

**Evaluation of Algorithm-Based Fault Tolerance for
Machine Learning and Computer Vision under
Neutron Radiation**

by

Seth Roffe

B. Phil. Physics, Astronomy, and Mathematics, University of
Pittsburgh, 2017

Submitted to the Graduate Faculty of
the Swanson School of Engineering in partial fulfillment
of the requirements for the degree of
Master of Science

University of Pittsburgh

2020

UNIVERSITY OF PITTSBURGH
SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Seth Roffe

It was defended on

March 6, 2020

and approved by

Alan D. George, Ph.D., RH Mickle Endowed Chair and Professor, Department of Electrical
and Computer Engineering

Zhi-Hong Mao, Ph.D., Professor, Department of Electrical and Computer Engineering

Ahmed Dallal, Ph.D., Assistant Professor, Department of Electrical and Computer
Engineering

Thesis Advisor: Alan D. George, Ph.D., RH Mickle Endowed Chair and Professor,
Department of Electrical and Computer Engineering

Copyright © by Seth Roffe
2020

Evaluation of Algorithm-Based Fault Tolerance for Machine Learning and Computer Vision under Neutron Radiation

Seth Roffe, M.S.

University of Pittsburgh, 2020

In the past decade, there has been a push for deployment of commercial-off-the-shelf (COTS) avionics due in part to cheaper costs and the desire for more performance. Traditional radiation-hardened processors are expensive and only provide limited processing power. With smaller mission budgets and the need for more computational power, low-cost and high-performance COTS solutions become more attractive for these missions. Due to the computational capacity enhancements provided by COTS technology, machine-learning and computer-vision applications are now being deployed on modern space missions. However, COTS electronics are highly susceptible to radiation environments. As a result, reliability in the underlying computations becomes a concern. Matrix multiplication is used in machine-learning and computer-vision applications as the main computation for decisions, making it a critical part of the application. Therefore, the large time and memory footprint of the matrix multiplication in machine-learning and computer-vision applications makes them even more susceptible to single-event upsets.

In this thesis, algorithm-based fault tolerance (ABFT) is investigated to mitigate silent data errors in machine learning and computer vision. ABFT is a methodology of data error detection and correction using information redundancy contained in separate data structures from the primary data. In matrix multiplication, ABFT consists of storing checksum data in vectors separate from the matrix to use for error detection and correction. Fault injection into a matrix-multiplication kernel was performed prior to irradiation. Irradiation was then performed on the kernel under wide-spectrum neutrons at Los Alamos Neutron Science Center to observe the mitigation effects of ABFT. Fault injections targeted towards the general-purpose registers show a $48\times$ reduction in data errors using data-error mitigation with ABFT with a negligible change in run-time. Cross-section results from irradiation show a $5.3\times$ improvement in reliability of using ABFT as opposed to no mitigation with

a > 99.9999 confidence level. The results of this experiment demonstrate that ABFT is a viable solution for run-time error correction in matrix multiplication for machine-learning and computer-vision applications in future spacecraft.

Table of Contents

1.0 Introduction	1
1.1 Background	2
1.1.1 Fault Injection	3
1.1.2 Neutron Beam Testing	4
1.1.3 Reliability Techniques	4
1.1.4 Algorithm-Based Fault Tolerance	5
1.1.5 Machine Learning and Computer Vision in Spacecraft	7
1.1.6 Computational Complexity	8
1.2 Platform Selection	9
2.0 Experimentation	10
2.1 Methodology	10
2.1.1 Fault Injection	10
2.1.2 Irradiation	11
2.1.3 Employed ABFT Configuration	13
2.1.4 Statistical Analysis	14
2.2 Experimental Results	15
2.2.1 High-Level Fault Injection	16
2.2.2 Low-Level Fault Injection	19
2.2.3 Irradiation	22
2.2.4 Statistical Significance	28
3.0 Conclusions and Future Work	30
Bibliography	33

List of Tables

1	Summary of Fault-Injection Reliability and Runtime Results for 1000 Executions.	20
2	Summary of Beam-Test Reliability and Runtime Results Over Four Days of Irradiation.	28
3	Summary of Primary Comparisons Between Matrix Multiplication With ABFT for Fault Mitigation and Without Error Mitigation	29

List of Figures

1	The experimental setup consisting of three PYNQ-Z1 boards irradiated in a line.	12
2	Neural network number of different elements in the output matrix differences and number of mispredictions vs. number of fault injections.	16
3	Neural network mean execution time vs. the number of injections.	18
4	Fault-injection outputs from DrSEUs without any form of error mitigation. .	19
5	Fault-injection outputs from DrSEUs with ABFT.	20
6	Neutron beam dosimetry over time over the days of irradiation: September 14, 2018 - September 17, 2018.	22
7	Results from four days of irradiation running a 500×500 matrix multiplication kernel without error mitigation.	23
8	Results from four days of irradiation running a 500×500 matrix multiplication kernel while using ABFT for data-error mitigation.	24
9	Number of errors corrected in the output matrix (SEC) and errors detected in the input matrices (SED).	25
10	Average runtime of 500×500 matrix multiplication on the processor side of the Xilinx Zynq 7020 for both tested configurations.	26
11	Execution times relative to the average runtime without any fault mitigation.	27

1.0 Introduction

Using commercial-off-the-shelf (COTS) electronics in hazardous environments, such as space, presents a number of difficulties, leading to the development of new technologies and strategies. For example, the NSF Center for Space, High-performance and Resilient Computing (SHREC), located at the University of Pittsburgh, created the CSP Space Computer, employing a Xilinx Zynq-7020 SoC with a dual-core ARM Cortex-A9 processor and Artix-7 field-programmable gate array (FPGA), for radiation-tolerant, high-performance, and reconfigurable space computing [16].

The upgraded version of the CSP, the SHREC Space Processor (SSP) is currently being developed at the NSF SHREC Center, employing a more powerful Xilinx Zynq-7030 or Zynq-7045 SoC with a dual-core ARM Cortex-A9 and Kintex-7 FPGA to increase the FPGA capabilities. In addition to the high performance of commercial processors, both space computers offer the benefits of FPGA reconfigurability and acceleration.

Traditional radiation-hardened electronic systems are often employed to tolerate a high total ionizing dose (TID) and single-event effects (SEEs) from the harsh space environment. However, these processors tend to be generations behind their COTS counterparts in terms of performance, energy-efficiency, and cost. Using commercial devices combined with radiation-hardened components and dependability-enhancing software allows for high-performance while still remaining a high degree of radiation tolerance. Better performance while still staying close to the reliability of a radiation-hardened system enables the use of more complex applications in harsh environments such as space. Therefore, the CSP and SSP offer a COTS solution for reliable and high-performance space computing. The CSP has been chosen and flown as part of two U.S. Department of Defense missions to the International Space Station, and have been adopted by several SHREC partners for deployment in future missions.

Using COTS solutions, the capability of on-board processing for machine-learning and computer-vision applications improves. However, COTS electronics are very susceptible to radiation. As a result, the reliability of computationally complex kernels, such as matrix

multiplication for machine-learning and computer-vision, becomes a concern. Namely, single-event upsets (SEUs) occur when a high-energy particle strikes a microelectronic component and either changes the state of the circuit [14]. These faults can then manifest as transient-data errors, corrupting the output of the application.

Algorithm-based fault tolerance (ABFT) was introduced by Huang and Abraham as a method of information redundancy in matrix operations [11]. Their algorithm focuses on encoding a vector of data using a checksum in order to detect and correct data errors caused by SEUs. Since the dimensionality of the data is reduced on encoding, the memory overhead is relatively insignificant.

In this thesis, the efficacy of ABFT for error mitigation in machine-learning and computer-vision applications was tested by means of fault injection and neutron irradiation with a focus specifically on data errors as opposed to control errors. The results demonstrate the use of ABFT to ensure that compute time is not being wasted on corrupted data. This technique's capabilities is then assessed by examining the reliability and performance impact on processing time.

1.1 Background

This section provides a cursory overview of reliability testing in computing, ABFT, machine learning and computer vision in spacecraft, and computational complexity. This thesis combines strategies from variously studied fields in order to demonstrate methods to create a more reliable system for applications.

1.1.1 Fault Injection

Fault injection is a methodology that is commonly used to validate the dependability of fault-tolerant systems[17]. Typically fault-injection experiments consist of controlled tests where the behavior of a system in the presence of faults is observed by writing faults directly into the system during operations. Therefore, the reliability of a system can be measured and tested before deployment. Similarly, any vulnerabilities in the system can be found and adjusted through fault injection.

The two main categories of fault injection used in this experiment consist of hardware and software fault injection. Hardware-based fault injection happens at a physical level, where faults are induced by disturbing the hardware from the environment. This disturbance could consist of causing a dip in the voltage of the system's power supply, radiation-beam testing to cause electromagnetic interference in the computing, or even manually modifying the value of specific pins on the device-under-test [17].

Software-based fault injection can happen at a variety of levels of abstraction. Namely, the registers can be affected to induce faults on a low-level. The objective of this kind of testing would be to observe the response of the system for data faults as well as control faults. Similarly this type of low-level injection gives a more realistic view of how the system would behave in a harsh environment. Conversely, faults can be injected at a high-level in software. This injection would consist of directly changing elements of data in software to observe the response of the tested reliability technique. Injecting directly in the data increases the number of samples by guaranteeing where the injection takes place, but it unfortunately does not give a good indication of how the system would behave in a harsh environment. This high-level fault injection would primarily be used to test the response of a specific reliability technique.

1.1.2 Neutron Beam Testing

Models describing galactic cosmic radiation show that the majority of radiation experienced in orbit would consist of protons and heavy ions [4]. However, neutron beam testing proves to be useful for testing error mitigation strategies with a lower risk of permanently damaging the device under test. Specifically, when testing the response of a mitigation technique to SEUs, the source of said SEU becomes irrelevant.

Beam testing is popular in the field of computing to classify SEUs of new systems and mitigation strategies. Knowledge of radiation data is critical to the design process for space missions by giving an overview of the form of upsets a system might see. For example, Anderson et al. performed a neutron beam experiment on the Xilinx UltraScale+ MPSoC to observe the number system crashes and silent data errors invoked from radiation [2]. With each new device destined for spaceflight, beam testing is vital in determining its viability in space.

Similarly, neutron beam testing is commonly used to test the robustness of systems to SEUs. NASA Langley Research Center and Honeywell used neutron radiation testing to test the robustness of their flight control computer architecture to test flight capabilities in a hazardous environment. In their experiment, they were able to measure the ability of their architecture to recover from neutron-induced faults.

The experiment detailed in this thesis follows a structure similar to the latter experiment. Namely, the ability for ABFT to recover faults in a radiation environment was measured to test its viability for applications in space missions.

1.1.3 Reliability Techniques

All reliability techniques center around one main concept: redundancy. In order to detect or correct errors in a fallen system, there must be some form of redundant information that is able to detect when something changes. Redundancy can take many forms and typically fall within one of five categories: hardware redundancy, information redundancy, network redundancy, software redundancy, and time redundancy [12]. The techniques presented in this thesis fall under hardware and information redundancy.

Hardware redundancy involves adding additional hardware to catch any faulty modules. One popular form of hardware redundancy is modular redundancy which involves having multiple modules run the same application and compare the outputs to ensure they match. One common forms of this kind of redundancy is Triple Modular Redundancy which involves using three redundant modules and taking a majority voter to get the output. This type of redundancy enables single error correction since a faulty module will be out-voted by the other two [12]. However, it tends to be expensive since three redundant hardware modules are required. Another form of modular redundancy is duplex redundancy which involves using two redundant modules and restarting the execution if the outputs differ. Duplex redundancy enables single error detection only since there is no method in determining which module has the correct output.

Conversely, information redundancy involves having additional bits of data that holds identifying information about the application. The number of errors that can be corrected or detected depends directly on how many extra pieces of information are stored [12]. This extra information causes the memory overhead to increase as the number of detectable errors increases. Information redundancy tends to be cheaper than hardware redundancy since no additional modules are needed at the cost of being less reliable. ABFT falls under this form of redundancy as it holds encoded data from a matrix in checksum vectors separate from the matrices they represent.

1.1.4 Algorithm-Based Fault Tolerance

To make matrix operations more reliable, ABFT uses extra bits to encode data in order to detect and correct errors. A popular encoding sequence for this method is performed by adding additional checksum vectors, called weighted checksums, to the matrix operands. These vectors contain containing redundant information about the matrix, such as the sum of the rows or the sum of the columns. These weighted checksums can be expanded to enable detection and correction of more errors within the matrix as long as it contains different types of redundant data. For example, two possible row checksum vectors could be the sum of the columns and the sum of 2^k times the columns where k is the index of the element's

row. If there are n checksum vectors, then either n errors can be detected, but not necessarily corrected, or $\lfloor \frac{n}{2} \rfloor$ errors can be corrected [11]. If an error were to exist in one of the elements of the operand, the information contained in the matrix and the checksums would be different, leading to an error detection. With multiple checksums, the error can be pinpointed in the matrix using the indices of the mismatched checksum vector elements. Then, through subtracting the checksum value from the sum of the row or column while excluding the error value, the corrupted data can be recovered in the original matrix without the need to restart the entire application. There has been much research into different applications and techniques of ABFT. There are many matrix operations that are of interest to the studies of fault-tolerant computing, such as lower-upper (LU) decomposition and gaussian elimination that use similar checksum vectors to detect errors in computation[13]. As mentioned, ABFT is specific to the application under study, so each operation or algorithm needs to be verified that the output preserves the encoding scheme prior to use.

NASA’s Jet Propulsion Laboratory (JPL) has investigated the use of ABFT for onboard data analysis in the presence of SEUs. They performed fault injection as a way to inject different amounts of faults into their application and view ABFT’s response. With this methodology, JPL is able to simulate a radiative environment by forcing SEUs into their application. As a test case for their ABFT techniques, they used prediction on an SVM binary classifier. Their proposed approach used checksums before and after critical matrix-multiplication operations. Testing with 100 different SVMs whose matrix computations were exposed with varying rates of SEU faults, they found that detection improved at higher SEU rates [9].

ABFT is also a popular approach in the high-performance computing (HPC) community for its ability to detect and correct errors without stopping the computation [7]. With increasingly large runtimes from increasingly more complex applications, the mean time to failure (MTTF) sharply declines. Therefore, effects from multiple fault tolerant techniques have been studied. For example, Du et al. employ ABFT methods with checkpointing methods for dense matrix factorizations to significantly increase fault tolerance while keeping the overhead low [8].

Aside from the research into the application side of ABFT, different encoding methods has also been studied to ensure the most efficient use of data redundancy. Anfinson and Luk detail a linear algebra model of the checksum method [3]. This model provides a method of selecting "proper" weights in a weighted checksum. In other words, using different weights defined in the checksum vectors gives methods to enable error correction as well as error detection.

This experiment focuses on evaluating the efficacy of ABFT for matrix multiplication specifically with machine learning and computer vision applications in spacecraft. Past research with ABFT has shown that it is an attractive solution to applications that use matrix operations. In order to use the methodology in the field intelligently, its performance needs to be measured. Due to the algorithm-specific nature of ABFT, the methodology needs to be applied to the application at hand. Therefore, it is much more efficient to test ABFT on kernels upon which many useful applications rely as opposed to testing specific applications in order to view its effects on a broad range of cases.

1.1.5 Machine Learning and Computer Vision in Spacecraft

To enable spacecraft autonomy, researchers have been examining the use of machine-learning algorithms that employ neural networks for sensing experiments. Benediktsson et al. show that neural networks are significantly more accurate at pattern recognition and classification in remote sensing data than traditional statistical methods [5]. However, they also cite that the neural network algorithm used is more computationally intensive than statistical methods, resulting in longer execution times. These long execution times leads to a greater higher susceptibility to radiation effects.

Similarly, computer-vision applications are essential in autonomous mission operations. Ho and McClamroch detail a formulation of automatic spacecraft docking using computer vision [10]. Any application built around image or video data will make heavy use of matrix operations. Therefore, the methodologies detailed are very complex, leading to a long runtime on embedded platforms. Therefore, data reliability is essential to avoid catastrophic failures.

In the realm of space applications, neural-network training would occur on the ground using previously acquired data as the training set. Once trained, the weights would be transmitted to the onboard processor, and only inference would occur onboard for classification purposes. The desire to classify objects in real-time necessitates the need for fast neural-network inference, which relies on both hardware and software that can perform matrix multiplications quickly and resiliently due to the radiative conditions in space.

1.1.6 Computational Complexity

To increase the reliability of matrix-multiplication-based applications in software, multiple methods can be considered. Either the data must have redundancy, or the application must be made faster to avoid vulnerability. Similarly, the probability of a radiation-induced error increases with the size of the data (n) and the amount of time that the data is under computation (t). Execution time of matrix multiplication is dependent on n and expressed with *Big O* notation, describing the worst-case execution time for a given algorithm, $O(f(n))$ for some function f as n becomes sufficiently large.

Stothers' thesis covers the details of the computational complexity behind matrix multiplication. The serial matrix multiplication algorithm is known to have a complexity of $O(n^3)$. Using the recursion methodology shown in Stothers' paper, he was able to reduce the complexity to approximately $O(n^{2.807})$ [15]. However, this improvement is insufficient to significantly reduce the risk involved in matrix multiplication in a radiative environment. This research shows that the complexity of matrix multiplication is bounded from below asymptotically. Therefore, the only viable way to reduce execution time of matrix multiplication is to implement parallelization techniques. However, adding parallel processing also increases the application's vulnerability to radiation due to the larger memory-footprint of running multiple cores. Thus, for large datasets and repeated computations, data redundancies such as ABFT present an attractive solution to the reliability problem.

1.2 Platform Selection

This experiment was tested on the processing system (PS) side of the Digilent PYNQ-Z1 with a Xilinx Zynq 7020 system-on-chip (SoC) to emulate the CSP. The CSP features the same commercial ZC7020 SoC along with radiation-hardened watchdog and power circuitry to ensure the reliability of critical components[16]. While COTS electronics offer more computational power to perform intensive algorithms, they are very susceptible to the radiation environment. Therefore, if any of these algorithms are used on a space processor such as the CSP, fault mitigation techniques are necessary to ensure the accuracy of the application.

2.0 Experimentation

This chapter describes the methodology and results of the experiment. The methodology of this experiment consists of fault-injection and irradiation to test the efficacy of ABFT.

2.1 Methodology

This section explains the methodology of each step of the experiment performed. The experiment can be separated into fault injection and irradiation. Fault injection was performed separately from irradiation as an original test for the capabilities of ABFT and expected results.

2.1.1 Fault Injection

Fault injection was performed at a high-level and a low-level with respect to the application. The high-level injector injected different amounts of errors into the matrix operands of a simple neural network by choosing a random element and changing its value. This high-level injection was used to compare ABFT to another popular reliability method, duplex redundancy. Duplex redundancy involves running the same application on two cores of the CPU and restarting if they have differing results. The number of faults injected varied to observe the response of the underlying method.

A neural-network classifier run on the MNIST dataset was used to measure the effects of the SEUs on accuracy and runtime for ABFT, duplex, and without any reliability techniques. 5,500 MNIST images were used for a training set over 15 epochs and 5,000 were used for a validation set. Inference was performed without any injections to obtain a golden, expected output that was used to compare to all injections. The results of these fault injections gave a direct comparison between the responses of ABFT and other reliability techniques to data errors.

Low-level fault injection involved performing bitflips on random registers within the processor. This method gives a response that is more realistic to radiation effects as it is hardware-dependent. Register fault injection was performed prior to radiation testing to simulate possible outcomes of the radiation experiment. These injections were performed on a 500×500 matrix multiplication kernel using the Dynamic robust Single Event Upset simulator (DrSEUs)[6] on the Digilent PYNQ-Z1 to ensure a consistent platform.

Low-level injections were performed at random times during executions on random registers on the ARM Cortex-A9 processor. General-purpose registers were targeted specifically to cause more data errors since other types of faults were not included in the scope of this experiment. Since ABFT is data-specific, it would be ineffective at detecting any kernel-related faults. The results of this fault injection show an idealistic view of the expected results of the radiation experiment.

2.1.2 Irradiation

Three Digilent PYNQ-Z1 boards were used as the device-under-test (DUT) for this experiment. Irradiation was done at the Los Alamos Neutron Science Center (LANSCE) Weapons Neutron Research (WNR) facility using the 4FP30R/ICE-II instrument for neutron irradiation. Despite the space environment being dominantly composed of proton and heavy-ion radiation, neutron irradiation provides valuable insight on the response of the mitigation strategy to induced errors without causing significant permanent damage to the device. The three boards were irradiated in a line and labeled as one, two, and three. Boards one and three use ABFT-fault mitigation in matrix multiplication, while board two does not use any mitigation techniques. Neutron dosimetry was tracked using a dosimeter to observe the relative neutron flux passing in 10-second intervals. The boards were placed with the Zynq SoC chips aligned to the 1-inch collimated beam, and irradiated simultaneously to ensure a similar neutron fluence on each one, as shown in Figure 1.

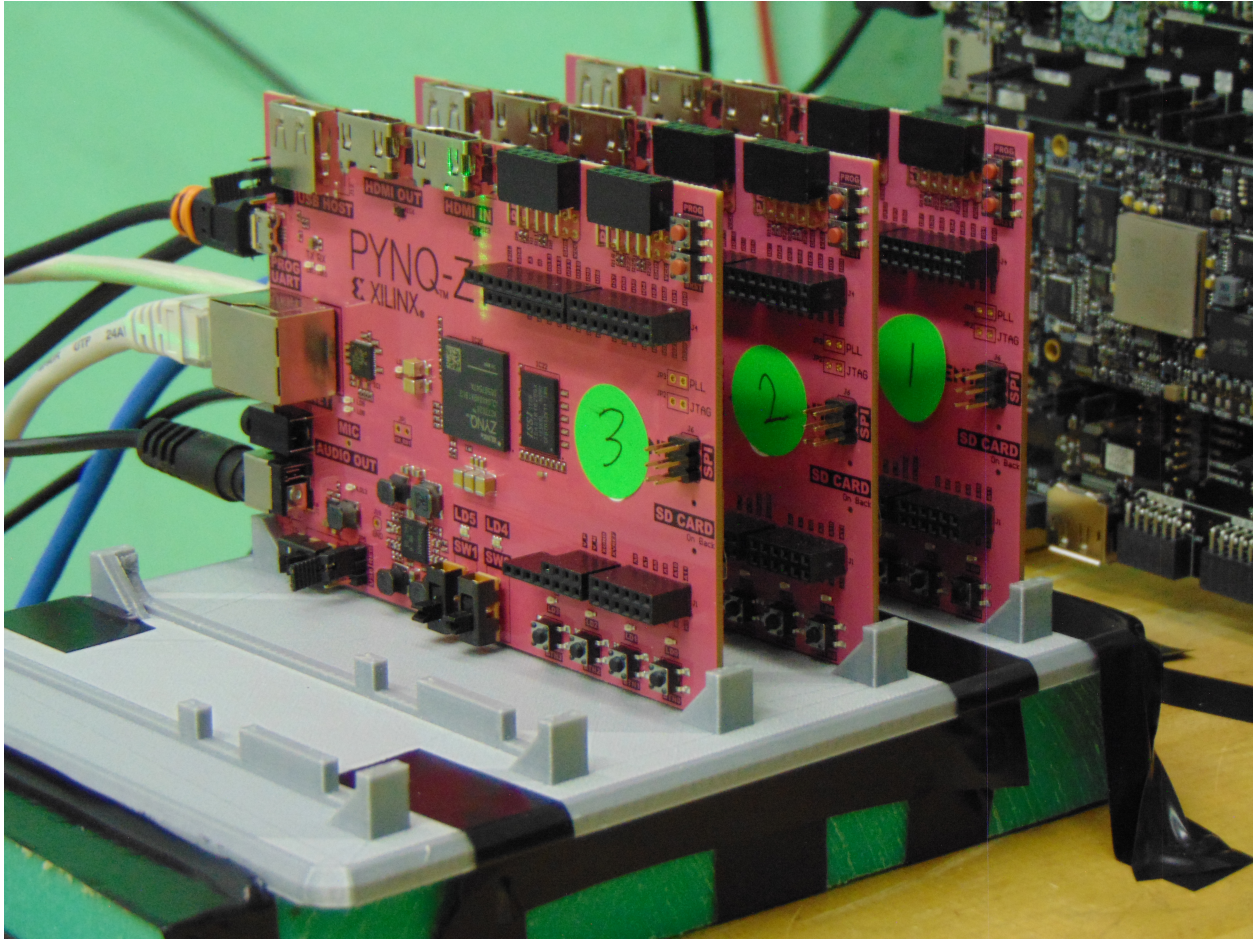


Figure 1: The experimental setup consisting of three PYNQ-Z1 boards irradiated in a line.

Each PYNQ board employed a matrix-multiplication kernel using the serial matrix-multiplication algorithm. Two 500×500 matrices of 16-bit integers were multiplied together to represent a large multiplication that would take place in a machine-learning or computer-vision algorithm, such as a neural network. These added checksum vectors give $2 \times 500 = 1000$ bytes memory overhead for each input matrix and $2 \times 2 \times 500 = 2000$ bytes of memory overhead for the output, totaling at 4000 bytes. This added memory is less than one percent of the memory needed to hold the matrices, so ABFT can be considered to have a negligible memory overhead. Boards one and three used ABFT mitigation techniques, while board two did not have any form of fault mitigation. This ordering was chosen to prevent any significant bias from specific boards being closer to the beam.

2.1.3 Employed ABFT Configuration

ABFT was chosen to enable single-error correction (SEC) for each of the three separate experiments, as recommended by Al-Yamani et. al since it shows an efficient correction-to-overhead ratio for matrix multiplication [1]. One checksum was generated for each input matrix: a row checksum representing the sum of the columns of the first and a column checksum representing the sum of the rows of the second. Multiplication of these two matrices and checksums results in an output matrix containing two checksums enabling single-error correction and double-error detection (SEC/DED). Similarly, this methodology allows for single-error detection (SED) between reading the data and the calculation in the initial matrices that are being multiplied. In order to keep the checksums separate from the data, each checksum was kept in a separate array data structure.

Before multiplying, the input matrices are compared with their checksums, and are read in again if there were any discrepancies. The focus of this experiment was on errors encountered during processing. External input/output errors were not considered due to the fact that external errors are sensor-specific and cannot be addressed with ABFT. After any multiplication, the checksum vectors are also multiplied by the matrix they are not associated with to calculate the two checksums for the output. The output matrix is then checked with these checksums and corrected if needed by cross referencing the location index

of the error by which elements of the checksums disagree with the output. The element is then restored by subtracting the sum of the rest of the row or column from the checksum element. More than one error cannot be corrected due to SEC being the assumed scope of the redundancy technique. The generalized matrix configurations for $A \times B = C$ with ABFT supporting SEC/DED in the output is used with A, B , and C defined as in Equation 2.1.

$$\begin{aligned}
 A &= \left[\begin{array}{ccc|c} A_{1,1} & \cdots & A_{1,n} & \\ \vdots & \ddots & \vdots & \\ A_{m,1} & \cdots & A_{m,n} & \end{array} \right] \\
 &\quad \left[\begin{array}{ccc|c} \sum_{i=1}^m A_{i,1} & \cdots & \sum_{i=1}^m A_{i,k} & \end{array} \right] \\
 B &= \left[\begin{array}{ccc|c} B_{1,1} & \cdots & B_{1,k} & \sum_{i=1}^k B_{1,i} \\ \vdots & \ddots & \vdots & \vdots \\ B_{n,1} & \cdots & B_{n,k} & \sum_{i=1}^k B_{n,i} \end{array} \right] \\
 \Rightarrow C &= \left[\begin{array}{ccc|c} C_{1,1} & \cdots & C_{1,k} & \sum_{i=1}^k C_{1,i} \\ \vdots & \ddots & \vdots & \vdots \\ C_{m,1} & \cdots & C_{m,k} & \sum_{i=1}^k C_{m,i} \end{array} \right] \\
 &\quad \left[\begin{array}{ccc|c} \sum_{i=1}^m C_{i,1} & \cdots & \sum_{i=1}^m C_{i,k} & \end{array} \right]
 \end{aligned} \tag{2.1}$$

2.1.4 Statistical Analysis

To evaluate statistical significance, various tests were performed on the final results. In order to ensure a good distribution of observed errors, a sample size of about 16,000 events was taken. The cross-section, calculated as the number of data errors divided by the neutron fluence, was used to calculate the reliability of ABFT as an error mitigation strategy. To evaluate the efficacy of adding ABFT to the operation, the mean-work-to-failure (MWTF) was calculated. Finally, to ensure that the results were statistically significant, a z-test was calculated between the distribution using ABFT and the distribution without error mitigation.

2.2 Experimental Results

The results of this experiment have been split into two separate events: fault injection and irradiation. Fault injection was performed to give an idealistic view of the results prior to irradiation. In this way, the two scenarios can be examined separately and then compared after analysis.

2.2.1 High-Level Fault Injection

High-level fault injection was performed to compare ABFT to duplex redundancy in regard to the accuracy and runtime of a neural network. The number of mispredictions was directly compared to the number of data faults that were injected.

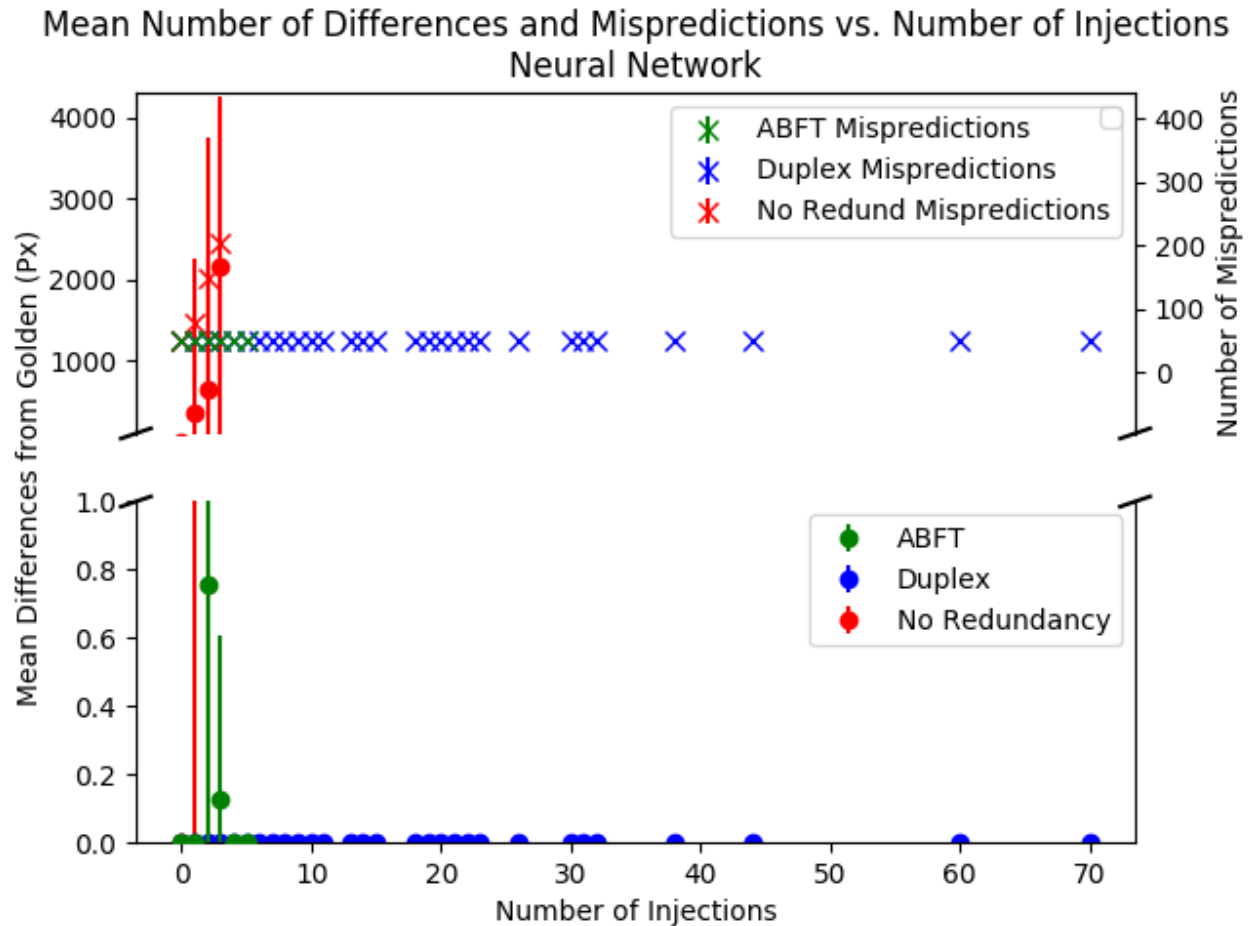


Figure 2: Neural network number of different elements in the output matrix differences and number of mispredictions vs. number of fault injections.

Figure 2 shows the mean number of mispredictions and the mean number of differences compared with the golden matrix containing classification probabilities with respect to the number of injections. In the figure, x symbols represent the average misprediction with a scale on the right side of the plot, while dots represent discrepancies with the golden output

matrix classification probabilities with a scale on the left side of the plot. Under normal operations with no injected faults, the neural network exhibited 51 mispredictions, or a 94.9% test accuracy. When faults were injected into the network, the classification accuracy degraded rapidly. For one, two, and three injections, the classification accuracy falls to 92.0%, 85.0%, and 79.8%, respectively. Adding ABFT and duplex redundancy brings the classification accuracy back to 94.9% as in golden operations. Duplex shows more injections overall since faults may be injected each time it restarts execution. Duplex did not show any difference with the golden output matrix. Conversely, ABFT showed slight discrepancies with the golden output matrix with more than one injection, as expected by the chosen design of ABFT. However, these discrepancies did not manifest as increased mispredictions.

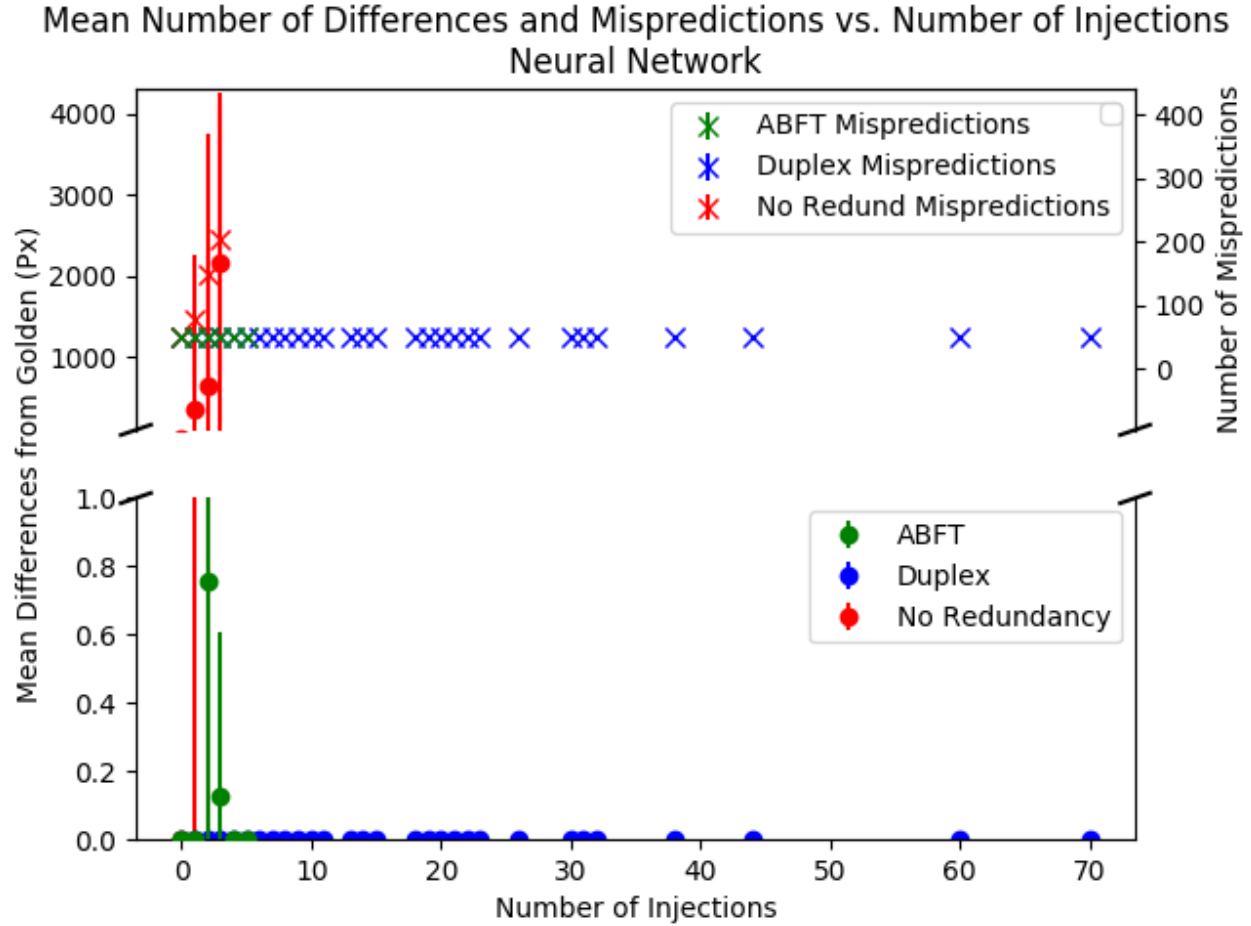


Figure 3: Neural network mean execution time vs. the number of injections.

Figure 3 shows the mean execution time for the neural network using all three methods. The runtime of the application with no redundancy technique stayed consistent around 3.04 seconds with a standard deviation on the order of 10^{-3} for any number of injections. Similarly, ABFT showed little variation in the runtime with an average around 6.62 seconds and a standard deviation on the order of 10^{-3} for any number of injections. The consistency in runtime is due to the ability to recover from errors without restarting executions. Meanwhile, duplex runtime showed a linear relationship with the number of injections performed. Therefore, while duplex showed the best reliability, it performed the worst in execution time since it needed to restart with every error.

2.2.2 Low-Level Fault Injection

Before irradiation, the reliability was examined with fault injection to compare the expected effects of data-error mitigation with ABFT. Errors seen in fault injection were categorized into “Injector Error” where the failure was internal to the fault injector, “Execution Error” where the fault manifested as a segmentation fault or system crash, and “Data Error” where the output did not match the expected golden output. This experiment focuses specifically on data errors.

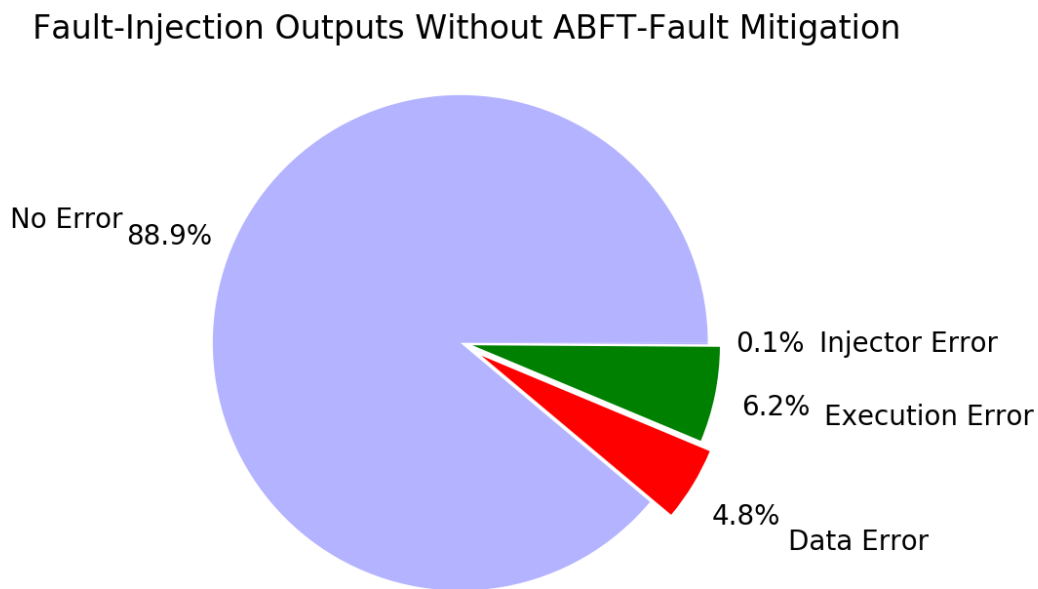


Figure 4: Fault-injection outputs from DrSEUs without any form of error mitigation.

Both forms were run and injected 1000 times. Without any mitigation, 4.8% of executions had outputs that differed from the expected matrix value giving a data error, as seen in Figure 4. The number of data errors are increased compared to what would be seen in a radiation experiment due to injecting specifically into the GPR. However, 6.2% of runs presented execution errors of the form of hangs or segmentation faults.

Fault-Injection Outputs With ABFT-Fault Mitigation

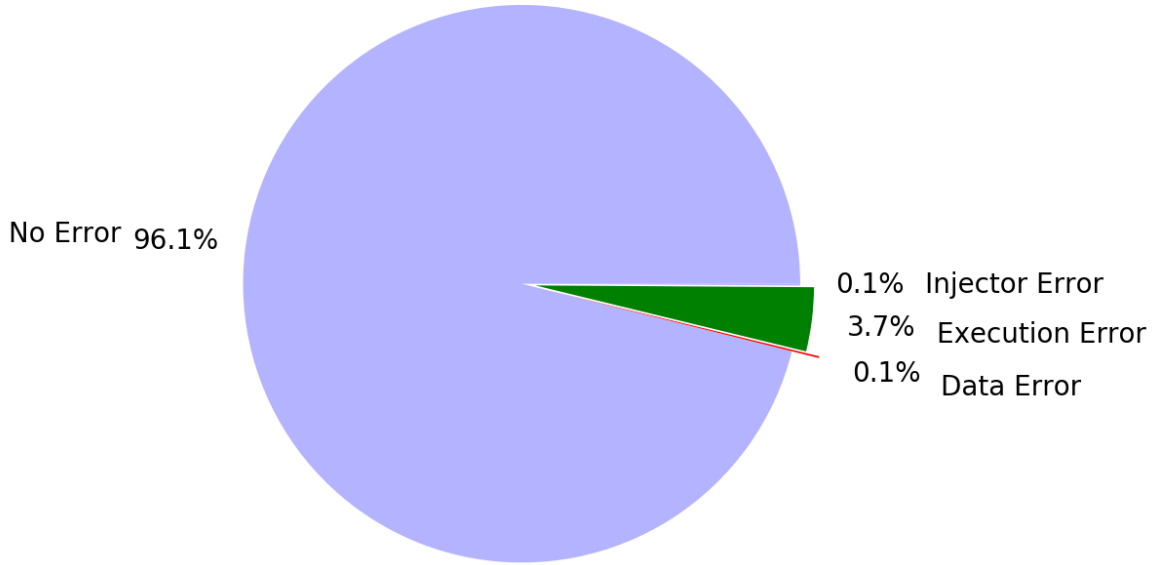


Figure 5: Fault-injection outputs from DrSEUs with ABFT.

Comparatively, only 0.1% of executions show data-output errors for the same number of injections, shown in Figure 5. Therefore, a $48\times$ improvement in reliability was observed using ABFT for data-error mitigation in fault injections. However, since these injections were targeted towards the general-purpose registers to force data errors instead of control errors, the improvement is inflated from what would be expected in irradiation. Any reduction in execution errors is an effect of having a lower sample size since ABFT has no bearing on control errors. As the number of runs for each injection experiment grows, the number of execution errors would approach roughly the same value.

Table 1: Summary of Fault-Injection Reliability and Runtime Results for 1000 Executions.

Design	Data Errors [%]	Runtime [s]
No Mitigation	4.8	15.19 ± 1.92
With ABFT	0.1	15.58 ± 1.39

A summary of the fault-injection results described above can be seen in Table 1. The two designs show no significant difference in runtime at about 15 seconds showing the minimal impact ABFT has on time overhead.

2.2.3 Irradiation

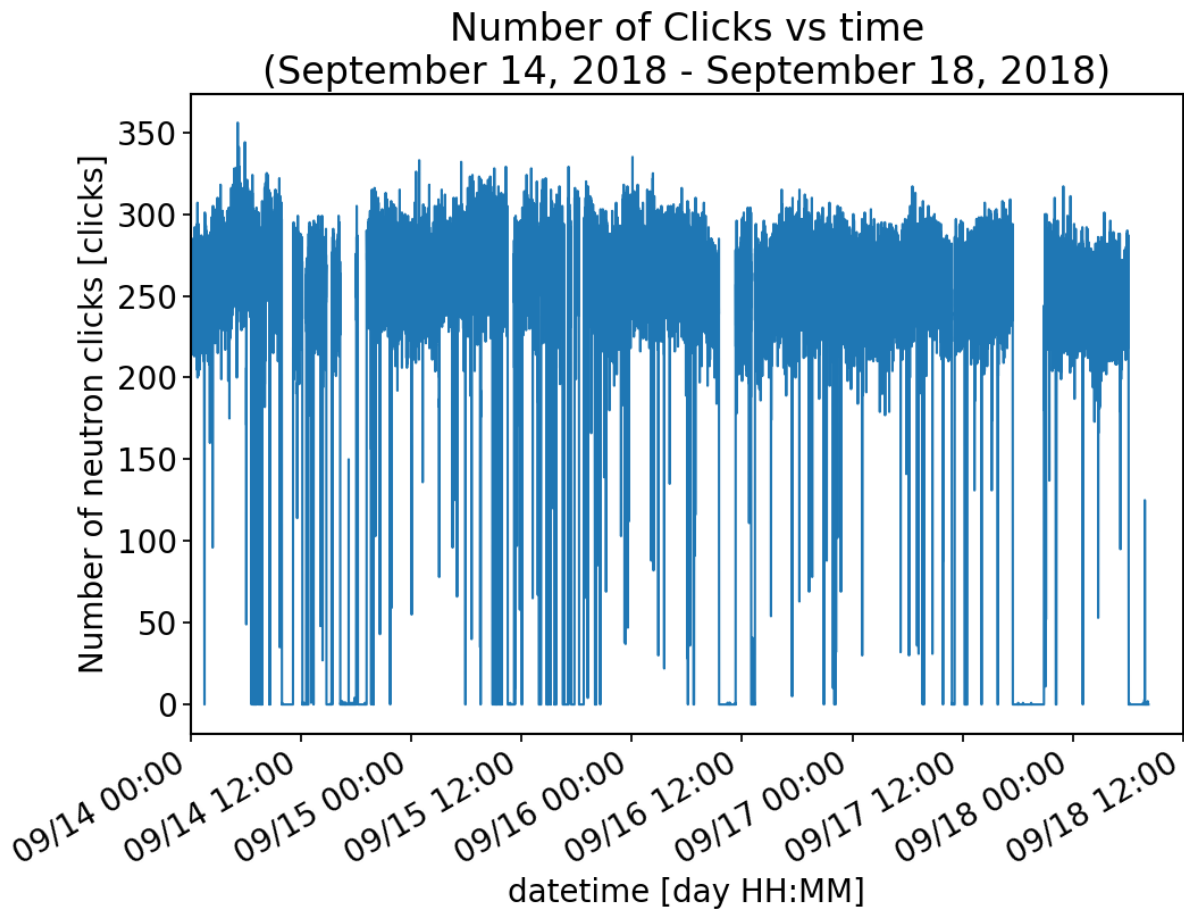


Figure 6: Neutron beam dosimetry over time over the days of irradiation: September 14, 2018 - September 17, 2018.

The measured dosimetry over the four days of experimentation provided by the dosimeter can be seen in Figure 6. Neutron beam dosimetry over the days of irradiation, September 14 to September 17, 2018, were represented by the number of clicks processed by the dosimeter. The number of clicks is multiplied by a constant for each day to represent an estimate of the number of neutrons of energies greater than 1 MeV and 10 MeV passing through the dosimeter.

This dosimetry gives a measurement of the number of neutrons passing through the dosimeter within a time interval of 10 seconds giving an estimate of the cross-section of neutrons passing through the device. The dosimetry was only considered when the devices under test were powered to calculate the cross-section

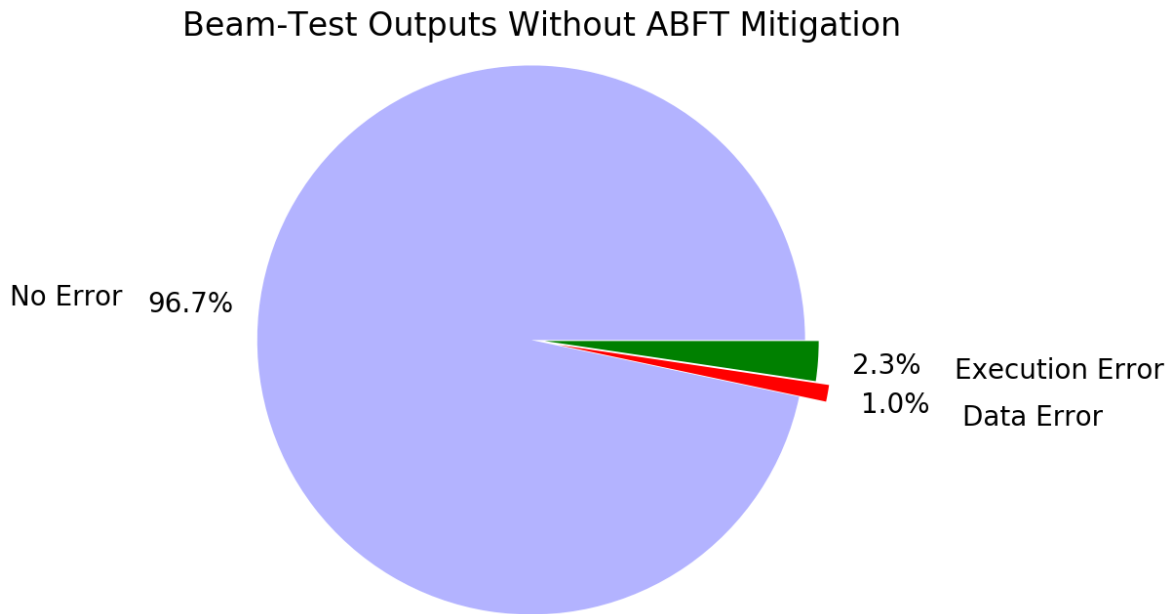


Figure 7: Results from four days of irradiation running a 500×500 matrix multiplication kernel without error mitigation.

The overall reliability in the output data was compared between the matrix multiplications employing SEU mitigation with ABFT and without under the wide-spectrum neutron radiation. Segmentation faults, system crashes, memory dumps etc. were classified as execution errors, and faults resulting in an out different from the expected was classified as a data error. Without any form of error mitigation, the matrix multiplication kernel show 1% of data errors over four days of irradiation. The ratio of unmitigated results can be seen in Figure 7.

Beam-Test Outputs With ABFT Mitigation

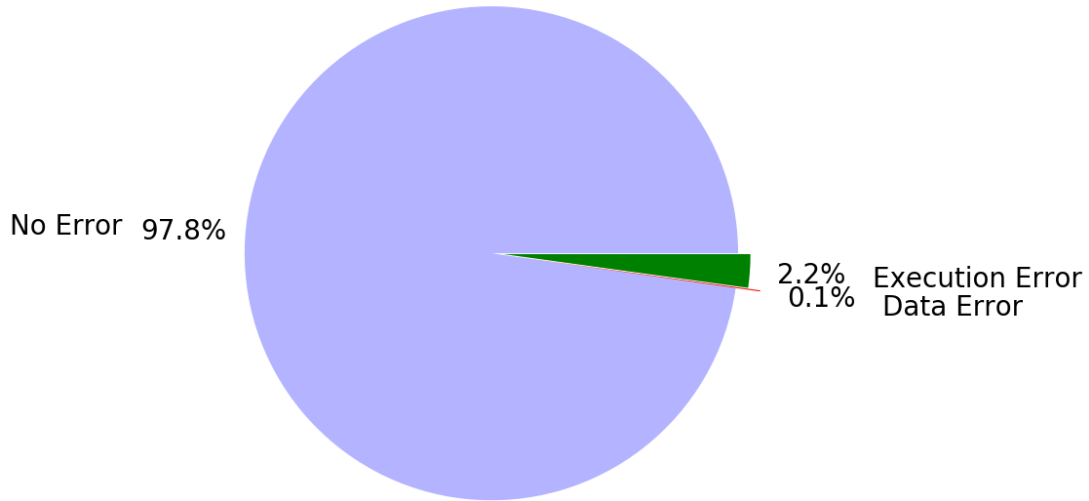


Figure 8: Results from four days of irradiation running a 500×500 matrix multiplication kernel while using ABFT for data-error mitigation.

Meanwhile, using ABFT for error mitigation shows only 0.1% data errors. Dividing this percentage by the number of data errors seen without mitigation gives a $10\times$ improvement in reliability. It can be seen that in both cases the number of execution errors remains the same at around 2.2% as expected. The beam test results with ABFT mitigation can be seen in Figure 8.

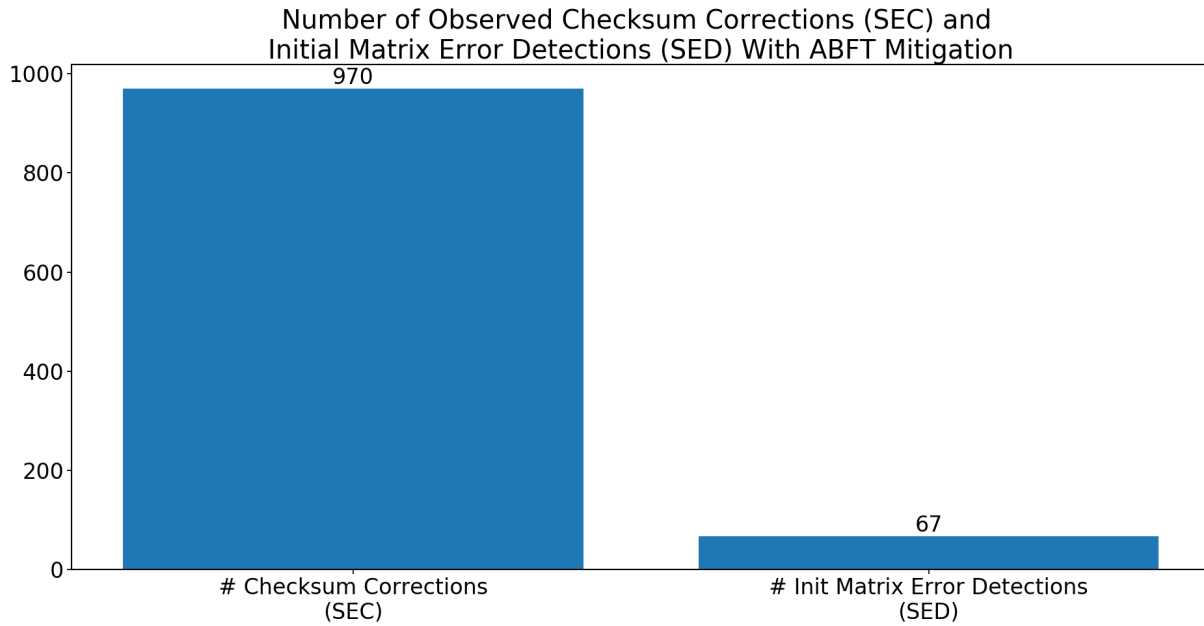


Figure 9: Number of errors corrected in the output matrix (SEC) and errors detected in the input matrices (SED).

To ensure that the reliability technique was, in fact, correcting errors, the number of errors caught and corrected in the output (SEC) were measured. Additionally, the number of errors detected in the input matrices (SED), restarting the calculation, was also measured. These values can be seen in Figure 9, showing a significant number of errors observed and corrected.

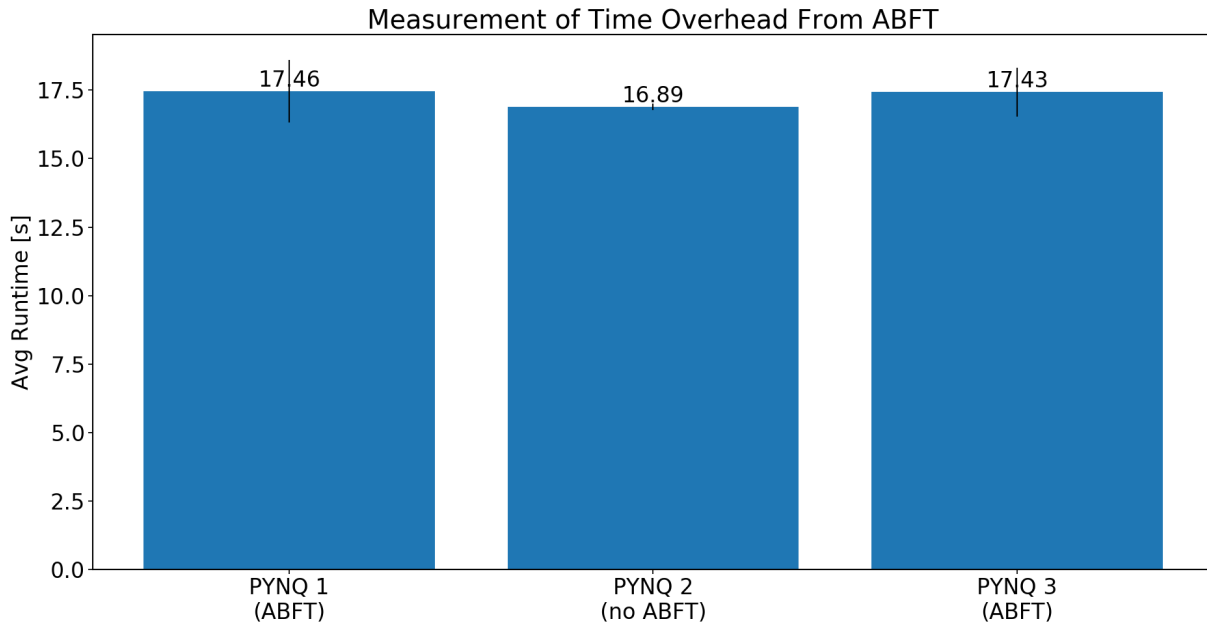


Figure 10: Average runtime of 500×500 matrix multiplication on the processor side of the Xilinx Zynq 7020 for both tested configurations.

In order to show overhead, the runtime of each execution was measured. Adding ABFT to the kernel did not result in a significant time increase. The average execution time increased from 16.89 seconds without mitigation to 17.43 seconds as seen in Figure 10, leading to a 3% increase in execution time.

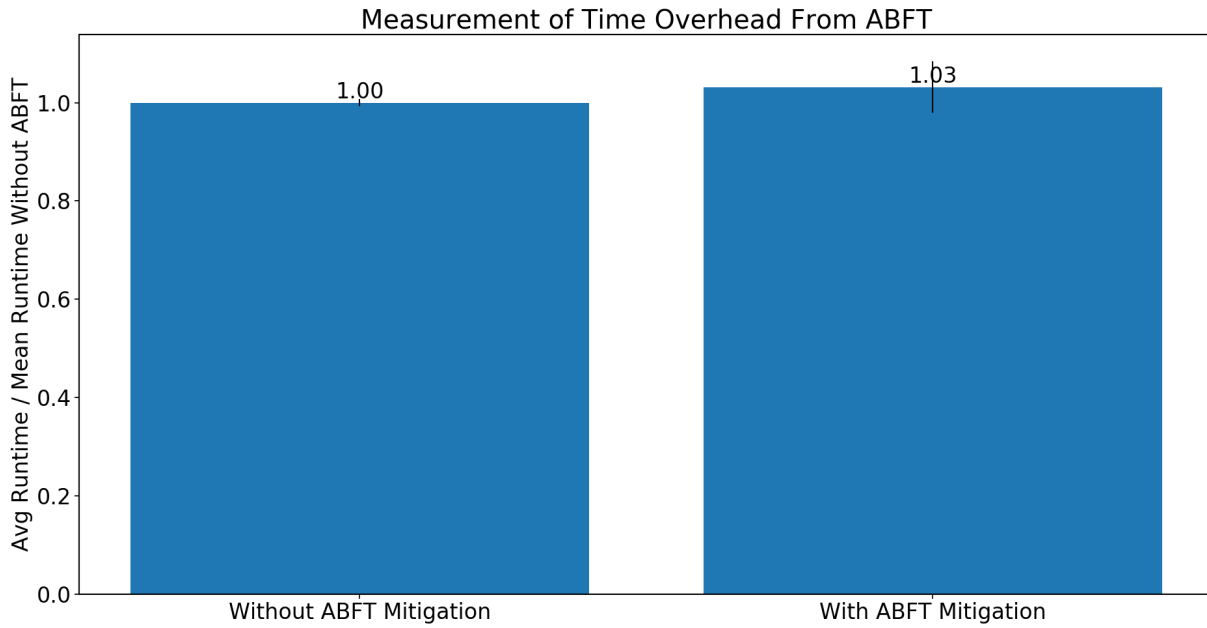


Figure 11: Execution times relative to the average runtime without any fault mitigation.

This increase in execution time is mostly due to the input matrices need to be reread if an error is detected in the inputs, and the calculation time for the checksum vectors. The relative increase in time can be seen in Figure 11.

Table 2: Summary of Beam-Test Reliability and Runtime Results Over Four Days of Irradiation.

Design	Data Errors [%]	Runtime [s]
No Mitigation	1.0	16.89 ± 0.11
With ABFT	0.1	17.43 ± 0.89

A summary of the beam test results can be seen in Table 2. These results agree with the results of fault injection and show a $10\times$ reduction in data errors with minimal time-overhead.

2.2.4 Statistical Significance

The cross-section was calculated by dividing the number of observed data errors by the calculated neutron fluence seen in Figure 6. To account for when the boards were powered off, only the effective fluence was calculated. Quantitative comparisons show a $5.33\times$ reduction in cross-section when ABFT is applied for fault-mitigation. In other words, there are about 5 times fewer errors per neutron that passes through the device-under-test. Therefore, adding ABFT as a fault-mitigation strategy will increase reliability in the data of a system by more than 5 times with a confidence level greater than 99% ($p < 10^{-5}$).

Similarly, the mean work to failure (MWTF) with fault mitigation was about 15.05 times that without any mitigation. This improvement shows that about 15 times more work can be done before a failure occurs than a system without any mitigation. Calculating a z-test between the two populations shows a confidence level of 99% with a p-value $< 10^{-5}$.

Table 3: Summary of Primary Comparisons Between Matrix Multiplication With ABFT for Fault Mitigation and Without Error Mitigation

Design	Eff. Fluence [n^0]	# Events	Exec. Errors	Data Errors	Cross-Section [cm^2]	Mean Work to Failure	95% Conf. Interval
No Mitigation	1.25×10^{11}	16649	389	163	1.308×10^{-9}	1668	$0.01 \pm 1.17 \times 10^{-5}$
With ABFT	4.49×10^{10}	16209	353	11	2.452×10^{-10}	25107	$0.001 \pm 3.82 \times 10^{-6}$
Improve					$5.33 \times$	$15.05 \times$	

The 95% confidence interval is calculated to be $0.01 \pm 1.17 \times 10^{-5}$ without mitigation and $0.001 \pm 3.82 \times 10^{-6}$ with ABFT. Since the true means do not overlap within the confidence interval, a statistically significant improvement in reliability is observed with high confidence. A summary of the comparison results can be seen in Table 3.

There was no significant difference in the number of execution errors between the two designs since execution errors were not addressed under the scope of this experiment. ABFT does not protect against segmentation faults, nor is it capable of detecting kernel errors. Therefore, the methodology presented in this thesis has no bearing on problems with the execution of the application under test or control of the system.

3.0 Conclusions and Future Work

The purpose of this experiment was to evaluate the use of algorithm-based fault tolerance to mitigate data errors in the matrix-multiplication kernels of machine-learning and computer-vision applications. The results show that ABFT is a viable solution to increase reliability on applications using a matrix-multiplication kernel.

The complexity of matrix multiplication has been studied exhaustively and show no improvement in runtime. Therefore, the only way of improving reliability of matrix multiplication applications is by adding data redundancy. Using ABFT to correct errors in the output matrix and detect errors in the input matrices show a reduction of data errors by $10\times$ under radiation. There were still a small amount of data errors in the mitigated design most likely due to upsets consisting of two or more bit-flips or damage to the board.

High-level fault injection was performed to compare ABFT's performance to duplex redundancy for matrix multiplication in a neural network. Both ABFT and duplex showed no increase in mispredictions as compared to the control with no redundancy, although ABFT did begin to exhibit matrix errors in the presence of multiple fault injections. The runtime for the ABFT enabled network was almost twice that of the base algorithm, but did not increase with an increasing number of faults. This increase in time overhead is inversely proportional to the size of the matrix multiplication kernel, as shown by the radiation experiment. On the other hand, duplex showed a linear increase in runtime with an increasing number of injections, culminating in a runtime that can be orders of magnitude larger than that of the control and the ABFT multiplications. As a result, ABFT demonstrates the best performance in terms of speed and accuracy in a radiation environment for a neural network application.

Fault-injection results show a more idealized effect of fault mitigation on the number of data-errors due the general-purpose registers being the target of injection. The fault-injection results agree with the beam-test results demonstrating consistency in the ability of ABFT to improve a system's reliability. Significant discrepancies in the fault-injection and radiation results are due to the randomized nature of SEUs in a radiation-beam experiment

and the ideal nature of fault injection into general-purpose registers. This lack of controlled variables in radiation testing causes control errors that were not considered in the scope of this experiment, reducing the overall percentage of observed data errors.

Irradiation results show a statistically significant improvement in reliability in terms of the cross-section and the MWTF with a negligible increase in runtime. Therefore, ABFT has been shown to improve the resilience of data in a radiative environment. Similarly, significantly more work can be done before an error is occurred, effectively increasing the number of applications that can be run before encountering a failure. This increase can thus help improve the autonomy of missions.

In both fault injection and irradiation, the time overhead of ABFT is shown to be minimal, averaging at about a 3% increase in execution time. This overhead is mostly due to calculation of the checksum vectors and when a problem was found in the input matrices with single-error detection. With the increase in reliability, ABFT is seen to have a significant improvement in dependability with minimal overhead.

Execution errors were shown to be more prevalent than data errors even without mitigation. However, due to the methodology ABFT is built on, execution errors were not addressed in the scope of this experiment. Data errors were specifically targeted for detection and correction in this experiment.

In general, data errors were not shown to be commonly occurring, accounting for less than 1% of all executions without mitigation. However, due to the number of decisions spacecraft can make using machine-learning or computer-vision applications, this small percentage can become non-negligible with time, especially if a single error can cause catastrophic failure. Therefore, employing ABFT as data redundancy has been shown to be a viable solution to significantly increase the amount of time and work done before failure while introducing minimal overhead.

Moving forward with error mitigation in complex applications requires two main steps. First, a new method of reducing errors in the execution of the application must be developed and addressed. ABFT is data specific, and does nothing to prevent control failures. With kernel protection, it is more likely that the application will complete. Then, combining that protection with ABFT, the system can guarantee with reasonable certainty that the application-under-test will produce not only a result, but an accurate one free from corruption.

The second step in future uses of ABFT is to test the reliability of commonly-used applications in spacecraft. Many computer-vision and machine-learning applications make use of the matrix multiplication kernel as their main computation. An example of one such application is the support-vector machine (SVM) used for image classification. Testing how ABFT affects the correctness of an SVM, or other classification algorithms would show ABFT's efficacy in ensuring accurate predictions and classifications in autonomous missions. If accurate predictions and classifications can be made, then proper and intelligent decisions can be assured in future spacecraft. Similarly, the benefit of using ABFT can be assessed when running applications parallelized with multiple cores. With the ability to run parallel processing reliably in radiation environment, the complexity of applications that missions are capable of increases even more.

Bibliography

- [1] Ahmad A Al-Yamani, Nahmsuk Oh, and Edward J McCluskey. Performance evaluation of checksum-based abft. In *Proceedings 2001 IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems*, pages 461–466. IEEE, 2001.
- [2] Jordan D Anderson, Jennings C Leavitt, and Michael J Wirthlin. Neutron radiation beam results for the xilinx ultrascale+ mpso. In *2018 IEEE Radiation Effects Data Workshop (REDW)*, pages 1–7. IEEE, 2018.
- [3] Cynthia J. Anfinson and Franklin T. Luk. A linear algebraic model of algorithm-based fault tolerance. *IEEE Transactions on Computers*, 37(12):1599–1604, 1988.
- [4] GD Badhwar and PM O’Neill. An improved model of galactic cosmic radiation for space exploration missions. *International Journal of Radiation Applications and Instrumentation. Part D. Nuclear Tracks and Radiation Measurements*, 20(3):403–410, 1992.
- [5] Jon A Benediktsson, Philip H Swain, and Okan K Ersoy. Neural network approaches versus statistical methods in classification of multisource remote sensing data. 1990.
- [6] Edward Carlisle IV and Alan George. Dynamic robust single-event upset simulator. *Journal of Aerospace Information Systems*, 15(5):282–296, 2018.
- [7] Zizhong Chen. Online-abft: An online algorithm based fault tolerance scheme for soft error detection in iterative methods. In *ACM SIGPLAN Notices*, volume 48, pages 167–176. ACM, 2013.
- [8] Peng Du, Aurelien Bouteiller, George Bosilca, Thomas Herault, and Jack Dongarra. Algorithm-based fault tolerance for dense matrix factorizations. *Acm sigplan notices*, 47(8):225–234, 2012.
- [9] Robert Granat, Kiri L Wagstaff, Benjamin Bornstein, Benyang Tang, and Michael Turmon. Simulating and detecting radiation-induced errors for onboard machine learning. In *2009 Third IEEE International Conference on Space Mission Challenges for Information Technology*, pages 125–131. IEEE, 2009.

- [10] Chi-Chang J Ho and N Harris McClamroch. Automatic spacecraft docking using computer vision-based guidance and control techniques. *Journal of Guidance, Control, and Dynamics*, 16(2):281–288, 1993.
- [11] Kuang-Hua Huang and Jacob A Abraham. Algorithm-based fault tolerance for matrix operations. *IEEE transactions on computers*, 100(6):518–528, 1984.
- [12] Israel Koren and C Mani Krishna. *Fault-tolerant systems*. Elsevier, 2010.
- [13] Franklin T Luk and Haesun Park. An analysis of algorithm-based fault tolerance techniques. *Journal of Parallel and Distributed Computing*, 5(2):172–184, 1988.
- [14] Joseph R Srour and James M McGarrity. Radiation effects on microelectronics in space. *Proceedings of the IEEE*, 76(11):1443–1469, 1988.
- [15] Andrew James Stothers. On the complexity of matrix multiplication. 2010.
- [16] Christopher Wilson, Jacob Stewart, Patrick Gauvin, James MacKinnon, James Coole, Jonathan Urriste, Alan George, Gary Crum, Elizabeth Timmons, Jaclyn Beck, et al. Csp hybrid space computing for stp-h5/isem on iss. 2015.
- [17] Haissam Ziade, Rafic A Ayoubi, Raoul Velazco, et al. A survey on fault injection techniques. *Int. Arab J. Inf. Technol.*, 1(2):171–186, 2004.