

# AR<sup>2</sup>T: Implementing a Truly SRAM-based FPGA On-Line Concurrent Testing

Manuel G. Gericota, Gustavo R. Alves

Miguel L. Silva, José M. Ferreira

## Abstract

*The new partial and dynamic reconfigurable features offered by new generations of SRAM-based FPGAs may be used to improve the dependability of reconfigurable hardware platforms through the implementation of on-line concurrent testing / fault-tolerance mechanisms. However, such mechanisms call for the development of new test strategies that do not interfere with the current system functionality.*

*The AR<sup>2</sup>T (Active Replication and Release for Testing) technique is a set of procedures that enables the implementation of a truly non-intrusive structural on-line concurrent testing approach, detecting and avoiding permanent faults and correcting errors due to transient faults. The experimental results presented prove the effectiveness of these solutions.*

*In relation to a previous technique proposed by the authors as part of the DRAFT FPGA concurrent test methodology, AR<sup>2</sup>T extends the range of circuits that can be replicated, by introducing a small replication aid block.*

## 1. Introduction

The advent of a new kind of SRAM-based FPGAs (Field Programmable Gate Arrays), capable of implementing fast run-time partial reconfiguration (e. g. the Virtex family from Xilinx [1]) and enabling the dynamic customisation of hardware functions to a particular system or application concurrently with system operation, considerably reinforced the advantages of using complex configurable logic devices in reconfigurable computing platforms.

Unfortunately, the smaller submicron scales used in the manufacturing of these devices increase the threat of

electromigration, due to higher electronic current density in metal traces. Also, the corresponding lower threshold voltages make them more susceptible to gamma particle radiation. Radiation interference is much more likely with larger dies, increasing the probability of failure [2, 3]. After large periods of operation, certain defects, namely those related to small manufacturing imperfections not detected by production testing, become exposed, emerging as either stuck-at (permanent) or transient faults [4].

An FPGA fault tolerance mechanism should be able to cover permanent and transient faults. In the case of permanent faults, and after the faulty elements (Configurable Logic Blocks - CLBs - or routing resources) are located, the FPGA can be reconfigured to exclude their usage. Previously unused FPGA resources can replace these faulty elements, improving dependability with a very small hardware redundancy. For transient faults, on-line partial reconfiguration enables the recovery of errors in the on-chip configuration memory cells, that modify the logic functionality, namely Single Event Upsets (SEU). Such upsets manifest themselves as permanent faults because of the change in functionality, and they cannot be recovered by traditional transient fault recovery techniques, such as rollback or roll-forward. However, the cause of the failure is actually transient [5].

A higher system dependability level can therefore only be achieved through the continuous testing of all FPGA resources throughout system lifetime, and by the introduction of fault tolerance features. In [6] a scanning methodology to dynamically test the CLBs in an FPGA was proposed, accompanied by the evaluation of the strategy adopted to implement the rotation and test schemes. This methodology guarantees that the whole FPGA is tested, if at least one unused CLB is available in the current implementation. The functionality of the CLBs currently being used by a given application is relocated in one of the CLBs already tested. After being successfully tested, unused CLBs remain available as spare blocks that may be used to replace other resources found defective, enabling the introduction of fault tolerance features,

---

♦ This work is supported by an FCT program under contract POCTI/33842/ESE/2000

provided that a pool of spare resources is continuously available to replace those eventually found defective. The replication of active CLBs, i.e. CLBs that are part of a functional block actually being used by the system, in order to release them for testing, was at the core of that approach. Their function had to be relocated in CLBs previously tested, in a way that had to be completely transparent for the system.

In this paper, a new set of replication procedures, called *AR<sup>2</sup>T* (*Active Replication and Release for Testing*), is presented. These procedures enable the implementation of a new replication and release for testing mechanism, capable of replicating active CLBs, concurrently with — but not affecting — system operation. Furthermore, they also enable the correction of errors due to transient faults in the associated configuration memory. The corresponding implementation does not require the usage of any FPGA I/O pins, since it reuses the IEEE 1149.1 Boundary Scan (BS) infrastructure [7] to access the configuration resources and to control the replication process.

This paper is organised as follows: recently proposed approaches to the test of SRAM-based FPGAs are first reviewed, followed by an overview of the problems faced when trying to implement a truly on-line testing mechanism. The following section details the *AR<sup>2</sup>T* procedures and their implementation. Some practical restrictions to their application are analysed and the possibilities offered for error recovery are also considered. Finally, directions for further research are introduced.

## 2. Background

Many different approaches based on off-line or on-line testing strategies have been proposed by different authors to test and diagnose FPGA faults.

An off-line Built-In Self-Test (BIST) technique is presented in [8-10], which exploits the reprogrammability of FPGAs in order to set up the BIST logic. A fixed number of reconfiguration test sessions are required after which the circuit is reconfigured for normal operation. Therefore, testability is achieved without any area overhead or performance penalty.

An off-line testing methodology based on a non-BIST approach, targeted to test the FPGA CLBs, is presented in [11, 12]. After a specific test configuration is set up, the FPGA Input/Output Blocks (IOBs) are used to support the external application of test vectors and to capture the test responses. In order to achieve 100% fault coverage at CLB level, different test configurations are programmed and specific sets of test vectors used in each case. Based on the same principles, a fault diagnosis method is present on [13]. Using a classical divide-to-conquer strategy,

extensive work on the structural testing of FPGA Look-Up Tables (LUT) and interconnections was also presented in [14, 15].

These approaches are restricted to manufacturing test, since they require the device to be off-line. This is unsatisfactory in highly fault-sensitive, mission-critical applications, since fault-detection latency increases greatly as a result of the off-line approach.

In order to overcome these limitations, on-line testing and diagnosis methods based on a scanning methodology were presented in [4, 16]. The idea underlying these methods is to have only a relatively small portion of the chip being tested off-line (instead of the whole chip as in previous proposals), while the remaining part continues its normal on-line operation. Testing is accomplished by sweeping the test functions across the entire FPGA. The functionality of a small number of FPGA elements is replicated on another portion of the device, before being taken off-line and tested. Then, another set of elements undergoes the same process, spanning the whole device. However, in [4] a modification in the FPGA cells structure is required to implement the replication mechanism. On the other hand, in [16] the whole system must be stopped in order to replicate an entire CLB column. Since reconfiguration is performed through the BS infrastructure, reconfiguration time is long, and it seems likely that halting the system will disturb its operation.

In [6, 17] some of the previous concepts were reused to establish an on-line FPGA testing approach that eliminated the two drawbacks that were referred. A much smaller unit of test — the CLB — was targeted and its replication accomplished without halting the system, even when the CLB is active. However, the technique was very restrictive concerning the sort of circuits that could be replicated.

In this paper, improvements on this replication technique are presented. Further to their replication function, the *AR<sup>2</sup>T* procedures also enable the correction of transient faults in the configuration memory.

Since the IEEE 1149.1 infrastructure is (re)used to access the FPGA configuration memory, in order to replicate and release for testing each CLB, we get the additional benefit of a reduced overhead at board level, since no other resources are used.

This approach can also be extended to the replication and release for testing of groups of interconnections, which proved to be easier to handle than CLBs.

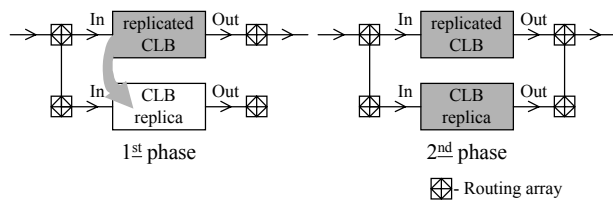
## 3. Replicating active elements

The release for testing of active CLBs requires their replication in CLBs already tested and available, in a way completely transparent to the implemented application(s). This task is not trivial due to two major issues: i)

configuration memory organisation, and ii) internal state information.

The configuration memory is partitioned into one-bit wide vertical frames grouped into larger units called columns. To each CLB column corresponds a configuration column, with multiple frames, which mixes internal CLB configuration and state information, and column routing and interconnect information.

The configuration process is a sequential mechanism that spans through some (or eventually all) CLB configuration columns. When replicating an active CLB more than one column may be affected, since its input and output signals (as well as those in its replica) may cross several columns before reaching its source or destination. Any partial reconfiguration action must therefore ensure that the signals from the replicated CLB are not broken before being totally re-established from its replica. Also important, to avoid output glitches, the functionality of the CLB replica must be perfectly stable before its outputs are connected to the system. The only possible solution is to divide the replication process in two phases, as illustrated in figure 1.



**Figure 1. Two-phase CLB replication process**

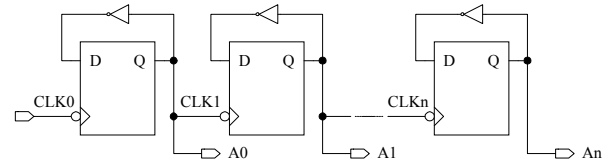
In the first phase, the internal configuration of the CLB is replicated and the inputs of both CLBs are placed in parallel. Due to the low-speed characteristics of the reconfiguration (BS) interface, the reconfiguration time is relatively long when compared with the system speed of operation. Therefore, the outputs of the CLB replica will be perfectly stable before being connected to the circuit, in the second phase. Both CLBs must remain in parallel for at least one system clock cycle to avoid output glitches. Notice that rewriting the same configuration data does not generate any transient signals, so this process does not affect the remaining resources covered by the rewriting of the configuration frames that are needed to carry out the replication.

Another major requirement for the success of the replication process is the correct transferring of state information. If the current CLB function is purely combinational, a simple read-modify-write configuration procedure will suffice to accomplish the replication process. However, in the case of a sequential function, the internal state information must be preserved and no writes could be lost during the copying process. In the Virtex FPGA family, it is possible to read the value of a register, but not to perform a direct write operation. Moreover,

when dealing with active CLBs, state information may change between the read and write of a register, causing a coherency problem. By this reason, no time gap between the two operations may exist. The solution to this problem depends on the type of implementation. In this paper we study three different implementation cases:

1. synchronous free-running clock circuits;
2. synchronous gated-clock circuits, and;
3. asynchronous circuits.

When dealing with synchronous free-running clock circuits, the two-phase replication process that was previously described may solve this problem. Between the first and the second phase, the CLB replica has the same inputs as the replicated CLB and all its four Flip-Flops (FFs) acquire the state information, even if the system frequency of operation is an order of magnitude lower than the BS test infrastructure frequency used for reconfiguration purposes. Several experiments made using this class of circuits have shown the effectiveness of this method. No loss of state information or the presence of output glitches was observed. This method is also effective when dealing with asynchronous circuits like the ripple counter illustrated in figure 2, if the slowest “clock” period (CLK<sub>n</sub>) is higher than the time interval between the first and the second phases.

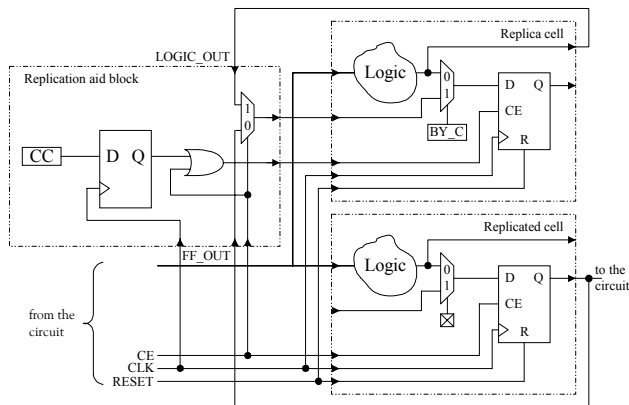


**Figure 2. Ripple Counter**

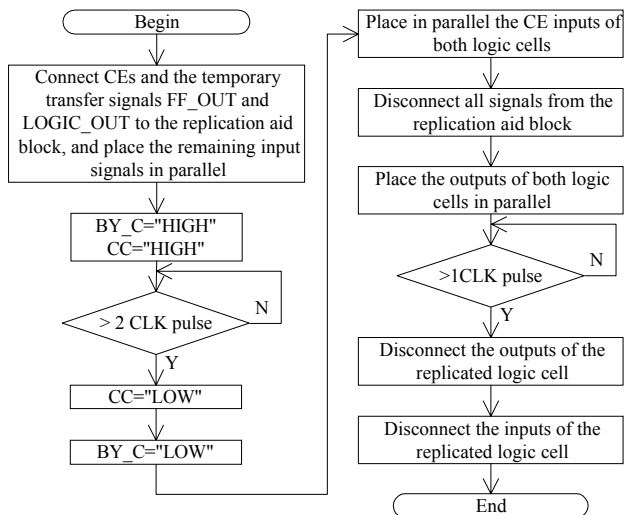
Despite the effectiveness of this solution, its usefulness is very restricted. A broad range of applications use synchronous gated-clock circuits, instead of free-running clocks, where input acquisition is controlled by the clock enable signal. In such cases, we cannot assure that this signal will be active during the replication process and that the value present at the input of the replica FFs will be captured. Setting this signal as part of the replication process is not viable either, since the value present at the input of the replica FFs, at this time, could be different from its last capture by the replicated FFs, and a coherency problem would arise. Furthermore, the state of FFs could be updated during the execution of the replication process.

To solve this problem we used a replication aid block, which manages the transference of the state value from the replicated FFs to the replica FFs, while enabling their update by the circuit at any moment, without losing the new state information or delaying the replication process. The whole replication scheme is represented in figure 3. By reasons of simplicity, only one logic cell is represented. Each CLB comprises four of these cells,

which, for this purpose, can be considered individually. In figure 4 is shown the flow diagram of the replication process.



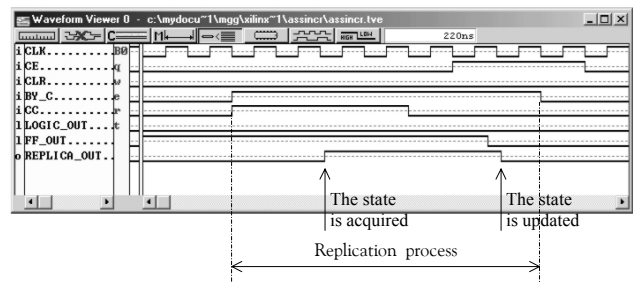
**Figure 3. Synchronous gated-clock circuit replication scheme**



**Figure 4. Replication process sequence algorithm**

The 2:1 multiplexer present in the replication aid block is controlled by the clock enable signal (CE) of the replicated FF. If this signal is not active, the output of the replicated FF (FF\_OUT) is applied to the input of the replica FF, through a multiplexer controlled by the bypass control signal (BY\_C). A clock enable signal, generated by the replication aid block (capture control signal - CC), forces the replica FF to hold the transferred value. The replica FF acquires the state information present in the replicated FF. If the CE signal is active or is activated during this process, the multiplexer selects the LOGIC\_OUT signal and applies it to the input of the replica FF, which is updated at the same time and with the same value as the replicated FF, guaranteeing state coherency. Figure 5 shows the waveform simulation of a

state transfer and update during the replication process. No loss of information is verified during this process.



**Figure 5. Simulation of a state transfer and update**

The control signals CC and BY\_C are configuration memory bits whose values are driven through reconfiguration of the configuration memory. BY\_C directs the state signal to the input of the replica FF, while CC enables its acquisition. It is therefore possible to control the whole replication process through the BS infrastructure, and as such no extra pins are required. Each replication aid block occupies one CLB slice, therefore two extra CLBs will be needed to implement this process since four of these blocks are required to replicate the four logic cells of a CLB. The FF included in the CC net is there simply as a consequence of the structure of the CLB slice, and does not play any role in the process.

After the state has been transferred, the input signals involved in the process are placed in parallel, all the signals to and from the replication aid block are disconnected, and the outputs are also placed in parallel. After at least one clock cycle the outputs of the replicated block are disconnected and then also the inputs. The CLB is released and is ready to be tested. Each of these steps (corresponding to a square in the flow diagram shown in figure 4) implies a new reconfiguration file. A total of 9 files are needed just to get the replication done, instead of the only 3 needed when dealing with free-running clock circuits. However, in most cases, their size is much smaller. To change the value of CC and/or BY\_C only one configuration frame is needed, which is around 2 Kbit (220  $\mu$ s for a 20 MHz BS test frequency).

Practical experiments performed using a Virtex XCV200 over the ITC'99 Benchmark Circuits from the Politécnico di Torino [18] demonstrated the effectiveness of our approach. The average number of bits needed to replicate each CLB using this strategy, including configuration and command frames, was around 230 Kbit. At an operation frequency of the BS infrastructure of 20MHz, the mean time for a complete replication of a CLB was 24 ms. Therefore, the replication of all CLBs of a XCV200 device (array size =28x42=1176 CLBs) will require 28 seconds, with only one CLB replicated at a time.

The ITC'99 Benchmark Circuits are purely synchronous with only one single-phase clock signal present. However, this approach is also applicable to multiple clock/multiple phase circuits, since only one clock signal is involved in the replication process at a time, provided that the slowest "clock" period is higher than the duration of the replication process.

This method is also effective when dealing with asynchronous circuits, where D latches are used instead of FFs. In this case, the CE signal is replaced in the latch by the input control signal. Data present in the D input is stored in the gated D latch when the input control signal changes from '1' to '0'. The same replication aid block is used and the same replication sequence is applied. The register present in the replication aid block may be configured as a latch, instead as a FF, if this is preferred or if no adequate clock signal is available.

The same two-phase procedure is effective on the replication and release for testing of local and global active interconnections. The interconnections to be replicated are first duplicated to establish an alternative path, and then disconnected, being free for testing.

The LUTs in the CLB can also be configured as memory modules (RAMs) for user applications. However, the extension of this on-line replication concept to the replication of LUT/RAMs is not viable. The content of the LUT/RAMs could be read and written through the configuration memory, but if there is a write attempt during the replication interval, there is no possible mechanism, other than to stop the system, capable of ensuring the coherency of the values, as stated in [5]. Furthermore, since frames span an entire column of CLB slices, the same LUT bit in all of them is updated with a single write command. We must ensure that either all the remaining data in the slice is constant, or it is also modified externally through partial reconfiguration. Even if not being replicated, LUT/RAMs should not lie in any column that could be affected by the replication procedure. However, while it is not possible to replicate LUTs configured as RAMs, nothing prevents the structural testing of the RAM mode of the LUTs when testing the CLBs.

This method could be used to replicate more than one CLB at a time, depending on the overall test strategy. This will improve the scalability of the process, which is important if we are dealing with large FPGAs. However, problems due to the limited number of interconnections available limit the number of CLBs able to be replicated at each time.

#### 4. Error recovery

The replication procedure used with synchronous free-running clock circuits did not perform a true state

transference operation, but rather an acquisition of the values present at the input of the replica FFs, which are the same as those present at the input of the replicated FFs. Therefore, the acquired state information is correct, despite any fault that may affect the replicated CLB FFs. As a consequence, and after the replication process, the outputs of the CLB replica always display the correct values, automatically correcting any faulty behaviour. On the other hand, when replicating synchronous gated-clock circuits (or asynchronous circuits), a truly state transference operation is performed. Therefore, if a permanent fault in the replicated CLB affects the value held by one of the FFs (or latches), this fault is propagated to the replica CLB and will remain active until that FF (or latch) is updated. The fault in the replicated CLB will be detected during the subsequent test phase and possibly the CLB will be flagged as defective, meaning that it will not be used again in a later reconfiguration.

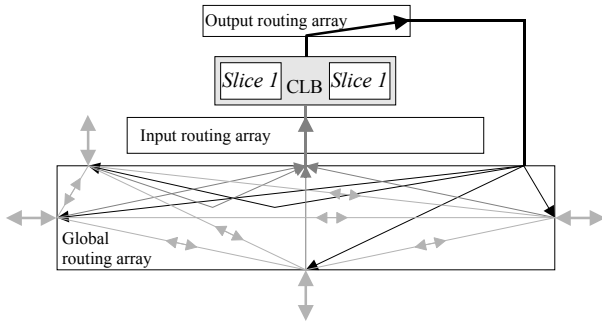
Depending on the method used to create the reconfiguration files, the replication procedure can recover errors caused by transient faults in the on-chip configuration memory cells that modify the circuit function. A typical example of such errors is SEUs in space environments. Since Virtex FPGAs enable readback operations, a completely automatic read-modify-write procedure could be implemented to replicate the CLBs using local processing resources. In this case, any transient fault in the configuration memory is not corrected, being propagated to the replica. On the other hand, if the reconfiguration files are generated from the initial configuration file stored in an external memory, any error due to SEUs is corrected when affected blocks are replicated.

#### 5. Behaviour of routing resources

The successful testing of the CLB replica assures its good functionality, but the replicated CLB might be faulty. When the inputs and outputs of both CLBs are placed in parallel, we may be interconnecting nodes with different voltage levels. Due to the impedance of the routing switches, this apparent "short-circuit" behaves as a voltage divider, limiting the current flow in the interconnection. Therefore, no damage results to the FPGA, as proved by extensive experimental essays. Since we are dealing with digital circuits, the analog value resulting from the voltage divider leads to a well defined value (logic 0 or logic 1) when it goes through a buffer during the routing, or at the input of the next CLB or IOB. No logic value instability was observed during the essays.

Each CLB has three associated routing arrays: two local arrays (input and output) and one global array. The routing resources in these arrays may be unidirectional or bi-directional, as indicated in figure 6. No routing

resources are available in the local arrays to establish direct interconnections with other CLB's, so the interconnections required by the replication process can only be done through the global routing array. Between local and global routing arrays only unidirectional routing resources are available.



**Figure 6. CLB routing resources**

Since no fault at any of the replicated CLB inputs may propagate backwards, the logic values present at the inputs of the replica CLB will not be affected by the interconnection, even if the replicated CLB is faulty. As such, all CLB replica inputs will always reflect the correct values.

This is also true when replicating active interconnections where faults in the replicated net are automatically corrected when the replication takes place. The test of interconnections could also be achieved using the same method described in [6], with CLBs under test being replaced by a set of wires under test.

## 6. Conclusion

The set of procedures presented in this paper enables the implementation of a truly on-line testing mechanism that, reusing the standard BS infrastructure and the novel partial dynamic reconfiguration features of recent FPGA devices, improves reconfigurable hardware systems dependability in a way that is completely transparent to the system operation.

Our current work focuses on the development of computational tools to introduce a higher degree of automation in the whole process. The final objective is to automatically generate the reconfiguration frames from the initial configuration file.

## 7. References

[1] The Programmable Logic Data Book, available at <http://www.xilinx.com>

- [2] F. Hanchek, S. Dutt, "Methodologies for Tolerating Cell and Interconnect Faults in FPGAs", *IEEE Trans. on Computers*, Vol. 47, No. 1, pp. 15-33, Jan. 1998.
- [3] J. Lach, H. W. Mangione-Smith, M. Potkonjak, "Low Overhead Fault-Tolerant FPGA Systems", *IEEE Trans. on VLSI Systems*, Vol. 6, No. 2, pp. 212-221, June 1998.
- [4] N. R. Shnidman, H. W. Mangione-Smith, M. Potkonjak, "On-Line Fault Detection for Bus-Based Field Programmable Gate Arrays", *IEEE Trans. on VLSI Systems*, Vol. 6, No. 4, pp. 656-666, Dec. 1998.
- [5] W. Huang, E. J. McCluskey, "A Memory Coherence Technique for Online Transient Error Recovery of FPGA Configurations", *Proc. 9<sup>th</sup> ACM Int. Symp. on FPGAs*, pp. 183-192, Feb. 2001.
- [6] M. G. Gericota, G. R. Alves, M. L. Silva, J. M. Ferreira, "DRAFT: An On-Line Fault Detection Method for Dynamic and Partially Reconfigurable FPGAs", *Proc. 7<sup>th</sup> IEEE Int. On-Line Testing Workshop*, pp. 34-36, July 2001.
- [7] *IEEE Standard Test Access Port and Boundary Scan Architecture* (IEEE Std 1149.1), IEEE Std. Board, May 1990.
- [8] C. Stroud, S. Konala, P. Chen, M. Abramovici, "Built-In Self-Test of Logic Blocks in FPGAs (Finally, A Free Lunch: BIST Without Overhead!)", *Proc. 14<sup>th</sup> IEEE VLSI Test Symp.*, pp. 387-392, April 1996.
- [9] C. Stroud, E. Lee, M. Abramovici, "BIST-Based Diagnostic of FPGA Logic Blocks", *Proc. Intl. Test Conf.*, pp. 539-547, Nov. 1997.
- [10] C. Stroud, S. Wijesuriya, C. Hamilton, M. Abramovici, "Built-In Self-Test of FPGA Interconnect", *Proc. Intl. Test Conf.*, pp. 404-411, Nov. 1998.
- [11] W. K. Huang, F. J. Meyer, X. Chen, F. Lombardi, "Testing Configurable LUT-Based FPGA's", *IEEE Trans. on VLSI Systems*, Vol. 6, No. 2, pp. 276-283, June 1998.
- [12] W. K. Huang, F. J. Meyer, F. Lombardi, "An approach for detecting multiple faulty FPGA logic blocks", *IEEE Trans. on Computers*, Vol. 49, No. 1, pp. 48-54, Jan. 2000.
- [13] T. Inoue, S. Miyazaki, H. Fujiwara, "Universal Fault Diagnosis for Look-up Table FPGAs", *IEEE Design and Test of Computers*, Vol. 15, N° 1, pp. 39-44, Jan.-Mar. 1998.
- [14] M. Renovell, J. M. Portal, J. Figueras, Y. Zorian, "RAM-Based FPGA's: A Test Approach for the Configurable Logic", *Proc. IEEE Int. Conf. on Design, Automation and Test in Europe*, pp. 82-88, Feb. 1998.
- [15] M. Renovell, J. M. Portal, J. Figueras, Y. Zorian, "Testing the interconnect of RAM-based FPGAs", *IEEE Design and Test of Computers*, Vol. 15, N° 1, pp. 45-50, Jan.-Mar. 1998.
- [16] M. Abramovici, C. Stroud, S. Wijesuriya, C. Hamilton, V. Verma, "On-Line Testing and Diagnosis of FPGAs with Roving STARS", *Proc. 5<sup>th</sup> IEEE Int. On-Line Testing Workshop*, pp. 2-7, July 1999.
- [17] M. G. Gericota, G. R. Alves, M. L. Silva, J. M. Ferreira, "Dynamic Replication: The Core of a Truly Non-Intrusive SRAM-based FPGA Structural Concurrent Test Methodology", *Proc. 3<sup>th</sup> IEEE Latin-American Test Workshop*, pp. 70-75, Feb. 2002.
- [18] Politécnico di Torino ITC'99 benchmarks, available at <http://www.cad.polito.it/tools/itc99.html>