This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2648978, IEEE Transactions on Nuclear Science

Analyzing Reliability and Performance Trade-offs of HLS-based Designs in SRAM-based FPGAs under Soft Errors

Lucas Antunes Tambara, Jorge Tonfat, André Santos, Fernanda Lima Kastensmidt, Nilberto H. Medina, Nemitala Added, Vitor A. P. Aguiar, Fernando Aguirre, and Marcilei A. G. Silveira

***Abstract*—The increasing system complexity of FPGA-based hardware designs and shortening of time-to-market have motivated the adoption of new designing methodologies focused on addressing the current need for high-performance circuits. High-Level Synthesis (HLS) tools can generate Register Transfer Level (RTL) designs from high-level software programming languages. These tools have evolved significantly in recent years, providing optimized RTL designs, which can serve the needs of safety-critical applications that require both high performance and high reliability levels. However, a reliability evaluation of HLS-based designs under soft errors has not yet been presented. In this work, the trade-offs of different HLS-based designs in terms of reliability, resource utilization, and performance are investigated by analyzing their behavior under soft errors and comparing them to a standard processor-based implementation in an SRAM-based FPGA. Results obtained from fault injection campaigns and radiation experiments show that it is possible to increase the performance of a processor-based system up to 5,000 times by changing its architecture with a small impact in the cross section (increasing up to 8 times), and still increasing the Mean Workload Between Failures (MWBF) of the system.**

***Index Terms*—FPGA, High-Level Synthesis, Soft Errors, Single Event Effects, Reliability.**

I. Introduction

RECENTLY, the increasing system complexity and heterogeneity of hardware designs embedded in state-of-the-art Commercial-Off-The-Shelf (COTS) SRAM-based Field Programmable Gate Array (FPGA) devices and shortening of time-to-market have motivated the development of new designing methodologies focused on addressing the current need for high-performance circuits. High-Level Synthesis (HLS) tools provide a design methodology able to generate optimized digital designs for different application needs. Moreover, HLS tools have evolved significantly in recent years, providing significant results in area and performance with very short development time.

HLS tools start from a high-level software programmable language (e.g. C, C++, SystemC) to automatically produce a Register Transfer Level (RTL) hardware described in a Hardware Description Language (HDL) (e.g. VHDL or Verilog) that performs the same function. High-level synthesis consists in extracting the control and data flow graphs from the source code to implement the hardware design based on default and user-applied special optimization directives. Such directives force the high-level synthesis to focus on particular objectives, such as performance, throughput, or area. As a result, one of the benefits of HLS is that CPU-intensive task functions that are usually performed in embedded processors in SRAM-based FPGAs can be offloaded to dedicated hardware accelerators implemented in the programmable logic array.

However, it is well known that one of the major reliability concerns for SRAM-based FPGAs is their susceptibility to soft errors, more specifically Single Event Upsets (SEUs) in their configuration memory bits and embedded memory cells [1]. When a charged particle passes through a sensitive junction of an SRAM cell transistor that composes the configuration memory of an SRAM-based FPGA, it can induce transient currents that can cause voltage changes in the hit circuit node, thus provoking a bit-flip (SEU) [2]. This bit-flip can change the configuration of a routing interconnection, Look-up Table (LUT), or Block RAM (BRAM) memory. Moreover, this bit-flip has a persistent effect, which can only be corrected when a new bitstream is loaded into the FPGA. A bit-flip can also occur in the Flip-Flop (FF) of a Configurable Logic Block (CLB) used to implement the user's sequential logic. In this case, the bit-flip has a transient effect and the next load of the flip-flop can correct it. Thus, the majority of the errors observed in SRAM-based FPGAs used in harsh environments come from bit-flips in the configuration memory bits, thus the susceptibility of a design synthesized into them must always be closely investigated.

Regarding the susceptibility of HLS-based designs, depending on the coding style of the high-level software programming language and optimization directives applied in the HLS tool, different amounts of FPGA resources and

Manuscript received July 12, 2016; revised January 2, 2017. This work was supported in part by CNPq, CAPES, FAPERGS, and FAPESP Brazilian Research Agencies.

L. A. Tambara, J. Tonfat, A. Santos, and F. L. Kastensmidt are with Instituto de Informática, PGMICRO, UFRGS, Porto Alegre, Brazil. e-mail: {latambara, jltseclen, afdsantos, fglima}@inf.ufrgs.br.

N. H. Medina, N. Added, V. A. P. Aguiar, and F. Aguirre are with Universidade de São Paulo, São Paulo, Brazil. e-mail: {medina, nemitala, vpaguiar, faguirre}@if.usp.br.

M. A. G. Silveira is with Centro Universitário da FEI, São Bernardo do Campo, Brazil. e-mail: marcilei@fei.edu.br.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2648978, IEEE Transactions on Nuclear Science

configuration bits, and distinct execution times are achieved by the generated hardware. Consequently, the cross section and Mean Workload Between Failures (MWBF) may present significantly different results. However, it is not only the amount of FPGA resources and configuration bits that determine the sensitivity of a hardware. The error masking effect [3] of the application algorithm implemented using HLS plays an important role, directly affecting the reliability of a design implemented in an FPGA, as shown in this work.

HLS-based designs have been employed in some safety-critical applications that require both high performance and high reliability levels, such as Advanced Driver Assistance Systems (ADAS) [4], medical systems [4], satellites [5], and even particle accelerators [6]. The trade-offs between area and performance in HLS-based designs have been studied in last years [7, 8]. However, a susceptibility evaluation of HLS-based designs aiming to evaluate their effectiveness under soft errors in SRAM-based FPGAs has not yet been developed.

The main objective of the present work is to investigate the trade-offs of different HLS-based designs in terms of not only resource utilization and performance but also reliability. Thus, an analysis of their behavior under soft errors was performed and compared to a standard processor-based implementation running on a soft-core processor embedded in an SRAM-based FPGA. Different HLS optimization strategies are considered, such as default, pipeline and loop unroll in different places, array partition with different configurations, and inline functions. The case-study designs consist of several architectures based on three benchmark algorithms: Matrix Multiplication (MM), Advanced Encryption Standard (AES), and Adaptive Differential Pulse-Code Modulation (ADPCM).

The designs are evaluated by several FPGA-based Fault Injection (FI) campaigns and radiation experiments with heavy ions, from which the dynamic cross section and MWBF are estimated for each design version. As a result, a reliability analysis for HLS-based designs is proposed. Results show that, in general, the estimation of the dynamic cross section and MWBF values of HLS-based designs through the proposed flow based on fault injection is a suitable method for predicting their trend before radiation experiments. Results also reveal that there are important trade-offs between the use of resources, optimization strategies, and execution time in order to achieve higher MWBF values.

II. High-Level Synthesis Overview

Currently, there is a large variety of HLS tools, which range from commercial products, such as Vivado HLS [9] (Xilinx), to open source tools developed from academic research initiatives, such as LegUp [10] (University of Toronto). An in-depth analysis and discussion of several commercial and academic HLS tools regarding performance and resources usage was performed in [7]. In this work, Vivado HLS was chosen to generate HLS-based designs because since it is a tool from Xilinx, there is a significant support for different Xilinx FPGA devices (7-series, Virtex, Artix, Kintex, Zynq-7000, etc.).

In Vivado HLS, the Register Transfer Level (RTL) generation is guided by synthesis directives, which are manually configured and require no source code modification. The most relevant directives are: i) *array partition*, which partitions large data arrays into multiple smaller data arrays or into individual registers in order to provide more data ports, improving access to data and removing memory bottlenecks; ii) *inline*, which inlines a function by removing all function hierarchy and enables logic optimization across function boundaries; iii) *pipeline*, which reduces the initiation interval by allowing the concurrent execution of operations within a loop or function; iv) *resource*, which specifies that a specific library resource (core, such as memory or bus) is used to implement a variable in RTL; and v) *unroll*, which unrolls *for* loops to create multiple independent operations rather than a single collection of operations [9]. Another feature of Vivado HLS is the possibility of creating hybrid designs with portions of code running on a soft-core or hard-core processor communicating with custom hardware accelerators. The Vivado HLS design flow is illustrated in Fig. 1.

In recent years, several works have investigated the trade-offs among area, performance, and types of resources utilized in HLS-based designs. In [6], for example, the authors performed a feasibility investigation by verifying if HLS tools are capable and mature enough to be applied in building critical data acquisition systems. Their case-study was the CERN CMS detector ECAL Data Concentrator Card (DCC). They concluded that the Vivado HLS fills all the constraints and can be used to build critical data acquisition systems. However, the authors did not perform a reliability evaluation of the design.

III. Proposed Reliability Analysis

The reliability analysis of a design implemented in an FPGA depends on the characteristics of the design and susceptibility of the underlying FPGA platform. This section presents the proposed methodology flow to estimate the reliability of HLS-based designs based on fault injection campaigns. It aims to accelerate the search for the design with the best trade-off between performance and reliability, i.e. the design that provides a performance enhancement higher than the cross section increase. Our methodology takes into account four parameter groups: A) Resources and performance in terms of execution time; B) Errors and critical bits;

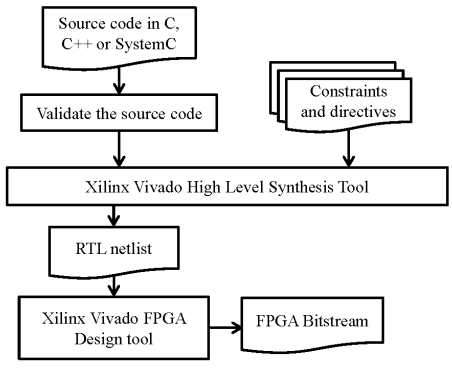

Fig. 1. Xilinx Vivado HLS design flow.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2648978, IEEE Transactions on Nuclear Science

C) Radiation measurements such as static and dynamic cross sections; and D) Mean Workload Between Failures. Fig. 2 summarizes the proposed reliability analysis in a flow diagram.

A. Resources and Performance

The area of an implemented design can be expressed in terms of the number of used resources, such as LUTs, flip-flops, BRAM blocks, Digital Signal Processor (DSP) blocks, etc. It is also possible to express the area in terms of configuration frames and configuration bits. A configuration frame is a group of configuration bits. It is the smallest addressable memory segment of the configuration memory (bitstream) of an SRAM-based FPGA. Since each frame is related to a specific resource and position in the floorplanning of the FPGA, the number of configuration frames (and configuration bits) used by a design can be calculated. In terms of reliability, the resource information is important since it is used to determine how much the design is physically exposed to radiation.

The performance of a design can be expressed in terms of execution time, operational frequency, and processed workload. The execution time can be defined by the number of clock cycles needed to perform an operation. According to the FPGA and embedded design architecture, a maximum clock frequency is achieved. Another important parameter is the workload processed by the design, which is the amount of data computed at each design execution. In terms of reliability, performance information is important to determine how much time the design is exposed to soft errors during the execution of the implemented function.

B. Errors and Critical Bits

An error is defined as any deviation from the expected behavior of a design. It can be classified as Silent Data Corruption (SDC) error (erroneous data in the design output) or timeout error (absence of data in the design output after a given time). With regard to critical bits, Xilinx defines critical bits in [11] as the amount of configuration bits that once flipped, they cause an error in the expected design behavior (SDC or timeout). Critical bits are obtained by means of fault injection in the Design Under Test (DUT). In this work, the critical bits of each design version are obtained by an exhaustive and sequential fault injection campaign, in which all the configuration bits of the injection area are flipped one at a time. A bit is considered critical if it causes a deviation from the expected design output (SDC or timeout). Thus, an entire fault injection campaign can last from a few hours to days depending on the amount of configuration bits that a design has or, in other words, the amount of bits that are going to be flipped.

C. Radiation Measurements

The static cross section (σ_{static}) is an intrinsic parameter of the device usually expressed in terms of area (cm^2/device or cm^2/bit) and is related to the minimum susceptible area of the device to a particle species (e.g. neutron, proton, heavy ion) and energy (Linear Energy Transfer - LET) [12]. The expression to estimate the static cross section per bit of a device is defined in (1, 2).

$$\sigma_{static} = \frac{N_{SEU}}{\Phi_{particle}} \qquad \sigma_{static-bit} = \frac{\sigma_{static}}{N_{bit}} \qquad (1, 2)$$

Where, N_{SEU} is the number of SEUs in the configuration memory bits and $\Phi_{particle}$ is the particle fluence. The fluence is measured by particle per cm^2 and is calculated by multiplying the particle flux by the time the device has been exposed to that flux. The cross section per bit is estimated by dividing the static cross section by the total number of bits in the device (N_{bit}). In SRAM-based FPGAs, the most important static cross section per bit is the one from the configuration memory (CRAM) and user embedded memory (BRAM).

The dynamic cross section ($\sigma_{dynamic}$) is defined as the ratio between the number of errors observed at the output of a design divided by the fluence of hitting particles. It quantifies the sensitivity of a design to a specific radiation source. The expression to estimate the dynamic cross section is defined in (3).

$$\sigma_{dynamic} = \frac{N_{errors}}{\Phi_{particle}} \qquad (3)$$

Where, N_{errors} is the number of errors observed in the design behavior and $\Phi_{particle}$ is the particle fluence.

D. Mean Workload Between Failures

The MWBF metric was first described in [13]. It evaluates the amount of data (workload) correctly processed by the design before an error. It is defined in (4), where, w is the workload processed (expressed in bits) in one execution, $\sigma_{dynamic}$ is the dynamic cross section, *flux* is the particle fluence per unit time, and t_{exec} is the execution time of the design.

$$MWBF = \frac{w}{\sigma_{dynamic} * flux * t_{exec}} \qquad (4)$$

A higher MWBF actually means that a higher workload was computed correctly before experiencing a failure and that the operational reliability of a design is higher.

Xilinx Vivado design tool provides the area of an implemented design in terms of resource utilization after the design is placed and routed, and Xilinx Vivado HLS tool provides the performance of the design in terms of clock cycles. The execution time of an HLS-based design is obtained by multiplying the number of clock cycles by the clock period. The execution time of the processor-based design is obtained at execution time.

The fault injector platform used in this work is based on the work presented in [14]. Our fault injection platform is composed of an ICAP controller circuit embedded in the FPGA and a script running on a monitor computer. The ICAP controller circuit controls the Internal Configuration Access Port (ICAP) and is connected to the script, which defines the injection area and type of fault injection (sequential or random) and controls the campaign. Faults are only injected in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2648978, IEEE Transactions on Nuclear Science

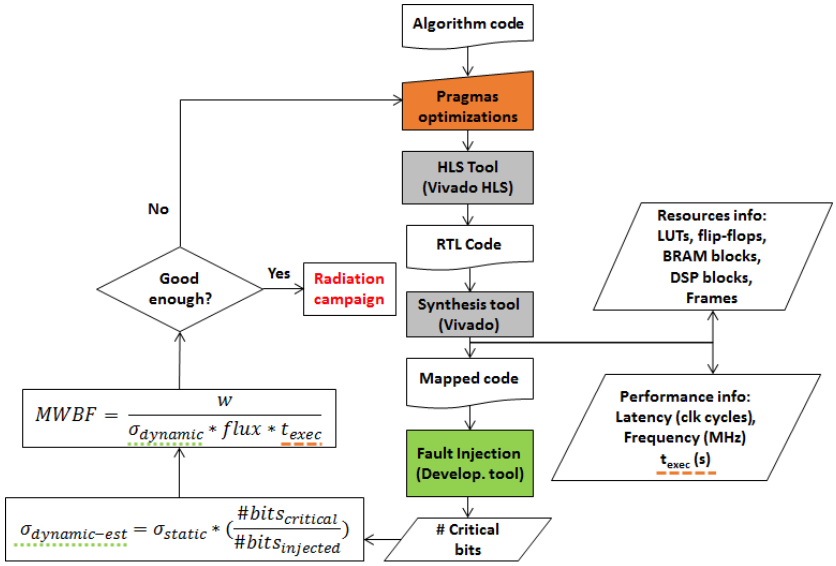

Fig. 2. Proposed reliability analysis flow.

the area of the DUT and in their configuration bits related to CLBs (LUTs, user FFs, and interconnections), DSP resources (DSP48E), and clock distribution interconnections. Faults are not injected in BRAM configuration bits in order not to affect the inputs and outputs of the DUT.

The method proposed by R. Velazco et al. in [15] was used to estimate the dynamic cross section of the designs before the radiation test campaigns. In this method, the dynamic cross section is calculated by multiplying the static cross section by the masking upset probability of the design, as shown in (5). The masking upset probability can be considered by using the information about the critical bits, which are first evaluated by means of fault injection. The static cross section for a specific particle and a specific device can be obtained from previous experiments or reports, such as [16].

$$\sigma_{dynamic-est} = \sigma_{static} * (\frac{\#bits_{critical}}{\#bits_{injected}}) \quad (5)$$

IV. Case-Study Designs and Resources and Performance Results

Three C language-based benchmark algorithms were evaluated: i) 32x32 floating-point matrix multiplication (MM) [17], ii) 128-bit Advanced Encryption Standard (AES) [18], and iii) Adaptive Differential Pulse-Code Modulation (ADPCM) [18]. These algorithms were chosen because they cover a variety of domains, such as arithmetic (MM), security (AES), and media processing (ADPCM). Moreover, they range from a data-flow oriented algorithm (MM) to a control-flow oriented algorithm (ADPCM). The processor-based design has a Microblaze (MB) soft-core processor, which is a 32-bit 5-state pipeline Reduced Instruction Set Computer (RISC) soft processor. The architecture of both HLS-based designs and processor-based design are shown in Fig. 3.

Three HLS-based designs for each benchmark application were generated in Xilinx Vivado HLS, each one with different optimization strategies. The optimizations applied are summarized in Table I. The optimizations were chosen focusing on speeding up the critical path of the algorithms. In the MxM benchmark, the main optimization effect was the increase in pipeline depth, which resulted in a 21-stage pipeline in *HLS1*, 220-stage pipeline in *HLS2*, and a 3235-stage pipeline in *HLS3*. It is worth noting that in *MxM-HLS3*, the *array partition* directive was applied aiming to increase the input data throughput, which significantly increased its pipeline depth and performance. In the AES and ADPCM benchmarks, the main optimizations applied were also related to the data parallelism (unroll loop and pipeline), although at different levels of granularity. This was done since they present a lot of data dependencies among their several internal functions and to not modify the original benchmark programs. The inputs and outputs of the three benchmarks use the *resource* directive to force the variables to be synthesized as Advanced eXtensible Interface Stream (AXI-S) channels in order that all the HLS-based designs have the same interface architecture, as shown in Fig. 3(b). Moreover, the input data of the three benchmarks are fixed.

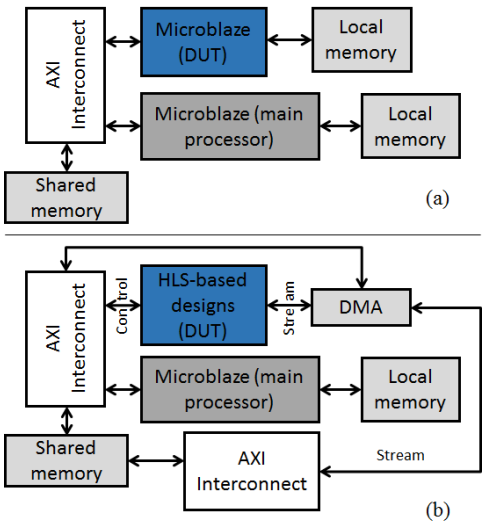

Fig. 3. Architecture of the processor-based design (a) and he HLS-based design (b).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2648978, IEEE Transactions on Nuclear Science

TABLE I
OPTIMIZATION STRATEGIES APPLIED IN EACH HLS-BASED DESIGN FOR EACH BENCHMARK PROGRAM

Benchmark	Optimization strategy		
	HLS1	HLS2	HLS3
MxM	- None	- *Pipeline* in the middle loop	- *Pipeline* in the middle loop - *Array partition* in the inputs - Function *inline*
AES	- None	- *Unroll* the loops of the main - *Pipeline* the encryption function - *Pipeline* the decryption function	- *Unroll* the loops of the main - *Pipeline* the encryption function - *Pipeline* the decryption function - *Pipeline* the key function
ADPCM	- None	- *Pipeline* fine grain	- *Pipeline* coarse grain

Thus, the outputs are known. All the input and output elements are implemented with Block RAM (BRAM) memories protected with Error Correction Code (ECC) for the designs tested under radiation. All designs were synthesized into a Xilinx Artix-7 FPGA, part XC7A100TCSG324-1, embedded into a Digilent Nexys 4 board and with an input clock of 100 MHz.

Resource usage, performance results in terms of clock cycles, and number of critical bits obtained by fault injection for each case-study design are presented in Table II. The workload considered for each benchmark was 32768 bits (1024 values of 32-bit each) for the MxM, 1024 bits (32 values of 32-bit each) for the AES, and 3200 bits (100 values of 32-bit each) for the ADPCM. Results show that as the optimization level of the HLS-based designs is increased, the number of resources used (15.6 times for the MxM in the worst case) and the number of critical bits increases as well (8.0 times for the MxM in the worst case). However, the highest increase is related to the performance (73.5 times for the MxM in the worst resource increase). This means that increasing the hardware parallelism using more resources, greatly increases the workload capability of the system. Moreover, the use of the *array partition* directive also plays an important role by decreasing memory bottlenecks. By comparing the use of a processor-based approach versus an HLS-based design, one can see that, in general, HLS-based designs use more area. However, the performance improvement they provide is even higher (31.0 times when comparing *ADPCM-MB* versus *ADPCM-HLS1* in the worst case).

The number of critical bits for each case-study design is also shown in Table II. Comparing the obtained results from fault injection with the number of configuration bits in the DUT area, one can observe that a small fraction of the configuration bits is, in fact, critical. Therefore, a slight variation in the percentage of critical bits over the total number of configuration bits among the different HLS-based designs of each benchmark circuit can be observed. From the obtained percentages, it is also possible to note that the more control-flow oriented an algorithm is, the higher the percentage of critical bits its generated HLS-based designs may present. Similarly, these results reflect in the comparison with the processor-based design.

V. CROSS SECTION AND MWBF RESULTS

Radiation experiments were carried out with heavy ions at Laboratório Aberto de Física Nuclear at Universidade de São Paulo (LAFN-USP), Brazil [19]. The ion beams were produced and accelerated by the São Paulo 8UD Pelletron Accelerator. A standard Rutherford scattering setup was included using a gold foil to achieve a very low particle flux in the range from 10^2 to 10^5 particles·cm^{-2}·s^{-1}, as recommended by the European Space Agency (ESA) for SEU tests [20]. The experiment was performed in vacuum. The SEU events were observed using a ^{16}O beam scattered by a 184 μg/cm² gold target, with an energy of 56 MeV, which provides an effective LET on the active region of 5 MeV/mg/cm² and penetration in

TABLE II
RESOURCE USAGE AND PERFORMANCE RESULTS OF EACH CASE-STUDY DESIGN

Design version	Resources and performance						
	# LUT	# FF	# DSP	# BRAM	# Configuration bits	# Critical bits (% of configuration bits)	t_{exec} (clock cycles)
MxM							
MB	1159	1452	0	0	524342	56450 (10.67%)	41611263
HLS1	755	944	5	3	435879	11733 (2.69%)	366354
HLS2	2397	3374	10	3	1000132	47164 (4.72%)	19613
HLS3	11787	13924	160	33	4606244	94692 (2.06%)	4982
AES							
MB	1159	1452	0	0	524342	56450 (10.67%)	104905
HLS1	5780	2380	0	17	896694	79205 (8.83%)	3188
HLS2	25243	6511	0	19	2023681	198661 (9.82%)	2269
HLS3	34725	12343	0	15	3823592	350657 (9.17%)	1317
ADPCM							
MB	1159	1452	0	0	524342	56450 (10.67%)	1091250
HLS1	5435	6385	103	8	1898542	230420 (12.14%)	35144
HLS2	4911	6157	103	14	1806674	232686 (12.88%)	17316
HLS3	14959	18681	240	14	6023727	817025 (13.56%)	8462

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2648978, IEEE Transactions on Nuclear Science

Si of 28.5 µm. To achieve the desired particle flow, the DUT was positioned at a scattering angle of 90º, which resulted in an average flux between $2.0x10^2$ and $2.5x10^2$ particles·cm^{-2}·s^{-1}. This configuration was chosen based on several trials and was the most suitable in terms of particle flux and number of errors. Each experiment run was set by the total number of errors observed (60) in the DUT output. Errors in the soft-core processor that sends data to the DUT are not considered in the count. All other uncertain errors observed are considered in the errors bars. For the heavy ion experiments, the package of an XC7A100TCSG324-1 device was thinned to allow irradiated particles to penetrate the active region of the silicon.

A small Finite State Machine (FSM) (one-hot encoding) implemented with Triple Modular Redundancy (TMR) performs the detection of incorrect outputs by comparing the computed results of the DUTs with their expected reference values stored in embedded BRAMs and protected with Error Correction Code (ECC). A host computer monitors the FSM through a serial interface. If there are no differences in the results, the FSM sends an alive signal to the host computer at each execution set. Otherwise, if differences are found, the FSM sends the number of mismatches to the host computer and then the FPGA is reset. Timeouts in the DUT are monitored through a watchdog circuit in the FSM. The host computer also has a watchdog circuit to monitor timeout occurrences in the FSM. In case of timeout, the FPGA is reset.

The obtained dynamic cross section results from both fault injections and radiation experiments are presented in Fig. 4, 5, and 6. With regard to the fault injection results, a reference static cross section of $1.11x10^{-9}$ cm² (effective LET = 5.25 MeV/mv/cm^2 and flux = $1.32x10^2$ particles·cm^{-2}·s^{-1}) was adopted, which was obtained from previous heavy ion experiments with the same device. Notably, there are differences in the obtained values from fault injections and radiation experiments (3.72 times in average and 11.9 times for the worst case - *ADPCM-HLS3*). In general, data from fault injection campaigns are more pessimistic than the ones from the radiation experiments. Such behavior was already expected, since our fault injection methodology is more deterministic and gives us a very fine fault granularity, which enables us to estimate the worst-case cross section of a design without considering side-effects, such as beam variations and fault masking by the design. Therefore, it is possible to determine precisely the exact number of bits that are critical and consider them in the estimated dynamic cross section. It is worth noting that radiation experiments cover the whole architecture, while the fault injection campaigns cover only the DUT (shown in Fig. 3). However, for a same benchmark application, the HLS-based design (DUT) is the only part of the system that changes among the different versions. This somehow normalizes the sensitivity of the processor among the designs, since the soft-core processor and peripherals never change among the designs. It is also important to highlight that the proposed reliability analysis is capable of estimating the dynamic cross section trend for different HLS-based designs of a same benchmark application, even with the mentioned differences.

By analyzing the obtained dynamic cross section values from fault injection for all HLS-based designs, one can notice a correlation with the number of critical bits and hence the resource usage. The higher the optimization level, the more resources a design uses and, consequently, the more critical bits it has. Moreover, there is also a correlation with the behavioral nature of the algorithm. Data-flow oriented algorithms are characterized by a significant amount of arithmetic operations with few control dependencies, while control-flow oriented algorithms contain many relational operations. Therefore, the variables and operations in control-flow designs may present more vulnerabilities to failures when

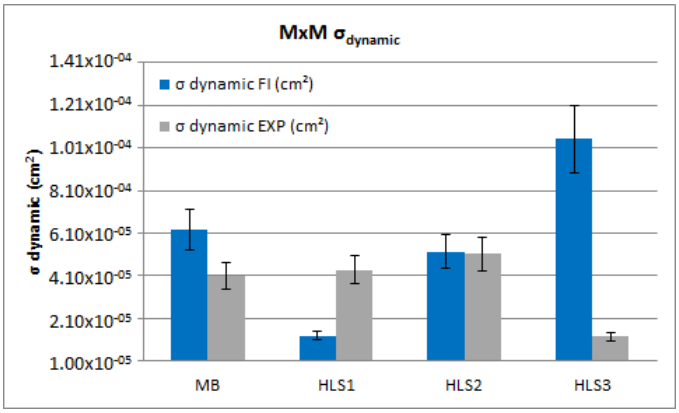

Fig. 4. Dynamic cross section results obtained for the MxM designs from both Fault Injection (FI) and Radiation Experiment (EXP).

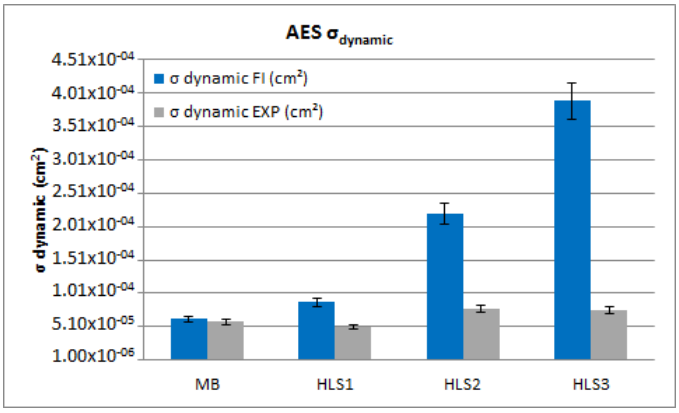

Fig. 5. Dynamic cross section results obtained for the AES designs from both Fault Injection (FI) and Radiation Experiment (EXP).

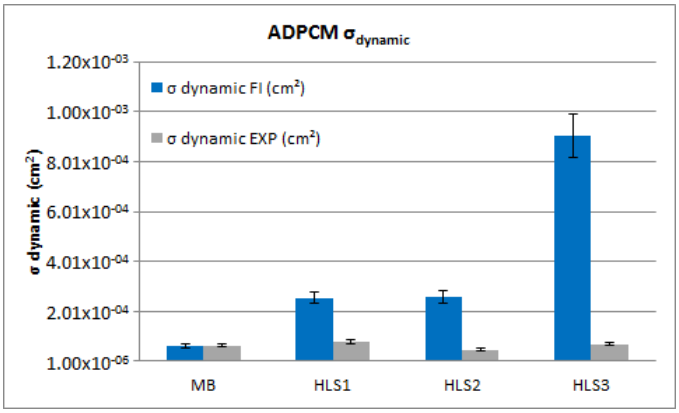

Fig. 6. Dynamic cross section results obtained for the ADPCM designs from both Fault Injection (FI) and Radiation Experiment (EXP).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2648978, IEEE Transactions on Nuclear Science

compared to data-flow ones, as observed when comparing the results of Table II with the obtained cross sections.

The obtained MWBF results from both fault injections and radiation experiments are depicted in Fig. 7, 8, and 9. A comparison between the fault injection and radiation experiment results shows that there are slight differences in the obtained values (6.8 times for the worst case - *MxM-HLS1*). However, it is worth mentioning that the proposed reliability analysis is also capable of estimating the MWBF trend of different HLS-based designs for the same benchmark application. By analyzing the obtained MWBF values from both fault injections and radiation experiments, it is possible to observe that the use of different optimization strategies has a direct impact in the final MWBF values of the different HLS-based. Such behavior is mainly guided by the execution time improvement achieved with the different optimization strategies chosen and cannot be observed if only the dynamic cross section measurements are taken into account. The only performance decrease observed after applying some optimization strategy is related to the AES benchmark. This is related to how the benchmark is encoded. Moreover, we chose not to modify the algorithm. In this case, the main AES function has the following structure: *for* loop, *encrypt* function call, *for* loop, *decrypt* function call, *for* loop. In addition, inside the *encrypt* and *decrypt* functions, there are calls to a *key* function, which calculates the key of the algorithm. With such organization, whatever the optimization strategy applied in the *for* loops or the internal functions, there will always be a data bottleneck among such blocks. This behavior cannot be observed in the MxM benchmark because the algorithm is quite simple and data-flow oriented, which facilitates data parallelization. Concerning the ADPCM benchmark, even though the algorithm is control-flow oriented, it is more modularized than the AES and with less data exchange among them. The lower MWBF values for the processor-based design were already expected due to their shorter execution times.

A comparison between the obtained dynamic cross section by fault injection (the more pessimistic values) and the performance rate improvement for each HLS-based design compared to its corresponding processor-based version is presented in Fig. 10, 11, and 12. The performance rate is calculated as shown in (6).

$$Performance\ rate = \frac{t_{proc} * \sigma\ dynamic_{proc}}{t_{HLS} * \sigma\ dynamic_{HLS}} \qquad (6)$$

The performance rate relation states that whenever the speed-up is higher than the increase in cross section ($\frac{t_{proc}}{t_{HLS}} > \frac{\sigma\ dynamic_{HLS}}{\sigma\ dynamic_{proc}}$), the faster configuration computes a larger workload before experiencing a failure, and the higher the *Performance rate* result is. Thus, it can be concluded that, the designs with the best trade-off between reliability and performance for each benchmark application are *MxM-HLS3*, *AES-HLS1*, and *ADPCM-HLS2*.

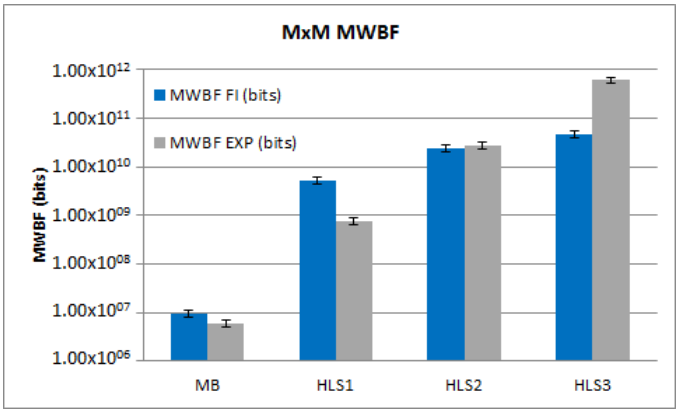

Fig. 7. MWBF results obtained for the MxM designs from both Fault Injection (FI) and Radiation Experiment (EXP).

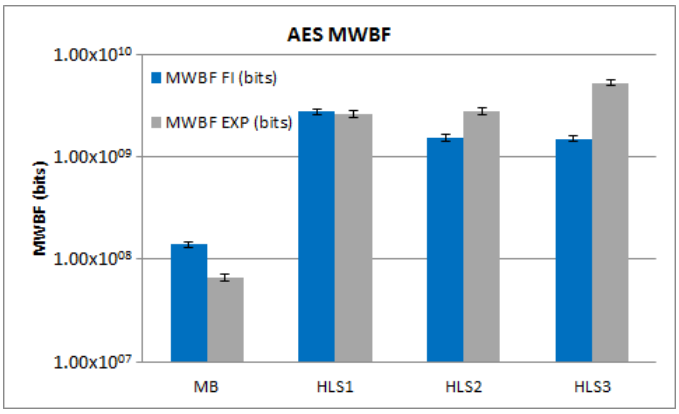

Fig. 8. MWBF results obtained for the AES designs from both Fault Injection (FI) and Radiation Experiment (EXP).

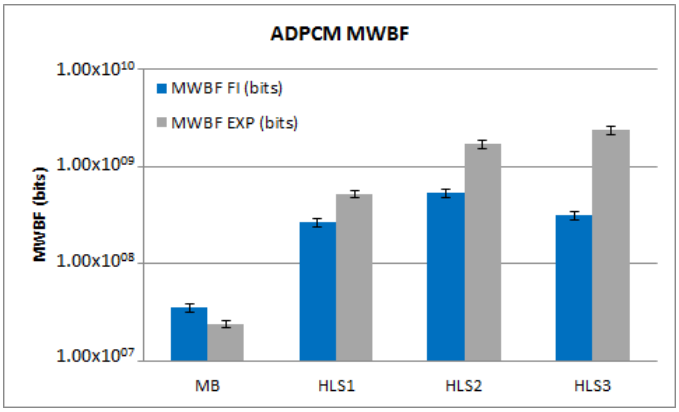

Fig. 9. MWBF results obtained for the ADPCM designs from both Fault Injection (FI) and Radiation Experiment (EXP).

VI. Conclusions

In this work, the reliability and performance trade-offs of HLS-based designs under soft errors in an SRAM-based FPGA were investigated. A method for analyzing the susceptibility of designs generated by an HLS tool was presented, which in this case was the Xilinx Vivado HLS tool. The method takes into account not only the susceptible area details of the designs, but also their performance efficiency in order to estimate the trade-offs between the dynamic cross section and MWBF. Results show that, in general, the estimation of the dynamic cross section and MWBF of an

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNS.2017.2648978, IEEE Transactions on Nuclear Science

HLS-based design through our proposed flow is a suitable method for predicting their trend before radiation experiments. Regarding the differences found between fault injections and radiation experiments, further investigations are required in order to reach to a proper conclusion.

With regard to the effects of optimization strategies in the final HLS-based designs, results reveal that HLS tools have evolved in the last decade by providing very structured and optimized RTL codes. The influence of HLS optimizations in the dynamic cross section HLS is very low (increases up to 8 times for the MxM benchmark) when compared to the design performance enhancement (increases up to 5000 times for the ADPCM benchmark), which contributes significantly to increase the MWBF of the designs.

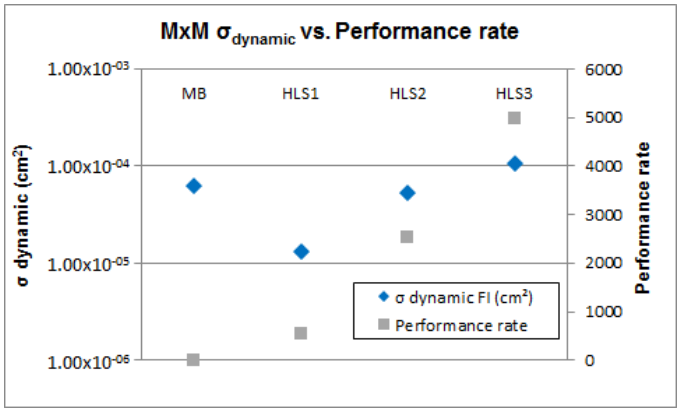

Fig. 10. Comparison between the dynamic cross section and performance rate for the MxM designs from Fault Injection (FI) results.

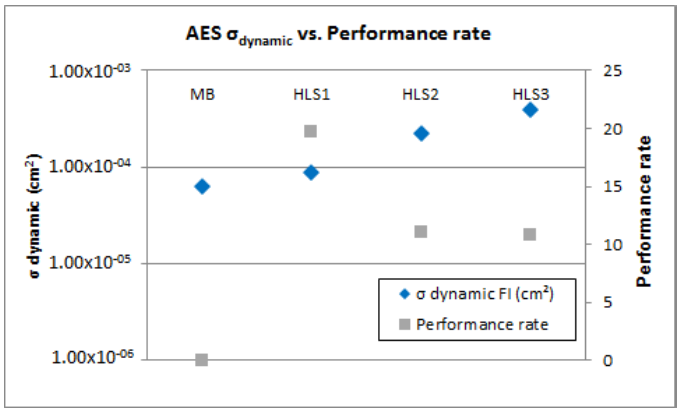

Fig. 11. Comparison between the dynamic cross section and performance rate for the AES designs from Fault Injection (FI) results.

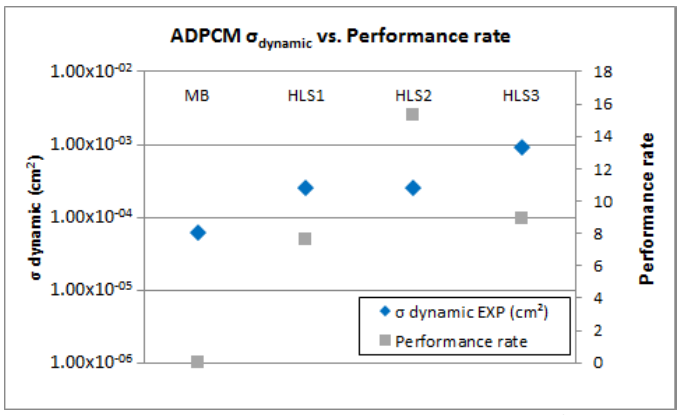

Fig. 12. Comparison between the dynamic cross section and performance rate for the ADPCM designs from Fault Injection (FI) results.

REFERENCES

[1] M. Wirthlin, D. Lee, G. Swift, and H. Quinn, "A Method and Case Study on Identifying Physically Adjacent Multiple-Cell Upsets Using 28-nm, Interleaved and SECDED-Protected Arrays," *IEEE Trans. Nucl. Sci.*, vol. 61, n. 6, pp. 3500-3505, Dec. 2014.

[2] M. Wirthlin, "High-Reliability FPGA-Based Systems: Space, High-Energy Physics, and Beyond," *Proc. of the IEEE*, vol. 103, n. 3, pp. 379-389, Mar. 2015.

[3] S. Rehman, M. Shafique, and J. Henkel, *Reliable Software for Unreliable Hardware: A Cross Layer Perspective*, 1st ed. Switzerland: Springer CH, 2016.

[4] Xilinx (2016). Applications [Online]. Available: http://www.xilinx.com/.

[5] X. Iturbe, D. Keymeulen, P. Yiu, D. Berisford, K. Hand, R. Carlson, and E. Ozer, "Towards a Generic and Adaptive System-on Chip Controller for Space Exploration Instrumentation," *Proc. of the 2014 NASA/ESA Conference on Adaptive Hardware and Systems*, Jun. 2015, pp. 1-8.

[6] M. Husejko, J. Evans, and J. C. R. da Silva, "Investigation of High-Level Synthesis Tools' Applicability to Data Acquisition Systems Design Based on the CMS ECAL Data Concentrator Card example," *Journal of Physics: Conference Series*, 664 (2015) 082019, pp. 1-8.

[7] R. Nane, V.-M. Sima, C. Pilato, J. Choi, B. Fort, A. Canis, Y. T. Chen, H. Hsiao, S. Brown, F. Ferrandi, J. Anderson, and K. Bertels, "A Survey and Evaluation of FPGA High-Level Synthesis Tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, n. 10, pp. 1591-1604, Oct. 2016.

[8] S. Windh, M. Xiaoyin, R. J. Halstead, P. Budhkar, Z. Luna, O. Hussaini, and W. A. Najjar, "High-Level Language Tools for Reconfigurable Computing," *Proc. of the IEEE*, vol. 103, n. 3, pp. 390-408, Mar. 2015.

[9] Xilinx. Vivado Design Suite - User Guide - High-Level Synthesis. UG902 (v2014.3) Oct. 1, 2014.

[10] A. Canis, J. Choi, M. Aldham, V. Zhang, A. Kammoona, J.H. Anderson, and T. Czajkowski, "LegUp: High-Level Synthesis for FPGA-based Processor/Accelerator Systems," *Proc. of the 19th ACM/SIGDA Int. Symp. on Field Programmable Gate Arrays*, Feb. 2011, pp. 33-36.

[11] Xilinx, "Soft Error Mitigation Using Prioritized Essential Bits," XAPP538 (v1.0), 2012.

[12] JEDEC. Measurement and Reporting of Alpha Particle and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices, JESD89A (Revision of JESD89, Aug. 2001), Oct. 2006.

[13] P. Rech, L.L. Pilla, P.O.A. Navaux and L. Carro, "Impact of GPUs Parallelism Management on Safety-Critical and HPC Applications Reliability," *Proc. of the 44th Annual IEEE/IFIP Int. Conf. on Dependable Systems and Networks*, Jun. 2014, pp. 455-466.

[14] J. Tonfat, L. A. Tambara, A. Santos, and F. L. Kastensmidt, "Method to Analyze the Susceptibility of HLS Designs in SRAM-Based FPGAs Under Soft Errors," *Lecture Notes in Computer Science*, vol. 9625, pp. 132-143, Mar. 2016.

[15] R. Velazco, F. Foucard, and P. Peronnard, "Combining Results of Accelerated Radiation Tests and Fault Injections to Predict the Error Rate of an Application Implemented in SRAM-Based FPGAs," *IEEE Trans. Nucl. Sci.*, vol. 57, n. 6, pp. 3500-3505, Dec. 2010.

[16] Xilinx, "Device Reliability Report," UG116 (v9.4), 2015.

[17] Xilinx, "A Zynq Accelerator for Floating Point Matrix Multiplication Designed with Vivado HLS," XAPP1170 (v2.0), 2016.

[18] Y. Hara, H. Tomiyama, S. Honda, H. Takada, and K. Ishii, "CHStone: A Benchmark Program Suite for Practical C-based High-Level Synthesis," *Proc. of the IEEE Int. Symp. on Circ. and Syst.*, May 2008, pp. 1192-1195.

[19] V. A. P. Aguiar, N. Added, N. H. Medina, E. L. A. Macchione, M. H. Tabacniks, F. R. Aguirre, M. A. G. Silveira, R. B. B. Santos, and L. E. Seixas, "Experimental Setup for Single Event Effects at the São Paulo 8UD Pelletron Accelerator," *Nuc. Instr. & Methods in Phys. Research*, vol. 332, 2014, pp. 397-400.

[20] ESA. ESA/SCC Basic Specification No. 25100: Single Event Effects Test Methods and Guidelines. ESA, Noordwijk, Netherlands, 2005.