capability by partial duplication

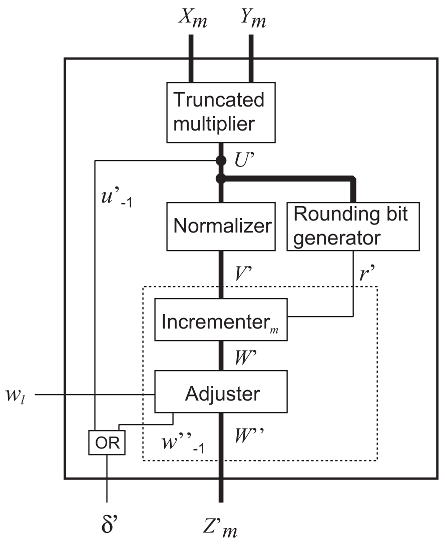


Fig. 2 Significand calculation circuit for checking

$$W'' = \begin{cases} W' & (\text{if } w_l = w'_l) \\ W' + 2^{-l} & (otherwise) \end{cases}$$

where w'_l is the least significant bit of W'. If no error exists, W'' is the same as W. The fraction part of significand Z'_m for checking is obtained as

$$Z'_m = [.w''_1 \cdots w''_l].$$

The significand calculation circuit sends increment $\delta'(= u'_{-1} \vee w''_{-1})$ to the exponent sub-circuit for checking. The normal result, i.e., z_s, Z_e, and Z_m, and the output from the circuit for checking, i.e., z'_s, Z'_e, and Z'_m, are compared by the checker.

Note that the adjuster works appropriately with any rounding mode. W is equal to either W' or $W' + 2^{-l}$ regardless of the rounding mode because of inequalities (2). Therefore, we can obtain the adjusted result by incrementing W' when the least significant bits of W and W' are not the same.

We use the least significant bit of Z_m, i.e., w_l, for adjusting the output of the truncated multiplier. The multiplier may not detect an erroneous significand output only when w_l is inverted by an error because the adjuster generates the same output to the incorrect Z_m. In other words, it detects any erroneous significand if it has an error larger than 1 ulp. Note that, when w_l is inverted by an error, the erroneous one is chosen for W''. In this case, the both inputs of the checker are erroneous. Even though the output of the normal significand calculation circuit has error of 1 ulp, the checker detects the error if the polarity of the two 1 ulp errors is different, i.e., one of the checker inputs is 1 ulp larger and the other is 1 ulp smaller than the correct result. Therefore, half cases of 1 ulp error of the final result are detected by the checker.

3.3 A Design Consideration of the Significand Calculation Circuit for Checking

We consider design of the significand calculation circuit for checking in detail.

We show an example of the part of partial product bits for a truncated multiplier for single-precision in Fig. 3. A dot represents a partial product bit. Every output value of a truncated multiplier for single-precision, which sums up the dots surrounded by the broken lines, satisfies inequalities (2) where $l = 23$. The truncated multiplier for double-precision can be designed in a similar way. Because the truncated multiplier sums up fewer bits than the normal one, it performs calculation in shorter time and occupies smaller circuit area. Note that, the delay of the overall circuit is longer than the normal significand calculation circuit because the checker output is obtained after two outputs.

We can shorten the critical path of the significand calculation circuit for checking by designing the part of the circuit surrounded by the broken lines in Fig. 2 cleverly. This part calculates W'' according to the values of r', w_l, and v'_l as follows:

$$W'' = \begin{cases} V' & (\text{if } r' = 0 \wedge w_l = v'_l) \\ V' + 2^{-l+1} & (\text{if } r' = 1 \wedge w_l = v'_l) \\ V' + 2^{-l} & (otherwise). \end{cases} \quad (3)$$

When $V' + 2^{-l+1}$ is represented as $T' = [t'_{-1}t'_0.t'_1 \cdots t'_l]$, $V' + 2^{-l}$ can be obtained from V' or T' as follows:

$$V' + 2^{-l} = \begin{cases} [t'_{-1}t'_0.t'_1 \cdots t'_{l-1}\ 0] & (\text{if } v'_l = 1) \\ [1.v'_1 \cdots v'_{l-1}\ 1] & (otherwise). \end{cases} \quad (4)$$

From (3) and (4), we can obtain W'' as follows:

$$W'' = \begin{cases} [t'_{-1}t'_0.t'_1t'_2 \cdots t'_{l-1}w_l] & (\text{if } sel_{Z'} = 1) \\ [01.v'_1v'_2 \cdots v'_{l-1}w_l] & (otherwise) \end{cases}$$

where $sel_{Z'} = (\overline{w_l} \wedge v'_l) \vee (v'_l \wedge r') \vee (r' \wedge \overline{w_l})$. Therefore, we can design this part as shown in Fig. 4. The design includes only one incrementer, Incrementer'$_m$, that calculates $V' + 2^{-l+1}$. Note that a naive design includes two incrementers. Though signal w_l is fed from the output of the normal circuit, influence of this part to the circuit delay of the overall circuit is small because output delay of an integer multiplier and an incrementer is not uniform at all positions, and the value of w_l is obtained earlier than the value of the latest delayed position generally.

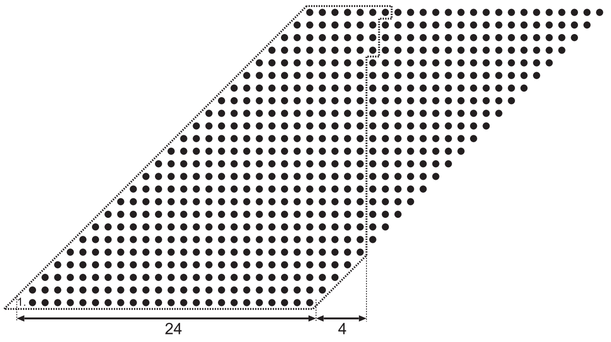


Fig. 3 Example of partial product bits for a truncated multiplier in a single-precision multiplier

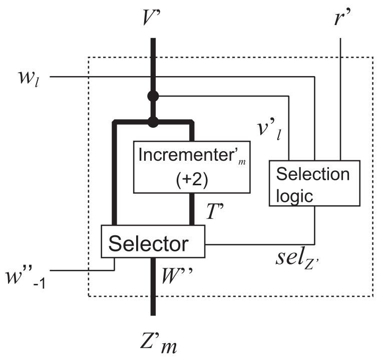


Fig. 4 Alternative design of Incrementer$_m$ and Adjuster for the significand calculation circuit for checking

4. Evaluation

We estimated the circuit area, and evaluated the error detection ratio and several metrics about error detection of the proposed floating-point multiplier. For comparison, we considered three other multipliers: a fully duplicated multiplier and two multipliers using residue checking based on [9]. For the circuits using residue checking, we chose 3 or 15 as the modulus. Modulo-3 checking is used widely [10], [17], and modulo-15 checking is also used in IBM z196 microprocessor [10], [18]. We used duplication for checking of the sign bit and the exponent for all the four multipliers to evaluate error detectability and area of the significand calculation circuits fairly. We used the design in Sect. 3.3 for the proposed multiplier. We used a two-rail checker as the checker.

We used Synopsys design compiler with NanGate

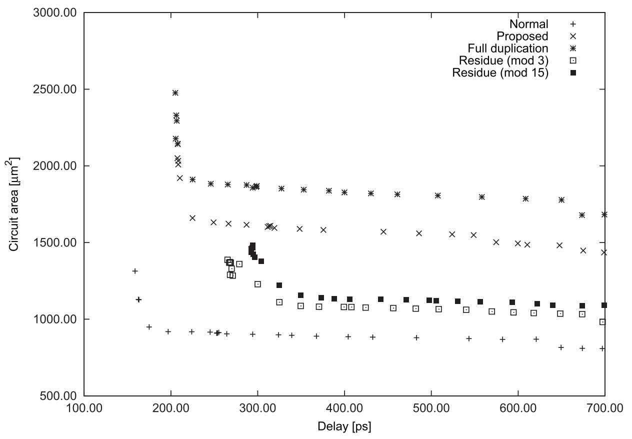


Fig. 5 Circuit area and delay time of single-precision floating-point multiplier designs

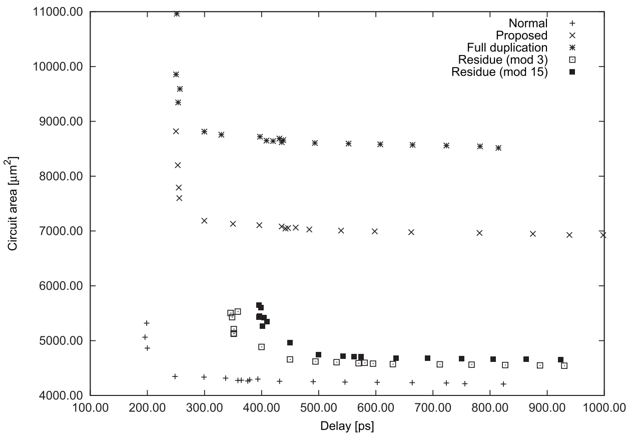


Fig. 6 Circuit area and delay time of double-precision floating-point multiplier designs

Table 1 Area overhead of floating-point multipliers

	Single-precision (delay: 300 ps)		Double-precision (delay: 400 ps)	
	Area [μm^2]	Overhead [%]	Area [μm^2]	Overhead [%]
Normal	910.3	-	4298.4	-
Proposed	1615.5	77.5	7106.2	65.3
Full duplication	1875.1	105.9	8719.1	102.8
Residue (mod 3)	1228.8	35.0	4885.7	13.7
Residue (mod 15)	1379.1	51.5	5344.6	24.3

Table 2 Evaluation results of error detections

	Error detection ratio [%]	Average magnitude of overlooked error [ulp]	Ratio of errors in exponent bits [%]	FP ratio [%]
Proposed	91.2	1 (exact)	0.0 (no error)	55.1
Full duplication	100.0 (no error)	0 (no error)	-	54.1
Residue (mod 3)	98.7	119545	9.1	52.7
Residue (mod 15)	99.2	5275	28.0	57.5

15nm open cell library [19] to synthesize the designs. We synthesized the designs with various timing constraints. Figures 5 and 6 show circuit area of single-precision designs and double-precision ones, respectively. Horizontal axis denotes delay time of the designs and vertical axis denotes circuit area. In the figures, “Normal” denotes the multiplier without error detection capability, “Proposed” denotes the proposed multiplier, “Full duplication” denotes the multiplier using full duplication, and “Residue (mod 3)” and “Residue (mod 15)” denote the multipliers using residue checking. The figures show that the proposed multiplier has shorter delay time compared with the multipliers using residue checking. Residue checking is timing-critical because residue generation is necessary after calculation of the normal result and the output for checking. In the figures, the shortest delay of the proposed multiplier is almost the same as that of the full duplication. The adjuster parts of the proposed multiplier have small influence to the circuit delay.

The circuit area and the area overhead are shown in Table 1. We show the estimation of single-precision designs synthesized with 300 ps delay constraint and double-precision designs synthesized with 400 ps delay constraint. The area overhead of the proposed multiplier is about 78% for the single-precision design and about 65% for the double-precision design. The area overhead is smaller compared with that of the multiplier using full duplication.

We evaluated the error detection ratio and other metrics about error detection of the four multipliers. We considered error caused by a single stuck-at fault for evaluation. We used Synopsys TetraMax for simulation of the multipliers inserting a stuck-at fault. We carried out simulation for single-precision multipliers synthesized with 300 ps delay constraint with 50,000 random patterns for every single stuck-at fault. The results are shown in Table 2.

The fault detection ratio of the proposed multiplier is over 90%. The magnitude of any overlooked error of the proposed multiplier is exactly 1 ulp. It is very small against the average magnitude of overlooked error of the multipliers using residue checking.

We show the ratio of erroneous outputs whose exponent bits are erroneous in all overlooked ones. The ratio for the proposed multiplier is 0%. The multiplier has two increment signals δ and δ' for the two exponent sub-circuits and erroneous outputs caused by erroneous incrementation on one of the two signals can be detected. On the other hand, the ratios for the multipliers using residue checking are over 9%.

We also evaluated the fault positive ratio (FP ratio) which is the ratio of correct outputs when the checker indicates error. Results of the four multipliers are almost the same.

5. Conclusion

We have proposed a floating-point multiplier with concurrent error detection capability by partial duplication. It detects any erroneous output with magnitude of error larger than 1 ulp which may be overlooked by residue checking. It works appropriately with any rounding mode defined in IEEE 754 floating-point standard. Area overhead of the proposed multiplier is small compared with the multiplier using full duplication.

The proposed multiplier utilizes the idea of tolerating overlooked small errors, considering the nature of floating-point multiplication. Further researches about trade-off between area overhead and magnitude of detectable error will be interesting.

Acknowledgements

This work was supported in part by JSPS KAKENHI Grant Numbers: 24300019 and 25730033. It was also supported by VLSI Design and Education Center (VDEC), the University of Tokyo with the collaboration with Synopsys Corporation.

References

[1] J.H. Stathis, “Reliability limits for the gate insulator in CMOS technology,” IBM Journal of Research and Development, vol.46, no.2-3, pp.265–286, 2002.

[2] J. Srinivasan, S.V. Adve, P. Bose, and J.A. Rivers, “The impact of technology scaling on lifetime reliability,” Proc. International

Conference on Dependable Systems and Networks (DSN 2004), pp.177–186, June 2004.

[3] D.K. Schroder and J.A. Babcock, "Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing," Journal of Applied Physics, vol.94, no.1, pp.1–18, 2003.

[4] R. Baumann, "Soft errors in advanced computer systems," IEEE Design & Test of Computers, vol.22, no.3, pp.258–266, May 2005.

[5] "IEEE standard for floating-point arithmetic," IEEE Std 754-2008, Aug. 2008.

[6] W.W. Peterson, "On checking an adder," IBM Journal of Research and Development, vol.2, no.2, pp.166–168, April 1958.

[7] T.J. Slegel, R.M. Averill III, M.A. Check, B.C. Giamei, B.W. Krumm, C.A. Krygowski, W.H. Li, J.S. Liptay, J.D. MacDougall, T.J. McPherson, J.A. Navarro, E.M. Schwarz, K. Shum, and C.F. Webb, "IBM's S/390 G5 microprocessor design," IEEE Micro, vol.19, no.2, pp.12–23, March 1999.

[8] T.J. Slegel, E. Pfeffer, and J.A. Magee, "The IBM eServer z990 microprocessor," IBM Journal of Research and Development, vol.48, no.3-4, pp.295–309, May 2004.

[9] J.-C. Lo, "Reliable floating-point arithmetic algorithms for error-coded operands," IEEE Trans. Comput., vol.43, no.4, pp.400–412, April 1994.

[10] D. Lipetz and E. Schwarz, "Self checking in current floating-point units," Proc. 20th IEEE Symposium on Computer Arithmetic, pp.73–76, July 2011.

[11] W.L. Gallagher and E.E. Swartzlander, "Fault-tolerant Newton-Raphson and Goldschmidt dividers using time shared TMR," IEEE Trans. Comput., vol.49, no.6, pp.588–595, June 2000.

[12] P.J. Eibl, A.D. Cook, and D.J. Sorin, "Reduced precision checking for a floating point adder," Proc. 24th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp.145–152, Oct. 2009.

[13] K. Seetharam, L.C.T. Keh, R. Nathan, and D.J. Sorin, "Applying reduced precision arithmetic to detect errors in floating point multiplication," Proc. 19th Pacific Rim International Symposium on Dependable Computing (PRDC), pp.232–235, Dec. 2013.

[14] M. Maniatakos, P. Kudva, B.M. Fleischer, and Y. Makris, "Low-cost concurrent error detection for floating-point unit (FPU) controllers," IEEE Trans. Comput., vol.62, no.7, pp.1376–1388, July 2013.

[15] W. Ouyang and X. Wang, "Joint deep learning for pedestrian detection," Proc. 2013 IEEE International Conference on Computer Vision (ICCV 2013), pp.2056–2063, Dec. 2013.

[16] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "DeepDriving: Learning affordance for direct perception in autonomous driving," Proc. 2015 IEEE International Conference on Computer Vision (ICCV 2015), pp.2722–2730, Dec. 2015.

[17] S. Mukherjee, Architecture design for soft errors, Morgan Kaufmann, 2011.

[18] B.W. Curran, L.E. Eisen, E.M. Schwarz, P.-K. Mak, J. Warnock, P.J. Meaney, and M. Fee, "The zEnterprise 196 system and microprocessor," IEEE Micro, vol.31, no.2, pp.26–40, March 2011.

[19] M. Martins, J.M. Matos, R.P. Ribas, A. Reis, G. Schlinker, L. Rech, and J. Michelsen, "Open cell library in 15nm FreePDK technology," Proc. 2015 International Symposium on Physical Design, pp.171–178, 2015.

Nobutaka Kito received the B.E. degree in information science, the master of information science degree, and the Dr. of information science degree from Nagoya University, Nagoya, Japan, in 2004, 2006, and 2009, respectively. He was a research associate since 2010 at Kyoto University, Kyoto, Japan. He moved to Chukyo University, Toyota, Japan, as an assistant professor in 2012. His current interests include computer arithmetic, dependable computing, and design automation of superconducting logic circuits.

Kazushi Akimoto received the B.E. degree in information science and the master of informatics degree from Kyoto University, Kyoto, Japan, in 2012 and 2014, respectively.

Naofumi Takagi received the B.E., M.E., and Ph.D. degrees in information science from Kyoto University, Kyoto, Japan, in 1981, 1983, and 1988, respectively. He joined Kyoto University as an instructor in 1984 and was promoted to an associate professor in 1991. He moved to Nagoya University, Nagoya, Japan, in 1994, and promoted to a professor in 1998. He returned to Kyoto University in 2010. His current interests include computer arithmetic, hardware algorithms, and logic design. He received Japan IBM Science Award and Sakai Memorial Award of the Information Processing Society of Japan in 1995, and The Commendation for Science and Technology by the Minister of Education, Culture, Sports, Science and Technology of Japan in 2005.