High-Performance Low-Power Carry Speculative Addition With Variable Latency

Ing-Chao Lin, *Senior Member, IEEE*, Yi-Ming Yang, and Cheng-Chian Lin

***Abstract*— Adders are one of the most critical arithmetic circuits in a system and their throughput affects the overall performance of the system. Traditional n-bit adders provide accurate results, but the lower bound of their critical path delay is $\Omega(\log n)$. To achieve a critical path delay lower than $\Omega(\log n)$, many approximate adders have been proposed. These approximate adders decrease the critical path delay and improve the speed by sacrificing computation accuracy or predicting the computation results. This paper proposes a high-performance low-power carry speculative adder (CSPA). This adder separates the carry generator and sum generator. Only one sum generator is used in a block adder to reduce the critical path delay and area overhead. In addition, to generate 100% accurate results, error detection and recovery circuits are added to the proposed CSPA to construct a variable-latency carry speculative adder (VLCSPA). Instead of recalculating all results, the error detection and recovery circuits find and correct the block adder that generates incorrect partial sum bits, reducing power consumption. The experimental results show that the proposed CSPA achieves a 26.59% delay reduction, a 14.06% area reduction, and a 19.03% power consumption reduction compared to the corresponding values for an existing speculative carry-select adder. The experimental results also show the proposed CSPA can be used to improve image denoising results as well.**

***Index Terms*— Approximate adder, error detection, error recovery, speculative adder, variable latency.**

I. Introduction

ADDERS are one of the most critical arithmetic circuits in digital systems. They are widely used in applications such as arithmetic logic units and digital signal processors. Many kinds of adders have been proposed including the ripple carry adder (RCA), carry look-ahead adder (CLA), and parallel prefix adders. In [1], it was shown that the lower bounds of the critical path delay and area overhead of an n-bit traditional adder are $\Omega(\log n)$ and $\Omega(n)$, respectively. This indicates that reliable adders cannot be implemented with a sub-logarithmic delay.

In order to implement adders with a sub-logarithmic delay, many approximate circuits have been proposed [2]–[8]. These approximate circuits decrease circuit delay power consumption, and verification and test costs by sacrificing computation accuracy. In [2] and [3] approximate full adders that reduce circuit complexity at the transistor level were proposed. Approximate full adder cells are used in the least significant bits (LSBs) to ensure higher quality output of the multibit adders. In [4], a logic synthesis approach was proposed for designing circuits that implement approximate versions of the given function and achieve a minimum area for a given error rate threshold. The authors complement min terms of the original function to maximize the literal reduction of the synthesized circuit.

Four approximate adders, namely ETAI, ETAII, ETAIIM, and ETAIII have been proposed [5]–[8] to achieve lower delay and power consumption. ETAI splits the input operands into two parts. The left part is the accurate part which is calculated using an accurate adder and the right part is the inaccurate part which is calculated by an approximate adder. These two parts are executed simultaneously to obtain approximate results. ETAII divides the adder into many smaller adders with shorter carry propagation paths. The carry-in bit of each smaller adder is set to 0 and each adder is operated concurrently to speed up addition operations. ETAIIM connects the carry chains of ETAII in higher order bits to obtain accurate results in the most significant bits (MSBs) and thus provides final results that are more accurate than those of ETAII. ETAIII is an improved version of ETAI. Unlike ETAI, it uses a selector circuit to determine the bit width of the accurate part. Although these approximate adders are simple, they also have higher error rates.

Verma *et al.* [9] observed that the carry propagation length in an n-bit adder is less than $\log(n)$ in most cases. They used many smaller adders that have input widths of less than $\log(n)$ to build an n-bit adder. Most input bits of a smaller adder overlap with those of another smaller adder to increase the accuracy of the results. However, overlapping adders also increase the primary input fan out, area overhead and power consumption. To reduce the primary input fan out and area overhead, Du *et al.* [11], [12] proposed a speculative carry select adder (SCSA). They segment the n-bit adder into several independent block adders with the same size. Each block adder has its own prediction circuit, which uses all input bits of its corresponding block adder to predict the carry-out bit of the block adder. Compared to [9], the primary input fan out and area are reduced because one input is only connected to one block adder. Compared to traditional adders, these approximate adders trade

Manuscript received October 22, 2013; revised May 30, 2014; accepted July 27, 2014. Date of publication September 29, 2014; date of current version August 21, 2015. This work was supported in part by the National Science Council of Taiwan under Grant 102-2221-E-006-281 and in part by the Ministry of Science and Technology, Taiwan, under Grant 103-2221-E-006-255.

I.-C. Lin and C.-C. Lin are with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan 701, Taiwan (e-mail: iclin@mail.ncku.edu.tw; p76021124@mail.ncku.edu.tw).

Y.-M. Yang is with VIA Technology, Inc., Taipei 23141, Taiwan (email: leo771031@gmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2014.2355217

computation accuracy for lower delay and power consumption However, these adders can only be used in error to lerant applications, such as video, audio and image processing. To allow approximate adders to be used in applications that require accurate results, error detection and recovery circuits need to be added. Verma *et al.* [9] added error detection and recovery circuits into an approximate adder to construct a variable latency speculative adder (VLSA). When input patterns arrive, the approximate adder and error detection circuit operate simultaneously. The error detection circuit detects whether an error has occurred. If no error is found, accurate results can be generated in a cycle. If an error has occurred, the error recovery circuit is used to obtain accurate results after two cycles. Because the error rate of the approximate adder is low, the average latency of VLSA is close to the delay of the approximate adder. Du *et al.* [11], [12] also added error detection and recovery circuits to their approximate adders and proposed SCSA-based variable latency adders. The error recovery circuits in [9], [11], and [12] use prefix adders to calculate correct carry-out bits. These prefix adders are executed simultaneously, leading to large power consumption.

In this paper, we propose a novel high performance low power approximate adder, called the carry speculative adder (CSPA). The CSPA divides an n-bit adder into several smaller block adders that can operate independently. Each block adder has its own carry predictor circuit to predict the carry-out bit. Instead of using all input bits in a block adder, the carry predictor circuit only uses the input bits that are near the MSB. Since the carry predictor uses less bits to predict the carry-out bit, the critical path delay is reduced. The proposed CSPA also separates the carry and sum generator circuits to reduce the critical path delay. Error detection and recovery circuits are added to the CSPA to construct a variable latency carry speculative adder (VLCSPA). The contributions of this paper are summarized as follows.

1) The carry generator and sum generator are separated in the proposed CSPA. Only one sum generator is needed in the block adder. Therefore, the area and critical path delay of the proposed CSPA are reduced.
2) The carry predictor circuits of the block adder only use the input bits that are near the MSB to predict the carry-out bit of the corresponding block adder, reducing area and power consumption with minimal accuracy loss.
3) The proposed error detection circuit can indicate which block adder generates an incorrect carry-out bit. As a result, the error recovery circuits only focus on recovering the block adder that has incorrect partial sum bits reducing the power consumption of the VLCSPA.
4) To determine the block adder size when segmenting an n-bit adder, an error-energy-delay product (EEDP) model is proposed to determine the appropriate block adder size to balance delay, power consumption, and error rate.
5) The experimental results show that the proposed CSPA can achieve a 26.59% delay improvement, a 14.06% area reduction, and a 19.03% power consumption reduction compared to the corresponding values for the SCSA-based adder. Compared to the SCSA-based variable latency adder, the delay, area and power consumption improvements of VLCSPA are 27.26%, 10.31%, and 19.25%, respectively.

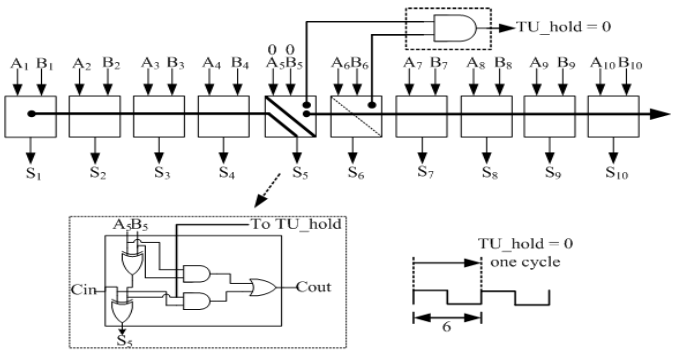

Fig. 1. Ten-bit RCA with hold logic.

The remaining sections are organized as follows: Section II introduces variable latency design related works and accuracy metrics for approximate design. Section III details the proposed CSPA, error detection circuit, error recovery circuit and the VLCSPA. Section IV shows the experimental results and the proposed EEDP model, and Section V concludes the paper.

II. Preliminaries

A. Variable-Latency Design

The main purpose of a variable latency design is to reduce the timing waste when using the critical path delay as the execution cycle period. Traditional circuit design uses the critical path delay as the circuit clock cycle to ensure that all computations complete in time. However, the probability of the critical paths that are triggered is low, and the path delay of most paths is shorter than the critical path delay. Thus, using the critical path delay as the cycle period for these non-critical paths it will causes significant timing waste.

A variable latency design divides the circuit into two parts, that with shorter paths and that with longer paths. The basic concept of a variable latency design is to use a shorter cycle to execute a shorter path, and use two cycles to execute a longer path. When most paths execute in shorter cycles, the average latency of the variable latency design is better than that of a traditional design. For example, Fig. 1 shows a 10-bit variable-latency RCA. Assume that the maximum delay for each full adder is one. If only one cycle is used for all paths, the clock period for the 10-bit RCA is at least 10. However, the possibility of a carry propagation delay of more than 6 is low, as obtained from simulation. The cycle period is set to 6 to allow most operations to be finished. The hold logic circuit is shown on the bottom left of Fig. 1. The function of the hold logic circuit is (A5 XOR B5) (A6 XOR B6). If A5 $\neq$ B5 and A6 $\neq$ B6, the TU_hold signal is 1, which notifies the system that this input pattern may trigger a path delay that is longer than 6. Therefore, the circuit requires two cycles to complete the current operation. Otherwise, when the

TU_hold signal is 0, the sixth adder will not have a carry out bit. Hence, the maximum delay will be less than one cycle period.

Based on the above, a comparison of variable-latency delay and fixed-latency delay is as follows. If the possibility of each input being 1 is 0.5 and the TU_hold signal being 1 is 0.25, then the average latency of the variable-latency design is $0.25 \times 12 + 0.75 \times 6 = 7.5$. Compared with the fixed-latency RCA, whose average latency is 10, the delay of the variable-latency design is 33% lower.

Hence, many studies have used the variable latency technique to decrease the average latency. In [13], a variable-latency pipelined multiplier with Booth encoding logic was proposed. Variable-latency adder consider the aging effect were proposed in [14] and [15]. An instruction scheduling algorithm was proposed in [16]. Operations are scheduled on nonuniform latency functional units and the delay of VLIW processors is reduced. In addition, a hold logic circuit is needed to determine whether the input pattern can complete the operation within a cycle period in the variable latency design. Su *et al.* [17] proposed a short path activation function algorithm to obtain a hold logic circuit with higher accuracy and improve the delay of the variable latency circuit.

B. Speculative Adders

The speculative technique is an optimization technique based on a prediction mechanism for improving the delay of arithmetic circuits. This technique has been implemented in many works. Baneres *et al.* [18] proposed a two-stage function speculative technique to reduce arithmetic circuit delay. They first identified gates that have functions similar to the gates on the critical paths. These gates, called speculated nodes, replaced the gates on the critical paths. Since these speculated nodes are simplified gates and their delay is less than the original gates on the critical path, the critical path delay is reduced. It has been shown the critical path delay of the arithmetic circuit is at most reduced by half with the two-stage function speculation technique. Liu *et al.* [19] extended the two-stage function speculative technique [18] to a multistage architecture. They also replaced gates on the critical path using simplified gates with similar functions. Since more stages are used, this technique can further reduce the critical path delay. This technique is also applied to design a speculative adder that achieves less area overhead because of simplified gates.

Some works modified the architecture of traditional adders to obtain the speculative results. Kahng and Kang [10] proposed a configurable-accuracy adder to reduce the delay of the traditional n-bit adder. They split the traditional n-bit adder into several overlapped sub-adders to obtain the approximate results. Du *et al.* [11], [12] proposed the SCSA-based adder shown in Fig. 2. Then-bit adder is segmented into mx-bit block adders with the same size, where x is the size of the block adder. Each block adder contains two internal adders (adder_0 and adder_1) and a multiplexer. The carry-in signal of an internal adder is 0, while the carry-in signal of another internal adder is 1. The internal adders can be implemented using any

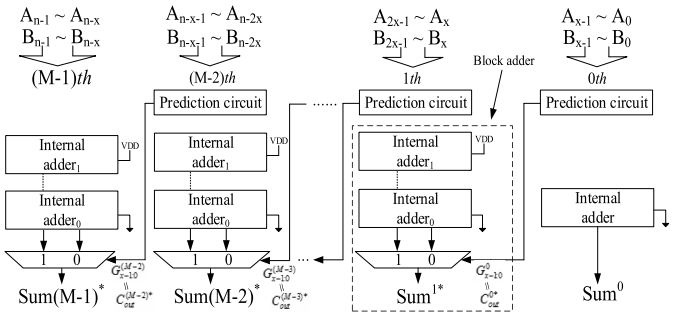

Fig. 2. SCSA-based adder.

traditional adder, and are used to calculate the approximate partial sum bits. The functional of the multiplexer is to select the partial sum bits from one of the internal adders according to the predicted carry-in signal generated by the prediction circuit in the previous stage. The prediction circuits of the adder use all input bits of the corresponding block adder to predict the carry-in bit of the next block adder, and use the bit to select the corresponding approximate partial sum bits of each block adder. This adder can significantly reduce the delay of a traditional adder because the adder carry chain becomes the length of block adder size.

In Fig. 2, the carry-out bit of the ith block adder, C^i_{out}, is given by

$$C^i_{\text{out}} = G^i_{x-1:0} + P^i_{x-1:0} \quad C^{i-1}_{\text{out}}, \quad 1 \le i < m \tag{1}$$

where $G^i_{x-1:0}$ and $P^i_{x-1:0}$ are the group propagate and generate signals at the $(x-1)$th bit position of the ith block adder. These parameters can be calculated as

$$P^i_{x-1:0} = \prod_{j=0}^{x-1} P^i_j$$

and

$$G^i_{x-1:0} = G_{x-1} + P_{x-1}G_{x-2} + \cdots + G^i_0 \prod_{j=1}^{x-1} P^i_j$$

where P^i_j is defined as $A^i_j \bigvee B^i_j$, and G^i_j is defined as $A^i_j \bigwedge B^i_j$.

C^{i-1}_{out} is assumed to be 0 in the ith block adder, and then C^i_{out} is approximated as C^{i*}_{out}

$$C^{i*}_{\text{out}} = G^i_{x-1:0}, \quad 1 \le i < m.$$

An error occurs if the ith block adder produces a group propagate signal $P^i_{x-1:0}$ with a value of 1 and the $(i-1)$th block adder produces a group generate signal $G^{i-1}_{x-1:0}$ with a value of 1. In this case $C^i_{\text{out}} \neq C^{i*}_{\text{out}}$ and an error result is produced from the adder. To obtain accurate results, error detection and recovery circuits are implemented. Fig. 3 shows the error detection circuit in [11]. An AND operation is performed between the group generate signal of the ith block adder and the group propagate signal of the $(i+1)$th block adder to detect whether an error has occurred.

Fig. 4 shows the error recovery circuit used in [11]. This error recovery circuit is implemented using a prefix adder.

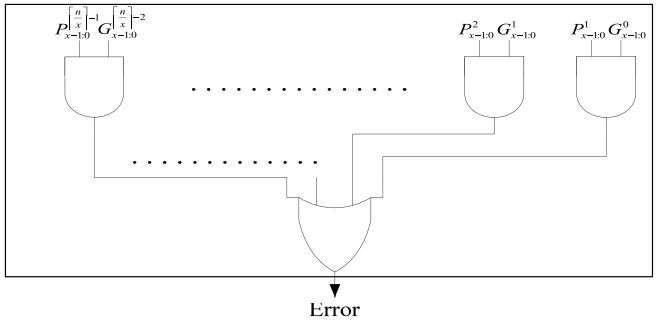

Fig. 3. Error detection circuit in [11].

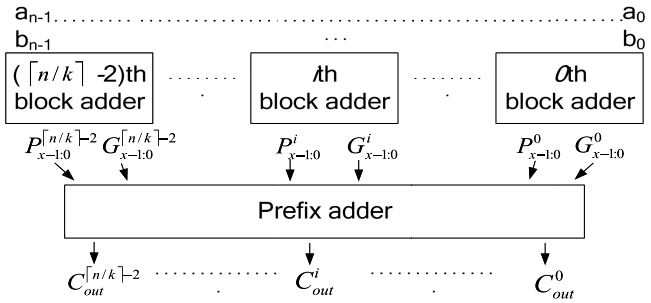

Fig. 4. Error recovery circuit in [11].

TABLE I
ACCURACY METRICS FOR APPROXIMATE DESIGNS

Metric	Definition
ACC_{amp}	$1 - \lvert R_c - R_e \rvert / R_c$
ACC_{inf}	$(1 - B_e)/B_w$

This prefix adder takes the group propagate and generate signals of block adders as inputs and calculates the correct carry-out bits of each block adder. Then the accurate results are computed using these correct carry-out bits. However, this error recovery circuit may cause larger delay and power consumption because all block adders need to be reexecuted to obtain the correct results. More details about the SCSA-based adder can be found in [11] and [12].

C. Metric for Approximate Designs

In this subsection, basic metrics are provided to quantify errors in an approximate design. Two accuracy metrics are defined in Table I. The accuracy for amplitude data (ACC_{amp}) used in [7] quantifies the amplitude of errors, where R_c and R_e are the correct and obtained results from the proposed adder, respectively. The accuracy for information data (ACC_{inf}) is used to measure error significance as a Hamming distance, where B_e is the number of error bits and B_w is the bit width of the data. For example, when the correct result is 10100000_2 and the obtained result is 10111000_2, ACC_{amp} and ACC_{inf} are $17/20$ and $3/4$, respectively.

These metrics are used in Section IV to evaluate the proposed speculative adder.

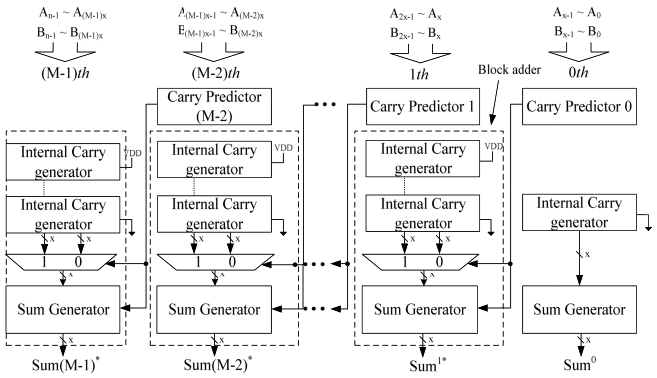

Fig. 5. Proposed architecture.

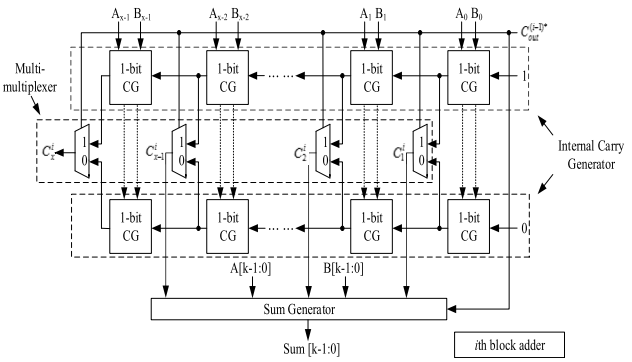

Fig. 6. Block adder architecture of proposed CSPA.

III. METHODOLOGY

This section first introduces the proposed CSPA, whose delay and area overhead are lower than those of the SCSA-based adder [11], [12]. Then an error model for analyzing the error rate of the CSPA and a method for estimating the critical path of the CSPA are proposed. This section also explains the proposed low-power error detection and correction circuits for the CSPA. Finally, a CSPA-based variable latency adder design with error detection and recovery circuits is proposed.

A. Carry Speculative Adder

Fig. 5 shows the proposed n-bit CSPA, which contains block adders and carry predictor circuits. Except for the leftmost block adder, the size of each block adder size is x. There are $m = [n/x]$ independent block adders and $(m - 1)$ carry predictor circuits in a CSPA. A carry predictor is used to predict the carry-out bit of the corresponding block adder. Each block adder has three key components: internal carry generator, multibit multiplexer, and sum generator. The internal carry generator is used to produce the internal carry signals that can be used in the sum generator. The output of the previous carry predictor circuit can be used as the selector of the multibit multiplexer to select one of the internal carry signals. The sum generator is used to calculate the partial sum bits of the block adder.

Fig. 6 shows the details of the block adder. An x-bit internal carry generator contains x 1-bit carry generators. The input

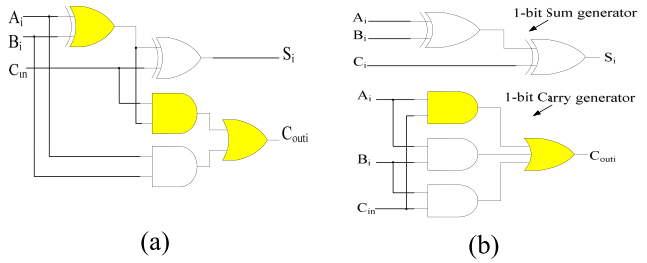

Fig. 7. (a) Traditional 1-bit full adder. (b) Modified 1-bit full adder.

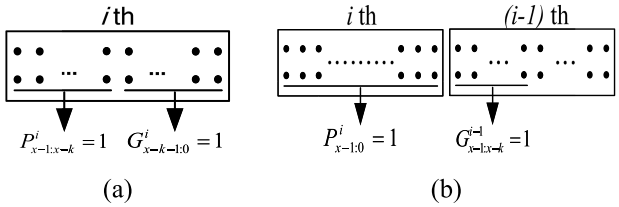

Fig. 8. Error occur when (a) $P^i_{x-1:x-k} = 1$ and $G^i_{x-k-1:0} = 1$ or (b) $P^i_{x-1:0} = 1$ and $G^{i-1}_{x-1:x-k} = 1$.

bit $C^{(i-1)*}_{\text{out}}$ is the carry-out bit of the *(i−1)*th block adder. This bit is used not only to choose one of the outputs from internal carry generators, but also as the carry-in bit of the sum generator. The sum generator produces the partial sum bits using the output from the internal carry generator and the input bit $C^{(i-1)*}_{\text{out}}$.

There are two differences between the proposed CSPA and the SCSA-based adder. The first difference is that each block adder in the SCSA-based adder uses two internal adders to generate possible partial sum bits. In contrast, our block adder uses two carry generators to generate possible internal carry signals and the selected carry signals are fed to a sum generator to generate the partial sum bits, as shown in Figs. 2 and 5, respectively. In the SCSA-based adder, the internal adders contain many 1-bit traditional full adders (TFA), as shown in Fig. 7(a). However, the carry-out bit of a TFA is produced after three gate delays. To reduce the delay of the carry-out bit, the block adder of the CSPA can be implemented using a modified full adder (MFA), as shown in Fig. 7(b). The MFA has a 1-bit carry generator and 1-bit sum generators that produce the carry and sum bits, respectively. The carry-out bit is produced after two gate delays. Therefore, the carry-out bit can be produced faster. However, the power consumption of an MFA is higher than that of a TFA since an extra logic gate is added. To reduce power consumption, the proposed CSPA uses two carry generators and a sum generator to implement the block adder instead of two internal adders. Therefore, the delay of the CSPA is lower than that of the SCSA-based adder since the carry generator and sum generator are separated, and the carry signals and the final sum bit can be calculated faster. In addition, the hardware cost of the proposed block adder is lower than that of the SCSA-based adder since only one copy of a sum generator is used.

The second difference is the carry predictor circuit. The prediction circuits of the SCSA-based adder consider all input bits of a block adder to predict the carry-out bit. To reduce the area overhead and hardware complexity of the prediction circuit, the proposed carry predictor circuit does not use all input bits of a block adder to predict the carry-out bit, but only uses the input bits that are near the MSB. The reason is the probability that a carry-out bit of the block adder depends on the k previous bit positions is $1/2^k$ since the probability of propagate signal Pi (a_i XOR b_i) having a value of 1 is $1/2$ in each bit position. When the input bits are near the LSB, they have a low possibility of affecting the carry-out bit. Hence, if the carry predictor circuit only usesthe input bits near the MSB to predict the carry-out bit, a low error rate can be maintained and the area overhead of the carry predictor circuits can be reduced.

The overall flow of the proposed CSPA is as follows. When input patterns arrive, the internal carry generators and the carry predictors operate simultaneously. The internal carry generator produces corresponding internal carry signals and the carry predictor predicts the carry-out bit of the block adder. The predicted carry-out bit is transferred to the next block adder, and used as the selector of the multiplexer and the carry-in bit of the sum generator. When the selector is determined, it can decide the corresponding internal carry signals, and send them to the sum generator. When the input signals of the sum generator (which includes input patterns, carry-in bit and internal carry signals) are all ready, the sum generator can calculate the partial sum bits of the block adder. The results generated from the proposed CSPA areal most correct since the possibility of incorrect prediction results is low.

B. Error Rate Analysis

This subsection analyzes the relationship between the block adder size, the carry predictor circuit size and the error rate of the CSPA. In the CSPA, the carry predictor circuit only uses k input bits near the MSB to predict the carry-out bit for each block adder. The approximate carry-out bit of the ith block adder, C^{i*}_{out} is given by

$$C^{i*}_{\text{out}} = G^i_{x-1:x-k}, \quad 1 \le i < m \tag{2}$$

where $G^i_{x-1:x-k}$ is the group generate signal from the $(x-k)$th to the $(x-1)$th bit position of the ith block adder, and m is the total number of block adders. Equation (1) can be expanded using (2) as follows:

$$C^i_{\text{out}} = G^i_{x-1:x-k} + P^i_{x-1:x-k} G^i_{x-k-1:0} + P^i_{x-k-1:0} C^{i-1}_{\text{out}}. \tag{3}$$

However, two cases result in erroneous results, as shown in Fig. 8. The first case is when $P^i_{x-1:x-k} = 1$ and $G^i_{x-k-1:0} = 1$, as shown in Fig. 8(a). $P^i_{x-1:x-k} = 1$ implies that $G_{x-1:x-k} = 0$. C^{i*}_{out} of the ith block adder is determined using (2) as

$$C^{i*}_{\text{out}} = G_{x-1:x-k} = 0. \tag{4}$$

The correct carry-out bit of the ith block adder is calculated using (3) as

$$C^i_{\text{out}} = G^i_{x-1:x-k} + P^i_{x-1:x-k} G^i_{x-k-1:0} + P^i_{x-k-1:0} C^{i-1}_{\text{out}} = 1.$$

Therefore, $C^i_{\text{out}} \neq C^{i*}_{\text{out}}$. This means that the carry predictor circuit generates an incorrect carry-out bit and the result of the CSPA is also incorrect.

The second case that leads to an incorrect carry-out bit is when $P_{x-1:0} = 1$ and $G^{i-1}_{x-1:x-k} = 1$, as shown in Fig. 8(b). $P^i_{x-k-1:0} = 1$ implies that $G^i_{x-k-1:0} = 0$ which means that the approximate carry-out bit of the ith block adder is 0. However, when the $(i-1)$ th block adder produces a group generate signal $G^{i-1}_{x-1:x-k} = 1$, it means that a carry-out bit with a logic value of 1 is produced from the carry predictor circuit of the $(i-1)$th block adder (i.e., $C^{i-1}_{\text{out}} = 1$). The real carry-out bit of the ith block adder is calculated using (1) as

$$C^i_{\text{out}} = G^i_{x-1:0} + P^i_{x-1:0} C^{i-1}_{\text{out}} = 1.$$

Therefore, $C^i_{\text{out}} \neq C^{i*}_{\text{out}}$ and the CSPA produces an incorrect result.

The probabilities of these two cases is calculated as follows. The error probability of the first case, $\boldsymbol{P}(P^i_{x-1:x-k} G^i_{x-k-1:0} = 1)$ is given by

$$\boldsymbol{P}\left(P^i_{x-1:x-k} G^i_{x-k-1:0} = 1\right) = \boldsymbol{P}\left(P^i_{x-k:x-k} = 1\right) \times \boldsymbol{P}\left(G^i_{x-k-1:0} = 1\right). \quad (5)$$

If the signal probability of the input variable is 0.5, the probabilities of the propagate and generate signals being 1 are

$$\boldsymbol{P}\left(P^i_{x-k:x-k} = 1\right) = \boldsymbol{P}\left(\prod_{j=x-k}^{x-1} p^i_j = 1\right) = \prod_{j=x-k}^{x-1} \boldsymbol{P}\left(p^i_j = 1\right) = \left(\frac{1}{2}\right)^k \quad (6)$$

$$\begin{aligned}\boldsymbol{P}\left(G^i_{x-k-1:0} = 1\right) &= \boldsymbol{P}\left(G^i_{x-k-1} + \cdots + G^i_0 \prod_{j=1}^{x-k-1} p^i_j = 1\right) \\ &= \boldsymbol{P}\left(G^i_{x-k-1} = 1\right) + \cdots + \boldsymbol{P}\left(G^i_0 \prod_{j=1}^{x-k-1} p^i_j = 1\right) \\ &= \left(\frac{1}{2}\right)\left[1 - \left(\frac{1}{2}\right)^{x-k}\right]. \end{aligned} \quad (7)$$

Based on (6) and (7), (5) is calculated as

$$\boldsymbol{P}\left(P^i_{x-1:x-k} G^i_{x-k-1:0} = 1\right) = \boldsymbol{P}\left(P^i_{x-k:x-k} = 1\right)\boldsymbol{P}\left(G^{i-1}_{x-k-1:0} = 1\right) = \left(\frac{1}{2}\right)^{x+1}\left[1 - \left(\frac{1}{2}\right)^{x-k}\right].$$

Similarly, the error probability of the second case, $\boldsymbol{P}(P^i_{x-1:x-k} G^{i-1}_{x-k-1:0}) = 1$ is calculated as follows:

$$\boldsymbol{P}\left(P^i_{x-1:x-k} G^{i-1}_{x-k-1:0} = 1\right) = \boldsymbol{P}\left(P^i_{x-1:0} = 1\right)\boldsymbol{P}\left(G^{i-1}_{x-k-1:0} = 1\right) = \left(\frac{1}{2}\right)^{x+1}\left[1 - \left(\frac{1}{2}\right)^{k}\right]. \quad (8)$$

Thus, the error probability of the ith block adder is obtained by adding (5) and (8)

$$\boldsymbol{P}^i_{\text{err}} = \left(\frac{1}{2}\right)^{k+1}\left[1 - \left(\frac{1}{2}\right)^{x-k}\right] + \left(\frac{1}{2}\right)^{x+1}\left[1 - \left(\frac{1}{2}\right)^{k}\right] \quad 1 \leq i < m.$$

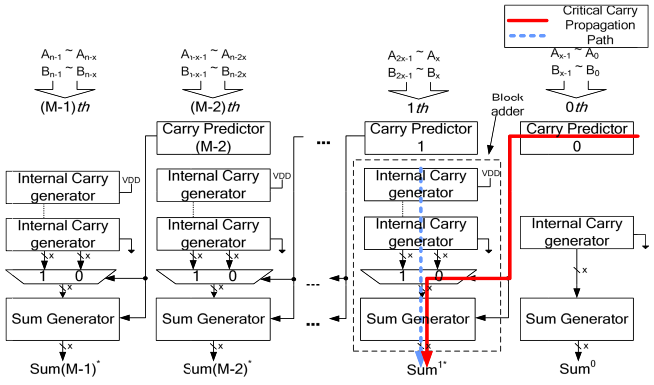

Fig. 9. Critical path delay analysis of CSPA.

The total error probability for the CSPA can be calculated by summing up the error probabilities for all block adders. The total error probability for the CSPA is

$$\begin{aligned}\boldsymbol{P}_{\text{err}} &= \sum_{i=0}^{\lceil \frac{n}{x} \rceil - 2} \boldsymbol{P}\left(P^i_{x-1:x-k} G^i_{x-k-1:0} = 1\right) + \boldsymbol{P}\left(P^i_{x-1:x-k} G^{i-1}_{x-k-1:0} = 1\right) \\ &= \sum_{i=0}^{\lceil \frac{n}{x} \rceil - 2} \left(\frac{1}{2}\right)^{k+1}\left[1 - \left(\frac{1}{2}\right)^{x-k}\right] + \left(\frac{1}{2}\right)^{x+1}\left[1 - \left(\frac{1}{2}\right)^{k}\right]. \end{aligned} \quad (9)$$

Equation (9) shows the relationship between the block adder size, the carry predictor circuit size and the error rate of the CSPA. The error rate of the adder is low when the block adder size and carry predictor size are large. An error rate comparison between the SCSA-based adder and the CSPA is shown in Section IV-A.

C. Critical Path Delay Analysis of CSPA

This section analyzes the critical path delay of the proposed CSPA. Fig. 9 shows the two possible critical carry propagation paths of the CSPA. The first possible path begins from the previous carry predictor circuit, through the multiplexer and ends at the sum generator (red solid line). Another one begins from the internal carry generator, through the multiplexer and ends at the sum generator (blue dashed line). The longest delay of each path is determined as [20]

$$D_{\text{red}} = k D_{\text{CG}} + D_{\text{MUX}(s)} + D_{\text{sum}} \quad (10)$$

$$D_{\text{blue}} = (x-1) D_{\text{CG}} + D_{\text{MUX(in)}} + D_{\text{sum}} \quad (11)$$

where k is the size of the carry predictor circuit and x is the block adder size. D_{CG}, $D_{MUX(s)}$, $D_{MUX(in)}$, and D_{SUM} are the 1-bit carry generator delay, multiplexer delay from selector, multiplexer delay from input and 1-bit sum generator delay, respectively. When the carry predictor circuit uses more input bits to predict the carry-out bit of a block adder, the red line has a higher chance of becoming the critical path of the CSPA; otherwise, the blue line is the critical path of CSPA.

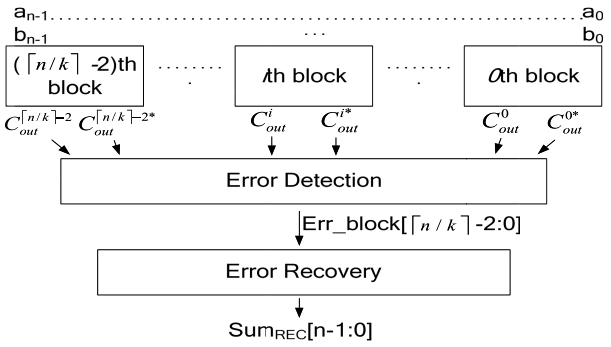

Fig. 10. Flow for error detection and recovery.

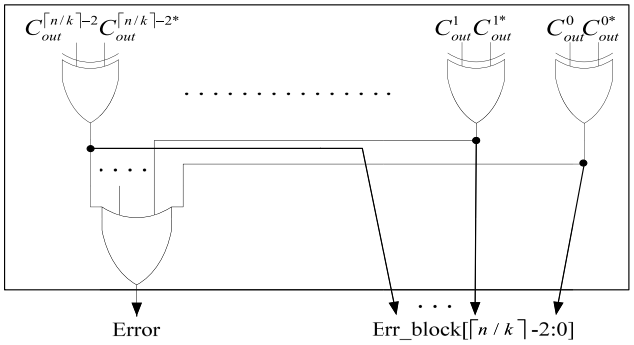

Fig. 11. Diagram of proposed error detection circuit.

D. Error Detection and Recovery

To obtain accurate results, error detection circuits are used to check whether an error has occurred, and error recovery circuits are used to correct the incorrect results. Fig. 10 shows the flow for the error detection and recovery circuits. First, C_{out}^{i} and C_{out}^{i*} are passed to the error detection circuit, where C_{out}^{i} is the correct carry-out bit of the ith block adder and C_{out}^{i*} is the approximated carry-out bit produced by the ith carry predictor circuit. The error detection circuit checks whether an error has occurred. Then the detection circuit transmits an *Err_block* signal to the error recovery circuit. The *Err_block* signal is generated to indicate which carry predictor circuit predicted the incorrect carry-out bit. Finally, the error recovery circuit modifies the incorrect partial sum bits and generates accurate results.

Fig. 11 shows the details of the proposed error detection circuit. When the error detection receives the C_{out}^{i} signal and C_{out}^{i*} signal of each block adder, it checks whether these two signals are the same. If C_{out}^{i} and C_{out}^{i*} are the same, it means that the carry-out bit of the ith block adder is correct. Otherwise, the predicted C_{out}^{i*} signal is not correct. Therefore, the C_{out}^{i} signal is decided before the detection circuit starts to operate. If the group propagate signal of the ith block adder (i.e., $p_{x-1:0}^{i}$) is 0, C_{out}^{i} is equal to the final carry signal of the ith internal carry generator (i.e., $C_{\text{out}}^{i} = C_{x}^{i}$). Otherwise, C_{out}^{i} is equal to the correct carry-out bit of the $(i-1)$th block adder (i.e., $C_{\text{out}}^{i} = C_{x}^{i}$).

Hence, the error detection circuit first performs an exclusive or (XOR) operation on C_{out}^{i} and C_{out}^{i*} to check which predicted carry-out bit is incorrect. Then the OR operation v is performed on the output of the XOR gate to determine the *Error* signal. When an error occurs, the error detection circuit sets the *Error* signal to 1; otherwise, the *Error* signal is 0. The major difference between the detection circuit in [11] and the proposed error detection circuit is that the former does not indicate which prediction circuit has generated the incorrect carry-out bit whereas the latter does, so our error recovery circuit only needs to correct the block adder that has incorrect partial sum bits.

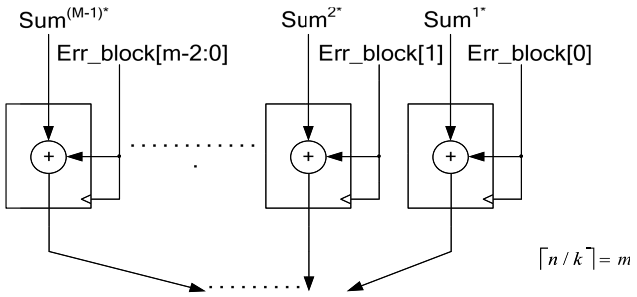

Fig. 12. Diagram of proposed error recovery circuit.

Fig. 12 shows the proposed error recovery circuit, which modifies the incorrect partial sum bits of the block adders according to the *Err_block* signal. In this figure, Sum^{i*} is the partial sum bits of the ith block adder and *Err_block* is the outputs of the error detection circuit. If *Err_block* [i] is 1, it means that the carry-out bit of the ith carry predictor circuit and the partial sum bits of the $(i+1)$th block adder are incorrect. If *Err_block* [i] is 0, it means that the $(i+1)$th block adder generates correct partial sum bits. Therefore, to decrease the power consumption of the recovery circuit, a gated recovery circuit technique is employed to suspend block adders with correct results. For example, when the *Err_block* signals {0100} are produced by error detection circuit, it indicates that the carry-out bit of the second block adder is incorrect, and the partial sum bits of the third block adder are incorrect. Hence, the error recovery circuit corrects the incorrect partial sum bits of the third block adder, with the other block adders suspended to reduce power consumption.

The major difference between the recovery circuit in [11] and the proposed error recovery circuit is that the former is implemented using an extra prefix adder and thus has higher area overhead and power consumption.

E. Variable-Latency Carry Speculative Adder

As mentioned in Section II-A, a variable latency design can reduce circuit timing waste and average latency. Fig. 13 shows the proposed VLCSPA with error detection and recovery circuits. VLCSPA produces the result of the CSPA (i.e., SUM*) in a cycle. The *ER* signal indicates whether an errorhas occurred. If the results are accurate, the *ER* signal is 0 and the *VALID* signal is 1; the results calculated by the CSPA are thus correct and are used as the output. If the results are inaccurate, the *ER* signal is 1 and the *VALID* signal is 0. The input registers are disabled and no new input is latched.

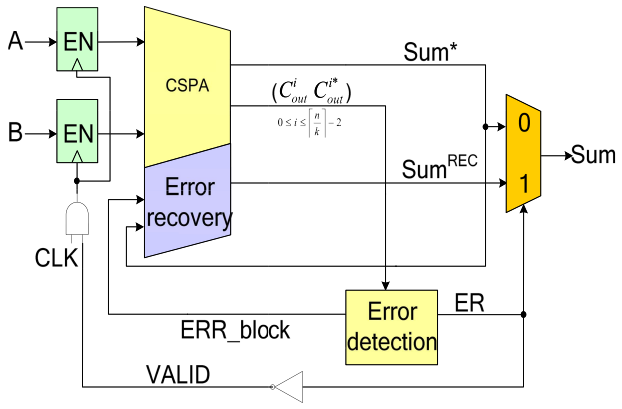

Fig. 13. Implementation of CSPA with variable latency.

The ERR_block signal generated by the error detection circuit indicates which block adder generated inaccurate results, and the accurate results, SUM^{REC} are calculated from the error recovery circuit. Note that since the error rate of the proposed CSPA is low, the average latency of the VLCSPA is close to that of the CSPA. Detailed experimental results are given in the next section.

IV. Experimental Setup and Results

This section shows experimental results of the CSPA and the SCSA-based adder for four adder widths. These adders were implemented in Veri log and synthesized to gate-level net lists using Synopsys Design Compiler with a standard TSMC 0.18-μm CMOS cell-library. The gate-level net list was converted to SPICE files using a commercial tool [21]. Then Synopsys Nanosim was used to analyze the delay and power of the adders. The error rate, accuracy, average latency and power consumption of the adders were analyzed and obtained by simulating 10 000 random inputs. Finally, an EEDP model is proposed to determine the appropriate block adder size for various adder widths.

In this paper, three kinds of SCSA-based adder were implemented. The first one is the original SCSA-based adder proposed in [11], called SCSAI. The second one is a modified SCSAI which only uses the input bits near the MSB of the corresponding block adder to predict the carry-out bit, called SCSAIM. The third one is SCSAII, which achieves a lower delay than that of SCSAI by reducing the block adder size. Two kinds of CSPA were implemented using the proposed architecture. The first one is CSPAI, whose block adder size is smaller than that of SCSAI. The second one is CSPAII, which has balanced delay, power consumption, and error rate The reason is detailed in Section IV-E.

The block adder size and number of block adders of the SCSA-based adders and the CSPAs are shown in Table II.

A. Error Rate and Accuracy Analysis for Various Input Widths in the Predictor Circuit

As mentioned in Section III, the carry-out prediction accuracy depends on the input width used in the carry predictor circuit, which is used to predict the carry-out bit of the block adder. If the block adder size is increased, the input width of the carry predictor circuit needs to increase as well so that the carry-out prediction accuracy does not decrease significantly.

TABLE II
Block Adder Size of SCSA-Based Adders and CSPAs

	Block adder size (number of block adders)			
Adder width	32	64	128	256
SCSAI	13 (3)	14 (5)	15 (9)	16 (16)
SCSAIM	13 (3)	14 (5)	15 (9)	16 (16)
SCSAII	9 (4)	10 (7)	11 (12)	12 (22)
CSPAI	12 (3)	13 (5)	14 (10)	15 (18)
CSPAII	11 (3)	13 (5)	13 (10)	16 (16)

TABLE III
Error Rate Comparison Between SCSA-Based Adders and CSPAs

		Adder width			
		32	64	128	256
SCSAI		0.02%	0.02%	0.01%	0.02%
SCSAIM	8-bit	0.92%	1.80%	2.45%	5.09%
	9-bit	0.46%	0.68%	1.25%	2.54%
	10-bit	0.21%	0.35%	0.72%	1.12%
	11-bit	0.10%	0.16%	0.34%	0.58%
SCSAII		1.05%	1.01%	1.04%	0.83%
CSPAI	8-bit	0.80%	1.20%	3.20%	5.94%
	9-bit	0.43%	0.61%	1.43%	2.87%
	10-bit	0.18%	0.34%	0.79%	1.62%
	11-bit	0.04%	0.16%	0.40%	0.69%
CSPAII	8-bit	0.77%	1.20%	2.47%	5.09%
	9-bit	0.38%	0.61%	1.27%	2.54%
	10-bit	0.04%	0.34%	0.73%	1.12%
	11-bit	0.02%	0.16%	0.35%	0.58%

Table III compares the error rate of 32-, 64-, 128-, and 256-bit speculative adders and four carry predictor circuits used in SCSAIM, CSPAI, and CSPAII. The carry predictor circuit input widths of SCSAI and SCSAII are the same as the block adder input width. Compared to CSPAI, CSPAII has a lower error rate in all cases. This is because the numbers of block adders are the same for each adder width and CSPAI has a larger block adder size than that of CSPAII. Therefore, the error rate of CSPAI is higher than that of CSPAII for a given carry predicator circuit. Although the error rates of CSPAI and CSPAII are higher than that of SCSAI, the difference is less than 0.8% in the worst case, which means that the increased error rate of the proposed CSPA is minimal.

Compared to SCSAIM, CSPAI and CSPAII can achieve lower error rates when the adder width is small. However, when the adder width is larger than 128, the error rate of CSPAI is higher than that of SCSAIM since the number of block adders used in CSPAI is more than that of SCSAIM. Compared to SCSAII, CSPAI and CSPAII have better error rates when the carry predictor circuit considers enough input bits to predict the carry-out bit.

Fig. 14(a) shows a comparison of ACC_{amp} values between SCSA-based adders and CSPAs. SCSAI has the highest ACC_{amp} for all adder widths since its error rate is low and few input patterns may cause errors. In contrast, SCSAIM has

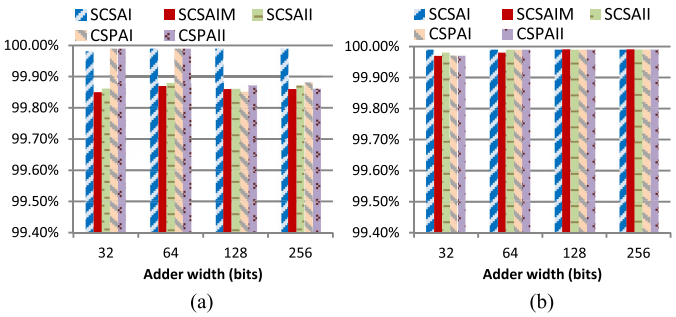

Fig. 14. (a) ACC_{amp} comparison. (b) ACC_{inf} comparison.

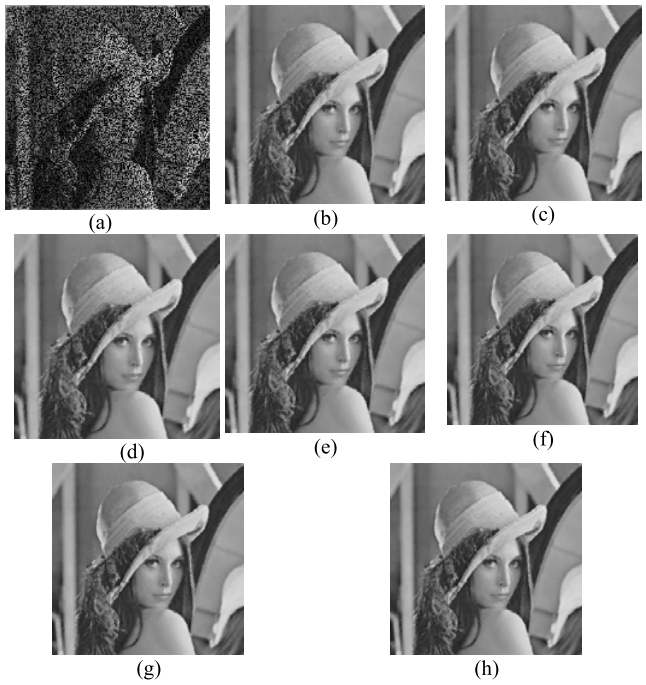

Fig. 15. Image denoising. (a) Original image with noise. (b) Accurate adder, PSNR: 28.17 dB. (c) SCSAI, PSNR: 28.09 dB. (d) SCSAIM, PSNR: 27.93 dB. (e) SCSAII, PSNR: 27.97 dB. (f) CSPAI, PSNR: 28.03 dB. (g) CSPAII, PSNR: 28.05 dB. (h) VLCSPA, PSNR: 28.17 db.

the lowest ACC_{amp} for all adder widths since its error rate is high, and it has a large block adder size. Although the error rate of SCSAII is higher than that of SCSAIM the ACC_{amp} of SCSAII is better than that of SCSAIM because SCSAII has a smaller block adder size. Compared to SCSAIM and SCSAII, both CSPAI and CSPAII have better ACC_{amp} values when the adder width is smaller than 64. However, when the adder width is increased, the ACC_{amp} values of CSPAI and CSPAII decrease because the block adder size increases. Fig. 14(b) shows ACC_{inf} values of the SCSA-based adders and CSPAs. The ACC_{inf} values of these adders are close to 99.99%, and thus their error significance is low.

These speculative adders were applied to image denoising. The results are shown in Fig. 15. The accurate adder and SCSAI have PSNR values of 28.17 and 28.09 dB, respectively. However, these two adders have higher delay, area and power consumption than those of the proposed CSPA. Thus, image processing with the proposed CSPA provides benefits with only a small loss in image quality.

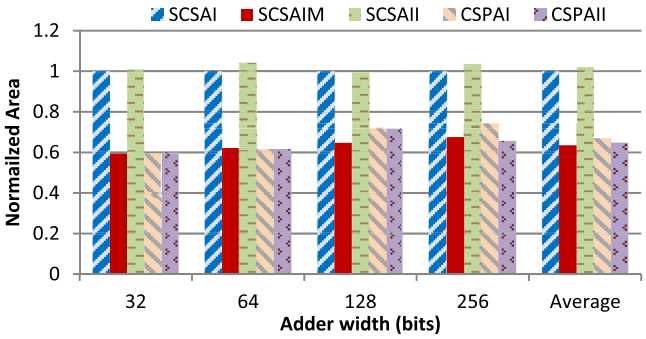

Fig. 16. Predictor circuit area comparison.

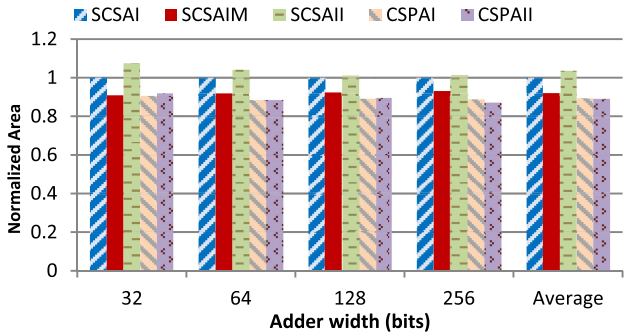

Fig. 17. Area comparison of SCSA-based adders and CSPAs.

B. Area Comparisons

Fig. 16 compares the area of the carry predictor circuit used in the SCSA-based adders and CSPAs. The carry predictor circuit areas of CSPAI and CSPAII are more than 33% lower than those of SCSAI and SCSAII. This is because SCSAI and SCSAII use all input bits in a block adder to predict the carry-out bit, but CSPAI and CSPAII only use the input bits near the MSB. Compared to SCSAIM, the carry predictor circuit areas of CSPAI and CSPAII area little higher (by 3.34% and 1.15%, respectively). This is because the numbers of block adders used in CSPAI and CSPAII are higher than that of SCSAIM, as shown in Table II. Therefore, more carry predictor circuits are needed.

Fig. 17 compares the normalized total area of the SCSA-based adders and CSPAs. It can be seen that although the carry predictor circuit areas of CSPAI and CSPAII are larger than that of SCSAIM, the total areas of CSPAI and CSPAII are smaller than that of the SCSA-based adders. This is because CSPAI and CSPAII only use one sum generator circuit to produce the partial sum bits in each block adder, and the carry predictor circuit only uses partial input bits to predict the carry-out bit. Compared to SCSAI, SCSAIM and SCSAII, the average area reductions of CSPAI (CPSAII) are 10.99% (11%), 3.24% (3.24%) and 14.03% (14.06%), respectively.

Fig. 18 compares the areas of Kogge–Stone prefix adder (KS), VLSCSAI, VLSCSAIM, VLSCSAII, VLCSPAI, and VLCSPAII. The area is estimated using the number of the transistors. It can be seen that the area of KS is smaller than thoseof the variable latency adders when the adder width is less than or equal to 64. This is because the variable latency

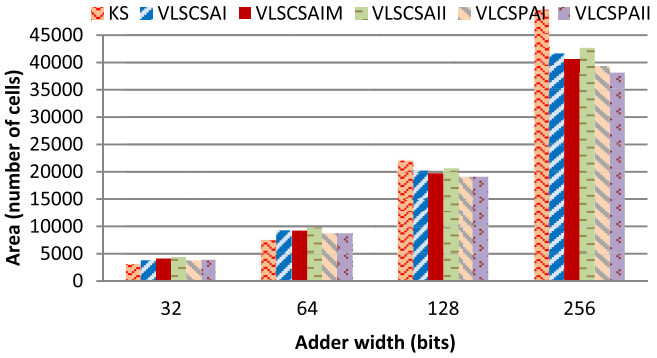

Fig. 18. Area comparison of KS, VLSCSAs, and VLCSPAs.

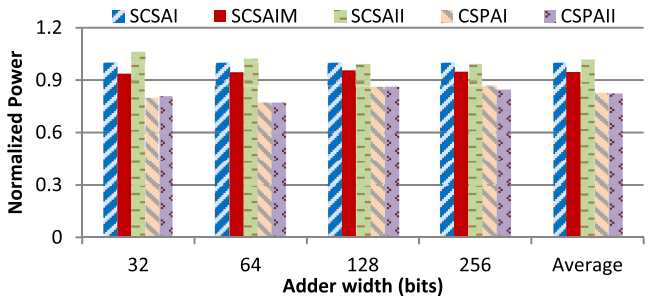

Fig. 19. Power consumption comparison of SCSA-based adders and CSPAs.

adders add extra error detection and recovery circuits to ensure correct results. However, the number of transistors in KS is larger than those for VLCSPAI and VLCSPAII when the adder width increases because the circuit complexity of KS increases at a rate faster than those of VLCSPAI and VLCSPAII. Compared to VLSCSAI, VLSCSAIM, and VLSCSAII, the average area reductions of VLCSPAI (VLCSPAII) are 5.04% (5.13%), 5.09% (5.18%), and 11.76% (11.86%), respectively, mainly because less area is needed in the CSPAs.

C. Power Consumption Comparisons

Fig. 19 compares the normalized power consumption between the SCSA-based adders and CSPAs. The power consumptions of CSPAI and CSPAII are smaller than that of the SCSA-based adders because CSPAI and CSPAII use only one sum generator in the block adder and fewer transistors in the carry predictor circuit. Compared to SCSAI, SCSAIM and SCSAII, CSPAI (CSPAII) has 17.53% (17.85%), 12.86% (13.19%), and 18.70% (19.03%) average power reductions, respectively. Note that the power consumption of CSPAI is higher than that of CSPAII when the adder width is 256 because more block adders and transistors are used in CSPAI, as shown in Table II.

Fig. 20(a) compares the normalized power consumptions of KS, VLSCSAI, VLSCSAIM, VLSCSAII, VLCSPAI, and VLCSPAII. It can be seen that the average power consumptions of VLCSPAI and VLCSPAII are higher than that of KS when the adder width is 32 and 64 and lower for higher adder widths. This is because the number of transistors in KS increases much higher when the adder width increases (as shown in Fig. 17). Compared to VLSCSAI, VLSCSAIM, and VLSCSAII,

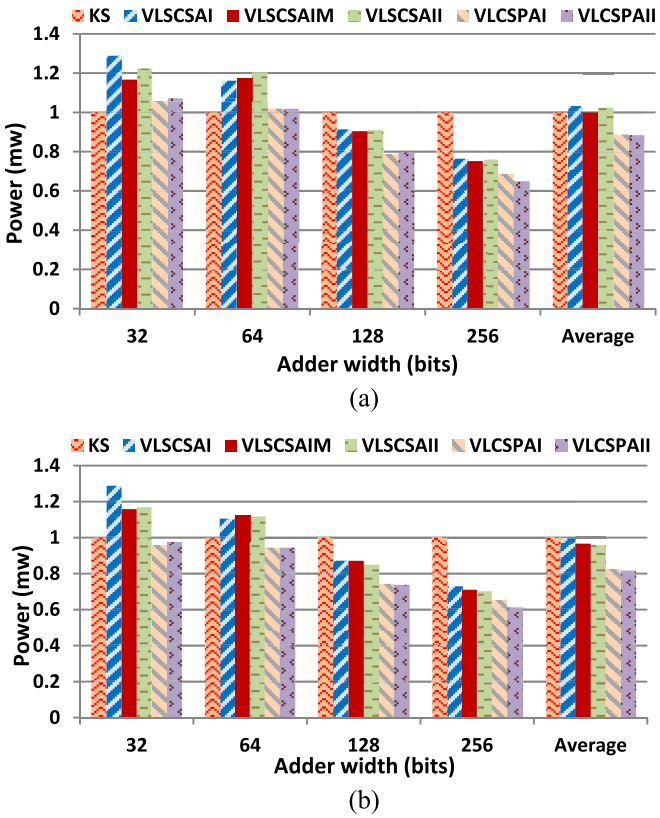

Fig. 20. (a) Power comparison of KS, VLSCSAs, and VLCSPAs. (b) Power comparison of KS, VLSCSAs and VLCSPAs with gated recovery circuit.

VLCSPAI (VLCSPAII) has 13.56% (14.14%), 11.15% (11.72%), and 12.90% (13.48%) average power reductions.

As mentioned in Section III-D, the power consumption of variable latency adders can be reduced if the inputs of the error recovery are gated when not in use. Fig. 20(b) compares the normalized power consumptions of KS and the variable latency adders with a gated recovery circuit. It can be seen that the average power consumptions of VLCSPAI and VLCSPAII are 8.41% and 9.36% lower than that of KS, respectively. Compared to VLSCSAI, VLSCSAIM, and VLSCSAII, VLCSPAI (VLCSPAII) has 1443% (1467%), 1125% (11.49%), and 13.52% (1376%) average power reductions respectively. From Figs. 19 and 20, the proposed speculative adders have the lowest average power consumption, with a gated recovery circuit leading to even more power reduction.

D. Critical Path Delay and Average Latency Comparisons

Fig. 21 compares the critical path delays of the SCSA-based adders and CSPAs. SCSAI and SCSAIM have the same critical path delays because the same block adder sizes are used in their circuits. SCSAII has a lower critical path delay than those of SCSAI and SCSAIM because the block adder size is reduced. It can be seen that CSPAI and CSPAII have critical path delays similar to that of SCSAII and much lower than those of SCSAI and SCSAIM. This is because the proposed CSPA separates the carry generator and sum generator, and thus the carry signals and final sum bits can be calculated faster to reduce the critical path delay.

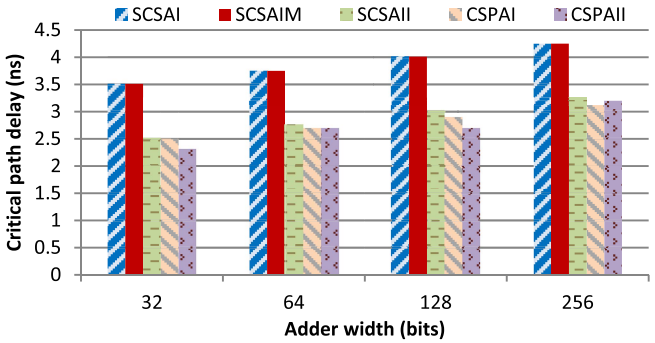

Fig. 21. Critical path delay comparison of SCSA-based adders and CSPAs.

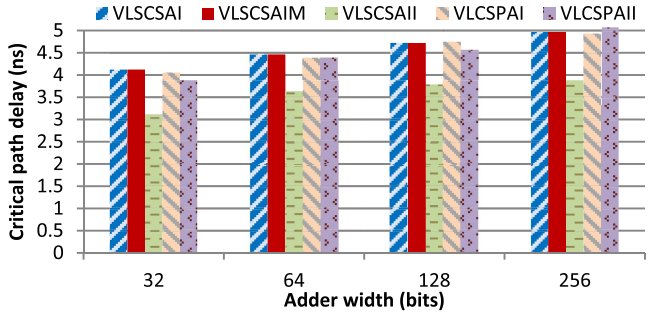

Fig. 22. Critical path delay comparisons of VLSCSAs and VLCSPAs.

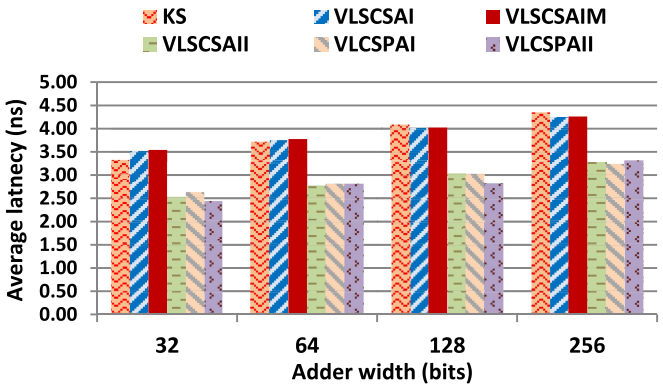

Fig. 23. Average latency comparison of KS, VLSCSAs, and VLCSPAs.

Fig. 22 compares the critical path delay of the variable latency adders. Compared to Fig. 21, since extra error detection and recovery circuits are needed to ensure the correct results the critical path delay of the variable latency adders is longer than that of the speculative adders. The critical path delays of VLCSPAI and VLCSPAII are similar to those of VLSCSAI and VLSCSAII. The critical path delay of VLSCSAII is lower than those of VLCSPAI and VLCSPAII because the block adder size in VLSCSAII is smaller. However, since VLSCSAII has a higher error rate (as shown in Table III), it is necessary to compare the average latency.

Fig. 23 compares the average latencies of KS, VLSCSAI, VLSCSAIM, VLSCSAII, VLCSPAI, and VLCSPAII. The average latencies of VLCSPAI and VLCSPAII are much lower than those of KS, VLSCSAI, and VLSCSAIM. Compared to KS, VLSCSAI and VLSCSAIM, the average delay reductions

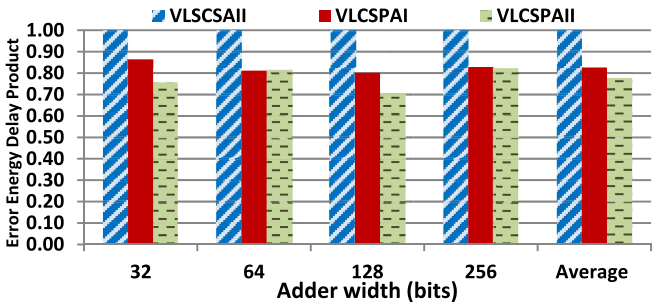

Fig. 24. EEDP comparison of VLSCSAII, VLCSPAI, and VLCSPAII.

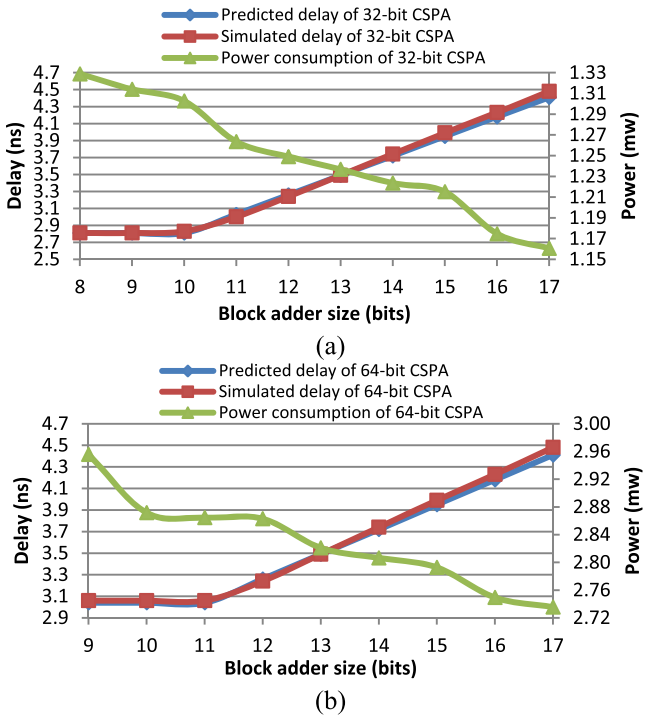

Fig. 25. Comparison of estimated and simulated delays for the (a) 32- and (b) 64-bit CSPAs.

of VLCSPAI are 24.34%, 24.76%, and 25.16% and those of VLCSPAII are 26.51%, 26.87%, and 27.26%, respectively. Compared to VLSCSAII, the average latency reductions of VLCSPAI and VLCSPAII are only 0.08% and 2.91%, respectively. Although the critical path delay of VLSCSAII is lower than those of VLCSPAI and VLCSPAII, VLSCSAII has a higher error rate. Therefore, the average latencies of VLCSPAI and VLCSPAII are similar to that of VLSCSAII.

In addition, Fig. 20 shows that VLSCSAII has higher power consumption than those of VLCSPAI and VLCSPAII. Fig. 24 compares the normalized EEDPs of VLSCSAII, VLCSPAI and VLCSPAII. The EEDPs of VLCSPAI and VLCSPAII are lower than that of VLSCSAII. The average EEDP reductions of VLCSPAI and VLCSPAII are 17.36% and 22.67%, respectively.

E. Error Energy-Delay Product Model

Fig. 25(a) and (b) shows the relationship between the critical path delay and power consumption for 32- and 64-bit CSPAs.

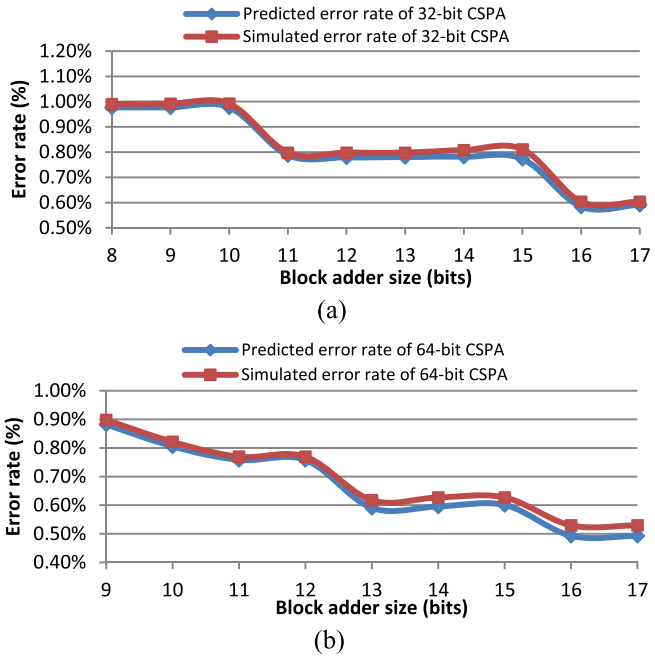

Fig. 26. Comparison of estimated and simulated error rates for (a) 32- and (b) 64-bit CSPAs.

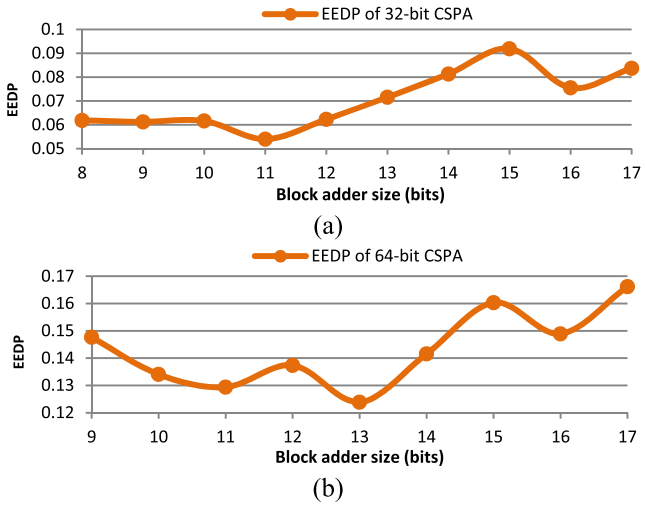

Fig. 27. EEDP values of (a) 32- and (b) 64-bit CSPAs.

Two methods were used to obtain the critical path delay. In the first method (10) and (11) were used to predict the critical path delay (blue line). In the second method Nanosim was used to obtain the simulated delay (red line). It can be seen that the predicted delay is close to the simulated delay, which means that using (10) and (11) is accurate for predicting delay. In addition, when the block adder size is increased, the power consumption of the CSPA decreases. This is because when the block adder size is increased, the number of block adders in the CSPA is reduced, and fewer carry predictor circuits are used. However, since the carry propagation length is increased when the block adder size is increased, the CSPA has a longer critical path delay.

Fig. 26(a) and (b) shows the error rates of 32- and 64-bit CSPAs for various block adder sizes. Two methods were used to obtain the error rate. In the first method, (9) was used to predict the error rate (blue line). In the second method NC-Veri log was used to obtain the simulated error rate (red line). It can be seen that the predicted error rate is close to the simulated error rate, which means that using (9) is accurate for predicting the error rate. In addition, the error rate is reduced when the block size is increased. This is because the number of carry predictor circuits is reduced.

In summary, in the proposed CSPA, the delay power consumption, and error rate are affected by the block adder size. When the block adder size is increased, the error rate and power consumption are reduced, but the critical path delay is increased. When the block adder size is decreased, the error rate and power consumption are increased and the critical path delay is decreased. Hence, determining a suitable block adder size to provide balanced delay, power consumption, and error rate is important. This study proposes a metric called EEDP to determine the block adder size. The metric is defined as

$$\text{Error-Energy-Delay Product} = P_{\text{err}}^{*} * \text{EDP}$$

where p_{err}^{*} and EDP are the error rate and the energy delay product of the CSPA, respectively. A block adder size that can achieve the minimum EEDP is regarded as most balanced block size. Fig. 27 shows EEDP values for 32- and 64-bit CSPAs with various block adder sizes. The minimum EEDP is obtained for block adder sizes of 11 and 13 for 32- and 64-bit CSPAs, respectively. These sizes are thus the most balanced values. This method can also be used to determine block adder size for other speculative adders with various input widths.

V. Conclusion

This paper proposed a high-performance low-power CSPA. There are five main contributions in this paper. First, the CSPA separates the carry generator and sum generator, and thus the carry signal and partial sum bit can be calculated faster. Second, the proposed carry predictor circuit of the block adder only uses the input bits near the MSB to predict the carry-out bit. The hardware cost of the prediction circuit is reduced and the CSPA has minimal error rate increase. Third, an error model was proposed to analyze the relationship between the block adder size, the carry predictor circuit size and the error rate of the CSPA. Fourth, the EEDP model was proposed to determine the appropriate block adder size to segment the n-bit adder for balanced delay, power consumption, and error rate. Fifth, when errors occur, the proposed error detection circuit indicates which block adder produced an incorrect carry-out bit, and the error recovery circuit only focuses on recovering the block adders with incorrect partial sum bits. Therefore, the power consumption of VLCSPA can be reduced. The experimental results show that the CSPA can achieve a 26.59% delay improvement, a 14.06% average area reduction and a 19.03% average power consumption reduction compared to the corresponding values for the SCSA-based adder. Compared to VLSCSA, the delay, area and power

consumption improvements of VLCSPA are 27.26%, 10.31%, and 19.25%, respectively.

ACKNOWLEDGMENT

The authors would like to thank G.-W. Wang and Prof. P.-Y. Chen of the National Cheng Kung University for their help in the setup of the image denoising experiment.

REFERENCES

[1] I. Koren, *Computer Arithmetic Algorithms*. Natick, MA, USA: A K Peters, Ltd., 2002.

[2] V. Gupta, D. Mohapatra, S. P. Park, A. Raghunathan, and K. Roy, "IMPACT: IMPrecise adders for low-power approximate computing," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2011, pp. 409–414.

[3] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 32, no. 1, pp. 124–137, Jan. 2013.

[4] D. Shin and S. K. Gupta, "Approximate logic synthesis for error tolerant applications," in *Proc. Design Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2010, pp. 957–960.

[5] N. Zhu, W. L. Goh, and K. S. Yeo, "Ultra low-power high-speed flexible probabilistic adder for error-tolerant applications," in *Proc. Int. SoC Design Conf. (ISOCC)*, Nov. 2011, pp. 393–396.

[6] N. Zhu, W. L. Goh, and K. S. Yeo, "An enhanced low-power high speed adder for error-tolerant application," in *Proc. Int. Symp. Intell. Control (ISIC)*, Dec. 2009, pp. 69–72.

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

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

[9] A. K. Verma, P. Brisk, and P. Ienne, "Variable latency speculative addition: A new paradigm for arithmetic circuit design," in *Proc. Design, Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2008, pp. 1250–1255.

[10] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs," in *Proc. Design Autom. Conf. (DAC)*, 2012, pp. 820–825.

[11] K. Du, P. Varman, and K. Mohanram, "High performance reliable variable latency carry select addition," in *Proc. Design Autom. Test Eur. Conf. Exhibit. (DATE)*, Mar. 2012, pp. 1257–1262.

[12] K. Du, P. Varman, and K. Mohanram, "Static window addition: A new paradigm for the design of variable latency adders," in *Proc. Int. Conf. Comput. Design (ICCD)*, Oct. 2011, pp. 455–456.

[13] M. Olivieri, "Design of synchronous and asynchronous variable-latency pipelined multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 2, pp. 365–376, Apr. 2001.

[14] Y. Chen, H. Li, J. Li, and C.-K. Koh, "Variable-latency adder (VL-adder): New arithmetic circuit design practice to overcome NBTI," in *Proc. Int. Symp. Low Power Electron. Design (ISLPED)*, Aug. 2007, pp. 195–200.

[15] Y. Chen *et al.*, "Variable-latency adder (VL-adder) designs for low power and NBTI tolerance," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 11, pp. 1621–1624, Nov. 2010.

[16] N. V. Mujadiya, "Instruction scheduling on variable latency functional units of VLIW processors," in *Proc. Int. Symp. Electron. Syst. Design (ISED)*, 2011, pp. 307–312.

[17] Y.-H. Su, D.-C. Wang, S.-C. Chang, and M.-S. Malgorzata, "Performance optimization using variable-latency design style," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 19, no. 10, pp. 1874–1883, Oct. 2011.

[18] D. Baneres, J. Cortadella, and M. Kishinevsky, "Variable-latency design by function speculation," in *Proc. Design Autom. Test Eur. Conf. Exhibit. (DATE)*, Apr. 2009, pp. 1704–1709.

[19] Y. Liu, Y. Sun, Y. Zhu, and H. Yang, "Design methodology of variable latency adders with multistage function speculation," in *Proc. 11th Int. Symp. Qual. Electron. Design (ISQED)*, Mar. 2010, pp. 824–830.

[20] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, *Digital Integrated Circuits, A Design Perspective*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2003.

[21] Synopsys. *Laker Custom Design and Layout Tool*. [Online]. Available: http://www.springsoft.com/ch/products/custom-design-and-layout/laker3, accessed May 2013.

Ing-Chao Lin (M'09–SM'14) received the M.S. degree in computer science from the National Taiwan University, Taipei, Taiwan and the Ph.D. degree from the Department of Computer Science and Engineering, Pennsylvania State University, State College, PA, USA, in 2007.

He was with Real Intent, Inc., Sunnyvale, CA, USA, from 2007 to 2009. Since 2009, he has been with the Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan, where he is currently an Associate Professor. His current research interests include VLSI design and computer-aided design for nanoscale silicon, low-power reliable system design, and computer architecture.

Yi-Ming Yang received the B.S. degree in computer science from the National Taichung University of Education, Taichung, Taiwan, in 2011 and the M.S. degree in computer science and information engineering from the National Cheng Kung University, Tainan, Taiwan, in 2013.

He is currently with VIA Technologies Inc., Taipei, Taiwan. His current research interests include low-power design for VLSI.

Cheng-Chian Lin received the B.S. degree in computer science from the National Central University, Zhongli, Taiwan, in 2013. He is currently pursuing the M.S. degree in computer science and information engineering at the National Cheng Kung University, Tainan, Taiwan.

His current research interests include low-power design for VLSI.