

Efficient squaring circuit using canonical signed-digit number representation

Yuuki Tanaka^{a)} and Shugang Wei

LETTER

Division of Mechanical Science and Technology, Faculty of Science and Technology, Gunma University, Ota, Gunma, 373–0057, Japan a) ytanaka@gunma-u.ac.jp

Abstract: Squaring and exponentiation of a number are fundamental arithmetic and widely used in the real-time applications such as image processing, digital filtering and cryptography. In this paper, we propose a squaring algorithm of an integer with canonical signed-digit (CSD) number representation. For an *n*-digit CSD number, our method generates n/4 CSD numbers of 2n-digit length as partial products. This result is half with respect to the conventional squaring algorithms. We implement the squaring circuit based on this algorithm and compare with some existing circuits. Our circuit is 40% faster than the known squaring circuit for binary numbers.

Keywords: canonical SD number, squaring circuit **Classification:** Integrated circuits

References

- [1] J.-T. Yoo: IEEE J. Solid-State Circuits 32 (1997) 90.
- [2] D. De Caro and A. G. M. Strollo: Electronics Letters **37** (2001) 346.
- [3] K. Sethi and R. Panda: Int. J. Advanced Computer Science and Applications 3 (2012) 111.
- [4] M. Yamada and A. Nishihara: ASP-DAC (2001) 7.
- [5] G. W. Reitwiesner: Advances in Computers 1 (1960) 231.
- [6] Y. Tanaka, Y. Zhang and S. Wei: ICDV (2013) 67.
- [7] J. P. Deschamps, G. J. A. Bioul and G. D. Sutter: Synthesis of arithmetic circuits (Wiley-Interscience, 2006).
- [8] N. Takagi, H. Yasuura and S. Yajima: IEEE Trans. Comput. 34 (1985) 789.
- [9] M. Zhang: Ph.D. thesis Gunma university (2013).

1 Introduction

Squaring and exponentiation are one of the important arithmetic operations. These operations are widely used in many applications such as image processing, digital filtering and cryptography. Therefore, developing low-power, small-area and fast algorithms and implementations are required. In response to those requirements, various squaring circuits based on the binary number





system have been proposed [1, 2, 3].

The canonical signed-digit (CSD) representation is one of the number representations in signed-digit (SD) number system which has two main properties:

- 1. The number of nonzero digits is minimal.
- 2. No two consecutive digits are both nonzeros.

For any integer, there exists only one representation which satisfies the above property. For example, $(010\overline{1})$ is the CSD representation of 3, where $\overline{1}$ stands for -1. Since the nonzero digits is minimal, the number of additions can be reduced if it is used for the multiplication. Thus, CSD representation is recently used in some applications such as FIR filter [4].

To apply CSD representation to various arithmetic operations, several recoding algorithms have been developed to obtain a CSD representation efficiently since the first CSD recoding algorithm was proposed in [5].

In this paper, we propose a squaring circuit based on the canonical signeddigit (CSD) number representation. Since the input of our circuit is CSD represented number, some conversion circuit does not need to squaring the CSD represented number. The output can be represented by SD number or CSD number. Efficient conversion algorithms and circuits from SD number to CSD number have been proposed in [6]. This circuit consists of a partial product generator, CSD adder and multi operand SD adder. We propose a new partial product generator based on a CSD number representation. For n-digit CSD number, our partial product generator generates n/4 partial products which are CSD number. The number of partial products is about 1/2 compare with the conventional multiplier with modified-Booth recoding. The CSD adder and multi operand SD adder stand for merging the partial products to obtain a square of the input.

2 SD number system

In this section, we introduce a number system and representation which are treated in this paper.

The signed-digit (SD) number system is a redundant binary representation. The SD representation has a fixed radix 2 and a digit set $\{\overline{1}, 0, 1\}$ where $\overline{1}$ stands for -1. An integer X can be represented by an *n*-digit SD number representation as follows:

$$X = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \dots + x_12^1 + x_0, \ x_i \in \{\overline{1}, 0, 1\}.$$
 (1)

A given *n*-digit SD number representation has a value range of $[-(2^n-1), 2^n-1]$. Obviously,

$$-X = (\overline{x_{n-1}} \ \overline{x_{n-2}} \cdots \overline{x_0}), \tag{2}$$

where $\overline{x_i}$ stands for $-x_i$.

There are several representation of an integer. For example, an integer 5 can be represented as (0101), $(1\overline{1}01)$ and $(10\overline{11})$. One famous property of the





SD number system is the carry-propagation-free addition and subtraction. Therefore, the SD number system is suitable in the field of large number of digit arithmetic, such as a cryptography.

3 A CSD-based partial product generator for squaring

In this section, we propose a CSD-based partial product generator for calculating a square of the input. This generator inputs an *n*-digit CSD number and outputs n/4 CSD numbers of 2n-digit. Throughout this paper, we assume that the length *n* is a multiple of 4.

For n = 4, numbers between -10 to +10 can be represented by 4-digit CSD form. The CSD representation of the square of each value is shown in Table I.

value	$x_3x_2x_1x_0$	$(x_3x_2x_1x_0)^2$	value	$x_3x_2x_1x_0$	$(x_3x_2x_1x_0)^2$
0	0000	000000000			
+1	0001	$0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1$	-1	$000\overline{1}$	$0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1$
+2	0010	00000100	-2	$00\overline{1}0$	$0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0$
+3	$010\overline{1}$	$0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1$	-3	$0\overline{1}01$	$0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1$
+4	0100	$0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0$	-4	$0\overline{1}00$	$0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0$
+5	0101	$0\ 0\ 1\ 0\ \overline{1}\ 0\ 0\ 1$	-5	$0\overline{1}0\overline{1}$	$0 \ 0 \ 1 \ 0 \ \overline{1} \ 0 \ 0 \ 1$
+6	$10\overline{1}0$	00100100	-6	$\overline{1}010$	$0\ 0\ 1\ 0\ 0\ 1\ 0\ 0$
+7	$100\overline{1}$	$0\ 1\ 0\ \overline{1}\ 0\ 0\ 0\ 1$	-7	$\overline{1}001$	$0\ 1\ 0\ \overline{1}\ 0\ 0\ 0\ 1$
+8	1000	01000000	-8	$\overline{1}000$	$0\ 1\ 0\ 0\ 0\ 0\ 0\ 0$
+9	1001	$0\ 1\ 0\ 1\ 0\ 0\ 1$	-9	$\overline{1}00\overline{1}$	$0\ 1\ 0\ 1\ 0\ 0\ 1\ 1$
+10	1010	$1 \ 0 \ \overline{1} \ 0 \ 0 \ 1 \ 0 \ 0$	-10	$\overline{1}0\overline{1}0$	$1\ 0\ \overline{1}\ 0\ 0\ 1\ 0\ 0$

Table I. 4-digit CSD and its square.

According to Table I, we obtain $Z = z_7 z_6 \cdots z_0 = (x_3 x_2 x_1 x_0)^2$ as follows:

z_7	=	$x_3 \odot x_1,$	z_6	=	$ x_3 \cdot (x_1'),$
z_5	=	$-(x_3\cdot x_1)+(x_2\odot x_0),$	z_4	=	$(x_3 \cdot x_0) + (x_2 \cdot (x'_0)),$
z_3	=	$-(x_2\cdot x_0),$	z_2	=	$ x_1 ,$
z_1	=	0,	z_0	=	$ x_0 .$

In the above equations,

- $x_i \odot x_j = 1$ when $x_i = x_j = 1$ or $x_i = x_j = \overline{1}$, otherwise $x_i \odot x_j = 0$.
- $x'_i = 1$ when $x_i = 0$, otherwise $x'_i = 0$.
- $|x_i| = 1$ when $x_i \neq 0$, otherwise $|x_i| = 0$.
- $x_i \cdot x_j = 1$ when $x_i = x_j = \pm 1$, $x_i \cdot x_j = \overline{1}$ when $x_i = -x_j$, otherwise $x_i \cdot x_j = 0$.

In the above equations, it seems that z_5 and z_4 need an addition of two terms. However, one of those two terms in z_5 and z_4 must be zero since X is





a CSD representation. Thus, "+" notation can be fulfilled by a simple OR logical operation. It is easy to verify that no consecutive digits of Z are both nonzero if X is a CSD representation. We implement a squaring circuit for 4-digit CSD by logical operations according to Table I.

For $n \geq 8$, we construct a squaring circuit recursively. Some of known squaring circuits are constructed by partition the *n*-bit input into two parts, that is, upper n/2-bit and lower n/2-bit and apply the squaring and multiplication recursively. Although our algorithm is based on the recursive structure, the partition scheme of the input digit is different from these. For any integer i > 1, we next show the construction of the partial product generator for $n = 4 \times i$. We decompose an *n*-digit CSD number

$$X = \sum_{i=0}^{n-1} x_i \cdot 2^i \tag{3}$$

into 2, (n-4) and 2 digits, that is,

$$X = 2^{n-2} \times \sum_{i=n-2}^{n-1} x_i \cdot 2^{i-(n-2)} + 2^2 \times \sum_{i=2}^{n-3} x_i \cdot 2^{i-2} + 2^0 \times \sum_{i=0}^{1} x_i.$$
 (4)

By using this decomposition, X^2 can be represented as

$$X^{2} = \left((x_{n-1}x_{n-2}) \times 2^{n-2} + (x_{n-3}x_{n-4} \cdots x_{2}) \times 2^{2} + (x_{1}x_{0}) \right)^{2}$$

= $A + B + C + D$,

where

1

$$A = (x_{n-1}x_{n-2})^2 \times 2^{2(n-2)} + (x_{n-3}x_{n-4}\cdots x_2)^2 \times 2^{2\cdot 2} + (x_1x_0)^2$$
(5)

$$B = 2 \times (x_{n-1}x_{n-2}) \times (x_{n-3}x_{n-4} \cdots x_2) \times 2^{n-2+2}$$
(6)

$$C = 2 \times (x_{n-1}x_{n-2}) \times (x_1x_0) \times 2^{n-2}$$
(7)

$$D = 2 \times (x_{n-3}x_{n-4} \cdots x_2) \times (x_1x_0) \times 2^2.$$
(8)

We show that X^2 can be represented as a sum of n/4 CSD numbers of 2*n*-digit by showing that A can be represented by sum of (n - 4)/4 CSD numbers and B + C + D can be represented by one CSD number with shift operation and inversion of the sign operation.

Let $X^2 = A + Z$ where Z = B + C + D. We place terms in A and Z according to Fig. 1. In Fig. 1, $(x_{n-1}x_{n-2})^2$, $(x_{n-3}x_{n-4}\cdots x_2)^2$, $(x_1x_0)^2$ and $(x_{n-1}x_{n-2}) \times (x_1x_0)$ are placed into a blue, green, magenta, and cyan rectangles, respectively. Shaded rectangles in Fig. 1 are filled by 0. These regions are discussed later.

By the construction scheme, $(x_{n-3}x_{n-4}\cdots x_2)^2$ can be represented by a sum of (n-4)/4 CSD numbers of 2(n-4)-digit. These are placed into the green rectangle shown in Fig. 1.

Next $(x_{n-1}x_{n-2})^2$ is placed into the blue rectangle shown in Fig. 1. The green rectangle and the blue rectangle do not overlap, two terms can be placed with shift operation and inversion of the sign operation. We should pay attention to the digits placed in the rightmost of the blue rectangle and upper-leftmost digit in the green rectangle. If both of these digits are nonzero, then the partial product of the first column is not a CSD representation.







Fig. 1. A partition of X^2 into partial products.

To prove that an SD number is a CSD representation, it is sufficient to show that there are no two adjacent nonzero digits since a representation of an SD number is CSD if and only if the representation has no two adjacent nonzero digits.

If $n \ge 8$, the leftmost digits of all partial products are zeros by the construction scheme and Fig. 1. If n = 4, the leftmost digit of a partial product does not always zero because a partial product is generated according to Table I. When n = 8, the leftmost digit in the green rectangle, that is, 11th digit of A is nonzero if $x_5 = x_3 = \pm 1$. In that case, $x_6 = 0$ and $(x_7x_6)^2$ is either 4 or 0. Therefore, the rightmost digit of the blue rectangle, that is, 12th digit of A is zero. Thus, the CSD property holds if $(x_{n-1}x_{n-2})^2$ is placed into the blue rectangle shown in Fig. 1.

Since $(x_{n-3}x_{n-4}\cdots x_2)^2 \times 2^{2\cdot 2} \geq 2^4$ and $(x_1x_0)^2 \leq 2^2$, $(x_1x_0)^2$ can be placed to the magenta rectangle shown in Fig. 1.

Therefore, A can be represented by sum of (n-4)/4 CSD numbers.

Next, we show that Z can be represented by one CSD number with shift operation and inversion of the sign operation. In Fig. 2, a notation "1*" in the cyan rectangle represents either 1 or $\overline{1}$. If $(x_1x_0) = 0$ or $(x_{n-1}x_{n-2}) = 0$, then at most one of three terms is nonzero. Then, it is easy to see that Z is represented by CSD. Therefore, we consider that $(x_1x_0) \neq 0$ and $(x_{n-1}x_{n-2}) \neq 0$. In Fig. 2, an assignment of C overlaps with that of B or D for the cases (a),(b) and (d). It seems that we need some additions to be placed in one partial product. However, as shown in Fig. 2, a nonzero digit in C does not overlap with B and D. Therefore, assignment of these digits can be processed by OR logical operation.

Next, we show that these assignments obtain a number with the CSD representation.

Case 1: $x_0 \neq 0$ and $x_{n-2} \neq 0$. The subproduct $(x_{n-3}x_{n-4}\cdots x_2) \times x_0$ is placed between (n-2)-nd digit and third digit. The subproduct $(x_{n-3}x_{n-4}\cdots x_2) \times x_{n-2}$ is placed between (2n-4)-th digit and (n+1)-st digit. Since X is a CSD representation, $x_{n-3} = 0$. Then, the leftmost





digit of $(x_{n-3}x_{n-4}\cdots x_2) \times x_0$ is zero and (n-2)-nd digit in Fig. 2 (a) is also zero.

- **Case 2:** $x_1 \neq 0$ and $x_{n-2} \neq 0$. The subproduct $(x_{n-3}x_{n-4}\cdots x_2) \times x_1$ is placed between (n-1)-st digit and 4th digit. The subproduct $(x_{n-3}x_{n-4}\cdots x_2) \times x_{n-2}$ is placed between (2n-4)-th digit and (n+1)st digit. Since X is a CSD representation, $x_2 = 0$ and $x_{n-3} = 0$. Then, the leftmost digit of $(x_{n-3}x_{n-4}\cdots x_2) \times x_1$ and the rightmost digit of $(x_{n-3}x_{n-4}\cdots x_2) \times x_{n-2}$ are zero. Therefore, (n-2)-nd digit and (n+1)-st digit in Fig. 2 (b) are zero.
- **Case 3:** $x_0 \neq 0$ and $x_{n-1} \neq 0$. The subproduct $(x_{n-3}x_{n-4}\cdots x_2) \times x_0$ is placed between (n-2)-nd digit and third digit. The subproduct $(x_{n-3}x_{n-4}\cdots x_2) \times x_{n-1}$ is placed between (2n-3)-rd digit and (n+2)-nd digit. In this case, it is easy to verify that there are no consecutive nonzero digits from Fig. 2 (c).
- **Case 4:** $x_1 \neq 0$ and $x_{n-1} \neq 0$. The subproduct $(x_{n-3}x_{n-4}\cdots x_2) \times x_1$ is placed between (n-1)-st digit and 4th digit. The subproduct $(x_{n-3}x_{n-4}\cdots x_2) \times x_{n-1}$ is placed between (2n-3)-rd digit and (n+2)-nd digit. Since $x_1 \neq 0$, $x_2 = 0$ and the rightmost digit of $(x_{n-3}x_{n-4}\cdots x_2) \times x_{n-1}$ is zero. Therefore, (n+2)-nd digit in Fig. 2 (d) is zero.

In all cases, the CSD representation of Z can be obtained from B, C and D with shift operation and inversion of the sign operation.

3.1 Some fixed digits in partial products

Next we analyze each digit in the partial products generated by the proposed generator.

Definition 1 In an array of partial products, a digit is called an inactive digit if the digit is always zero, that is, the digit does not change according to the input. A digit which is not an inactive digit is called an active digit.

If some inactive digits are specified, we can reduce adders in multi-operand adder used to obtain a square of the input.

In Fig. 1, shaded rectangles represent some inactive digits. Moreover, there exist other inactive digits in green rectangle. We can specify such digits by the recursive structure of the partial product generator. The following theorem shows the number of active digits in the partial products obtained by the proposed generator.

Theorem 1 For an n-digit CSD input, there are n(n+3)/4 active digits in outputs from the proposed partial product generator.

Proof: Prove by induction on n. When n = 4, a digit z_1 is inactive digit and the number of active digit is 7. For a digit n which is a multiple of 4, the number of active digit is denoted by ad(n). Next we assume that the statement holds for some n. ad(n + 4) is a sum of the following terms:







Fig. 2. One CSD representations on Z = B + C + D.

- $\operatorname{ad}(n)$ digits for $(x_{n+1}x_n\cdots x_2)^2$,
- 2 digits for $(x_{n+3}x_{n+2})^2$,
- 2 digits for $(x_1x_0)^2$,
- 2(n+1) + 1 = 2n + 3 digits for Z.

Therefore, ad(n + 4) = ad(n) + 2n + 7. By solving this recurrence with ad(4) = 7, we obtain ad(n) = n(n+3)/4.

Most conventional squaring circuits output n(n + 1)/2 active digits [7]. In contrast, a number of active digits of our method is almost half and our method can reduce the additions in the multi-operand SD adder to obtain a square of the input.

4 Addition of two CSD numbers

In this section, we show that an addition of two CSD numbers is accomplished simpler than the conventional addition of SD numbers. This adder inputs two CSD numbers and outputs one SD number (may not CSD).

Lemma 1 Let $X = (x_{n-1}x_{n-2}\cdots x_0), Y = (y_{n-1}y_{n-2}\cdots y_0)$ be n-digit CSD numbers. Then, i-th digit of $Z = (z_n z_{n-1} \cdots z_0) = X + Y$ is obtained by





 $z_i = x_i \oplus y_i + (x_{i-1} \times y_{i-1}), \text{ where } (x_{i-1} \times y_{i-1}) = x_{i-1} \text{ if } x_{i-1} = y_{i-1}, \text{ otherwise } 0.$

Proof: Since no two consecutive digits are nonzeros in the CSD representation, if $(x_{i-1} \times y_{i-1})$ gets nonzero, then x_i and y_i should be zero. If x_i or y_i is nonzero, $(x_{i-1} \times y_{i-1})$ should be zero. Therefore, a carry propagation does not occur.

5 Evaluation of proposed circuit

In this section, we evaluate the proposed squaring circuit. In this paper, we implement a 32-digit length circuit by using VHDL to describe the proposed algorithm. A block diagram is shown in Fig. 3. The partial product generator is discussed in Section 3 and the CSD adder is discussed in Section 4. Since the partial product generator outputs 8 CSD numbers, the CSD adder consists of 4 CSD adders operated in parallel and outputs 4 SD numbers. The multi-operand SD adder forms a 4 inputs redundant binary addition tree structure [8]. In the multi-operand SD adder, we use SD adders proposed in [9].

If someone wants to obtain an output as CSD, some efficient SD to CSD recoding circuits [6] are used for a requirement.



Fig. 3. A block diagram of proposed circuit.

One competitor is a squaring circuit using arithmetic operator "*" in VHDL, that is, a circuit given by dataout <= datain * datain;.

Another competitor is an implementation of the squaring circuit proposed in [3]. This squaring circuit is based on Vedic multiplier. Firstly, an *n*-bit input is partitioned into upper n/2-bit and lower n/2-bit. Secondly, square of upper n/2-bit, square of lower n/2-bit and products of upper n/2-bit and lower n/2-bit are calculated in parallel. To obtain squares of upper n/2-bit and lower n/2-bit, they use their algorithm recursively. Thirdly, square of the *n*-bit input is obtained by adding these terms with appropriate bit shift.

Since these competitors input a binary number and output a binary number, we cannot compare those circuits with proposed circuit directly. However, we need some references to show the efficiency of our method. All of them are synthesized with Synopsys design compiler with $0.18 \,\mu m$ CMOS library. Results are shown in Table II.





	Operator	Vedic	Proposed with	Proposed with
		[3]	SD output	CSD output
Area(# cell)	1327	2042	2473	3639
$\operatorname{Area}(\mu m^2)$	47668.40	53093.85	61648.89	90159.85
Delay(ns)	17.24	16.11	7.16	9.51

Table II. Comparison of area and delay time.

In Table II, last column is a modified proposed circuit which connects a 32-digit SD to CSD recoder [6] following Fig. 3. The difference between last two columns is derived from the recoder.

Although our circuits represent one digit by two bits, the total area of our proposed circuit with CSD output is 70% larger than the vedic squaring circuit. The delay time of our circuit is 40% faster than the vedic squaring circuit.

6 Conclusion

In this paper, we have proposed a new squaring circuit based on the Canonical SD number representation. By using a CSD representation as an input of the partial product generator, n/4 partial products with CSD number representation are generated, which are half with respect to the conventional squaring algorithm. We consider not only the number of partial products but also a number of active digit in the partial products. Our circuit outputs partial products with less active digits, which leads us to a low-power implementation.

Acknowledgments

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

