EASILY TESTABLE ITERATIVE UNIDIMENSIONAL CMOS CIRCUITS

A. Rubio, R. Anglada and J. Figueras
Department of Electronic Engineering
Polytechnical University of Catalunya
Diagonal 647, 08028 Barcelona, Spain

Abstract

VLSI digital circuits structured as iteration of identical cells offer to the designer the advantage, among others, that in general they can be tested easily. In this work we consider the C-testability of combinational unidimensional iterative circuits implemented in CMOS technology. In this technology the stuck-open faults impose additional requirements to the well known C-testability conditions. The needed requirements are considered for the classical adder and incrementer circuits and these circuits are shown to be potentially C-testable, too, when stuck-open faults are taken into account. C-testable sets of test patterns that detect all the single stuck-open and all the single stuck-at faults for the above mentioned circuits are presented. Results obtained from the consideration of Moore-type sequential iterative circuits are also presented.

1 INTRODUCTION

The increase in the complexity of the integrated circuits and the inherent increase in the cost of the test carried out on them are making it necessary to look for ways of improving the testability of VLSI circuits.

The integrated circuits structured as iteration of identical cells, because their regularity have a set of advantages that make them attractive for many applications. Among these advantages are their simplicity of design, because the structural repetition of the basic cell, manufacturing, test [1-9], fault tolerance and their interest for concurrent algorithmic structure implementation [1,2,10].

In the iterative structures a set of identical cells are interconnected in a regular way following a topology, usually unidimensional arrays. Structures of this kind have a spread field of applications, mainly in the mathematical operation circuits and, in general, in the bit-sliced circuits [1,2,10,11].

As a result of their interconnection regularity it is possible to generate sets of test patterns of very reduced length, and because of this they are attractive in easily testable and self-testable digital circuits [3,12].

This work focuses its attention on the unidimensional iterative circuits implemented in CMOS technology. Most of the works published on the iterative circuit test consider the stuck-at or the basic cell truth table modification [4] fault models.

Section 2 is a review of the well-known fundamental concept of iterative circuits testability, in view of the known conditions for linear testability [7]. In CMOS circuits some frequent faults cannot be modeled following the stuck-at and truth table fault models [13]. This set of faults can be improved by adding stuck-open-type faults. Section 3 constitutes an analysis of the additional requirements that have to be introduced into the test conditions of CMOS circuits when stuck-open faults are considered.

Sections 4 and 5 provide an analysis of the testability of the adder and incrementer CMOS circuits generating C-testable sets of test patterns. Finally section 6 presents the results of the work.

![Figure 1: Unidimensional iterative circuit](image-url)
2 TESTABILITY OF ITERATIVE CIRCUITS

Figure 1 illustrates the typical organization of an N-cells iterative unidimensional circuit. In the figure, and in this work, we consider unilateral circuits (all the signals go from left to right), however the results can be extended to stable classes of bilateral circuits [5].

The N cells have identical functionality. Every cell (i) has an external input $y_i$ and an internal input $x_i$ coming from the previous cell (i-1). Every cell generates a circuit output signal $p_i$ and an internal output $z_i$ that goes to the following cell (i+1).

The following assumptions about these signals are considered:

1) All the $y_i$ vectors are independent.

2) Only the $x_1, y_1, x_2, y_2, ..., x_N$ signals are directly controllable for test procedures.

3) Only the $y_1, y_2, ..., y_N, z_N$ signals are directly observable.

4) The $x_i$ and $z_i$ signals are called the states (input and output states respectively) of the ith-cell and are not directly controllable (except $x_1$) nor observable (except $z_N$).

In [7], Kautz gives the conditions of the basic cell functionality that warrant the exhaustive testing of each of the cells of the array. These conditions assure the controllability and observability of the states. In circuits that verify these conditions the length of the test increases linearly with the number of cells of the array with a resulting length that is inferior to the corresponding length for other implementation structures [7].

A fundamental contribution to the easy testability of iterative circuits was made by Friedman [6]. In his work the concept of C-testability is introduced: an iterative circuit is C-testable if a cell-level exhaustive test with a constant length can be generated. This means the length is independent of the number of cells composing the array (N). The results of [6] are generalised in several ways in [5].

In all these works it is assumed that there is only one faulty cell in the array. Cell level stuck-at (single or multiple) and truth-table fault models are considered. The set $T$ of test vectors of the basic cell is formed by a sequence (whatever the order) of input vectors to the cell. It can be written as follows:

$$T = \{v^1, v^2, ..., v^r\}$$

where

$$v^j = (x^j, y^j).$$

The conditions of C-testability can be enunciated in the following manner:

A) It must be possible to apply every $v^j$ vector from $T$ to every cell $i$ of the array. This means that

$$\forall v^j \text{ and } i \text{ there exist a spacial sequence}$$

$$CA = [x_1, y_1, x_2, y_2, ..., y_{i-1}]$$

that applies the $v^j$ to the $i$-th cell. We can write:

$$x_1 \xrightarrow{CA} x_i^j.$$ 

B) It must be possible to generate the test in a cyclical manner in order to get C-testability. $\forall v^j_i = (x_i^j, y_i^j)$ there exists a spacial sequence:

$$CR = [y_{i+1}, y_{i+2}, ..., y_{i+l-1}]$$

that repeats the vector in a cyclical way in each $i$ cells.

$$x_i^j \xrightarrow{y_i^j, CR} x_{i+l}^j.$$ 

The parameter $l$ is the length cycle for the application of the vector $v^j_i$.

All the circuits where the basic cell has a strongly connected state transition graph verifies the conditions A and B.

C) All the discrepancies of states have to be propagable or observable.

If $x_i^j \neq x_i$ one or both of the following conditions have to be verified:

- $x_i^j \xrightarrow{y_i^j, CR} x_i^j$ with $x_i^j \neq x_i$ (state propagation condition)

- $x_i^j \xrightarrow{y_i^j, CR} x_i(y_i^j)$ with $y_i^j \neq y_i$ (output observation condition).

Proves of this conditions are presented in [4,5,6,7].

3 ADDITIONAL REQUIREMENTS IN CMOS CIRCUITS

In 1978 Wadsack showed [13] that because of the duality of function of the PMOS and NMOS transistor networks some physical defects changed the combinational behaviour of a classical gate to sequential, these defects are modeled using the concept of stuck-open.
In the presence of a stuck-open fault the output of a gate can depend on the preceding value of the output. Faults of this kind are not modelable with stuck-at and truth-table fault models and introduce specific requirements into the test procedures.

Many works [1,4,8,15,16,17] show that to test stuck-open faults two actions have to be performed: initialization and testing. This differs from the test procedures for stuck-at where just one action was needed. In stuck-open tests, the test patterns must be applied without any kind of insertions (due for example to glitch or race signals).

In accordance with the nomenclature of section 2 the test set $T$ for the basic cell, if stuck-open and stuck-at faults are considered, will look like this:

$$T = \{S^1, S^2, ..., v^{s1}, ..., v^h\}$$

where $S^i$ is a temporal sequence (with a fixed order) of test vectors:

$$S^i = \{v^s, ..., v^h\}$$

where the vector $v^j$ has the form:

$$v^j = (x^j, y^j)$$

The need of the sequential test ($S^i$) introduce the following conditions in order to assure C-testability:

D) For each sequence $S^i$ of $T$ there must exist sequences of repetition $CR$ of equal length ($l_i$) for all the vectors $v^j$ that compose $S^i$. This means that it is possible to repeat the sequence with the length $l_i$ (cycle length).

E) In CMOS technology the transient logic values originated for differential propagation delays can invalidate the stuck-open tests [17]. The iterative circuits have, as an inherent characteristic, the strong propagation delay of state vectors. This means that a test set can found to be robust at cell level and be invalidated at array level.

Procedures must be established to avoid this kind of invalidation during the test phase. Among the different procedures to obtain this the shifted application of the test vectors [12], the keeping of the outputs of the cells in third state during the transition time [17] and the utilization of sincronizing elements [12], can be considered.

F) Because of the sequential behaviour of the stuck-open faults the error must remain apparent during the time of propagation.

In the following section C-test vectors are determined for the adder and incrementer circuits, it is assumed that that the conditions E and F are verified.

4 ADDER CIRCUIT

The adder circuit of two n-bits length words, $A$ and $B$, is a typical example of iterative structure. In this example the adder organized in 1-bit adder cells is considered. The state vector of the basic cell is the carry signal, with a dimension of 1 bit. The external inputs are $a$ and $b$ and the input state is the carry $c$ of the previous cell. The outputs are the adder output $S$ and the output state $c'$.

- The transition graph of the basic cell in function of $a$ and $b$ is shown in figure 2. The adder function verifies the conditions A, B, C and D with a cycle length of 2. In our example the following implementation is considered: The 1-bit adder is configured as the aggregation of 2 half-adders and an OR gate (see figure 3). The half-adders are implemented with AND and XOR gates. All the gates are full-complemented except the XOR that have the dual structure presented in [17]. In order to generate the test pattern for the basic cell the method of application of sequences 010 or 101 with propagation of discrepancies to the output of networks of full-complemented gates has been followed.

With these considerations a test pattern set of 15 vectors (see Table 1) can be generated (in this work the test set has been generated manually) to detect all the single stuck-open, all the single stuck-at and truth-table faults.

The presented test is longer than the exhaustive one (15 instead of 8) because of the the need to keep the sequences.

The C-test of the iterative circuit is formed by 26 vectors (it would be 8 if we consider just the stuck-at faults[11]) all with a constant cycle of 2 (or 1). The test makes it possible to detect all the single stuck-open faults, all the single stuck-at and truth table faults. The length of the test (26 vectors) is independent of the size (N) of the adder.

![Figure 2: State transition graph for the 1-bit adder cell](image-url)
5 INCREMENTER CIRCUIT

The incrementer circuit can also be implemented as the iteration of a basic cell. In this case the basic cell has one input variable \( a \), one output variable \( S \) and the state variables \( (c, c') \) all of them of a 1 bit-length. The basic cell is shown in figure 4 where:

![Figure 4: Basic and modified incrementer circuits](image)

In table 2 such a test is illustrated, with the fault-free outputs for the even and odd cells of the array.

In order to apply these test patterns it is necessary to have access to the variable \( c_1 \). In [15] an adder circuit of 2 bits implemented with NOR gates is presented, requiring 30 test vectors (no \( C \)-test) to test it. In the next section the incrementer circuit is analysed.
The state transition graph is shown in figure 5. The circuit is not C-testable because the combination \( c_0 = 0X \) does not have a repeating sequence. The non C-testable circuits can be modified to C-testable [5,9]. In figure 4 an elemental incrementer circuit modified to be C-testable is presented. The circuit has some additional logic and one additional test input \( P \). In figure 6 the new state transition graph is presented. For this circuit and an implementation based on the gates of the preceding section test patterns detecting all the stuck-at and truth-table faults and all the single stuck-open can be obtained (14 vectors, generated manually, see Table 3).

The horizontal lines in Table 3 indicate the separation of sequences. The C-test at array level can be implemented with 23 vectors, independently of the array size \( N \) (see Table 4).

The circuit presents only two additional inputs for all the iterative circuit \( (P_{even}, P_{odd}) \).

The results of the C-testable incrementer circuit can be applied to the design of counters. In [11] it can be seen that a Moore-type sequential circuit is C-testable if the corresponding combinational circuit is C-testable. A counter can be designed using incrementer circuits and D-type master-slave flipflops. In [9] it is shown that a counter circuit can be C-testable with a test length of 18 vectors (taking stuck-at faults into consideration). Following the same strategy and considering the flipflop implementation presented in [18] a C-testable basic cell for a counter can be designed to have a test length of 31 vectors taking stuck-open faults into consideration. The overall counter circuits has a C-testable vector set with a length of 65 vectors.

![Figure 5: State transition graph of the basic 1-bit incrementer](image)

![Figure 6: State transition graph of the modified 1-bit incrementer](image)

**Table 3: Test patterns for the modified incrementer cell**

<table>
<thead>
<tr>
<th>vector</th>
<th>( C )</th>
<th>( A )</th>
<th>( P )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Table 4: Test patterns for C-testable incrementer array**

<table>
<thead>
<tr>
<th>vector</th>
<th>( c_1 )</th>
<th>( a_1 )</th>
<th>( P_1 )</th>
<th>( P_2 )</th>
<th>( S(\text{odd}) )</th>
<th>( S(\text{even}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
6 CONCLUSIONS

The results of easy testable iterative circuits can be extended to CMOS circuits where the stuck-open faults introduce additional requirements. Some special conditions have to be considered in order to avoid the invalidity of the test at array level due to internal glitches and races. Our conclusion is that in this type of circuits very reduced sequences of test can be obtained, even with a length that is independent of the size of the array (C-testable). The length of the test is greater than that of circuits with stuck-at faults, because of the need to maintain the sequence of the test vectors and the difficulty of compactation. C-tests have been presented for the classical adder and C-testable modified incrementer circuits, attention has been given to the possibility of designing C-testable counters. Although all the tests have been obtained for single-stuck-open faults, these tests have a very high coverage of multiple faults, due to the known dominance of stuck-open faults in series circuits.

REFERENCES


