Robustness of Different TMR Granularities in Shared Wishbone Architectures on SRAM FPGA

U. Kretzschmar, A. Astarloa, *Member, IEEE*, J. Lázaro,
Department of Electronics and Telecommunications
University of the Basque Country UPV/EHU
Bilbao, Spain
{uli.kretzschmar, armando.astarloa, jesus.lazaro}@ehu.es

M. Garay, J. Del Ser, *Senior Member, IEEE*
TELECOM Unit
TECNALIA RESEARCH & INNOVATION
Zamudio, Spain
{mikel.garay, javier.delser}@tecnalia.com

***Abstract*—Triple Module Redundancy (TMR) is a popular technique for protecting critical FPGA designs. Although automatic tools for TMR generation mostly use triplication on flip-flop level, designers may opt for different approaches. This work analyses the impact of different granularities on TMR architectures based on a coarse- and a medium-grained TMR implementation of a shared Wishbone interconnection. The actual robustness of these different implementations is measured on a Xilinx Virtex-5 FPGA by using error injection into the configuration bitstream. A specialized test setup comprising two FPGAs boards is introduced so as to allow for the execution of the robustness testing. Based on the coarse-grained architecture, a fine categorization of errors in TMR architectures can be obtained.**

I. Introduction

SRAM-based FPGAs are a widely used basis for applications of low to medium volume magnitude, the reason being their low non-recurring engineering costs, the high achievable performance and the availability of high capacity devices based on this technology. In addition, the possibility of reconfiguring devices in the field provides the means to correct implementation issues or to react to changes such as, for example, new protocols.

Unfortunately, along with these advantages this technology has a high susceptibility to the so-called Single Event Upsets (SEU). A SEU stands for the inversion of a memory bit caused by heavy ions, protons and/or ground level radiation. When a SEU occurs in the configuration memory of a FPGA, the behavior of the device might be altered. This is due to the fact that the configuration memory holds the information of the design currently implemented on the device, including the operational function of the device resources such as Look-Up Tables (LUT), routing and other elements.

To lessen the aforementioned susceptibility, a popular technique for improving the robustness in digital circuits is TMR, which implements the complete design or sub-parts of it multiple times in the same device. This approach has been used in many studies, e.g. [1] and [2], especially for SRAM based FPGAs aimed at improving the overall system reliability of these so-called Commercial-Off-The-Shelf (COTS) devices. In contrast to device shielding, TMR relies on redundancies in the design to avoid wrong outputs, instead of preventing the SEUs from occurring at all. But the importance of TMR is not restricted to mitigating the effects of upsets in the configuration memory. Also permanent errors caused by device ageing can be avoided applying this technology. Furthermore working designs might be also implemented on FPGAs, which are known to undergo production faults. In this way the challenge of producing error-free FPGA devices based on continuously shrinking fabrication processes could be relaxed.

In order to get a highly reliable overall system, the design should have none or as little as possible single points of failure. In the context of robustness analysis, the term *single point of failure* refers to a single piece of logic within a sub-design or routing which, when corrupted, will cause a wrong system behavior. Such single points of failure could be an input sourced by a single FPGA pin, a non-triplicated communication net between many modules inside the FPGA, or a single output pin. The TMR protected shared Wishbone architectures introduced in this paper provide a coarse- and medium-granularity protected communication net, which enables communication between multiple TMR-modules in a design without causing single points of failure.

This paper is structured as follows: first, Section II gives an overview of the TMR technique and different SEU injection methods. The SEU injection method used in this work is next presented in Section III. Section IV follows by introducing two different TMR implementations of a shared Wishbone bus, with a test setup using two Xilinx Development Boards detailed in Section V. Finally, the obtained test results are presented in Section VI, continued by some conclusions in Section VIII.

II. Related Work

The main idea of TMR is to implement a critical design (or sub-parts of it) three times, and to use voters for determining the final output. So, even if there was a fault in one of the three sub designs, the voter would still be fed with two replica of the correct value, so the wrong third result would be overruled by a simple majority voting. If desired, the voter can give additional feedback to the system stating which of the three redundant parts produced a failure. It is important to observe that identifying the erroneous element in case of a voting discrepancy is not possible with Double Module Redundancy (DMR).

In [3] all logic is categorized into four groups, which require different handling in regard to TMR: throughput logic, state-machine logic, I/O logic and so-called special features (e.g.

DLL or DSP48). For all these categories different approaches for implementing TMR must be adopted. For instance, in state-machine logic a voter needs to be implemented to avoid wrong values to be fed back into the different redundant modules. On this purpose, the Xilinx owned tool *TMRTool* supports the automatic generation of triple redundancy.

The finest granularity of TMR is achieved when voting is applied for all registers in the design as done by the *TMRTool* or in [3]. A more coarse granularity can be achieved if the triplication and voting are applied through the device on a basis of bigger modules and their respective outputs. An example of this approach is shown in [1] where a complete microcontroller was triplicated and the voting was applied on its bus signals. Depending on the size of the triplicated modules, the characteristics of the TMR implementation change accordingly. The robustness increases when voting is performed over smaller modules, but so does the implementation cost in terms of FPGA utilization. Furthermore, a fine granularity reduces the highest achievable system speed, because voters are added to the critical path. The relation between TMR granularity and robustness is thoroughly analysed in [4]. Indeed, applying triplication only on critical parts of the logic as proposed in [5] can lead to an interesting trade-off between implementation size and robustness, but this approach is only viable in applications allowing the existence of single points of failures.

The general result of a robustness analysis can be assessed by means of the percentage of *critical* bits in the configuration bitstream, where a configuration bit is considered *critical* if its inversion results in wrong operational behavior of an implemented design. Common robustness measures such as Mean Time Between Failures (MTBU) or Failures In Time (FIT) can be directly derived from this percentage [6]. Besides, different methods of analysing the robustness of a SRAM based FPGA designs against SEU have been proposed in the literature. Since exposing the device under test to actual radiation is an expensive and poorly controllable way of injecting errors, different ways of emulating the effects of SEUs in the configuration memory have been extensively studied instead. One way of emulation is to inject the SEUs at run-time as shown in [7], where the so called SEU controller is used to control the Internal Configuration Access Port (ICAP, described in [8]) of Virtex-5 FPGAs. One or more bits of the configuration memory are flipped during the runtime, whose effects are equivalent to those of a SEU. The benefit of this methodology is that SEUs occur some time after the device startup, just like in SEUs provoked by radiation. However, a drawback of this method is that the SEU controller requires additional resources on the FPGA, which are vulnerable to SEU and consequently, the robustness measurement itself might be distorted.

Another way of injecting SEUs into the configuration memory of a FPGA hinges on manipulating the configuration bitstream prior to the device configuration and the programming of the modified bitstream onto the device. This manipulation can be done on a host computer as in [6] and [9], or by using additional hardware as in [10]. Bitstream manipulation on the PC is easy to implement, it does not require additional resources on the FPGA under test (which avoids adding critical bits not belonging to the tested implementation), and allows evaluating the effects of SEUs in the configuration of the device under test.

For evaluating the correct operation of the device, data considered significant is put out and compared to a reliable source. The source for comparison could be a golden run [9] or a golden device as used in [11].

III. SEU Test Flow

All measurement results for the different TMR implementation robustnesses are obtained using the SEU test flow introduced in [6]. A high level chart of this test flow is given in Figure 1.

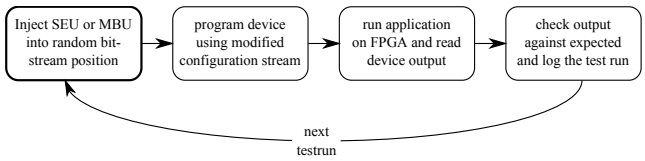


Fig. 1. SEU injection and test-flow.

Its basic idea is to program the device with a configuration bitstream, where a SEU or MBU (Multiple Bit Upset) has been injected beforehand to check if the design is still behaving as expected after programming it on the FPGA. The checking procedure of the devices' correct operation, despite of the presence of a SEU or MBU, is done through iterations, each of which is called testrun or run.

The configuration bitstream generated by the Xilinx `bitgen` generator has a packet-oriented format. When injecting SEUs into this bitstream, only data of the bitstream file which is within the payload of a packet with the destination register FDRI (Virtex-5 devices) may be changed, since this data represents the content of the configuration memory. When generating the bitstream using `bitgen`, it needs to be ensured that the CRC checking is disabled, to avoid CRC errors when programming a modified bitstream.

A UART channel is used for validating the correct functionality of the device. A set of internal registers, representing the applications state, are sent to a PC in order to assess if the application behaves as expected. This expected behavior can be obtained from a golden run, where the device gets programmed without injection of a SEU or MBU. Therefore, this evaluation compares the received data from the internal registers with the expected data from the golden run, comparison that is done for each injected error and the corresponding bit positions. As a consequence, the compared bits are deemed critical if the UART data received differs from that corresponding to the golden run or uncritical, if it was identical.

For robustness measurements it is recommendable to make the design under test fill the FPGA up to a high degree, because unused logic likely results in corresponding configuration bits to be reported as uncritical. It is also important to implement a validation testcase, which utilizes the maximum

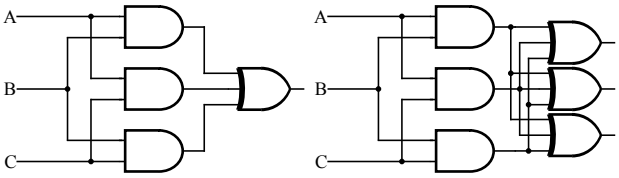
implemented parts and logic of the design. To execute the flow for a given number of SEUs and MBEs without any need of manual interaction all sub-steps need to be automated. This is achieved by using a custom set of PHP scripts for error injection, device programming, behavior validation and testflow management together with the command line tools [12] provided by Xilinx.

IV. Shared Wishbone Implementations of Different Granularity

This work is based on the shared Wishbone interconnection specified in the Wishbone specification [13] and develops two TMR protected versions of this. Both can be used for all Wishbone compliant devices. In this work the test design consists of 64 masters and 64 slaves. Each master is built using the PicoBlaze microcontroller extended by a wrapper providing Wishbone master capabilities. The slaves are implemented as distributed memory within the FPGA.

A. Coarse-grained TMR

A very easy and straight forward way of generating triple module redundancy protected designs is implementing the design three times and adding voters, of type (a) from Figure 2, for all outputs.



(a) final voter (b) intermediate voter

Fig. 2. Voting structure for different granularities.

A block diagram of the coarse-grained TMR architecture is shown in Figure 3. A single voter and three layers of the same logic are implemented, requiring a triplication of the input signals (reset and clock). This triplication results in exactly a triplication in FPGA resource requirements compared to an unprotected architecture when the voting is done externally. Alternatively internal voting for the one output signal needs a very small amount of logic on the FPGA, but it then uses only one I/O pin for the UART output instead of three. Such an internal voting will introduce a single point of failure into the FPGA, which can be avoided by implementing minority voters [3], which require three I/O pins as well.

Besides its simplicity, the coarse-grained TMR is able to eliminate all single points of failure, i.e. single bit errors such as SEUs will not affect the system operation. The drawbacks of the coarse-grained triplication are only visible when single errors accumulate or multiple errors occur. In case of a first error corrupting one of the three layers, any error in the remaining two layers will render a wrong overall device behavior. Due to the big size of each layer, the probability of having this phenomenon results to be rather high.

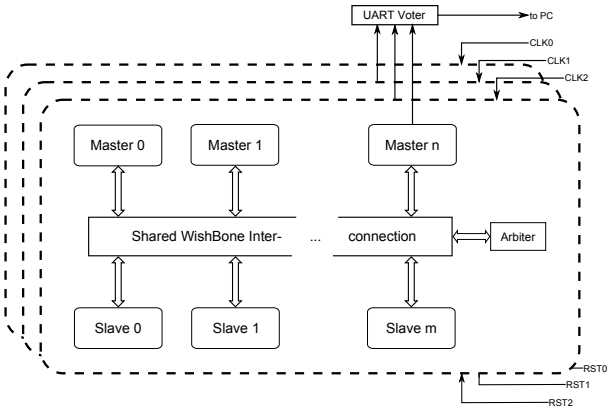


Fig. 3. Coarse-grained TMR for shared Wishbone interconnection.

B. Medium-grained TMR

Choosing smaller modules for the triplication and insertion of intermediate voters has the benefit that errors only affect smaller modules and therefore, a smaller area of the FPGA. Accumulation of errors or multiple errors can be tolerated to a higher extend because the probability of two errors occurring in two redundant instances of the same module is lower as the granularity is made finer.

This work proposes a medium-granularity shared Wishbone interconnection with the intermediate voting scheme shown in Figure 2(b) on the Wishbone bus interfaces. Consequently the modules, which will be triplicated and voted are the masters, the slaves and the Wishbone interconnection structure including the arbiters. As in the coarse-grained approach the structure of Figure 2(a) is used for the final voting of the outputs (e.g. the UART output). This voter can be implemented directly, as a minority or as an external voter with the consequences mentioned above. Figure 4 summarizes the medium-grained TMR architecture.

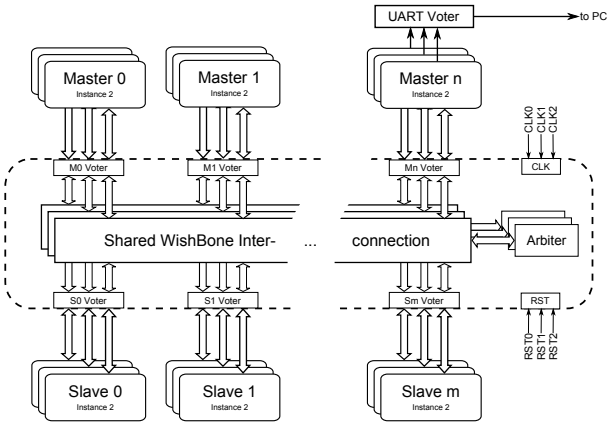


Fig. 4. Medium-grained TMR for shared Wishbone interconnection.

The benefits of the medium- over the coarse-grained approach are illustrated in the following example: if two single errors occur, where the first error affects the *Master 1* and the second affects *Slave 1* in any of the three redundant instances of the medium-grained architecture, the overall system oper-

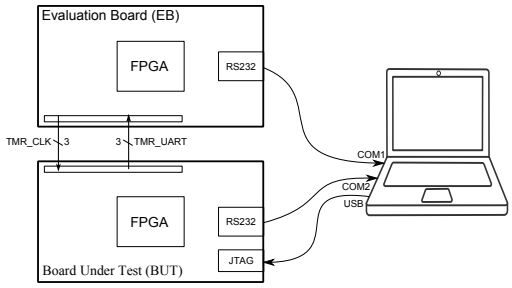


Fig. 5. Test setup using two Xilinx ML507 boards.

ation will be unaffected. Contrarily, the coarse-grained TMR is not able to tolerate this situation in the majority of cases (except if the errors in *Master 1* and *Slave 1* are in the same coarse-grain layer). The benefits of the medium-grained TMR come at a very low cost. Resource-wise it only implements 64 internal voters for masters and slaves more than the coarse-grained TMR, causing only an 1% increase in required Slices. A performance impact in terms of f_{max} is only visible if the bus communication builds part of the on the critical path.

C. Shared Wishbone Application

A test application is run on all masters for testing the behavior of the two proposed TMR shared Wishbone implementations in presence of SEUs and MBU. This application programmes all PicoBlazes to generate traffic on the bus directly after reset and without requiring any further external inputs. All but one of the triplicated PicoBlaze Wishbone masters write to the Wishbone slaves in an interleaved manner using a predefined pattern. The remaining triplicated PicoBlaze master reads the values of the slaves once it has been written, and transmits these values via RS232 to a connected PC. Since this master is triplicated, there are three independent UART signals which can be voted as described before. From the UART output it can be inferred whether the injected SEU or MBU did affect a critical bit of the configuration bitstream.

V. Test Setup for TMR Testing

For execution of the SEU injection flow a specialized test setup specialized for TMR testing was developed, which is summarized Figure 5. A two-board approach was used utilizing common FPGA boards (Xilinx Evaluation Platform ML507 [14]). These provide neither external voting nor multiple inputs of the same clock frequency:

- **Board Under Test (BUT)**: The board under test gets programmed with the SEU affected bitstreams. TMR-clock inputs and the TMR-UART outputs are implemented in a triplicated form. In addition, there is a single UART output driven by an internal voter.
- **Evaluation Board (EB)**: The evaluation board generates the clock signals for the BUT. Besides the clock generation, it contains an external voter for the triplicated UART output of the board BUT.

In order to exclude single points of failures, all the inputs and outputs are triplicated on the board BUT. For the architectures analysed in this paper, this strategy entails the triplication of the reset- and clock-inputs, as well as the triplication of the UART outputs preparing the connection of an external voter. Furthermore, a single UART output with a voted UART signal is available, driven by a voter implemented using the FPGAs logic. By evaluating this output it is possible to determine the robustness of an internal voter. Its signal is connected to the COM2 port of the test computer and the triplicated in- and output ports are connected to the ML507s on-board connector J4.

The EB serves as clock generator for BUT, external voter and it contains additional evaluation logic. The voted UART signal on the EB is connected to the COM1 port of the host PC. By comparing the data on COM1 to the data on COM2, the different reliability levels of the internal and external voting schemes can be identified. Since for the coarse-grain architecture each of the unvoted UART signals represents the correct operation of the corresponding layer, an additional evaluation can be executed based on three counters implemented on the EB corresponding to the three UART inputs of the external voting. Every time one of the three TMR signals differs from the other two, the corresponding counter is incremented 1 unit. A watchdog timer on the EB detects the end of the application on the voted UART signal and appends 16 additional bytes: one header and three 32bit values of the error counters.

Thanks to the two-board approach adopted in this work, it is possible to infer the robustness of an internal voter, the robustness of an external voter and detailed information on which of the internal TMR layers failed in the case of the coarse-grained TMR shared Wishbone architecture.

VI. Test Results

Both shared Wishbone architectures were tested using the SEU injection flow from [6]. The tests consisted of SEUs and MBUs of two to five bits. Injecting SEUs 17500 testruns where executed, for the five types of MBU emulated 12500 testruns were executed each. This accumulates to 80000 testruns per architecture and 160000 overall testruns each providing results for both internal- and external voting.

A. Internal versus External Voting

For all error injections executed in this work, no difference between internal and external voting was found. This is despite of the fact, that the voter implemented in the BUT does cause single points of failure. Any error affecting the logic of the voter, the routing to the used I/O pin or the I/O pin definition itself is potentially able to distort the correct output of the FPGA. But within all the 160000 SEU and MBU injections performed no difference between internal and external voter was observed, i.e. critical configuration bits of the internal voter were never affected. A reason for this result is that both shared Wishbone architectures have only one single-bit output signal (e.g. UART TX). Consequently there is a small number of configuration bits associated to the internal voter, thus the probability of a SEU or MBU being injected into it is extremely low.

TABLE I
TEST RESULTS FOR EXTERNAL AND INTERNAL VOTING

Error type	Testruns	Coarse-grained		Medium-grained	
		Errors	Percent	Errors	Percent
SEU	17500	0	0	36	0.21
2bit-MBU	12500	58	0.46	51	0.41
2bit-MBU*	12500	36	0.29	67	0.54
3bit-MBU	12500	185	1.48	67	0.54
4bit-MBU	12500	337	2.70	103	0.82
5bit-MBU	12500	558	4.46	140	1.12

B. Differences between Coarse- and Medium-grained TMR

The simulation results are summarized in Table I. Rows with an asterisk represent MBUs with sequential bit locations; likewise, rows without an asterisk indicate random error locations. The following observations can be made:

1) Emulation Error due to Optimization in the Design Flow: For the coarse-grained TMR implementation none of the injected SEU caused wrong system behavior, which represents an expected result of any TMR architecture. On the contrary, the medium-grained architecture does show errors in the presence of SEUs. However, these errors are not a consequence of errors in the architecture of the medium-grained TMR implementation, but they are caused by the fact that in this evaluation all architectures are implemented directly in HDL. For the medium-grained implementation the design tools performed an optimization by removing some of the voting logic. This optimization could not be avoided by choosing adequate tool settings, neither was it possible to check all of the voters manually due to the complexity of the chosen architecture (which itself was required to fill a high percentage of the FPGA).

Nevertheless the results of the executed tests disclose important valuable information despite of the removal of a small number of the voters by the design tools. Thereupon, all simulation results obtained can be interpreted as a worst case estimation on the real robustness of the medium-grained architecture. This type of error did not occur for the coarse-grained architecture because it contains one single voter. Summarizing, signals of triplicated masters could not be done by the design tools, because therefore the software of all PicoBlaze masters would need to be analysed.

2) Differences for random error locations: Figure 6 summarizes the test results for random injection. Despite the aforementioned negative impact of the design tools optimization, for MBUs of two bits the medium-grain TMR architecture encompasses a better robustness than its coarse-grained counterpart. This is based on the observation given in [4] where it was shown that for triplication executed on smaller modules, the probability of two errors affecting two triplicated instances reduces as well. For multi-bit errors affecting three bits and more, this advantage of the medium-grained architecture becomes more substantial.

3) Differences for consecutive error locations: The results in Table I remarked with an asterisk represent a special type of two-bit MBUs. For these remarked cases, a position within the configuration bitstream was selected and the corresponding bit and its successor where flipped so as to emulate effects where a high energy particle induces a bitflip in two consecutive configuration bits. This type of error reveals a potential weakness of a medium-grained TMR architecture. When redundant modules are not distributed over the device this type of error is more likely to affect two redundant instances of the same module. On the other hand, the coarse-grain architecture reveals an increased robustness for this case, because two erroneous configuration bits are likely to affect the same TMR layer, leaving two functional layers undergoing a correct system behavior.

The system robustness for both consecutive and randomly injected errors are expected to converge to the same level when distributing the triplicated instances of the medium-grained architecture evenly over the FPGA.

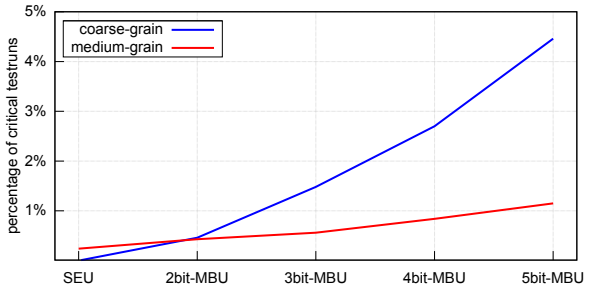


Fig. 6. Injection results for random error positions.

TABLE II
CATEGORIES OF TEST RESULTS

Category	Description	System behavior
A	All TMR layer counters are zero	Correct
B	One TMR layer counter differs from zero	
C	Two or three of the TMR layer counters differ from zero	
D	Two or three of the TMR layer counters differ from zero	Erroneous
E	Two or three of the TMR layer counters are zero	

VII. CATEGORIZATION OF TMR ERRORS

The coarse-grained TMR architecture together with the test setup presented in Section V permits to classify the errors in the TMR architectures into different categories. To this end, counters monitoring the external voting of the UART are implemented on the EB. Each time any of the TMR layers on the BUT differs from the other two, the corresponding error counter on the EB is incremented one unit. This monitoring is executed with a multiple of the BUT's clock frequency. Depending on the overall system behavior and the state of the error counters, the categories introduced in Table II are used for evaluation.

Categories A, B and C represent the cases where a SEU or MBU injection did not provoke a wrong system behavior. In *category A* all errors were injected to uncritical bits, whereas

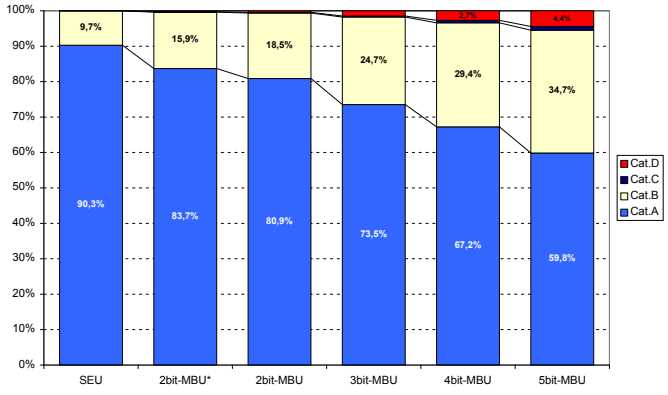


Fig. 7. Categorized results in percentage.

in *category B* the injection only affected a critical bit in one layer and the overall system behavior was guaranteed by the voting. But even when critical bits were hit in two or all three redundant layers as in *category C*, this does not necessarily disturb the overall system behavior. If the errors in each layer articulate themselves at different points in time but for each discrete instant there are at least two correct working layers, the overall system operation will remain correct. A concrete example for testrun belonging to *category C* is produced when in one layer there has been an error with master 1 and in another layer there has been an error with master 14, which obviously did write at different times to the bus.

The testruns resulting in a wrong system behavior belong either to *category D*, where the injected errors affected at least two TMR layers and caused the voting to fail, or *category E*. The latter shows no errors in the error counters, but surprisingly undergoes a wrong system behavior. This situation may occur if bitflips are injected in a way that the same errors occur in all three layers. Henceforth, the voter is fed with three times the same signal and the error counter mechanism is not triggered. This category is only mentioned for the sake of completeness, because only a negligibly small percentage of the testruns resulted to be classified as such.

Figure 7 depicts the test results for the different categories and the coarse-grained TMR architecture. It is important to note that with a growing number of injected SEUs, the percentage of testruns belonging to *category A* decreases quickly. Since *category A* represents the cases where no critical bits where affected by an error injection, the results for this category can be regarded as a measure of the robustness of unprotected architectures, despite it being pessimistic due to the extra hardware needed for the triplication. Furthermore, a comparison of the percentage of testruns belonging to *category A* to the percentage of testruns with correct system behavior reveals that TMR is not only able to completely prevent SEUs from negatively impacting on the system behavior, but also offers a strong mitigation of the effects of multiple bit errors. This mitigation is stronger if the granularity of the TMR is finer, as measured in the previous Section.

VIII. Conclusion

This work presents two TMR protected architectures of different granularity levels for a shared Wishbone interconnection. Their robustness is analysed by using error injection into the configuration bitstream of the FPGA. The performed analysis elucidates the positive impact of a finer granularity on the robustness of the design. By injecting errors not only on random locations but also in sequential order, the importance of the distribution of triplicated modules on the FPGA is underlined. A performed test set-up based on two FPGAs allows for the categorization of TMR errors, which sheds further light on the operational behavior of the TMR strategy.

Acknowledgements

This work was carried out in the R&D Unit UFI11/16 of the UPV/EHU, and supported by the Ministerio de Ciencia e Innovacion of Spain within the projects TEC2011-28250-C02-01/2, and by the Basque Governments Department of Education, Universities and Research within the research fund of the Basque university system IT394-10.

References

[1] Y. Ichinomiya, S. Tanoue, M. Amagasaki, M. Iida, M. Kuga, and T. Sueyoshi, "Improving the Robustness of a Softcore Processor against SEUs by Using TMR and Partial Reconfiguration," in *2010 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, May 2010, pp. 47 –54.

[2] L. Sterpone and M. Violante, "Analysis of the robustness of the TMR architecture in SRAM-based FPGAs," *IEEE Transactions on Nuclear Science*, vol. 52, no. 5, pp. 1545–1549, October 2005.

[3] C. Carmichael, "Triple Module Redundancy Design Techniques for Virtex FPGAs," Xilinx Application Notes XAPP197, http://www.xilinx.com, July 2006.

[4] X. Wang, "Partitioning Triple Modular Redundancy for Single Event Upset Mitigation in FPGA," in *2010 International Conference on E-Product E-Service and E-Entertainment (ICEEE)*, November 2010, pp. 1–4.

[5] B. Pratt, M. Caffrey, P. Graham, K. Morgan, and M. Wirthlin, "Improving FPGA Design Robustness with Partial TMR," in *IEEE International Reliability Physics Symposium*, March 2006, pp. 226–232.

[6] U. Kretzschmar, A. Astarloa, J. Lazaro, J. Jimenez, and A. Zuloaga, "An Automatic Experimental Set-up for Robustness Analysis of Designs Implemented on SRAM FPGAs," in *2011 International Symposium on System on Chip (SoC)*, October-November 2011, pp. 96–101.

[7] K. Chapman, "Virtex-5 SEU Critical Bit Information Extending the capability of the Virtex-5 SEU Controller," Xilinx Documentation SEU lounge, http://www.xilinx.com.

[8] Xilinx Corp., "Xilinx-5 FPGA Configuration User Guide," Xilinx Documentation UG191, http://www.xilinx.com, 2010.

[9] Baker, Z. K. and Monson, J.S., "Fault Injection Into SRAM-Based FPGA For The Analysis Of SEU Effects," in *IEEE International Conference on FieId-Programmable Technology (FPT)*, 2003.

[10] M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, A. Marmo, S. Pastore, and G. Sechi, "A Tool for Injecting SEU-like Faults into the Configuration Control Mechanism of Xilinx Virtex FPGAs," in *18th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, November 2003, pp. 71–78.

[11] P. Bernardi, M. Reorda, L. Sterpone, and M. Violante, "On the Evaluation of SEU Sensitiveness in SRAM-based FPGAs," in *10th IEEE International On-Line Testing Symposium*, July 2004, pp. 115–120.

[12] Xilinx Corp., "Command Line Tools User Guide," Xilinx Documentation UG628, http://www.xilinx.com, 2010.

[13] Silicore Corporation, "Wishbone System-on-Chip (SoC) Interconnection Architecture for Portable IP Cores Revision: B.3," http://www.opencores.org, September 2002.

[14] Xilinx Corp., "ML505/ML506/ML507 Evaluation Platform User Guide," Xilinx Documentation UG347, http://www.xilinx.com, 2009.