ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
UNIVERSITY OF PIRAEUS

Embedded Systems Lab (ESLab)
Department of Informatics

SINGLE EVENT UPSETS (SEU) TEST REPORT

Single-Event Upsets Characterization of the 28nm Artix-7-based Programmable Logic of Xilinx Zynq-7000 FPGA

Radiation Testing Date	12-18 Nov 2018
Participants	University of Piraeus/Embedded System Lab, ESA/ESTEC
DUT	Xilinx Zynq-7000 AP SoC XC7Z020-CLG484
Board	Zedboard
Radiation type	Very High Energy Heavy ions (Pb ions)
Accelerator	CERN Super-Proton-Synchrotron North Area (SPS-NA)
Radiation source	Energy: up to 150A GeV/c, effective LETs: 8.8, 12.45 MeVcm2/mg
Test type	Single Event Upsets (SEUs)

Reference: ESL-TR-1901

Version: 1.0

Date: 1 July 2019

Authors: Vasileios Vlagkoulis, Mihalis Psarakis (University of Piraeus)

Revision History

Version	Date	Authors	Comments
1.0	1 July 2019	V.Vlagkoulis, M.Psarakis	First release for review

Table of Contents

List of Figures

List of Tables

1. Scope

1.1 Scope of the document

This document reports the results of the radiation testing of the 28nm Xilinx Zynq-7000 FPGA device with high energy heavy ions (Pb) in the **CERN Super-Proton-Synchrotron North Area (SPS-NA)**.

1.2 References

[1] Vasileios Vlagkoulis et al., "Analysis of the Single Event Upsets in the Programmable Logic of 28 nm Xilinx Zynq-7000 FPGA due to Heavy Ion Irradiation", Single Event Effects (SEE) Symposium and Military and Aerospace Programmable Logic Devices (MAPLD) Workshop, 2019.

[2] Vasileios Vlagkoulis et al., "Configuration Memory Scrubbing of the Xilinx Zynq-7000 FPGA using a Mixed 2-D Coding Technique", European Conference on Radiation and its Effects on Components and Systems (RADECS), 2019.

[3] R. G. Alía et al., "Ultraenergetic Heavy-Ion Beams in the CERN Accelerator Complex for Radiation Effects Testing," in IEEE Trans. on Nucl. Sci., vol. 66, no. 1, pp. 458-465, Jan. 2019. doi: 10.1109/TNS.2018.2883501

[4] Amrbar, Mehran, et al. "Heavy ion single event effects measurements of Xilinx Zynq-7000 FPGA." 2015 IEEE Radiation Effects Data Workshop (REDW). IEEE, 2015.

[5] Tambara, Lucas Antunes, et al. "Heavy ions induced single event upsets testing of the 28 nm xilinx zynq-7000 all programmable soc." 2015 IEEE Radiation Effects Data Workshop (REDW). IEEE, 2015.

[6] Stoddard, Aaron, et al. "A hybrid approach to FPGA configuration scrubbing." IEEE Transactions on Nuclear Science 64.1 (2017): 497-503.

[7] Furano, Gianluca, et al. "FPGA SEE Test with Ultra-High Energy Heavy Ions." 2018 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT). IEEE, 2018.

[8] Yang, Weitao, et al. "Microbeam Heavy-Ion Single-Event Effect on Xilinx 28-nm System on Chip." IEEE Transactions on Nuclear Science 65.1 (2018): 545-549.

[9] A. Sari, V. Vlagkoulis and M. Psarakis, "An Open-source Framework for Xilinx FPGA Reliability Evaluation", Workshop on Open Source Design Automation (OSDA) 2019, Florence, Italy, in conjunction with DATE 2019.

[10] Xilinx, Inc. " 7 Series FPGAs Configuration User Guide v1. 13 (UG470)." (2018)

[11] Markus Happe, Andreas Traber, and Ariane Keller. "Preemptive Hardware Multitasking in ReconOS." Applied Reconfigurable Computing – 11th International Symposium (ARC), Bochum, Germany, 2015.

[12] B. Gill, M. Nicolaidis, C. Papachristou. "Radiation induced single-word multiple-bit upsets correction in SRAM." 11th IEEE International On-Line Testing Symposium, French Riviera, France, 2005.

[13] ESCC Basic Specification No. 25100: SINGLE EVENT EFFECTS TEST METHOD AND GUIDELINES, Issue 2, 2014

[14] JESD89-3A: Test Method for Beam Accelerated Soft Error Rate, Nov 2007

[15] Wirthlin, Michael, et al. "A method and case study on identifying physically adjacent multiple-cell upsets using 28-nm, interleaved and SECDED-protected arrays." IEEE transactions on nuclear science 61.6 (2014): 3080-3087.

[16] ESA-ESTEC. "Space product assurance - Techniques for radiation effects mitigation in ASICs and FPGAs handbook (ECSS-Q-HB-60-02A)" (2016)

[17] Xilinx, Inc. "LogicCORE IP: Soft Error Mitigation (SEM) Controller v4.1 (PG036)." (2018).

[18] Park, Sang Phill, Dongsoo Lee, and Kaushik Roy. "Soft-error-resilient FPGAs using built-in 2-D Hamming product code." IEEE transactions on very large scale integration (VLSI) systems 20.2 (2012): 248-256.

[19] Venkataraman, Shyamsundar, et al. "A bit-interleaved embedded hamming scheme to correct single-bit and multi-bit upsets for SRAM-based FPGAs." Field Programmable Logic and Applications (FPL), 2014 24th International Conference on. IEEE, 2014.

[20] Ebrahimi, Mojtaba, and Mehdi B. Tahoori. "Stepped parity: A low-cost multiple bit upset detection technique." 2015 IEEE International Test Conference (ITC). IEEE, 2015.

[21] Ebrahimi, Mojtaba, et al. "Low-cost multiple bit upset correction in SRAM-based FPGA configuration frames." IEEE Transactions on Very Large Scale Integration (VLSI) Systems 24.3 (2016): 932-943.

[22] Mandal, Swagata, et al. "Efficient dynamic priority based soft error mitigation techniques for configuration memory of FPGA hardware." Microprocessors and Microsystems 51 (2017): 313-330.

2. Introduction

This document presents the **Single-Event Upsets (SEUs)** characterization of the **28nm Artix-7-based Programmable Logic (PL) of a Xilinx Zynq-7000** device due to heavy ion irradiation. All the embedded memories of the PL part of the Xilinx Zynq-7000 device, i.e. Configuration memory (CRAM), Block RAM (BRAM) and user Flip-Flops (FFs), are investigated.

Two different experiments (test setups) were conducted in parallel (in two different boards):

i) **Test #1**: Recording of the Single Bit Upsets (SBUs), Multiple Cell Upsets (MCUs) and Multiple Bit Upsets (MBUs) and calculation of cross section of the embedded memories of the PL part of the Xilinx Zynq-7000

ii) **Test #2**: Evaluation of a configuration memory scrubbing approach for the Xilinx Zynq-7000 FPGA based on a mixed 2D coding scheme

The experimental results of Test #1 have been presented in [1] while the results of Test #2 have been described and submitted in [2].

This work was partially funded by ESA/ESTEC under the purchase order N.5201016857.

3. Test objectives

The main purpose of the radiation tests is to analyze in-depth the SEU vulnerability of the PL part of the Xilinx Zynq-7000 FPGA device under heavy ion irradiation and study the feasibility of using Xilinx Zynq-7000 in space applications.

The objectives of the radiation test #1 are:

i) Calculate the cross sections for all the different memory types of the PL part: CRAM, Flip-Flops (FFs), Shift Register LUTs (SRLs) and Block RAMs (BRAMs)

ii) Calculate the cross sections for the essential bits of the CRAM, i.e. including only upsets in the essential bits of the CRAM. Essential bits are the device configuration bits associated with the circuitry of the design

iii) Determine and analyze the Single Bit Upsets (SBUs) and the Multiple Cell Upsets (MCUs) in CRAM

The objectives of the radiation test #2 are:

i) Evaluate the efficiency of a configuration memory scrubbing approach for the Xilinx Zynq-7000 based on 2-D coding scheme, i.e. to check whether the proposed scrubbing approach detects and corrects all SBUs and MCUs observed during the radiation experiments

4. Test facility

The radiation tests were performed in the CERN Super-Proton-Synchrotron North Area (SPS-NA) [3] under heavy ions (Pb) irradiation. The energy of the lead ions reached 150A GeV/c. Note that the very high energy of the ion beams allows for testing in air, with packaged parts and enabling tilting up to large angles. The beam arrives at the facility in form of spills of length between 4.5 and 10 s, with a periodicity of ~41s. During the tests, a beam size of ~40 mm × 40 mm was set, and the beam flux was varied between 1×10^2 and 2×10^3 ions/cm^2 per spill.

The beam size was measured through a Delay Wire Chamber with a 2 mm accuracy, whereas the beam intensity was measured for every spill with a scintillator detector. The alignment of the Device Under Test (DUT) with the beam was ensured by means of a movable table and independently confirmed through the readout on the ESA reference SEU monitor throughout the tests.

5. Test setup

The FPGA device under test (DUT) is the Xilinx Zynq-7000 SoC device. These devices integrate a processing system (PS) based on a single or dual-core ARM Cortex-A9 CPU and Xilinx programmable logic (PL) in a single device built on a 28nm, high-k metal gate process technology. The PL includes several different types of resources including among others: configurable logic blocks (CLBs), BRAMs, DSP slices, etc. The radiation tests have been performed in an Avnet Zedboard (see **Figure 1**) which integrates an XC7Z020-1CLG484C Zynq-7000 APSoC device. The PL part of the Xilinx XC7Z020 device is derived from the Xilinx Artix-7 series technology and is the main target of this study.

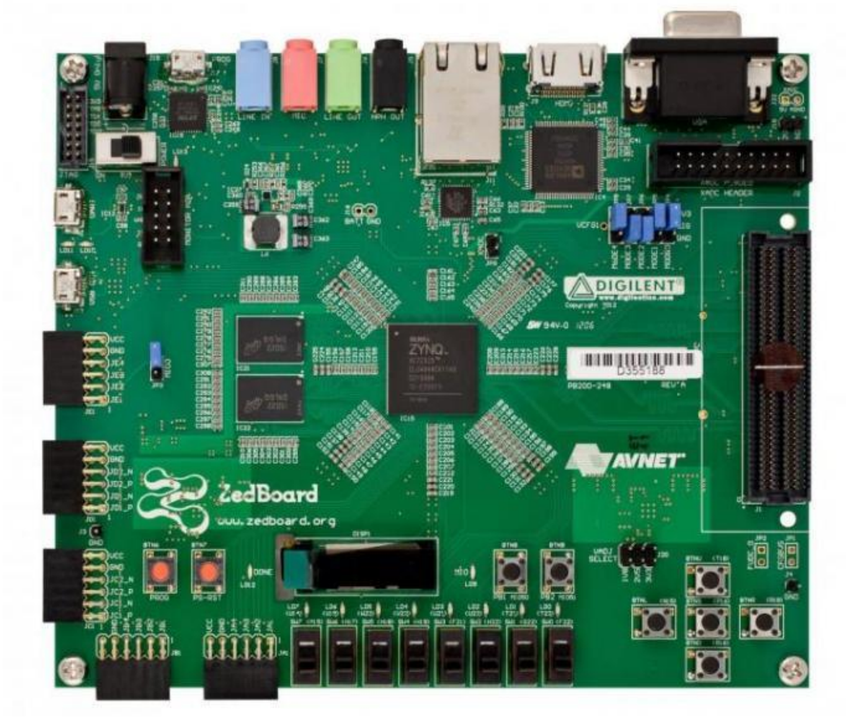


Figure 1: Board under test (Avnet Zedboard)

The test setup is shown in **Figure 2**. Two experiments were conducted in parallel in two different Zedboards, Zedboard #1 and Zedboard #2:

- The Zedboards are powered by an N6705 power supply which is remotely controlled by the Control Room Laptop 3 (CRL 3).
- The Zedboards #1 and #2 are connected through the JTAG port with the Beam Room Laptops 1 & 2 (BRL 1 & BRL 2), respectively, for bitstream configuration and readback purposes.
- The BRL 1 and BRL 2 are remotely controlled and monitored through Windows Remote Desktop by CRL1 and CRL2.
- The BRL 1 and BRL 2 run the test_app #1 and test_app #2, respectively. Test_app #1 performs FPGA configuration, data acquisition (FPGA readback) and logging. Test_app #2 performs FPGA configuration and data logging and runs the proposed configuration memory scrubbing algorithm. More information about the test applications #1 and #2 are given in Sections 6 and 8, respectively.
- The Beam signal is monitored by two portable USB Oscilloscopes (Digilent Analog Discovery 2) which are connected to the BRL 1 and BRL 2. The Beam signal indicates the start and the end of the irradiation periods (1: beam on, 0: beam off). When the Oscilloscope detects a falling edge in the Beam signal, it actually identifies the end of spill. This event detection triggers the test applications running on the BRL 1 and BRL 2.
- All system clocks were synchronized before the tests with the facility's clock. All instruments' (PSU, BRLs, CRLs) logs are time-stamped for post processing.

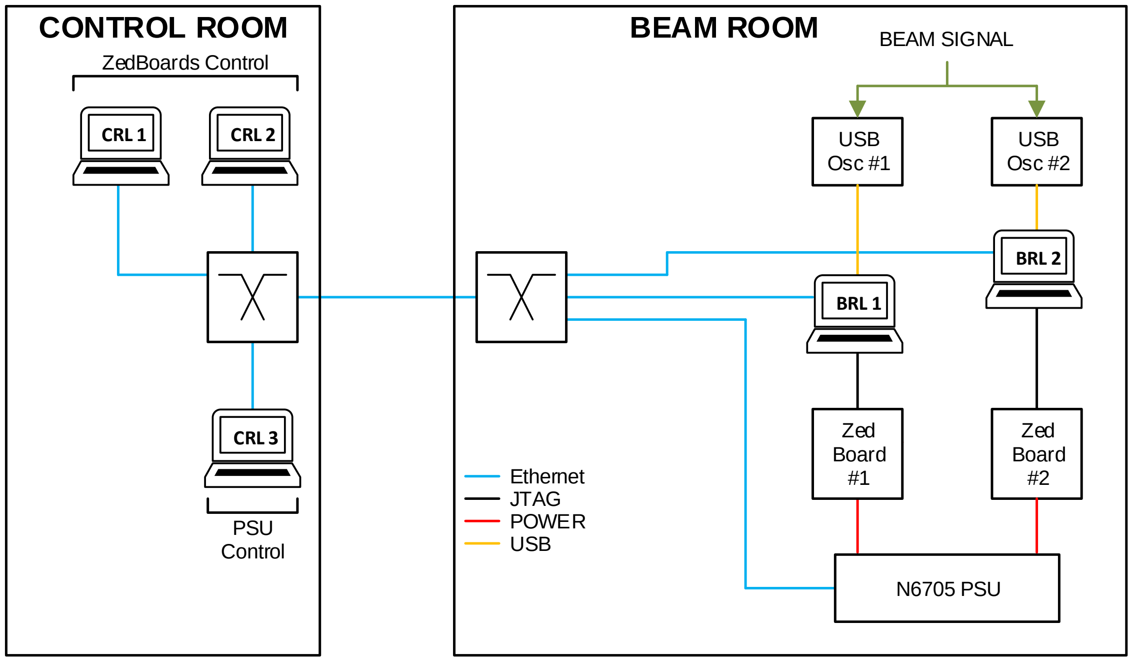


Figure 2: Test setup

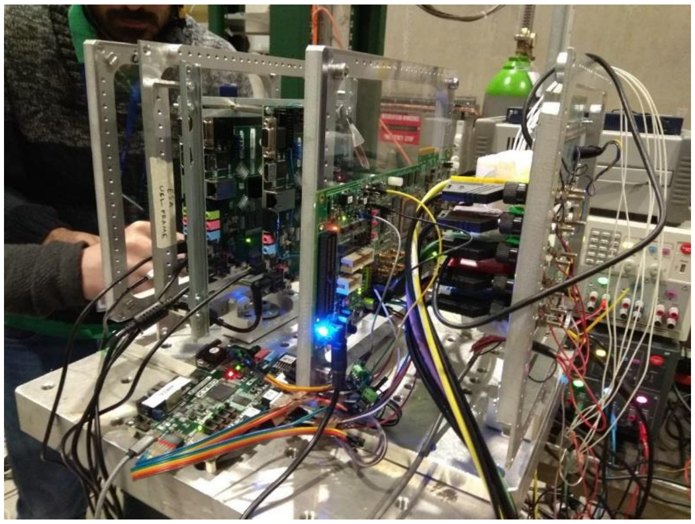


Figure 3: Photo of boards under test

Figure 3 photo depicts the Zedboards connected in the beam room. Actually, multiple (four) boards are irradiated in parallel. The front board (Myriad chip) and the second board are for other radiation experiments. Our Zedboards are the 3rd and the 4th boards in line.

<u>Note:</u> The high energy of the beam ions allows the irradiation of stacked boards in parallel.

6. Test #1: SEU characterization

The goal of this test is to study the upsets in the embedded memories of the Xilinx Zynq-7000 under heavy ion irradiation. Last years, several studies of the radiation effects in Xilinx Zynq-7000 devices have been presented in the literature. These experiments have investigated the SEUs in the various memories of the device [4],[5],[6],[7], i.e. configuration memory (CRAM) and BRAM of the programmable logic (PL) and caches and on-chip memory (OCM) of the processing system (PS) or the SEE impact in the entire PS area [8].

However, in this radiation experiment we provide a deeper insight in the SEU vulnerability of the device:

- Analyze the SEU vulnerability of all different memory types of the FPGA device: CRAM, Shift Register LUTs (SRLs), BRAM and user FFs. Previous approaches mainly focus on the upsets in the CRAM.
- Separate the SBUs and MCUs and analyze the characteristics of MCUs that may affect an MCU mitigation approach, e.g. the number of upsets per configuration frame, the shapes of upsets, etc.
- Identify upsets occurred in SRLs and FFs due to SETs (in global signals, e.g. clock tree)
- Analyze how the SEUs are affected by the memory contents (i.e. the likelihood to occur upset in memory cells preloaded with ‘0’ or with ‘1’)
- Use a synthetic benchmark that fully utilizes the FPGA device to improve the statistical insignificance of the experiment

Note that no SEU detection and correction approach is applied during irradiation. The configuration memory is readback when the beam goes off, the upsets are recorded and the configuration memory is re-written to remove upsets.

6.1 Circuit under test (CUT)

A synthetic, parameterized benchmark has been designed for the purposes of the radiation tests.

- The FPGA CUT communicates with the BRL1 through the JTAG interface (use of BSCANE2 primitive) as shown in Figure 27 (The figures of Test #1 CUT are included in Appendix A).
- The CUT is not clocked during irradiation, so we can capture Single Event Upsets (SEUs) in the user memories of the PL part, i.e. BRAM, SRL and FFs using the Xilinx Readback Capture[1] command. Otherwise (i.e. if the CUT is clocked), upsets in user memory elements would be overwritten by normal circuit operation. Given that the clock is paused, any difference

[1] There are two styles of readback: Readback and Readback Capture. During Readback, the configuration memory cells are read, including the current values on all user memory elements (LUT RAM, SRL, and BRAM). Readback Capture is a superset of Readback. In addition to reading all configuration memory cells, the current state of all internal CLB and IOB registers is read, storing all CLB and IOB register values into configuration memory cells. The register values are stored in the same configuration memory cell on which the corresponding register initial values have been programmed, thus sending the GRESTORE/GSR command to the FPGA configuration logic after the Readback Capture can cause registers to return to an unintended state [10].

observed in the BRAM, SRL and FFs contents compared to their initial values can be caused by either an SEU in the configuration bits or a single event transient (SET) in the global signals (e.g. clock or reset) of the corresponding DUT chains.

- All available slices, FFs, BRAMs and DSPs have been instantiated in the CUT as shown in Figure 28. Specifically:
- All slices are connected in long register chains (see Figure 29 and Figure 30).
    - The SLICEL LUTs are configured as route-through, the SLICEM LUTs are configured as 32-bit Shift Registers LUTs (SRLs) and LUT outputs are connected with CLB FFs to form long register chains.
    - The FFs are preloaded with alternate 0 and 1, while the SRLs with continuous 0s and 1s or alternate 0-1 patterns.
    - Register chain is not clocked (clock signal input is pulled down into the IOB), the CE signal of the FFs and SRLs is connected to '1' (pulled up into the IOB) and the FFs reset is configured to be synchronous and pulled-down into the IOB; this means that only transients in the clock tree are captured.
- All available BRAMs of the device are instantiated in the CUT (see Figure 31)
    - Initialized with a predefined pattern (to test SEUs in BRAMs), i.e. data are set to all 1s or all 0s or checkerboard values and parity bits to 0s.
    - Cascaded through the Data Bus either horizontal (raw) or vertical (column).
    - BRAM chain is not clocked and the WREN/RDEN signals of the BRAMs are connected to '0' (pulled-down into the IOB); this means that upsets in the BRAMs due to transients in the clock tree are unlikely to happen.
- All available DSP slices of the device are instantiated in the CUT (see Figure 32)
    - Connected in cascade mode either horizontal (raw) or vertical (column) -configurable
    - Configured to implement multiply and accumulate (MAC) operation
    - DSP chain is not clocked.

The outcome is a highly utilized and densely routed design (100% slice, BRAM and DSP utilization) (see Figure 33). The following Tables presents details about the configuration bitstream of the Test #1 CUT.

Table 1: Test #1 CUT - bits type details

Bit Type	Number of Bits	%
Configuration bits (unmasked)	18.031.484	55.74
CLB FF bits (masked)	106.400	0.33
CLB SRL bits (masked)	1.113.600	3.44
Unknown masked bits	146.980	0.46
BRAM bits (masked)	5.791.744	17.90
Other unused masked bits (PS area or dummy frames)	7.158.880	22.13
Total bits	32.349.088	100.00

Table 2: Test #1 CUT – initial bit values

Bit Value	Number of Bits	%
Configuration bits (unmasked) zeros	15.419.274	85.51
Configuration bits (unmasked) ones	2.612.210	14.49
Total Configuration bits (unmasked)	18.031.484	100.00
CLB FF bits (masked) zeros	53.200	50.00
CLB FF bits (masked) ones	53.200	50.00
Total CLB FF bits (masked)	106.400	100.00
CLB SRL bits (masked) zeros	610.000	50.00
CLB SRL bits (masked) ones	610.000	50.00
Total CLB SRL bits (masked)	1.366.980	100.00
BRAM bits (masked) zeros	3.661.824	63.22
BRAM bits (masked) ones	2.129.920	36.78
Total BRAM bits (masked)	5.791.744	100.00
Unknown & Unused masked bits zeros	7.305.860	100.00
Unknown & Unused masked bits ones	0	0.00
Total Unknown & Unused masked bits	7.305.860	100.00

Table 3: Test #1 CUT – Essential bits

Bit Type	Number of Bits	%
Configuration Essential bits	7.850.897	43.54
Configuration Non-Essential bits	10.180.587	56.46
Total Configuration bits	18.031.484	100.00
CLB FF Essential bits	106.400	100.00
CLB FF Non-Essential bits	0	0.00
Total CLB FF bits	106.400	100.00

* All the other bits (SRL, BRAM, unused) are non-essential bits

For test purposes an open-source platform has been developed [9] described in Section 6.3. The platform provides access to the FPGA configuration memory and circuit logic via the JTAG protocol. It provides a non-intrusive tool to perform various configuration memory functions, such as bitstream readback and verify, configuration frames/registers write and read, etc.

6.2 Test flow

The test #2 flow is shown in **Figure 4**. It performs the following steps:

1. The Zedboard is configured through the JTAG interface
2. The irradiation period starts. No configuration action (readback or scrubbing) is performed during the irradiation to avoid the injection of errors due to abnormal behavior of the configuration interface
3. The beam line signals the end of irradiation period and triggers the BRL1, which first reads back the CUT configuration memory through JTAG and records the readback data.
4. The BRL1 captures and reads back the device configuration memory and records the readback data. This step is used to record the content of user state elements (i.e. FFs); given that the CUT is paused during the irradiation and the reset is synchronous, any changes in their content

would be due to upsets or SETs in the clock signal. Readback capture uses the same process as readback but requires additional commands to be issued during the readback sequence to read the user state of the internal CLB registers

5. The Zedboard is reconfigured and a readback-verify command is performed to check for uncorrected upsets or JTAG malfunction (SEFI). In case of success, the test goes to step (2). Readback verify compares the user design bitstream against the readback data using the design's generated mask file (.msk/.msd). The mask file determines which components have dynamically changing values in the user's design and ignores them during the comparison. Examples of such components are CLBs and look-up tables (LUTs) that have been configured as distributed RAM or Shift Registers (SRLs).
6. In case of readback fail, a power cycle (power-off and power-on) is executed and the test goes to step (1).

All the configuration memory write and readback operations are performed by Vivado TCL scripts using JTAG commands.

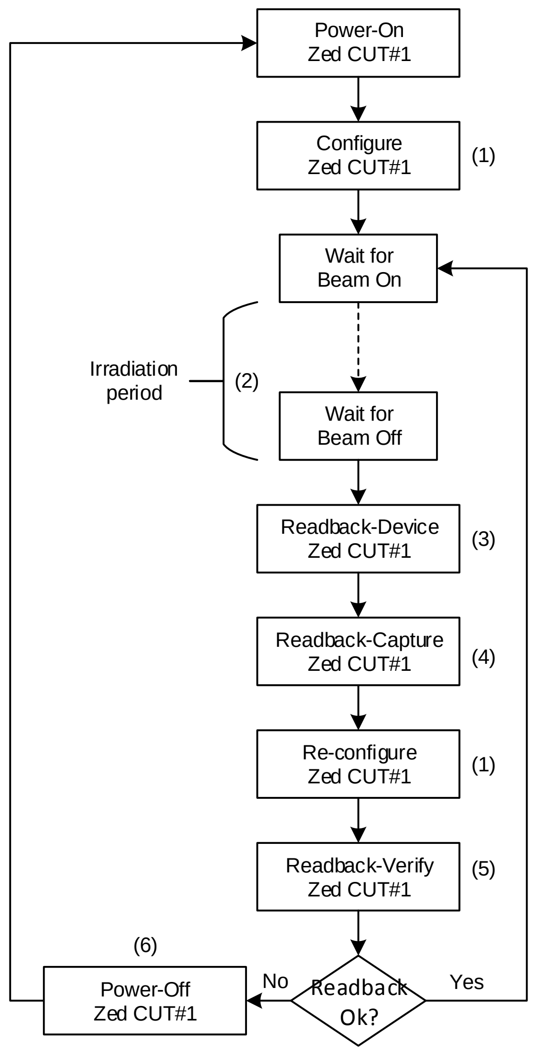


Figure 4: Test #1 flow

6.3 Test software

For the purposes of the radiation experiments, we developed an open-source framework that provides several FPGA memory configuration functions, such as fault injection, memory scrubbing, memory readback and configuration, etc. The proposed framework provides access to the FPGA CUT through the JTAG interface using Xilinx Vivado TCL commands.

Except its benefits as an open-source tool, it also provides a low-cost, non-intrusive solution since it does not require: i) dedicated hardware (e.g. extra boards) and ii) CUT modifications (only the minimum hardware logic is integrated within the target FPGA device to enable access to the configuration memory). Furthermore, due to its generic and open architecture, the framework is extendable (i.e. new FPGA families can be easily integrated in the list of the supported FPGA devices) and user friendly (i.e. the generic GUI enables the easy development of the target reliability applications).

The proposed framework provides a complete set of functions for accessing the FPGA configuration memory, and thus, to support the establishment of test environments, such as fault injection, radiation test monitoring, that need to monitor and modify the configuration memory. Meanwhile, it aims to simplicity, providing an out-of-the-box and cross-platform tool which can be modified and extended easily to support different FPGA devices without requiring any specialized hardware tools. The framework is composed of three major components as shown in **Figure 5**:

i. *on-chip logic*: provides access to the FPGA configuration memory and user logic. It is a custom logic that serves as an interface of the configuration memory and the user logic to the JTAG configuration port.

ii. *JTAG configuration engine (JTAG-CE)*: provides the implementation of a TCP server and low-level JTAG functions (in the form of TCL functions) for accessing the FPGA configuration memory/registers and the user logic through the Vivado tool.

iii. *High-level configuration functions (GUI)*: consists of the user application which supports widget implementation and the interface functionality (TCL interface handler) with the JTAG configuration engine.

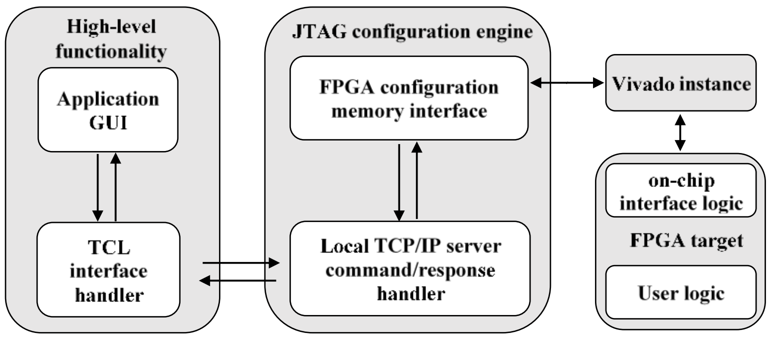


Figure 5: Test framework architecture

The target (user) application, such as a fault injection tool or a configuration memory scrubbing manager, is built on top of the framework. The proposed framework has been designed using the Qt and PySide2 frameworks. Qt is a well-known and highly-appreciated cross-platform application and User Interface (UI) development framework, while PySide2 is a Python binding for Qt which gives the opportunity to use the rich set of functionalities provided by the native implementation of Qt using the Python programming language.

The user application typically uses the Application Programming Interfaces (APIs) exposed by the TCL interface handler to communicate with the JTAG-CE in order to read/write device's configuration memory and configuration registers. Initially, the TCL interface handler starts a Vivado instance in batch mode and a TCP client. The TCP client is used as Inter-Process Communication (IPC) between the application thread and the JTAG-CE which runs in a separate thread as a TCP server. The use of TCP client-server scheme improves significantly the execution time overhead introduced by the Vivado instance. Instead of running a Vivado instance upon every script/command execution, the framework creates a single instance during the execution lifetime of the application, eliminating thus the overhead for setting-up the JTAG connection for every script. Notice that, in our experimental setup, the time required for initiating a JTAG connection is about 10 seconds including Vivado start-up, open the hardware target and connect to the Xilinx hardware server.

Furthermore, the TCP client-server solution, enables the development of multi-server and distributed approaches, where the application can communicate with multiple servers targeting either different FPGA devices (assuming this feature is enabled in the Vivado instance) or multiple user applications running in the same platform targeting the same FPGA device. For example, a multi-server scenario could be as follows: a fault injection tool and a memory scrubbing process are built on top of our framework and run in parallel for the same target FPGA. In this case, the scrubbing process executes continually or periodically to detect and correct any errors in the configuration memory of the device, while the fault injection tool introduces single or multiple faults in the FPGA configuration memory periodically or on-demand. Thus, the fault injection and the memory scrubbing tasks are performed asynchronously and independently allowing SEU mitigation techniques to be evaluated more effectively. The TCP client-server approach is illustrated in **Figure 6**.

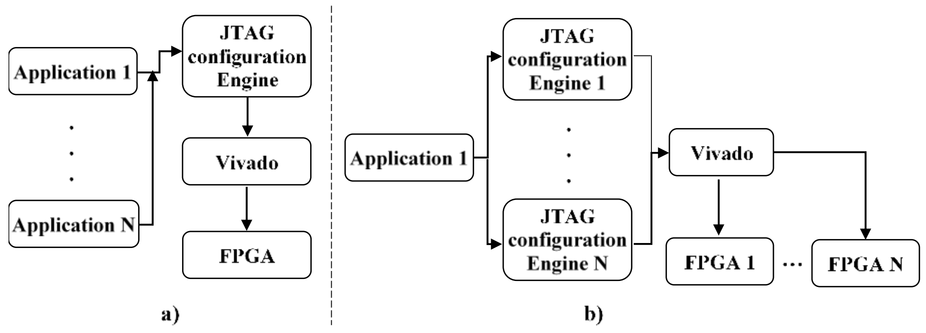

Application 1
Application N
JTAG configuration Engine
Vivado
FPGA
a)
Application 1
JTAG configuration Engine 1
JTAG configuration Engine N
Vivado
FPGA 1
FPGA N
b)

Figure 6: Using the TCP client-server approach: a) Multiple applications using a single JTAG configuration engine and b) Single application using multiple JTAG configuration engines

Although multiple applications and/or JTAG configuration engines (servers) may exist, only a single Vivado instance runs in the host machine to keep the resource allocation as low as possible since main memory requirements of Vivado may be high. This is not mandatory though but can be adjusted according to the final application requirements. As already mentioned, Vivado is the key component to handle the JTAG protocol of the target FPGA.

On-chip Logic

The framework communicates with the FPGA device through the standard boundary-scan JTAG port (IEEE standard 1149.1). The write and read to the configuration memory or the configuration registers are the most common JTAG operations. The JTAG also allows access to and from the internal FPGA logic for monitoring or debugging purposes. To activate a general-purpose communication port between the JTAG interface and the user design, one or more (up to 4 for the recent Xilinx FPGA device families) BSCAN primitive(s) must be instantiated while special JTAG USER instruction(s) provides access to the internal logic.

Three different alternatives of the on-chip logic have been implemented, based on the requirements of the target test environment (e.g. for fault injection or scrubbing or radiation testing monitoring):

i. *Basic setup (used in Test #1)*: This corresponds to a basic setup in order to enable write and read operations of the configuration frames (configuration memory) and the configuration registers, as shown in **Figure 7**. The monitoring of the user logic using the BSCAN primitive is optional and was not activated in Test #1. Tasks that can be performed by this setup include: full FPGA configuration, readback of the configuration memory and registers, fault injection to the configuration frame(s) and monitoring of the user logic (to detect erroneous behavior).

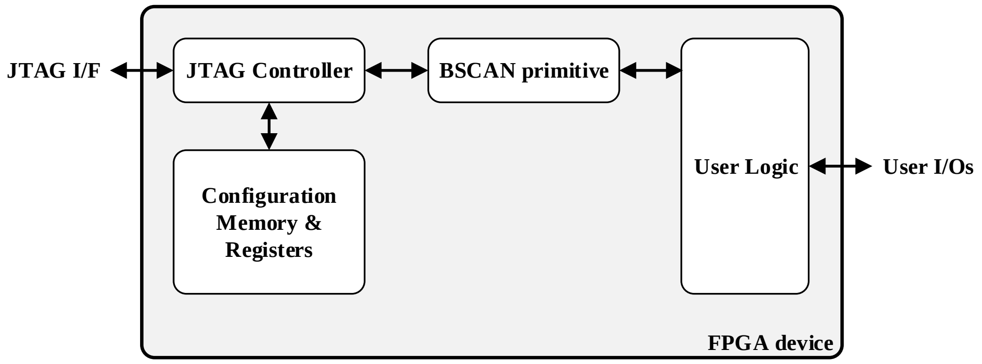


Figure 7: On-chip Logic - Basic setup

ii. *Frame ECC-based scrubbing setup*: This setup can be part of a configuration memory scrubber, which uses the embedded configuration frame-level ECC to detect SEUs (**Figure 8**). A FRAME ECC primitive provides access to the embedded ECC logic of the configuration frames. A FIFO retains the erroneous configuration frames (frame address, syndrome) detected by the FRAME ECC logic. The FIFO can be read through the BSCAN primitive by the host PC. An extra block named HeartBeat Logic stores the state of the FRAME ECC to enable the host PC to execute a watchdog process and detect malfunction of the frame ECC logic. For more information about the operation of the FRAME ECC and the heartbeat logic see the description of the on-chip logic of Test #2 CUT in Section 8.1.

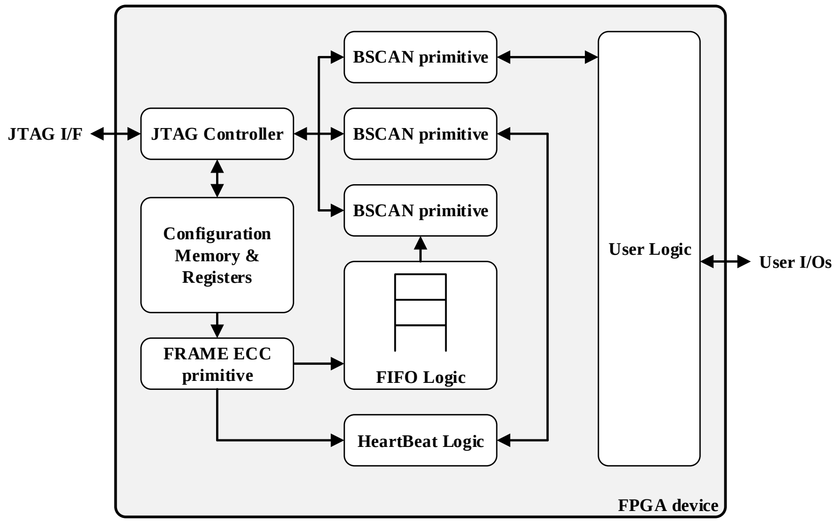


Figure 8: On-chip Logic – Frame ECC-based scrubbing setup

iii. *Hardened (TMR) version (used in Test #2)*: This setup version is an enhancement of setup (ii) for use in a radiation environment (e.g. radiation experiments). The FIFO and HeartBeat modules are triplicated and voted (in the fashion of triple modular redundancy – TMR), as shown in **Figure 9**. This scheme tolerates all the programmable resources of the on-chip logic against SEUs in order to provide a more robust and reliable setup under radiation conditions.

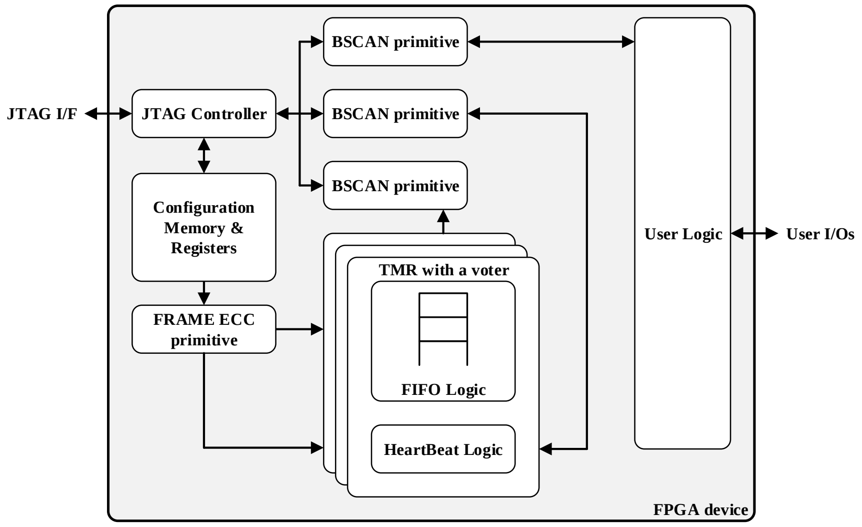


Figure 9: On-chip Logic – Hardened (TMR) version

<u>JTAG Configuration Engine (JTAG-CE)</u>

The low-level APIs provided by the JTAG configuration engine runs in a separate thread as TCP server listening in specific TCP port. The JTAG-CE accepts predefined commands from the target application and executes the associated low-level API while sending back a response to the application asynchronously. This means the user application should: i) send a dedicated command using the TCP client, ii) wait asynchronously for the response from the JTAG-CE component and iii) process the response as needed. **Table 4** describes the low-level APIs implemented in the JTAC-CE as well as the associated TCP commands.

Table 4: Low-level APIs for the configuration memory access

API	Parameters	Description
Configure	bit_filepath: bitstream file path mask_filepath: mask file path	Configures the FPGA device given a bitstream file and a maskfile (the maskfile is used when verification is issued; see ReadbackVerify API)
Readback	rdb_filepath: readback file path	Reads-back the FPGA device and saves its content in the target file
ReadbackCapture	rdb_filepath: readback file path	Reads-back the FPGA device in capture mode and saves its content in the target file
ReadbackVerify	-	Verifies the FPGA against the configured bitstream file while using also the maskfile (see Configure API)
RegisterWrite	register_address: target register address register_value: value to be written	Writes a configuration register
RegisterRead	register_address: target register address	Reads a configuration register
FrameWrite	frame_address: start address frame_filepath: file holding the frames data size: number of frames to write append_dummy_frame: True to append a dummy frame use_hex_format: True if frame data is in HEX reset_fifo: True to reset the internal FIFO	Writes a given number of frames to the configuration memory
FrameRead	readback_filepath: target file path (where data be saved) frame_address: start address size: number of frames to be read overwrite: True to overwrite the file	Reads a given number of configuration frames from the FPGA. The read data is saved in the target file.

<u>High-level Configuration Functions (GUI)</u>

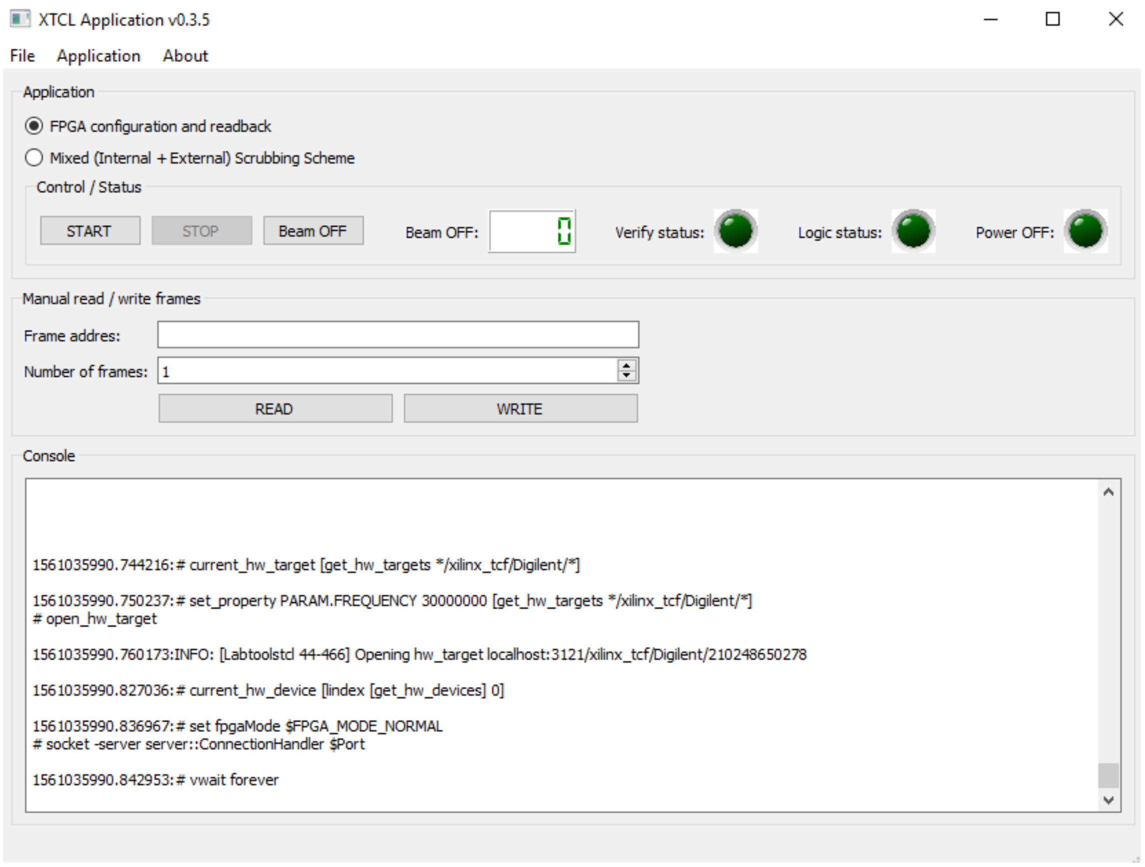
The High-level configuration functions include the user application and the TCL interface handler, which adds an abstraction layer between the user application and the TCP client commands. As mentioned earlier, the TCL interface includes a TCP client to communicate with the JTAG-CE by sending predefined commands and getting any response sent from the JTAG-CE. As in JTAG-CE, the TCL interface component can be easily modified and expanded to support new functions. This is a three-step procedure:

i. implement the low-level API inside JTAG-CE,

ii. map the implemented function with a new server command and

iii. implement the new high-level function inside the TCL interface (sending the mapped command to the TCP server).

Figure 10 shows the GUI used for Test #1.



```
1561035990.744216:# current_hw_target [get_hw_targets */xilinx_tcf/Digilent/*]

1561035990.750237:# set_property PARAM.FREQUENCY 30000000 [get_hw_targets */xilinx_tcf/Digilent/*]
# open_hw_target

1561035990.760173:INFO: [Labtoolstcl 44-466] Opening hw_target localhost:3121/xilinx_tcf/Digilent/210248650278

1561035990.827036:# current_hw_device [lindex [get_hw_devices] 0]

1561035990.836967:# set fpgaMode $FPGA_MODE_NORMAL
# socket -server server::ConnectionHandler $Port

1561035990.842953:# vwait forever
```

Figure 10: Test framework GUI for Test #1

The execution times of the FPGA configuration memory access functions (e.g. FPGA memory configuration, readback, readback capture and readback verify) used in both Test #1 and Test #2 are presented in **Table 5**.

Table 5: Execution time of API commands

API command	Execution time (sec)
configure	1.45
readback	4.10
readbackCapture	4.18
readbackVerify	14.11
writeFrames (1 frame)	0.060
readFrames (32 frames)	1.36
readRegister (ECC FIFO read, check HeartBeat)	0.020
writeRegister	0.016

7. Test #1 results

Different test sessions were performed as shown in **Table 6**:

- During the period 17.11.18 1:00-8:00 the board was not properly aligned. Thus, although the radiation effects of this period were analyzed, they were not considered for the calculation of cross section for the various embedded memories.
- Test #1 was performed for two different angles of incidence (θ), 0^o and 45^o obtaining two effective LETs of 8.8 and 12.45 MeVcm^2/mg. The two tests were comprised 890 and 126 runs; the duration of each run was around 40sec, 10sec the beam was active and 30sec inactive.

Table 6: Test #1 sessions

Duration		**Angle of incidence (θ)/ Effective LET** (MeVcm^2/mg)	**Runs**	**Fluence (ions/cm^2)**	**Comments**
Start (CET)	**End (CET)**				
17.11.18 1:00	17.11.18 8:00	0^o / 8.8	419	4.4×10^5	Test #1: Not properly aligned board
17.11.18 8:00	17.11.18 23:30	0^o / 8.8	890	9.5×10^5	Test #1
18.11.18 1:15	18.11.18 2:30	45^o / 12.45	126	0.9×10^5	Test #1 with angle

7.1 Post-processing software

The Zynq device is divided into two halves, the top and the bottom. All configuration frames in the device have a fixed, identical length of 3,232 bits (101 32-bit words). The configuration frame address is divided into five fields (**Table 7**).

Table 7: Configuration frame address fields

Address fields	Bit index	Description
Block Type	[25:23]	0: Interconnect & Block Configuration (CLB, IOB, DSP, CLK) and FF & LUTRAM values 1: BRAM content 2: CFG_CLB 3: Unknown type (a normal bitstream does not include this type) 4: Unknown type
Top/Bottom	[22]	0: Top 1: Bottom
Row Address	[21:17]	Defines a row of the device. The row addresses start at 0 and increment from center to top and from center to bottom
Column Address	[16:7]	Defines a column of the device. Column addresses start at 0 on the left and increase to the right
Minor Address	[6:0]	Defines a frame within a column

Either the configuration bit file or the readback file (generated using Xilinx Readback commands) includes the configuration frames with block type 0 and 1. The internal Readback CRC feature, which is used in the test #2, scans the configuration frames with block type 0, 2 and 3. Xilinx does not provide documentation for the block types 2, 3 and 4. In [11], it is mentioned that the CFG_CLB block type 2 defines which part of the FPGA needs to be reset or reconfigured. It only appears in a partial bitstream, if the RESET_AFTER_RECONFIG attribute has been set for this region.

For the SEU/MCU analysis, a database with all the erroneous bits was created comparing the readback and readback-capture files with the readback golden and readback-capture golden files, respectively. For each bit difference between a readback and readback-capture with the corresponding golden file, a database entry is created with the fields shown below.

Table 8: Test #1 Post-Processing Database fields

Database fields	Valid values	Description
Timestamp		The timestamp of the readback or readback-capture file
Readback Capture	False or True	False: readback True: readback-capture
Golden Bit value	0 or 1	Bit value of the golden file, e.g. if this bit is 0 then the bit is flipped from 0 to 1
Frame Address	32 bit hex value	Address of the configuration frame of the erroneous bit
Block Type	Interconnect & Block Configuration or Block RAM Content	The value of the frame address bits [25:23] 000: Interconnect & Block Configuration 001: BRAM Content These block types are only included in the readback/readback capture files
Top/Bottom	Top or Bottom	The value of the frame address bit [22]
Row Address		The value of the frame address bits [21:17]
Column Address		The value of the frame address bits [16:7]
Minor Address		The value of the frame address bits [6:0]
Word of frame	integer (0 to 100)	The word position in the frame of the erroneous bit
Bit of Word	integer (0 to 31)	The bit position in the word
Bit of Frame	integer (0 to 3231)	The bit position in the frame
Masked Bit	0 or 1	0: The bit is unmasked 1: The bit is masked
Masked Frame	False or True	False: The bit is located into a non-masked frame (at least one bit of the frame is unmasked) True: The bit is located into a masked frame (all the bits of the frame are masked). Such frames are a) the BRAM content (block type 1) frames, b) the dummy frames and the PS area frames of all the block types
Essential Bit	0 or 1	0: The bit is non-essential 1: The bit is essential
Non Essential Frame	False or True	False: The bit is located into a frame which has at least one essential bit True: The bit is located into a non-essential frame (all the bits of the frame are not essential)
Logic Block	String (SLICE_XnYm or RAMB36_XnYm)	In case of masked bit, this field identifies the specific logic block position (SLICE, RAMB)
Specific Logic	String (e.g. Latch=D5FF.Q or RAM=A:1)	In case of masked bit, this field identifies the specific logic of the above block (FF, SRL, BRAM bit)

All the results below are extracted from the database using queries.

7.2 SEU analysis

To analyze the SEUs we studied the effects in four different memory categories: CRAM, BRAM, FFs and SRLs.

Note: For the calculation of the cross sections only the results of the 2nd and 3rd test sessions of **Table 6** were considered (the 1st test session where the board was not properly aligned was ignored). However, the upsets occurred in the 1st test session were taken into account for the analysis of the MBUs (upsets in a frame) and the MCUs.

7.2.1 CRAM testing

CRAM category includes all the unmasked bits of the configuration bitstream plus the masked bits for the CLB FFs (initial FF values) and has been obtained by the analysis of the readback files. Since all these bits are static to the device operation, any upset is supposed to be an SEU (**Table 9**). The CRAM SEUs and the cross section are shown in **Table 10**.

We have also calculated the cross sections considering only the upsets in the essential CRAM bits, as extracted by the Xilinx ebd file, in order to provide a more realistic probability metric of SEUs affecting the CUT behavior.

Table 9: Test #1 CRAM Upsets in a frame

Upsets in a frame	CRAM Bits Occurrences	
	θ=0°	θ=45°
1	26072	2845
2	574	194
3	139	28
4	42	6
5		2

Table 10: Test #1 CRAM SEUs

		θ=0°	θ=45°
CRAM	SEUs	27805	3351
	Cross section [cm2/bit]	1.62×10^{-9}	2.09×10^{-9}
CRAM Essential	SEUs	14057	1822
	Cross section [cm2/bit]	0.82×10^{-9}	1.13×10^{-9}

7.2.2 BRAM testing

BRAM category include the masked bits of the configuration bitstream for the BRAM data and has been obtained by the analysis of the readback-capture data. Upsets in these bits are mostly due to SEUs. Given that the CUT clock is paused during the radiation experiments and the WREN/RDEN signals are ‘0’, upsets due to transients in the clock tree or the data busses are unlikely to happen. The address input signals and the data input and output signals of each BRAM of the BRAM chain are also set to ‘0’. The data output of each BRAM are initialized after configuration to ‘0’ irrespective of the value of the memory cells. The ECC feature is configured as “disabled”, so the parity cells can be used for additional data cells. The parity input signals are also set to ‘0’.

Table 11 presents the number of upsets in a frame and the number of occurrences and **Table 12** presents the total SEUs and the calculated cross section.

Table 11: Test #1 BRAM Upsets in a frame

Upsets in a frame	**BRAM Bits Occurrences**	
	θ=0°	**θ=45°**
1	12413	1124
2	48	5
8	256	
16	378	
17	6	
24	765	256
25	3	

Table 12: Test #1 BRAM SEUs

		θ=0°	**θ=45°**
BRAM	**Total SEUs**	39142	7278
	SEUs (1K, 2K, 3K per BRAM)	12509	1134
	Cross section [cm2/bit]	7.13×10^{-9}	14.15×10^{-9}

It was observed a phenomenon where 1K, 2K or 3K bits of the same RAM block were affected by a single event. The root cause of this phenomenon is currently investigated. In all these cases, the BRAM cells were preloaded with 1s and changed to 0, while the corresponding parity bits upset from 0 to 1.

In **Figure 11**, the 1K bits upsets case is described. On the left side of the figure, 128 consecutive configuration frames of BRAM content type are presented which corresponds to 10 BRAMs 36Kb of the same BRAM column. Each BRAM (green color of the figure) consists of 36864 bits (32768 data bits + 4096 parity bits) and it extends to the same 320 bits of all the 128 frames. The corresponding BRAM bits of each block RAM are shown on the right side of the figure. Each frame includes 256 BRAM data bits with their corresponding 32 parity bits. For example, in case of the second BRAM of a BRAM column, the BRAM bit 0 corresponds to the bit 320 of the configuration frame with frame address FA, the BRAM bit 64 corresponds to the bit 321 of the configuration frame FA, the BRAM bit 256 corresponds to the bit 320 of the configuration frame FA + 0x01 and so on. It must be noted that the frame FA includes the BRAM data bits [255:0] and the BRAM parity bits [31:0] (one parity bit for each byte), the second frame FA + 0x01 includes the BRAM data bits [511:256] and the BRAM parity bits [64:32], etc. So, two consecutive bits of a BRAM36 in the CRAM x-coordinate are located at a distance of 256 BRAM bits. In case of CRAM y-coordinate, two consecutive bits are located at a distance of 64 BRAM bits, but this applies for every 4 bits. The 1Kbits phenomenon (red color of the figure) affects specific 8 bits (e.g. bits

[327:320]) of all the 128 frames. These bits correspond to all BRAM data bits with distance of 32 bits (0, 32, 64, …, 32736).

In case of the 2K and 3K bits cases, the phenomenon is similar. Two or three blocks of 8 x 128 CRAM bits, which are not adjacent to one another, are affected. These bits belong to the same BRAM and they are either sequence of BRAM data bits with distance of 32 bits or sequence of BRAM parity bits with distance of 4 bits.

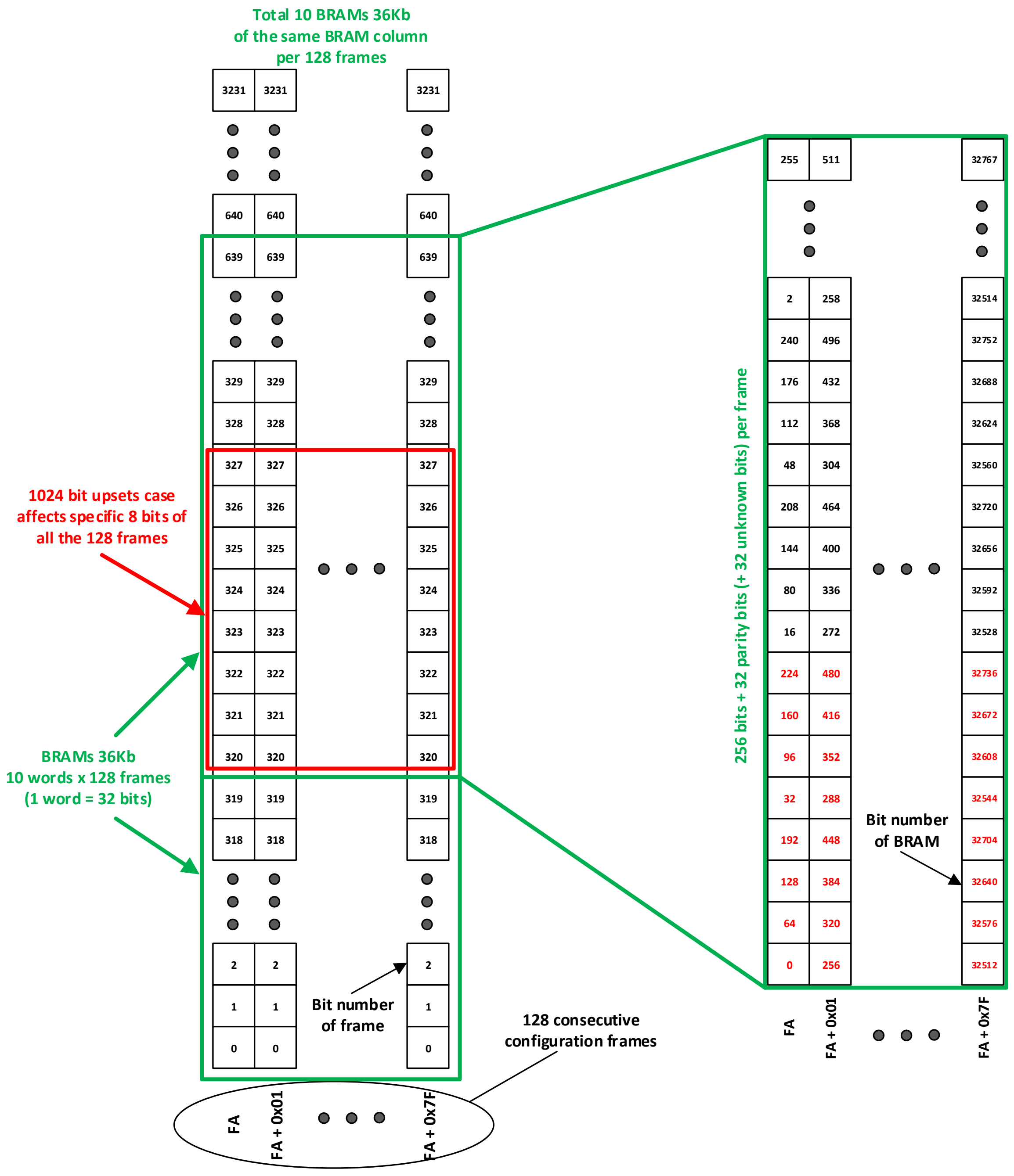


Figure 11: BRAM 1Kbits upsets

Table 13 presents the number of occurrences of this phenomenon and the specific BRAM bit upsets.

Table 13: 1K, 2K & 3K upsets in a BRAM

Upsets in a BRAM	BRAM data/parity bits	Bit flip type	Occurrences	
			θ=0°	θ=45°
1K	data bits 0, 32, 64, …, 32736	1 to 0	2	
	data bits 1, 33, 65, …, 32737	1 to 0	3	
2K	data bits 1, 33, 65, …, 32737 & parity bits 1, 5, 9, …, 4093	1 to 0 & 0 to 1	3	
3K	data bits 0, 32, 64, …, 32736 & data bits 8, 40, 72, …, 32744 & parity bits 0, 4, 8, …, 4092	1 to 0 & 1 to 0 & 0 to 1	5	
	data bits 1, 33, 65, …, 32737 & data bits 9, 41, 73, …, 32745 & parity bits 1, 5, 9, …, 4093	1 to 0 & 1 to 0 & 0 to 1	2	2

The cause of this finding is currently under investigation, given that we do not have information on the BRAM physical structure. In [12], it is mentioned that Single Event Multi-bit Upsets (SEMU) in SRAM occur when a particle strike upsets more than one cell in the memory. It is not necessary that every SEMU is Single-word Multi-bit Upset (SMU). SMU only occur when SEMU causes two or more upsets in a single data word.

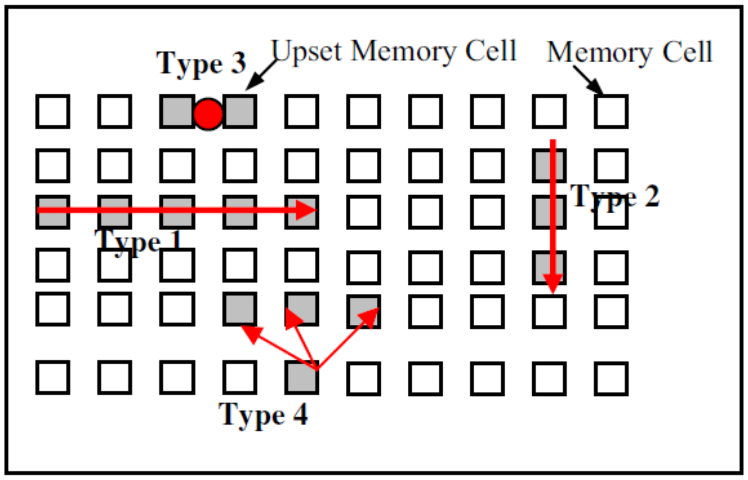


Figure 12: Types of Single Event Multiple Upsets in SRAM

Figure 12 shows some types of SEMU. Type 1 SEMU occurs when particle strike (red dot or arrows) passes through multiple adjacent cells. This type can cause SMU depending on memory structure.

For example, if adjacent affected cells belong to a single data word then it is SMU. In some memory architecture adjacent cells belong to distinct data words. Type 2 occurs when particles strike passes through multiple adjacent cells in a column, it causes only SEMU since it upsets one cell per row or data word. Type 3 arises when particle strike has an effect on two adjacent cells and it can be SMU if both the affected cells belong to same data word. Type 4 takes place when striking neutron interaction with silicon, oxygen or other atoms of the die generates multiple secondary particles. Again depending on the memory structure this type can also cause SMU.

7.2.3 Flip-Flop testing

FF and SRL categories include the masked bits of the configuration bitstream for the CLB FFs and shift register LUT data, respectively and have been obtained by the analysis of the readback-capture data. The CUT clock is also paused for the FFs and SRLs, so any upset in these bits compared to their initial values can be caused by either an SEU in the configuration bits or a single event transient (SET) on the clock net of the corresponding CUT chains (remember that all FFs and SRLs are cascaded in long chains).

Table 14: Test #1 FF + SRL Upsets in a frame

Upsets in a frame	FF Bits Occurrences		SRL Bits Occurrences	
	θ=0°	θ=45°	θ=0°	θ=45°
1	527	66	3938	483
2	3		36	6
3	3	1	27	4
4	69	10		
5		1		
8	87	16	10	
9	1			
11			1	
12	5		1	
16	10	1		
32			20	8
49				1
50				1
52				2
63			2	
64			24	2
200			1	
400	5			
1600			6	

In the case of FFs, these SET events were observed to affect several FFs: in most cases 8, 16 or 24 FFs were struck extending in one, two or three slices located in one or two CLBs. Similarly, in the case of SRLs, these SET events affects a large number of SRL LUT bits (i.e. in most cases 8, 128 and 256 bits) which are all located in the same slice. The number of upsets depends on the preloaded values of the chain. In three cases only, we observed a large number of upsets in the FFs (400 and 800 bits) and the SRLs (200, 3200 and 6400) extending in several slices that belong to adjacent physical half-row columns (**Table 14**). This observation is in accordance with the results presented in [6] regarding the transients in FF reset signals. However, in our experiments (remember that CE of the CLB chain is '1' and reset is configured as synchronous) these events are due to transients in the clock tree (and not in reset signals like in [6]). A SET event may strike a clock branch that drives one slice or one CLB or two adjacent half-row columns affecting the corresponding FFs and SRLs. Removing all the above cases from the upset lists, we calculated the SEUs and the cross section (**Table 15**).

Table 15: Test #1 FF + SRL Upsets & SEUs

		θ=0º	**θ=45º**
FFs	**Upsets**	3743	258
	SEUs	818	114
	Cross section [cm2/bit]	8.11×10^{-9}	12.06×10^{-9}
SRLs	**Upsets**	16296	1094
	SEUs	4091	507
	Cross section [cm2/bit]	3.87×10^{-9}	5.13×10^{-9}

The cross section of the different memory categories for the 2 LETs along with the error bars are shown in **Figure 13**. For the calculation of the error bars, we assumed a Poisson distribution of the SEUs, confidence level 95% and uncertainty on the measured fluence 10% [13],[14].

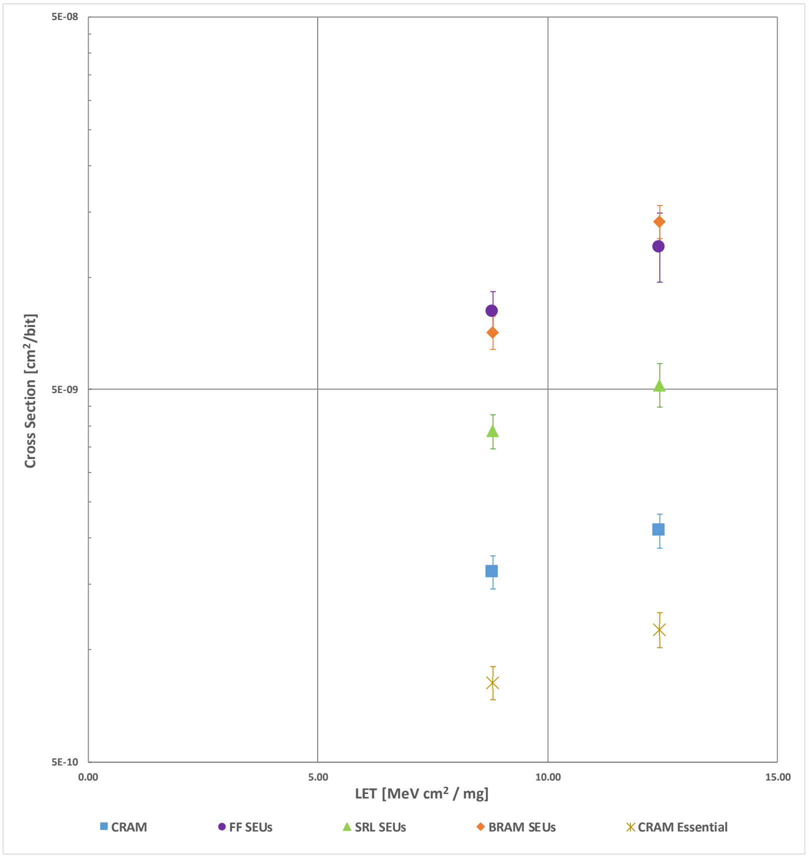


Figure 13: Test #1 cross section vs. LET

7.2.4 Initial values analysis

Table 16 presents: a) the percentage of the occurrences of the bit flip 0 to 1 and 1 to 0 for all the memory categories (columns 3 and 5) and b) the ratios of the number of 0-to-1 (1-to-0) upsets/number of bits preloaded with 0 (1) (columns 4 and 6). For the calculation of the ratios, we took into account the percentage of the initial values for each memory category as shown in **Table 2**.

Table 16: Test #1 SEU according to bit flip type

Memory Category	Bit Flip Type	θ=0°		θ=45°	
		% occurrences	Upsets per preloaded bits ratio	% occurrences	Upsets per preloaded bits ratio
CRAM	**0 to 1**	82.08%	1.69×10^{-3}	78.37%	0.16×10^{-3}
	1 to 0	17.92%	2.17×10^{-3}	21.63%	0.27×10^{-3}
BRAM	**0 to 1**	39.60%	5.02×10^{-3}	37.44%	0.74×10^{-3}
	1 to 0	60.40%	13.16×10^{-3}	62.56%	2.13×10^{-3}
FFs	**0 to 1**	36.10%	6.54×10^{-3}	18.42%	0.39×10^{-3}
	1 to 0	63.90%	11.57×10^{-3}	81.58%	1.75×10^{-3}
SRLs	**0 to 1**	51.51%	4.11×10^{-3}	51.08%	0.41×10^{-3}
	1 to 0	48.49%	3.87×10^{-3}	48.92%	0.42×10^{-3}

Observing the results of the above Table we can conclude (**Table 17**) that:

i. for the cases of CRAM and SRLs the likelihood to occur upset in memory cells preloaded with ‘0’ is approximately equal with those preloaded with ‘1’, as also observed in [6] and

ii. for the cases of BRAMs, the ‘1’ cells upset more frequently (~2.5-3.0X) compared to the ‘0’ cells.

iii. for the cases of FFs, the likelihood of 1->0 upset seems to be higher, while the results differ significantly for the two different LETs. This may be due to statistical error given the fact that the number of upsets in FFs is low (258 in 45°).

Table 17: Test #1 likelihood to upset 0->1 : 1->0

Memory Category	Likelihood to upset 0->1 : 1-> 0	
	θ=0°	θ=45°
CRAM	1 : 1.3	1 : 1.7
BRAM	1 : 2.6	1 : 2.9
FFs	1 : 1.8	1 : 4.5
SRLs	1 : 1	1 : 1

7.3 MBU/MCU analysis

Table 18 presents the number of bit upsets occurred in a configuration frame in a single radiation period (in one readback file) for both angles of incidence (θ), 0^o and 45^o (sum of the CRAM upsets in a frame of the **Table 9**); we can assume that when two or more upsets are observed in a frame, they are multiple bit upsets (MBUs), i.e. caused by a single particle. The results from both effective LETs are considered. Although most cases are single bit upsets (SBUs), there is a significant number of MBUs (mainly 2-bit upsets per frame). These upsets cannot be corrected by the embedded error correction code (ECC) of the Xilinx Zynq-7000 devices and require a scrubbing scheme executing a more complex error correction mechanism.

Table 18: Test #1 multiple upsets in a frame

Upsets in a frame	1	2	3	4	5
# of Occurrences	28917	768	167	48	2

Moreover, we observed multiple upsets expanding in more than one (neighboring) configuration frames called Multiple Cell Upsets (MCUs). We adopted the approach proposed in [15] to identify the MCUs in different frames.

The MCU extraction technique uses the radiation test data and the dimensions of the Configuration Memory array to determine statistically which events in the test data are MCUs. This process involves the following steps:

- Collect upset data from static radiation testing and organize them into logical addresses. To perform physical adjacency analysis, the configuration bits are represented by two dimensional coordinates, the x-coordinate represents the consecutive configuration frames (continuous frame addresses) and the y-coordinate the number of bits in the frame.
- Determine the hamming distances between all upset pairs of an irradiation period (e.g. let's assume that for a specific readback file, N upsets have been observed; then we calculate the hamming distances of all the N*(N-1)/2 pairs). To reduce the number of upset pairs (and thus the complexity of the algorithm), a subset of upset pairs based on the hamming distance can be explored (e.g. only pairs with hamming distance less than 10) assuming that the physical adjacency is most likely in configuration bits with relatively close logical addresses (frame numbers and bit numbers). However, here, the search algorithm is not limited by the hamming distances.
- Create physical adjacency model from statistical data. The algorithm is based on the assumption that if a particular upset pair (hamming distance) appears several times this is an evidence of physical adjacency and thus it is very likely that the upset pair is an MCU and not the result of two different SEUs. Two bits are classified as physically adjacent when their specific hamming distance has been observed several times, i.e. more than a specific threshold (e.g. 5).
- Extract MCUs using physical adjacency model. After the physical adjacency model has been identified, all upset pairs within individual irradiation runs, that their hamming distance has been marked in the previous step, are classified as 2-bit MCUs. The process continues iteratively to create larger MCUs: when two MCUs have a common bit then their union is also considered an MCU.

Figure 14 presents the patterns (shapes) of the MCUs and their frequency of occurrence. The x-axis of the shapes represents consecutive frame address, while the y-axis consecutive bits in a frame. The total number of SBUs are slightly higher than the number of MCUs. The second and fourth shapes of the first row prove the bit interleaving scheme adopted in the FPGA configuration memory [15]. For example, the hamming distance of the two upsets of the second shape is (1,1), i.e. the first upset is in the i-bit of the frame j and the second upset is in the i+1-bit of the frame j+1. Note that in the above analysis, only the unmasked bits of the configuration memory have been considered, i.e. that are not dynamically altered by the CUT and possible upsets in these bits can be detected and corrected by a scrubbing mechanism.

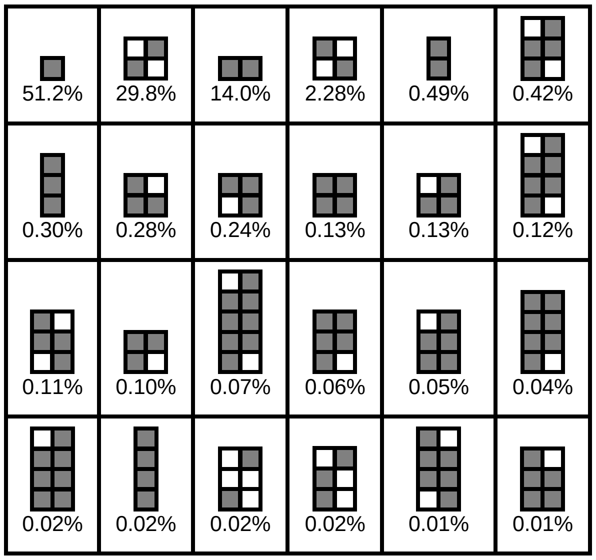


Figure 14: Test #1 MCU patterns (unmasked bits) – angle 0^o

Figure 15 presents the MCUs patterns in case of masked bits. The occurrence of these patterns depend on the CUT, because the masked bits of the configuration memory corresponds to the LUTs used to shape the SRLs and the configuration bits used to preload the FFs.

Given that for the FFs, SRLs and BRAMs masked configuration bits, the slice/LUT position can be extracted (see fields Logic block and Specific Block of the upsets database in Section 7.1), it is possible to correlate the hamming distances (from the frame/bit logical addresses) of **Figure 15** with the bit positions of SRLs and BRAMs. For example, the relation between the bits of four consecutive configuration frames containing the LUTs of a SLICEM and the bits of the four corresponding SRLs is shown in **Figure 16**. Specifically, it shows how the bits 0 to 63 of four consecutive configuration frames (address offsets 0x0, 0x1, 0x2 and 0x3) form four 64-bit SRLs (LUTA, LUTB, LUTC and LUTD). The observation is that the first, third and fourth MCU shapes of **Figure 15** may occur in two contiguous bits of an SRL (bits of **Figure 16** with red color). For example, the MCU of the first shape may affect the bits 3 and 4 of the LUTA SRL, the MCU of the third shape may affect the bits 31 and 32 of the LUTA SRL and the MCU of the fourth shape may affect the bits 23 and 24 of the LUTB SRL. Also, the MCU of the last shape of **Figure 15** may affect the same bit position of three contiguous SRLs (LUTB, LUTC and LUTD shown by yellow color).

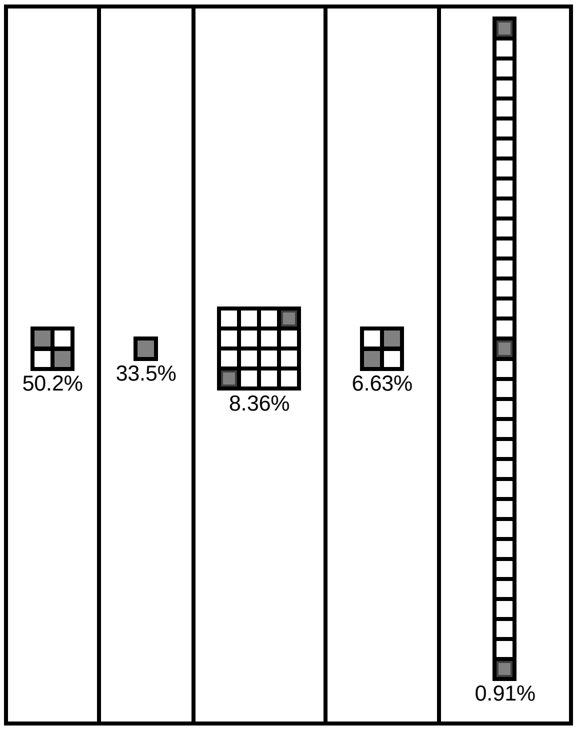


Figure 15: Test #1 MCU patterns (masked bits) – angle 0°

consecutive frames →

bit of frame ↓

	FA	FA + 0x1	FA + 0x2	FA + 0x3
Bit 0	LUTA BIT1	LUTA BIT0	LUTA BIT9	LUTA BIT8
Bit 1	LUTA BIT3	LUTA BIT2	LUTA BIT11	LUTA BIT10
Bit 2	LUTA BIT5	LUTA BIT4	LUTA BIT13	LUTA BIT12
Bit 3	LUTA BIT7	LUTA BIT6	LUTA BIT15	LUTA BIT14
Bit 4	LUTA BIT17	LUTA BIT16	LUTA BIT25	LUTA BIT24
Bit 5	LUTA BIT19	LUTA BIT18	LUTA BIT27	LUTA BIT26
Bit 6	LUTA BIT21	LUTA BIT20	LUTA BIT29	LUTA BIT28
Bit 7	LUTA BIT23	LUTA BIT22	LUTA BIT31	LUTA BIT30
Bit 8	LUTA BIT33	LUTA BIT32	LUTA BIT41	LUTA BIT40
Bit 9	LUTA BIT35	LUTA BIT34	LUTA BIT43	LUTA BIT42
Bit 10	LUTA BIT37	LUTA BIT36	LUTA BIT45	LUTA BIT44
Bit 11	LUTA BIT39	LUTA BIT38	LUTA BIT47	LUTA BIT46
Bit 12	LUTA BIT49	LUTA BIT48	LUTA BIT57	LUTA BIT56
Bit 13	LUTA BIT51	LUTA BIT50	LUTA BIT59	LUTA BIT58
Bit 14	LUTA BIT53	LUTA BIT52	LUTA BIT61	LUTA BIT60
Bit 15	LUTA BIT55	LUTA BIT54	LUTA BIT63	LUTA BIT62

	FA	FA + 0x1	FA + 0x2	FA + 0x3
Bit 16	LUTB BIT1	LUTB BIT0	LUTB BIT9	LUTB BIT8
Bit 17	LUTB BIT3	LUTB BIT2	LUTB BIT11	LUTB BIT10
Bit 18	LUTB BIT5	LUTB BIT4	LUTB BIT13	LUTB BIT12
Bit 19	LUTB BIT7	LUTB BIT6	LUTB BIT15	LUTB BIT14
Bit 20	LUTB BIT17	LUTB BIT16	LUTB BIT25	LUTB BIT24
Bit 21	LUTB BIT19	LUTB BIT18	LUTB BIT27	LUTB BIT26
Bit 22	LUTB BIT21	LUTB BIT20	LUTB BIT29	LUTB BIT28
Bit 23	LUTB BIT23	LUTB BIT22	LUTB BIT31	LUTB BIT30
Bit 24	LUTB BIT33	LUTB BIT32	LUTB BIT41	LUTB BIT40
Bit 25	LUTB BIT35	LUTB BIT34	LUTB BIT43	LUTB BIT42
Bit 26	LUTB BIT37	LUTB BIT36	LUTB BIT45	LUTB BIT44
Bit 27	LUTB BIT39	LUTB BIT38	LUTB BIT47	LUTB BIT46
Bit 28	LUTB BIT49	LUTB BIT48	LUTB BIT57	LUTB BIT56
Bit 29	LUTB BIT51	LUTB BIT50	LUTB BIT59	LUTB BIT58
Bit 30	LUTB BIT53	LUTB BIT52	LUTB BIT61	LUTB BIT60
Bit 31	LUTB BIT55	LUTB BIT54	LUTB BIT63	LUTB BIT62

	FA	FA + 0x1	FA + 0x2	FA + 0x3
Bit 32	LUTC BIT1	LUTC BIT0	LUTC BIT9	LUTC BIT8
Bit 33	LUTC BIT3	LUTC BIT2	LUTC BIT11	LUTC BIT10
Bit 34	LUTC BIT5	LUTC BIT4	LUTC BIT13	LUTC BIT12
Bit 35	LUTC BIT7	LUTC BIT6	LUTC BIT15	LUTC BIT14
Bit 36	LUTC BIT17	LUTC BIT16	LUTC BIT25	LUTC BIT24
Bit 37	LUTC BIT19	LUTC BIT18	LUTC BIT27	LUTC BIT26
Bit 38	LUTC BIT21	LUTC BIT20	LUTC BIT29	LUTC BIT28
Bit 39	LUTC BIT23	LUTC BIT22	LUTC BIT31	LUTC BIT30
Bit 40	LUTC BIT33	LUTC BIT32	LUTC BIT41	LUTC BIT40
Bit 41	LUTC BIT35	LUTC BIT34	LUTC BIT43	LUTC BIT42
Bit 42	LUTC BIT37	LUTC BIT36	LUTC BIT45	LUTC BIT44
Bit 43	LUTC BIT39	LUTC BIT38	LUTC BIT47	LUTC BIT46
Bit 44	LUTC BIT49	LUTC BIT48	LUTC BIT57	LUTC BIT56
Bit 45	LUTC BIT51	LUTC BIT50	LUTC BIT59	LUTC BIT58
Bit 46	LUTC BIT53	LUTC BIT52	LUTC BIT61	LUTC BIT60
Bit 47	LUTC BIT55	LUTC BIT54	LUTC BIT63	LUTC BIT62

	FA	FA + 0x1	FA + 0x2	FA + 0x3
Bit 48	LUTD BIT1	LUTD BIT0	LUTD BIT9	LUTD BIT8
Bit 49	LUTD BIT3	LUTD BIT2	LUTD BIT11	LUTD BIT10
Bit 50	LUTD BIT5	LUTD BIT4	LUTD BIT13	LUTD BIT12
Bit 51	LUTD BIT7	LUTD BIT6	LUTD BIT15	LUTD BIT14
Bit 52	LUTD BIT17	LUTD BIT16	LUTD BIT25	LUTD BIT24
Bit 53	LUTD BIT19	LUTD BIT18	LUTD BIT27	LUTD BIT26
Bit 54	LUTD BIT21	LUTD BIT20	LUTD BIT29	LUTD BIT28
Bit 55	LUTD BIT23	LUTD BIT22	LUTD BIT31	LUTD BIT30
Bit 56	LUTD BIT33	LUTD BIT32	LUTD BIT41	LUTD BIT40
Bit 57	LUTD BIT35	LUTD BIT34	LUTD BIT43	LUTD BIT42
Bit 58	LUTD BIT37	LUTD BIT36	LUTD BIT45	LUTD BIT44
Bit 59	LUTD BIT39	LUTD BIT38	LUTD BIT47	LUTD BIT46
Bit 60	LUTD BIT49	LUTD BIT48	LUTD BIT57	LUTD BIT56
Bit 61	LUTD BIT51	LUTD BIT50	LUTD BIT59	LUTD BIT58
Bit 62	LUTD BIT53	LUTD BIT52	LUTD BIT61	LUTD BIT60
Bit 63	LUTD BIT55	LUTD BIT54	LUTD BIT63	LUTD BIT62

Figure 16: SRLs (LUTRAMs) – Configuration Memory mapping

In case of BRAM, the MCU patterns are shown below. There are a few more infrequent MCU shapes (<0.05%) not presented here, including the phenomenon described in **Figure 11**. The MCU of the second shape below may affect two bits of a BRAM36 located at a distance of 256 bits, according to the BRAM bits location shown in **Figure 11**.

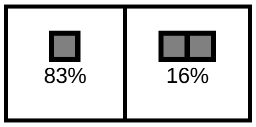


Figure 17: Test #1 BRAM MCU patterns – angle 0°

Moreover, we observed that the shapes of the MCUs for the two different angles of incidence (θ), 0° and 45° are the same, comparing the **Figure 14**, **Figure 15** and **Figure 17** with the **Figure 18**, **Figure 19** and **Figure 20** respectively. The basic difference is that the number of single unmasked bits upsets (SBU) for θ = 45° is less than the number of SBUs for θ = 0°.

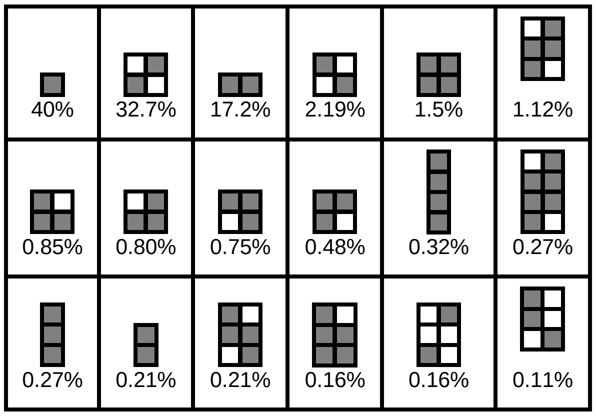


Figure 18: Test #1 MCU patterns (unmasked bits) – angle 45°

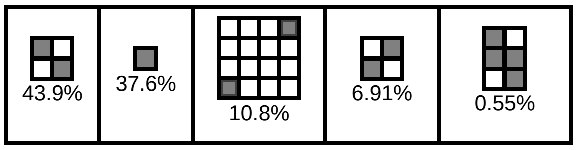


Figure 19: Test #1 MCU patterns (masked bits) – angle 45°

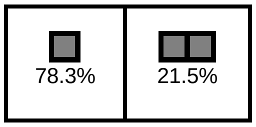


Figure 20: Test #1 BRAM MCU patterns – angle 45°

7.4 SEFI results

In general, there are two main types of SEFI (Single-Event Functional Interrupt) depending on the actions required to restore operability [16]: reset by software or power cycling. Both SEFI types were observed in our experiment: the first SEFI type occurred once after a readback-verify command (**Figure 4**) and we re-run the test software to restore operability while the second SEFI type occurred three times and we powered-off/on the DUT. The two power-cycle SEFI cases occurred after the readback-verify command and the third one during readback command.

8. Test #2: Configuration memory scrubbing

The goal of this test is to evaluate a configuration memory scrubbing approach for the Xilinx Zynq-7000 devices. The approach combines the embedded Error Correction Code of the configuration memory frames and an inter-frame, interleaved parity scheme to form a mixed two-dimensional (2-D) error correction code. The 2-D coding scheme detects and corrects single and multiple bit upsets in the configuration memory of the Xilinx Zynq-7000 FPGA device.

Several Single Event Upsets (SEUs) mitigation approaches have been proposed in the past for SRAM-based FPGAs. Most of these usually combine redundancy techniques for error detection and masking with memory scrubbing to correct upsets in the configuration memory. Scrubbing is based on scanning the configuration memory, detecting upsets and rewriting the affected configuration frames with the correct data. In many cases, the scrubbing mechanism is supported by embedded Error Correction Codes (ECCs). Most Xilinx FPGA device families are protected by frame-level ECC and device-level Cyclic Redundancy Check (CRC). Frame-level ECC is a typical single-error correction, double error detection (SECDED) Hamming code, while global CRC algorithm supports the error detection of MCUs but without locating the errors; in the case of device-level CRC activation, the configuration memory is fully reprogrammed to correct the erroneous frame(s). The Xilinx Soft Error Mitigation (SEM) IP [17] core incorporates a frame-level CRC scheme that enables the correction of double bit adjacent errors. The hybrid scrubbing approach proposed in [6] combines the aforementioned internal ECC-based scrubbing mechanism with an external scrubber. The internal scrubber continuously scans the FPGA configuration frames and corrects single-bit errors or detects multi-bit errors per frame. Multi-bit errors are repaired by an external scrubbing mechanism that identifies the frames with multiple-bit errors by monitoring the internal scrubber, derives the golden configuration frame data from an external storage medium and re-writes these frames through the JTAG port. Both approaches rely on an external expensive radhard memory to store the golden configuration bitstream which must be accessed in the case of multi-bit errors increasing significantly the error correction latency.

Alternatively, efficient ECC schemes [18]-[22] have been proposed to improve the FPGA correction capabilities for multi-bit errors. In [18], a two-dimensional (2-D) Hamming code is employed to correct multi-bit errors in a configuration frame; 2-D ECCs are applied to each frame arranged as a 2-D matrix. In [19], a bit-interleaved hamming scheme is proposed to correct single-bit upsets (SBUs) and multiple-bit upsets (MBUs); it uniformly interleaves the bits of each frame into a number of sub-frames and uses the non-essential bits of the frames to store the parity bits. In [20] a parity scheme, called stepped parity, is presented, where each configuration bit is associated with several parity bits so as the most common MCU patterns can be detected by at least one parity bit. In [21], an interleaved parity code is combined with erasure codes to protect the FPGA configuration memory from MCUs. The parameters of the coding scheme are tuned for the specific FPGA technology, i.e. the interleaving distance is determined based on actual MCU patterns, while the frames are divided into clusters in a way to enable the correction of MCUs expanding in adjacent configuration frames. Finally, in [22] the authors use specific parity equations (Modified matrix code) and Erasure codes (EVENODD) to improve the error correction capability of [21]. All the above approaches provide efficient coding schemes for the protection of the FPGA configuration memory against both SBUs and MCUs but without exploiting the embedded ECC scheme of the Xilinx FPGA devices.

In this test, we combine the features of the hybrid scrubbing [6] and the 2-D coding schemes [20]-[22] to provide a low-cost scrubbing solution for the configuration memory of the Xilinx Zynq-7000 FPGA. Our approach:

i) relies on the embedded ECC scheme to detect SBUs and MCUs; the 2-D coding schemes proposed in previous works [18]-[22] provide better error detection capabilities compared to the internal ECC imposing significant area overhead and

ii) uses an external scrubber connected to the FPGA device through the JTAG port that corrects both SBUs and MCUs based on an interframe, interleaved parity code.

Our scrubbing approach consists of two components: the on-chip scrubber and the external error correction mechanism which communicates with the on-chip logic through the standard boundary-scan JTAG port (IEEE standard 1149.1). The on-chip logic is based on the built-in 7-series Readback CRC to detect the erroneous configuration frames while the error correction mechanism uses a two-dimensional (2-D) coding scheme to detect and correct the upsets in these frames.

8.1 Circuit under test (CUT)

The CUT is similar with the synthetic benchmark used in Test #1, with the difference that the CUT is clocked.

- Consists of FF, BRAM and DSP chains as described for the Test #1 CUT
    - A specific region is left unused (no chain is allocated in this region) to host the internal scrubber
- The FPGA CUT communicates with the BRL2 via BSCANE2 primitives and JTAG interface (similar to Test #1) as shown in Figure 34.

The on-chip scrubbing logic, as shown in **Figure 21**, uses the Readback CRC feature in conjunction with the FRAME_ECC primitive to gain access to the internal ECC logic. The Readback CRC is set to the CONTINUE mode in order to detect the presence of upsets in the configuration memory. It performs readbacks continuously and calculates the frame-level ECC syndromes and the global CRC. The FRAME_ECC provides the frame-level ECC error flag, the syndrome, the current frame address and the CRC error flag. Note that the FRAME_ECC does not correct the detected upsets.

A FIFO is used to store the FRAME_ECC outputs when an ECC or CRC error occurs. The external scrubber (in our experiment it runs on the host laptop) reads the FIFO data and monitors the status of the FRAME_ECC. BSCAN primitives are instantiated to provide general-purpose communication ports between the JTAG interface and the internal logic. The external scrubber reads the FIFO periodically and in case of error (e.g. one or more configuration frames have been stored in the FIFO by the FRAME_ECC), initiates the error correction process described in the next subsection.

HeartBeat logic stores the state of the FRAME_ECC at the end of each device scan cycle. A Flip-Flop Register is set to '1', when the Readback CRC mechanism reads the last configuration frame and it is auto-cleared when it is read through the JTAG interface. This enables the external scrubber to perform a watchdog process in order to alarm the malfunction of the frame ECC. An extra logic named Active Status Logic is also used, as shown in **Figure 35**, in order to monitor the status of the FF, BRAM and DSP chains. Specifically, the FF chain is driven by a toggle bit logic and the Active Status Logic stores the toggle action at the end of the chain. The Active Status Logic flag is auto-cleared when it is read through the JTAG interface. The BRAM and DSP chains are also monitored using similar logic. The external scrubber accesses periodically the Active Status Logic flags through the BSCAN primitive in order to perform a watchdog functionality. Reading a zero flag means that the corresponding chain does not operate correctly. Errors are logged into a file and a LED is also set in the test sw GUI.

The FIFO and HeartBeat modules are triplicated and voted in order to tolerate all the programmable resources of the on-chip logic against SEUs and provide a more robust and reliable setup under radiation conditions. Notice that separate BSCAN primitives are connected into the different resources (i.e. user logic, FIFO logic, Heartbeat logic) instead of multiplexing all into a single primitive in order to simplify the routing between the BSCAN and the corresponding components and, thus, facilitate the implementation of the Isolation Design Flow typically used for the design of hardened circuits.

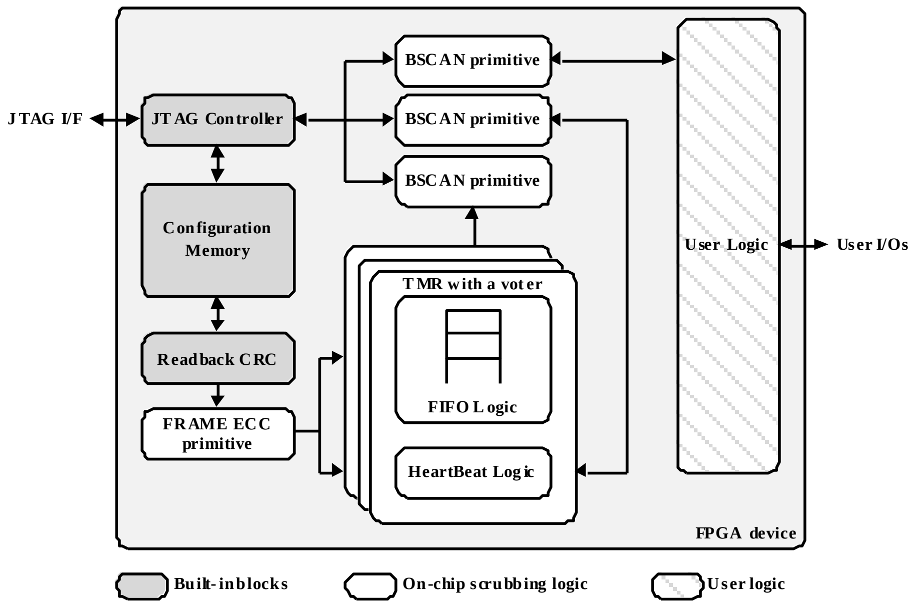


Figure 21: Test #2 on-chip logic

As mentioned above, the Readback CRC is set to the CONTINUE mode. In this mode, the global LUT mask (GLUTMASK) configuration option is enabled in order to mask changeable memory cell readback LUTRAM value. By design, certain configuration memory bits can change value during design operation. This is frequently the case where logic slice resources are configured to implement LUTRAM functions such as Distributed RAM or Shift Registers (SRL). It also occurs when other resource types with Dynamic Reconfiguration Ports are updated during design operation. The memory bits associated with these resources are masked so that they are excluded from CRC and ECC calculations to prevent false error detections. Xilinx FPGA devices implement configuration memory masking to prevent these false error detections through GLUTMASK. The masked bits are no longer monitored by the controller. Configuration memory reads of bits associated with masked resources return constant values (either logic one or logic zero). This prevents false error detections.

The following Tables presents details about the configuration bitstream of the Test #2 CUT.

Table 19: Test #2 CUT - bits type details

Bit Type	Number of Bits	%
Configuration bits (unmasked)	19.257.500	55.74
CLB FF bits (masked)	85.740	0.33
CLB SRL bits (masked)	898.368	3.44
Unknown masked bits	152.184	0.46
BRAM bits (masked)	4.796.416	17.90
Other unused masked bits (PS area or dummy frames)	7.158.880	22.13
Total bits	32.349.088	100.00

Table 20: Test #2 CUT – Configuration bit initial values

Bit Value	Number of Bits	%
Configuration bits (unmasked) zeros	17.136.982	88.99
Configuration bits (unmasked) ones	2.120.518	11.01
Total Configuration bits (unmasked)	19.257.500	100.00

Table 21: Test #2 CUT – Essential bits

Bit Type	Number of Bits	%
Configuration Essential bits	6.394.118	33.20
Configuration Non-Essential bits	12.863.382	66.80
Total Configuration bits	19.257.500	100.00
CLB FF Essential bits	85.740	80.58
CLB FF Non-Essential bits	20.660	19.42
Total CLB FF bits	106.400	100.00

* All the other bits (SRL, BRAM, unused) are non-essential bits

8.2 Error correction algorithm

Considering the configuration memory as a 2-D matrix, i.e. configuration frames are the columns of the matrix, our error correction algorithm relies on a 2-D coding scheme consisting of the internal frame-level ECC code (vertical direction) and an interframe, interleaved parity code (horizontal direction). Given that the internal ECC is a SECDED code, it is able to detect per configuration frame (column) all SBUs, all odd-numbered MBUs, all double MBUs and the vast majority of even-numbered MBUs (detection is not guaranteed in rare circumstances). In the case of SBUs, the syndrome indicates the precise location of the errors within the frame, otherwise just indicates the existence of errors.

Regarding the (horizontal) parity code, the configuration frames of the FPGA device are divided into groups as in [21]. Each group *G* contains *N* frames, $R_0, R_1, \ldots, R_{N-1}$ which are further divided into two or more subgroups in a interleaved manner. In our experiments, each group contains 32 (*N*) frames and is divided into two subgroups, $G_A = \{R_0, R_2, \ldots, R_{30}\}$ and $G_B = \{R_1, R_3, \ldots, R_{31}\}$. However, we are currently investigating how these parameters (the group size and the number of subgroups) affect the scrubbing performance (correction coverage, correction latency and storage requirements). All the frames (columns) of each subgroup are XORed to calculate a parity frame, e.g. PG_A and PG_B. The rationale behind dividing each group into two subsets is to allocate the adjacent frames into different subsets, thus reducing the possibility to occur upsets in multiple frames of the same subset.

Let’s assume that the list of erroneous frames $F = \{F_0,F_1, \ldots,F_{M-1}\}$ with syndromes $S = \{S_0,S_1,\ldots,S_{M-1}\}$, all belonging to subgroup G_A, has been read by FIFO. Notice that the error correction process is performed per subgroup; thus, if frames from more than one (sub)groups are present in the FIFO, the scrubber creates one fault list per subgroup and runs the error correction process several times. All the N configuration frames of group G are readback and the PG_A parity frame is calculated and compared with the stored PG_A parity frame. Let’s also assume that the golden and the calculated PG_A parity frames differ in the following bit positions $B = \{B_0,B_1,\ldots B_{K-1}\}$. The error correction algorithm shown in **Figure 22** is performed next.

```
1   if M = 1                // Single erroneous frame
2       correct all upsets in F based on B          // SBU or MBU (case A)
3   else                    // Multiple erroneous frames
4       for every frame Fi in F i=0…M-1 do
5           if Si belongs to B                      // when syndrome matches parity
                                                    // i.e. Si is equal with an element of B
6               correct upset Si in Fi              // SBU in Fi (case B)
7           else
8               for every Fj in F j=i+1…M-1 do
9                   if Si = Sj                      // 2 SBUs in the same row
                                                    // have masked the corresponding parity bit
10                      correct upset Si in Fi and Fj   // SBUs in Fi and Fj (case C)
```

Figure 22: 2-D error correction algorithm

The proposed coding scheme guarantees the correction of the following cases (all the frames belong to the same subgroup). Figure 23 presents some different fault scenarios, assuming a group of 8 configuration frames (4 frames per subgroup) each one containing 8 bits.

a) all SBUs and MBUs in a single configuration frame; these faults are corrected by the case A of the algorithm (line 2), while Figure 23.(i) and Figure 23.(ii) present an SBU and an MBU scenario, respectively, that are corrected by the algorithm.

b) all SBUs in multiple configuration frames; these faults are corrected by the case B of the algorithm when each row contains a single SBU (line 6) or case C when some rows contain multiple SBUs (line 10). Figure 23.(iii) and Figure 23.(iv) presents two SBU scenarios in multiple configuration frames that are corrected by the case B and C of algorithm, respectively.

c) all SBUs in multiple configuration frames and the MBUs in a single frame, assuming that the MBUs has occurred in different rows than the SBUs. In this case, the correction will be performed in two stages: in the first stage, the algorithm will correct all the SBUs (cases B or C), and thus, in the second stage there will be only one configuration frame with MBU (case A). Figure 23.v presents such a fault scenario, while Figure 23.vi presents an SBU/MBU scenario that cannot be corrected by the algorithm.

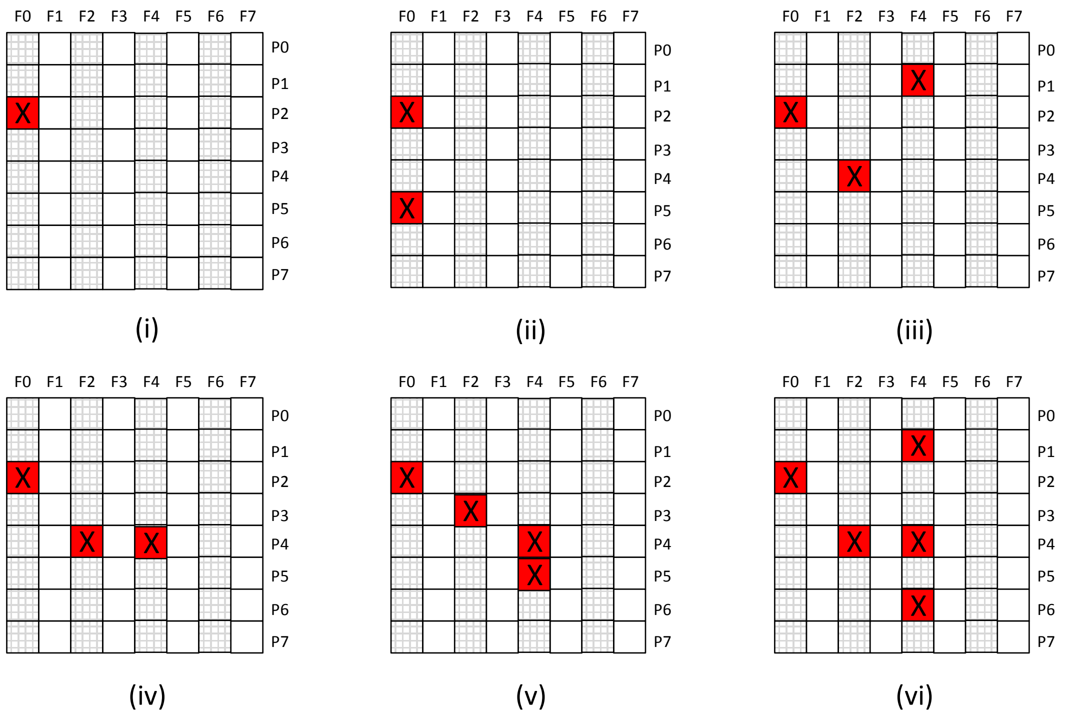


Figure 23: SBU/MBU fault scenarios

Table 22: MCU correction coverage

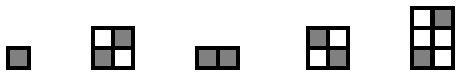
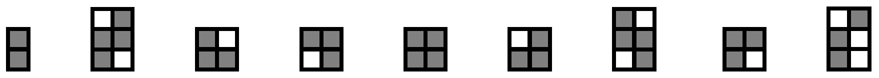
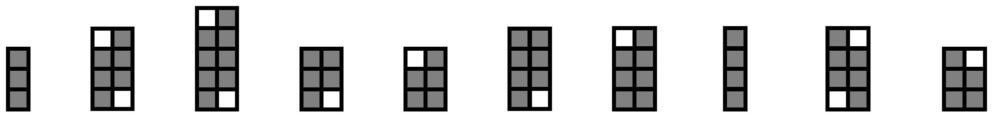
Corrected by SEM, Hybrid Internal Scrubber and Proposed Approach									
									
51.2%	29.8%	14.0%	2.28%	0.02%					
Corrected by SEM, Hybrid External Scrubber and Proposed Approach									
									
0.49%	0.42%	0.28%	0.24%	0.13%	0.13%	0.11%	0.10%	0.02%	
Corrected by Hybrid External Scrubber and Proposed Approach									
									
0.30%	0.12%	0.07%	0.06%	0.05%	0.04%	0.02%	0.02%	0.01%	0.01%

Table 22 compares three scrubbing approaches, the Xilinx SEM [17], the Hybrid Scrubber [6] and the proposed approach, in terms of the SBU/MBU correction coverage. The MCU patterns (shapes) considered in the comparison are those observed in the Test #1 (see Section 7.3). The x-axis of the shapes represents consecutive frame addresses, while the y-axis consecutive bits in a frame. The SEM in enhanced repair mode supports correction of single bit or double bit adjacent errors per frame. The Hybrid Scrubber is separated into the internal and external part. The internal

part corrects only SBUs, while the external part corrects all the other cases but with increased latency (i.e. due to the delay of reading the golden data from an external storage device and writing them back to the FPGA). The SEM core does not correct the MCUs which contain more than two upsets in a frame, i.e. the 0.7% of the total events according to the observations of Test #1. The Hybrid scrubber and the proposed scrubber correct all the upsets presented in **Table 22**. The Hybrid scrubber corrects the 97.3% of the upsets through its internal part and the rest through its external part. Our approach corrects all the upsets using the 2-D coding scheme.

8.3 Test flow

The test flow is shown in **Figure 24**. It performs the following steps:

1. The Zedboard is first powered and configured through the JTAG interface
2. During the irradiation period, no configuration action (readback or scrubbing) is performed to avoid the injection of errors due to abnormal behavior of the configuration interface.
3. When the beam gets off, a beamoff timer is activated (which counts the time that the beam is OFF) and the error correction process initiates.
4. The active status logic and the heartbeat logic are checked and in case that the heartbeat logic alarms a system malfunction, a DUT power cycle is performed (power-off, power-on and configure the board).
5. Otherwise, the FIFO is read and in the case of ECC errors, the correction process is performed: it reads the corresponding frame groups, runs the error correction algorithm and writes back the corrected configuration frames.
6. Two different scenarios have been tested. In the first scenario, called as ***non-free run***, the error correction process runs only during the beam-off period (in order to readback first the CUT and record the configuration memory image that the correction process runs on). In this case a beam-off timer is set and if it expires before the error correction process has been completed and there are uncorrected errors in the FIFO, the CUT is fully reconfigured and the process is repeated.
7. In the second scenario, called as ***free-run***, the error correction process runs continuously (the beam-off timer is set to a MAX value and it never expires). When all the faults of the FIFO list are corrected, then the FIFO is read again and the correction process continues. The risk of this scenario is that during the correction steps (and before all upsets are corrected), new soft errors may occur due to irradiation causing the algorithm to fail.
8. In both cases, if a CRC-only error is read from the FIFO, which means that the frame-level ECC failed to detect an MBU, the DUT is fully reconfigured. This also occurs when the error correction process failed to correct some errors.

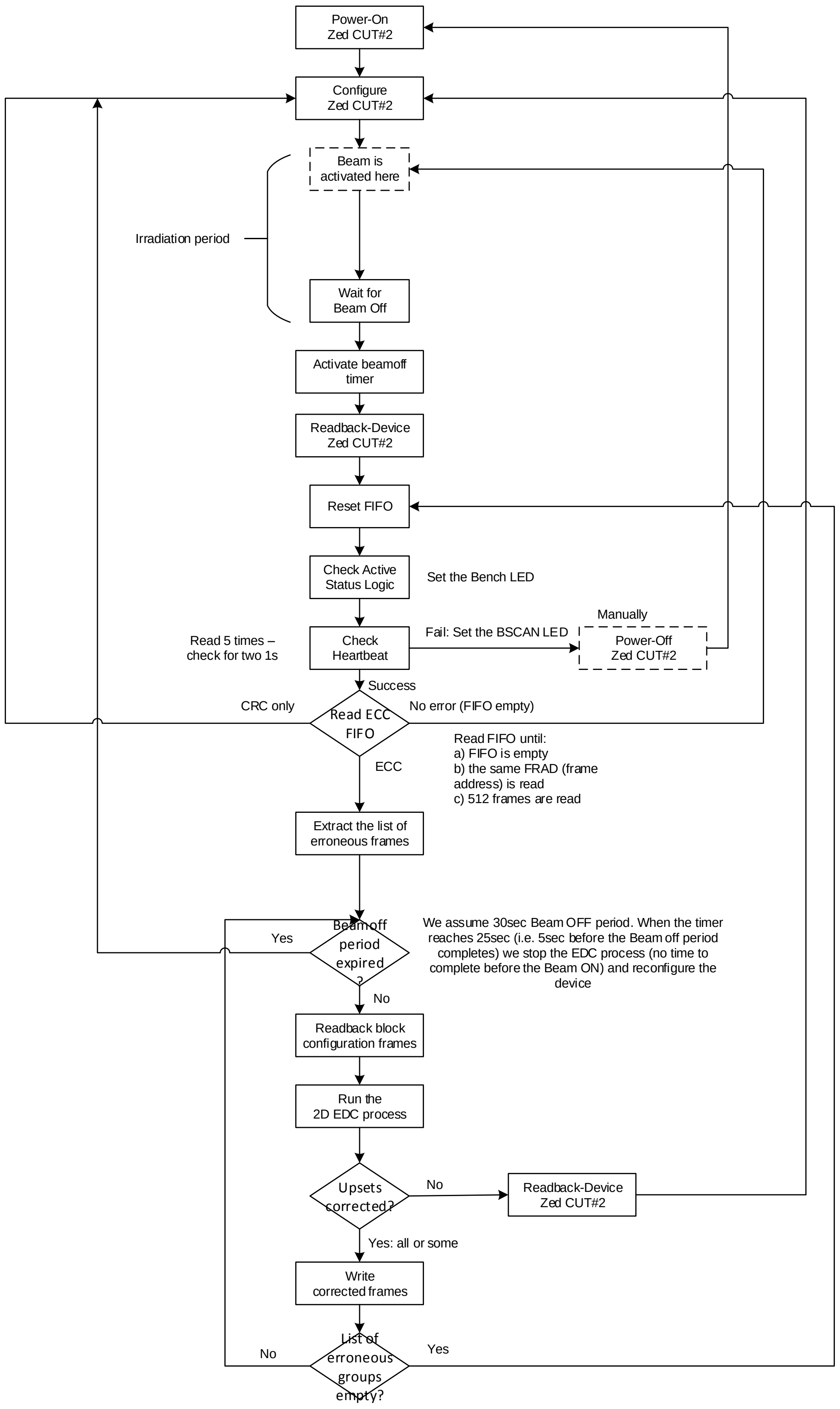

Power-On Zed CUT#2
Configure Zed CUT#2
Beam is activated here
Irradiation period
Wait for Beam Off
Activate beamoff timer
Readback-Device Zed CUT#2
Reset FIFO
Check Active Status Logic
Set the Bench LED
Manually
Read 5 times – check for two 1s
Check Heartbeat
Fail: Set the BSCAN LED
Power-Off Zed CUT#2
Success
CRC only
Read ECC FIFO
No error (FIFO empty)
Read FIFO until:
a) FIFO is empty
b) the same FRAD (frame address) is read
c) 512 frames are read
ECC
Extract the list of erroneous frames
Yes
Beamoff period expired?
We assume 30sec Beam OFF period. When the timer reaches 25sec (i.e. 5sec before the Beam off period completes) we stop the EDC process (no time to complete before the Beam ON) and reconfigure the device
No
Readback block configuration frames
Run the 2D EDC process
Upsets corrected?
No
Readback-Device Zed CUT#2
Yes: all or some
Write corrected frames
No
List of erroneous groups empty?
Yes

Figure 24: Test #2 flow

9. Test #2 results

Different test sessions were performed as shown in the following Table:

- Although during the period 17.11.18 2:00-8:00 the board under test was not properly aligned, the effectiveness of the proposed scrubbing approach for this period was investigated.
- Test #2 non-free run mode was performed for two different angles of incidence (θ), 0^o and 45^o obtaining two effective LETs of 8.8 and 12.45 MeVcm^2/mg.
- Test #2 free run mode was performed for 0^o angle of incidence and an effective LET of 8.8 MeVcm^2/mg.

Table 23: Test #2 sessions

Duration		**Angle of incidence/ Effective LET** (MeVcm^2/mg)	**Runs**	**Fluence (ions/cm^2)**	**Comments**
Start (CET)	**End (CET)**				
17.11.18 2:00	17.11.18 8:00	0^o / 8.8	392	4.23×10^5	Test #2 non-free run: Not properly aligned board
17.11.18 8:00	17.11.18 12:00	0^o / 8.8	349	3.86×10^5	Test #2 non-free run
17.11.18 14:30	17.11.18 23:30	0^o / 8.8	571	6.33×10^5	Test #2 free running mode
18.11.18 00:45	18.11.18 1:15	45^o / 12.45	41	0.45×10^5	Test #2 non-free run with angle

9.1 Post-processing software

A database with all the erroneous bits is created with the same format as Test #1. The Test #2 has run in 2 different modes, as mentioned above, the non-free run mode and the free-run mode. In case of non-free run, the database is made by comparing the readback file, derived after the beam off, with the readback golden file. For each bit difference, a database entry is created with the fields described in **Table 8**. In case of free-run (where readback process is performed only once at the first round of the test), the database is made comparing the block frames data (which are readback when the error correction algorithm is activated) with the corresponding golden block frames. This block contains all the frames of the block that includes erroneous frame which has been fetched by the ECC FIFO.

During post-processing the verification of the entire test flow shown in **Figure 24** is performed. It is checked:

- If all the erroneous frames read by the ECC FIFO (that have been recorded in the log file) match with the upsets stored in the database, i.e. we check whether all ECC FIFO frames are also written in the database and all erroneous frames of the database have been also read by the ECC FIFO. Moreover, when the ECC Syndrome indicates that there is a single bit upset (or odd-numbered bit upset) in the frame, the location of the erroneous words and bit are extracted by the ECC Syndrome and crosschecked with the database.

- In case of the non-free run mode, if the errors in the readback block configuration frames are also in the database.
- If the corrected frames written after the run of the 2-D EDC process is identical with the corresponding golden frames.

9.2 SEU/MCU analysis

Given that in Test#2 the CUT is clocked, the memory cell readback LUTRAM values are changeable and thus masked (GLUTMASK enabled). For this reason, the CRAM data have been obtained by the readback files for SEU analysis and include all the unmasked bits of the configuration bitstream plus the masked bits for the CLB FFs (initial FF values). The CRAM SEUs and the cross section for the non-free run case are shown in **Table 24** and **Table 25**.

Note: For the calculation of the cross sections only the results of the 2nd and 4th test sessions of **Table 23** were considered (the 1st test session where the board was not properly aligned was ignored). However, the upsets occurred in the 1st test session were taken into account for the analysis of the MBUs (upsets in a frame) and the MCUs. The results of the free running mode (3rd test session) were not taken into account for the calculation of the cross sections, because the correction algorithm failed in some cases disturbing the results, as explained in the next section.

Table 24: Test #2 CRAM Upsets in a frame

Upsets in a frame	**CRAM Bits Occurrences**	
	θ=0º	**θ=45º**
1	14525	1029
2	330	85
3	70	13
4	25	2
5	7	3
6	1	3

Table 25: Test #2 CRAM SEUs

		θ=0º	**θ=45º**
CRAM	**SEUs**	11243	1279
	Cross section [cm2/bit]	1.51×10^{-9}	2.06×10^{-9}
CRAM Essential	**SEUs**	4431	338
	Cross section [cm2/bit]	0.59×10^{-9}	0.55×10^{-9}

Comparing with **Table 10**, the CRAM cross section of both tests is similar for both angles of incidence. On the other hand, the cross sections considering only the upsets in the essential CRAM bits are lower than those of test #1, due to the fact that the test #2 essential bits are less than the test #1 essential bits (see **Table 3** and **Table 21**).

Moreover, we observed that the shapes of the MCUs expanding in more than one neighboring frames (**Figure 25** and **Figure 26**) are more or less the same with those of Test #1 (**Figure 14** and **Figure 18**).

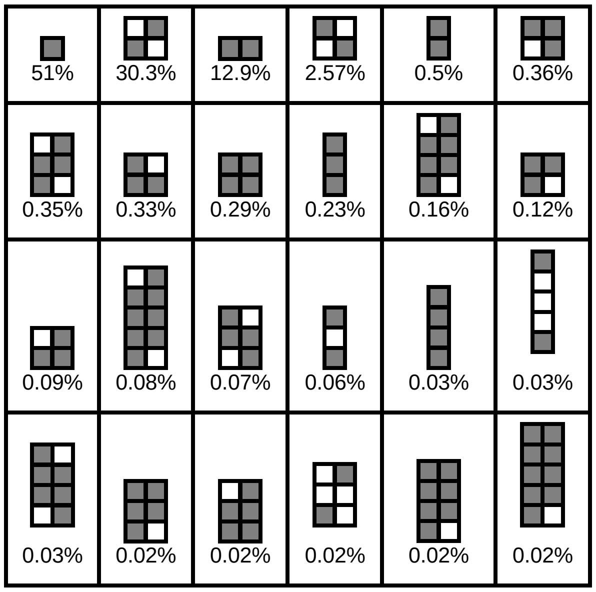


Figure 25: Test #2 MCU patterns (CRAM bits) – angle 0º

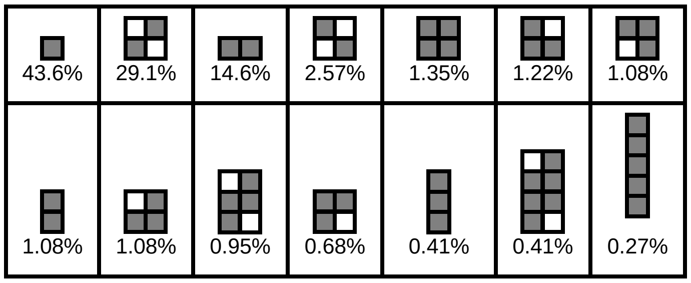


Figure 26: Test #2 MCU patterns (CRAM bits) – angle 45º

9.3 2-D error correction algorithm analysis

Non-free run mode

According to the test flow (**Figure 24**), when the beam-off timer expires and the error detection process has not been completed, the process is paused and the CUT is fully reconfigured before the beam goes on. This scenario occurred several times during the non-free run mode (2nd test session of **Table 23**); the beam-off timer expired 344 times out of 349 runs. In the other 5 runs, because the number of errors in the ECC FIFO was large, the ECC FIFO reading latency was longer than the timer expiration period; as a consequence, the timer was not expired and the ECC process inserted in the next beam period without reconfiguring the CUT. Notice that the number of the erroneous frames per spill stored in the FIFO in this test session varies from 11 to 52 frames.

This phenomenon is mainly due to the accelerated nature of the radiation testing experiment where a large number of upsets (e.g. up to 52 frames located to a large number of groups) may occur during a single spill. Our ECC algorithm tries to correct all these upsets within the beam off period (~30 sec). Given that the group read latency is 1.36 sec (see **Table 5**), the 2-D EDC process cannot handle all the erroneous frames read by the FIFO. On the other hand, a post-process analysis of the upsets showed that the proposed 2D EDC algorithm can correct all SEUs and MCUs occurred during the non-free run mode (without timing restrictions), i.e. it achieved **100% SEU/MBU correction coverage.** As a conclusion it is important to reduce the error correction latency (e.g. by implementing the external error correction mechanism in hardware).

The number of frames per group corrected using the 2-D EDC algorithm is shown in **Table 26**.

Table 26: Frames per group corrected (non-free run)

Frames per group	θ=0°	θ=45°
1	2122	45
2	2010	32
3	86	-
4	38	-

The user logic of the FF (FFs+SRLs), BRAM and DSP chains is monitored through the Active Status Logic (**Figure 35**) periodically after the beam is off, as explained in section 8.1. The percentage of error occurrence in each chain (e.g. how many times the Active Status flag read via the JTAG indicates failure), is shown in **Table 27**. Notice that these results are largely depend on the CUT (i.e. whether an upset affects the chain output). The likelihood to occur error in the FF chain is higher compared with the BRAM and DSP chains, given also the fact that the FF (CLB) chain occupies more reconfigurable resources than the BRAMs or DSPs. Moreover, the likelihood to occur error, in case of 45° angle of incidence is slightly higher.

Table 27: Active Status Logic statistics (non-free run)

	FF chain		BRAM chain		DSP chain	
	θ=0°	θ=45°	θ=0°	θ=45°	θ=0°	θ=45°
Error occurrence	39%	51%	0.3%	2.4%	4.0%	4.9%

Free run mode

In the free run mode, where the error correction process runs continuously, the following scenario occurred several times: within the time interval between the ECC FIFO read and the group frames readback, a new upset has hit another frame of the same subgroup, causing the correction algorithm to fail. However, the likelihood this scenario to occur in a non-accelerated radiation environment is very low and will become even lower by reducing the error detection latency of the algorithm. Two types of failure were observed in the above cases: the algorithm decides that cannot correct the erroneous frame(s) of a group of frames or it takes wrong decisions inserting new errors. The first failure type occurred 40 times while the second occurred 922 times. From the cases of the second type, in 808 times the ECC algorithm injected (incorrectly) 1-bit upset in a frame, in 93 times 2-bit upsets in a frame, in 11 times 3-bit upsets in a frame and in 10 times 4-bit upsets in a frame.

The frames per group corrected by the ECC algorithm including the above false operations are shown in **Table 28**.

Table 28: Frames per group corrected (free run)

Frames per group	Number of corrections
1	3371
2	3560
3	530
4	303
5	42
6	23
7	3

The percentage of error occurrence in each chain (e.g. how many times the Active Status flag read via the JTAG indicates failure) is shown in **Table 29**. The results are similar to the non-free run case (**Table 27**).

Table 29: Active Status Logic statistics (free run)

	FF chain	BRAM chain	DSP chain
Error occurrence	48%	0.6%	4.4%

APPENDIX A: Test #1 CUT

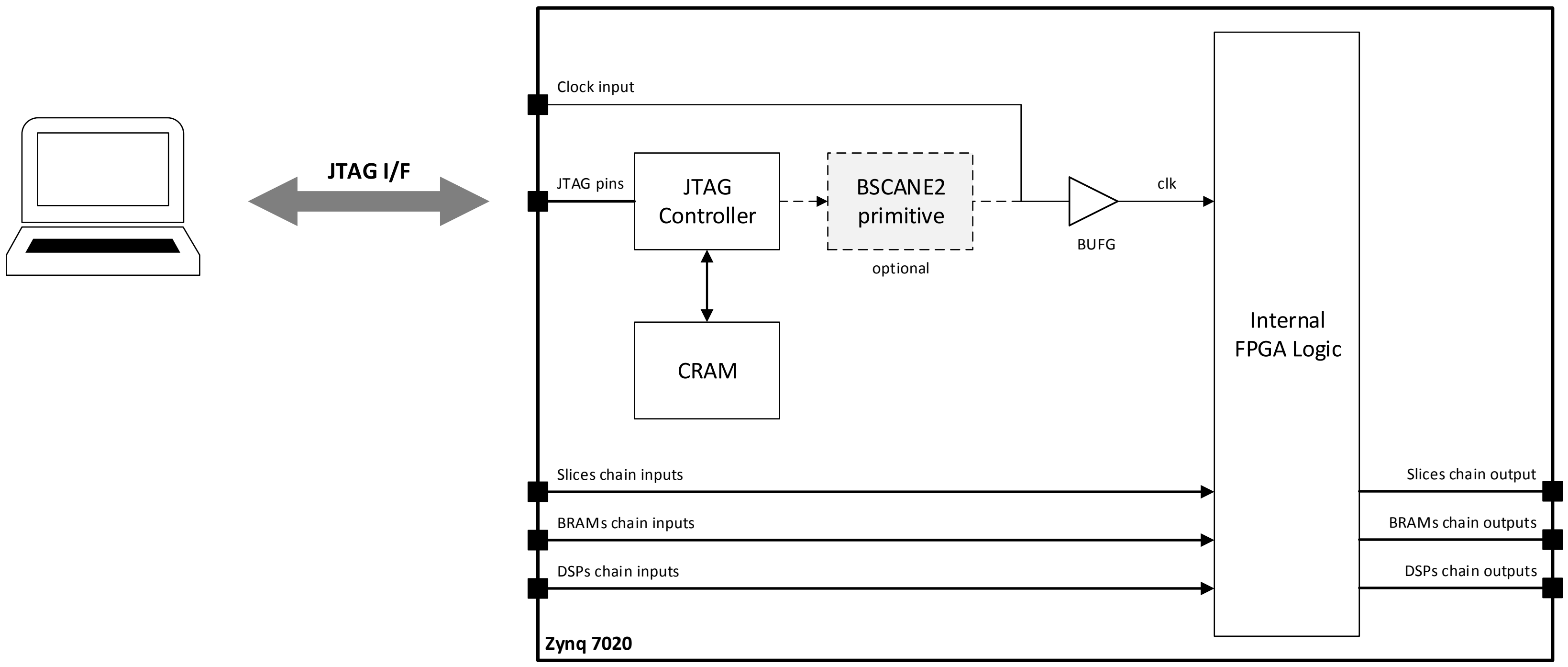


Figure 27: CUT for test #1 - Communication between the Host PC and the CUT

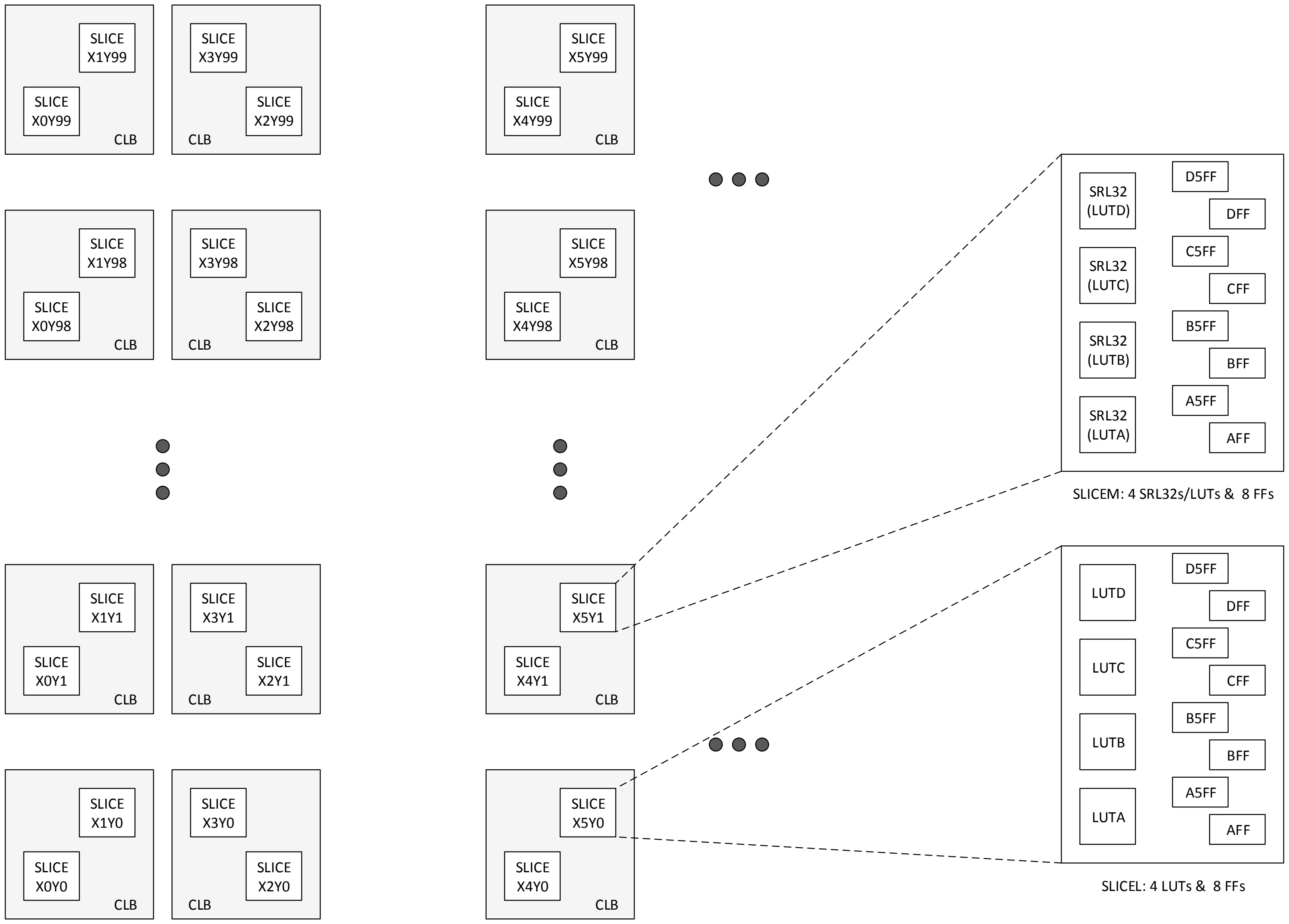
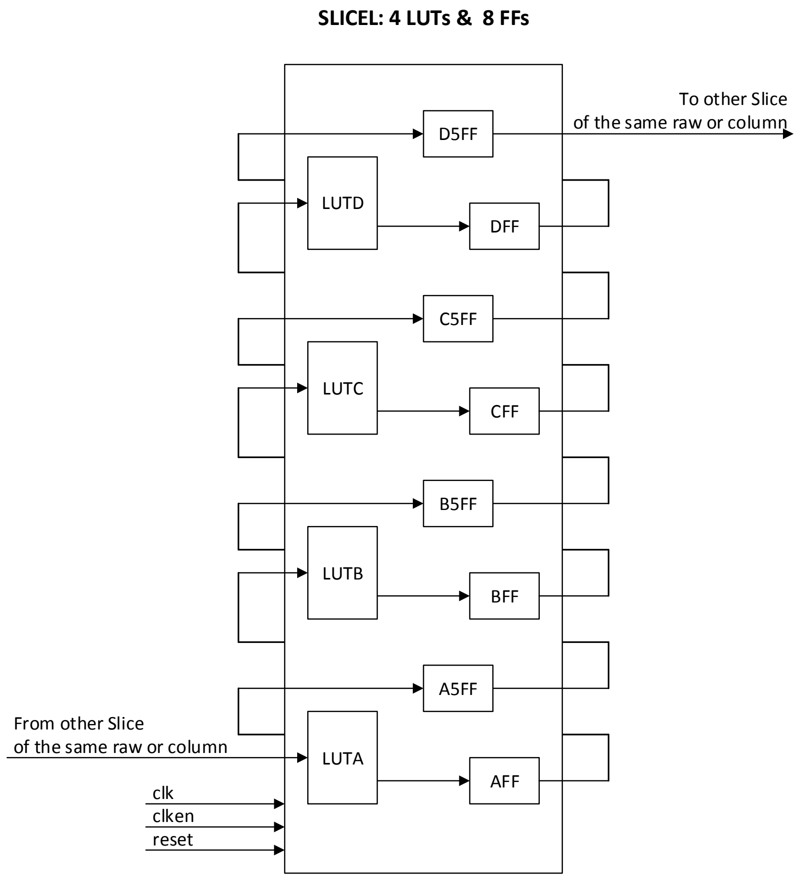
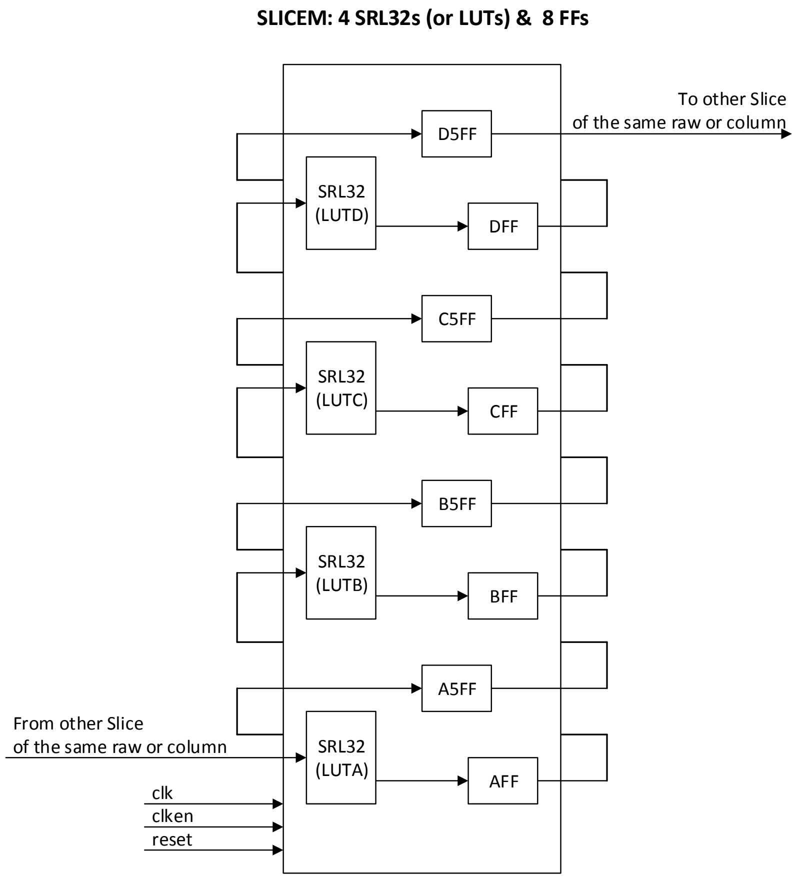


Figure 28: Zynq-700 CLB architecture (all slices are connected in a row or a column fashion)



- 8-bit shift register for each SLICEL
- LUTs used as route-thrus
- FFs initialization values of a SLICEL configurable

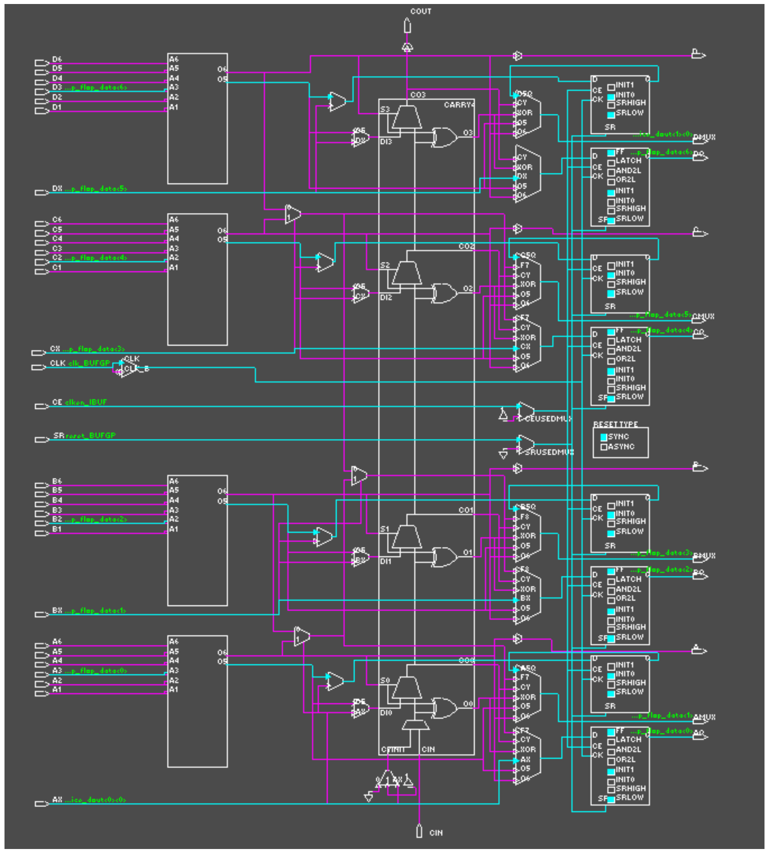


- 136-bit shift register for each SLICEM
- LUTs used as 32-bit shift register
- FFs and SRLs initialization values of a SLICEM configurable

Figure 29: Slice configuration as FF chains

SLICEL: 4 LUTs & 8 FFs

SLICEM: 4 SRL32s (or LUTs) & 8 FFs



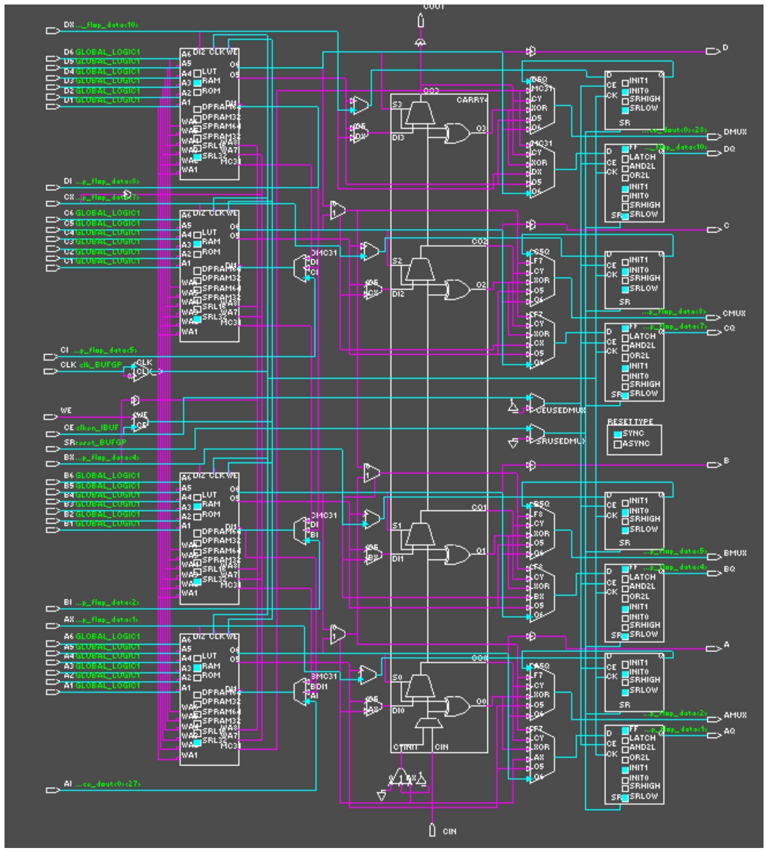
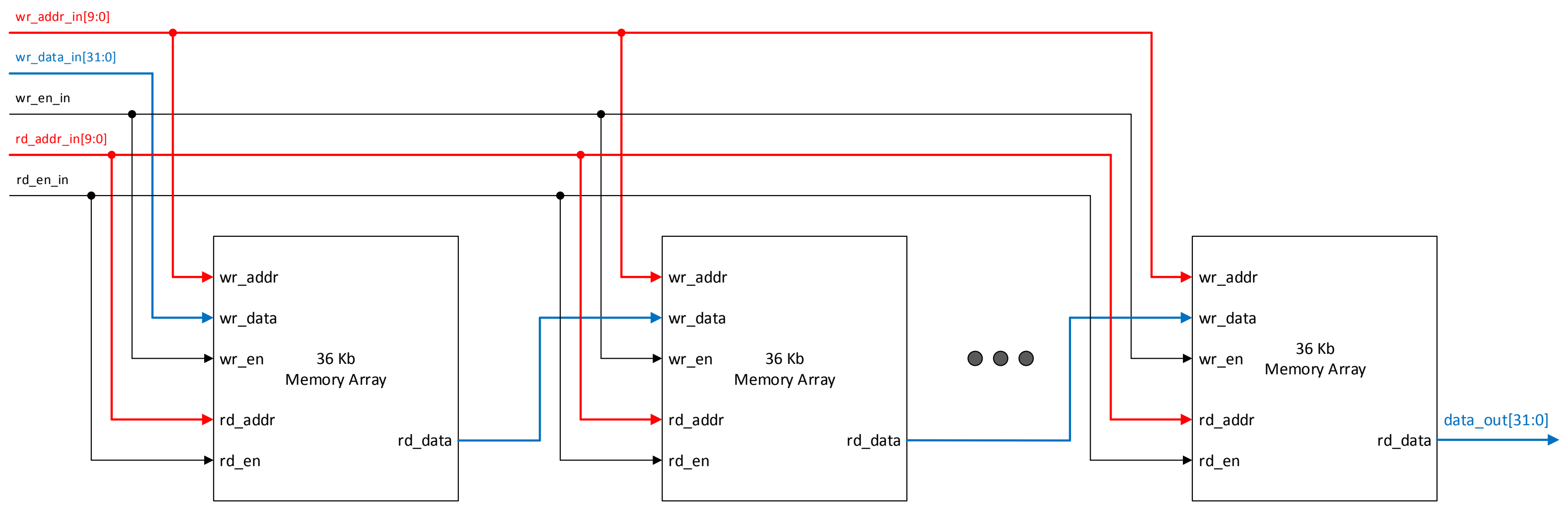
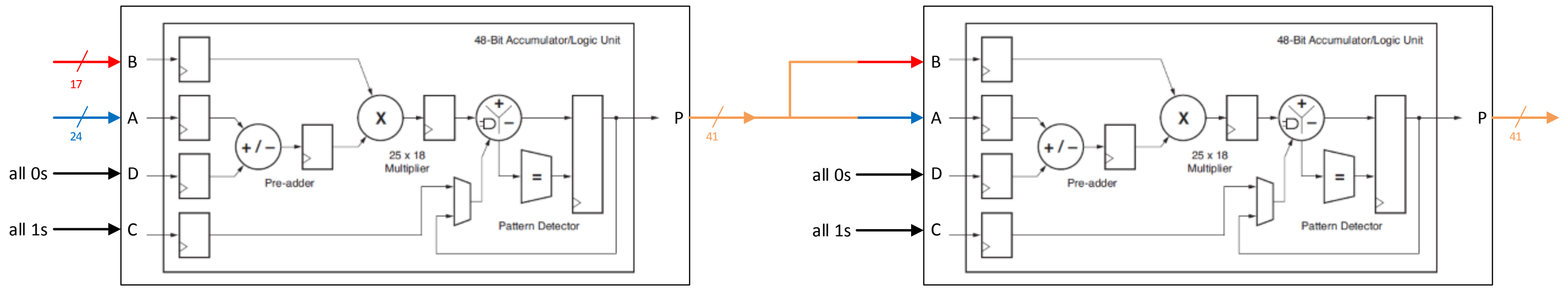


Figure 30: Slice configuration and routing



- BRAMs initialized with a predefined pattern
- BRAMs connected in cascade mode through Data Bus either horizontal (raw) or vertical (column) -> configurable
- Each Memory size 1024 x 32 = 32.768 bits (36 Kb)

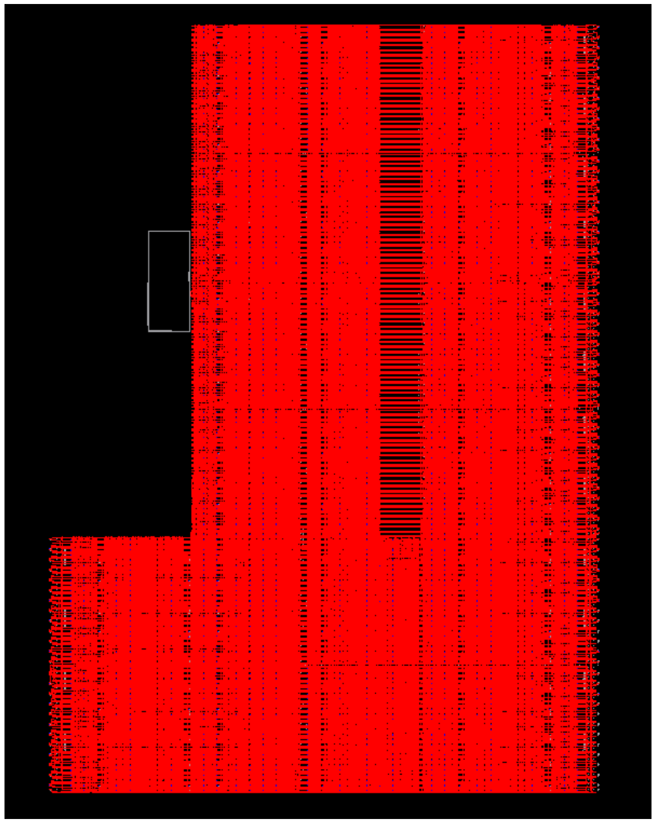
Figure 31: BRAM configuration and connection in cascade mode

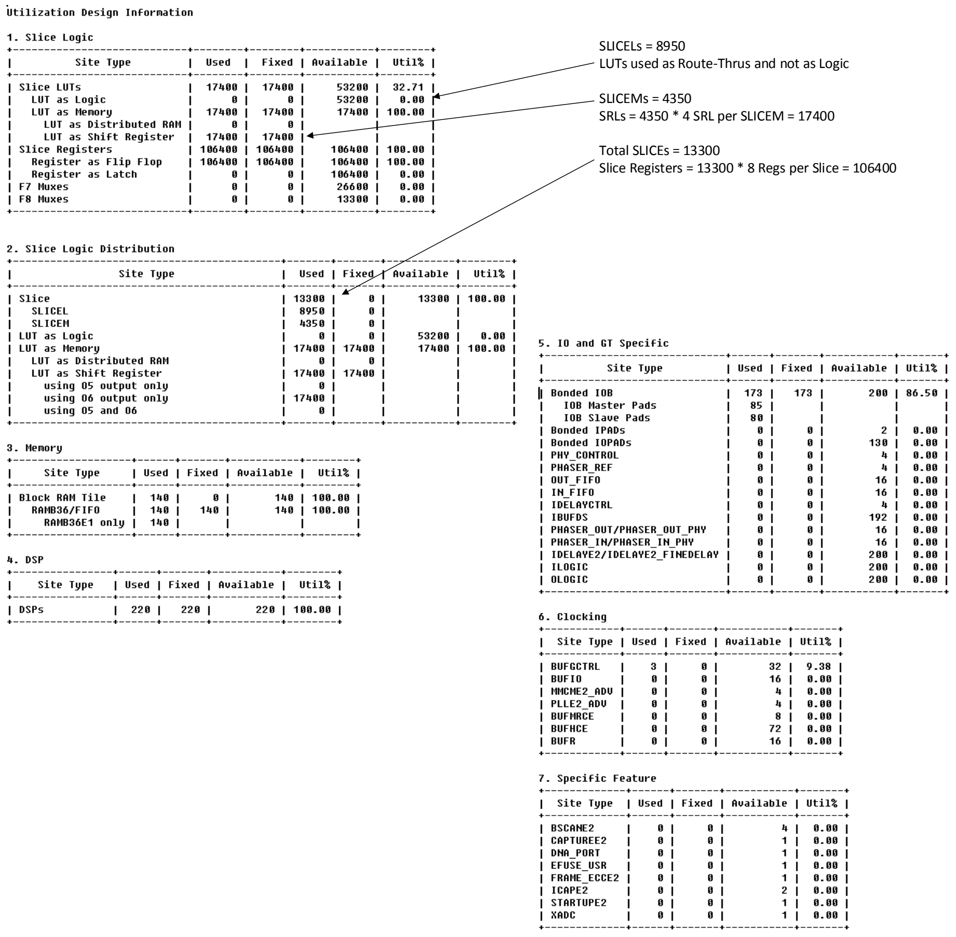


- DSPs connected in cascade mode
- DSPs implement multiplication
- DSPs connected in cascade mode either horizontal (raw) or vertical (column) -> configurable

Figure 32: DSP slices configuration and connection in cascade mode

Full FPGA routing





```
Utilization Design Information

1. Slice Logic
+----------------------------+--------+--------+-----------+--------+
|         Site Type          |  Used  | Fixed  | Available | Util%  |
+----------------------------+--------+--------+-----------+--------+
| Slice LUTs                 |  17400 |  17400 |     53200 |  32.71 |
|   LUT as Logic             |      0 |      0 |     53200 |   0.00 |
|   LUT as Memory            |  17400 |  17400 |     17400 | 100.00 |
|     LUT as Distributed RAM |      0 |      0 |           |        |
|     LUT as Shift Register  |  17400 |  17400 |           |        |
| Slice Registers            | 106400 | 106400 |    106400 | 100.00 |
|   Register as Flip Flop    | 106400 | 106400 |    106400 | 100.00 |
|   Register as Latch        |      0 |      0 |    106400 |   0.00 |
| F7 Muxes                   |      0 |      0 |     26600 |   0.00 |
| F8 Muxes                   |      0 |      0 |     13300 |   0.00 |
+----------------------------+--------+--------+-----------+--------+

2. Slice Logic Distribution
+------------------------------------------+-------+-------+-----------+--------+
|                 Site Type                | Used  | Fixed | Available | Util%  |
+------------------------------------------+-------+-------+-----------+--------+
| Slice                                    | 13300 |     0 |     13300 | 100.00 |
|   SLICEL                                 |  8950 |     0 |           |        |
|   SLICEM                                 |  4350 |     0 |           |        |
| LUT as Logic                             |     0 |     0 |     53200 |   0.00 |
| LUT as Memory                            | 17400 | 17400 |     17400 | 100.00 |
|   LUT as Distributed RAM                 |     0 |     0 |           |        |
|   LUT as Shift Register                  | 17400 | 17400 |           |        |
|     using O5 output only                 |     0 |       |           |        |
|     using O6 output only                 | 17400 |       |           |        |
|     using O5 and O6                      |     0 |       |           |        |
+------------------------------------------+-------+-------+-----------+--------+

3. Memory
+-------------------+------+-------+-----------+--------+
|     Site Type     | Used | Fixed | Available | Util%  |
+-------------------+------+-------+-----------+--------+
| Block RAM Tile    |  140 |     0 |       140 | 100.00 |
|   RAMB36/FIFO     |  140 |   140 |       140 | 100.00 |
|     RAMB36E1 only |  140 |       |           |        |
+-------------------+------+-------+-----------+--------+

4. DSP
+----------------+------+-------+-----------+--------+
|   Site Type    | Used | Fixed | Available | Util%  |
+----------------+------+-------+-----------+--------+
| DSPs           |  220 |   220 |       220 | 100.00 |
+----------------+------+-------+-----------+--------+

5. IO and GT Specific
+-----------------------------+------+-------+-----------+-------+
|          Site Type          | Used | Fixed | Available | Util% |
+-----------------------------+------+-------+-----------+-------+
| Bonded IOB                  |  173 |   173 |       200 | 86.50 |
|   IOB Master Pads           |   85 |       |           |       |
|   IOB Slave Pads            |   80 |       |           |       |
| Bonded IPADs                |    0 |     0 |         2 |  0.00 |
| Bonded IOPADs               |    0 |     0 |       130 |  0.00 |
| PHY_CONTROL                 |    0 |     0 |         4 |  0.00 |
| PHASER_REF                  |    0 |     0 |         4 |  0.00 |
| OUT_FIFO                    |    0 |     0 |        16 |  0.00 |
| IN_FIFO                     |    0 |     0 |        16 |  0.00 |
| IDELAYCTRL                  |    0 |     0 |         4 |  0.00 |
| IBUFDS                      |    0 |     0 |       192 |  0.00 |
| PHASER_OUT/PHASER_OUT_PHY   |    0 |     0 |        16 |  0.00 |
| PHASER_IN/PHASER_IN_PHY     |    0 |     0 |        16 |  0.00 |
| IDELAYE2/IDELAYE2_FINEDELAY |    0 |     0 |       200 |  0.00 |
| ILOGIC                      |    0 |     0 |       200 |  0.00 |
| OLOGIC                      |    0 |     0 |       200 |  0.00 |
+-----------------------------+------+-------+-----------+-------+

6. Clocking
+------------+------+-------+-----------+-------+
|  Site Type | Used | Fixed | Available | Util% |
+------------+------+-------+-----------+-------+
| BUFGCTRL   |    3 |     0 |        32 |  9.38 |
| BUFIO      |    0 |     0 |        16 |  0.00 |
| MMCME2_ADV |    0 |     0 |         4 |  0.00 |
| PLLE2_ADV  |    0 |     0 |         4 |  0.00 |
| BUFMRCE    |    0 |     0 |         8 |  0.00 |
| BUFHCE     |    0 |     0 |        72 |  0.00 |
| BUFR       |    0 |     0 |        16 |  0.00 |
+------------+------+-------+-----------+-------+

7. Specific Feature
+-------------+------+-------+-----------+-------+
|  Site Type  | Used | Fixed | Available | Util% |
+-------------+------+-------+-----------+-------+
| BSCANE2     |    0 |     0 |         4 |  0.00 |
| CAPTUREE2   |    0 |     0 |         1 |  0.00 |
| DNA_PORT    |    0 |     0 |         1 |  0.00 |
| EFUSE_USR   |    0 |     0 |         1 |  0.00 |
| FRAME_ECCE2 |    0 |     0 |         1 |  0.00 |
| ICAPE2      |    0 |     0 |         2 |  0.00 |
| STARTUPE2   |    0 |     0 |         1 |  0.00 |
| XADC        |    0 |     0 |         1 |  0.00 |
+-------------+------+-------+-----------+-------+
```

SLICELs = 8950
LUTs used as Route-Thrus and not as Logic

SLICEMs = 4350
SRLs = 4350 * 4 SRL per SLICEM = 17400

Total SLICEs = 13300
Slice Registers = 13300 * 8 Regs per Slice = 106400

Figure 33: Test #1 - FPGA resource utilization and routing

APPENDIX B: Test #2 CUT

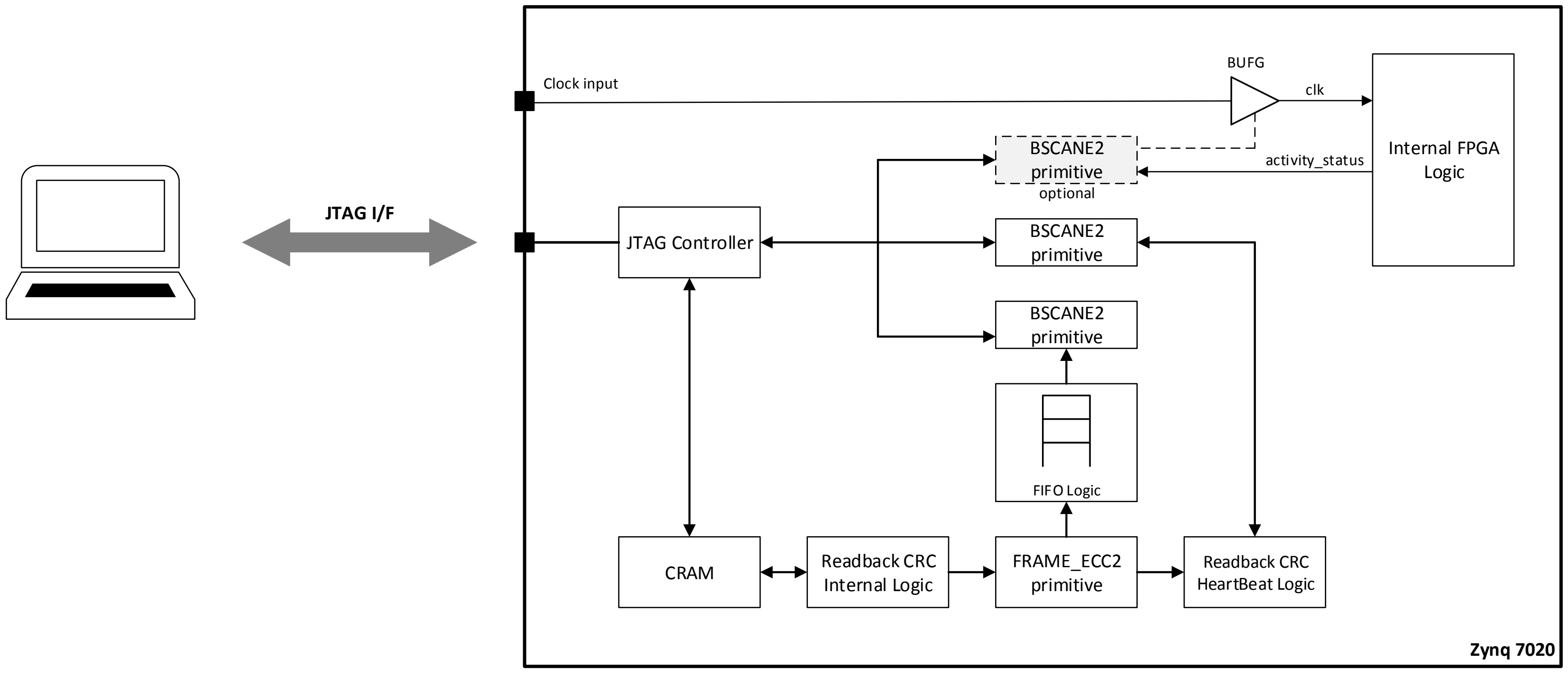


Figure 34: CUT for test #2 - Communication between the Host PC and the CUT

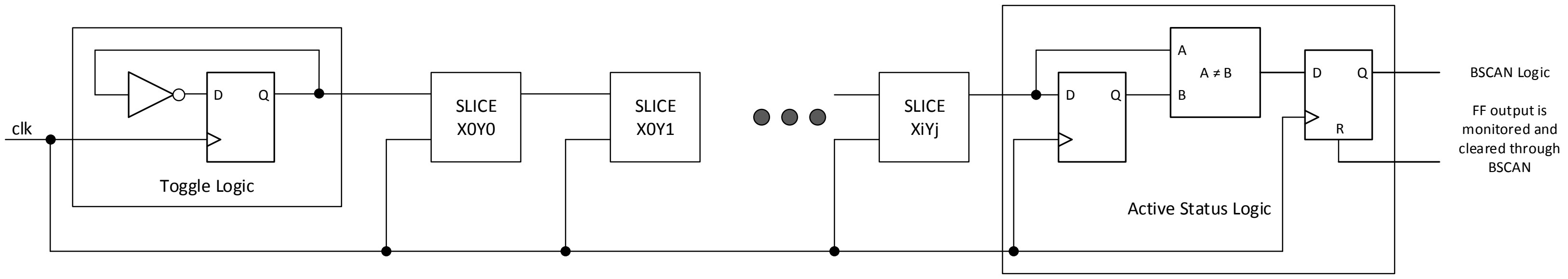


Figure 35: Test #2 – Active status logic

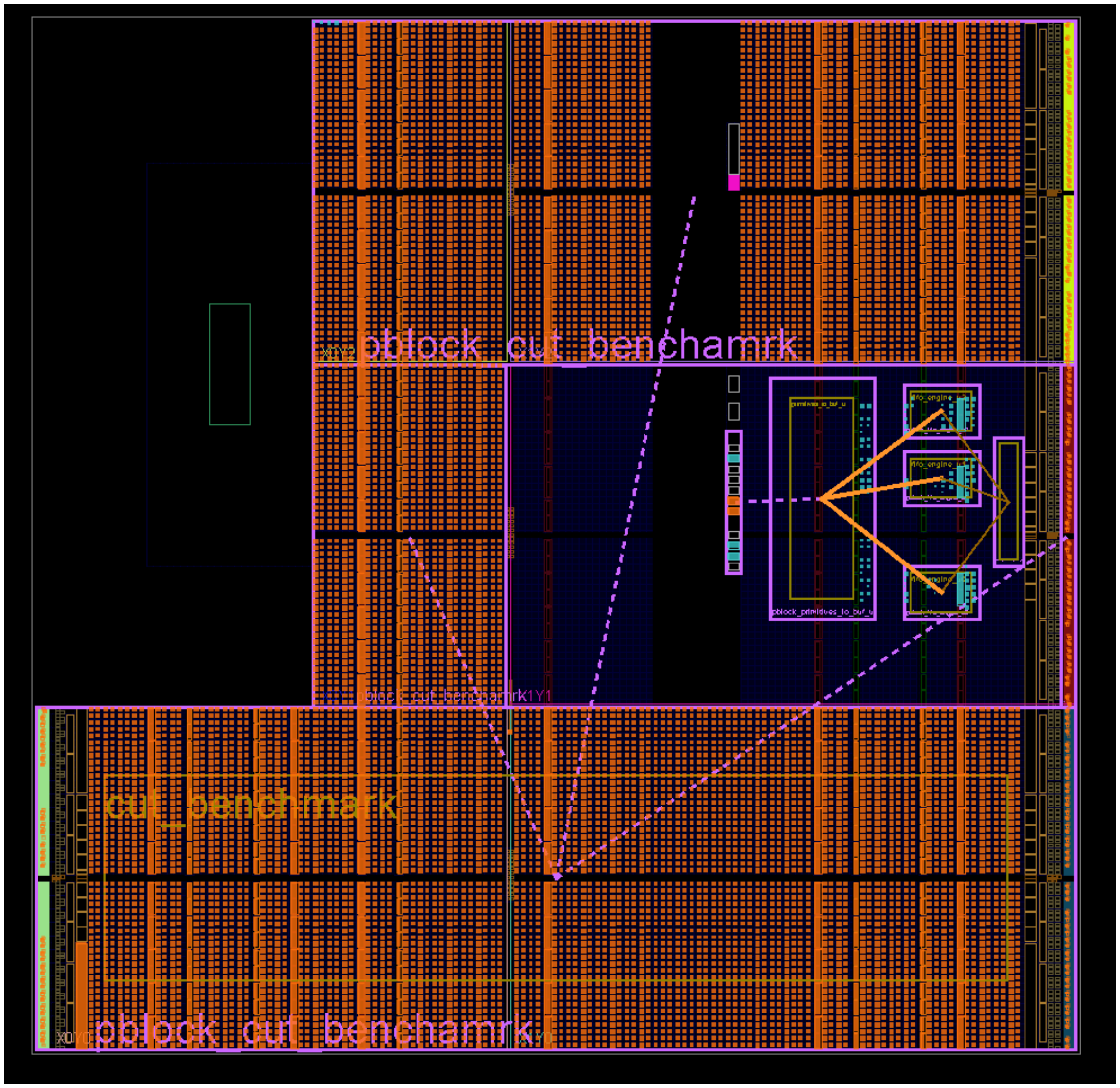


Figure 36: Test #2 - FPGA resource utilization and routing