AN EFFICIENT ALGORITHM FOR TESTING THE TWO-LEVEL CARRY SKIP ADDER (CSKA)

Daniela E. Popescu

University of Oradea, Str. Armatei Romane nr.5, Oradea, România
depopescu@uoradea.ro

Abstract: The paper is the result of the author's activities concerning the testing and the design for testability for Computer arithmetic systems. By applying the C-testability concept, presented in the first part of the paper, we present a method of identifying a minimal complete set of test vectors for detecting all single stuck-at faults for the multi-level Carry Skip Adder (CSkA), together with our results.

Key words: C-testability, single stuck-at faults, iterative logic arrays

1. INTRODUCTION

Pseudoexhaustive testing techniques based on partitioning are perfectly suited for circuits structured as iterative logic arrays (ILAs), composed from identical cells interconnected in a regular pattern [3].

The partitioning problem is naturally solved by using every cell as a segment that is exhaustively tested. We consider one-dimensional and unilateral ILAs, that is, ILAs where the connections between cells go in only one direction. First we assume that cells are combinational. A ripple-carry adder is a typical example of such an ILA.

Some ILAs have the useful property that can be pseudoexhaustively tested with a number of tests that does not depend on the number of cells in the ILA. These ILAs are said to be C-testable [2].

In Fig.2. we have the truth table of a cell of the ripple-carry adder. We can observe that CI=CO in 6 entries. Thus, if we apply the *X, Y* values corresponding to one of these entries to every cell, then every cell will also receive the same *CI* value. Hence, these six tests can be applied concurrently to all cells. In the remaining two entries, $CO = \overline{CI}$. The tests corresponding to these two entries can be applied to alternative cells, as shown in [4]. Hence a ripple-carry adder of arbitrary size can be pseudoexhaustively tested with only eight tests; therefore, it is a C-testable ILA.

Verifying the truth table of a cell in the ILA corresponds to verifying every entry in the state table of the equivalent sequential circuit. The truth table of the cell of the ripple-carry adder, can be interpreted as the state table of the sequential circuit modeling the ripple-carry adder.

Let us consider the checking of an entry (y, x) of the truth table of an ILA cell. To have the total number of tests independent of the number of cells in the ILA, the same input combination (y, x) must be simultaneously applied to cells spaced at a regular interval - say, k - along the array. For the sequential circuit modeling the ILA [4], this

means that the k-step sequence starting with x, which is applied in state y, must bring the circuit back to state y. The one test applies the input combination (y, x) to every *k*-th cell, and *k* tests are sufficient to apply it to every cell in the ILA. The following result characterizes a C-testable ILA:

- The state diagram of a sequential circuit modeling a C-testable ILA is either a strongly connected graph, or it is composed of disjoint strongly connected graphs [2].

2. THE C-TESTABILITY ANALYZE OF THE MULTI –LEVEL *CARRY SKIP ADDER (CSkA)*

Our paper focuses on finding the minimal test set for detection the single stuck-at 0 or 1 faults for the multi-level *Carry Skip Adder* (CSkA) [1], which uses the carry skip idea for computing the carries between blocks.

The CSkA with one level of carry-skipping is obtained from a RCA (*Ripple Carry Adder*) by dividing the adder stages into several blocks, which need not be of the same size, and equipping each block with carry skip logic that determines when the carry into the block can be passed directly to the next block (Figure 1).

A carry may skip a block, i, of size m if:

$$(A_i + B_i)\cdot(A_{i+1} + B_{i+1})\cdot\ldots\cdot(A_{i+m-1} + B_{i+m-1}) \overset{D}{=} T_i \cdot T_{i+1} \cdot\ldots\cdot T_{i+m-1} = 1 \qquad (1)$$

(T_i is the *CY transfer* for stage *i* - $T_i = A_i + B_i$).

The carry that enters block $i+1$ from block i is either the carry, C_{i-1}, that skips block i, or is the carry, C_{i+m-1}, that is produced by block i. The carry skip logic for every block therefore consists of one OR gate for each T signal, one AND gate to combine the T signals, and an OR gate to select either source of carry. Together, these implement the equation:

$$block\ CY-out = T_i \cdot T_{i+1} \cdot \ldots \cdot T_{i+m-1} \cdot C_{i-1} + C_{i+m-1} \qquad (2)$$

The organization of part of a carry-skip adder is shown in Figure 1. The design assumes that a carry C_i is produced by equation:

$C_i = A_i B_i + (A_i \oplus B_i) C_{i-1}$, so an extra OR gate is needed to produce T_i .

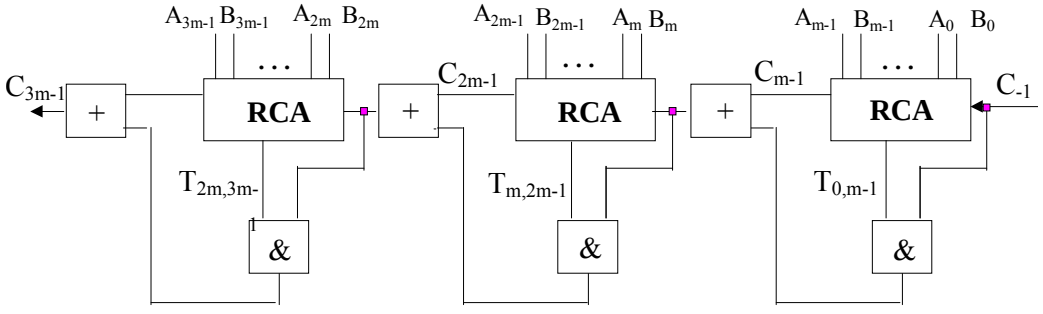

Figure 1

Note that in order to obtain a good result, the *CY-out* signals must be reseted before each operation (ex. By pre-charging in CMOS).

The same principles used to develop the CskA from the RCA can be applied to develop a two-level CskA from the basic CskA in order to obtain further improvements. In this case the skip blocks are grouped into superblocks, with additional carry-skip logic supplied for each superblock.

Suppose we have a block size of m and a superblock size of mM. A carry entering superblock I will skip through the superblock if:

$$\left(A_i + B_i\right)\cdot\left(A_{i+1} + B_{i+1}\right)\cdot \ldots \cdot \left(A_{i+mM-1} + B_{i+mM-1}\right) \overset{\Delta}{=} T_i \cdot T_{i+1} \cdot \ldots \cdot T_{i+mM-1} = 1 \quad (3)$$

Taking into account the carry skip logic already available for each block, and defining:

$$K_j = P_j \cdot P_{j+1} \cdot \ldots \cdot P_{j+m-1} \quad (4)$$

the superblock skipping condition may now be written as:

$$K_i \cdot K_{i+1} \cdot \ldots \cdot K_{i+M-1} = 1 \quad (5)$$

Hence, the additional carry-skip logic required at the superblock level is the implementation of the following equation for each superblock i:

$$Super-block\ CY-out = C_{i+mM-1} + K_{i+M-1} \cdot C_{i+(M-1)\cdot m-1} + K_i \cdot K_{i+1} \cdot \ldots \cdot K_{i+M-1} \cdot C_{i-1} \quad (6)$$

This leads to the organization of Figure 2:

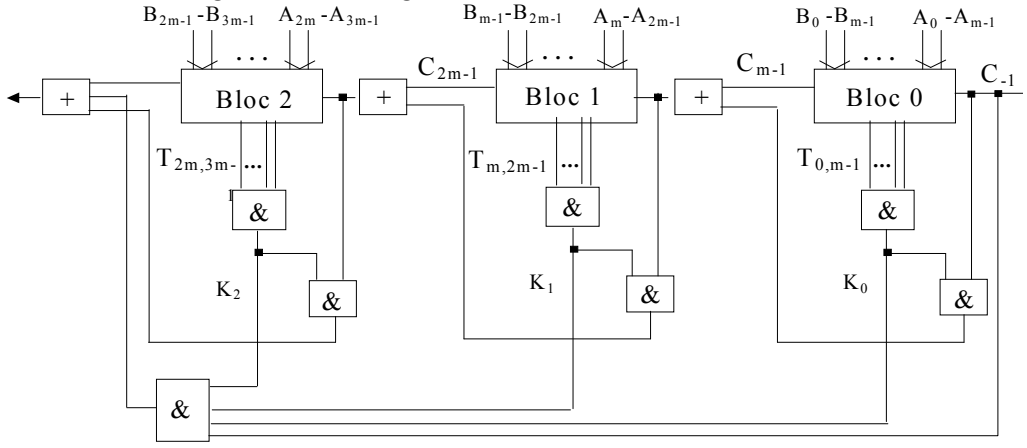

Figure 2

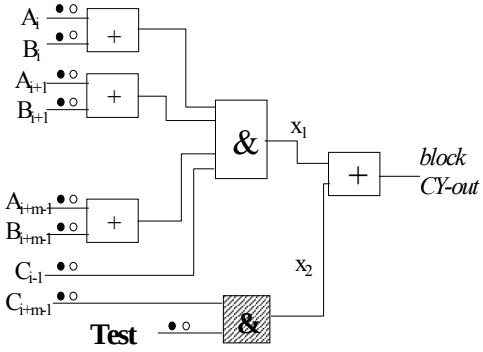

Figure 3

So, the design of CSkA is obtained by interconnecting in an ILA structure the RCA cells (which are C-testable ILA [6]) together by an external combinational logic that give out the *block carry-out* signal. The fault model used suppose that the faulty cell is still combinational.

The considered ILA is unilateral, one dimensional and composed of combinational cells (the ripple-carry blocks together with the CY-skip unit that generate the carry for the next block) with directly observable outputs, whose sequential model has *m* primary inputs (A_i, B_i) and one state input (C_{i-1}).

The test pattern needed for complete testing the CskA are obtained from the pseuoexhaustive test patterns that test the ILA composed by the RCA cells, together with the test patterns that realize the complete testing of the circuit that gives the block carry-out signal.

Based on the above mentioned and on the C-testability property, I demonstrated in [5][4] the following two original theorems:

Theorem 1 By adding of a supplementary gate to each block (the gate marked in Figure 3) of a CSkA whose blocks' dimension is *m*, we obtain a C-testable CSkA; the total number of tests that detects the single stuck-at faults is constant and independent the number of blocks links together by the *block carry-out* logic, having the expression: $N_{TCSkA} = 10 + 2 \cdot m$.

Theorem 2 By adding of a supplementary gate to each superblock (the gate marked Figure 4) of a two-level CskA, whose blocks' dimension is *m*, and the number of blocks in a superblock is M, we obtain a C-testable two-level CSkA; the total number of tests that detects the single stuck-at faults is constant and independent the number of blocks and superblocks links together by the *block and superblock carry-out* logic, having the expression: $N_{TCSkAmn}(8 + 2m + M + 3)$.

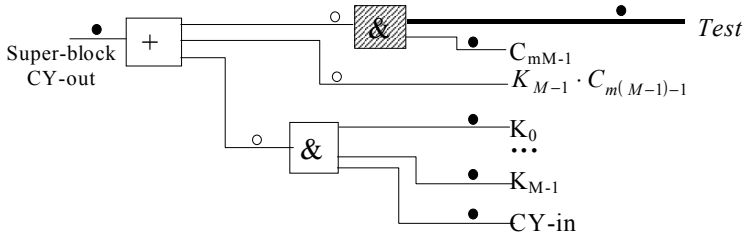

Figure 4

We give here (Figure 5) in pseudocod the algorithm for obtaining the set of test vectors for an *n* bit two-level CSkA that has the blocks' dimension *m* and the superblock dimension M:

Algoritm CSkA_pe_2_nivele()
Begin
For $i = 0, n-1$
Assign $A_i = 0$, $B_i = 0$, $Test = 1$ *and* $C_{-1} = 0$
Apply the obtained test
For $i = 0, n-1$
Assign $A_i = 0$, $B_i = 1$, $Test = 1$ *and* $C_{-1} = 0$
Apply the obtained test
To the same assignment for the operands A i B , assign $C_{-1} = 1$
Apply the obtained test
To the same assignment for the operands A i B, as for C_{-1}, *assign* $Test = 0$
Apply the obtained test
end
For $i = 0, n-1$
Begin
Assign $A_i = 1$, $B_i = 0$, $C_{-1} = 0$ *and* $Test = 1$
Apply the obtained test
To the same assignment for the operands A i B , assign $C_{-1} = 1$
Apply the obtained test
To the same assignment for the operands A i Bas for C_{-1}, *assign* $Test = 0$
Apply the obtained test
end
$Test = 1$

For $i = 0, n-1$

Assign $A_i = 1$, $B_i = 1$ *and* $C_{-1} = 0$

Apply the obtained test

$C_{-1} = 0$

Repeat

i=0

Assign $A_i = 1$, $B_i = 1$, $A_{i+1} = 0$, $C_{-1} = 0$

i=i+2

until $i \geq n$

Apply the obtained test

$C_{-1} = 1$

Repeat

i=0

Assign $A_i = 0$, $B_i = 0$, $A_{i+1} = 1$, $B_{i+1} = 1$

i=i+2

until $i \geq n$

Apply the obtained test

$C_{-1} = 1$

$Test = 0$

For the first block of each superblock, assign

For $i = 1, m$ *the values* $A_i = 0$ *and* $B_i = 0$

For all superblocks

For all blocks j, $j = 2, M$

For $i = 1, m$ *assign the values* $A_i = 1$ *i* $B_i = 1$

Apply the obtained test

$Test = 1$

/ for blocks with constant dimension m */*

For $j = 1, m$

Begin

For the j stage of all blocks assign the values $A_j = 0$, $B_j = 0$

For all the adder's blocks ***Do***

For $(i = 1, m), i \neq j$ *assign for the i stage the values* $A_i = 0$, $B_i = 1$

For the last stage of even blocks assign the values: $A_m = 1$, $B_m = 1$

Apply the obtained test

Inverse the input values from the last stage of even blocks with those from the odd blocks

Apply the obtained test

end

/ for superblocks with M blocks (M=constant) */*

For $j = 1, M$

Begin

$C_{-1} = 1$

For the j block of all the superblocks

For $i = (1, m)$ *assign the values* $A_i = 0$, $B_i = 0$

For all the superblocks

For all blocks k, $k = 1, M$ $(k \neq j)$

For $i = (1, m)$ *assign the values* $A_i = 0$, $B_i = 1$

For the last block of the even superblocks

For $i=(1,m)$ *assign* $A_i=1,\ B_i=1$

Apply the obtained test

Inverse the input values from the last stage of even blocks with those from the odd blocks

Apply the obtained test

end

end

Figure 5

4. CONCLUSIONS

The applying of the C-testability concept for determining the complete tests set for detection of the singular stuck-at faults in the two-level CSkA permits to obtain a number of tests, which can be used in testing a two-level CSkA of any size. This leads to an important gain in time testing for such an adder.

REFERENCES

1. Omondi, A. R., (1994), Computer Arithmetic Systems - Algorithms, Architecture and Implementation, Prentice Hall International, pp.39-49
2. Friedman, A.D., (1973), Easily Testable Iterative Systems, *IEEE Trans on Computers*, Vol. C-22, No.12, pp.1061-1064, Dec.1973
3. Abramovici, M., Breuer, M.A., Friedman, A.D., (1996), Digital Systems and Testable Design, Computer Science Press, pp.97-110
4. Popescu D, (1998), Contribu ii la testarea asistat de calculator i sinteza testabil a echipamentelor de calcul cu fiabilitate sporit –teza de doctorat
5. Popescu, D. and Popescu, C., (1999) The C-testability Analyze of the Carry Skip Adder (CSkA), *Proceedings of Oradea EMES' 99*,
6. Popescu, D. and Popescu, C., (1999), An algorithm for determining the minimal test set for the Block Carry-Lookahead Adder (BCLA), *Proceeding FEI International Conference Electronic Cpmputers & Informatics, Slovakia*, pp.108-113
7. Popescu, D. and Popescu, C, (2000), An Efficient Algorithm for Testing The Carry Skip adder, *A&Q00, International Conference and Quality Control, QUALITY, Design and Development,* Tome I, Cluj, pp. 51-56