Improving Fault Tolerance for FPGA SoCs through Post-Radiation Design Analysis

ANDREW ELBERT WILSON, NATHAN BAKER, ETHAN CAMPBELL, and MICHAEL WIRTHLIN, Brigham Young University, Provo, Utah, USA

FPGAs have been shown to operate reliably within harsh radiation environments by employing single-event upset (SEU) mitigation techniques, such as configuration scrubbing, triple-modular redundancy, error correction coding, and radiation aware implementation techniques. The effectiveness of these techniques, however, is limited when using complex system-level designs that employ complex I/O interfaces with single-point failures. In previous work, a complex SoC system running Linux applied several of these techniques only to obtain an improvement of 14× in mean time to failure (MTTF). A detailed post-radiation fault analysis found that the limitations in reliability were due to the DDR interface, the global clock network, and interconnect. This article applied a number of design-specific SEU mitigation techniques to address the limitations in reliability of this design. These changes include triplicating the global clock, optimizing the placement of the reduction output voters and input flip-flops, and employing a mapping technique called "striping." The application of these techniques improved MTTF of the mitigated design by a factor of 1.54× and thus provides a 22.8X× MTTF improvement over the unmitigated design. A post-radiation fault analysis using BFAT was also performed to find the remaining design vulnerabilities.

CCS Concepts: • **Computer systems organization** → **Redundancy**; *Embedded systems*; • **Hardware** → **Reconfigurable logic and FPGAs**; **Fault tolerance**; *Test-pattern generation and fault simulation*;

Additional Key Words and Phrases: FPGA, TMR, RISC-V, soft processor, radiation testing, fault injection, fault analysis, reliability

ACM Reference format:
Andrew Elbert Wilson, Nathan Baker, Ethan Campbell, and Michael Wirthlin. 2024. Improving Fault Tolerance for FPGA SoCs through Post-Radiation Design Analysis. *ACM Trans. Reconfig. Technol. Syst.* 17, 3, Article 50 (September 2024), 21 pages.
https://doi.org/10.1145/3674841

This work was supported by the I/UCRC Program of the National Science Foundation under Grant No. 1738550. The authors are associated with the BYU site of the NSF Center for Space, High-performance, and Resilient Computing (SHREC). This work was also supported by the Los Alamos Neutron Science Center (LANSCE) under proposal NS-2022-9138-A for the radiation testing experiment.
Authors' Contact Information: Andrew Elbert Wilson (Corresponding author), Brigham Young University, Provo, Utah, USA; e-mail: andrew.e.wilson@byu.edu; Nathan Baker, Brigham Young University, Provo, Utah, USA; e-mail: nathangarybaker@byu.edu; Ethan Campbell, Brigham Young University, Provo, Utah, USA; e-mail: ecampbell@byu.edu; Michael Wirthlin, Brigham Young University, Provo, Utah, USA; e-mail: wirthlin@byu.edu.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 1936-7414/2024/9-ART50
https://doi.org/10.1145/3674841

1 Introduction

The high density and reconfigurability of **static RAM (SRAM)**-based **field programmable gate array (FPGA)** devices make them an attractive option for providing high-performance computing, high-bandwidth I/O processing, and in-system reconfigurability for satellite systems [13, 31]. The large amount of reconfigurable resources found in FPGAs enables the use of complex **system-on-chip (SoC)** computing systems that include programmable processors, complex bus interfaces, high-bandwidth bus interfaces and a variety of system components, such as direct memory access engines, interrupt controllers, and so on. The European Space Agency is exploring the use of FPGA-implemented soft processors to replace application-specific integrated circuit processors [3]. National Aeronautics and Space Administration has developed an FPGA-based processing system with **double data rate (DDR)** and Flash memory that implements internal configurable processing systems to perform high-performance application processing [6].

Although SRAM-based FPGAs provide many benefits for space computing systems, they are sensitive to the ionizing radiation found in space environments [11, 23]. Ionizing radiation can cause **single-event upsets (SEUs)** within the **configuration RAM (CRAM)**, internal block memory, **flip-flops (FFs)**, and other internal FPGA state. Upsets within the CRAM are especially troublesome as they may *change* the way the FPGA design is configured to operate. Such upsets may change the logic functions, routing, clocking, or other design functions configured onto the FPGA. With such unwanted changes, the configured FPGA design may fail within the spacecraft system.

A variety of techniques have been shown to mitigate the effects of SEUs within FPGA systems. Configuration scrubbing is the process of continually writing the CRAM of the FPGA to repair any upsets that may occur due to ionizing radiation [21]. **Triple modular redundancy (TMR)** is also used to tolerate failures that occur due to CRAM upsets by performing a majority voting of internal circuit signals between three circuit copies. A variety of other SEU mitigation techniques have demonstrated improvements in reliability including the use of error-correction codes [18], software consistency checks [22], and a variety of other techniques [19]. The use of these techniques has demonstrated significant improvements in reliability for relatively simple circuits [8]. For the simple state machine circuit named 'b13' from the ITC'99 benchmark suite [10], the use of these techniques resulted in a 367× improvement in the **mean-time to failure (MTTF)** of the circuit under high energy neutron testing.

While these techniques are effective for relatively simple, internal FPGA circuits, the benefits of these techniques are limited when applied to more complex, system-level circuits. A fault tolerant **reduced instruction set computer five (RISC-V)** processor-based system running Linux was created with these SEU mitigation techniques that includes a VexRiscv open source RISC-V processor, DDR controller, Ethernet controller, UART, and a variety of other system components. When subjected to a neutron radiation test, this TMR SoC system demonstrated only a 14× improvement in MTTF. A post-radiation test analysis of this design was performed to identify the reasons for the limited improvements in reliability [27]. The primary reasons for the limited improvement were due to the single-point failures of the I/O pins, the DDR controller interface, and the global clock.

In this article, the results from the detailed post-radiation failure analysis drove several improvements that were implemented within the Linux SoC system. These changes include triplicating the global clock, optimizing the placement of the reduction output voters and input FFs, and employing a mapping technique called "striping." A fault injection campaign and a follow-up neutron radiation test were performed on this improved design to measure the improvements in reliability. The application of these techniques improved the TMR CRAM sensitivity of the design described in [27] by a factor of 2× (see Table 6). The results of the neutron radiation test showed an improved MTTF of 22.8× over the unmitigated design (see Table 7). A post-radiation fault analysis using

bitstream fault analysis tool (BFAT) was also performed to find the remaining design vulnerabilities. This work demonstrates how this detailed fault analysis provides insight into identifying and categorizing failure modes to effectively apply low-cost high-performance mitigation.

This article will begin by summarizing the baseline RISC-V SoC system, the fault tolerant techniques used to improve its reliability, and the results from the original radiation test in Section 2. Section 3 will describe the BFAT tool used to analyze the design failures from this original radiation test and Section 4 will describe the changes that were made to the design based on the BFAT analysis tool. The follow-up radiation test conducted to measure the improvements in reliability is described in Sections 5 and 6 and uses the same BFAT analysis to identify yet more design vulnerabilities.

2 Baseline RISC-V Linux Fault Tolerant System

The baseline SoC system used in this work is based on a RISC-V open source processor and the LiteX SoC framework. This section will describe baseline RISC-V Linux system used in this work. The SEU mitigation approach used to improve the reliability of the system will also be presented as well as the results from the previous radiation experiments.

2.1 VexRiscv Linux System

The open-source LiteX framework for building SoCs for multiple FPGA families [15] was used to create the SoC systems for this work. This project enables users to construct complex embedded systems around a variety of open source processor cores. We chose to build the system around the VexRiscv processor [26], which is a 32-bit implementation of the RISC-V instruction set architecture that supports extensions, such as multiply, divide, and 32-bit floating point. The LiteX framework also provides a large set of useful system IP cores, such as memory controllers, **universal asynchronous receiver-transmitter (UART)**, serial peripheral interface, peripheral component interconnect, and so on for composing complex on-chip systems.

A related project called "Linux on LiteX-VexRiscv" [17] provides a Linux implementation on the LiteXVexRiscv SoC. This project is based on the popular Buildroot tool [4, 24] for simplifying the process of supporting Linux on a variety of embedded processor environments. The Linux environment for this project was configured with a minimal file system and unix tools. The operating system was configured to initiate the "Dhrystone" benchmark at startup and to run this benchmark continuously. This benchmark displays the output over the serial UART, and this output is continuously monitored by the external host to identify processor or system failures.

The LiteX, VexRiscv Linux system was targeted for the Digilent Nexys Video development board [2]. This board is based on the Xilinx 7-Series 28nm XC7A200T FPGA and provides a UART, Ethernet, video physical interface, and **Secure Digital (SD)** card. An overview of the implemented SoC architecture is shown in Figure 1.

2.2 Mitigated VexRiscv System

To generate a radiation-hardened version of the SoC, TMR was applied to the unmitigated design. TMR is a mitigation technique which involves triplicating all resources of a design and placing majority voters at the outputs of the triplicated segments (see Figure 2). In a system with TMR implemented, a corrupted output due to a failure in one triplicated module will be masked by the correct outputs of the other two functional modules [14]. This additional redundancy comes at the cost of greater power consumption, higher resource utilization, and a slower maximum frequency.

The additional hardware cost of TMR designs will increase the area affected by radiation-induced upsets, and over time multiple upsets will overcome the error masking of TMR if repair is not employed [20]. To address this issue, TMR systems must be paired with a repair mechanism, such as CRAM scrubbing to prevent error accumulation. Scrubbing involves continuously reading the entire CRAM of the FPGA, comparing the readback to a golden file, and repairing any discrepancies that are found between the two. The TMR system in this work is paired with an external hardware

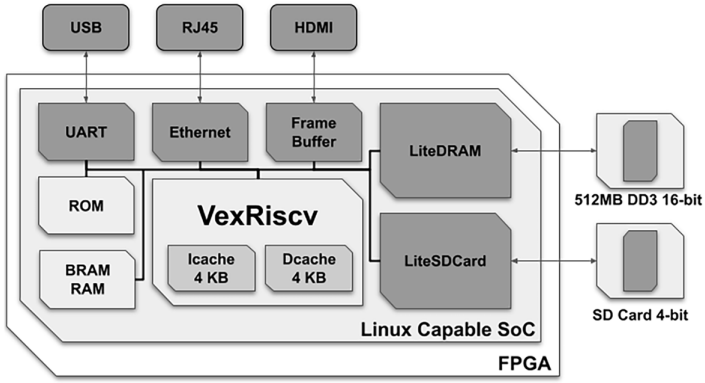

Fig. 1. VexRiscv system architecture.

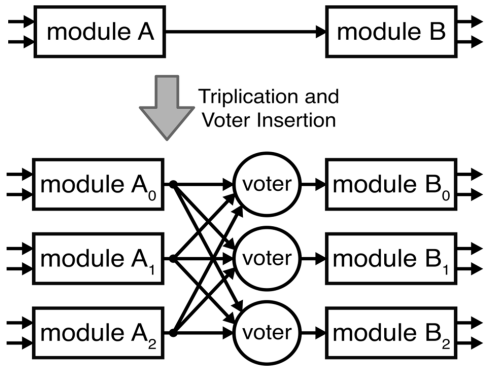

Fig. 2. TMR and majority voting.

Table 1. VexRiscv SoC Design Utilization for Non-TMR and TMR Designs

Design	LUT	LUTRAM	FF	BRAM	DSP	FMAX
Unmitigated	9,131 (6.8%)	521 (1.1%)	7,641 (2.8%)	36 (9.9%)	4 (0.5%)	111.6 MHz
TMR	37,846 (28.1%)	1,611 (3.5%)	22,905 (8.5%)	108 (29.6%)	12 (1.6%)	91.4 MHz
Cost Ratio	4.14×	3.09×	3.00×	3.00×	3.00×	0.81 ×

scrubber that reads and writes the FPGA CRAM through the device's **Joint Test Action Group (JTAG)** interface. More detail will be given on the configuration scrubber in Section 2.3.

The SpyDrNet TMR tool was used to apply fine-grain TMR to the post-synthesis electronic design interchange format netlist of the VexRiscv SoC [25]. This tool triplicated all FFs, **lookup tables (LUTs)**, **block RAMs (BRAMs)**, and **digital signal processings (DSPs)** and placed voters between small groups of these primitives. The resulting triplicated netlist was then imported back into Vivado, where a placed and routed design could be generated. The resource utilization of the triplicated design is increased by a factor of about 3.6× due to the additional LUTs that are used to implement majority voting. The maximum possible clock frequency is also reduced due to the lengthened critical path created by the inclusion of voters. A full comparison of the utilization between the unmitigated and the mitigated designs is in Table 1.

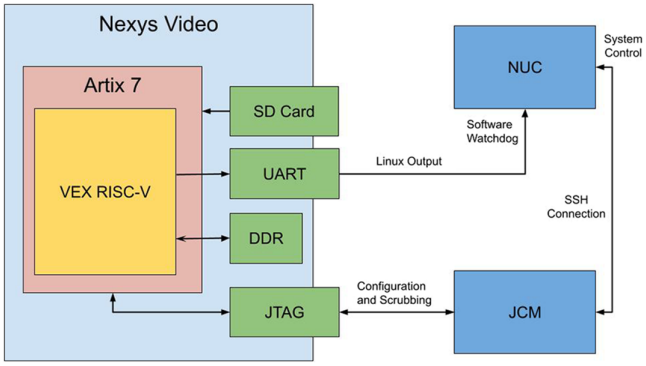

Fig. 3. VexRiscv Linux system test diagram.

2.3 High Energy Neutron Radiation Test

In order to measure the reliability of the non-TMR and TMR systems, both designs were subjected to radiation testing at the **Los Alamos Neutron Science Center (LANSCE)** in September of 2021 [27]. While under the facility's high energy neutron beam, single-event effects can be created in a device at an accelerated rate [16]. Because the energy spectrum of neutrons produced by the beam matches the environment of the Earth, the reliability of a system can be estimated based on the failure rate while under the beam. The radiation test setup involved three primary components as shown in Figure 3:

- —*Nexys Video board*: a FPGA development board that is configured with the Linux system and serves as the **device-under-test (DUT)**,
- —*JTAG Configuration Manager (JCM)*: a JTAG controller that handles configuration of the FPGA and correction of CRAM faults [12], and
- —*Intel **Next Unit of Computing (NUC)** Computer*: a remote host computer thatt monitors and logs the status of the DUT and issues commands to the JCM.

The JCM is used to configure the initial configuration of the Artix 7 FPGA and to perform continuous configuration scrubbing, performing a full scrub every 500 ms. The JCM is an embedded Linux system with a direct connection to the FPGA's JTAG port to perform all configuration functions on the Artix FPGA. It is connected to the JTAG port on the Nexys Video board in order to access the CRAM of the device. Each of the CRAM locations that are repaired by the JCM are logged and timestamped. The NUC monitors the Linux system by reading the Dhrystone results which are published over UART. If a failure is detected, the FPGA is fully reconfigured with the bitstream and the error is logged by the NUC.

2.4 Test Results

The results of the LANSCE radiation test experiment are summarized in the bottom two rows of Table 2. There were 76 failures observed on the unmitigated design and 27 failures observed in the TMR mitigated design. The amount of neutron fluence required to cause a system failure (on average) serves as a metric for the reliability of each system. The mean "fluence to failure" is 4.79×10^8 n/cm^2 for the non-TMR system and 7.07×10^9 n/cm^2 for the TMR system. Assuming a terrestrial neutron flux of 14 n/cm^2 per hour [1], the non-TMR system will have a MTTF of 3,903 years and the TMR system will have a MTTF of 57,642 years in a terrestrial environment. The improvement in MTTF of the mitigated system over the unmitigated system is 14.7×. Taking into account the slower clock rate of the TMR design, the improvement in **"mean work between failure" (MWBF)** is 12.1×.

Table 2. Neutron Radiation Test Data

Design	Fluence (n/cm^2)	Observed CRAM Upsets	Failures	Cross Section (cm^2)	+95% Confidence -95% Confidence	Reduction	MWBF Improvement
VexRiscv Unmitigated	7.08×10^{9}[a]	3,473[a]	422[a]	5.09×10^{-10}	5.59×10^{-10} 4.60×10^{-10}	1×	1×
VexRiscv TMR	5.38×10^{10}[a]	21,783[a]	30[a]	1.99×10^{-11}	2.84×10^{-11} 1.34×10^{-11}	25.57×	23.80×
Linux-VexRiscv Unmitigated	3.64×10^{10}	11,908	76	2.09×10^{-9}	2.57×10^{-9} 1.61×10^{-9}	1×	1×
Linux-VexRiscv TMR	1.91×10^{11}	59,014	27	1.42×10^{-10}	3.47×10^{-10} 4.82×10^{-11}	14.76×	12.07×

[a]Results calculated for individual processors from multiple processor implementations on Xilinx Kintex UltraScale+[30].

A previous work performed a radiation test on bare metal VexRiscv systems [28] (summarized in the top two rows of Table 2). The arrangement contained the same VexRiscv processor as the system described in this work, but without the Linux operating system, DDR, and other peripherals. Like the experiment described in this work, both non-TMR and TMR implementations of the system were tested. Despite being based on the same processor as the one used in the Linux SoC, the bare metal system with TMR achieved a MTTF improvement of 25.6× over the non-TMR system. This is a much better improvement than the 14.7× improvement seen in the mitigated Linux SoC. These differing results indicate that there are vulnerabilities in the Linux system that are not present in the bare metal system. Locating these vulnerabilities was the primary goal for the fault analysis described in the next section.

3 Post-Radiation Fault Analysis

Although the single-event mitigation methods employed did improve the reliability of the system, the overall reliability improvement over the non-TMR system was disappointing. In order to understand why the sensitivity of the design did not see the same improvement as that of the simplified embedded processors [30] we carefully analyzed the failures which were observed during the radiation test. By employing several fault injection approaches, we were able to narrow down the CRAM upsets which caused each failure. We then used a static fault analysis tool to expose the vulnerabilities which were exploited by each upset. The data collection and analysis processes which we utilized are described in this section as well as the discovered design vulnerabilities.

3.1 Fault Injection

The first step in the fault analysis process was to identify which CRAM locations that were observed during the radiation test caused design failure. We can identify the FPGA and design resources that were affected by upsets in these sensitive CRAM cells in order to pinpoint the most vulnerable locations in the design. CRAM fault injection was used to identify the sensitive bits observed at the radiation test. Fault injection involves the deliberate modification of the CRAM cells to emulate SEUs in the radiation test. A series of fault injection experiments were performed using the JCM based on upset data collected from the radiation test in order to reproduce failures that were observed during the test and narrow down the sensitive bit(s) which caused each failure.

We performed several SEU "playbacks" of all CRAM faults which were observed during the test for the TMR Linux system. This campaign aimed to reproduce every failure that occurred during the radiation test. Playback fault injection involves manually injecting every fault that occurred during

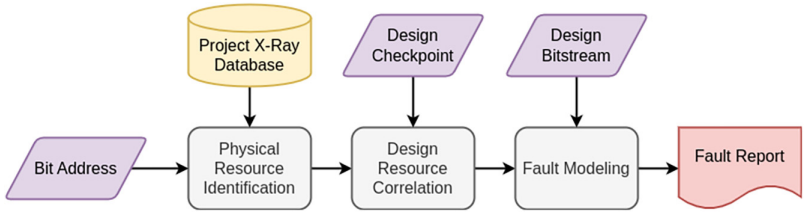

Fig. 4. BFAT flow.

the radiation test in the order that they were found by the JCM scrubber. If a failure is observed, the bits that occurred during the cycle are tagged, the system is reset, and the fault injection continues with the next cycle in the list. We were able to correlate a sensitive scrub cycle to 15 of the 27 failures from the radiation test using this method. 13 failures required one bit upset to reproduce, while 2 failures required two bit upsets in order to be reproduced. In total, 17 bits were identified during these fault injection experiments.

3.2 BFAT

After gathering all bits which caused a failure in the radiation test, we analyzed the effects of these upsets on the FPGA design using the BFAT [7]. This tool correlates CRAM addresses to specific FPGA resources and their corresponding design elements of the configured FPGA design. By knowing which design elements caused a system failure, we can get a better idea of where vulnerabilities exist in the design.

The overall flow of the BFAT tool is summarized in Figure 4. The BFAT tool requires a list of bitstream addresses to analyze, the design bitstream, and a post-implementation design checkpoint from the Vivado tool. For each CRAM address, BFAT identifies the physical FPGA resource(s) that depend on the bit.

Next, BFAT will query the design checkpoint in order to associate the affected physical resource with an element of the design configured on the FPGA. Finally, the tool will simulate the upset with a model to test if the upset corrupts the intended operation of the affected design resources. For example, upsets in routing bits can activate or deactivate routing connections, disconnecting existing signals or shorting two existing signals together.

After the analysis, a human-readable fault report is generated which includes all relevant information that was gathered for each bit during the analysis. The report includes the physical resource, relevant cells and nets, and cause of failure (if one was found) for all CRAM addresses provided during the analysis. Designers can then review this data in order to identify common points of failure that constitute vulnerabilities in the design.

3.3 Fault Analysis of Linux SoC

All 17 sensitive bits identified from the radiation test for the TMR design were applied to the BFAT tool to find design failures. These 17 bits are a relatively small sample size, and more sensitive bits are needed to provide a more accurate statistical overview of design failure. In order to increase our statistical confidence we conducted a large-scale random fault injection campaign to increase the size of our analysis set. Over the course of 34 days of continuous fault injection, we identified 106 more sensitive CRAM bits that were also input to BFAT. The FPGA device resources that caused the design to fail for both sets of CRAM bits (those gathered in radiation testing and random fault injection) are summarized in Table 3. The estimated sensitive bits for the whole design are reported in the right-most column.

Table 3. FPGA Device Resources Causing Design Failure

Resource	Occurrences (radiation test)	Occurrences (random fault injection)	Occurrences (total)	Estimated Bits
Interconnect	13 (76.4%)	75 (70.8%)	88 (71.5%)	17,855
I/O Pin	3 (17.6%)	19 (17.9%)	22 (17.9%)	4,464
Clocking	1 (5.9%)	6 (5.7%)	7 (5.7%)	1,420
CLB	0 (0.0%)	4 (3.8%)	4 (3.3%)	812
Undefined	0 (0.0%)	2 (1.9%)	2 (1.6%)	406
Total	17	106	123	24,956 (0.042%)*

*Percentage is in comparison to the 59,145,600 type 0 CRAM locations in the XC7A200T.

Table 4. Design Resources Causing System Failure

Design Function	Occurrences (radiation test)	Occurrences (random fault injection)	Occurrences (total)
DDR Interface	10 (66.7%)	81 (77.9%)	91 (76.5%)
Global Clock	3 (20.0%)	19 (18.3%)	22 (18.5%)
Constant Logic	0 (0.0%)	2 (1.9%)	2 (1.7%)
Reset Circuit	0 (0.0%)	1 (1.0%)	1 (0.8%)
Misc. Circuitry	2 (13.3%)	1 (1.0%)	3 (2.5%)
Total	15	104	119

There were four primary failure modes which were identified by the BFAT analysis of the sensitive CRAM bits: interconnect, I/O pins, clocking, and **configurable logic blocks (CLBs)**. Previous works have demonstrated that bits that configure the interconnect (routing) resources are the most common points for an SEU to occur [5, 11]. It was unsurprising that over 75% of design failures were caused by failures within the interconnects. Of the 88 interconnect failures, 48 were caused by an "open" in a design net and 40 were caused by a "short" between two nets. The next most common fault type involved upsets within either the I/O pins or the Serial/Deserial (SerDes) blocks used within the I/O. Though less common, there were some upsets that affected the clocking resources (clock H tree, phase lock loop, and so on) and CLB tiles. There were two bits for which a device resource could not be identified.

3.4 Design Vulnerabilities

After identifying what types of FPGA resources are associated with these sensitive bits we analyzed the actual design resources to understand which portions of the design are failing. By reviewing the affected design nets and cells for each of the 119 failures for which an affected device resource was identified, we identified four components of the design which were most vulnerable. The results are summarized in Table 4.

The DDR interface stands out as the weakest portion of the design. This is unsurprising, as the DDR interface claims 48 (52%) of the 93 utilized I/O pins for the design, which are not triplicated. As such, faults that affect I/O resources which are used by the design are not protected by TMR. What was not expected was that 71 of the 91 faults that are related to the DDR interface were related to DDR signals in the FPGA interconnect, rather than the I/O resources themselves. Only 17 of the faults were related to the I/O resources, and 3 additional faults occurred within CLB tiles. Clearly, the interconnect associated with the DDR interface was responsible for the bulk of these failures.

Fig. 5. Vulnerable segments of signal trunk originating from an untriplicated I/O.

After examining the signals related to these faults, we discovered that the failures occur in untriplicated regions of the DDR nets for both inputs and outputs. For example, a net which is sourced by an untriplicated I/O resource may travel some distance within the FPGA resources before splitting into three TMR domains. From the three branches and onward, the signal is protected by TMR. However, if the trunk is corrupted all three branches will have incorrect output, causing the majority voting mechanism to fail. This phenomenon is displayed in Figure 5. A similar situation exists with output signals relating to the DDR interface and the reduction voters that are used to converge signals from the three domains into a single output I/O pin.

The next most vulnerable location was the global clock network. After examining the BFAT results for these failures, the global clock network was entirely untriplicated in the implemented design. This was perplexing, as SpyDrNet had successfully generated a triplicated clock network in the netlist for the TMR design. Further investigation revealed that Vivado's implementation tools had unknowingly optimized away the two additional global clock buffers, resulting in only one clock signal driving all three TMR domains. As the clock is one of the largest signals in the design, it is unsurprising that it would contribute to a large portion of design failures if it were not triplicated.

There are also a small number of failures related to global constant signals which represent voltage common collector and ground. These signals have many possible sources throughout the device to provide local access to the constant values 1 and 0 when needed. Due to their special nature compared to other nets in the device, SpyDrNet is unable to triplicate these signals. Constant signals use the same routing resources as other nets, leaving them vulnerable to the same "short" and "opens" that are possible with routing bit upsets. In the two failures from the test, we observed that local regions of global logic signals were shared between cells of differing domains. Because these segments of global logic were faulted, both domains were corrupted, leading to a system failure.

Finally, there were three failures that occurred in miscellaneous regions of the design. Two of these failures were caused by multi-bit upsets. Each individual bit upset affected one TMR domain, so these upsets do not cause failures independently. However, when both upsets in the group occur simultaneously, two domains are corrupted at the same time, which is not protected by the TMR schema. An additional failure in miscellaneous logic occurred due to a single bit upset. The upset affected the clock invert flag in a CLB slice, inverting the clock signal for all cells in the slice. This particular slice happened to contain cells from multiple TMR domains. Since all of these cells experienced errors simultaneously, the system failed. A failure in multiple TMR domains due to an SEU is referred to as common-mode failure.

4 SoC Design Improvements

With this detailed analysis of the single-point failures for the TMR design, the opportunity arises to apply targeted mitigation techniques to the design with minimal overhead and effective coverage.

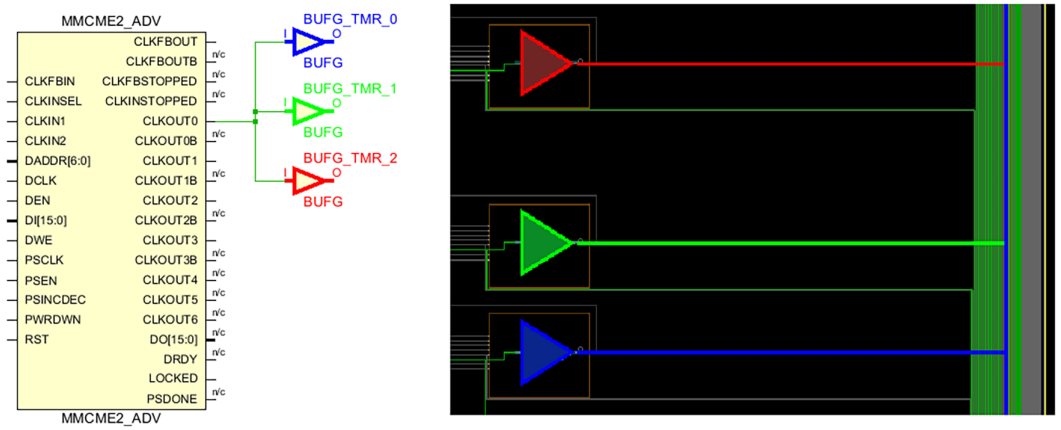

Fig. 6. (Left) schematic diagram of MMCME2 clock generator and three BUFG primitives, (right) floorplan with colored TMR clock network.

As indicated by the data in Table 4, specific attention should be directed toward addressing vulnerabilities within the DDR IO and Clock components since these particular elements contribute to over 90% of all single-point failures encountered.

The simplest way to address most of these issues is to provide placement constraints to the FPGA vendor tools. The tools allow for the different netlist primitives within the design to be placed at specific locations. Regular expressions and large placement regions can be used to simplify this process. It is worth noting that while these methods will not alter the synthesized design, they might pose challenges in meeting timing constraints. These mitigation methods will offer increased reliability without incurring additional design overhead. This section will summarize the additional design mitigation techniques that were employed with this Linux SoC design. The techniques used for this design include clock triplication, custom mapping for the reduction output voters and input FFs, and striping.

4.1 Clock Triplication

Part of the placement process within the vendor tools is the optimization of BUFGs (global clock buffers). In scenarios where a TMR design incorporates triplicated BUFGs, the tools will automatically consolidate them into a solitary primitive. The user can enforce that the tools "don't touch" these primitives during the optimization stages, as shown in Listing 1.

```
set_property DONT_TOUCH true [get_cells BUFG_TMR_0]
set_property DONT_TOUCH true [get_cells BUFG_TMR_1]
set_property DONT_TOUCH true [get_cells BUFG_TMR_2]
```

Listing 1. Vendor Contraints for TMR Clock BUFGs

Utilizing three BUFGs, as illustrated in Figure 6, effectively eliminates the vulnerability of single-point failures within a single-clock network. However, the FPGA's ability to isolate these three unique clock networks remains restricted. These networks continue to rely on shared routing resources, thereby allowing a failure of a single routing MUX to potentially affect multiple clock domains. With regard to inter-clock skew, there is an expectation that the global H clock tree, integrated into the FPGA, will rectify any inconsistencies without introducing timing-related complexities linked to the utilization of three distinct clocks.

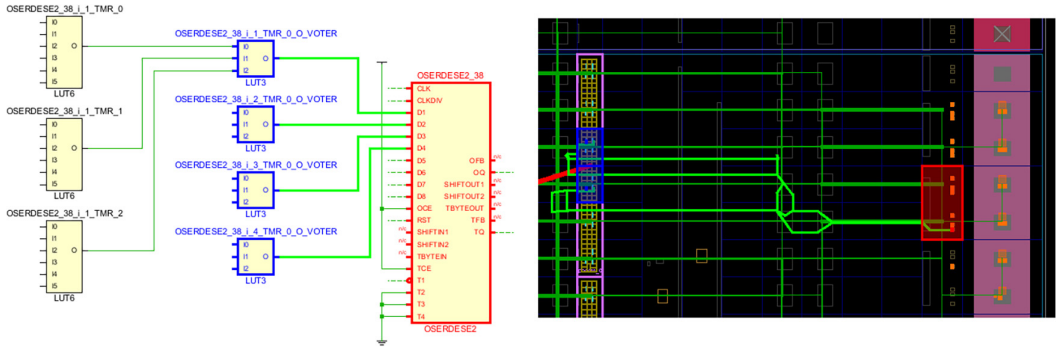

Fig. 7. (Left) OSERDES schematic with reduction LUT voters, (right) floorplan of same elements.

4.2 Reduction Output Voters

The DDR interface encompasses **input serializer (ISERDES)** and **output deserializer (OSERDES)** primitives, facilitating high-speed SerDes operations with the DDR3 module. In the TMR domain, the utilization of reduction LUT voters is imperative when transitioning from TMR logic to the OSERDES outputs. Although the occurrence of single-point failures within the LUT is inevitable, there exists an opportunity to minimize the routed net between the LUT and the OSERDES primitives. The vendor tools will position this LUT to optimize timing constraints, which might inadvertently lead to a lengthy net traversing numerous routing MUXs.

By manually identifying the reduction LUT voters associated with each OSERDES cell, they can be placed as close as possible to the output. Figure 7 shows the OSERDES cell in red, the LUTs in blue, and the routed nets in green. By using the placement constraint in Listing 2, the number of routing MUXs are greatly reduced, though not completely removed. The individual placement constraints include the LUTs connected to small neighboring groups of OSERDES tiles and placed in nearby groups of CLB tiles. This style of constraints simplified the constraints and offered greater flexibility to the vendor tools during the placement phase.

```
create_pblock poserd_0
add_cells_to_pblock [get_pblocks poserd_0] [get_cells -quiet [list \
    OSERDESE2_38_i_1_TMR_0_O_VOTER OSERDESE2_38_i_2_TMR_0_O_VOTER  \
    OSERDESE2_38_i_3_TMR_0_O_VOTER OSERDESE2_38_i_4_TMR_0_O_VOTER]]
resize_pblock [get_pblocks poserd_0] -add {SLICE_X162Y194:SLICE_X163Y199}
```

Listing 2. Vendor placement contraints for OSERDES TMR LUT Voters

To determine how successful these placement constraints were, we can measure the lengths of the output nets of the reduction voter cells. The lengths are measured in the number of **programmable interconnect points (PIPs)** that are used by a given net. A PIP represents a possible connection that can be made in a routing MUX between two physical nodes in the FPGA routing network. These connections are configured by CRAM bits, so a net which utilizes more connections will depend on more sensitive CRAM bits. Thus, the sensitivity of a net will be roughly proportional to the number of PIPs that are utilized by the net. In the naive TMR design, the output voter nets utilized 10.4 PIPs on average. In the improved TMR design with placement constraints, these nets only used 4.1 PIPs on average. So, we can expect that the sensitivity of output signals associated with the DDR interface will be reduced by about 2.5×.

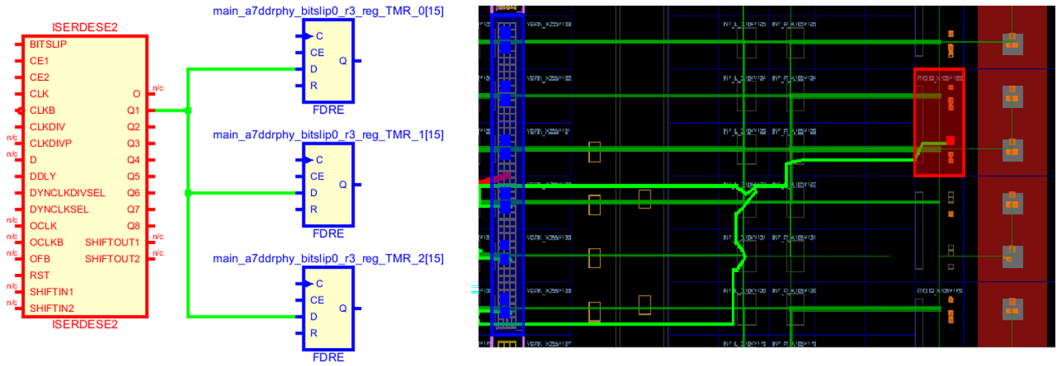

Fig. 8. (Left) ISERDES schematic with triplicated FFs, (right) floorplan of same elements.

4.3 Triplicated FFs Inputs

Similar to the DDR outputs, the ISERDES primitives connect to TMR FFs when transitioning to the TMR domain. The routed net is connected to three different FFs as shown in Figure 8. Similar to the reduction LUT voters, the vendor placement tools are unaware of the sensitive nature of this routed net and may introduce unnecessary singe-point failures. Placement constraints are used again to reduce the length of the routed nets from the ISERDES primitives to the triplicated FFs.

Once the triplicated FFs are identified and associated with each ISERDES cell, they can be placed as close as possible to the input. Figure 8 shows the ISERDES cell in red, the FFs in blue, and the routed nets in green. The individual placement constraints include the FFs connected to small neighboring groups of ISERDES tiles and placed in nearby groups of CLB tiles as shown in Listing 3. Similar to the output constraints, this guides the vendor placement tools to reduce the routing MUXs used for these sensitive nets.

```
create_pblock piserd_0
add_cells_to_pblock [get_pblocks piserd_0] [get_cells -quiet [list \
    main_a7ddrphy_bitslip0_r3_reg_TMR_0[15] \
    main_a7ddrphy_bitslip0_r3_reg_TMR_2[15] \
    main_a7ddrphy_bitslip0_r3_reg_TMR_2[15]]]
resize_pblock [get_pblocks piserd_0] -add {SLICE_X162Y180:SLICE_X163Y185}
```

Listing 3. Vendor placement contraints for ISERDES TMR FFs

We can predict the sensitivity improvement attained by these placement constraints by comparing the number of PIPs used by the input DDR nets. In the naive TMR design, the nets utilized 24.6 PIPs on average. In the improved TMR design with placement constraints, these nets only used 7.8 PIPs on average. Based on the reduction in average net length, we can expect that the sensitivity of input signals associated with the DDR interface will be improved by about 3.2×.

4.4 Striping

TMR clock networks share routing resources and thus introduce single-point failures that can affect multiple clock domains. Other works have shown that general placement constraints can reduce the number of routing related single point failures [8, 9]. These methods attempt to isolate the placement and routing of the different TMR domains.

The method chosen in this situation is called striping. It simply divides the FPGA configuration columns and assigns them to different TMR domains. If a consistent naming convention is used

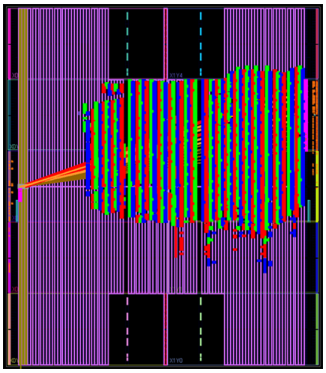

Fig. 9. Striped TMR design with each TMR domain represented as blue, green, and red.

when generating the TMR netlist, a regular expression can be used within the vendor tools to identify each domain and assign it to an area covering separated segments. Listing 4 is an example of creating a segmented partition for one of the triplicated domains ending with the suffix of "TMR_0." The design is split into three different segmented partitions for each domain as shown in Figure 9. Though previous work has shown this to not be as effective with complex designs like embedded soft processors [29], it still provides some benefits for this design as shown in fault injection.

```
create_pblock pblock_dut_tmr_0
resize_pblock [get_pblocks pblock_dut_tmr_0] -add {
    SLICE_X0Y0:SLICE_X1Y249
    SLICE_X6Y0:SLICE_X7Y249
    ...
    SLICE_X96Y0:SLICE_X97Y249
    SLICE_X102Y0:SLICE_X103Y249
}
set_property IS_SOFT 0 [get_pblocks pblock_dut_tmr_0]
add_cells_to_pblock [get_pblocks pblock_dut_tmr_0] \
    [get_cells -hierarchical -regexp .*TMR(_VOTER)?_0.* -filter IS_PRIMITIVE==1]
```

Listing 4. Vendor placement contraints for TMR Striping

5 Evaluation of Improved SoC Design

Each of these mitigation techniques will result in a reduction of single-point failures that were pinpointed through the analysis of faults from both fault injection and neutron radiation data. By utilizing the limited dataset of 106 failures from the estimated 25,000 sensitive CRAM bits within the design, an estimation of improvement can be inferred. If all single-point failures linked to routing are eliminated, as illustrated in Table 5, the revised design could yield a potential fivefold improvement in MWBF.

Furthermore, additional fault injection on the improved design is used to emulate radiation and measure sensitive CRAM bits. Faults were injected into different variations of the design to see how the different mitigation methods affected the results. Table 6 shows the CRAM sensitivity and MTTF improvement for each design iteration of the soft Linux SoC that was tested. Each mitigation method described in Section 4 was added incrementally within the tested designs, with the last

Table 5. Estimated Reduction in System Failures with Design Improvements

Device Resource	Observed Failures	Estimated Failures
Interconnect	75	8
I/O Pin	19	19
Clocking	6	1
CLB	4	3
Undefined	2	2
Total	106	23

Table 6. Fault Injection of Mitigated Designs

Design	Injections	Failures	Sensitivity	MTTF Improvement
Unmitigated	24,218	139	0.574%	1.0×
TMR with One Clock	603,366	235	0.039%	14.74×
TMR Clocks	121,162	37	0.031%	18.79×
TMR Clocks + Striping	155,635	45	0.029%	19.85×
TMR Clocks + DDR IO	103,884	25	0.024%	23.85×
TMR and All Mitigations	270,209	50	0.019%	31.02×

row using all the methods. The outcome of fault injection yielded a twofold improvement from the preceding TMR design, which contrasts with the initially projected fivefold enhancement.

Using these mitigation methods identified by the fault analysis, the reliability of the design measured by MWBF can be drastically improved without any overhead to functionality or performance. The newly mitigated design was able to achieve the same max frequency as the original design at 91.4 MHz as shown in Table 1.

The original unmitigated and improved TMR versions of the VexRiscv system were tested with high energy radiation at the LANSCE [16] in December 2022. This facility provides a high energy neutron beam that provides a wide spectrum of neutron energies (up to 800 MeV) that closely matches the energy spectrum of neutrons found on earth (called terrestrial neutrons). The Nexys Video boards were placed in the neutron radiation beam path with the Artix 7 FPGA centered with the beam as shown in Figure 10. The FPGA board is placed at a normal angle of incidence and operates at room temperature. A collimator with a diameter of 2 inches was placed within the beam to limit exposure of the high-energy radiation to other board components.

The reliability of the system is measured in terms of "fluence to failure" or total amount of neutron fluence (neutrons/cm^2) divided by the number of system failures. A related metric is the sensitive cross section, σ, which is the inverse of fluence to failure (in units cm^2). The beam testing time was divided between the non-mitigated system and the TMR mitigated system across two FPGA development boards. One-tenth of the time was scheduled for testing the non-TMR system and nine-tenths of the time was scheduled for the improved TMR design. More time is needed for the mitigated TMR design to gather sufficient statistical results since much more neutron fluence will be needed for each system failure.

The results from the new experiment are summarized in the bottom two rows of Table 7. The baseline of the unmitigated design is reevaluated during this experiment to account for any changes in testing. Considering the data from the new experiment, 95 system failures were observed on the

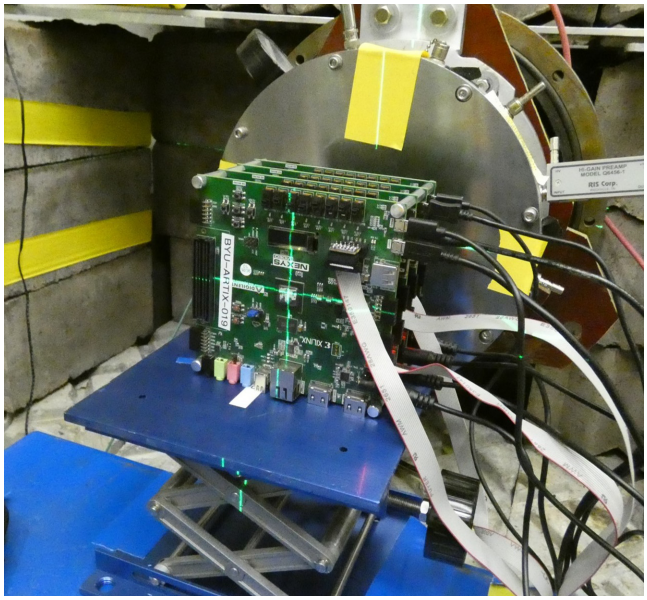

Fig. 10. Testing at LANSCE neutron radiation source.

Table 7. Radiation Test Data of Improved SoC Design

Design	Fluence (n/cm^2)	Observed CRAM Upsets	Failures	Cross Section (cm^2)	+95% Confidence -95% Confidence	Reduction	MWBF Improvement
Linux-VexRiscv Unmitigated	3.64×10^{10}	11,908	76	2.09×10^{-9}	2.57×10^{-9} 1.61×10^{-9}	1×	1×
Linux-VexRiscv TMR	1.91×10^{11}	59,014	27	1.42×10^{-10}	3.47×10^{-10} 4.82×10^{-11}	14.76×	12.07×
Linux-VexRiscv Unmitigated	3.67×10^{10}	16,239	95	2.59×10^{-9}	3.12×10^{-9} 2.06×10^{-9}	1×	1×
Linux-VexRiscv TMR (Improved)	3.26×10^{11}	144,841	37	1.14×10^{-10}	2.70×10^{-10} 3.38×10^{-11}	22.78×	18.66×

non-TMR system and 37 failures were observed in the improved TMR system. The mean fluence to failure is 3.87×10^8 n/cm^2 for the non-TMR system and 8.80×10^9 n/cm^2 for the improved TMR system. In a terrestrial environment, the non-TMR system will have a MTTF of 3,150 years, and the improved TMR system MTTF is estimated at 71,744 years. The overall improvement in MTTF of the improved TMR system over the non-TMR system is 22.78×. If the slower clock rate of the improved TMR system is considered, the design has a MWBF improvement of 18.66×.

6 Improved SoC Post-Radiation Fault Analysis

In the radiation test of our design with targeted improvements, we were able to achieve an improvement in the MWBF of 1.54×. While impressive given the low-cost of our improvements, this is still a far cry from the estimated fivefold improvement from our initial predictions after fault analysis of the initial design. This prompted us to perform a similar fault analysis procedure to the one described in Section 3 on the radiation test data for the improved design. This section will describe that procedure and compare the vulnerability distributions for the original and improved TMR

Table 8. Comparison between of FPGA Device Resources Causing Design Failure

Resource	Occurrences (original)	Occurrences (improved)	Estimated Bits (original)	Estimated Bits (improved)
Interconnect	88 (71.5%)	21 (48.8%)	17,855	7,510
I/O Pin	22 (17.9%)	12 (27.9%)	4,464	4,291
Clocking	7 (5.7%)	2 (4.7%)	1,420	715
CLB	4 (3.3%)	8 (18.6%)	812	2,861
Undefined	2 (1.6%)	0 (0.0%)	406	0
Total	123	43	24,956 (0.042%)[a]	15,379 (0.026%)[a]

[a]Percentages are in comparison to the 59,145,600 type 0 CRAM locations in the XC7A200T.

designs to see where our targeted modifications were successful. We will also discuss potential additional improvements that can be made to the reliability of the system.

6.1 Fault Injection

The first step was to once again perform playback fault injection on all recorded CRAM upset locations to correlate the observed failures from the test to one or more sensitive CRAM upsets. Bits from scrub cycles that cause a failure during the playback and occur within 60 second prior to a failure from the test can be reasonably assumed to be the cause of the failure. We can further narrow down the sensitive bits to find the individual bit from each cycle that caused the failure by injecting each bit independently. The injection that causes the failure is marked as the sensitive bit from the cycle.

During this fault injection campaign, we were able to reproduce 34 of the 37 failures that occurred during the radiation test (not all failures can be reproduced as reported in Section 3). Of the 34 failures we were able to reproduce, 9 failures contained multiple sensitive bits that individually could cause a system error. The presence of a relatively large number of **multiple bit upsets (MBU)** failure modes suggests that more than one CRAM bit may be upset by ionizing particles [32]. Mitigation techniques to address these MBUs are beyond the scope of this article.

6.2 Fault Analysis

After gathering all sensitive bits that caused failures during the radiation test, we analyzed the causes of failure with BFAT. The FPGA device resources which were affected for each of the 37 bits can be found in Table 8. Though there were only 34 failures, 9 of these failures contained two sensitive bits, leading to a total of 43 possible failures. The results for the naive TMR design from Section 2.3 are also provided for comparison.

Considering the failure results of the improved design in Table 8, interconnect faults clearly contribute less to the overall failure rate of the improved system, with an estimated 7,510 sensitive bits compared to the 17,855 that were estimated in the naive TMR design. This 2.4× reduction of sensitivity in the interconnects is due to the changes we have made in reducing the size of non-triplicated net segments related to the DDR and global clock. Faults in clocking resources have also been reduced significantly (by 2.0×) due to the proper triplication of the global clock network. The estimated number of possible errors occurring in the I/O resources appears to be unchanged. This is unsurprising, as we applied no additional mitigation techniques that targeted the I/O.

Interestingly, the number of sensitive CLB bits appears to have increased by a factor of 3.5×. This was somewhat perplexing, as only 149 additional LUTs are utilized by the improved iteration of the designs. The additional LUT cells should also be protected by TMR voters. The issue became more

Table 9. Comparison between Original and Improved Design of Design Resources Causing Design Failure

Design Function	Occurrences (original)	Occurrences (improved)	Estimated Bits (original)	Estimated Bits (improved)
DDR Interface	91 (76.5%)	23 (67.6%)	19,084	10,402
Global Clock	22 (18.5%)	2 (5.9%)	4,614	905
Constant Logic	2 (1.7%)	0 (0.0%)	419	0
Reset Circuit	1 (0.8%)	0 (0.0%)	210	0
Misc. Circuitry	3 (2.5%)	9 (26.5%)	629	4,071
Total	119	34	24,956 (0.042%)[a]	15,378 (0.026%)[a]

[a]Percentages are in comparison to the 59,145,600 type 0 CRAM locations in the XC7A200T.

clear when examining the failures that each of these CLB faults was associated with. Of the 8 faults which were associated with CLB resources, 5 of them occurred in scrub cycles where MBU were required in order to reproduce the failure. Extrapolating from the 3 standalone CLB bits, only 1,073 of the 2,861 estimated sensitive CLB bits are related to SPFs. If the multi-bit failures are ignored, this is much closer to the 812 estimated CLB bits in the naive TMR design.

6.3 Comparison of Vulnerabilities

Once again, examining the design logic elements related to each failure can paint a better picture than simply examining the affected FPGA resources. Investigation of the 34 failures found for the improved TMR system revealed that most of the failure modes matched the original design failure modes for the 119 failures found when testing the naive TMR design. So, we can directly compare the occurrences for each vulnerable design function between the original and improved TMR designs. The results of this comparison can be found in Table 9.

6.3.1 DDR Interface. Though failures in the DDR interface are still the most common source of failure, their frequency has been decreased significantly by about 1.8×. All of these improvements have occurred in the interconnect. In the naive TMR design, 71 of the 91 (78.0%) failure bits associated with the DDR interface were related to the interconnect. With the improved design, we used manual placement and routing to lessen the amount of interconnect resources used by untriplicated DDR nets. This resulted in a significant reduction in interconnect-related failures in the DDR interface, with only 8 of the 23 (34.8%) DDR failures being caused by faults in the interconnect. These results indicate that we have removed around 76% of bits related to DDR interconnect errors. The other possible locations where we observed DDR failures were in CLB tiles and the I/O resources. Since we did not apply additional mitigation techniques to these areas, the sensitivity of these parts of the DDR interface is relatively unchanged.

As an example, consider the `ISERDESE2_15_n_4` signal, which is an input signal associated with the DDR interface. In our previous work [27], BFAT revealed that one of the bits which caused a failure created a short between this signal and another nearby signal. The short was created before the net split into three branches, thus corrupting the value of this signal for all TMR domains. As evident in Figure 11, this untriplicated net trunk is quite large, leaving many places for this short to occur. Figure 12 shows this same signal in the improved TMR design. The entire net has been shortened immensely due to the constraints on the destination cells of the net, leaving far fewer possible locations for interconnect faults to occur.

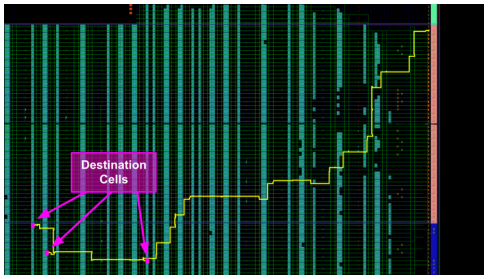

Fig. 11. Input net routing in the original design.

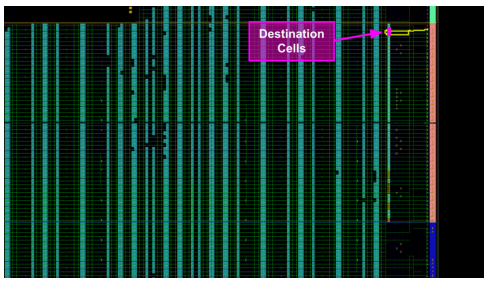

Fig. 12. Input net routing in the improved design.

6.3.2 Global Clock. Failures related to the global clock network have been reduced even further than those related to the DDR interface. The 5.1× reduction in clock-related errors is entirely a result of eliminating the optimization by Vivado's implementation tools which caused the clock to not be properly triplicated. Now that the system clock is triplicated, the only locations where errors can occur are in the source clock which the triplicated clock signals are generated from and in the `MMCME2` primitive which generates the triplicated clocks. Thus, it is unsurprising that of the two clock-related bits, one upset affected the `MMCME2` primitive and the other corrupted the routing of the source clock.

6.3.3 Constant Logic. During the radiation test of the improved design, the system did not experience any errors related to the global constant signals. The proportion of errors which were related to global constants was quite small in the original naive TMR design, at only 1.7%. Due to that and the low number of observed errors in the improved design, it is unsurprising that an error in the global signals happened to not occur. However, it is still possible that the true proportion of errors related to this part of the design has been reduced in the new design. Global constants have many local sources within CLB slices and the interconnects adjacent to them which are only depended upon by local cells (see Figure 13). The striping technique that was implemented ensures that any given CLB tile will only be populated with cells of one TMR domain. So, regardless of the global logic segment that is affected, only local cells which belong to the same domain will be corrupted. Since a failure in one domain will be masked by TMR voters, faults in constant logic are protected by TMR with striping.

6.3.4 Reset Circuit. There were also no failures detected in the reset circuitry in the improved design. In the original design, one failure was detected in the I/O pin for the system reset signal. Since the I/O cannot be triplicated, there was nothing we could do to improve the reliability of this

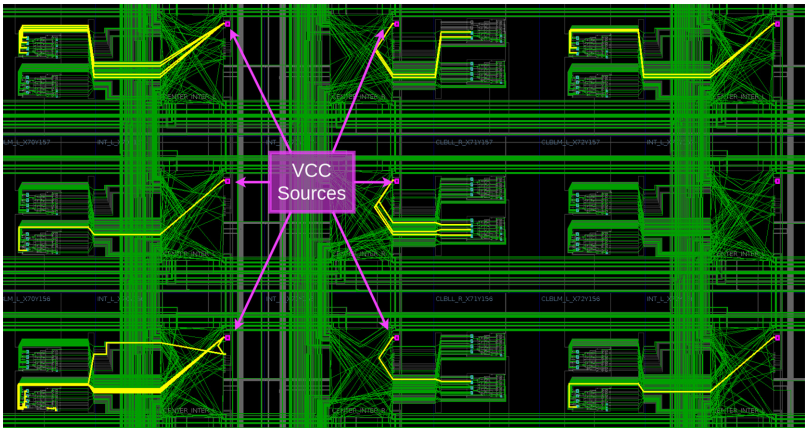

Fig. 13. Demonstration of local TIEOFF sites for the GLOBAL_LOGIC1 constant signal.

part of the design. So, the lack of errors in the reset line is most likely not due to any differences in the design, but rather the small amount of sample data collected during the radiation test.

6.3.5 Miscellaneous Circuitry. There are a significant amount of errors that occurred in miscellaneous locations throughout the design. All nine of these errors are due to the multi-bit upsets where each bit affected different TMR domains. Three of these failures were caused by upsets in physically adjacent locations and were thus likely caused by a single event. The other six pairs of bits were not physically adjacent, so they are most likely caused by two separate events.

Recall that only 2 of the 15 radiation-induced errors in the initial test were caused by multiple bits, whereas 9 of the 34 errors in the follow-up test of the improved design were caused by MBUs. The proportion of this type of error appears to have increased in the improved TMR system compared to the initial design. It seems as though failures caused by MBU make up a higher proportion of the errors due to the reduction in single-point failures that was achieved in the improved iteration of the design. Because there are less possible locations for SPFs to occur, the failures that do occur will be more heavily made up of multi-bit upsets.

7 Conclusion

This work presents a detailed failure analysis utilizing fault injection and radiation test results. The article provides an example of how these results guide several SEU mitigation techniques to a soft Linux SoC design. The improved design used the insights from a previous fault analysis effort to identify the vulnerable aspects of the design (DDR, clocking, and constants). The improvements made in this design include constrained placement of primitives associated with the DDR interface, locking of the clocking primitives, and striping. The improvements in reliability of this design were measured through fault injection and a radiation test experiment at LANSCE. The improved designs resulted in a 1.54× improvement to MWBF without any cost to utilization, performance, or even changes to the synthesized design.

This experiment demonstrates that measurable improvements can be made by targeting specific regions of the circuit that have been identified as vulnerable. Additional post-radiation analysis was performed on the improved design to understand the cause of failures. One insight from this new analysis was that in almost one third of the cases, the design failed due to two or more upsets occurring within the CRAM, suggesting the need for techniques to protect against multi-bit upsets.

Another insight suggests that the DDR controller is still the primary failure mode and additional DDR controller mitigation techniques are needed.

The results from these detailed failure analyses use real failure data to identify specific single-point failures which can be used to target possible mitigation techniques. Additional placement constraints can be performed to further reduce the single-point failures associated with the I/O. Mitigation of the DDR interface could include the exploration of DDR Data ECC and address/command parity checking. Data ECC could protect the DDR in the time interval when a DDR data I/O signal has been corrupted and when it has been repaired with configuration scrubbing. Address/command parity checking could be used to detect failed DDR transactions and thus issue a retry. This could also include the separation of DDR reset lines to prevent multiple IOSERDES failures from a single SEU. As future experiments and fault analysis provide more information, this can help guide the vendor tools in eliminating more single-point failures during implementation.

References

[1] 2001. *Measurement and Reporting of Alpha Particles and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices*. Technical Report. JEDEC, New York, NY. JESD89.

[2] 2020. Nexys Video Reference Manual. Retrieved from https://digilent.com/reference/programmable-logic/nexys-video/reference-manual

[3] Luis Alberto Aranda, Nils Johan Wessman, Lucana Santos, Alfonso Sánchez-Macián, Jan Andersson, Roland Weigand, and Juan Antonio Maestro. 2020. Analysis of the critical bits of a RISC-V processor implemented in an SRAM-based FPGA for space applications. *Electronics (Switzerland)* 9, 1 (2020). Article 20799292. DOI: https://doi.org/10.3390/electronics9010175

[4] Buildroot Association. [n. d.]. Buildroot Making Embedded Linux Easy. Retrieved from https://buildroot.org/

[5] Marco Bellato, P. Bernardi, D. Bortolato, A. Candelori, M. Ceschia, A. Paccagnella, M. Rebaudengo, M. S. Reorda, M. Violante, and P. Zambolin. 2004. Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA. In *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, Vol. 1, 584–589. DOI: https://doi.org/10.1109/DATE.2004.1268908

[6] Cody Brewer, Nicholas Franconi, Robin Ripley, Alessandro Geist, Travis Wise, Sebastian Sabogal, Gary Crum, Sabrena Heyward, and Christopher Wilson. 2020. NASA SpaceCube intelligent multi-purpose system for enabling remote sensing, communication, and navigation in mission architectures. In *Proceedings of the Small Satellite Conference 2020*.

[7] BYUCCL. [n. d.]. Bitstream Fault Analysis Tool. Retrieved December 26, 2022 from https://github.com/byuccl/bfat

[8] Matthew Cannon, A. Keller, and M. Wirthlin. 2018. Improving the effectiveness of TMR designs on FPGAs with SEU-aware incremental placement. In *Proceedings of the IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM '18)*. 141–148. DOI: https://doi.org/10.1109/FCCM.2018.00031

[9] Matthew J. Cannon, A. M. Keller, H. C. Rowberry, C. A. Thurlow, A. Pérez-Celis, and M. J. Wirthlin. 2019. Strategies for removing common mode failures from TMR designs deployed on SRAM FPGAs. *IEEE Transactions on Nuclear Science* 66, 1 (Jan 2019), 207–215. DOI: https://doi.org/10.1109/TNS.2018.2877579

[10] Fulvio Corno, M. S. Reorda, and G. Squillero. 2000. RT-level ITC'99 benchmarks and first ATPG results. *IEEE Design & Test of Computers* 17, 3 (2000), 44–53. DOI: https://doi.org/10.1109/54.867894

[11] Paul Graham, Michael Caffrey, Jason Zimmerman, D. Eric Johnson, Prasanna Sundararajan, and Cameron Patterson. 2003. Consequences and categories of SRAM FPGA configuration SEUs. *Proceedings of the 5th Annual International Conference Military Aerospace Programmable Logic Devices*.

[12] Ammon Gruwell, P. Zabriskie, and M. Wirthlin. 2016. High-speed FPGA configuration and testing through JTAG. In *Proceedings of the 2016 IEEE AUTOTESTCON*, 218–225. DOI: https://doi.org/10.1109/AUTEST.2016.7589601

[13] Matt Hamblen. 2020. NASA Mars rover perseverance launches on time Thursday to find evidence of life on red planet. Retrieved December 26, 2022 from https://www.fierceelectronics.com/electronics/nasa-mars-rover-perseverance-launches-thursday-to-find-evidence-life-red-planet

[14] Yoshihiro Ichinomiya, S. Tanoue, M. Amagasaki, M. Iida, M. Kuga, and T. Sueyoshi. 2010. Improving the robustness of a softcore processor against SEUs by using TMR and partial reconfiguration. In *Proceedings of the 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines*, 47–54. DOI: https://doi.org/10.1109/FCCM.2010.16

[15] Florent Kermarrec, Sébastien Bourdeauducq, Hannah Badier, and Jean-Christophe Le Lann. 2019. LiteX: An open-source SoC builder and library based on Migen Python DSL. In *Proceedings of the Oregon Society of Dermatology Associates* (*OSDA '19), Colocated with DATE 2019 Design Automation and Test in Europe*.

[16] Paul. W. Lisowski, C. D. Bowman, G. J. Russell, and S. A. Wender. 1990. The Los Alamos national laboratory spallation neutron sources. *Nuclear Science and Engineering* 106, 2 (1990), 208–218. DOI: https://doi.org/10.13182/NSE90-A27471

[17] LiteX-Hub. [n. d.]. Linux on LiteX VexRiscv. Retrieved February 3, 2022 from https://github.com/litex-hub/linux-on-litex-vexriscv

[18] Shih-Fu Liu, G. Sorrenti, P. Reviriego, F. Casini, J. A. Maestro, M. Alderighi, and H. Mecha. 2012. Comparison of the susceptibility to soft errors of SRAM-based FPGA error correction codes implementations. *IEEE Transactions on Nuclear Science* 59, 3 (2012), 619–624. DOI: https://doi.org/10.1109/TNS.2012.2193417

[19] Thandassery S. Nidhin, Anindya Bhattacharyya, R. P. Behera, and T. Jayanthi. 2018. A review on SEU mitigation techniques for FPGA configuration memory. *IETE Technical Review* 35, 2 (2018), 157–168. DOI: https://doi.org/10.1080/02564602.2016.1265905

[20] Patrick Ostler, Michael Caffrey, Derrick Gibelyou, Paul Graham, Keith Morgan, Brian Pratt, Heather Quinn, and Michael Wirthlin. 2010. SRAM FPGA reliability analysis for harsh radiation environments. *Nuclear Science, IEEE Transactions on* 56 (Jan. 2010), 3519–3526. DOI: https://doi.org/10.1109/TNS.2009.2033381

[21] Heather Quinn, Paul S. Graham, Keith Morgan, Jim Krone, Michael P. Caffrey, and Michael J. Wirthlin. 2008. An introduction to radiation-induced failure modes and related mitigation methods for Xilinx SRAM FPGAs. In *Proceedings of the ERSA*, 139–145.

[22] Heather Quinn, William H. Robinson, Paolo Rech, Miguel Aguirre, Arno Barnard, Marco Desogus, Luis Entrena, Mario Garcia-Valderas, Steven M. Guertin, David Kaeli, Fernanda Lima Kastensmidt, Bradley T. Kiddie, Antonio Sanchez-Clemente, Matteo Sonza Reorda, Luca Sterpone, and Michael Wirthlin. 2015. Using benchmarks for radiation testing of microprocessors and FPGAs. *IEEE Transactions on Nuclear Science* 62, 6 (2015), 2547–2554.

[23] Felix Siegle, Tanya Vladimirova, Jørgen Ilstad, and Omar Emam. 2015. Mitigation of radiation effects in SRAM-based FPGAs for space applications. *ACM Computing Surveys (CSUR)* 47, 2 (2015), 1–34.

[24] Alexander Sirotkin. 2011. Roll your own embedded Linux system with buildroot. *Linux Journal* 2011, 206 (2011), 7.

[25] Dallin Skouson, Andrew Keller, and Michael Wirthlin. 2020. Netlist analysis and transformations using SpyDrNet. In *Proceedings of the Python in Science Conference,* 41–47.

[26] SpinalHDL. [n. d.]. VexRiscv. Retrieved December 26, 2022 from https://github.com/SpinalHDL/VexRiscv

[27] Andrew Elbert Wilson, Nathan Baker, Ethan Campbell, Jackson Sahleen, and Michael Wirthlin. 2023a. Post-radiation fault analysis of a high reliability FPGA Linux SoC. In *Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA '23)*. ACM, New York, NY, 123–133. DOI: https://doi.org/10.1145/3543622.3573191

[28] Andrew E. Wilson, Sam Larsen, Christopher Wilson, Corbin Thurlow, and Michael Wirthlin. 2021. Neutron radiation testing of a TMR VexRiscv soft processor on SRAM-based FPGAs. *IEEE Transactions on Nuclear Science* 68, 5 (2021), 1054–1060. DOI: https://doi.org/10.1109/TNS.2021.3068835

[29] Andrew E. Wilson, C Thurlow, and M. Wirthlin. 2021. Fault injection testing of fault tolerant RISC-V soft processors on Xilinx SRAM-based FPGAs. *Journal of Research and Educational Research Evaluation* 39, 1 (Aprile 2021), 356–361.

[30] Andrew E. Wilson, Michael Wirthlin, and Nathan G. Baker. 2023b. Neutron radiation testing of RISC-V TMR soft processors on SRAM-based FPGAs. *IEEE Transactions on Nuclear Science* 70, 4 (2023), 603–610. DOI: https://doi.org/10.1109/TNS.2023.3235582

[31] Michael Wirthlin. 2015. High-reliability FPGA-based systems: space, high-energy physics, and beyond. *Proceedings of the IEEE* 103, 3 (2015), 379–389. DOI: https://doi.org/10.1109/JPROC.2015.2404212

[32] Michael Wirthlin, David Lee, Gary Swift, and Heather Quinn. 2014. 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.

Received 2 September 2023; revised 11 April 2024; accepted 2 June 2024