García-Astudillo, L., Lindoso, A., Entrena, L., Martín, H., & García-Valderas, M. (2021). Error sensitivity study of FFT architectures implemented in FPGA. *Microelectronics Reliability*, 126, 114298.

DOI: 10.1016/j.microrel.2021.114298

Error sensitivity study of FFT architectures implemented in FPGA

L. A. García-Astudillo, A. Lindoso, L. Entrena, H. Martín, M. García-Valderas

Electronic Technology Department, University Carlos III de Madrid
Avda. Universidad, 30, 28911 Leganés (Madrid), Spain

Abstract

This work studies the impact that the architectural choices can have in the error mitigation of a digital processing module such as the Fast Fourier Transform. To this purpose, several serial and pipelined architectures were implemented in FPGA using Block Triple Modular Redundancy and analysed under a fault injection approach that was previously validated with radiation. These architectures were compared with respect to error rate, common mode failure rate and signal-to-noise ratio. Experimental results show that the error rate is strongly correlated with the use of resources when using a similar architecture. However, pipelined architectures tend to have more common mode failures but with lower signal-to-noise ratio than a serial architecture.

1. Introduction

FPGAs (Field-Programmable Gate Arrays) are widely used today for Digital Signal Processing (DSP) applications. Since the introduction of embedded multiplier-accumulator blocks, usually called DSP blocks, FPGAs can provide very high performance for DSP applications, exceeding that of state-of-the-art microprocessors and digital signal processors. Furthermore, the main advantage of FPGAs is the ability to tailor the implementation to match system requirements. FPGA inherent parallelism can be used to boost performance, while serial architectures can be used to save resources in low rate applications. As a matter of fact, DSP is a major application field for FPGAs.

FPGA-implemented DSP designs are increasingly used in safety and mission-critical applications, such as automotive, avionics and space. In this case, soft errors are an increasing concern and fault-tolerant designs must be used to mitigate them. For space, a few types of radiation-hardened FPGAs are available, but they are expensive and lag way behind their commercial counterparts. Therefore, designers are considering the use of COTS (Commercial Off-The-Shelf) FPGAs and mitigate errors by design. A typical solution is TMR (Triple Modular Redundancy) [1] combined with scrubbing to prevent error accumulation in the configuration memory. This solution is quite effective [2]. Nevertheless, it cannot mask all errors, because some configuration bits can cause Common Mode Failures (CMFs) [3].

In this work, we perform a study of soft error mitigation for FFT (Fast Fourier Transform) architectures. The FFT is a prime example of a DSP algorithm and it is widely used in a large variety of applications. Because it is a very important and complex algorithm, many architectures have been proposed to optimize performance and resource usage. The goal of this work is to evaluate some of the most representative architectures with respect to the reliability they can provide when implemented in an FPGA using TMR, and determine the impact that the choice of architecture has on the soft error resilience. To this purpose, we compare several parameters such as the error rate, the CMF rate and the Peak-Signal-to-Noise Ratio (PSNR).

In order to be able to compare a significant amount of different designs, the evaluation has been carried out by extensive fault injection campaigns. Nevertheless, radiation test results obtained with the same experimental setup show a very good correlation between radiation results and fault injection results with our approach. The results of this work demonstrate that although the error rate is primarily determined by the size of the design, the choice of architecture has a significant impact on the amount of unmitigated errors (CMFs) and the relevance of the errors. In particular, pipelined architectures tend to have more CMFs but with higher PSNR than a serial architecture.

* Corresponding author. luisagar@ing.uc3m.es
Tel: +34 916 24 5979; Fax: +34 916 24 9430

The remaining of the paper is as follows. Section 2 summarizes related work. Section 3 describes the FFT architectures under study. Section 4 presents and discusses the experimental results. Finally, section 5 shows the conclusions of this work.

2. Related work

The problem of designing fault-tolerant circuits for DSP, and for FFT networks in particular, has traditionally received significant attention [4]-[12]. Early work focused on weighted checksum codes [4], [5], Concurrent Error Detection (CED) [6], [7], [8], [9] and time-redundant networks using redundant processing elements [10], [11], [12]. Some works proposed ad-hoc architectures that take advantage of the properties of the FFT algorithm, such as Algorithm-Based Fault Tolerance (ABFT) [13]. These ideas have been further developed in more recent works [14].

For the early approaches, the target technology was mainly ASIC (Application Specific Integrated Circuit) because FPGAs were hardly large enough to implement these complex algorithms. Performance and area overheads were of the utmost importance, and hence the solutions tried to provide fault-tolerance with much lower overhead than TMR. Moreover, these approaches were usually validated using simplified error models or theoretical estimations of error bounds. These models may not be appropriate for SRAM-based FPGAs, which have fine-grain configurable architectures that are affected by configuration memory errors.

Today, FFT designs for real-time applications are often implemented in FPGAs, taking advantage of the inherent parallelism of these devices. In this case, TMR can be a convenient solution that does not enforce a particular FFT implementation architecture. However, TMR is not fully effective because there are single configuration bits that can cause multiple errors across domains [3], usually known as Common Mode Failures [15]. For instance, a single configuration bit error can create a short or a double open, that may affect two of the three TMR domains or the voter circuit. F. L. Kastensmidt et al. [16] analyse the optimal placement of voters using a DSP case study to reduce CMFs. Another source of CMFs are Multiple Cell Upsets (MCUs), which are multiple bit errors caused by the strike of a single particle. An MCU can create a CMF if the multiple upsets are not in the same TMR domain. Nevertheless, although MCUs are becoming relevant for advanced technologies, they are still much less frequent than SEUs [15]. Thus, SEUs are still the cause of most CMFs in SRAM-based FPGAs [17].

3. FFT architectures under study

FFT architectures can be divided in two main groups: serial and pipelined. They can be selected according to the application requirements. Serial architectures offer resource efficiency, while pipelined architectures aim at higher throughput. In particular, pipelined architectures can support the continuous processing of input data which is needed in real-time processing applications.

3.1. Serial FFT

Serial architectures perform the required mathematical operations in an iterative way over the same circuit, storing the intermediate results in banks of memory. Internally, the circuit can be designed in various ways, of which the most popular are the Radix-2 and Radix-4 algorithms. They differ in how the input data are split in groups and passed to the *butterfly unit* (BU) that performs complex additions. Butterflies in Radix-2 perform 2 complex additions, while those in Radix-4 execute 4 additions. After the butterflies, the data are multiplied by the *twiddle factors*, which are complex constant numbers related to the length of the transform. The Radix-4 algorithm is considered more efficient, as it divides the number of iterations by 2 with respect to the Radix-2 algorithm. In this work, we will study the performance of a Radix-4 serial circuit, implemented using Xilinx' configurable FFT IP.

3.2. Pipelined FFT

Pipelined versions of the FFT algorithm are based on chains of *stages*, comprising Butterfly Units (BUs) and complex multipliers. The output of each stage feeds the next stage until all input data have been processed. To feed the BUs with the correct data, it is necessary to add data delays between stages. The Single- and Multi-path Delay Commutator (SDC and MDC, respectively) use memory blocks and multiplexors to create these buffers [18]. The Single-path Delay Feedback (SDF) approach [19] introduces FIFO memories at the output of the BUs and feeds their contents back to the BUs when necessary using control signals. SDF designs are generally more efficient in the usage of memory and we selected this approach for our designs.

As in the case of the serial FFT, various radix algorithms can be used. In this work we explored the features of Radix-2, Radix-4 and Radix-2^2. The Radix-2 architecture (R2SDF), depicted in Fig. 1.a, is the simplest of the three. It encompasses log2(N) stages, where N is the transform length. Each stage

has a 2-input BU, a FIFO memory to implement data delays and a complex multiplier.

Fig. 1.b represents a Radix-4 architecture (R4SDF). In this case, log4(N) stages are needed. The stages enclose a 4-input BU, 3 FIFO memories and one complex multiplier. The FIFOs are all of the same size within the stage, but their length is reduced as the data advances through the pipeline.

The Radix-2^2 ($R2^2SDF$) architecture, shown in Fig. 1.c, also contains log4(N) stages. However, each stage contains two slightly different BUs, a complex multiplier and two FIFO memories, whose depth depends on the order of the stage. In the first stage the FIFOs are N/2 and N/4 long, and then they are halved in each stage. The Radix-2^2 approach combines features of the Radix-2 and Radix-4 architectures, and it is more efficient in the usage of resources than both of them.

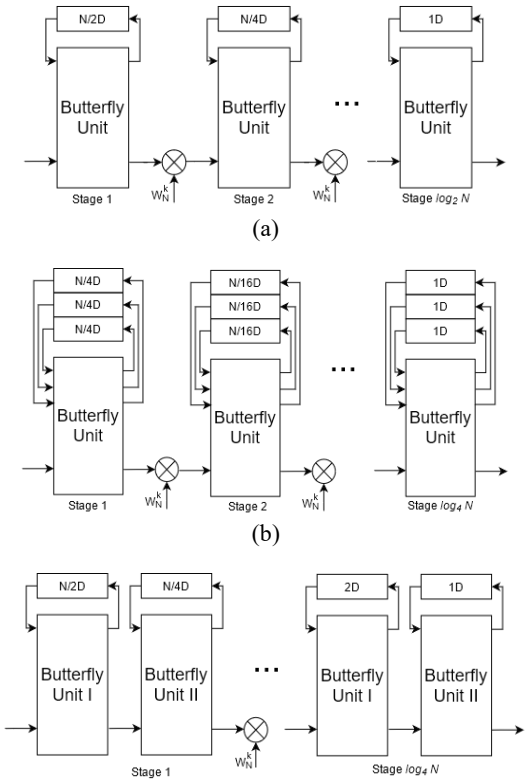


Fig. 1. Pipelined FFT architectures: a) R2SDF, b) R4SDF, and c) $R2^2SDF$

4. Experimental results

To assess the performance of the proposed architectures, we developed 64 and 256 points versions of the FFTs and implemented fault-tolerant versions using a Block TMR approach. Each of them was wrapped on a testbench circuit capable of providing inputs for the FFTs, comparing the outputs against the expected results and collecting error data for off-line analysis. Upon error, the results from each instance of the FFT and their voted output are sent through a UART connection to the external host that controls the experiments.

The programmable logic of a Xilinx Zynq-7010 All Programmable SoC (APSoC) was used in these experiments to implement the designs. The dual core ARM processor included in this device was not used in the tests. We used a default effort in the synthesis and implementation of all designs, with a common clock constraint. We have just used constraints to ensure redundant FFT instances were not optimized away.

Fault injection was implemented using Xilinx Soft Error Mitigation (SEM) IP [20]. This IP can be used to detect or correct errors in the configuration memory, and also to inject faults. Faults were injected randomly at a sufficiently slow rate to ensure that a complete FFT can be computed between faults. This setup can also be used for radiation test experiments, by simply disabling fault injection. We performed one radiation test experiment at Centro Nacional de Aceleradores (CNA) in Seville with low energy protons (up to 15 MeV) using the 64-point serial TMR design and the results matched very well with fault injection. Specifically, the ratio of CMFs to the total errors was 1.29% under radiation and 1.04% under fault injection. However, we used fault injection for the analysis because the amount of beam time required to test all the versions would have been prohibitive.

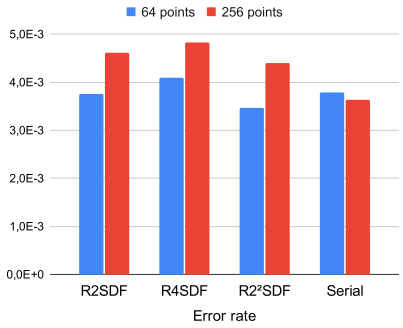
Table 1 reports the hardware utilized by each architecture for the 64-point and 256-point FFT versions. It is evident from the data in the table that the Radix-2^2 architecture is the most efficient of the pipelined versions regardless the transform length, while Radix-2 and Radix-4 are similar in terms of total area used. The Serial Radix-4 consumes far more resources in the 64-point FFT, but it is with longer FFTs where this architecture shows its efficiency. The addition of more stages in the pipeline greatly increases the resources necessary in the R2, R4 and $R2^2$ architectures, whereas the serial FFT is barely affected by the change.

Table 2 summarizes the fault injection results. The first row shows the number of injections performed in each design. The next two rows show the total amount of erroneous FFT calculations (faulty frames) and the error rate, respectively. The error rate is calculated as the ratio of faulty frames to the number of injected faults. We define a word as the real or imaginary part of each complex point in an FFT frame. This way, a frame of length N, with N being 64

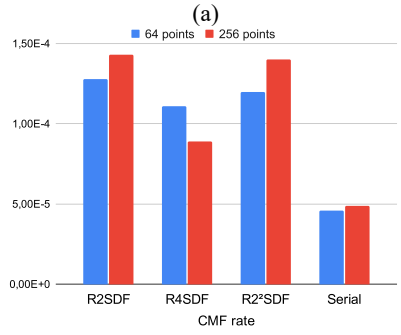
Table 1
Resource utilization of the different architectures.

	R2SDF FFT		R4SDF FFT		R2²SDF FFT		Serial R4 FFT	
Points	64	256	64	256	64	256	64	256
Stages	6	8	3	4	3	4	1	1
LUT as logic	1065	1627	1231	1939	971	1445	1072	1106
LUT RAM	218	802	236	821	221	805	181	188
FF	1069	1437	705	930	723	930	2134	2227
BRAM	0	0.5	0	0	0	0	3.5	3.5
DSP Slices	15	21	6	9	6	9	9	9

or 256 points in this study, would have 2N words. The results of the three FFTs and the voted output are collected when an error occurs, and the words of each frame can be later compared with correct results. This way, the comparison can establish whether errors affected one FFT, several FFTs or the voting logic, and we can classify the errors according to this information. In case an error affects a word in just one FFT, the voter mitigates the error and it will be classified as a masked error. On the contrary, if the fault affects more than one of the FFTs, the triplicated voters, or the voting logic and at least one of the FFT instances, the voter is not capable of correcting the fault and the error will be classified as an unmitigated error or Common Mode Failure (CMF). CMFs are reported in the fourth row of Table 2. Finally, the last row in Table 2 reports the CMF rate, which is calculated as the ratio of CMF frames to the number of injected faults. The error rate and the CMF rate of the designs are shown in Fig. 2 for comparison.



(a)



(b)

Fig. 2. Error rates (a) and CMF rates (b) of the tested architectures.

By carefully analysing the error rates in Fig. 2a, we can clearly see that all the 64-point designs behave in a similar way, but when the FFT is enlarged to 256 points, only the pipelined designs increase their error rate. This suggests a strong correlation of the error rate with the resources used by the architecture, as shown in Fig. 3.

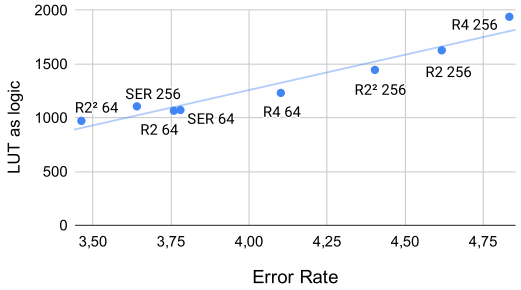


Fig. 3. LUT as logic utilization vs. error rate.

A similar correlation can be established between the CMF rate and the resources. These two correlations seem to suggest that a higher utilization of resources leads to a higher incidence of faults, which, in turn, increases the appearance of CMFs. However, as seen in Fig. 2b, the CMF rate is between 2 and 3 times smaller in the serial case than in the pipelined designs, even in cases where the difference in area favours the pipeline architecture, such as the 64-point R2²SDF architecture. Thus, the architecture has a significant impact on the criticality of the errors. This is also noticeable when comparing the pipeline designs: although the R4 architecture is significantly larger than the others, its CMF rate is lower than the CMF rate of R2 and R2² architectures.

A more detailed analysis is shown in Table 3, which reports the total amount of faulty words in the faulty frames. What stands out in Table 3 is a lower incidence of erroneous words in pipelined designs than in the serial architecture, which is even more evident when the transform length is increased to 256 points. Contrary to expectations, the data obtained suggest that a longer pipeline has a lower percentage of its outputs affected by faults. This result may be explained by the separation of computations in stages and the truncation performed between the stages to

Table 2
Results of the fault injection campaigns.

	R2SDF FFT TMR		R4SDF FFT TMR		R2²SDF FFT TMR		Serial R4 FFT TMR	
Points	64	256	64	256	64	256	64	256
Number of injections	216000	357120	216000	367200	216000	422640	216000	388800
Faulty frames (total)	812	1649	886	1775	748	1861	816	1416
Error Rate (10^{-3})	3.76	4.61	4.09	4.83	3.46	4.40	3.78	3.64
CMF frames	30	51	24	33	26	48	10	19
CMF Rate (10^{-4})	1.28	1.43	1.11	0.89	1.20	1.14	0.46	0.49

Table 3
Error classification of the reported faults.

	R2SDF FFT TMR		R4SDF FFT TMR		R2²SDF FFT TMR		Serial R4 FFT TMR	
Points	64	256	64	256	64	256	64	256
Total words in faulty frames	103936	844288	113408	908800	95744	952832	104448	724992
Faulty words (real or imaginary)	39751 (38.2%)	267648 (31.7%)	34444 (30.37%)	220484 (24.3%)	33572 (35.1%)	305808 (32.1%)	48854 (46.7%)	350785 (48.4%)
Masked errors (%)	92.6	93.6	92.9	95.8	92.2	94.6	98.1	98.2
Words with CMFs (%)	7.4	6.4	7.1	4.2	7.8	5.4	1.9	1.8
PSNR (dB)	70.1	77.3	78.9	82.1	72.6	78.8	69.5	72.9

avoid overflow. An error affecting the least significant bits would be mitigated by this truncation. Thus, the more stages are in the pipeline, the less likely error propagation is.

This effect has another positive consequence: the percentage of words affected by CMFs is also lowered with longer pipelined FFTs. However, the serial implementation does not exhibit this kind of behaviour. Because all the stages are performed by the same hardware, a configuration memory error in the butterfly or the multiplier would affect the result of every stage, while in a pipeline, only one result would be corrupted, and chances are higher that it may be mitigated in the following stages.

A graphical representation of these two phenomena is provided in Fig. 4. This figure shows the error classification of all the words in the faulty frames found, expressed as a percentage of the total. The percentage of correct words is significantly higher in longer pipeline FFTs, and the percentage of words with CMFs is also reduced.

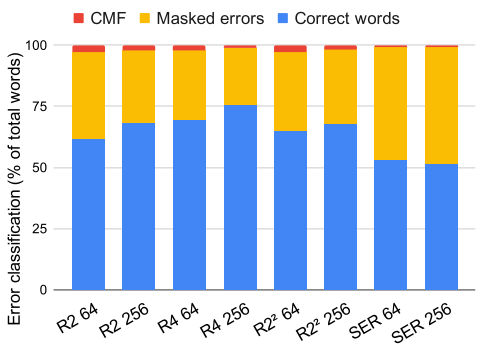


Fig. 4. Error classification of the words in faulty frames.

The last row of Table 3 sheds additional light over this finding. This row presents the average Peak Signal-To-Noise Ratio (PSNR) of all the faulty frames, a measure of the distortion of a signal due to noise caused by errors. Eq. 1 describes the method to compute the PSNR of a faulty frame, where *MAX* is the highest possible value of the signal and the *MSE* is the Mean Squared Error of the FFT frame, calculated as the sum of the squared error of each point in the frame divided by the number of points in the frame.

$$PSNR = 10 \cdot log_{10}\left(\frac{MAX_I^2}{MSE}\right) \qquad (1)$$

From the data in Table 3, we can see that the PSNR improves significantly in the longer pipelined designs. Not only the number of total errors is decreased due to truncation, but also the importance of the errors is lowered. All pipeline designs have a better performance in terms of PSNR than the serial architecture. Although seemingly contradictory with the previous results presented, this finding is, in fact, very interesting. The experiments hint that pipelined FFT designs are more sensitive to radiation effects due to their larger area overhead, but at the same time, they boast of an intrinsic error mitigation mechanism that allows them to eliminate small errors and lower the impact of more significant errors.

A comparison between the pipeline designs can also be carried out in these terms. As in the case of the CMF rate, the R4 architecture seems to perform better, while the R2 and R2² architectures are similar. The percentage of correct words per frame in the R4

architecture is the highest of the three pipeline designs and, accordingly, the PSNR of the R4 architecture is also outstanding.

5. Conclusions

In this work we performed a comparison of the error sensitivity of several FFT architectures implemented in a SRAM-based FPGA using fault injection. Our results showed a clear relationship between the usage of resources and the sensitivity to Single Event Upsets, giving an evident advantage to serial architectures, that only increase their size slightly when computing larger FFTs, whereas the addition of more stages in the pipeline versions imposes a penalty on their sensitivity. However, we also showed that the architecture has a significant impact on the criticality of the errors. Our data suggest that the incidence of critical errors (Common-Mode Failures) in serial implementations is much lower than in pipeline architectures, even in cases where the pipeline architecture has a smaller area. There are also significant differences among the pipeline architectures that do not correlate with the size of the implementation.

Nevertheless, regarding the quality of the results, the reutilization of resources in a serial architecture leads to a higher number of erroneous data points in the FFT frame and noisier results due to error accumulation in the stages.

Acknowledgements

This work has been supported in part by the Spanish Ministry of Science and Innovation under project PID2019-106455GB-C21 and by the Community of Madrid under project no. 49.520608.9.18.

References

[1] M. Berg, "Single Event Effects in FPGA Devices 2014-2015", NASA Electronic Parts and Packaging Program (NEPP) Electr. Tech. Workshop (ETW), June 2015.

[2] K. S. Morgan, D. L. McMurtrey, B. H. Pratt, and M. J. Wirthlin, “A comparison of TMR with alternative fault-tolerant design techniques for FPGAs”, IEEE Transactions on Nuclear Science, vol. 54, no. 6, pp. 2065–2072, Dec. 2007.

[3] H. Quinn et al. “Domain crossing errors: Limitations on single device triple-modular redundancy circuits in Xilinx FPGAs”, IEEE Transactions on Nuclear Science, vol. 54, no. 6, pp. 2037–2043, Dec. 2007.

[4] J.-Y. Jou and J. A. Abraham, "Fault-tolerant matrix arithmetic and signal processing on highly concurrent computing structures", Proceedings of the IEEE, vol. 74, no. 5, pp. 732-741, May 1986.

[5] J.-Y. Jou and J. A. Abraham, "Fault-tolerant FFT networks", IEEE Transactions on Computers, vol. 37, no. 5, pp. 548-561, 1988.

[6] D. L. Tao and C. R. P. Hartmann, "A novel concurrent error detection scheme for FFT networks", IEEE Transactions on Parallel and Distributed Systems, vol. 4, no. 2, pp. 198-221, 1993.

[7] C. G. Oh and H. Y. Youn, "On concurrent error location and correction of FFT networks", IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 2, no. 2, pp. 257-260, June 1994.

[8] F. Lombardi and J. C. Muzio, "Concurrent error detection and fault location in an FIT architecture", IEEE J. Solid- State Circuits, vol. 27, no. 5, pp. 728-736, 1992.

[9] J.-F. Li, S.-K. Lu, S.-A. Hwang and C.-W. Wu, "Easily testable and fault-tolerant FFT butterfly networks", IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing, vol. 47, no. 9, pp. 919-929, Sept. 2000,

[10] A. Antola, R. Negrini, M. G. Sami and N. Scarabottolo, "Fault-tolerance in FFT arrays: time-redundancy approaches", IEEE Int. Conf. on Communications, vol.3, pp. 779-785, 1990.

[11] Y.-M. Hsu and E. E. Swartzlander, "FFT arrays with built-in error correction", Proc. 28th Asilomar Conf. on Signals, Systems and Computers, vol.1, pp. 172-176, 1994.

[12] T.-H. Chen and L.-G. Chen, "Concurrent error-detectable butterfly chip for real-time FFT processing through time redundancy", IEEE J. Solid-state Circuits, vol. 28, no. 5, pp. 537-547, 1993.

[13] K.-H. Huang and J. A. Abraham, "Algorithm-Based Fault Tolerance for Matrix Operations", IEEE Transactions on Computers, vol. C-33, no. 6, pp. 518-528, June 1984.

[14] P. Reviriego, C. Bleakley, J. A. Maestro and A. O'Donnell, "Offset DMR: A Low Overhead Soft Error Detection and Correction Technique for Transform-Based Convolution", IEEE Transactions on Computers, vol. 60, no. 10, pp. 1511-1516, Oct. 2011.

[15] M. Wirthlin, D. Lee, G. Swift, and H. Quinn, “A method and case study on identifying physically adjacent multiple-cell upsets using 28-nm, interleaved and SECDED-protected arrays”, IEEE Transactions on Nuclear Science, vol. 61, no. 6, pp. 3080–3087, Dec. 2014.

[16] F. L. Kastensmidt, L. Sterpone, L. Carro and M. Sonza Reorda, “On the Optimal Design of Triple Modular Redundancy Logic for SRAM-Based FPGAs”, Proc. Design, Automation and Test in Europe, vol. 2, pp. 1290–1295, 2005.

[17] M. J. Cannon, A. M. Keller, H. C. Rowberry, C. A. Thurlow, A. Pérez-Celis and M. J. Wirthlin, "Strategies for Removing Common Mode Failures From TMR Designs Deployed on SRAM FPGAs", IEEE Transactions on Nuclear Science, vol. 66, no. 1, pp. 207-215, Jan. 2019.

[18] S. He and M. Torkelson, "Designing pipeline FFT processor for OFDM (de)modulation", Proc. URSI Int. Symp. on Signals, Systems, and Electronics, pp. 257-262, 1998.

[19] E.H. Wold and A.M. Despain. “Pipeline and parallel-pipeline FFT processors for VLSI implementation”. IEEE Trans. Comput., C-33(5), pp. 414-426, May 1984.
[20] “Soft error mitigation controller v4.1 Product guide”, Xilinx Inc., White Paper PG036, Nov. 2014.