CA-CSTP: A new BIST Architecture for Sequential Circuits

F. Corno, M. Sonza Reorda, G. Squillero, M. Violante
Politecnico di Torino - Dipartimento di Automatica e Informatica
Torino, Italy
http://www.cad.polito.it/

Abstract

Circular Self-Test Path (CSTP) is an attractive technique for implementing BIST in sequential circuits; unfortunately, there are cases in which the fault coverage it attains is unacceptably low. This paper proposes a new architecture, named CA-CSTP, which overcomes these limitations and always reaches a high fault coverage by exploiting a slightly more complex chain cell based on a Cellular Automata architecture. Experimental results show the effectiveness of our proposal.

1.Introduction

In the ASIC design and production process, testing is known to be a critical issue from an economical and technical point of view. The evolution of technology and the increasing complexity of circuits are making traditional approaches, based on sophisticated ATPGs tools and expensive ATEs, less and less viable; therefore, BIST solutions are gaining popularity. Several parameters are taken into account when evaluating the practical applicability of the many BIST approaches proposed in the past years: their capability to detect faults, the test time they involve, the area overhead and performance slow-down they introduce, the easiness of their integration in the existing design flows, etc.

Circular Self-Test Path (CSTP) was first introduced in [1] and represents a general solution to implement BIST in sequential circuits. CSTP is based on transforming each memory element in the circuit into a special cell (Fig. 1); moreover, a similar cell is added to each Primary Input (PI) and Primary Output (PO); all the cells are finally connected to form a circular chain. When in test mode, the chain acts at the same time as a test vector generator and as an output response analyzer (Fig. 2). The CSTP technique is potentially very effective, since it involves relatively low hardware and time overheads, it can be easily adapted to most design flows, and it performs an at-speed testing. On the other side, CSTP has a major limitation in the low fault coverage it sometimes reaches, which can hardly be foreseen using the theoretical analysis reported in [2]. This phenomenon is due to several different causes:

- Correlations between the input and output signals of different cells, due to the function implemented by the circuit, can limit the capability of the chain to generate good pseudo-random input sequences, and to compact the circuit output response.
- After CSTP insertion, some circuits exhibit a cyclic behavior, so that the circuit enters a short cycle, where it remains blocked indefinitely; in this case, the number of states reached by the CSTP chain does not grow any more when increasing the test time, nor does the attained fault coverage.
- For hard to test circuits, the number of clock cycles needed to generate all the required test vectors can be very high, thus limiting CSTP real applicability.

Several papers faced the issue of how to increase the fault coverage provided by CSTP for the most critical circuits. Avra and McCluskey [3] propose a different cell, able to avoid any problem created by dependencies between cells. Corno et al. [4] show that re-ordering the chain is insufficient for eliminating dependencies in the general case, and propose a new method, based on symbolic computation, for determining the best initial state for avoiding short cycles. Touba [5] describes a method named *state skipping* which greatly enhances the reached fault coverage, at the cost of an increase in the complexity of the adopted CSTP cell; moreover, his method requires the availability of the test cubes for hard to test faults in the circuit.

This paper proposes a new method that enhances the performance of CSTP by increasing the number of faults it detects without introducing any significant increase in the required area overhead. When in test mode, the chain implements a hybrid Cellular Automaton (CA), and is able to both generate pseudo-random vectors and compact the output behavior of the circuit. In this new architecture, called CA-CSTP, the classical CSTP chain can be obtained as a particular case. The approach requires some method to compute the rule implemented by each CA cell. For the purpose of this paper, we exploited a straightforward hill-climbing algorithm; the starting solution for the algorithm is carefully chosen, so that the resulting rules require a minimal amount of logic to be implemented.

Section 2 describes the new CA-CSTP architecture. Section 3 outlines the method we followed to determine the rules implemented by the chain cells, and Section 4 reports some experimental results. Section 5 draws some conclusions.

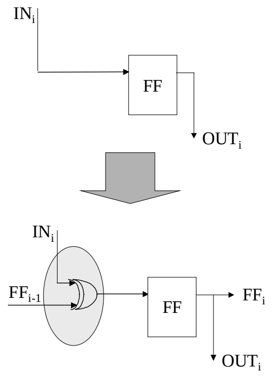

Fig. 1: basic CSTP cell.

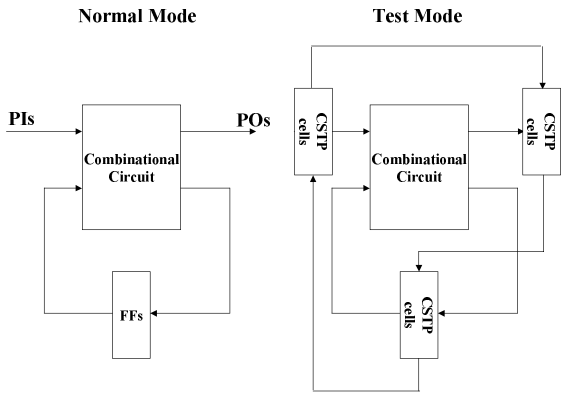

Fig. 2: CSTP architecture.

2. CA-CSTP Architecture

The basic CSTP cell is composed of a flip flop and an exor gate. The exor gate is fed by the output of the flip flop of the previous cell in the chain, and by either a circuit output (for cells on the POs), or the output of the circuit combinational part (Fig. 1 and 2). In [3] and [5] some new cells are proposed, which differ from the original one because of a more complex Boolean function feeding the flip flop input. We propose a new approach able to overcome all the limits of the basic CSTP cell. The approach adopts a more general cell (Fig. 3), where the 4-input function f_i is fed by the flip flop in the previous cell of the chain (FF_{i-1}), the flip flop in the same cell (FF_i), the flip flop in the following cell (FF_{i+1}), and the circuit output (IN_i). In this way, when in Test Mode the chain becomes a non-linear hybrid Cellular Automaton, which evolves according to its previous global state, and the value coming from the circuit outputs (Fig. 4).

By suitably selecting the rules (i.e., the Boolean functions f_i) implemented by CA-CSTP cells, it is possible to overcome the limits of the original CSTP architecture, avoiding dependencies and loops, and possibly letting the CA generate the test vectors required to detect hard to test faults.

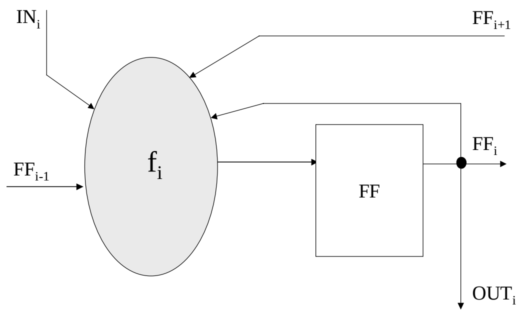

Fig. 3: CA-CSTP cell.

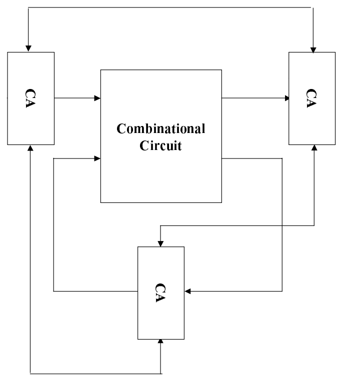

Fig. 4: CA-CSTP architecture.

The CA-CSTP architecture exploits the characteristics of CA for both generating test vectors and compacting output responses. Although properly configured CA are well-known to exhibit very good properties from both aspects [6] [7], the reader should note that in this case there are two main novelties:

- the CA acts *at the same time* as a generator and as a compactor
- the CA is configured in a circular way, while the standard configuration includes null boundary conditions.

For the above reasons, it is impossible to adopt the standard methods for selecting linear rules to be implemented by the CA cells. Moreover, in order to find better solutions we do not limit our search to linear CA, but we accept that a limited number of cells implement not linear functions. In the following Section we will propose a suitable method for selecting the optimal configuration of the CA.

3. Rule Selection

In the proposed CA-CSTP architecture each CA cell has 4 inputs and implements a 4-input Boolean function out of the 2^{16} possible ones. Therefore, each rule can be represented by a 16-bit number. The rule selection process explores the space of all sequences of n 16-bit numbers, where n is the number of CA cells in the CA-CSTP version of the considered circuit.

The goal of the rule selection process is to find a point in this space (i.e., a set of rules, one for every CA cell), so that the corresponding CA-CSTP architecture, when run for a given number of clock cycles, reaches a maximum fault coverage. According to the original CSTP principles, we assume that a fault is detected if, at the end of the test cycle, at least one of the CSTP chain flip flops holds a different value in the good and faulty circuit. In this way, aliasing effects are taken into account.

An important additional constraint that should be taken into account when defining the rule selection process is the final area overhead coming from implementing the selected rules. Therefore, the proposed approach introduces some additional area overhead with respect to CSTP only if this is strictly required to improve FC.

The selection process is based on a first-improvement hill-climbing procedure which at each step aims at improving the current solution by exploring neighboring solutions, evaluating them on the basis of the value of a given evaluation function.

The pseudo-code of the proposed rule selection procedure is shown in Fig. 5.

```
SOLUTION rule_selection( SOLUTION start)
{ SOLUTION curr, new;
  int   best, val;
  curr = start;
  best = eval( curr);
  while( stopping_condition() == FALSE)
  {  new = get_neighbor( curr);
     val = eval( new);
     if( val > best)
     {  best = val;
        curr = new;
     }
  }
  return( curr);
}
```

Fig. 5: Rule selection procedure pseudo-code.

Four issues have to be faced in order to optimally tune the rule selection process: the initial solution choice, the neighborhood exploration mechanism, the evaluation function definition, and the stopping condition.

3.1 The Initial Solution

The choice of the set of CA rules the selection process starts from strongly affects the algorithm ability of both maximizing the fault coverage and minimizing the area overhead. An effective solution is to start from a chain composed of CA cells implementing the rule #27030, corresponding to the exor of the cell four inputs (the previous and following cell, the cell itself, and the circuit output). In this way we can often obtain an initial fault coverage which is close to the one attained by the original CSTP architecture, with a small area overhead.

3.2 Neighborhood Exploration

Each solution considered by the rule selection process corresponds to a string of 16n bits. The *distance-d neighborhood* of a solution S is composed of the solutions having a Hamming distance equal to d from S, i.e., those that can be obtained from S by complementing d of its bits. For the purpose of this paper we adopted a neighborhood exploration mechanism based on randomly selecting a solution among those in the distance-1 neighborhood.

Following this mechanism, space exploration is first performed in the sub-space close to the initial solution (which is very cheap in terms of area), thus reducing the chance of unnecessary jumps into any area of high-overhead solutions.

3.3 Evaluation Function

We adopted an evaluation function corresponding to the percent number of faults detected by the current CA-CSTP architecture when run for a fixed number K of clock cycles. Aliasing effects are taken into account by marking a fault as detected only when it causes the value of at least one flip flop in the CSTP chain to differ from the good value at the end of the K clock cycles.

3.4 Stopping Condition

The hill-climbing procedure is stopped when either 100% fault coverage is reached, or a maximum number MAX_ITER of iterations is achieved.

4. Experimental Results

In order to evaluate the proposed CA-CSTP architecture, as well as the related rule selection procedure, we set up an experimental environment composed of three modules (Fig. 6):

- a *search engine*, which implements the hill-climbing procedure outlined in Fig. 5
- a *CA interface*, which receives from the search engine a set of CA rules, and generates the corresponding hardware cells
- a *fault simulator*, which receives the CA cells produced by the previous module and simulates them for K clock cycles, starting from the given starting solution, together with the combinational part of the circuit; the module returns the number of detected faults at the end of the experiment.

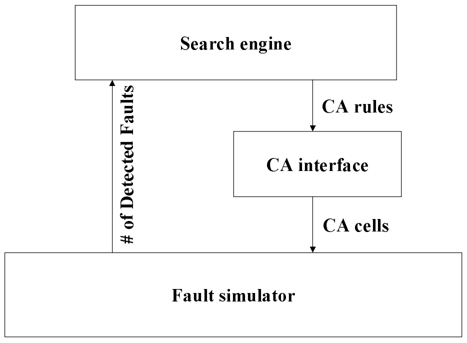

Fig. 6: experimental environment.

A prototypical version of this environment has been implemented in C language resorting to an in-house developed gate-level fault simulator. Synthesis of the CA is not necessary, because the simulator can handle special gates that implement user-defined functions: the behavior of these gates can be dynamically customized at any simulation experiment without re-synthesizing the net. To assess the effectiveness of the proposed approach we run the prototype on the same subset of ISCAS89 benchmark circuits reported in [5]. We assumed K=50,000 and MAX_ITER=2,000. The attained FC is computed by fault simulating the obtained CA configuration for M=500,000 clock cycles. All the experiments have been run on a Sun Ultra 5 running at 333 MHz with 256MB of memory.

Tab. 1 summarizes the results we gathered. For every circuit we reported in the second column the number n of CA-CSTP cells; the third column shows the percent testable fault coverage reached by the best CA configuration found by our algorithm. The fourth column reports the CPU time required for both the rule selection and the final fault simulation phase. We determined the untestable faults in the combinational part of each circuit by running a commercial ATPG on the same fault list.

In order to evaluate the cost of the proposed approach in terms of area overhead, we synthesized the CA identified by our system using Synopsys Design Compiler with a generic library. We then compared the area required by the CA-CSTP architecture with that required by the standard CSTP one. The last column of Tab. 1 reports the percent area overhead resulting from this evaluation.

Circuits can be divided in two main categories. The first category includes s298, s344, s382, s510, and s526: for these circuits zero or very few iterations are sufficient for reaching a 100% testable fault coverage, avoiding possible configurations affected by dependencies and/or leading to loops. The flexibility of the CA-CSTP architecture is exploited in these cases. The second category includes the other circuits, for which some hard to test faults require the generation of specific vectors; several hours of CPU time are needed in these cases to find the CA rules able to reach such a target using the current search algorithm.

In order to allow a more detailed evaluation of our results, we reported in Tab. 2 a comparison between the results of our approach with the ones published in [5], and with the performance of the original CSTP architecture, also reported in [5]. Although in that paper the overhead was computed resorting to the number of literals, instead of synthesizing the circuit, the comparison shows that our method is far more effective in terms of area overhead, while it has practically the same performance in terms of attained fault coverage.

5. Conclusions

We proposed a new BIST architecture (named CA-CSTP) for sequential circuits, which is based on CSTP and improves it by removing some of its drawbacks.

In particular, the architecture is based on a new cell that may implement any 4-input Boolean function. By suitably selecting the functions implemented by the chain cells, the architecture can reach higher fault coverage by solving the problems related to inter-cell dependencies, loops, and hard to test faults. A straightforward hill-climbing procedure is also described, which is able to heuristically search an optimal solution in the space of all the possible rules for the CA cells.

Experimental results show the effectiveness of the approach in terms of attained fault coverage and introduced area overhead. They prove that the CA-CSTP method represents a very effective solution for implementing BIST in sequential circuits. As a matter of fact, the described method is able to always provide nearly complete stuck-at fault coverage with negligible additional area overhead with respect to CSTP, while maintaining all its advantages (low timing intrusiveness, easy integration into design flow, at-speed testing).

References

[1] P.H. Bardell, W.H. McAnney, "Self-Testing of Multichip Logic Modules," *Proc. IEEE International Test Conf.*, 1982, pp. 200-204

[2] A. Krasniewski, S. Pilarski, "Circular Self-Test Path: a Low-cost BIST Technique for VLSI Circuits," *IEEE Trans. on Computer-Aided Design*, Vol. 8, No. 1, Jan. 1989, pp. 46-55

[3] L.J. Avra, E.J. McCluskey, "Synthesizing for Scan Dependence in Built-in Self-testable Designs," *Proc. IEEE International Test Conf.*, 1993, pp. 734-743

[4] F. Corno, P. Prinetto, M. Sonza Reorda, "Circular Self-Test Path for FSMs," *IEEE Design & Test of Computers*, Winter 1996, pp. 50-60

[5] N.A. Touba, "Obtaining High Fault Coverage with Circular BIST via State Skipping," *IEEE VLSI Test Symposium*, 1997

[6] P. D. Hortensius, R. D. McLeod, W. Pries, D.M. Miller, H.C. Card, "Cellular Automata-Based Pseudorandom Number Generators for Built-In Self-Test," *IEEE Trans. on Computer-Aided Design*, Vol. 8, No. 8, August 1989, pp. 842-859

[7] M. Serra, T. Slater, J. C. Muzio, D. M. Miller, "The Analysis of One-Dimensional Linear Cellular Automata and Their Aliasing Properties," *IEEE Trans. on Computer-Aided Design*, Vol. 9, No. 7, July 1990, pp. 767-778

Circuit	*n*	*TFC* %	*CPU time* *[s]*	*Overhead* %
s208	18	100.00	4,446	7.05
s298	17	100.00	292	2.90
s344	24	100.00	474	3.09
s382	24	100.00	535	2.91
s420	34	99.78	18,458	9.16
s510	25	100.00	833	2.72
s526	24	100.00	410	3.82
s641	54	99.97	77,894	5.26
s1196	32	100.00	1,911	3.52
s1423	91	100.00	55,637	3.10
s5378	199	100.00	150,401	3.49
s9234	247	98.43	52,089	4.02
s13207	700	100.00	21,421	5.16

Table 1: CA-CSTP performance

	CSTP	*[5]*		*CA-CSTP*	
Circuit	*TFC %*	*TFC %*	*Area Overhead %*	*TFC %*	*Area Overhead %*
s208	92.6	100.00	27	100.00	7.05
s298	99.2	100.00	8	100.00	2.90
s344	77.4	100.00	12	100.00	3.09
s382	95.8	100.00	10	100.00	2.91
s420	92.0	100.00	36	99.78	9.16
s510	98.4	100.00	13	100.00	2.72
s526	98.6	100.00	14	100.00	3.82
s641	98.5	100.00	12	99.97	5.26
s1196	98.9	100.00	5	100.00	3.52
s1423	100	100.00	0	100.00	3.10
s5378	86.0	100.00	32	100.00	3.49
s9234	93.7	100.00	15	98.43	4.02
s13207	99.0	100.00	4	100.00	5.16
Avg.	94.6	100.00	21	99.86	4.32

Table 2: comparing CA-CSTP with original CSTP and the state skipping method ([5]).