

# High Performance Reliable Variable Latency Carry Select Addition

Kai Du<sup>†</sup> Peter Varman<sup>†</sup> Kartik Mohanram<sup>‡</sup>

<sup>†</sup>Department of Electrical and Computer Engineering, Rice University, Houston

<sup>‡</sup>Department of Electrical and Computer Engineering, University of Pittsburgh, Pittsburgh

kai.du@rice.edu p.jv@rice.edu kartik.mohanram@gmail.com

## Abstract

Speculative adders have attracted strong interest for reducing critical path delays to sub-logarithmic delays by exploiting the trade-offs between reliability and performance. Speculative adders also find use in the design of reliable variable latency adders, which combine speculation with error correction to achieve high performance for low area overhead over traditional adders. This paper describes speculative carry select addition (SCSA), a novel function speculation technique for the design of low error-rate speculative adders and low overhead, high performance, reliable variable latency adders. We develop an analytical model for the error rate of SCSA to facilitate both design exploration and convergence. We show that for an error rate of 0.01% (0.25%), SCSA-based speculative addition is 10% faster than the DesignWare adder with up to 43% (56%) area reduction. Further, on average, variable latency addition using SCSA-based speculative adders is 10% faster than the DesignWare adder with area requirements of -19% to 16% (-17% to 29%) for unsigned random (signed Gaussian) inputs.

## 1. Introduction

Theoretical research has established that the lower bound on the critical path delay of the adder has complexity  $O(\log n)$ , where  $n$  is the adder width. Several high performance adder designs have achieved logarithmic delays [1]. Whereas theoretical bounds indicate that no traditional adder can achieve sub-logarithmic delay, it has been shown that speculative adders can achieve sub-logarithmic delays by neglecting rare input patterns that activate the critical paths [2–4]. Furthermore, by augmenting speculative adders with error detection and recovery, one can construct reliable variable latency adders whose average performance is very close to speculative adders [5–8].

Speculative adders are built upon the observation that *the critical path is rarely activated in traditional adders*. In traditional adders, each output depends on all previous (lower or equal significance) bits. In particular, the most significant output depends on all the  $n$  bits, where  $n$  is the adder width. In contrast, in speculative adders [2–6], each output depends only on the previous  $k$  bits rather than all previous bits, where  $k$  is slightly larger than  $O(\log n)$ . However, the cumulative error grows linearly with  $n$  since each speculative output can independently be in error. Moreover, the calculation of each speculative output requires an individual  $k$ -bit adder; hence, such designs also incur large area overhead and large fanout at the primary inputs. Techniques such as effective sharing [5] can mitigate but not eliminate fanout and area problems. Although the speculative adder in [9] can mitigate the area problem, it incurs a fairly high error rate that limits its application. For applications where errors cannot be tolerated, a reliable variable latency adder can be built upon the speculative adder by adding error detection and recovery [5–8]. When error detection flags no error,

the speculative result is correct; when error detection flags an error, error recovery provides correct results within one or more extra cycles. Variable latency adders are designed mainly for unsigned random inputs [5, 7, 8]. If the error rate is low, the average latency of the variable latency adder should be similar to the speculative one. However, existing variable latency adders have several drawbacks. The critical path delay of error detection is always longer than that of the speculative adder [5, 6]. Hence, the benefit of speculation is limited by the delay of error detection [7, 8]. Furthermore, the area overhead of the error detection and recovery circuitry is not negligible in practice.

This paper makes the following three contributions. First, we describe a novel function speculation technique, called speculative carry select addition (SCSA). The key idea is to segment the chain of propagate signals in addition into blocks of the same size. Specifically, the input bits of addends are segmented into blocks, and the carry bits between blocks are selectively truncated to 0. All outputs of a block, instead of each output, are speculated together, which mitigates the area overhead problem. This paper extends the preliminary work described in [10] by deriving an analytical model to determine the error rate of SCSA. We present a high performance speculative adder design for low error rates (0.01% and 0.25%). Second, we describe a reliable variable latency adder design that augments the speculative adder with error detection and recovery. The speculative adder produces correct results in a single cycle in most cases, and error recovery provides correct results in an extra cycle (worst case). In this paper, the preliminary work in [10] is extended to optimize the design of the block adder with two advantages: (i) the critical path delay of error detection is no higher than that of the speculative adder and (ii) the error detection and recovery circuitry incurs low area overhead by using intermediate results from the speculative adder. Finally, since the existing variable latency adders [5, 7, 8] are designed mainly for unsigned random inputs, we propose a modified variable latency adder suitable for both unsigned random and signed Gaussian inputs.

Simulations using 10 million unsigned random inputs were used to validate the high accuracy of the analytical error model. Simulation results indicate that for an error rate of 0.01% (0.25%), the SCSA-based speculative addition is 10% faster than the DesignWare adder with up to 43% (56%) area reduction. Simulation results also suggest that on average, variable latency addition using SCSA-based speculative adders is about 10% faster than the DesignWare adder with area requirements of -19% to 16% (-16% to 29%) for unsigned random (signed Gaussian) inputs.

This paper is organized as follows. Section 2 provides the motivation for the speculative and reliable variable latency adder. Section 3 introduces SCSA. Section 4 describes the speculative adder design. Section 5 proposes the corresponding reliable variable latency adder design. Section 6 presents a modified reliable variable latency adder design suitable for both unsigned uniform and signed Gaussian inputs. Section 7 validates the above models and designs. Section 8 summarizes our contributions.

$$\begin{array}{r}
A: a_{15} a_{14} a_{13} a_{12} a_{11} a_{10} a_9 a_8 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0 \\
+ B: b_{15} b_{14} b_{13} b_{12} b_{11} b_{10} b_9 b_8 b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0 \\
\hline
S: s_{15} s_{14} s_{13} s_{12} s_{11} s_{10} s_9 s_8 s_7 s_6 s_5 s_4 s_3 s_2 s_1 s_0
\end{array}$$

Figure 1: An example of 16-bit unsigned binary addition.

## 2. Background and motivation

Due to the importance of addition, various adders have been designed for high performance and low power [1], including *ripple carry adder*, *look-ahead adder*, and *parallel prefix adder*. There is an interesting observation regarding adders and indeed many other designs: *The critical path is rarely activated*. This observation indicates that the traditional worst-case design methodology may require a large design margin. Speculative adders have achieved significantly higher performance by neglecting rare input patterns that exercise the critical paths [2–4]. Furthermore, error-free variable latency adders can be constructed from speculative adders by adding error detection and recovery to achieve average performance comparable to speculative adders [5–8].

Variable latency adders mainly fall into two categories. The first category detects and removes all timing errors at design time. Telescopic units [11] fall in this category. However, the synthesis of an exact function that covers all input patterns that violate the timing constraint is expensive in practice. It has been shown that this problem is  $\mathcal{NP}$ -complete [12], which limits the application of this technique to large circuits. The second category is built upon the function speculation, wherein the original logic function is replaced by an approximate logic function. In the asynchronous domain, [13] first proposed a variable latency adder. In the synchronous domain, it has been suggested that the complete logic function be replaced by a simplified logic function that provides correct results most of the time [2–4]. However, the techniques in [2–4] have no error correction capability and may also suffer from large area and large fanout at the primary inputs. Recently, [5] proposed an error-free variable latency adder design wherein the speculative addition is similar to [2–4]. [6] studied an extension of [5] for signed non-random inputs. Both [5] and [6] have the same area and fanout problems noted above. Furthermore, in [5, 6], the critical path delay of error detection is always longer than that of the speculative adder. The approach in [5] was generalized in [7], wherein an automatic synthesis technique that transforms a combinational design to a two-stage variable-latency design was described. This was extended in [8] to multi-stage function speculation. In both [7] and [8], speculation is limited by the latency and overhead of error detection. Further, [7, 8] are both designed for unsigned random inputs. Finally, although the speculative adder design in [9] can mitigate the area problem, it exhibits a fairly high error rate that limits its application.

Besides, the closest approaches to our work is [8]. In [8], a combinational adder is transformed into a multi-cycle variable-latency one, which requires multi-cycle timing analysis. This adder is designed for unsigned random inputs. Besides, it is difficult to incorporate this technique within the traditional EDA flow due to complicated multi-cycle timing constraints. In contrast, the SCSA-based variable latency design is suitable for both unsigned random and signed Gaussian inputs, and is a simple deterministic design with one or two cycles of latency for addition.

Finally, speculative or reliable variable latency designs have been discussed for exploiting the tradeoffs between reliability and power. The Razor technique [14] dynamically adjusts the supply voltage by detecting and correcting errors. Similar work [15] has been done for signal processing applications. A non-uniform voltage scaling approach, called probabilistic arithmetic [16], was proposed to save energy.

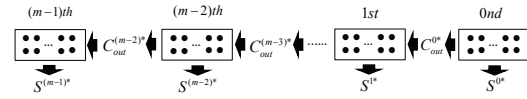


Figure 2: Dot graph to illustrate the operation of SCSA in [10]. A dot represents an input bit.

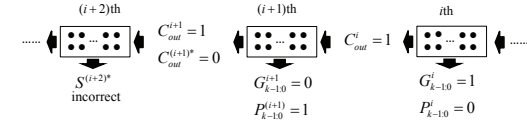


Figure 3: Error when  $G_{k-1:0} = 1$  in the  $i$ th window and  $P_{k-1:0} = 1$  in the  $(i+1)$ th window.

## 3. Speculative carry select addition (SCSA)

In this section, we discuss the proposed speculative addition, called speculative carry select addition (SCSA). SCSA comes from the observation of the carry chain in addition. During the discussion, we employ unsigned binary addition and random inputs to illustrate SCSA. An example of 16-bit addition is shown in Figure 1. We represent two input numbers as A and B. The  $i$ th least significant bit of A and B are represented as  $a_i$  and  $b_i$ , respectively. Then we define propagate ( $P_i$ ) and generate ( $G_i$ ) signals at the  $i$ th bit position as

$$P_i = a_i \oplus b_i \quad \text{and} \quad G_i = a_i b_i.$$

The sum bit  $s_i$  and carry-out (carry) bit  $c_i$  at the  $i$ th bit position are given by

$$s_i = P_i \oplus c_{i-1} \quad \text{and} \quad c_i = G_i + P_i c_{i-1}.$$

If  $P_i = 1$ ,  $c_i = c_{i-1}$ , which indicates that changing the value of  $c_{i-1}$  directly changes the value of  $c_i$ . This condition is defined as  $c_i$  depends on  $c_{i-1}$ , denoted as  $c_i \rightarrow c_{i-1}$ . All other conditions are defined as  $c_i$  does not depend on  $c_{i-1}$ , denoted as  $c_i \nrightarrow c_{i-1}$ . Let us consider how  $c_i$  depends on  $c_{i-k}$ ,  $0 < k \leq i$ . If  $\exists P_j = 0$ ,  $i-k+1 \leq j \leq i$ ,  $c_i \nrightarrow c_{i-k}$ . In other words,  $c_i \rightarrow c_{i-k}$  iff  $\forall P_j = 1$ ,  $i-k+1 \leq j \leq i$ . The number of consecutive propagate signals  $P_i$  with value 1 is called the *carry chain length*. Since  $\mathbf{P}(P_i = 1) = 1/2$ , the probability of a  $k$ -bit carry chain is  $1/2^k$ . The average longest carry chain length in an  $n$ -bit addition has been extensively studied [1]: it is widely recognized that the average longest carry chain length in an  $n$ -bit addition is  $O(\log n)$  for uniform inputs. This interesting fact suggests that it is possible to quickly and accurately estimate the output bit using only several consecutive input bits.

### 3.1 Operation of SCSA

Long carry chains rarely happen in addition for random inputs. In other words, by grouping input bits into blocks as shown in Figure 2, the carry chain length can be made comparable to the block size with high probability. Input bits are divided into blocks of the same size, as shown in Figure 2. A block, called a *window*, includes several consecutive input bits. The SCSA operation for adder width  $n$  and window size  $k$  is shown in Figure 2. The total number of windows is  $m = \lceil n/k \rceil$ . The carry-out bit of the  $i$ th window is called  $C_{\text{out}}^i$ ,  $0 \leq i < m$ . The carry-out bit of a window is speculated using only all  $k$  input bits of the window. Combining 1 speculative carry-in bit with  $k$  input bits of the window,  $k$  speculative sum bits of the window are computed. Any bit position in the window is affected by at least previous  $k$  bit positions. As argued earlier, the probability that an output bit depends on more than  $k$  previous bit positions is less than  $1/2^k$ . However, the relation between the window size and the overall error rate of the adder remains unclear. An analytical error model for SCSA is presented below and provides critical guidance for SCSA-based adder design.

### 3.2 Error rate analysis

An error occurs if a window produces a group generate signal with value 1 and the next window produces a group propagate signal with value 1, as shown in Figure 3. Specifically, for adder width  $n$  and window size  $k$ , the total number of windows is  $m = \lceil \frac{n}{k} \rceil$ . The group propagate and generate (P/G) signals at the  $j$ th bit position of the  $i$ th window, denoted by  $P_{j:0}^i$  and  $G_{j:0}^i$  are given by

$$P_{j:0}^i = \prod_{l=0}^j P_l^i \quad \text{and} \quad G_{j:0}^i = G_j^i + P_j^i G_{j-1}^i + \dots + G_0^i \prod_{l=1}^j P_l^i,$$

where  $P_l^i$  and  $G_l^i$  are the P/G signals at the  $l$ th bit position of the  $i$ th window. The group P/G signals of the  $i$ th window are defined as  $P_{k-1:0}^i$  and  $G_{k-1:0}^i$ ,  $0 \leq i < m$ . The carry-out bit of the  $i$ th window,  $C_{\text{out}}^i$ , is given by

$$C_{\text{out}}^i = G_{k-1:0}^i + P_{k-1:0}^i C_{\text{out}}^{i-1}, \quad 1 \leq i < m. \quad (1)$$

In SCSSA,  $C_{\text{out}}^{i-1}$  is truncated to 0, and  $C_{\text{out}}^i$  is approximated as  $C_{\text{out}}^{i*}$

$$C_{\text{out}}^{i*} = G_{k-1:0}^i, \quad 1 \leq i < m. \quad (2)$$

As shown in Figure 3, the  $i$ th window has a group generate signal with value 1,  $G_{k-1:0}^i = 1$ . According to (1), we have

$$C_{\text{out}}^i = 1.$$

$P_{k-1:0}^{i+1} = 1$  indicates that  $C_{\text{out}}^i$  passes through the  $(i+1)$ th window, which also implies  $G_{k-1:0}^{i+1} = 0$ . The carry-out bit of the  $(i+1)$ th window  $C_{\text{out}}^{i+1}$  is approximated using (2) as

$$C_{\text{out}}^{i+1*} = G_{k-1:0}^{i+1} = 0.$$

In fact, the correct carry-out bit of the  $(i+1)$ th window  $C_{\text{out}}^{i+1}$  is given by

$$\begin{aligned} C_{\text{out}}^{i+1} &= G_{k-1:0}^{i+1} + P_{k-1:0}^{i+1} C_{\text{out}}^i \\ &= C_{\text{out}}^i = 1. \end{aligned}$$

Thus,  $C_{\text{out}}^{i+1} \neq C_{\text{out}}^{i+1*}$ . SCSSA incorrectly speculates the result if  $P_{k-1:0}^{i+1} G_{k-1:0}^i = 1$ .

The probability of the above event is calculated as follows. Since group P/G signals from two different windows are fully independent, the error probability,  $\mathbf{P}(P_{k-1:0}^{i+1} G_{k-1:0}^i = 1)$  is given by

$$\mathbf{P}(P_{k-1:0}^{i+1} G_{k-1:0}^i = 1) = \mathbf{P}(P_{k-1:0}^{i+1} = 1) \mathbf{P}(G_{k-1:0}^i = 1). \quad (3)$$

Also, the probabilities that group P/G signals equal 1 is

$$\begin{aligned} \mathbf{P}(P_{k-1:0}^{i+1} = 1) &= \mathbf{P}\left(\prod_{j=0}^{k-1} P_j^{i+1} = 1\right) \\ &= \prod_{j=0}^{k-1} \mathbf{P}(P_j^{i+1} = 1) = (1/2)^k, \end{aligned} \quad (4)$$

$$\begin{aligned} \mathbf{P}(G_{k-1:0}^i = 1) &= \mathbf{P}(G_{k-1}^i + \dots + G_0^i \prod_{j=1}^{k-1} P_j^i = 1) \\ &= \mathbf{P}(G_{k-1}^i = 1) + \dots + \mathbf{P}(G_0^i \prod_{j=1}^{k-1} P_j^i = 1) \\ &= (1/2)[1 - (1/2)^k], \end{aligned} \quad (5)$$

where  $G_{k-1}^i, P_{k-1}^i G_{k-2}^i, \dots, G_0^i \prod_{j=1}^{k-1} P_j^i$  are mutually exclusive. Based on (4) and (5), (3) is given by

$$\begin{aligned} \mathbf{P}(P_{k-1:0}^{i+1} G_{k-1:0}^i = 1) &= \mathbf{P}(P_{k-1:0}^{i+1} = 1) \mathbf{P}(G_{k-1:0}^i = 1) \\ &= (1/2)^{k+1} [1 - (1/2)^k]. \end{aligned} \quad (6)$$

The total error probability for SCSSA can be approximated by summing up probabilities of these events for all windows. The approximate total error probability is stated as

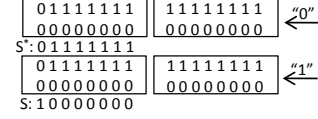


Figure 4: Example to illustrate low error magnitude of SCSSA. The error magnitude is  $1/2^7$ .

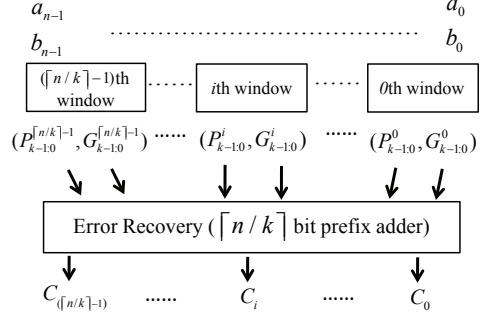


Figure 5: Speculative adder and the error recovery block in [10]. Intermediate values from the speculative adder largely simplifies the design of error recovery.

$$\begin{aligned} P_{\text{err}}^* &= \sum_{i=0}^{\lceil \frac{n}{k} \rceil - 2} \mathbf{P}(P_{k-1:0}^{i+1} G_{k-1:0}^i = 1) \\ &= \sum_{i=0}^{\lceil \frac{n}{k} \rceil - 2} (1/2)^{k+1} [1 - (1/2)^k]. \end{aligned} \quad (7)$$

(7) describes the relation between the window size and error rate. The error rate becomes negligible if the window size is large enough. For example, if  $n = 256, k = 16, P_{\text{err}}^* \simeq 0.01\%$ .

Moreover, the error magnitude (the ratio of the error to the correct result) is low when an error occurs. An example is shown in Figure 4. If the carry-in bit of the right window is truncated to 0, the sum bits of the left window are speculated as 01111111. However, the actual carry-in bit for the right window is 1, and the correct sum bits are 10000000. The error magnitude is  $1/2^7$ . This error affects all outputs of the left window rather than an individual output, which amortizes the effect of the error. In contrast, if only an individual output is incorrect, the error magnitude can be as large as the significance of the most significant bit in addition, e.g., the speculative addition in [5].

### 4. SCSSA-based speculative adder design

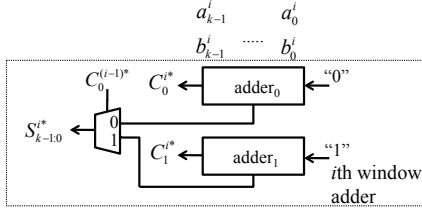
A high performance, low area overhead speculative adder (SCSSA-based speculative adder) can be built upon SCSSA for applications where errors can be tolerated, e.g., data mining, cryptography, and signal processing.

The  $n$ -bit adder is equally segmented into  $\lceil \frac{n}{k} \rceil$   $k$ -bit windows, as shown in the upper part of Figure 5. As shown in Figure 6, the window adder consists of two small adders,  $\text{adder}_0$  and  $\text{adder}_1$ . The two small adders can be implemented using any traditional adder, such as the Kogge-Stone adder. The  $j$ th sum bit of the  $i$ th window,  $s_{j,i}^*$ , is estimated as

$$\begin{aligned} s_{j,i}^* &= P_j^i \oplus [G_{j-1:0}^i + P_{j-1:0}^i C_{\text{out}}^{i-1*}] \\ &= P_j^i \oplus [G_{j-1:0}^i + P_{j-1:0}^i G_{k-1:0}^{i-1}], \quad 0 \leq j < k \end{aligned}$$

where  $C_{\text{out}}^{i-1*} = G_{k-1:0}^{i-1}$ . We employ a carry-select structure to compute the two cases when  $G_{k-1:0}^{i-1}$  is 1 and 0

$$\begin{aligned} s_{j,1}^* &= P_j^i \oplus [G_{j-1:0}^i + P_{j-1:0}^i], \\ s_{j,0}^* &= P_j^i \oplus G_{j-1:0}^i, \quad 0 \leq j < k. \end{aligned}$$



**Figure 6: Window adder implementation in speculative adder.**

Then we select one of them as the speculative sum bit when  $C_{out}^{i-1*}$  is ready, given by  $C_0^{i-1*}$  in Figure 6.

Assume we implement the small adder in the window adder using Kogge-Stone. The complexity of the critical path delay of the speculative adder is  $O(\log k)$ , which is similar to that of a  $k$ -bit traditional prefix adder. In contrast, the critical path delay of an  $n$ -bit traditional adder has complexity  $O(\log n)$ . Hence, the speculative adder can be significantly faster than a traditional adder. At each step in the speculative adder, there are at most  $k$  levels for intermediate group P/G signals. The total number of windows is  $\lceil \frac{n}{k} \rceil$ . Thus, the space complexity of an  $n$ -bit speculative adder is  $O(\lceil \frac{n}{k} \rceil k \log k)$ . This is in contrast to traditional fast adders such as the Kogge-Stone adder, which has a space complexity of  $O(n \log n)$ .

## 5. SCSA-based variable latency adder design

For applications in which errors cannot be tolerated, a reliable variable latency adder can be built upon the SCSA-based speculative adder by adding error detection and recovery. We call this adder as variable latency carry selection adder (VLCSA). Error detection flags if speculation is incorrect. Error recovery produces the correct result when error detection flags an error. VLCSA works with one or two cycles of latency for addition, whose operation is similar to [5]. The average performance of VLCSA is close to the speculative one. We first describe a VLCSA design directly derived from SCSA, called VLCSA-1. In the next section, we discuss a modified VLCSA design suitable for both unsigned random and signed Gaussian inputs, called VLCSA-2.

### 5.1 Error detection

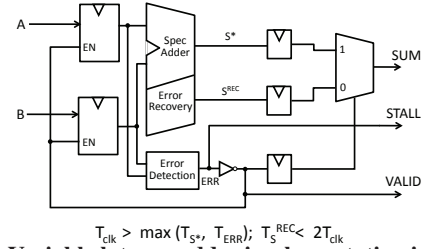
Error detection flags all the events that the speculation is incorrect. Based on the analytical error model for SCSA, the error detection signal is stated as

$$ERR = \sum_{i=0}^{\lceil \frac{n}{k} \rceil - 2} P_{k-1:0}^{i+1} G_{k-1:0}^i.$$

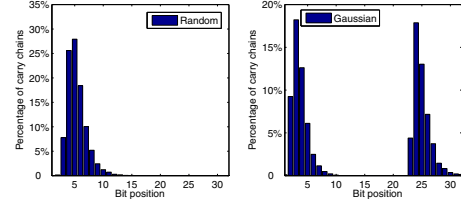
The ERR signal flags an error if  $\exists P_{k-1:0}^{i+1} G_{k-1:0}^i = 1$ ,  $0 \leq i < \lceil \frac{n}{k} \rceil - 1$ . It takes  $O(\log k)$  steps to generate the group P/G signals of the window, and an additional  $O(\log \lceil \frac{n}{k} \rceil)$  steps to produce the ERR signal. The critical path delay of error detection has complexity  $O(\log \lceil \frac{n}{k} \rceil + \log k)$ . We observe that  $\log \lceil \frac{n}{k} \rceil$  is quite small in practice. For example, when  $n = 512$  and  $k = 17$ ,  $\log \lceil \frac{n}{k} \rceil \simeq 5$ . In contrast, it takes several constant steps to compute sum bits after generating group P/G signals in the speculative adder. Error detection has comparable or even shorter critical path delays than the speculative adder in VLCSA-1, so the benefit of speculation is maintained. Besides, the space complexity of error detection is  $O(\lceil \frac{n}{k} \rceil \log \lceil \frac{n}{k} \rceil)$ . The computation uses only simple gates and requires low area overhead.

### 5.2 Error recovery

Error recovery produces correct results when error detection flags errors. The lower part of Figure 5 shows an area-efficient implementation for error recovery using intermediate results from the



**Figure 7: Variable latency adder implementation in [10], over-arching idea is similar to [5].**



**Figure 8: Example of statistics of carry chain lengths in 32-bit addition for unsigned random and signed Gaussian inputs.**

speculative adder. An  $\lceil \frac{n}{k} \rceil$ -bit prefix adder takes the group P/G signals of windows as inputs and computes the accurate carry-out bits for all windows. The speculative adder has also computed the group P/G signals at each bit position of the window. Thus, the correct sum bits are computed using the outputs of this prefix adder. The space complexity of this prefix adder is  $O(\lceil \frac{n}{k} \rceil \log \lceil \frac{n}{k} \rceil)$ . The complexity of the critical path delay of error recovery, through the speculative adder and the prefix adder, is  $O(\log k + \log \lceil \frac{n}{k} \rceil)$ .

### 5.3 Operation of VLCSA-1

VLCSA-1 — with core architecture similar to [5] — is shown in Figure 7. The clock cycle,  $T_{clk}$ , is slightly longer than the critical path delays of the speculative adder and error detection. The speculative result and error detection signal ERR are computed in a single cycle. If ERR flags no error, the speculative result is correct. Otherwise, error recovery produces the correct result in a single additional cycle. The effective cycles required by this design for addition,  $T_{ave}$ , is given by

$$T_{ave} = T_{clk}(1 - P_{err}) + 2T_{clk}P_{err},$$

where  $P_{err}$  is the error rate of VLCSA-1. If  $P_{err}$  is small, say 0.01%,  $T_{ave} \simeq T_{clk}$  and the average performance of the variable latency adder is close to that of the speculative adder.

## 6. VLCSA-2 for signed Gaussian inputs

In previous sections, speculative and variable latency adders were designed for unsigned random inputs. However, in practice the distribution of inputs will affect the performance of speculative and variable latency adders. We employ mathematical distributions to approximately profile the practical inputs. In practice, it has been reported in [6, 17] that (i) the 2's complement representation is widely used for signed numbers and (ii) small numbers appear more frequently than large ones. In particular, signed Gaussian inputs in 2's complement representation capture the basics of the practical workloads such as [6]. An example of statistics of carry chain lengths for different inputs is shown in Figure 8. For signed Gaussian inputs, a nontrivial portion of carry chains is as long as the adder size. These long carry chains significantly increase the error rate of the speculative addition and make VLCSA even slower than that of the traditional adder. Therefore, we propose a modified VLCSA design, called VLCSA-2, for both unsigned random and

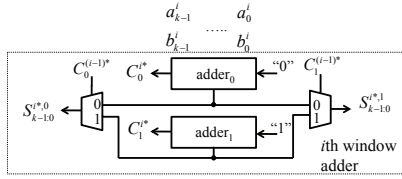


Figure 9: Window adder implementation in VLCSA-2.

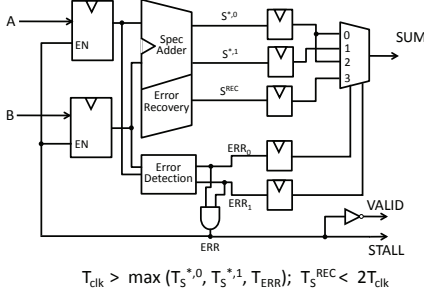


Figure 10: VLCSA-2 implementation.

signed Gaussian inputs.

The key idea of VLCSA-2 is to correctly speculate when very long carry chains occur. We design VLCSA-2 based on the key observation for 2's complement: long carry chains are usually triggered by the addition of a small positive and a small negative number that then affects the most significant bit position.

First, we describe the modified window adder shown in Figure 9. Compared with Figure 6, another speculative result,  $S^{i*,1}$ , is calculated, which is selected by one speculative carry-out bit of the previous window adder,  $C_1^{i-1*}$ .

Second, we introduce an additional error detection signal,  $ERR_1$ , to detect long carry chains. The error detection signal in VLCSA-1 is denoted by  $ERR_0$ .  $ERR_1$  is defined as

$$ERR_1 = \sum_{i=0}^{\lceil \frac{n}{k} \rceil - 2} \overline{P}_{k-1:0}^{i+1} P_{k-1:0}^i.$$

The main idea of this error detection is similar to [6]. Let us see why  $ERR_0$  and  $ERR_1$  can detect long carry chains. (i)  $ERR_0 = 1$ ,  $ERR_1 = 0$ :  $\exists P_{k-1:0}^{i+1} G_{k-1:0}^i = 1, \forall \overline{P}_{k-1:0}^{i+1} P_{k-1:0}^i = 0$ , which implies that a long carry chain generates at a bit position and propagates to the MSB position. In other words, if  $ERR_0 = 1$ ,  $ERR_1 = 0$ , we know that a long carry chain occurs and does not incur errors.

(ii)  $ERR_0 = 1$ ,  $ERR_1 = 1$ :  $\exists P_{k-1:0}^{i+1} G_{k-1:0}^i = 1, \exists \overline{P}_{k-1:0}^{i+1} P_{k-1:0}^i = 1$ , which indicates that a carry chain starts at a bit position and ends before reaching the MSB position. Error detection flags an error. (iii)  $ERR_0 = 0$ . The speculative result is correct and the same as that in VLCSA-1.

Finally, VLCSA-2 is shown in Figure 10. Error detection signals,  $ERR_0$  and  $ERR_1$ , are used to select speculative results and flag errors. (i)  $ERR_0 = 0$ . The speculative result,  $S^{*,0}$ , is correct. (ii)  $ERR_0 = 1$ ,  $ERR_1 = 0$ . The speculative result,  $S^{*,1}$ , is correct. (iii)  $ERR_0 = 1$ ,  $ERR_1 = 1$ . The speculation is incorrect, and error recovery provides the correct result.

## 7. Simulation and validation

We have implemented C++ programs which take the adder width  $n$  and the window size  $k$ , and generate Verilog files for speculative and variable latency adders. Circuits are synthesized using a common standard library for UMC 65 nm CMOS technology in a commercial synthesis tool. The DesignWare building block IP can generate high-quality adder designs for timing, area, and power. The DesignWare adder is synthesized for the minimal achievable

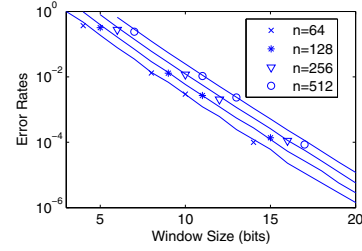


Figure 11: Comparison of analytical error model for SCSA and simulation results for different adder widths ( $n$ ).

delay. We implemented a hybrid Kogge-Stone carry-select adder and observed that the DesignWare adder is faster than the hybrid one. The details of the DesignWare building block IP are available here [18]. We compare the delay and area of SCSA-based speculative adder, VLCSA-1 and VLCSA-2 with the DesignWare adder.

### 7.1 Error model validation

For different adder widths, we compare the analytical error model for SCSA (solid lines) with the error rate obtained by running Monte Carlo simulations for 10 million unsigned random inputs (marked points) in Figure 11. From the figure, it is clear that there is good agreement between the analytical model and the simulation results.

### 7.2 Error rates for Gaussian inputs

We also estimate the error rates of speculative addition in VLCSA-1 and VLCSA-2 by running Monte Carlo simulations for 1 million 2's complement Gaussian inputs. For the Gaussian distribution, the mean is  $\mu = 0$  and the standard deviation is  $\sigma = 2^n$ , where  $n$  is the adder width. The error rate in VLCSA-1 is greater than 1% and we obtain more than a 100 $\times$  reduction in the error rate with VLCSA-2 for the same inputs, as shown in Table 1.

adder width	window size	$P_{\text{err}}(ERR_0 = 1, ERR_1 = 1)$
64	14	< 0.01%
128	15	< 0.01%
256	16	< 0.01%
512	17	< 0.01%

Table 1: Error rates in VLCSA-2 for signed Gaussian inputs, according to simulation results.

### 7.3 Comparison with the DesignWare adder

The parameters of the speculative adder are reported in Table 2. Two small adders of the window adder are implemented using DesignWare IP block. In comparison to the DesignWare adder, we targeted 10% critical path delay reduction and zero area overhead during synthesis.

adder width	window size $P_{\text{err}}=0.01\%$	window size $P_{\text{err}}=0.25\%$
64	14	10
128	15	11
256	16	12
512	17	13

Table 2: Parameters of SCSA-based speculative adder and VLCSA-1 for error rates of 0.01% and 0.25% for unsigned random inputs, according to error models and simulation results.

As shown in Figure 12, the delays of the speculative adder are 10% lower than those of the DesignWare adder for error rates 0.01% and 0.25%. For an error rate of 0.01%, as the adder width increases,

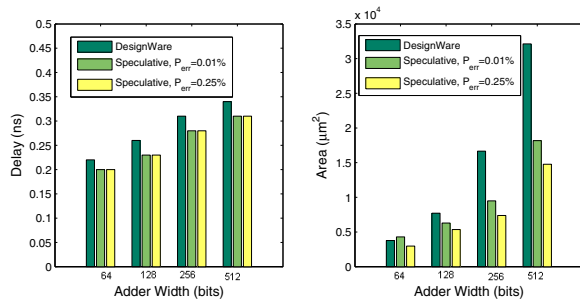


Figure 12: Comparison of delay and area of SCSA-based speculative adder and DesignWare adder.

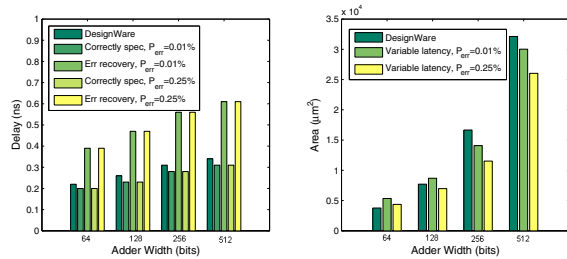


Figure 13: Comparison of delay and area of VLCSA-1 and DesignWare adder.

the area of the speculative adder can be 43% smaller than that of the DesignWare adder. For an error rate of 0.25%, the area of the speculative adder is 21% to 56% smaller than that of the DesignWare adder. The speculative adder with a lower error rate has larger area than the one with a higher error rate. This tradeoff between the error rate and area can be employed to rapidly reduce area by slightly increasing the error rate.

Next we compare VLCSA-1 with the DesignWare adder. The parameters of the speculative adder in VLCSA-1 also follow Table 2. The maximum critical path delay of the speculative adder and error detection block is stated as the “correctly speculated” delay. As shown in Figure 13, the critical path delays of VLCSA-1 are 10% lower than those of the DesignWare adder when speculation is correct. The critical path delays of the error recovery block are less than twice of the “correctly speculated” delays. For an error rate 0.25% (0.01%), VLCSA-1 has area requirements of -19% to 16% (-6% to 42%) over the DesignWare adder. If the error rate is 0.25% instead of 0.01%, on average, we can save 17.39% area by increasing the average latency by 0.12%. The tradeoff between the error rate and area is valuable for saving area.

adder width	window size $P_{err}=0.01\%$	window size $P_{err}=0.25\%$
64	13	9
128	13	9
256	13	9
512	13	9

Table 3: Parameters of VLCSA-2 for the error rates of 0.01% and 0.25%, according to simulation results.

Finally, we compare VLCSA-2 with the DesignWare adder. The parameters of VLCSA-2 are reported in Table 3. As shown in Figure 14, the critical path delays of VLCSA-2 are 10% lower than those of the DesignWare adder when speculation is correct. For an error rate of 0.25% (0.01%), VLCSA-2 has area requirements of -17% to 29% (1% to 62%) over the DesignWare adder. The area overhead of VLCSA-2 is larger than that of VLCSA-1 due to additional circuitry for speculative addition and error detection.

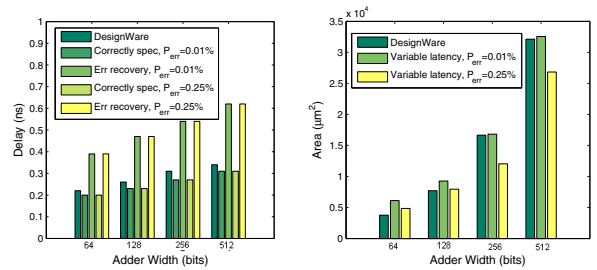


Figure 14: Comparison of delay and area of VLCSA-2 and DesignWare adder.

## 8. Conclusion

In this paper, we described a novel function speculation technique, called speculative carry select addition (SCSA). We described a speculative adder based on SCSA. We also proposed a reliable variable latency adder that augments the speculative adder with error detection and recovery. Furthermore, we described a modified variable latency adder suitable for both unsigned random and signed Gaussian inputs. Simulation results suggest that the speculative adder can be faster and smaller than the DesignWare adder for very low error rates and that the reliable variable latency adder can outperform the DesignWare adder in both delay and area. We plan to generalize SCSA for other arithmetic operations such as multiplication and multi-operand addition.

## References

- [1] I. Koren, *Computer Arithmetic Algorithms*. A K Peters, Ltd., 2002.
- [2] T. Liu and S. Lu, “Performance improvement with circuit-level speculation,” in *Proc. Intl. Symp. on Microarchitecture*, pp. 348–355, 2000.
- [3] S. Lu, “Speeding up processing with approximation circuits,” in *Computer*, vol. 37, pp. 67–73, 2004.
- [4] T. Austin *et al.*, “Opportunities and challenges for better than worst-case design,” in *Proc. Asia and South Pacific Design Automation Conference*, pp. 2–7, 2005.
- [5] A. K. Verma *et al.*, “Variable latency speculative addition: a new paradigm for arithmetic circuit design,” in *Proc. Design, Automation and Test in Europe*, pp. 1250–1255, 2008.
- [6] A. Cilaro, “A new speculative addition architecture suitable for two’s complement operations,” in *Proc. Design, Automation and Test in Europe*, pp. 664–669, 2009.
- [7] Bañeres *et al.*, “Variable-latency design by function speculation,” in *Proc. Design, Automation and Test in Europe*, pp. 1704–1709, 2009.
- [8] Y. Liu *et al.*, “Design methodology of variable latency adders with multi-stage function speculation,” in *Proc. Intl. Symp. on Quality Electronic Design*, pp. 824–830, 2010.
- [9] N. Zhu *et al.*, “An enhanced low-power high-speed adder for error-tolerant application,” in *Proc. 12th Intl. Symp. on Integrated Circuits*, pp. 69–72, 2009.
- [10] K. Du *et al.*, “Static window addition: a new paradigm for the design of variable latency adders,” in *Proc. Intl. Conf. on Computer Design*, pp. 455–456, 2011.
- [11] L. Benini *et al.*, “Telescopic units: A new paradigm for performance optimization of VLSI designs,” *IEEE Trans. Computer-aided Design*, vol. 17, no. 3, pp. 220–232, 1998.
- [12] Y. Su *et al.*, “An efficient mechanism for performance optimization of variable-latency designs,” in *Proc. Design Automation Conference*, pp. 976–981, 2007.
- [13] S. Nowick *et al.*, “Speculative completion for the design of high-performance asynchronous dynamic adders,” in *Proc. of 3rd Intl. Symp. on Advanced Research in Asynchronous Circuits and Systems*, pp. 210–223, 1997.
- [14] D. Ernst *et al.*, “Razor: a low-power pipeline based on circuit-level timing speculation,” in *Proc. Intl. Symp. on Microarchitecture*, pp. 7–18, 2003.
- [15] R. Hegde and N. R. Shanbhag, “Soft digital signal processing,” *IEEE Trans. VLSI Systems*, vol. 9, no. 6, pp. 813–823, 2001.
- [16] L. Chakrapani *et al.*, “Highly energy and performance efficient embedded computing through approximately correct arithmetic,” in *Proc. Intl. Conf. on Compilers, Architectures and Synthesis for Embedded Systems*, pp. 187–196, 2008.
- [17] D. Kelly and J. Phillips, “Arithmetic data value speculation,” in *Advances In Computer Systems Architecture, Lecture Notes in Computer Science*, pp. 353–366, 2005.
- [18] “DesignWare building block IP user guide.” <https://www.synopsys.com/dw/doc.php/doc/dwf/manuals/dwbb.userguide.pdf>.