INTEGRATION the VLSI journal xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

A high-accuracy approximate adder with correct sign calculation

Junjun Hu, Zhijing Li, Meng Yang, Zixin Huang, Weikang Qian*

University of Michigan-Shanghai Jiao Tong University Joint Institute, Shanghai Jiao Tong University, Shanghai, China

ARTICLE INFO

Keywords:
Approximate adder
Approximate computing
Low relative error
Sign error correction
Low-power design

ABSTRACT

Conventional precise adders take long delay and large power consumption to obtain accurate results. Exploiting the error tolerance of some applications such as multimedia, image processing, and machine learning, a number of recent works proposed to design approximate adders that generate inaccurate results occasionally in exchange for reduction in delay and power consumption. However, most of the existing approximate adders have a large relative error. Besides, when applied to 2's complement signed addition, they sometimes generate a wrong sign bit. In this paper, we propose a novel approximate adder that exploits the generate signals for carry speculation. Furthermore, we introduce a low-overhead module to reduce the relative error and a sign correction module to fix the sign error. Compared to the conventional ripple carry adder and carry-lookahead adder, our adder with block size of 4 reduces power-delay product by 66% and 32%, respectively, for a 32-bit addition. Compared to the existing approximate adders, our adder significantly reduces the maximal relative error and ensures correct sign calculation with comparable area, delay, and power consumption. We further tested the performance of our adders with and without the sign error correction module in three real applications, mean filter, edge detection, and *k*-means clustering. The experimental results demonstrated the importance of reducing the relative error and ensuring the correct sign calculation for 2's complement signed additions. The outputs produced using our adder with the sign error correction module are very close to those produced using accurate adder.

1. Introduction

As CMOS devices are scaled into the sub-nanometer regime, power consumption has become a major bottleneck in sustaining Moore's law. Thus, energy efficiency has become a critical concern in designing VLSI circuits. At the same time, with the prevalence of mobile computing, there is an increasing demand for signal processing, multimedia, machine learning, and pattern recognition applications [1]. These applications are essentially error tolerant due to various reasons such as limitation of human perception, redundancy in the input signal, and lack of a unique golden answer [2]. As a result, an inaccurate computation result may still lead to an output with acceptable quality. The relaxation of the accuracy requirement for these applications potentially enlarges the design space, which may contain some solutions with smaller area, delay, and power consumption than those targeted for accurate computation. This leads to a new design paradigm, known as *approximate computing*, which deliberately sacrifices a small amount of accuracy to achieve improvement in performance and power consumption [3].

In this work, we focus on designing approximate adder. As adders are key building blocks in many applications that are suitable for approximate computing, many previous works propose various designs of approximate adders [4–7] (A detailed review of several existing representative approximate adders can be found in Section 2). These adders have smaller areas, delays, and power consumption compared to the accurate ones. Many of them also have small error rates. However, most of them cannot guarantee a small relative error in their outputs. As a result, they may degrade the output quality for some applications. Furthermore, these approximate adders are subject to sign calculation error when doing signed addition for 2's complement numbers.

In this paper, to address the above problems of the existing approximate adders, we propose a novel approximate adder design. It exploits the generate signals to produce the speculated carry signals. This approach leads to a simplified circuity to calculate carry, resulting in a dramatic reduction in adder area and power consumption compared to the accurate adders. To reduce the maximal relative error, we introduce a low-overhead error reduction module. Furthermore, to eliminate the potential sign error in 2's complement signed addition, we introduce a lightweight sign error correction

* Corresponding author.
E-mail addresses: wujunjun_sh@hotmail.com (J. Hu), liuyuedtian@sjtu.edu.cn (Z. Li), yangm.meng@sjtu.edu.cn (M. Yang), zxhuang14@sjtu.edu.cn (Z. Huang), qianwk@sjtu.edu.com (W. Qian).

http://dx.doi.org/10.1016/j.vlsi.2017.09.003
Received 5 May 2017

module. We provide a rigorous analysis on two error measures, error rate and maximal relative error, of the proposed adders with and without the sign correction module. The analysis shows that both adders have a low error rate and a low relative error.

Compared to the conventional ripple carry adder (RCA) and carry-lookahead adders (CLA), our approximate adder with sign error correction module reduces power-delay product by 66% and 32%, respectively, for a 32-bit addition. Compared to the existing approximate adders, our adder significantly reduces the maximal relative error and ensures correct sign calculation with comparable area, delay, and power consumption. To verify the performance of our adders with and without the sign correction module in real situations, we also apply them to three applications, mean filter, edge detection, and *k*-means clustering, which have additions and/or subtractions as their major computation. The output quality for those applications using the proposed adder with the sign correction module is very close to that using an accurate adder.

A preliminary version of this work was published in [8]. Compared to that version, in this work, we further rigorously analyze the relative error and error rate of the proposed sign-correct approximate adder. Furthermore, we empirically study the performance of the proposed adder in real applications and demonstrate its good performance.

The remainder of this paper is organized as follows. In Section 2, we discuss the related works. In Section 3, we introduce some preliminaries on conventional adders. In Section 4, we describe our proposed approximate adder with low relative error and analyze its relative error and error rate. In Section 5, we present an enhanced version with sign correction. In Section 6, we experimentally studied the different metrics of the proposed approximate adders and evaluate them in real applications. The conclusion is drawn in Section 7.

2. Related works

The emerging paradigm of approximate computing has been applied at various levels of modern computing systems. For example, at the algorithm level, random sampling-based approaches can be viewed as approximate algorithms that trade off accuracy for the improvement in runtime. They have significant runtime advantage over the deterministic counterparts for many compute-demanding applications, such as high-dimensional integral [9] and large matrix factorization [10]. At the compiler level, loop perforation was proposed to skip a number of loops to accelerate the program at the cost of introducing some amount of error [11]. At the architecture level, Esmaeilzadeh et al. proposed to execute compute-intensive approximable code on a neural network-based accelerator [12]. Imani et al. proposed a resistive configurable associative memory that enables approximate matching induced by voltage overscaling [13]. At the circuit level, many approximate arithmetic circuits, such as adders [4–7], multipliers [14,15], and dividers [16], were proposed. For some applications, it is desirable to reconfigure the approximate circuits to achieve different accuracy requirements at runtime. For this purpose, several works also proposed additional error detection and correction modules [17–19]. Also, techniques for analyzing specific approximate circuits were developed. For example, Mazahir et al. proposed a probabilistic error modeling method for a specific class of approximate adders that comprise of sub-adder units [20]. They also proposed a method for analyzing the error of approximate multipliers constructed from approximate partial product modules [21]. Furthermore, several other works designed approximate circuits for specific error-tolerant applications, such as video encoding [22] and artificial neural network [23]. For a detailed survey of approximate computing, the readers can refer to the papers [24–26].

Since adder is a basic module in many error-tolerant applications, researchers have proposed a number of approximate adders in the previous works. One type of approximate adder uses an accurate adder to calculate the sum bits at the most significant positions, while applying a simple but inaccurate digital circuit to the remaining bits. For example, the Low-part-OR Adder (LOA) [4] uses a simple OR gate to obtain the sum at lower bit positions and Error-Tolerant Adder I (ETAI) [27] uses a modified XOR gate for the same purpose. This kind of design has a high error rate. Also, if the bit length of the accurate part is long, the delay and power consumption of the adder are still very large. Furthermore, its relative error will be large when doing addition on small input values. In [28], the author proposed a *k*-bit lookahead approximate adder to limit the carry chain for each bit in order to reduce the critical path. However, the area of this adder is large due to the fact that the computation of each bit needs an individual carry generator. The Error-Tolerant Adder II (ETAII) [5], Error-Tolerant Adder IV (ETAIV) [29], Speculative Carry Select Adder (SCSA) [18], Accuracy-Configurable Adder (ACA) [30], Carry Skip Approximate Adder (CSAA) [6], Carry Speculative Adder [31], and Generic Accuracy Configurable Adder [7] use the block-based design to truncate the carry propagation chain. The entire adder is divided into a number of blocks. The sum of each block is computed based on a speculated carry-in signal, which is obtained from the bits before the current block. This method effectively reduces the critical path delay. However, it could introduce large relative error for the computation results, which may decrease the output quality for some applications. Furthermore, all of the approximate adders are subject to sign calculation error when doing signed addition for 2's complement numbers. Although after introducing an error reduction module, the relative error of CSAA can be reduced, it still fails to solve the problem on sign calculation. In contrast, the adder proposed in this work guarantees a small relative error. Furthermore, with the extra light-weight sign error correction module, it can always ensure the correct sign calculation.

A type of adder related to approximate adder is variable-latency adder, such as Variable Latency Carry Selection Adder (VLCSA) [18] and the adder proposed in [32]. However, it is still an accurate adder. It consists of an underlying approximate adder and an error correction module. When the output of the approximate adder is correct, the computation is finished within one clock cycle. However, when an error occurs, a second clock cycle is needed to fix the error. Its performance highly depends on the applications. For situations where the probability of an error is large, a variable-latency adder has a higher chance to need the second clock cycle, which increases the total delay significantly. In contrast, our proposed adder is still an approximate adder. Although not 100% accurate, it ensures high accuracy and the correct sign calculation within one clock cycle. Furthermore, the relaxation in the accuracy leads to smaller area and power consumption than a variable-latency adder.

3. Preliminaries on conventional adder

We discuss the preliminaries on the conventional adder in this section. Assume that the bit width of the adder is n. We use the following notations in the paper:

- $A = (a_{n-1}a_{n-2}\ldots a_0)$ and $B = (b_{n-1}b_{n-2}\ldots b_0)$ represent the two inputs of the adder, where a_i and b_i represent the i-th bits of A and B, respectively.
- s_i and c_{i+1} $(0 \le i \le n-1)$ represent the sum bit and the carry-out bit at the i-th bit position, respectively. This also means that c_i $(1 \le i \le n-1)$ is the carry-in bit at the i-th bit position. The carry-in bit at the 0-th bit position, which is a primary input, is denoted as c_0. By the rule of addition, we have

$$s_i = a_i \oplus b_i \oplus c_i, \text{ for} i = 0, \ldots, n-1. \quad (1)$$

$$c_{i+1} = a_i \cdot b_i + (a_i + b_i) \cdot c_i, \text{ for} i = 0, \ldots, n-1. \quad (2)$$

- p_i, g_i, and k_i represent the propagate, generate, and kill signal at the i-th bit position, respectively. They are defined as

$$p_i = a_i \oplus b_i,\ g_i = a_i \cdot b_i,\ k_i = \overline{a_i} \cdot \overline{b_i}. \tag{3}$$

If $g_i = 1$ ($k_i = 1$), then the i-th carry-out $c_{i+1} = 1(0)$ regardless of the value of the i-th carry-in. If $p_i = 1$, then $c_{i+1} = c_i$, which indicates the propagation of the carry-out from the $(i-1)$-th position to the i-th position.

From Eq. (3), s_i and c_i shown in Eqs. (1) and (2) can also be calculated as

$$s_i = p_i \oplus c_i, \text{ for } i = 0, \ldots, n-1. \tag{4}$$

$$c_{i+1} = g_i + p_i \cdot c_i, \text{ for } i = 0, \ldots, n-1. \tag{5}$$

Applying Eq. (5) recursively, we can compute the carry-out signal c_{i+1} as

$$c_{i+1} = g_i + g_{i-1} \cdot p_i + \cdots + g_0 \cdot \prod_{j=1}^{i} p_j + c_0 \cdot \prod_{j=0}^{i} p_j. \tag{6}$$

There are a number of different ways to realize an adder, among which the simplest form is the ripple carry adder (RCA). Fig. 1(a) shows the block diagram of one variation of RCA, where the n-bit adder is divided into a number of blocks, each with k bits. Each block is composed of a k-bit propagate/generate (P/G) signal generator, a k-bit carry generator, and a k-bit sub-adder. The number of blocks is $m = \lceil \frac{n}{k} \rceil$. In practice, we usually choose n as a multiple of k. Thus, in what follows, we assume that $n = m \cdot k$. The block indices increase from the right to the left, with the index of the rightmost block being 0.

For this block-based RCA, we use the following notations:

- $A^i_{k-1:0}$ and $B^i_{k-1:0}$ represent the two k-bit inputs of the i-th block.
- $P^i_{k-1:0}$ and $G^i_{k-1:0}$ represent the k propagate signals and the k generate signals of the i-th block, respectively.
- $S^i_{k-1:0}$ represents the k-bit partial sum of the i-th block.
- C^i_o represents the carry-out of the i-th block.
- a^i_j, b^i_j, p^i_j, g^i_j, s^i_j and c^i_{j+1} $(0 \le j \le k-1)$ represent the input bit of the first addend, the input bit of the second addend, the propagate signal, the generate signal, the sum bit, and the carry-out bit, respectively, at the j-th position in the i-th block. Note that c^i_j $(1 \le j \le k-1)$ is also the carry-in bit at the j-th bit position in the i-th block and $c^i_k = C^i_o$. The carry-in bit at the 0-th bit position in the i-th block is denoted as c^i_0.

For the adder shown in Fig. 1(a), $P^i_{k-1:0}$ and $G^i_{k-1:0}$ are produced by the P/G generator i. The carry generator i takes the propagate and generate signals of the block i and a carry-in as inputs, and produce the carry-out signal C^i_o. For $1 \le i \le m-1$, the carry-in is the carry-out from the previous block, i.e., C^{i-1}_o. For $i = 0$, the carry-in is the primary carry-in c_0. For consistency, we define $C^{-1}_o = c_0$. For any $0 \le i \le m-1$, C^i_o is obtained as

$$C^i_o = g^i_{k-1} + g^i_{k-2} p^i_{k-1} + \cdots + g^i_0 \prod_{j=1}^{k-1} p^i_j + C^{i-1}_o \prod_{j=0}^{k-1} p^i_j. \tag{7}$$

C^i_o feeds into the carry generator and the sub-adder in the $(i+1)$-th block as their carry-in signal.

The function of the i-th sub-adder is to generate the lowest k bits $S^i_{k-1:0}$ of the sum of the k-bit input $A^i_{k-1:0}$, the k-bit input $B^i_{k-1:0}$, and the carry-in bit to the sub-adder c^i_0. Note that this sum could have $(k+1)$ bits due to the generation of the carry-out. However, the sub-adder does not compute the carry-out. For example, if $A^i_{k-1:0} = (0000)_2$, $B^i_{k-1:0} = (1111)_2$, and $c^i_0 = 1$, the sub-adder will produce $S^i_{k-1:0} = (0000)_2$. For the RCA, c^i_0 equals C^{i-1}_o, the carry-out bit of the previous block.

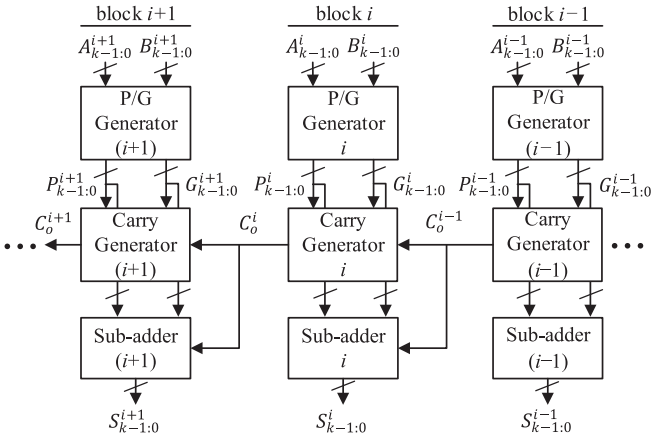
To realize its function, the sub-adder takes the propagate and generate signals of the current block and the carry-out from the previous block, C^{i-1}_o, as inputs. The logic to get the sum bit s^i_j $(0 \le j \le k-1)$ is as follows

$$s^i_j = p^i_j \oplus c^i_j, \text{ for } j = 0, \ldots, k-1, \tag{8}$$

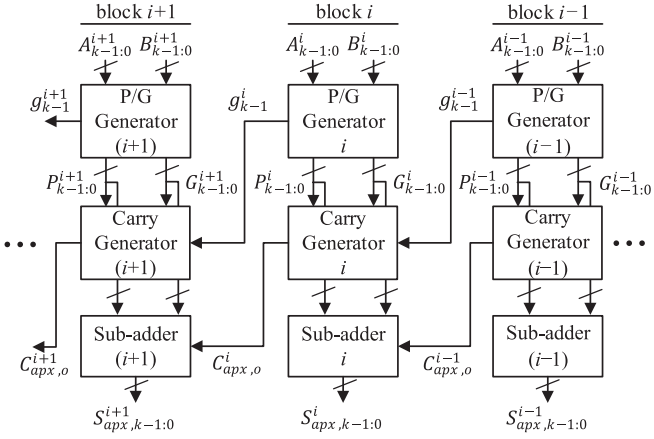
where $c^i_0 = C^{i-1}_o$ and c^i_{j+1} $(0 \le j \le k-2)$ is generated in the sub-adder internally as follows

$$c^i_{j+1} = g^i_j + p^i_j \cdot c^i_j. \tag{9}$$

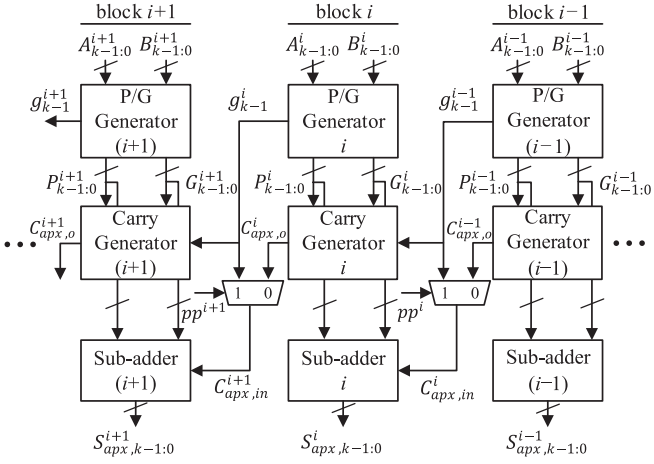
An important signal we will use later is the *block propagate* signal pp^i, which is defined as



(a) Conventional adder.



(b) Proposed approximate adder without error reduction.



(c) Proposed approximate adder with error reduction.

Fig. 1. Block diagram of adders.

$$pp^i = \prod_{j=0}^{k-1} p_j^i. \tag{10}$$

C_o^i depends on C_o^{i-1} if and only if $p_{k-1}^i = p_{k-2}^i = \cdots = p_0^i = 1$, or $pp^i = 1$. When $pp^i = 0$, C_o^i is independent of C_o^{i-1}.

4. Proposed approximate adder with low relative error

4.1. Approximate adder architecture

The longest critical path for the adder shown in Fig. 1(a) occurs when the propagate signals at all the bit positions equal 1, which makes $c_n = c_0$. Thus, to reduce the critical path delay, one effective method is to reduce the length of the carry propagation chain.

Fig. 1(b) shows the block diagram of the proposed approximate adder. It is based on the RCA shown in Fig. 1(a). To reduce the long carry chain, we introduce a carry speculation strategy to truncate the carry chain: for any $1 \le i \le m-2$, the carry-in signal of the carry generator in block i is set as the generate signal of the most significant bit in the previous block, i.e., g_{k-1}^{i-1}. The carry generator 0 has its carry-in signal remain as c_0. For consistency, we define $g_{k-1}^{-1} = c_0$. For any $0 \le i \le m-2$, the output of the carry generator in block i is now an approximation to the correct carry-out, which is denoted as $C_{apx,o}^i$. It can be obtained by replacing C_o^{i-1} in Eq. (7) with g_{k-1}^{i-1}:

$$C_{apx,o}^i = g_{k-1}^i + \cdots + g_0^i \prod_{j=1}^{k-1} p_j^i + g_{k-1}^{i-1} \prod_{j=0}^{k-1} p_j^i. \tag{11}$$

For the speculated carry-out, it has the following two properties specified by Lemmas 1 and 2, respectively.

Lemma 1. *If the speculated carry - out $C_{apx,o}^i$ produced by the i - th carry generator is 1, then the correct carry - out of the i - th block, C_o^i, must be 1.* □

Proof. By the definition of the signals and Eq. (9), we have

$$C_o^{i-1} = c_k^{i-1} = g_{k-1}^{i-1} + p_{k-1}^{i-1} c_{k-1}^{i-1}.$$

Replacing C_o^{i-1} in Eq. (7) with the above equation, we get

$$\begin{aligned} C_o^i &= g_{k-1}^i + \cdots + g_0^i \prod_{j=1}^{k-1} p_j^i + (g_{k-1}^{i-1} + p_{k-1}^{i-1} c_{k-1}^{i-1}) \prod_{j=0}^{k-1} p_j^i \\ &= C_{apx,o}^i + p_{k-1}^{i-1} c_{k-1}^{i-1} \prod_{j=0}^{k-1} p_j^i = 1 \end{aligned}$$ □

Lemma 2. *If $pp_i = 0$, then the speculated carry - out $C_{apx,o}^i$ produced by the i - th carry generator equals the correct carry - out of the i - th block, C_o^i.* □

Proof. $pp_i = 0$ means that there is no carry propagation chain from the carry-in of the i-th carry generator to its carry-out. Thus, the carry-out does not depend on the carry-in. Therefore, the speculated carry-out equals the correct carry-out. □

Given Lemma 1, we immediately get the following corollary.

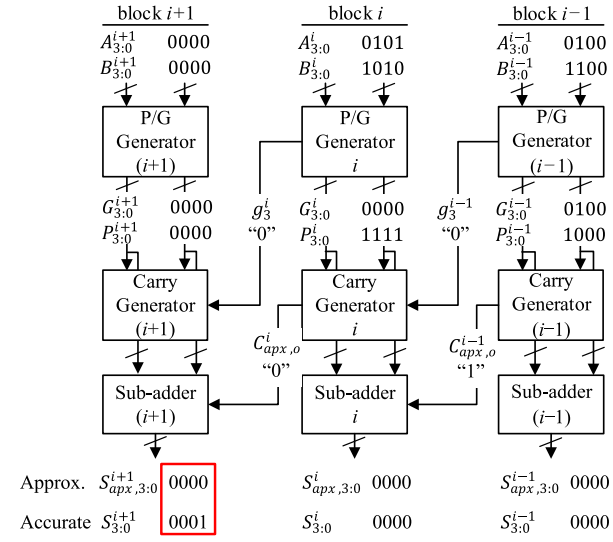
Corollary 1. *If the correct carry - out of the i - th block, C_o^i, is 0, then the speculated carry - out $C_{apx,o}^i$ produced by the i - th carry generator must be 0.* □

As shown in Fig. 1(b), for any $1 \le i \le m-1$, the sub-adder i takes the speculated carry-out $C_{apx,o}^{i-1}$ as its carry-in signal. The sub-adder 0 has its carry-in signal remain as c_0. For consistency, we define $C_{apx,o}^{-1} = c_0$. The output of the sub-adder will be a partial approximate sum, which is denoted as $S_{apx,k-1:0}^i$. We use $s_{apx,j}^i$ $(0 \le j \le k-1)$ to represent the j-th approximate sum bit in the i-th block.

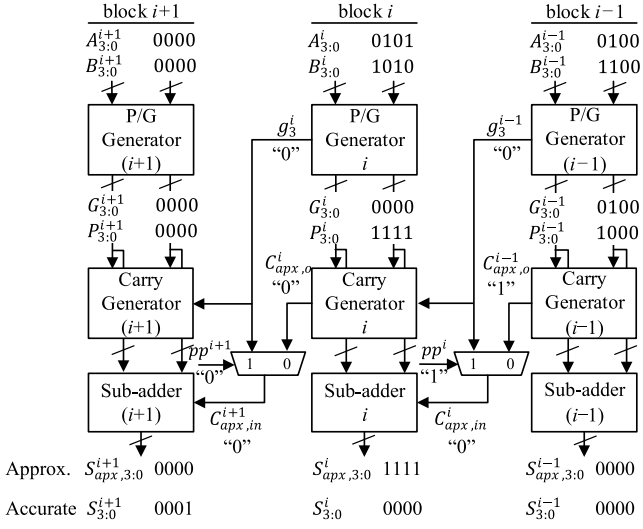
Note that in the proposed design, we omit the $(m-1)$-th carry generator in order to reduce the area. The carry-out of the entire approximate adder, denoted as $C_{apx,add}$, is produced by the $(m-1)$-th sub-adder.

The length of the critical path is thus reduced from the span of the entire m blocks to the span of 2 blocks (i.e., a sub-adder followed by a carry generator), which significantly reduces the circuit delay.

However, $C_{apx,o}^i$ is just a speculated carry-out signal of the i-th block, which could be wrong. This may lead to a large relative error.



(a) Proposed approximate adder without error reduction. It has a large relative error for the inputs shown in the figure.



(b) Proposed approximate adder with error reduction. It reduces the relative error.

Fig. 2. An example of addition error due to wrong carry speculation.

Example 1. Consider an example shown in Fig. 2(a), in which $pp^i = 1$, $g_{k-1}^{i-1} = 0$, and $C_{apx,o}^{i-1} = 1$. We further assume that all the input bits of the blocks on the left of the i-th block and on the right of the $(i-1)$-th block are all 0's. On the one hand, since there is a carry propagation chain from the carry-in bit of the i-th carry generator to its carry-out, we have $C_{apx,o}^i = g_{k-1}^{i-1} = 0$. Since the carry-in to the i-th sub-adder is $C_{apx,o}^{i-1} = 1$ and the propagate signals $p_{k-1}^i = p_{k-2}^i = \cdots = p_0^i = 1$, we have $s_{apx,j}^i = 0$, for all $j = 0, \ldots, k-1$. On the other hand, since $C_{apx,o}^{i-1} = 1$, based on Lemma 1, the correct carry-in to the i-th block $C_o^{i-1} = 1$. Thus, the correct carry-out of the i-th block is $C_o^i = 1$ and the sum bits are $s_j^i = 0$, for all $j = 0, \ldots, k-1$. Since all the input bits of the blocks on the left of the i-th block and on the right of the $(i-1)$-th block are all 0's, the approximate sum is $(00\ldots0)_2$. The corresponding accurate sum is $(00\ldots0100\ldots0)_2$, where the only 1 is at the 0-th bit of the $(i+1)$-th

block. This leads to a relative error of 100%. □

To reduce the large relative error, we further introduce an error reduction module into the approximate adder in Fig. 1(b). The modified approximate adder is shown in Fig. 1(c), where we only insert a 2-to-1 multiplexer between block i and block $(i-1)$, for all $i = 1, 2, \ldots, m-2$. The two data inputs of the multiplexer are the generate signal g_{k-1}^{i-1} in the previous block and the speculated carry-out signal produced by the carry generator in the previous block, $C_{apx,o}^{i-1}$. The selection signal is the block propagate signal pp^i. Since pp^i has already been generated in the carry generator of block i (as shown in Eq. (11)), we do not need to include additional circuits to produce it. If $pp^i = 1$, the output of the multiplexer is chosen as g_{k-1}^{i-1}; otherwise, it is chosen as $C_{apx,o}^{i-1}$. The output signal of the multiplexer is used as the carry-in signal of the sub-adder of the current block, which is denoted as $C_{apx,in}^{i}$. Note that since the $(m-1)$-th sub-adder produces both the approximate sum and the carry out, it can be viewed as a sub-adder of length $(k+1)$ with two $(k+1)$-bit inputs $a_n a_{n-1}\ldots a_{n-k}$ and $b_n b_{n-1}\ldots b_{n-k}$ where $a_n = b_n = 0$. Under this model, since the propagate signal of bit n is $p_n = a_n \oplus b_n = 0$, the block propagate signal of block $(m-1)$ is

$$pp^{m-1} = \prod_{j=0}^{k} p_{n-j} = 0. \tag{12}$$

As a result, the multiplexer between block $(m-1)$ and block $(m-2)$ always chooses $C_{apx,o}^{m-2}$ as the carry-in signal to the $(m-1)$-th sub-adder. Therefore, that multiplexer is omitted.

As can be seen, this modification causes nearly no overhead to the original approximate adder. In the following subsections, we will first demonstrate that the proposed adder has a very small relative error. Then, we will show that another error metric, error rate, is also small. Finally, we will analyze its delay.

4.2. Relative error analysis

In this section, we will analyze the relative error of the approximate adder with the error reduction module. The relative error is defined as

$$E_{re} = \left|\frac{S - S_{apx}}{S}\right|, \tag{13}$$

where S and S_{apx} are the correct and approximate sums for the given inputs, respectively.

First, consider the case we introduced in Example 1, which results in a large relative error for the original approximate adder.

Example 2. For the case we introduced in Example 1, the computation result of the modified adder is shown in Fig. 2(b). Given $pp^i = 1$, the carry-in to the i-th sub-adder is chosen as $g_{k-1}^{i-1} = 0$. Thus, the approximate sum bit $s_{apx,j}^{i} = 1$, for all $j = 0, \ldots, k-1$. Given $pp^{i+1} = 0$, the carry-in to the $(i+1)$-th sub-adder is selected as $C_{apx,o}^{i} = 0$. Since the input bits to the blocks on the left of block i and on the right of block $(i-1)$ are all 0's, the approximate sum is $(00\ldots011\ldots100\ldots0)_2$ (k 1's at block i). This is very close to the correct sum $(00\ldots0100\ldots0)_2$ (the only 1 is at the 0-th bit of the $(i+1)$-th block). The relative error is significantly reduced, i.e., from 100% to $\frac{1}{2^k}$. □

Indeed, for all possible input combinations, the relative error of our design is bounded by $\frac{1}{2^k}$. We prove this claim in the rest of this subsection.

For the purpose of proving, we add back the multiplexer in between block $(m-1)$ and block $(m-2)$, which selects either g_{k-1}^{m-2} or $C_{apx,o}^{m-2}$ as the speculated carry-in $C_{apx,in}^{m-1}$ to the $(m-1)$-th sub-adder. The selecting input of the multiplexer is pp^{m-1}, which, as shown by Eq. (12), is 0. Consequently, $C_{apx,in}^{m-1}$ is always chosen as $C_{apx,o}^{m-2}$.

Now, we consider an arbitrarily fixed input combination. We first partition all m blocks of the input into groups. Each group consists of two sequences of blocks, where all the blocks in the left sequence have their block propagate signals as 0 and all the blocks in the right sequence have their block propagate signals as 1. Fig. 3 gives an example of 8 blocks divided into 3 groups. Note that for the rightmost group, it is possible that all of its blocks have their block propagate signals as 0.

Block:	7	6	5	4	3	2	1	0
pp:	0	0	1	0	0	1	1	0
Group:	III			II				I

Fig. 3. Partition the blocks in the approximate adder into groups.

First, we have the following claim.

Theorem 1. *For any groupsuppose its correct sum and approximate sum are SG and SG_{apx}, respectively. Then, we have* $0 \le SG - SG_{apx} \le \frac{1}{2^k} SG$. □

Proof. See Appendix A. □

We use the following example to demonstrate the correctness of Theorem 1.

Example 3. Consider the case shown in Example 2. Given the input, we have $pp^{m-1} = \cdots = pp^{i+1} = 0$, $pp^i = 1$, and $pp^{i-1} = 0$. Therefore, the blocks $m-1, \ldots, i$ form a group. By the discussion in Example 2, the approximate sum and the correct sum for this group are $SG_{apx} = (00\ldots011\ldots1)_2$ (k 1's in total) and $SG = (00\ldots0100\ldots0)_2$ (k 0's on the right of 1), respectively. Therefore, we have $SG - SG_{apx} = (00\ldots01)_2$. We conclude that $0 \le SG - SG_{apx} \le \frac{1}{2^k} SG$. □

Now, suppose the m blocks are partitioned into l groups and the group i $(0 \le i \le l-1)$ ends at block d_i. For group i, denote its correct sum as SG_i and its approximate sum as $SG_{apx,i}$. Then, the correct sum of the entire adder is $S = \sum_{i=0}^{l-1} SG_i 2^{d_i k}$. The approximate sum is $S_{apx} = \sum_{i=0}^{l-1} SG_{apx,i} 2^{d_i k}$. By Theorem 1, we have $0 \le SG_i - SG_{apx,i} \le \frac{1}{2^k} SG_i$, for all $0 \le i \le l-1$. Thus, the relative error for the approximate adder is

$$E_{re} = \left|\frac{S - S_{apx}}{S}\right| = \frac{\sum_{i=0}^{l-1} (SG_i - SG_{apx,i}) 2^{d_i k}}{\sum_{i=0}^{l-1} SG_i 2^{d_i k}} \le \frac{\sum_{i=0}^{l-1} \frac{1}{2^k} SG_i 2^{d_i k}}{\sum_{i=0}^{l-1} SG_i 2^{d_i k}} = \frac{1}{2^k}.$$

Thus, the relative error of our proposed design is bounded by $\frac{1}{2^k}$, which is small for a moderate block size k.

4.3. Error rate analysis

Besides relative error, another commonly-used error metric is error rate, which is the ratio of the number of input vectors that produce incorrect outputs to the total number of input vectors [3]. For some applications, error rate may be also important. For example, peak signal-to-noise ratio (PSNR), which is a commonly used quality metric in image and video processing applications, is affected by both error rate and relative error [33]. We analyze the error rate of our proposed adder in this section. We assume that the inputs are uniformly distributed.

Instead of directly calculating the error rate $Pr(E)$, we calculate the probability of getting a correct result, $Pr(C)$, from which we can obtain $Pr(E) = 1 - Pr(C)$. First, we define the following three events:

1. U_i: the approximate carry-out of block i, $C_{apx,o}^{i}$, is equal to the correct carry-out C_o^i, and for all $0 \le j \le i$, the approximate sum of block j, $S_{apx,k-1:0}^{j}$, is correct.
2. V_i: $g_{k-1}^{i} = 0$, the correct carry-out $C_o^i = 0$, and for all $0 \le j \le i$, the approximate sum of block j is correct.

3. W_i: $g^i_{k-1} = 1$, the correct carry-out $C^i_o = 1$, and for all $0 \le j \le i$, the approximate sum of block j is correct.

Denote $Pr(U_i)$, $Pr(V_i)$, and $Pr(W_i)$ as u_i, v_i, and w_i, respectively. The result of the approximate adder is correct if and only if for all $0 \le j \le m-1$, the approximate sum of block j is correct. Moreover, since the carry-in to the $(m-1)$-th sub-adder is $C^{m-2}_{apx,o}$, the approximate sum of block $(m-1)$ is correct if and only if $C^{m-2}_{apx,o}$ is correct. Therefore, the result of the approximate adder is correct if and only if the event U_{m-2} happens. Thus, $Pr(C) = u_{m-2}$.

First, consider when the event U_i happens. We divide the event into two cases, based on whether $pp^i = 0$ or 1, since this determines the speculated carry-in to the i-th sub-adder.

1. $pp^i = 0$. Then, $C^i_{apx,o}$ equals the correct carry-out C^i_o by Lemma 2. Moreover, because $pp^i = 0$, the carry-in to the i-th sub-adder is chosen as $C^{i-1}_{apx,o}$. Thus, the correctness of the partial sum at block i requires that $C^{i-1}_{apx,o} = C^{i-1}_o$. Thus, under the condition that $pp^i = 0$, the event U_i occurs if and only if $C^{i-1}_{apx,o} = C^{i-1}_o$ and for all $0 \le j \le i-1$, the approximate sum of block j is correct. In other words, the event U_i occurs if and only if the event U_{i-1} occurs.
2. $pp^i = 1$. Then, $C^i_{apx,o} = g^{i-1}_{k-1}$, due to carry propagation. At the same time, the correct carry-out satisfies that $C^i_o = C^{i-1}_o$. Thus, $C^i_{apx,o} = C^i_o$ requires that $g^{i-1}_{k-1} = C^{i-1}_o$. Note that when $pp^i = 1$, the carry-in to the i-th sub-adder is g^{i-1}_{k-1}. Thus, $g^{i-1}_{k-1} = C^{i-1}_o$ will also make the approximate sum at block i correct. Therefore, under the condition that $pp^i = 1$, the event U_i occurs if and only if $g^{i-1}_{k-1} = C^{i-1}_o$ and for all $0 \le j \le i-1$, the approximate sum of block j is correct. In other words, the event U_i occurs if and only if either the event V_{i-1} or the event W_{i-1} occurs.

From the above two cases, we have

$$u_i = Pr(pp^i = 0)\cdot u_{i-1} + Pr(pp^i = 1)\cdot(v_{i-1} + w_{i-1}). \tag{14}$$

Now, consider when the event V_i happens. We still divide the event into two cases, based on whether $pp^i = 0$ or 1.

1. $pp^i = 0$. Then, $C^{i-1}_{apx,o}$ is chosen to be the carry-in to the i-th sub-adder. To ensure the correctness of the partial sum at block i, we require $C^{i-1}_{apx,o} = C^{i-1}_o$. Furthermore, we require that for all $0 \le j \le i-1$, the approximate sum of block j is correct. Thus, the event U_{i-1} must occur. Moreover, since $C^i_o = 0$ and $pp^i = 0$, there must exist a kill signal in block i which propagates all the way to the most significant bit position of block i. Thus, we require one of the following events to occur: $k^i_{k-1} = 1$, $p^i_{k-1} = k^i_{k-2} = 1$, …, or $p^i_{k-1} = \cdots = p^i_1 = k^i_0 = 1$. Note that one of the above events occurring also guarantees that $g^i_{k-1} = 0$.
2. $pp^i = 1$. This ensures that $g^i_{k-1} = 0$. Since $pp^i = 1$, we have $C^i_o = C^{i-1}_o$. Thus, $C^i_o = 0$ requires that $C^{i-1}_o = 0$. Moreover, since the carry-in to the i-th sub-adder is g^{i-1}_{k-1} and the approximate sum at block i is correct, we require that g^{i-1}_{k-1} is equal to the correct carry-in C^{i-1}_o. Thus, $g^{i-1}_{k-1} = C^{i-1}_o = 0$. Thus, under the condition that $pp^i = 1$, the event V_i occurs if and only if $g^{i-1}_{k-1} = C^{i-1}_o = 0$ and for all $0 \le j \le i-1$, the approximate sum of block j is correct. In other words, the event V_i occurs if and only if the event V_{i-1} occurs.

From the above two cases, we have

$$\begin{aligned} v_i = {} & [Pr(k^i_{k-1} = 1) + \cdots + Pr(p^i_{k-1} = \cdots = p^i_1 = k^i_0 = 1)]\cdot u_{i-1} \\ & + Pr(pp^i = 1)\cdot v_{i-1}. \end{aligned} \tag{15}$$

Finally, we consider when the event W_i occurs. One requirement is $g^i_{k-1} = 1$. If $g^i_{k-1} = 1$, then we guarantee that $C^i_o = 1$. Moreover, when $g^i_{k-1} = 1$, pp^i must be 0. Thus, the carry-in to the i-th sub-adder is chosen to be $C^{i-1}_{apx,o}$. To ensure the correctness of the partial sum at block i, we require that $C^{i-1}_{apx,o} = C^{i-1}_o$. Also, we require that for all $0 \le j \le i-1$, the approximate sum of block j is correct. In summary, the event W_i occurs if and only if $g^i_{k-1} = 1$ and the event U_{i-1} occurs. Thus, we have

$$w_i = Pr(g^i_{k-1} = 1)\cdot u_{i-1}. \tag{16}$$

Eqs. (14), (15) and (16) give us a recursive way to obtain u_i, v_i and w_i. To eventually obtain these values, we only need to get the values u_0, v_0 and w_0, corresponding to the base case.

To get the base case values, we analyze block 0. The carry-ins to both the carry generator and the sub-adder in block 0 are the primary carry-in c_0, so the speculated carry-out $C^0_{apx,o}$ and the partial approximate sum $S^0_{apx,k-1:0}$ are always correct. It means that the event U_0 always happens. Thus, $u_0 = 1$. The event V_0 happens if and only if $g^0_{k-1} = C^0_o = 0$. Since $C^0_o = 0$ indicates that $g^0_{k-1} = 0$, we have $v_0 = Pr(C^0_o = 0)$. Since the inputs are uniformly distributed, we have $v_0 = Pr(C^0_o = 0) = \frac{1}{2}$. The event W_0 happens if and only if $g^0_{k-1} = C^0_o = 1$. Since $g^0_{k-1} = 1$ indicates that $C^0_o = 1$, we have $w_0 = Pr(g^0_{k-1} = 1) = \frac{1}{4}$.

Under the assumption that the inputs are uniformly distributed, the full set of recursive equations is:

$$\begin{aligned} u_0 = {} & 1,\ v_0 = \frac{1}{2},\ w_0 = \frac{1}{4}; \\ u_i = {} & \left(1 - \frac{1}{2^k}\right)u_{i-1} + \frac{1}{2^k}(v_{i-1} + w_{i-1}),\ 0 < i < m; \\ v_i = {} & \frac{1}{2}\left(1 - \frac{1}{2^k}\right)u_{i-1} + \frac{1}{2^k}v_{i-1},\ 0 < i < m; \\ w_i = {} & \frac{1}{4}u_{i-1},\ 0 < i < m. \end{aligned}$$

Finally, we can get the error rate as

$$Pr(E) = 1 - Pr(C) = 1 - u_{m-2}.$$

The error rates of the proposed approximate adders with different n and k will be shown in Section 6.2.

4.4. Delay analysis

We analyze the delay of the proposed approximate adder in this section. As shown in Fig. 1(c), the propagate signals and generate signals are first produced by the P/G generators and delivered to the carry generators and the sub-adders. Next, all the carry generators simultaneously produce the speculated carry-out signals $C^i_{apx,o}$ with the speculated carry-in signal as g^{i-1}_{k-1}. Then, the speculated carry-in signals to the sub-adders, $C^i_{apx,in}$, are chosen from the two sets of signals $C^{i-1}_{apx,o}$ and g^{i-1}_{k-1} by the multiplexers based on block propagate signals pp^i. Finally, the sub-adders take the speculated carry-in signals $C^i_{apx,in}$ to compute the approximate sums $S^i_{apx,k-1:0}$. Thus, the critical path delay of our proposed approximate adder is

$$t_{apx} = t_{PG} + t_{CG} + t_{MUX} + t_{SA},$$

where t_{PG}, t_{CG}, t_{MUX}, and t_{SA} are the delays of the P/G generator, carry generator, multiplexer, and sub-adder, respectively. Note that the sub-adder can be implemented by any kind of conventional adders like RCA or carry-lookahead adder (CLA). If RCA is chosen, then the approximate adder has small area and power consumption. The asymptotic upper bounds for the delay values are

$$\begin{aligned} t_{PG} = {} & O(1),\ t_{CG} = O(\log k),\ t_{MUX} = O(1),\ t_{SA} = O(k), \\ t_{apx} = {} & O(k). \end{aligned}$$

On the other hand, if CLA is used as the sub-adders, then both t_{SA} and t_{apx} will be reduced to $O(\log k)$, but the area cost and power consumption will increase.

5. Proposed sign-correct approximate adder

The approximate adder introduced in the previous section is targeted for unsigned addition. However, there exist some applications where 2's complement signed additions are required. Most existing approximate adders are not targeted for signed addition, and hence could generate incorrect sign bit when applied for signed additions, leading to catastrophic consequences. In our work, based on the proposed low-relative-error approximate adder, we further introduce a lightweight sign correction module to solve the potential sign error in the approximate addition. Our basic idea of fixing the sign error is to detect the situation when the error occurs and then correct the leading bits that are wrong.

5.1. Sign correction module

In 2's complement signed addition, the input A (B) has an extra bit a_n (b_n) at the beginning, which denotes the sign. The inputs to the leftmost block (i.e., block $(m-1)$) have $(k+1)$ bits, which are $a_n \ldots a_{n-k}$ and $b_n \ldots b_{n-k}$[1]. The carry-in to the sign bit is produced by the leftmost sub-adder based on the inputs $a_{n-1} \ldots a_{n-k}$, $b_{n-1} \ldots b_{n-k}$, and the speculated carry-in to that sub-adder, i.e., $C^{m-2}_{apx,o}$. We denote the carry-in to the sign bit as $C_{apx,sign}$. The correct carry-in to the sign bit is denoted as C_{sign}. The sign bit is correct if and only if $C_{apx,sign} = C_{sign}$.

First, we show an example where a sign error occurs when we perform a 2's complement signed addition using the approximate adder shown in Fig. 1(c).

Example 4. Consider the example shown in Fig. 4, in which $m = 3$. The two signed inputs, represented as decimal numbers, are 85 and −84. The correct sum should be 1. However, the result of our approximate adder is −15, which is quite different from the correct one.□The above example shows that developing an additional module to correct the sign error is necessary.

Next, we study the general situation under which a sign error occurs for the proposed approximate adder. We define the block propagate signal of the leftmost block (i.e., block $(m-1)$) as $pp^{m-1} = \prod_{j=1}^{k} P_{n-j}$, which does not include the propagate signal at the sign bit. The block propagate signals of the remaining blocks are defined as before.

The following theorem gives a **necessary condition** for a sign error to occur.

Theorem 2. *If a given input causes a sign errorthen there must exist an* $1 \le i \le m-1$ *such that* $pp^{m-1} = \cdots = pp^i = 1$ *and* $C^{i-1}_{apx,o} = 1$. □

Proof. See Appendix B. □

We use the following example to illustrate Theorem 2.

Example 5. Consider the case shown in Example 4, which has a sign error. By the inputs shown in Fig. 4, it satisfies that $pp^{m-1} = pp^{m-2} = 1$ and $C^{m-3}_{apx,o} = 1$, where $m = 3$. □

Now, consider an input for which there exists an $1 \le i \le m-1$ such that $pp^{m-1} = \cdots = pp^i = 1$ and $C^{i-1}_{apx,o} = 1$. Because Theorem 2 only gives a necessary condition for the existence of a sign error, this input may or may not cause a sign error. However, no matter whether there is a sign error or not, the correct sign value can be determined for this input. Because $C^{i-1}_{apx,o} = 1$, by Lemma 1, the correct carry-out $C^{i-1}_{o} = 1$. Given that $pp^{m-1} = \cdots = pp^i = 1$, the correct carry-in to the sign bit is $C_{sign} = 1$ and the correct partial sums $S^{m-1}_{k-1:0}$, …, $S^{i}_{k-1:0}$ are all $(00\ldots0)_2$ (k 0's in total). This indicates a way to correct sign error.

First, we define the following signal for any $1 \le i \le m-1$:

$$sp^i = pp^{m-1} \cdot \cdots \cdot pp^i \cdot C^{i-1}_{apx,o}, \tag{17}$$

which can be realized by an $(m-i+1)$-input AND gate, as shown in Fig. 5(a). $sp^i = 1$ indicates that the input satisfying that $pp^{m-1} = \cdots = pp^i = 1$ and $C^{i-1}_{apx,o} = 1$. We further define

$$CS^i = sp^1 + \cdots + sp^i,\ 1 \le i \le m-1, \tag{18}$$

which can be realized by an i-input OR gate, as shown in Fig. 5(b).

If $CS^{m-1} = 1$, then there exists an $1 \le i \le m-1$ such that $pp^{m-1} = \cdots = pp^i = 1$ and $C^{i-1}_{apx,o} = 1$. Thus, we should set the speculated carry-in to the sign bit to 1. Otherwise, the original speculated carry-in is correct and we just keep it. Therefore, we modify the speculated carry-in to the sign bit as follows:

$$C_{sapx,sign} = CS^{m-1} + C_{apx,sign},$$

where $C_{sapx,sign}$ denotes the modified carry-in to the sign bit, which can be obtained by an OR gate as shown in Fig. 5(c).

Furthermore, to reduce the error magnitude, if $sp^i = 1$, we also set the partial approximate sums at blocks $m-1$, $m-2$, …, i to $(00\ldots0)_2$ (k 0's in total). Correspondingly, the partial approximate sum at block i ($1 \le i \le m-1$) is set to $(00\ldots0)_2$ if any of sp^1, …, sp^i is 1, which is equivalent to $CS^i = 1$. If $CS^i = 0$, we just keep the original partial approximate sum at block i. Thus, we modify the approximate sum bit as follows:

$$s^i_{sapx,j} = s^i_{apx,j} \cdot \overline{CS^i},\ 0 \le j \le k-1,\ 1 \le i \le m-1, \tag{19}$$

where $s^i_{sapx,j}$ denotes the modified sum bit at the j-th position in block i, which can be obtained by an AND gate as shown in Fig. 5(c). Note that the partial approximate sum at block 0 is not changed, i.e., $s^0_{sapx,j} = s^0_{apx,j}$ for all $0 \le j \le k-1$.

By the above construction, the sign correction module ensures the result of the approximate addition has no sign error. Next, we will discuss the relative error and error rate of this modified adder.

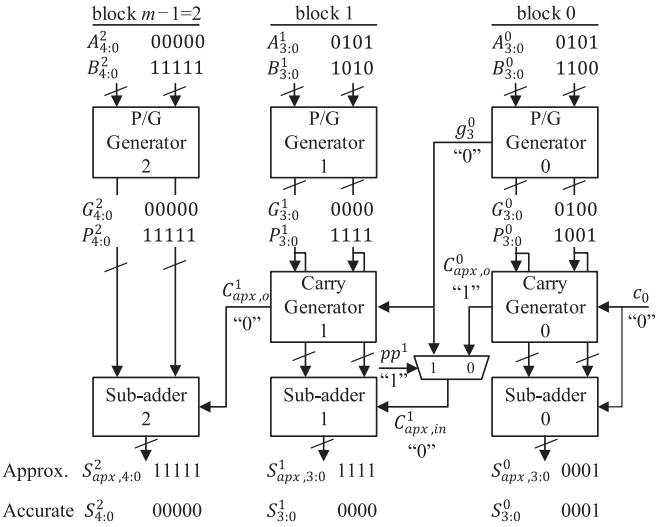


Fig. 4. Example that the proposed approximate adder produces an incorrect sign bit.

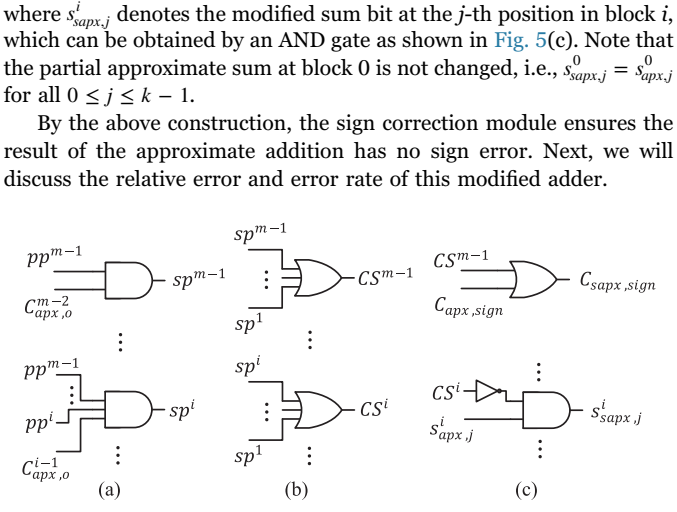


Fig. 5. Circuits of the sign error correction module.

[1] Different from unsigned addition, in signed addition, the carry-out of the entire adder is ignored.

5.2. Relative error analysis

We first ignore the sign bit and focus on the sum composed of all the remaining output bits of the approximate adder. Given an arbitrarily fixed input combination, based on the block propagate signals pp^i, we partition all the blocks into groups as defined in Section 4.2. Suppose the m blocks are partitioned into l groups and the group i $(0 \le i \le l-1)$ ends at block d_i. Since the sign-correct approximate adder is modified from the proposed approximate adder without sign correction, our analysis idea is to connect the former to the latter. For group i, we denote its correct sum as SG_i, its approximate sum produced by the proposed approximate adder without sign correction as $SG_{apx,i}$, and its approximate sum produced by the sign-correct approximate adder as $SG_{sapx,i}$.

First, for groups other than the leftmost one, we have the following claim.

Lemma 3. *For any* $0 \le i \le l-2$, $SG_{sapx,i} = SG_{apx,i}$. □

Proof. See Appendix C. □

For the leftmost group, we have the following claim.

Lemma 4. *If* $pp^{m-1} = 0$, *then* $SG_{sapx,l-1} = SG_{apx,l-1}$. *If* $pp^{m-1} = 1$, *then* $SG_{sapx,l-1} = SG_{l-1}$. □

Proof. See Appendix D. □

Based on Lemmas 3 and 4 and Theorem 1, we have the following claim.

Theorem 3. *For any group* $0 \le i \le l-1$, $0 \le SG_i - SG_{sapx,i} \le \frac{1}{2^k} SG_i$. □

If the sign bit is 0, then the final sum $S_{spax} = \sum_{i=0}^{l-1} SG_{sapx,i} 2^{d_i k}$. Given Theorem 3, by the same argument as we used in Section 4.2, the relative error of the approximate sum is bounded by $\frac{1}{2^k}$.

Now, consider the case where the sign bit is 1. The correct sum is $S = \sum_{i=0}^{l-1} SG_i 2^{d_i k} - 2^n$, and the approximate sum is $S_{sapx} = \sum_{i=0}^{l-1} SG_{sapx,i} 2^{d_i k} - 2^n$. The absolute error is

$$|S - S_{sapx}| = \sum_{i=0}^{l-1} (SG_i - SG_{sapx,i}) 2^{d_i k}. \tag{20}$$

Define $d_l = m$. For $0 \le i \le l-1$, define $SG'_i = 2^{(d_{i+1}-d_i)k} - 1 - SG_i$. Note that SG'_i is equivalent to the bit-wise negation of SG_i. For the magnitude of S, we have

$$\begin{aligned} |S| &= 2^n - \sum_{i=0}^{l-1} SG_i 2^{d_i k} \\ &= \sum_{i=0}^{l-1} (2^{d_{i+1}k} - 2^{d_i k} - SG_i 2^{d_i k}) + 2^{d_0 k} \\ &= \sum_{i=0}^{l-1} SG'_i 2^{d_i k} + 1. \end{aligned} \tag{21}$$

To further derive the relative error, we will use the following claim, which bounds $(SG_i - SG_{sapx,i})$ using SG'.

Lemma 5. *For any* $0 \le i \le l-1$, $SG_i - SG_{sapx,i} \le \frac{1}{2^k - 1} SG'_i$. □

Proof. See Appendix E. □

When the sign bit is 1, based on Eqs. (20), (21), and Lemma 5, the relative error of the approximate sum is

$$\begin{aligned} E_{re} = \frac{|S - S_{sapx}|}{|S|} &= \frac{\sum_{i=0}^{l-1} (SG_i - SG_{sapx,i}) 2^{d_i k}}{\sum_{i=0}^{l-1} SG'_i 2^{d_i k} + 1} \\ &< \frac{\sum_{i=0}^{l-1} \frac{1}{2^k - 1} SG'_i 2^{d_i k}}{\sum_{i=0}^{l-1} SG'_i 2^{d_i k}} = \frac{1}{2^k - 1} \end{aligned}$$

By the above analysis, the relative error of the sign-correct approximate adder is bounded by $\frac{1}{2^k - 1}$, which is close to that of the proposed approximate adder without sign correction.

5.3. Error rate analysis

In this section, we analyze the error rate of the sign-correct approximate adder. We assume the inputs are uniformly distributed. First, we have the following claim.

Theorem 4. *The error rate of the sign - correct approximate adder is smaller than that of the approximate adder without sign correction.* □

Proof. See Appendix F. □

Next, we will calculate the exact error rate for the sign-correct approximate adder. We consider when the inputs produce the correct sum. We distinguish all the inputs into the following three cases.

1. $pp^{m-1} = 0$. In this case, by Lemmas 3 and 4, we have $SG_{sapx,i} = SG_{apx,i}$ for all $0 \le i \le l-1$. Therefore, the sum of the sign-correct approximate adder is correct if and only if that of the approximate adder without sign correction is correct, which happens when the event U_{m-2} occurs by our discussion in Section 4.3.
2. $pp^{m-1} = \cdots = pp^t = 1$ and $pp^{t-1} = 0$, where $2 \le t \le m-1$. In this case, by Lemmas 3 and 4, we have $SG_{sapx,l-1} = SG_{l-1}$ and for all $0 \le i \le l-2$, $SG_{sapx,i} = SG_{apx,i}$. Therefore, the sum of the sign-correct approximate adder is correct if and only if for the approximate adder without sign correction, $SG_{apx,i} = SG_i$ for all $0 \le i \le l-2$. In this case, the group $l-2$ begins at block $t-1$. Thus, we require that for all $0 \le j \le t-1$, the sum of block j of the approximate adder without sign correction is correct. Since $pp^{t-1} = 0$, we have $C^{t-1}_{apx,in} = C^{t-2}_{apx,o}$. Therefore, an equivalent condition is that $C^{t-2}_{apx,o}$ is equal to the correct carry-out C^{t-2}_o and for all $0 \le j \le t-2$, the sum of block j of the approximate adder without sign correction is correct. In other words, the event U_{l-2} defined in Section 4.3 must occur.
3. $pp^{m-1} = \cdots = pp^1 = 1$. In this case, blocks $m-1$, $m-2$, …, 1 belong to group $(l-1)$. By Lemma 4, $SG_{sapx,l-1} = SG_{l-1}$. As a result, the sum from block $m-1$ to block 1 of the sign-correct approximate adder is correct. Furthermore, by our design, the sum at block 0 is always correct. Therefore, in this case, the sum of the sign-correct approximate adder is always correct.

By the above discussion, we conclude that the probability of the sign-correct approximate adder to produce a correct output is

$$\begin{aligned} Pr(C) = {} & Pr(pp^{m-1} = 0) u_{m-2} \\ & + \sum_{i=2}^{m-1} \left(\prod_{k=i}^{m-1} Pr(pp^k = 1) \right) Pr(pp^{i-1} = 0) u_{i-2} \\ & + \prod_{k=1}^{m-1} Pr(pp^k = 1). \\ = {} & \sum_{i=2}^{m} \left(\frac{1}{2^k}\right)^{m-i} \left(1 - \frac{1}{2^k}\right) u_{i-2} + \left(\frac{1}{2^k}\right)^{m-1}, \end{aligned}$$

where u_i's are obtained in Section 4.3. The error rate is

$Pr(E) = 1 - Pr(C)$. The error rates of the sign-correct approximate adders with different n and k will be shown in Section 6.2.

6. Experimental results

We evaluate our proposed approximate adders with and without sign correction in this section. They were designed in Verilog HDL, synthesized with a NANGATE 45 nm cell library [34] using Synopsys Design Compiler [35].

6.1. Comparison of the proposed adders with other adders

In this section, we compared our proposed adders with other adders, including two conventional adders (RCA and Kogge-Stone-based CLA) and six other approximate adders (LOA [4], ETAII [5], SCSA [18], ACA [30], LUA [28], and CSAA [6]) for a 32-bit addition. We conducted two sets of experiments. In the first set of experiments, we chose the block size $k = 4$ for our proposed approximate adders, while in the second set, we chose $k = 8$. For each set of experiments, the block sizes for ETAII, SCSA, and CSAA were also chosen as k, the block size of the proposed approximate adders. For LOA, we set its accurate and inaccurate segments to both have 16 bits. For ACA, we used $2k$-bit sub-adders, to make it have an equivalent block size of k. For LUA, we set the look-ahead as k bits. The same designs of the k-bit carry-lookahead module were used as the carry generators in the ETAII, SCSA, LUA, CSAA, and our proposed adders. For SCSA, ACA, LUA, and CSAA, their sub-adders were implemented as RCAs. For LOA, we implemented two versions, one with its accurate part as RCA and the other as Kogge-Stone-based CLA. We also implemented two versions for ETAII and our proposed adders with and without sign correction module. The first version of each adder has the sub-adders as RCAs and the second version as Kogge-Stone-based CLAs. For ACA, it has its own error detection and correction modules, but such modules incur additional area overhead and require an additional clock cycle to correct the error. Thus, we did not include such modules in our implementation of ACA. We compared seven metrics of our proposed adders to those of the other adders, which are area, delay, power consumption, power-delay product, error rate, maximal relative error, and the capability to ensure the correct sign calculation.

The results for $k = 4$ are listed in Table 1. For the proposed adder with or without sign correction module, the implementation with RCAs as the sub-adders has smaller area and power consumption but larger delay than the implementation with CLAs as the sub-adders, which is expected. However, due to the small size of the sub-adder (i.e., of only 4 bits), their differences in area, delay, and power consumption are not large. As a result, these two implementations have close power-delay product (PDP).

Compared to RCA, the proposed RCA-based approximate adder without sign correction module has smaller area, delay, and power consumption, while the proposed one with sign correction module has larger area, but smaller delay and power consumption. In terms of PDP, the proposed RCA-based approximate adders without and with sign correction are smaller than RCA by 76% and 66%, respectively. Compared to CLA, the proposed RCA-based approximate adders with and without sign correction module have slightly larger delay, but much smaller area and power consumption. In terms of PDP, the proposed RCA-based adders without and with sign correction are smaller than CLA by 51% and 32%, respectively. Similar conclusions can be drawn for the CLA-based implementation of the proposed adders.

Compared with the other approximate adders, our adders also show advantages. In terms of area, our proposed RCA-based approximate adder without sign correction is only inferior to LOA and ETAII; it is superior to SCSA, ACA, LUA, and CSAA. In terms of PDP, it is better than RCA-based LOA, SCSA, and ACA. The PDP of our proposed RCA-based adder with sign correction is larger than all the other existing approximate adders except ACA, due to the extra hardware for correcting sign bit error. However, it can ensure the correct sign calculation for 2's complement signed addition, while the other approximate adders cannot. The error rates of our adders are only larger than CSAA among all the other approximate adders. It is because CSAA looks ahead two blocks for carry speculation, while ours look ahead only one block. All the other approximate adders have large maximal relative error. In contrast, the maximal relative errors of our designs are limited to a small value. According to our analysis, with $k = 4$, the maximal relative error is $\frac{1}{2^k} = 6.25\%$ for the proposed approximate adder without sign correction and $\frac{1}{2^k - 1} = 6.67\%$ for the proposed sign-correct approximate adder.

To give a clearer comparison, we also plot the power-delay product (PDP) versus error rate and the PDP versus maximum relative error for each adder in Table 1 in Fig. 6(a) and Fig. 6(b), respectively. A Pareto optimal curve is identified in each figure.

From Fig. 6(a), we can see that the proposed approximate adder without sign correction module (either RCA-based or CLA-based) is close to the Pareto optimal PDP-versus-error-rate curve, while the one with sign correction module (either RCA-based or CLA-based) deviates farther from the curve. Since the main purpose of our proposed adders is to reduce the relative error and to ensure the correct sign calculation, their relative positions to the Pareto optimal curve are satisfying.

From Fig. 6(b), we can see that our CLA-based approximate adder without sign correction module is on the Pareto optimal PDP-versus-

Table 1
Comparison of the proposed adders with other 32-bit adders ($k = 4$).

Adders	Area (μm^2)	Delay (ns)	Power (μW)	Power- delay product (fJ)	Error rate (%)	Max relative error (%)	Sign always correct
RCA	254.03	4.04	176	711.04	0	0	Yes
CLA	700.11	1.54	229	352.66	0	0	Yes
LOA(RCA)	127.95	2.37	87	206.90	99	50	No
LOA(CLA)	192.32	1.80	92	165.60	99	50	No
ETAII(RCA)	214.12	1.48	91.5	135.42	16.94	100	No
ETAII(CLA)	225.23	1.47	93	136.71	16.94	100	No
SCSA	405.64	1.58	125	197.50	20.51	100	No
ACA	335.16	1.57	153	240.21	16.34	100	No
LUA	323.72	1.37	114	156.18	34.64	100	No
CSAA[a]	251.37	1.74	84.3	146.68	0.91	100	No
Proposed(RCA)[b]	244.72	1.69	102	172.38	8.67	6.25	No
Proposed(CLA)[b]	251.68	1.61	106	170.66	8.67	6.25	No
Proposed(RCA)[c]	350.22	1.68	142	238.56	8.58	6.67	Yes
Proposed(CLA)[c]	402.01	1.55	155	240.25	8.58	6.67	Yes

[a] Without error reduction module.
[b] Without sign correction module.
[c] With sign correction module.

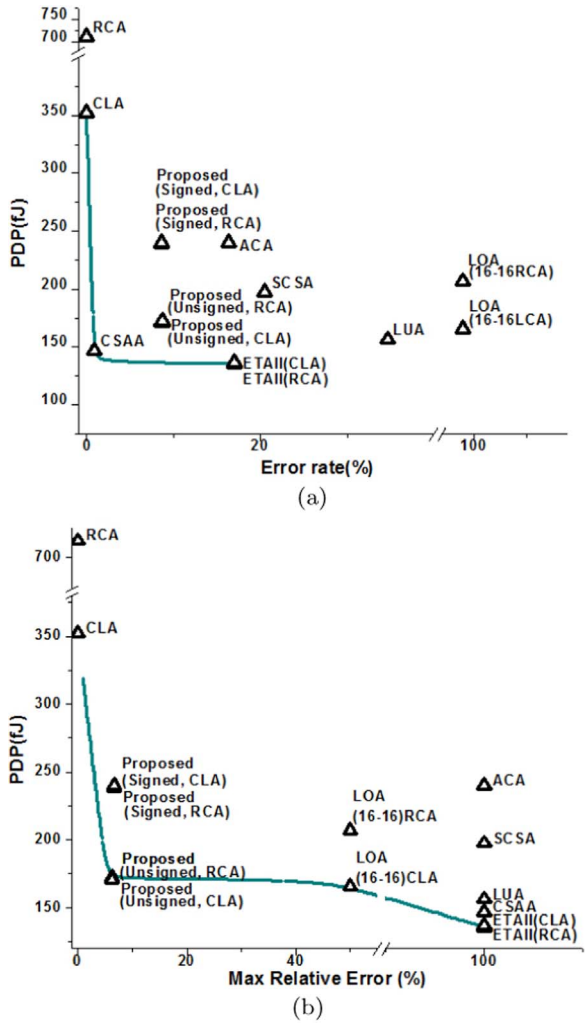


Fig. 6. Plots of power-delay product (PDP) versus (a) error rate and (b) maximal relative error for all the adders in Table 1.

maximal-relative-error curve. The RCA-based implementation is very close to the optimal curve. After adding the sign correction module, the PDP of our adder (either RCA-based or CLA-based) increases, but it does not deviate much from the optimal curve. Considering its feature of always ensuring the correct sign calculation, one may find this trade-off worthy in some applications.

To further study the overhead of the various components in our proposed approximate adder with sign correction module, we present the power breakdown of the major components in Table 2. Both RCA and CLA-based implementations are considered. Each adder can be decomposed into two major parts, the approximate adder without sign correction and the sign correction module. The adder without sign correction can be further decomposed into the main part (which includes the P/G signal generators, the carry generators, and the sub-adders) and the set of multiplexers (which are introduced to reduce the relative error). From Table 2, we can see that for both the RCA and CLA-based implementations, the sign correction module takes less than 1/3 of the total power of the proposed adder. The power consumption of the multiplexers is very small, accounting for less than 2.5% of the total value.

Table 2
Power breakdown of the proposed approximate adder with sign correction module (32-bit and $k = 4$).

Component		RCA-based		CLA-based	
		Power (μW)	%	Power (μW)	%
Approximate adder w/o sign correction	Main part	93.06	65.5	100.7	65.0
	MUX	3.54	2.5	3.6	2.3
Sign correction		45.4	32	50.7	32.7
Total		142	100	155	100

Table 3 shows the results of all the adders for $k = 8$. The proposed approximate adder without sign correction module (either RCA-based or CLA-based) has a much smaller PDP than RCA and CLA. With the additional sign correction module, the proposed adder has an increased PDP, but it is still smaller than those of RCA and CLA. Compared with the other approximate adders, our proposed adders with and without sign correction (either RCA-based or CLA-based) have smaller PDP than SCSA and ACA. Again, the main advantages of the proposed approximate adders over the existing approximate adders lie in the small relative error and the capability to ensure the correct sign calculation in the signed addition. Furthermore, we can see that the increase of the block size significantly reduces the error rate and maximal relative error of the proposed adders.

We also plot the PDP versus error rate and the PDP versus maximal relative error for all the adders in Table 3 in Fig. 7(a) and Fig. 7(b), respectively. A Pareto optimal curve is identified in each figure.

From Fig. 7(a), we can see that the proposed approximate adder without sign correction module (either RCA-based or CLA-based) is very close to the Pareto optimal PDP-versus-error-rate curve. The proposed adder with sign correction module (either RCA-based or CLA-based) does not deviate much from the curve. This demonstrates that the proposed adders achieve a good trade-off between PDP and error rate.

From Fig. 7(b), which focuses on the maximal relative error, we find that our proposed CLA-based approximate adder without sign correction module lies on the Pareto optimal curve. The RCA-based implementation is very close to the optimal curve. The approximate adder with sign correction module (either RCA-based or CLA-based) has a larger PDP and hence, is off the optimal curve. However, it has the unique feature of the always correct sign calculation, which is desirable for some applications, as we will demonstrate in Section 6.4.

For the case where the block size $k = 8$, we also present the power breakdown of the proposed adder with sign correction in Table 4. Both RCA and CLA-based implementations are considered. For both implementations, the power consumptions of the sign correction module and the multiplexers account for less than 27% and 1.1% of the total power, respectively. Compared to the previous results for $k = 4$, their percentages reduce, since the number of the blocks of the adder reduces.

6.2. Error rates of the proposed adders with different sizes

In this section, we evaluated the error rates of the proposed approximate adders with different bit lengths n and block sizes k. The error rates were obtained by the methods discussed in Sections 4.3 and 5.3. Fig. 8 shows the error rates of the proposed adder without sign correction for different bit lengths n and block sizes k. For a fixed bit length n, the error rate decreases exponentially with the block size k. For a fixed block size k, the error rate increases with the bit length n, because the block number $m = n/k$ increases, causing a large error rate. For the adders with $n \leq 64$ and $k = 8$, the error rates are less than 10^{-2}. For the adders with $n \leq 256$ and $k = 16$, the error rates are less than 10^{-4}.

The error rates of the proposed approximate adder with sign correction are very close to that of the approximate adder without sign correction and hence, are of the similar trend as shown in Fig. 8. Table 5 shows the detailed comparison of the error rates of the proposed approximate adders with and without sign correction for a number of combinations of n and k. For all combinations of n and k,

Table 3
Comparison of the proposed adders with other 32-bit adders ($k = 8$).

Adders	Area (μm^2)	Delay (ns)	Power (μW)	Power- delay product (fJ)	Error rate (%)	Max relative error (%)	Sign always correct
RCA	254.03	4.04	176	711.04	0	0	Yes
CLA	700.11	1.54	229	352.66	0	0	Yes
LOA(RCA)	127.95	2.37	87	206.90	99	50	No
LOA(CLA)	192.32	1.80	92	165.60	99	50	No
ETAII(RCA)	247.65	1.82	118	214.76	0.4	100	No
ETAII(CLA)	297.65	1.71	126	215.46	0.4	100	No
SCSA	463.37	1.71	196	335.16	0.58	100	No
ACA	332.23	2.46	182	447.72	0.39	100	No
LUA	516.83	1.50	158	237.00	2.22	100	No
CSAA[a]	302.70	1.84	135	248.40	< 0.1	100	No
Proposed(RCA)[b]	296.86	1.84	135	248.40	0.19	0.39	No
Proposed(CLA)[b]	335.96	1.76	140	246.4	0.19	0.39	No
Proposed(RCA)[c]	379.58	1.92	172	330.24	0.19	0.39	Yes
Proposed(CLA)[c]	426.93	1.85	178	329.30	0.19	0.39	Yes

[a] Without error reduction module.
[b] Without sign correction module.
[c] With sign correction module.

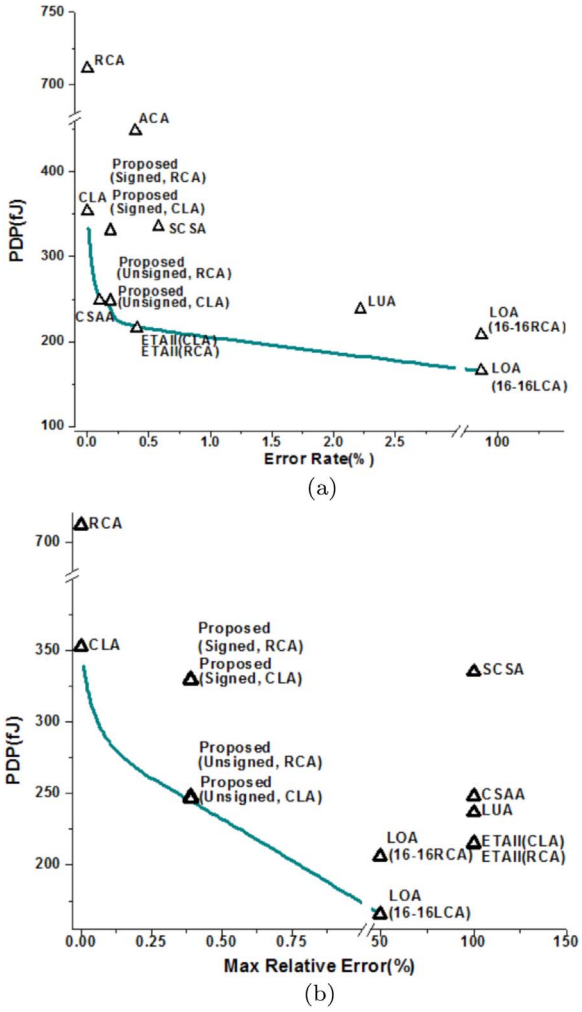


Fig. 7. Plots of power-delay product (PDP) versus (a) error rate and (b) maximal relative error for all the adders in Table 3.

the error rate of the adder with sign correction is smaller than that of the adder without sign correction, which is indicated by Theorem 4. The last column of the table shows the reduction ratio of the error rate of the former over the latter. It can be seen that only when $n = 8$ and $k = 2$, the difference is larger than 10%. For a fixed k, as n increases, the reduction ratio drops. When $n \geq 64$, the reduction ratios for different

Table 4
Power breakdown of the proposed approximate adder with sign correction module (32-bit and $k = 8$).

Component		RCA-based		CLA-based	
		Power (μW)	%	Power (μW)	%
Approximate adder w/o sign correction	Main part	124.6	72.5	128.0	72.2
	MUX	1.87	1.1	1.88	1.1
Sign correction		45.5	26.4	47.5	26.7
Total		172	100	178	100

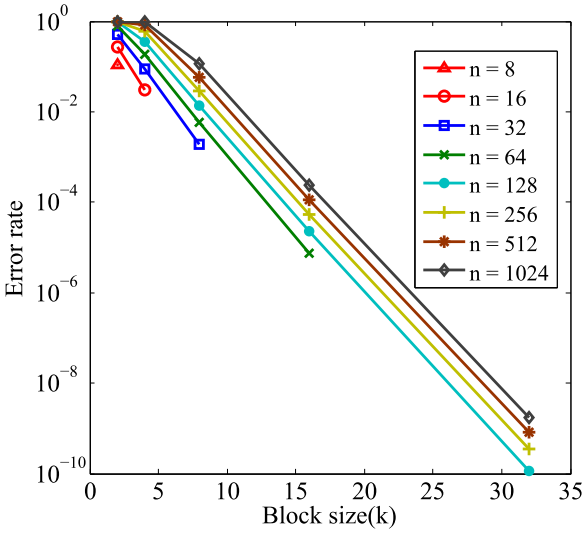


Fig. 8. Error rates of the proposed approximate adders without sign correction for different bit lengths n and block sizes k.

block sizes k are all very close to zero.

6.3. Areas, delays, and power consumptions of the proposed adders with different sizes

In this section, we studied the effects of different bit lengths n and block sizes k on the area, delay, and power consumption of our proposed adders. We focused on the RCA-based implementations. We chose $n = 16, 32, 64$. For $n = 16$, we chose $k = 2, 4$. For $n = 32$, we chose $k = 2, 4, 8$. For $n = 64$, we chose $k = 4, 8, 16$. The plots of area, power consumption, and delay are shown in Fig. 9(a), (b), and (c), respectively.

Table 5
Comparison of the error rates of the proposed approximate adders with and without sign correction.

n	k	ER without sign correction (e_1)	ER with sign correction (e_2)	Reduction ratio $\left(\frac{e_1-e_2}{e_1}\cdot 100\%\right)$
8	2	0.109	0.094	14.3
16	2	0.275	0.262	4.72
	4	3.03E-2	2.93E-2	3.23
32	2	0.520	0.512	1.65
	4	8.67E-2	8.58E-2	1.06
	8	1.95E-3	1.95E-3	0.195
64	2	0.790	0.786	0.478
	4	0.190	0.189	0.430
	8	5.83E-3	5.83E-3	6.52E-2
	16	7.63E-6	7.63E-6	7.86E-4
128	2	0.960	0.959	7.56E-2
	4	0.363	0.362	0.177
	8	1.35E-2	1.35E-2	2.80E-2
	16	2.29E-5	2.29E-5	0
	32	1.16E-10	1.16E-10	0

From Fig. 9(a), we can see that the area is mainly affected by the bit length n: it increases with n. Each proposed approximate adder without sign correction lies below its counterpart with sign correction, since the sign correction module takes additional area. We can also observe that when n is fixed, the area difference between the approximate adder with sign correction and its counterpart without sign correction decreases with the block size k. This is because the difference is due to the area of the sign correction module, which, as shown in Fig. 5, increases with the number of blocks. As the block size k increases, the number of blocks decreases. This leads to a smaller sign correction module and reduces the area difference between the two adders.

From Fig. 9(b), we can see that the power consumption is still mainly affected by the bit length n. Furthermore, the power consumptions of the proposed approximate adders have a similar trend as their areas. This is because the power consumption of a circuit is usually proportional to its area.

Fig. 9(c) shows the delays of the proposed adders with different bit lengths and block sizes. We can see that for the approximate adders without sign correction, their delays are independent of the bit length. For example, the point for $n = 16$ and $k = 2$ overlaps that for $n = 32$ and $k = 2$. This is because their delays are only affected by the block size k, as we discussed in Section 4.4. As k increases, the delay of the approximate adder without sign correction increases. The delay of a proposed approximate adder with sign correction is usually larger than its counterpart without sign correction. This is due to the existence of the additional AND gates added to the outputs of the sub-adders, as shown in Fig. 5(c). As k increases, the delay of the approximate adder with sign correction also increases.

6.4. Performance of the proposed adders on real applications

In this section, we applied our proposed approximate adders with and without sign correction module to three different applications to study their performance on real applications. These three applications are 1) mean filter for digital images, 2) Roberts cross-based edge detection, and 3) k-means clustering. For all these applications, we used the proposed approximate adders to do the additions and subtractions. For comparison purpose, we also applied other approximate adders to the same applications.

6.4.1. Mean filter for digital images

Mean filter algorithm is commonly applied to reduce noise in an image. It replaces the value for each pixel in the image by the average value of pixels in an $n \times n$ window centered at that pixel [36].

In this section, we consider implementing a mean filter of a 3×3 window size. It does the following calculation

$$z_{i,j} = (x_{i-1,j-1} + x_{i-1,j} + x_{i-1,j+1} + x_{i,j-1} + x_{i,j} + x_{i,j+1} + x_{i+1,j-1} + x_{i+1,j} + x_{i+1,j+1})/9, \quad (22)$$

where $z_{i,j}$ and $x_{i,j}$ denote the input and output pixel values, respectively, at row i and column j of the image. Notice that the calculation involves only addition; there is no subtraction.

Fig. 10 shows the output images of applying various adders to perform the additions involved in Eq. (22). The adders tested include an accurate adder, the proposed approximate adders with and without sign correction, and four other approximate adders, which are CSAA, LOA, LUA, and ETAII. The bit lengths of all the adders were set as 15. The block size for the approximate adders other than the LUA and the LOA was set as 3. The bit length of the carry-lookahead in LUA was set as 3. The bit length of the inaccurate part of LOA was set as 7. The input image is shown in Fig. 10(a) and the output images by applying various adders are shown in Fig. 10(b-h). Compared with the result of the accurate one, the output images produced by applying LOA, LUA, and ETAII are not good, while that by CSAA is acceptable. The proposed approximate adders with and without sign correction module give results very close to the accurate one. Indeed, since there is no subtraction involved, the two proposed adders produce the same result. It is interesting to note that CSAA, although with low error rate, could occasionally produce relative error close to 100% and hence, introduce some visible noise rather than eliminate it. This application demonstrates the importance of reducing the maximal relative error and the advantage of our proposed adders.

To numerically evaluate the qualities of different adders, we also calculated the peak signal-to-noise ratio (PSNR) for each output image. It is defined as:

$$PSNR = 10 \cdot \log_{10}\left(\frac{MAX_I^2}{MSE}\right) \quad (23)$$

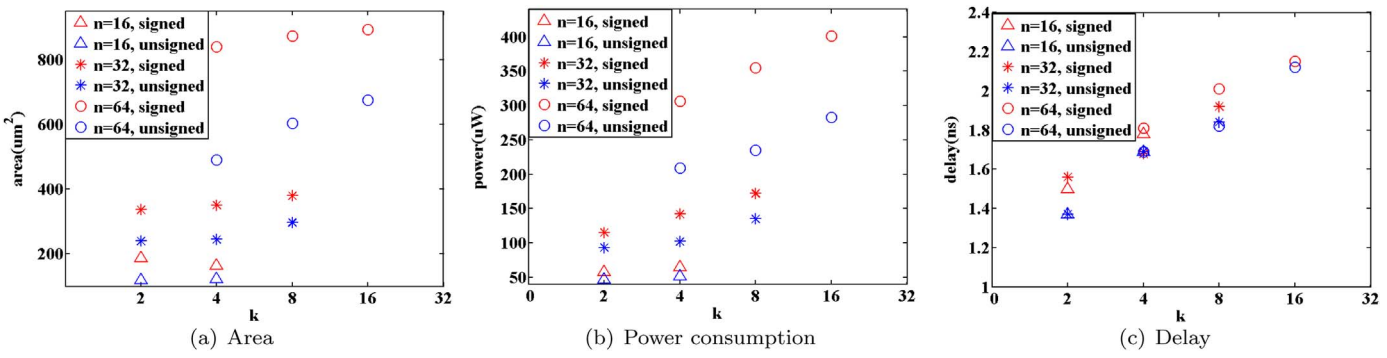
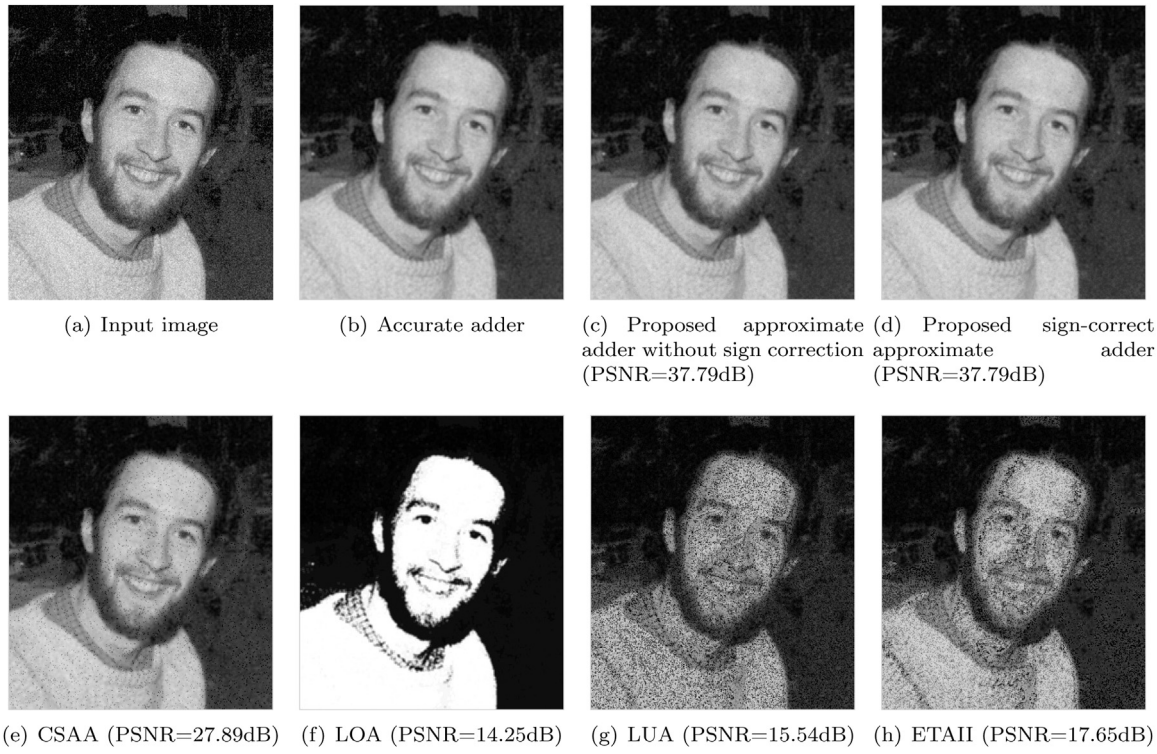


Fig. 9. The areas, power consumptions, and delays of the proposed adders with different bit lengths and block sizes.



(a) Input image (b) Accurate adder (c) Proposed approximate adder without sign correction (PSNR=37.79dB) (d) Proposed sign-correct approximate adder (PSNR=37.79dB)

(e) CSAA (PSNR=27.89dB) (f) LOA (PSNR=14.25dB) (g) LUA (PSNR=15.54dB) (h) ETAII (PSNR=17.65dB)

Fig. 10. Mean filter for a digital image using different adders.

where MAX_I is the maximum possible pixel value in an image, which is 255 in our experiment. MSE is the mean squared error, defined as:

$$MSE = \frac{1}{W \times H} \sum_{i=0}^{W-1} \sum_{j=0}^{H-1} [z_{i,j} - z'_{i,j}]^2$$

where W and H represent the width and height of the image, respectively, and $z_{i,j}$ and $z'_{i,j}$ represent the accurate value and the value obtained by an approximate adder, respectively, at location (i, j) in the output image. The larger the PSNR is, the higher the quality of the output image is. The PSNR for each approximate adder is shown in the caption below the image produced by that adder. We can see that the PSNR of the image produced by the proposed approximate adder, whether with sign correction module or not, is higher than that by any other approximate adder.

6.4.2. Roberts cross-based edge detection

Roberts cross algorithm [37] is a simple method to detect edges in a given image. In this section, we consider implementing the following variation of the Roberts cross algorithm:

$$z_{i,j} = |x_{i,j} - x_{i+1,j+1}| + |x_{i+1,j} - x_{i,j+1}|, \tag{24}$$

where $z_{i,j}$ and $x_{i,j}$ denote the input and output pixel values, respectively, at row i and column j of the image. Note the calculation involves one addition and two subtractions, which are realized through 2's complement signed addition.

Fig. 11 shows the output images of applying various adders to perform the addition and subtractions involved in Eq. (24). The adders tested include an accurate adder, the proposed approximate adders with and without sign correction, and four other approximate adders, which are CSAA, LOA, LUA, and ETAII. The bit lengths of all the adders were set as 10. The block sizes for the proposed approximate adders, CSAA, and ETAII were all set as 2. The bit length of the carry-lookahead in LUA was set as 2. The bit length of the inaccurate part of LOA was set as 5, which is half of the total bit length. The input image is shown in Fig. 11(a) and the output images by applying various adders are shown in Fig. 11(b-h). Compared with the result of the accurate adder, the output images produced by using CSAA, LUA and ETAII are not good, while that by LOA is acceptable. However, there are still a number of noisy points in the image produced by LOA. In contrast, the result of the proposed sign-correct approximate adder is almost the same as that of the accurate adder. This shows the proposed approximate adder performs much better than those previously proposed adders. It should be noted that the proposed approximate adder without sign correction does not perform well neither. This demonstrates the importance of the correct sign bit calculation in this application.

To numerically evaluate the qualities of different adders, we also calculated the peak signal-to-noise ratio (PSNR) for each output image using Eq. (23). The PSNR for each approximate adder is shown in the caption below the image produced by that adder. We can see that the PSNR of the image produced by the proposed sign-correct approximate adder is much higher than that by any other approximate adder.

To further demonstrate the low error rate and low relative error of the proposed sign-correct approximate adder, we collected the first subtraction results in a Roberts cross calculation over all pixels, calculated their relative errors, and plotted the distribution of the relative errors in Fig. 12(a). The x-axis is the value of relative error and the y-axis is the number of pixels with the corresponding relative error.

The size of the figure used in our experiment is 640×480, so there are 307,200 samples in total. As we show in Section 5.2, the relative error of our proposed sign-correct approximate adder is bounded by $\frac{1}{2^k - 1}$. With the choice of block size $k = 2$ in our experiment, the relative error is bounded by $\frac{1}{2^2 - 1} = 33.33\%$. This is consistent with Fig. 12(a), which shows that the maximal relative error is about 27%. Besides low relative error, we can also see that majority of the computation results are correct (i.e., the relative error is zero). Indeed, the error rate of the proposed approximate adder is only 1.12%.

For comparison purpose, we plotted the same relative error distributions for other four approximate adders, CSAA, LOA, LUA, and ETAII, in Fig. 12(b-e), respectively. Since these approximate adders cannot guarantee the correct sign bit calculation, there exist some inputs from the sample image for which the accurate results

(a) Input image (b) Accurate adder (c) Proposed approximate adder without sign correction (PSNR=8.84dB) (d) Proposed sign-correct approximate adder (PSNR=43.44dB)

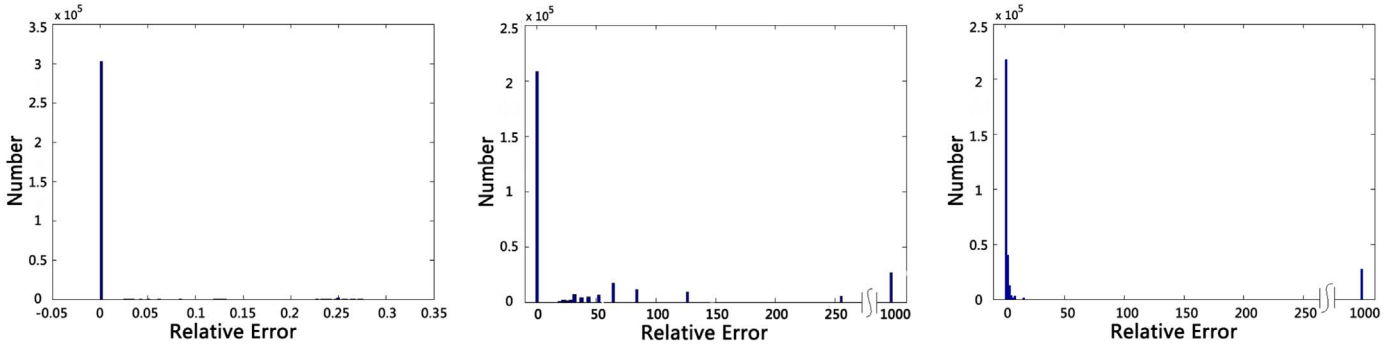
(e) CSAA (PSNR=6.34dB) (f) LOA (PSNR=24.77dB) (g) LUA (PSNR=8.01dB) (h) ETAII (PSNR=7.4dB)

Fig. 11. Roberts cross-based edge detection using different adders.

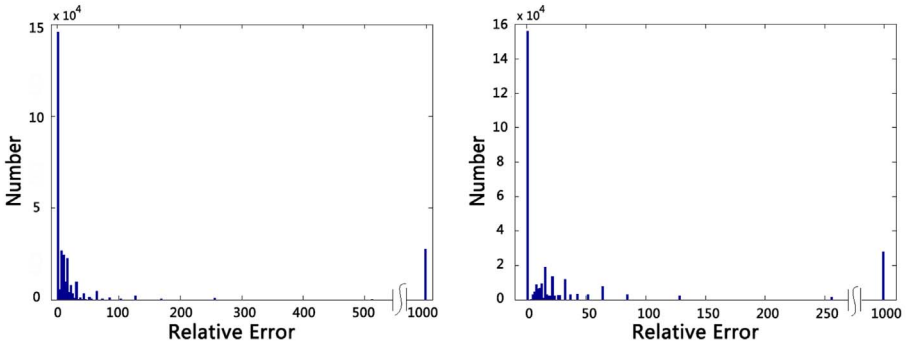
should be 0, but the approximate results are not. The corresponding relative error is infinity. We indicate this situation by setting the relative error as 1000 in the histograms. Note that this situation will never happen by using our proposed sign-correct approximate adder. Comparing the five error distributions shown in Fig. 12, we can see that our proposed adder has significant advantages. Due to the inaccurate sign calculation, some results given by the other adders could be hundreds of times different from the accurate ones, while the results given by our adder have at most 27% difference. Furthermore, although some of the other approximate adders, such as CSAA, could have a low error rate for unsigned additions, their error rates are all very large for 2's complement signed additions. From our experiment, the error rates of CSAA, LOA, LUA, and ETAII are 32.09%, 77.74%, 56.14%, and 51.31%, respectively, which are much larger than that of our adder.

6.4.3. k-means clustering

In this section, we studied the performance of our approximate adders in the *k*-means clustering algorithm, which is widely-used in data mining and unsupervised learning [38]. The target of the algorithm is to partition a set of *n* data points into *k* clusters so that the sum of the distances from each point to the centroid of its belonging cluster is minimized. In other words, the following objective function, known as within-cluster distance (WCD), is minimized:



(a) Proposed sign-correct approximate adder (error rate=1.12%) (b) CSAA (error rate=32.09%) (c) LOA (error rate=77.74%)



(d) LUA (error rate=56.14%) (e) ETAII (error rate=51.31%)

Fig. 12. The distributions of the relative errors on the subtraction step in the edge detection application for five approximate adders.

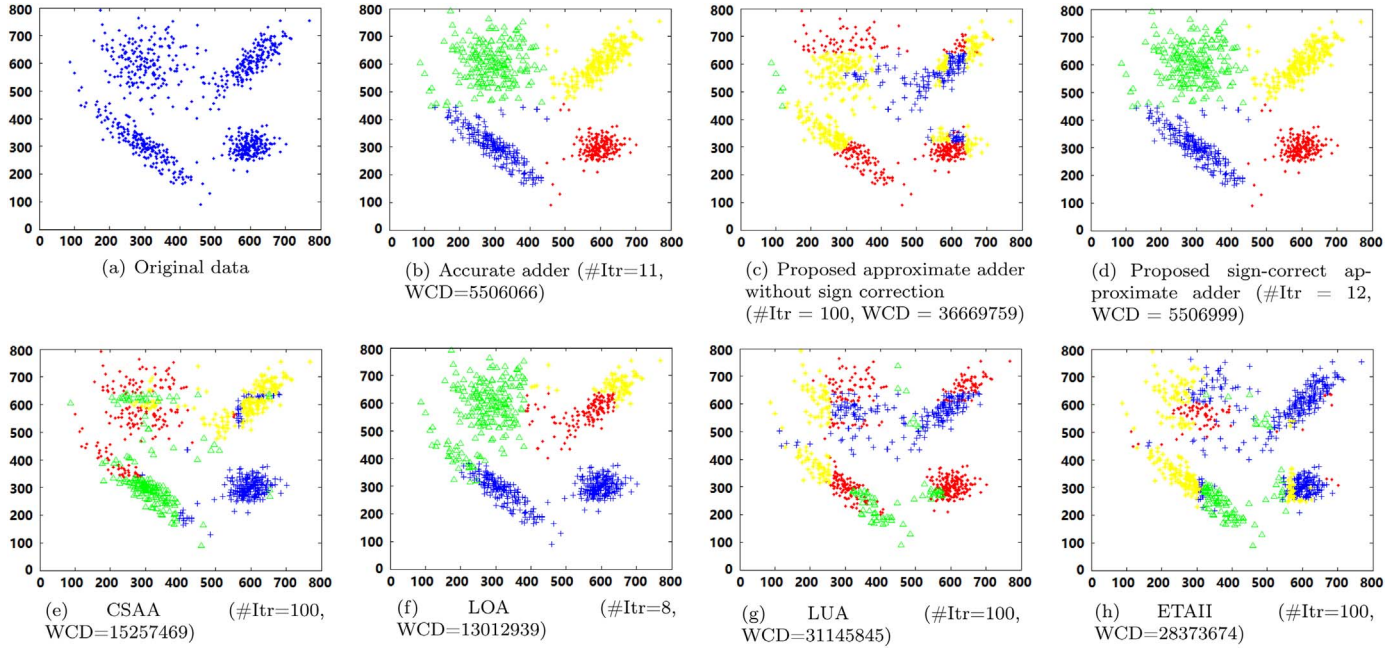


Fig. 13. *k*-means clustering using different adders.

$$F = \sum_{i=1}^{k} \sum_{x_j^{(i)} \in C_i} \left| x_j^{(i)} - \mu_i \right|^2, \tag{25}$$

where C_i is the i-th cluster, μ_i is the centroid of the cluster C_i, and $x_j^{(i)}$ is the j-th point in the cluster C_i.

The basic idea of the algorithm is to iteratively update each cluster by assigning each point to its closest cluster until there is no more change to the clusters. The major computation of the algorithm is to calculate the distance between a point and the centroid of a cluster, which involves a number of subtractions and additions. We applied our proposed approximate adders and various other approximate adders to do these operations. All the adders are of bit length 12. The block lengths of the approximate adders are 2.

A synthetic 2-D data set shown in Fig. 13(a) was used to test the performance of the *k*-means clustering algorithm using various adders. The data set contains 800 points in total. The number of clusters is set as 4. The termination condition is either the centers of all the clusters do not change between two successive iterations or the number of iterations is over 100. Fig. 13(b-h) show the clustering results given by using the accurate adder, our proposed approximate adder without sign correction, our proposed sign-correct approximate adder, CSAA, LOA, LUA, and ETAII, respectively. The final four clusters are indicated by four different types of points. For each adder, we also show the total number of iterations of the algorithm and within-cluster distance (WCD) calculated by Eq. (25) in the caption below the corresponding figure.

Comparing our proposed approximate adders with the accurate adder, we can see that the sign-correct one can give a result very close to that of the accurate adder, while the one without sign correction could not. This observation is similar to what we had for the Roberts cross-based edge detection. Again, it shows the importance of correct sign calculation for applications involving subtractions. We can see that using the approximate adder without sign correction, the result could not converge in 100 iterations and the WCD of the final clustering result is much larger than that of the accurate adder. In contrast, if we use the sign-correct approximate adder, although it needs one more iteration than the accurate adder to converge, it only takes 12 iterations, which is still very fast. The WCD of the final result is only 0.017% larger than that of the accurate adder.

Furthermore, we can see that among the other four previous approximate adders, only the clustering using LOA converges, taking only 8 iterations, which is even fewer than that of the accurate adder. However, its final clustering result is still quite different from the correct one. Its final WCD is about 140% larger than that of the accurate adder. The *k*-means clusterings by the other three approximate adders all fail to converge after 100 iterations. In summary, our proposed sign-correct approximate adder has much better performance than the other approximate adders in *k*-means clustering.

7. Conclusion

In this paper, we proposed a novel approximate adder which significantly reduces the power-delay product over the conventional accurate adders. The efficient carry speculation and error reduction technique proposed in this work can make our adder have a low error rate and, more importantly, a low maximal relative error. Furthermore, a lightweight error correction module was proposed that ensures the correct sign calculation in 2's complement signed addition. By applying our adders in three real applications, we found that our proposed approximate adders with and without sign correction can produce high-quality results in applications that require small relative error. Furthermore, our proposed sign-correct approximate adder can perform very well when doing 2's complement signed additions. The final result of the adder is very close to that given by the accurate one.

Acknowledgments

This work is supported by National Natural Science Foundation of China (NSFC) under Grant No. 61574089.

Appendix A. Proof of Theorem 1

We first consider any group except the rightmost one. Suppose it starts at block r and ends at block d, where $r > d$. By the definition of a group, there exists an integer $d \le t \le r-1$ such that $pp^r = \cdots = pp^{t+1} = 0$ and $pp^t = \cdots = pp^d = 1$. Also, we have $pp^{d-1} = 0$. We distinguish the following two cases.

1. The case where $t > d$. Given the block propagate signals, the speculated carry-ins to the sub-adders in the group are

$$C^i_{apx,in} = \begin{cases} C^{i-1}_{apx,o}, & t+1 \le i \le r \\ g^{i-1}_{k-1}, & d \le i \le t \end{cases} \tag{A.1}$$

For $t+2 \le i \le r$, since $pp^{i-1} = 0$, by Lemma 2, we have $C^{i-1}_{apx,o} = C^{i-1}_o$. Since $pp^t = 1$, $C^t_{apx,o}$ is propagated from the generate signal g^{t-1}_{k-1} and hence, $C^t_{apx,o} = g^{t-1}_{k-1}$. Because $pp^{t-1} = 1$, we must have $g^{t-1}_{k-1} = 0$. Therefore, $C^t_{apx,o} = 0$. Because $pp^{t-2} = \cdots = pp^d = 1$, we also have $g^{t-2}_{k-1} = \cdots = g^d_{k-1} = 0$. Given the above values, Eq. (A.1) changes to

$$C^i_{apx,in} = \begin{cases} C^{i-1}_o, & t+2 \le i \le r \\ 0, & d+1 \le i \le t+1 \\ g^{d-1}_{k-1}, & i = d \end{cases} \tag{A.2}$$

Since $pp^t = \cdots = pp^d = 1$, for the accurate adder, we have

$$C^t_o = \cdots = C^{d-1}_o. \tag{A.3}$$

Next, we discuss based on the following three different combinations of g^{d-1}_{k-1} and C^{d-1}_o.

(a) The case where $g^{d-1}_{k-1} = C^{d-1}_o = 0$. For this case, by Eqs. (a) and (b), we can conclude that for all $d \le i \le r$, $C^i_{apx,in} = C^{i-1}_o$. Therefore, $SG = SG_{apx}$.

(b) The case where $g^{d-1}_{k-1} = C^{d-1}_o = 1$. Fig. A.14 shows the speculated carry-in to the sub-adder, the correct carry-in to the sub-adder, the approximate sum, and the correct sum for each block in the group for this situation. The speculated carry-ins and the correct carry-ins are obtained by Eqs. (A.2) and (A.3). For any $t+2 \le i \le r$, given that $C^i_{apx,in} = C^{i-1}_o$, the approximate sum at block i is equal to the correct one. Given that $pp^{t+1} = 0$, $C^{t+1}_{apx,in} = 0$, and $C^t_o = 1$, the correct sum at block $(t+1)$ is larger than the approximate sum at that block by 1. For any $d+1 \le i \le t$, given that $pp^i = 1$, the approximate sum at block i is $(11\ldots1)_2$ (k 1's in total), while the correct sum at block i is $(00\ldots0)_2$ (k 0's in total). Both the approximate sum and the correct sum at block d are $(00\ldots0)_2$ (k 0's in total). Therefore, we have $SG - SG_{apx} = 2^k$. Since $t > d$, we have $SG \ge 2^{2k}$. Therefore, we conclude that $0 \le SG - SG_{apx} \le \frac{1}{2^k} SG$.

(c) The case where $g^{d-1}_{k-1} \ne C^{d-1}_o$. In this case, we must have $g^{d-1}_{k-1} = 0$ and $C^{d-1}_o = 1$, because $g^{d-1}_{k-1} = 1$ implies that $C^{d-1}_o = 1$. The correct sum is same as that shown in Fig. A.14. The approximate sum is same as that shown in Fig. A.14 except that the sum at block d is $11\cdots1$ (k 1's in total). Therefore, we have $SG - SG_{apx} = 1$. Since $SG \ge 2^{2k}$, we conclude that $0 \le SG - SG_{apx} \le \frac{1}{2^k} SG$.

2. The case where $t = d$. Given the block propagate signals, it can be shown that the speculated carry-ins to the sub-adders within the group are

$$C^i_{apx,in} = \begin{cases} C^{i-1}_o, & d+2 \le i \le r \\ g^{d-1}_{k-1}, & d \le i \le d+1 \end{cases} \tag{A.4}$$

Since $pp^d = 1$, for the accurate adder, we have

$$C^d_o = C^{d-1}_o. \tag{A.5}$$

Next, we discuss based on whether $g^{d-1}_{k-1} = C^{d-1}_o$.

(a) The case where $g^{d-1}_{k-1} = C^{d-1}_o$. In this case, by Eqs. (A.4) and (A.5), we have that for all $d \le i \le r$, $C^i_{apx,in} = C^{i-1}_o$. Therefore, $SG = SG_{apx}$.

(b) The case where $g^{d-1}_{k-1} \ne C^{d-1}_o$. In this case, as we discussed before, we must have $g^{d-1}_{k-1} = 0$ and $C^{d-1}_o = 1$. Give this, we have $C^{d+1}_{apx,in} = C^d_{apx,in} = g^{d-1}_{k-1} = 0$, while $C^d_o = C^{d-1}_o = 1$. Fig. A.15 shows the speculated carry-in to the sub-adder, the correct carry-in to the sub-adder, the approximate sum, and the correct sum for each block in the group for this situation. Given that $pp^d = 1$, the approximate sum and

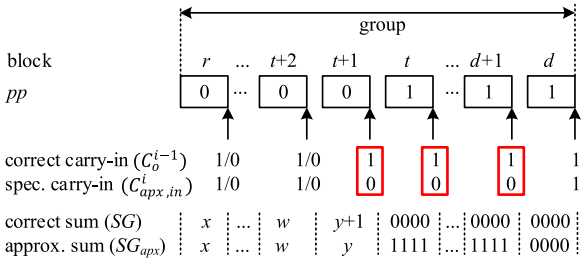


Fig. A.14. The addition in a group when $t > d$ and $g^{d-1}_{k-1} = C^{d-1}_o = 1$. We assume the block size $k = 4$.

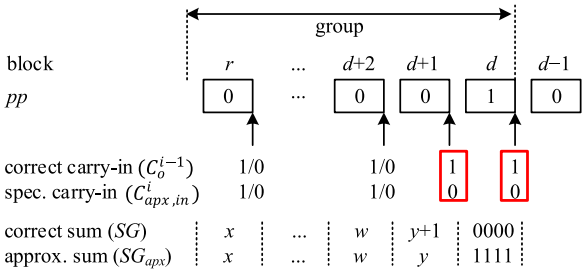


Fig. A.15. The addition in a group when $t = d$ and $g_{k-1}^{d-1} \neq C_o^{d-1}$. We assume the block size $k = 4$.

the correct sum at block d are $(11\ldots1)_2$ (k 1's in total) and $(00\ldots0)_2$ (k 0's in total), respectively. Given that $pp^{d+1} = 0$, $C_{apx,in}^{d+1} = 0$, and $C_o^d = 1$, the correct sum at block $(d + 1)$ is larger than the approximate sum at that block by 1. For any $d + 2 \le i \le r$, since the speculated carry-in to the sub-adder i is correct, the approximate sum at block i is equal to the correct one. From Fig. A.15, we can see that $SG - SG_{apx} = 1$ and $SG \ge 2^k$. Therefore, we have $0 \le SG - SG_{apx} \le \frac{1}{2^k} SG$.

In summary, for both cases, we have $0 \le SG - SG_{apx} \le \frac{1}{2^k} SG$. Thus, we have shown that the claim in the theorem holds for any group except the rightmost one.

Now, consider the rightmost group. A difference of it from the other groups is that all of its blocks may have the block propagate signals as 0. If it is like the other groups, then the claim in the theorem also holds for the rightmost group. If all the block propagate signals in the group are 0, it can be easily shown that the sum of the rightmost group is correct. Therefore, the claim in the theorem also holds for the rightmost group.

Appendix B. Proof of Theorem 2

First, we have the following lemma.

Lemma 6. *If $pp^{m-1} = 1$, then there exists an $1 \le i \le m - 1$ such that $pp^{m-1} = \cdots = pp^i = 1$ and $C_{apx,o}^{i-1} = C_o^{i-1}$.*□

Proof. For any input combination such that $pp^{m-1} = 1$, it belongs to one of the following two cases:

a. There exists an $2 \le i \le m - 1$ such that $pp^{m-1} = \cdots = pp^i = 1$ and $pp^{i-1} = 0$.
b. $pp^{m-1} = \cdots = pp^1 = 1$.

For Case 1, since $pp^{i-1} = 0$, by Lemma 2, we have $C_{apx,o}^{i-1} = C_o^{i-1}$. For Case 2, since the carry-in signal to the carry generator 0 is the correct value c_0, the output $C_{apx,o}^0$ of the carry generator 0 is equal to the correct value C_o^0. For both cases, the claim in the lemma holds.□

Now, consider an input that causes a sign error. If $pp^{m-1} = 0$, then by Lemma 2, $C_{apx,sign}$ is equal to the correct carry-in to the sign bit C_{sign} and hence, the sign bit is correct. Therefore, we must have $pp^{m-1} = 1$. By Lemma 6, there must exist an $1 \le i \le m - 1$ such that $pp^{m-1} = \cdots = pp^i = 1$ and $C_{apx,o}^{i-1} = C_o^{i-1}$.

We argue that if there is a sign error, $C_{apx,o}^{i-1}$ must be 1. We prove this by contradiction. Assume $C_{apx,o}^{i-1} = 0$. Then, $C_o^{i-1} = C_{apx,o}^{i-1} = 0$. Since $pp^{m-1} = \cdots = pp^i = 1$, the correct carry-in to the sign bit is $C_{sign} = 0$. On the other hand, since $pp^{m-2} = \cdots = pp^i = 1$ and $C_o^{i-1} = 0$, we have $C_o^{m-2} = 0$. By Corollary 1, we have $C_{apx,o}^{m-2} = 0$. Since $pp^{m-1} = 1$, we have $C_{apx,sign} = C_{apx,o}^{m-2}$ due to carry propagation. Therefore, we have $C_{apx,sign} = C_{apx,o}^{m-2} = 0 = C_{sign}$, which contradicts with the fact that the input produces a wrong sign bit. Therefore, we have $C_{apx,o}^{i-1} = 1$ and the claim in the theorem holds.

Appendix C. Proof of Lemma 3

First, consider any $1 \le i \le l - 2$. The group i includes the blocks $d_{i+1} - 1$, $d_{i+1} - 2$, …, d_i. Since the group i is not the leftmost one, we must have $pp^{d_{i+1}-1} = 0$. Since $i \ge 1$, we have $d_{i+1} - 1 \ge d_i \ge d_1 \ge 1$. Thus, by Eq. (17), $sp^j = 0$ for all $1 \le j \le d_{i+1} - 1$. In turn, by Eq. (18), $CS^j = 0$ for all $1 \le j \le d_{i+1} - 1$. Specifically, $CS^j = 0$ for all $d_i \le j \le d_{i+1} - 1$. Finally, by Eq. (19), we have $SG_{sapx,i} = SG_{apx,i}$.

Now, consider the group 0, which includes the blocks $d_1 - 1$, …, d_0. By the same argument above, we have $CS^j = 0$ for all $1 \le j \le d_1 - 1$. Thus, by Eq. (19), the approximate sums at the blocks $d_1 - 1$, …, 1 are the same as those in the approximate adder without sign correction. Furthermore, by our design, the approximate sum at the block 0 is the same as that in the approximate adder without sign correction. In summary, we have $SG_{sapx,0} = SG_{apx,0}$.

Appendix D. Proof of Lemma 4

If $pp^{m-1} = 0$, by the same argument used in the proof of Lemma 3, we can show $SG_{sapx,l-1} = SG_{apx,l-1}$.

Now, consider the case where $pp^{m-1} = 1$. For simplicity, denote $t = d_{l-1}$. First, we consider the case where $t \ge 1$. Since the group $l - 1$ ends at block t, we have $pp^{m-1} = \cdots = pp^t = 1$ and $pp^{t-1} = 0$. Since $pp^{t-1} = 0$, by Lemma 2, we have $C_{apx,o}^{t-1} = C_o^{t-1}$. We further distinguish between $C_o^{t-1} = 1$ and $C_o^{t-1} = 0$.

1. $C_o^{t-1} = 1$. Then, we have $C_{apx,o}^{t-1} = C_o^{t-1} = 1$. By Eq. (17), we have $sp^t = 1$. By Eq. (18), we further have $CS^j = 1$ for all $t \le j \le m - 1$. Finally, by Eq. (19), we have $s_{sapx,j}^i = 0$ for all $t \le i \le m - 1$ and $0 \le j \le k - 1$. Because $C_o^{t-1} = 1$ and $pp^{m-1} = \cdots = pp^t = 1$, the accurate sum bit $s_j^i = 0$ for all

$t \le i \le m-1$ and $0 \le j \le k-1$. Thus, $SG_{sapx,l-1} = SG_{l-1}$.

2. $C_o^{t-1} = 0$. It can be easily shown that $CS^i = 0$, for all $t \le i \le m-1$. Therefore, $SG_{sapx,l-1} = SG_{apx,l-1}$. Now, we analyze $SG_{apx,l-1}$, the approximate sum at group $(l-1)$ produced by the approximate adder without sign correction. Since $pp^{m-2} = \cdots = pp^t = 1$ and $C_o^{t-1} = 0$, the speculated carry-ins to the sub-adders at blocks $m-2, \ldots, t$ are $C_{apx,in}^{m-2} = g_{k-1}^{m-3} = 0, \ldots, C_{apx,in}^{t} = g_{k-1}^{t-1} = 0$. Furthermore, the carry-in to the $(m-1)$-th sub-adder is $C_{apx,o}^{m-2} = 0$. Given that $pp^{m-1} = \cdots = pp^t = 1$, the sum bit of the approximate adder without sign correction $s_{apx,j}^i = 1$ for all $t \le i \le m-1$ and $0 \le j \le k-1$. Since $C_o^{t-1} = 0$, the accurate sum bit $s_j^i = 1$ for all $t \le i \le m-1$ and $0 \le j \le k-1$. Thus, $SG_{apx,l-1} = SG_{l-1}$. Given that $SG_{sapx,l-1} = SG_{apx,l-1}$, we conclude that $SG_{sapx,l-1} = SG_{l-1}$.

Therefore, when $t \ge 1$, we have $SG_{sapx,l-1} = SG_{l-1}$.

Finally, we consider the remaining case where $t = 0$. For this situation, we have $pp^{m-1} = \cdots = pp^0 = 1$. By the same argument used above, we can show that the approximate sums at blocks $m-1, \ldots, 1$ are the same as those in the correct adder. Furthermore, by our design, the approximate sum at block 0 is the same as that in the correct adder. In summary, we have $SG_{sapx,l-1} = SG_{l-1}$.

Appendix E. Proof of Lemma 5

For any $0 \le i \le l-1$, group i starts at block $d_{i+1}-1$ and ends at block d_i. In normal case, there exists an integer $d_i \le t \le d_{i+1}-2$ such that blocks $d_{i+1}-1, \ldots, t+1$ have their block propagate signals as 0 and blocks $t, \ldots, d_i$ have their block propagate signals as 1. However, there are two exceptions. The first is that for the leftmost group, all of its blocks could have their block propagate signals as 1. For this case, we have $pp^{m-1} = 1$. By Lemma 4, we have $SG_{sapx,l-1} = SG_{l-1}$. Thus, the claim in the lemma holds. The second is that for the rightmost group, all of its blocks could have their block propagate signals as 0. For this case, it can be shown that $SG_{sapx,0} = SG_0$ and hence, the claim also holds.

Now, consider any normal case. If $i = l-1$, then since it is a normal case, we must have $pp^{m-1} = 0$. Therefore, by Lemma 4, we have $SG_{sapx,l-1} = SG_{apx,l-1}$. If $0 \le i \le l-2$, by Lemma 3, we also have $SG_{sapx,i} = SG_{apx,i}$. In summary, for any normal case, we must have $SG_{sapx,i} = SG_{apx,i}$. Thus, we only need to focus on the approximate adder without sign correction and prove that

$$SG_i - SG_{apx,i} \le \frac{1}{2^k - 1} SG'_i. \tag{E.1}$$

Eq. (E.1) obviously holds when $SG_i = SG_{apx,i}$. Next, we consider the situation in which $SG_i \ne SG_{apx,i}$.

From the proof of Theorem 1, we can see that only the following three cases can cause $SG_i \ne SG_{apx,i}$:

1. The case where $t > d_i$ and $g_{k-1}^{d_i-1} = C_o^{d_i-1} = 1$.
2. The case where $t > d_i$ and $g_{k-1}^{d_i-1} \ne C_o^{d_i-1}$.
3. The case where $t = d_i$ and $g_{k-1}^{d_i-1} \ne C_o^{d_i-1}$.

For Case 1, as shown in the proof of Theorem 1, $SG_i - SG_{apx,i} = 2^k$. Since SG'_i is the bit-wise negation of SG_i and the last $(t-d_i+1)k$ bits of SG_i are 0 (as shown in Fig. A.14), we have

$$SG'_i \ge 2^{(t-d_i+1)k} - 1 \ge 2^{2k} - 1 > 2^{2k} - 2^k.$$

Therefore, $\frac{1}{2^k-1} SG'_i > 2^k = SG_i - SG_{apx,i}$, which proves Eq. (E.1).

For Case 2, we have $SG_i - SG_{apx,i} = 1$ and $SG'_i \ge 2^{(t-d_i+1)k} - 1 > 2^{2k} - 2^k$. Thus, Eq. (E.1) also holds.

For Case 3, as shown in the proof of Theorem 1, $SG_i - SG_{apx,i} = 1$. Furthermore, we have $SG'_i \ge 2^k - 1$. Therefore, $\frac{1}{2^k-1} SG'_i \ge 1 = SG_i - SG_{apx,i}$, which proves Eq. (E.1).

In summary, for any normal case, the claim in the lemma holds.

Appendix F. Proof of Theorem 4

Consider any input that produces a correct output for the approximate adder without sign correction. Suppose for that input the m blocks are partitioned into l groups. Since the output is correct, we have $SG_{apx,i} = SG_i$, for $0 \le i \le l-1$. By Lemma 3, we have $SG_{sapx,i} = SG_{apx,i} = SG_i$ for all $0 \le i \le l-2$. Now, consider the $(l-1)$-th group. If $pp^{m-1} = 1$, by Lemma 4, we directly have $SG_{sapx,l-1} = SG_{l-1}$. If $pp^{m-1} = 0$, by Lemma 4, we also have $SG_{sapx,l-1} = SG_{apx,l-1} = SG_{l-1}$. Therefore, we conclude that for all $0 \le i \le l-1$, $SG_{sapx,i} = SG_i$. Given that the sign bit is correct, the input also produces the correct output for the sign-correct approximate adder.

Furthermore, there exist inputs such that $pp^{m-1} = 1$, $SG_{apx,l-1} \ne SG_{l-1}$, and for all $0 \le i \le l-2$, $SG_{apx,i} = SG_i$. For these inputs, since $SG_{apx,l-1} \ne SG_{l-1}$, the output of the approximate adder without sign correction is not correct. On the other hand, since $pp^{m-1} = 1$, by Lemma 4, we have $SG_{sapx,l-1} = SG_{l-1}$. Furthermore, by Lemma 3, we have $SG_{sapx,i} = SG_{apx,i} = SG_i$ for all $0 \le i \le l-2$. Therefore, the output of the sign-correct approximate adder is correct. Thus, the error rate of the sign-correct approximate adder is smaller than that of the approximate adder without sign correction.

References

[1] Y.-K. Chen, et al., Convergence of recognition, mining, and synthesis workloads and its implications, Proc. IEEE 96 (5) (2008) 790–807.

[2] V. Chippa, S. Chakradhar, K. Roy, A. Raghunathan, Analysis and characterization of inherent application resilience for approximate computing, in: Design Automation Conference, 2013, pp. 113:1–113:9.

[3] J. Han, M. Orshansky, Approximate computing: An emerging paradigm for energy-efficient design, in: European Test Symposium, 2013, pp. 1–6.

[4] H.R. Mahdiani, A. Ahmadi, S.M. Fakhraie, C. Lucas, Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications, IEEE Trans. Circuits Syst. I: Regul. Pap. 57 (4) (2010) 850–862.

[5] N. Zhu, W.L. Goh, K.S. Yeo, An enhanced low-power high-speed adder for error-tolerant application, in: International Symposium on Integrated Circuits, 2009, pp. 69–72.

[6] Y. Kim, Y. Zhang, P. Li, An energy efficient approximate adder with carry skip for error resilient neuromorphic VLSI systems, in: International Conference on Computer-Aided Design, 2013, pp. 130–137.
[7] M. Shafique, W. Ahmad, R. Hafiz, J. Henkel, A low latency generic accuracy configurable adder, in: Design Automation Conference, 2015, pp. 86:1–86:6.
[8] J. Hu, W. Qian, A new approximate adder with low relative error and correct sign calculation, in: Design, Automation and Test in Europe, 2015, pp. 1449–1454.
[9] W. Yu, Applications of Monte Carlo method to 3-D capacitance calculation and large matrix decomposition, in: International Conference on Solid-State and Integrated Circuit Technology, 2016, pp. 227–230.
[10] N. Halko, P.G. Martinsson, J.A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288.
[11] S. Misailovic, S. Sidiroglou, H. Hoffmann, M. Rinard, Quality of service profiling, in: International Conference on Software Engineering, 2010, pp. 25–34.
[12] H. Esmaeilzadeh, A. Sampson, L. Ceze, D. Burger, Neural acceleration for general-purpose approximate programs, in: International Symposium on Microarchitecture, 2012, pp. 449–460.
[13] M. Imani, A. Rahimi, T.S. Rosing, Resistive configurable associative memory for approximate computing, in: Design, Automation and Test in Europe, 2016, pp. 1327–1332.
[14] P. Kulkarni, P. Gupta, M. Ercegovac, Trading accuracy for power with an under-designed multiplier architecture, in: International Conference on VLSI Design, 2011, pp. 346–351.
[15] S. Rehman, W. El-Harouni, M. Shafique, A. Kumar, J. Henkel, Architectural-space exploration of approximate multipliers, in: International Conference on Computer-Aided Design, 2016, pp. 80:1–80:8.
[16] S. Hashemi, R.I. Bahar, S. Reda, A low-power dynamic divider for approximate applications, in: Design Automation Conference, ACM, 2016, pp. 105:1–105:6.
[17] A.K. Verma, P. Brisk, P. Ienne, Variable latency speculative addition: A new paradigm for arithmetic circuit design, in: Design, Automation and Test in Europe, 2008, pp. 1250–1255.
[18] K. Du, P. Varman, K. Mohanram, High performance reliable variable latency carry select addition, in: Design, Automation and Test in Europe, 2012, pp. 1257–1262.
[19] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, J. Henkel, An area-efficient consolidated configurable error correction for approximate hardware accelerators, in: Design Automation Conference, 2016, pp. 96:1–96:6.
[20] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, J. Henkel, Probabilistic error modeling for approximate adders, IEEE Trans. Comput. 66 (3) (2017) 515–530.
[21] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, Probabilistic error analysis of approximate recursive multipliers, IEEE Trans. Comput. (2017) 1.
[22] W. El-Harouni, et al., Embracing approximate computing for energy-efficient motion estimation in high efficiency video coding, in: Design, Automation and Test in Europe, 2017, pp. 1384–1389.
[23] Q. Zhang, T. Wang, Y. Tian, F. Yuan, Q. Xu, ApproxANN: an approximate computing framework for artificial neural network, in: Design, Automation and Test in Europe, 2015, pp. 701–706.
[24] S. Mittal, A survey of techniques for approximate computing, ACM Comput. Surv. 48 (4) (2016) 62:1–62:33.
[25] Q. Xu, T. Mytkowicz, N.S. Kim, Approximate computing: A survey, IEEE Des. Test. 33 (1) (2016) 8–22.
[26] M. Shafique, R. Hafiz, S. Rehman, W. El-Harouni, J. Henkel, Cross-layer approximate computing: From logic to architectures, in: Design Automation Conference, 2016, pp. 99:1–99:6.
[27] N. Zhu, W.L. Goh, W. Zhang, K.S. Yeo, Z.H. Kong, Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 18 (8) (2010) 1225–1229.
[28] S.-L. Lu, Speeding up processing with approximation circuits, Computer 37 (3) (2004) 67–73.
[29] N. Zhu, W.L. Goh, G. Wang, K.S. Yeo, Enhanced low-power high-speed adder for error-tolerant application, in: International SoC Design Conference, 2010, pp. 323–327.
[30] A.B. Kahng, S. Kang, Accuracy-configurable adder for approximate arithmetic designs, in: Design Automation Conference, 2012, pp. 820–825.
[31] I.-C. Lin, Y.-M. Yang, C.-C. Lin, High-performance low-power carry speculative addition with variable latency, IEEE Trans. Very Large Scale Integr. (VLSI) Syst. 23 (9) (2015) 1591–1603.
[32] A. Cilardo, A new speculative addition architecture suitable for two's complement operations, in: Design, Automation and Test in Europe, 2009, pp. 664–669.
[33] J. Miao, A. Gerstlauer, M. Orshansky, Multi-level approximate logic synthesis under general error constraints, in: International Conference on Computer-Aided Design, 2014, pp. 504–510.
[34] NANgate, ⟨http://www.nangate.com/⟩.
[35] Synopsys, ⟨http://www.synopsys.com/⟩.
[36] C.-S. Lee, Y.-H. Kuo, P.-T. Yu, Weighted fuzzy mean filters for image processing, Fuzzy Sets Syst. 89 (2) (1997) 157–180.
[37] L.G. Roberts, Machine perception of three-dimensional solids, Ph.D. thesis, Massachusetts Institute of Technology, 1963.
[38] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, in: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1967, pp. 281–297.