IEICE TRANS. INF. & SYST., VOL.E96–D, NO.9 SEPTEMBER 2013

PAPER Special Section on Dependable Computing

Low-Overhead Fault-Secure Parallel Prefix Adder by Carry-Bit Duplication

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

SUMMARY We propose a low-overhead fault-secure parallel prefix adder. We duplicate carry bits for checking purposes. Only one half of normal carry bits are compared with the corresponding redundant carry bits, and the hardware overhead of the adder is low. For concurrent error detection, we also predict the parity of the result. The adder uses parity-based error detection and it has high compatibility with systems that have parity-based error detection. We can implement various fault-secure parallel prefix adders such as Sklansky adder, Brent-Kung adder, Han-Carlson adder, and Kogge-Stone adder. The area overhead of the proposed adder is about 15% lower than that of a previously proposed adder that compares all the carry bits.

key words: *parity prediction, parallel prefix adder, fault secure, carry-bit duplication*

1. Introduction

As manufacturing technology of integrated circuits progresses by reducing feature size and providing more integration density, reliability problems have been an issue. Designs for reliability have become increasingly important. Concurrent error detection is crucial for enhancing reliability. For example, in microprocessors for enterprise servers, concurrent error detection techniques have been used [1], [2].

Adders are important components in VLSI chips. Many adder structures have been proposed to meet various requirements, i.e., delay time and area. Among them, parallel prefix adders are widely used [3]. A parallel prefix adder calculates "propagate signals" and "generate signals" in parallel, and these signals are used to calculate the carry bit for each bit position. Various prefix calculation circuits having different delay times and areas can be designed to generate these signals.

In this paper, we propose a low-overhead fault-secure parallel prefix adder. A circuit is said to be fault-secure, if errors are detectable by observing the circuit output whenever it outputs erroneous values. We can implement fault-secure parallel prefix adders using various prefix calculation circuits, such as Sklansky adder [4], Brent-Kung adder [5], Han-Carlson adder [6], and Kogge-Stone adder [7]. The adder is fed with two operands and their parity bits, and produces the sum as well as the predicted parity of the sum and a pair of check bits. We assume that the adder consists of circuit blocks referred to as functional cells. We intend to detect errors caused by a single faulty cell at the output of the adder.

For concurrent error detection of the adder, we duplicate carry bits, i.e., we introduce redundant carry bits for checking purposes, and one half of normal carry bits are compared with the corresponding redundant carry bits using a two-rail checker. We also predict the parity of the result (the sum). Because the bit width of the checker is smaller, the overhead of the adder is lower than that of the previously proposed fault-secure adders. We detect errors by comparing the predicted parity with the actual parity of the adder result and observing the check bit pair from the checker. The adder is suitable for systems with parity-based error detection. In such a system, we can easily feed two operands and their parities to the adder, and can use the predicted parity as the parity bit.

Several concurrent error detectable adders using parity bits have been proposed [8]–[11]. In [8], a fault-secure ripple carry adder is shown. Although a ripple carry adder is basic and simple, it is too slow for many applications. In [9] and [10], a fault-secure and self-testing carry lookahead adder and a fault-secure and self-testing carry skip adder are shown, respectively. A circuit is said to be self-testing, if, for any fault, at least one input pattern exists to detect the fault. Additional hardware for these adders is shown at the gate level. These adders are less flexible when implementing designs, unlike the adder proposed in this paper, and have a large hardware overhead because of the large hardware required for sum-bit duplication and a larger checker. In [11], a fault-secure adder using carry-bit duplication is shown. In this adder, all the carry bits are compared with the corresponding redundant carry bits for checking. In the previously proposed adders using carry-bit (or sum-bit) duplication [9]–[11], all the carry bits (or sum bits) are compared with the corresponding redundant bits for checking. On the other hand, in the adder proposed in this paper, only one half of the carry bits are compared. Thus, the proposed adder has a smaller wiring area and lower hardware overhead. Evaluation results show that the hardware overhead for a 32-bit design of the proposed adder is about 15% lower than that for a design of the adder proposed in [11].

This paper is organized as follows. In the next section, we describe a parallel prefix adder, the fault model, and a two-rail checker. In Sect. 3, we propose a low-overhead fault-secure parallel prefix adder, and discuss its error de-

Manuscript received January 18, 2013.
Manuscript revised April 27, 2013.
[†]The author is with the School of Engineering, Chukyo University, Toyota-shi, 470–0393 Japan.
[††]The author is 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.E96.D.1962

tection mechanism. In Sect. 4, we evaluate a design of the proposed adder, and in Sect. 5, we conclude this paper.

2. Preliminaries

2.1 Parallel Prefix Adder

We consider an unsigned n-bit parallel prefix adder. We let the augend X be $[x_{n-1} \cdots x_0]$, the addend Y be $[y_{n-1} \cdots y_0]$, the carry input to the least significant position be c_0, and the sum S be $[c_n s_{n-1} \cdots s_0]$.

In a parallel prefix adder, the carry bit c_{l+1} of each bit position l ($n > l \geq 0$) is calculated regarding the "generate signal" and the "propagate signal" spanning bit positions from 0 to l. We calculate the generate signal g_i and the propagate signal p_i of bit position i as follows:

$$g_i = x_i \cdot y_i$$
$$p_i = x_i \oplus y_i.$$

With the above signals, we can calculate the generate signal and the propagate signal spanning bit positions from k to i ($n > i \geq j > k \geq 0$) as follows:

$$G_{i,k} = G_{i,j} + P_{i,j} \cdot G_{j-1,k}$$
$$P_{i,k} = P_{i,j} \cdot P_{j-1,k}.$$

We treat $G_{i,i} = g_i$ and $P_{i,i} = p_i$.

The carry bit c_{l+1} and the sum bit s_l can be calculated as follows:

$$c_{l+1} = G_{l,0} + P_{l,0} \cdot c_0$$
$$s_l = c_l \oplus p_i.$$

As an example of a parallel prefix adder, we show a 16-bit Sklansky adder in Fig. 1. We also show the prefix calculation circuits of Brent-Kung adder, Han-Carlson adder, and Kogge-Stone adder in Figs. 2 (a), (b), and (c), respectively. We use several circuit blocks, hereafter referred to as cells, to construct a parallel prefix adder. We use five types of cells, whose functions are shown in Table 1. A GP cell calculates a generate signal and a propagate signal from four inputs. We let the GP cell calculating $G_{i,k}$ and $P_{i,k}$ be GP(i,k).

In this paper, we assume that a parallel prefix adder holds the following properties for any GP cell.

- Inputs of GP(i,k) come from GP(i,j) and GP($j-1$,k) ($i \geq j > k$), i.e., there is no overlap in bit positions between the both source GP cells.
- $dm(i, j)$ is equal to n, or at least one of $dm(i, j)$ and i is odd. Here, $dm(i, j)$ denotes the maximum d such that all carry bits c_h ($d \geq h > i$) depend on the outputs of GP(i,j).

In general, parallel prefix adders without redundant GP cells satisfy the first property. The prefix calculation circuits shown in Figs. 1 and 2 satisfy the above two properties.

For example, in a prefix calculation circuit of the Sklansky adder shown in Fig. 1, $dm(i, j)$ of GP(i,j) cell is either n or $i+1$. For example, $dm(6, 4)$ is 7 ($= 6+1$) because the carry bit c_7 depends on the outputs of GP(6,4) while the

Table 1 Cells for a parallel prefix adder.

Cell	Function
g	$g = x \cdot y$
p	$p = x \oplus y$
GP	$g = G_h + P_h \cdot G_l$ $p = P_h \cdot P_l$
C	$c = G + P \cdot c_{in}$
S	$s = p \oplus c$

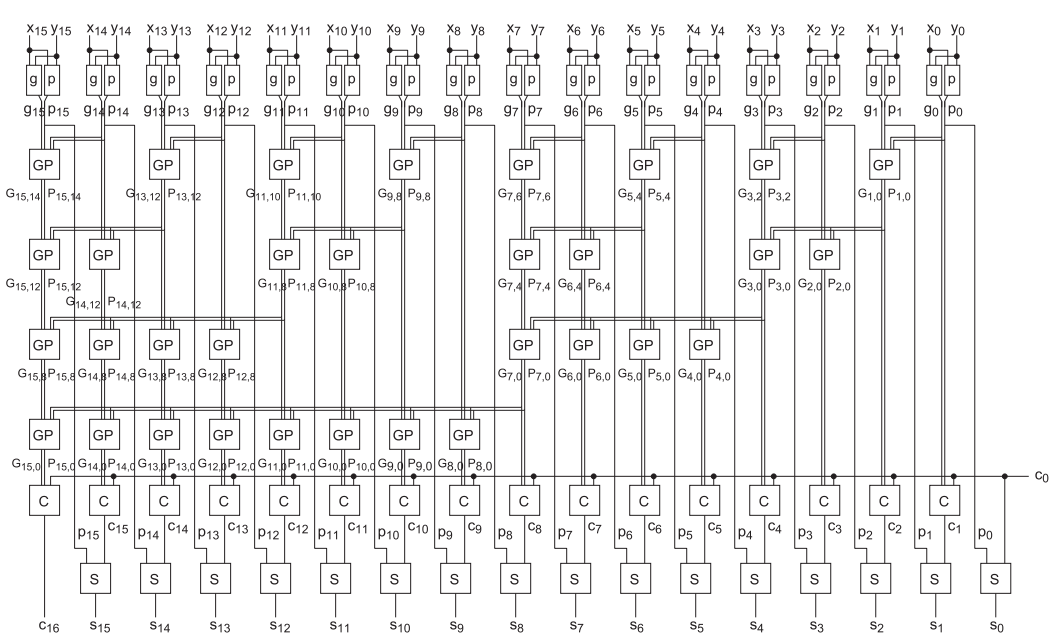

Fig. 1 Example of a parallel prefix adder (a 16-bit Sklansky adder).

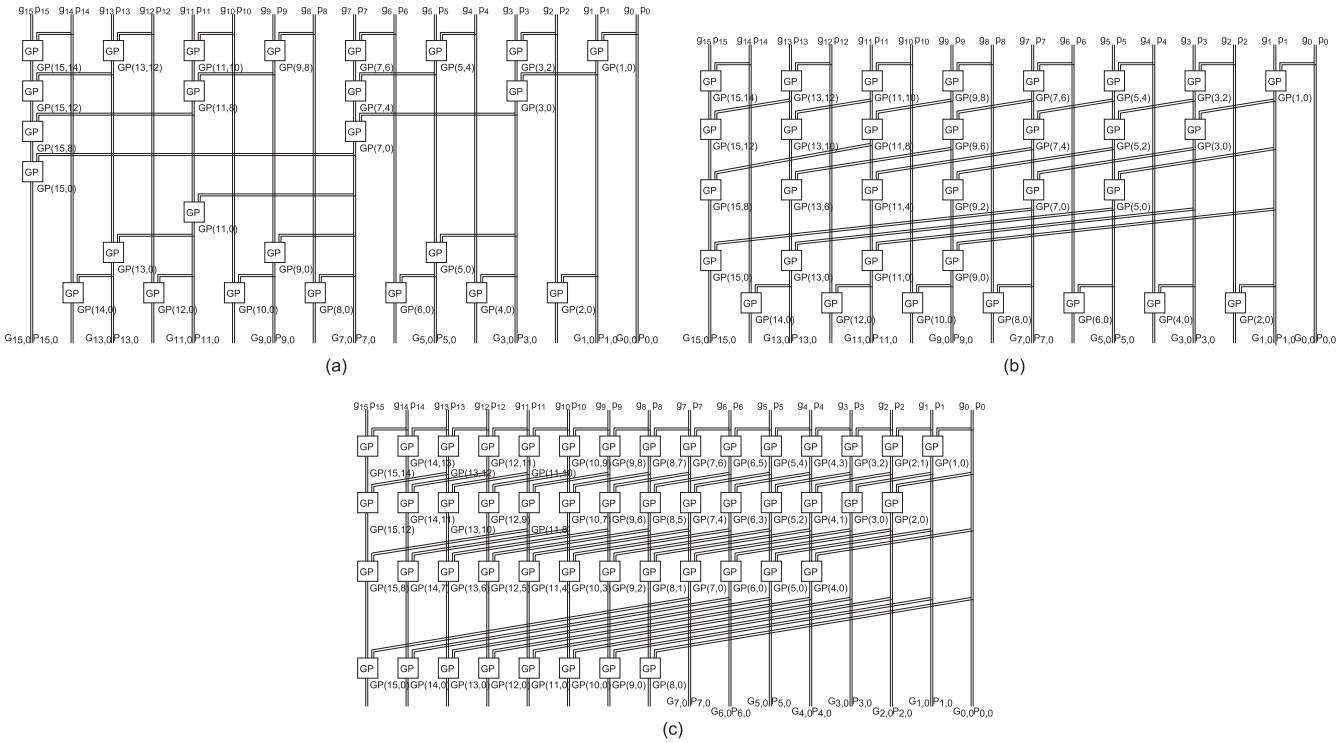

Fig. 2 Prefix calculation circuits in 16-bit parallel prefix adders. (a) Brent-Kung adder, (b) Han-Carlson adder, (c) Kogge-Stone adder.

carry bit c_8 does not depend on them. $dm(7,6)$ is 16 ($= n$) because there are paths from GP(7,6) to the carry bits from c_8 to c_{16}. When $dm(i, j)$ is $i + 1$, either $dm(i, j)$ or i is odd. Therefore, all GP cells satisfy the properties.

We also show another example. In the prefix calculation circuit of the Brent-Kung adder shown in Fig. 2 (a), we can classify GP cells into two groups. One group contains GP(i_o,j) and the other contains GP(i_e,j), where i_o and i_e denote odd and even-numbers, respectively. GP(i_o,j) satisfies the second property because i_o is odd. GP(i_e,j) also satisfies the property because $dm(i_e, j)$ is equal to $i_e + 1$. Therefore, all GP cells satisfy the properties.

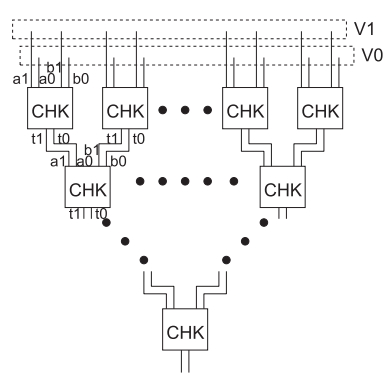

Fig. 3 Two-rail checker.

2.2 Fault Model

In this paper, we assume that the considered circuit consists of functional circuit blocks, called cells. In the circuit, at most one cell is faulty. We consider that the logical function of the faulty cell is different from the expected function. A faulty cell with multiple output lines may output erroneous logic values at the multiple output lines. For example, when a GP cell is faulty, we consider that its two outputs may be erroneous simultaneously. When the traditional single stuck-at fault model is adopted, this fault is difficult to handle or can not be handled because the single stuck-at fault model consider a fault of a single signal.

Because we do not assume specific gate-level implementations of cells, any gate-level implementation can be used. Thus, for example, an S cell that calculates $p \oplus c$ can be implemented by an XOR gate or by a combination of gates. Therefore, various gate-level implementations exist for an adder that consists of cells, and implementation flexibility of circuits that consist of cells is higher than that of the gate-level circuit considering the traditional single stuck-at fault model. Moreover, multiple faults in a gate-level implementation of a cell can be treated as a single cell fault.

2.3 Two-Rail Checker

We use the two-rail checker shown in Fig. 3 for the parallel prefix adder proposed in this paper. The two-rail checker takes two bit vectors $V1$ and $V0$ as inputs. $V1$ is expected to be the bitwise logical inverse of $V0$.

The two-rail checker consists of CHK cells. A CHK cell has four input lines ($a1$, $a0$, $b1$ and $b0$) and calculates

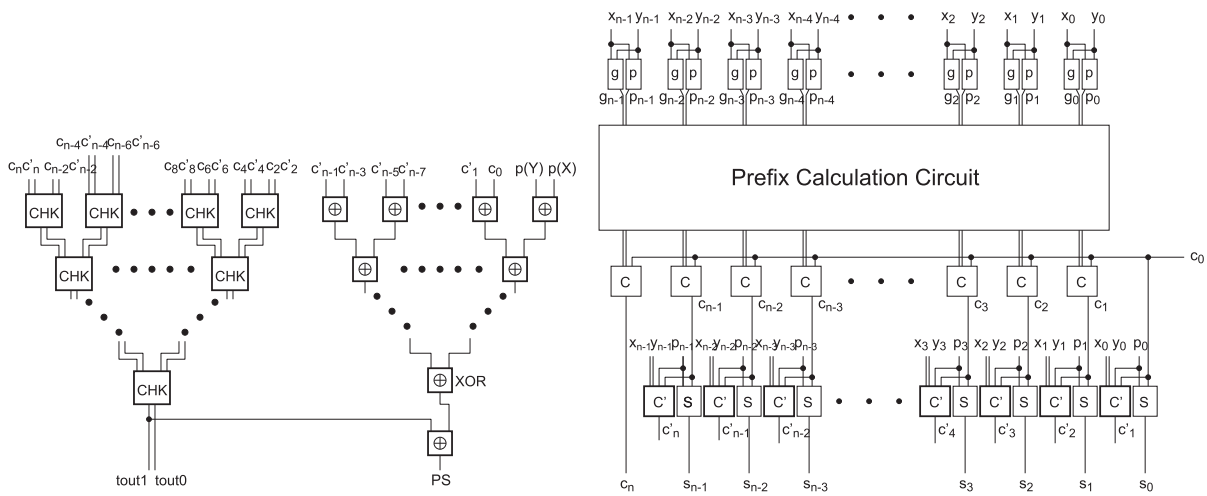

Fig. 4 Carry-bit duplicated fault-secure parallel prefix adder.

two outputs ($t1$ and $t0$) as follows:

$$t1 = a1 \cdot b0 + a0 \cdot b1$$
$$t0 = a1 \cdot b1 + a0 \cdot b0.$$

The outputs $t1$ and $t0$ have the same value if the input value for $a1$ is equal to that for $a0$, or if the input value for $b1$ is equal to that for $b0$. When input values for $a1$ and $b1$ are the logical inverses of those for $a0$ and $b0$, respectively, the output values of $t1$ and $t0$ are equal to $a1 \oplus b1$ and $\overline{a1 \oplus b1}$, respectively.

In a two-rail checker, we connect $t1$ and $t0$ of the CHK cells to $a1$ and $a0$ or $b1$ and $b0$, respectively. We feed the bits of $V1$ to $a1$ and $b1$ of the leaf CHK cells and the corresponding bits of $V0$ to $a0$ and $b0$ of the leaf CHK cells. Normally, $t1$ and $t0$ of the root CHK cell output the parity of bits in $V1$ and the logical inverse of the parity, respectively. When $V1$ is not the bitwise logical inverse of $V0$, two outputs of the root CHK cell have the same logical value. Therefore, this checker is code-disjoint, i.e., we can always detect errors when unexpected input values are fed to the circuit.

3. Low-Overhead Fault-Secure Parallel Prefix Adder

We propose a low-overhead fault-secure parallel prefix adder. We assume that, in addition to the augend X and the addend Y, parity bits $p(X)$ and $p(Y)$ that correspond to X and Y, respectively, are fed to the adder. The outputs of the adder are the sum S, the predicted parity PS, and a pair of check bits $tout0$ and $tout1$. In the adder, we introduce redundant carry bits for checking purposes (hereafter referred to as check carry bits) and compare only one half of carry bits with the corresponding check carry bits. We also predict the parity of the result for error detection purposes.

In this section, we assume that if not otherwise specified, the input parities $p(X)$ and $p(X)$ are consistent with the operands X and Y, respectively.

Table 2 Cells for a fault-secure parallel prefix adder.

Cell	Function
C'	$c' = \overline{x \cdot y + p \cdot c_{in}}$
XOR	$p = p1 \oplus p2$

3.1 Structure

The proposed low-overhead fault-secure parallel prefix adder is shown in Fig. 4. In the figure, the structure for a $4w$-bit adder ($w > 0$) is shown. The cells in bold lines are the additional ones for the fault-secure adder. We introduce two types of cells (C' and XOR) shown in Table 2. We need no additional cell for calculating the sum bits. Therefore, the adder has a low delay overhead for calculating the sum bits.

In a parallel prefix adder, the value of carry bit c_{i+1} is equal to $g_i + c_i \cdot p_i$. In the proposed adder, with this relation, we duplicate the carry bit for each bit position except the least significant position. We name the check carry bit c'_i. c'_i has the inverse value of the carry bit c_i. The check carry bit c'_{i+1} is calculated using the carry bit c_i according to $\overline{x_i \cdot y_i + c_i \cdot p_i}$. We use a C' cell to calculate the check carry bit, whose function is shown in Table 2.

In an adder, the following relations hold among X, Y, $c_n, c_{n-1}, \ldots, c_0$, and $s_{n-1}, \ldots, s_0$

$$c_n \oplus s_{n-1} \oplus s_{n-2} \oplus \cdots \oplus s_0 = (x_{n-1} \oplus \cdots \oplus x_0) \oplus (y_{n-1} \oplus \cdots \oplus y_0) \oplus (c_n \oplus \cdots \oplus c_0)$$

Thus, we can predict the parity of the sum using $p(X)$, $p(Y)$, and the parity of carry bits $p(C)$ $(= c_n \oplus \cdots \oplus c_0)$. The predicted parity PS is calculated according to $p(X) \oplus p(Y) \oplus p(C)$ in the proposed adder. Note that, by using only the predicted parity, we can not detect erroneous values caused by even number of carry-bit inversions. With the checker described below, we can obtain fault-secure property.

We use a two-rail checker as shown in Fig. 4. We treat

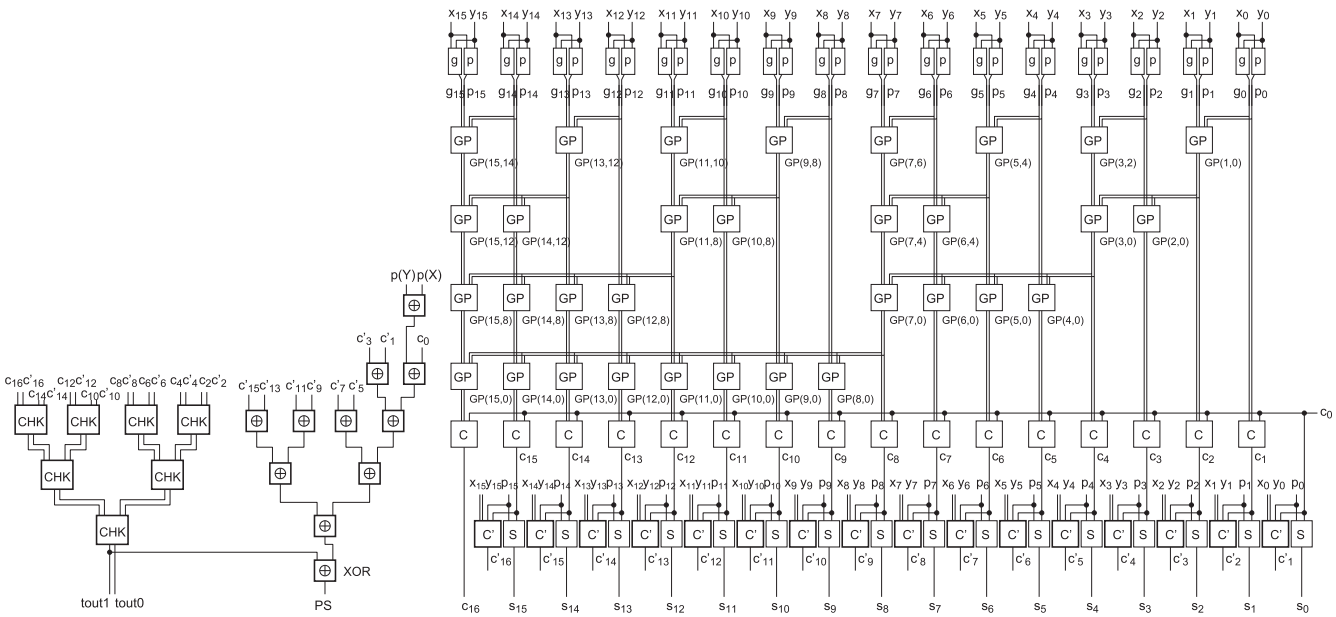

Fig. 5 Example of a fault-secure parallel prefix adder (a 16-bit Sklansky adder).

the original carry bits at even-numbered positions c_{i_e} and check carry bits at even-numbered positions c'_{i_e} $(n \geq i_e > 0)$ as input vectors $V1$ and $V0$ of the checker, respectively. We let the outputs $t1$ and $t0$ of the root CHK cell be $tout1$ and $tout0$, respectively. Normally, the values of $tout1$ is equal to the parity of all the original carry bits at even-numbered positions c_{i_e} $(n \geq i_e > 0)$, and the values of $tout0$ is the inverse value of $tout1$. If a carry bit and a corresponding check carry bit have the same value because of a fault, $tout0$ and $tout1$ output the same value.

We let the parity of all the check carry bits at odd-numbered positions c'_{i_o} be $p(C'_{odd})$. When $n = 4w$ or $n = 4w - 1$ $(w > 0)$, $c_{4w-1} \oplus c_{4w-3} \oplus \cdots \oplus c_1$ is equal to $c'_{4w-1} \oplus c'_{4w-3} \oplus \cdots \oplus c'_1$ $(= p(C'_{odd}))$, and therefore, we can obtain the predicted parity PS by calculating $p(X) \oplus p(Y) \oplus p(C'_{odd}) \oplus tout1 \oplus c_0$ because $p(C)$ is equal to $p(C'_{odd}) \oplus tout1 \oplus c_0$. In the other cases, i.e., $n = 4w - 2$ or $n = 4w - 3$, we can obtain the predicted parity PS by calculating $p(X) \oplus p(Y) \oplus p(C'_{odd}) \oplus tout0 \oplus c_0$.

When the adder works correctly, the predicted parity PS is equal to the parity of the sum S and the values of the checker outputs $tout1$ and $tout0$ are the parity of bits in $V0$ and its inverse, respectively. When the adder outputs an erroneous value because of a fault, either the predicted parity PS differs from the parity of the sum S or the values of the checker outputs $tout0$ and $tout1$ are the same.

In the previously proposed fault-secure adders [9], [11], all the carry (or sum) bits are compared with the corresponding redundant ones. Because only one half of the carry bits are compared in our adder, the two-rail checker and wiring area are smaller.

As an example, in Fig. 5, we show the proposed adder with a prefix calculation circuit in Sklansky adder corresponding to Fig. 1. Note that we can reduce the delay time of PS, $tout0$, and $tout1$ by optimizing the tree structure of CHK cells and the tree structure of XOR cells considering the signal delay.

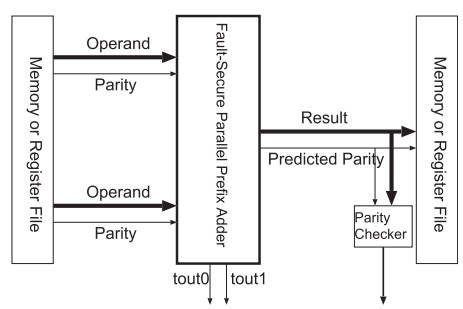

Fig. 6 Example of a datapath circuit with the fault-secure adder in a system using parity-based error detection.

We can easily use the proposed adder in systems using parity-based error detection. We show an example system in Fig. 6. We can use parity bits of the memory or the register file as parities of operands $p(X)$ and $p(Y)$ for the adder. We can use the predicted parity of the adder as the parity bit. In this case, errors in the adder will be detected by a parity checker of the system or by observing the checker output lines of the adder.

Note that, when we want to compare the predicted parity with the parity of the sum in the adder, we append a parity calculation circuit consisting of n XOR cells and compare the predicted parity and the parity of the sum using a CHK cell and an inverter to invert one of the two parities.

While we assume that the input parities $p(X)$ and $p(Y)$ are consistent with the operands X and Y in this section, we can detect an odd number of bit inversions in X, Y, $p(X)$, and $p(Y)$ by comparing the predicted parity PS and the parity of the sum S, because the predicted parity is calculated by $p(X) \oplus p(Y) \oplus p(C)$. In this sense, the adder is code-disjoint.

3.2 Error Detection

Now, we show that we can detect any error caused by a single cell fault by observing the output of the adder. For each type of cells in the adder, we discuss propagation of effects caused by a faulty cell, and show the adder's detectability of erroneous outputs.

First, we discuss an error caused by a faulty S cell, a faulty XOR cell, or a faulty CHK cell. An S cell calculates a sum bit of the adder. Thus, we can detect the error by comparing the predicted parity PS and the parity of the sum S. We can also detect an error caused by a faulty XOR cell for parity prediction by comparing the predicted parity and the parity of the sum. The effect of a faulty CHK cell propagates to outputs $tout0$, $tout1$, and PS, and we can detect it by observing the checker outputs $tout0$ and $tout1$ or by comparing the predicted parity and the parity of the sum.

Next, we discuss an error caused by a faulty C cell or a faulty C' cell. If the C cell that generates c_i is faulty, its effect propagates to the sum bit s_i and/or the check carry bit c'_{i+1}. If both c'_{i+1} and c_i are erroneous, we can detect one of them by observing the checker outputs $tout0$ and $tout1$. If the effect of the C cell does not propagate to c'_{i+1}, we can detect the error by observing the checker outputs (when i is even) or by comparing the predicted parity PS and the parity of the sum S (when i is odd). When a C' cell is faulty, it only affects the circuit for the parity prediction and the checker. Thus, we can detect an error caused by a faulty C' cell in the same manner as that by a faulty CHK cell.

To discuss an error caused by a faulty GP cell, a faulty g cell, or a faulty p cell, we introduce a new notation. For a pair of operands, we let $pm(i)$ be the bit position such that $x_{m_i} \neq y_{m_i}$ for all bit positions m_i between $pm(i)$ and i ($pm(i) > m_i > i$), and $x_{pm(i)} = y_{pm(i)}$. For example, when $(\ldots, x_{i+3}, x_{i+2}, x_{i+1}, \ldots) = (\ldots, 0, 0, 0, \ldots)$ and $(\ldots, y_{i+3}, y_{i+2}, y_{i+1}, \ldots) = (\ldots, 0, 1, 1, \ldots)$, $pm(i)$ is $i + 3$ because x_{i+3} is equal to y_{i+3} while x_{i+2} and x_{i+1} are not the same values as y_{i+2} and y_{i+1}, respectively. Note that we consider $pm(i) = n$ when $x_{m_i} \neq y_{m_i}$ for all bit positions higher than i ($n > m_i > i$).

Now, we discuss an error caused by a faulty g cell generating g_i or a faulty GP cell GP(i, j) such that $dm(i, j) \geq pm(i)$. In this case, the effect of the faulty g cell or the faulty GP cell propagates to the carry bits from c_{i+1} to $c_{pm(i)}$ and to the check carry bits from c'_{i+2} to $c'_{pm(i)}$ because the effect propagates to upper positions while the carry bit $c_{pm(i)+1}$ and the check carry bit $c'_{pm(i)+1}$ are determined only by $x_{pm(i)}$ and $y_{pm(i)}$. We show an example in Fig. 7. In the figure, $(x_{11}, x_{10}, \ldots, x_4, x_3) = (0, 1, \ldots, 1, 0)$, and $(y_{11}, \ldots, x_3) = (0, \ldots, 0)$ are fed to the adder. We assume GP(3,2) is faulty and the output value $G_{3,2}$ is inverted. In this example, $dm(3, 2) = 16$ and $pm(3) = 11$ because propagate signals from p_4 to p_{10} are 1 and $p_{11} = 0$. As described above, carry bits from c_4 ($= c_{(3+1)}$) to c_{11} ($= c_{pm(3)}$), check carry bits from c'_5 to c'_{11}, and sum bits from s_4 to s_{11} are inverted.

While the effect propagates to c_{i+1}, the effect does not propagate to c'_{i+1}. Thus c_{i+1} is equal to c'_{i+1}. When $i + 1$ is even, i.e., i is odd, we can detect the error by observing the checker outputs $tout0$ and $tout1$. In Fig. 7, c_4 and c'_4 have the same value, and they are compared by the checker. When i is even, we calculate the sum output with the erroneous carry bit c_{i+1} while we calculate the predicted parity PS with the correct check carry bit c'_{i+1}. Thus, the predicted parity PS is not equal to the parity of the sum S, and we can detect the error by comparing the predicted parity with the parity of the sum. Therefore, we can detect an error caused by a faulty GP cell or a faulty g cell whether i is even or not.

We also discuss an error caused by a faulty GP cell GP(i, j) such that $pm(i) > dm(i, j)$. In this case, the effect of the faulty cell propagates to carry bits at least from c_{i+1}

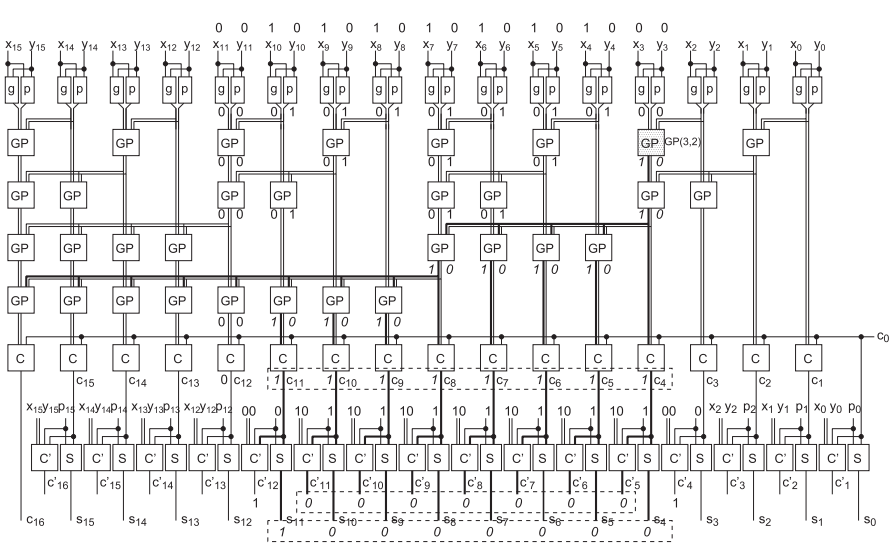

Fig. 7 Example of fault effect propagations by a faulty GP cell when $dm(i, j) \geq pm(i)$ (GP(3,2) is faulty, and the checker circuit and the parity prediction circuit are omitted).

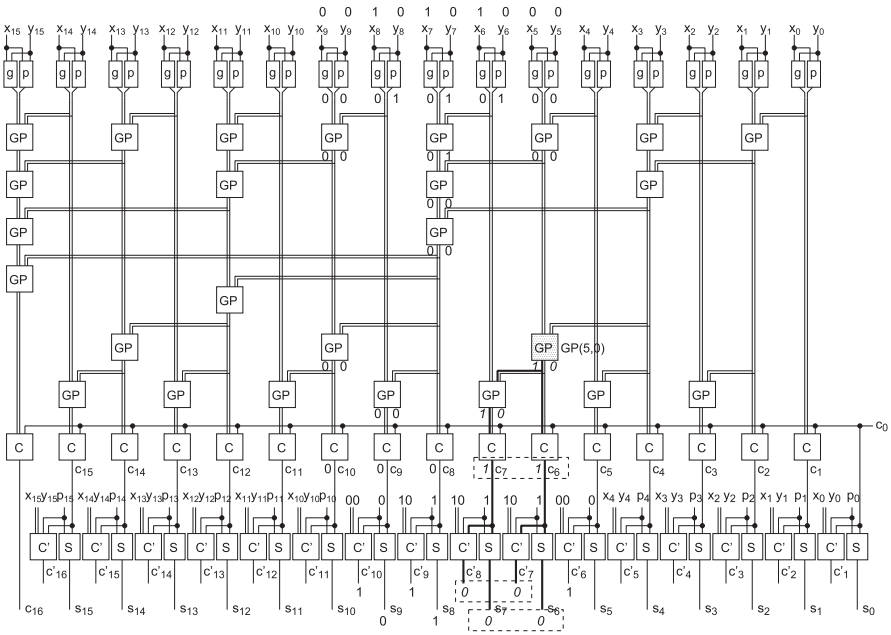

Fig. 8 Example of fault effect propagations by a faulty GP cell when $dm(i, j) < pm(i)$ (GP(5,0) is faulty).

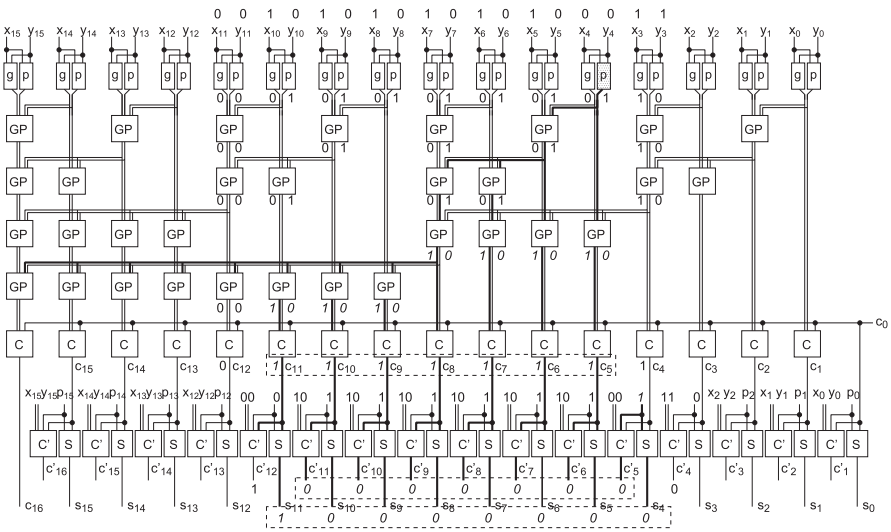

Fig. 9 Example of fault effect propagations by a faulty p cell (p cell that generates p_4 is faulty).

to $c_{dm(i,j)}$, and check carry bits at least from c'_{i+2} to $c'_{dm(i,j)+1}$. Therefore, c_{i+1} and $c_{dm(i,j)+1}$ are equal to c'_{i+1} and $c'_{dm(i,j)+1}$, respectively. Owing to the properties of adders shown in Sect. 2.1, $dm(i, j)$ or i is odd. $c_{dm(i,j)+1}$ or c_{i+1} is compared with the corresponding check carry bit, and errors caused by the faulty GP cell can be detected. We show an example in Fig. 8. In the figure, $(x_9, x_8, x_7, x_6, x_5) = (0, 1, 1, 1, 0)$, and $(y_9, y_8, y_7, y_6, y_5) = (0, 0, 0, 0, 0)$ are fed to the adder. We assume GP(5,0) is faulty and the output value $G_{5,0}$ is inverted. In this example, $dm(5, 0) = 7$ and $pm(5) = 9$. As shown in the figure, c_6, c_7, c'_7, and c'_8 are inverted by the faulty GP cell. Because $dm(5, 0)$ is odd, the bit position of the most significant erroneous check carry bit is even (c'_8 in this case). Thus, the check carry bit is compared with the corresponding correct carry bit by the checker.

Finally, we discuss an error caused by a faulty p cell that generates p_i. The effect of the faulty p cell propagates to the sum bit s_i. The effect also propagates either to no other bit or to the carry bits, the check carry bits, and the sum bits of bit positions from $i + 1$ to $pm(i)$. Thus, the number of inversions in the sum bits are larger by one than the numbers of inversions in the original carry bits and the check carry bits. Therefore, we can detect the error by comparing the predicted parity PS and the parity of the sum S. In Fig. 9, we assume the p cell that generates p_4 is faulty. In the figure, carry bits from c_5 to c_{11} and the corresponding check carry bits are inverted, and the sum bits from s_4 to s_{11} are inverted. Though the number of bit inversions in carry bits and that in

check carry bits are odd, the number of bit inversions in sum bits is even. Thus, the predicted parity PS is not equal to the parity of the sum.

As a result, we can detect an error caused by a faulty GP cell, a faulty g cell, or a faulty p cell. Therefore, by observing the output of the adder, we can detect any error caused by a single cell fault.

4. Hardware Overhead

We estimate hardware overhead of the proposed adder and compare it with that of the previously proposed adder [11]. In Table 3, we show the overhead in a design of the proposed adder and that in a design based on [11] that was implemented using the same cells as in this paper. In the bottom row of the table, we show estimated hardware overhead in the number of equivalent 2-input NAND gates. We consider that an XOR cell is 2.5 gate equivalents because in CMOS technology without transmission gates, an XOR gate and a 2-input NAND gate can be realized by 10 transistors and 4 transistors, respectively. We also consider a C' cell and a CHK cell as 2 gate equivalents and 4.5 gate equivalents, respectively. The overhead of the proposed adder is 15% ($\approx \frac{2}{13}$) lower than that in [11].

To acquire the actual overhead, we designed the proposed adder and the previously proposed adder [11] having the same prefix calculation circuit in a Sklansky adder. We synthesized them with Synopsys design compiler and adopted the Rohm 0.18 μm CMOS process cell library. In Tables 4 and 5, we show the synthesis results. Area constraints are used for synthesis. The hardware overhead in the design of the proposed adder is about 15% lower than that in the design based on [11]. The delay time of the design of the proposed adder is smaller than or nearly equal to the delay time of the design based on [11]. The impact of the modification on delay time of the sum outputs is small.

Note that, in the proposed adder, ratio of the hardware overhead in designs varies depending on the number of GP cells because the additional hardware are the same for designs. For example, a design of Brent-Kung adder has smaller amount of GP cells and a design of Kogge-Stone adder has larger amount of GP cells than a design of Sklansky adder. Thus, ratio of the overhead in a design of Kogge-Stone adder is smaller than that in a design of Sklansky adder, and that in a design of Brent-Kung adder is larger. We show hardware overhead in designs of Brent-Kung adder, Han-Carlson adder, and Kogge-Stone adder in Tables 6, 7, and 8, respectively.

Table 3 Comparison of hardware overhead.

	Proposed	Based on [11]
Number of C' cells	n	n
Number of CHK cells	$\frac{n}{2}-1$	$n-1$
Number of XOR cells	$\frac{n}{2}+3$	3
Total gate equivalents	$\frac{11n}{2}+3$	$\frac{13n}{2}+3$

5. Conclusion

We have proposed a low-overhead fault-secure parallel prefix adder. We duplicate carry bits for checking purposes, and compare only one half of the carry bits with the corresponding check carry bits using a two-rail checker. We also predict the parity of the result for error detection purposes. We can detect errors by observing the output lines of the

Table 4 Comparison of area overhead.

N	Normal	Proposed		Based on [11]		Overhead ratio
	Area [μm^2]	Area [μm^2]	Area overhead [%]	Area [μm^2]	Area overhead [%]	Proposed/Based on [11] [%]
8	1486.6	2145.9	44.3	2249.4	51.3	86.4
16	3361.0	4640.8	38.1	4847.8	44.2	86.2
32	7497.8	10018.6	33.6	10432.6	39.1	85.9
64	16585.7	21588.7	30.2	22416.7	35.2	85.8

Table 5 Comparison of delay overhead.

N	Normal	Proposed				Based on [11]			
	Sum [ns]	Sum [ns]	PS [ns]	$tout0$ [ns]	$tout1$ [ns]	Sum [ns]	PS [ns]	$tout0$ [ns]	$tout1$ [ns]
8	1.83	2.06	2.71	2.26	2.32	2.06	2.83	2.58	2.63
16	2.70	2.92	3.86	3.35	3.40	2.92	3.91	3.66	3.71
32	4.02	4.25	5.44	4.88	4.93	4.25	5.45	5.19	5.24
64	6.49	6.71	8.17	7.54	7.60	6.71	8.12	7.86	7.91

Table 6 Comparison of area overhead (Brent-Kung adder).

N	Normal	Proposed		Based on [11]		Overhead ratio
	Area [μm^2]	Area [μm^2]	Area overhead [%]	Area [μm^2]	Area overhead [%]	Proposed/Based on [11] [%]
8	1438.1	2097.4	45.8	2200.9	53.0	86.4
16	3070.0	4349.8	41.7	4556.8	48.4	86.2
32	6382.4	8903.3	39.5	9317.3	46.0	85.9
64	13055.7	18058.6	38.3	18886.6	44.7	85.7

Table 7 Comparison of area overhead (Han-Carlson adder).

N	Normal	Proposed		Based on [11]		Overhead ratio
	Area [μm^2]	Area [μm^2]	Area overhead [%]	Area [μm^2]	Area overhead [%]	Proposed/Based on [11] [%]
8	1486.6	2145.8	44.3	2249.3	51.3	86.4
16	3360.9	4640.7	38.1	4847.7	44.2	86.2
32	7497.5	10018.3	33.6	10432.3	39.1	85.9
64	16546.3	21549.2	30.2	22377.2	35.2	85.8

Table 8 Comparison of area overhead (Kogge-Stone adder).

N	Normal	Proposed		Based on [11]		Overhead ratio
	Area [μm^2]	Area [μm^2]	Area overhead [%]	Area [μm^2]	Area overhead [%]	Proposed/Based on [11] [%]
8	1729.0	2388.3	38.1	2491.8	44.1	86.4
16	4185.1	5464.9	30.6	5671.9	35.5	86.2
32	9873.0	12393.9	25.5	12807.9	29.7	85.9
64	22800.3	27803.3	21.9	28631.3	25.6	85.5

checker or by comparing the predicted parity with the parity of the result. The hardware overhead of the proposed adder is lower compared to the previously proposed fault-secure adders because the proposed adder's checker has a smaller bit width than that of the checker in the previously proposed adders. We can implement adders with various prefix calculation circuits such as Sklansky adder, Brent-Kung adder, Han-Carlson adder and Kogge-Stone adder. Evaluation results show that the hardware overhead of the proposed adder is about 15% lower than that of the previously proposed adder. By the results, we can conclude that a reduction in the bit width of the checker in the proposed adder is effective for total area reduction of a fault-secure parallel prefix adder.

Acknowledgement

This work is supported by VLSI Design and Education Center (VDEC), The University of Tokyo with the collaboration with Synopsys Corporation.

References

[1] H. Ando, Y. Yoshida, A. Inoue, I. Sugiyama, T. Asakawa, K. Morita, T. Muta, T. Motokurumada, S. Okada, H. Yamashita, Y. Satsukawa, A. Konmoto, R. Yamashita, and H. Sugiyama, "A 1.3 GHz fifth generation SPARC64 microprocessor," Proc. 40th Annual Design Automation Conference, DAC '03, pp.702–705, 2003.

[2] J. Rivers, M. Gupta, J. Shin, P. Kudva, and P. Bose, "Error tolerance in server class processors," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol.30, no.7, pp.945–959, July 2011.

[3] S. Vangal, M. Anders, N. Borkar, E. Seligman, V. Govindarajulu, V. Erraguntla, H. Wilson, A. Pangal, V. Veeramachaneni, J. Tschanz, Y. Ye, D. Somasekhar, B. Bloechel, G. Dermer, R. Krishnamurthy, K. Soumyanath, S. Mathew, S. Narendra, M. Stan, S. Thompson, V. De, and S. Borkar, "5-GHz 32-bit integer execution core in 130-nm dual-Vt CMOS," IEEE J. Solid-State Circuits, vol.37, no.11, pp.1421–1432, 2002.

[4] J. Sklansky, "Conditional-sum addition logic," IRE Trans. Electronic Computers, vol.EC-9, no.2, pp.226–231, June 1960.

[5] R.P. Brent and H.T. Kung, "A regular layout for parallel adders," IEEE Trans. Comput., vol.31, no.3, pp.260–264, March 1982.

[6] T. Han and D.A. Carlson, "Fast area-efficient VLSI adders," Proc. 8th IEEE Symposium on Computer Arithmetic (ARITH), pp.49–56, May 1987.

[7] P.M. Kogge and H.S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," IEEE Trans. Comput., vol.22, no.8, pp.786–793, Aug. 1973.

[8] M. Nicolaidis, R.O. Duarte, S. Manich, and J. Figueras, "Fault-secure parity prediction arithmetic operators," IEEE Des. Test Comput., vol.14, pp.60–71, 1997.

[9] E.S. Sogomonyan, V. Ocheretnij, and M. Gössel, "A new code-disjoint sum-bit duplicated carry look-ahead adder for parity codes," Proc. 10th Asian Test Symposium, pp.365–370, Nov. 2001.

[10] D. Marienfeld, E.S. Sogomonyan, V. Ocheretnij, and M. Gössel, "A new self-checking code-disjoint carry-skip adder," Proc. 8th IEEE International On-Line Testing Workshop, pp.39–43, 2002.

[11] M. Nicolaidis, "Carry checking/parity prediction adders and ALUs," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.11, no.1, pp.121–128, Feb. 2003.

Nobutaka Kito received the B.E., M.I.S. and Dr. of Information Science degrees in information engineering 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 CAD algorithms for single flux quantum circuits, design for testability, and computer arithmetic.

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 was 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.