Dynamically Rotate And Free for Test: The Path for FPGA Concurrent Test

Manuel G. Gericota, Gustavo R. Alves, José M. Ferreira

Abstract

Dynamically reconfigurable systems have benefited from a new class of FPGAs, recently introduced into the market, that enables partial and dynamic reconfiguration.

While enabling concurrent reconfiguration without disturbing system operation, this technology also raises a new test challenge: to assure a continuously fault free operation independently of the circuit present after the reconfiguration process. A new structural concurrent test method, recently proposed by the authors and based in the principle of replicating and freeing the resources to be tested, raised several questions, one of them being: What strategy to follow in the process of dynamically replicate and free those resources?

This paper presents a strategy on how to free the resources to be tested and the results of a series of simulation experiments made with the objective of finding the best methodology to achieve it.

1. Introduction

Reconfigurable logic devices, namely Field Programmable Gate Arrays (FPGAs) experienced a considerable expansion in the last few years. These components enable savings in board space, quicker turn-around time and an unsurpassed degree of flexibility, when compared to traditional off-the-shelf LSI/VLSI components with pre-defined functionality. These advantages have been reinforced and new possibilities added, with the advent of dynamic and partially reconfigurable SRAM-based FPGAs (e. g. the Virtex family from Xilinx), that enable the dynamic customisation of hardware functions to a particular system or application "on-the-fly" [1].

The need to reprogram the whole device, halting its operation (and possibly halting the operation of the circuit or system where they are inserted), is one main limitation associated with classic SRAM-based FPGAs. Additionally, the contents of all registers (state information) are lost when the component is reprogrammed. In recent FPGA generations, manufacturers addressed these issues by supporting partial device reconfiguration, which can take place concurrently with the system operation.

Dynamic reconfiguration goes beyond "in-system reprogramming", since it does not interrupt the operation of the device. However, the very high levels of integration and sub-micron technologies used in these FPGAs lead to a higher occurrence of defects [2], thus creating a critical need for fault tolerance features. High reliability levels can only be achieved by continuously testing all FPGA blocks throughout system lifetime, in search of emerging

* This work was supported by an FCT program under contract PCTI/1999/ESE/33842

defects that pop out in result of the reconfiguration process.

The structural concurrent test method proposed in [3] reuses the dynamic and partially reconfigurable features introduced by these devices and the well established Boundary-Scan (BS) test infrastructure [4] defined in the IEEE 1149.1 standard [5], thus presenting a very low test overhead at chip and board level.

This paper is organised as follows: recently proposed approaches to the test of SRAM-based FPGAs are first examined, followed by a general description of the structural concurrent test solution envisaged for the Configurable Logic Blocks (CLBs) of an FPGA. The next section presents some particular aspects of the proposed solution, namely the strategies to dynamically rotate the free CLB among the used CLBs, in order to individually test all of them, and some simulation results that validate the selected option. The last section presents some directions for further research.

2. Proposed FPGA test approaches

In recent publications, different methodologies and aspects concerning the test of SRAM-based FPGAs have been considered, most of them focusing on off-line test methodologies applied after manufacturing and employing several different Built-In Self-Test (BIST) strategies. More recently, on-line/functional test methods, and structural/application-oriented tests, have also been proposed, as referred below.

An FPGA test approach based on BIST techniques, exposed in [6, 7], exploits the reprogrammability of FPGAs in order to create the BIST logic, by configuring it only during off-line testing such that this approach is applicable to all levels of testing, including the system level. In this way, testability is achieved without any area overhead or performance penalty, since the BIST logic is replaced when the circuit is reconfigured for its normal system operation.

A slightly different BIST technique, which includes a modification of the original configuration memory, is proposed in [8]. The advantages of this method, compared to similar BIST techniques, are a reduction on test time and on the off-chip memory required, and the automation of the test process. The modification required at the internal hardware level of the FPGA is one main drawback, implying the non-universality of the solution.

An on-line method based on a self-testing area is proposed in [9]. The main idea is to have only a relatively small portion of the chip off-line being tested (instead of the whole chip as in previous proposals), while the rest of the device is on-line carrying out its normal operation. Roving the self-testing area across the FPGA eventually tests the entire chip. It is assumed that the reconfiguration process is controlled by a module external to the FPGA.

A different approach specific to look-up table -based FPGAs, featuring design for testability characteristics, was proposed in [10]. An algorithm, which maps optimised Boolean expressions into look-up tables, automatically incorporates testability features into the implemented design, allowing on-line detection of faults within an FPGA. This is accomplished by using a unique set of cells to implement a design. These cells operate on the premise of a two-rail checker, thus producing both the normal and complemented output when a cell is operating correctly, and two outputs of the same value in the presence of a fault. A fault generated in an intermediate cell is propagated to the final outputs, thus allowing on-line testability of an FPGA-based system. This method could be considered an error detection technique rather than a structural or functional test approach, since it is not targeted to test the resources of the FPGA, in itself, neither it take into consideration its logic structure (it is rather aimed to detect the presence of faults in current implemented applications). In a newly configured application, the same defect may not be detected using this method because the new application may not try to use the faulty resource. If the system is always being reconfigured, this method could result in an intermittent fault, depending on whether or not a logic function is placed into a faulty resource.

A new application-oriented method that generates a functional test for the configured circuit, while considering the logic structure of the FPGA where it is implemented, was proposed in [11]. This method is an off-line field oriented test to be used by an FPGA user in a given application, thus presenting the same drawbacks of the previously referred method.

Our concurrent test proposal re-uses some of the previous concepts, while adding other innovative aspects, namely the use of an atomic unit of test – the CLB – introducing a negligible overhead at the FPGA level. The reuse of the BS infrastructure to access and control all necessary operations in order to achieve the final goal also greatly reduces the overhead at the board level. Being application-independent, and oriented to test the logic FPGA structure, it guarantees the FPGA reliability (even after many reconfigurations), thus helping to ensure the correct operation of the implemented applications.

3. The DRAFT Technique

An FPGA comprises an array of independent CLBs, surrounded by a periphery of Input/Output Blocks (IOBs), which are interconnectable by configurable routing resources, as shown in figure 1.

In most cases, only a portion of the FPGA resources is required to implement a given functional specification. Even in those cases where independent hardware blocks share the same FPGA device, with the aim of reducing

board space, it is highly improbable that all of the FPGA resources are used.

It is therefore possible to consider a test strategy where temporarily unused CLBs are released for test (without compromising the operation of the FPGA), while the remaining CLBs are requested and in use by the current implementation. Unused CLBs that have been successfully tested become available as spare blocks, which may then be used to replace other resources found defective. This solution guarantees that the whole FPGA can be tested, without disturbing the system operation, provided that at least one unused CLB is available in the current implementation. Fault tolerance features will however require more than one unused CLB, since a pool of spare resources has to be continually available.

This test method uses a technique named Dynamically Rotate And Free for Test (DRAFT), as represented in figure 2. This technique scans the whole set of CLBs, replicating an atomic subset of the current implementation in a previously unused CLB, which has been successfully tested. The previously used CLB is then released for test, and passed on to the pool of spare resources in the event of a positive test outcome. This replicating process covers all the FPGA, dynamically rotating the free CLB among the used CLBs (without disturbing the system operation).

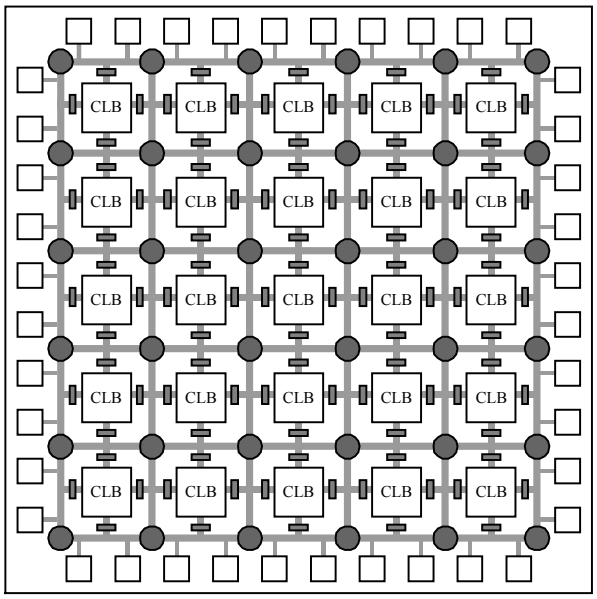

Figure 1. Schematic representation of an FPGA

This method is only feasible with devices that support dynamic partial reconfiguration, where the replication of the CLB (functional and state information) and the re-establishment of the interconnections are possible without disrupting the operation of the device. Our solution also requires that the partial dynamic reconfiguration is done through the 1149.1 Test Access Port (TAP) such as in the Xilinx Virtex family.

This novel structural concurrent test method has very low test overhead, since it reuses the BS test infrastructure for all reconfiguration operations, for test vector application to the CLB under test and response capturing.

4. Dynamic rotation strategies

This paper concentrates on the aspects related to the strategy adopted for the implementation of the dynamic rotation of the free CLB. Two goals were fixed initially: the rotation method should have a minimum impact (preferably none) in the system operation, as well as a reduced overhead in terms of reconfiguration cost. This cost depends on the number of frames needed to replicate and free each CLB, as 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. Note that the rotation process might imply a longer path, thus diminishing the maximum system operation frequency (in an FPGA the longest path delay determines the maximum frequency of operation). Thus, if the re-routing procedure originates a path delay higher than the previous maximum, the higher frequency of operation is reduced, leading to an undesirable impact in the system operation.

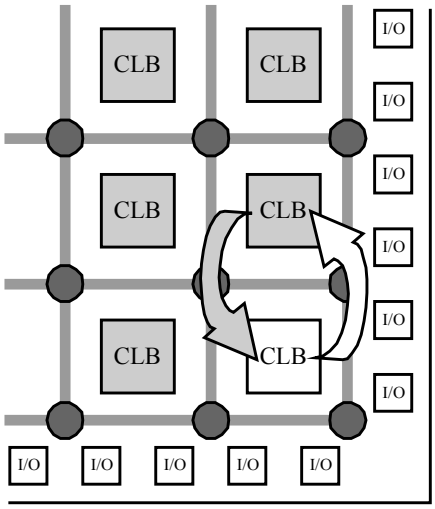

Figure 2. CLB replication and rotation of free resources

Three possibilities were considered for establishing the rule to rotate the free CLB among the entire CLB array: random, vertical and horizontal rotation.

The first strategy considered (random rotation) was rejected for several reasons. If the placement algorithm (in a strategy to reduce path delays) gathers in the same

area the logic needed to implement the components of a given application, it would not be logical to disperse the blocks by two main reasons. First, it would generate longer paths (and hence, an increase in path delays); secondly, it would put too much stress in the limited routing resources. Furthermore, the rotation strategy could not be totally random since it is necessary to assure the same degree of defect coverage for the whole CLB array within a predictable latency.

The second strategy (vertical rotation) illustrated in figure 3, would rotate the free CLB following a vertical shifting path that would cover all the CLBs in the array. In the Virtex family, the configuration memory can be visualised as a rectangular array of bits, which are grouped into vertical frames, one-bit wide, that extend from the top of the array to the bottom. Frames grouped together into larger units are called columns, which correspond to the configuration bits of a column of CLBs [12]. Apparently, the best strategy could be the replication of the resources to a nearby up or down CLB . Note that, in this case, only one CLB configuration column would be affected, and thus the number of reconfiguration frames needed for the vertical rotation would be less than the number needed for random or horizontal rotation.

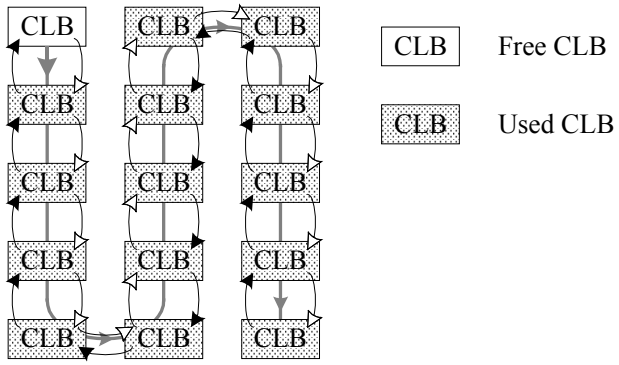

Figure 3. Dynamic rotation of the free CLB, using a vertical strategy

The third strategy corresponds to the horizontal rotation of the free CLB, as illustrated in figure 4. The free CLB would rotate following a horizontal shifting path that would also cover all the CLBs in the array. In the two previous cases the replication is only performed between neighbouring CLBs for the reasons referred before (scarcity of routing resources and increase in path delay). However, this strategy would probably result in a number of reconfiguration frames greater than for the vertical rotation, as it would affect two CLB configuration columns.

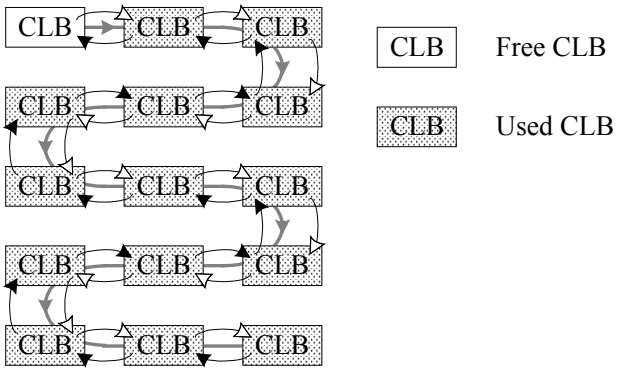

Figure 4. Dynamic rotation of the free CLB, using a horizontal strategy

5. Simulation results

A series of simulations were performed to find out the dynamic reconfiguration strategy with fewer costs. First, we used a microcontroller implementation described in [13], and tried to find out the influence of the two strategies in the number of reconfiguration frames needed to rotate the free CLB all along the CLBs occupied by the microcontroller implementation.

The simulations were performed on a microcontroller circuit, due to its internal complexity, characterised by a diversity of circuits covering a large range of possible fundamental components: decoders, shift registers, counters, and other arithmetic functions.

In the first approach we tried to find out what strategy, between the remaining two – vertical and horizontal rotation strategies – was the most advantageous in terms of the number of reconfiguration frames needed to perform each rotation. The results were not entirely satisfactory as the horizontal rotation solution had only, in terms of mean value, a cost which was 10% higher than the vertical rotation solution, a value lower than the expected difference. The dispersion of results was also very large, as shown in the graphic presented in figure 5. In face of the results, we concluded that the number of frames affected by swapping the configuration setup between CLBs was smaller than those affected by re-routing the interconnected nets. Shifting the free CLB vertically or horizontally doesn't result in a great difference in the number of configuration frames since the number of the routing configuration frames dominates over the number of CLB configuration frames.

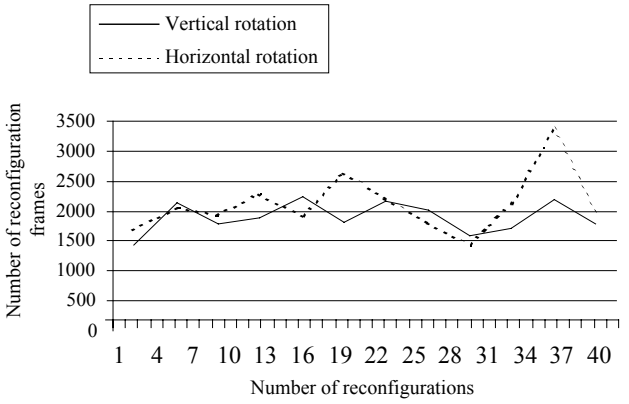

Figure 5. Cost of the rotation in terms of reconfiguration frames

Nevertheless, during the simulation process we noticed that the routing algorithm tended to spend a large effort when the rotation process affected some types of routing resources. In fact, Virtex FPGAs present several dedicated routing resources, one of them being a pair of dedicated nets per CLB that propagate carry signals vertically to adjacent CLBs [14]. When, in the rotation process, one dedicated carry path was broken due to the introduction of the free CLB in the middle, the propagation of this carry signal between the nearest adjacent CLBs (over and under the free CLB) went on through generic routing resources.

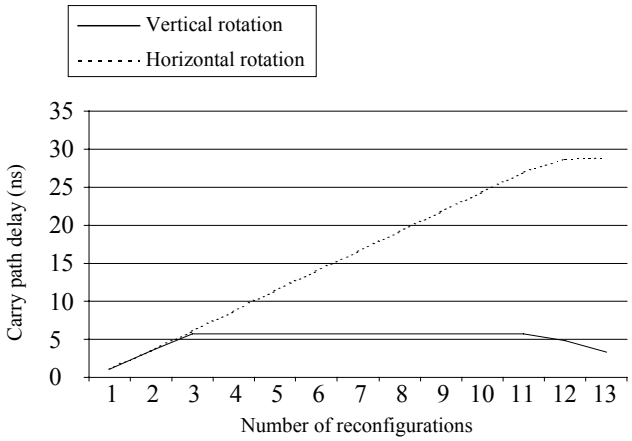

Figure 6. Changes in the carry path delay introduced by the rotation process

In order to understand the truly implications of this fact we tried the two strategies with a 24-bit counter, a component which operation relies mainly in the use of carry signals. We particularly noticed the changes in the carry path delay when the two strategies were applied. The effect of not using the dedicated carry nets on the delay path was evident, as shown in the graphic presented in figure 6. With the vertical rotation technique, the dedicated carry nets were kept, except between the two cells separated during the test by the free CLB. With the horizontal rotation strategy, all dedicated carry nets were gradually broken, thus requiring the use of generic resources. The consequence was a linear increase in the carry path delay that further resulted in a huge reduction on the maximum operating frequency of the counter, as shown in figure 7.

By applying the same simulation process to the microcontroller implementation we found out that its maximum operating frequency dropped when using the horizontal rotation strategy, and, surprisingly, raised when using the vertical strategy. This pointed out that the reconfiguration algorithm performed better than the placement algorithm, as shown in figure 8.

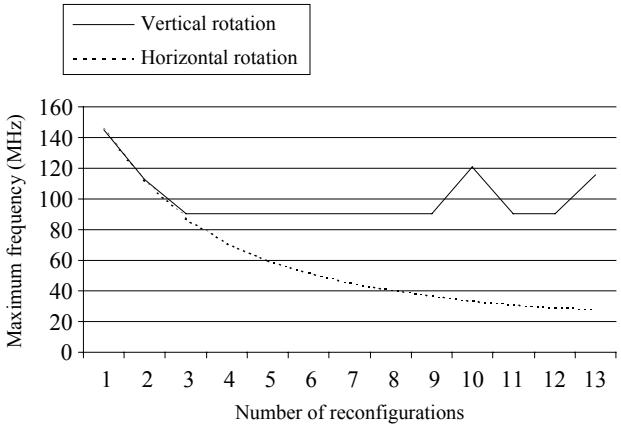

Figure 7. Changes in the maximum allowed frequency of the counter introduced by the rotation process

6. Conclusion

The solution proposed in this paper enables the implementation of a concurrent test method that reuses the standard BS infrastructure and the novel partial dynamic reconfiguration features of recent FPGA devices, in order to improve the reliability of reconfigurable hardware systems, with minimal test overhead and in a way that is completely transparent to the system operation.

Simulation sessions done with the objective of finding the best rotation strategy showed that the vertical rotation strategy does not affect the normal system operation and is preferable to the horizontal rotation strategy that tends to rise path delays and, consequently, imposes a decrease in the maximum operating frequency.

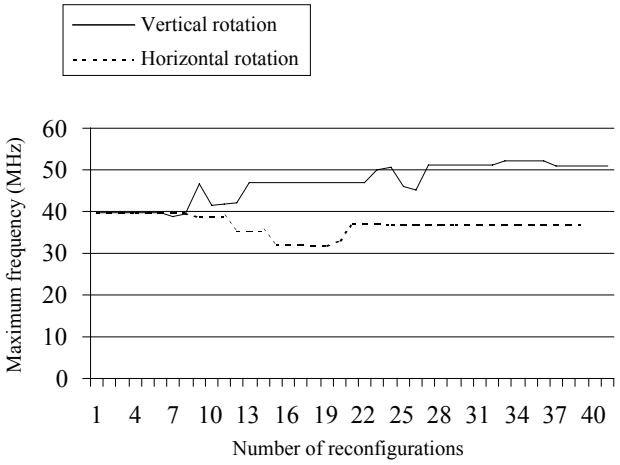

Figure 8. Changes in the maximum allowed frequency of the microcontroller introduced by the rotation process

Experiments are currently under way addressing other aspects of the proposed method, namely looking for an optimal solution (test application time, fault coverage), in terms of the trade off between the number of reconfiguration operations and the number of test vectors. Future research work is also planned to address other FPGA resources, namely interconnections and RAM--blocks, while still reusing the BS infrastructure.

7. References

[1] C. Maxfield, "Logic that mutates while-u-wait", *EDN Magazine*, N.º 23, November 1996.

[2] F. Hanchek, S. Dutt, "Methodologies for Tolerating Cell and Interconnect Faults in FPGAs", *IEEE Transactions on Computers*, Vol. 47, N. 1, pp. 15-33, January 1998.

[3] Gericota, M. G., Alves, G. R., Ferreira, J. M., "The RaT technique for concurrent test of dynamically reconfigurable hardware", *Proceedings of the XV Conference on Design of Circuits and Integrated Systems (DCIS'2000)*, November 2000.

[4] C. Hamilton, G. Gibson, S. Wijesuriya, C. Stroud, "Enhanced BIST-Based Diagnosis of FPGAs via Boundary Scan Access", *Proceedings of the 17th IEEE VLSI Test Symposium*, pp. 413-418, April 1999.

[5] *IEEE Standard Test Access Port and Boundary Scan Architecture* (IEEE Std 1149.1), IEEE Standards Board, May 1990.

[6] Stroud, C., Lee, E., Konala, S., Abramovici, M., "Using ILA Testing for BIST in FPGAs", *Proceedings of the International Test Conference (ITC96)*, pp. 68-75, October 1996.

[7] Stroud, E., Konala, S., Chen, P., Abramovici, M., "Built-In Self-Test of Logic Blocks in FPGAs (Finally, A Free Lunch: BIST Without Overhead!)", *Proceedings of the 14th IEEE VLSI Test Symposium (VTS96)*, pp. 387-392, April 1996.

[8] Doumar, A., Ohmameuda, T., Ito, H., "Design of an automatic testing for FPGAs", *Compendium of Papers IEEE European Test Workshop (ETW99)*, May 1999.

[9] M. Abramovici, C. Stroud, S. Wijesuriya, C. Hamilton, V. Verma, "On-Line Testing and Diagnosis of FPGAs with Roving STARs", *Proceedings of the 5th IEEE International On-Line Testing Workshop*, pp. 2-7, July 1999.

[10] Burress, A. L., Lala, P. K., "On-Line Testable Logic Design for FPGA Implementation", *Proceedings of the International Test Conference (ITC97)*, pp. 471-478, November 1997.

[11] Renovell, M., Portal, J. M., Faure, P., Figueras, J., Zorian, Y., "Analyzing the Test Generation Problem for an Application-Oriented Test of FPGAs", *Proceedings of the IEEE European Test Workshop (ETW'2000)*, pp. 75-80, March 2000.

[12] "Virtex Series Configuration Architecture User Guide", Xilinx Application Note 151, version 1.3, February 2000 (available at http://www.xilinx.com).

[13] "8-bit microcontroller for Virtex devices", Xilinx Application Note 213, version 1.1, October 2000 (available at http://www.xilinx.com).

[14] "Virtex 2.5V Field Programmable Gate Arrays", Xilinx Final Product Specification, v. 2.2, May 2000 (available at http://www.xilinx.com).