Analysis of H-Parallel Prefix Circuits with Dense Faulty Nodes

Hatem M. El-Boghdadi

Islamic University of Madinah, Faculty of Computer& Information, Saudi Arabia

Abstract
Parallel prefix circuits compute the prefix operation over a given input and are widely used in hardware systems. They play an important role in applications such as cryptography and high-performance adders, and any hardware design that incorporates adders can benefit from efficient prefix computation. Existing prefix circuits vary in performance, cost, and size, and most rely on operation nodes with fan-in and fan-out of two. One way to classify these circuits is by their width: those whose width matches the input size, and those whose width is smaller. In this paper, we focus on the first category. A well-known example of this type is the L-circuit, which performs well when the circuit width matches the input size. We analyze the behavior of the L-circuit in the presence of faulty nodes, estimating both the time delay and the number of idle nodes resulting from a fault at a given location. Based on this analysis, we propose several new designs that provide improved resilience to node failures.

Keywords:
parallel prefix circuits, prefix computation, faulty elements.

1. Introduction

The prefix operation is a fundamental computation with a wide range of applications. A prefix over a set of inputs can be computed either sequentially or in parallel. Parallel prefix computation, in particular, has attracted significant interest because it provides substantial speedups and supports various efficient techniques. For an associative operation *, the prefix of inputs $x_1, x_2, \dots, x_n$is the sequence $y_1, y_2, \dots, y_n$, where

$$y_i = x_1 * x_2 * \cdots * x_i.$$

Prefix computation plays a crucial role across numerous domains. It is directly used in applications such as encryption, biological sequence processing, and many forms of adders, especially high-performance or fast adders.

Historically, two primary approaches have been pursued for prefix computation: algorithmic methods and combinational circuit implementations. The algorithmic approach relies on established parallel computation models—such as the Parallel Random Access Machine (PRAM) [1], reconfigurable mesh (R-Mesh) [3], hypercubes [2], and others—to efficiently solve the prefix problem.

Prefix computation plays a critical role in many areas. It is directly used in applications such as encryption, biological sequence analysis, and various types of adders, particularly high-speed adders. Two main approaches have been explored for performing prefix computation: algorithmic methods and combinational circuit designs. The algorithmic approach relies on established parallel computation models—such as the Parallel Random Access Machine (PRAM) [1], reconfigurable mesh (R-Mesh) [3], and hypercube architectures [2]—to efficiently solve the prefix problem.

The second approach relies on computational and combinational circuits to solve the prefix problem. A combinational circuit can be represented as a directed acyclic graph (DAG) with $m$inputs and $m$outputs, and is therefore said to have width m. This DAG contains several operation nodes and at least one duplication node.

An operation node has two inputs and one or two outputs, and applies the associative operation $*$ to its input values. A duplication node, by contrast, simply replicates its input without performing any operation. The width of the circuit corresponds to the number of inputs it can process simultaneously, while its size is defined as the number of operation nodes it contains. The circuit depth refers to the number of levels in the DAG. Figure 1 illustrates both the operation and duplication nodes.

Figure 2 presents an example of a prefix circuit, $L(9)$[7]. The inputs enter at the top of the circuit, and after a

Manuscript received December 5, 2025
Manuscript revised December 20, 2025
https://doi.org/10.22937/IJCSNS.2025.25.12.1

number of time steps—corresponding to the circuit's depth—the prefix results are produced.

Existing prefix circuit designs generally fall into three categories: circuits whose width matches the input size, circuits whose width is smaller than the input size, and reconfigurable circuits that allow flexible width. In this paper, we consider the analysis

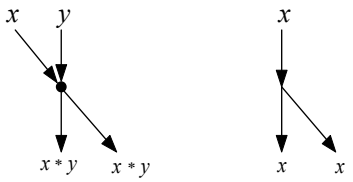

Figure 1 (a) operation node (b) duplication node

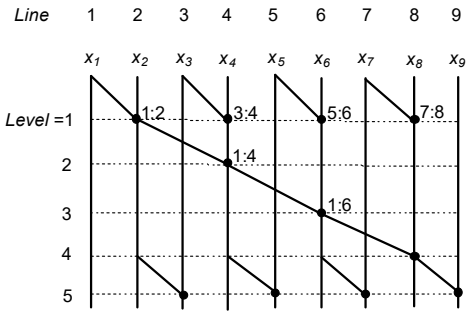

Figure 2 Example of a prefix circuit, $L(9)$ [7].

and design of the first class of circuits in the existence of dense faulty nodes. Many circuits were proposed in literature where they differ in performance, size, width and waist.

Figure 2 illustrates an example of a prefix circuit, $L(9)$[7]. The inputs are applied at the top of the circuit, and after a number of time steps—equal to the circuit's depth—the corresponding prefix outputs are generated.

Prefix circuit designs typically fall into three categories: circuits whose width equals the input size, circuits whose width is smaller than the input size, and reconfigurable circuits capable of adjusting their width as needed.

Based on this analysis, we propose several parallel prefix circuits designs capable of tolerating faulty nodes. For each design, we estimate the additional operation nodes required to compensate for a single faulty node. These results build upon our earlier work presented in [8].

The next section presents the analysis of the L-circuit [7]. Section 3 presents the new proposed circuits. Section 4 summarizes our results and makes some concluding remarks.

2. Prefix Circuits with Dense Faulty Nodes

In this section we consider the analysis of circuits that perform well when the -input size is of the same width as the circuit width.

2.1 Analysis of L-circuit with dense Faulty nodes

In this section, we examine a depth–size optimal parallel prefix circuit known as the L-circuit. Prior work [7] established that, among all depth–size optimal circuits with fan-out 2, the L-circuit achieves the minimum possible depth. It is also known to perform well when the input size matches the circuit width.

Here, we analyze the behavior of the L-circuit in the presence of faulty nodes and evaluate the performance impact introduced by such faults.

Let **L(m)** denote an L-circuit of width m, and let **d(L)** represent its depth. When an input sequence $x_1, x_2, \dots, x_m$ is applied to the circuit, it produces the corresponding prefix sums after $d(L)$time steps. Figure 2 illustrates an L-circuit of width 9, $L(9)$, which has a depth of 5—that is, it consists of 9 lines and 5 levels.

We first demonstrate that an L-circuit of width m, $L(m)$, can be used to compute prefix sums for an input of size n, where $n < m$. Examining the structure of the L-circuit, we observe that there are no connections from line $j$to line $i$when $i < j$, and that the node levels in line

$j$are always greater than those in line i. Consequently, the output of line $i$depends only on inputs from preceding lines and is unaffected by any line $j > i$. This observation leads to the following result.

Lemma 1[8]. *Let L(m) be an L-circuit of width m. L(m) can be used to compute the prefix computation for an input of width n<m* ■

Next, we consider the presence of faulty nodes in an L-circuit. Let a faulty node occur in line $i \leq m$. Since this node influences all outputs in lines $j$where $i \leq j \leq m$, these outputs will be affected and cannot be used. In contrast, the outputs of all lines $k < i$remain unaffected. Therefore, the circuit can still perform prefix computation for inputs of size at most $n \leq i - 1$. Note that the computation still requires $d(L)$ time steps, corresponding to the depth of the circuit. From this, we obtain the following result.

Lemma 2[8]. *Let L(m) be an L-circuit of width m and depth d(L). If there is a faulty node in line i, then L(m) can be used to compute the prefix computation for an input of width h<i using the first h lines in d(L) time steps.* ■

Lemma 2 demonstrates that an L-circuit $L(m)$with a faulty node still requires $d(L)$time steps to compute prefix sums for an input of size $h < i$. In contrast, in a fault-free L-circuit, the depth required for computing prefix sums of size $h < i$would be less than that for the full input size m. We now analyze the performance penalty incurred when using $L(m)$with a faulty node at line $i$to compute prefix sums for an input of size $h < i$.

The L-circuit $L(m)$has a depth of $d(L) = \lceil m/2 \rceil$. When $L(m)$is used to compute prefix sums for an input of size $h < i$, this results in a time-step penalty of $\lceil m/2 \rceil - \lceil h/2 \rceil$. Consequently, we obtain the following result.

Corollary 3. *Let L(m) be an L-circuit of width m and depth d(L). If there is a faulty node in line i, then the time penalty of using L(m) to compute the prefix computation for an input of width h<i is* $\left(\left\lceil \frac{m}{2} \right\rceil - \left\lceil \frac{h}{2} \right\rceil\right)$ *time steps.* ■

Now we consider the number of idle operation nodes in the *L*-circuit if there exists a faulty node in line *i*. Recall the structure of the *L*-circuit, starting the third line, lines alternates in having either one operation node or two operation nodes. If there is a faulty node in line *i*, then all nodes in line *j*, $i \leq j \leq m$, are idle nodes. Thus. It is straightforward to show that the total number of idle nodes, *N*, in all lines from line *i* to line *m*, $\left(N = \left\lceil \frac{3}{2}(m - i) \right\rceil\right)$.

Lemma 3. *Let L(m) be an L-circuit of width m and depth d(L). If there is a faulty node in level d(L) and line i, then L(m) can be used to compute the prefix computation for as L(m) except for y_i in d(L) time steps. The output y_i will be in error.* ■

Lemma 3 shows that there is minimal effect on the prefix computation in case of the operation nodes in the last level are in errors. If there is a faulty node in the last level, it will affect only one output of the prefix computation as it is clear from Figure 2.

Lemma 4. *Let L(m) be an L-circuit of width m and depth d(L). If there is a faulty node in level 1 and line i, then L(m) can be used to compute the prefix computation for as L(m) from line 1 t ln i-1 in d(L) time steps.* ■

Lemma 4 shows that there is a larger effect on the prefix computation in case of the operation nodes in the first level are in errors. If there is a faulty node in the first level, it will affect the prefix computation of all lines > *i*.

3. Design of New Prefix circuits

In this section we propose several parallel prefix circuits that can handle dense faulty nodes better. The proposed designs are based on the analysis done in section 3 for the *L*-circuits.

3.1 Fault tolerant *L*-circuit

In this section we consider the *L*-circuit that was investigated in section 2.1. The importance of the *L*-circuit is that it belongs to the class of depth-size optimal circuits if the fan-out is 2. The *L*-circuit is shown to have good performance when the input size is the same as the width of the circuit.

Recall the analysis done in section 2.1, if a faulty node exists in line *i*, then *L*-circuit can be used to generate prefix computation for input of size *i*-1. It follows that if the faulty node exists in a line that is closer to line *m*, then the number of idle nodes is smaller. We will use this observation in designing several fault tolerant *L*-circuits (*FL*-circuits) that can better handle faulty nodes. Each proposed circuit will have a different number of duplicate nodes.

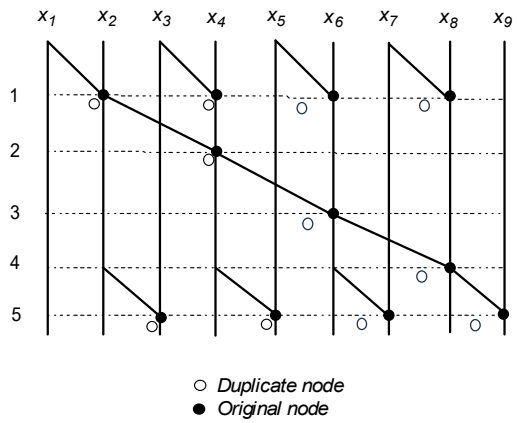

Figure 3 Prefix circuit FL-1(9)

Definition 1. A parallel prefix circuit is called an *FL-1 circuit*, if it has the same structure as *L*-circuit [7] in addition to one duplicate node for each of the original nodes in all the *m* lines. ■

The proposed design, *FL-1-circuit*, depends on having a duplicate operation node for each original operation node in all the lines. A duplicate operation node will be used if the original operation node is faulty. Such duplicate node if used (if a node is faulty), will save a large number of idle operation nodes in the circuit. Thus, *FL-1* circuit will be equipped with a duplicate node for each original node. If any original node goes faulty, then the duplicate node will be active and the whole circuit operate as originally designed. If any original node goes faulty, then the circuit can generate the prefix computation as the original circuit. Figure 3 shows an *FL-1* circuit of width 9. Consequently, we have the following theorem.

Theorem 5. *Let FL-1(m) be an FL-1-circuit of width m and depth d(FL-1). If there is a faulty node in any line, then FL-1(m) can generate prefix computation for the inputs as the original circuit in d(FL-1) time steps. Moreover, FL-1(m) has twice the number of original nodes.* ■

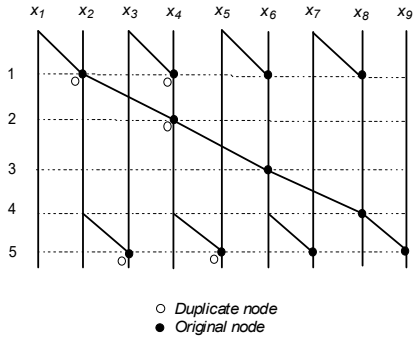

Figure 4 Prefix circuit FL-2(9).

Definition 2. A parallel prefix circuit is called an *FL-2 circuit*, if it has the same structure as *L*-circuit [7] in addition to one duplicate node for each of the original nodes in the first $\left\lceil \frac{m}{2} \right\rceil$ lines. ■

The proposed design in [8], *FL-2-circuit*, depends on having a duplicate operation node for each original operation node in the first few lines. A duplicate operation node will be used if the original operation node is faulty. Such duplicate node if used (if a node is faulty), will save a large number of idle operation nodes in later lines. For example, having a duplicate node in line 2 will save the whole circuit if the node in line 2 is faulty. Thus, *FL-circuit* will be equipped with a number of duplicate nodes for the first *m*/2 lines. If any original node in line $i<\left\lceil \frac{m}{2} \right\rceil$ goes faulty, then the duplicate node will be active and the whole circuit operate as originally designed.

If any original node in line $i> \left\lceil \frac{m}{2} \right\rceil$ goes faulty, then the circuit can generate the prefix computation for an input of width $h<i$ (Lemma 2). Figure 4 shows an *FL-2 circuit* of width 9. Consequently, we have the following theorem.

Theorem 6 [8]. *Let FL-2(m) be an FL-circuit of width m and depth d(FL-2). If there is a faulty node in line* $i<\left\lceil\frac{m}{2}\right\rceil$ *(resp.* $i\geq\left\lceil\frac{m}{2}\right\rceil$*), then FL-2(m) can generate prefix computation for input of width m (resp.* $h<i$*) in* $\left\lceil\frac{m}{2}\right\rceil$ *time steps. Moreover, FL-2(m) has* $(\frac{m-3h}{2}+2)$ *duplicate nodes.* ■

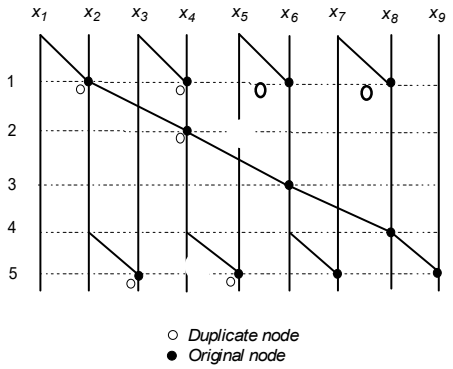

Figure 5 Prefix circuit FL-3(9).

Definition 3. A parallel prefix circuit is called an *FL-3 circuit*, if it has the same structure as *L*-circuit [7] in addition to one duplicate node for each of the original nodes in level 1. ■

The design, *FL-3-circuit*, depends on having a duplicate operation node for each original operation node in the first line. A duplicate operation node will be used if the original operation node is faulty. Thus, *FL-3* circuit will be equipped with a duplicate node for each original node in level 1. If any original node goes faulty, then the duplicate node will be active and the whole circuit operate as originally designed.
If any original node of level 1 goes faulty, then the circuit can generate the prefix computation as the original circuit. Figure 5 shows an *FL-3* circuit of width 9. Consequently, we have the following theorem.

Theorem 7. *Let FL-3(m) be an FL-3-circuit of width m and depth d(FL-3). If there is a faulty node in level 1, then FL-3(m) can generate prefix computation for the inputs as the original circuit in d(FL-3) time steps. Moreover, FL-3(m) has m/2 duplicate nodes.* ■

4. Conclusions

In this paper, we have investigated different classes of parallel prefix circuits in the existence of dense faulty nodes in terms of time penalty and number of idle nodes. The analysis shows that some classes of prefix circuits can handle faulty nodes better than others. Also, we have proposed new designs for prefix circuits that can handle faulty nodes. The idea is based on some duplicate nodes that can be active when some nodes go faulty. One direction to extend this work is to analyze different circuits. Other directions include proposing other designs that could handle faulty nodes.

References

[1] Jaja J (1992) An introduction to parallel algorithms. Addison Wesley, Redwood City, CA, USA.

[2] Leighton FT (1992) Introduction to parallel algorithms and architectures: arrays, trees, hypercubes. Morgan Kaufmann, San Mateo

[3] Vaidyanathan R, Trahan JL (2004) Dynamic reconfiguration: architectures and algorithms, vol 13.Springer Series in Computer Science. Kluwer Academic/Plenum Publishers, New York

[4] Lin Y-C, Hung L-L (2009) Fast problem-size-independent parallel prefix circuits. J Parallel Distrib Comput 69:382–388

[5] Hatem M El-Boghdadi (2013), A class of almost-optimal size-independent parallel prefix circuits. J Parallel Distrib Comput 73:888–894

[6] Hatem M. El-Boghdadi (2015), Dynamic-width reconfigurable parallel prefix circuits. The Journal of Supercomputing 71(4): 1177-1195

[7] Y.-C. Lin, C.-K. Liu, Finding optimal parallel prefix circuits with fan-out 2 in constant time, Inform. Process. Lett. 70 (4) (1999) 191–195.

[8] Hatem M. El-Boghdadi, Fazal Noor, Mostafa Mahmoud, "Analysis and Design of Parallel Prefix Circuits with Faulty Nodes", International Journal of Computer Science and Network Security, VOL.19 No.12, December 2019.

[9] Y.-C. Lin, C.-C. Shih, Optimal parallel prefix circuits with fan-out at most 4, in: Proc. 2nd

IASTED Int. Conf. on Parallel and Distributed Computing and Networks, Brisbane, Australia, 1998, pp. 312–317.

[10] T. D. Sawarkar, L. Chawle and N Narole, " Survey paper on Implementation of FPGA based Parallel Microprogrammed FIR Architecture", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 5 Issue 1, January