

RICE UNIVERSITY

**High Performance Reliable
Variable Latency Carry Select Addition**

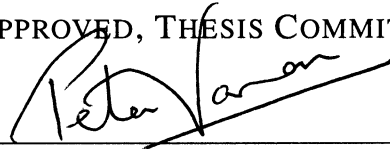
by

Kai Du

A THESIS SUBMITTED
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE

Master of Science

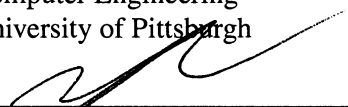
APPROVED, THESIS COMMITTEE:



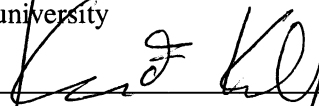
Peter Varman, Chairman
Professor of Electrical and Computer
Engineering
Rice University



Kartik Mohanram
Associate Professor of Electrical and
Computer Engineering
University of Pittsburgh



Lin Zhong
Associate Professor of Electrical and
Computer Engineering
Rice University



Kevin Kelly
Associate Professor of Electrical and
Computer Engineering
Rice University



Ray Simar
Professor in the Practice of Electrical and
Computer Engineering
Rice University

HOUSTON, TEXAS

NOVEMBER, 2011

Abstract

High Performance Reliable Variable Latency Carry Select Addition

by

Kai Du

This thesis describes the design and the optimization of a low overhead, high performance variable latency carry select adder. Previous researchers believed that the traditional adder has reached the theoretical speed bound. However, a considerable portion of hardware resources of the traditional adder is only used in the worst case. Based on this observation, variable latency adders have been proposed to improve on the theoretical limit, but such adders incur significant area overhead. By combining previous variable latency adders with carry select addition, this work describes a novel variable latency carry select adder. Applying carry select addition in the variable latency adder design significantly reduces the area overhead and increases its performance. This variable latency adder is faster and smaller than previous variable latency adders. Furthermore, this variable latency adder can be optimized to be faster and smaller than the fastest adder generated by the Synopsys DesignWare building block IP.

Acknowledgements

I express my sincere gratitude to my advisors, Prof. Kartik Mohanram and Prof. Peter Varman for all their guidance, patience and support throughout this thesis. I am also grateful for Prof. Lin Zhong, Prof. Kevin Kelly and Prof. Ray Simar for their time and valuable comments on this thesis.

I am also grateful to my friends at Rice, Xuebei Yang, Mihir Choudhury, Masoud Ros-tami and Ahmed ElNably. Thanks for their time and help.

Finally, I thank my families for their love and support.

Contents

Abstract	ii
Acknowledgements	iii
List of Figures	vi
List of Tables	viii
1 Introduction	1
2 Background	4
3 Speculative carry select addition (SCSA)	7
3.1 Operation of SCSA	8
3.2 Error rate analysis	9
3.3 Error magnitude analysis	13
4 SCSA-based speculative adder	14
4.1 Window adder design	15
4.2 Implementation of SCSA	16
4.3 Time and space complexity of SCSA	17
5 Variable latency carry selection adder (VLCSA)	19
5.1 Error detection design	19

5.2	Error recovery design	22
5.3	Operation of VLCSA	23
6	Modified VLCSA (VLCSA 2)	25
6.1	Motivation	25
6.2	Profiling practical inputs	26
6.3	Approximating practical inputs	28
6.4	Key idea of Modified VLCSA	30
6.5	Modified speculative addition	31
6.6	Modified error detection	32
6.7	Operation of VLCSA 2	34
7	Results	37
7.1	Simulation setup	37
7.2	Error model validation	37
7.3	Error rates for 2's complement Gaussian inputs	38
7.4	Comparison with existing variable latency adders	40
7.4.1	Speculative addition in VLCSA 1 VS speculative addition in VLSA	40
7.4.2	VLCSA 1 VS VLSA	42
7.5	Comparison with DesignWare adder	44
7.5.1	Speculative addition in VLCSA 1 vs DesignWare adder	44
7.5.2	VLCSA 1 vs DesignWare adder	45
7.5.3	VLCSA 2 vs DesignWare adder	47
8	Conclusion	50

List of Figures

3.1	Dot graph for addition. A dot represents an input bit.	7
3.2	Input bits grouped into windows.	8
3.3	Dot graph to illustrate the operation of SWA.	9
3.4	Error when $G_{k-1:0} = 1$ in the i^{th} window and $P_{k-1:0} = 1$ in the $(i + 1)^{th}$ window.	10
3.5	Predicted error rates for different adder widths (n) and window sizes.	12
3.6	Example to illustrate low error magnitude.	13
4.1	Speculative adder is consisted of $\lceil \frac{n}{k} \rceil$ k -bit small adders.	14
4.2	Window adder implementation in speculative adder. Carry-select adder structure is employed to increase performance.	15
4.3	Tree structure for calculating the group P/G signals of Kogge-Stone adder.	16
5.1	Error detection implementation using 2-input AND and OR gates.	20
5.2	Area-efficient implementation for error recovery using intermediate results from the speculative adder.	22
5.3	Variable latency adder implementation, similar to [17]	24
6.1	Example of statistics of carry chain lengths for unsigned random inputs. The adder size is 32 bits.	26
6.2	Examples of statistics of carry chain lengths from a cryptographic workload [6].	27

6.3	Example of statistics of carry chain lengths for 2's complement random inputs. The adder size is 32 bits.	28
6.4	Example of statistics of carry chain lengths for unsigned Gaussian inputs. The adder size is 32 bits.	29
6.5	Example of statistics of carry chain lengths for 2's complement Gaussian inputs. The adder size is 32 bits.	30
6.6	Window adder implementation in SCSA 2.	31
6.7	Modified error detection implementation using 2-input AND and OR gates.	32
6.8	VLCSA 2 implementation.	34
7.1	Comparison of analytical error model for SCSA and simulation results for different adder widths (n).	38
7.2	Comparison of delay of speculative adders and Kogge-Stone adder.	41
7.3	Comparison of area of speculative adders and Kogge-Stone adder.	41
7.4	Comparison of delay of variable latency adders and Kogge-Stone adder.	42
7.5	Comparison of area of variable latency adders and Kogge-Stone adder.	43
7.6	Comparison of delay of speculative addition in VLCSA 1 and DesignWare adder.	45
7.7	Comparison of area of speculative addition in VLCSA 1 and DesignWare adder.	46
7.8	Comparison of delay of VLCSA 1 and DesignWare adder.	47
7.9	Comparison of area of VLCSA 1 and DesignWare adder.	48
7.10	Comparison of delay of VLCSA 2 and DesignWare adder.	49
7.11	Comparison of area of VLCSA 2 and DesignWare adder.	49

List of Tables

7.1	Experimental and nominal error rates in VLCSA 1 for 2's complement Gaussian inputs. $\mu = 0, \sigma = 2^{32}$	39
7.2	Experimental and nominal error rates in VLCSA 2 for 2's complement Gaussian inputs. $\mu = 0, \sigma = 2^{32}$	39
7.3	Parameters of SCSA and the speculative adder in [17] for an error rate of 0.01% , according to analytical error models and simulation results	40
7.4	Parameters of SCSA and VLCSA 1 for the error rates of 0.01% and 0.25% , according to analytical error models and simulation results.	44
7.5	Parameters of VLCSA 2 for the error rates of 0.01% and 0.25% , according to simulation results. $\mu = 0, \sigma = 2^{32}$	47

Chapter 1

Introduction

Addition, one of the most frequently used arithmetic operations, is employed to build advanced operations such as multiplication and division. Theoretical research has found that the lower bound on the critical path delay of the adder has complexity $O(\log n)$, where n is the adder width. The design of high performance adders has been extensively studied [10] [15], and several adders have achieved logarithmic delays. 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 exercise the critical paths [2, 11, 13]. 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 [3, 6, 12, 17].

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, 11, 13, 17], each output only depends on the previous k bits rather than all previous bits, where k is much smaller than n . However, the cumulative error grows linearly with the adder width 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 [17] can mitigate but not eliminate fanout and area problems. Although the speculative adder in [18] 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 [3, 6, 12, 17]. For the vast majority of input combinations, the speculative adder produces correct results; when error detection flags an error, error recovery provides correct results in one or more extra cycles. Ideally, the average performance of the variable latency adder should be similar to the speculative one. However, existing variable latency adders have several drawbacks. When error detection indicates no error, the actual delay is the longer of the speculative adder and error detection. The delay of error detection is always longer than the speculative adder [6] [17]. Hence, the benefit of speculation is limited by the delay of error detection [3] [12]. Besides, the circuitry for error detection and recovery incurs nontrivial area overhead. Finally, variable latency adders are mostly restricted for random inputs [3, 12, 17].

This thesis first describes 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. SCSA is less susceptible to errors, since it is only applied for blocks instead of individual outputs. A single individual adder is required to compute all outputs of a block instead of each output, which mitigates the area overhead problem. An analytical model to determine the error rate of SCSA is formulated, and the accurate relation between the block size and output error is developed. A high performance speculative adder design is presented for low error rates (e.g. 0.01% and 0.25%).

Secondly, this thesis describes 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 in worst cases. The performance of the variable latency adder is close to that of the speculative adder. This approach has two advantages. First, the critical path delay of the error detection block is lower or comparable to that of the speculative adder. Second, the error detection and recovery circuitry incurs low area overhead by using intermediate results from the speculative adder.

Finally, the previous variable latency and speculative adders are mainly designed for unsigned random inputs, so this thesis proposes the modified variable latency and speculative adders suitable for both random and Gaussian inputs. With modified speculative adder and error detection block, the variable latency adder still achieves high performance when 2's complement Gaussian inputs present. This shows that the variable latency adder design is feasible for practical applications.

Simulations using 10 million unsigned random inputs are used to validate the analytical error model, and analytical and simulation results match well. Simulation results indicate 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. 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 (2's complement Gaussian) inputs.

This thesis is organized as follows. Chapter 2 presents the background of the speculative adder and reliable variable latency adder. Chapter 3 introduces the SCSA and the corresponding error analysis. Chapter 4 describes the SCSA-based speculative adder design. Chapter 5 proposes the reliable variable latency adder design using the SCSA-based speculative adder with error detection and recovery, called variable latency carry select adder (VLCSA). Chapter 6 presents a modified reliable variable latency adder design suitable for both unsigned uniform and 2's complement Gaussian inputs. Chapter 7 validates above models and designs. Section 8 is a conclusion.

Chapter 2

Background

Due to the importance of addition, various adders have been proposed for achieving high performance and low power [10] [15], such as ripple carry adder, carry select adder, carry skip 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*. The actual paths in typical cases are much shorter than the critical one. This observation indicates that the traditional worst-case design methodology may require large design margin. Speculative adders have achieved significantly higher performance by neglecting rare input patterns that exercise the critical paths [2, 11, 13]. Furthermore, error-free variable latency adders can be constructed from speculative adders by adding error detection and recovery, and can achieve average performance comparable to speculative ones [3, 6, 12, 17].

Speculative or variable latency adders fall into two main categories. The first category detects the input patterns that violate the timing constraint and remove these errors at design time. Telescopic units [4] 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 [16], which limits the application of this technique to large circuits. The second category is called function speculation, wherein the original logic function is replaced by an approximate logic function. In the asynchronous domain, [14] first proposed a speculative variable latency adder. In the synchronous do-

main, 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, 11, 13]. However, the techniques in [2, 11, 13] have no error correction capability and may also suffer from large area and large fanout at the primary inputs. Recently, [17] proposed an error-free variable latency adder design wherein the speculative addition is similar to [2, 11, 13]. [6] studied an extension of [17] for the inputs extracted from practical benchmarks, which incurs additional area overhead. Both [17] and [6] have the same area and fanout problems noted above. Furthermore, in [17] [6], the critical path delay of error detection is always longer than that of the speculative adder. The approach in [17] was generalized in [3], wherein an automatic synthesis technique that transforms a combinational design to a two-stage variable-latency design was described. This was extended in [12] to multi-stage function speculation. The design of speculation is strictly limited by error detection in [3] [12]. Besides, [3] [12] are both restricted for random inputs. Finally, although the speculative adder design proposed in [18] can mitigate the area problem, it exhibits a fairly high error rate that limits its application.

Besides, other speculative designs or variable latency designs for low energy operation have been reported. The Razor technique [7] dynamically adjusts the supply voltage by detecting and correcting errors. Similar energy saving technique [8] has been proposed for signal processing applications. Signal processing applications can be error-tolerable, and do not require error detection and recovery. Besides, a non-uniform voltage scaling approach, called probabilistic arithmetic [5], was proposed to adjust voltage for each bit position in adder. This technique can achieve more energy saving due to the special treatment for each bit position.

In this thesis, we describe a novel function speculative addition, and the corresponding speculative adder and reliable variable latency adder. The closest approaches to our work are [18] [12]. Although the speculative adder design proposed in [18] can mitigate the area problem, it exhibits a fairly high error rate that limits its application. The error rate is estimated by running the simulation for a 32-bit adder, which is not scalable for large adders.

The relation between speculation and error also remains unclear. In contrast, we present an analytical error model for SCSA, by which the accurate relation between speculation and error is formulated. In [12], a combinational adder is transformed into a multi-cycle variable-latency one, which requires multi-cycle timing analysis. The design is assumed to work for unsigned random inputs. Besides, it is difficult to incorporate this technique with 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 2's complement Gaussian inputs, and is a simple deterministic design with $1/2$ cycles of latency for addition.

Chapter 3

Speculative carry select addition (SCSA)

Speculative carry select addition (SCSA) comes from the observation about the carry chain in addition. During the discussion in this chapter, we employ unsigned binary addition to illustrate SCSA. The input are assumed to be uniformly distributed, called random inputs. The addition is shown as the dot graph in Figure 3.1, where dots indicate input bits.



Figure 3.1: Dot graph for addition. A dot represents an input bit.

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 and generate (P/G) signals at the i^{th} bit position:

$$P_i = a_i \oplus b_i, \quad (3.1)$$

$$G_i = a_i b_i. \quad (3.2)$$

The sum bit, s_i , and carry-out (carry) bit, c_i , at the i^{th} bit position are rewritten as:

$$s_i = P_i \oplus c_{i-1}, \quad (3.3)$$

$$c_i = G_i + P_i c_{i-1}. \quad (3.4)$$

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 situation is defined as c_i depends on c_{i-1} , written as $c_i \rightarrow c_{i-1}$. All other situations are defined as c_i does not depend on c_{i-1} , written 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*. The probability $P_i = 1$ is $1/2$, so the probability of a k -bit carry chain is $1/2^k$. This implies that c_i may be locally approximated using several consecutive input bits. The average longest carry chain length in an n -bit addition has been extensively studied [10] [15]. Although there is no closed-form solution, it is widely recognized that the carry chain length in the n -bit addition is $O(\log n)$ for unsigned uniform inputs [10] [15]. This interesting fact suggests that it is possible to quickly and accurately estimate the output bit using only several consecutive input bits.

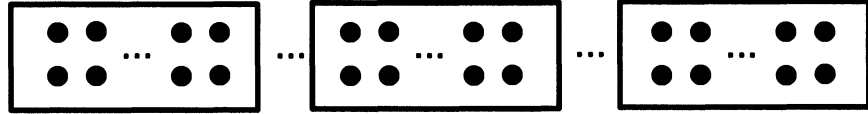


Figure 3.2: Input bits grouped into windows.

3.1 Operation of SCSA

Long carry chains rarely happen in addition for unsigned random inputs. In another word, by grouping input bits into blocks as shown in Figure 3.2, the carry chain length can be

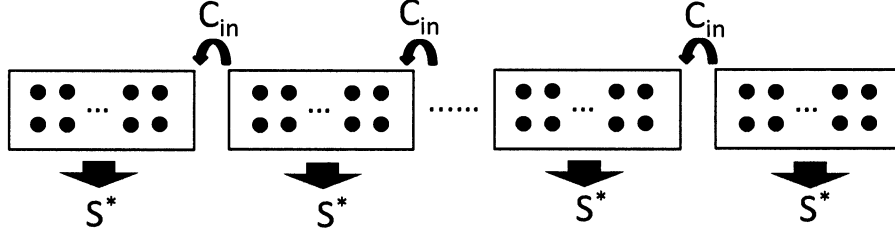


Figure 3.3: Dot graph to illustrate the operation of SWA.

made comparable to the block size with high probability. In SCSA, input bits are divided into blocks of the same size, as shown in Figure 3.2. A block, called *window*, includes several consecutive input bits. The SCSA operation is shown in Figure 3.3: The adder width is n . The window size is k . The total number of windows is $m = \lceil \frac{n}{k} \rceil$. The carry-out bit of the i^{th} window is called C_{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 error remains unclear. An analytical error model for SCSA is presented and provides critical guidance for the SCSA-based adder design.

3.2 Error rate analysis

We start from when an error occurs in SCSA. We observe that 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 3.4.

We state this event in a rigorous way. The adder width is n . The window size is k . The total number of windows is $m = \lceil \frac{n}{k} \rceil$. The group P/G signals at the j^{th} bit position of the

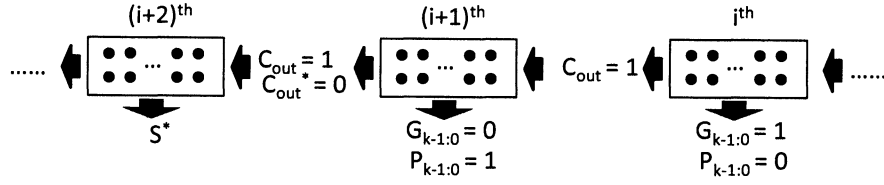


Figure 3.4: Error when $G_{k-1:0} = 1$ in the i^{th} window and $P_{k-1:0} = 1$ in the $(i+1)^{th}$ window.

i^{th} window are stated as:

$$G_{j:0}^i = G_j^i + P_j^i G_{j-1}^i + \dots + G_0^i \prod_{l=1}^j P_l^i, \quad (3.5)$$

$$P_{j:0}^i = \prod_{l=0}^j P_l^i. \quad (3.6)$$

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_{out}^i , is written as:

$$C_{out}^i = G_{k-1:0}^i + P_{k-1:0}^i C_{out}^{i-1}, \quad 1 \leq i < m \quad (3.7)$$

In SCSA, C_{out}^{i-1} is truncated to 0, and C_{out}^i is approximated as C_{out}^{i*} :

$$C_{out}^{i*} = G_{k-1:0}^i, \quad 1 \leq i < m \quad (3.8)$$

As shown in Figure 3.4, the i^{th} window has a group generate signal with value 1, $G_{k-1:0}^i = 1$. It is straightforward to see that:

$$C_{out}^i = 1$$

$P_{k-1:0}^{i+1} = 1$ indicates that C_{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_{out}^{i+1} is approximated using (3.8) as:

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

In contrast, in traditional addition, the correct carry-out bit of the $(i + 1)^{th}$ window C_{out}^{i+1} is written as:

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

Thus, $C_{out}^{i+1} \neq C_{out}^{i+1*}$. SCSA 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, $P(P_{k-1:0}^{i+1} G_{k-1:0}^i = 1)$, is written as:

$$P(P_{k-1:0}^{i+1} G_{k-1:0}^i = 1) = P(P_{k-1:0}^{i+1} = 1)P(G_{k-1:0}^i = 1) \quad (3.9)$$

Besides, the probabilities that group P/G signals of the window equal 1 can be derived as:

$$\begin{aligned} P(P_{k-1:0}^{i+1} = 1) &= P\left(\prod_{j=0}^{k-1} P_j^{i+1} = 1\right) \\ &= \prod_{j=0}^{k-1} P(P_j^{i+1} = 1) = (1/2)^k, \end{aligned} \quad (3.10)$$

$$\begin{aligned} P(G_{k-1:0}^i = 1) &= P(G_{k-1}^i + P_{k-1}^i G_{k-2}^i + \dots + G_0^i \prod_{j=1}^{k-1} P_j^i = 1) \\ &= P(G_{k-1}^i = 1) + P(P_{k-1}^i G_{k-2}^i = 1) + \dots + P(G_0^i \prod_{j=1}^{k-1} P_j^i = 1) \\ &= (1/2)[1 - (1/2)^k]. \end{aligned} \quad (3.11)$$

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 (3.10) and (3.11), (3.9) is computed as:

$$\begin{aligned} P(P_{k-1:0}^{i+1} G_{k-1:0}^i = 1) &= P(P_{k-1:0}^{i+1} = 1)P(G_{k-1:0}^i = 1) \\ &= (1/2)^{k+1}[1 - (1/2)^k] \end{aligned} \quad (3.12)$$

The total error probability for SCSA is approximated by summing up probabilities of

these events for all windows. The approximate total error probability is stated as:

$$\begin{aligned}
 P_{err}^* &= \sum_{0 \leq i < \lceil \frac{n}{k} \rceil - 1} P(P_{k-1:0}^{i+1} G_{k-1:0}^i = 1) \\
 &= \sum_{0 \leq i < \lceil \frac{n}{k} \rceil - 1} (1/2)^{k+1} [1 - (1/2)^k]
 \end{aligned} \tag{3.13}$$

(3.13) describes the relation between the window size and error rate for unsigned random inputs.

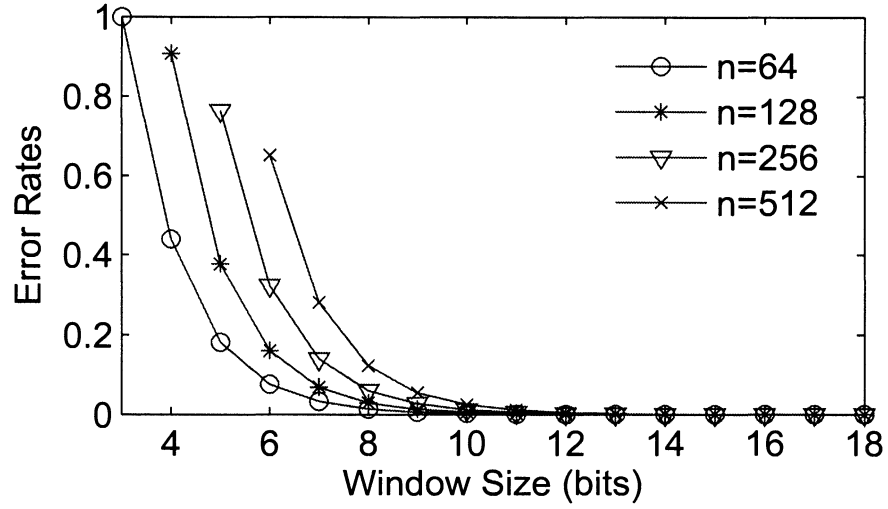


Figure 3.5: Predicted error rates for different adder widths (n) and window sizes.

Based on (3.13), predicted error rates for different adder widths and window sizes are plotted in Figure 3.5. Figure 3.5 suggests that the error rate rapidly decreases as the window size increases. The error rate becomes negligible if the window size is large enough. For example, if $n = 256$, $k = 16$, $P_{err}^* \simeq 0.01\%$. In other words, a 256-bit adder is replaced with 16 16-bit adders for an error rate of 0.01%. The predicted error rate is critical for guiding the SCSA-based speculative adder design.

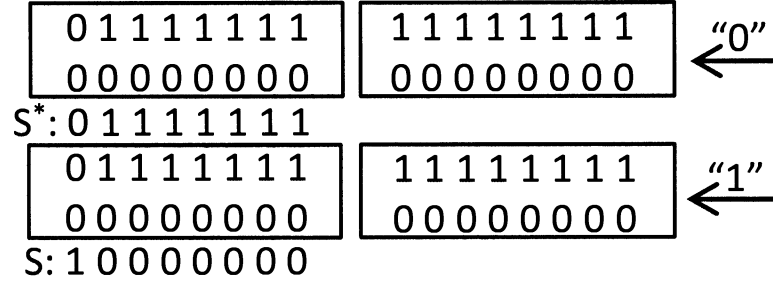


Figure 3.6: Example to illustrate low error magnitude.

3.3 Error magnitude analysis

Error magnitude is defined as the ratio of the error to the correct result. For example, the correct result is 11001, the speculative result is 10001. The error is $11001 - 10001 = 01000$. So the error magnitude is $01000/11001 = 0.32$. It is preferable to have small error magnitude when the speculation is incorrect. For some error-tolerable applications, the speculative result with small error magnitude may still be acceptable.

In SCSA, the error magnitude is low when an error occurs. An example is shown in Figure 3.6. The carry-in bit of the right window is truncated as 0. Then 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 $1/2^7$, which is quite small. 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 such as the speculative addition in [17]. For example, the correct result is 11111111, and the speculated result is 01111111. The error magnitude is $2^7/(2^8 - 1) = 50.2\%$, which is quite large.

Chapter 4

SCSA-based speculative adder

A high performance, low area overhead speculative adder can be built upon SCSA, called SCSA-based speculative adder. We call this adder design as SCSA for simplicity. This design may be used for those applications where errors are tolerable, such as data mining, machine learning, cryptography and signal processing. In these scenarios, the incorrect result generated by the speculative addition may still result in the correct final result.

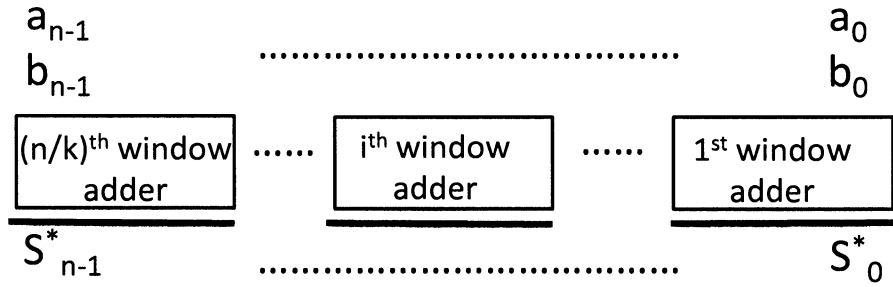


Figure 4.1: Speculative adder is consisted of $\lceil \frac{n}{k} \rceil$ k -bit small adders.

Ideally, the n -bit adder is segmented into several small identical *window adders*. However, since $n \% k = 0$ doesn't always hold, at least one of the window adders is smaller than others. Specifically, this window adder has $n - k(\lceil \frac{n}{k} \rceil - 1)$ bits. Similar to the optimization of the carry select adder design, this adder is placed as the 1st window adder for reducing

the delay of the speculative adder. Other window adders all have k bits. In summary, the n -bit adder is almost equally segmented into $\lceil \frac{n}{k} \rceil$ k -bit window adders, as shown in the Figure 4.1.

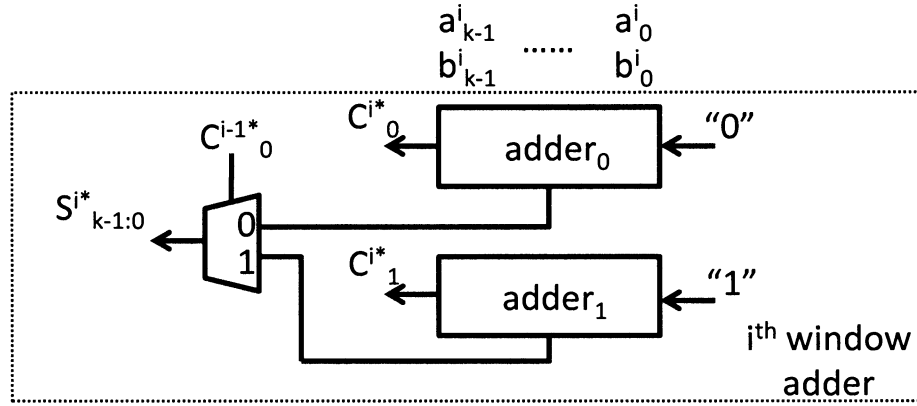


Figure 4.2: Window adder implementation in speculative adder. Carry-select adder structure is employed to increase performance.

4.1 Window adder design

Next we discuss the design of the window adder. The speculative result of the window adder is computed using the input bits of the window adder and the carry-in bit from the previous window adder. Since the window adder works as the same as the traditional adder, the window adder can be implemented using any traditional adder. Assume all inputs arrive at the same time. For the window adder, input bits arrive earlier than the carry-in bit provided by the previous window adder. Thus we employ the carry-select adder structure to increase performance. As shown in Figure 4.2, the window adder is consisted of two small adders, $adder_0$ and $adder_1$. Two small adders can be implemented using any traditional adder, such as Kogge-Stone or Brent-Kung adder. Kogge-Stone adder is considered as the possible fastest adder design in traditional adders [10]. We can employ Kogge-Stone to

further increase performance of the speculative adder.

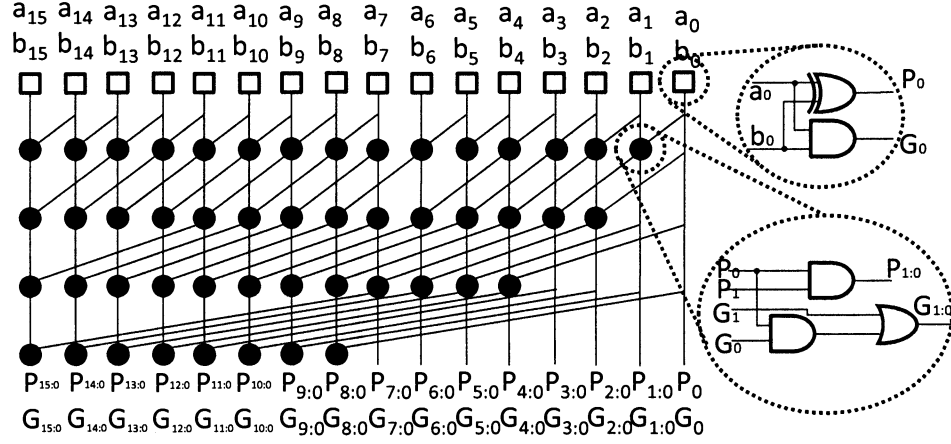


Figure 4.3: Tree structure for calculating the group P/G signals of Kogge-Stone adder.

For the i th bit position of the adder, we have:

$$c_i = G_{i-1:0} + P_{i-1:0}c_0, \quad (4.1)$$

$$s_i = c_i \oplus P_i. \quad (4.2)$$

where c_i is the carry bit at the i th bit position, $G_{i-1:0}$ and $P_{i-1:0}$ are the group P/G signals at the i th bit position, c_0 is the carry bit at the 0th bit position, s_i is the sum bit at the i th bit position, P_i is the propagate signal at the i th bit position. For example, the tree structure for calculating the group P/G signals of Kogge-Stone adder is shown in Figure 4.3. After calculating the group P/G signals, the sum bit is computed.

4.2 Implementation of SCSA

Then we how to implement SCSA. The group propagate and generate (P/G) signals of the i^{th} window adder are computed. The speculative carry-in bit of the i^{th} window adder is the group G signal of the $(i-1)^{th}$ window adder, $G_{k-1:0}^{i-1}$. Then the j^{th} sum bit of the i^{th}

window, s_j^{i*} , is estimated as:

$$s_j^{i*} = P_j^i \oplus [G_{j-1:0}^i + P_{j-1:0}^i C_{\text{out}}^{i-1*}] \quad (4.3)$$

$$= P_j^i \oplus [G_{j-1:0}^i + P_{j-1:0}^i G_{k-1:0}^{i-1}], 0 \leq j < k \quad (4.4)$$

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

$$s_{j,1}^{i*} = P_j^i \oplus [G_{j-1:0}^i + P_{j-1:0}^i], \quad (4.5)$$

$$s_{j,0}^{i*} = P_j^i \oplus G_{j-1:0}^i, 0 \leq j < k. \quad (4.6)$$

One of the results above is selected and output as the speculative sum bit when C_{out}^{i-1*} is ready, noted as C_0^{i-1*} in Figure 4.2.

4.3 Time and space complexity of SCSA

We first discuss the time complexity of SCSA. Assume we implement the small adder in the window adder using Kogge-Stone. The critical path of SCSA is consisted of a traditional adder and a multiplexer. Thus, the critical path of SCSA is equivalent to that of a k -bit Kogge-Stone adder plus a multiplexer. The complexity of the critical path delay of SCSA is $O(\log k)$. In contrast, the critical path delay of a n -bit traditional adder has at least complexity $O(\log n)$. Hence, the speculative adder can be significantly faster than the traditional adder.

Then we estimate the space complexity of SCSA. At each step in SCSA, there are at most k times computations for intermediate group P/G signals. The total number of window adders is $\lceil \frac{n}{k} \rceil$. Thus, the space complexity of a n -bit speculative adder is $O(\lceil \frac{n}{k} \rceil k \log k)$. In contrast, traditional parallel prefix adders generally have larger area than the speculative adder. For example, the space complexity of n -bit Kogge-Stone adder is $O(n \log n)$.

Finally, it is worth comparing SCSA with existing speculative adders. The adder in [17], one of the best state-of-the-art speculative adders, has the delay complexity $O(\log l)$ and

space complexity $O(n \log l)$. l is the speculative carry chain length, which is similar to the window size in SCSA. The window size k of SCSA is smaller than the speculative carry length l of the design in [17] for achieving similar error rates, which will be shown in Table 7.4. Our experiments will also validate that SCSA has smaller area than the speculative adder in [17] with similar performance and error rate. In summary, SCSA can be faster and smaller than the counterpart in [17].

Chapter 5

Variable latency carry selection adder (VLCSA)

For applications that error can be tolerated, the speculative adder can increase performance by introducing a certain level of inaccuracy. However, many applications are not error-tolerable. The incorrect result generated by speculation will result in incorrect final result. For applications where errors cannot be tolerated, a reliable variable latency adder can be built upon the SCSA-based speculative adder by adding error detection and recovery, called variable latency carry selection adder (VLCSA). Error detection flags if speculation is correct. Error recovery produces the correct result when error detection flags an error. VLCSA works with one or two cycles of latency for addition, which is similar to [17]. Ideally, the average performance of VLCSA is close to the speculative one when error rate is low. In this chapter, unsigned random inputs are still assumed.

5.1 Error detection design

We first develop error detection in VLCSA. Error detection flags if speculation is incorrect. In another word, error detection flags when the speculative carry-in bit of the window adder is incorrect. The error detection block in VLCSA is designed to overestimate the error rate.

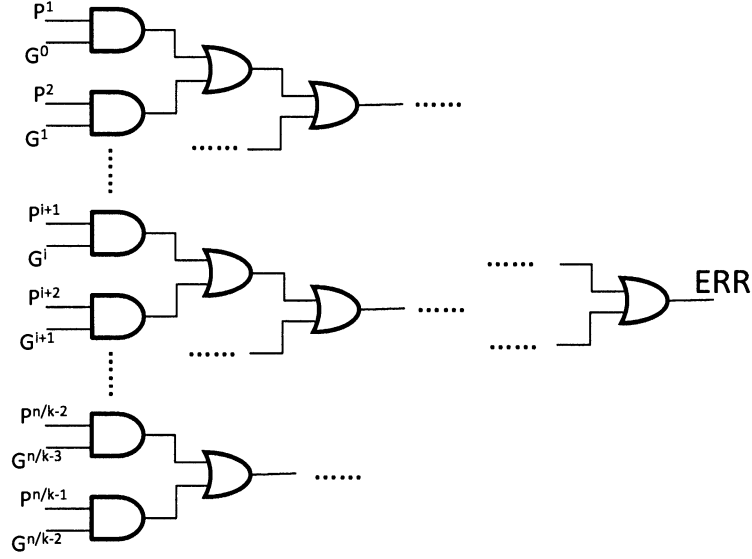


Figure 5.1: Error detection implementation using 2-input AND and OR gates.

There are two reasons: (1) Since the exact carry-in bit of the window is a global signal, it incurs nontrivial latency to wait and detect this carry-in bit. Conceptually, the advantage of speculation is lost if error detection requires the exact carry-in bit of the window adder. (2) It is crucial that all errors are detected in VLCSA. Although some correct results are flagged as incorrect ones, overestimation of error is helpful for detecting all errors. Besides, the false negative rate of error detection can be controlled under a low level.

In Chapter 3, we discuss the analytical error model for SCSA. This error model accurately describes the event that the speculative carry-in bit of a window adder is incorrect. Thus, we employ this model to implement error detection. The error detection signal, ERR, is stated as:

$$ERR = \sum_{0 \leq i < \lceil \frac{n}{k} \rceil - 1} P_{k-1:0}^{i+1} G_{k-1:0}^i \quad (5.1)$$

where $P_{k-1:0}^{i+1}$ is the group propagate signal of the $(i+1)$ window adder, $G_{k-1:0}^i$ is the group

generate signal of the i window adder. ERR flags an error if $\exists i, 0 \leq i < \lceil \frac{n}{k} \rceil - 1, P_{k-1:0}^{i+1} = 1, G_{k-1:0}^i = 1$. In another word, ERR flags an error if $G_{k-1:0}^i$ with value 1 affects the carry-in bit of the next next window adder.

The group propagate and generate signals of the window adder are computed during the speculative addition. So it greatly simplifies the error detection design by using these intermediate values from the speculative adder. Assume group propagate and generate signals are obtained from the speculative adder, we focus on the combination of these group propagate and generate signals. The error detection block is implemented using 2-input AND and OR gates, as shown in Figure 5.1.

Next we discuss the time complexity of the error detection block. It takes $\log k$ steps to generate the group P/G signals of the window adder. As shown in Figure 5.1, it takes additional $\log \lceil \frac{n}{k} \rceil$ steps to produce ERR. Therefore, the critical path delay of error detection has complexity $O(\log \lceil \frac{n}{k} \rceil + \log k)$. In particular, we observe that $\log \lceil \frac{n}{k} \rceil$ is quite small in practice. For example, when $n = 512$ and $k = 17$, $\log \lceil \frac{n}{k} \rceil \simeq 5$. On the other hand, it takes several constant steps to compute sum bits after generating group P/G signals in the speculative adder. In VLCSPA, error detection has comparable or even shorter critical path delays than the speculative addition. The actual critical path delay is the longer one of the speculative addition and error detection when error detection flags no error, so the benefit of speculation is maintained.

Finally, the space complexity of the error detection block is estimated as below. At each step, there are at most $\lceil \frac{n}{k} \rceil$ computations. Besides, there are $\log \lceil \frac{n}{k} \rceil$ steps in the error detection block. Thus, the space complexity of the error detection block is $O(\lceil \frac{n}{k} \rceil \log \lceil \frac{n}{k} \rceil)$. This is similar to that of SCSA, but the computation only uses simple gates and requires low area overhead.

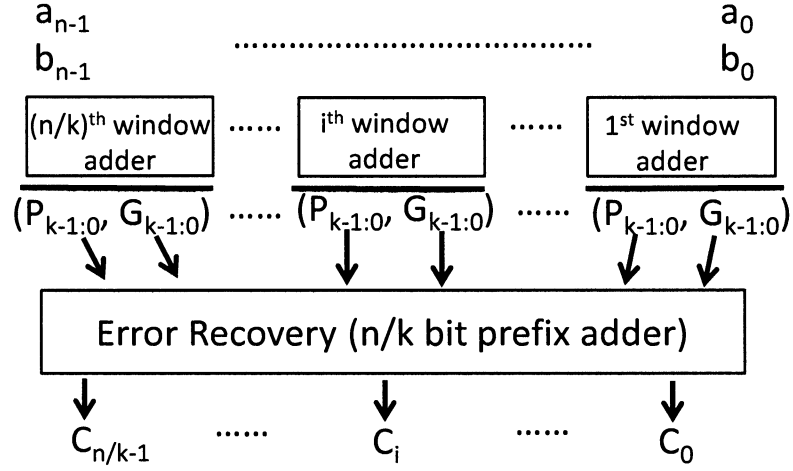


Figure 5.2: Area-efficient implementation for error recovery using intermediate results from the speculative adder.

5.2 Error recovery design

Error recovery produces the correct result when error detection flags an error. The simplest solution is to employ a traditional adder to compute the result when the speculation is incorrect. But this incurs long delay penalty and large area overhead. The area overhead can be larger than that of SCSA. Alternatively, we employ the intermediate results from SCSA to simplify the error recovery design.

Figure 5.2 shows an area-efficient implementation for error recovery using intermediate results from the speculative adder. The upper part of Figure 5.2 is SCSA. The lower part of Figure 5.2 is a $\lceil \frac{n}{k} \rceil$ -bit prefix adder. This prefix adder takes the group propagate and generate (P/G) signals of window adders as the input and computes the correct carry-out bits for all window adders. SCSA has computed group P/G signals at each bit position of the window adder. Thus, correct sum bits of all window adders are computed using the

outputs of this prefix adder.

Then we discuss the time complexity of the error recovery block. For this $\lceil \frac{n}{k} \rceil$ -bit prefix adder, there are $\log \lceil \frac{n}{k} \rceil$ steps in computation. Thus the time complexity of this prefix adder is $O(\log \lceil \frac{n}{k} \rceil)$. The critical path delay of the error recovery block is through the speculative adder and the prefix adder. There are $\log k$ steps for computing the intermediate results in SCSA. Thus, the time complexity of the critical path delay of the error recovery block, through the speculative adder and the prefix adder, is $O(\log k + \log \lceil \frac{n}{k} \rceil)$. Since the error recovery block introduces an extra prefix adder, it incurs nontrivial delay penalty. If the clock cycle is chosen to be slightly longer than the delays of SCSA and the error detection block, we observe that the delay of error recovery can be shorter than two cycles.

Finally, we discuss the space complexity of the error recovery block. There are most $\lceil \frac{n}{k} \rceil$ computations in each steps in the $\lceil \frac{n}{k} \rceil$ -bit prefix adder. The total number of steps is $\log \lceil \frac{n}{k} \rceil$. Thus, the space complexity of this prefix adder is $O(\lceil \frac{n}{k} \rceil \log \lceil \frac{n}{k} \rceil)$. The computation in this prefix adder may use complex gates and requires nontrivial area overhead. The major area overhead of VLCSA comes from the error recovery block. We can employ synthesis tools to reduce the area overhead.

5.3 Operation of VLCSA

The implementation of VLCSA is shown in Figure 5.3, which is similar to [17]. When inputs A and B are ready, SCSA computes the speculative result, S^* . If error detection flags no error, $ERR = 0$, the output signal VALID flags that the speculative result is correct. Then the speculative result S^* is selected and output as the final result of VLCSA. If error detection flags an error, $ERR = 1$, the output signal STALL indicates that the speculative result is incorrect. VLCSA stalls an extra cycle and waits for the correct result generated by the error recovery block. The error recovery block uses the intermediate results from SCSA and re-computes the correct result. Then the result from error recovery, S^{REC} , is selected and output as the final result of VLCSA.

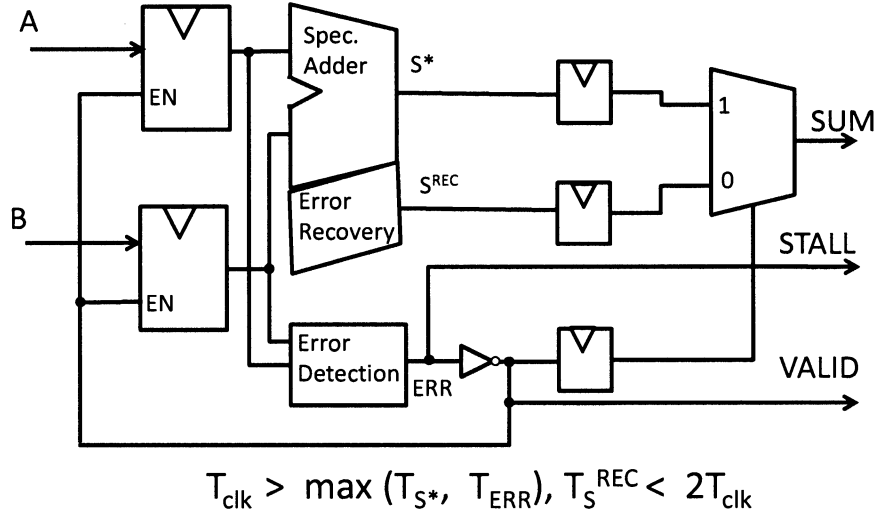


Figure 5.3: Variable latency adder implementation, similar to [17]

Next we discuss the timing issues in the design of VLCSA. The delay of the error detection block is designed to be similar to that of SCSA. Thus, we will know whether the speculative result is correct or not when the speculative result is ready. Then speculative result can be selected and output without additional delays. The clock cycle, T_{clk} , is slightly longer than the delays of SCSA and the error detection block. The speculative result and error detection signal, ERR , are computed in a single cycle. The error recovery block produces the correct result in two cycles. If ERR flags no error, the speculative result is correct. Otherwise, error recovery produces the correct result in an additional cycle. The effective cycle of VLCSA, T_{ave} , is stated as:

$$T_{ave} = T_{clk}(1 - P_{err}) + 2T_{clk}P_{err} \quad (5.2)$$

where P_{err} is the error rate of SCSA. If P_{err} is quite small such as 0.01%, $T_{ave} \simeq T_{clk}$, the average latency of the variable adder is slightly longer than the delays of the speculative adder and the error detection block. This indicates that the average performance of VLCSA is close to that of SCSA.

Chapter 6

Modified VLCSA (VLCSA 2)

6.1 Motivation

In previous chapters, speculative and variable latency adders are assumed to work mainly for unsigned random inputs. SCSA and VLCSA work well for unsigned random inputs. From the discussion, we obtain valuable insight for the design of speculative and variable latency adders. This is very crucial for guiding speculative and variable latency adder designs. Furthermore, it is crucial to evaluate the performance of the variable latency adder for other inputs. Different inputs for the adder will largely affect the performance of speculative and variable latency adders.

The key idea of speculative addition is to utilize the carry chain distribution for unsigned random inputs. An example of statistics of carry chain lengths for unsigned random inputs is shown in Figure 6.1. The adder size is 32 bits. We run 10^6 simulations for unsigned random inputs to gather records. We observe that the portion of the carry chains rapidly decreases as the carry chain length increases. There is a gap between unsigned random inputs and practical inputs: the distribution of carry chain lengths in practical applications may be quite different from that of the unsigned random inputs. It is meaningful to evaluate the carry chain distribution of the practical inputs.

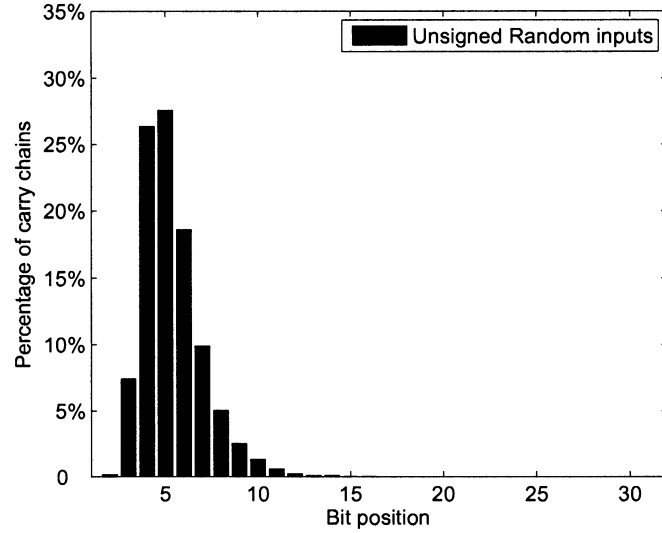


Figure 6.1: Example of statistics of carry chain lengths for unsigned random inputs. The adder size is 32 bits.

6.2 Profiling practical inputs

We start from the difference between unsigned random inputs and practical inputs. There are two important observations for practical inputs [6] [9]: (1) The 2's complement representation is widely used in practice. It's convenient to implement subtraction with addition using the 2's complement representation. (2) Small numbers appear more frequently than large ones. The computations between two small numbers has a high frequency. Thus, the practical inputs can be quite different from unsigned random inputs.

In [6] [9], the distributions of the carry chain lengths are extracted from practical workloads. There are nontrivial portions of carry chains with very long lengths. One example from [6] is employed here. They calculated the statistics of carry chain lengths in addition from a cryptographic workload, including RSA encryption/decryption (RSA), Elliptic curve ElGamal encryption/decryption over prime fields (ECELGP), Diffie-Hellman key exchange (DH), and Elliptic curve digital signature algorithm over prime fields (ECDSP).

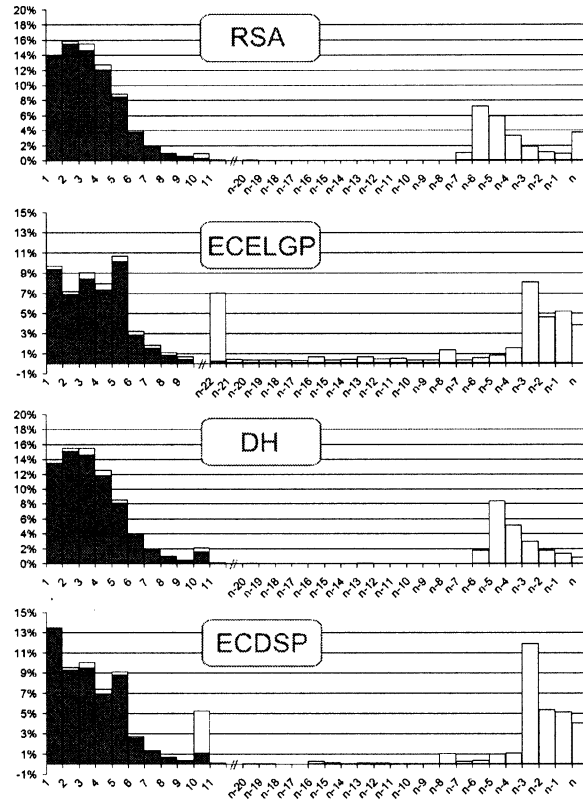


Figure 6.2: Examples of statistics of carry chain lengths from a cryptographic workload [6].

The statistics of carry chain lengths for these benchmarks are shown in Figure 6.2. We can find that very long carry chains frequently occur in these benchmarks, and carry chains mainly concentrate in two ranges. This is quite different from that of unsigned random inputs. The speculative addition designed for unsigned random inputs is unable to handle those very long carry chains, and thus error rate significantly increases. This will largely hurt the performance of speculative and variable latency addition.

6.3 Approximating practical inputs

Practical inputs can be quite different from each other, so it is impractical to capture the basics of all practical inputs. We target to employ the mathematical distribution to approximately profile the practical inputs similar to those in [6]. In particular, we observe that Gaussian inputs seems able to reflect the basics of the practical workloads such as [6]. There are two reasons: (1) Small numbers occur more frequently than large numbers. (2) It is straightforward to introduce 2's complement representation for Gaussian inputs. We will examine this observation using several examples. The adder size is 32 bits. We run 10^6 simulations for different inputs to gather records.

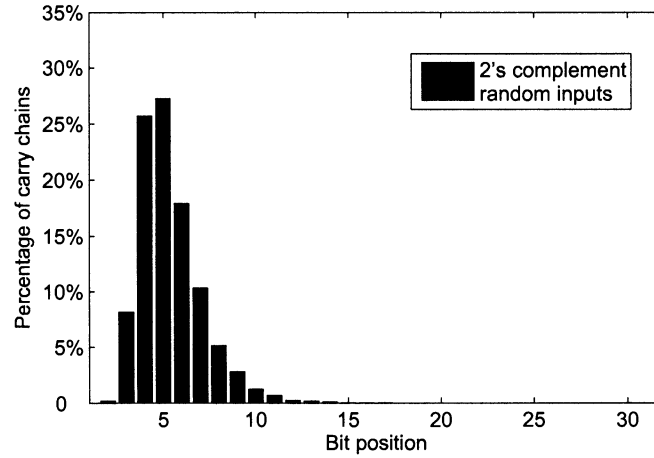


Figure 6.3: Example of statistics of carry chain lengths for 2's complement random inputs. The adder size is 32 bits.

We first compare the difference between unsigned and 2's complement random inputs by generating the statistics of carry chain lengths. An example of statistics of carry chain lengths for 2's complement random inputs is shown in Figure 6.3. we observe that carry chains still concentrate in the range of short carry chains. As the carry chain length increases, the portion of carry chains rapidly decreases, which is similar to that of unsigned random inputs.

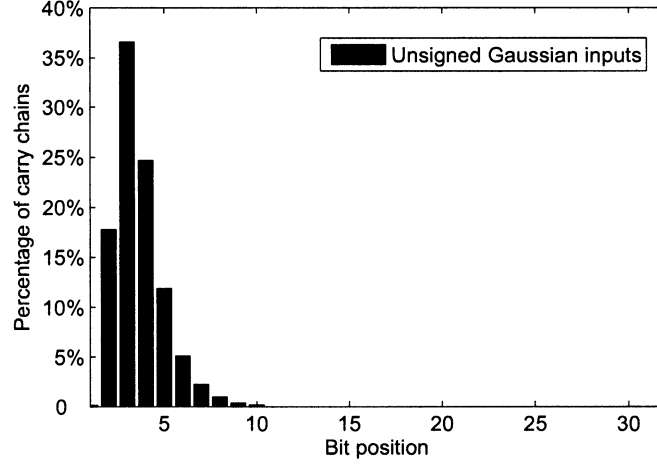


Figure 6.4: Example of statistics of carry chain lengths for unsigned Gaussian inputs. The adder size is 32 bits.

Then we produce the statistics of carry chain lengths for unsigned Gaussian inputs. An example of statistics of carry chain lengths for unsigned Gaussian inputs is shown in Figure 6.4. We see that carry chains still concentrate in the range of short carry chains. As the carry chain length increases, the portion of carry chains rapidly decreases, which is still similar to that of unsigned random inputs.

Next we profile the statistics of carry chain lengths for 2's complement Gaussian inputs. This case combines both 2's complement representation and Gaussian inputs. An example of statistics of carry chain lengths for 2's complement Gaussian inputs is shown in Figure 6.5. we see that carry chains concentrate in two separate ranges. For 2's complement Gaussian inputs, a nontrivial portion of carry chains is as long as the adder size. Although the frequency of long carry chains in Figure 6.5 seems higher than that in Figure 6.2, the distribution in Figure 6.5 is quite close to that in Figure 6.2. In another word, 2's complement Gaussian inputs captures the basics of the inputs in [6]. We will propose a modified speculative addition and the corresponding variable latency addition for 2's complement Gaussian inputs as below.

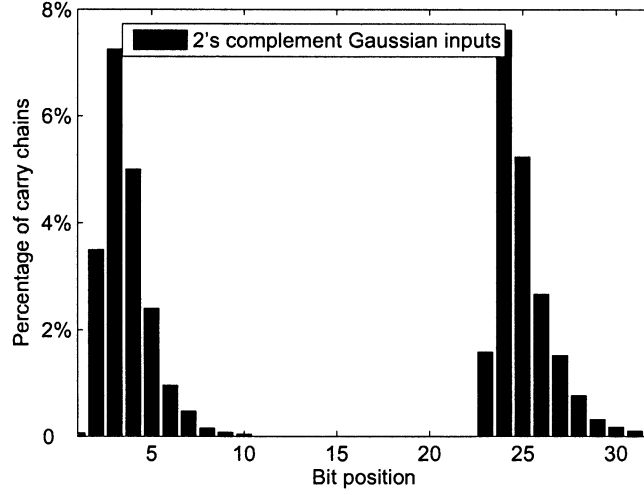


Figure 6.5: Example of statistics of carry chain lengths for 2's complement Gaussian inputs. The adder size is 32 bits.

6.4 Key idea of Modified VLCSA

As shown in Figure 6.2 and Figure 6.5, these long carry chains significantly increases the error rate of speculative addition in VLCSA and makes VLCSA even slower than the traditional adder. Therefore, it is necessary to modify the speculative addition and the variable latency addition for practical inputs. In this thesis, we target to the design for 2's complement Gaussian inputs. The speculative adder design, SCSA, is called SCSA 1, and the variable latency adder design, VLCSA, is called VLCSA 1. The modified speculative adder design is called SCSA 2, and the modified variable latency adder design is called VLCSA 2.

The key idea of VLCSA 2 is to correctly speculate results when long carry chains like those in Figure 6.5 occur. We design VLCSA 2 based on the observation: long carry chains are usually triggered by the addition of a small positive and a small negative number, and affect the most significant bit position. The error detection in VLCSA 1 flags these long carry chains as errors since they are much longer than the window size. If we can detect

and remove these errors, the error rate will decrease to the low level. In another word, we target to correctly speculate results and flags no errors when such long carry chains occur. Gaussian inputs in the 2's complement representation are assumed for the design.

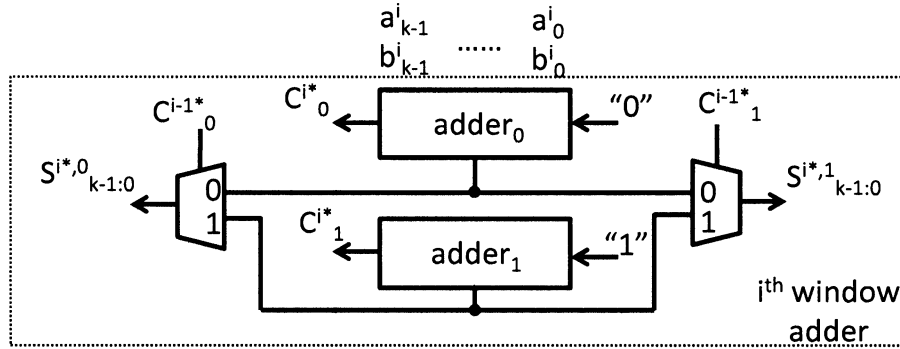


Figure 6.6: Window adder implementation in SCSA 2.

6.5 Modified speculative addition

We first describe the implementation of modified speculative addition, SCSA 2. The modified window adder design is shown in Figure 6.6. Compared with the window adder in Figure 4.2, an additional speculative result, $S^{i*,1}$, is calculated in Figure 6.6. This speculative result is selected by one of speculative carry-out bits of the previous window adder, C_{i-1}^{i-1*} , which is discarded in SCSA 1. By combining two speculative results, the speculative adder can correctly calculate the result. In another word, $S^{i*,1}$ is correct when the very long carry chains occur. However, how to select the speculative result still remains a challenge. We will handle this issue in the design of error detection.

Then we discuss the time complexity of SCSA 2. The complexity of the critical path of SCSA 1 is $O(\log k)$. In SCSA 2, we add another speculative result $S^{i*,1}$. $S^{i*,1}$ has the complexity $O(\log k)$ as the same as that of $S^{i*,0}$. This indicates that no extra delay is added.

Finally, we estimate the space complexity of SCSA 2. Compared with SCSA 1, the major cost of SCSA 1 is to add an extra 2-input multiplexer in the window adder. The total number of window adders is $\lceil \frac{n}{k} \rceil$. Therefore, the area overhead of SCSA 2 has the complexity $O(\lceil \frac{n}{k} \rceil)$. The space complexity of a n -bit SCSA 2 is still $O(\lceil \frac{n}{k} \rceil k \log k)$.

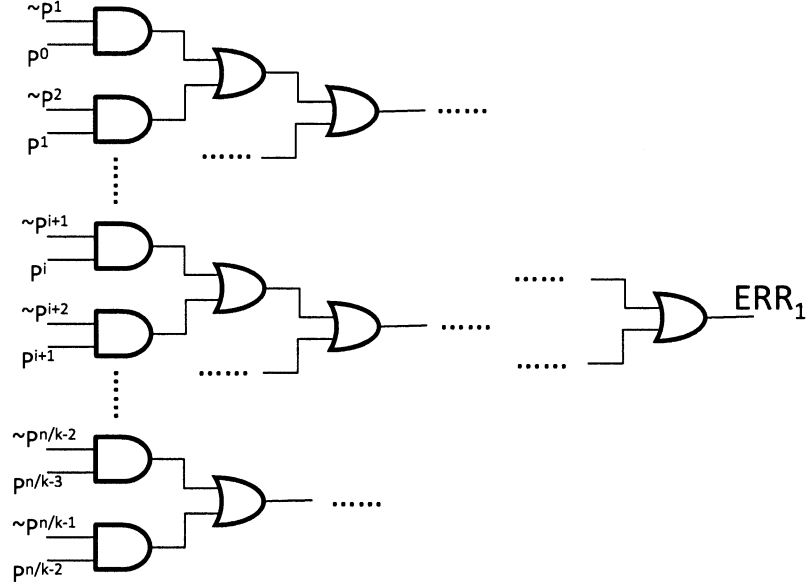


Figure 6.7: Modified error detection implementation using 2-input AND and OR gates.

6.6 Modified error detection

Since we target to detect those long carry chains, new error detection signal is introduced. We define an additional error detection signal, ERR_1 , for detecting long carry chains. The error detection signal for detecting short carry chains in VLCSA 1 is noted as ERR_0 . It is noted that the error should be detected by both ERR_1 and ERR_0 . ERR_1 in VLCSA 2 is

stated as:

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

where $P_{k-1:0}^i$ is the group propagate signal of the i^{th} window, $\overline{P}_{k-1:0}^{i+1}$ is the complement of the group propagate signal of the $(i+1)^{th}$ window.

The key idea of this error detection is inspired by the discussion in [6]. Let us see why long carry chains can be flagged by ERR_1 and errors can be detected by ERR_0 and ERR_1 : (1) $ERR_0 = 0$. This means that $\forall i, 0 \leq i < \lceil \frac{n}{k} \rceil - 1, P_{k-1:0}^{i+1} G_{k-1:0}^i = 0$. Only short carry chains occur in this case. The speculative result is correct and just as the same as the one in VLCSA 1. It is not necessary to check ERR_1 . (2) $ERR_0 = 1, ERR_1 = 0$. $\exists i, 0 \leq i < \lceil \frac{n}{k} \rceil - 1, P_{k-1:0}^{i+1} G_{k-1:0}^i = 1, \forall i, 0 \leq i < \lceil \frac{n}{k} \rceil - 1, \overline{P}_{k-1:0}^{i+1} P_{k-1:0}^i = 0$. ERR_0 flags an error. It indicates that there may be carry chains longer than the window size. The speculative result in VLSCA 1 is incorrect. Besides, $ERR_1 = 0$ implies that a long carry chain generates at a bit position and propagates to the most significant bit (MSB) position. The new speculative result in VLSCA 2 provides the correct result. In another word, if $ERR_0 = 1, ERR_1 = 0$, long carry chains occur, but do not incur actual errors. (3) $ERR_0 = 1, ERR_1 = 1$. $\exists i, 0 \leq i < \lceil \frac{n}{k} \rceil - 1, P_{k-1:0}^{i+1} G_{k-1:0}^i = 1, \exists i, 0 \leq i < \lceil \frac{n}{k} \rceil - 1, \overline{P}_{k-1:0}^{i+1} P_{k-1:0}^i = 1$. ERR_0 flags an error. It indicates that there are carry chains longer than the window size. Besides, $ERR_1 = 1$ implies that such an error occurs since a carry chain starts at a bit position and ends before reaching the MSB position. Both speculative results in VLSCA 2 are incorrect. In this case, error recovery is activated and produces the correct result.

Next we discuss the time complexity of the error detection block. It takes $\log k$ steps to generate the group propagate signals of the window adder. As shown in Figure 6.7, it takes additional $\log \lceil \frac{n}{k} \rceil$ steps to produce ERR_1 . Therefore, the critical path delay of error detection has complexity $O(\log \lceil \frac{n}{k} \rceil + \log k)$. In VLCSA 2, the error detection block has comparable or even shorter critical path delays than that of the speculative addition. The actual critical path delay is the longer one of the speculative addition and the error detection

block when error detection flags no error, so the benefit of speculation is maintained.

Then the space complexity of the error detection block is estimated as below. At each step, there are at most $\lceil \frac{n}{k} \rceil$ computations. Besides, there are $\log \lceil \frac{n}{k} \rceil$ steps in the error detection block. Thus, the space complexity of the error detection block is $O(\lceil \frac{n}{k} \rceil \log \lceil \frac{n}{k} \rceil)$. This is similar to that of ERR_0 .

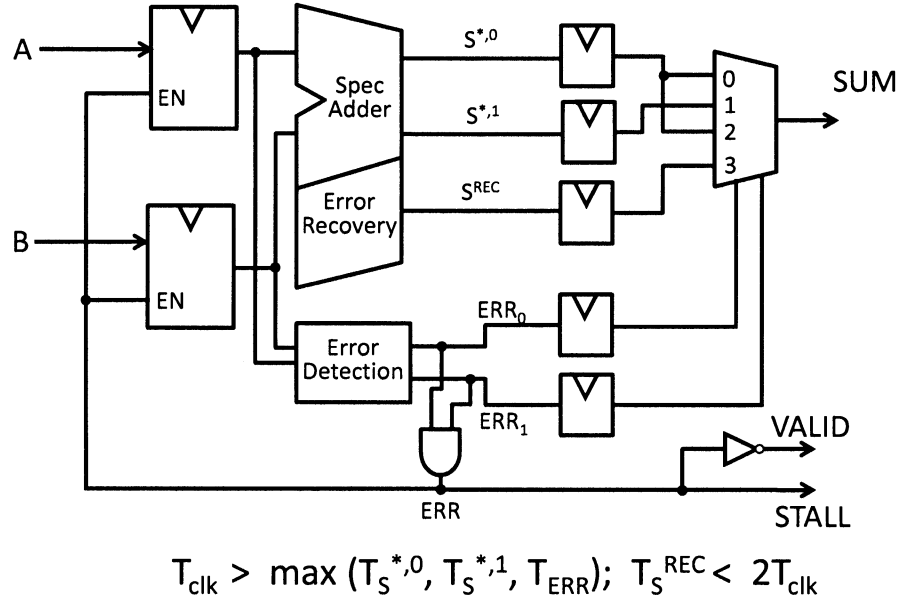


Figure 6.8: VLCSA 2 implementation.

6.7 Operation of VLCSA 2

Based on the discussion, VLCSA 2 is shown in Figure 6.8. There are two speculative results, $S^{*,0}$ and $S^{*,1}$. $S^{*,0}$ ($S^{*,1}$) is the speculative result when speculative carry-in bit is value 0 (1). There are two error detection signals, ERR_0 and ERR_1 . ERR_0 flags if there is any carry chain longer than the window size. ERR_1 flags if there are long carry chains reach the MSB position. If $ERR_0 = 1$ and $ERR_1 = 1$, the speculative addition is incorrect. Then error recovery provides the correct result, which is as same as that in

VLCSA 1.

In general, VLCSA 2 works in the similar way to VLCSA 1. Error detection signals, ERR_0 and ERR_1 , are used to select speculative results and flag errors: (1) $ERR_0 = 0$. The output VALID is 1 and STALL is 0. This indicates the speculative result is correct. The speculative result $S^{*,0}$ is selected and output. (2) $ERR_0 = 1, ERR_1 = 0$. The output VALID is 1 and STALL is 0. This indicates the speculative result is also correct. The speculative result $S^{*,1}$ is selected and output. (3) $ERR_0 = 1, ERR_1 = 1$. The output VALID is 0 and STALL is 1. This implies that the speculative addition is incorrect. The variable latency adder stalls for an additional cycle, and error recovery provides the correct result, S^{rec} .

Next we discuss the timing issue in the design of VLCSA 2. The delay of the error detection block is designed to be similar to that of the speculative adder. Thus, we can know if the speculative result is correct or not when the speculative result is ready. Then speculative result can be output without additional delays. The clock cycle, T_{clk} , is slightly longer than the delays of the speculative adder and the error detection block, $T_{clk} > \max(T_S^{*,0}, T_S^{*,1}, T_{ERR})$. The speculative results and the error detection signal, ERR, are computed in a single cycle. The error recovery block produces the correct result in two cycles, $T_S^{REC} < 2T_{clk}$. If ERR flags no error, the speculative result is correct. Otherwise, error recovery produces the correct result in an additional cycle. The effective cycle of the design, T_{ave} , is stated as the same as that of VLCSA 1:

$$T_{ave} = T_{clk}(1 - P_{err}) + 2T_{clk}P_{err} \quad (6.1)$$

where P_{err} is the error rate of the speculative addition. Ideally, if the error rate is tiny, the average performance of VLCSA 2 is also close to that of VLCSA 1.

Compared with VLCSA 1, we add new speculative result and error detection signal in VLCSA 2. This causes extra delay penalty and area overhead. Besides, there is no analytical error rate model for 2's complement Gaussian inputs. For similar adder settings, the error rate for 2's complement Gaussian inputs may be higher than that for unsigned random inputs. We will employ the experimental results for 2's complement Gaussian

inputs to profile the error rate.

Chapter 7

Results

7.1 Simulation setup

We have implemented C++ programs which take the adder width n and the window size k , and generate Verilog files for the SCSA-based speculative adder, VLCSA 1 and 2. Circuits are synthesized using a common standard library for UMC 65 *nm* CMOS technology in the Synopsys Design Compiler.

We first compare the delay and area of SCSA-based speculative adder, VLCSA 1 and 2 with the Kogge-Stone adder and the variable latency adder in [17]. Furthermore, we compare the delay and area of SCSA-based speculative adder, VLCSA 1 and 2 with the adder generated by the DesignWare building block IP [1], called DesignWare adder.

7.2 Error model validation

We verify the analytical error model by comparing it with simulation results. The simulation results are obtained by running Monte Carlo simulations for 10 million unsigned random inputs for different adder widths and window sizes. As shown in Figure 7.1, the solid lines are generated using the analytical error model for different adder widths. The marked points are simulation results. The analytical and experimental results fit quite well

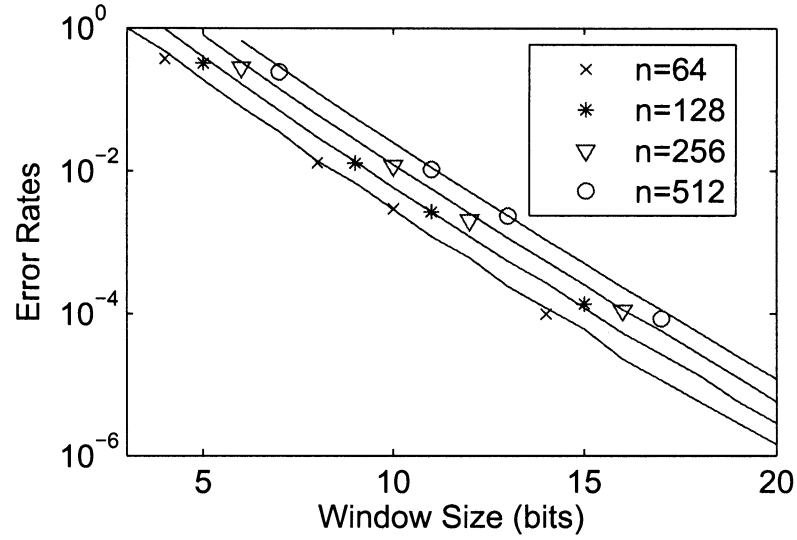


Figure 7.1: Comparison of analytical error model for SCSA and simulation results for different adder widths (n).

for different adder widths and window sizes. Thus, the analytical model accurately predicts simulation results.

7.3 Error rates for 2's complement Gaussian inputs

As discussed in Chapter 6, there is no analytical error model for 2's complement Gaussian inputs. We estimate the error rate of speculative addition by running Monte Carlo simulations. For the Gaussian distribution, the mean is $\mu = 0$, and the standard deviation is $\sigma = 2^{32}$.

We first discuss the speculative addition in VLCSA 1. An error occurs if the speculative result is different from that of the traditional addition. We also report the nominal error rate based on error detection, ERR . Simulation results are generated by running Monte Carlo simulations for 1 million 2's complement Gaussian inputs, as shown in Table 7.1. We observe that the error rate is clearly larger than that for unsigned random inputs. This

adder width	window size	P_{err} (Monte Carlo)	P_{err} ($ERR = 1$)
64	14	25.01%	25.01%
128	15	25.01%	25.01%
256	16	25.01%	25.01%
512	17	25.01%	25.01%

Table 7.1: Experimental and nominal error rates in VLCSA 1 for 2's complement Gaussian inputs. $\mu = 0, \sigma = 2^{32}$.

implies that VLCSA 1 becomes much slower for 2's complement Gaussian inputs: every one out of four computations in VLCSA 1, the error recovery produces correct result and incurs extra delay penalty.

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

Table 7.2: Experimental and nominal error rates in VLCSA 2 for 2's complement Gaussian inputs. $\mu = 0, \sigma = 2^{32}$.

Then we discuss the speculative addition in VLCSA 2. There are two speculative results in VLCSA 2. If either speculative result is the same as the traditional result, the speculation is correct. We also report the nominal error rate based on the error detection signals, ERR_0 and ERR_1 . The simulation results are also generated by running Monte Carlo simulations for 1 million Gaussian inputs, as shown in Table 7.2. We can observe that the error rate is clearly smaller than that of VLCSA 1, which is similar to that for unsigned random inputs. In another word, the error rate effectively reduces from 25.01% in VLCSA 1 to

0.01% in VLCSA 2 for same 2's complement Gaussian inputs. This implies that VLCSA 2 can successfully handle the 2's complement Gaussian inputs by introducing additional speculative result and error detection signal. The performance of VLCSA 2 can be close to that of VLCSA 1.

7.4 Comparison with existing variable latency adders

adder width	window size	carry chain length [17]
64	14	17
128	15	18
256	16	20
512	17	21

Table 7.3: Parameters of SCSA and the speculative adder in [17] for an error rate of 0.01% , according to analytical error models and simulation results

It is worthwhile to compare the proposed adders with the traditional adder and existing variable latency adders. The adder in [17], called *variable latency speculative adder* (VLSA) , is one of the best state-of-the-art variable latency adders and employed here. VLSA is designed for unsigned random inputs. We compare the delays and areas among the Kogge-Stone adder, VLSA and VLCSA 1 for unsigned random inputs. The parameters of speculative adders are shown in Table 7.4 for an error rate of 0.01%. Two small adders of the window adder in VLCSA 1 are implemented using Kogge-Stone adder. In all cases, we optimize for minimal delay during synthesis.

7.4.1 Speculative addition in VLCSA 1 VS speculative addition in VLSA

The speculative addition in VLCSA 1 is SCSA 1. As shown in Figure 7.2, for an error rate of 0.01%, the critical path delay of SCSA 1 is 18 to 38% lower than that of Kogge-Stone

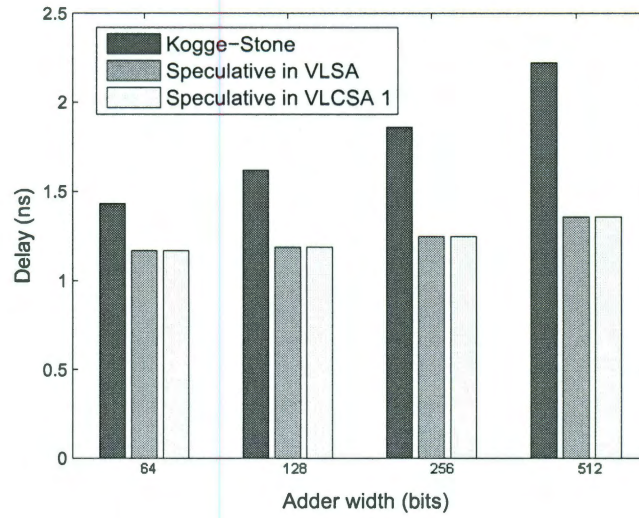


Figure 7.2: Comparison of delay of speculative adders and Kogge-Stone adder.

adder. For an error rate of 0.01%, the critical path delay of SCSA 1 is similar to that of the speculative addition in VLSA. This indicates that SCSA 1 can perform quite well.

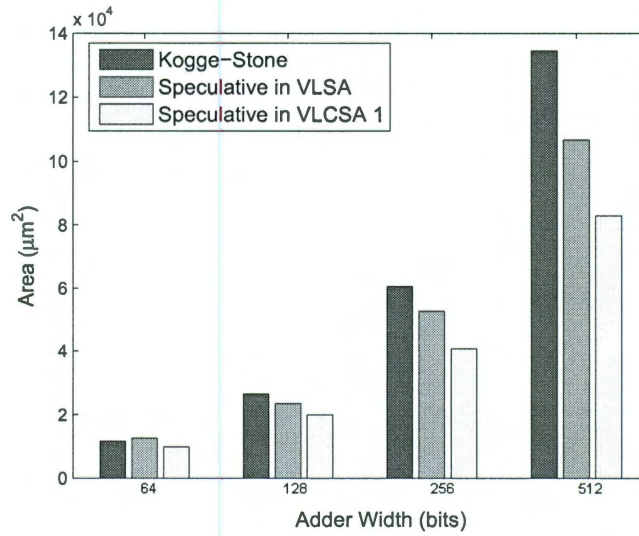


Figure 7.3: Comparison of area of speculative adders and Kogge-Stone adder.

As shown in Figure 7.3, for an error rate of 0.01%, the area of SCSA 1 is 15 to 38%

lower than Kogge-Stone adder. For an error rate of 0.01%, the area requirement of the speculative adder in VLSA is -20 to 8%. We can observe that the area of SCSA 1 is always smaller than that in VLSA for different bitwidths. This is mainly because the speculation in SCSA 1 is on the level of the window while the counterpart in VLSA speculates on the level of the individual bit position.

7.4.2 VLCSA 1 VS VLSA

We compare delays of Kogge-Stone adder, VLCSA 1 and VLSA. For the variable latency adder, there are three delays: speculative addition, error detection and error recovery. As

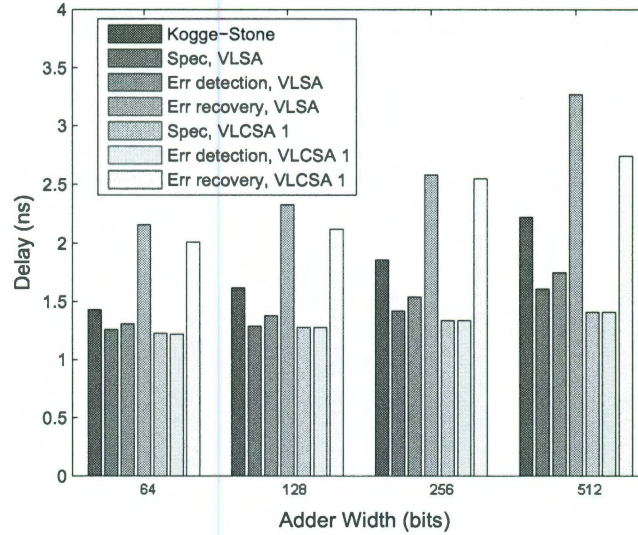


Figure 7.4: Comparison of delay of variable latency adders and Kogge-Stone adder.

shown in Figure 7.4, the critical path delay of speculative adder in VLSA is 12 to 27% shorter than that of Kogge-Stone adder. However, the critical path delay of the error detection block is 4 to 8% higher than that of the speculative adder, offsetting the advantage of speculation. The delay of the error recovery block is less than twice of the longer of speculative addition and error detection. In contrast, error detection and speculative addition

in VLCSA 1 has almost the same critical path delay, both are 14 to 36% shorter than that of Kogge-Stone adder. The simple circuitry of error detection in our design results in the low latency. In general, the critical path delay of VLCSA 1 is 6 to 19% lower than that of VLSA when speculation is correct. The critical path delay of the error recovery block is less than twice of the longer of speculative addition and error detection.

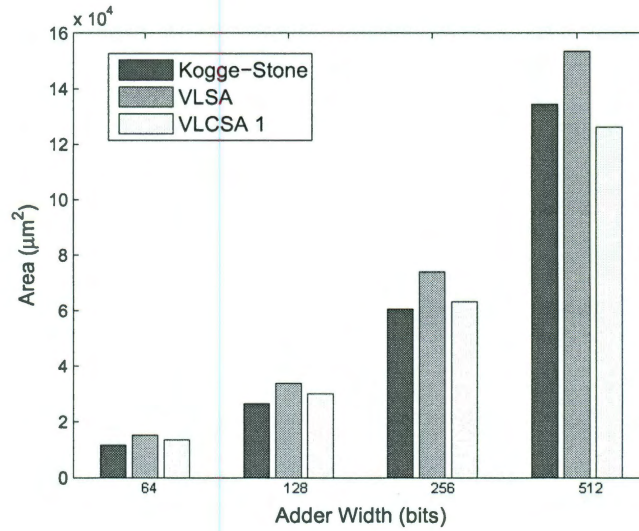


Figure 7.5: Comparison of area of variable latency adders and Kogge-Stone adder.

As shown in Figure 7.5, the area of VLSA is 14 to 32% larger than that of Kogge-Stone adder. This is due to the area overhead of the error detection and recovery blocks. On the other hand, the area requirement of VLCSA 1 is -6 to 17% smaller than that of Kogge-Stone adder. In particular, the area of VLCSA 1 is 6% smaller than that of Kogge-Stone adder when the bitwidth is 512. In another word, VLCSA 1 can be smaller and faster than Kogge-Stone adder.

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

Table 7.4: Parameters of SCSA and VLCSA 1 for the error rates of 0.01% and 0.25% , according to analytical error models and simulation results.

7.5 Comparison with DesignWare adder

The Synopsys DesignWare building block IP is a collection of highly optimized reusable IP blocks, which can quickly provide desirable designs during synthesis [1]. In particular, the DesignWare building block IP can generate high-quality adder designs for timing, area and power [19]. The DesignWare adder is synthesized for the minimal achievable delay, called *DesignWare adder*. We implemented a hybrid Kogge-Stone carry-select adder and observed that the adder generated by the DesignWare building block IP is faster than the hybrid one. The further details of the DesignWare building block IP can be referred to [1].

We compare SCSA, VLCSA 1 and 2 with the DesignWare adder. Note that SCSA stands for the SCSA-based speculative adder design. The parameters of SCSA are reported in Table 7.4 for error rates 0.01% and 0.25%. Two small adders of the window adder are implemented using DesignWare IP block. We also discuss the effect of different error rates. In comparison to the DesignWare adder, we target to achieve 10% critical path delay reduction and zero area overhead over the DesignWare adder during synthesis.

7.5.1 Speculative addition in VLCSA 1 vs DesignWare adder

The speculative addition in VLCSA 1 is SCSA 1. As shown in Figure 7.6, the critical path delays of SCSA 1 are 10% lower than those of the DesignWare adder for error rates 0.01%

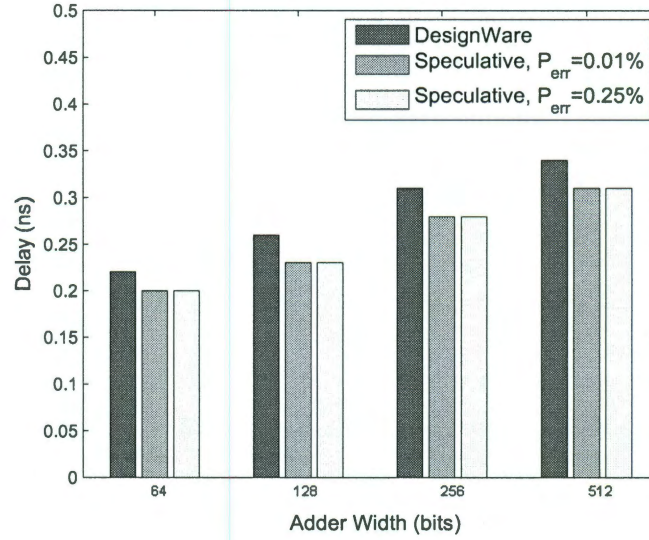


Figure 7.6: Comparison of delay of speculative addition in VLCSA 1 and DesignWare adder.

and 0.25%. In another word, if a certain level of inaccuracy is acceptable, SCSA 1 is faster than the DesignWare adder.

As shown in Figure 7.7, for an error rate 0.01%, as the adder width increases, the area of the SCSA 1 can be 43% smaller than that of the DesignWare adder. For an error rate 0.25%, the area of the SCSA 1 is 21 to 56% smaller than that of the DesignWare adder.

SCSA 1 with a lower error rate has larger area than the one with a higher error rate. Thus, there is a tradeoff between the error rate and area. When certain application is more error-tolerable, the error rate may slightly increase to clearly reduce area. Similarly, there is a tradeoff between the error rate and delay.

7.5.2 VLCSA 1 vs DesignWare adder

Next we compare VLCSA 1 with the DesignWare adder. The parameters of the speculative adder in VLCSA 1 also follow Table 7.4. As the delays of the speculative adder and the error detection block are designed to be very close, we only show one of them for simplicity.

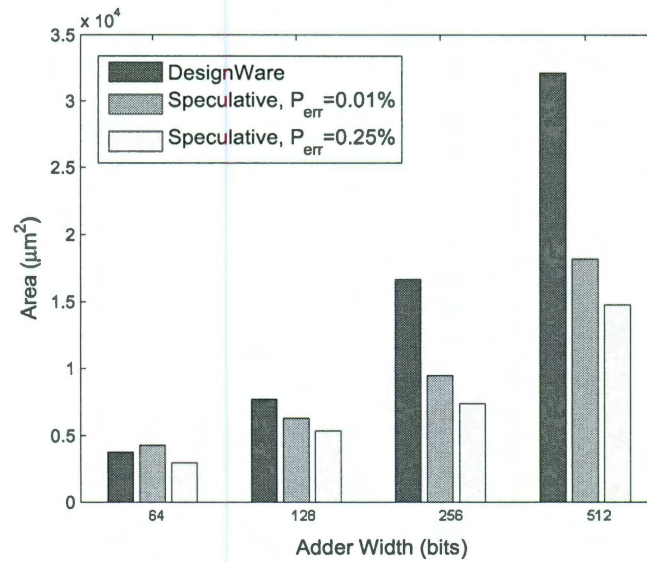


Figure 7.7: Comparison of area of speculative addition in VLCSA 1 and DesignWare adder.

The longer one of the critical path delays of the speculative adder and error detection block is stated as the "correctly speculated" delay.

As shown in Figure 7.8, 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 lower than twice of the "correctly speculated" delays.

As shown in Figure 7.9, for an error rate 0.25% (0.01%), VLCSA 1 has area requirements of -19 to 16% (-6 to 42%) over the DesignWare adder. As the adder width increases, the VLCSA 2 has less area requirements over the DesignWare adder. If the error rate is 0.25% instead of 0.01%, on average, we can save 17% area by increasing 0.12% average cycle. For example, we may choose an error rate 0.25% to save significant area while decreasing a little bit performance. The tradeoff between the error rate and area is valuable for saving area.

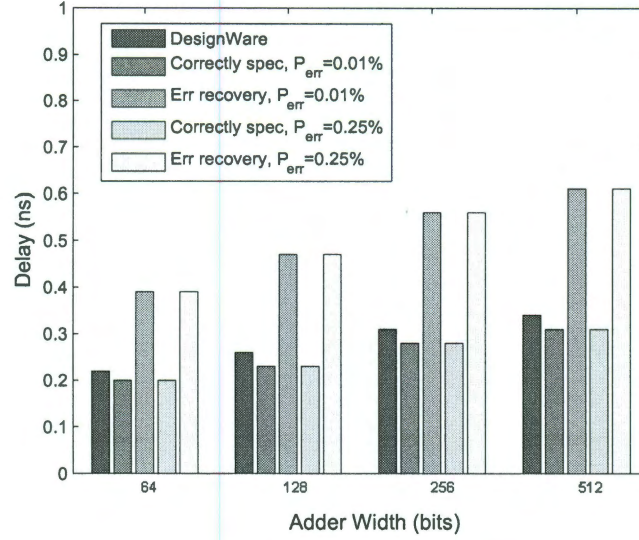


Figure 7.8: Comparison of delay of VLCSA 1 and DesignWare adder.

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

Table 7.5: Parameters of VLCSA 2 for the error rates of 0.01% and 0.25% , according to simulation results. $\mu = 0, \sigma = 2^{32}$.

7.5.3 VLCSA 2 vs DesignWare adder

Finally, we compare VLCSA 2 with the DesignWare adder. The error rates of the speculative addition in VLCSA 2 are obtained using simulation results. The parameters of the speculative adder in VLCSA 2 are reported in Table 7.5.

As shown in Figure 7.10, the critical path delays of VLCSA 2 are 10% lower than those of the DesignWare adder when speculation is correct. If the error rate is tiny, the average

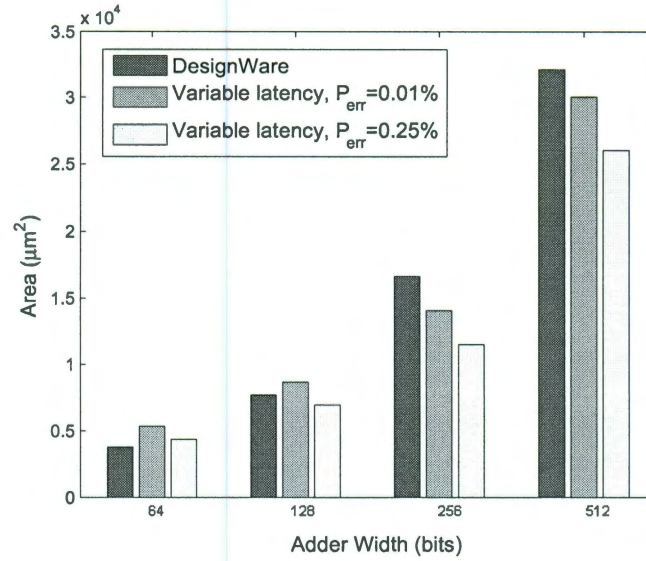


Figure 7.9: Comparison of area of VLCSA 1 and DesignWare adder.

performance of VLCSA 2 is close to that of the speculative adder.

As shown in Figure 7.11, for an error rate 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 of speculative addition and error detection. As the adder width increases, VLCSA 2 has less area requirements over the DesignWare adder. Similarly, if the error rate is 0.25% instead of 0.01%, on average, we can save 20% area by increasing 0.12% average cycle.

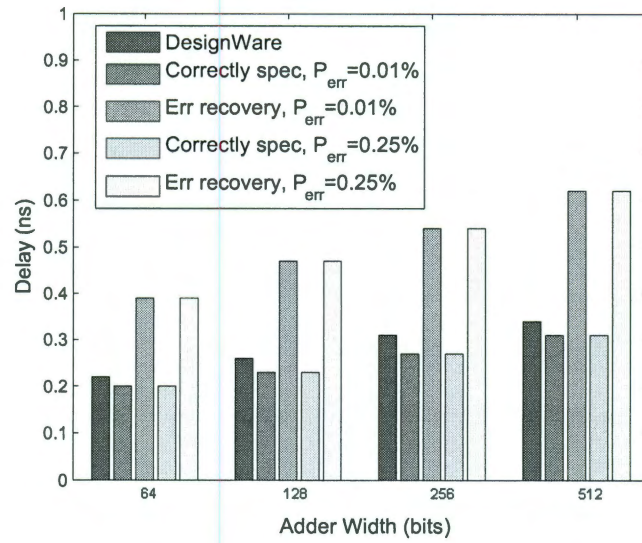


Figure 7.10: Comparison of delay of VLCSA 2 and DesignWare adder.

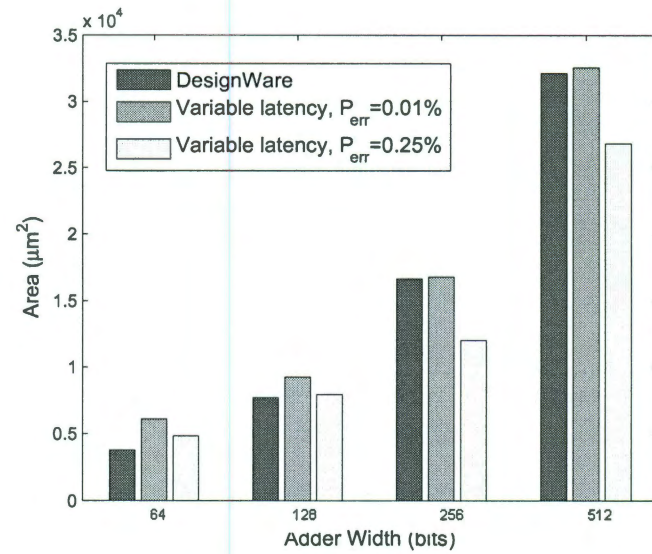


Figure 7.11: Comparison of area of VLCSA 2 and DesignWare adder.

Chapter 8

Conclusion

In this thesis, we propose a novel function speculation technique, called speculative carry select addition (SCSA). We develop an analytical error model for unsigned random inputs. Then We develop a speculative adder design based on SCSA. Simulation results show that, for an error rate of 0.01% (0.25%), SCSA-based speculative adder can be 10% faster and 43% (56%) smaller than the fastest adder generated by the DesignWare building block IP, called Designware adder.

Next we present a reliable variable latency adder design that augments the speculative adder with error detection and recovery for unsigned random inputs, called variable latency carry selection adder 1 (VLCSA 1). Simulation results show that the critical path delay of VLCSA 1 is 10% lower than that of the DesignWare adder when speculation is correct. For an error rate 0.25% (0.01%), VLCSA 1 has area requirements of -19 to 16% (-6 to 42%) over the Designware adder.

Furthermore, we develop a modified variable latency adder design suitable for both unsigned random and 2's complement Gaussian inputs, called VLCSA 2. The key idea of VLCSA 2 is to correctly speculate results when long carry chains occur. We re-design the speculative adder and the error detection block in VLCSA 2. Simulation results show that the critical path delay of VLCSA 2 is still 10% lower than that of the Designware adder when speculation is correct. For an error rate 0.25% (0.01%), VLCSA 2 has area

requirements of -17 to 29% (1 to 62%) over the DesignWare adder.

In summary, simulation results suggest that the proposed speculative adder can be faster and smaller than the DesignWare adder for very low error rates. The reliable variable latency adder can outperform the DesignWare adder in both delay and area. Besides, the proposed speculative and reliable variable latency adders are smaller than one of the best speculative and reliable latency adders [17] for similar design settings. The proposed reliable variable latency adder is also faster than the counterpart in [17].

In future, we plan to generalize the speculative and reliable variable latency carry select addition for floating-point numbers, or other arithmetic operations such as multiplication and multi-operand addition. We also plan to apply the speculative and reliable variable latency carry select addition for certain applications such as digital signal processing.

Bibliography

- [1] DesignWare building block IP user guide. <http://www.synopsys.com/dw/dwlibdocs.php>.
- [2] T. Austin et al. Opportunities and challenges for better than worst-case design. In *Proc. Asia and South Pacific Design Automation Conference*, pages 2–7, 2005.
- [3] Bannères et al. Variable-latency design by function speculation. In *Proc. Design, Automation and Test in Europe*, pages 1704–1709, 2009.
- [4] L. Benini et al. Telescopic units: A new paradigm for performance optimization of VLSI designs. *IEEE Trans. Computer-aided Design*, 17(3):220–232, 1998.
- [5] L. Chakrapani et al. Highly energy and performance efficient embedded computing through approximately correct arithmetic. In *Proc. Intl. Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pages 187–196, 2008.
- [6] A. Cilardo. A new speculative addition architecture suitable for two’s complement operations. In *Proc. Design, Automation and Test in Europe*, pages 664–669, 2009.
- [7] D. Ernst et al. Razor: a low-power pipeline based on circuit-level timing speculation. In *Proc. Intl. Symposium on Microarchitecture*, pages 7–18, 2003.
- [8] R. Hegde and N. R. Shanbhag. Soft digital signal processing. *IEEE Trans. VLSI Systems*, 9(6):813–823, 2001.
- [9] D. Kelly and J. Phillips. Arithmetic data value speculation. In *Advances In Computer Systems Architecture, Lecture Notes in Computer Science*, pages 353–366, 2005.

- [10] I. Koren. *Computer Arithmetic Algorithms*. A K Peters, Ltd., 2002.
- [11] T. Liu and S. Lu. Performance improvement with circuit-level speculation. In *Proc. Intl. Symposium on Microarchitecture*, pages 348–355, 2000.
- [12] Y. Liu et al. Design methodology of variable latency adders with multistage function speculation. In *Proc. Intl. Symposium on Quality Electronic Design*, pages 824–830, 2010.
- [13] S. Lu. Speeding up processing with approximation circuits. In *Computer*, volume 37, pages 67–73, 2004.
- [14] S.M. Nowick et al. Speculative completion for the design of high-performance asynchronous dynamic adders. In *Proc. of 3rd Int. Symp. on Advanced Research in Asynchronous Circuits and Systems*, pages 210–223, 1997.
- [15] B. Parhami. *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, New York, 2000.
- [16] Y.S. Su et al. An efficient mechanism for performance optimization of variable-latency designs. In *Proc. Design Automation Conference*, pages 976–981, 2007.
- [17] Ajay K. Verma et al. Variable latency speculative addition: a new paradigm for arithmetic circuit design. In *Proc. Design, Automation and Test in Europe*, pages 1250–1255, 2008.
- [18] N. Zhu et al. An enhanced low-power high-speed adder for error-tolerant application. In *Proc. 12th International Symposium on Integrated Circuits*, pages 69–72, 2009.
- [19] R. Zimmermann. Datapath Synthesis for Standard-Cell Design. In *Proc. Intl. Symposium on Computer Arithmetic*, pages 207–211, 2009.