FLIPPER

Executive Summary

ESA/ESTEC Contract n. 18559/04/NL/LvH/gm

MITIGATION OF SEUs AFFECTING CONFIGURATION MEMORY AND RECONFIGURATION LOGIC IN VIRTEX II FPGAs

DOCUMENT CHANGE RECORD

Issue	Date	Paragraphs affected	Change information
1.0	29/09/08	All	New document
2.0	27/01/09	5., 5.1-5.5 (new), 7.1, 7.3 (new)	Major changes

Prepared by
Monica Alderighi, IASF – INAF, Milano.

1 TABLE OF CONTENTS

2 INTRODUCTION

Field Programmable Gate Arrays based on SRAM (SRAM-FPGAs) have gained a primary role in several application areas due to their high density and unlimited on-field re-configuration capability. Nevertheless when used in high reliability applications and specifically space applications, the Single Event Effects (SEEs) have to be addressed [1] [2]. Single Event Upsets (SEU) are of particular concern, because in SRAM-FPGA they affect not only flip-flops and RAM blocks of the user design, but also the device configuration memory, they can therefore change the logical function of the circuit. Many mitigation techniques are available to designers for dealing with SEU, among them Triple Modular Redundancy (TMR) is a well known approach to design hardening [3] [4]. Xilinx provides a tool for automatic implementation of a TMR scheme (XTMR) suitable for their SRAM-FPGA devices, called TMRTool [9]. To prevent multiple error accumulation, TMR should be used in conjunction with a periodic configuration memory refresh, also referred to as scrubbing [24][25].

For complexity reasons (XTMR increases the FPGA resource usage by a factor > 3), mitigation may have to be applied only on part of the design, or different mitigation schemes are mixed. Prediction of the resulting overall SEU sensitivity is then difficult, and ground radiation testing is often an unavoidable step. Fault injection techniques like the hardware fault emulation based on partial re-configuration [5], or the laser fault injection [6] are usually cheaper to implement, achieve a higher upset rate, and they are more deterministic with respect to the SEU locations; they can be applied to prepare or to complement radiation tests. The circuit robustness can also be assessed by static analysis, based on the topological analysis of the placed circuit [7].

In the frame of the contract n. 18559/04/NL/LvH/gm "MITIGATION OF SEUs AFFECTING CONFIGURATION MEMORY AND RECONFIGURATION LOGIC IN VIRTEX II FPGAs" a fault injection system has been developed for Xilinx Virtex II devices in order to evaluate mitigation techniques offering different protection degrees. The tool has been applied to two sample designs. In this document results and analyses are reported together with a brief description of the tool.

Section 3 gives an overview of related works, in section 4 the FLIPPER fault injection system is described.
In section 5 the results concerning the injection activity into the reconfiguration logic are presented while in sections 6 and 7 the two case studies are reported. Conclusions are drawn in section 8.

3 RELATED WORKS

Various systems have been reported in the literature for the evaluation of SEU fault tolerance of VLSI circuits for hi-rel/space applications. Such systems check the response of a circuit in presence of faults by comparing the behavior of the fault free circuit and the faulty one for a given set of stimuli or test patterns.
Besides radiation testing in e.g. particle accelerator facilities, SEU injection can also be performed by simulation [10] [11] [12] and laser. The first two methods are relatively slow due to either a limited upset rate, or a reduced device operation speed respectively. Laser sources offer the advantages of a greater availability and lower cost than particle accelerators, while being more deterministic with respect to the SEU locations. However, techniques are needed to bypass the difficulties due to the increasing metallization present in devices [13].

FPGA emulation based fault injection has proven effective to reduce fault evaluation time in this context, because it allows a high upset rate as well as a high speed operation of the design under test. FPGA fault injection can be used to evaluate the SEU sensitivity of an ASIC design, injection is then performed in flip-flops or memories of the target design only, by either reconfiguring the device in order to present the faulty behavior [15] [16] or modifying the original circuit adding extra hardware to modify the state of the circuit [17] [18]. These approaches do not consider the FPGA as the target technology.

In case SRAM-FPGA themselves are employed in avionic or space applications, SEU injection must be done also in the device configuration memory and not in flip-flops only. As the configuration memory is not modeled in the netlist of the target design, logic simulation cannot be used. But SEU fault injection by partial reconfiguration can be exploited to evaluate the degree of protection of mitigation techniques, or to get a rough evaluation of circuit behaviors prior to radiation testing.

A configuration bitstream SEU emulator for SRAM-FPGA was described in [19]. The goal of the simulator was to provide a map of the sensitive bits of the device configuration memory. A prototypal system aimed at studying the SEU sensitivity of the reconfiguration logic of Xilinx Virtex devices was described in [20]. A similar system was used for radiation test activity for the same devices reported in [21].

4 FLIPPER FAULT INJECTION SYSTEM

FLIPPER is a system that was mainly conceived to inject bit-flip faults within the internal memory of FPGA's. It consists of a hardware platform and a software application running on a PC. The hardware platform is actually a flexible FPGA-based general purpose board that can be employed in other SEE evaluation activities (namely, ground radiation test), and also in more general applications as a testbed equipment or a digital I/O board [8].

The main FLIPPER board (Figure 1a) manages the overall fault injection by means of a Xilinx Virtex 2 Pro (XC2VP20) device. This control board contains 128 MByte of either SDRAM or DDR memory and 16 MByte Flash, and communicates to a PC via a USB 2.0 port controlled by a dedicated microcontroller. It also has two 240 pin connectors plus one 60 pin connector for the communication of test data and control data to/from a Device Under Test (DUT). Two further P160 standard connectors allow to use standard expansion boards with the Flipper control board.

The piggy-back style DUT board (Figure 1b) contains the Device Under Test. The DUT device used in the experiment described in this document is a Xilinx Virtex 2 FPGA (XQR2V6000). Up to 416 signals are driven by the main board's FPGA, corresponding to a maximum of 416 triplets (1240 DUT I/Os) in case of XTMR-ed designs. Virtually any component can be used as a DUT device for functional test and radiation test, provided that some constraints are met as far as pin number and device accessibility are met.

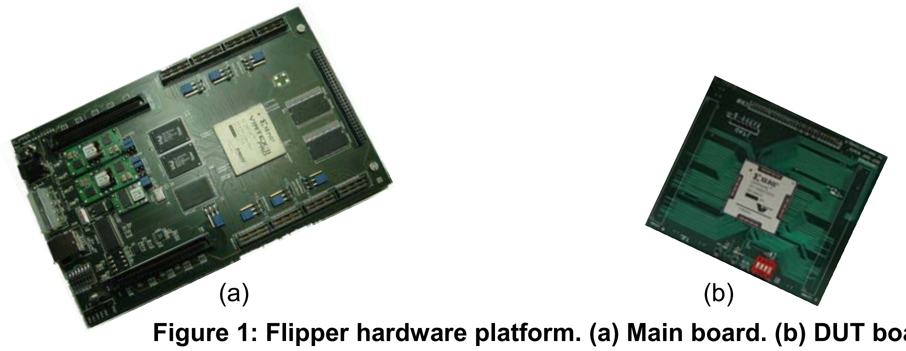
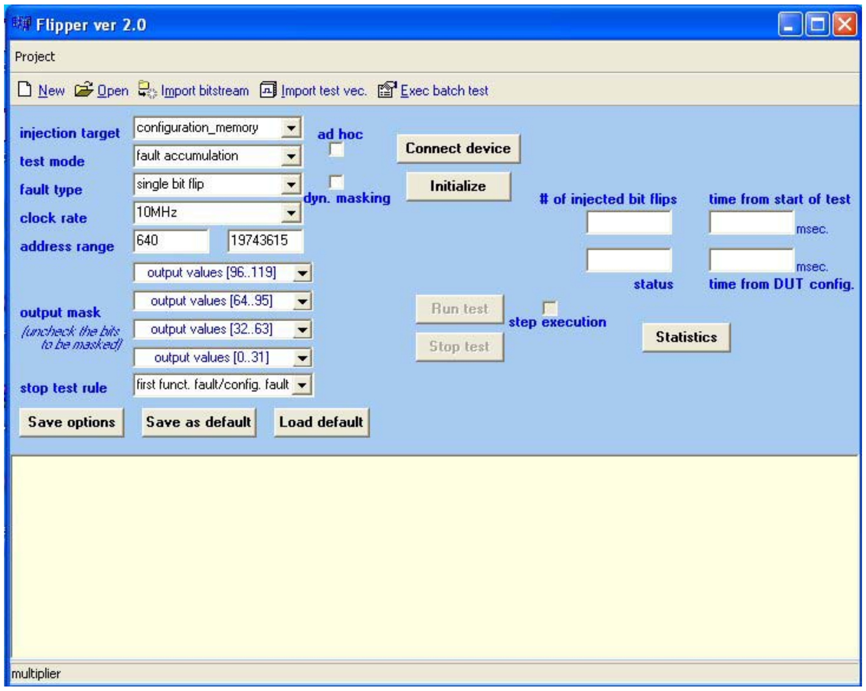


Figure 1: Flipper hardware platform. (a) Main board. (b) DUT board

Fault injection in FPGA devices is based on frame modification and active partial re-configuration. Both single bit and multiple bit upset [26] can be injected. The injection experiment is run by means of a software application that was specifically developed for the FLIPPER system. Figure 2 shows snapshot of the GUI interface for such application.



injection target: configuration_memory
test mode: fault accumulation
fault type: single bit flip
clock rate: 10MHz
address range: 640 19743615
output mask *(uncheck the bits to be masked)*: output values [96..119]; output values [64..95]; output values [32..63]; output values [0..31]
stop test rule: first funct. fault/config. fault
multiplier

Figure 2: FLIPPER GUI interface

By means of this application it is possible to set up the experiment options, and in particular:

i) the target of injection,
ii) the test mode (internal memory bit may be randomly addressed and faults may accumulate until a functional failure occurs, or the bits may be addressed sequentially, one at a time),
iii) the fault type (either single bit flip or multiple bit flip),
iv) the DUT clock rate
v) the address range of memory bits that are involved in the current experiment.

Within the application it is possible to import input and output values from an HDL simulation, at each clock edge. These data will be used as test vectors and reference (gold) values during the fault injection experiment. Test stimuli vectors may be up to 150 bit wide, and gold output vectors may be up to 120 bit wide.

Test vectors and gold vectors are stored in the on board RAM, before the injection procedure begins, to allow a high speed functional test. The DUT is exercised with the whole set of test vectors anytime a bit flip is injected to verify the functional influence of such a fault on the device. Actual outputs from the DUT are compared on board with corresponding gold values to perform this functional verification. When a mismatch is detected, a fault packet is sent to the PC including all the information relevant to the system behaviour. Once the test is run, it proceeds until the test stop condition (set by the software) is met.

Compared to other solutions, the main assets of FLIPPER are:

- Piggy-back DUT board allowing an easy exchange of the DUT devices and hence keeping the test system up to date with the evolution of FPGA technology
- Memory based test pattern handling ensures a smooth flow to derive stimuli and expected outputs from a simulation testbench. The entire target FPGA can be used for the target design, it is neither necessary to duplicate the target design, nor to have stimuli generators implemented in hardware on the FPGA.

5 RECONFIGURATION LOGIC

5.1 Experiment

In order to study SEU sensitivity of the logic devoted to device reconfiguration (reconfiguration logic), the fault injection is performed by executing a write operation to the address of the target register. This is the way to modify the content of a given register, and the only mean to emulate a register upset through a bitstream manipulation-based injection.

The injection procedure operates modifying the content of writable registers. Table 1 lists the whole set of reconfiguration logic registers[1], while the CMD command register codes are given in Table 2[2].

Register Name	Read	Write	Address	Description	Notes
CRC	Y	Y	00000	CRC register	Stores the CRC value embedded into the input bitstream.
FAR	Y		00001	Frame address register	Sets the starting frame address for the next configuration data write cycle.
FDRI	N	Y	00010	Frame data input register (write configuration data)	Contains inputs to the SRAM cell module.
FDRO	Y	N	00011	Frame data output register (readback configuration data)	Contains read-back data from the configuration memory.
CMD	Y	Y	00100	Command register	Contains the instruction for the built-in FSM.
CTL	Y	Y	00101	Control register	Controls internal functions as, for instance, the status of DONE and pins after configuration.
MASK	Y	Y	00110	Masking register for CTL	Contains the mask for write operation to the CTL register; a “1” in bit N of the mask allows that bit position to be written in the CTL register.
STAT	Y	N	00111	Status register	Indicates the value of numerous global signals.
LOUT	N	Y	01000	Legacy output register (DOUT for daisy chain)	Is used to rite data to the DOUT pin when configuring downstream devices in a serial configuration daisy-chain.
COR	Y	Y	01001	Configuration option register	Loads user selected options.
MFWR	N	Y	01010	Multiple frame write register	Is used when the bitstream compression option is enabled in DitGen.
FLR	Y	Y	01011	Frame length register	Indicates the frame size.
KEY	N	Y	01100	Initial key address register	Indicates the number of DES keys that are used for bitstream encryption.
CBC	N	Y	01101	Initial CBC value register	Stores the DES encryption keys.
IDCODE	Y	Y	01110	Device ID register	Contains the device-specific identification code.

Table 1: Virtex II Configuration Registers

[1] “Virtex-II Platform FPGA User Guide", UG002, section "Configuration Details", pp 314-341
[2] Virtex-II Platform FPGA User Guide", UG002, section "Configuration Details", pp 322-323

Command	Code	Description
WCFG	0001	Write Configuration Data. Used prior to writing configuration data to the FDRI.
MFWR	0010	Multiple Frame Write Register. Used to perform a write of a single frame to multiple frame addresses.
LFRM	0011	Last Frame. Deasserts the GHIGH_B signal, activating all inter terconnect. The GHIGH_B signal is asserted with the AGHIGH command.
RCFG	0100	Read Configuration Data. Used prior to reading configuration data from the FDRO.
START	0101	Begin Startup Sequence. Initiates the startup sequence. The startup sequence begins after a successful CRC check and a DESYNC command are performed.
RCAP	0110	Reset Capture. Resets the CAPTURE signal after performing readback-capture in single-shot mode.
RCRC	0111	Reset CRC. Resets the CRC register
AGHIGH	1000	Assert GHIGH_B Signal. Places all interconnect in a high-Z state to prevent contention when writing new configuration data. This command is only used during shutdown reconfiguration and readback. Interconnect is reactivated with the LFRM command.
SWITCH	1001	Switch CCLK Frequency. Updates the frequency of the Master CCLK to the value specified by the OSCFSEL bits in the COR.
GRESTORE	1010	Pulse the GRESTORE Signal. Sets/resets (depending on user configuration) IOB and CLB flip-flops.
SHUTDOWN	1011	Begin Shutdown Sequence. Initiates the shutdown sequence, disabling the device when finished. Shutdown activates on the next successful CRC check or RCRC instruction (typically an RCRC instruction).
GCAPTURE	1100	Pulse GCAPTURE. Loads the capture cells with the current register states.
DESYNCH	1101	Reset DALIGN Signal. Used at the end of configuration to desynchronize the device. After desynchronization, all values on the configuration data pins are ignored.

Table 2: Command Register Codes

The control signals, INIT, BUSY, and DONE are monitored during device (re)configuration and their value compared to the expected one. In case of a mismatching, the error codes shown in Table 3 are reported.

Phase	Error Code	Description
	0	No error
Memory Clear	1	INIT is not driven low by the FPGA at the beginning of the memory clear phase
	2	INIT is not driven high by the FPGA when the memory clear phase is completed
Bitstream Load	6	BUSY is asserted while the bitstream is being loaded
	7	INIT gets low while the bitstream is being loaded
	8	DONE is released during active reconfiguration

Table 3: Configuration Errors

In the following sections, the results of two injection campaigns, namely systematic and random, are illustrated.

5.2 Results of systematic injection campaign

The goal of this test is to classify the effects of single event upsets in VIRTEX II configuration logic registers. The initial condition is that the DUT XQR2V6000 has been programmed with a design. SEU and MBUs are injected into registers content only in terms of bit-flips in the value actually loaded.

One single injection at a time is performed, after which its effects are evaluated. Then, the register content is restored. In this test, the effects of an injection do not sum up to the effects of the previous ones.

The test mode is sequential. For each register, every bit/couple of bits is accessed and modified in sequential order.

- *CRC.* The register contains the value against which the calculated CRC value is compared. No anomalous behaviour was observed after SEU or MBU injection.

- *CTL*. The Control Register is used to set the configuration security level, the persist setting, and to toggle the Global Three-State signal. Writes to the Control Register are masked by the value in the MASK Register, which allows the GTS_USR_B signal to be toggled without re-specifying the Security and Persist bits. Anomalous behaviours are observed after SEU or MBU injection. The error is the same in both cases is a BUSY error. The explanation is that, an SEU in the persist bit, which is the only unmasked bit within the generated bitstream, disables the SelectMAP port; any further attempt to use this port results in a BUSY error. In case of MBUs the persist bit is affected twice, so the error is observed twice.

- *FAR*. The register addresses the configuration frames. Writes to the FDRI and reads from the FDRO act on the address specified in the FAR. Each time the FAR is updated with a new value, the command in the Command Register (CMD) is executed. No anomalous behaviour was observed after SEU or MBU injection.

- *FDRI*. Configuration data frames are written to memory by shifting frame data into the Frame Data Input Register. Data written to the FDRI is placed in the configuration memory address specified by the FAR. For how injection is performed, SEUs are injected in the first DWORD of a configuration data frame. We have not detected configuration errors and anomalous behaviours of configuration signals. Functional errors have not occurred. No anomalous behaviour was observed after SEU or MBU injection.

- *CMD*. The Command Register is used to instruct the configuration control logic to strobe global signals and perform other configuration functions. Commands are executed immediately after being written to the CMD register. An anomalous behaviour is observed after an MBU injection, resulting in a DONE error. Actually, the DESYNCH command (“1101”), which is the last executed command, turns to SHUTDOWN (“1011”), as an injection effect, and this prevents reconfiguration.

- *MASK*. The Mask Register masks writes into the Control Register. A '1' in any bit within the MASK register allows the corresponding bit position to be written in the Control Register. The default value of the MASK Register is all '0's. No anomalous behaviour was observed after SEU or MBU injection.

- *LOUT*. The Legacy Output Register is used to write data to the DOUT pin when configuring downstream devices in a serial configuration daisy-chain. The LOUT register is not available for SelectMAP and JTAG configuration modes. No anomalous behaviour was observed after SEU or MBU injection.

- *COR*. The Configuration Options Register is used to set certain configuration options for the device. Anomalous behaviours are observed after SEU or MBU injection. An SEU in one reserved bit de-asserts the DONE signal, yet the configuration is preserved, as it is re-asserted in the reconfiguration phase.

- *MFWR*. The Multiple Frame Write Register is used when the bitstream compression option is enabled in BitGen (-g Compress:yes). When more than one frame has identical data, the frame data can be loaded once into the Multiple Frame Write Register then copied into multiple memory address locations. In some cases this can decrease the size of the bitstream considerably. No anomalous behaviour was observed after SEU or MBU injection.

- *FLR*. The Frame Length Register (FLR) stores the device frame length, and must be written to before any FDRI or FDRO operation can be performed. No anomalous behaviour was observed after SEU or MBU injection.
- *KEY*. The Initial Key Address Register (KEY) indicates the number of DES keys that are used for bitstream encryption. This is a write-only register; there is no way to read this register through any configuration interface or user logic resources. No anomalous behaviour was observed after SEU or MBU injection.
- *CBC.* The Cipher Block Chaining Register stores the DES encryption keys. This is a write-only register; there is no way to read the keys out of the device through any configuration interface or user logic resources. No anomalous behaviour was observed after SEU or MBU injection.
- *IDCODE*. The Device ID (IDCODE) Register contains the device-specific identification code. This register is separate from the JTAG IDCODE register although it uses the same 32-bit identification code. The purpose of this register is to ensure that the bitstream has been created for the correct device. No anomalous behaviour was observed after SEU or MBU injection.

5.3 Results of random injection campaign

The goal of this test is to evaluate configuration failures due to accumulated SEUs in configuration control registers. Actually, only the value of CTL, CRC, LOUT, MFWR, CBC, and KEY registers is progressively modified without restoring their content, as all other registers are restored by the reconfiguration operation.

As for the previous campaign, the initial condition is a configured XQR2V6000, the test proceeds by performing one single injection at a time, and only the register content is bit-flipped by an injection. The test mode is random. Bit locations in registers are randomly addressed.

After the last functional test, the process continues with the next injection until the active reconfiguration is no longer possible or the predefined number of injections is reached. The active reconfiguration always swaps between the two configuration bitstreams involved.

Three runs were performed, increasing the number of injected SEUs from 500 to 10000. Results are summarized in Table 4, Table 5, and Table 6.
Generally speaking, the behavior is similar to the systematic case: no errors are observed until CTL and CMD registers are altered.
Indeed, most observed errors are BUSY errors, due to an SEU in the persist bit of the CTL register, which disables the SelectMAP port and thus configuration. In two cases an SEU affecting the CMD register turns it to an unknown command (“1111”) and a SWITCH respectively, generating a DONE error as a configuration error.

# Run	# Injections	Configuration Signals			Configuration Error	Functional Error		Notes
		INIT_B	BUSY	DONE		After injection	After re-config.	
0	188	1	0	0	8	PASSED	NON PASSED	CMD (1111)
1	378	1	1	1	6	PASSED	NON PASSED	CTL PERSIST
2	3	1	1	1	6	PASSED	NON PASSED	CTL PERSIST
3	254	1	0	0	8	PASSED	NON PASSED	CMD SWITCH
4	214	1	1	1	6	PASSED	NON PASSED	CTL PERSIST

Table 4: SEU random injection (500 max)

# Run	# Injections	Configuration Signals			Configuration Error	Functional Error		Notes
		INIT_B	BUSY	DONE		After injection	After re-config.	
0	71	1	1	1	6	PASSED	NON PASSED	CTL PERSIST
1	657	1	1	1	6	PASSED	NON PASSED	CTL PERSIST
2	54	1	1	1	6	PASSED	NON PASSED	CTL PERSIST
3	254	1	1	1	6	PASSED	NON PASSED	CTL PERSIST
4	425	1	1	1	6	PASSED	NON PASSED	CTL PERSIST

Table 5: SEU random injection (1000 max)

# Run	# Injections	Configuration Signals			Configuration Error	Functional Error		Notes
		INIT_B	BUSY	DONE		After injection	After re-config.	
0	879	1	1	1	6	PASSED	NON PASSED	CTL PERSIST
1	131	1	1	1	6	PASSED	NON PASSED	CTL PERSIST
2	553	1	1	1	6	PASSED	NON PASSED	CTL PERSIST
3	115	1	1	1	6	PASSED	NON PASSED	CTL PERSIST
4	184	1	1	1	6	PASSED	NON PASSED	CTL PERSIST

Table 6: SEU random injection (10000 max)

5.4 Results of accumulate & clean-up campaign

The goal of this test is to evaluate configuration failures due to accumulated SEUs in all configuration control registers between two successive reconfigurations. As for the previous campaigns, the initial condition is a configured XQR2V6000, the test then proceeds by performing a predefined number of injections, and only the register content is bit-flipped by an injection. Bit locations in registers are randomly addressed. Differently from the previous campaigns, no reconfiguration operation is performed in the iterative loop, meaning that all register values are progressively modified without restoring their content.

After the last functional test, the injection is re-iterated until the predefined number of injections is reached, or injection is no longer possible. The procedure continues with the functional test, a reconfiguration attempt by using the same bitstream, and a further functional test.

Three runs were performed, increasing the number of injected SEUs from 500 to 10000. Results are summarized in Table 7, Table 8, and Table 9.
No configuration errors were observed, except a BUSY error due to an SEU in the persist bit of the CTL register. Yet, in a few cases the injection de-asserted the DONE signal which was re-asserted in the final configuration.

# Run	# Injections	Configuration Signals			Configuration Error	Functional Error		Notes
		INIT_B	BUSY	DONE		After injection	After re-config.	
0	500	1	0	1	0	PASSED	PASSED	-
1	15	1	0	1[3]	0	PASSED	PASSED	COR
2	447	1	0	1*	0	PASSED	PASSED	COR
3	500	1	0	1	0	PASSED	PASSED	-
4	500	1	0	1	0	PASSED	PASSED	-

Table 7: SEU Accumulate & Clean (500 max)

[3] The DONE signal is de-asserted after injection, thus inhibiting further injections. The signal is re-asserted in the reconfiguration phase.

# Run	# Injections	Configuration Signals			Configuration Error	Functional Error		Notes
		INIT_B	BUSY	DONE		After injection	After re-config.	
0	1000	1	0	1	0	PASSED	PASSED	-
1	1000	1	0	1	0	PASSED	PASSED	-
2	354	1	0	1*	0	PASSED	PASSED	COR
3	1000	1	0	1	0	PASSED	PASSED	-
4	1000	1	0	1	0	PASSED	PASSED	-

Table 8: SEU Accumulate & Clean (1000 max)

# Run	# Injections	Configuration Signals			Configuration Error	Functional Error		Notes
		INIT_B	BUSY	DONE		After injection	After re-config.	
0	10000	1	0	1	0	PASSED	PASSED	-
1	587	1	1	1	6	PASSED	PASSED	-
2	1063	1	0	1*	0	PASSED	PASSED	COR
3	10000	1	0	1	0	PASSED	PASSED	-
4	143	1	1	1	6	PASSED	PASSED	CTL

Table 9: SEU Accumulate & Clean (10000 max)

5.5 Result analysis

The fault injection procedure based on register write operation may induce internal state transitions that do not correspond to real effects of particle ionization. This is not dependent on the specific system implemented, rather it represents a limit in performing such analysis by using artificially emulated faults. This suggests not to rely completely on FLIPPER to analyze SEU effects on the configuration logic.

However, inducing those transitions at each injection leads to two important considerations. First, the number of critical bits (thus the failure probability) is very low, with only one case (CTL register) requiring a total device configuration. A normal scrubbing recovers from most part of injected faults. Based on on-orbit failure rate that can be found in [23], this seems to encourage the use of such devices in critical applications, as far as their configuration registers are concerned.

6 CASE STUDY 1: CUC-CTM

6.1 Experiment

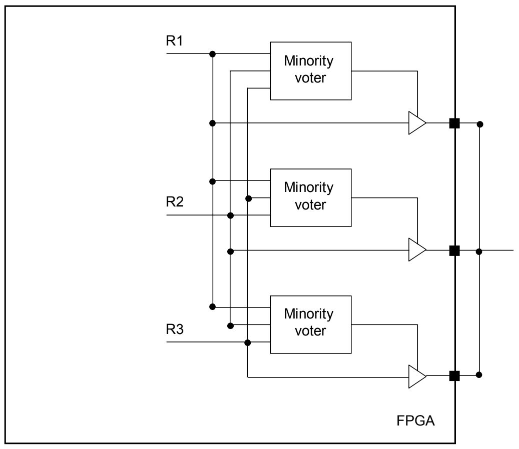
In this case study an analysis of a sample design using FLIPPER is presented. The sample design is a CUC-CTM VHDL IP-core from the European Space Agency (ESA) that provides datation and alarm services for space applications. The original (plain) version of the design is analysed along with the XTMR protected design, the latter in two variants depending on the design output protection scheme: double or triple voted [9].

Sample Design

CUC-CTM stands for CCSDS Unsegmented Code and CCSDS Time Manager (CCSDS = Consultative Committee for Space Data Systems). The CUC-CTM is a synthesisable VHDL IP-CORE from ESA [22]. The CUC contains a frequency synthesiser and an elapsed time counter, and the CTM implements datation (time tagging) and alarm services.

All services provided by the CTM are accessible via two AMBA APB slave interfaces, through which standard Spacecraft Time Source Packets according to the ESA Packet Telemetry Standard are available. There are several discrete output signals carrying the alarm or pulse-per-second (PPS), and input signals which allow time tagging of external events.

The netlist obtained by the XST synthesis tool from the CUC-CTM source code has been processed by the TMRTool from Xilinx. TMRTool offers different triplication options and output schemes. For this study the triplication has been applied to the entire design by using the XTMR “Standard” triplication. It means that all internal logic (combinatorial and sequential) is triplicated, and majority voters are inserted in all the state machine feedback paths. Two different output schemes have been implemented: triple and double minority voted output. They are quite similar, in both cases, the three redundant replicas (R1, R2, R3) of an output signal participate to the voting scheme, and the output signal that differs from the others is put in high impedance by a minority voter. In the double voted output scheme only two out of three outputs of the replicas are brought to the device pad as shown in the right part of Figure 3.



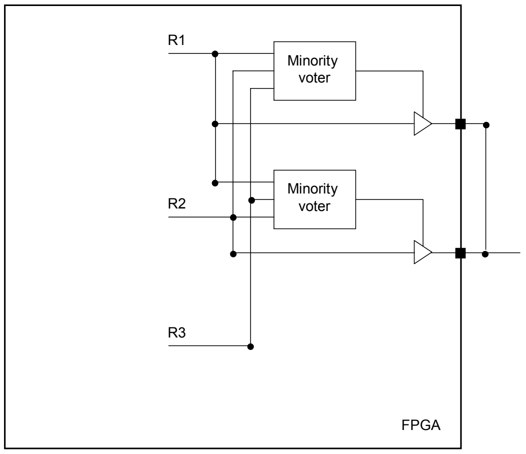


Figure 3: Triple and Double Voted Output Schemes

This allows saving one third of the output pads, but the drawback concerning the lower level of protection has to be assessed. The device resource usage is reported in Table 10. It has to be noticed that the double voted scheme, for the CUC-CTM, saves 73 out of 569 output pads.

Resource	Plain	XTMR triple voted output	XTMR double voted output
Slice FF	785 out of 67584	2361 out of 67584	2361 out of 67584
LUT	1789 out of 67584	7167 out of 67584	7094 out of 67584
IOB	212 out of 824	569 out of 824	496 out of 824
GCLK	1 out of 16	3 out of 16	3 out of 16

Table 10: XQR2V6000 resource usage for CUC-CTM (two versions)

The VHDL test-bench provided by ESA together with the CUC-CTM source code, has been converted into stimuli for the fault injection campaigns. The test-bench is structured into several procedures, each of them verifies a specific function: after the reset verification, the access to both AMBA APB interface is tried, then the frequency synthesiser and time counter are configured. The verification of high level functions is performed afterwards. Among these functions there are time sample, time correlation, alarm and periodic pulse generation, interrupt, and time packet read out.

Procedure
A fault injection campaign is usually composed by many runs as to obtain statistically relevant data. The goal of testing all configuration bits and all possible combination bits, however appears unreachable because of performance limitations. For each of the two XTMR design variants and for the plain design a fault injection campaign has been performed. In both cases the XQR2V6000 [24] is initially configured and, after checking the configuration signals (INIT, BUSY and DONE), the whole set of stimuli is applied for verifying the experimental set up and the correct design behaviour.
FLIPPER then injects an SEU by active partial re-configuration into a randomly chosen configuration memory location and applies the stimuli.
In each run, this procedure iterates, successively accumulating bit flips (SEU) in the configuration memory, until a functional fault is detected, i.e. the values on one or more output signals deviate from the expected outputs (golden pattern). The fault signature is logged to the workstation, the user can analyse how many and which bits of the configuration bitstream have been upset, and which outputs of the design have failed at which cycle of the stimuli.

6.2 Results

In the plain version, 1000 injection runs were accomplished, for a total of about 3×10^5 SEU injected. For the triple voted output (TVO) and double voted output (DVO) versions the injection runs were 2500 and 1000 corresponding to 2×10^6 and 1×10^6 SEU injected respectively. From the result file produced by FLIPPER a few graphs have been extracted, showing the probability of observing an output error as a function of accumulated SEUs. The distribution and the cumulative distribution of the number of injections to the first functional fault, for the plain and both design variants, are shown in Figure 4 and Figure 5 respectively, where the results are binned in class width of 25.

The frequency is computed as the ratio between the numbers of accumulated SEUs to the output error in the class and the total number of injected SEUs. The cumulative distribution is the integral of the distribution (adding the actual frequency in a class to the frequencies of the previous ones); it indicates the probability of a functional failure to occur after a given number of configuration bit upsets. The mean value of the distribution of the number of injections to the first functional fault is 337 for the plain design, 1330 for the triple voted output version, and 1,152 for the double voted output. The corresponding highest output error probabilities (8%-2.27%-2.87%) are after 25, 1,000, and 775 injections respectively (Figure 4).

A comparison of the three cumulative frequency distribution diagrams is given in Figure 5. It clearly shows the two design variants behave similarly (mach better than plain) and the triple voted output scheme gives only a little improvement. This is quite satisfactory as one third of the output pins could be saved in this design.

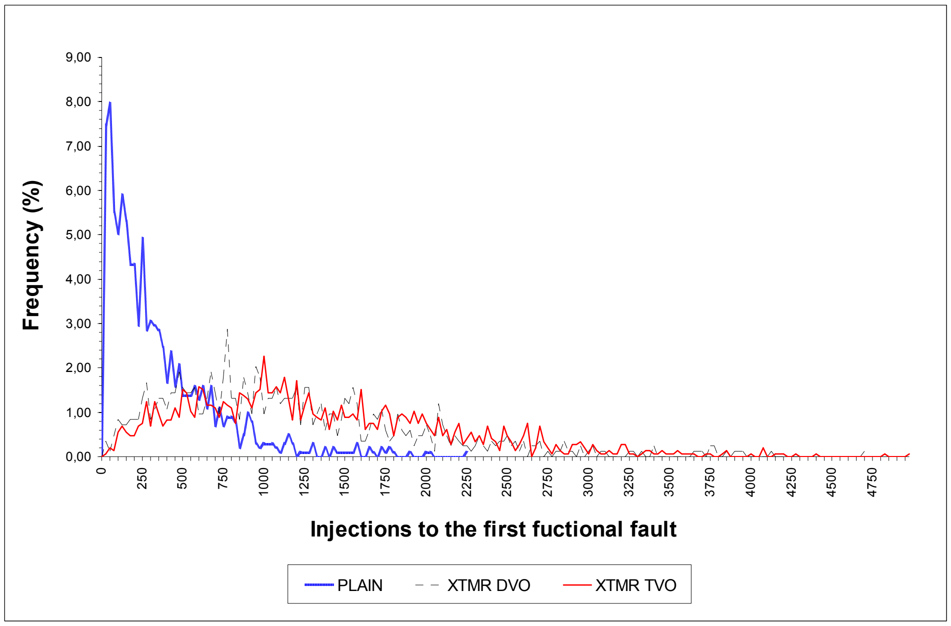


Figure 4: Distribution of the number of injections to the first functional fault

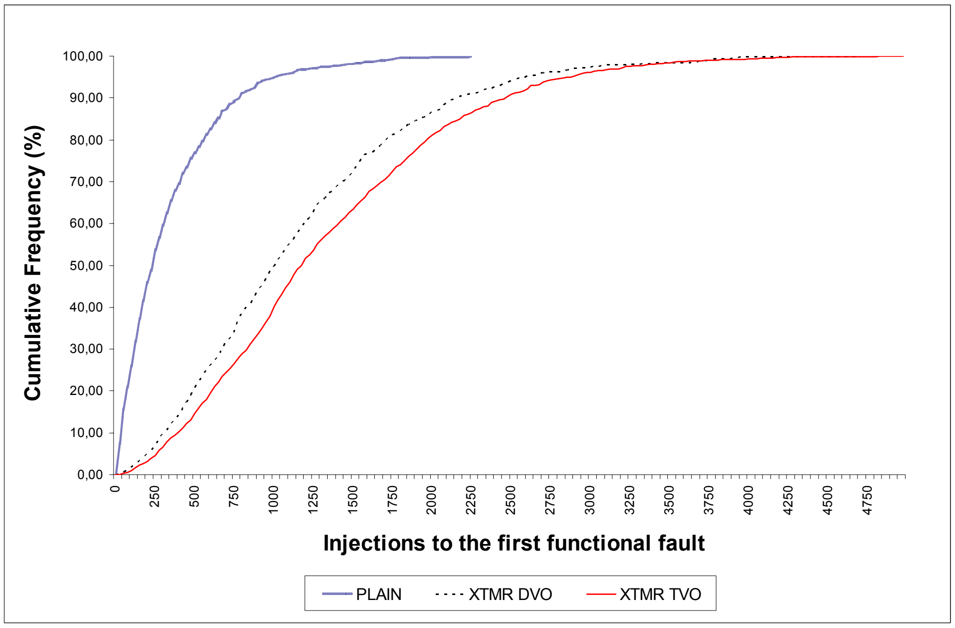


Figure 5: Cumulative distribution of the number of injections to the first functional fault

Zooming the Figure 5 into the interval [0,50] results in the curves shown in Figure 6, where again the double voted version closely follows the triple voted one. As shown by the figure, the distribution details the error probability for a few injected SEUs.

Figure 6 provides the indication that up to 8 accumulated SEUs, no failure has been observed. This suggests the conclusion that even a scrubbing rate allowing up to 8 upsets may be enough to prevent malfunction. Assuming a bit-error rate of 8,000 bit-errors/day for this device (geostationary orbit, worst case solar activity [25]), this corresponds to a scrubbing rate of 1/1,000 day or 1.44 min. However in a practical case, the statistical nature of the processes and the certainly limited coverage of test stimuli derived from a simulation test-bench will require a margin to be taken.

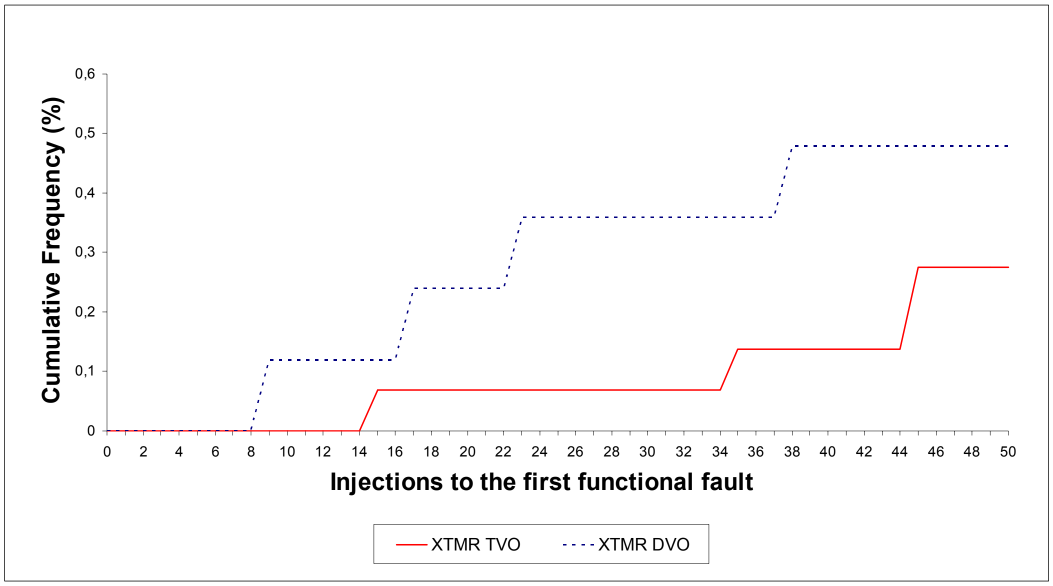


Figure 6: Cumulative frequency distribution (histograms class width = 1)

The experiment (3 mitigation schemes, 4,000 runs and 3.3×10^6 injected faults) took a global execution time of 17 hours. This corresponds to an injection time of 18 ms/fault. Part of this time (50 µsec) is for partial reconfiguration, while the rest comprises the functional test execution (26,000 testbench cycles), the generation of fault data packets and communication to the host. Compared to other reconfiguration based systems [18], FLIPPER shows higher injection efficiency.

7 CASE STUDY 2: “SAAB” DESIGN

The approach presented in this section has a twofold goal: to offer a methodology for estimating, in a first phase, the design sensitivity to soft errors, and secondly, to investigate various SEU hardening schemes. We used several tools in order to achieve this goal. At first, the FLIPPER fault injection platform [27] has been used in conjunction with the static analyzer tool STAR [7], in order to give a precise estimation of the soft error sensitivity; secondly, the reliability oriented place and route algorithm called RoRA [28] has been used to modify and harden the implemented circuits. In a previous work a detailed comparison between FLIPPER and the STAR static analysis tool was presented [29].
While statistically all bits of the configuration memory are equally exposed to radiation effects, the effect of a bit flip on the functionality of a circuit can be classified in three categories.

Some bits may not be used at all by a given circuit. Other bits may only affect the functionality if a combination of them is modified, for example by accumulation of Single Event Upsets (SEU) over time or by Multiple Bit Upsets (MBU). Accumulation effects may be mitigated by configuration memory scrubbing with an appropriate scrub cycle time. The most critical bits are the Single Points of Failure (SPF) of a design, meaning that a single bit may corrupt the circuit functionality. Mitigation against these SPF usually requires modification of the circuit by introducing redundancy as for example Triple Modular Redundancy (TMR) or by changing its topology, the placement and routing in the FPGA.

While the SPF are assessed and mitigated by analytic tools and structural methods like STAR, RoRA and TMR insertion, the effect of multiple bit flips, due to the high number of their combinations, can only be assessed with the help of experimental, statistical data. The main objective of the present work is to assess the robustness of different modules of a design against accumulation of soft errors with STAR and FLIPPER, and then harden the circuit with the RoRA tool. The robustness of the hardened design is verified again with FLIPPER.

The experimental results we achieved on a benchmark circuit provided by ESA demonstrate the feasibility of the methodology. Furthermore, the results show how RoRA can also be fruitfully used for reducing the sensitivity of a mapped circuit versus multiple soft errors accumulation.

7.1 Experiment

For the experimental analysis, a sample design composed of different functional modules has been provided by ESA. The same benchmark design, implemented in an XQR2V3000, had previously undergone radiation testing by SAAB Space [30], allowing for correlation of results with injection analysis. Two variants of the design have been analyzed by means of injection campaigns in accumulation mode. Both variants have been hardened by using the Xilinx TMRtool.

The first design variant (V1) has been derived from the unprotected design by straightly applying the TMRtool. The complete design is triplicated, outputs are triple voted and connected together on the PCB, and voters are inserted to the existing feedback paths of the design. The second variant (V2) has been obtained by modifying the plain design before the use of TMRtool. A local dummy feedback path, controlled by a multiplexer, has been inserted for each flip-flop. The switch input of the multiplexer is connected to a top-level port, which is never enabled during operation. As a consequence of that modification, TMRtool creates a voter for every flip-flop in the design.

Both variants have been analyzed by STAR. The outcomes of the analysis have been used as inputs to RoRA to produce more robust versions of the design (RoRA V1 and RoRA V2). The robustness of these four designs against the accumulation of soft errors has been evaluated by FLIPPER.

Sample Design
The sample design consists of various modules exploiting different resources inside the Virtex II architecture. Serial inputs and outputs are used in all modules, except those involving IO tests. Modules work independently of each other. Clock and reset nets are the only common resources. A brief description of the modules is given below.

- *FFT*: As reported in [30], the FFT module performs a Fourier Transform on a data matrix. The data, stored in the matrix, are generated by randomized input data. A signature of the computed transform is then shifted out to the serial output of the module.
- *MULT16_LUT*: The module is a 2-stage 16x16 bit multiplier instantiated twice. The upper 16 bits in a 32-bit register are multiplied with the lower 16 bits. The result is stored in a 32-bit register. In a first stage, new data are shifted into the 32-bit register from an input pin. One of the 32 bits of the last stage is connected to an output pin. A comparator circuit compares all results bitwise in each clock cycle and pipes the result to an output pin. Multipliers are implemented by LUTs.

- *MULT16_MULT18*: This is similar as the previous module, but each instance has 10 stages, and embedded multipliers are used in place of LUTs.
- *FFmatrix*: It consists of two identical copies of a 480 bit long chain of shift registers. It also includes a comparator circuit where each bit in the two copies is compared to each other. Any mismatch is brought to an output pin.
- *IOff_A/B*: It consists of a register linking 16 input pins to 16 output pins.
- *ROMff*: Two copies of a 256 bit shift register are instantiated. The former is loaded and holds the stored values; the latter cyclically reads the values stored by the former.

Resource	Plain	V1	V2	RoRA V1	RoRA V2
Slice FF	2,926 (4%)	8,778 (12%)	8,778 (12%)	8,778 (12%)	8,778 (12 %)
LUT	3,806 (5%)	13,437 (19%)	29,217 (43%)	13,437 (19 %)	29,214 (43 %)
IOB	87 (10%)	264 (32%)	267 (32%)	264 (32 %)	267 (32 %)
MULT 18x18	32 (22%)	96 (66%)	96 (66%)	96 (66 %)	96 (66 %)
GCLK	1 (6%)	3 (18%)	3 (18%)	3 (18 %)	3 (18 %)

Table 11: XQR2V6000 resource usage for different versions of SAAB design

The resource usage per module shown in Table 12. It is extracted from the mapped design netlist (.ncd).

			FFmatrix module		
Logic Utilization	**Plain**	**V1**	**V2**	**RoRA V1**	**RoRA V2**
FF (DFF)	1,104	3,313	3,313	3,313	3,313
LUT (FG)	271	813	7,437	813	7,437
MULT 18x18	0	0	0	0	0
Sensitive bit count		71,874	300,485	72,853	288,130
			Mult16_LUT		
Logic Utilization	**Plain**	**V1**	**V2**	**RoRA V1**	**RoRA V2**
FF (DFF)	193	579	579	579	579
LUT (FG)	1,181	3543	4701	3543	4701
MULT 18x18	0	0	0	0	0
Sensitive bit count		135,903	180,839	137,635	177,953
			FFTout		
Logic Utilization	**Plain**	**V1**	**V2**	**RoRA V1**	**RoRA V2**
FF (DFF)	360	1080	1080	1080	1080
LUT (FG)	1,435	5382	6468	5382	6468
MULT 18x18	12	36	36	36	36
Sensitive bit count		249,241	290,515	254,736	288,060
			Mult16_Mult18		
Logic Utilization	**Plain**	**V1**	**V2**	**RoRA V1**	**RoRA V2**
FF (DFF)	713	2139	2139	2139	2139
LUT (FG)	169	507	4785	507	4785
MULT 18x18	20	60	60	60	60
Sensitive bit count		58,854	220,628	69,334	203,962
			ROMff		
Logic Utilization	**Plain**	**V1**	**V2**	**RoRA V1**	**RoRA V2**
FF (DFF)	524	1572	1572	1572	1572
LUT (FG)	538	2421	4758	2421	4758
MULT 18x18	0	0	0	0	0
Sensitive bit count		107,626	187,122	108,120	178,948

Table 12: Resource usage per module

An independent set of stimuli is provided in input to each module. The complete set of stimuli is composed by about 28,000 vectors.
The occupancy of the design variants with respect to the plain design is reported in Table 11. For each design variant a fault injection campaign of about 1000 runs in accumulation mode has been performed. The predefined maximum number of injections per run is 100,000.

Procedure
The injection campaigns have been performed according to same procedure as described in Section 6.1. There are several independent modules in the sample design. When a functional failure occurs in one of the module, the outputs of the failed module are then masked, thus removing the module from further consideration in the test. The injection run ends when the totality of modules has failed or the maximum (predefined) number of injection has been reached.

7.2 Injection results

In this section, we report the results of the fault injection campaigns performed. We made a first analysis with the STAR tool for identifying SPFs and warnings of the TMR circuits. In order to remove SPFs and reduce the number of warnings reported by STAR, RoRA was applied. For the test designs, we reduced the number of warnings from 658 for the V1 to 140 for the RoRA V1, and from 690 for the V2, to 163 for the RoRA V2.
We then performed cumulative fault injection campaigns. The cumulative results of the various modules are illustrated from Figure 7 through Figure 11. The cumulative frequency (Y axis) is determined in the same way as described above in Section 6.2.

Two distinct categories of behaviour can be identified. For the modules FFmatrix, MULT16_MULT18, and ROMff the circuit RoRA V2 outperforms as expected the V2, V1 and RoRA V1 variants, as clearly depicted in Figure 7, Figure 10, and Figure 11 respectively. In particular, for the module FFmatrix, shown in Figure 7, RoRA V1 shows only a small advantage with respect to V1 below about 300 flipped bits; above 300, the two curves cannot be distinguished. Viceversa, the advantage of RoRA V2 against V2 is clear for the entire injection range.
As Figure 10 shows, the situation for MULT16_MULT18 is almost similar. The benefit of RoRA V1 adoption with respect to V1 is evident until 1200 injections. For ROMff, V1 behaves better than RoRA V1 for all injections, as shown in Figure 11.
In contrary to this general tendency, for the FFT module, V2 is worse than all the others (Figure 8), and for MULT16_LUT, the RoRA V2 variant is worst (Figure 9).

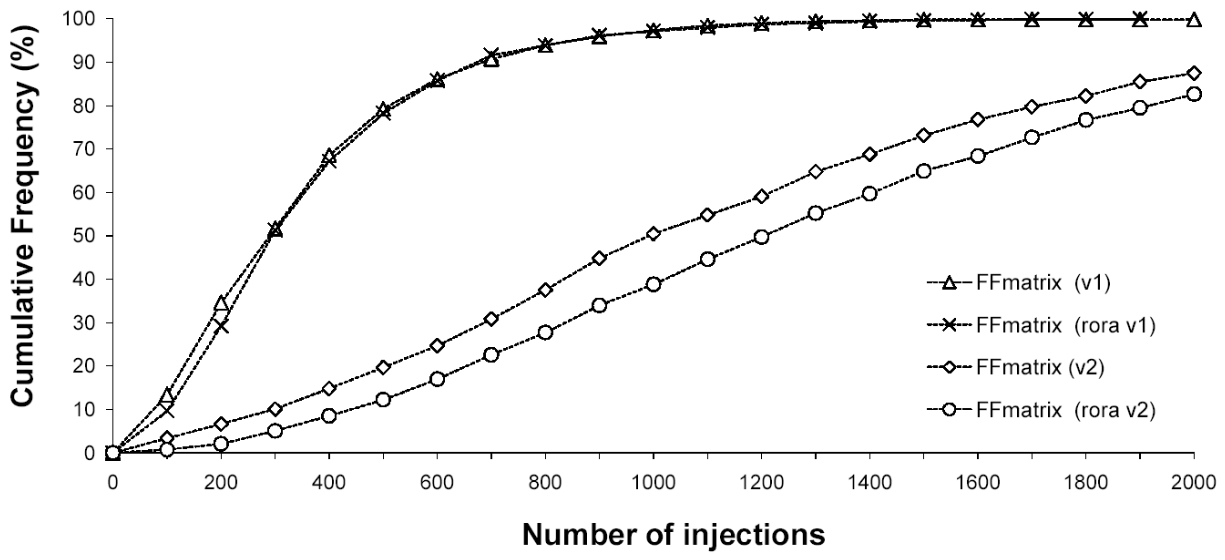


Figure 7: Cumulative frequency to the first failure (FFmatrix)

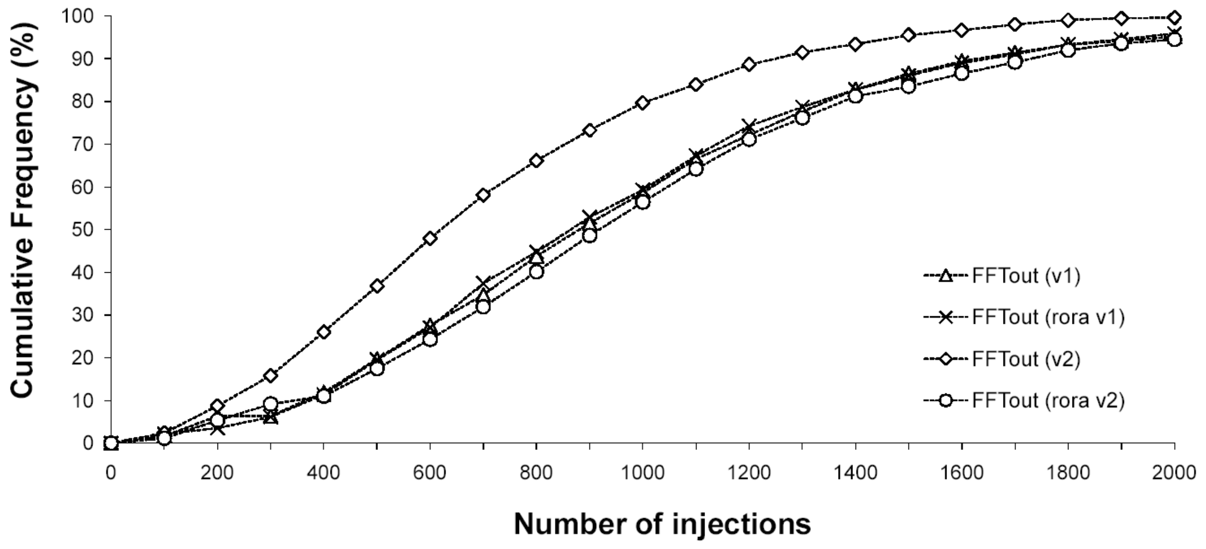


Figure 8: Cumulative frequency to the first failure (FFTout)

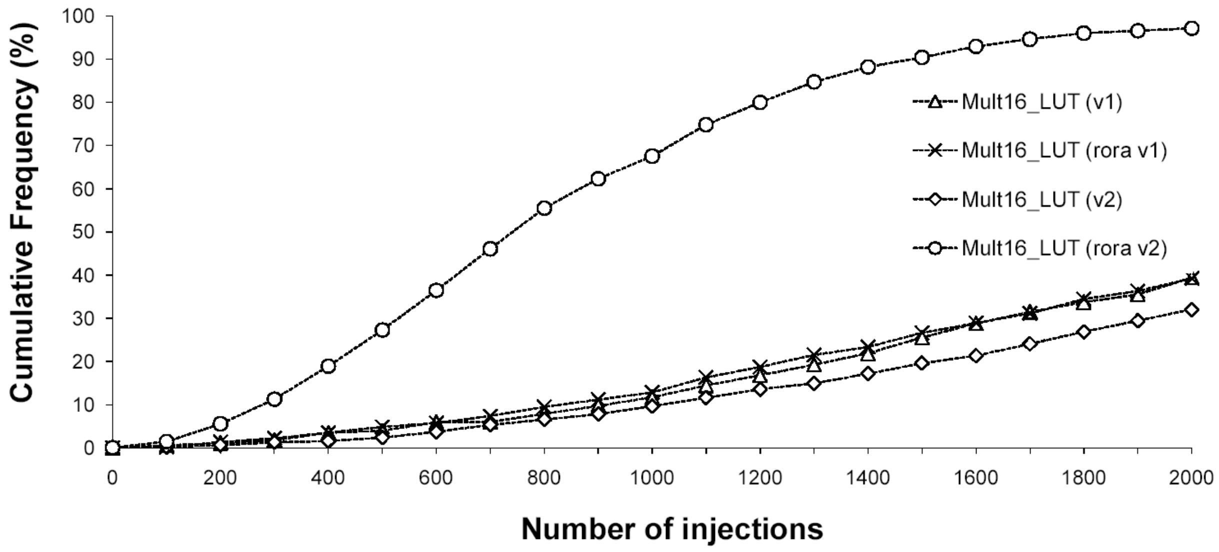


Figure 9: Cumulative frequency to the first failure (Mult16_LUT)

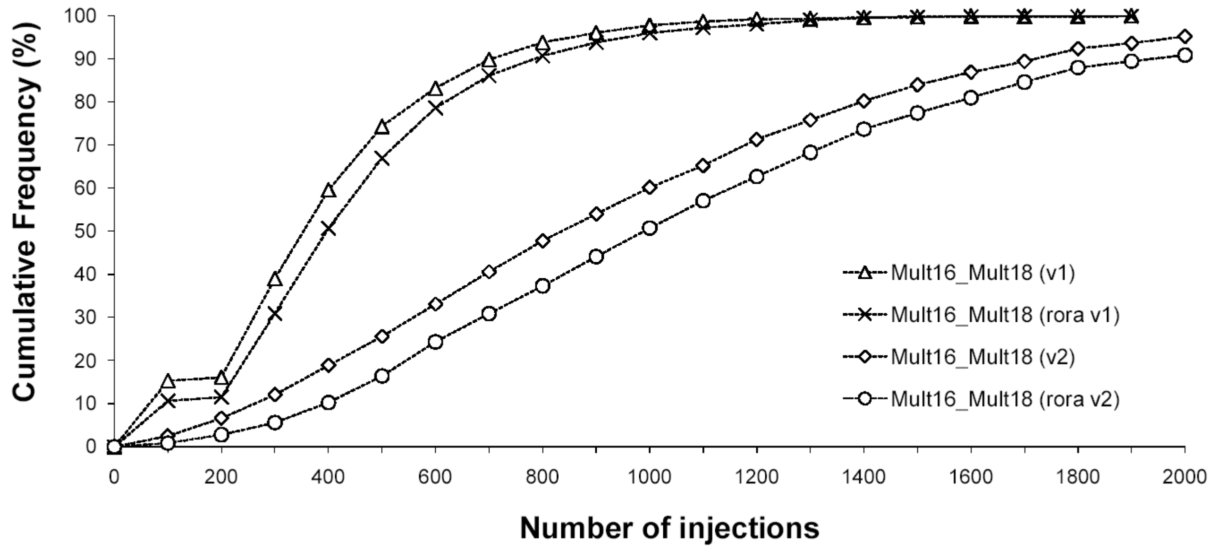


Figure 10: Cumulative frequency to the first failure (Mult16_Mult18)

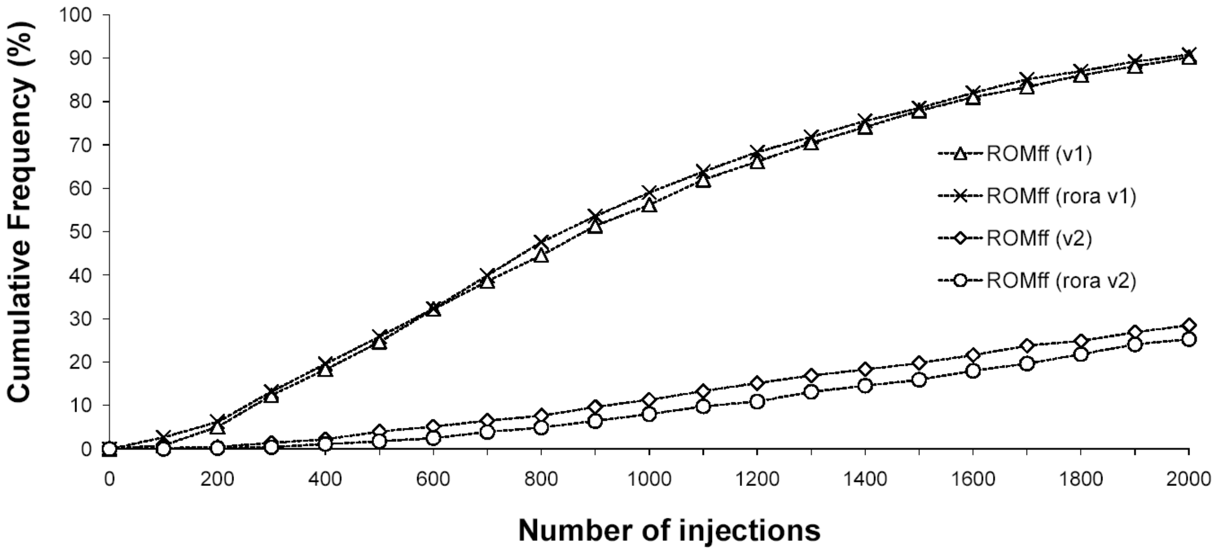


Figure 11: Cumulative frequency to the first failure (ROMff).

7.3 Injection results and irradiation data

In this section FLIPPER results are discussed against the radiation data of the test [30]. In comparing these data, the following main differences in the experimental set-up should be noticed: the target FPGA (XQR2V3000 for [30], XQR2V6000 for FLIPPER experiments), the placement, and VHDL parameters. For the sake of completeness, data from RoRA variants have been included in the graphs, although radiation data are not available for them.

The failure probabilities of FFmatrix, FFTout, Mult16_LUT, and Mult16_Mult18 modules are plotted in Figure 12 - Figure 19. For each module, the graph with injection results is accompanied by the one in which the irradiation data are also displayed (the RAD prefix is used in the legenda).

Plots from FLIPPER fault injection experiments show the failure probability for the 1-100 injection range. With respect to results presented in the previous section, an enlarged statistic population has been used, in order to increase the number of samples in that range. As expected, considering the differences in resource usages, V1 and V2 design variants exhibit a lower sensitivity in the XQR2V6000 than the XQR2V3000 device.

7.3.1 FFmatrix

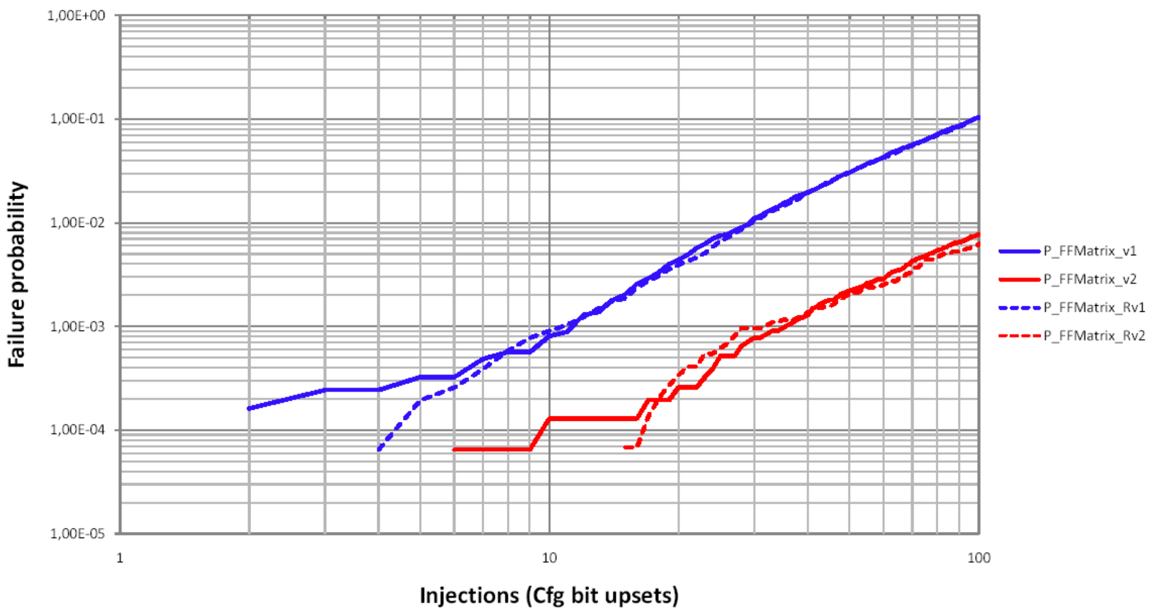


Figure 12: Error frequency for FFmatrix module - Injection

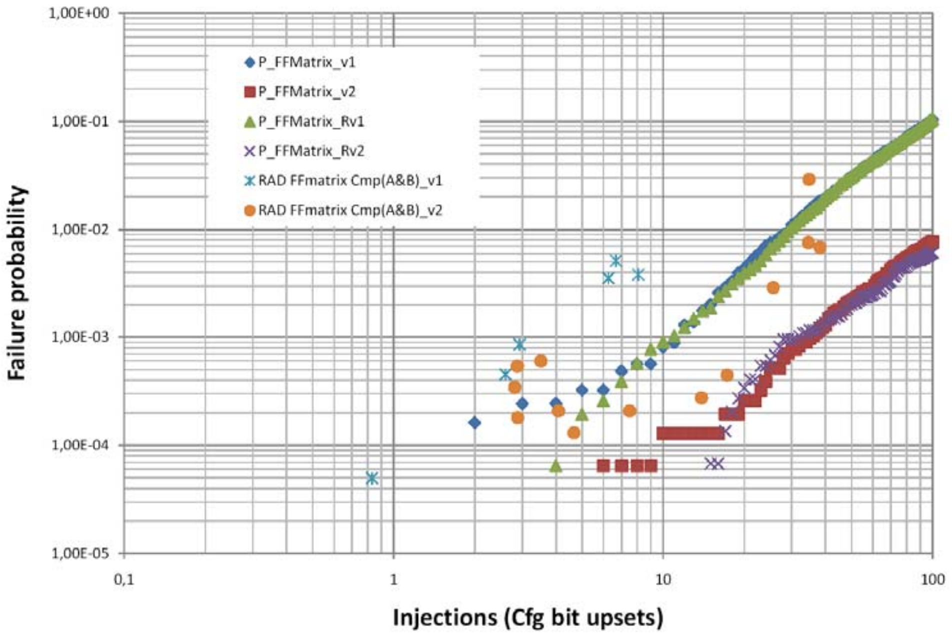


Figure 13: Comparison between injection and radiation data for the FFmatrix module

The graph in Figure 13 shows the improvement trend of V2 with respect to V1 reported by both injection and radiation test.

7.3.2 FFTout

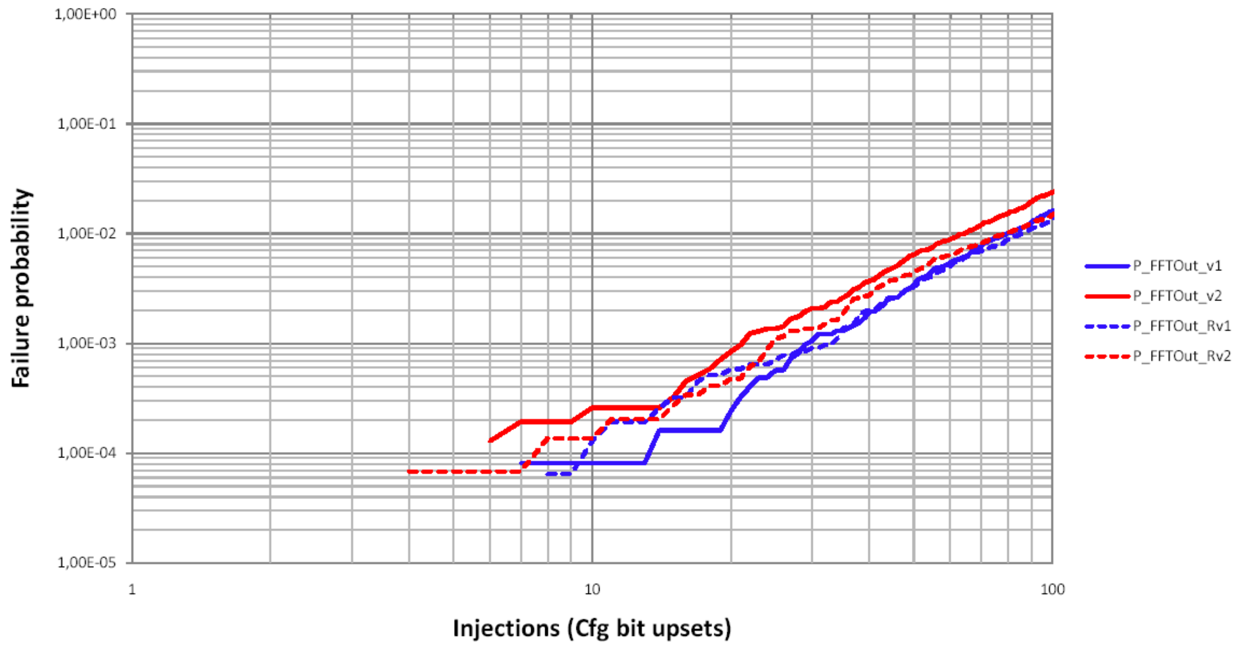


Figure 14: Error frequency for FFTout module - Injection

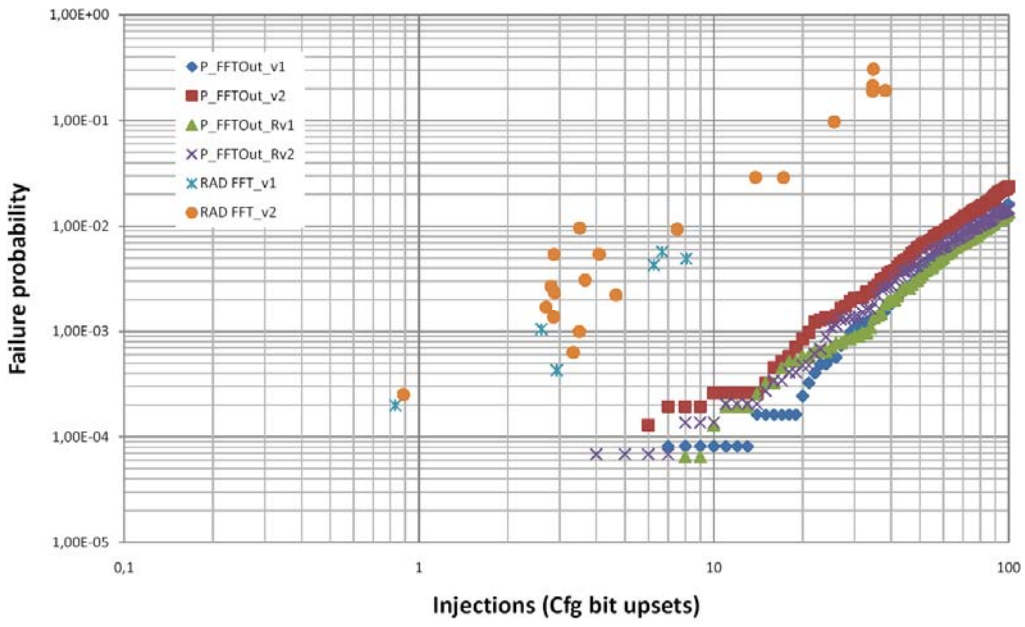


Figure 15: Comparison between injection and radiation data for the FFTout module

For this module, V2 appears to have a higher sensitivity than V1, and this results highlighted by injection in Figure 14 is also present in the radiation testing, as displayed in Figure 15 .

7.3.3 Mult16_LUT

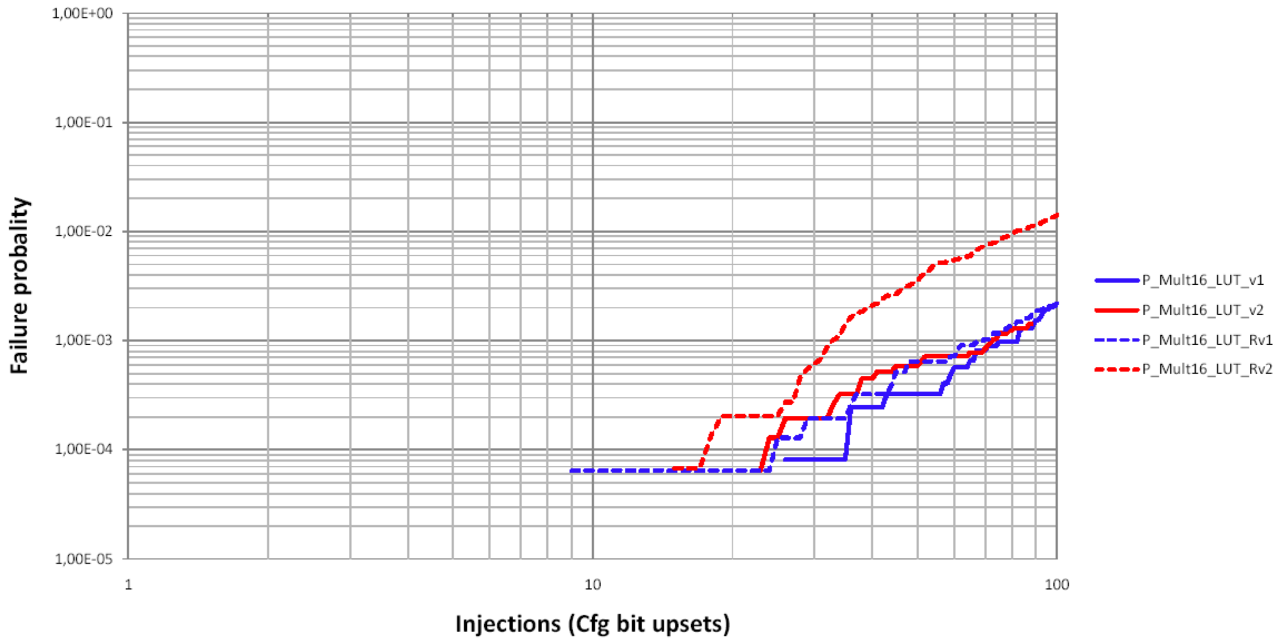


Figure 16: Error frequency for Mult16_LUT module - Injection

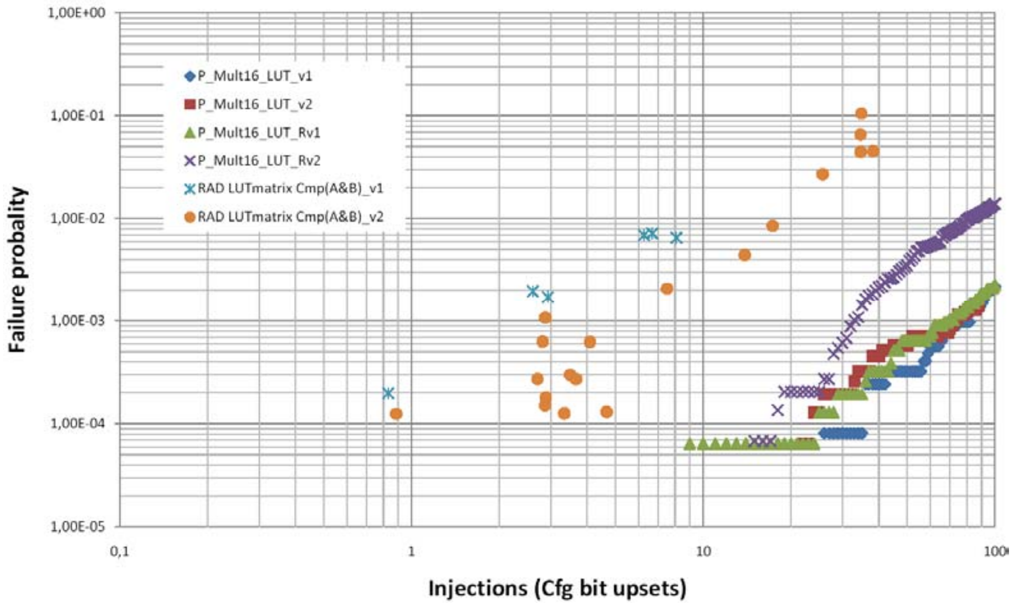


Figure 17: Comparison between injection and radiation data for the MULT16_LUT module

In this case opposite effects have been observed in injection and irradiation. While V1 shows an higher sensitivity than V2 in the radiation testing, V2 is more vulnerable than V1 in injection.

7.3.4 Mult16_Mult18

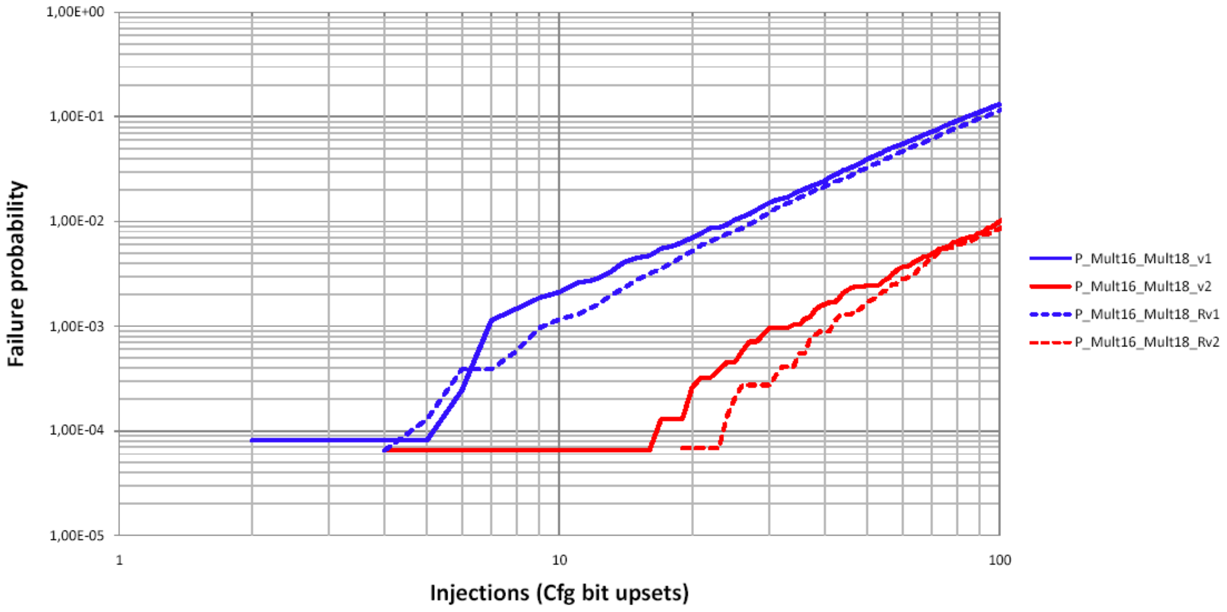


Figure 18: Error frequency for Mult16_Mult18 module - Injection

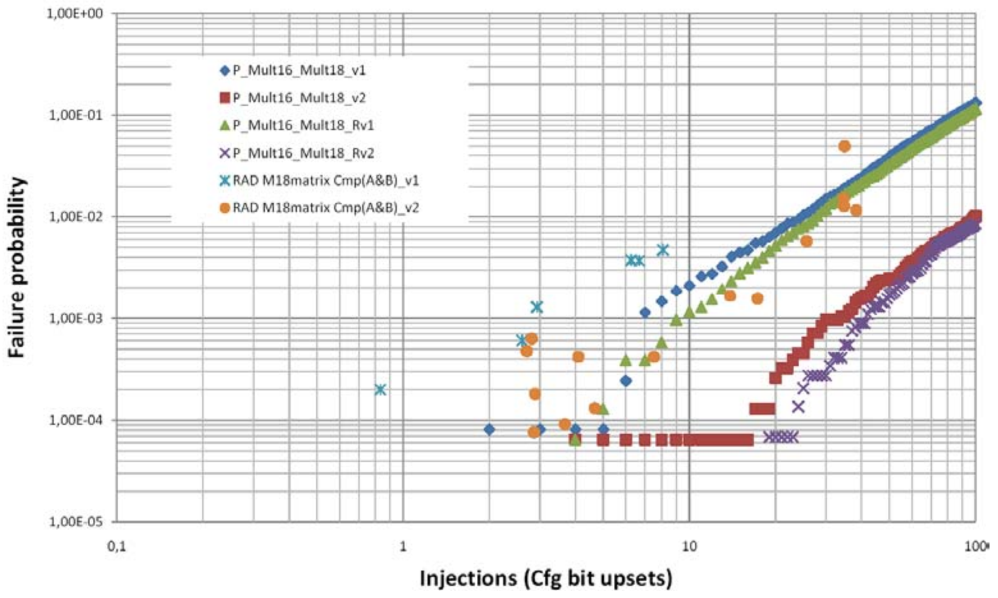


Figure 19: Comparison between injection and radiation data for the M18matrix (Mult16_Mult18) module

The improvement trend of V2 with respect to V1 is observed in both irradiation and injection. As shown in Figure 19, the improvement is higher in injection than in irradiation.

8 CONCLUSIONS

In this study contract an hardware fault injection system for Virtex II devices was designed and implemented. Results on benchmark designs showed the tool is effective for the evaluation of their SEU sensitivity. Various mitigation techniques were analyzed.
Moreover, we presented a methodology based on FLIPPER and two tools developed by the Politecnico di Torino, STAR and RoRA, for evaluating and hardening designs implemented by SRAM-based FPGAs against the accumulation of soft error within their configuration memory.

The results demonstrated the feasibility of both the analysis and hardening method, thus enabling the study of multiple error sensitivity of circuits hardened according to standard redundancy approaches.

It has been demonstrated that the robustness of designs protected by TMRtool against accumulated SEU or MBU can be improved by applying the RoRA algorithm and by increasing the number of TMR voters in the circuit. Such additional mitigation may allow operating the designs at reduced scrubbing rates. Some exceptions however show that the mitigation to be applied needs to be validated for every individual design, because in some cases, also a deterioration is observed.

In future work, more designs shall be assessed with the proposed method. This will include also designs not protected by TMRtool, with the objective to find alternative mitigation methods with a reduced area overhead, not using full triplication. In order the investigate on the use of these devices in real applications, where scrubbing is generally employed, future activities should enable injection campaigns in which a reduced number of bit flips (two or three at maximum) is allowed to accumulate.

In order to achieve a representative statistics of design sensitivity, even with a reduced number of accumulated bit-flips, future injection campaigns should maximize the number of different target locations of the configuration memory.

Furthermore is it planned to upgrade the FLIPPER test system to Virtex-4 devices, and to allow injection also after the start of the test sequence, in order to achieve a more realistic modelling of the real situation in orbit, where SEU occur at any time during operation of the circuit.

The upgrade will exploit the experience gained with the actual system, and maximize system performances and injection efficiency. A powerful Virtex 5 device will replace the current Virtex II Pro in the control board. An Ethernet link at 1 Gbit/s will substitute the USB connection to the PC, and DDR2 SODIMM will provide a faster and larger on board memory.

9 REFERENCE DOCUMENTS

[1] P.E. Dodd and L.W. Massengill, “Basic Mechanisms and Modeling of Single-Event Upset in Digital Microelectronics”, IEEE Trans. on Nucl. Sci., vol. 50, n. 3, pp 583-602, June 2003.
[2] M. Caffrey, P. Graham, E. Johnson and M. Wirthlin,, “Single-Event Upsets in SRAM FPGAs”, in Proc. of the Military and Aerospace Applications of Programmable Devices Int’l Conference (MAPLD), September 2002.
[3] C.C. Yui, G.M. Swift, C. Carmichael, R. Koga and J.S. George, “SEU Mitigation Testing of Xilinx Virtex II FPGAs”, Radiation Effects Data Workshop Record, 2003.
[4] S. Rezgui, G.M. Swift, K. Somervill, J. George, C. Carmichael, and G. Allen, “Complex Upset Mitigation Applied to a Re-Configurable Embedded Processor”, IEEE Trans. on Nucl. Sci., vol. 52, n. 6, pp 2468-2474, Dec. 2005.
[5] G.M. Swift, S. Rezgui, J. George, C. Carmichael, M. Napier, J. Maksymowicz, M. Moore, A. Lesea, R. Koga, T.F. and Wrobel, “Dynamic Testing of Xilinx Virtex-II Field Programmable Gate Array (FPGA) Input/Output Blocks (IOBs)”, IEEE Trans. on Nucl. Sci., vol. 51, n. 6, pp 3469-3474, Dec. 2004.

[6] V. Pouget, A. Douin, D. Lewis, P. Fouillat, G. Foucard, P. Peronnard, V. Maingot, J.B. Ferron, L. Anghel, R. Leveugle and R. Velazco, "Tools and Methodology Development for Pulsed Laser Fault Injection in SRAM-Based FPGAs", 8th Latin American Test Workshop (LATW 2007), [Cusco (Peru), 11- 14 Mars 2007].
[7] L. Sterpone and M. Violante, "A new analytical approach to estimate the effects of SEUs in TMR architectures implemented through SRAM-based FPGA's", IEEE Trans. Nucl. Sci., vol. 52, no. 6, pp. 2217–2223, Dec. 2005.
[8] FLIPPER product sheet, http://microelectronics.esa.int/techno/Flipper_ProductSheet.pdf.
[9] TMR Tool User Guide, UG156 (v1.0), 30 Sept. 2004, Xilinx.
[10] Daniel González Gutiérrez, "Single Event Upsets Simulation Tool Functional Description", ESA ESTEC, July 2004 http://microelectronics.esa.int/asic/SST-FunctionalDescription1-3.pdf
[11] H. Cha, E. M. Rudnick, J. Patel, R. K. Iyer, G.S.Choi, "A gate-level simulation environment for alpha-particle-induced transient faults", IEEE Trans. Comput., vol. 45, pp. 1248–1256, Nov. 1996.
[12] M. Rebaudengo, M. Sonza Reorda, M. Violante, B. Nicolescu, R. Velazco, " Coping With SEUs/SETs in Microprocessors by Means of Low-Cost Solutions: A Comparison Study", IEEE Trans. on Nucl. Sci., Vol. 49, No. 3, June 2002
[13] V. Pouget, A. Douin, D. Lewis, P. Fouillat, G. Foucard, P. Peronnard, V. Maingot, J.B. Ferron, L. Anghel, R. Leveugle and R. Velazco, "Tools and Methodology Development for Pulsed Laser Fault Injection in SRAM-Based FPGAs", in 8th Latin American Test Workshop (LATW 2007), [Cusco (Peru), 11- 14 Mars 2007].
[14] C. Carmichael, "Correcting Single-Event Upsets Through Virtex Partial Configuration", XAPP216 (v1.0) June 1, 2000, Xilinx.
[15] L. Antoni, R. Leveugle, B. Fehér, "Using Run-Time Reconfiguration for Fault Injection in Hardware Prototypes", in Proc. of the 2000 Int'l Symposium on Defect and Fault Tolerance in VLSI Systems, [Monte Yamanashi, October 25-27, 2000], pp. 405-413.
[16] M. Aguirre, J.N. Tombs, F. Muñoz, V. Baena, A. Torralba, A. Fernández-León, F. Tortosa, and D. González-Gutiérrez, "An FPGA based hardware emulator for the insertion and analysis of Single Event Upsets in VLSI Designs", in Proc. Radiation Effects on Components and Systems Conf. (RADECS), Madrid, Spain, Sept. 2004.
[17] P. Civera, L. Macchiarulo, M. Rebaudengo, M. Sonza Reorda, and M. Violante, "Exploiting Circuit Emulation for Fast Hardness Evaluation", in IEEE Trans. on Nucl. Sci., vol. 48, n. 6, pp 2210-2216, Dec. 2001
[18] C. López-Ongil, M. García-Valderas, M. Portela-García, and L. Entrena, "Autonomous Fault Emulation: A New FPGA-Based Acceleration System for Hardness Evaluation", IEEE Trans. on Nucl. Sci., vol. 54, n. 1, pp 252-261, Feb. 2007.
[19] E. Johnson, M. Caffrey, P. Graham and M. Rollins, "Accelerator Validation of an FPGA SEU Simulator", IEEE Trans. on Nucl. Sci., vol. 50, n. 6, pp 2147-2157, Dec. 2003.
[20] M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, A. Marmo, S. Pastore, and G.R. Sechi, "A Tool for Injecting SEU-like Faults into the Configuration Control Mechanism of Xilinx Virtex FPGAs", in Proc. of the 2003 Int'l Symposium on Defect and Fault Tolerance in VLSI Systems, [Cambridge, November 3 - 5, 2003, USA], pp. 71-78.
[21] M. Alderighi, A. Candelori, F. Casini, S. D'Angelo, M. Mancini, A. Paccagnella, S. Pastore and G.R. Sechi, "SEU Sensitivity of Virtex Configuration Logic", in IEEE Trans. on Nucl. Sci., vol. 52, n. 6, pp 2462-2467, Dec. 2005.
[22] CCSDS Unsegmented Code (CUC) & CCSDS Time Manager (CTM) Synthesizable VHDL Cores Data Sheet, Dec. 2003, ESA ESTEC. http://microelectronics.esa.int/core/corepage.html
[23] C. Carmichael, "Radiation effects and mitigation overview", XDAF, April 2004.
[24] Virtex-II Platform FPGA User Guide, UG002 (v2.0) 23 March 2005, Xilinx.
[25] C. Carmichael, B. Bridgford, G. Swift, and M. Napier; "A Triple Module Redundancy Scheme for SEU Mitigation of Static Latch-Based FPGAs", Military Applications of Programmable Logic Devices (MAPLD) Conference 2004, available at http://www.klabs.org/mapld04/presentations/session_l/5_l189_carmichael_s.ppt.

[26] H. Quinn, P. Graham, J. Krone, M. Caffrey and S. Rezgui, “Radiation-Induced Multi-Bit Upsets in SRAM-Based FPGAs”, IEEE Transactions on Nuclear Science, Vol. 52, No. 6, December 2005, pp 2455 – 2461.

[27] M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, S. Pastore, G.R. Sechi, and R. Weigand, “Evaluation of Single Event Upset Mitigation Schemes for SRAM based FPGAs using the FLIPPER Fault Injection Platform”, in Proc. of the 22nd IEEE Int'l Symposium. on Defect and Fault Tolerance in VLSI Systems, Rome, Italy, IEEE CS Press, Sept. 2007, pp. 105-113.

[28] L. Sterpone, M. Violante, “A new reliability-oriented place and route algorithm for SRAM-based FPGAs”, *IEEE Trans. on Computers*, Vol. 55, No. 6, June 2006, pp. 732 – 744.

[29] M. Alderighi, F. Casini, S. D'Angelo, M. Mancini, S. Pastore, L. Sterpone, and M. Violante, "Soft errors in SRAM-based FPGAs: a comparison of two complementary approaches", *IEEE Trans. on Nuclear Science,* 2008, in print.

[30] “Particle Test of Xilinx Virtex-II FPGA using XTMR Mitigation Technique”, European Space Agency Contract Report, 20 Sept. 2006, Saab Ericsson Space AB.