A Fast Optimal Robust Path Delay Fault Testable Adder

Bernd Becker[1] Rolf Drechsler[1] Rolf Krieger[2] Sudhakar M. Reddy[3]*

[1]Institute of Computer Science
Albert-Ludwigs-University
79110 Freiburg i.Br., Germany

[2]Department of Computer Science
Johann Wolfgang Goethe-University
60054 Frankfurt a.M., Germany

[3]Department of ECE
University of Iowa
Iowa City, IA 52242, USA

Abstract

In this paper we explore the test complexity of the adder function with respect to the robust path delay fault model. A lower bound of $\Omega(n^2)$ *for the cardinality of a complete test set for a combinational* n*-bit adder is proven. This result is valid for any adder design known until now. In addition we present a fast* $O(\sqrt{n})$*-time adder that is fully robust path delay fault testable with a test set of size* $\Theta(n^2)$*.*

1 Introduction

Even if chips are correctly designed, a non negligible fraction of them will have physical defects caused by imperfections occurring during the manufacturing process (e.g., open connections induced by dust particles). Therefore, there has to be a test phase in which 'production' verification is performed, i.e., in which the 'good' chips are sorted from the 'bad' ones. For a detailed treatment of the topic see [2]. Due to the variety of possible defects restrictions on a subset of the possible faults are necessary; these simplifying assumptions based on experience of many years are manifested in fault models. Since tests are generated to test for the fault mechanisms described by the assumptions, the reliability of the chip is at least partially determined by the accuracy and effectiveness of the fault model (measured, e.g., in detected physical failures). However, it has been observed by many authors that a large amount of defects typical of today's VLSI technologies are not covered by static fault models, like the stuck-at [12] or cellular fault model [7]. In many cases dynamic fault models which allow to model stuck-open faults or timing issues are necessary [16, 19, 23]. Therefore in this paper we consider the *Robust Path Delay Fault Model* (RPDFM) as introduced in [19]. The RPDFM is a very powerful fault model whose objective is to check the propagation delay of every path in the circuit. It has already been observed in [18] that delay fault coverage is quite poor for many combinational circuits and thus many paths are not robust path delay fault testable. In [14, 8] it was shown that not all possible path delay faults of a circuit have to be tested to ascertain correct timing behavior and methods to approximate sets of *Robust Dependent* (RD) paths, that need not to be considered for testing, were given [20, 14].

In order to reduce the test costs, testability issues have to be considered from the very beginning of the design process to guarantee the testability of the circuit at the end of the manufacturing process. This requires a detailed analysis of the relation between structural properties of the circuit design and testability properties. Several successful steps in this direction can be observed: Relevant subclasses of circuits, which, due to their structural properties, allow to generate complete and small test sets efficiently. Typical examples of classes which have been successfully treated by this method are PLAs, memories, arithmetical units and tree-like structures, e.g. [1, 6, 5, 9, 21, 22].

Over the years various VLSI designs for fast addition have been proposed [21, 22]. The delays of the adders presented are optimal from the asymptotic point of view. In [3, 4] the testability of adders based on the *Conditional Carry* principle or the *Conditional Sum* principle are investigated. Especially, it was shown that these time-optimal adders have a complete test set of size $\Theta(n^2 \cdot log(n))$ with respect to RPDFM where n denotes the size of the operands. On the contrary in [11] it was shown that the standard design of a *Carry Ripple Adder* has a complete test set of size $O(n^2)$. Therefore two questions arise:

1. Is there any fast adder having test complexity $O(n^2)$?
2. Is $O(n^2)$ optimal for the test complexity of the adder function?

To answer the first question we show that the *Carry*

*This author has been supported by a *von Humboldt Foundations Senior Research Fellowship.*

Select Adder (CSA), whose run time is bounded by $O(\sqrt{n})$, can be tested by $O(n^2)$ patterns. Moreover we show that any realization of the adder function with *Immediate Operand Reconvergence* (IOR) has a test complexity of $\Omega(n^2)$. IOR is a very reasonable property fulfilled by all adder designs known until now. This answers the second question and shows that the test complexity of the carry select adder is optimal.

The paper is structured as follows: We start with some preliminaries in Section 2. The CSA is described in Subsection 2.1. Subsection 2.2 provides an introduction to the RPDFM. The testability of the CSA is examined in Section 3. We show that the CSA is completely testable with respect to the RPDFM independent of the word size. In Section 4 we prove a tight lower bound for the test size of any adder with respect to the RPDFM. The proof on testability is independent of the real structure of the adder up to the fact that IOR is required. We end with a discussion of the results in Section 5.

2 Preliminaries

2.1 The Carry Select Adder

We start with a description of the *Carry Select Adder* (CSA). Let a, b and s be three binary numbers with

$$a = \sum_{i=0}^{n-1} a_i 2^i \qquad b = \sum_{i=0}^{n-1} b_i 2^i \qquad s = \sum_{i=0}^{n} s_i 2^i$$

where s is the sum of a and b. The relation between the sum s and the operands a and b can be described by the following equations:

$$c_i = \begin{cases} 0 & \text{if } i = 0 \\ a_i b_i \vee a_i c_{i-1} \vee b_i c_{i-1} & \text{otherwise} \end{cases}$$

for $i, 0 \leq i \leq n-1$, and

$$s_i = a_i \oplus b_i \oplus c_{i-1}$$

for $i, 0 \leq i \leq n$. The bit c_i, $0 \leq i \leq n-1$, is called carry bit.

The overall structure of an adder is shown in Figure 1. The primary input to which the input bit a_i (b_i) is assigned is denoted by A_i (B_i), $0 \leq i \leq n-1$. The primary output at which the sum bit s_i is computed is denoted by S_i, $0 \leq i \leq n$. The n-bit CSA is split up into k n_i-bit adders ADD_{n_i}, so that

$$n = \sum_{i=1}^{k} n_i.$$

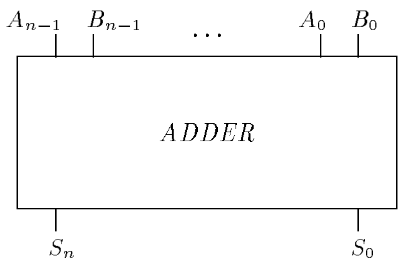

Figure 1: Structure of an adder

The choice of k and the size of the n_i's, $1 \leq i \leq k$, determine the structure of the adder. The computation scheme is shown in Figure 2. In the sequel we describe the structure of an n_i-bit adder ADD_{n_i}, $1 \leq i \leq k$. Thereby, we assume that $p_i = \sum_{l=1}^{i-1} n_l$. The adder ADD_{n_i} receives $(a_{p_i+n_i-1}, b_{p_i+n_i-1}, \dots, a_{p_i}, b_{p_i})$ and c_{p_i-1} as inputs and computes the sum bits $(s_{p_i+n_i-1}, \dots, s_{p_i})$ and the carry bit $c_{p_i+n_i-1}$ as output. ADD_{n_i} is realized using a *Carry Ripple Stage* (CRS), that is defined next. The basic component of a CRS is a *Full Adder* (FA) that adds three bits a_j, b_j and c_{j-1} and computes the sum bit s_j and the carry bit c_j according to

$$s_j = a_j \oplus b_j \oplus c_{j-1} \qquad c_j = a_j b_j \vee a_j c_{j-1} \vee b_j c_{j-1}.$$

Figure 3 depicts an implementation of a FA based on *AND*, *OR* and *NOT* gates. It will be proved later that this implementation is fully testable using the RPDFM. Based on the FA we construct a *Full Adder Circuit* (FAC) that is used to define the CRS. The FAC consists of two FAs and a multiplexer that selects the correct result dependent on the carry of the less significant positions c_{p_i-1} as shown in Figure 4. The upper FA (FA$_u$) computes s_j under the assumption that c_{p_i-1} is 1. The lower FA (FA$_l$) computes s_j under the assumption that c_{p_i-1} is 0. Using this definition it is possible to combine several FACs. If we connect the rightmost upper (lower) carry input $C_{p_i-1,1}$ ($C_{p_i-1,0}$) to constant 1 (0), the linear composition of n_i such cells realize a CRS. At the leftmost position an additional cell is needed to select the correct carry value dependent on the value of c_{p_i-1}. Therefore we use the *Carry Cell* (CC) shown in Figure 5. This cell is functionally equivalent to a multiplexer, since the value $(0, 1)$ cannot be applied to the northern inputs of the cell. A multiplexer is not used because of the redundant value $(0, 1)$ at the northern inputs the multiplexer is not fully testable [4].

By the construction described above we obtain the

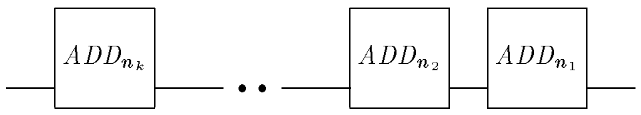

Figure 2: Computation scheme of the CSA

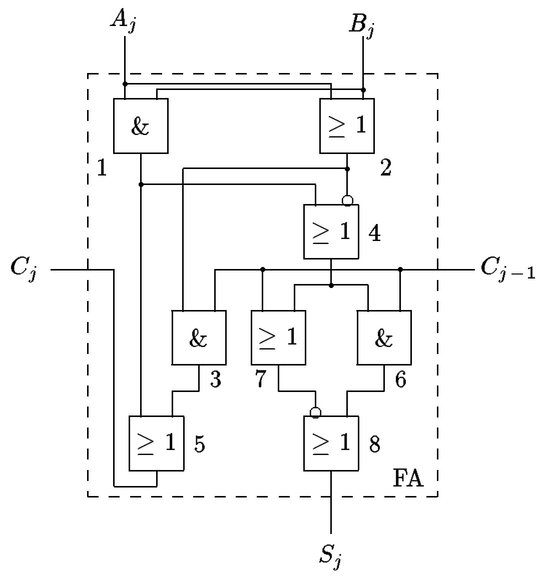

Figure 3: Realization of an FA

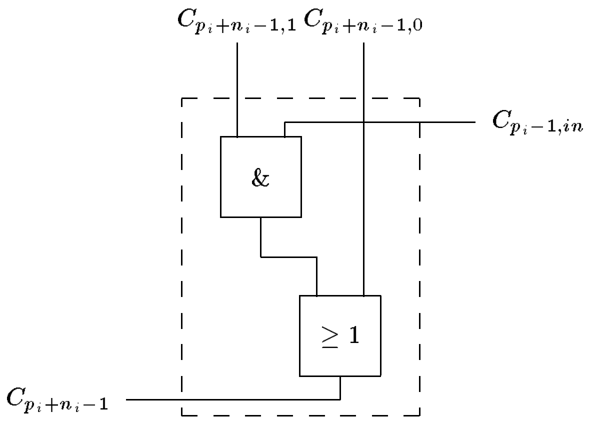

Figure 5: Realization of the CC

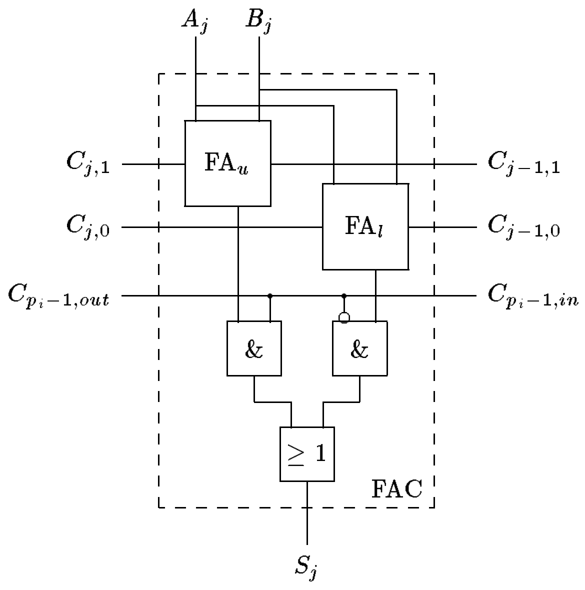

Figure 4: Realization of the FAC

circuit in Figure 6 consisting of one CC and n_i FACs. As mentioned before this circuit is called a carry ripple stage. If we cascade k carry ripple stages of bit length n_i, $0 \leq i \leq k$, we obtain the n-bit CSA.

The choice of k determines the structure and as a consequence the runtime of adder. If we choose $k = O(\sqrt{n})$ and $n_i = O(\sqrt{n})$ for all i we obtain an n-bit adder of runtime $O(\sqrt{n})$.

2.2 The Robust Path Delay Fault Model

The fault model adopted in this paper is the *Path Delay Fault Model* (PDFM) [19]. In this model it is checked whether the propagation delays of all paths in a given combinational circuit are less than the system clock interval.

A CSA can be viewed as a directed acyclic graph whose vertices correspond to the gates (*AND*, *OR*, *NOT*) and primary inputs and outputs. The edges correspond to the leads. A *(pyhsical) path* π is then given by an alternating sequence of nodes and edges $(v_0, e_0, v_1, \ldots, v_n, e_{n+1}, v_{n+1})$ starting at a PI v_0 and ending at a PO v_{n+1}. Inputs of nodes on the path where no edge e_i of the path ends are called *side inputs*.

As usual in delay fault test generation we will associate two *logical paths* with each physical path. A

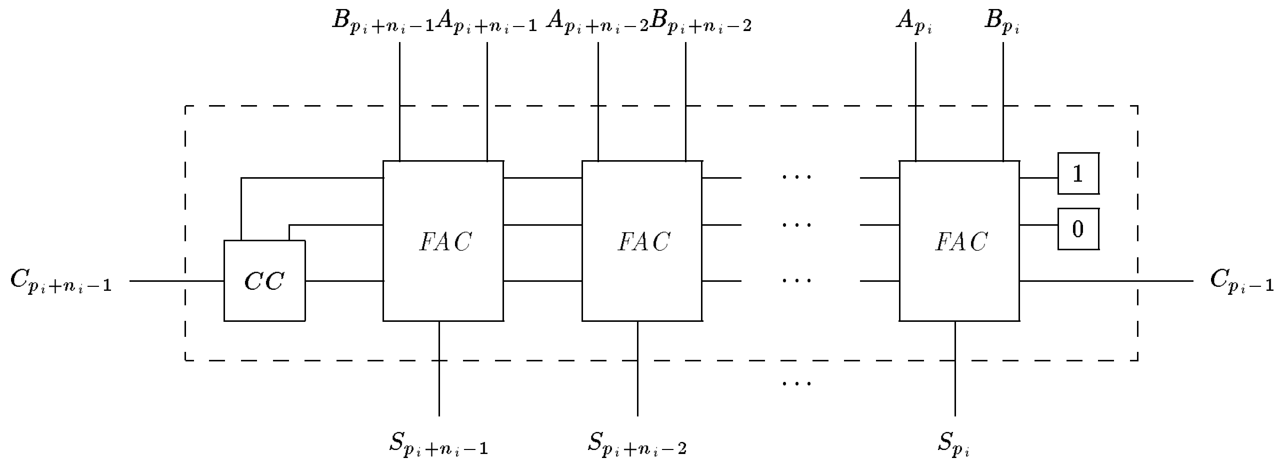

Figure 6: Realization of the CRS

logical path is given as a tuple (π, t) with t being a transition at the PI v_0 of π. A *transition* ($0 \rightarrow 1 =$ *rising* or $1 \rightarrow 0 =$ *falling*) *propagates along* π, if a sequence of transitions $t_0, t_1, \ldots, t_{n+1}$ occur at the nodes $v_0, v_1, \ldots, v_{n+1}$, such that t_i occurs as a result of t_{i-1}. π has a *Path Delay Fault* (PDF), if the actual propagation delay of a (rising or falling) transition along π exceeds the system clock interval. For the detection of such a fault a pair of patterns (I_1, I_2) is required rather than a single pattern as in case of static faults: The initialization vector I_1 is applied and all signals of the circuit C are allowed to stabilize; then the *propagation vector* I_2 is applied and after the system clock interval the outputs of C are controlled. A two-pattern test is called a *robust test* for a PDF on π, if it detects that fault independently of all other delays in the circuit and all other delay faults not located on π. For detailed classification of PDFs see [15, 17]. In this paper we focus our attention on the robust testing of PDFs. If each path of C is robustly testable, C is called *fully testable* with respect to the RPDFM.

It has already been observed in [18] that delay fault coverage is quite poor for many combinational circuits and thus many paths are not robust path delay fault testable. In [14, 8] it was shown that not all possible path delay faults of a circuit have to be tested to ascertain correct timing behavior and methods to approximate a set of *Robust Dependent* (RD) paths, that need not to be considered for testing, were given [20, 14]. A set of robust tests is called a *complete RPDF test set* for a circuit, if it contains a robust test for a set of paths, that guarantees the correct timing of the whole circuit. That is, all paths that are not RD are robust testable by a complete RPDF test set.

3 Test Complexity of the Carry Select Adder

In this section we prove an upper bound for the complexity of a complete RPDF test set for a CSA. The bound is valid for any CSA independent of the choice of k and n_i, $0 \leq i \leq k$. We also show that each CSA can be made fully testable with respect to the RPDFM by a slight modification of the upper FAs in each FAC.

The proof will be done by constructing tests (I_1, I_2) which fulfill the following property: (I_1, I_2) sets all side inputs to the non controlling values on application of I_1, that remain stable during application of I_2, i.e., the values on the side inputs are not subjected to hazards or races.

First, we prove the testability of the FA and of the FAC (see Subsection 2.1) which are basic components of a CSA.

Lemma 1 The FA as shown in Figure 3 is fully testable with respect to the RPDFM. The test set consists of 15 patterns given in Table 1.

Proof: The proof is given by generating the test patterns. In doing so we look at each path seperately. In Table 1 a complete test set is enumerated. The paths are described by the input variable and a sequence of numbers corresponding to the labels at the basic cells as shown in Figure 3 where the path to be tested runs through. The value 0/1 describes the transition at the input to be tested. Each row in the table describes a

paths	a_j	b_j	c_{j-1}	s_j	c_j
A_j,1,4,6,8	0/1	1	1	0/1	1
A_j,1,4,7,8	0/1	1	0	1/0	0/1
A_j,2,4,6,8	0/1	0	1	1/0	0/1
A_j,2,4,7,8	0/1	0	0	0/1	0
A_j,1,5	0/1	1	0	1/0	0/1
A_j,2,3,5	0/1	0	1	1/0	0/1
B_j,1,4,6,8	1	0/1	1	0/1	1
B_j,1,4,7,8	1	0/1	0	1/0	0/1
B_j,2,4,6,8	0	0/1	1	1/0	0/1
B_j,2,4,7,8	0	0/1	0	0/1	0
B_j,1,5	1	0/1	0	1/0	0/1
B_j,2,3,5	0	0/1	1	1/0	0/1
C_{j-1},6,8	1	1	0/1	0/1	1
C_{j-1},7,8	1	0	0/1	1/0	0/1
C_{j-1},3,5	1	0	0/1	1/0	0/1

Table 1: Complete test set for the FA

test pattern consisting of two test vectors for exactly one path in the circuit. □

Lemma 2 The FAC described is fully testable by $2 \cdot 15 = 30$ patterns with respect to the RPDFM.

Proof: The proof uses the result of Lemma 1. The FAC consists of two FAs and a multiplexer whose select input is controlled by a less significant ripple stage. Thus the select input can be directly controlled independent of the inputs to be tested. Since the paths cannot branch out in the multiplexer each FA is tested by 15 test patterns. The select input itself is tested by two patterns. □

We now consider the paths running through several FACs. Special attention will be required for paths running from from a PI A_i (B_i) through gates 1 and 5 of the upper FA (see Figures 3 and 4) across at least one CC cell to a PO S_j ($j > i$). We call these paths *Upper CC paths* (UCC paths).

The following lemma proves the most important property for the upper bound.

Lemma 3 All logical paths from an arbitrary input to an arbitrary output, that are not UCC paths with a rising transition, can be tested by a test set of constant size.

Proof: We assume w.l.o.g. that the input A_i is the input to be tested. For the output S_i the assertion holds by Lemma 2. Thus, we only have to examine the outputs S_j, $i+1 \leq j \leq n$. It follows easily from the construction of the adder that the pyhsical paths from A_i to S_j can only branch out at bit position i or bit position j. Thus, the number of logical paths from A_i to S_j can be bounded by a constant. The construction of the tests for all paths except the UCC paths (with rising transition) now is straightforward. □

We now consider the rising transition on UCC paths. Let π be a UCC path π from A_i to S_j, $j > i$. To test a rising transition it is necessary to apply 1 to B_i. This directly leads to the propagation of a rising transition also in the lower FA of bit postion i. A test for a rising transition on π then simultaneously propagates a rising transition also through the lower FAs. Both paths meet at the OR gate of the carry cell CC. Thus, a rising transition on π is not robustly testable. On the other hand, the propagation of a rising transition within the given clock interval is guaranteed, as long as the path through the lower FAs is robustly testable. We conclude that the logical path with rising transition on π is RD.

Lemma 4 A test set containing robust tests for all paths except the rising transition on UCC paths is a complete RPDF test set for the CSA of Figure 2.

From Lemmas 1 through 4 we obtain an upper bound for the size of a complete RPDF test set for a CSA. (Notice that the upper bound is independent of the real structure of the adder, i.e., the choice of k.)

Theorem 1 The CSA is completely RPDF testable by a test set of size $O(n^2)$.

Proof: The number of test patterns from one input to all outputs is bounded by $O(n)$ using Lemma 3. Since the adder has $2 \cdot n$ inputs $O(n^2)$ clearly is an upper bound for the size of the complete test set. □

Next, we point out, that full RPDF testability of the CSA can be obtained by a slight modification of the upper FA in each FAC. Similar to the methods proposed in [3], all UCC paths can be isolated by duplication of gates, such that no S_j's are reachable before a carry cell is visited. Then the untestable rising transition corresponds to a redundant stuck-at 0 fault. This fault can be removed without introducing new redundancies. We give the resulting modification of the FA cell in Figure 7. Compared to Figure 3 the carry lines C_{j-1}, C_j are duplicated to $C_{j-1,s}, C_{j,s}$ and $C_{j-1,c}, C_{j,c}$. $C_{j-1,s}, C_{j,s}$ ($C_{j-1,c}, C_{j,c}$) are responsible for the computation of the sum outputs (computation of the top left carry input of the carry cell CC). A detailed analysis shows that not only the untestable

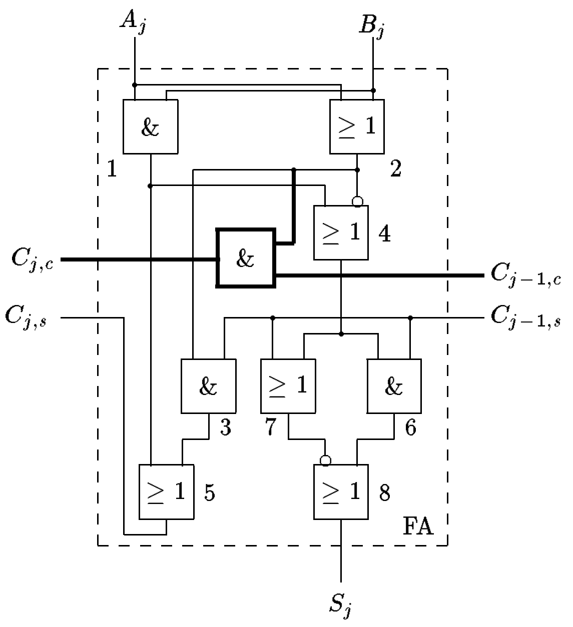

Figure 7: Realization of a modified FA

paths have been removed but also all other paths are RPDF testable. We summarize

Theorem 2 The (modified) CSA is fully RPDF testable by a test set of size $O(n^2)$.

In the following we show that this is an asymptotically optimum bound for the size of the test set.

4 Lower Bound of the Test Complexity

In this section, we prove a lower bound for the cardinality of a test set for an adder with respect to the RPDFM. We make almost no assumptions about the realization of the function. That is, we assume that an adder is a black box that contains circuitry performing the addition of two binary numbers a and b as illustrated by Figure 1. the assumptions we make are:

1. The circuit consists of primitive gates.
2. We have *Immediate Operand Reconvergence* (IOR), i.e. for all i A_i is connected to the same gates with two inputs as B_i.
3. The structure has a complete RPDF test set.

Property 2 is an assumption fulfilled by all practically relevant adder designs [13], e.g., all adders making use of *generate* and *propagate* bits have IOR.

To prove the lower bound we use a basic property of any RPDF test.

Lemma 5 Let C be a combinational circuit with n primary inputs. If (I_1, I_2) with $I_2 = (t_{n-1}, \ldots, t_0)$ is an RPDF test for the path π from the i-th primary input of C to a primary output of C then (I_2', I_2) with $I_2' = (t_{n-1}, \ldots, t_{i+1}, \bar{t_i}, t_{i-1}, \ldots, t_0)$ is also a robust test for this fault.

The correctness of this lemma was proven in [15]. Using this lemma we can prove a lower bound for the test complexity of any adder which fulfills the assumptions above. Let $0 \leq i, i_1, i_2 \leq n-1$ and $0 \leq j, j_1, j_2 \leq n$.

Lemma 6 If $i \leq j$, then there is a path from A_i (B_i) to S_j.

Proof: Consider any input with $a_l = b_l = 0$, $0 \leq l \leq i-1$, $a_l \neq b_l$, $i < l \leq j$ and $b_i = 1$. If $a_i = 0$, we get $s_j = 1$ and if $a_i = 1$, we get $s_j = 0$. Consequently, there must be a path from A_i to S_j. Analogously it can be shown that there is a path from B_i to S_j. □

Now we define the parity of a path as follows:

Definition 1 The parity of a path π is *even (odd)* if the number of inversions on the path is even (odd).

Lemma 7 If $i \leq j$ then there is a non RD path π from A_i (B_i) to S_j having an even parity.

Proof: Consider any input with $a_l = b_l = 0$, $0 \leq l \leq i-1$, $a_l \neq b_l$, $i < l < j$, $a_j = b_j = 0$, $b_i = 1$ if $i \neq j$ and $b_i = 0$ if $i = j$. If $a_i = 0$, we get $s_j = 0$ and if $a_i = 1$, we get $s_j = 1$. Consequently, there must exist paths from A_i to S_j with even parity. Of course, at least one of these paths is non RD. Analogously it can be shown that there is a non RD path from B_i to S_j having an even parity. □

Lemma 8 Let (I_1, I_2) be a test for a path from an input A_i (B_i) to an output S_j. Then $I_2 = (a_{n-1}, b_{n-1}, \ldots, a_0, b_0)$ satisfies $a_l \neq b_l$ for all $i < l < j$.

Proof: W.l.o.g. we consider the path π from A_i to S_j. According to Lemma 5 it follows that $(I_2^{a_i}, I_2)$ with $I_2^{a_i} = (a_{2,n-1}, b_{2,n-1}, \ldots, \overline{a}_i, b_i, \ldots, a_0, b_0)$ is also test for the path π with respect to the RPDFM. Now assume that there exists an l with $i < l < j$ and $a_l = b_l$. Then it follows that the carry c_l computed by

$$c_l = a_l b_l \vee a_l c_{l-1} \vee b_l c_{l-1}$$

is independent of c_{l-1}. Therefore the bits s_m, $l < m \leq n$, are independent of the value assigned to A_i.

It follows that $(I_2^{a_i}, I_2)$ is not a test for π. Moreover using Lemma 5 we can conclude that (I_1, I_2) is not a test for π. But this contradicts the assumption and proves the lemma. $\square$

Lemma 9 Let (I_1, I_2) be a test for a path from an input A_i (B_i) to an output S_j. Then $I_2 = (a_{n-1}, b_{n-1}, \ldots, a_0, b_0)$ satisfies

$$a_i b_i \vee a_i c_{i-1} \vee b_i c_{i-1} \neq \bar{a}_i b_i \vee \bar{a}_i c_{i-1} \vee b_i c_{i-1}$$

(Remark: This means the ith carry bit implied by I_1 is unequal to the carry bit c_i implied by I_2. Moreover if and only if $a_i = 1$ $(b_i = 1)$ it holds $c_l = 1$, $i < l \leq j-1$.)

Proof: W.l.o.g. we consider a path π from A_i to S_j. To prove the lemma we construct the test $(I_2^{a_i}, I_2)$ as described in the previous lemma. Let $\tilde{c_l}$, $0 \leq l \leq n-1$, and $\tilde{s_l}$, $0 \leq l \leq n$, be the carry bits and sum bits implied by $I_2^{a_i}$, respectively.

Obviously we get:

$$\tilde{c_i} = \overline{a}_i b_i \vee \overline{a}_i c_{i-1} \vee b_i c_{i-1}$$

It follows $\tilde{c_i} \leq c_i$ if $a_i = 1$ and $\tilde{c_i} \geq c_i$ otherwise. Now assume that $\tilde{c_i} = c_i$. Then $\tilde{s_l} = s_l$, $i < l \leq n$ and applying $(I_2^{a_i}, I_2)$ we have no transition at S_j. But this is a contradiction since $(I_2^{a_i}, I_2)$ is a test for the path from A_i to S_j. Therefore it must hold $\tilde{c_i} \neq c_i$ and we get $c_i = 1$ if $a_i = 1$ and $c_i = 0$ otherwise.

According to Lemma 8 it holds $a_l \neq b_l$, $i < l < j$. Consequently, we get $c_l = 1$ if $a_i = 1$ and $c_l = 0$ otherwise, $i < l < j$. $\square$

Lemma 10 Let (I_1, I_2) be a test for a path π from A_i (B_i) to S_j with even parity. Then $I_2 = (a_{n-1}, b_{n-1}, \ldots, a_0, b_0)$ satisfies $a_j = b_j$.

Proof: W.l.o.g. we consider a path from A_i to S_j having even parity. Assume that $a_j \neq b_j$. We distinguish two cases:

Case 1: $a_i = 0$

We have a falling transition at A_i. According to Lemma 9 c_{j-1} must be 0. Evaluating

$$s_j = a_j \oplus b_j \oplus c_{j-1}$$

we get $s_j = 1$ if $a_j \neq b_j$. Therefore we must have a rising transition at S_j since (I_1, I_2) is a test for π. But this contradicts the assumption that the parity of π is even.

Case 2: $a_i = 1$

We have a rising transition at A_i. According to Lemma 9 c_{j-1} must be 1. Evaluating

$$s_j = a_j \oplus b_j \oplus c_{j-1}$$

we get $s_j = 0$ if $a_j \neq b_j$. Therefore we must have a falling transition at S_j since (I_1, I_2) is a test for π. But this contradicts the assumption that the parity of π is even.

This completes the proof of the lemma. $\square$

Lemma 11 Let $0 \leq i_1, i_2 \leq \lfloor n/2 \rfloor$ $n \geq j_1, j_2 > \lfloor n/2 \rfloor$ with $j_1 \neq j_2$, π_1 be a path from A_{i_1} to S_{j_1} and π_2 be a path from A_{i_2} to S_{j_2}. If both, π_1 and π_2, have an even parity, then π_1 and π_2 have disjoint test sets.

Proof: W.l.o.g. we assume that $i_1 \leq i_2$, $j_2 < j_1$. Any test (I_1, I_2) for π_2 having an even parity with $I_2 = (a_{n-1}, b_{n-1}, \ldots, a_0, b_0)$ must satisfy the equation $a_{j_2} = b_{j_2}$ according to Lemma 10. Due to Lemma 8, a test for π_1 must satisfy the inequation $a_{j_2} \neq b_{j_2}$. Therefore, (I_1, I_2) cannot be a test for the path π_1. $\square$

Lemma 12 Let $0 \leq i_1, i_2 \leq \lfloor n/2 \rfloor$ with $i_1 > i_2$, $n \geq j_1, j_2 > \lfloor n/2 \rfloor$. Then there exist non RD paths π_1, π_2 with even parity and rising transitions from A_{i_1} to S_{j_1} and from A_{i_2} to S_{j_2}, respectively, whose test sets are disjoint.

Proof: According to Property 2 A_{i_1} is combined with B_{i_1} at the first gate on π_1. Assume w.l.o.g. that this is an AND gate and that B_{i_1} is non inverted. Then a robust test of π_1 for the falling transition requires the final values 1 at A_{i_1} and B_{i_1}. Using Lemma 8, it follows that path π_2 cannot be tested at the same time.

$\square$

Based on Lemmas 11 and 12, we can now prove a lower bound for the cardinality of a test set for an adder with respect to the RPDFM.

Theorem 3 The cardinality of a test set for any completely testable adder with respect to the RPDFM is $\Omega(n^2)$.

Proof: There are at least $\lfloor n^2/2 \rfloor$ physical paths with even parity from a primary input A_i (B_i), $i \leq \lfloor n/2 \rfloor$, to a primary output S_j, $j > \lfloor n/2 \rfloor$. According to Lemmas 11 and 12, testing these paths requires $\Omega(\lfloor n^2/2 \rfloor)$ different tests. $\square$

Notifying this general result we directly obtain:

Theorem 4 The test size of the (modified) n-bit CSA is asymptotically optimal.

5 Conclusion and Open Problems

We presented a completely testable adder with run time $O(\sqrt{n})$ using the powerful robust path delay fault model. We have proven this adder to be optimally testable by a test set of size $\Theta(n^2)$. In addition we have proven a lower bound of $\Omega(n^2)$ which is valid for all combinational n-bit adders with immediate operand reconvergence.

We want to mention that even if IOR is not fulfilled, we can show a lower bound of $\Omega(n\sqrt{n})$ by the consideration of paths with even parity to a fixed output. It is an open problem whether this bound can be improved to $\Omega(n^2)$ in this case. Nevertheless, to our knowledge this is the first result concerning the test complexity of an important class of Boolean functions which makes completely no assumptions about the logical realization of the functions.

References

[1] M. Abadir and H. Reghbati. Functional testing of semiconductor random access memories. *Computing Surveys*, 15:175–198, 1983.

[2] M. Abramovici, M. Breuer, and A. Friedman. *Digital Systems Testing and Testable Design.* Computer Science Press, 1990.

[3] B. Becker and R. Drechsler. A time optimal robust path-delay-fault self-testable adder. In *European Design Automation Conf.*, pages 376–381, 1992.

[4] B. Becker, R. Drechsler, and P. Molitor. On the generation of area-time optimal testable adders. *IEEE Trans. on CAD*, 14(9):1049–1066, 1995.

[5] B. Becker and U. Sparmann. A uniform test approach for rcc-adders. In *Proceedings of the 3rd Aegean Workshop on Parallel Computation and VLSI Theory, LNCS 319*, pages 288–300, 1988.

[6] R. Brent and H. Kung. A regular layout for parallel adders. *IEEE Trans. on Comp.*, 31:260–264, 1982.

[7] M. Breuer and A. Friedman. *Diagnosis & reliable design of digital systems.* Computer Science Press, 1976.

[8] K.-T. Cheng and H.-C. Chen. Delay testing for non-robust untestable circuits. In *Int'l Test Conf.*, pages 954–961, 1993.

[9] W. Daehn and J. Mucha. A hardware approach to self-testing of large PLA's. *IEEE Trans. on Circ. and Systems*, 28, 1981.

[10] S. Devadas and K. Keutzer. Synthesis of robust delay - fault - testable circuits: Practice. *IEEE Trans. on CAD*, 11(3):277–300, 1992.

[11] S. Devadas and K. Keutzer. Synthesis of robust delay - fault - testable circuits: Theory. *IEEE Trans. on CAD*, 11(1):87–101, 1992.

[12] R. Eldred. Test routines based on symbolic logical statements. *Journal of the ACM*, 6(1):33–36, 1959.

[13] I. Koren. *Computer Arithmetic Algorithms.* Prentice Hall, Englewood Cliffs, New Jersey, USA, 1993.

[14] W.K. Lam, A. Saldanha, R.K. Brayton, and A.L. Sangiovanni-Vincentelli. Delay fault coverage and performance tradeoffs. In *Design Automation Conf.*, pages 446–451, 1993.

[15] C.-J. Lin and S.M. Reddy. On delay fault testing in logic circuits. *IEEE Trans. on CAD*, 6(5):694–703, 1987.

[16] E. Lindbloom, J. Waicukauski, B. Rosen, and V. Iyengar. Transition fault simulation by parallel pattern single fault propagation. In *Int'l Test Conf.*, pages 542–549, 1986.

[17] A. Pramanick and S. Reddy. On the design of path delay fault testable combinational circuits. In *Int'l Symp. on Fault-Tolerant Comp.*, pages 374–381, 1990.

[18] S.M. Reddy, C.J. Lin, and S. Patil. An automatic test pattern generator for the detection of path delay faults. In *Int'l Conf. on CAD*, pages 284–287, 1987.

[19] G. Smith. Model for delay faults based upon paths. In *Int'l Test Conf.*, pages 342–349, 1985.

[20] U. Sparmann, D. Luxenburger, K.-T. Cheng, and S.M. Reddy. Fast identification of robust dependent path delay faults. In *Design Automation Conf.*, pages 119–125, 1995.

[21] E. Swartzlander. *Computer Arithmetic (Volume I).* IEEE Computer Society Press Tutorial, 1990.

[22] E. Swartzlander. *Computer Arithmetic (Volume II).* IEEE Computer Society Press Tutorial, 1990.

[23] R. Wadsack. Fault modeling and logic simulation of CMOS and MOS integrated circuits. *Bell System Technical Jour.*, 57, 1978.