I. Villalta, U. Bidarte, G. Santos, A. Matallana and J. Jiménez, "*Fault injection system for SEU emulation in Zynq SoCs,*" **Design of Circuits and Integrated Systems, Madrid, Spain, 2014**, pp. 1-6, doi: 10.1109/DCIS.2014.7035579.

Fault Injection System for SEU Emulation in Zynq SoCs

Igor Villalta, Unai Bidarte, Gorka Santos, Asier Matallana, Jaime Jiménez
Department of Electronics And Telecommunications
University of the Basque Country UPV/EHU
Bilbao, Spain

Abstract— This paper presents a fault injection method for SEU (Single Event Upset) emulation in FPGAs based on loading at the programmable logic a configuration file with an erroneous bit. A "Xilinx Zynq®-7000 All Programmable SoC" device has been used to implement it, which combines a hard microprocessor (Processing System PS) with Programmable Logic (PL). The emulation tool is fully implemented on the Zynq chip, which means that neither additional external equipment nor PCB modifications are needed. Communications to external devices that slow down the configuration process are avoided, so a high fault-injection rate is achieved. Previous works consider including fault injection circuitry at the PL. This circuitry can be affected by a faulty configuration file, leading the device to an unrecoverable state, which is named as "injection side effects". In the method proposed in this paper the injection is implemented in the processing system of the Zynq device, making the injection system independent to the programmable logic and avoiding the previously mentioned effect. This method allows using complete bitstreams, partial bitstreams and one-frame bitstreams to inject faults. A comparison is done so as to find the most appropriate bitstream type.

Keywords—SEU, FPGA, emulation; fault injection; fault tolerance; ZYNQ;

I. INTRODUCTION

Modern manufacturing technologies are leading to smaller device sizes. As a result, chips performing more complex functions and operating at a higher frequency are being developed. However, they are becoming more and more vulnerable to radiation-induced SEUs. A radiation-induced SEU happens when a high-energy particle hits the semiconductor storing an erroneous value at a memory cell [1]. The device-size reduction makes more likely for these particles to hit inverse-polarized p-n junctions, which are the most sensitive parts of electronic devices.

According to [2] a fault is a physical phenomenon that leads to an error. And an error is an incorrect part of the system that can lead to a failure, which means that the equipment does not deliver correctly the service it has been designed for. Electronic devices are widely used in systems called safety critical, where a failure can lead to environmental damage, injury or death. So measures have to be taken when a radiation-induced SEU (fault) leads to a bit flip (error) that may provoke a malfunction (failure), which is not allowed in any way.

SRAM based FPGAs are more vulnerable to SEUs than other semiconductor devices. In [1] it is mentioned that SEU events have a greater impact in SRAM cells than in DRAM or in Flash cells. In [3] the architecture of a SRAM based FPGA is presented and the effects produced by radiation are analyzed. The key element of a FPGA is the configuration memory, where the information about hardware resources is stored. If a critical bit of this memory flips, the implemented design's functionality changes and a failure is provoked.

The aim of SEU emulation systems is to quantify the amount of critical bits of a design. Device manufacturers as Xilinx publish a reliability report [5], where it is specified the failure rate per configuration memory Mb. So when the amount of critical bits in the configuration memory is known, the failure rate is immediately calculated.

In [4] a method for critical bit estimation for standard designs is provided by Xilinx. Here it is mentioned that FPGA designs have a criticality ratio between 1% and 11%, as a result, it can be assumed that no more than 11% of the configuration bits are critical, even when the device is almost full. So according to [4] and Xilinx reliability report [5], where it is mentioned that SEU is 86FIT/Mb for 7series FPGAs at sea level, the failure rate of a ZYNQ7020 device, which has 32Mb configuration bits, would be:

32Mb x 86FIT/Mb x 0.11= 302.72 FIT

(1 FIT = 1 failure / 10^9 hours)

The effect of radiation-induced SEU at sea level is quite low, when the device is used at another altitude for avionics or aerospace application the FIT value has to be multiplied by a derating factor as it is explained in [10]. At 40000ft the FIT value has to be multiplied by 561.70 [10], so the FIT value is:

302.72FIT x 561.70 = 170037.82FIT

However, when analyzing redundant fault tolerant systems this is not suitable. Only a few bits corresponding to majority voters or routing resources are critical in this case. So a fault injection environment is needed to evaluate the fault tolerance of these systems.

II. BITSTREAM FILE

A file named bitstream is used to configure Xilinx FPGAs. Here configuration and initial values of different elements of the FPGA are stored. Configuration values define the implemented hardware and remain unmodified at configuration memory during execution. Initial values are loaded into memory elements while configuration and are modified during operation.

Most of the bits of a bitstream configure the routing resources and the CLBs (Configurable Logic Blocks), which are the key elements of FPGAs.

The bitstream also contains initial values of flip-flops. If the LUT is used as distributed RAM or shift register, bits belonging to it are also initial value bits and not configuration bits.

Besides, the bitstream contains configuration information of other elements as BRAMs, IOBs, DSP48 or clock resources (BUFGP, MCMM, PLL...).

In figure 1 the structure of different bitstream files is represented. Xilinx bitstream file begins with a set of 32-bit commands. Different kinds of bitstream commands are documented at [11]. The file begins with a synchronization sequence, which is followed by a set of commands depending on the bitstream type. After these commands are sent the configuration data frames are loaded.

The most common bitstream is the complete bitstream. It is generated by Xilinx development tools following the normal design flow and it contains configuration and initial value data for all the elements of the programmable device. Complete bitstream commands contain timing, synchronization, security and encryption. Then the configuration data is loaded and CRC check is done in order to assure bitstream integrity. If it is correct a command is sent in order to start the device and the file ends with a DESYNC command, which ends the configuration interface transfer.

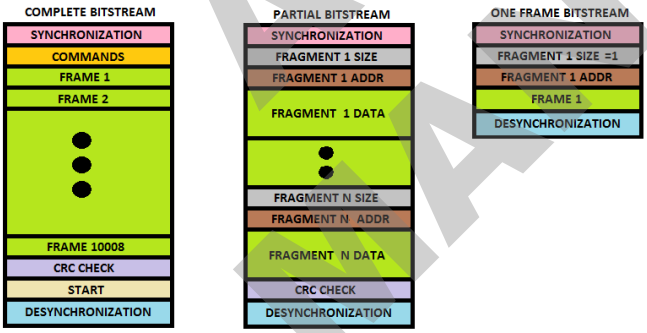

Fig. 1. Structure of different types of bitstream files

Another bitstream type is the partial bitstream. The purpose of this file is to reconfigure only a part of the device while other parts work normally, which is named DPR (Dynamic Partial Reconfiguration). It is generated by following the partial reconfiguration design flow of Xilinx design tools. As it is thought to be loaded while the FPGA is working it does not contain the start order. The number of words loaded into the configuration memory has to be specified as well as the address of the programmable logic where these data has to be stored. It is organized in fragments and the size and address are specified for each fragment. In the end a CRC check is performed in order to check the integrity of the file.

The time needed for a bitstream to be loaded is proportional to its size. As a result, partial bitstreams are loaded in a much shorter time than complete ones.

Since configuration data is structured in frames, the optimum bitstream for fast fault injection is the one-frame bitstream. There are two ways to generate them, using the SEU Controller [7] or sending a read order to the configuration memory [9].

Fault injection in FPGAs is based on reprogramming the device with a bitstream that has an erroneous bit. Once it has been reprogrammed a test vector is sent to circuit's inputs and the device operation is observed. If a malfunction appears the flipped bit is considered to be critical. If a bit of the bitstream has been artificially modified, CRC check would not match. Thus, it must be disabled. The result of the check would be incorrect and the new data would never be stored.

The content of sequential elements such as flip-flops, distributed RAM, LUT implemented shift registers, DSP registers and BRAMs should always be taken as critical. However, the first step after having reconfigured the device and before the verification process starts, the system has to be taken into a known state, which means that values of sequential elements are overwritten (more information at section 4.B.4 of this article). So the effect of having modified a bit containing the initial value of a sequential element is cancelled and the bit is wrongly considered as no critical. The amount of initialization bits of BRAMs, distributed RAM and flip-flops can be easily counted by the designer and they should be added to overall critical bits.

Modifying bits belonging to commands has to be avoided as well. These are not going to be stored at the configuration memory, and they are not going to result affected by a radiation-induced SEU.

III. BACKGROUND ON FPGA SEU INJECTION METHODS

A. Internal Fault Injection

An internal subsystem decides which bit is flipped and controls the interface. As no cables to other boards or equipments are needed the injection rate is high. Up to now an internal configuration port implemented in the programmable logic has been used. In devices from Xilinx, a provider of programmable technologies and devices, it is called ICAP (Integrated Configuration Access Port). However, if a configuration bit belonging to the logic controlling the internal configuration port is flipped, the

injection subsystem can be damaged and the system stops working properly. Errors produced by this phenomenon are called injection side effects [6].

In [8] a system based on partial reconfiguration is presented. The unit under test is placed at a reconfigurable region while the injection subsystem is located at the static region. This does not avoid injection side effects because static part can be routed across reconfigurable partitions. The time needed to reconfigure the logic with this partial bitstream depends on its size, but overall it is notably faster than loading a complete bitstream.

When the unit under test is placed at a bounded region smaller than the whole device, the routing tool has fewer resources to carry the signals. This means that the final routing is more complex and signals have to go across more interconnection points, leading to a higher error rate. A SEU emulation platform should be as close to the final design as possible, and unluckily there is a great difference between an implementation occupying the whole FPGA and the same circuit implemented in a smaller bounded region.

Xilinx has an IPcore called SEM (soft error mitigation controller) [7] that is used for SEU mitigation. It can be used for SEU emulation, although it is not its main functionality. After the flipped bit being addressed, the SEM performs a readback of the needed frame, flips the desired bit and partially reconfigures the FPGA. A one-frame partial bitstream needs very little time to be programmed. Nevertheless, the SEM has also influence on injection side effects.

B. *External Fault Injection Methods*

The flipped bit is decided by an entity out of the FPGA device, and it is programmed through an external interface, such as JTAG or SelectMAP . The main advantage of external fault injection system is the fact that the emulation platform stays always unmodified and it is totally independent to the implemented design. The difficulty lies in managing the interface. JTAG, which is used in [9] is slow. And SelectMAP, although it can be fast, it is not easy to cope with. It usually requires a CPLD controlling it, which means that the PCB has to be modified. Note that all the systems mentioned [6][7][8][9] and the proposed in this paper are general methods that can operate on any board .

C. *Internal-External fault injection*

In [6] a mixed internal-external approach is proposed. The SEU controller is used, which is the previous version of the SEM [7]. And to minimize the effects of self damage an external reconfiguration is periodically done. In this way the high rate of internal systems is mixed with the independence of external systems.

D. *Specific Board Based Fault Injection*

In [14][15] fault injection platforms based on a specific board are presented. They achieve high injection performance and they avoid injection side-effects. The design has to be placed and routed for these platforms to perform the test. After having performed the test, the design has to be placed and routed again for the product board. So the obtained test results don't belong to the implementation that would operate under SEU conditions.

IV. PRESENTED METHOD

A. *System Architecture*

A Xilinx ZYNQ-7000 All Programmable SoC device is chosen to implement this SEU emulation system. This system-on-chip combines a hard dual core ARM9 with a FPGA. To configure the programmable logic a hard interface controlled by the processing system is provided, which is called PCAP (Processor Configuration Access Port). In figure 2 a general scheme of the system is shown. Note that the UUT (Unit Under Test) is placed in the PL, and the fault injection system is set at the non-reprogrammable part of the ZYNQ device. The system has been developed at a ZedBoard, which is a standard board containing a ZYNQ7020 device.

The UUT is placed at the PL. Data transfer between PS (Processing System) and PL (Programmable Logic) is EMIO (Extendable Multiplexed I/O) signals using the GPIO peripheral of the ARM system.

The bitstream manipulation software is placed at the PS, and faulty bitstreams are loaded to PL through PCAP (Processor Configuration Access Port), which is a hardware implemented configuration interface. As it is hard implemented, it is never modified by faulty bitstream carrying the system to an unrecoverable state. A SD card is used to store the correct bitstream generated by Xilinx tools.

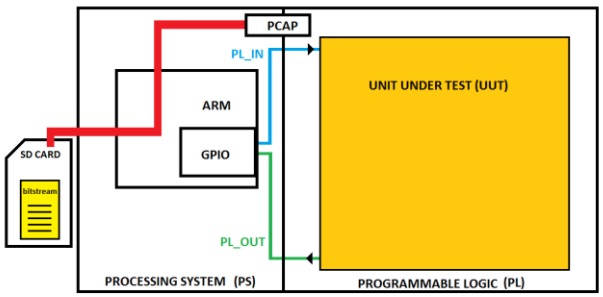

Fig. 2. System architecture

B. *Fault Injection Flow*

First of all a bitstream is obtained, then it is modified to emulate a SEU, afterwards the device is reconfigured, then verification is done and finally the injected fault is repaired. The test continues by modifying a new bit at the present bitstream. When all of the bits of the bitstream are analyzed a new bitstream is obtained (only if partial reconfiguration has been chosen). If all possible partial bitstreams have been analyzed the test is ended. In figure 3 the fault injection flow diagram is presented.

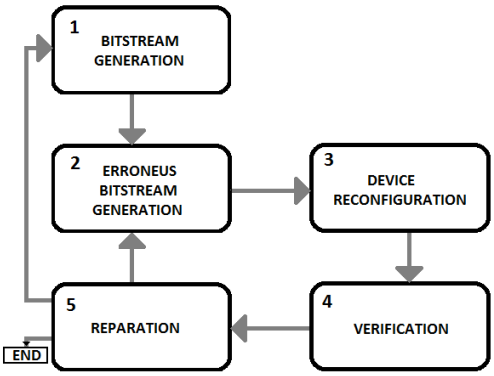

Fig. 3. Flow diagram

1) Bitstream Generation

First of all the bitstream has to be obtained. As it has been mentioned before, there are 3 types of fault injection bitstreams; the complete bitstream, the partial bitstream and the one-frame bitstream.

If a complete reconfiguration is chosen, the bitstream file is picked directly from the Xilinx BitGen tool. To generate a partial bitstream is not so immediate if the unit under test has not been placed at a bounded region in the programmable logic. After the address and the size of the bitstream has been specified, it has to be fulfilled with the content, which is stored in the complete bitstream. The addres has to be specified in a Xilinx format, and it is no way to convert from this format to linear address of the complete bitstream.

This can be solved by brute-force attack, generating partial bitstreams and complete bitstreams containing those partials. Then the content of the partial is searched in the complete bitstream and in this way the position can be found.

Once the position is found the content of the complete bitstream at that point is copied in the partial one.

It is not possible to generate one-frame bitstreams at Xilinx Tools. It has been mentioned before that there are two ways to generate them, using the SEM IPcore or reading the content from the configuration memory. The first option has been discarded because it is implemented in the PL and injection side-effects are present.

So the one-frame bitstream generation process starts with a readback order. Then the interface copies the content of the configuration memory at the processor's memory. Finally bitstream headers are created.

2) Erronoeous Bitstream Generation

Once the bitstream has been generated the value of a bit is changed emulating a SEU. CRC check has to be disabled. This action is done always after having repaired the previous injected error, unless there are no more bits remaining at the present bitstream. When all of the bits of the present bitstream have been tested a new bitstream is generated.

3) Device reconfiguration

This is one of the most critical part of the system in terms of time performance. The PCAP is a 32bit 100MHz interface, so the time needed to load the full bitstream of ZYNQ7020, which has 4MB is:

4MB *(1s/100E6 cycles)*(1cycle/4B) = 10ms

However, in [12] it is mentioned that due to software overheads and DMA transfer time the overall time needed to load the whole file is 32ms. So the time needed to analyze the 4MB*8 bits of the full bitstream when complete reconfiguration is used is:

4MB* (8bits/Byte) * 32ms = 1024000s = 284h

In [12] it is also mentioned that the time needed to load a partial bitstream of 134kB is 1060µs, so the time needed to analyze the 4MB of the complete bitstream is:

4MB * (8bits/Byte) * 1060µs = 33920s = 9.42h

Doing an extrapolation from these data it can be concluded that the time needed for loading a 1 frame bitstream of 404 Bytes is 3.18 µs. So the time needed to analyze 4MB of configuration bits is:

4MB * (8bits/Byte) * 3.18µs = 101s

So there is no doubt that using 1 frame bitstream is optimum in terms of time performance.

While a partial bitstream is being configured uncontrolled outputs may be generated. In order to not disrupt the static logic the DPR flow [13] requires the reconfigurable partition to be placed at a bounded region independent from the static part. It is also defined that some glue logic should be placed between dynamic and static parts so as to create a safe interface between them.

In this method there is not a bounded region for the dynamic part isolated from the static part. If a partial bitstream for fault injection is loaded, a region of the design is reconfigured while the rest continues working. During reconfiguration, dynamic part can provide uncontrolled outputs to the surrounding logic provoking damage out of the reconfigurable region. These effects will be named in this document as "DPR side-effects".

4) Verification

The objective of this part is to decide if a functional error has occurred after having reconfigured the device with a faulty bitstream. It is also very sensitive to overall testing time. An input is sent to the PL and after having waited the time the hardware needs to perform the operation the output of the PL is read and compared to the golden value.

Before sending any input it is recommendable to carry the unit under test to a known state by overwriting register and memory contents. The easiest way to do this is to apply a reset. However, sequential circuits not containing a reset are becoming more and more typical nowadays. For example, in [16] it is recommended to use resets unless they are totally necessary. In this case it is more complex to return to the initial state. One idea is to apply a set of inputs that would carry the values of sequential elements to a known state. This can be complex and can require long initialization vectors.

The other possibility is to apply complete bitstreams to inject faults. In this case the sequential elements are initialized while a fault is being injected.

5) Reparation

After having verified a faulty configuration the system has to return to a known state. This step is not necessary when complete reconfiguration has been chosen for fault injection. A complete reconfiguration loads initial values of all the elements of the device bringing it to a known state.

When a faulty bitstream is loaded, the erroneous bit can provoke a fault in a region out of the influence area of the loaded bitstream (DPR side-effects). This means that if the device is reconfigured again with the correct partial bitstream that error would not be solved, and next injections would be taken as critical bits even when they are not.

In order to mitigate the effects of DPR side-effects, the partial bitstream without the erroneous bit is loaded. A test is done after reparation, and if the error remains a complete reconfiguration is done.

C. Comparing to Other fault Injection Systems

In the following table a comparison between injection and verification subsystems is presented. Note that the presented method can test an unbounded design. It has high-speed internal injection having solved the issue of injection side effects.

TABLE I. INJECTION SUBSYSTEM COMPARISON

Methods	Fault injection subsystem			
	Injection type	***Dynamic part***	***Bitstream type***	***Injection side effects***
Xilinx SEM	Internal	Whole FPGA	1 Frame	Yes
[6]	Internal & External	Whole FPGA	1 Frame	Yes but mitigated with external reconfiguration
[8]	Internal	Bounded region	Partial bitstream	Yes
[9]	External	Bounded region	1 Frame	No
[11] [12]	Specific board	Whole FPGA	-	No
Proposed	Internal	Whole FPGA	Complete/ Partial /One frame	No

V. RESULTS

The objective of this document, in addition to presenting the method, is to compare the different bitstream types find out which is the most appropriate to do the fault injections. A 1500 adder chain has been implemented as UUT at the PL, occupying the 70% of the device. After a faulty bitstream is loaded a vector of 30 elements is sent to the UUT and the result is compared to the golden value. All the sequential circuits of the design have a reset, so there is no trouble at carrying the system to the initial state.

Then reparation is applied, (if complete bitstream has not been used) and verification is performed. If the error persists it is classified as type_2 error (and requires a complete reconfiguration). If the reparation removes the error the modified bit is classified as type_1.

A. One column test

One CLB column of the PL has been tested using the three mentioned bitstream types; one-frame, partial and complete. The used device has 3 rows of 72 column and one CLB column contains 50 CLBs. The same test has been done three times for each bitstream type. The result is never the same. It is not possible to know which element is addressed by each bit of the configuration file. As a result it is not possible to find out why some bits are sometimes classified as critical and sometimes no. However, it can be concluded from the following table that the accuracy of the method is above 99%. In brackets the amount of type_2 errors is represented, which are only present when one-frame bitstream is used.

TABLE II. ONE COLUMN TEST

Injection type	*Injections*	*Test1 errors*	*Test2 errors*	*Test3 errors*
Complete	116352	10873 (0)	10846 (0)	10860 (0)
Partial	116352	10891 (0)	10888 (0)	10886 (0)
One frame	116352	10896 (227)	10914 (233)	10891 (228)

Then the position of the first 1964 errors has been monitored for the three bitstream types. It has been observed that 2 errors that are classified as critical at the complete bitstream are not detected at the partial and one-frame bitstreams. 3 errors detected at the one-frame bitstream are not detected at the others. 9 errors are detected at the partial bistream and at the one-frame bitstream, and not at the complete. The other 1950 errors are the same for the three bitstream types. From this it can be concluded that the accuracy of the method is above 99%.

All the bits classified as type_2 errors at the one-frame bitstream are detected as type_1 errors at the other bitstream types.

B. Critical Bit calculation

Finally, the amount of critical bits has been measured by testing all the bits for the different kind of bitstream types. This is done in order to evaluate the coverage level of the injection part. The complete bitstream has 32349088 bits, so 32349088 different fault injections can be done. The partial bitstream method allows doing 18645408 injections. If injections are done at block RAM the number is higher. Injection in block RAMs has been discarded because they are not used by the unit under test and errors at that point are always 0. Something similar happens with the one-frame bitstream method. This method allows (without block RAM) 17782464 injections. The FIT value is given for 40000ft

Even when fault injection is done at block RAM, the maximum possible fault injection number does not reach the 32349088 bits of the complete bitstream. This happens because Xilinx tools don't allow creating partial bitstreams containing IO blocks, clocking or other special configurable resources. This is not a big problem because failures at IO blocks can't be detected in internal verification approaches and the other resources are usually not used in normal designs.

When complete bitstream has been used only the 25% of the total bits have been tested in order to cut down the test time. 15 days are needed to analyze the 100% of the bits of the complete bitstream. The obtained result is multiplied by 4 to obtain the amount of critical bits of the device. This is the result of an extrapolation, thus, it is not very exact.

TABLE III. CRITICAL BITS CALCULATION

Injection type	*Injections*	*Errors*	*Time*	*FIT*
Complete	8087272	346027	76h	66860
Partial	18645408	1512430	19h	73059
One frame	17782464	1470651	8h	71041

VI. CONCLUSIONS

A fast and accurate method for SEU characterization has been presented. It is fast because the injection and the verification are done internally Since the final implemented design can be tested an accurate result is provided. The three presented bitstream types give an accurate result, (above 99%). Complete bitstream is more appropriate when sequential elements without reset are used. When it is easy to carry the sequential elements of the system to a known state, a fast injection test based on one-frame bitstreams can be done.

VII. 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, by the UPV/EHU within the project US13/13 and by the Basque Governments Department of Education, Universities and Research within the research fund of the Basque university system IT394-10.

VIII. REFERENCES

[1] Robert C. Baumann, Radiation-Induced Soft Errors in Advanced Semiconductor Technologies, IEEE Transactions on Device and Materials Reliability, Volume 5, no. 3, pp 305-316

[2] Avizienis, A, Laprie, J.C, Dependable Computing: From Concepts to Design Diversity, Proceedings of the IEEE, Volume 74, Issue 5, pp 629-638

[3] Luca Sterpone, Electronics System Design Techniques for Safety Critical Applications, Lecture Notes in Electrical Engineering, Volume 26

[4] Ken Chapman, Virtex-5 SEU Critical Bit Information Extending the capability of the Virtex-5 SEU controller

[5] Device Reliability Report, Third Quarter 2013, UG116 (v9.6) November 19, 2011

[6] Uli Kretzschmar, Armando Astarloa, Member, IEEE, Jesús Lázaro, Jaime Jiménez, Aitzol Zuloaga, An Automatic Experimental Set-Up for Robustness Analysis of Designs Implemented on SRAM FPGAs, 2011 International Symposium on System on Chip (SoC), pp. 96 - 101Xilinx, "User guide: Logicoretm ip soft error mitigation controllerv1.1. " UG764 March 1, 2011

[7] L. Sterpone and M. Violante A new partial reconfiguration-based fault-injection system to evaluate SEU effects in SRAM-based FPGAs, IEEE Transactions on Nuclear Science, Volume 54 , Issue: 4, pp. 965 - 970

[8] M. Straka, J. Kastil, and Z. Kotasek, SEU Simulation Framework for Xilinx FPGA: First Step towards Testing Fault Tolerant Systems, 14th Euromicro Conference on Digital System Design (DSD), 2011, pp. 223 – 230

[9] Xilinx, NSEU Mitigation in Avionics Applications XAPP1073 (v1.0) May 17, 2010

[10] Xilinx, User guide: 7 Series FPGAs Configuration UG470 (v1.7) Octover 33, 2013

[11] Cristian Kohn, Partial Reconfiguration of a Hardware Accelerator on Zynq-7000 All Programable SoC Devices XAPP1159 (v1.0) January 21, 2013

[12] Xilinx, Vivado Design Suite User Guide, Partial Reconfiguration UG909 (v2013.3) October 30, 2013

[13] M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, S. Pastore, G.R. Sechi, R. Weigand Evaluation of Single Event Upset Mitigation using the FLIPERR Faut Injection Platform, 22nd IEEE International Symposium on Defect and Fault Tolerance inVLSI Systems pp.105,113, 26-28 Sept. 2007

[14] J. Nápoles, H. Guzmán, M. Aguirre, J.N. Tombs, F. Muñoz, V. Baena, A. Torralba, L.G. Franquelo, Radiation Environment Emulation for VLSI Designs: A Low Cost Platform based on Xilinx FPGA's Industrial Electronics, 2007. ISIE 2007. IEEE International Symposium on, pp.3334,3338, 4-7 June 2007

[15] Xilinx, UltraFast Design Methodology Guide for the Vivado Design Suite, UG949 (v2014.1) April 2, 2014