International Journal of Circuit Theory and Applications

WILEY

ORIGINAL ARTICLE

VLPPLAs: Variable Latency Parallel Prefix Ling Adders

Yongqiang Zhang[1] | Yili Yuan[1] | Jie Han[2] | Xin Cheng[1] | Guangjun Xie[1]

[1]School of Microelectronics, Hefei University of Technology, Hefei, China | [2]Department of Electrical and Computer Engineering, University of Alberta, Edmonton, Alberta, Canada

Correspondence: Guangjun Xie (gjxie8005@hfut.edu.cn)

Received: 29 May 2025 | **Revised:** 30 June 2025 | **Accepted:** 21 October 2025

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 62404067, in part by the Fundamental Research Funds for the Central Universities under Grant JZ2025HGTB0231, and in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada under Grant RES0048688.

Keywords: energy | error rate | latency | Ling adder | parallel prefix adder

ABSTRACT

Adders are ubiquitous in computer systems. Parallel prefix adders (PPAs) provide a way to speed up the addition. To further increase the performance, variable latency parallel prefix Ling adders (VLPPLAs) are proposed based on Ling adders in this work, respectively, using Brent–Kung, Beaumont Smith, Knowles, Kogge–Stone, and Sklansky topologies. The parallel prefix processing stage is designed to generate correct sums for most input operands. The Ling carries of these operands propagate for no more than a predetermined maximum carry chain length. The results of the remaining cases are speculated and detected through an overall error detection signal (OEDS) and then may be corrected using exact circuits if it is asserted in the next clock cycle. The OEDS is divided into several block error detection signals (BEDSs), for which the error rates are computed to estimate the computing accuracy of the designed VLPPLAs. Simulation and experimental results indicate that the proposed VLPPLAs achieve reductions in error rates by up to 96.50%, and energy performance with respect to average latency by about 15.68% for 64-bit designs on average, respectively, compared with previous variable latency parallel prefix adders (VLPPAs).

1 | Introduction

Addition is of importance in computer arithmetic, which dominates the computation speed of systems. Techniques for high-speed adders have been put forward, such as carry select adders (CSAs), carry look-ahead adders (CLAs), and parallel prefix adders (PPAs). In particular, adders using parallel prefix processing topologies, such as Brent–Kung [1], Beaumont Smith [2], Knowles [3], Kogge–Stone [4], Sklansky [5], and Ladner Fisher [6], can realize various performance enhancements concerning latency, area, fan-out, fan-in, logic depth, and wiring complexity. However, the latency of exact n-bit PPAs designed using these topologies is still limited by $O(\log(n))$ [7].

Adders for applications with resilience to errors can be loosely designed to speed up the addition. The latency of adders designed in such an approximate way is about sublogarithmic by sacrificing computing accuracy. In this case, inexact adders aimed at exploiting the tradeoff between computing accuracy and hardware costs, especially latency, have been extensively investigated at transistor, gate, logic, and algorithm levels, such as imprecise adders and carbon nanotube field-effect transistor-based adders [8, 9], lower-part OR adders (LOAs) and reconfigurable approximate carry look-ahead (RAP-CLA) adders [10–12], generic accuracy configurable (GeAr) adders, dual subadders based low latency adders (DSLLAs), approximate parallel prefix adders (AxPPAs), and block-based carry speculative approximate adders [13–16], and multiobjective Cartesian genetic programming (CGP) based adders [17]. If the generated results significantly deviate from the exact ones, an error correction module can be embedded into these inexact adders to alleviate such issues [14, 16].

Besides, variable latency adders (VLAs) show advantages in balancing computing accuracy and latency [18]. Previous works have shown that the average latency of an exact n-bit adder is $\log(n)$ for uniformly distributed unsigned input operands [18]. That is, each output generally depends on only previous $\log(n)$ bits, whereas unsigned inputs with a latency longer than $\log(n)$ are rare cases. Thus, the ideas behind VLAs are to generate correct results through approximate circuits for most input operands with a carry chain shorter than a predefined maximum length in one clock cycle. The results for other cases are speculated, where errors may occur. These results are detected using an overall error detection signal (OEDS) and can then be corrected using exact circuits if the OEDS is asserted in the next clock cycle. This concept has been applied to adders using various parallel prefix processing topologies [19–23], multipliers [24], and even dividers [25].

In [26], a 32-bit VLA based on a Ling adder detects the erroneously speculated results using a derived OEDS for it, as in other designs [19–23]. Experimental results show that the VLA based on a Ling adder surpasses conventional VLAs in latency since it achieves a smaller average latency than previous designs. However, the logic expression of the OEDS in [26] is redundant, and it corrects too many results, containing some correctly speculated ones. Thus, this VLA incurs large error rates. This work is a thorough extension and optimization of the design in [26] using different parallel prefix processing topologies, named variable latency parallel prefix Ling adder (VLPPLAs). A 64-bit VLPPLA using Brent–Kung topology is realized by removing specific rows from the original structure. It is analyzed in detail to derive an OEDS to check the speculated results. The OEDS is divided into several block error detection signals (BEDSs) to compute the error rate of each block and estimate the computing accuracy of the VLPPLA using independently and uniformly distributed unsigned operands.

The main contributions of this work are summarized as follows. (1) The general expression for the Ling carry is derived through the defined intermediate signals. (2) The OEDSs are derived from the truth tables of the proposed VLPPLAs using Brent–Kung, Beaumont Smith, Knowles, Kogge–Stone, and Sklansky topologies to check the correctness of speculated results. (3) A general architecture for the VLPPLAs is designed. (4) Various 64-bit VLPPLAs with different maximum carry chain lengths using these topologies are presented to validate the proposed design and analysis method.

This paper proceeds as follows. Section 2 introduces conventional PPAs, variable latency parallel prefix adders (VLPPAs), and Ling adders. Section 3 illustrates the proposed VLPPLAs. Section 4 provides the experimental results, including error rate and hardware costs. Section 5 concludes this work.

2 | Background

2.1 | PPAs

The addition of two n-bit unsigned binary operands $A = a_{n-1}a_{n-2}\cdots a_1a_0$ and $B = b_{n-1}b_{n-2}\cdots b_1b_0$ generates an n-bit sum $S = s_{n-1}s_{n-2}\cdots s_1s_0$ and a 1-bit carry-out c_{n-1} in a final result $c_{n-1}s_{n-1}s_{n-2}\cdots s_1s_0$. A PPA computes in three stages: preprocessing, parallel prefix processing, and postprocessing [7].

In the preprocessing stage, the generate g_i and propagate p_i for each bit are defined as

$$\begin{aligned} g_i &= a_i \cdot b_i \\ p_i &= a_i \oplus b_i \end{aligned} \tag{1}$$

where $0 \le i \le n-1$, the symbols $\cdot$ and $\oplus$ respectively denote the logic AND and XOR, and the symbol $\cdot$ is omitted later for convenience. Thus, the sum s_i and carry-out c_i for each bit are computed using g_i and p_i as

$$\begin{aligned} s_i &= p_i \oplus c_{i-1} \\ c_i &= g_i + p_i c_{i-1} \end{aligned} \tag{2}$$

where the symbol + denotes the logic OR and c_{-1} is the carry-in of an adder. So, the carry-out c_i can be iteratively expanded as

$$\begin{aligned} c_i = {} & g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + \cdots + p_i p_{i-1} \cdots p_1 g_0 \\ & + p_i p_{i-1} \cdots p_1 p_0 c_{-1} \end{aligned} \tag{3}$$

By modifying the definition of g_0 from x_0y_0 to $x_0y_0 + (x_0 \oplus y_0)c_{-1}$ or assuming $c_{-1} = 0$, the carry-out c_i for each bit becomes

$$c_i = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + \cdots + p_i p_{i-1} \cdots p_1 g_0 \tag{4}$$

For brevity, in the parallel prefix processing stage, the block generate $G_{[i:j]}$ and block propagate $P_{[i:j]}$, from the jth bit to the ith bit, are respectively defined as

$$\begin{aligned} G_{[i:j]} &= \begin{cases} g_i & i = j \\ G_{[i:m]} + P_{[i:m]} G_{[m-1:j]} & i \ge m > j \end{cases} \\ P_{[i:j]} &= \begin{cases} p_i & i = j \\ P_{[i:m]} P_{[m-1:j]} & i \ge m > j \end{cases} \end{aligned} \tag{5}$$

If $j = 0$, $G_{[i:0]}$ can be written as

$$G_{[i:0]} = g_i + p_i g_{i-1} + p_i p_{i-1} g_{i-2} + \cdots + p_i p_{i-1} \cdots p_i g_0 \tag{6}$$

According to (4) and (6), one has

$$c_i = G_{[i:0]} \tag{7}$$

The computation of $G_{[i:0]}$ is realized in the parallel prefix processing stage, where a Boolean operator° is introduced to compute the carry-outs as

$$(G, P) \circ \left(\widetilde{G}, \widetilde{P}\right) = \left(G + P\widetilde{G}, P\widetilde{P}\right) \tag{8}$$

The operator is associative and idempotent, so

$$\left(c_i, \sim\right) = \left(G_{[i:0]}, \sim\right) = \left(g_i, p_i\right) \circ \left(g_{i-1}, p_{i-1}\right) \circ \cdots \circ \left(g_1, p_1\right) \circ \left(g_0, p_0\right) \tag{9}$$

In the postprocessing stage, the sum s_i for each bit and carry-out c_{n-1} are respectively computed as

$$\begin{aligned} s_i &= p_i \oplus G_{i-1} \\ c_{n-1} &= G_{[n-1:0]} \end{aligned} \tag{10}$$

The key to a PPA is the parallel prefix processing stage, which computes the block generate $G_{[i:j]}$ and block propagate $P_{[i:j]}$. Since the associativity of the operator in (8), different combinations

of the generate and propagate pairs bring forth various parallel prefix processing topologies, such as Brent–Kung [1], Beaumont Smith [2], Knowles [3], Kogge–Stone [4], Sklansky [5], and Ladner Fisher [6]. Ladner Fisher topology searches for a design for PPAs: The one with a minimal logic depth shares the same topology with Sklansky. All these PPAs are designed to balance the logic level, latency, area, fan-in, fan-out, and overall wiring.

Figure 1a shows the parallel prefix processing stage of a 64-bit PPA using Brent–Kung topology (PPA_{BK64}), where the white squares in the top row are used to produce the generate g_i and propagate p_i according to (1), the black nodes represent the associative operators, the white nodes are buffers, and the black connection means that the result of an operator is put into the next one in the next row. It has a total of nine rows to compute the carry-outs.

2.2 | VLPPAs

In [21, 23], VLPPAs using parallel prefix processing topologies are studied to save the addition time for most input operands. The operands with a carry chain length shorter than a predefined maximum length are accurately computed, whereas the carry-outs of other operands are speculated, detected, and corrected if the OEDS is true. The maximum length can be realized by removing specific rows in the original structure of a PPA. For example, Figure 1b shows the speculative parallel prefix processing stage of a 64-bit VLPPA using the Brent–Kung topology ($VLPPA_{BK64\_7}$) by deleting seven rows in the red-dotted rectangle in Figure 1a to realize the maximum carry chain length of $L=7$. For example, the computation of c_i ($i=6, 10, 14, 18, 22, 26, 30, 34, 38, 42, 46, 50, 54, 58$, and 62) from g_j ($j=0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56$, and 60) involves 7 bits, respectively. Thus, the carry-outs of the less significant 7 bits ($c_0 \sim c_6$) are accurately computed, but they are speculated for the most significant 57 bits ($c_7 \sim c_{63}$). The speculated carry-out c_i^* for each bit of the $VLPPA_{BK64\_7}$ is

$$c_i^* = \begin{cases} G_{[i:0]} & 0 \le i \le 6 \\ G_{[i:4]} & 7 \le i \le 10 \\ G_{[i:8]} & 11 \le i \le 14 \\ G_{[i:12]} & 15 \le i \le 18 \\ G_{[i:16]} & 19 \le i \le 22 \\ G_{[i:20]} & 23 \le i \le 26 \\ G_{[i:24]} & 27 \le i \le 30 \\ G_{[i:28]} & 31 \le i \le 34 \\ G_{[i:32]} & 35 \le i \le 38 \\ G_{[i:36]} & 39 \le i \le 42 \\ G_{[i:40]} & 43 \le i \le 46 \\ G_{[i:44]} & 47 \le i \le 50 \\ G_{[i:48]} & 51 \le i \le 54 \\ G_{[i:52]} & 55 \le i \le 58 \\ G_{[i:56]} & 59 \le i \le 62 \\ G_{[i:60]} & i = 63 \end{cases} \tag{11}$$

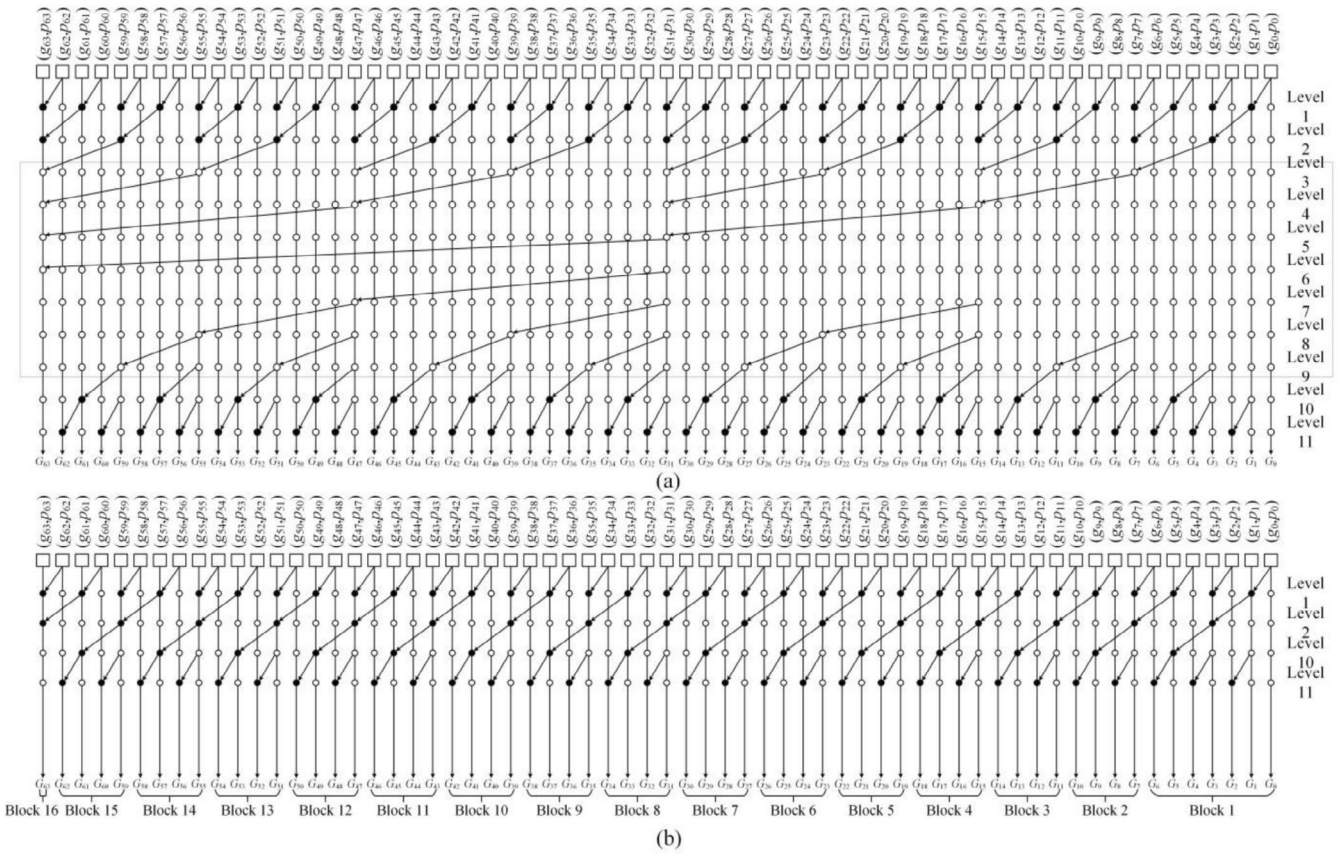


FIGURE 1 | The parallel prefix processing stage of 64-bit adders using Brent–Kung topology. (a) The original structure. (b) The speculative parallel prefix processing stage by deleting seven rows in the red-dotted rectangle.

With these 16 divided blocks, the accurately computed carry-outs $c_i = G_{[i:0]}$ $(0 \le i \le 63)$ in (7) can be written as

$$c_i^* = \begin{cases} G_{[i:0]} & 0 \le i \le 6 \\ G_{[i:4]} + P_{[i:4]} G_{[3:0]} & 7 \le i \le 10 \\ G_{[i:8]} + P_{[i:8]} G_{[7:0]} & 11 \le i \le 14 \\ G_{[i:12]} + P_{[i:12]} G_{[11:0]} & 15 \le i \le 18 \\ G_{[i:16]} + P_{[i:16]} G_{[15:0]} & 19 \le i \le 22 \\ G_{[i:20]} + P_{[i:20]} G_{[19:0]} & 23 \le i \le 26 \\ G_{[i:24]} + P_{[i:24]} G_{[23:0]} & 27 \le i \le 30 \\ G_{[i:28]} + P_{[i:28]} G_{[27:0]} & 31 \le i \le 34 \\ G_{[i:32]} + P_{[i:32]} G_{[31:0]} & 35 \le i \le 38 \\ G_{[i:36]} + P_{[i:36]} G_{[35:0]} & 39 \le i \le 42 \\ G_{[i:40]} + P_{[i:40]} G_{[39:0]} & 43 \le i \le 46 \\ G_{[i:44]} + P_{[i:44]} G_{[43:0]} & 47 \le i \le 50 \\ G_{[i:48]} + P_{[i:48]} G_{[47:0]} & 51 \le i \le 54 \\ G_{[i:52]} + P_{[i:52]} G_{[51:0]} & 55 \le i \le 58 \\ G_{[i:56]} + P_{[i:56]} G_{[55:0]} & 59 \le i \le 62 \\ G_{[i:60]} + P_{[i:60]} G_{[59:0]} & i = 63 \end{cases} \tag{12}$$

where the block generate $G_{[i:j]} = 1$ and block propagate $P_{[i:j]} = 1$ are mutually exclusive. Comparing (11) with (12), the OEDS for the $\text{VLPPA}_{\text{BK64\_7}}$ is

$$\begin{aligned} E = & \bigcup_{i=63}^{63} P_{[i:60]} G_{[59:0]} + \bigcup_{i=59}^{62} P_{[i:56]} G_{[55:0]} + \bigcup_{i=55}^{58} P_{[i:52]} G_{[51:0]} \\ & + \bigcup_{i=51}^{54} P_{[i:48]} G_{[47:0]} + \bigcup_{i=47}^{50} P_{[i:44]} G_{[43:0]} + \bigcup_{i=43}^{46} P_{[i:40]} G_{[39:0]} \\ & + \bigcup_{i=39}^{42} P_{[i:36]} G_{[35:0]} + \bigcup_{i=35}^{38} P_{[i:32]} G_{[31:0]} + \bigcup_{i=31}^{34} P_{[i:28]} G_{[27:0]} \\ & + \bigcup_{i=27}^{30} P_{[i:24]} G_{[23:0]} + \bigcup_{i=23}^{26} P_{[i:20]} G_{[19:0]} + \bigcup_{i=19}^{22} P_{[i:16]} G_{[15:0]} \\ & + \bigcup_{i=15}^{18} P_{[i:12]} G_{[11:0]} + \bigcup_{i=11}^{14} P_{[i:8]} G_{[7:0]} + \bigcup_{i=7}^{10} P_{[i:4]} G_{[3:0]} \end{aligned} \tag{13}$$

where the symbol $\cup$ represents the logic OR. If the OEDS E is detected as a 1, it means that at least one of the speculated carry-outs is different from the exact one for the input operands.

Considering the last term in (13) and (5), we have

$$\begin{aligned} \bigcup_{i=7}^{10} P_{[i:4]} G_{[3:0]} &= \left(P_{[7:4]} + P_{[8:8]} P_{[7:4]} + P_{[9:8]} P_{[7:4]} + P_{[10:8]} P_{[7:4]} \right) G_{[3:0]} \\ &= P_{[7:4]} G_{[3:0]} \end{aligned} \tag{14}$$

Because $c_{10}^* = P_{[10:4]} G_{[3:0]}$, $c_9^* = P_{[9:4]} G_{[3:0]}$, $c_8^* = P_{[8:4]} G_{[3:0]}$, and $c_7^* = P_{[7:4]} G_{[3:0]}$ have the same parent node c_7^* as marked in green in Figure 1b, this simplification is obvious. The OEDS for the $\text{VLPPA}_{\text{BK64\_7}}$ is simplified as

$$\begin{aligned} E = & P_{[63:60]} G_{[59:0]} + P_{[59:56]} G_{[55:0]} + P_{[55:52]} G_{[51:0]} + P_{[51:48]} G_{[47:0]} \\ & + P_{[47:44]} G_{[43:0]} + P_{[43:40]} G_{[39:0]} + P_{[39:36]} G_{[35:0]} \\ & + P_{[35:32]} G_{[31:0]} + P_{[31:28]} G_{[27:0]} + P_{[27:24]} G_{[23:0]} \\ & + P_{[23:20]} G_{[19:0]} + P_{[19:16]} G_{[15:0]} + P_{[15:12]} G_{[11:0]} \\ & + P_{[11:8]} G_{[7:0]} + P_{[7:4]} G_{[3:0]} \end{aligned} \tag{15}$$

Furthermore, considering the last two terms in (15) and (5), we obtain

$$\begin{aligned} & P_{[11:8]} G_{[7:0]} + P_{[7:4]} G_{[3:0]} = P_{[11:8]} \left(G_{[7:4]} + P_{[7:4]} G_{[3:0]} \right) + P_{[7:4]} G_{[3:0]} \\ & = P_{[11:8]} G_{[7:4]} + P_{[7:4]} G_{[3:0]} \end{aligned} \tag{16}$$

Finally, the OEDS for the $\text{VLPPA}_{\text{BK64\_7}}$ can be written as

$$\begin{aligned} E = & P_{[63:60]} G_{[59:56]} + P_{[59:56]} G_{[55:52]} + P_{[55:52]} G_{[51:48]} + P_{[51:48]} G_{[47:44]} \\ & + P_{[47:44]} G_{[43:40]} + P_{[43:40]} G_{[39:36]} + P_{[39:36]} G_{[35:32]} + P_{[35:32]} G_{[31:28]} \\ & + P_{[31:28]} G_{[27:24]} + P_{[27:24]} G_{[23:20]} + P_{[23:20]} G_{[19:16]} + P_{[19:16]} G_{[15:12]} \\ & + P_{[15:12]} G_{[11:8]} + P_{[11:8]} G_{[7:4]} + P_{[7:4]} G_{[3:0]} \end{aligned} \tag{17}$$

If the OEDS is asserted, the erroneously speculated carry-outs are corrected by restoring the removed rows to the original structure in the next clock cycle [21, 23]. Thus, the average latency T_{avg} of a VLPPA is computed as

$$T_{avg} = \left(1 - P_{\text{OEDS}}\right) T_{clk} + P_{\text{OEDS}} 2 T_{clk} = \left(1 + P_{\text{OEDS}}\right) T_{clk} \tag{18}$$

where P_{OEDS} is the probability of an OEDS being asserted and T_{clk} is the clock period.

2.3 | Ling Adders

Different from the PPAs above, the generate g_i, propagate p_i, and half-sum d_i for each bit are respectively defined in a Ling adder [27] as

$$\begin{aligned} g_i &= a_i b_i \\ p_i &= a_i + b_i \\ d_i &= a_i \oplus b_i \end{aligned} \tag{19}$$

where $0 \le i \le n-1$. These definitions are critical to compute

$$g_i p_i = a_i b_i \left(a_i + b_i\right) = a_i b_i = g_i \tag{20}$$

In contrast to the PPAs, a Ling adder uses the Ling carry H_i instead of the carry-out c_i for each bit in the parallel prefix processing stage, which is defined as

$$H_i = c_i + c_{i-1} \tag{21}$$

According to (2) and (4), it can be expanded as

$$H_i = g_i + g_{i-1} + p_{i-1} g_{i-2} + p_{i-1} p_{i-2} g_{i-3} + \cdots + p_{i-1} p_{i-2} \cdots p_1 g_0 \tag{22}$$

Consider the first six terms $H_0 \sim H_5$ as

$$\begin{aligned} H_0 &= g_0 \\ H_1 &= g_1 + g_0 \\ H_2 &= g_2 + g_1 + p_1 g_0 \\ H_3 &= g_3 + g_2 + p_2 g_1 + p_2 p_1 g_0 \\ H_4 &= g_4 + g_3 + p_3 g_2 + p_3 p_2 g_1 + p_3 p_2 p_1 g_0 \\ H_5 &= g_5 + g_4 + p_4 g_3 + p_4 p_3 g_2 + p_4 p_3 p_2 g_1 + p_4 p_3 p_2 p_1 g_0 \end{aligned} \tag{23}$$

Since $g_i p_i = g_i$, (23) can be rewritten as

$$\begin{aligned} H_0 &= (g_0 + g_{-1}) \\ H_1 &= (g_1 + g_0) \\ H_2 &= (g_2 + g_1) + p_1 p_0 (g_0 + g_{-1}) \\ H_3 &= (g_3 + g_2) + p_2 p_1 (g_1 + g_0) \\ H_4 &= (g_4 + g_3) + p_3 p_2 (g_2 + g_1) + p_3 p_2 p_1 p_0 (g_0 + g_{-1}) \\ H_5 &= (g_5 + g_4) + p_4 p_3 (g_3 + g_2) + p_4 p_3 p_2 p_1 (g_1 + g_0) \end{aligned} \tag{24}$$

where $g_{-1} = 0$. Thus, the Ling carry H_i can be classified into even and odd bits as

$$\begin{aligned} H_{2k} &= (g_{2k} + g_{2k-1}) + p_{2k-1} p_{2k-2} (g_{2k-2} + g_{2k-3}) + \cdots \\ &+ p_{2k-1} p_{2k-2} \cdots p_1 p_0 (g_0 + g_{-1}) H_{2k+1} = (g_{2k+1} + g_{2k}) \\ &+ p_{2k} p_{2k-1} (g_{2k-1} + g_{2k-2}) + p_{2k} p_{2k-1} \cdots p_2 p_1 (g_1 + g_0) \end{aligned} \tag{25}$$

where $0 \le k \le n/2 - 1$. Define two intermediate signals α_i and β_i for each bit as

$$\begin{aligned} \alpha_i &= g_i + g_{i-1} \\ \beta_i &= p_i p_{i-1} \end{aligned} \tag{26}$$

where $p_{-1} = 0$. The Ling carries H_{2k} and H_{2k+1} for even and odd bits are respectively represented as

$$\begin{aligned} H_{2k} &= \alpha_{2k} + \beta_{2k-1}\alpha_{2k-2} + \beta_{2k-1}\beta_{2k-3}\alpha_{2k-4} + \cdots + \beta_{2k-1}\beta_{2k-3} \cdots \beta_1 \alpha_0 \\ H_{2k+1} &= \alpha_{2k+1} + \beta_{2k}\alpha_{2k-1} + \beta_{2k}\beta_{2k-2}\alpha_{2k-3} + \cdots + \beta_{2k}\beta_{2k-2} \cdots \beta_2 \alpha_1 \end{aligned} \tag{27}$$

Taking the associative operator [28] into consideration, (27) becomes

$$\begin{aligned} (H_{2k}, \sim) &= (\alpha_{2k}, \beta_{2k-1}) \circ (\alpha_{2k-2}, \beta_{2k-3}) \circ \cdots \circ (\alpha_0, \beta_{-1}) \\ (H_{2k+1}, \sim) &= (\alpha_{2k+1}, \beta_{2k}) \circ (\alpha_{2k-1}, \beta_{2k-2}) \circ \cdots \circ (\alpha_1, \beta_0) \end{aligned} \tag{28}$$

where $\beta_{-1} = 0$.

In the postprocessing stage of a Ling adder, the sum for each bit and carry-out c_{n-1} are respectively computed as

$$\begin{aligned} s_i &= d_i \oplus (p_{i-1} H_{i-1}) \\ c_{n-1} &= p_{n-1} H_{n-1} \end{aligned} \tag{29}$$

where $s_0 = d_0 \oplus c_{-1}$ is computed as a special case.

3 | The Proposed VLPPLAs

A VLPPLA operates in three stages, including preprocessing, parallel prefix processing, and postprocessing, with error detection and error correction modules [26].

3.1 | Preprocessing Stage

The preprocessing stage in a VLPPLA generates the intermediate signals α_i and β_i defined in (26).

3.2 | Speculative Parallel Prefix Processing Stage

In this subsection, different speculative parallel prefix processing stages in the VLPPLAs with various maximum carry chain lengths are respectively designed using Brent–Kung [1], Beaumont Smith [2], Knowles [3], Kogge–Stone [4], and Sklansky [5] topologies. The following definitions for the intermediate signals α_i and β_i are made to simplify the formulation as

$$\begin{aligned} &\alpha_{[2k:2(k-j)]} = \\ &\begin{cases} \alpha_{2k} & j = 0 \\ \alpha_{[2k:2(k-m)]} + \beta_{[2k-1:2(k-m)-1]} \alpha_{[2(k-m-1):2(k-j)]} & else \end{cases} \\ &\beta_{[2k:2(k-j)]} = \\ &\begin{cases} \beta_{2k} & j = 0 \\ \beta_{[2k:2(k-m)]} \beta_{[2(k-m-1):2(k-j)]} & else \end{cases} \\ &\alpha_{[2k+1:2(k-j)+1]} = \\ &\begin{cases} \alpha_{2k+1} & j = 0 \\ \alpha_{[2k+1:2(k-m)+1]} + \beta_{[2k:2(k-m)]} \alpha_{[2(k-m)-1:2(k-j)+1]} & else \end{cases} \\ &\beta_{[2k+1:2(k-j)+1]} = \\ &\begin{cases} \beta_{2k+1} & j = 0 \\ \beta_{[2k+1:2(k-m)+1]} \beta_{[2(k-m)-1:2(k-j)+1]} & else \end{cases} \end{aligned} \tag{30}$$

where $0 \le m < j \le k \le n/2 - 1$. The expressions for the Ling carries H_{2k} and H_{2k+1} for even and odd bits in (28) can be written as

$$\begin{aligned} H_{2k} &= \alpha_{[2k:0]} \\ H_{2k+1} &= \alpha_{[2k+1:1]} \end{aligned} \tag{31}$$

In what follows, we detail the parallel prefix Ling adders (PPLAs) and then derive the VLPPLAs from them. A 64-bit Ling adder using the Brent–Kung topology ($\text{PPLA}_{\text{BK64}}$) is presented in detail to illustrate the proposed design and analysis method. Figure 2a shows the parallel prefix processing stage of the $\text{PPLA}_{\text{BK64}}$, where the white squares in the top row are used to generate the intermediate signals α_i and β_i according to (26), the black nodes represent the associative operators [28], the white nodes are buffers, and the black connection means that the result of an operator is connected to the next one in the next row.

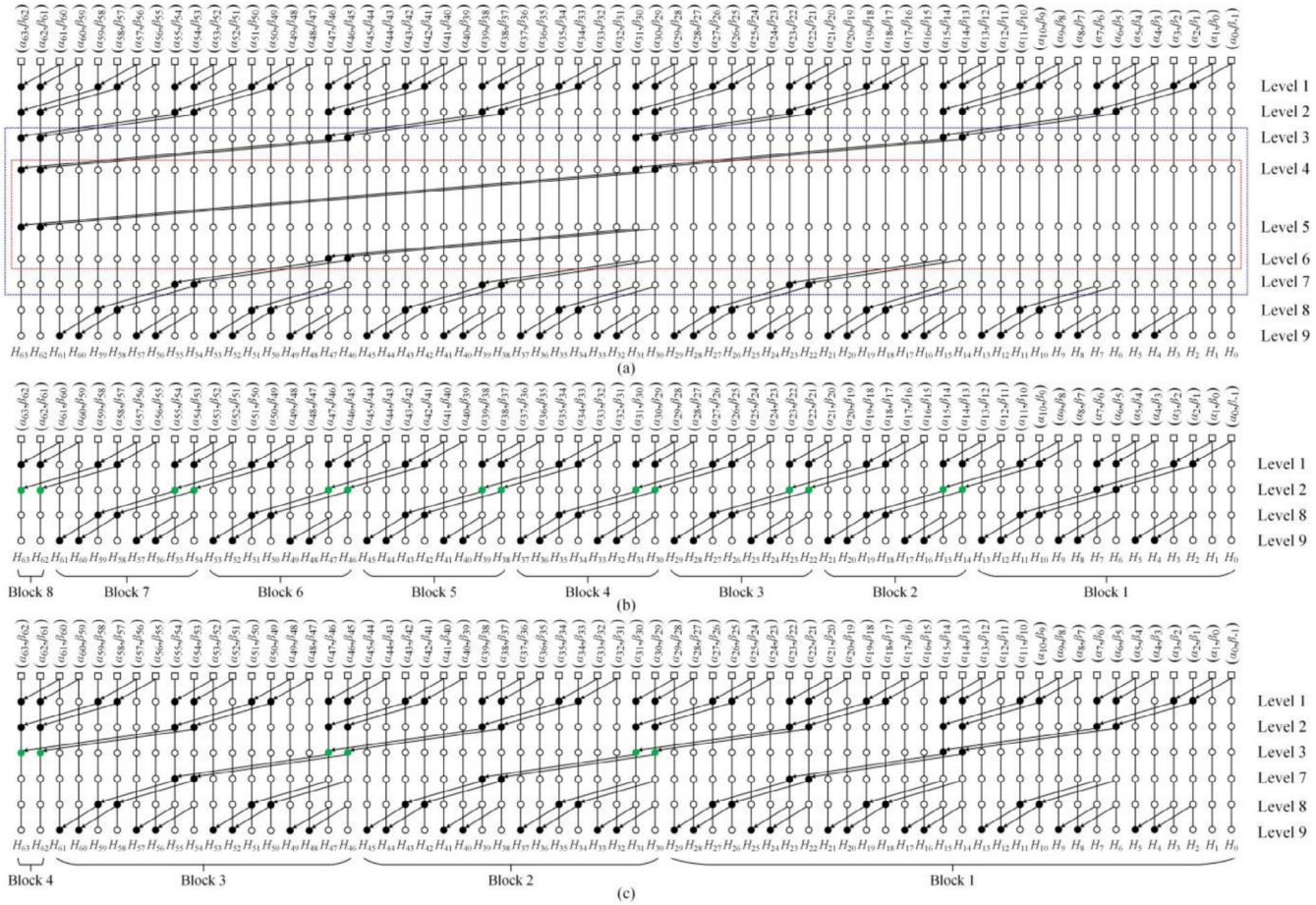


FIGURE 2 | The parallel prefix processing stages of 64-bit Ling adders using Brent–Kung topology. (a) The original structure. (b) The speculative parallel prefix processing stage with a maximum carry chain length of 7. (c) The speculative parallel prefix processing stage with a maximum carry chain length of 15.

The adder has a total of eight rows to compute the Ling carries, where the even and odd bits share the same connectivity.

As corroborated in 23, the maximum carry chain length for 64-bit unsigned inputs is mostly less than 7. The maximum carry chain length is therefore set to $L=7$ in the speculative parallel prefix processing stage of the 64-bit VLPPLA using Brent–Kung topology ($\text{VLPPLA}_{\text{BK64\_7}}$). The speculative parallel prefix processing stage can be constructed, as shown in Figure 2b, by removing five rows in the blue-dotted rectangle from the $\text{PPLA}_{\text{BK64}}$ in Figure 2a. This structure results in three rows to compute the Ling carries, where the less significant 14 bits ($H_0 \sim H_{13}$) are accurately generated and the most significant 50 bits ($H_{14} \sim H_{63}$) are speculated. It can be observed that the maximum carry chain length is 7, such as the carry chain lengths of $H_{12}, H_{13}, H_{20}, H_{21}, H_{28}$, and H_{29}. In addition, the speculative parallel prefix processing stage of the $\text{VLPPLA}_{\text{BK64\_7}}$ presents the same connectivity for even and odd bits, as that of the $\text{PPLA}_{\text{BK64}}$.

3.3 | Error Detection

According to (31), the speculated Ling carry H_i^* for each bit of the $\text{VLPPLA}_{\text{BK64\_7}}$ is computed as

$$H_{2k}^* = \begin{cases} \alpha_{[2k:0]} & 0 \le k \le 6 \\ \alpha_{[2k:8]} & 7 \le k \le 10 \\ \alpha_{[2k:16]} & 11 \le k \le 14 \\ \alpha_{[2k:24]} & 15 \le k \le 18 \\ \alpha_{[2k:32]} & 19 \le k \le 22 \\ \alpha_{[2k:40]} & 23 \le k \le 26 \\ \alpha_{[2k:48]} & 27 \le k \le 30 \\ \alpha_{[2k:56]} & k = 31 \end{cases} \quad H_{2k+1}^* = \begin{cases} \alpha_{[2k+1:1]} & 0 \le k \le 6 \\ \alpha_{[2k+1:9]} & 7 \le k \le 10 \\ \alpha_{[2k+1:17]} & 11 \le k \le 14 \\ \alpha_{[2k+1:25]} & 15 \le k \le 18 \\ \alpha_{[2k+1:33]} & 19 \le k \le 22 \\ \alpha_{[2k+1:41]} & 23 \le k \le 26 \\ \alpha_{[2k+1:49]} & 27 \le k \le 30 \\ \alpha_{[2k+1:57]} & k = 31 \end{cases} \tag{32}$$

It is shown that the speculative parallel prefix processing stage of the $\text{VLPPLA}_{\text{BK64\_7}}$ is divided into eight blocks for even and odd bits, including Block 1 (from H_0 to H_{13}), Block 2 (from H_{14} to H_{21}), Block 3 (from H_{22} to H_{29}), Block 4 (from H_{30} to H_{31}), and so on, respectively, as shown in Figure 2b. The Ling carries in Block 1 are accurately computed, while they are speculated in Blocks 2, 3, 4, 5, 6, 7, and 8. With these divided eight blocks, the exact Ling carry H_i for each bit of the

$\mathrm{PPLA}_{\mathrm{BK64}}$ in Figure 2a can be respectively written for even and odd bits as

$$H_{2k}=\begin{cases}\alpha_{[2k:0]} & 0\le k\le 6\\ \alpha_{[2k:8]}+\beta_{[2k-1:7]}\alpha_{[6:0]} & 7\le k\le 10\\ \alpha_{[2k:16]}+\beta_{[2k-1:15]}\alpha_{[14:0]} & 11\le k\le 14\\ \alpha_{[2k:24]}+\beta_{[2k-1:23]}\alpha_{[22:0]} & 15\le k\le 18\\ \alpha_{[2k:32]}+\beta_{[2k-1:31]}\alpha_{[30:0]} & 19\le k\le 22\\ \alpha_{[2k:40]}+\beta_{[2k-1:39]}\alpha_{[38:0]} & 23\le k\le 26\\ \alpha_{[2k:48]}+\beta_{[2k-1:47]}\alpha_{[46:0]} & 27\le k\le 30\\ \alpha_{[2k:56]}+\beta_{[2k-1:55]}\alpha_{[54:0]} & k=31\end{cases} \tag{33}$$

$$H_{2k+1}=\begin{cases}\alpha_{[2k+1:1]} & 0\le k\le 6\\ \alpha_{[2k+1:9]}+\beta_{[2k:8]}\alpha_{[7:1]} & 7\le k\le 10\\ \alpha_{[2k+1:17]}+\beta_{[2k:16]}\alpha_{[15:1]} & 11\le k\le 14\\ \alpha_{[2k+1:25]}+\beta_{[2k:24]}\alpha_{[23:1]} & 15\le k\le 18\\ \alpha_{[2k+1:33]}+\beta_{[2k:32]}\alpha_{[31:1]} & 19\le k\le 22\\ \alpha_{[2k+1:41]}+\beta_{[2k:40]}\alpha_{[39:1]} & 23\le k\le 26\\ \alpha_{[2k+1:49]}+\beta_{[2k:48]}\alpha_{[47:1]} & 27\le k\le 30\\ \alpha_{[2k+1:57]}+\beta_{[2k:56]}\alpha_{[55:1]} & k=31\end{cases}$$

By comparing (32) with (33), it is observed that the exact Ling carry H_i for each bit is truncated and speculated by the Ling carry H_i^*. Among them, the speculated Ling carries H_{2k}^* and H_{2k+1}^* $(0\le k\le 6)$ in Block 1 are always equal to H_{2k} and H_{2k+1}, respectively, since they are accurate. For Blocks 2, 3, 4, 5, 6, 7, and 8, an error may occur, for which the error detection is different from that of the VLPPAs simply using PG in (17).

TABLE 1 | The truth table of s_i^*, s_i, H_{i-1}^*, and H_{i-1} for the $\mathrm{VLPPLA}_{\mathrm{BK64\_7}}$

d_i	p_{i-1}	α	$\beta\alpha$	H_{i-1}^*	H_{i-1}	s_i^*	s_i
0	0	0	0	0	0	0	0
0	0	0	1	0 (x)	1	0 (✓)	0
0	0	1	0	1	1	0	0
0	0	1	1	1	1	0	0
0	1	0	0	0	0	0	0
0	1	0	1	0 (x)	1	0 (x)	1
0	1	1	0	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0
1	0	0	1	0 (x)	1	0 (✓)	0
1	0	1	0	1	1	0	0
1	0	1	1	1	1	0	0
1	1	0	0	0	0	1	1
1	1	0	1	0 (x)	1	1 (x)	0
1	1	1	0	1	1	0	0
1	1	1	1	1	1	0	0

According to (29), Table 1 lists the truth table of speculated sum s_i^*, accurate sum si, speculated Ling carry H_{i-1}^*, and accurate Ling carry H_{i-1} for the $\mathrm{VLPPLA}_{\mathrm{BK64\_7}}$, where the subscripts of α and $\beta\alpha$ are omitted for simplicity and can be seen in (33). If the error correction is made for H_{i-1}^* according to the values of α and $\beta\alpha$, four erroneously speculated cases have to be done among these 16 conditions as marked using (x), as in the previous work [26]. This leads to large error rates. On the other hand, the Ling carries are not the final goal of a VLPPLA. Thus, the error correction is made for s_i^* directly instead of H_{i-1}^* for the proposed VLPPLAs in this work, leading to two erroneously speculated cases marked using (x) among 16 conditions in Table 1.

With these analyses, the OEDS E for the speculated sums of the $\mathrm{VLPPLA}_{\mathrm{BK64\_7}}$ is

$$\begin{aligned}E=&\,p_{63}\overline{\alpha_{[63:57]}}\beta_{[62:56]}\alpha_{[55:1]}+p_{62}\overline{\alpha_{[62:56]}}\beta_{[61:55]}\alpha_{[54:0]}\\&+\bigcup_{k=27}^{30}p_{2k+1}\overline{\alpha_{[2k+1:49]}}\beta_{[2k:48]}\alpha_{[47:1]}\\&+\bigcup_{k=27}^{30}p_{2k}\overline{\alpha_{[2k:48]}}\beta_{[2k-1:47]}\alpha_{[46:0]}\\&+\bigcup_{k=23}^{26}p_{2k+1}\overline{\alpha_{[2k+1:41]}}\beta_{[2k:40]}\alpha_{[39:1]}\\&+\bigcup_{k=23}^{26}p_{2k}\overline{\alpha_{[2k:40]}}\beta_{[2k-1:39]}\alpha_{[38:0]}\\&+\bigcup_{k=19}^{22}p_{2k+1}\overline{\alpha_{[2k+1:33]}}\beta_{[2k:32]}\alpha_{[31:1]}\\&+\bigcup_{k=19}^{22}p_{2k}\overline{\alpha_{[2k:32]}}\beta_{[2k-1:31]}\alpha_{[30:0]}\\&+\bigcup_{k=15}^{18}p_{2k+1}\overline{\alpha_{[2k+1:25]}}\beta_{[2k:24]}\alpha_{[23:1]}\\&+\bigcup_{k=15}^{18}p_{2k}\overline{\alpha_{[2k:24]}}\beta_{[2k-1:23]}\alpha_{[22:0]}\\&+\bigcup_{k=11}^{14}p_{2k+1}\overline{\alpha_{[2k+1:17]}}\beta_{[2k:16]}\alpha_{[15:1]}\\&+\bigcup_{k=11}^{14}p_{2k}\overline{\alpha_{[2k:16]}}\beta_{[2k-1:15]}\alpha_{[14:0]}\\&+\bigcup_{k=7}^{10}p_{2k+1}\overline{\alpha_{[2k+1:9]}}\beta_{[2k:8]}\alpha_{[7:1]}\\&+\bigcup_{k=7}^{10}p_{2k}\overline{\alpha_{[2k:8]}}\beta_{[2k-1:7]}\alpha_{[6:0]}\end{aligned} \tag{34}$$

Since

$$\overline{g_i}p_i=d_i \tag{35}$$

considering (30) and the last term in (34), we have

$$\bigcup_{k=7}^{10}p_{2k}\overline{\alpha_{[2k:8]}}\beta_{[2k-1:7]}\alpha_{[6:0]}=\bigcap_{i=7}^{14}d_ip_6\alpha_{[6:0]} \tag{36}$$

where the symbol ∩ represents the logic AND. Thus, (34) is simplified as

$$
\begin{aligned}
E = & \bigcap_{j_7=56}^{63} d_{j_7} p_{55} \alpha_{[55:1]} + \bigcap_{i_7=55}^{62} d_{i_7} p_{54} \alpha_{[54:0]} \\
& + \bigcap_{j_6=48}^{55} d_{j_6} p_{47} \alpha_{[47:1]} + \bigcap_{i_6=47}^{54} d_{i_6} p_{46} \alpha_{[46:0]} \\
& + \bigcap_{j_5=40}^{47} d_{j_5} p_{39} \alpha_{[39:1]} + \bigcap_{i_5=39}^{46} d_{i_5} p_{38} \alpha_{[38:0]} \\
& + \bigcap_{j_4=32}^{39} d_{j_4} p_{31} \alpha_{[31:1]} + \bigcap_{i_4=31}^{38} d_{i_4} p_{30} \alpha_{[30:0]} \cdot \\
& + \bigcap_{j_3=24}^{31} d_{j_3} p_{23} \alpha_{[23:1]} + \bigcap_{i_3=23}^{30} d_{i_3} p_{22} \alpha_{[22:0]} \\
& + \bigcap_{j_2=16}^{23} d_{j_2} p_{15} \alpha_{[15:1]} + \bigcap_{i_2=15}^{22} d_{i_2} p_{14} \alpha_{[14:0]} \\
& + \bigcap_{j_1=8}^{15} d_{j_1} p_7 \alpha_{[7:1]} + \bigcap_{i_1=7}^{14} d_{i_1} p_6 \alpha_{[6:0]}
\end{aligned} \tag{37}
$$

Although the logic expression is complex, it shares the same elements with the Ling carries H_{2k} and H_{2k+1} in (33) or sums in (29). The error detection module needs only several AND gates more than the exact circuits to compute the OEDS.

3.4 | BEDS

For the VLPPAs and VLPPLAs, the BEDS is introduced in this subsection.

$VLPPA_{BK64\_7}$: By separately considering the elements in the OEDS E for the $\text{VLPPA}_{\text{BK64\_7}}$ in (17), we have E_{16}, E_{15}, ..., E_6, E_5, E_4, E_3, E_2, and E_1 for Blocks 16, 15, ..., 2, and 1 in Figure 2b as

$$
\begin{aligned}
& E_{16} = P_{[63:60]} G_{[59:56]}, E_{15} = P_{[59:56]} G_{[55:52]}, E_{14} = P_{[55:52]} G_{[51:48]}, \\
& E_{13} = P_{[51:48]} G_{[47:44]}, E_{12} = P_{[47:44]} G_{[43:40]}, E_{11} = P_{[19:16]} G_{[15:12]}, \\
& E_{10} = P_{[39:36]} G_{[35:32]}, E_9 = P_{[35:32]} G_{[31:28]} \\
& E_8 = P_{[31:28]} G_{[27:24]}, E_7 = P_{[27:24]} G_{[23:20]}, \\
& E_6 = P_{[23:20]} G_{[19:16]}, E_5 = P_{[19:16]} G_{[15:12]} \\
& E_4 = P_{[15:12]} G_{[11:8]}, E_3 = P_{[11:8]} G_{[7:4]}, E_2 = P_{[7:4]} G_{[3:0]}, E_1 = 0
\end{aligned} \tag{38}
$$

where E_1 is 0 since the carry-outs in Block 1 are accurately computed. We name these eight elements the BEDSs of the $\text{VLPPA}_{\text{BK64\_7}}$ in this work. Thus, the OEDS E of the $\text{VLPPA}_{\text{BK64\_7}}$ equals the ORed results of these BEDSs as

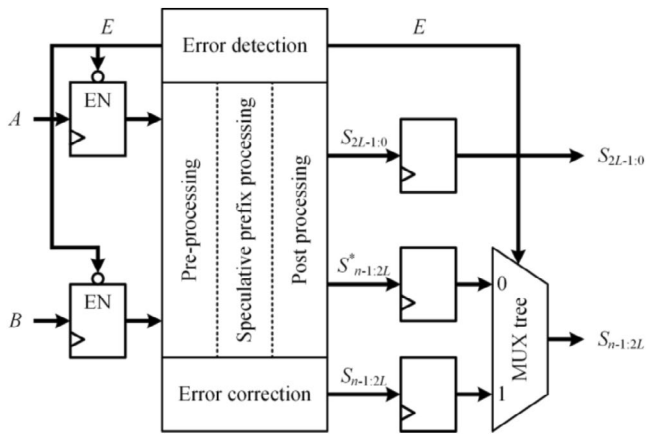


FIGURE 3 | The architecture for n-bit VLPPLAs with a maximum carry chain length L.

$$
\begin{aligned}
E = & E_{16} + E_{15} + E_{14} + E_{13} + E_{12} + E_{11} + E_{10} + E_9 + E_8 \\
& + E_7 + E_6 + E_5 + E_4 + E_3 + E_2 + E_1
\end{aligned} \tag{39}
$$

$VLPPLA_{BK64\_7}$: By separately considering the elements in the OEDS E in (37), we obtain E_8, ..., E_4, E_3, E_2, and E_1 for bits in Blocks 8, ..., 3, 2, and 1 of the $\text{VLPPLA}_{\text{BK64\_7}}$ as

$$
\begin{aligned}
E_8 &= \bigcap_{j_7=56}^{63} d_{j_7} p_{55} \alpha_{[55:1]} + \bigcap_{i_7=55}^{62} d_{i_7} p_{54} \alpha_{[54:0]}, \\
E_7 &= \bigcap_{j_6=48}^{55} d_{j_6} p_{47} \alpha_{[47:1]} + \bigcap_{i_6=47}^{54} d_{i_6} p_{46} \alpha_{[46:0]}, \\
E_6 &= \bigcap_{j_5=40}^{47} d_{j_5} p_{39} \alpha_{[39:1]} + \bigcap_{i_5=39}^{46} d_{i_5} p_{38} \alpha_{[38:0]}, \\
E_5 &= \bigcap_{j_4=32}^{39} d_{j_4} p_{31} \alpha_{[31:1]} + \bigcap_{i_4=31}^{38} d_{i_4} p_{30} \alpha_{[30:0]}, \\
E_4 &= \bigcap_{j_3=24}^{31} d_{j_3} p_{23} \alpha_{[23:1]} + \bigcap_{i_3=23}^{30} d_{i_3} p_{22} \alpha_{[22:0]}, \\
E_3 &= \bigcap_{j_2=16}^{23} d_{j_2} p_{15} \alpha_{[15:1]} + \bigcap_{i_2=15}^{22} d_{i_2} p_{14} \alpha_{[14:0]}, \\
E_2 &= \bigcap_{j_1=8}^{15} d_{j_1} p_7 \alpha_{[7:1]} + \bigcap_{i_1=7}^{14} d_{i_1} p_6 \alpha_{[6:0]}, E_1 = 0
\end{aligned} \tag{40}
$$

where E_1 is 0 since the sums for the bits in Block 1 are accurately computed. Thus, the OEDS E of the $\text{VLPPLA}_{\text{BK64\_7}}$ equals the ORed results of these BEDSs as

$$
E = E_8 + E_7 + E_6 + E_5 + E_4 + E_3 + E_2 + E_1 \tag{41}
$$

3.5 | Error Correction

Error correction for the speculated sums is made if the OEDS is asserted, as in previous works [21, 23], by using the removed rows to compute the exact Ling carries and sums.

3.6 | Postprocessing Stage

With the computed Ling carries, propagate p_i, and half-sum d_i, the speculated and exact results can be computed according to (29).

From the analyses above, it can be concluded that the less significant $2L$ sums are correctly computed, whereas the most significant n-$2L$ sums are speculated, for an n-bit VLPPLA with a maximum carry chain length L. Figure 3 shows the architecture for the designed VLPPLAs, where the speculated sums $s^*_{n-1:2L}$, accurate sums $s_{n-1:2L}$, and OEDS E are computed. If the OEDS E of a VLPPLA is 0, the multiplexer (MUX) tree will select the speculated sums or the exact sums in the next clock cycle.

4 | Experimental Results

The error rates of the VLPPAs and VLPPLAs are evaluated by computing the probabilities of BEDSs being asserted, assuming independent unsigned input operands. One million Monte Carlo trials are carried out for each design to validate the computations. In VS Code, Verilog hardware description language is used for programming PPAs, PPLAs, VLPPAs, and previous designs [23]. Additionally, in Vivado, the simulation chip XC7A35TFGG484I

from the Artix-7 series is used to conduct simulations and verify logic correctness. Besides, they are synthesized by the Synopsys Design Compiler with a TSMC 65-nm standard library at the typical corner [29] according to the same timing constraints. The command "set_multicycle_path" is employed to mark non-speculative outputs of a speculative adder, aiming to find the minimum tclk for comparing the performance of different adders. Finally, MATLAB is used to verify the correctness of the simulation results and compute the error rates of these adders.

4.1 | Simulation Results

From the simulation results shown in Figure 4, it can be observed that the design presented in this paper has fully achieved the intended objectives: When $E=0$ (indicating no error), the result is generated within the current clock cycle; when $E=1$, the modified and accurate result is produced in the next clock cycle. Additionally, a flag signal has been incorporated to verify the correctness of the adder's outputs.

4.2 | Error Rate

This subsection is to illustrate the error rates of the VLPPAs and VLPPLAs using different parallel prefix processing topologies. The probabilities of the bits a_i and b_i being 1 of the independent input operands $A=a_{n-1}a_{n-2}\cdots a_1a_0$ and $B=b_{n-1}b_{n-2}\cdots b_1b_0$ are respectively

$$\begin{aligned} P(a_i=1)&=P_{a_i}\\ P(b_i=1)&=P_{b_i} \end{aligned} \tag{42}$$

$VLPPA_{BK64\_7}$: The probabilities of the generate g_i and propagate p_i being 1 for a VLPPA are respectively

$$\begin{aligned} P(g_i=1)=P_{g_i}=P(a_ib_i=1)=P_{a_i}P_{b_i}\\ P(p_i=1)=P_{p_i}=P(a_i\oplus b_i=1)=P_{a_i}+P_{b_i}-2P_{a_i}P_{b_i} \end{aligned} \tag{43}$$

Take the BEDS E_6 in the $VLPPA_{BK64\_7}$ as an example as follows. According to (5),

$$\begin{aligned} P(E_{16}=1)&=P(p_{63}p_{62}p_{61}p_{60}G_{[59:56]}=1)\\ &=\prod_{i=60}^{63}(P_{p_i})(P(G_{[59:56]}=1)) \end{aligned} \tag{44}$$

where $p_{63}, p_{62}, p_{61}, p_{60}$, and $G_{[59:56]}$ are independent to each other. Note that

$$P(g_i+p_i=1)=P(g_i=1)+P(p_i=1) \tag{45}$$

The last term in (44),

$$P(G_{[27:24]}=1)=P(g_{27}+p_{27}G_{[26:24]}=1)=P_{g_{27}}+P_{p_{27}}P(G_{[26:24]}=1) \tag{46}$$

Thus,

$$P(E_{16}=1)=\prod_{i=60}^{63}(P_{p_i})\left(\begin{pmatrix}P_{g_{59}}+P_{p_{59}}\\ \begin{pmatrix}P_{g_{58}}+P_{p_{58}}\\ \begin{pmatrix}P_{g_{57}}+P_{p_{57}}\\ (P_{g_{56}})\end{pmatrix}\end{pmatrix}\end{pmatrix}\right) \tag{47}$$

$VLPPLA_{BK64\_7}$: The probabilities of the generate g_i, propagate p_i, and half-sum d_i being 1 for a VLPPLA are respectively

$$\begin{aligned} P(g_i=1)=P_{g_i}=P(a_ib_i=1)=P_{a_i}P_{b_i}\\ P(p_i=1)=P_{p_i}=P(a_i+b_i=1)=P_{a_i}+P_{b_i}-P_{a_i}P_{b_i}\\ P(d_i=1)=P_{d_i}=P(a_i\oplus b_i=1)=P_{a_i}+P_{b_i}-2P_{a_i}P_{b_i} \end{aligned} \tag{48}$$

Thus, the probabilities of the intermediate signals α_i and propagate β_i being 1 are respectively

$$\begin{aligned} P(\alpha_i=1)=P_{\alpha_i}=P(g_i+g_{i-1}=1)=P_{g_i}+P_{g_{i-1}}-P_{g_i}P_{g_{i-1}}\\ P(\beta_i=1)=P_{\beta_i}=P(p_ip_{i-1}=1)=P_{p_i}P_{p_{i-1}} \end{aligned} \tag{49}$$

With these probabilities, the BEDSs for each block can be computed respectively as follows by taking the BEDS E_4 of Block 4 in the $VLPPLA_{BK64\_7}$ as an example.

$$\begin{aligned} P(E_8=1)&=P\left(\bigcap_{j_7=56}^{63}d_{j_7}p_{55}\alpha_{[55:1]}+\bigcap_{i_7=55}^{62}d_{i_7}p_{54}\alpha_{[54:0]}=1\right)\\ &=P\left(\bigcap_{j_7=56}^{63}d_{j_7}p_{55}\alpha_{[55:1]}=1\right)+P\left(\bigcap_{i_7=55}^{62}d_{i_7}p_{54}\alpha_{[54:0]}=1\right)\cdot\\ &-P\left(\bigcap_{j_7=56}^{63}d_{j_7}p_{55}\alpha_{[55:1]}\bigcap_{i_7=55}^{62}d_{i_7}p_{54}\alpha_{[54:0]}=1\right) \end{aligned} \tag{50}$$

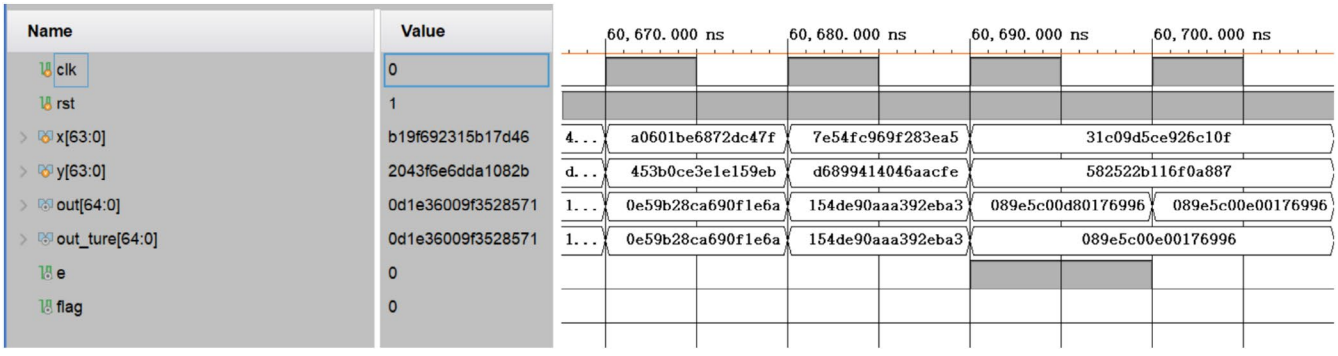


FIGURE 4 | Simulation results of the $VLPPLA_{BK64}$ in Vivado.

Each of the elements in (50) is computed as

$$P\left(\bigcap_{j_7=56}^{63} d_{j_7} p_{55}\alpha_{[55:1]}=1\right)=\prod_{j_7=56}^{63}\left(P_{d_{j_7}}\right)\left(P_{p_{55}}\left(\cdots\left(P_{p_1}\begin{pmatrix}P_{g_1/p_1}+\\(P_{g_0})\end{pmatrix}\right)\right)\right)\begin{matrix}P_{g_{55}/p_{55}}+\\\end{matrix} \tag{51}$$

$$P\left(\bigcap_{i_7=55}^{62} d_{i_7} p_{54}\alpha_{[54:0]}=1\right)=\prod_{i_7=55}^{62}\left(P_{d_{i_7}}\right)\left(P_{p_{54}}\left(\cdots\left(P_{p_0}\begin{pmatrix}P_{g_0/p_0}+\\(P_{g_{-1}})\end{pmatrix}\right)\right)\right)\begin{matrix}P_{g_{54}/p_{54}}+\\\end{matrix} \tag{52}$$

$$\begin{aligned}&P\left(\bigcap_{j_7=56}^{63} d_{j_7} p_{55}\alpha_{[55:1]}\bigcap_{i_7=55}^{62} d_{i_7} p_{54}\alpha_{[54:0]}=1\right)\\&=\prod_{i=55}^{63}\left(P_{d_i}\right)\left(P_{p_{54}}\left(\cdots\left(P_{p_1}\begin{pmatrix}P_{g_1/p_1}+\\(P_{g_0})\end{pmatrix}\right)\right)\right)\begin{matrix}P_{g_{54}/p_{54}}+\\\end{matrix}\end{aligned} \tag{53}$$

where $P_{g/p}$ means the probability of g being 1 under the condition that p is 1.

Figure 5 shows the sums of simulated and computed probabilities of the BEDSs for 32- and 64-bit VLPPAs and VLPPLAs. The computed results match the simulated ones well. These data show that the probabilities of the BEDSs for 32- and 64-bit VLPPLAs are smaller than those for VLPPAs by more than an order of magnitude. That is, the probability of errors occurring in VLPPLAs is significantly lower than that of VLPPAs. In detail, the probabilities are respectively lowered by up to 98.31% and 98.96% for 32- and 64-bit designs on average. Indeed, $\text{VLPPLA}_{\text{KN64\_16}}$ and $\text{VLPPLA}_{\text{KS64\_16}}$ get 0 BEDS errors in one million Monte Carlo trials, whereas their counterparts have about 195 BEDS errors on average. In addition, $\text{VLPPA}_{\text{BK32\_7}}$, $\text{VLPPA}_{\text{BS32\_8}}$, $\text{VLPPA}_{\text{SK32\_8}}$, $\text{VLPPA}_{\text{BK64\_7}}$, $\text{VLPPA}_{\text{BS64\_8}}$, and $\text{VLPPA}_{\text{SK64\_8}}$ generate larger error rates than others.

Figure 6 shows the simulated probabilities of the OEDSs for 32- and 64-bit VLPPAs and VLPPLAs. The error probabilities of 32- and 64-bit VLPPLAs are reduced by about 96.37% and 96.50% on average compared with those of VLPPAs. For example, $\text{VLPPA}_{\text{BK32\_7}}$, $\text{VLPPA}_{\text{BS32\_8}}$, and $\text{VLPPA}_{\text{SK32\_8}}$ realize error probabilities of about 0.062, 0.086, and 0.086, whereas the error probabilities of $\text{VLPPLA}_{\text{BK32\_7}}$, $\text{VLPPLA}_{\text{BS32\_8}}$, and $\text{VLPPLA}_{\text{SK32\_8}}$ are 0.0087, 0.00148, and 0.00148, respectively. Note that the error probabilities of $\text{VLPPLA}_{\text{KN32\_8}}$ and $\text{VLPPLA}_{\text{KS32\_8}}$ are only 6.40×10^{-5} on average, and $\text{VLPPLA}_{\text{KN64\_16}}$ and $\text{VLPPLA}_{\text{KS64\_16}}$ produce no error in one million Monte Carlo trials.

Figure 7 shows the differences between the summed probabilities of the BEDSs and the probabilities of the OEDSs for 32- and 64-bit VLPPAs and VLPPLAs. Take the $\text{VLPPA}_{\text{BK32\_7}}$ as an example.

$$\begin{aligned}&P(E=1)\\&=P(E_4+E_3+E_2+E_1=1)\\&<P(E_4=1)+P(E_3=1)+P(E_2=1)+P(E_1=1)\end{aligned} \tag{54}$$

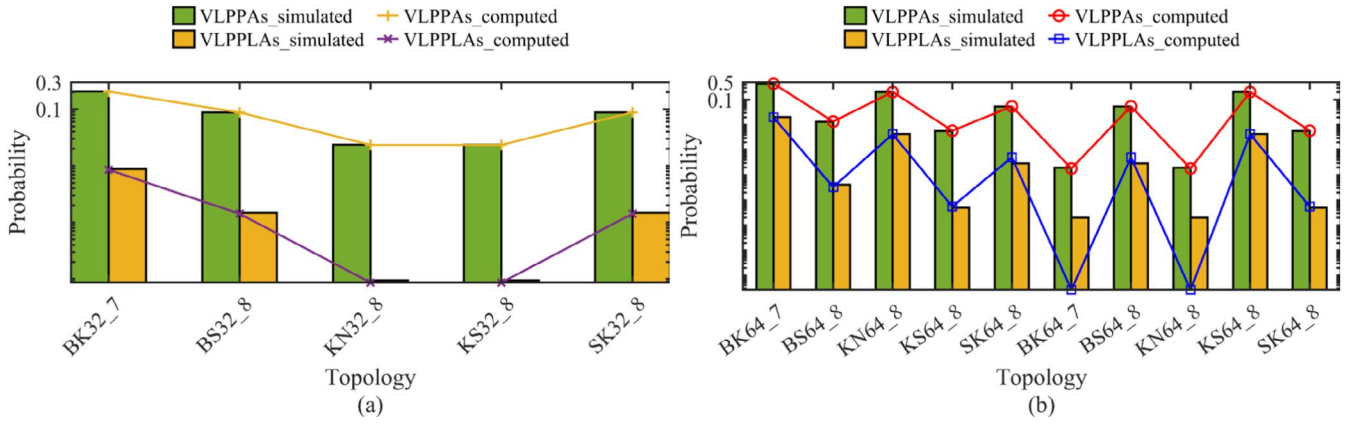


FIGURE 5 | The sum of simulated and computed probabilities of the BEDSs for (a) 32-bit VLPPAs and VLPPLAs and (b) 64-bit VLPPAs and VLPPLAs.

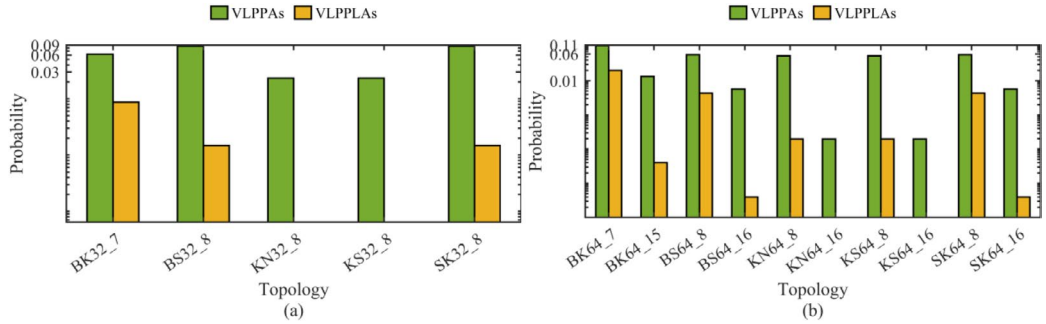


FIGURE 6 | The simulated probabilities of the OEDSs for (a) 32-bit VLPPAs and VLPPLAs and (b) 64-bit VLPPAs and VLPPLAs.

Thus, the probability of the OEDS E in the $\text{VLPPA}_{\text{BK32\_7}}$ is smaller than the summed probabilities of the BEDSs E_4, E_3, E_2, and E_1. The difference between them indicates the dependence between the divided blocks. The larger the difference is, the higher the dependence becomes for a VLPPA or VLPPLA. That is, the predefined maximum carry chain length is too small for most input operands, which will increase the error rates and probabilities of error correction. For example, $\text{VLPPA}_{\text{BK32\_7}}$ and $\text{VLPPA}_{\text{BK64\_7}}$ with a carry chain length of 7 bits generate large error probabilities. The error probabilities can be lowered by about 93.35% and 99.92% by increasing the maximum carry chain length from 7 (or 8) to 15 (or 16) for 64-bit VLPPAs and VLPPLAs. Besides, $\text{VLPPA}_{\text{BS32\_8}}$, $\text{VLPPA}_{\text{SK32\_8}}$, $\text{VLPPA}_{\text{BS64\_8}}$, and $\text{VLPPA}_{\text{SK64\_8}}$ are inferior designs since they generate large error rates if considering computing accuracy only.

4.3 | Hardware Costs

Figure 8 shows the hardware costs of the considered 64-bit VLPPAs [23] and VLPPLAs in terms of area, power, and $\text{ED}_{\text{avg}}\text{P}$ versus average latency, where the average latency is computed using (18) and simulated error probabilities of the OEDSs above.

The EDavgP is given as

$$\begin{aligned} ED_{avg}P &= energy \cdot T_{avg} \\ energy &= power \cdot T_{avg} \end{aligned} \tag{55}$$

For 64-bit designs, $\text{VLPPLA}_{\text{BK64}}$, $\text{VLPPLA}_{\text{KN64}}$, $\text{VLPPLA}_{\text{KS64}}$, $\text{VLPPLA}_{\text{SK64}}$, and $\text{VLPPLA}_{\text{BS64}}$ provide shorter operation time at the cost of smaller area and certain power consumption compared with their corresponding original structures. In particular, $\text{VLPPLA}_{\text{SK64}}$ and $\text{VLPPLA}_{\text{BS64}}$ also have a certain reduction in power consumption. Specifically, in terms of latency, the improvements of $\text{VLPPLA}_{\text{KN64}}$ and $\text{VLPPLA}_{\text{KS64}}$ are the most significant, with an increase of approximately over 20%, whereas the improvements of the other three structures are relatively weak, only around 10%. Additionally, in terms of $\text{ED}_{\text{avg}}\text{P}$, nearly all topological structures have been improved. Particularly for $\text{VLPPLA}_{\text{SK64}}$ and $\text{VLPPLA}_{\text{BS64}}$, there is an improvement of approximately 23% application.

All the data in Figure 9, the area, power, and EDavgP versus average latency for (a), (b), and (c) 64-bit VLPPLA, VLPPA, PPLA, PPA, CLA, CSA, and RCA, are obtained through simulation tests using the 65-nm process library with the same timing constraints. The VLPPLA structure proposed in this paper exhibits the lowest latency among all the architectures. However, it also incurs higher costs in terms of area and power

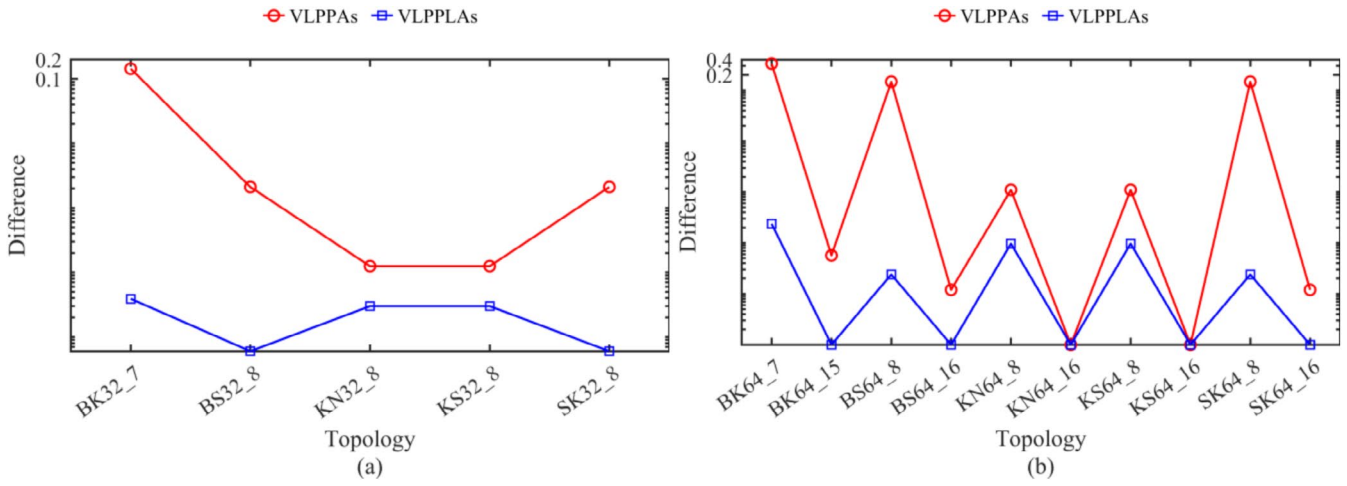


FIGURE 7 | The differences between the summed probabilities of the BEDSs and the probabilities of the OEDSs for (a) 32-bit VLPPAs and VLPPLAs and (b) 64-bit VLPPAs and VLPPLAs.

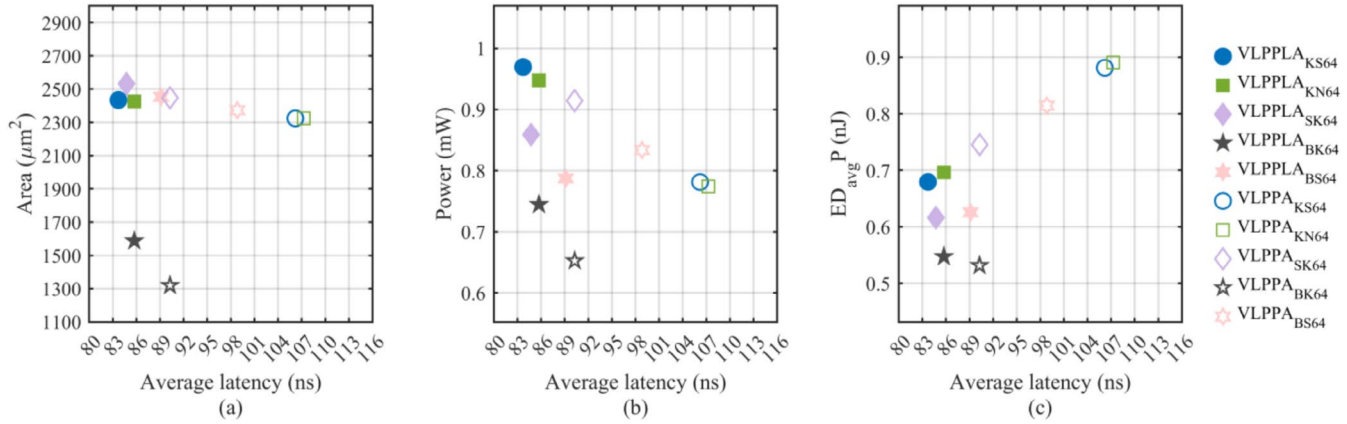


FIGURE 8 | The area, power, and $\text{ED}_{\text{avg}}\text{P}$ versus average latency for (a), (b), and (c) 64-bit VLPPAs and VLPPLAs.

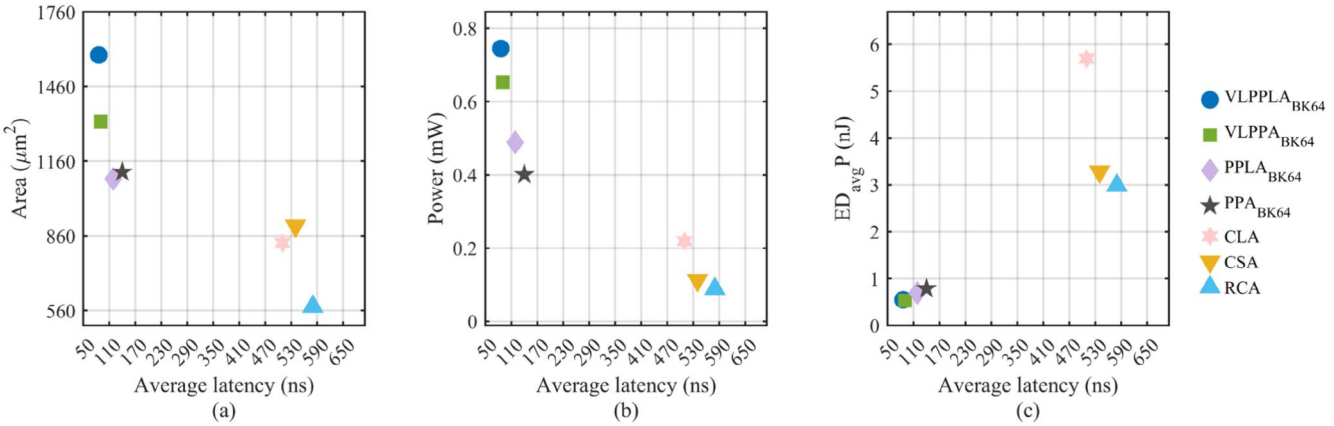


FIGURE 9 | The area, power, and EDavgP versus average latency for (a), (b), and (c) 64-bit VLPPLA, VLPPA, PPLA, PPA, CLA, CSA, and RCA.

TABLE 2 | The average latency of the five topological structures for VLPPLAs, VLPPAs, PPLAs, and PPAs.

	—	BS64	BK64	SK64	KN64	KS64
VLPPLA	—	0.89	0.85	0.84	0.85	0.83
VLPPA	—	0.98	0.90	0.90	1.07	1.06
PPLA	—	0.99	1.19	0.92	0.86	0.84
PPA	—	1.36	1.4	1.29	1.04	0.98
CLA	5.1	—	—	—	—	—
CSA	5.4	—	—	—	—	—
RCA	5.4	—	—	—	—	—

consumption compared with other structures. Therefore, the design presented in this paper is suitable for applications with extremely high latency requirements. In addition, since the hardware costs of topological structures such as BS and KN are relatively close to that of the BK structure, comparing them in the same graph would confuse the originally obvious conclusions. Therefore, other topological structures are not added to the graph for comparison here. The delay of other topological structures can be found in Table 2. The average latency of the five topological structures for VLPPLAs, VLPPAs, PPLAs, and PPAs.

Table 2 shows the average latency of five topological structures of four adders: PPAs, PPLAs, VLPPAs, and the VLPPLAs designed in this paper, as well as the average latency of classic adders CLA, CSA, and RCA. It can be easily seen from the table that, in terms of average latency, after using the Ling structure, variable latency processing, and optimizing the error detection signal, the latency of the VLPPLAs designed has been optimized. This makes it achieve the shortest latency under various topological structures, demonstrating the extensibility of the method proposed in this paper. Moreover, under the same simulation conditions, compared with the CLA, CSA, and RCA structures in existing literature, the latency has been significantly improved.

From the analyses above, the OEDS of VLPPLA derived through truth tables significantly reduces the error rate compared with VLPPA, which significantly reduces the error rate. Moreover, compared with the existing designs, because of the reduced error rate and the structural advantages of the VLPPLAs itself, the average latency is reduced. Although area and power consumption are sacrificed, the overall energy metrics are optimized. And because of the lower error rate, it can be well applied to 2's complement, which gives it certain practical value. Implementation results confirm that, as pointed out in [21], VLAs can be useful when the highest speed is required; otherwise, the standard, nonspeculative adders remain the best choice.

5 | Conclusion

In this work, VLPPLAs are designed based on Ling adders, using Brent–Kung, Beaumont Smith, Knowles, Kogge–Stone, and Sklansky topologies, respectively. The OEDS and BEDSs are derived from the truth table to evaluate the error rates of these VLPPLAs. Computing and simulation results demonstrate that the VLPPLAs realize significantly lower error rates than conventional VLPPAs. Physically synthesized results show that the VLPPLAs in this paper are superior to VLPPAs in energy performance and average latency.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62404067, in part by the Fundamental Research Funds for the Central Universities under Grant JZ2025HGTB0231, and in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada under Grant RES0048688.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

Research data are not shared.

References

1. R. Brent and H. Kung, "A Regular Layout for Parallel Adders," *IEEE Transactions on Computers* C-31, no. 3 (1982): 260–264.

2. A. Beaumont-Smith and C. Lim, "Parallel Prefix Adder Design," in *15th IEEE Symposium on Computer Arithmetic, Vail, CO, USA* (IEEE, 2001), 218–225.

3. S. Knowles, "A Family of Adders," in *Proceedings 14th IEEE Symposium on Computer Arithmetic (Cat. No.99CB36336)* (IEEE, 1999), 30–34.

4. P. Kogge and H. Stone, "A Parallel Algorithm for the Efficient Solution of a General Class of Recurrence Equations," *IEEE Transactions on Computers* C-22, no. 8 (1973): 786–793.

5. J. Sklansky, "Conditional-Sum Addition Logic," *IEEE Transactions on Electronic Computers* EC-9, no. 2 (1960): 226–231.

6. R. Ladner and M. Fischer, "Parallel Prefix Computation," *Journal of the ACM* 27, no. 4 (1980): 831–838.

7. I. Koren, *Computer Arithmetic Algorithms* (A K Peters/CRC Press, 2002).

8. V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-Power Digital Signal Processing Using Approximate Adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, no. 1 (2013): 124–137.

9. A. Sadeghi, R. Ghasemi, H. Ghasemian, and N. Shiri, "High Efficient GDI-CNTFET-Based Approximate Full Adder for Next-Generation of Computer Architectures," *IEEE Embedded Systems Letters* 15, no. 1 (2022): 33–36.

10. H. Mahdiani, A. Ahmadi, S. Fakhraie, and C. Lucas, "Bio-Inspired Imprecise Computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," *IEEE Transactions on Circuits and Systems I, Regular Papers* 57, no. 4 (2010): 850–862.

11. A. Dalloo, A. Najafi, and A. Garcia-Ortiz, "Systematic Design of an Approximate Adder: The Optimized Lower Part Constant-OR Adder," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26, no. 8 (2018): 1595–1599.

12. O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "RAP-CLA: A Reconfigurable Approximate Carry Look-Ahead Adder," *IEEE Transactions on Circuits and Systems II: Express Briefs* 65, no. 8 (Aug. 2018): 1089–1093.

13. M. Shafique, W. Ahmad, R. Hafiz, and J. Henkel, "A Low Latency Generic Accuracy Configurable Adder," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), San Francisco, CA, USA* (IEEE, 2015), 1–6.

14. H. Seo and Y. Kim, "A Low Latency Approximate Adder Design Based on Dual Sub-Adders With Error Recovery," *IEEE Transactions on Emerging Topics in Computing* 11, no. 3 (2023): 811–816.

15. M. Rosa, G. Paim, P. Costa, E. Costa, R. Soares, and S. Bampi, "AxPPA: Approximate Parallel Prefix Adders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 31, no. 1 (2023): 17–28.

16. F. Ebrahimi-Azandaryani, O. Akbari, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Block-Based Carry Speculative Approximate Adder for Energy-Efficient Applications," *IEEE Transactions on Circuits and Systems II: Express Briefs* 67, no. 1 (Jan. 2020): 137–141.

17. V. Mrazek, L. Sekanina, and Z. Vasicek, "Libraries of Approximate Circuits: Automated Design and Application in CNN Accelerators," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 10, no. 4 (2020): 406–418.

18. A. Verma, P. Brisk, and P. Ienne, "Variable Latency Speculative Addition: A New Paradigm for Arithmetic Circuit Design," in *2008 Design, Automation and Test in Europe, Munich, Germany* (IEEE, 2008), 1250–1255.

19. A. Cilardo, "A New Speculative Addition Architecture Suitable for Two's Complement Operations," in *Design, Automation & Test in Europe Conference & Exhibition* (IEEE, 2009), 664–669.

20. Y. Chen, H. Li, C. Koh, et al., "Variable-Latency Adder (VL-Adder) Designs for Low Power and NBTI Tolerance," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 18, no. 11 (2010): 1621–1624.

21. D. Esposito, D. De, E. Napoli, N. Petra, and A. Strollo, "Variable Latency Speculative Han-Carlson Adder," *IEEE Transactions on Circuits and Systems I: Regular Papers* 62, no. 5 (2015): 1353–1361.

22. I. Lin, Y. Yang, and C. Lin, "High-Performance Low-Power Carry Speculative Addition With Variable Latency," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23, no. 9 (2015): 1591–1603.

23. D. Esposito, D. De, and A. Strollo, "Variable Latency Speculative Parallel Prefix Adders for Unsigned and Signed Operands," *IEEE Transactions on Circuits and Systems I: Regular Papers* 63, no. 8 (2016): 1200–1209.

24. S. Chen, C. Liu, T. Wu, and A. Tsai, "Design and Implementation of High-Speed and Energy-Efficient Variable-Latency Speculating Booth Multiplier (VLSBM)," *IEEE Transactions on Circuits and Systems I: Regular Papers* 60, no. 10 (2013): 2631–2643.

25. P. Seidel, "Variable Latency Division," in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS), Windsor, ON, Canada* (IEEE, 2018), 944–947.

26. P. Guo, G. Xie, X. Chen, J. Han, and Y. Zhang, "A Variable Latency Ling Adder Based on Brent-Kung Parallel-Prefix Topology," in *IEEE 23rd International Conference on Nanotechnology (NANO)* (Republic of Korea, 2023), 1–5.

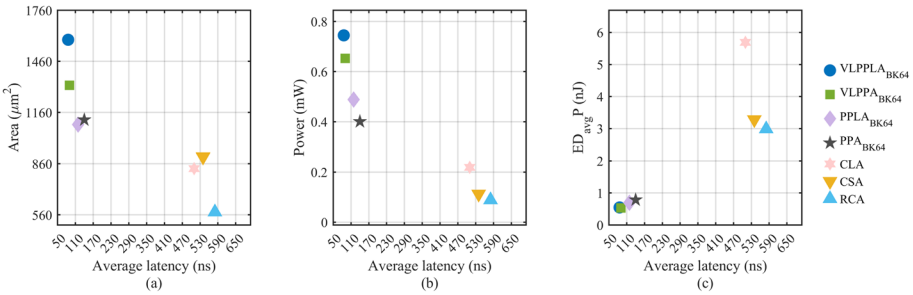
27. H. Ling, "High-Speed Binary Adder," *IBM Journal of Research and Development* 25, no. 3 (1981): 156–166.

28. G. Dimitrakopoulos and D. Nikolos, "High-Speed Parallel-Prefix VLSI Ling Adders," *IEEE Transactions on Computers* 54, no. 2 (2005): 225–231.

29. "TCBN40LPBWP TSMC N40LP Standard Cell Library Datasheet," *Taiwan Semiconductor Manufacturing Company*, 2013.

Graphical Abstract

Please note that Graphical Abstracts only appear online as part of a table of contents and are not part of the main article (therefore, they do not appear in the article HTML or PDF files).



The area, power, and EDavgP versus average latency for (a), (b), and (c) 64-bit VLPPLA, VLPPA, PPLA, PPA, CLA, CSA, and RCA.