JOURNAL OF ELECTRONIC TESTING: Theory and Applications 19, 285–298, 2003

Easily Testable Cellular Carry Lookahead Adders

DIMITRIS GIZOPOULOS
Department of Informatics, University of Piraeus, 18534 Piraeus, Greece
dgizop@unipi.gr

MIHALIS PSARAKIS
Teletel S.A., 124 Kifissias Av, 11526 Athens, Greece
M.Psarakis@teletel.gr

ANTONIS PASCHALIS
Department of Informatics & Telecommunications, University of Athens, 15771 Athens, Greece
paschali@di.uoa.gr

YERVANT ZORIAN
Virage Logic, 46501 Landing Parkway, Fremont, CA 94538, USA
zorian@viragelogic.com

Received February 25, 2000; Revised January 28, 2003

Editor: J.P. Hayes

Abstract. Cellular Carry Lookahead (CLA) adders are systematically implemented in arithmetic units due to their regular, well-balanced structure. In terms of testability and with respect to the classical Cell Fault Model (CFM), cellular CLA adders have poor testability by construction. Design-for-testability (DFT) modifications for cellular CLA adders have been proposed in the literature providing complete CFM testability making the adders either level-testable or C-testable. These designs impose significant area and performance overheads. In this paper, we propose DFT modifications for cellular CLA adders to achieve complete CFM testability with special emphasis on the minimum impact in terms of area and performance. Complete CFM testability is achieved without adding any extra inputs to the adder, with very small area and performance overheads, thus providing a practical solution. The proposed DFT scheme requires only 1 extra output and it is not necessary to put the circuit in a special test mode, while the earlier schemes require the addition of 2 extra inputs to set the circuit in test mode. A rigorous proof of the linear-testability of the adder is given and a sufficient linear-sized test set is provided that guarantees 100% CFM fault coverage. Surprisingly, the size of the proposed linear-sized test set is, in most practical cases, comparable or even smaller than a logarithmic-sized test set proposed in the literature.

Keywords: cellular carry lookahead adders, design-for-testability, cell fault model, linear-testability

1. Introduction

High-performance processors require fast adders within their datapath, either as a separate module or as part of the Arithmetic Logic Unit (ALU). *Carry Lookahead* (*CLA*) adder architecture is the most common architecture for fast addition. Regular, cellular CLA adders [1] are preferred by many designers, since their enhanced regularity leads to efficient circuit implementation. Additionally, cellular CLA adders can be easily generated by automatic generators either as stand-alone fast adders or embedded in fast ALU modules.

The demand for high test quality in today's dense and complex processors (such as dense cellular structures) cannot be based on traditional fault models, such as the stuck-at fault model. To achieve high test quality, cellular adder architectures should adopt the combinational cell-level fault model, namely the *Cell Fault Model* (*CFM*) [2], which has been widely used [3–18]. A CFM-based test strategy is cell implementation independent, i.e. valid for any internal gate-level implementation of the cells. Unfortunately, cellular CLA adders due to their convergent-tree structure have serious testability problems when CFM is considered [3, 4].

Cellular CLA adder architecture is based on regular convergent tree structures to reduce the delay associated with carry generation. Lynch and Swartzlander [1] proposed efficient, modular CLA adders with high regularity, suitable for VLSI implementation. This architecture has been adopted in the *Advanced Micro Devices* AM29050 microprocessor [1, 19]. The adder design is based on a combination of a tree circuit (carry lookahead) for the generation of a subset of regularly distributed carry signals and one-dimensional arrays (connected in a ripple carry—RC fashion) which use the carry signals generated by the CLA logic for the computation of the final sum and carry outputs. This adder architecture, (termed spanning tree CLA adder [1], or cellular CLA adder which is the term that we use in this paper), combines the advantages of fast carry generation and propagation of a CLA adder with the great regularity of an RC adder. Thus, it provides a balance between circuit speed and complexity resulting in a very efficient CLA adder design.

When CFM testability is considered in this cellular adder architecture, serious problems appear due to poor testability of the internal generate/propagate logic, as it is demonstrated in the paper. In order to provide a complete CFM testability solution to these problems, Blanton and Hayes relied on their convergent tree circuit testing methodology [3, 4]. Based on this methodology, they proposed Design For Testability (DFT) modifications to the adder design to make it either *level-testable* (testable with a number of patterns growing linearly with the number of levels of the convergent tree) or *C-testable* (testable with a constant number of test patterns), with respect to CFM. These DFT modifications resolved the testability problems with the addition of two extra inputs that make the circuit to operate in test mode, when they are asserted. The CFM testability problems are solved putting the adder to operate in test mode. The major drawback of the designs proposed in [3, 4] is that the DFT modifications impose significant hardware and delay overheads over the original CLA adder. For this reason, there is a difficulty in the implementation of the DFT modifications of [3, 4] in practical arithmetic units.

In this paper, we propose a novel DFT modification to the cellular CLA adder design which makes it completely CFM testable. According to the proposed method, the adder design is CFM testable without requiring extra control inputs and thus can be completely tested in its normal mode. The adder's poor testability is resolved by properly modifying the functionality of the cells without affecting the overall circuit behavior and taking advantage of don't cares without adding extra cell inputs as in [3] and [4]. Few observability problems of the original design are resolved with the insertion of only one extra output.

The imposed area overhead of the proposed CFM testable cellular CLA adder design ranges between 5.29% and 6.53% for adders of lengths between 8 bits and 64 bits, while the performance overhead ranges between 4.59% and 8.97% for the same word lengths range. This is a significant improvement over the designs of [3, 4] and makes the proposed DFT modifications appropriate for a practical CFM testable implementation.

We prove the linear-testability of the proposed adder based on the theory of the testability of convergent trees of monoid operations given in [16, 17]. We also present a sufficient linear-sized test set that achieves 100% CFM testability. The test set size is $\mathbf{16(n-r-1)+2}$, and growths linearly to $\mathbf{n}$, the word length of the adder ($\mathbf{r}$ is the length of the ripple carry chains). Even for large CLA adders of up to 64-bits wide, the test set size is practically very small and provides the high quality of a classical cellular fault model. The length of the proposed linear test set is comparable or even smaller than the logarithmic-sized test set for the level-testable design of [3] for most practical word lengths.

The organization of the paper is as follows. In Section 2, the adopted fault model, CFM, is briefly presented. In Section 3, we describe the architecture of the cellular CLA adder design. In Section 4 the proposed DFT modifications are presented and the proposed linear-testable scheme is compared with previously proposed approaches [3, 4]. In Section 5, the linear-testability of the proposed adder is proven and the linear-sized test set is given. Section 6 concludes the paper.

2. Cellular Circuits Fault Modeling

Cell Fault Model (*CFM*) proposed in [2] is the classical and well adopted combinational fault model for testing cellular circuits. It is an implementation independent fault model which does not depend on the gate-level implementation of the circuit cells since it provides exhaustive application of all possible cell input combinations. At most one cell can be faulty at a time and any fault inside a faulty cell can happen as long as its function remains combinational. In order to be tested for CFM, each cell must receive all possible input combinations and all possible faulty cell outputs must be propagated to primary circuit outputs.

CFM is very important today, because of the rapid growth in the use of high-level synthesis, and the parallel increase in complexity and density of ICs. Using CFM as the target fault model, allows the test to be independent of the adopted synthesis tool and vendor standard cells library.

Particularly, arithmetic modules (adders, subtracters, shifters, multipliers, dividers etc.), due to their high regularity are designed in a very dense fashion. The use of a more comprehensive fault model than single stuck-at fault model is required to cover actual failure mechanisms of these dense designs. CFM covers all combinational faults that may happen in a single faulty cell. As a subset of all combinational faults, all single or multiple stuck-at faults inside the faulty cell are completely detected.

CFM has been extensively used in research so far for testing general array structures and specific arithmetic module architectures [3–18]. In [14] and [18] two methodologies are presented so that stuck-at fault based Automatic Test Pattern Generators and Fault Simulators can be used to perform automatic test generation and fault simulation with respect to CFM. This way, no special ATPGs or fault simulators need to be developed for CFM. In this paper, we have verified complete CFM testability using the approach of [14].

3. Cellular CLA Adder Architecture

In [1], the structure of the "spanning tree" carry lookahead adder (we call it a cellular carry lookahead adder) is introduced and a special layout-level implementation of the adder for the *Advanced Micro Devices* AM29050 microprocessor [19] is described. In [3, 4] a modified version of the adder is described based on cells of specific functionality. We describe the adder architecture based on the 16-bit cellular CLA adder depicted in Fig. 1. Its architecture is a combination of:

- Two convergent trees for the generation of three regularly distributed carry signals ($\mathbf{c}_4$, $\mathbf{c}_8$, $\mathbf{c}_{12}$). The trees consist of the well-known types of cells used in CLA adders for the extraction of generate/propagate logic: gp-cells (first level generate/propagate), GP-cells (higher levels generate/propagate) and C-cells (carry cells). The cells functions and their corresponding number of gate equivalents are given in Table 1 (we use a resolution of 0.5 gate equivalent so that the calculations following in this paper are as accurate as possible; for example, an inverter is equivalent to 0.5 gates, a 2-input NAND gate and a 2-input NOR gate are equivalent to 1.0 gates and a 2-input AND gate and a 2-input OR gate are equivalent to 1.5 gates).
  - ⇒ a **gp-cell** produces a pair of generate/propagate signals **($\mathbf{g}_i$, $\mathbf{p}_i$)** from a pair of operand bits **($\mathbf{a}_i$, $\mathbf{b}_i$)**.
  - ⇒ a **GP-cell** receives two generate/propagate signal pairs **($\mathbf{g}_i$, $\mathbf{p}_i$), ($\mathbf{g}_j$, $\mathbf{p}_j$)** and produce a pair of next level generate/propagate signals **($\mathbf{g}_{i:j}$, $\mathbf{p}_{i:j}$)**. For example, the leftmost GP-cell of the top level receives (g_1, p_1) and (g_2, p_2) and produces ($g_{1:2}$, $p_{1:2}$), while the leftmost GP-cell of the second level (the root GP-cell of the first convergent tree) receives ($g_{1:2}$, $p_{1:2}$) and ($g_{3:4}$, $p_{3:4}$) and produces ($g_{1:4}$, $p_{1:4}$).
  - ⇒ a **C-cell** receives a generate/propagate pair and a carry input signal to produce a carry output signal. For example the leftmost C-cell uses the generate/propagate pair ($\mathbf{g}_{1:4}$, $\mathbf{p}_{1:4}$) and the carry input signal $\mathbf{c}_0$ to produce the carry output signal $\mathbf{c}_4$.

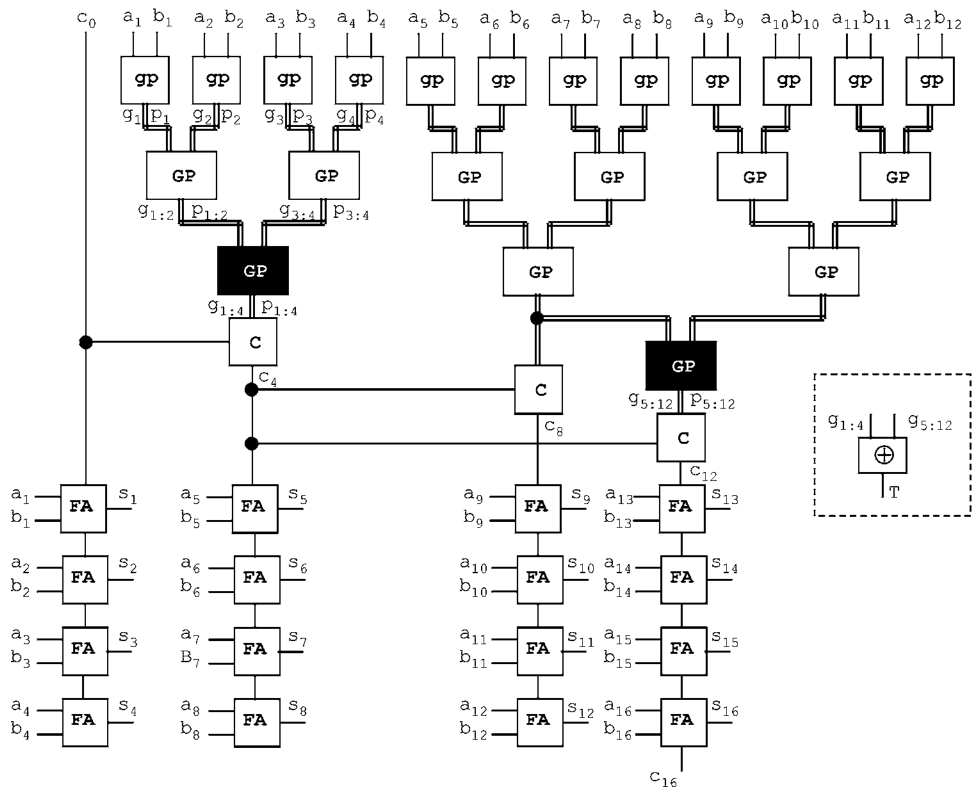

Fig. 1. 16-bit cellular CLA adder.

- Four one-dimensional arrays (ripple-carry adder chains) for the computation of the final sum and carry outputs. Each ripple carry adder takes the carry signal produced by the corresponding C-cell to compute the sum values. (The first stage ripple carry addition uses as carry input the carry input c_0 of the adder).

The XOR gate shown in the figure is a testability enhancement and will be explained in the related section.

In general, an $\mathbf{n}$-bit cellular CLA adder consists of $\mathbf{q}$ independent convergent trees. The $\mathbf{i}$-th convergent tree has $\mathbf{d_i}$ inputs ($\mathbf{i} = \mathbf{1}, \ldots, \mathbf{q}$) and consists of $\mathbf{d_i}$ gp-cells, and $\mathbf{d_i} - \mathbf{1}$ GP-cells. The number of levels of the GP-cells of the $\mathbf{i}$-th convergent tree is $\mathbf{log_2(d_i)}$ when a regular binary tree structure is used and for this reason $\mathbf{d_i}$ is selected to be a power of 2 whenever possible. The outputs of the root GP-cell of the $\mathbf{i}$-th convergent tree are connected to a C-cell (root C-cell) to generate the carry signal of the $(\mathbf{i} + \mathbf{1})$-th convergent tree. If a convergent tree is required to generate more than one carry signals, additional C-cells are connected to the outputs of intermediate level GP-cells of the tree (see for example the C-cell that generates signal c_8 in Fig. 1).

Table 1. Original cell functions.

Cell function	Gate equivalents
$p = a \oplus b$ $g = a\,b$	4.5
$P = p_1 p_0$ $G = g_1 + g_0 p_1$	4.0
$C = G + P\,c_{in}$	2.5

An $\mathbf{n}$-bit adder also consists of ripple carry chains of full adders of the same length, $\mathbf{r}$, that depends on the implementation technology and the tradeoff between the propagation delays of the convergent trees and the ripple carry chains [1]. All carries with distance $\mathbf{r}$ are generated by the C-cells, i.e. $\mathbf{c_{r \cdot i}}$ ($\mathbf{i} = 1, \ldots, \lceil \mathbf{n}/\mathbf{r} \rceil - 1$).

It follows from the above discussion that the adder consists of: $\mathbf{n} - \mathbf{r}$ gp-cells, $\mathbf{n} - \mathbf{r} - \mathbf{q}$ GP-cells, $\lceil \mathbf{n}/\mathbf{r} \rceil - \mathbf{1}$ C-cells and $\mathbf{n}$ FA-cells.

Thus, using the above equations and the gate equivalents of Table 1, the total number of gate equivalents for the original $\mathbf{n}$-bit adder is: $17{,}5\,\mathbf{n} + 2,5 \lceil \mathbf{n}/\mathbf{r} \rceil - 8{,}5\,\mathbf{r}$ $-4\,\mathbf{q} - 2,5$

The above equation will be used in the following for the evaluation of the proposed design and earlier approaches in terms of hardware overhead. Besides, the maximum propagation delay of the **n**-bit cellular CLA adder is approximately: D[gp-cell] + $\log_2(\mathbf{d_q})\cdot$ D[GP-cell] + D[C-cell] + $\mathbf{r}\cdot$ D[FA-cell] where D[X] is the propagation delay of module X which in any case depends on the implementation of the logic functions of the module in a specific technology library.

The worst-case propagation delays in any of the subsequent designs depend on the implementation library and thus we do not provide any formula, but instead we implement all different testable designs for a set of common word lengths, i.e. 8-bits, 16-bits, 32-bits and 64-bits.

4. Novel CFM Testable CLA Adder Design

In this section, we present our novel linear-testable cellular CLA adder design for complete CFM testability. The section begins with a demonstration of the testability problems of the cellular CLA adders. Then we discuss earlier attempts to increase CFM testability of the cellular CLA adders [3, 4, 15]. Finally, we propose our novel CFM testable cellular CLA adder design and compare it with the previous approaches.

4.1. Cellular CLA Adder Testability Problems

The original implementation of cellular CLA adders [1] has the following testability problems as described in [4], when CFM is the target fault model.

- The original function of the gp-cells (Table 1) is not *surjective*, i.e. there is one output combination (gp = 11) which *never appears at the cell outputs*. This missing combination causes CFM testability problems to the GP-cells whose inputs are directly fed by gp-cells outputs, since they cannot receive all possible input combinations required for complete CFM testability.
- The original GP-cell function (Table 1) requires the application of value gp = 11 to at least one of the two generate/propagate pairs in order to produce the output combination GP = 11. Thus the above mentioned CFM testability problem of GP-cells that directly receive inputs from gp-cells is transferred to *all* subsequent GP-cells which also cannot be completely tested with respect to CFM. Thus, no GP-cell can receive the input combination 11 in any of its generate/propagate pairs. This means that 7 of the total 16 input combinations of the GP-cells cannot be applied to them, i.e. at most 56.25% CFM testability can be achieved for the GP-cells.
- The original function of the C-cells is not able to distinguish all values appearing at their GP inputs, i.e. the input combinations GP = 10 and 11 produce the same output value independent on the value of c_{in}. Considering a fault that occurs on any gp-cell or GP-cell and affects the GP-input of a C-cell changing it from 10 to 11 or vice versa, the C-cell cannot propagate the fault to its output and thus the fault is not detected. This is therefore a fault propagation problem to be resolved.

DFT modifications have been proposed in [3] and [4] to make the cellular CLA adder design either *level-testable* (testable with a number of test vectors increasing linearly with the number of levels in the convergent tree) or *C-testable* (testable with a constant number of test vectors not depending on the size of the adder) with respect to CFM. In both cases the DFT modifications require the addition of two extra inputs to the design and the modification of the cell functions. Due to the use of extra inputs significant area and performance overheads are imposed, as we will see in the following subsections.

A *linear-testable* cellular CLA adder design (testable with a number of test vectors increasing linearly with the adder word length) has been proposed in [15]. This design does not provide complete CFM testability, since as we show in the following, its C-cell does not distinguish all values appearing at its GP-inputs and thus it has the same propagation problem as the original function. No observability alternative is proposed.

4.2. Level-Testable Cellular CLA Adder

In [3] the gp-cell functions are transformed into surjective ones with the addition of one extra cell input $\mathbf{z}_1$ to the gp-cells. One more extra input $\mathbf{z}_2$ is added to C-cells. Additionally, GP-cell functions are modified to achieve level-testability. As a result the proposed design is level-testable with respect to CFM with an excessive hardware overhead as we show below. Additionally, we note that the adder has to operate both in normal mode ($\mathbf{z}_1 = \mathbf{z}_2 = 0$) and test mode ($\mathbf{z}_1 = 1$ and/or $\mathbf{z}_2 = 1$), where in test mode it does not perform addition. The functions of the gp, GP and C cells used in the level-testable CLA adder proposed in [3] and their corresponding gate equivalents are:

Table 2. Cell functions for the level-testable CLA design of [3].

Cell function	Gate equivalents
$p = z_1'(a \oplus b) + z_1 b$ $g = z_1'(a\,b) + z_1 a$	10.5
$P = g_1' p_1 p_0 + g_1 p_1' g_0' + g_1 p_1' p_0$ $+ g_1 g_0' p_1 + g_1 p_1 g_0 p_0'$ $G = g_1 + g_0 p_0 + p_1 g_0$	17.0
$C = z_2'(G + P c_{in}) + z_2 P$	5.5

Using the equations of Section 3 and the gate equivalents of Table 2, the total number of gate equivalents for the level-testable adder is: 36,5 $\mathbf{n}$ + 5,5 $\lceil \mathbf{n/r} \rceil - 27{,}5\mathbf{r} - 17\mathbf{q} - 5{,}5$

Total circuit area (gate equivalents) and execution time (critical path) and the corresponding hardware and delay overheads for the 8 up to 64-bits versions of the level-testable design of [3] compared to the original design are presented in Table 3. *AMS* 0.8 *um* standard cell library [20] was used for the implementation of all adders. *Veritime* timing simulator from *Cadence* was used for the calculation of critical path delay.

The size of the test set proposed in [3] grows logarithmically with the adder word length **n** and is $\mathbf{64(log_2 n + 1) + 25}$.

4.3. C-Testable Cellular CLA Adder

Similarly to the level-testable design of [3], a C-testable design is proposed in [4] which adds two extra inputs $\mathbf{z_1}$, $\mathbf{z_2}$ to the gp-cells and C-cells, respectively. Additionally, GP-cell functions are modified to achieve C-testability. As a result the proposed design is C-testable with respect to CFM with an even more excessive hardware overhead than the linear-testable design. Also in this case, the adder has to operate both in normal mode ($\mathbf{z_1} = \mathbf{z_2} = 0$) and test mode ($\mathbf{z_1} = 1$ and/or $\mathbf{z_2} = 1$), where in the test mode it does not perform addition.

Table 4. Cell functions for the C-testable CLA design of [4].

Cell function	Gate equivalents
$p = z_1'(a \oplus b) + z_1 b$ $g = z_1'(a\,b) + z_1 a$	10.5
$P = g_1' p_1 p_0 + g_1 p_1 p_0' + g_1 p_1' g_0 p_0$ $G = g_1 p_1 g_0' + g_1 g_0' p_0 + g_1' g_0 p_0$ $+ p_1 g_0 p_0' + g_1 g_0 p_0'$	19.5
$C = z_2'(G + P c_{in}) + z_2 P$	5.5

The functions of the gp, GP and C cells used in the C-testable cellular CLA adder proposed in [4] and their corresponding gate equivalents are:

Using the equations of Section 3 and the gate equivalents of Table 4, the total number of gate equivalents for the C-testable adder is: 39 $\mathbf{n}$ + 5,5 $\lceil \mathbf{n/r} \rceil - 30\mathbf{r} - 19{,}5\mathbf{q} - 5{,}5$

Total circuit area and execution time and the corresponding hardware and delay overheads for the 8-bits, 16-bits, 32-bits and 64-bits versions of the C-testable design of [4] compared again to the original design are presented in Table 5. The hardware and delay overheads imposed by both the level-testable and C-testable designs are prohibitively large and cannot be adopted in a well-optimised CLA adder core in terms of area and speed.

We next present our novel linear-testable CLA adder design which achieves complete CFM testability without excessive hardware and delay overheads and thus can be used in practical CLA adder designs.

4.4. Proposed Linear-Testable CLA Adder Design with Complete CFM Testability

In order to overcome the practical problem of the earlier modified designs, we introduce a novel CLA adder

Table 3. Overheads for the level-testable CLA design of [3].

	Area (g.e.)			Speed (ns)		
Size (bits)	Original	Level-testable	Overhead (%)	Original	Level-testable	Overhead (%)
8	122,5	219,5	79,18	5,91	9,68	63,79
16	245,5	456,5	85,95	8,73	13,54	55,10
32	491,5	930,5	89,31	13,92	19,77	42,03
64	1053,5	2086,5	98,05	18,27	25,05	37,11

Table 5. Overheads for the C-testable CLA design of [4].

	Area (g.e.)			Speed (ns)		
Size (bits)	Original	C-testable	Overhead (%)	Original	C-testable	Overhead (%)
8	122,5	229,5	87,35	5,91	9,62	62,77
16	245,5	481,5	96,13	8,73	13,48	54,41
32	491,5	985,5	100,50	13,92	19,71	41,59
64	1053,5	2216,5	110,39	18,27	24,99	36,78

Table 6. Cell functions for the novel linear-testable CLA design.

Cell function	Gate equivalents
$p = a \oplus b$ $g = a$	3.0
$P = p_1 p_0$ $G = g_1 p_1' + g_0 p_1 + g_1 p_0$	6.5
$C = GP' + Pc_{in}$	3.5

design which solves the CFM testability problems of the cellular CLA adders. The proposed design is superior over the two earlier approaches in two key points:

(a) It can be tested with respect to CFM operating in its *normal operating mode* (no special test mode is required as in [3] and [4]).
(b) The hardware and delay overheads imposed by the DFT modifications that we propose are very small and in all cases significantly reduced compared to the designs of [3, 4].

The functions of the cells in the proposed design are modified accordingly in order to solve the CFM testability problems previously mentioned *without adding extra inputs* to the modified cells which is not the case with the modifications proposed in [3, 4]. The proposed design takes advantage of the missing combination (gp = 11) and modifies the functions of the cells as follows (Table 6):

The new cell functions have the following characteristics:

- Table 7 shows the truth table of the modified gp-cell function. The modified gp-cell function is *surjective* since it produces all 4 possible output combinations without adding any extra circuit input. Consequently, GP-cells directly connected to the gp-cells outputs can receive all possible input combinations. The gp-cell function we use is *propagate-dominant*, i.e. when both g and p are 1 (gp = 11, the missing combination which now appears) the meaning is *propagate and not generate*. In other words, states gp = 01 and 11 are equivalent and both denote propagate.
- Table 8 shows the truth table of the modified GP-cell function, where g_1 p_1 is the most significant generate/propagate pair and g_0 p_0 is the least significant

Table 7. gp-cell truth table.

a	b	g	P
0	0	0	0
0	1	0	1
1	0	**1**	1
1	1	1	0

Table 8. GP-cell truth table.

g_0	p_0	g_1	P_1	G	P
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	1	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	1
0	1	1	0	1	0
0	1	1	1	1	1
1	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	1	1
1	1	1	0	1	0
1	1	1	1	1	1

pair. The GP-cell function is modified to preserve the equivalence between the gp states 01 and 11. It can be easily proved that the GP-cell function is also surjective. Thus, all the subsequent GP-cells can be fully tested with respect to CFM. Also, modified GP-cell function propagates any faulty input to its outputs as it can be easily verified from Table 8. Let's consider an error generated in a single gp-cell or GP-cell. It results in a faulty generate/propagate pair at the inputs of a GP-cell. The error can be propagated to the cell outputs by setting the side generate/propagate pair to the value 01 (Table 8). This way, the error is propagated to at least one GP-cells tree output, regardless of the size of the tree. The 01 value 01 of a generate/propagate pair is the identity (or neutral) element of the GP-cell function which is a monoid operation [16, 17]. The existence of such an identity element guarantees the propagation of faulty cell outputs and provides the linear-testability of such convergent trees. We elaborate further on the monoid operation of the GP-cells and the linear-testability in the next section.

- Table 9 shows the truth table of the modified C-cell function. The modified C-cell function holds a similar fault propagation problem as the original cell function since it cannot distinguish between the equivalent states gp = 01 and 11 (the original function does not distinguish between 10 and 11). This problem can be easily overcome by monitoring the G outputs of only the root GP-cells using one extra output T, as shown in Fig. 1. Root GP-cells are the GP-cells whose outputs are only connected to C-cells inputs. The root GP-cells (for any size of the cellular CLA adder the number of root GP-cells is **q** i.e. the number of the independent convergent trees) are shown shaded in Fig. 1. The extra output T is the XOR function of the G outputs of the root **q** GP-cells. The generate/propagate pair of only one root GP-cell can present an error due to a single faulty gp-cell or GP-cell since every convergent tree of GP-cells ends to a unique root GP-cell. Thus the XOR function will propagate to the extra output T all the faults that result to a faulty G output of a root GP-cell and consequently all the faults that change the GP outputs of a root GP cell to 01 instead of 11 and vice versa.

Table 9. C-cell truth table.

G	P	c_{in}	c_{out}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Before proceeding to the calculation of the circuit area and execution time of the proposed adder, we will prove that it still functions correctly as an adder with the new functions of its cells. This proof is necessary in the case of the proposed design since it does not have a special test mode as the previously proposed designs and its improved testability is due to the use of the new functions.

Theorem 1. *The cellular design of Fig.* 1 *implementing the cell functions of Table* 6 *correctly performs addition.*

Proof: Table 6 shows the proposed cell functions for the gp-cells, GP-cells and C-cells. Therefore, in order to prove that the new design is still an adder it is sufficient to prove that the carry signals driven from the convergent trees to the ripple-carry adder chains are the same as in the adder design with the original cell functions.

The new gp-cell function (truth table given in Table 7) is propagate-dominant, meaning that the gp = 11 combination means for the new design *propagate*. Thus, gp-cell outputs 01 and 11 both denote carry propagate. In the original gp-cell function combination 11 never appears, so the only combination of gp outputs denoting propagate is 01.

The new GP-cell function (truth table given in Table 8) preserves the equivalence of gp states 01 and 11 which both denote propagate. Therefore, when a root GP-cell gives at its outputs the GP combination 01 or 11 this means that a carry is propagated at this stage. In the original functions a root GP-cell would never produce a 11 output. A carry propagate in the original adder is denoted by a root GP-cell giving a 01 output, while in the new design it is denoted by a root GP-cell output of either 01 or 11. What is left is the new C-cell function to equivalently deal with the two GP-cell output combinations 01 and 11.

The new C-cell function as shown in its truth table of Table 9 produces exactly the same carry output

Table 10. Overheads for the proposed linear-testable CLA design.

	Area (g.e.)			Speed (ns)		
Size (bits)	Original	Linear-testable	Overhead	Original	Linear-testable	Overhead
8	122,5	130,5	6,53%	5,91	6,44	8,97%
16	245,5	259,5	5,70%	8,73	9,50	8,82%
32	491,5	517,5	5,29%	13,92	14,92	7,18%
64	1053,5	1118,5	6,17%	18,27	19,11	4,59%

when GP is equal to 01 and when it is equal to 11. In the original function of the C-cell no 11 GP input appears.

Therefore, the carry signals that are produced in the original design and the new design by the C-cells are in all cases the same. Since, these signals drive the same ripple adder chains, the new adder design correctly performs addition. ■

Using the equations of Section 3 and the gate equivalents of Table 6, the total number of gate equivalents for the proposed novel linear-testable adder is (the XOR gate is also included):

$$18,5\,\mathbf{n} + 3,5\lceil \mathbf{n}/\mathbf{r}\rceil - 9,5\,\mathbf{r} - 2,5\,\mathbf{q} - 7,5$$

We present circuit area and execution time and the corresponding hardware and delay overhead in Table 10 for the 8-bits, 16-bits, 32-bits and 64-bits versions of the proposed design. It is obvious that the hardware and delay overheads of the proposed design are much less than the overheads imposed in the designs of [3, 4].

This significant improvement in terms of overheads is due to the fact that the missing state 11 from the outputs of the original gp-cells now participates *in the normal operation* of the circuit and not only in special test mode that requires the addition of extra inputs as in [3, 4]. This fact gives us the capability of using the original P function for the GP-cells and to add only a very small overhead to the G function.

In [15], an attempt for a linear-testable cellular CLA adder design is presented. The gp-cells are modified as in [3, 4] by using an extra input $\mathbf{z_1}$, the original GP-cell function is used and the C-cell function of Table 9 is also used. This adder design *is not completely testable with respect to CFM* since, as we analysed above, the C-cell function of Table 9 is by construction not able to distinguish between the GP values 01 and 11 and thus any fault causing a C-cell GP-inputs to change from 01 to 11 and vice versa cannot be detected.

5. CLA Adder Linear Test Set

We first prove that the proposed CLA adder is linear-testable with respect to CFM based on the testability theory of convergent tree structures of *monoid* operations [16, 17]. Then we give a sufficient linear test set.

A *finite* set M and an *associative* operation $* : M \times M \rightarrow M$ constitute a *finite monoid* where there exists an *identity* (or *neutral*) *element* e so that for every element m of M it is satisfied that: $m * e = e * m = m$.

In [17] the testability of two types of convergent tree structures of monoid operations is studied. Conditions are provided that characterize the tree structures as C-testable, level-testable and linear-testable with constant, logarithmic and linear test complexity, respectively. The two types of convergent trees are the *expression evaluation* (*EEV*) *tree* and the *parallel prefix computation* (*PPC*) *tree.* The EEV tree calculates the expression $e_n = m_1 * \cdots * m_n$ where $m_i (i = 1 \ldots n)$ are elements of M, i.e. it is a single output convergent tree structure. The PPC tree on the other side, calculates *all* the expressions $e_i = m_1 * \cdots * m_i$ where $m_i (i = 1 \ldots n)$ are elements of M, i.e. it is a multiple output convergent tree structure.

The convergent tree of GP-cells in the cellular CLA adder studied in this paper is a subset of a PPC tree since not all GP expressions are outputs of the tree. Characterizing the testability of the PPC tree we can characterize also the testability of its subset which has lower observability that the PPC tree.

In the following, we first formally prove that the set of GP signal pairs $M = \{00, 01, 10, 11\}$ supplied with the GP-cell function of the proposed adder design (see Table 8) constitute a monoid and then we will prove that a convergent tree of GP-cells of the adder is linear-testable with respect to CFM since it does not satisfy the conditions described in [17] for C-testability and level-testability.

Theorem 2. *The set of GP signal pairs $M = \{00, 01, 10, 11\}$ and the GP-cell function of the proposed adder design constitute a monoid.*

Proof: As it is mentioned earlier a monoid is characterized by the existence of an identity element and the associativity property. The identity element of the GP-cell function is the element 01, as it can be easily verified from Table 8. The P operation of the GP-cell function is the boolean AND operation and therefore it is associative. We prove below that G operation is also associative. The G operation of the GP-cell function is given in Table 6.

Let us consider 3 elements of M (pairs of gp signals at the first level): g_2p_2, g_1p_1, g_0p_0. We do not use capital G and P for the cell inputs to avoid confusion between GP-cell inputs and outputs and to be consistent with the notation of Tables 6, 7 and 8. What we need to prove is that $G_{(2:1):0} = G_{2:(1:0)}$. Note that from Table 6: $G_{j:i} = g_j\, p_j' + g_i\, p_j + g_j\, p_i$ (when $j > i$). Therefore: $G_{(2:1):0} = G_{2:1}\, P_{2:1}' + g_0\, P_{2:1} + G_{2:1}\, p_0 = (g_2\, p_2' + g_1\, p_2 + g_2\, p_1)\,(p_2\, p_1)' + g_0\,(p_2\, p_1) + (g_2\, p_2' + g_1\, p_2 + g_2\, p_1)\, p_0 = (g_2\, p_2' + g_1\, p_2 + g_2\, p_1)\,(p_2' + p_1' + p_0) + g_0\, p_2\, p_1 = \mathbf{g_2(p_2' + p_1p_0) + g_1(p_2\, p_1' + p_2\, p_0) + g_0p_2\, p_1}$

Also: $G_{2:(1:0)} = g_2\, p_2' + G_{1:0}\, p_2 + g_2\, P_{1:0} = g_2\, p_2' + (g_1\, p_1' + g_0\, p_1 + g_1\, p_0)\, p_2 + g_2\,(p_1\, p_0) = \mathbf{g_2\,(p_2' + p_1\, p_0) + g_1(p_2\, p_1' + p_2\, p_0) + g_0p_2\, p_1}$

Therefore, the associativity property holds for both G and P operations and thus the set M supplied with the GP-cell function constitute a monoid. ■

According to [17] a convergent tree of monoid operations is, in the worst case, linear-testable. If conditions are satisfied [17] such tree may be either C-testable or level-testable. We show that the proposed GP-cell function does not satisfy the conditions given in [17]. First, we show that the convergent tree of GP-cells using the GP-cell function of Table 8 is not C-testable.

According to [17] a convergent tree of monoid operations is C-testable with respect to CFM if and only if M supplied with the monoid operation is a group. In other words, each element m of set M has an inverse element m^{-1} such that $m * m^{-1} = m^{-1} * m = e$. We easily verify from Table 8 that only element 01 of set M has an inverse element (it is inverse of itself). Elements 00, 10 and 11 do not have an inverse element.

Secondly, the convergent tree of GP-cells using the GP-cell function of Table 8 is not level-testable.

As we have mentioned the GP-cells tree of the adder is a subset of a complete PPC type convergent tree. Therefore, if we prove that a complete PPC type convergent tree is not level-testable then its subset (which has a restricted observability) cannot be level-testable as well.

According to [17] a convergent tree of monoid operations of the PPC type is level-testable with respect to CFM if and only if M supplied with the monoid operation is not a group and for all elements m, m' of M, $m \neq m'$ there exists an element y of M such that:

(i) $m * y = (m * y)^2$
(ii) $m * y * m \neq m * y * m'$

As it was shown above M is not a group.

Let us consider the elements $m = 00$ and $m' = 01$. Condition (i) is satisfied for any y of M, but condition (ii) cannot be satisfied for any y of M.

Therefore, there exists at least one pair of elements m, m' of M (the pair 00, 01) for which the condition for level-testability is not satisfied. Thus, the complete PPC type tree is not level-testable and as a consequence its subset, the GP-cells tree of the cellular CLA adder is not level-testable.

In what follows, we provide a sufficient linear-sized test set for the CLA adder. The basic idea behind the proposed linear test set is the way to control the inputs of the GP-cells and observe their faulty outputs, using the identity element 01 of the monoid operation described above. The following two Theorems support the effectiveness of the proposed linear-sized test set.

Theorem 3. *Assume a CLA convergent tree with inputs $\boldsymbol{a_k} \ldots \boldsymbol{a_l}$ and $\boldsymbol{b_k} \ldots \boldsymbol{b_l}$ where $\boldsymbol{k} < l$. A test set that applies all 16 different input combinations to the 4 inputs $(\boldsymbol{a_i}, \boldsymbol{b_i}, \boldsymbol{a_{i+1}}, \boldsymbol{b_{i+1}})$ of each pair of adjacent gp-cells, for every $\boldsymbol{i}$ between $\boldsymbol{k}$ and $\boldsymbol{l} - 1$ while setting the inputs of the remaining gp-cells to 01, completely tests the convergent tree with respect to CFM, i.e.:*

- *applies to all the gp and GP-cells of the convergent tree all their possible cell input combinations and*
- *any faults occuring in a single cell are propagated to the outputs of the tree (outputs of root GP-cell)*

Proof: For each GP-cell of the convergent tree, there is a unique pair of adjacent gp-cells that have a path (either directly or through other GP-cells) towards the two

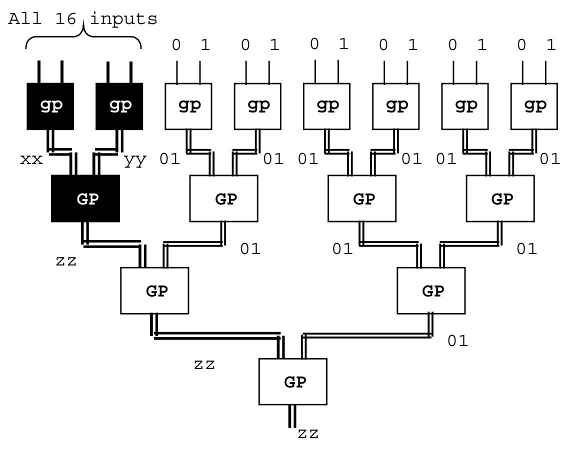

Fig. 2. Testing of the leftmost GP-cell of a tree.

gp-inputs pairs of GP-cell. For example, see Fig. 2 and Fig. 3 which depict the largest (rightmost) convergent tree for the 16-bit cellular CLA adder of Fig. 1.

In Fig. 2 the two shaded gp-cells–that have direct access to the inputs of the shaded GP cell—receive all 16 input combinations. Since gp-cell function is surjective they produce all 16 possible combinations to their outputs and thus the inputs of the shaded GP-cell. By applying the identity element 01 of the monoid operation (GP-function) to the remaining gp-cells, the propagation of any faults of the shaded gp or GP-cells to the outputs of the tree (the outputs of the root GP-cell) is guaranteed. Similarly, in Fig. 3, the inputs of shaded GP-cell (root) are fully controlled by the inputs of the shaded (adjacent) gp-cells if the inputs of the remaining gp-cells are set to the identity element 01. Thus the

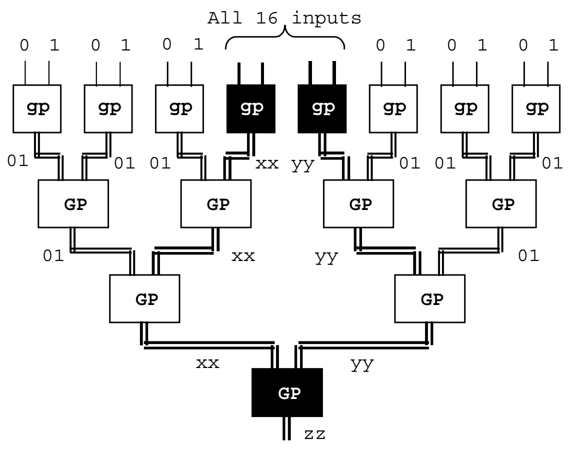

Fig. 3. Testing of the root GP-cell of a tree.

shaded GP-cell along with the two adjacent gp-cells are tested together by applying all 16 combinations to the 4 inputs of the shaded gp-cells. Concluding, to test the gp-cells and GP-cells of a convergent tree for CFM it is sufficient that all the adjacent gp-cells receive all 16 input combinations while the remaining gp-cells receive the identity element 01. ■

Theorem 4. *Assume the proposed cellular CLA adder. When the linear test set described in Theorem* 3 *is applied to each convergent tree of the adder, any fault that occurs in the convergent tree is propagated either in the primary outputs of the adder or the extra output T* (*the* XOR *function of the G outputs of the root GP-cells of the convergent trees*) *if the input* $\boldsymbol{c_{in}}$ *of the C-cell that follows the root GP-cell of the tree receives the value* $\boldsymbol{a_i}' \boldsymbol{a_{i+1}}' + \boldsymbol{a_{i+1}}' \boldsymbol{b_{i+1}}' + \boldsymbol{a_i}' \boldsymbol{b_i}' \boldsymbol{b_{i+1}}'$ (*where* $\boldsymbol{a_i}, \boldsymbol{a_{i+1}}, \boldsymbol{b_i}, \boldsymbol{b_{i+1}}$ *are the* 4 *inputs of adjacent gp-cells*).

Proof: Let us consider the different scenaria for the propagation of the fault from the root GP-cell outputs to either the primary outputs of the adder or extra output T (a fault exists either in a gp-cell or in a GP-cell):

(a) The error free output of the root GP-cell is 00:

If a fault changes the output of root GP-cell to 10 or 11, then the fault will be propagated to the extra output T, since only one of the G outputs of the root GP-cells is affected by the fault.

If a fault changes the output of root GP-cell to 01, then the fault cannot be detected in T. Such a fault will be propagated to the output of the C-cell that follows the root GP-cell and consequently to the primary outputs of the adder via a subset of full adder cells if $c_{in} = 1$, since $C = G\,P' + P\,c_{in}$ and $G = 0$. Therefore if the root GP-cell is 00 then $c_{in} = 1$ during testing.

(b) The error free output of the root GP-cell is 01:

If a fault changes the output of root GP-cell to 10 or 11, then the fault will be propagated to the extra output T, since only one of the G outputs of the root GP-cells is affected by the fault.

If a fault changes the output of root GP-cell to 00, then the fault cannot be detected in T. Such a fault will be propagated to the output of the C-cell that follows the root GP-cell and consequently to the primary outputs of the adder via a subset of full adder cells if $c_{in} = 1$, since $C = G\,P' + P\,c_{in}$ and $G = 0$. Therefore if the root GP-cell is 01 then $c_{in} = 1$ during testing.

(c) The error free output of the root GP-cell is 10:

Table 11. Value of the c_{in} of the C-cell.

g_i p_i g_{i+1} p_{i+1}	G P	a_i b_i a_{i+1} b_{i+1}	c_{in}
0 0 0 0	0 0	0 0 0 0	1
0 0 0 1	0 0	0 0 0 1	1
0 0 1 1	0 0	0 0 1 0	1
0 0 1 0	1 0	0 0 1 1	0
0 1 0 0	0 0	0 1 0 0	1
0 1 0 1	0 1	0 1 0 1	1
0 1 1 1	1 1	0 1 1 0	0
0 1 1 0	1 0	0 1 1 1	0
1 1 0 0	0 0	1 0 0 0	1
1 1 0 1	1 1	1 0 0 1	0
1 1 1 1	1 1	1 0 1 0	0
1 1 1 0	1 0	1 0 1 1	0
1 0 0 0	0 0	1 1 0 0	1
1 0 0 1	1 0	1 1 0 1	0
1 0 1 1	1 0	1 1 1 0	0
1 0 1 0	1 0	1 1 1 1	0

If a fault changes the output of root GP-cell to 00 or 01, then the fault will be propagated to the extra output T, since only one of the G outputs of the root GP-cells is affected by the fault.

If a fault changes the output of root GP-cell to 11, then the fault cannot be detected in T. Such a fault will be propagated to the output of the C-cell that follows the root GP-cell and consequently to the primary outputs of the adder via a subset of full adder cells if $c_{in} = 0$, since $C = G P' + P c_{in}$ and $G = 1$. Therefore if the root GP-cell is 10 then $c_{in} = 0$ during testing.

(d) The error free output of the root GP-cell is 11:

If a fault changes the output of root GP-cell to 00 or 01, then the fault will be propagated to the extra output T, since only one of the G outputs of the root GP-cells is affected by the fault.

If a fault changes the output of root GP-cell to 10, then the fault cannot be detected in T. Such a fault will be propagated to the output of the C-cell that follows the root GP-cell and consequently to the primary outputs of the adder via a subset of full adder cells if $c_{in} = 0$, since $C = G P' + P c_{in}$ and $G = 1$. Therefore if the root GP-cell is 11 then $c_{in} = 0$ during testing.

In Table 11 we summarize the requirements. Column 1 shows inputs of the tested GP-cell. Column 2 shows outputs of the tested GP-cell and thus the outputs of the root GP-cell. In column 3 the inputs of the adjacent gp-cells required to generate the corresponding inputs to the tested GP-cell are shown. In column 4 the values of c_{in} according to the previous reasoning are given. From columns 3 and 4 after minimization of the logic function it is derived that $\mathbf{c_{in}} = \mathbf{a_i}' \mathbf{a_{i+1}}' + \mathbf{a_{i+1}}' \mathbf{b_{i+1}}' + \mathbf{a_i}' \mathbf{b_i}' \mathbf{b}'_{\mathbf{i+1}}$, where $\mathbf{a_i}$ $\mathbf{b_i a_{i+1}}$ $\mathbf{b_{i+1}}$ are the inputs of the adjacent gp-cells. This provides an easy way of calculating the necessary $\mathbf{c_{in}}$ value for the proposed test set. ■

Based on Theorems 3 and 4 we conclude that in order to test the gp-cells and GP-cells of each convergent tree with respect to CFM it is sufficient that all the adjacent gp-cells receive all 16 input combinations, the remaining gp-cells receive the value 01 and c_{in} input of the C-cell takes value $\mathbf{a_i}' \mathbf{a_{i+1}}' + \mathbf{a_{i+1}}' \mathbf{b_{i+1}}' + \mathbf{a_i}' \mathbf{b_i}' \mathbf{b_{i+1}}'$.

Furthermore, in order to test *all* convergent trees with respect to CFM, it is sufficient that all the adjacent gp-cells receive all 16 input combinations, the remaining gp-cells receive the value 01 and the $\mathbf{c_0}$ input of the C-cell takes the value $\mathbf{a_i}' \mathbf{a_{i+1}}' + \mathbf{a_{i+1}}' \mathbf{b_{i+1}}' + \mathbf{a_i}' \mathbf{b_i}' \mathbf{b_{i+1}}'$.

If the tested cell belongs to the left convergent tree then $\mathbf{c_0}$ must take the value $\mathbf{a_i}' \mathbf{a_{i+1}}' + \mathbf{a_{i+1}}' \mathbf{b_{i+1}}' + \mathbf{a_i}' \mathbf{b_i}' \mathbf{b_{i+1}}'$. If the tested cell belongs to any of the following convergent trees then the $\mathbf{c_{in}}$ input of the C-cell of this tree must take the above value. This is accomplished by simply setting input $\mathbf{c_0}$ to this value since the value will be propagated through the intermediate C-cells since $c_{in} = G P' + P c_0$ and $GP = 01$.

We can easily verify that ***all*** remaining cells of the adder are completely tested for CFM when the linear test is applied to the adder with the addition of two extra test vectors described in the following. Particularly:

- From Theorem 4 it is verified that the C-cells receive the input combinations $(G, P, c_{in}) = (0, 0, 1), (0, 1, 1), (1, 0, 0), (1, 1, 0)$ during testing of the their corresponding convergent GP-cell tree. When the 16 input combinations are applied to the following adjacent gp-cells: the rightmost gp-cell of the leftmost tree and the leftmost gp-cell of the rightmost tree, two of the remaining input combinations, i.e. $(G, P, c_{in}) = (0, 1, 0), (1, 1, 1)$ are applied to the C-cells. In order to apply the remaining two input combinations $(G, P, c_{in}) = (0, 0, 0), (1, 0, 1)$ to all C-cells, we only need to apply two extra input patterns to the adder:

the all 0's input to both operands with $c_{in} = 0$ and the all 1's input to both operands with $c_{in} = 1$.

- The XOR gate implementing the function T is also completely tested. For example, when the 16 input combinations are applied to the following adjacent gp-cells: the rightmost gp-cell of the leftmost tree and the leftmost gp-cell of the rightmost tree, the XOR gate receives all 4 possible input combinations.
- The full adder cells in the CLA adder are tested with all 8 input combinations, since all pairs of adjacent FA cells receive all 16 inputs sufficient for complete CFM coverage as it shown earlier in the literature. We have also verified that if each ripple-carry full adder chain is replaced by a carry-select adder (two parallel ripple-carry chains, one calculating the sum bits assuming a 0 carry-in for the chain, and one calculating the sum bits assuming a 1 carry-in for the chain, and a final multiplexing stage to select the sum bits based on the actual carry-in signal), then the cells of this structure (half and full adders and multiplexers) are also completely tested with respect to CFM.

As a conclusion, in order to test an entire n-bit cellular CLA adder (with inputs $\mathbf{a_1a_2\ldots a_n}$, $\mathbf{b_1b_2\ldots b_n}$ and $\mathbf{c_0}$) with respect to CFM, it is sufficient that each 4-bits group of operand inputs $\mathbf{a_i}$ $\mathbf{b_ia_{i+1}}$ $\mathbf{b_{i+1}}$ for $i = 1\ldots \mathbf{n} - \mathbf{r} - 1$, receives all 16 different values while at the same time input $\mathbf{c_0}$ receives the values shown in Table 11, bits $\mathbf{a_kb_k}$ with $k \neq$ i and $k \neq$ i + 1 and $k \leq \mathbf{n} - \mathbf{r}$ receive values $\mathbf{a_k} = 0$ and $\mathbf{b_k} = 1$, respectively and bits $\mathbf{a_mb_m}$ with m $> \mathbf{n} - \mathbf{r}$ bits (which are the inputs of the full adder cells of the last chain) receive the same values as the tested 4-bit group ($\mathbf{a_i}$ $\mathbf{b_i}$ $\mathbf{a_{i+1}}$ $\mathbf{b_{i+1}}$). Additionally, two extra test vectors, the all 0's input with $\mathbf{c_{in}} = 0$ and the all 1's input with $\mathbf{c_{in}} = 1$ must be applied to the adder.

Therefore, the test set size is linear to the word length $\mathbf{n}$ of the adder and equal to $\mathbf{16(n - r - 1) + 2}$. Compared to the logarithmic-sized test set for the level-testable adder design of [3], our linear-sized test set is surprisingly comparable or even smaller for practical word lengths of the adder. This is due to the fact that in our design, $\mathbf{n}$ is multiplied only by a factor of 16 while in the test set of [3] the multiplication factor is 64 ($\mathbf{64\,(log_2n + 1) + 25}$). Table 12 shows the test set sizes for most common word lengths.

The complete CFM testability of the cellular CLA adder for the proposed linear-sized test set was additionally verified with the Verifault-XL single stuck-at fault simulator, according to a technique that allows evaluation of CFM testability of a cellular circuit using classical stuck-at fault simulators [14].

Table 12. Test set sizes comparison.

Size (bits)	Linear test set size (Proposed)	Logarithmic test set size [3]
8	82	281
16	178	345
32	370	409
64	882	473

6. Conclusion

We have proposed a completely CFM testable cellular CLA adder which is tested without setting the circuit in special test mode. No extra inputs are required to achieve complete CFM testability. The required DFT modifications impose significantly less hardware and delay overhead than the modifications of the designs proposed earlier in the literature and therefore the new design can be clearly applied to practical designs. We have shown that the new design is linear-testable with respect to CFM, based on the theory of the testability of finite monoids. A linear-sized test set was also proposed for the cellular CLA adder which is, surprisingly, comparable or even smaller that the logarithmic-sized test set proposed in [3] for practical word lengths.

References

1. T. Lynch and E.E. Swartzlander, "A Spanning Tree Carry Lookahead Adder," *IEEE Transactions on Computers*, vol. 41, no. 8, pp. 931–939, Aug. 1992.
2. W.H. Kautz, "Testing for Faults in Cellular Logic Arrays," in *Proc. 8th Annual Symposium Switching and Automata Theory*, 1967, pp. 161–174.
3. R.D. Blanton and J.P. Hayes, "Design of a Fast, Easily Testable ALU," in *Proc. of the 14th IEEE VLSI Test Symposium*, 1996, pp. 9–16.
4. R.D. Blanton and J.P. Hayes, "Testability of Convergent Tree Circuits," *IEEE Transactions on Computers*, vol. 45, no. 8, pp. 950–963, Aug. 1996.
5. D. Gizopoulos, A. Paschalis, and Y. Zorian, "Effective Built-In Self-Test for Booth Multipliers," *IEEE Design & Test of Computers*, vol. 15, no. 3, pp. 105–111, July–Sept. 1999.
6. D. Gizopoulos, A. Paschalis, and Y. Zorian, "An Effective Built-In Self-Test Scheme for Array Multipliers," *IEEE Transactions on Computers*, vol. 48, no. 9, pp. 936–950, Sept. 1999.