On-line Testing of FPGA Logic Blocks Using Active Replication

Manuel G. Gericota[1], Gustavo R. Alves[1], Miguel L. Silva[2], José M. Ferreira[2,3]

[1] Department of Electrical Engineering – DEE/ISEP
Rua Dr. António Bernardino de Almeida, 4200-072 Porto – PORTUGAL
{mgg, galves}@dee.isep.ipp.pt

[2] Dep. of Electrical and Computers Engineering – DEEC/FEUP
Rua Dr. Roberto Frias, 4200-465 Porto – PORTUGAL
{mlms, jmf}@fe.up.pt

[3] Institutt for datateknikk - Høgskolen i Buskerud
Frogsvei 41 - 3603 Kongsberg - NORWAY

Abstract. This paper presents a novel on-line test method for partial and dynamically reconfigurable FPGAs, based on the active replication of configurable logic blocks (CLBs). Each CLB in the FPGA is released for test and again made available to be reused if fault-free, or removed from operation if faulty. The proposed method continuously scans the whole FPGA without disturbing system operation. It implies a very low overhead at chip level, since all the test actions are carried out through the IEEE 1149.1 standard boundary-scan architecture and test access port. Moreover, it presents the additional benefit of correcting transient faults, such as single event upsets in space environments, which would otherwise become permanent faults and most likely introduce functional restrictions or even halt the system.

1 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 gains in board space and flexibility. With the advent of a new kind of SRAM-based FPGAs, capable of implementing fast run-time partial reconfiguration (e. g. the Virtex family from Xilinx used to validate this work), the advantages of these devices were considerably reinforced, wide-spreading their usage as a base for reconfigurable computing platforms.

Unfortunately, as in the rest of the semiconductor industry, the technological trends that made newer FPGAs more appealing and affordable also threaten their reliability. Smaller submicron scales increase the probability of electromigration, due to higher electronic current density in metal traces. Also, the corresponding lower threshold voltages make them more susceptible to gamma particle radiation. Interference from such radiation is much more likely with larger FPGA dies, increasing the probability of failure [1, 2]. After large periods of operation, certain defects, related to manufacturing imperfections, that are not large enough to influence initial testing, become exposed, emerging as either stuck-at (permanent) or transient faults [3].

An on-line FPGA test methodology should be able to cover permanent and transient faults. In the case of permanent faults, and after the faulty elements (either CLBs or routing resources) are located, the FPGA can be reconfigured to exclude their usage. Previously unused FPGA resources can replace these faulty elements, improving dependability with a very small hardware redundancy. For transient faults, on-line

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

partial reconfiguration enables the recovery of errors in the on-chip configuration memory cells, that modify the logic functionality, namely Single Event Upsets (SEUs). Such upsets manifest themselves as permanent faults because of the change in functionality, and cannot be recovered by traditional transient fault recovery techniques, such as rollback or roll-forward. However, the cause of the failure is actually transient [4].

A higher level of dependability for a reconfigurable system can therefore only be achieved through the continuous testing of all its FPGAs throughout its lifetime, and by the introduction of error correction/fault tolerance features. Using the same partial Run--Time Reconfigurable (RTR) features that originated this test challenge, a novel on-line testing approach, based on a scanning strategy, is proposed in this paper, with the following main characteristics:

1. it is the first truly non-intrusive CLB test method proposed in the literature;
2. it is able to detect any permanent structural fault in the CLB emerging at any stage during the system lifetime;
3. it is able to correct any transient fault affecting the CLB functionality;
4. no FPGA I/O pins are occupied with test functions, since its complete implementation is based on the IEEE 1149.1 infrastructure [5].

This paper is organised as follows: recently proposed approaches for testing SRAM--based FPGAs are first reviewed, followed by a general description of the on-line testing solution envisaged for the FPGA CLBs. The next three sections present the components of the proposed methodology: the replication procedures, the dynamic rotation mechanism used to release the CLBs for testing, and the strategy used to test them. All the proposed solutions were experimentally validated. We conclude this paper by presenting some directions for further research.

2 Background

Many different approaches based on off-line or on-line testing strategies have been proposed by different authors to test and diagnose FPGA faults. An off-line Built-In Self-Test (BIST) technique is presented in [6-7], which exploits the FPGA reprogrammability features in order to set up the BIST logic. Some of the logic blocks are configured as pattern generators or response analysers, 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 to all levels of testing (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 its normal operation.

A slightly different BIST technique, which implies structural modifications on the original configuration memory, is proposed in [8]. When compared to similar BIST techniques, this method reduces test time and the required off-chip memory, while enabling the automation of the test process. The modification required at the internal hardware level of the FPGA is a major disadvantage, implying the non-universality of the solution.

An off-line testing methodology based on a non-BIST approach, targeted to test the FPGA CLBs, is presented in [9]. After a specific test configuration is set up, the FPGA Input/Output Blocks (IOBs) are used to support the external application of test vectors and to capture the test responses. In order to achieve 100% fault coverage at CLB level, and due to the configurable characteristic of the FPGAs, different test configurations must be programmed and specific sets of test vectors applied in each case. Using a

classical divide-to-conquer strategy, extensive work on the structural testing of FPGA Look-Up Tables (LUT) and interconnections was also presented in [10, 11].

The previous approaches are restricted to manufacturing test, since they require the device to be off-line, increasing fault-detection latency, which may not be admissible in highly fault-sensitive, mission-critical applications. In order to overcome these limitations, on-line testing and diagnosis methods based on a scanning strategy were presented in [3, 12]. The idea underlying these methods is to have only a relatively small portion of the chip being tested off-line (instead of the whole chip as in previous proposals), while the rest continues its normal on-line 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 can be taken off-line and tested in a completely transparent way (i.e. without interrupting the device functionality). This fault scanning procedure then moves on to copy and test another set of elements, sweeping through the whole FPGA, systematically testing for faults. However, in the first approach [3], a modification in the FPGA cells structure is required to implement the replication procedure. On the other hand, in the second approach [12], known as Roving STARs (Self-Testing AReas), the whole system must be stopped in order to relocate an entire CLB column. Since reconfiguration is performed through the Boundary Scan (BS) infrastructure, reconfiguration time is long, and it seems likely that halting the system will disturb its operation.

The on-line testing approach proposed in this paper reuses some of the previous ideas, but eliminates their drawbacks by using a new concept – the active replication – which enables the relocation of each CLB functionality without halting the system, even if the CLB is active, i.e. the CLB is part of an implemented function that is actually being used by the system [13]. A dynamic rotation mechanism ensures that the whole FPGA CLBs are released and tested within a given latency.

The exclusive (re)use of the BS test infrastructure to release and test the CLBs brings the additional benefit of a reduced overhead at board level, since no other resources (than those of the FPGA itself) are used. Being application-independent, and oriented to test the FPGA structure, the proposed strategy guarantees FPGA reliability after many reconfigurations, thus helping to ensure the correct operation throughout the system lifetime.

3 The proposed on-line testing approach

Conceptually, an FPGA could be visualised as an array of uncommitted CLBs, surrounded by a periphery of IOBs, which are interconnectable by configurable routing resources, whose configuration is controlled by a set of memory cells that lies beneath.

On a given application, 100% usage of the FPGA resources is hardly ever achieved, even when independent hardware blocks dynamically share the same device (in the case of a dynamically reconfigurable hardware system). For that reason, a few blocks will always be free. Therefore, it is possible to consider a strategy to test temporarily unused blocks, without disturbing system operation, taking advantage of the dynamic and partially reconfigurable features offered by newer FPGAs.

After being tested, unused defect free CLBs remain available as spare parts, ready to replace others eventually found defective. Through a dynamic rotation mechanism the CLBs currently being used are released for testing, after their current functionality have been relocated to other CLBs already tested. For simplicity of analysis we divide our approach in three parts: the replication procedure, the dynamic rotation mechanism and the testing strategy. In the next three sections each of these phases is analysed in detail.

4 Replicating active CLBs

The configuration memory is partitioned into one-bit wide vertical frames grouped into larger units called configuration columns. To each CLB column corresponds a configuration column, with multiple frames, which mixes internal CLB configuration and state information, and column routing and interconnect information. The configuration process is a sequential mechanism that spans through some (or eventually all) CLB configuration columns. When replicating an active CLB more than one column may be affected, since its input and output signals (as well as those in its replica) may cross several columns before reaching its source or destination. Any reconfiguration action must therefore ensure that the signals from the replicated CLB are not broken before being totally re-established from its replica, otherwise its operation will be disturbed or even halted. Furthermore, the functionality of the CLB replica must be perfectly stable before its outputs are connected to the system to avoid output glitches. A set of experiments performed with a Virtex FPGA from Xilinx demonstrated that the only possible solution is to divide the replication process in two phases, as illustrated in figure 1.

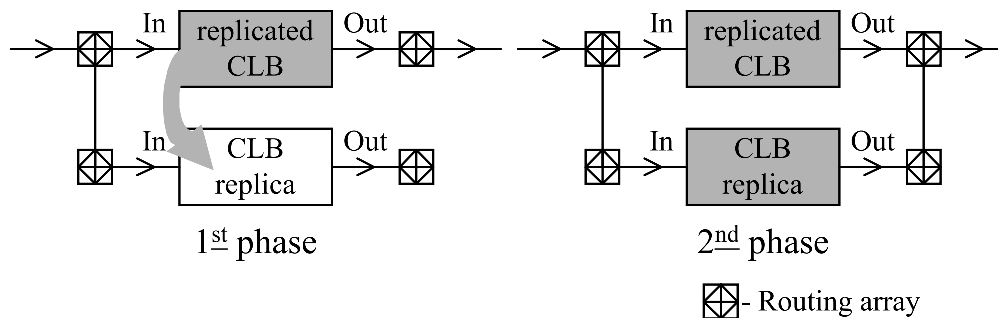

Fig. 1. Two-phase CLB replication process.

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

Another major requirement for the success of the replication process is the correct transferring of state information. If the current CLB function is purely combinational, a simple read-modify-write configuration procedure will suffice to accomplish the replication process. However, in the case of a sequential function, the internal state information must be preserved and no write-operation may fail during the copying process. In the Virtex FPGA family, each CLB comprises four logic cells with one register each, which can be configured as a latch or as a flip-flop (FF). Although it is possible to read the value of a register, it is not possible to perform a direct write operation. Moreover, when dealing with active CLBs, if state information changes between the read and write of a register, it will cause a coherency problem. By this reason, no time gap between the two operations may exist.

The solution to this problem depends on the type of implementation. In this paper we shall consider three implementation cases: synchronous free-running clock circuits, synchronous gated-clock circuits and asynchronous circuits.

When dealing with synchronous free-running clock circuits, the two-phase replication process that was previously described solves this problem. Between the first and the second phase, the CLB replica has the same inputs as the replicated CLB and all its four FFs acquire the state information. This statement holds true even when the system has a lower frequency than the BS clock (which is used for reconfiguration purposes), due to the low-speed nature of the serial BS infrastructure and also due to the overhead bits of the reconfiguration file. Several experiments made using this class of circuits have shown the effectiveness of this method in the replication of active CLBs. No loss of state information or the presence of output glitches was observed. Notice that this procedure is valid even when dealing with asynchronous circuits, if the longest update interval of the latches is shorter than the time interval between the first and the second phases.

Despite the effectiveness of this solution, its usefulness is very restricted. A broad range of applications use synchronous gated-clock circuits, instead of free-running clocks, where input acquisition by the FF is controlled by the clock enable signal. In such cases, we cannot 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 this time at the input of the replica FFs could be different from the last one captured by the replicated FFs, and a coherency problem will then occur. Furthermore, an updating of the state of the FFs could occur at any time during the replication process.

To solve this problem we used a replication aid block, which manages the transference of the state value from the replicated FFs to the replica FFs, while enabling its update by the circuit, at any instant, without losing the new state information or delaying the replication process. The whole replication scheme is represented in figure 2 for a single CLB cell (for this purpose each of the four logic cells of a CLB can be considered individually), while figure 3 represents the flow diagram of the replication process.

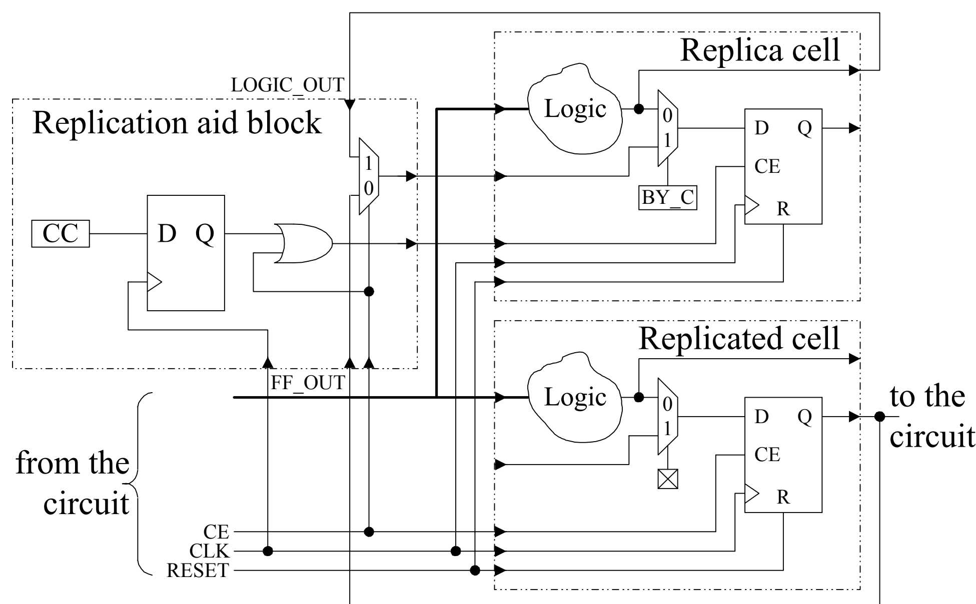

Fig. 2. Implementation of the synchronous gated-clock flip-flop replication scheme.

The inputs of the 2:1 multiplexer present in the replication aid block receive one temporary transfer path from the output of the replicated FF (FF_OUT), and another one from the output of the combinational logic block configured 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, generated by the replication aid block (capture control signal - CC), forces the replica FF to store the transferred value. The replica FF acquires the state information present in the replicated FF. If the CE signal is active or is activated during this process, the multiplexer selects the LOGIC_OUT signal and applies it to the input of the replica FF, which is updated at the same time and with the same value as the replicated FF, therefore guaranteeing state coherency. The control signals CC and BY_C are configuration memory bits whose values are driven through reconfiguration of this memory. BY_C directs the state signal to the input of the replica FF, while CC enables its acquisition. It is therefore possible to control the whole replication process through the BS infrastructure, and as such no extra pins are required. The implementation of the replication aid block occupies a single CLB slice. Since four of these blocks are required to replicate the four logic cells of a CLB, two extra CLBs will be needed to implement this process. Since we decided to control all signals through the configuration memory, the CC net includes the FF shown in figure 2. However, it is there simply as a consequence of the structure of the CLB slice, and does not play any role in the process.

After the state has been transferred, the CLB inputs involved in the process are placed in parallel, all the signals to and from the replication aid block are disconnected, and the outputs are also placed in parallel. After at least one clock cycle, the outputs of the replicated block are disconnected, followed by its inputs, becoming free to be tested. Each of these steps (corresponding to a square in the flow diagram shown in figure 3) implies a new reconfiguration file. A total of 9 files are therefore needed just to get the replication done, instead of two, as would only 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/or BY_C, only one configuration frame is needed, which is less than 1 Kbit (50 μs for a 20 MHz BS clock frequency).

Practical experiments performed using a Virtex XCV200 over the ITC’99 Benchmark Circuits from the Politécnico di Torino [14] demonstrated the effectiveness of our approach. These circuits are purely synchronous with only one single-phase clock signal present. Nevertheless, these procedures are also applicable to multiple clock/multiple phase circuits, since only one clock signal is involved in the replication process at a time, provided that the slowest clock period is higher than the duration of the replication process.

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

The same two-phase procedure is effective on the replication and release for testing of local and global active interconnections. The interconnections to be replicated are first duplicated in order to establish an alternative path, and then disconnected, and made available for test. In fact, handling interconnects proves to be easier than handling CLBs.

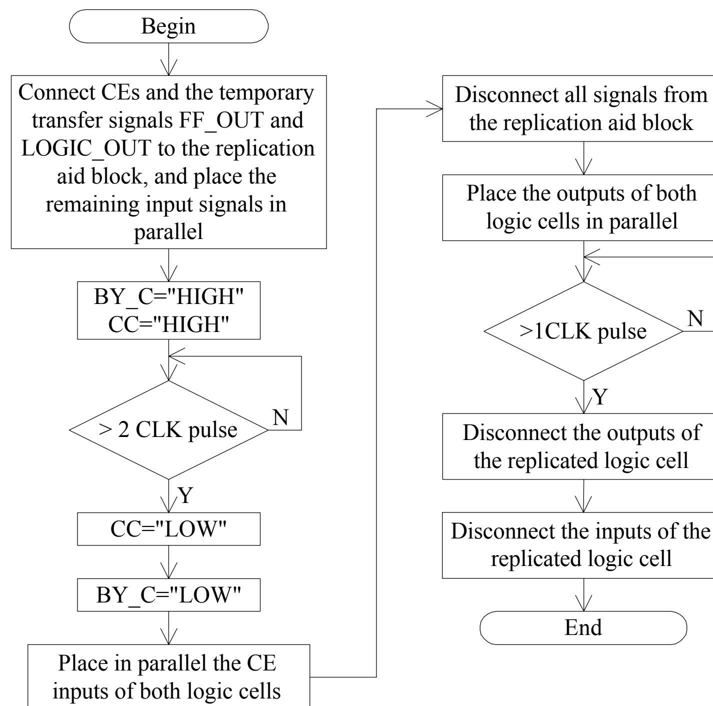

Fig. 3. Replication process flow.

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 could 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 a write-operation takes place during the replication interval, as stated in [4]. Furthermore, since frames span an entire column of CLB slices, the same LUT bit for all slices is written with a single command (e. g. a bit-write operation for LUT bit 0 in one slice affects the same bit on all slices of that particular column). We must ensure that either all the remaining data in the slice is constant, or it is also changed externally through partial reconfiguration. Even not being replicated, LUT/RAMs should not lie in any column that could be affected by the replication process.

According to the overall test strategy, this method could be used to replicate more than one CLB at once, improving the scalability of the process, which is important if we are dealing with large FPGAs. However, problems due to the limited number of interconnections available restrict the number of CLBs that can be replicated at once.

4.1 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 input of the replica FFs, which are the same as those present at the input of the replicated FFs. Therefore, the acquired state information is correct, despite any fault that may affect the replicated CLB FFs. As a consequence, and after the replication process, the outputs of the CLB replica always display the correct values, automatically correcting any faulty behaviour. On the other hand, when replicating synchronous gated-clock circuits (or asynchronous circuits), a truly state transfer operation is performed, so if a permanent fault in the replicated CLB affects the value held by one of the FFs (or latches), this fault is propagated to the replica CLB and will remain active until that FF (or latch) is updated. The fault in the replicated CLB will be detected

during the subsequent test phase and possibly the CLB will be flagged as defective, meaning that it will not be used again in a later reconfiguration.

Depending on the method used to create the reconfiguration files, the replication procedure can recover errors that modify the circuit function, caused by transient faults in the on-chip configuration memory cells. SEUs in space environments are a typical example of such errors, which reveal themselves as permanent faults, regardless of the fact that the cause of the failure is actually transient [4]. Since Virtex FPGAs enable readback operations, a completely automatic read-modify-write procedure could be implemented to replicate the CLBs using local processing resources. In this case, any transient fault in the configuration memory is not corrected, being propagated to the replica. On the other hand, if the reconfiguration files are generated from the initial configuration file stored in an external memory, any error due to SEUs is corrected when the affected blocks are replicated.

4.2 Behaviour of routing resources

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

Each CLB comprises, in addition to its logic resources, three associated routing arrays: two local arrays (input and output) and one global array. With the exception of a pair of dedicated paths that provides high-speed connections between vertically adjacent CLBs, for the propagation of carry signals, no routing resources are available in the local arrays to establish direct interconnections with other CLBs. As such, the interconnections required by the replication process can only be done through the global routing array.

Between local and global routing arrays only unidirectional routing resources are available. To place the inputs in parallel, interconnection segments between global arrays may be unidirectional (from the replicated CLB inputs towards the CLB replica inputs), or bi-directional. Concerning the outputs, interconnection segments between global arrays may also be unidirectional (from the CLB replica outputs towards the replicated CLB output), or bi-directional. Otherwise, since signals do not propagate backwards, no signals will exist at the inputs of the CLB replica, and the outputs of both CLBs will not be placed in parallel. As a result, output glitches will occur when the outputs of the replicated CLB are disconnected from the system, and no signals will be propagated to the rest of the circuit.

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

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

5 The dynamic rotation mechanism

The dynamic rotation mechanism used to release the CLBs to be tested should have a minimal influence (preferably none) in the system operation, as well as a reduced overhead in terms of reconfiguration cost. This cost depends on the number of reconfiguration frames needed to replicate and release each CLB, since a great number of frames would imply a longer test time. The impact of this process in the overall system operation is mainly related to the delays imposed by re-routed paths, since the rotation process might imply a longer path, thus diminishing the maximum frequency of operation (the longest path delay determines the maximum frequency of operation).

Supposing that there is only one free CLB, three possibilities were considered for establishing 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 the components of a given application, it would be unwise to disperse the blocks: 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 defect coverage latency, which it is not acceptable.

In the horizontal rotation strategy the free CLB rotates along an horizontal path covering all the CLBs in the array, and the replication is performed between neighbouring CLBs, due to scarcity of routing resources and to avoid higher path delays. The same principle applies to the vertical rotation strategy, where the free CLB is rotated along a vertical path.

Practical experiments performed over a subset of the ITC'99 benchmarks using the last two strategies, have shown that the size of the reconfiguration files obtained by the application of the horizontal strategy was approximately 20% higher than the reconfiguration files obtained by the application of the vertical strategy to the same circuit implementations. However, the influence over the maximum frequency of operation was substantially different, mainly due to the pair of dedicated paths per CLB that propagate carry signals vertically between adjacent CLBs. When the rotation process breaks a dedicated carry path, due to the insertion of the free CLB, the propagation of this carry signal between the nearest adjacent CLBs (above and below the free CLB) is re-established through generic routing resources, increasing the path delay. If the implemented circuit has one or more of these carry signals, the horizontal rotation would break all the carry nets, increasing path delays, but the vertical rotation would break only those in the top or bottom of the CLB columns. As such, in this case, the vertical strategy becomes preferable.

When no carry signals are used, two other factors must be considered: i) the number of signals with high fanout, and ii) the placement shape (rectangular, square, circular) and orientation (horizontal, vertical) of the circuits implemented inside the FPGA. In rectangular/horizontal implementations, and when many high fanout signals are present, the horizontal strategy becomes preferable, since the maximum frequency of operation is less degraded (this could be a more important factor than reconfiguration files size when dealing with high-speed applications).

Generally, on the 14 circuits that were considered, the vertical strategy performs better, with a mean value reduction in the maximum frequency of approximately 7% of its initial value, against 18% found for the horizontal strategy.

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

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

The minimum fault detection latency is in turn given by

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

where:

t_{reconf}: time needed to complete a CLB replication

t_{test}: time needed to test a free CLB

CLB_{rows}: number of CLB rows in the FPGA

$CLB_{columns}$: number of CLB columns in the FPGA

When this back and forth rotation process is complete, the initial routing is restored. The scanning process may then be repeated or paused, depending on the overall test strategy.

The average replication time of each CLB implementing synchronous gated-clock circuits is about 22,6 ms, for a 20 MHz frequency of the BS clock, when reconfiguration is done through the BS infrastructure (as it was the case in this experiment).

6 The test strategy

The BS infrastructure is also reused to apply test vectors and to capture test responses, with the outputs of the CLB(s) under test being routed to unused BS register cells associated to the IOBs. Nevertheless, it is not possible to apply the test vectors through the BS register without affecting the values present at each FPGA input, so an alternative User Test Register must be used (the Virtex family enables the definition of two user registers controlled through the BS infrastructure). Figure 4 illustrates the test infrastructure set up that is required to carry out this procedure.

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 each CLB 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.

To completely test the structure of the CLB a number of test configurations has to be used and several test vectors applied. Since the implementation structure of the CLB elements (LUTs, multiplexers, FFs) is not known, a hybrid fault model was considered [10]. To achieve 100% fault coverage at CLB level, the carry logic associated to each CLB has to be tested as well. Since it is not possible to access the CLB carry input and output ports directly (only through the vertically adjacent CLBs), at least two vertically consecutive CLBs must be released and tested at the same time. To test the SRAM elements of the LUT, each bit is set to both '0' and '1'. By programming the LUTs (four in each CLB) 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 bits in the configuration memory). A total of four test configurations are enough to completely test the combinational part of the CLB. All FFs are tested during these four phases for data input and hold, clock enable, initialise and reverse, and stuck-at faults. Since reconfiguration is slower than vector test 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 four phases. In the first test phase, the initial test configuration has to be set up. In the three subsequent test phases, only a few configuration bits, related to the LUT function and to the programmable multiplexers, are changed. Therefore, test reconfiguration time is smaller. Table 1 details the content of each CLB structural test session.

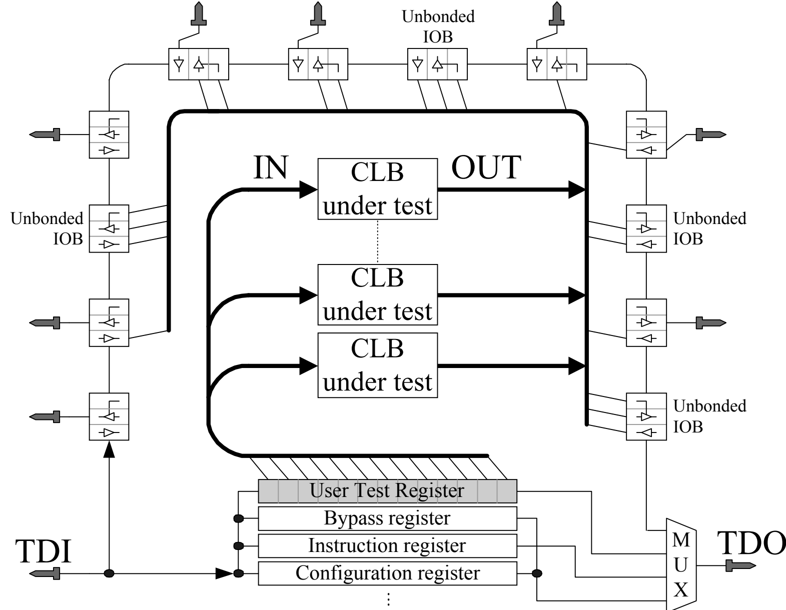

Fig. 4. Test of CLBs through the BS infrastructure.

The test time of each CLB is about 4,2 ms, for a 20 MHz frequency of the test clock, thus the complete replication and test of a single CLB takes about 26,8 ms.

The User Test Register comprises 13 cells, corresponding to the maximum possible number of CLB inputs. The number of CLBs occupied by this register (7) and the two CLBs occupied by the replication aid blocks, associated to the CLB needed to perform the rotation, are the only hardware overhead at FPGA level implied by our test methodology. This accounts for less than 1% of the CLB resources of the Xilinx Virtex XCV200 device (array size = 28x42 CLBs).

Table 1. Number of phases and test vectors in a test session.

1st test phase	18 test vectors
2nd test phase	3 test vectors
3rd test phase	2 test vectors
4th test phase	16 test vectors

Notice also that it is not possible to replicate LUTs configured as RAMs, but nothing prevents the structural testing of the RAM mode of the LUTs. The test of interconnections could also be achieved using the same strategy, with the CLBs under test being replaced by a set of wires under test.

7 Conclusion

A novel approach to the on-line testing of FPGA logic blocks for partial and dynamically reconfigurable SRAM-based FPGAs was proposed in this paper, presenting the following advantages:

1. it is truly non-intrusive and therefore completely transparent at system level;

2. the overhead at chip level is very low;
3. test pattern generation is easy because it is done for a single CLB;
4. fault location is resolved to a single CLB slice;
5. transient fault correction is achieved;
6. the implementation of fault tolerance mechanisms may take place as an extension of the proposed approach;
7. the dependability of reconfigurable hardware platforms is improved.

Our current work focuses on the development of computational tools to introduce a higher degree of automation in the whole process. These tools will read the initial configuration file and automatically generate the partial reconfiguration files needed to perform the replication process.

References

1. F. Hanchek and S. Dutt. Methodologies for Tolerating Cell and Interconnect Faults in FPGAs. *IEEE Trans. on Computers*, Vol. 47, No. 1, pp. 15-33, Jan. 1998.
2. J. Lach, H. W. Mangione-Smith and M. Potkonjak. Low Overhead Fault-Tolerant FPGA Systems. *IEEE Trans. on VLSI*, Vol. 6, No. 2, pp. 212-221, June 1998.
3. N. R. Shnidman, H. Mangione-Smith and M. Potkonjak. On-Line Fault Detection for Bus-Based Field Programmable Gate Arrays. *IEEE Trans. on VLSI Systems*, Vol. 6, No. 4, pp. 656-666, December 1998.
4. W. Huang and E. J. McCluskey. A Memory Coherence Technique for Online Transient Error Recovery of FPGA Configurations. *Proc. of the 9th ACM Int. Symposium on Field-Programmable Gate Arrays*, pp. 183-192, February 2001.
5. IEEE Standard Test Access Port and Boundary Scan Architecture (IEEE Std 1149.1). *IEEE Std. Board*, May 1990.
6. C. Stroud, E. Lee and M. Abramovici. BIST-Based Diagnostic of FPGA Logic Blocks. *Proc. of the International Test Conference*, pp. 539-547, Nov. 1997.
7. C. Stroud, S. Wijesuriya, C. Hamilton and M. Abramovici. Built-In Self-Test of FPGA Interconnect. *Proc. of the International Test Conf.*, pp. 404-411, Nov. 1998.
8. A. Doumar, T. Ohmameuda and H. Ito. Design of an automatic testing for FPGAs. *IEEE European Test Workshop Compendium of Papers*, May 1999.
9. W. K. Huang, F. J. Meyer and F. Lombardi. An approach for detecting multiple faulty FPGA logic blocks. *IEEE Trans. on Computers*, Vol. 49, No. 1, pp. 48-54, Jan. 2000.
10. M. Renovell, J. M. Portal, J. Figueras and Y. Zorian. RAM-Based FPGA's: A Test Approach for the Configurable Logic. *Proc. of the IEEE Int. Conference on Design, Automation and Test in Europe*, pp. 82-88, February 1998.
11. M. Renovell, J. M. Portal, J. Figueras and Y. Zorian. Testing the interconnect of RAM-based FPGAs. *IEEE Design and Test of Computers*, Vol. 15, No. 1, pp. 45-50, January-March 1998.
12. M. Abramovici, C. Stroud, S. Wijesuriya, C. Hamilton and V. Verma. On-Line Testing and Diagnosis of FPGAs with Roving STARs. *Proc. of the 5th IEEE Int. On-Line Testing Workshop*, pp. 2-7, July 1999.
13. M. G. Gericota, G. R. Alves, M. L. Silva and J. M. Ferreira. Dynamic Replication: The Core of a Truly Non-Intrusive SRAM-based FPGA Structural Concurrent Test Methodology. *3nd IEEE Latin-American Test Workshop*, pp. 70-75, February 2002.
14. Politécnico di Torino ITC'99 benchmarks, available at http://www.cad.polito.it/tools/itc99.html