A New Approximate Adder with Low Relative Error and Correct Sign Calculation

Junjun Hu and Weikang Qian
University of Michigan-Shanghai Jiao Tong University Joint Institute
Shanghai Jiao Tong University, Shanghai, China
Email: hujunjun81@hotmail.com, qianwk@sjtu.edu.cn

***Abstract*—Conventional precise adders need long delay and large power consumption to obtain accurate results. However, in recognition of the error tolerance of some applications such as multimedia processing and machine learning, a few recent works proposed approximate adders that generate inaccurate results occasionally to reduce the delay and power consumption. However, existing approximate adders rarely control the relative error and the potential sign error of the calculation results. In this paper, we propose a novel approximate adder that exploits the generate signals for carry speculation. Furthermore, we introduce a very low-cost error reduction module to effectively control the maximal relative error and a low-overhead sign correction module to fix the sign errors. Compared to the conventional adders, our adder is up to 4.3x faster and saves 47% power 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.**

I. Introduction

The continued scaling of CMOS technology causes the power consumption on a chip to continuously increase, which becomes a major bottleneck in sustaining the Moore's law. Thus, energy-efficiency has become a critical concern in designing VLSI circuits [1]. 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]–[8]. One feature of these applications is that they can tolerate some error in the computation results. For example, since a small amount of error embedded in images cannot be caught by human beings, we could allow an image processing application to occasionally produce an incorrect value for a pixel. The relaxation of the accuracy requirement for these applications potentially enlarges the design space, which may contain some solutions with smaller 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.

Since adders are a commonly-used hardware in those error-tolerant applications, in the previous works, people have proposed several different kinds of approximate adders. The Low-part-OR Adder (LOA) proposed in [9] applies the precise adder to the leading bits in the addends, while only uses a simple OR gate to obtain the sum bits for the trailing bits. This method, however, has the drawback of high error rate. Also, if the bit length of the accurate part is long, the time delay is still very large. Furthermore, the relative error will be large when doing addition on small values. Similarly, Error-Tolerant Adder I (ETAI) proposed in [3] utilizes a kind of modified XOR gate to compute the less significant part in the sum, which has the same drawbacks as LOA. In [10], the author proposed a k-bit lookÍCahead approximate adder to limit the carry propagate chain for each bit in a segment of length k, in order to reduce the critical path. However, the area cost of this adder is large due to the fact that the computation of each bit needs an individual carry generator. In [4], the authors proposed an Accuracy-Configurable Adder (ACA). However, the sub-adders used in it results in a large area cost and power consumption. The ETAII [5], ETAIV [11], Variable Latency Carry Select Addition (VLCSA-1) [6] and Carry Skip Approximate Adder (CSA) [7] use the segmentation method to truncate the carry propagation chain. The computation of each segment is based on the carry signal speculated using the bits in the lower segment. This method could effectively reduce the critical path delay, but, at the same time, it could introduce large relative error for the computation results, which may not be proper 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 CSA can be controlled, it still fails to solve the sign problem. Sign correction modules are introduced in VLSCA-2 [6] and the adder proposed in [12].However, both of them require an additional clock cycle and accurate adder to fix the sign error, which increases the delay and power consumption.

In this paper, we propose a novel approximate adder with an effective carry speculating method by exploiting generate signals. The segmentation of carry chain can significantly reduce the carry propagation delay, increasing the speed of computation. We further introduce a low-cost error reduction module into the adder, which significantly reduces the maximal relative error. We provide a rigorous analysis of two error measures, the error rate and the maximal relative error, of the proposed adder containing the error reduction module, which shows that both error measures are very low. Furthermore, we add a low-overhead sign correction module into the adder, which eliminates the potential sign error in 2's complement signed addition without the need of an additional clock cycle. Compared to the conventional adders, our adder is up to 4.3x faster and saves 47% power 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.

The remainder of this paper is organized as follows. In Section II, we introduce some background on conventional adders. In Section III, we describe our proposed approximate adder and give the error analysis. In Section IV, we present an enhanced adder with the sign correction. In Section V, we give the comparison of our proposed adder with two conventional adders and six other approximate adders. The conclusion is drawn in Section VI.

II. Background on Conventional Adder

We discuss the background on the conventional adder in this section. We use the following notations in the paper. A and B represent the two inputs of the adder. C_{in} represents the carry-in bit of the adder. a_i and b_i represent the i-th least significant bits of A and B, respectively. 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

$$p_i = a_i \oplus b_i, g_i = a_i \cdot b_i, k_i = \bar{a_i} \cdot \bar{b_i},$$

where s_i and c_i represent the sum bit and the carry-out bit at the i-th bit position, respectively. They are given by

$$\begin{aligned} &s_0 = p_0 \oplus C_{in}, c_0 = g_0 + p_0 \cdot C_{in} \\ &s_i = p_i \oplus C_{i-1}, c_i = g_0 + p_0 \cdot C_{i-1}, \quad i > 0. \end{aligned} \tag{1}$$

If $g_i = 1$, then $c_i = 1$, which indicates the "generate" of the carry-out. If $k_i = 1$, then $c_i = 0$, which indicates the "kill" of the carry-out. If $p_i = 1$, then $c_i = c_{i-1}$, which indicates the "propagate" of the carry-out from the $(i-1)$-th position to the i-th position. Applying Eq. (1) recursively, we can represent the carry-out signal c_i as

$$c_i = g_i + g_{i-1} \cdot p_i + \cdots + g_0 \cdot \prod_{j=1}^{i} p_j + C_{in} \cdot \prod_{j=0}^{i} p_j.$$

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 several 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 sum generator. The number of blocks is $m = \lceil \frac{n}{k} \rceil$. We order the rightmost block as block 0 and the leftmost block as block m-1.

978-3-9815370-4-8/DATE15/©2015 EDAA

In the figure, $A^i_{k-1:0}$ 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 propagate and the generate signals of the i-th block, respectively, which are produced by the P/G generators; $S^i_{k-1:0}$ represents the partial sum of the i-th block, which is produced by the sum generator; C^i_o represents the carry-out of the i-th block, which is the output of the carry generator. C^i_o feeds into the (i+1)-th carry generator and sum generator as the carry-in signal. We further use a^i_j, b^i_j, p^i_j, g^i_j, s^i_j and c^i_j to represent the first input bit, the second input bit, 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. In each segment, s^i_j and c^i_j are calculated based on the propagate and generate signals of the current block together with the carry-in C^{i-1}_o as follows

$$C^i_o = g^i_{k-1} + g^i_{k-2}p^i_k + \cdots + g^i_0 \cdot \prod_{x=1}^{k-1} p^i_x + C^{i-1}_o \cdot \prod_{x=0}^{k-1} p^i_x. \quad (2)$$

The sum bits s^i_j is calculated in the sum generator as follows

$$\begin{aligned} s^i_0 &= C^{i-1}_o \oplus p^i_0 \\ s^i_j &= c^i_{j-1} \oplus p^i_j, \quad j = 1, \ldots, k-1 \\ c^i_0 &= g^i_0 + p^i_0 \cdot C^{i-1}_o \\ c^i_j &= g^i_j + p^i_j \cdot c^i_{(j-1)}, \quad j = 1, \ldots, k-2, \end{aligned}$$

where c^i_j is the internal carry signals in the sum generator. We further define

$$pp^i = \prod_{x=0}^{k-1} p^i_x. \quad (3)$$

From the above equation, it can be seen that C^i_o depends on C^{i-1}_o if and only if pp^i=1. When pp^i=0, C^i_o is independent of C^{i-1}_o.

III. Proposed Approximate Adder

A. Approximate Adder Architecture

The longest critical path for the adder shown in Fig. 1(a) appears when the propagates at all the bit positions equal 1, which leads to $c_{n-1} = C_{in}$. Thus, to reduce 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 potential long carry chain, we introduce a carry speculation strategy in the carry generator: we use the generate signal of the most significant bit in the previous block as the carry-in signal of the carry generator, i.e., $g^{i-1}_{k-1} = a^{i-1}_{k-1} \cdot b^{i-1}_{k-1}$ is used as the carry-in signal of the carry generator in block i. Thus, the output of the carry generator in block i is now an approximate to the correct carry-out, which is denoted as $C^i_{apx,o}$

$$C^i_{apx,o} = g^i_{k-1} + \cdots + g^i_0 \cdot \prod_{x=1}^{k-1} p^i_x + g^{i-1}_{k-1} \cdot \prod_{x=0}^{k-1} p^i_x. \quad (4)$$

As shown in Fig. 1(b), the approximate carry-out is used as the carry-in signal to the sum generator. Thus, the sum bits at block i are also an approximate result, which we denote as $S^i_{apx,k-1:0}$.

The length of the longest carry propagate path is thus reduced from n to $2k$, which significantly reduces the circuit delay. However, $C^i_{apx,o}$ is just a speculated carry-out signal of the i-th block, which could be wrong.

We first study when the speculated carry-out is different from the correct carry-out. Clearly, this happens when the following two conditions both hold:

1. $g^{i-1}_{k-1} \neq C^{i-1}_o$, i.e., the carry-in is different from the correct carry-in.
2. pp^i (shown in Eq. (3)) equals 1, which results in the wrong carry-in to propagate to the carry-out.

Note that when the generate signal $g^{i-1}_{k-1} = 1$, the correct carry-in signal C^{i-1}_o must be 1. Thus, the first condition does not hold when $g^{i-1}_{k-1} = 1$. As a result, a necessary condition that the speculated carry-out could be wrong is that $pp^i = 1$ and $g^{i-1}_{k-1} = 0$. Furthermore, if the condition happens, the maximal relative error could be large. Indeed, consider a special case where the above necessary condition happens at the leftmost block, i.e., $pp^{m-1} = 1$ and $g^{m-2}_{k-1} = 0$. We further assume that at the same time, $C^{m-2}_{apx,o} = 1$. The calculation result of the adder

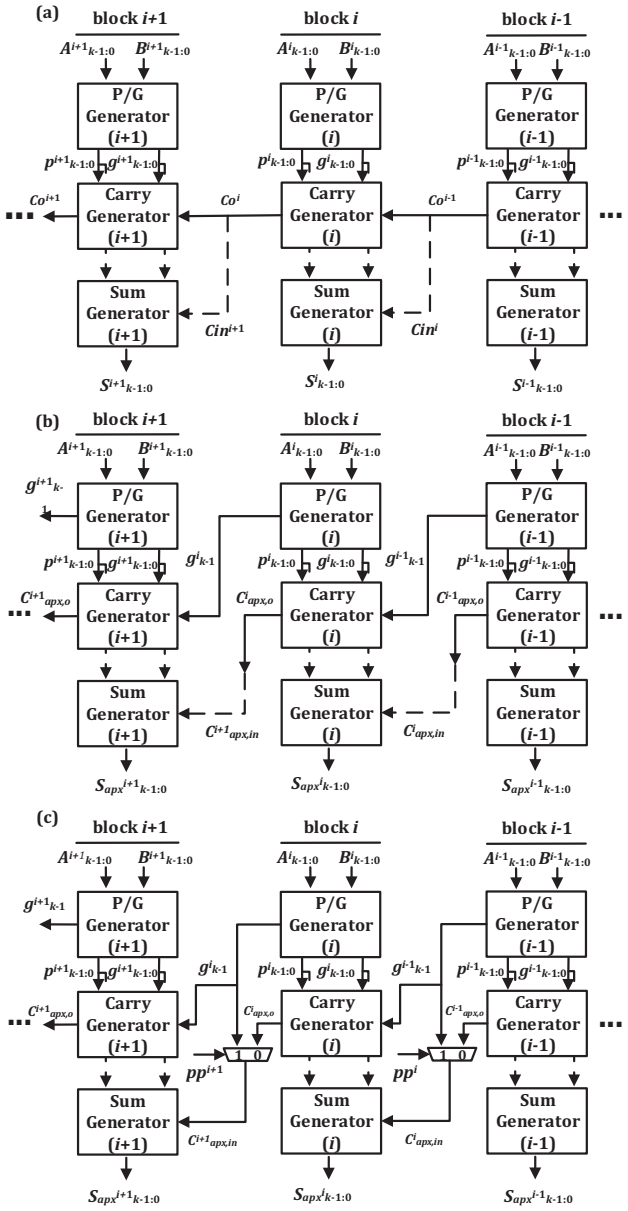

Fig. 1: Block diagram of adders: (a) Conventional adder. (b) Proposed approximate adder without error reduction. (c) Proposed approximate adder with error reduction.

is illustrated in Fig. 2 with $i = m - 1$. On the one hand, according to Eq. (4), we have $C^{m-1}_{apx,o} = 0$. Since the carry-in signal of calculating sum bits is $C^{m-2}_{apx,o} = 1$, we have $S^{m-1}_{apx,j} = 0$, for all $j = 0, \cdots, k-1$. On the other hand, based on Eq. (2), we can show that $C^{m-2}_o = 1$, which means that the correct carry-in to the $(m-1)$-th block is $C^{m-2}_o = 1$. This results the correct carry-out and the sum bits at block $(m-1)$ to be $C^{m-1}_o = 1$ and $s^{m-1}_j = 0$, for all $j = 0, \ldots, k-1$.

In summary, the approximate sum begins with $00\ldots0$, while the correct sum should begin with $10\ldots0$. This could lead to a relative error up to 100%. Similar large relative error can be obtained for cases where $pp^i = 1$, $g^{i-1}_{k-1} = 0$, $C^{i-1}_{apx,o} = 1$ and all the input bits to the blocks to the left of the block i are all 0. To reduce the large relative error, we further introduce an error reduction module into the approximate adder shown in Fig. 1(b). The modified approximate adder is shown in Fig. 1(c), where we only insert a 2-to-1 multiplexor between each pair of adjacent blocks. The two inputs of the multiplexor are the generate signal of the most significant bit in the previous block, g^{i-1}_{k-1}, and the carry-out signal produced by the carry generator in the previous block, $C^{i-1}_{apx,o}$. The selection signal is $pp^i = \prod_{j=0}^{k-1} p^i_j$. Such a signal has already been

generated in the carry generator of block i (as shown in Eq. (4)), so we do not need to include additional circuits to obtain it. The output signal of the multiplexor is used as the carry-in signal of the sum generator of the current block, which is denoted as C^i_{in}. This improvement comes from the idea of lowering the bit position where error occurs, which is achieved by forcing some bits in lower positions to be wrong rather than letting one bit in higher position to be incorrect.

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, of it is also small. Finally, we will analyze its delay.

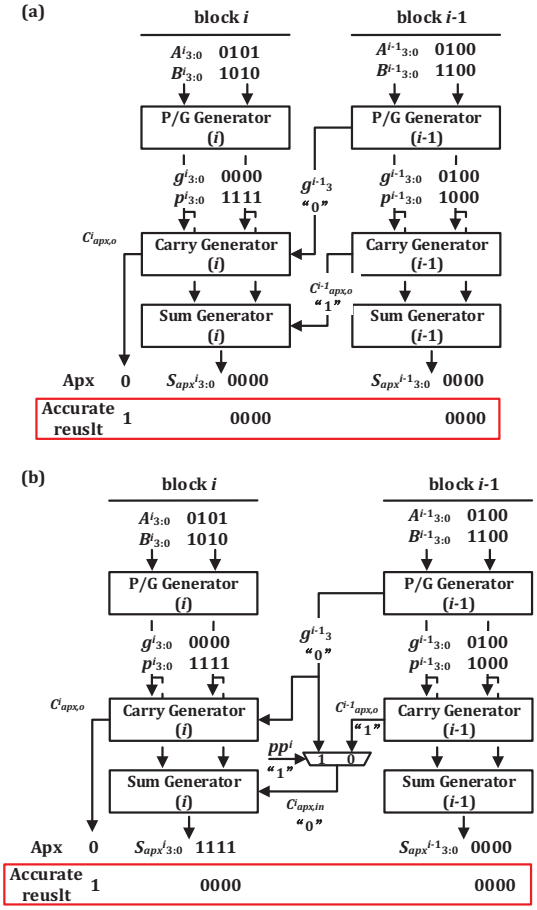

Fig. 2: An example of addition error due to wrong carry speculation: (a) without error reduction (b) with error reduction.

B. Relative Error Analysis

In this section, we will show the effect of the modification in reducing the relative error, which is defined as

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

where E_{re} is the relative error and S and S_{apx} are the correct and approximate outputs for the given inputs, respectively.

First, consider the case we introduced in the previous section which results in a large relative error for the original approximate adder, i.e., the case where $pp^{m-1} = 1$, $g^{m-2}_{k-1} = 0$, and $C^{m-2}_{apx,o} = 1$. The calculation result of the modified adder is shown in Fig. 2(b). Since $pp^{m-1} = 1$ and $g^{m-2}_{k-1} = 0$, the carry-in to the sum generator is also 0 due to the multiplexor, which causes $s^{m-1}_{apx,j} = 1$, for all $j = 0, \ldots, k-1$.

Furthermore, $C^{m-1}_{apx,o} = 0$. Thus, the approximate sum begins with $011\ldots1$. This is very close to the correct sum, which begins with $100\ldots0$. The relative error is significantly reduced, i.e., from up to 100% to $\frac{1}{2^k}$.

Indeed, we can show that not only for the above special cases, but also for all possible cases, our design can effectively limit the relative error to be no more than $\frac{1}{2^k}$, where k is the bit length of each block.

To show this, we will partition the m blocks into several groups. Suppose that a group is composed of blocks $i+s, \ldots, i$. Then, it has the following property: there exits a $t \in [0, s-1]$ such that $pp^{i+x} = 0$ for any $t+1 \le x \le s$ and $pp^{i+x} = 1$ for any $0 \le x \le t$. Not that the leftmost group may consist of block(s) with all pp^i equal to 1, and the rightmost group may consist of block(s) with all pp^i equal to 0. Fig. 3 gives an example of 8 blocks divided into 4 groups.

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

Fig. 3: An illustration of groups of bit blocks, used in relative error analysis.

Consider a group in the middle, which starts at block $i+s$ and ends at block i. By the definition of a group, we have $pp^{i+s+1} = 1$, $pp^{i+s} = 0$, $pp^i = 1$ and $pp^{i-1} = 0$. Also assume $pp^{i+x} = 0$ for $t+1 \le x \le s$ and $pp^{i+x} = 1$ for $0 \le x \le t$, where $0 \le t \le s-1$. Then, the carry-in to the sum generator at block i is $C^i_{apx,in} = g^{i-1}_{k-1}$ and the carry-in to the carry generator at block i is also g^{i-1}_{k-1}. Assume the approximate carry-out of block $i-1$ is $C^{i-1}_{apx,o}$. Since $pp^{i-1} = 0$, $C^{i-1}_{apx,o} = C^{i-1}_o$, which is the accurate carry-out. Consider the following three cases:

1) The case where $C^{i-1}_o = g^{i-1}_{k-1} = 0$. It is not hard to see that $C^{i+x}_{apx,in} = g^{i+x-1}_{k-1} = C^{i+x-1}_o = 0$ for all $0 \le x \le t$. For $t < x \le s$, since $pp^{i+x} = 0$, we have $C^{i+x}_{apx,in} = C^{i+x}_{apx,o} = C^{i+x-1}_o$. Therefore, no error occurs in the sum for this group.
2) The case where $C^{i-1}_o = 1$ and $g^{i-1}_{k-1} = 0$. Then, the carry-in fails to be propagated into the group. For any $0 \le x \le t$, the approximate sum bits of block $i+x$ will be $11\ldots1$, while the accurate ones should be $00\ldots0$. The carry-in signal of block $i+t+1$ will be $C^{i+t+1}_{apx,in} = C^{i+t}_{apx,o} = 0$, while the accurate value should be 1. Thus, the sum at block $i+t+1$ will be smaller than the accurate result by 1. No any other error occurs in block $i+x$, $t < x \le s$, since their pp signals are 0. Thus, the error of sum bit will be the difference between $100\ldots0$ (total $(t+1)\cdot k$ 0s) and $011\ldots1$ (total $(t+1)\cdot k$ 1s), which equals 1 at the lowest bit position of the group, i.e., bit position $i \cdot k$. Suppose the sum result of block $i+x$, $t < x \le s$ is S. Then the relative error of this group is:
$$E_{re} = \frac{1 \cdot 2^{i \cdot k}}{(S + 2^{(t+1)\cdot k}) \cdot 2^{i \cdot k}} = \frac{1}{S + 2^{(t+1)\cdot k}}, \quad t \ge 0.$$
The maximal relative error for the group in this case is $\frac{1}{2^k}$, which occurs when $S = 0$ and $t = 0$.
3) The case where $C^{i-1}_o = 1$ and $g^{i-1}_{k-1} = 1$. Then, we have $C^i_{apx,in} = g^{i-1}_{k-1} = C^{i-1}_o = 1$. The sum bits of block i are correct, which are $(00\ldots0)$, and the approximate carry-out signal is also correct which is $C^i_{apx,o} = C^i_o = 1$. If $t = 0$, then $pp^{i+x} = 0$ for any $1 \le x \le s$. Thus, the sum bits of these blocks will always be correct, since $C^{i+x}_{apx,in} = C^{i+x-1}_{apx,o} = C^{i+x-1}_o$. Therefore, when $t = 0$, no error occurs. However if $t > 0$, then the other blocks in this group will face the situation as in Case 2. The difference between approximate and accurate result will be "1" at bit position $(i+1)\cdot k$. Then the relative error of such group will be
$$E_{re} = \frac{1 \cdot 2^{(i+1)\cdot k}}{(S + 2^{(t+1)\cdot k}) \cdot 2^{i \cdot k}} = \frac{2^k}{S + 2^{(t+1)\cdot k}}, \quad t > 0.$$
The maximal relative error for this case is also $\frac{1}{2^k}$, which occurs when $S = 0$ and $t = 1$.

Besides, it can be shown that the relative error for the first and the last group are also bounded by $\frac{1}{2^k}$. Thus, the maximal relative error of our proposed approximate adder with error reduction is $\frac{1}{2^k}$.

C. Error Rate Analysis

Besides relative error, another commonly-used error metric is error rate, which is the ratio of incorrect outputs out of a total number of inputs [1]. 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) A_x: the approximate carry-out of block x, $C^x_{apx,o}$, is correct, and the approximate sums of the block x and all the less significant blocks are correct, i.e., $S^i_{apx,k-1:0}, 0 \le i \le x$ are correct.
2) B_x: $g^x_{k-1} = 0$, the correct carry-out $C^x_o = 0$, and the approximate sums of the block x and all the less significant blocks are correct.
3) C_x: $g^x_{k-1} = 1$, the correct carry-out $C^x_o = 1$, and the approximate sums of the block x and all the less significant blocks are correct.

Let $Pr(A_x) = a_x$, $Pr(B_x) = b_x$ and $Pr(C_x) = c_x$. Clearly, the result of the approximate adder is correct if and only if A_{m-1} happens. Thus, $Pr(C) = Pr(A_{m-1})$.

First, consider when the event A_x happens. We divide the event into two cases, based on whether $pp^x = 0$ or 1, since this determines what the carry-in to the sum generator of block x is.

1) If $pp^x = 0$, then $C^x_{apx,o}$ is equal to the correct carry-out C^x_o, since the carry-out does not depend on the carry-in to the current block. Moreover, since $C^{x-1}_{apx,o}$ is chosen to be the carry-in to the sum generator due to $pp^x = 0$, the correctness of the partial sum at block x requires the correctness of $C^{x-1}_{apx,o}$. Thus, the event A_x occurs if and only if the event A_{x-1} occurs.
2) If $pp^x = 1$, then based on Eq. (4), $C^x_{apx,o} = g^{x-1}_{k-1}$. At the same time, in the accurate adder, we have $C^x_o = C^{x-1}_o$. Thus, the correctness of $C^x_{apx,o}$ requires g^{x-1}_{k-1} to be the same as the correct carry-out of block (x-1), i.e., $g^{x-1}_{k-1} = C^{x-1}_o = 0$ or $g^{x-1}_{k-1} = C^{x-1}_o = 1$. Thus, the event A_x occurs if and only if either the event B_{x-1} or the event C_{x-1} occurs.Thus, we have

$$a_x = Pr(pp^x = 0) \cdot a_{x-1} + Pr(pp^x = 1) \cdot (b_{x-1} + c_{x-1}). \quad (5)$$

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

1) If $pp^x = 0$, then $C^{x-1}_{apx,o}$ is chosen to be the carry-in to the sum generator. To ensure the partial sum at block x to be correct, we require $C^{x-1}_{apx,o}$ to be the correct carry-out of the block (x-1). Furthermore, we require that the approximate sums of all the less significant blocks are correct. Thus, the event A_{x-1} must occur. What's more, $C^x_o = 0$ and $pp^x = 0$ mean that there is a kill signal in block x which propagates all the way to the most significant bit position of block x. Thus, we require one of the following events to occur: $k^x_{k-1} = 1, p^x_{k-1} = k^x_{k-2} = 1, \ldots,$ or $p^x_{k-1} = \cdots = p^x_1 = k^x_0 = 1$. Also, as long as one of the above events happens, it guarantees that $g^x_{k-1} = 0$.
2) If $pp^x = 1$, then of course $g^x_{k-1} = 0$. Due to the carry propagate, we have $C^{x-1}_o = C^x_o = 0$. Further, since the carry-in to block x is g^{x-1}_{k-1} and the approximate partial sum at block x is correct, we require g^{x-1}_{k-1} to be the same as the true carry-in C^{x-1}_o. Thus, $g^{x-1}_{k-1} = C^{x-1}_o = 0$. Thus, B_x happens if and only if B_{x-1} happens.

So we have

$$\begin{aligned} b_x =& [Pr(k^x_{k-1} = 1) + \cdots + Pr(p^x_{k-1} = \cdots = p^x_1 = k^x_0 = 1)] \cdot a_{x-1} \\ &+ Pr(pp^x = 1) \cdot b_{x-1}. \end{aligned} \quad (6)$$

Finally, we consider the occurrence of the event C_x. Under this event, we require $g^x_{k-1} = 1$. Once $g^x_{k-1} = 1$, $C^x_o = 1$ is always guaranteed. Furthermore, pp^x can only be 0. Thus, the carry-in to the sum generator in block x is always chosen to be $C^{x-1}_{apx,o}$. To ensure the partial sum at block x to be correct, we require $C^{x-1}_{apx,o}$ to be the correct carry-out of the block $(x-1)$. Also, we require that the approximate sums of all the less significant blocks are correct. Thus, the event $A(x-1)$ must occur. As a consequence, we have

$$c_x = Pr(g^x_{k-1} = 1) \cdot a_{x-1}. \quad (7)$$

Note that Eq. (5), (6) and (7) give us a recursive way to obtain a_x, b_x and c_x. To eventually obtain these values, we only need to get the values a_0, b_0 and c_0, corresponding to the base case.

To get the base case values, we analyze the least significant block, block 0. For this block, the carry-in of both the carry generator and sum generator are both the input C_{in}, so the carry-out signal, C^0_o, given by the carry generator, and the partial sum, $S^0_{apx,k-1:0}$, given by the sum generator, are always correct. It means that the event A_0 always happens, i.e., $a_0 = 1$.

Clearly, the event B_0 happens if and only if $g^0_{k-1} = 0$ and $C^0_o = 0$ both happen. This indicates one of the following events happens: $k^0_{k-1} = 1$, $p^0_{k-1} = k^0_{k-2} = 1, \ldots, p^0_{k-1} = \cdots = p^0_1 = k^0_0 = 1$, or $p^0_{k-1} = \cdots = p^0_0 = 1$ and $C_{in} = 0$. Thus, we have

$$\begin{aligned} b_0 =& Pr(k^0_{k-1} = 1) + \cdots + Pr(p^0_{k-1} = \cdots = p^0_1 = k^0_0 = 1) \\ &+ Pr(p^0_{k-1} = \cdots = p^0_0 = 1) \cdot Pr(C_{in} = 0) = \frac{1}{2}. \end{aligned}$$

The event C_0 happens if and only if $g^0_{k-1} = 1$ and $C^0_o = 1$ both happen. Since $g^0_{k-1} = 1$ guarantees that $C^0_o = 1$, we have $c_0 = Pr(g^0_{k-1} = 1) = \frac{1}{4}$. Thus, we obtain the full set of recursive equations:

$$\begin{aligned} &a_0 = 1, b_0 = \frac{1}{2}, c_0 = \frac{1}{4}; \\ &a_x = \left(1 - \frac{1}{2^k}\right) \cdot a_{x-1} + \frac{1}{2^k} \cdot (b_{x-1} + c_{x-1}),\ 0 < x < m; \\ &b_x = \frac{1}{2}\left(1 - \frac{1}{2^k}\right) \cdot a_{x-1} + \frac{1}{2^k} \cdot b_{x-1},\ 0 < x < m; \\ &c_x = \frac{1}{4} \cdot a_{x-1},\ 0 < x < m. \end{aligned}$$

Finally, we can get the error rate as

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

Fig. 4 shows some error rates of the proposed adder with different bit length and different block size. For the block size of 8 bits, the error rate will be in the magnitude between about 10^{-1} and 10^{-3}; for the block size of 16 bits, the error rate will be in the magnitude of no more than 10^{-4}. It shows that the error rate of our design is quite small.

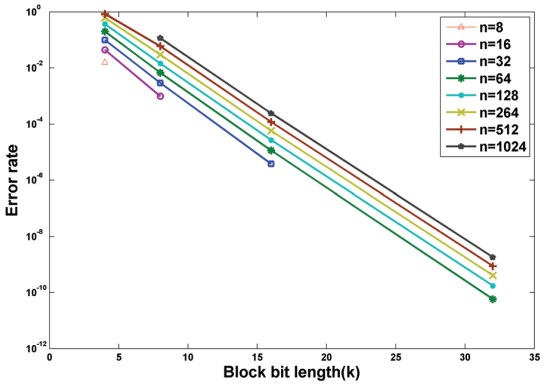

Fig. 4: Error rates of adders with different bit length and different block size.

D. Delay Analysis

We analyze the delay of the proposed approximate adder in this section. The addition operation works as follows. As shown in Fig. 1(c), the propagate signals and generate signals are first created by the P/G generators, and delivered to the carry generators and the sum generators. Next, all the carry generators in each blocks will simultaneously create the carry-out signals $(\cdots, C^{i+1}_{apx,o}, C^i_{apx,o}, C^{i-1}_{apx,o}, \cdots)$ with the speculated carry-in signal as $(\cdots, g^i_{k-1}, g^{i-1}_{k-1}, g^{i-2}_{k-1}, \cdots)$. Then, the carry-in signals to the sum generators $(\cdots, C^{i+1}_{in}, C^i_{in}, C^{i-1}_{in}, \cdots)$ are simultaneously chosen from the two sets of signals $(\cdots, C^i_{apx,o}, C^{i-1}_{apx,o}, C^{i-2}_{apx,o}, \cdots)$ and $(\cdots, g^i_{k-1}, g^{i-1}_{k-1}, g^{i-2}_{k-1}, \cdots)$ by the multiplexors based on the propagate chain detective signals $(\cdots, pp^{i+1}, pp^i, pp^{i-1}, \cdots)$ obtained from the carry generators. Finally, the sum generators take the carry-in signals to compute the approximate sums $(\cdots, S^{i+1}_{apx,k-1:0}, S^i_{apx,k-1:0}, S^{i-1}_{apx,k-1:0}, \cdots)$. Thus, the critical path delay of our proposed approximate adder can be calculated as

$$t_{apx} = t_{PG} + t_{CG} + t_{mux} + t_{SG},$$

where t_{PG}, t_{CG}, t_{mux} and t_{SG} are the delays of P/G generator, carry generator, multiplexor, and sum generator. Note that the sum generator could be implemented by any kind of conventional adders like RCA or carry-lookahead adder (CLA). In our implementation, we use RCA. Then, the asymptotic upper bounds for these delay values are

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

As demonstrated by our experimental results, when k is small the delay is also small. Then, using RCA to realize the sum generator has the advantage of small area and low power consumption. However, when k is large and we want to further reduce the delay, we can use CLA for the sum generator. Then both t_{SG} and t_{apx} reduce to $O(\log k)$.

IV. Sign Correction Module

The approximate adder introduced in the previous section is targeted for unsigned addition where the inputs are assumed to be two unsigned numbers. However, there exist many situations 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 design, based on the proposed unsigned approximate adder, we further proposed a low-overhead sign correction module to solve the potential sign error in the approximate addition. Our basic idea of fixing the sign error in 2's complement signed addition is to detect the situation when sign error occurs and then correct the leading bits that are wrong.

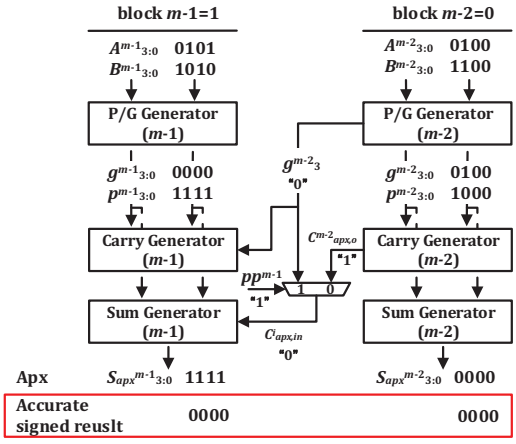

Fig. 5: Case of approximate addition with potential sign error.

Fig. 5 shows an example where sign error occurs when we perform a 2's complement signed addition using the approximate adder. With $m = 2$, the two signed inputs, represented as decimal numbers, are 84 and -84. The correct sum should be 0. However, the result by our approximate adder is -16, which is quite different from the correct one.

We first study what kind of conditions will cause the incorrect sign calculation for the adder shown in Fig. 1(c). In 2's complement signed addition, the sign bit is the most significant bit of the sum.

One necessary condition is that the propagate signals in the most significant block (i.e., block $m-1$) should be all true, i.e., $pp^{m-1} = 1$. Otherwise, $pp^{m-1} = 0$, which indicates either $p_{k-1}^{m-1} = 0$ or $\prod_{j=0}^{k-2} p_j^{m-1} = 0$. If the first case where $p_{k-1}^{m-1} = 0$ occurs, then the sign bit only depends on the two most significant bits in the inputs, i.e., a_{k-1}^{m-1} and b_{k-1}^{m-1}. Thus, the sign bit is always correct. If the second case where $\prod_{j=0}^{k-2} p_j^{m-1} = 0$ occurs, then the carry-in to the most significant position,c_{k-2}^{m-1}, is always correct, no matter of the value of the speculated carry-in to block $(m-1)$, $C_{apx,in}^{m-1}$. Thus, the sign bit is always correct.

Now we assume that $pp^{m-1} = 1$. Then, there exists an s $\in [1, m]$ such that $pp^{m-1} = \cdots = pp^{m-s} = 1$ and $pp^{m-s-1} = 0$, which means that the propagate signals in blocks $m-1, \ldots, m-s$ are all true, while a propagate signal at block $m-s-1$ is false[1]. There are two cases depending on whether C_o^{m-s-1} is 0 or 1.

1. The case where $C_o^{m-s-1} = 0$. Due to $pp^{m-1} = \cdots = pp^{m-s} = 1$, the correct partial sums of the s most significant blocks are all $11\ldots1$, i.e., $S_{apx,k-1:0}^{i} = 11\ldots1$, $m-s < i < m-1$. Since $C_o^{m-s-1} = 0$, we must have $g_{k-1}^{m-s-1} = 0$. Thus, the speculated carry-in to block $(m-s)$ is $C_{apx,in}^{m-s} = g_{k-1}^{m-s-1} = 0$, due to $pp^{m-s} = 1$. Furthermore, since $pp^{m-1} = \cdots = pp^{m-s} = 1$, we have $g_{k-1}^{m-2} = \ldots = g_{k-1}^{m-s} = 0$ and $C_{apx,in}^{m-1} = \ldots = C_{apx,in}^{m-s+1} = 0$. Based on the block diagram of the proposed approximate adder shown in Fig. 1a, the approximate partial sums of the s most significant blocks are all $11\ldots1$, which are the same as the correct ones. Clearly, no sign error occurs.
2. The case where $C_o^{m-s-1} = 1$. Due to $pp^{m-1} = \cdots = pp^{m-s} = 1$, the correct partial sums of the s most significant blocks are all $00\ldots0$. First, we consider the situation where $s \geq 2$. Since $pp^{m-1} = \cdots = pp^{m-s} = 1$, we have $g_{k-1}^{m-2} = \ldots = g_{k-1}^{m-s} = 0$ and $C_{apx,in}^{m-1} = \ldots = C_{apx,in}^{m-s+1} = 0$. Based on the block diagram of the proposed approximate adder shown in Fig. 1(a), the approximate partial sums of the $(s-1)$ most significant blocks are all $11\ldots1$, which are different from the correct ones. Clearly, a sign error occurs. Now consider the remaining situation where $s = 1$. If the generate signal at position $k-1$ in block $m-2$ is 0, i.e., $g_{k-1}^{m-2} = 0$, it can be easily shown that the approximate partial sum of the most significant blocks is $11\ldots1$. Thus, a sign error occurs. Otherwise, if $g_{k-1}^{m-2} = 1$, the approximate partial sum of the most significant blocks is $00\ldots0$, which means that no sign error occurs.

By the above analysis, we can see that a sign error occurs only if there exists an s $\in [1, m]$ such that $pp^{m-1} = \cdots = pp^{m-s} = 1$, $pp^{m-s-1} = 0$, and $C_o^{m-s-1} = 1$. We notice that the occurrence of the conditions $pp^{m-s-1} = 0$ and $C_o^{m-s-1} = 1$ indicates that $C_{apx,o}^{m-s-1} = 1$. Thus, a necessary condition for a sign error to occur is that there exists an s $\in [1, m]$ such that $pp^{m-1} = \cdots = pp^{m-s} = 1$ and $C_{apx,o}^{m-s-1} = 1$.

To correct the sign error, we first define the following signal for checking the necessary condition for any $1 \leq s \leq m$:

$$sp^{m-s} = pp^{m-1} \cdot \cdots \cdot pp^{m-s} \cdot C_{apx,o}^{m-s-1},$$

which can be realized by an $(s+1)$-input AND gate, as shown in Fig. 6(a). If there is $1 \leq s \leq m$ such that $sp^{m-s} = 1$, then the partial approximate sums of blocks $m-1, \ldots, m-s$ may be incorrect. Furthermore, if $sp^{m-s} = 1$, then it indicates that $pp^{m-1} = \cdots = pp^{m-s} = 1$ and $C_{apx,o}^{m-s-1} = 1$. Note that $C_{apx,o}^{m-s-1} = 1$ indicates the correct carry-out at block $(m-s-1)$ is $C_o^{m-s-1} = 1$. As a consequence, the correct partial sums of the blocks $m-1, \ldots, m-s$ must all be $00\ldots0$. This means that if $sp^{m-s} = 1$, no matter whether the partial approximate sums of blocks $m-1, \ldots, m-s$ are correct or incorrect, they should be set to $00\ldots0$, to achieve the correct calculation. Thus, the partial sum of block i should be forced to $00\ldots0$ when one of $sp^i, sp^{i-1}, \ldots, sp^0$ is 1. Thus, we modify the approximate partial sum at block i as follows:

$$S_{sapx,k-1:0}^{i} = S_{apx,k-1:0}^{i} \cdot \overline{CS^i}, 0 < i < m,$$

where CS^i is a sign-correction signal for each block, defined as

$$CS^i = sp^i + \cdots + sp^0, 0 \leq i < m,$$

which can be realized by an i+1-input OR gate, as shown in Fig. 6(b). The final sign-correct approximate sum bits, $S_{sapx,k-1:0}^{i}$, can then be obtained by ANDing the unsigned approximate sum bits with the negation of the signal CS^i, as shown in Fig. 6(c).

This sign correction module offers us an alternative choice to apply the approximate adder for 2's complement signed addition. It ensures that the result given by the approximate addition has no sign error. Its maximal relative error can be shown to be equal to that of the approximate adder shown in Fig. 1(c). Its error rate is smaller than the approximate adder shown in Fig. 1(c).

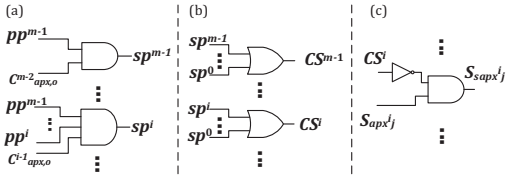

Fig. 6: Circuits of the sign correction module.

V. Experimental Results

We evaluate our proposed approximate adders (with and without the correction module) in this section. They are designed in Verilog HDL and synthesized with a 45nm NAN-gate cell library [13] using Design Compiler [14]. We chose the length of the addition to be 32 and the size of each block to be 4.

For comparison purpose, we also implemented two conventional adders (RCA and CLA) and six other approximate adders(LOA [9],

[1]We define: $pp^{-1} = 0$, $g_{k-1}^{-1} = C_{apx,o}^{-1} = C_o^{-1} = C_{in}$

ETAII [5], VLCSA-1 [6], ACA [4], LUA [10] and CSA [7]) for a 32-bit addition. We used the same block size 4 for ETAII, VLCSA-1, and CSA. For LOA, we set its accurate and inaccurate segments to both have 16 bits. For ACA, we use 8-bit sub-adders, to make it have an equivalent block size of 4. For LUA, we set the look-ahead as 4 bits. Furthermore, the same designs of the 4-bit RCA and the 4-bit carry-lookahead module were used as sum generator and carry generator, respectively, in the ETAII, VLCSA-1, CSA, LUA, and our proposed adders. For VLCSA-1 and ACA, they have their own error detection and correction modules, but such modules incur additional area overhead and require an additional clock cycle to realize the error correction. Thus, we did not include such modules in our implementation of them. We compared six metrics of our proposed adders to those of the other adders, which are area, delay, power consumption, error rate, maximal relative error, and the capability to ensure the correct sign calculation. The results are listed in Table I.

TABLE I: Comparison of the proposed adders with other 32-bit adders ($k = 4$).

Adders	Area	Delay (ns)	Power	Error rate(%)	Max relative error(%)	Solve sign error
RCA	161.7	5.62	29.0	0	0	Yes
CLA	587.9	1.28	79.4	0	0	Yes
LOA(16-16)	99.5	3.09	15.8	99	50	No
ETAII	208.5	0.94	28.2	16.94	100	No
VLCSA-1	389.7	0.93	44.7	20.51	100	No
ACA	283.0	1.55	31.1	16.34	100	No
LUA	322.4	0.85	40.0	34.64	100	No
CSA [a]	240.5	1.29	33.6	0.91	100	No
Proposed [b]	238.6	1.23	33.1	10.03	6.25	No
Proposed [c]	307.8	1.3	41.7	<10.03	6.25	Yes

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

Although RCA has smaller area and power consumption than our proposed approximate adders, our designs are much faster than RCA. Our adders have similar delay as CLA, but the area and power consumption of our adders are much smaller than CLA. Indeed, our proposed approximate adder with the sign correction module is 4.3x faster than RCA and saves 47% power over CLA. Compared with the six approximate adders, our adders also show advantages. Although like RCA, LOA has smaller area and power consumption than ours, its delay is much larger. Because of its simple structure, its error rate and relative error are also quite large. Compared with the other five approximate adders (i.e., ETAII, VLCSA-1, ACA, LUA, and CSA), our adders have comparable area, delay, and power consumption. The error rates of our adders are only inferior to CSA among all the six approximate adders. It is because CSA looks ahead two blocks for carry speculation, while ours look ahead only one block.

The most significant advantages of our adders are demonstrated in the last two columns in the table, which show the maximal relative error and the capability to ensure the correct sign calculation. We can see that all the other approximate adders have large maximal relative error and are unable to always guarantee the correct sign calculation when applied for 2's complement signed addition. In contrast, the maximal relative errors of our designs are limited to a small value. In the experiment, we choose $k = 4$. According to our analysis, the maximal relative error is $\frac{1}{2^k} = 6.25\%$. With a larger block size, the maximal relative error can be further reduced. Furthermore, by including the proposed low-overhead sign correction module, we can ensure that our approximate adder has no sign error in signed additions, which cannot be achieved by all the other approximate adders.

Table II shows another set of experiments on 32-bit adders with block length $k = 8$. Similar as the situation where $n = 32$ and $k = 4$, our proposed adders have comparable area, delay, and power consumption compared with the other adders. Our main advantages are the small maximal relative error and the capability to ensure the correct sign calculation in the signed addition. Comparing the proposed adders with $k = 4$ (Table I) to the ones with $k = 8$ (Table II), we can see that when the block length k increases, the delay of the proposed adders increases. This is because the critical path of each block increases. With the block length k increasing, the area of the proposed adder *without* sign correction module increases because the increase of block length causes the area of the carry generator and the sum generator to increase. In contrast, with the block length k increasing, the area of the proposed adder *with* sign correction module decreases. This is because the increase of block length leads to the decrease of the number of blocks and hence, the area of the sign correction module is reduced. Furthermore, we can see that the increase of the block length significantly reduces the error rate and maximal relative error of the proposed adders.

TABLE II: Comparison of the proposed adders with other 32-bit adders ($k = 8$).

Adders	Area	Delay (ns)	Power	Error rate(%)	Max relative error(%)	Solve sign error
RCA	161.7	5.62	29.0	0	0	Yes
CLA	587.9	1.28	79.4	0	0	Yes
LOA(16-16)	99.5	3.09	15.8	99	50	No
ETAII	223.4	1.6	29.56	0.4	100	No
VLCSA-1	384.9	1.98	50.69	0.58	100	No
ACA	242.6	2.97	31.14	0.39	100	No
LUA	482.8	1.19	56.9	2.22	100	No
CSA [a]	250.1	2.09	35.56	<0.1	100	No
Proposed [b]	248.2	2.03	34.31	0.29	0.39	No
Proposed [c]	270.26	2.09	39.72	<0.29	0.39	Yes

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

VI. Conclusion

In this paper, we proposed a novel approximate adder which significantly reduces the delay and power consumption over the conventional accurate adders. For a 32-bit addition, our adder is up to 4.3x faster and saves up to 47% power, compared with those accurate adders. The efficient carry speculating and error reduction technique proposed in this work can make our adders have a low error rate, and, more importantly, a low maximal relative error. Furthermore, a low-overhead enhancement to our adder was proposed that ensures the correct sign calculation in 2's complement signed addition.

Acknowledgement

This work is supported by National Natural Science Foundation of China (NSFC) under Grant No. 61204042 and 61472243.

References

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Test Symposium (ETS), 2013 18th IEEE European*. IEEE, 2013, pp. 1–6.

[2] R. Venkatesan, A. Agarwal, K. Roy, and A. Raghunathan, "Macaco: Modeling and analysis of circuits for approximate computing," in *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2011, pp. 667–673.

[3] N. Zhu, W. L. Goh, W. Zhang, K. S. Yeo, and Z. H. Kong, "Design of low-power high-speed truncation-error-tolerant adder and its application in digital signal processing," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 18, no. 8, pp. 1225–1229, 2010.

[4] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 820–825.

[5] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," in *Integrated Circuits, ISIC'09. Proceedings of the 2009 12th International Symposium on*. IEEE, 2009, pp. 69–72.

[6] K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2012*. IEEE, 2012, pp. 1257–1262.

[7] Y. Kim, Y. Zhang, and P. Li, "An energy efficient approximate adder with carry skip for error resilient neuromorphic vlsi systems," in *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 2013, pp. 130–137.

[8] J. Miao, K. He, A. Gerstlauer, and M. Orshansky, "Modeling and synthesis of quality-energy optimal approximate adders," in *Proceedings of the International Conference on Computer-Aided Design*. ACM, 2012, pp. 728–735.

[9] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 57, no. 4, pp. 850–862, 2010.

[10] S.-L. Lu, "Speeding up processing with approximation circuits," *Computer*, vol. 37, no. 3, pp. 67–73, 2004.

[11] N. Zhu, W. L. Goh, G. Wang, and K. S. Yeo, "Enhanced low-power high-speed adder for error-tolerant application," in *SoC Design Conference (ISOCC), 2010 International*. IEEE, 2010, pp. 323–327.

[12] A. Cilardo, "A new speculative addition architecture suitable for two's complement operations," in *Proceedings of the Conference on Design, Automation and Test in Europe*. European Design and Automation Association, 2009, pp. 664–669.

[13] http://www.nangate.com/.

[14] http://www.synopsys.com/.