Reliability and Availability in Reconfigurable Computing: A Basis for a Common Solution

Manuel G. Gericota, *Member, IEEE*, Gustavo R. Alves, Miguel L. Silva, and José M. Ferreira

***Abstract*—Dynamically reconfigurable SRAM-based field-programmable gate arrays (FPGAs) enable the implementation of reconfigurable computing systems where several applications may be run simultaneously, sharing the available resources according to their own immediate functional requirements. To exclude malfunctioning due to faulty elements, the reliability of all FPGA resources must be guaranteed. Since resource allocation takes place asynchronously, an online structural test scheme is the only way of ensuring reliable system operation. On the other hand, this test scheme should not disturb the operation of the circuit, otherwise availability would be compromised. System performance is also influenced by the efficiency of the management strategies that must be able to dynamically allocate enough resources when requested by each application. As those resources are allocated and later released, many small free resource blocks are created, which are left unused due to performance and routing restrictions. To avoid wasting logic resources, the FPGA logic space must be defragmented regularly. This paper presents a non-intrusive active replication procedure that supports the proposed test methodology and the implementation of defragmentation strategies, assuring both the availability of resources and their perfect working condition, without disturbing system operation.**

***Index Terms*—Active replication, availability, field-programmable gate array (FPGA), online structural testing, reliability.**

I. Introduction

RECONFIGURABLE logic devices, namely field-programmable gate arrays (FPGAs), experienced a considerable expansion in the last few years due in part to an increase in their size and complexity, with advantages in terms of board space and flexibility. The availability of SRAM-based FPGAs supporting fast runtime partial reconfiguration (e.g., the Virtex family from Xilinx used to validate this work, the only family of FPGAs that supports dynamic reconfiguration so far) considerably reinforced these advantages, wide-spreading their usage as a base for reconfigurable computing platforms.

Manuscript received May 10, 2006; revised October 10, 2007. Current version published October 22, 2008. This work was supported by an FCT Program under Contract POSC/EEA-ESE/55680/2004.

M. G. Gericota and G. R. Alves are with the Department of Electrical Engineering, Polytechnic Institute of Porto, 4200-072 Porto, Portugal (e-mail: mgg@isep.ipp.pt; gca@isep.ipp.pt).

M. L. Silva is with the Department of Electrical and Computer Engineering, Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal (e-mail: mlms@fe.up.pt).

J. M. Ferreira is with the Department of Electrical and Computer Engineering, Faculty of Engineering, University of Porto, 4200-465 Porto, Portugal, and also with the Buskerud College of Engineering, 3603 Kongsberg, Norway (e-mail: jmf@fe.up.pt).

Digital Object Identifier 10.1109/TVLSI.2008.2001141

Dynamically reconfigurable FPGAs enable the implementation of virtual hardware as defined in [1] in the beginning of the 1990's, by using temporal partitioning to implement those applications whose area requirements exceed the available logic space (as if there were unlimited hardware resources). This approach is viable because each application comprises a set of functions, predominantly executed sequentially or with a low degree of parallelism, making simultaneous availability hardly ever required. The static implementation of an application is separated in two or more independent hardware contexts, which may be swapped during runtime [2]. Extensive work was done to improve the multi-context handling capability of these devices, by storing several configurations and enabling quick context switching [3], [4]. The main goal was to improve the execution time by minimizing external memory transfers, assuming that some amount of on-chip data storage was available in the reconfigurable architecture. However, this solution was only feasible if the functions implemented on hardware were mutually exclusive on the temporal domain, e.g., context-switching between coding/decoding schemes in communication, video or audio systems; otherwise, the length of the reconfiguration intervals would lead to unacceptable delays in most applications.

The reduction of manufacturing scales contributed significantly to eliminate these restrictions, by enabling higher levels of integration and higher frequencies of operation. The increasing amount of logic available in FPGAs and the smaller reconfiguration times, partly owing to the possibility of partial reconfiguration [5], extended the concept of virtual hardware to the implementation of multiple applications sharing the same logic resources in the spatial and temporal domains. However, if the functions required by the different applications cannot be scheduled in advance, all resource allocation decisions will have to be made at runtime, preventing a long term resource allocation strategy. In this case, fragmentation of the logic space is inevitable, leading to poor resource utilization. Since the type and amount of resources required by different functions varies greatly, as resources are allocated to functions and later released, many "islands" of free resources are created. These areas tend to become so small that they are left unused due to performance and routing restrictions. To avoid wasting resources and degrading system availability, the defragmentation of the FPGA logic space must be performed systematically.

Smaller submicrometer scales also have disadvantages, such as the higher electronic current densities in metal traces—which increase the threat of electromigration—and the lower threshold voltages. As scale goes down the number of defects related to small manufacturing imperfections that are not detected by production testing goes up. These defects are especially prone to

electromigration phenomena and lead to permanent faults after long operation periods [6]. On the other hand, the exponential growth in the number of configuration memory cells, and their lower threshold voltages, make these components more susceptible to gamma particle radiation and to the appearance of transient faults, such as single event upsets (SEU) and multi-bit upsets (MBU) [7]–[9]. These faults do not physically damage the chip, but their effects are permanent, since the functionality of the circuits mapped into the device is modified [10], [11]. Yet, and since the cause of the failure is actually transient, online reconfiguration is sufficient to restore the original functionality.

In the case of permanent faults, and after the faulty elements are located—either configurable logic blocks (CLBs) or routing resources—they must be excluded and replaced by previously unused fault-free resources. A non-intrusive online concurrent test strategy, performing a structural test of all FPGA resources is required to detect and diagnose any emerging permanent fault. This solution involves the periodic release of all resources from their active functions for the system to be able to perform their structural test.

Previous considerations about the advantages and disadvantages of new FPGA features show that to increase the availability and reliability of reconfigurable computing systems, transparent online FPGA logic space defragmentation and transparent online test operations must be run simultaneously [12], [13]. Both procedures require resource reallocation strategies that do not interfere with any currently running functions. This article presents a non-intrusive procedure for concurrent replication of active logic blocks and resource interconnections (i.e., logic resources and interconnections that are currently being used to implement running functions from one or more applications). This procedure is able to support the implementation of the proposed structural test methodology and also to serve as a basis for the implementation of defragmentation strategies.

This paper is organized as follows. Section II presents a survey of FPGA test strategies proposed in the literature. A detailed account of the proposed procedure for the active replication of logic resources is presented on Section III, while Section IV describes the online structural concurrent test methodology. A brief introduction to the defragmentation issue and an overview of how the implementation of defragmentation strategies may benefit from the proposed active replication procedure are provided in Section V. Section VI presents the software tool developed to support the active replication procedure and the test methodology. Finally, conclusions are drawn in Section VII.

II. Testing Dynamically Reconfigurable FPGAs

To achieve higher reliability in reconfigurable computing systems, the structure of all FPGA resources has to be continuously tested and error correction/fault tolerance has to be introduced. These requirements will ensure that any function will perform correctly independently of its type. For SRAM-based FPGAs, they translate into the following features:

1) to be non-intrusive;
2) to be able to detect any permanent structural fault emerging during system lifetime;
3) to be able to correct transient faults affecting function functionality.

Several offline and online strategies have been proposed to test and diagnose FPGA faults. An offline built-in self-test (BIST) technique that uses reprogrammability to set up the BIST logic is presented in [14]–[16]. Some of the logic blocks are configured as test pattern generators or response analyzers, while testing the other blocks, and vice versa. Since the test sequences are a function of the FPGA architecture and independent of its functionality, this approach is applicable at all levels (wafer, packaged device, board, and system). This technique requires a fixed number of reconfiguration sessions and presents no area overhead or performance penalty, since the BIST logic is eliminated when the circuit is reconfigured for normal operation.

A slightly different BIST technique, involving a structural modification of the original configuration memory, is proposed in [17]. This technique enables the automation of the test process while reducing test time and off-chip memory. However, the modification required to the FPGA hardware is a major disadvantage, implying the non-universality of the solution.

An offline test methodology based on a non-BIST approach, targeting the FPGA CLBs, is presented in [18] and [19]. After setting up a specific test configuration, 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 must be programmed and specific sets of test vectors applied in each case. Based on the same principles, a fault diagnosis method is presented in [20]. Extensive work on the structural testing of FPGA lookup tables (LUT) and interconnections is also presented in [21] and [22].

The previous approaches require the device to be offline, increasing fault-detection latency, and as such are not admissible in highly fault-sensitive, mission-critical applications. In order to overcome these limitations, online testing and diagnosis methods based on a scanning strategy were presented in [6] and [12]. The idea underlying these methods is to have a relatively small portion of the chip being tested offline (instead of the whole chip, as considered in previous proposals), while the rest continues its normal online operation. Testing is accomplished by sweeping the test functions across the entire FPGA. If the functionality of a small number of FPGA elements can be relocated on another portion of the device, then those elements may be taken offline and tested in a completely transparent way. This fault scanning procedure moves on to copy and test another set of elements, systematically testing the whole FPGA. However, the replication procedure of the first approach [6] requires a modified cell structured, while the second approach [12] halts the whole system to relocate an entire CLB column. Since reconfiguration is performed through the boundary scan (BS) infrastructure (IEEE 1149.1 Standard) [23], reconfiguration time is long, and it seems likely that repeatedly halting the system will severely disturb its operation.

The design for test features proposed in [24] are essentially concerned with fault detection, instead of carrying out any structural or functional test functions (the FPGA logic structure is not taken into account). Their main goal consists of detecting the presence of faults in the current application, and therefore, a physical defect may escape detection if that particular application is not using the damaged resource.

A new application-oriented method that generates a functional test for the configured circuit, taking into account the logic structure of the FPGA where it is implemented, was proposed in [25]. However, this method corresponds to an offline field-oriented test to be used with a given application, thus presenting the same drawbacks of the previous method.

The online test approach proposed in this article reuses some of the previous ideas, but eliminates their disadvantages by using a novel concept herein referred as *active replication*, which enables the functionality of a given set of resources to be relocated without halting the system. This approach is feasible even when the resources are active, i.e., when they are being used by a function that is currently running [26], [27].

Conceptually, an FPGA may be visualized as an array of uncommitted CLBs, surrounded by a periphery of IOBs, which are interconnectable by configurable routing resources. A set of memory cells that lies beneath controls the configuration of the whole structure.

Complete (100%) usage of the FPGA resources is hardly ever achieved, even when independent hardware blocks, from different applications, dynamically share the same device. The dynamic and partially reconfigurable features that are offered by some FPGAs make it possible to test all free CLBs and interconnection resources, without disturbing system operation.

After being tested, defect-free CLBs and interconnection resources remain available as spare blocks, ready to replace others that are found defective. The CLBs and routing resources currently being used are released for test following a dynamic rotation mechanism, after having their current functionality relocated into other areas already tested. That dynamic rotation mechanism ensures that all FPGA resources are released and tested within a given latency.

The active replication of the FPGA resources is therefore at the core of this proposed non-intrusive online structural test approach, which can be carried out concurrently with system operation. Since all FPGA resources are released and tested using the BS test infrastructure, there is no overhead at board level. Being application-independent, and oriented to test the FPGA structure, the proposed strategy guarantees FPGA reliability after many reconfigurations, and helps to ensure correct operation throughout the system lifetime.

III. Concurrent Replication of Active Logic Blocks

The replication of CLBs and interconnections is required to release any active resources for testing. However, it is not trivial to do it non-intrusively due to two major issues: 1) configuration memory organization and 2) 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 interconnecting information. The organization of the entire configuration memory into frames enables the online concurrent partial reconfiguration of the FPGA.

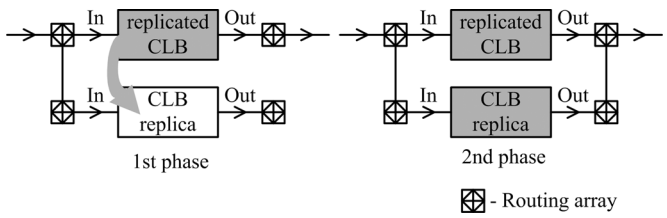

Fig. 1. Two-phase CLB replication process.

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 reestablished from its replica. It is also important to ensure that the functionality of the CLB replica is perfectly stable before its outputs are connected to the system, to avoid output glitches.

A set of experiments performed with Virtex FPGAs from Xilinx demonstrated that the replication process has to be divided into two phases, as illustrated in Fig. 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 infrastructure), the reconfiguration time is relatively long when compared to the system speed. 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. Therefore, the remaining resources covered during this process by the rewritten configuration frames are not affected, even if in an active state.

The correct transference of state information is another major requirement for the success of the replication process. If the current CLB function is purely combinational, a simple read-modify-write procedure will suffice to accomplish a successful replication. 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. In the Virtex FPGA family, each CLB slice comprises two storage elements, which can be individually configured as a latch or flip-flop (FF). Although a read back operation of the configuration memory may be performed to read the value of a storage element, it is not possible to perform a direct write operation. In addition, when dealing with active CLBs during a replication procedure, if state information changes between read and write operations, a coherency problem will occur. For this reason, no time gap is allowed between the two operations.

The solution to this problem depends on the type of implementation. The following three cases are considered:

1) synchronous free-running clock circuits;
2) synchronous gated-clock circuits, and;

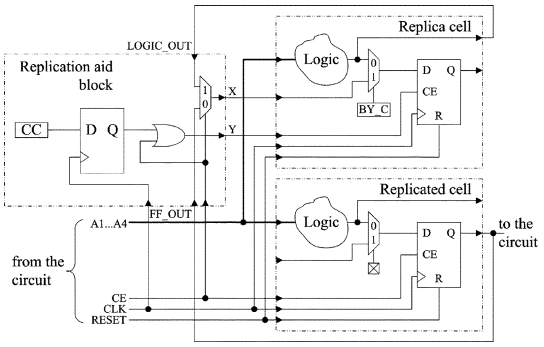

Fig. 2. Implementation of the synchronous gated-clock FF replication scheme.

3) asynchronous circuits.

When dealing with synchronous free-running clock circuits, the two-phase replication process that was previously described solves the state transfer problem. Between the first and the second phase, the CLB replica has the same inputs as the replicated CLB, and all its storage elements acquire the state information, even if the system clock frequency is an order of magnitude lower than the clock frequency (of the BS infrastructure) used for reconfiguration purposes. Several experiments were carried out and showed the effectiveness of this method to replicate active CLBs. No loss of state information and no output glitches were observed. Notice that this procedure is valid even when dealing with asynchronous circuits. If the longest interval between consecutive update operations of asynchronous latches is lower than the interval between the first and the second phases, the replicated latch always acquires the correct state information.

Despite the effectiveness of this solution, its usefulness is very restricted. A broad range of applications use synchronous gated-clock circuits, where input acquisition 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 FFs 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 FFs may differ from the one captured by the replicated FFs, resulting in a coherency problem. Furthermore, the FFs could be updated during the replication process, since this procedure is asynchronous in relation to system operation.

A replication aid block is used to solve this problem. This block manages the transfer of state information from the replicated FFs to their replicas. State information may also be updated by the circuit at any moment, without losing information or delaying the replication process. The replication scheme is represented in Fig. 2 for a single CLB logic cell (for this purpose each CLB logic cell in the Virtex FPGA family can be considered individually). Fig. 3 represents the flow diagram of the replication process.

One input of the 2:1 multiplexer in the replication aid block is connected to one temporary transfer path from the output of the replicated FF (FF_OUT). The other one is connected to the output of the combinational logic block in the replica cell (LOGIC_OUT), which is normally applied to the input of the FF. If the clock enable (CE) signal—controlling the multiplexer—is not active, the output of the replicated FF (FF_OUT) is applied to the input of the replica FF. A clock enable signal, coming from the replication aid block (capture control signal—CC), forces the replica FF to store the transferred value. 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. This FF is, therefore, updated simultaneously with the replicated FF, and captures the same value, guaranteeing state coherency. Neither simulations, nor the ensuing practical experiments, have shown any loss of information.

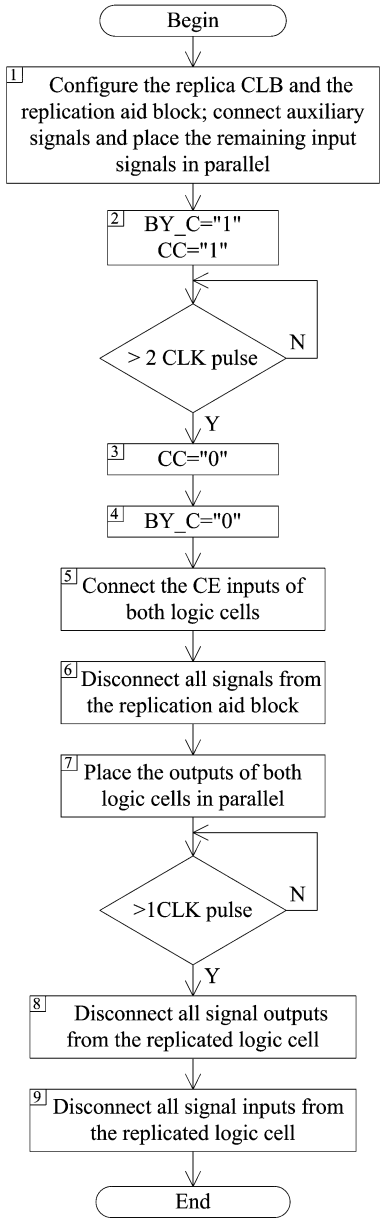

Fig. 3. Replication process flow.

The control signals CC and BY_C are driven by configuration memory bits. 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. Fig. 4

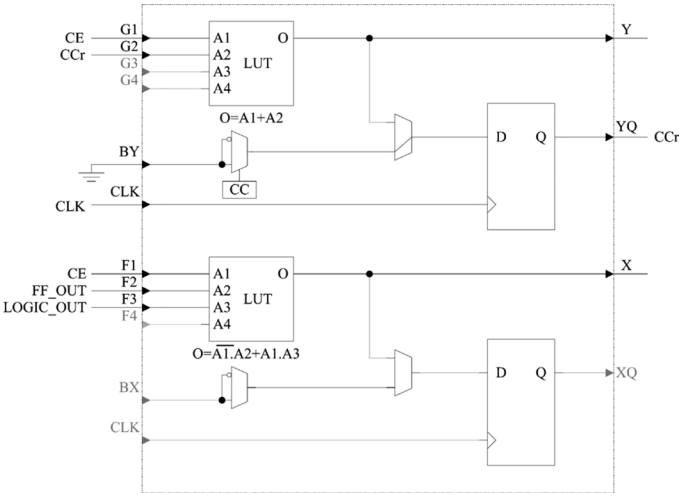

Fig. 4. Simplified representation of the replication aid block implemented on a CLB slice.

shows a schematic implementation of the replication aid block in a CLB slice. To enable all signals to be controlled through the configuration memory, the CC net includes the FF shown in Figs. 2 and 4. However, it is there simply as a consequence of the structure of the CLB slice, and does not play any role in this process.

After the transference of state information BY_C is driven low, disconnecting the replica FF from the replication aid block. The state transfer ends and the replica FF may now be directly updated by the circuit. The CE signal of both CLBs is placed in parallel, all the signals to/from the replication aid block are disconnected, and the outputs are also placed in parallel. After at least one clock cycle, the replicated block is disconnected, and the resources used in its implementation are released. Each of these steps (corresponding to a rectangle in the flow diagram shown in Fig. 3) requires a new reconfiguration file. A total of nine files are therefore needed to complete the replication process, instead of four, as would be necessary when dealing with synchronous free-running clock circuits. However, in most cases, their size is much smaller (to change the value of CC and BY_C only one configuration frame is needed). Table I details the average sizes of the partial reconfiguration files and their respective reconfiguration times, when using a 20-MHz test clock [the TestClock (TCK) signal of the BS infrastructure] [23]. Replication of synchronous free-running clock circuits takes roughly 18 ms, as steps 2–6 are not necessary.

Practical experiments performed using a Virtex device to implement the ITC'99 Benchmark Circuits from the Politecnico di Torino [28], demonstrated the effectiveness of the proposed approach. These circuits are purely synchronous with only one single-phase clock. 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 a time. Still, the slowest "clock" period must be shorter than the duration of the replication process, thus enabling the FFs to be updated meanwhile.

The proposed method is also effective when dealing with asynchronous circuits, where storage elements are configured

TABLE I
COST OF EACH PARTIAL RECONFIGURATION FILE DURING REPLICATION

Steps	No. of bytes	Time (ms)
Internal logic functionality is copied and input signals are placed in parallel	11 289	9.705
BY_C=1∧CC=1	441	0.379
CC=0	277	0.238
BY_C=0	277	0.238
Clock enable inputs of both CLBs are connected	2 145	1.844
All the auxiliary relocation circuit signals are disconnected	2 217	1.906
CLB outputs are placed in parallel	4 129	3.550
Original CLB outputs are disconnected	1 333	1.146
Original CLB inputs are disconnected	3 986	3.438
Total	**26 094**	**22.444**

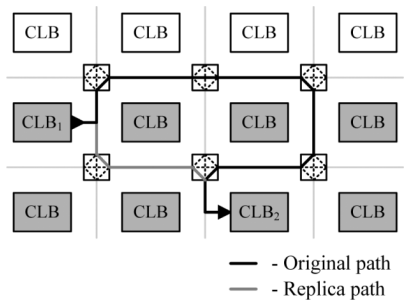

Fig. 5. Relocation of routing resources.

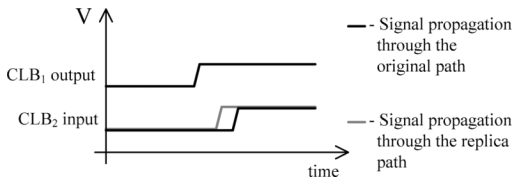

Fig. 6. Propagation delay during the relocation of routing resources.

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

The replication of routing resources does not pose any special problems, since the same two-phase replication procedure is also effective to relocate local and global interconnections. The interconnections involved are first duplicated in order to establish an alternative path, and then disconnected, becoming available to be reused, as illustrated in Fig. 5.

A last remark must be made about the replication of routing resources. The different paths used while paralleling the original and replica interconnections will likely have different propagation delays. This means that if the logic level at the output of the source CLB changes, there will be an interval of fuzziness at the input of the destination CLB, as shown in Fig. 6. However, the impedance of the routing switches will limit the current flow in

the interconnection, and hence this behavior does not damage the FPGA. Nevertheless, and for transient analysis, the propagation delay associated to parallel interconnections shall be the longer of the two paths [29].

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 [30]. 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 all the remaining data in the slice is constant, or else it must also be changed externally through partial reconfiguration. Even if not being replicated, LUT/RAMs should not lie in any column that could be affected by the replication process.

According to the overall test (and/or defragmentation) strategy, this method could be used to replicate more than one CLB simultaneously, improving scalability aspects. Considering that the smallest configuration unity is a frame, and that frames span the FPGA from top to bottom, it takes exactly the same time to replicate one CLB or the whole CLB column (the number of reconfiguration frames involved is the same in both situations). Scalability is therefore not an issue. The time required increases proportionally to the number of CLB columns in the FPGA and is independent of the number of CLB rows.

IV. Online Structural Concurrent Test

Our online structural concurrent test method is divided in the following three parts:

1) the replication procedure;
2) the test strategy;
3) the dynamic rotation mechanism.

The replication procedure has already been presented. A detailed presentation of the proposed test strategy and of the dynamic rotation mechanism used to release resources for test will follow.

A. Fault Detection and Error Recovery

The replication procedure used with synchronous free-running clock circuits did not perform a true state transfer operation, but rather an acquisition of the values present at the inputs of the replica CLB FFs. For this reason, the acquired state information is correct, despite any permanent or transient fault that may affect the content of 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 behavior. On the other hand, when replicating synchronous gated-clock circuits (or asynchronous circuits), a true state transfer operation is performed. In this case, the replica CLB FFs (or latches) will acquire exactly the same value held by the replicated FFs (or latches). Erroneous state information may therefore be propagated to the replica CLB, and will survive until an update occurs. A permanent fault in the replicated CLB will be detected during the subsequent test phase and 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 also recover from errors caused by transient faults in the on-chip configuration memory. Typical examples of such errors are SEUs, which modify the logic function originally implemented in the FPGA. Until now, they used to be a major concern only for space applications. Yet for designs manufactured at advanced technology nodes—such as 90, 65 nm, and downward—system-level soft errors become an issue also at ground level. They are now much more frequent than in previous generations [31]. Since Virtex FPGAs enable read back operations, a completely automatic read-modify-write procedure may 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.

B. Interconnection Resources and I/O Blocks

Successful structural testing of the CLB replica ensures its good functionality, but the replicated CLB may be faulty. When the inputs and outputs of both CLBs are placed in parallel, nodes with different voltage levels may be interconnected. 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 proven 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 propagates through a routing buffer, or at the input of the next CLB or IOB. No logic value instability was observed in our experiments [26].

In the FPGA, signals are routed using the global routing resources, which are located in horizontal and vertical routing channels between each routing array. The routing resources may be unidirectional or bidirectional. Besides a pair of dedicated paths providing high-speed connections between vertically adjacent CLBs (to propagate carry signals), few routing resources are available to establish direct interconnections with other CLBs. As such, the majority of interconnections required by the replication process can only be done through global routing resources.

To place the inputs in parallel, the interconnection segments to be used between routing arrays may be unidirectional (from the replicated CLB inputs towards the CLB replica inputs), or bidirectional. Concerning the outputs, interconnection segments between routing arrays may also be unidirectional (from the CLB replica outputs towards the replicated CLB output), or bidirectional, as illustrated in Fig. 7. Since signals do not propagate backwards, if propagation direction is not taken into account, no signals would exist at the inputs of the CLB replica, and the outputs of both CLBs would not be placed in parallel. As a result, when the outputs of the replicated CLB were disconnected, no signals would be propagated to the rest of the circuit.

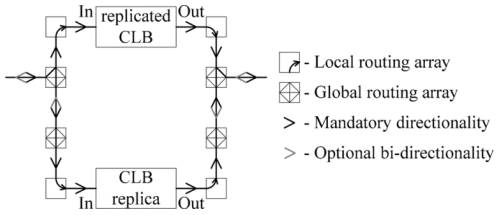

Fig. 7. Replication CLB interconnection.

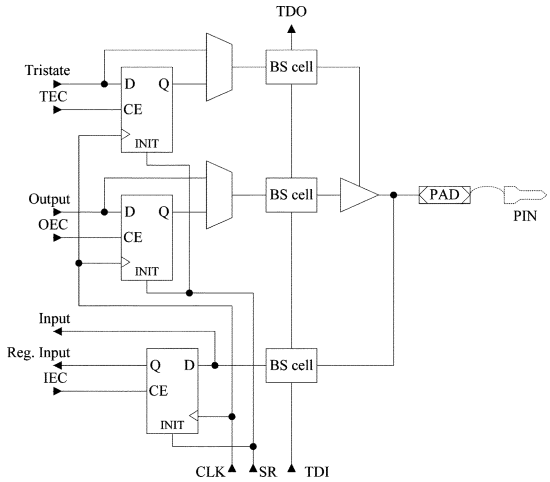

Fig. 8. Internal architecture of an IOB and associated BS cells.

Local routing, at the inputs (and outputs) of the CLB, is unidirectional and therefore 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, because no fault at any of the replicated CLB inputs may propagate backward.

This is also true when replicating active interconnections, with faults in the replicated net being automatically corrected when the replication takes place. Depending on the location of the fault in the replicated interconnection, it may be corrected while the path is duplicated, or only after it is disconnected.

Any FPGA pin could be used as an input, an output, a tristate output, or a bidirectional pin. The output and tristate signals may or may not be registered. The IOB circuitry provides an FF for each of these signals and two multiplexers, controlled through the configuration memory. The input signal is available to the internal logic both in registered and nonregistered form. A generic implementation of an IOB is illustrated in Fig. 8.

In spite of the configuration of each IOB or of its use (or not) to implement a system function, the number of BS cells of the BS register remains constant. All IOBs are considered as independent tristate bidirectional pins, placed in a single BS chain. For this reason, BS cells are provided on the input, output, and tristate signal paths, as required by the IEEE 1149.1 Standard [23]. Notice that, even when a bidirectional pin is used only as an input, its tristate and output BS cells are still part of the BS register, as well as the three BS cells of unused bidirectional pins.

All IOBs have a pad, as seen in Fig. 8, but not all of them have an associated output pin. IOBs without a bond wire connecting the pad to a pin on the package are called unbonded IOBs. These IOBs may be used on register intensive applications or as tristate buffers in internal bus implementations, with the bus signals being returned to the internal logic through the input path. Usually, design tools offer an option that enables the user to pack registers into IOBs. Despite not being true input/outputs, these IOBs have BS cells and, therefore, are part of the BS register.

Test vector application to the IOBs and response capturing should take account the following factors:

1) BS register enables controllability of the input signal path and observability of the output and tristate signal paths;
2) observability of the input signal paths and controllability of the output and tristate signal paths are not possible through the BS register;
3) observability and controllability of the control and clock signals are not possible through the BS register;
4) not all IOBs have an attached pin; therefore, external access to improve the controllability/observability of the IOB can not be assumed; however, since they all have BS cells, this limitation is not a problem.

These remarks lead to the conclusion that a feasible and reliable online test of the IOBs is not possible. The observability and controllability of all the paths in the IOB implies the direct access through the external pin (if it exists), or the execution of intrusive operations through the BS register. An offline test method for the IOB structure and its interconnections at board level, which presents no area overhead or performance penalty (since the logic functionality required to implement it is eliminated when the circuit is reconfigured for its normal operation), is presented in [32].

C. Test Configurations

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, FFs) is not known, a hybrid fault model was considered [18] (see also [21] and [22] for an extensive study concerning FPGA fault models). To test the SRAM elements of the LUT, each bit is set to both "0" and "1". By programming the LUTs to implement XOR and XNOR functions—which requires at least two test phases—it is easy to propagate any activated faults to a primary CLB output. Due to the XOR/XNOR functions, all LUT input stuck-at faults, together with their respective addressing faults, are also detected. For test purposes, Virtex CLB multiplexers have to be divided in two types: conventional and programmable multiplexers (those where selection lines are controlled through configuration memory bits). Since the existing maximum number of selection lines is two (in both cases), at least four test configurations are needed to exhaustively test each programmable multiplexer.

The CLB structure presents a chain of three configurable primitives, which requires at least six test configurations to completely test its combinational part. Notice that the test of primitives in a chain could not take place simultaneously, because the controllability and observability of a primitive under test depends on the configuration of its immediate neighbours

TABLE II
NUMBER OF TEST VECTORS PER TEST PHASE

1st test phase	16 test vectors
2nd test phase	16 test vectors
3rd test phase	2 test vectors
4th test phase	2 test vectors
5th test phase	2 test vectors
6th test phase	2 test vectors

TABLE III
COST OF EACH PARTIAL RECONFIGURATION

Configurations	No. of bytes	Time (ms)
1st	18 392	15.813
2nd	3 115	2.678
3rd	623	0.536
4th	634	0.545
5th	613	0.527
6th	512	0.440
Total	**23 889**	**20.539**

in the propagation path, except in the case of primitives with primary inputs and/or outputs. All FFs are tested during these six phases for data input and hold, clock enable, initialize and reverse, and stuck-at faults. Since reconfiguration is slower than test vector application, the small number of test phases is a good measure of our reduced test time. Notice also that test reconfiguration time is not constant through all six phases. In the first test phase the initial test configuration has to be set up. In the five subsequent test phases, only a few configuration bits related to the LUT function, to the programmable multiplexers and to the FFs configuration, are changed. Therefore, test reconfiguration time is smaller. Table II details the content of each CLB structural test session. The average values for the partial reconfiguration file sizes and reconfiguration times (using a 20-MHz TCK) are shown in Table III.

D. Test Application

The BS infrastructure is also reused to apply the test vectors and to capture the test responses, with the outputs of the CLB(s) under test being routed to unused BS register cells associated to the IOBs. However, the application of test vectors by means of the BS register would be intrusive, so an alternative User Test Register is needed (the Virtex family enables the definition of two user registers controlled through the BS infrastructure). The User Test Register created for this purpose comprises 13 cells, corresponding to the maximum number of CLB inputs in the Virtex family, and is fully compliant with the IEEE 1149.1 Standard [23]. The schematic representation of a User Test Register cell is illustrated in Fig. 9.

The seven CLBs occupied by this register and the two CLBs occupied by the replication aid block, associated to the CLB needed to perform the replication, are the only FPGA hardware overhead that is implied by the proposed test methodology. In total, it accounts for less than 1% of the CLB resources of a Xilinx Virtex XCV200 device (array size $= 28 \times 42$ CLBs), one of the FPGAs used to validate this work. Fig. 10 illustrates the

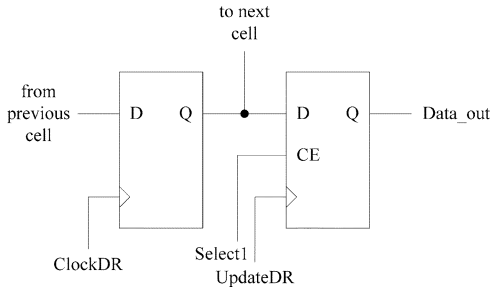

Fig. 9. User Test Register cell.

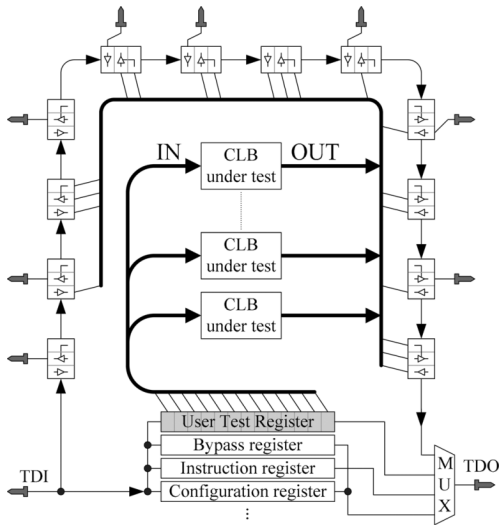

Fig. 10. Test of CLBs through the BS infrastructure.

TABLE IV
SHIFTING TIME FOR TEST VECTOR APPLICATION

No. of vectors	Length (bits)	Total (bits)	Shifting time (ms)
40	13	520	0.066

test infrastructure setup that is required to implement this procedure. Notice that more than one CLB may be under test at the same time, provided that enough routing resources and unused BS register cells are available. Since the same set of test vectors are applied simultaneously to all CLBs under test, the length of the User Test Register (13 bits) is fixed. Therefore, scalability of the test procedure is also possible, although dependent on the usage of the FPGA resources.

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 same principles apply to Virtex-II CLBs. Experimental results, obtained using a Virtex XCV200 with a TCK of 20 MHz, are shown in Table IV—the shifting time for each test vector application—and in Table V—the shifting time for the test vector responses from a CLB under test.

The test of global interconnections is achieved using the same principles, with the CLB under test being replaced by a set of

TABLE V
SHIFTING TIME FOR TEST VECTOR RESPONSE

No. of cells of the BS register in a XCV200	No. of vectors	Shifting time (ms)
1022	40	4.088

wires under test, limited only by the length of the User Test Register. Each wire from the set does not need to be a single wire, as single wires can be interconnected forming longer wires for test purposes [22]. This is not only useful, since it improves the scalability of the procedure, but also desirable, as it may improve the test of the global interconnection array blocks. Notice that local routing (inside the CLB and in its boundary) and local routing array blocks are tested over time, simultaneously with the CLB under test. The same happens, in an implicit way, with two other types of interconnections: those used to route test application vectors from the User Test Register to the CLB under test; and those used to route test responses from that CLB to the Boundary Scan Register. However, interconnect faults will not be recognised as such, being detected instead as CLB faults.

E. Rotation and Release for Test Strategy

A dynamic rotation mechanism assures that all CLBs are released for test. This mechanism should have a minimum influence (preferably none) in the system operation, as well as a reduced overhead in terms of reconfiguration cost. As seen before, this cost depends on the number of reconfiguration frames needed to replicate and release each CLB. The impact of this process in the overall system operation is mainly related to the delays imposed by rerouted paths, which may become longer due to the rotation process, thus degrading the maximum frequency of operation [33].

Assuming that there is only one free CLB, three possibilities were considered to define the rotation rule among the entire CLB array: random, horizontal, and vertical.

The random strategy was rejected for several reasons. If the placement algorithm (in an attempt to reduce path delays) gathers in the same area the logic needed to implement a given function, it would be unwise to disperse it: firstly, 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, a random rotation strategy would imply an unpredictable fault coverage latency, which it is not acceptable.

In the horizontal rotation strategy, illustrated in Fig. 11(a), the free CLB rotates along an horizontal path covering all the CLBs in the array, and the replication is performed between neighboring CLBs, due to scarcity of routing resources, and to avoid longer path delays. The same principle applies to the vertical rotation strategy, illustrated in Fig. 11(b), where the free CLB is rotated along a vertical path.

Practical experiments performed over a subset of the ITC'99 benchmarks using the last two strategies have shown that the application of the horizontal strategy leads to reconfiguration files that are in the average approximately 20% larger. This fact is a consequence of the configuration memory organization. When the rotation is done vertically, only one column of CLBs is involved in the process and thus the number of reconfiguration frames is restricted to one configuration column. However, at least two columns are involved when it is performed horizontally, and therefore, the number of reconfiguration frames is higher. The total number does not double because other frames related with the configuration of the interconnections—which may be established in global routing arrays belonging to neighbouring CLB columns—must also be reconfigured. Furthermore, the CLBs located in the top and bottom of the columns are effectively displaced horizontally, as can be seen in Fig. 11(b). The experimental results are presented in Table VI.

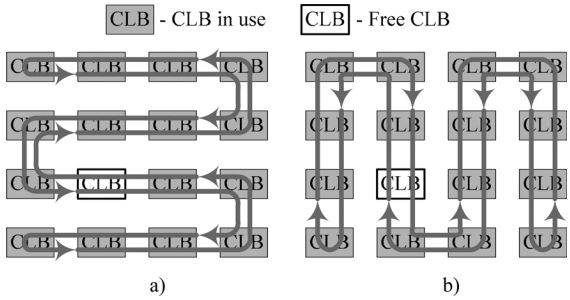

Fig. 11. Dynamic rotation of the free CLB: (a) Horizontal strategy and (b) vertical strategy.

TABLE VI
COST COMPARISON BETWEEN THE TWO ROTATION STRATEGIES

Ref.	Δ freq. (%)		Total reconf. files size (bytes)	
	Vertical	Horizontal	Vertical	Horizontal
B01	-5.5	0.0	48 350	56 102
B02	0.0	0.0	7 016	10 623
B03	-1.9	-4.9	120 705	138 484
B04	-6.1	-29.3	548 595	665 419
B05	-17.3	-36.9	1 130 985	1 286 031
B06	-2.7	0.0	45 291	53 503
B07	-23.6	-37.8	354 367	425 214
B08	-5.8	-5.8	150 093	178 339
B09	-1.8	-4.9	112 107	129 855
B10	-7.5	-7.6	195 571	245 455
B11	-10.5	-36.0	500 261	614 093
B12	0.0	-1.2	1 275 804	1 631 953
B13	-4.3	-42.8	258 827	332 954
B14	-13.5	-47.8	5 195 444	6 070 485

Concerning the influence over the maximum frequency of operation, there is no clearly defined pattern. The most relevant differences are related to the existence of a pair of dedicated paths 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 reestablished through generic routing resources, increasing the path delay. If the implemented circuit uses one or more 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. As such, in this case, the vertical strategy becomes preferable [33].

TABLE VII
AVERAGE DURATION OF A COMPLETE CLB TEST

CLB test time using the replication aid block	43.991 ms
CLB test time without using the replication aid block	39.797 ms
Free CLB test time	24.984 ms

When no carry signals are used, two other factors must be considered: 1) the number of signals with high fan-out and 2) the placement shape (rectangular, square, circular) and orientation (horizontal, vertical) of the circuits implemented inside the FPGA. In rectangular/horizontal implementations, and when many high fan-out signals are present, the horizontal strategy becomes preferable, since the maximum frequency of operation is less degraded, as proved by our experiments with a subset of the ITC'99 benchmarks. This could be a more important factor than the size of reconfiguration files, when dealing with high-speed applications.

Apart from those specific cases, we concluded that the vertical strategy performs better in the 14 circuits that were considered, with an average reduction in the maximum frequency of approximately 7% of its initial value, against 18% found for the horizontal strategy.

The "come and go" dynamic free-CLB rotation across the chip implies a variable test latency. The time to reach a given CLB once again alternates, according to the rotation direction, between a maximum and a minimum value, depending on the device size (number of CLB columns $(\mathrm{CLB}_{\mathrm{columns}})$ and rows $(\mathrm{CLB}_{\mathrm{rows}})$). The maximum fault detection latency is given by

$$\tau_{\mathrm{scan}_{\mathrm{MAX}}} = ((\#\mathrm{CLB}_{\mathrm{rows}} \times \#\mathrm{CLB}_{\mathrm{columns}}) - 1) \times 2 \times (t_{\mathrm{reconf}} + t_{\mathrm{test}}).$$

The minimum fault detection latency is in turn given by

$$\tau_{\mathrm{scan}_{\mathrm{min}}} = 2 \times (t_{\mathrm{reconf}} + t_{\mathrm{test}})$$

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

When this "come and go" rotation process is complete, the initial routing is restored. Therefore, no cumulative performance degradation is implied by the rotation mechanism. The whole process may be repeated or paused, depending on the overall test strategy.

It is not possible to give an exact value for the maximum fault detection latency, because it depends on the FPGA size and occupancy, and on the nature of the implemented circuits. The values obtained for the complete test of a CLB (including replication and release for test, if needed), taking into account the different CLB configuration types, and considering the replication of only one CLB at a time, are shown in Table VII.

These values may be used to accurately determine the test time considering FPGA size and occupancy rate, and the combinational or sequential circuits that are present. As an example, Table VIII shows the maximum test latency for a XCV200 FPGA with 1176 logic blocks, considering an occupancy rate of 75%, and with one third of the occupied CLBs requiring the use of the replication aid block. The time values shown correspond to two different TCK frequencies used in our experiments. These values were obtained considering the replication of only one CLB at a time and therefore are the worst possible values obtainable for this FPGA, as no scalability of the process was considered.

TABLE VIII
MAXIMUM TEST LATENCY OF A CLB

43 679.188 ms	TCK = 20 MHz
26 472.235 ms	TCK = 33 MHz

V. DEFRAGMENTATION

While the test of all FPGA resources ensures that the functionality of current and incoming functions will not be affected by structural faults, the absence of such faults is by no means able to guarantee a sustainable performance of the reconfigurable system as a whole. A good management of logic resources is also necessary to avoid delays due to temporary unavailability of resources to implement any function required at a given moment.

The possibility of dynamically reconfigure the FPGA enables considerable improvements over multiple context switching on the implementation of the virtual hardware concept [1]. Contrary to (static) full reconfiguration, dynamic partial configuration allows multiple applications to share the same logic resources in the spatial and temporal domains. However, the implementation of incoming functions may be disrupted by the fragmentation of the logic space. Since each function sharing the same FPGA has different logic requirements, many small pools of resources are created as those functions cease. These unallocated areas tend to become so small that they fail to satisfy any request and for that reason remain unused, leading to the fragmentation of the logic space [34]. This problem is illustrated in Fig. 12 in the form of successive reconfigurations of the same floorplan [35]. Each shadowed area corresponds to the optimal space occupied by the implementation of a single function. Despite the existence of enough free resources in the configurable logic space, the implementation of the incoming function on the nth partial reconfiguration is not possible. The free resources are scattered all over the FPGA logic space and not enough contiguous resources are available for its immediate implementation.

The fragmentation problem has been studied by some authors [13], [35]–[38]. Apart from presenting an analysis of the mechanism that leads to the fragmentation of the logic space, some algorithms are also proposed to perform (partial) rearrangements, in order to increase the allocation rate of waiting functions, while minimizing disruptions to running functions that have to be relocated. The problem with those approaches is their tendency to model the FPGA as a regular array structure and to regard defragmentation as a strictly packing problem. This assertion was true in the first generations of FPGAs, regarding the CLBs position inside the array, but it was inaccurate when other resources were considered. The presence of dedicated routing resources available to enhance specific applications, like counters and adders, are largely responsible for this irregularity. This problem became more complex in more recent generations, due

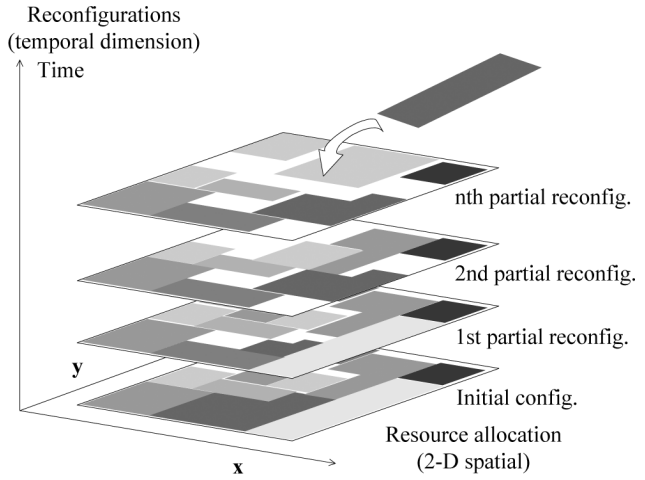

Fig. 12. Representation of the fragmentation problem.

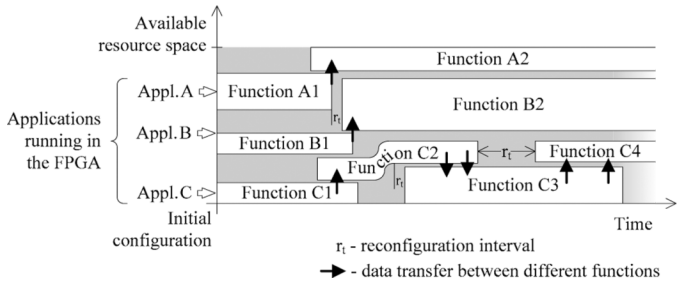

Fig. 13. Temporal scheduling of applications in the temporal and spatial domains.

to the introduction of memory and of dedicated digital signal processing (DSP) blocks distributed in the FPGA array. These aspects were taken into account in the work of Koester *et al.* [39]. However, to limit the complexity of the problem, the authors restrict the analysis to a 1-D-system approach, where the atomic unit for a partial reconfiguration is a CLB column and tasks can only be placed along the horizontal axes of the FPGA.

Suitable arrangements can be designed if the requirements of each function and their execution schedule are known in advance, but an increase in the available resources will in most cases be necessary to cope with the allocation problem [36]. However, when placement decisions have to be made at runtime, or when the need for extra space is only temporary, an increase in the available resources is a poor solution.

On the other hand, the reconfiguration intervals offered by new FPGAs are sufficiently small to enable functions to be swapped in real time. Partial reconfiguration times are in the order of a few milliseconds, depending on the configuration interface and on the complexity (and thus on the size) of the function being implemented. If a proper execution schedule is devised, it becomes feasible to use a single device to run a set of applications, which in total might require more than 100% of the FPGA resources, by swapping functions in and out of the FPGA as needed.

Furthermore, the reconfiguration time overhead may be virtually zero, if new functions are swapped in advance with those already out of use, as illustrated in Fig. 13 (where a number of applications share the same reconfigurable logic space in the temporal and spatial domains) [35]. After the execution of a given function, its logic resources may be reused to implement a new function during the interval r_t, so that it will already be available when required by the application flow (r_t should not be mistaken by the reconfiguration time). Notice that an increase in the degree of parallelism may retard the reconfiguration of incoming functions, due to lack of space in the FPGA. Consequently, delays will be introduced in application execution, systematically or not, depending on the application flow. Besides, an incoming function may require the relocation of other functions that are already implemented and running, in order to release enough contiguous space for its implementation (see function C2 in Fig. 13).

Although being a good solution, some relocation problems remain unsolved. In all the methods presented so far to perform defragmentation, the function to be relocated will be halted while partial reconfiguration is performed. This means that some or all the currently running functions are stopped regularly, erasing the gains obtained by running computing intensive functions in hardware rather than in software. Another issue is state preservation, related to the possibility of reading and writing register contents. Reading is straightforward, but writing is a different problem. In current Virtex FPGAs it is possible to read the content of FFs by a partial or full readback of the configuration memory. However, writing is not possible. A solution exists to get around this problem, but coherency problems can only be avoided if running functions are halted [39].

The same active replication procedure used to replicate the functionality of a CLB or an interconnection may also be used to defragment the FPGA logic space, by rearranging the placement of currently running functions. This procedure has the following two advantages:

1) defragmentation is performed concurrently with all running functions, without needing to halt their execution (no execution delay is introduced);
2) coherency of the register contents is guaranteed, preserving function state information.

As mentioned before, the placement algorithms (in an attempt to reduce path delays) gather in the same area the logic resources that are needed to implement a given function. Dispersing these resources, even if only during the relocation procedure, would generate longer paths (and hence, an increase in path delays), besides putting too much stress upon the limited routing resources. Therefore, the relocation process should take place between neighboring CLBs. If necessary, the relocation of a complete function may be carried out in several stages, to avoid an excessive increase in path delays during the relocation interval. A methodology to evaluate the impact of relocation over different functions is presented in [33].

Due to the scarcity of routing resources, it may be necessary to perform a rearrangement of the existing interconnections, to optimize their occupancy after the relocation of one or more CLBs, and to increase the availability of routing paths to incoming functions.

Since the access to the reconfiguration mechanism and the replication procedure are independent from the operation of the

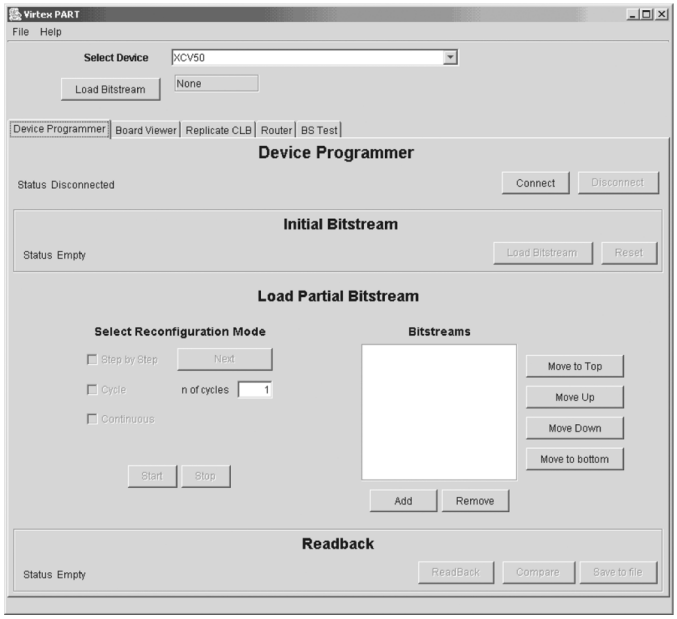

Fig. 14. Interface of the FPGA rearrangement and programming tool.

running functions, defragmentation may be implemented as a concurrent background process. A metric to determine when to perform defragmentation is proposed in [37]. Therefore, defragmentation may be run in advance and not only when a new incoming function is claiming area to be implemented. Waiting times will in this case be reduced improving the overall system performance.

The scalability of the proposed approach is guaranteed by the replication procedure (as seen before), but the defragmentation algorithm will dictate the end result. As an example, the defragmentation time in the methodology proposed by Koester *et al.* [39] increases proportionally with the number of columns involved.

VI. Replication, Rerouting, Test Management, and Programming Tool

The tool shown in Fig. 14 was developed to support the implementation of the active replication procedure. This tool is based on the JBits software—a set of Java classes that provide an application programming interface (API) to access the Xilinx FPGA bitstream [40]. This tool automatically produces the partial configuration files for the active replication procedure, greatly simplifying the task of the designer. The input information may be provided in the form of a complete configuration file (generated by the Xilinx design environment [41]) or by providing the coordinates (source and destination) of the CLB to be relocated.

This tool retains a complete copy of the current configuration, knowing exactly which resources are being used and which are free. Furthermore, it enables system recovery in case of a general failure.

The configuration of the CLB replica and of the replication aid block (when needed) is exclusively based on functions available on the JBits set. The configuration of the interconnections is based on a mix of functions from the JBits set, and on other functions that were specially developed to implement the proposed active replication procedure. Routing is based on the A$*$ algorithm, first described in 1968 by Hart *et al.* [42]. A$*$ incrementally builds paths leading from the starting point until it finds one that reaches the final point. Still, as an informed search algorithm, it only builds paths that appear to lead towards the final point, employing a heuristic estimation of the distance from any given point to the final one. Additionally, considering the specificities of the application, the search space was limited to the currently available interconnection resources. A$*$ is complete and optimal, i.e., A$*$ will always find the shortest path inside the search space, if it exists, since it takes into account the distance already travelled.

Another problem had yet to be solved. During the replication of the CLBs, their outputs have to be placed in parallel, meaning that the new path will have to intersect the old path somewhere. This is not a usual procedure since it "short-circuits" two signals coming from two outputs, and as such it was not supported by JBits. An algorithm was therefore developed to perform an exhaustive search along the original output path until finding an interconnection where the signal may be intersected. Due to Virtex architectural restrictions, only those points located in the far end of single-length lines may be intersected. After the original placement of a subset of circuits from the ITC'99 Benchmark Circuits [28] using the Xilinx design environment [41], many signals did not satisfy this condition. A Java based tool was developed to edit the original configuration and analyze all paths between CLBs, changing those that did not use a single-length line. All functions implemented in the FPGA should adhere to this constraint to be replicable.

Another module of this tool addresses test application and response capturing and comparison, fully implementing the proposed test solution. All the interaction between the FPGA and the outside world is carried out via the BS interface [23].

VII. Conclusion

This paper describes an active replication procedure that enables a non-intrusive online relocation of active functions implemented in dynamically reconfigurable FPGAs. The proposed procedure:

1) releases currently occupied resources for test, allowing the implementation of a truly online concurrent structural test of the FPGA; additionally, transient faults in the configuration memory and—to some extent—in function registers, may also be corrected;
2) enables the online management of FPGA resources, supporting the rearrangement of active functions and the defragmentation of the FPGA logic space, aiming to facilitate the allocation of incoming functions.

Based on this procedure an online concurrent test methodology for the structural test of dynamically reconfigurable FPGAs was presented. The tool developed to support the implementation of the replication procedure also automates the test of Xilinx Virtex FPGAs.

Extensive experimental work was conducted and the results presented in this article demonstrate the effectiveness of the proposed active replication procedure and of the non-intrusive online concurrent structural test methodology.

A common solution to test and defragment dynamically reconfigurable SRAM-based FPGAs improves the reliability and availability of reconfigurable computing systems, which are largely based on this type of devices.

REFERENCES

[1] X. P. Long and H. Amano, "WASMII: A data driven computer on a virtual hardware," in *Proc. 1st IEEE Workshop FPGAs Custom Comput. Mach.*, 1993, pp. 33–42.

[2] J. M. P. Cardoso and H. C. Neto, "An enhanced static-list scheduling algorithm for temporal partitioning onto RPUs," in *Proc. 10th Int. Conf. VLSI*, 1999, pp. 485–496.

[3] R. Maestre, F. J. Kurdahi, R. Hermida, N. Bagherzadeh, and H. Singh, "A formal approach to context scheduling for multicontext reconfigurable architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 1, pp. 173–185, Feb. 2001.

[4] M. Sanchez-Elez, M. Fernandez, R. Maestre, R. Hermida, N. Bagherzadeh, and F. J. Kurdahi, "A complete data scheduler for multi-context reconfigurable architectures," in *Proc. ACM/IEEE Int. Conf. Des., Autom. Test Eur.*, 2002, pp. 547–552.

[5] S. Lange and M. Middendorf, "Models and reconfiguration problems for multi task hyperreconfigurable architectures," in *Proc. 18th Int. Parallel Distrib. Process. Symp.*, 2004, pp. 135–142.

[6] N. R. Shnidman, H. Mangione-Smith, and M. Potkonjak, "On-line fault detection for bus-based field programmable gate arrays," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 4, pp. 656–666, Dec. 1998.

[7] M. Ceschia, M. Violante, M. S. Reorda, A. Paccagnella, P. Bernardi, M. Rebaudengo, D. Bortolato, M. Bellato, P. Zambolin, and A. Candelori, "Identification and classification of single-event upsets in the configuration memory of SRAM-based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 6, pp. 2088–2094, Dec. 2003.

[8] M. Bellato, P. Bernardi, D. Bortolato, A. Candelori, M. Ceschia, A. Paccagnella, M. Rebaudengo, M. S. Reorda, M. Violante, and P. Zambolin, "Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA," in *Proc. ACM/IEEE Int. Conf. Des., Autom. Test Eur.*, 2004, pp. 584–589.

[9] H. Quinn, P. Graham, J. Krone, M. Caffrey, S. Rezgui, and C. Carmichael, "Radiation-induced multi-bit upsets in Xilinx SRAM-based FPGAs," in *Proc. Military Aerosp. Appl. Program. Logic Devices Conf.*, 2005, p. E6.

[10] F. Hanchek and S. Dutt, "Methodologies for tolerating cell and interconnect faults in FPGAs," *IEEE Trans. Computers*, vol. 47, no. 1, pp. 15–33, Jan. 1998.

[11] J. Lach, H. W. Mangione-Smith, and M. Potkonjak, "Low overhead fault-tolerant FPGA systems," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 2, pp. 212–221, Jun. 1998.

[12] M. Abramovici, C. Stroud, S. Wijesuriya, C. Hamilton, and V. Verma, "On-line testing and diagnosis of FPGAS with roving STARs," in *Proc. 5th IEEE Int. On-Line Testing Workshop*, 1999, pp. 2–7.

[13] M. Handa and R. Vemuri, "An efficient algorithm for finding empty space for online FPGA placement," in *Proc. ASP/DAC Des. Autom. Conf.*, 2004, pp. 960–965.

[14] C. Stroud, S. Konala, P. Chen, and M. Abramovici, "Built-in self-test of logic blocks in FPGAs (finally, a free lunch: BIST without overhead!)," in *Proc. 14th IEEE VLSI Test Symp.*, 1996, pp. 387–392.

[15] C. Stroud, E. Lee, and M. Abramovici, "BIST-based diagnostic of FPGA logic blocks," in *Proc. Int. Test Conf.*, 1997, pp. 539–547.

[16] C. Stroud, S. Wijesuriya, C. Hamilton, and M. Abramovici, "Built-in self-test of FPGA interconnect," in *Proc. Int. Test Conf.*, 1998, pp. 404–411.

[17] A. Doumar, T. Ohmameuda, and H. Ito, "Design of an automatic testing for FPGAs," in *Proc. IEEE Eur. Test Workshop*, 1999, pp. 152–157.

[18] W. K. Huang, F. J. Meyer, X. Chen, and F. Lombardi, "Testing configurable LUT-based FPGA's," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 6, no. 2, pp. 276–283, Jun. 1998.

[19] W. K. Huang, F. J. Meyer, and F. Lombardi, "An approach for detecting multiple faulty FPGA logic blocks," *IEEE Trans. Computers*, vol. 49, no. 1, pp. 48–54, .Jan 2000.

[20] T. Inoue, S. Miyazaki, and H. Fujiwara, "Universal fault diagnosis for look-up table FPGAs," *IEEE Des. Test Comput.*, vol. 15, no. 1, pp. 39–44, Jan.–Mar. 1998.

[21] M. Renovell, J. M. Portal, J. Figueras, and Y. Zorian, "RAM-based FPGA's: A test approach for the configurable logic," in *Proc. ACM/IEEE Int. Conf. Des., Autom. Test Eur.*, 1998, pp. 82–88.

[22] M. Renovell, J. M. Portal, J. Figueras, and Y. Zorian, "Testing the interconnect of RAM-based FPGAs," *IEEE Des. Test Comput.*, vol. 15, no. 1, pp. 45–50, Jan.–Mar. 1998.

[23] *IEEE Standard Test Access Port and Boundary Scan Architecture*, IEEE Standard 1149.1, 2001.

[24] A. L. Burress and P. K. Lala, "On-line testable logic design for FPGA implementation," in *Proc. Int. Test Conf.*, 1997, pp. 471–478.

[25] M. Renovell, J. M. Portal, P. Faure, J. Figueras, and Y. Zorian, "Test generation optimization for a FPGA application-oriented test procedure," in *Proc. 15th Des. Circuits Integr. Syst. Conf.*, 2000, pp. 330–336.

[26] M. G. Gericota, G. R. Alves, M. L. Silva, and J. M. Ferreira, "Active replication: Towards a truly SRAM-based FPGA on-line concurrent testing," in *Proc. 8th IEEE On-Line Testing Workshop*, 2002, pp. 165–169.

[27] M. G. Gericota, G. R. Alves, M. L. Silva, and J. M. Ferreira, "On-line defragmentation for run-time partially reconfigurable FPGAs," in *Proc. 12th Int. Conf. Field Program. Logic Appl.*, 2002, pp. 302–311.

[28] Politecnico di Torino, ITC'99 Benchmarks 1999. [Online]. Available: http://www.cad.polito.it/tools/itc99.html

[29] M. G. Gericota, G. R. Alves, M. L. Silva, and J. M. Ferreira, "Run-time defragmentation for dynamically reconfigurable hardware," in *New Algorithms, Architectures and Applications for Reconfigurable Computing*, 1st ed. Dordrecht, The Netherlands: Springer, 2005, ch. 10, pp. 117–129.

[30] W. Huang and E. J. McCluskey, "A memory coherence technique for online transient error recovery of FPGA configurations," in *Proc. 9th ACM Int. Symp. Field-Program. Gate Arrays*, 2001, pp. 183–192.

[31] S. Mitra, N. Seifert, M. Zhang, Q. Shi, and K. S. Kim, "Robust system design with built-in soft-error resilience," *Computer*, vol. 38, no. 2, pp. 43–52, Feb. 2005.

[32] M. G. Gericota, G. R. Alves, M. L. Silva, and J. M. Ferreira, "Programmable logic devices: A test approach for the input/output blocks and pad-to-pin interconnections," in *4th IEEE Latin-Amer. Test Workshop Dig. Papers*, 2003, pp. 72–77.

[33] M. G. Gericota, L. F. Lemos, G. R. Alves, and J. M. Ferreira, "A new approach to assess defragmentation strategies in dynamically reconfigurable FPGAs," in *Proc. 2nd Int. Workshop Appl. Reconfigurable Comput.*, 2006, pp. 262–267.

[34] M. Vasilko, "DYNASTY: A temporal floorplanning based CAD framework for dynamically reconfigurable logic systems," in *Proc. 9th Int. Workshop Field-Program. Logic Appl.*, 1999, pp. 124–133.

[35] O. Diessel, H. El Gindy, M. Middendorf, H. Schmeck, and B. Schmidt, "Dynamic scheduling of tasks on partially reconfigurable FPGAs," *IEE Proc.-Comput. Digit. Technol.*, vol. 147, no. 3, pp. 181–188, May 2000.

[36] J. Teich, S. Fekete, and J. Schepers, "Compile-time optimization of dynamic hardware reconfigurations," in *Proc. Int. Conf. Parallel Distrib. Process. Techn. Appl.*, 1999, pp. 1097–1103.

[37] A. Ejnioui and R. F. DeMara, "Area reclamation strategies and metrics for SRAM-based reconfigurable devices," in *Proc. Int. Conf. Eng. Reconfig. Syst. Algorithms*, 2005, pp. 196–202.

[38] P. C. Vinh and J. P. Bowen, "Continuity aspects of embedded reconfigurable computing," *Innovations Syst. Softw. Eng.*, vol. 1, no. 1, pp. 41–53, Apr. 2005.

[39] M. Koester, M. Porrmann, and H. Kalte, "Relocation and defragmentation for heterogeneous reconfigurable systems," in *Proc. Int. Conf. Eng. Reconfig. Syst. Algorithms*, 2006, pp. 70–76.

[40] S. A. Guccione, D. Levi, and P. Sundararajan, "JBits java based interface for reconfigurable computing," in *Proc. 2nd Military Aerosp. Appl. Program. Devices Technol. Conf.*, 1999, pp. 1–9.

[41] Xilinx, San Jose, CA, "Xilinx ISE tools," 2003. [Online]. Available: http://www.xilinx.com

[42] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

Manuel G. Gericota (M'00) was born in Porto, Portugal, in 1968. He received the B.S., M.Sc., and Ph.D. degrees in electrical and computer engineering from the University of Porto, Porto, Portugal, in 1992, 1995, and 2003, respectively.

Since 1994, he has been a Professor with the Department of Electrical Engineering, School of Engineering, Polytechnic Institute of Porto (IPP), Porto, Portugal. His research interests include FPGA test, test methodologies for reconfigurable hardware systems, and dynamic allocation of hardware resources

in dynamically reconfigurable systems. He has authored or coauthored more than 40 research papers in these areas and served as a reviewer for several journals and conferences.

Dr. Gericota is a member of the IEEE Computer Society, the European Design and Automation Association, the IEEE European Test Technology Technical Council, and the Portuguese Association of Engineers.

Gustavo R. Alves was born in Porto, Portugal, in 1968. He received the B.S., M.Sc., and Ph.D. degrees in electrical and computer engineering from the University of Porto, Porto, Portugal, in 1991, 1995, and 1999, respectively.

Since 1994, he has been an Adjunct Professor with the Department of Electrical Engineering, School of Engineering, Polytechnic Institute of Porto (IPP), Porto, Portugal. His research interests include design for debug and test, reconfigurable systems, and remote experimentation in e-learning contexts. He has authored or coauthored 2 book chapters and more than 80 research papers in these areas.

Dr. Alves is a member of the Portuguese Association of Engineers. He has served as a reviewer for several journals and conferences, including the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS and the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS.

Miguel L. Silva was born in Porto, Portugal, in 1976. He received the B.Sc. in electrical and computer engineering and the M.Sc. degree in artificial intelligence and computation from the University of Porto, Porto, Portugal, in 2000 and 2004, respectively, where he is currently pursuing the Ph.D. degree in electrical and computer engineering.

His research interests include dynamic reconfigurable systems, FPGAs, configurable resource management, and CAD tools.

José M. Ferreira was born in Porto, Portugal, in 1959. He received the B.S., M.Sc., and Ph.D. degrees in electrical and computer engineering from the University of Porto, Porto, Portugal, in 1982, 1987, and 1992, respectively.

In 1982, he joined the University of Porto (FEUP), where he is currently an Associate Professor. He also holds a position of Professor II with the Buskerud College of Engineering, Kongsberg, Norway. His nonacademic experience includes Texas Instruments and Efacec. He is the author of one book on digital hardware design, coauthor of approximately 100 papers, and served as a reviewer for several journals and conferences. His interests are related to research and education in digital hardware development and test, an area where he participated in various national and international projects.