FPGA In Reliability and Safety Critical Application

Neaz Mahmud
Electronic Engineering
Hochschule Hamm Lippstadt
Lippstadt , Germany
Neaz.mahmud@stud.hshl.de

***Abstract*—The growing complexity of FPGA-based systems Material designs and reduced time to market have driven the adoption of new, more focused design methodologies to meet today's need for high performance circuits. High Level Synthesis (HLS) tools can generate Register Transfers Level designs (RTL) from high-level software programming Languages. The very high levels of integration achieved by VLSI technologies for SRAM-based Field Programmable Gate Arrays (FPGAs) lead to a high frequency of transient errors caused by Single Event Upsets (SEU) in the configuration memory of the FPGAs. Since the configuration memory determines which circuit implements a SRAM-based FPGA, any changes caused by SEUs can make dramatic change to the implemented circuit. When such devices are used in safety critical applications, fault tolerance techniques are required to reduce the effects of SEUs in the FPGA configuration memory.**

I. Introduction

Since Xilinx developed the first FPGA device with the XC2064 chip in 1984, FPGA technology has grown tremendously in terms of flexibility, reliability and computing power. While not comparable to ASIC technology in terms of computational power or silicon surface area, FPGA technology offers outstanding performance, low one-time design costs, and extremely short time to market, and has established itself in many application areas. SRAM-based FPGA devices are particularly used in many application areas such as broadcasting, wired and wireless communication systems [1], cryptography and network security [2], air [3], aerospace and defense cite r4, railways [4], control of industrial and nuclear power plants [5]. FPGA devices can be reconfigured for each task, resulting in better resource utilization. However, SRAM-based FPGA devices are still rarely used in those parts of systems related to system security, due to its vulnerability to SRAM-based memory configuration errors [6]. On the other hand, in recent years, a number of special conferences and workshops, such as the NASA/ESA Conferences on Adaptive Hardware and Systems and the Workshops on Programmable Logic Devices for the Military and Aerospace, have shown great interest in using of SRAM FPGA based devices. devices in security and mission-critical applications.

The remainder of this paper is organized as follows: in Section 2, we get a brief overview of safety critical system ; in Section 3, a review of the fault models are discussed ;lastly on Section 4, the solutions are presented.

II. Safety Critical System

A. What Is Safety Critical System

A safety-critical system is a system whose failure could result in death or serious injury to persons, loss or serious damage to equipment, or damage to the environment. In the functional safety standard IEC 61508-2 [7], relating to safety electrical / electronic / programmable electronic systems (E / E / PE) operating in a low demand operating mode, the lower limit of the target fault measurements is set with an average probability of 10^{-5} dangerous failures for hours of operation. On the other hand, for E / E / PE safety systems operating in a demanding continuous mode of operation, the lower limit is set to a average probability of 10^{-9}

B. Safety Requirements Specifications

In this phase, the requirements for the FPGA-based system are extracted and analyzed. In particular, it is recommended to identify requirements that include features that allow the system to achieve and maintain a certain level of security. The system requirements specification should contain details relevant to the design to achieve the Safety integrity level and required target failure measure for the safety function [8]. In particular, ECSS-Q-ST-60-02C provides the following additional requirements for the appearance of radiation interference:

- Error Management,
- Test Equipment on The Ground and In Flight,
- Evidence of failure coverage required during testing.

At the end of this phase, there is a document that fully captures and defines the FPGA-based system requirements. This document must be complete, unambiguous, clear and accurate, verifiable and testable.

C. Standards

A general framework for the design and development of safety-critical hardware and software systems is the IEC 61508-2 [7] and IEC 61508-3 [9] for equipment and Software systems. There is currently specific regulation for the design of FPGAs in safety-critical systems Aerospace application only: ECSS-Q-ST60-02C [10]. Moreover, in the airborne field the RTCA/DO-254 [8] standard can be applied to design and development of electronic systems and therefore also on FPGA-based systems. In all other areas of application, the applicable regulations must be adopted for the design and

development of software and hardware systems. These are the reference standards, particularly in the automotive application field ISO/DIS 26262-5 [8] and ISO/DIS 26262-6 [8], and in the railway field CENELEC EN 50128 [11] and CENELEC EN 50129 [12]. Since the design of modern VLSI systems always includes the use of high-level hardware description languages, all standards agree on the application of a V-shaped design flow, which is inspired by the classic software design flow. Fig. 2 shows the V-shaped design current derived from standard IEC 61508-2 [7] for ASIC development life cycle which could also be used for FPGA development process.

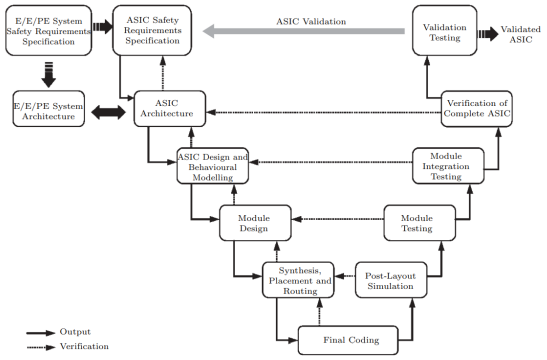

Fig.2. ASIC development lifecycle (the V-shaped model)[10].

D. *Advantages,Disadvantages and Safety Related Issues*

As analyzed in depth by Kuon and Rose [13], FPGA-based designs tend to be larger, slower and consumes much more energy than fully customized designs. However, they are becoming more common in all application areas and interest in using FPGA devices in safety related applications such as: space missions or rail systems, is growing. This is due to the fact of two main factors of low cost and short time to market [14]. A fully customized design requires very expensive CAD tools for simulation, verification, synthesis and floor design. In addition, a fully customized design has a great number of engineers working for many months. To finish, Fully custom designs may require the use of masks and might costs several million dollars to drive the lithography process. On the other hand, the cost of an FPGA device varies from a few tens to several thousand dollars. Plus CAD tool licenses for FPGA-based designs are much cheaper than full customization tools Drafts. The only major cost associated with FPGA-based Designs is the cost of the development team. Thus, especially for small and medium-sized productions, FPGAs are often the best technology choice. From a time-to-market perspective. The main problems related to the design of FPGA-based systems and their acceptance in safety-critical application areas are the lack of specific standards and the strong sensitivity of FPGA devices to the effects of radiation. Given the relative youth of FPGA technology, and its not yet widespread adoption in the field of security fields of application, there are very few design standards specifically focused on FPGA technology.Radiation, both in space and at ground level, can cause a system to fail. In particular, radiation striking the silicon surface of digital circuits can alter the content of storage elements. Radiation hitting SRAM-based FPGA devices can have even worse effects,they could thus permanently damage the contents of the configuration memory (until the device is reconfigured) and thus affecting the functionality implemented in the device.

III. FAULT MODELS

Various types of failures can be occurred in a FPGA in Safety Critical System. Those types mainly can be divided into two groups **(i) Random Faults** and **(ii) Systematic Faults**

A. *Random Faults*

A fault in a module can occur for various reasons. A random hardware fault such as a faulty transistor can result in the misbehavior of a module. For example, **in Error! Reference source not found.**, a random hardware failure can cause failures in the clock generation function or the reset feature. Latch up, bridging, single event upset/transient could be some examples of Random Faults. Random faults can be group into **Hard error** and **Soft error**

1) Hard Error: Hard errors in FPGA can be also defined as permanent errors which can not be solved or mitigated

2) Soft Error: Earlier, soft errors were generally ignored and reliability predictions concentrated on hard errors but when IEC 61508-2:2010 mentioned soft errors they could no longer be ignored. Soft errors are largely caused by alpha particles from the packaging materials and neutron particles caused by galactic sources. At ground altitudes the two contribute roughly equally. While the alpha particles cannot penetrate deeply into silicon they are coming from on top of the die so they are hard to shield against but the literature suggests polyamide can help. On the other hand, neutron particles are hard to shield against without using several meters of cement or lead. Therefore, mitigation is needed either at the CMOS device level, the module level on the integrated circuit , at the system level on the IC or at the top-level system level.

B. *Systematic Faults*

Systematic faults may be caused during design, development, and manufacturing due to errors in specification or implementation. Systematic errors can be divided into two categories (i) Design Errors and (ii) Tools Errors . Systematic fault injection is widely used in the analysis of the fault tolerance of FPGA designs. Systematic fault injection is used on Xilinx Virtex-6 FPGAs to identify critical bits that cannot be repaired by scrubbing alone because they cause the design to enter an incorrect state, requiring reset or power cycle. TURTLE, a dual-FPGA fault injection platform based on Xilinx Artix-7 FPGAs is proposed in [15]. The authors state that by using multiple TURTLE platforms in parallel, systematic fault injection tasks may be accelerated. A system for accelerating systematic fault injection by parallelization

across multiple Xilinx Zynq-7000 FPGAs of the same type is proposed in [15].

The works discussed above do not consider device- or temperature dependency, even though results described in [15] hint at (presumably) random processes that lead to varying criticality results in subsequent fault injection campaigns on the same design. It is implicitly assumed that a single systematic fault injection run on a single device is sufficient to determine the critical bits in a design.

IV. SOLUTIONS

As discussed in [16], testing alone cannot guarantee such requirement; combination of fault-tolerant approaches, such as replication and diversity, as well as testing and other techniques such as Failure Mode and Effects Analysis (FMEA) and reliability analysis methods improve the reliability of the system, but the result is still far from the goal of 10^{-9}. It is then necessary to complete the debugging (i.e. testing) and fault tolerance strategies with a fault avoidance strategy with the goal of producing high quality systems that are as free as possible of Systematic errors. This goal can be achieved with rigorous development processes conducted according to standards that explicitly address the Requirements for safety-critical systems. Many standards are available to developers of safety-critical systems, but most not directly address issues specific to FPGA technology, or offer only limited advice.

A. *Handling of Hardware Faults*

This section should give a brief comparative overview on the handling of hardware faults in MCU and FPGA based embedded systems.

1) Hardware Design Process: The hardware design process has a high impact on the quality of the resulting hardware. Steps to achieve specific levels of quality (testing, burn-in) are very important to avoid malfunction in the next device.

2) Device Technology: Dealing with hardware reliability, publications often focuses on specific problems in aerospace or space applications (e.g.: SEU (Single Event Upset), SEL (Single Event Latch-up), TID (Total Ionizing Dose)). These effects are less critical at ground level, but still exist [17], [18], [19]. Especially [19] gives a good overview how memory cells could be affected at ground level. These effects have to be considered in safety-critical systems since a fault in the hardware of such a system could lead to catastrophic consequences.To harden the silicon structure, certain measures have been introduced.For example, alpha particles can be protected with a thick layer of polyimide before packaging. However, this does not work for high energy cosmic radiation [17]. TID ratings can be improved through special circuit design techniques and shielding methods [20], and SEL has been all but eliminated for many FPGA devices through the incorporation of special current limiting circuitry [20].SEU cannot be completely avoided by device techniques and must be addressed in every CMOS device. One question is, whether some device techniques are suited better than others to build reliable devices. For example," old" silicon techniques could be more robust since increasing device density and larger dies of current devices makes these less reliable [21]. However, there have been improvements in silicon techniques as can be seen in [17]. One-time programmable (OTP) memories, as for example anti fuse technology, are very robust to soft errors [22]. However, they are less flexible and it should be borne in mind that even devices based on anti-fuse program/configuration memories contain SRAM for dynamic data storage. In-circuit re-configurable memories allow much higher flexibility than OTP devices. In [21] they state that the most promising in-circuit reconfigurable devices from the TID, SEL, and SEU point of view are those that are based upon FLASH or EEPROM.

3) Hardware Architecture (Fault Avoidance): Certain hardware architectures allow techniques to avoid and to tolerate hardware faults.The hardware architecture requires a lot of thought, as it is the hardest to change later in the design cycle. Standard hardware considerations should apply, e.g. signal integrity, power decoupling and bypass, input protection, and component derating.Aspects of the design that don't normally require attention may also need attention, such as implementing communication buses between devices.Aspects of the design that don't normally require attention may also need attention, such as implementing communication buses between devices.Both parallel and serial buses have advantages and disadvantages. Serial buses make it easier to identify stuck errors (signal is permanently high or low), but can reduce system data throughput. MCUs and FPGAs are affected in different ways by transient hardware failures. In the case of MCUs, the program memory, data memory, and general and special purpose registers can be damaged. In case of FPGAs the configuration memory, Flip-Flops and, if available, the data memory could be corrupted.Changes in configuration memory could fundamentally alter the behavior of the FPGA. However, the same applies if certain registers of the [23] MCUs are affected. One design area that requires careful design and analysis in FPGA and MCU based systems is often one of the most easily overlooked areas, and that is the Joint Test Action Group (JTAG) interface (IEEE 1149.1 standard) [**?**] . JTAG interface is used in digital electronics to program FPGA, Flash, etc. and it can also be used to debug FPGAs and MCUs, and is increasingly used in production tests to ensure that the board is manufactured correctly. It is important to ensure that this interface remains unusable under normal use. However, no general differences in hardware fault avoidance in the architectures of FPGAs and MCUs could be found.

4) Hardware Verification: Another important aspect in safety-critical systems is to show that the hardware will always behave as intended. One approach is the formal verification of the hardware built (with respect to specification) which is applicable for both MCU and FPGA

in the same way. If devices are taken which have been used in other applications without failures for a certain time, they can be considered as" proven in use" which is sufficient for less critical systems [7]. However, this implies not to use new state of the art technologies which could lead to the problems aforementioned. It might be interesting to analyze whether FPGA or MCU hardware have the higher risk of unrevealed hardware design faults after being used in a number of applications. If new hardware is to be qualified (temperature, humidity, altitude, radiation, shock vibration, acceleration etc.).it is possible to use only one type of FPGA which can subsequently be used for a large number of applications (generic hardware, as long as it is reasonably sized), while a single MCU is unlikely to be used for many different applications due to specialization (fixed point/ floating point, on-chip devices, etc.).

B. Handling of Software Faults

Increasing complexity, real-time requirements and hardware/software dependencies make developing reliable software for embedded systems a non-trivial task. Additionally, software measures for hardware fault mitigation, as described in the previous section, further increase the software complexity. Handling these challenges involves several which will be looked at in the following.. . .

1) Design Process: The software design processes of MCUs and FPGAs are different, especially from the programming language point of view. The two main programming languages used for MCUs today are C/C++ and ADA and there has been much work conducted on which is more suitable for safety-critical applications. Many governmental and international organizations which develop safety-critical systems and safety standards as [9] prefer ADA rather than C. However, the automotive industry has developed a subset of the C language called MISRA C which has been designed to address C's failings in the safety-critical arena. Although currently most FPGAs are programmed in VHDL and Verilog, recent publications propose the use of languages developed for CPU based systems as ADA [24] or Esterel [25], [26]. They argue, that classical HDLs are not suited for more complex designs and that languages as Esterel allow formal verification. However, it is important that PLD's cannot be treated as CPU based systems, especially regarding the HDL used to implement the design. While the HDL may look like a software program it is not, instead it describes how the implemented hardware is to function. Unlike a software program which will execute sequentially, the implemented FPGA will be operating simultaneously. Therefore, the HDL development process will be subject to different rules and procedures than the software development to ensure fault tolerance or avoidance. However, in embedded software design the underlying hardware has to be considered in safety-critical applications, since many faults appear at this hardware/software boundary (especially timing problems). This consideration has to be done manually or by suitable tools and might be more challenging for FPGA than for MCU designs. The reuse of common components across the system reduces the possibility of errors being introduced by different design engineers implementing the same function, this approach can also lead to a reduced time to market. This library of components can be either written at the start of a project or built up over a series of projects. A similar advantage using FPGAs is the possibility of using Commercial Off The Shelf (COTS) components provided by the third parties (in VHDL or Verilog). These COTS components are available from both the device manufacturers and third-party specialists, and due to the similar architecture of FPGA's these third-party components are normally suitable for use across different FPGA families. The use of these modules will further reduce development time, provided they are subjected to an in-depth test bench which checks the accuracy, time response and behavior during failure modes. COTS components also have a much wider user base across many different applications. This wider user base and the varied applications of the components will inevitably mean that errors within the component design are identified quicker and corrected by the manufacturer.

2) Software Verification: According to [27], verification has become the bottle neck of the design process. Verification is especially important for safety-critical systems [28], [29], and it is desirable to achieve very high-test coverage (100 percent is desired) on all of the required metrics for this reason. While this does add to the development timescales, work done by [30] indicates that an increase in the code coverage is likely to increase reliability. While this work was conducted on traditional software there is no reason to believe the results could not be extended to HDL code coverage (in [27] an introduction to different coverage metrics used in HDL design can be found). FPGAs offer a benefit over a more traditional MCU based approach in that execution is more deterministic on a PLD than on an embedded microprocessor using techniques as interrupt processing and cache memory [17]. This eases verification of critical real-time systems and the fact that a frequently encountered source of failures in hard real-time systems is the inability of the system to deliver the required services by the required deadline under various workload conditions [31] makes this a very important aspect. Techniques like assertion-based verification and code coverage analysis demand modifications of the code. In case of CPU based systems, this changes the timing properties which is problematic for real-time systems. In FPGAs, assertions usually do not affect the timing properties of the application. Formal verification of the software developed is another important issue. Approaches are available for systems written for FPGAs [32], [33], [34] and for MCUs [35] used in embedded systems. However, formal verification of real-time properties remains a challenge for model checking of MCU code. Formal Equivalence Checking used in FPGA designs verifies that the synthesis and place and route stages (which typically occur after functional verification has been completed) have not introduced or removed logic therefore

changing the function of the logic. This verification is needed, since design tools will attempt to modify and optimize the logical structures described within the HDL to make it best fit the logical structures available within the target architecture. It is therefore important that in a device intended for safety-critical applications these downstream processes do not change the behavior of the module under error or error free conditions. Formal Equivalence Checking provides the ability to check that the functionality described in a HDL (typically Register Transfer Level) is the same as that of the implemented net-list. Figure 2 from an example located within [36] illustrates one of the primary reasons why formal equivalence checking is so important. While the original design was incapable of asserting both Q and !Q at the same time even if a SEU occurred, the implementation would allow both Q and !Q to be asserted if a SEU were to occur possibly resulting in undefined behavior of downstream modules. Utilizing formal equivalence checking does in the authors experience require the disabling of many of the synthesis tools optimization options. The author found equivalence could be achieved within a reasonable timescale through the disabling of pipe-lining, re-timing, replication, not gate push back, resource sharing and hard limiting the fan out. This therefore results in the design engineer implementing many of the optimizations within RTL as opposed to allowing the synthesis tool free reign, if high clock frequencies are to be utilized. State machines and counters must also be carefully designed to ensure that any unused states are addressed and do not lead to differing behavior between the RTL and the final net-list. It is often best to ensure all state machines are encoded using a sequential or grey code, as one hot implementation can often require transformations or constraints within the Formal Equivalence tool which reduce the confidence that all registers and all their possible values have been checked. The use of synthesis / place and route settings to implement safe state machines will lead to the addition of hardware within the final net-list which is not present within the Golden RTL and will therefore result in a Formal Equivalence Checking mismatch. Work done by the NASA Office of Logic Design in and [37] corresponds with the authors experience with regard to synthesis and place and route optimizations. The state machine implementation technique described within [38] is very interesting and deserve further investigation with regard to the ease of formal equivalence checking.

C. SEU Effects Analysis Techniques

The SEU sensitivity of SRAM-based FPGAs Systems can be analyzed using four main approaches: Accelerated Radiation Ground Testing, Fault Emulation Boards, Analytical Computation, and Fault Simulation.

Accelerated Radiation Ground Testing [39] emulates the radioactive environment in which the system is used by exposing a prototype of the FPGA-based system to a flow of radiation. The disadvantages Accelerated Radiation Tests are:

1) the impossibility of inserting SEUs only in configuration store the FPGA,
2) A possibility that the the device may be permanently damaged after the experiment and
3) high cost.

A number of **Fault emulation boards** have been developed to evaluate the impact of SEUs in the configuration memory of SRAM-based FPGA systems [40], [41]. These boards mimic the behavior of the SEUs by modifying the bitstream of the target system whose behavior is then dynamically evaluated. Error emulation can be performed before downloading the bitstream on the device under test or partially exploits it during execution dynamic reconfiguration. Unlike radiation testing experiments, error emulation allows for specific focusing on the SEUs in the FPGA configuration memory. Additionally, error emulation avoids the risk of damaging the device during analyses. The disadvantages of SEU emulation are:

1) high cost,
2) complex usability,
3) heavy reliance on chips and suppliers.

Analytical approaches, such as those presented in [42], [43], [44], they are designed to avoid the high cost of radiation test and the long experimental time of error emulation. In [44], a structure-based model of the project implemented on the FPGA is built, and the topological changes caused by the SEU in each configuration bits are derived, finding out which one SEU influence design. Finally, in [44], a probabilistic model for estimating the reliability of SRAM-based FPGA system is presented. Given the opportunity to occurrence of an SEU, the model estimates the probability of a system failure after a certain amount time. The downside to these approaches is that since the analysis is performed without taking into account the input patterns entered into the system which can lead to: provide a worst case analysis when they can't provide information on the behavior of the system in its normal operating conditions.

Lastly, there are very few **Simulators** that specifically address FPGA technology. In [45], [46], two SEU simulators involving digital circuits was proposed. Both simulators operate at the gate level of the circuit, ensuring accurate results, but neither takes into account the specific details of FPGA technology. The only simulator that targets SEUs in FPGAs is SST [8]which works at the level of personal data transfer, the representation of the system. This allows SST to emulate only the effects of SEU in logic resources, such as flip-flops and memories, and cannot reproduce the effects of SEU in configuration memory.

D. SEU Mitigation and Correction Techniques

Many SEU mitigation techniques are discussed in Literature. In [47] SEU mitigation techniques are divided into two large families: **fabricating process based and design based**.

Fabrication process based techniques aim at reducing the effects of radiation through the use of non-standard CMOS logic gates, such as the silicon-on insulator (SOI) technology from IBM [48]. In [8], IBM relies on the placement of a thin layer of Silicon on insulator during manufacturing process. All the transistors of the device are then made up on this layer of silicon, which is characterized by a reduced capacity, followed by reduced susceptibility to the effects of radiation. This technique can alleviate and reduce the long-term effects of radiation exposure (i.e. TID), but can not delete the SEU. Design-based techniques rely on hardware redundancy, i.e., the use of extra components of the FPGA device (duplication or triplication) and voting systems to detect (when duplication is used) or correct (when triplication is used) the occurrence of SEUs[55]. Design-based techniques are widely accepted because they are much cheaper than process-based techniques. Hardware redundancy uses spare components (if available) from the FPGA device and therefore its cost (apart from the increase in electricity consumption) is actually zero. An additional advantage of design-based techniques is that they can be applied to bring and address different levels of creative abstraction and different types of errors.

E. Safety Mechanisms for Random and Systematic Failures

1.Triple modular redundancy (TMR) 2.Duplication with Compare (DWC)

Triple Modular Redundancy **TMR** uses a voting mechanism to identify which module has failed and to enable error detection and recovery. However, this method has the side effect of increased gate count. A Duplication with compare scheme **DWC** may be sufficient in some cases, which can help in fault detection (but not in recovery from the fault) Fig 7.

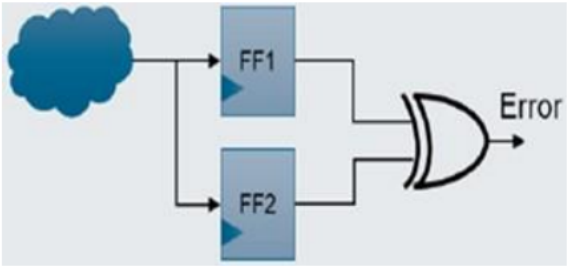

Figure 7: Duplication with Compare

A safety mechanism based on redundancy can be achieved by providing CPUs operating in Dual Core Lock Step mode (Fig 8).

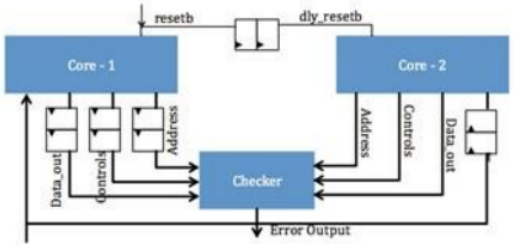

Figure 8: Dual Core Lock Step

Error Correction methods can ensure the system can continue running in the event of a failure by entering a Safe state. e.g., SEC: Single-bit error correction using ECC for memory corruption in Embedded RAMs.For example,**Application of ISO 26262 on FPGA**

o Detection of Data Integrity errors
o Errors in clock generation logic
o Errors in the timing of interface signals
o 2-bit memory error detection via ECC

These errors can be indicated by setting error bits in status registers, resulting in resetting the whole system or the host discarding the error packets.

Removal of systematic failures may include the removal of the potential failure through testing or inspection. The inspection effectiveness must match the level of severity that the hazard may impose on the consumer. o Various validation and error injection tools can be used for adequate testing, such as HDL Simulators, Fault Simulators, and Static Code Analysis tools. The process for tool selection is described in the sections below.

V. Conclusion

This document summarizes the design standards for the development of systems based on safety critical FPGAs applications. Three main points about FPGA based design systems in a safety-critical application area can be identified. The first point is that it is highly recommended to only start designing a safety-critical FPGA-based system after a well-structured and well-documented design flow is identified. The second recommendation is never quite rely on the CAD tools of the FPGA device vendor and always check the intermediates of all phases of the design process using external tools. Finally, even though the design and development process of an FPGA-based system is very similar in design and the development process of a software system, the designer must be familiar with all the technological details of the final destination device that will host the system. Tools and methods that address these issues will allow Increasing the application of SRAM-based FPGA devices in safety critical systems.

References

[1] Francisco Cardells-Tormo, Javier Valls-Coquillat, Vicenc Almenar-Terre, and Vicente Torres-Carot. Efficient fpga-based qpsk demodulation loops: Application to the dvb standard. In *International Conference on Field Programmable Logic and Applications*. Springer, 2002.

[2] Antonino Mazzeo, Luigi Romano, Giacinto Paolo Saggese, and Nicola Mazzocca. Fpga-based implementation of a serial rsa processor. In *2003 Design, Automation and Test in Europe Conference and Exhibition*. IEEE, 2003.

[3] Henrik Christopherson, Wayne Pickell, Adrian Koller, Suresh Kannan, and Eric Johnson. Small adaptive flight control systems for uavs using fpga/dsp technology. In *AIAA 3rd" Unmanned Unlimited" Technical Conference, Workshop and Exhibit*, page 6556, 2004.

[4] Radek Dobias and Hana Kubatova. Fpga based design of the railway's interlocking equipments. In *Euromicro Symposium on Digital System Design, 2004. DSD 2004.*, pages 467–473. IEEE, 2004.

[5] Jingke She and Jin Jiang. Application of fpga to shutdown system no. 1 in candu. In *6th American Nuclear Society International Topical Meeting on NPIC&HMIT, April, Knoxville, Tennessee, USA*, 2009.

[6] A Sutton. No room for error: Creating highly reliable, high-availability fpga designs, april 2012, 2014.

[7] International Electrotechnical Commission et al. Functional safety of electrical/electronic/programmable electronic safety-related systems-part 2, examples of methods for the determination of safety integrity levels. *IEC 61508-5*, 1998.

[8] Cinzia Bernardeschi, Luca Cassano, and Andrea Domenici. Sram-based fpga systems for safety-critical applications: A survey on design standards and proposed methodologies. *Journal of Computer Science and Technology*, 30(2):373–390, 2015.

[9] International Electrotechnical Commission et al. Functional safety of electrical/electronic/programmable electronic safety-related systems-part 5, examples of methods for the determination of safety integrity levels. *IEC 61508-5*, 1998.

[10] Luciano Balestra. European cooperation for space standardization (ecss). In *Trilateral Safety & Mission Assurance Conference (TRISMAC 2008)*, 2008.

[11] İlhan Mutlu, Tolga Ovatman, M Turan Söylemez, and Leyla Gören Sümer. A new test environment for plc based interlocking systems. In *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*, pages 686–690. IEEE, 2011.

[12] Dave Macdonald. *Practical industrial safety, risk assessment and shutdown systems*. Elsevier, 2003.

[13] Ian Kuon. *Measuring and navigating the gap between FPGAs and ASICs*. University of Toronto, 2007.

[14] Ian Kuon, Russell Tessier, Jonathan Rose, et al. Fpga architecture: Survey and challenges. *Foundations and Trends® in Electronic Design Automation*, 2(2):135–253, 2008.

[15] Christian Fibich, Martin Horauer, and Roman Obermaisser. Device- and temperature dependency of systematic fault injection results in artix-7 and ice40 fpgas. In *2021 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1600–1605, 2021.

[16] Jonathan Bowen and Victoria Stavridou. Safety-critical systems, formal methods and standards. *Software engineering journal*, 8(4):189–209, 1993.

[17] Robert Baumann. The impact of technology scaling on soft error rate performance and limits to the efficacy of error correction. In *Digest. International Electron Devices Meeting,*, pages 329–332. IEEE, 2002.

[18] Paolo Bernardi, L Bolzani, Maurizio Rebaudengo, Matteo Sonza Reorda, Fabian Vargas, and Massimo Violante. On-line detection of control-flow errors in socs by means of an infrastructure ip core. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, pages 50–58. IEEE, 2005.

[19] Eugene Normand. Single event upset at ground level. *IEEE transactions on Nuclear Science*, 43(6):2742–2750, 1996.

[20] B Earl Wells and Sin Ming Loo. On the use of distributed reconfigurable hardware in launch control avionics. In *20th DASC. 20th Digital Avionics Systems Conference (Cat. No. 01CH37219)*, volume 2, pages 8B1–1. IEEE, 2001.

[21] John Lach, William H Mangione-Smith, and Miodrag Potkonjak. Low overhead fault-tolerant fpga systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6(2):212–221, 1998.

[22] John McCollum, Roy Lambertson, Jeewicka Ranweera, Jennifer Moriarta, Jih-Jong Wang, Frank Hawley, and Arun Kundu. Reliability of antifuse-based field programmable gate arrays for military and aerospace applications. In *2001 MAPLD International Conference*, 2001.

[23] Raoul Velazco and Sana Rezgui. Transient bitflip injection in microprocessor embedded applications. In *Proceedings 6th IEEE International On-Line Testing Workshop (Cat. No. PR00646)*, pages 80–84. IEEE, 2000.

[24] Adrian Hilton and Jon Hall. On applying software development best practice to fpgas in safety-critical systems. In *International Workshop on Field Programmable Logic and Applications*, pages 793–796. Springer, 2000.

[25] Jerker Hammarberg and Simin Nadjm-Tehrani. Formal verification of fault tolerance in safety-critical reconfigurable modules. *International Journal on Software Tools for Technology Transfer*, 7(3):268–279, 2005.

[26] Adrian Hilton and Jon G Hall. Developing critical systems with pld components. In *Proceedings of the 10th international workshop on Formal methods for industrial critical systems*, pages 72–79, 2005.

[27] Jing-Yang Jou and Chien-Nan Jimmy Liu. Coverage analysis techniques for hdl design validation. *Proc. Asia Pacific CHip Design Languages*, pages 48–55, 1999.

[28] N Leveson. Safeware: System safety and computers. addison wesley, reading, 1995.

[29] Xiaocheng Ge, Richard F Paige, and John A McDermid. An iterative approach for development of safety-critical software and safety arguments. In *2010 Agile Conference*, pages 35–43. IEEE, 2010.

[30] Fabio Del Frate, Praerit Garg, Aditya P Mathur, and Alberto Pasquini. On the correlation between code coverage and software reliability. In *Proceedings of Sixth International Symposium on Software Reliability Engineering. ISSRE'95*, pages 124–132. IEEE, 1995.

[31] Jaynarayan H Lala and Richard E Harper. Architectural principles for safety-critical real-time applications. *Proceedings of the IEEE*, 82(1):25–40, 1994.

[32] Dominique Borrione and Philippe Georgelin. Formal verification of vhdl using vhdl-like acl2 models. In *Electronic Chips & Systems Design Languages*, pages 273–284. Springer, 2001.

[33] S Dellacherie, L Burgaud, and P Di Crescenzo. Improve-hdl-a do-254 formal property checker used for design and verification of avionics protocol controllers. In *Digital Avionics Systems Conference, 2003. DASC'03. the 22nd*, volume 1, pages 1–A. IEEE, 2003.

[34] Félix Nicoli and Laurence Pierre. Formal verification of behavioral vhdl specifications: a case study. In *Proceedings of the conference on European design automation*, pages 560–565, 1994.

[35]

[36] Robert Katz, K LaBel, JJ Wang, B Cronquist, R Koga, S Penzin, and G Swift. Radiation effects on current field programmable technologies. *IEEE Transactions on Nuclear Science*, 44(6):1945–1956, 1997.

[37] M Berg. Vhdl synthesis for high-reliability systems. In *2004 MAPLD International Conference*, 2004.

[38] John L Goodman. Application of gps navigation to space flight. In *2005 IEEE Aerospace Conference*, pages 1837–1852. IEEE, 2005.

[39] F Sturesson, S Mattsson, C Carmichael, and R Harboe-Sorensen. Heavy ion characterization of seu mitigation methods for the virtex fpga. In *RADECS 2001. 2001 6th European Conference on Radiation and Its Effects on Components and Systems (Cat. No. 01TH8605)*, pages 285–291. IEEE, 2001.

[40] Monica Alderighi, Fabio Casini, Sergio D'Angelo, Marcello Mancini, Sandro Pastore, Giacomo R Sechi, and Roland Weigand. Evaluation of single event upset mitigation schemes for sram based fpgas using the flipper fault injection platform. In *22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT 2007)*, pages 105–113. IEEE, 2007.

[41] Miguel Angel Aguirre, Jonathan Noel Tombs, Vicente Baena, F Muñoz-Chavero, A Torralba, A Fernández-León, and F Tortosa. Ft-unshades: A new system for seu injection, analysis and diagnostics over post synthesis netlist. *NASA Military and Aerospace Programmable Logic Devices (MAPLD 2005), Washington DC (USA)*, 2005.

[42] Luca Sterpone and Massimo Violante. A new analytical approach to estimate the effects of seus in tmr architectures implemented through sram-based fpgas. *IEEE Transactions on Nuclear Science*, 52(6):2217–2223, 2005.

[43] Ghazanfar Asadi and Mehdi B Tahoori. An analytical approach for soft error rate estimation of sram-based fpgas. *Proc. Military and Aerospace Applications of Programmable Logic Devices (MAPLD)*, pages 149–160, 2004.

[44] Olivier Héron, Talal Arnaout, and H-J Wunderlich. On the reliability evaluation of sram-based fpga designs. In *International Conference on Field Programmable Logic and Applications, 2005.*, pages 403–408. IEEE, 2005.

[45] Simon Schulz, Giovanni Beltrame, and David Merodio-Codinachs. Smart behavioral netlist simulation for seu protection verification. In *2008 European Conference on Radiation and Its Effects on Components and Systems*, pages 406–411. IEEE, 2008.

[46] Walter Calienes Bartra and Ricardo Reis. Set and seu simulation toolkit for labview. In *2011 12th European Conference on Radiation and Its Effects on Components and Systems*, pages 829–836. IEEE, 2011.

[47] Fernanda Lima Kastensmidt, Luigi Carro, and Ricardo Augusto da Luz Reis. *Fault-tolerance techniques for SRAM-based FPGAs*, volume 1. Springer, 2006.

[48] Teodor Calin, Michael Nicolaidis, and Raoul Velazco. Upset hardened memory design for submicron cmos technology. *IEEE Transactions on nuclear science*, 43(6):2874–2878, 1996.