Improving the dependability of dynamically reconfigurable hardware by concurrent replication of active resources

José M. M. Ferreira, Manuel G. Gericota

FEUP / DEEC, R. Roberto Frias, 4200-465 Porto PORTUGAL jmf@fe.up.pt
ISEP / DEE, R. Ant. Bern. Almeida, 4200-072 Porto PORTUGAL mgg@dee.isep.ipp.pt

Abstract - This presentation describes a low-level technique to replicate active resources (i.e. resources that are being used by functions that are currently running) in dynamically reconfigurable FPGAs, with the main objective of releasing them to be tested in a non-intrusive way. This technique may be used to support i) Online concurrent testing to detect any faults that emerge during system operation, ii) Enhanced fault tolerance[1] (restoring the reliability index by replacing a defective resource), and iii) Reallocation of the FPGA logic space to prevent excessive delays or wasting resources due to fragmentation. All solutions proposed reuse the IEEE 1149.1 (JTAG) test access port and boundary-scan architecture to ensure a low-cost / low overhead implementation.

1 Introduction

SRAM-based field-programmable gate arrays (FPGAs) comprise an array of uncommitted configurable logic blocks (CLBs) and input / output blocks (IOBs), which are interconnectable via configurable routing resources. A large number of SRAM cells define the operation of all such blocks and interconnections. Dynamically reconfigurable FPGAs (DR-FPGAs) that support partial reconfiguration enable the device logic space to be reconfigured selectively, i.e. the redefinition of logic functions will only address the required subset of the device logic space. Any function implemented on the remaining logic space will continue to operate undisturbed while the reconfiguration process takes place. Due to some discrepancies in terminology among various authors, it is useful to state that the expression *dynamically reconfigurable FPGA* will be used throughout this work to refer to those devices that support partial reconfiguration. DR-FPGAs enable the implementation of virtual hardware by appropriate scheduling of applications. Efficient time and space management enable the implementation of applications which in total may exceed 100% of the logic space available.
Technological improvements enabled the recent introduction of self-reconfigurable FPGAs (SR-FPGAs), where an internal function may control the reconfiguration of the device logic space. SR-FPGAs are able to further reduce the cost and size of adaptive systems, by implementing online management tasks within the FPGA itself.

[1] The extension of this technique to enhance fault-tolerant architectures has just started in May 2005 and is being financed by the Fundação para a Ciência e a Tecnologia (FCT contract number POSC/EEA-ESE/55680/2004).

The increasing amount of logic available in FPGAs and the reduction of the reconfiguration time, partly due to the possibility of partial reconfiguration, extended the concept of virtual hardware to the implementation of multiple applications sharing the same logic resources in the spatial and temporal domains. However, higher complexity comes hand-in-hand with higher vulnerability. Transient phenomena e.g. single-event upsets (SEUs) or single-event transients (SETs), may lead to modifications in the configuration memory or to state modifications, particularly for larger die sizes [1, 2]. This problem is non-negligible at ground level, and it is further aggravated when these devices are used in space applications, where the cosmic radiation causing SEUs and SETs is far more important. On the other hand, the threat of electromigration also increases with smaller technological scales, and may lead to permanent physical damage. Even when the cause of the problem is not permanent (i.e. modifications in the configuration memory due to an SEU), the circuit may fail if corrective action is not taken in due time. Altogether, these factors indicate that good production tests are no longer enough to guarantee fault-free operation. Error conditions or physical defects may (and will) emerge in the field, and the only way to ensure reliability is to implement online concurrent fault detection and mitigation solutions. The concurrent replication of active resources herein presented enables an effective framework to ensure dependable system design, comprising the following components:

- **Online concurrent testing**: Active replication is used as the basis of a non-intrusive concurrent testing strategy, whereby each resource is replicated (functional and state information) and released for testing. Fault detection latency is related to the size of the FPGA — an emerging defect will only be detected when the affected resource is again under test. Error conditions may eventually cause system malfunction, if the fault latency is higher than the system inertia.
- **Enhanced fault tolerance**: Spatial redundancy architectures may be a solution when fault detection latency is not acceptable. However, if more than one module fails, the system may also fail (a triple redundancy system can only tolerate single module failures). Concurrent testing will identify the defective resource, which will be replaced to reestablish the reliability index. A simpler form of replication suffices in this case, since state information does not have to be transferred.

Concurrent testing and fault tolerance are necessary, but may not be sufficient to guarantee sustainable performance after many reconfiguration sessions. Increasing propagation delays due to poor rerouting and excessive fragmentation of the FPGA logic space are two major reasons of concern in this context. A dependable framework for dependable system design must therefore also support concurrent defragmentation, to enable the activation / deactivation of functions as needed at any given moment. The research work done so far is not divided equally among concurrent testing, enhancement of fault-tolerance, and concurrent defragmentation. Online concurrent testing concentrated most of our efforts, and has been validated via practical experimentation using a DR-FPGA from Xilinx. Extensive experimental data is available in this case, and an extract will be presented to validate the solutions presented. Experimental data is not yet available for the two other areas. Fault tolerance strategies are the main focus of current research, particularly in what concerns the implementation of self-repair methods using SR-FPGAs. Logic space defragmentation is an area that waits for the opportunity to develop appropriate higher

level management solutions that are able to exploit the replication technique proposed. Those areas will be therefore presented in less detail, but may lead to stimulating discussions and also to possible cooperation efforts in the near future.

2 Concurrent replication of active resources

Several off-line and online strategies have been proposed to test and diagnose FPGA faults [3-16]. The concurrent test approach herein presented reuses some of the ideas described in the literature, but eliminates their drawbacks by using *active replication*, which enables the relocation of the functionality allocated to each CLB, without halting the system. This approach is feasible even when the CLB is active, i.e. when it is part of an implemented function that is actually being executed [16]. A dynamic rotation mechanism ensures that all FPGA CLBs are released and tested within a given latency. The exclusive (re)use of the Boundary Scan (BS) test infrastructure to release and test the CLBs brings the additional benefit of reduced overhead at board level, since no other resources (other than those of the FPGA itself) are used.

Releasing active CLBs for testing requires their replication into CLBs already tested and available, in a way that is completely transparent to the application(s) that are currently running. This task is not trivial due to two major issues: i) configuration memory organization, and ii) internal state information.

The configuration memory may be visualized as a rectangular array of bits, which are grouped into one-bit wide vertical frames, extending from the top to the bottom of the array. The atomic unit of configuration is one frame — it is the smallest portion of the configuration memory that can be written to or read from. These frames are grouped together into larger units called columns. Each CLB column has an associated configuration column, with multiple frames, which mixes internal CLB configuration and state information, and column routing and interconnect information. The organization of the entire configuration memory into frames enables the online concurrent partial reconfiguration of the FPGA.

The configuration process is a sequential mechanism that spans through some (or eventually all) CLB configuration columns. More than one column may be affected during the replication of an active CLB, 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 procedure must ensure that the signals from the replicated CLB are not broken before being totally re-established from its replica. It is also important to ensure that the functionality of the CLB replica must be perfectly stable before its outputs are connected to the system, so as to avoid output glitches. The replication of CLBs is divided into two phases, as illustrated in figure 1.

In the first phase, the internal configuration of the CLB is copied and the inputs of both CLBs are placed in parallel. Due to the low-speed characteristics of the configuration interface used (the 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 the remaining resources addressed by the configuration frames are not affected. Another major requirement for the success of the replication process is the correct transferal of state information. If the current CLB function is purely combinational, a simple read-modify-write 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 write-operations may be lost while this process goes on.

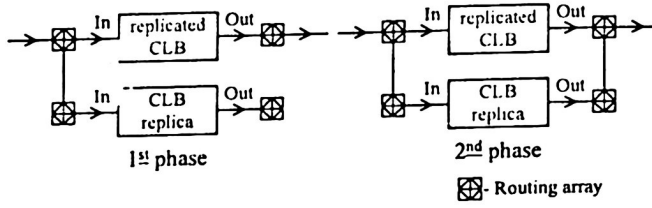


Fig. 1. Two-phase CLB replication process

When dealing with synchronous free-running clock circuits, the two-phase replication process that was previously described solves 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 acquire the state information, even if the system clock frequency is an order of magnitude lower than the clock frequency of the BS infrastructure, which is used for reconfiguration purposes. Several experiments made using this class of circuits have shown the effectiveness of this method in the replication of active CLBs. No loss of state information or the presence of output glitches was observed.

A different situation is present in the case of synchronous gated-clock circuits, where input acquisition by the flip-flop is controlled by a clock enable signal. In such cases, it is not possible to ensure that this signal will be active during the replication process, and that the value at the input of the replica flip-flops will be captured. On the other hand, it is not feasible to set this signal as part of the replication process, because the value present at the input of the replica flip-flops might differ from the one captured by the replicated flip-flops, in which case a coherency problem will occur. Furthermore, the state of the flip-flops could be updated during the replication process. A replication aid block is used to solve this problem. This block manages the transferal of state information from the replicated flip-flops to the replica flip-flops, while enabling its update by the circuit, at any moment, without losing new state information or delaying the replication process.

Practical experiments performed using a Virtex XCV200 over the ITC'99 Benchmark Circuits from the Politecnico di Torino [17], demonstrated the effectiveness of the proposed approach. These circuits are purely synchronous with only one single-phase clock signal present. However, the procedures presented are also applicable to multiple clock / multiple phase circuits, since only one clock signal is involved in the replication process at each time, provided that the slowest "clock" period is higher than the duration of the replication process. The proposed method is also effective

when dealing with asynchronous circuits (using D latches instead of flip-flops), where the same replication aid block and the same replication sequence may be used.
A further remark must be made concerning the relocation of routing resources. Since different paths are used while paralleling the original and replica interconnections, each of them will have a different propagation delay. This means that if the signal level at the output of the CLB source changes, the signal at the input of the CLB destination will show an interval of fuzziness. However, the impedance of the routing switches will limit the current flow in the interconnection, and hence this behaviour does not damage the FPGA. Nevertheless, and for transient analysis, the propagation delay associated to the parallel interconnections, shall be the longer of the two paths.
The LUTs in the CLB can also be configured as memory modules (RAMs) for user applications. However, the extension of this concept to the replication of LUT/RAMs is not feasible. The content of the LUT/RAMs may be read and written through the configuration memory, but there is no mechanism, other than to stop the system, capable of ensuring the coherency of the values, if there is a write attempt during the replication interval, as stated in [2]. Furthermore, since frames span an entire column of CLB slices, a given bit in all slices is written with the same command. Therefore, it is necessary to ensure that either all the remaining data in the slice is constant, or it is also changed externally through partial reconfiguration. Even not being replicated, LUT/RAMs should not lie in any column that could be affected by the replication process.
Depending on the method used to create the reconfiguration files, the replication procedure can also recover from errors caused by transient faults in the on-chip configuration memory cells. A typical example of such errors are SEUs in space environments, which modify the logic function originally implemented in the FPGA. 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 propagated and will affect the functionality of the CLB 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 the affected blocks are replicated.

3 Online concurrent testing

The configurable structure of the CLB requires the use of a minimum number of test configurations to perform a complete test of its structure, with a specific set of test vectors applied to each test configuration. Since the implementation structure of the CLB primitives (LUTs, multiplexers, flip-flops) is not known, a hybrid fault model was considered [7] (see also [10, 11] for an extensive study concerning FPGA fault models). The BS infrastructure is reused to apply the 40 test vectors required to test each CLB, and to capture the test responses. Since the application of test vectors via the BS register would be intrusive, a user test register must be used (the Virtex family enables two BS user registers). The register created for this purpose comprises 13 cells, corresponding to the maximum number of CLB inputs, and is fully compliant

with the IEEE 1149.1 standard. The seven CLBs occupied by this register and the two CLBs occupied by the replication aid block, are the only FPGA hardware overhead that is implied by our test methodology. It accounts for less than 1% of the CLB resources of the Xilinx Virtex XCV200 device (array size = 28x42 CLBs). Each Virtex CLB comprises two slices that are exactly equal. In total, each CLB has 13 inputs (test vectors are applied to both slices of all CLBs under test simultaneously) and 12 outputs (6 from each slice). Since the outputs of each slice are captured independently, fault location can be resolved to a single slice.

The dynamic rotation mechanism used for releasing CLBs to be tested should have a minimum influence in the system operation, as well as reduced reconfiguration cost overhead. This cost depends on the number of reconfiguration frames needed to replicate and release each CLB, since a great number of frames would imply a longer test time. The impact of this process in the overall system operation is mainly related to the delays imposed by re-routed paths, which may now be longer (reducing the maximum frequency of operation). Two strategies were considered to rotate a free CLB across the FPGA logic space: horizontal and vertical. In the horizontal rotation strategy the free CLB rotates along a horizontal path covering all the CLBs in the array. The replication is performed between neighbouring CLBs, due to scarcity of routing resources, and to avoid higher path delays. The same principle applies to the vertical rotation strategy, where the free CLB is rotated along a vertical path.

The results of practical experiments performed over a subset of the ITC'99 benchmarks using these two strategies are presented in table 1. Generally, the vertical rotation scheme is seen to perform much better, be it in terms of the reduction in the maximum frequency of operation (7% in the average for vertical rotation, against 18% average for horizontal rotation) or in what concerns the size of the partial reconfiguration files. The vertical organization of the reconfiguration vectors explains why the size of the reconfiguration files is in the average 20% higher for horizontal rotation, which always involves two columns.

The influence over the maximum frequency of operation is explained by the pair of dedicated paths per CLB that propagate carry signals vertically between adjacent CLBs. When the rotation process breaks a dedicated carry path, due to the insertion of the free CLB, the propagation of this carry signal between the nearest adjacent CLBs (above and below the free CLB) is re-established through generic routing resources, increasing the path delay. If the implemented circuit has one or more of these carry signals, the horizontal rotation would break all the carry nets, increasing path delays, but the vertical rotation would break only one of them at a time. In this case, the vertical strategy becomes preferable. When no carry signals are used, two other factors must be considered: i) the number of signals with high fanout, and ii) the placement shape (rectangular, square, circular, etc.) and orientation (horizontal, vertical) of the circuits implemented in the FPGA. In rectangular / horizontal implementations, and when many high fanout signals are present, the horizontal strategy is preferable, since the maximum frequency of operation is less degraded (this could be more important than the size of reconfiguration files, when dealing with high-speed applications).

If we look into the mean size of the reconfiguration files per CLB, table 1 shows that the vertical rotation scheme also performs better (nearly 10% in average). This table shows that the mean size of the reconfiguration files per CLB increases as the number

of CLBs used in the implementation also increases, because it becomes more difficult to find good routing alternatives to those signals involved in the replication process.

Table 1. Frequency deviation, reconfiguration file size, and size per CLB

Circuit Refer.	Number of CLBs	Maximum frequency deviation (%)		Size of reconfiguration files in total (bytes)		Size of reconfiguration files per CLB (bytes)		Ratio (horiz. / vert.)
		Vert.	Horiz.	Vert.	Horiz.	Vert.	Horiz.	
B01	6	-5,5	0,0	48 350	56 102	8 058	9 350	16,0
B02	1	0,0	0,0	7 016	10 623	7 016	10 623	51,4
B03	11	-1,9	-4,9	120 705	138 484	10 973	12 589	14,7
B04	54	-6,1	-29,3	548 595	665 419	10 159	12 322	21,3
B05	103	-17,3	-36,9	1 130 985	1 286 031	10 980	12 485	13,7
B06	5	-2,7	0,0	45 291	53 503	9 058	10 700	18,1
B07	31	-23,6	-37,8	354 367	425 214	11 431	13 716	20,0
B08	17	-5,8	-5,8	150 093	178 339	8 829	10 490	18,8
B09	12	-1,8	-4,9	112 107	129 855	9 342	10 821	15,8
B10	20	-7,5	-7,6	195 571	245 455	9 778	12 272	25,5
B11	39	-10,5	-36,0	500 261	614 093	12 827	15 745	22,8
B12	119	0,0	-1,2	1 275 804	1 631 953	10 721	13 713	27,9
B13	37	-4,3	-42,8	258 827	332954	6 995	8 998	28,6
B14	333	-13,5	-47,8	5 195 444	6 070 485	15 601	18 229	16,8

The back and forth dynamic free-CLB rotation across the chip implies a variable latency. The time to again reach a given CLB alternates, according to the rotation direction, between a maximum and a minimum value, depending on the device size (number of CLB columns and rows). The fault detection latency is bounded by the following limits:

$$\tau_{scan_{max}} = ((\#CLB_{rows} \times \#CLB_{columns}) - 1) \times 2 \times (t_{reconf} + t_{test})$$

$$\tau_{scan_{min}} = 2 \times (t_{reconf} + t_{test})$$

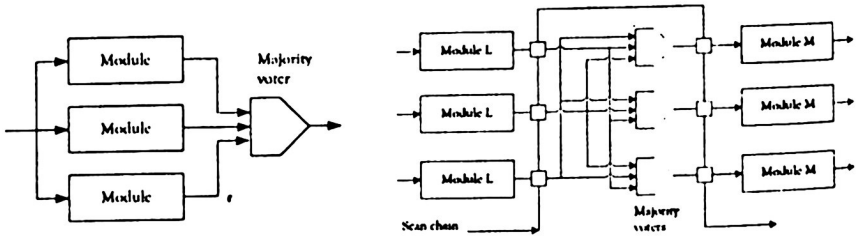
(t_{reconf} is time needed to complete a CLB replication and t_{test} is the time needed to test a free CLB)

When the rotation process is complete, the initial routing is restored. The whole process may then be repeated or paused, depending on the overall test strategy. In order to estimate the worst case fault detection latency, we must take into account the time needed to carry out each step of the proposed approach. The replication of each CLB dominates the total time per CLB, in particular in the case of those circuits requiring a replication aid block (in which case approx. 35 ms are required to complete the replication process). The various test configurations required to complete a structural test of each CLB, together with the time required to shift in and out test data, contribute with approximately 10 ms, leading to a total time per CLB

that is close to 45 ms. In a typical scenario, like those that correspond to the ITC'99 benchmark circuits (25% of CLBs requiring a replication aid block, 50% of CLBs not requiring such blocks, and 25% of empty CLBs), the total test time to cover the 1.176 CLBs in the XVC200 is above 40 seconds.

4 Enhanced fault tolerance

Active replication enables non-intrusive concurrent testing and logic space defragmentation, but cannot avoid performance degradation due to higher propagation delays, in the case of signals that are re-routed via longer interconnection paths. On the other hand, fault detection latency enables the propagation of fault effects, which may eventually lead to irreversible malfunction of the whole system. These restrictions may not be important in many application domains, and namely when the fault detection latency is small compared to system inertia. However, in the case of mission-critical applications, a higher reliability solution must be devised.
Triple modular redundancy (TMR) is the best known form of spatial redundancy, and may be represented as shown in figure 2.a. Three identical modules (M) receive the same inputs and drive a majority-voter that produces the circuit output. When more than one module fails, the circuit fails. Tolerance to multiple module failures may of course be achieved at the cost of providing higher redundancy, leading to NMR architectures. Replication may take place at various hierarchical levels, so each module may be a simple gate, or a much more complex resource, including mid-range functional blocks, components (e.g. a microprocessor), or even a complete system. In the basic configuration shown in figure 2.a, the reliability of the circuit will depend on the reliability of the voter (if the voter fails, the circuit fails). Special design and implementation techniques may be used to improve the reliability of the voter circuit. However, if such solutions are not considered satisfactory, the voting element itself may be replicated as well, leading to what is known as N-NMR architectures. Additionally, design diversity may be enforced to further enhance reliability.



a. TMR with a single voting element **b.** T-TMR with scan fault detection

Fig. 2. Fault tolerance via spatial redundancy

Particularly in the case of low level modular redundancy, DR-FPGAs bring two important advantages to the cost x benefit model of NMR implementations: **i)** since

each function may be implemented only when needed (and afterwards removed to release FPGA floor space), the additional spatial requirements of modular redundancy solely address those blocks that have to be implemented at any given time; **ii)** any defective resource that caused module failure may be identified and replaced, restoring the reliability index. The occurrence of a fault will be indicated by a discrepancy at the output of a module or voter. An internal 1149.1 scan chain able to capture the output of modules and voters, as shown in figure 2.b, enables the identification of the defective block. The fault masking properties of modular redundancy will ensure that circuit operation will not be affected, provided that a second module / voter in the same set does not fail, before the fault correction procedure is complete. The fault detection procedure launches a background task to readback the configuration bitstream of the area where the affected module is located. If an incoherency is found, the test controller restores the original configuration and eliminates the cause of the failure. If no error in the configuration bitstream is detected after the readback-and-compare operation, but the fault persists, the most probable reason is the existence of a physical defect. When the defective resource is found, it is flagged to avoid further usage and replaced to restore the reliability index. Notice that this solution confines the fault effect to the defective module (its output is masked), but does not eliminate the fault detection latency. However, the worst case fault latency is no longer given by the time taken to test the full CLB matrix, since the concurrent fault detection procedure is now looking for incoherencies at the module and voter outputs. The important thing to do is to identify the cause of the incoherency and to eliminate it. Appropriate action (i.e. correcting the contents of the reconfiguration memory or replicating the defective CLB) will then reestablish the reliability index and bring the circuit back to its full fault-tolerance features.

5 Conclusion

The active replication technique described in this presentation enabled the proposal of a truly non-intrusive online concurrent testing solution that was validated using an XCV200-based prototyping board. Notice that the structure of the current Xilinx Virtex II devices is the same as in the older XCV200 that was available when this project started (the embedded microprocessor cores and the higher number of reconfigurable resources make no difference from this point of view). When fault detection latency cannot be tolerated, an enhanced T-TMR architecture enables fault diagnosis and guarantees fault-free operation and fast recovery of the reliability index. The same active replication technique may also be used to defragment the FPGA logic space, ensuring sustainable performance by preventing excessive path delays and reducing the waiting time imposed on incoming functions [18-21]. The development of enhanced fault-tolerant T-TMR architectures is the subject of a new R&D project that has just started in May 2005. The main objective of this new project is to improve the solution proposed in the previous section, by using SR-FPGAs to develop self-repair architectures. An embedded T-TMR microprocessor controls the internal configuration access port (ICAP) of SR-FPGAs and triggers the self-reconfiguration procedure when a defective resource is identified [22].

References

1. R. Baumann, "Soft Errors in Advanced Computer Systems," *IEEE Design and Test of Computers*, Vol. 22, No. 3, pp. 258–266, May-June 2005.
2. W. Huang, E. J. McCluskey, "A Memory Coherence Technique for Online Transient Error Recovery of FPGA Configurations," *Proc. of the 9th ACM Int. Symposium on Field-Programmable Gate Arrays*, pp. 183-192, February 2001.
3. C. Stroud et al., "Built-In Self-Test of Logic Blocks in FPGAs (Finally, A Free Lunch: BIST Without Overhead!)," *14th IEEE VLSI Test Symposium*, pp. 387-392, April 1996.
4. C. Stroud, E. Lee, M. Abramovici, "BIST-Based Diagnostic of FPGA Logic Blocks," *Proc. of the International Test Conference*, pp. 539-547, Nov. 1997.
5. C. Stroud et al., "Built-In Self-Test of FPGA Interconnect," *Proc. of the International Test Conference*, pp. 404-411, Nov. 1998.
6. Doumar, T. Ohmameuda, H. Ito, "Design of an automatic testing for FPGAs," *IEEE European Test Workshop Compendium of Papers*, pp. 152-157, May 1999.
7. 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.
8. 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.
9. T. Inoue, S. Miyazaki, H. Fujiwara, "Universal Fault Diagnosis for Look-up Table FPGAs," *IEEE D&T of Computers*, Vol. 15, No. 1, pp. 39-44, January-March 1998.
10. M. Renovell et al., "RAM-Based FPGA's: A Test Approach for the Configurable Logic," *IEEE Int. Conference on Design, Automation and Test in Europe*, pp. 82-88, Feb. 1998.
11. M. Renovell, J. M. Portal, J. Figueras, Y. Zorian, "Testing the interconnect of RAM-based FPGAs," *IEEE D&T of Computers*, Vol. 15, No. 1, pp. 45-50, January-March 1998.
12. N. R. Shnidman et al., "On-Line Fault Detection for Bus-Based Field Programmable Gate Arrays," *IEEE Trans. on VLSI Systems*, Vol. 6, No. 4, pp. 656-666, December 1998.
13. M. Abramovici et al., "On-Line Testing and Diagnosis of FPGAs with Roving STARs," *Proc. 5th IEEE Int. On-Line Testing Workshop*, pp. 2-7, July 1999.
14. L. Burress, P. K. Lala, "On-Line Testable Logic Design for FPGA Implementation," *Proc. of the International Test Conference*, pp. 471-478, November 1997.
15. M. Renovell et al., "Test Generation Optimization for a FPGA Application-Oriented Test Procedure," *Proc. of the 15th Design of Circuits and Integrated Systems Conference*, pp. 330-336, November 2000.
16. M. G. Gericota, G. R. Alves, M. L. Silva, J. M. Ferreira, "Active Replication: Towards a Truly SRAM-based FPGA On-Line Concurrent Testing," *Proc. of the 8th IEEE On-Line Testing Workshop*, pp. 165-169, July 2002.
17. Pol. di Torino ITC'99 benchmarks, available at http://www.cad.polito.it/tools/itc99.html
18. M. Gericota, G. Alves, M. Silva, J. Ferreira, "Run-time Defragmentation for Dynamically Reconfigurable Hardware", in *New Algorithms, Architectures and Applications for Reconfigurable Computing*, pp. 117-129, Springer, 2005, ISBN 1-4020-3127-0.
19. O. Diessel, H. El Gindy, M. Middendorf, H. Schmeck, B. Schmidt, "Dynamic scheduling of tasks on partially reconfigurable FPGAs," *IEE Proc.-Computer Digital Technology*, Vol. 147, No. 3, pp. 181-188, May 2000.
20. M. Vasilko, "DYNASTY: A Temporal Floorplanning Based CAD Framework for Dynamically Reconfigurable Logic Systems," *Proc. 9th Intl. Workshop on Field-Programmable Logic and Applications*, pp.124-133, Aug.-Sep. 1999.
21. M. Teich, et al., "Compile-time optimization of dynamic hardware reconfigurations," *Proc. Intl. Conf. on Par. and Distr. Proc. Techniques and Applications*, pp. 1097-1103, 1999.
22. M. G. Gericota, G. R. Alves, J. M. Ferreira, "A Self-Healing Real-Time System Based on Run-Time Self-Reconfiguration," *10th IEEE International Conference on Emerging Technologies and Factory Automation*, Catania, Italy, September 2005.