Embedded Fault-Tolerant Accelerator Architecture for SAR Backprojection

Helena Cruz [*] and Rui Policarpo Duarte[†] and Horacio Neto[‡]
INESC-ID/Instituto Superior Tecnico, Lisbon, 1000-049 Lisbon, Portugal

This paper presents an optimized multi-core embedded architecture to speedup and improve the resilience of the execution of a Backprojection Algorithm targeting on-board SAR imaging systems for the space environment. The proposed architecture produces SAR images by computing approximations instead of the full precision of some arithmetic operations. The proposed solution was implemented on a Xilinx SoC device with a dual-core processor. The novel fault-tolerant system exhibits graceful degradation characteristics as it is able to produce images considered acceptable, i.e. SNR > 100dB and on average with an SNR 0,65dB less than the fault-free image, while providing a time reduction up to 33%, and a decrease up to 40% in energy consumption.

Nomenclature

$g(n)$	=	Wave sample in the previous adjacent range bin.
$g(n+1)$	=	Wave sample in the following adjacent range bin.
$r(n)$	=	Corresponding range to the previous adjacent bin.
$r(n+1)$	=	Corresponding range to the following adjacent bin.
r_k	=	Range from pixel $f(x, y)$ to aperture point θ_k.
dr	=	Differential range from platform to each pixel versus center of swath.
$f(x, y)$	=	Value of each pixel (x, y).
$g_{x,y}(r_k, \theta_k)$	=	Wave reflection received at r_k at θ_k
r_c	=	Range to center of the swath from radar platform.
r_k	=	Range from pixel $f(x, y)$ to aperture point θ_k.
θ_k	=	Aperture point.
ω	=	Minimal angular velocity of wave.
x_k, y_k, z_k	=	Radar platform location in Cartesian coordinates.
x, y, z	=	Pixel location in Cartesian coordinates.

[*]PhD candidate, Department of Computer Science and Engineering, INESC-ID/Instituto Superior Tecnico, Av. Rovisco Pais, 1, Portugal
[†]Lecturer, Department of Computer Science and Engineering, INESC-ID/Instituto Superior Tecnico, Av. Rovisco Pais, 1, Portugal
[‡]Senior Lecturer, Department of Electrical and Electronic Engineering, INESC-ID/Instituto Superior Tecnico, Av. Rovisco Pais, 1, Portugal

I. Introduction

Space is a severe environment for digital electronic systems on-board spacecrafts and satellites, therefore they must be reliable and tolerate errors induced by the radiation from space. Yet, it is very attractive for the development of novel services, to have the computations performed on-board and delivered to simpler receivers, making it possible to deploy in *ad-hoc* scenarios such as emergency and rescue teams.

Synthetic-Aperture Radar (SAR) is an enticing method for Earth monitoring since it does not require a light source. It is capable of operating even under adverse weather conditions and night time. Several observation missions, including Esa Sentinel and Envisat, Alos Palsar, TerraSAR-X, Cosmo-SkyMed and Radarsat, are being used to monitor: ice caps, land and ocean monitoring, vegetation, soil and coastal areas, sea-surface topography, sea and land surface temperature, ocean and land color, weather forecasting, atmosphere and air quality, ozone and UV radiation, and natural disasters such as hurricanes, volcano eruptions and earthquakes [1].

SAR systems are a challenge to design and implement because they require the implementation of computationally intensive algorithms and are usually processed off-board, hence it is of great interest to have high-performance embedded on-board systems capable of implementing such algorithms and broadcasting SAR images, while being fault-tolerant.

Backprojection (BP) is an algorithm for SAR image generation that is capable of generating high quality images of the terrain. It is considered a reference algorithm for image formation since it produces better images with more detail than other algorithms. Notwithstanding, it is a computationally intensive algorithm.

This paper presents a novel fault-tolerant implementation of the Backprojection algorithm targeting a low-power and low-cost System-on-Chip (SoC) Field-Programmable Gate Arrays (FPGAs). SoC FPGAs were chosen because they offer more processing power than a traditional embedded system. The dual-core ARM processor on the device fabric is capable of implementing the Backprojection algorithm, while offering the flexibility to offload the most computational intensive parts of the algorithm to the reconfigurable fabric in the future. Moreover, the non-critically of the component regarding the avionic or satellite system, associated with performance results reported in [2] makes it an attractive solution. So far, the novel fault tolerance mechanism for the Backprojection algorithm has been implemented and evaluated in software, and applied in the protection of components and registers available at the application level, since the device is not large enough to contain a dedicated hardware accelerator. Even if the slowest arithmetic operations in the algorithm were to be accelerated, the control-flow would still be performed the processor level. Results demonstrate the benefit of the proposed approach by offering robustness to faults while requiring less resources than typical fault tolerance mechanisms.

This paper is organized as follows: section 2 contextualizes the application and the problem behind the proposed architecture. Section 3 is devoted to the proposed architecture. Section 4 presents the evaluation of the architecture from previous section. The paper ends with the conclusions and final remarks.

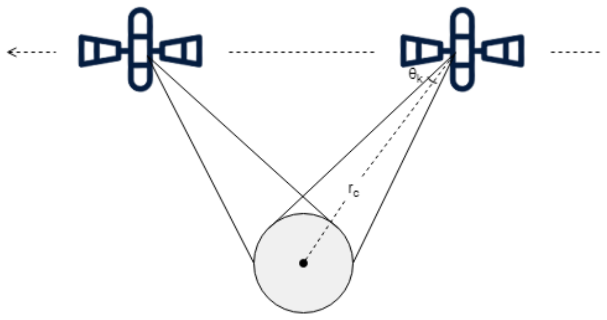


Figure 1 Example of a spotlight SAR system mounted on a moving platform and the respective illuminated region.

II. Background

A. Synthetic-Aperture Radar

SAR is a technology used to generate high-resolution ground imagery. Unlike other radars, SAR uses relatively small antennas and uses the motion of the platform where the system is mounted on to emulate a larger aperture. SAR can operate under severe weather conditions such as rain, fog, clouds, and even darkness [3, 4]. The illuminated region is reconstructed using the radar echoes [5]. SAR has multiple applications, such as surveillance, reconnaissance, medical tomography, agriculture, cartography, and geology [5, 6]. SAR sensors are mounted on moving platforms, as displayed in Fig. 1. SAR systems move in the direction of the azimuth and illuminate a region corresponding to its range. As a consequence of the satellite movement, the illuminated region varies in time.

Terrain images are generated by processing the radar echoes received. The echo processing algorithms can be divided into two categories: frequency-domain and time-domain. Frequency-domain algorithms are based on Fast-Fourier Transformer (FFT) and are usually faster than time-domain algorithms, however, the quality of the generated images is inferior [5–8]. The decrease in quality is caused by assumptions of planarity on the reconstruction surface and the wavefront within the imaged scene and also assume an idealized trajectory for the platform [9]. The mostly used FFT algorithms are range Doppler, range migration, chirp scaling and Polar Format Algorithm (PFA). FFT-based algorithms have a time complexity of $O(N^2 \log N)$ [10, 11]. Time-domain algorithms do not require the assumptions frequency-domain algorithms do, thus leading to the generation of higher-quality images. However, these algorithms have a higher computation cost. This paper is focused on the Backprojection which is one of the most used algorithm in SAR. It has a time complexity of $O(N^3)$.

B. Backprojection Algorithm

The Backprojection is a time-domain algorithm which takes the following values as input: number of pulses, location of the platform for each pulse, the carrier wave number, the radial distance between the plane and target, the range bin resolution, the real distance between two pixels and the measured heights. The Backprojection algorithm, from [12], performs the following steps for each pixel and each pulse:

1) Computes the distance from the SAR platform to the target on the ground, Eq. 1.

$$R = ||r_k - r(n)|| \tag{1}$$

2) Converts the distance to an associated position (range) in the dataset (received echoes).
3) Samples at the computed range using linear interpolation, using Eq. 2.

$$g_{x,y}(r_k) = g(n) + \frac{g(n+1) - g(n)}{r(n+1) - r(n)} \cdot (r_k - r(n)) \tag{2}$$

4) Scales the sampled value by a matched filter to form the pixel contribution. This value is calculated using the Euler expansion, in Eq. 3, and dr is calculated using Eq. 4.

$$e^{i\omega 2\left|\overrightarrow{r_k}\right|} = \cos(2 \cdot \omega \cdot dr) + i\sin(2 \cdot \omega \cdot dr) \tag{3}$$

$$dr = \sqrt{(x - x_k)^2 + (y - y_k)^2 + (z - z_k)^2} - r_c \tag{4}$$

5) Accumulates the contribution into the pixel. The final value of each pixel is given by Eq. 5 [6].

$$f(x, y) = \sum_k g_{x,y}(r_k, \theta_k) \cdot e^{i \cdot \omega \cdot 2\left|\overrightarrow{r_k}\right|} \tag{5}$$

Algorithm 1 presents the pseudocode to compute these steps. k_u represents the wave number and is given by $2\pi f_c/c$, where f_c is the carrier frequency of the waveform and c is the speed of light, a_k refers to the position of the pixel, N_{Bp} is the number of pixels in the image and v_p, corresponds to the platform position.

The implementation of the Backprojection algorithm used in this work was written in C and taken from the PERFECT Suite [12]. The implementation was evaluated using three datasets of different sizes: small (512×512), medium (1024×1024), and large (2048×2048).

Algorithm 1 Backprojection algorithm pseudocode.
Source: PERFECT Manual Suite [12].

```
for all x pixels $x$ do
    for all y pixels $y$ do
        $f(x, y) \leftarrow 0$
        for all pulses $n$ do
            $R \leftarrow ||a_k - v_n||$
            $bin \leftarrow \lfloor (R - r_c)/dr \rfloor$
            if $bin \in [0, N_{Bp} - 2]$ then
                $w \leftarrow \lfloor (R - r_c)/dr \rfloor - bin$
                $s \leftarrow (1 - w) \cdot g(n, bin) + w \cdot g(n, bin + 1)$
                $f(x, y) \leftarrow f(x, y) + s \cdot e^{i \cdot k_u \cdot R}$
            end if
        end for
    end for
end for
```

C. Image Quality Evaluation

The quality of the resulting image is assessed in terms of Signal-To-Noise Ratio (SNR) against the reference, or expected, image. It quantifies the difference between the value of the pixels in both images, Eq. 6. In this equation, r_k is the expected value for *k-th* pixel, t_k the test value for *k-th* pixel, and N the number of pixel to be compared. According to [12], values of SNR above 100dB are considered acceptable.

$$SNR_{dB} = 10 \log_{10} \left(\frac{\sum_{k=1}^{N} |r_k|^2}{\sum_{k=1}^{N} |r_k - t_k|^2} \right) \tag{6}$$

D. Fault-Tolerant SAR Image Generation

Fault tolerance is the ability of a system to be able to remain functional even in the presence of failures. Fault-tolerant systems are able to detect faults and to recover from them. Precise fault tolerance mechanisms rely on repetitions of the same operations in one or more processing units followed by a comparison between them to assess which is the most common, and assume it the correct result. The most common fault tolerance mechanism is Triple Modular Redundancy (TMR), and it consists of having three entities computing the same operation and have a voter entity to compare the results. This mechanism is explained in detail in [13, 14].

Another fault-tolerance mechanism, common in dual-core processors, is the Lock-step and it consists of having both

CPU cores producing the same computations in parallel, and then verify if they achieved the same result frequently. If they disagree, both computations are repeated. The main drawback in the aforementioned mechanisms is that they consume, at least, more than twice the power, and introduce a latency overhead. Devices such as FPGAs with an architecture implemented in reconfigurable fabric allow the use of techniques such as scrubbing, which consists of reconfiguring the architecture repeatedly with a fresh copy of the configuration bitstream to avoid its corruption.

Previous works on implementations of fault-tolerant SAR image generation algorithms can be found in [13–15]. [14] proposes a fault tolerance mechanism for the FFT algorithm based on range and azimuth compression by implementing Concurrent Error Detection (CED) and using weighted sum, and FPGA scrubbing. [15] also presents a mechanism for FFT algorithm based on a weighted checksum encoding scheme. [13] describes a fault-management unit which is responsible for periodically scrubbing of the FPGA configuration data, test a fault condition, removing a faulty processor from the circuit and replace it by an alternative processor, and responsible for a TMR mechanism used in the execution of a SAR algorithm.

E. Approximate Computing Applied to Fault Tolerance

If small variations are introduced in the computations of image processing algorithms, they may not be perceptible in the resulting image. Therefore, such algorithms allow to have some deviations from the correct value while still having valid images. Extensive work in this research area is devoted to trade off the amount of computations required, and their complexity, for the accuracy in the results. Often, the proposed methods rely on the computation of truncated or approximate functions, Look-Up Tables (LUTs), and loop-perforation [16].

In the context of SAR imaging application there is interest in exploring the computation of approximate results to increase the system's reliability, by implementing a reduced-precision fault tolerance mechanism, such as the one described in [17, 18]. Here, the authors proposed a novel hardware mechanism for Single Event Upset (SEU) mitigation, which relies on the comparison of a full-precision result against only one approximate result, obtained from a LUT. Upon comparison of the difference between results against a threshold defined at design time, the mechanism selects either the full-precision or the approximation results, if the difference is greater or smaller, respectively. To minimize the size of the LUT, only the Most Significant Bits (MSbs) are compared. The work proposed in this paper borrows the idea of performing a comparison with the results from a reduced implementation of the computation, but in software. Moreover, the previous works targeted LUTs in hardware whereas the present work is focused on the approximations for the different arithmetic functions, which is more suitable for a CPU implementation.

Even though it is not possible to determine which unit is faulty, it is assumed the result from a full-precision computation is statistically more prone to error than a reduced-precision one. This is justified by the fact that the computations involved in producing an approximate result tend to use smaller wordlengths in their variables, and execute less and faster instructions than a full precision computation, therefore having a smaller memory footprint and being exposed less time to radiation particles. A draft of this concept can be found in [19], however many fundamental details and important discussions are missing from that early contribution.

III. Embedded Fault-Tolerant Backprojection SAR Imaging

This section of the paper is dedicated to the implementation of the novel fault tolerance mechanism in the Backprojection algorithm, consuming less resources than traditional approaches. The most time consuming part in the execution of the algorithm is the generation of the pixel values, which consists of three nested loops, performing the same computations for all the pulses and the x and y coordinates of the image. The inner loop iterates through all the pulses for each position.

The implementation of the algorithm on a Zynq embedded platform from Xilinx, similar device as used or Software-Defined Radio for a CubeSat presented in [20], starts with the characterization of the most time-consuming instructions and then explore fast approximate implementations. It ends with the introduction of the reduced-precision fault tolerance mechanism using the approximation which provides the best results in the shortest time. For clarity and simplification, Backprojection Unit (BPU) designates the segment of code in the inner loop of the Backprojection algorithm.

A. Algorithm Profiling

The implementation of the algorithm was profiled to determine which were the most time consuming operations executed in the computation of the BPUs. They are the natural candidates to be substituted by smaller and faster approximations. This profiling will become relevant in the future to determine which operations to port into a fault-tolerant hardware accelerator in the reconfigurable fabric.

For profiling, the software implementation of the Backprojection algorithm ran as a baremetal application on the target device, Zynq FPGA from Xilinx, for a 512×512 pixels image. The Backprojection algorithm was profiled using GNU gprof*. It took approximately 8 minutes to generate the output image, using the -o3 compiler level. Table 1 shows the times measured to compute images of different sizes. Since these times fit almost a straight line, it is possible to infer that the time required to process a complete image, in minutes, is approximated by Eq.7.

$$ProcTime(pixel) = 0.09556 * p2 - 38.5[min]; \tag{7}$$

Table 2 shows the percentage of time dedicated to the most time consuming instructions in the Backprojection algorithm. Since sin and cos account for nearly 85% of the time, any reduction on their computation introduces significant speedups in the pixel computation, which is given by Amdahl's Law, Eq. 8.

*http://sourceware.org/binutils/docs/gprof/

Table 1 Backprojection algorithm execution time for different images sizes.

Image Size (pixels)	Execution Time (sec)
512×512	480
1024×1024	39780
2048×2048	9360

Table 2 Backprojection algorithm profiling.

Operation	Execution Time (%)
Sine	42.05
Cosine	42.54
Others	15.41

$$S_{\text{latency}}(s) = \frac{1}{(1-f) + \frac{f}{s}} \tag{8}$$

where,

- $S_{latency}$ is the theoretical speedup of the execution of the pixel;
- s is the speedup of the trigonometric functions that can be reduced and accelerated;
- f is the proportion of execution time that taken by the full-precision trigonometric functions.

B. Proposed Architecture

The main objective of this work is to develop an effective method to reduce the total overhead introduced by the fault tolerance mechanism in the system. This was done via approximate computations instead of using full-precision computations. Moreover, in Algorithm 1 the pixel value is given by the sum of all pulse contributions, therefore, it can be parallelized. Using the second CPU core it is possible to compute more than one pixel at a time.

A scheme of the operation of the architecture is displayed in Fig. 2, where it is possible to observe which parts of the BP algorithm are computed in full-precision (fp) and reduced-precision (rp). The approximations are computed after the software computations for each pixel. The approximated computation is represented in Fig. 2 as Reduced-Precision Backprojection Pixel Units (rp-BPUs), while the unaltered part in software is known as Full-Precision Backprojection Pixel Units (fp-BPUs).

C. Algorithm Parallelization

As aforementioned, in this algorithm, the pixel computations have no dependencies, therefore, they can be computed in parallel. The workload was divided between the cores statically since both cores execute the same sequence of instructions and the fault tolerance mechanism doesn't introduce any extra computations in it. After the initial setup by the first CPU core, when the pixels are ready to be computed, the first CPU core will start to compute the odd pixels,

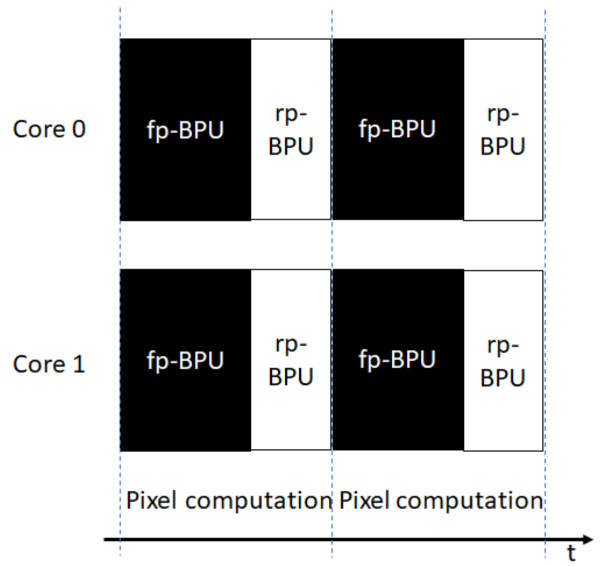


Figure 2 Full-precision and reduced-precision pixel computation workload distribution on the two CPU cores of the SoC.

and notifies the second one to start the computation of the even pixels.

D. Modified Reduced-Precision Redundancy

This work uses a modified version of the Reduced-Precision Redundancy (RPR) mechanism, which computes only one approximation (rp-BPU) after computing the full precision result (BPU) to perform the comparison. The architecture of the modified RPR mechanism is presented in Fig. 3.

Both full-precision and reduced-precision values are compared by computing their difference. If the absolute value of the difference is greater than an acceptable threshold from the reduced-precision value, it is assumed the value is incorrect and the reduced-precision value is used instead. If not, the full-precision result is assumed correct and is used. The reduced-precision value is copied to the output when an error is detected because it is calculated in a shorter amount of time, and thus on average it is less likely to be affected by a fault. The following section discusses the approximation methods and techniques considered.

E. Word-Length Optimization

In addition to the trigonometric optimization, the impact of the floating-point precision was also tested. Reducing the precision of all variables from double-precision to single-precision in the BP Algorithm resulted in a SNR of 15,7dB, which can be seen in Fig. 4. The dataset used is taken from the Perfect suit [12], and consists of a sequence of single-precision values of the pulses for a 512×512 pixel image.

The SNR for the single-precision implementation was $15.7dB$ and it took 224.7 seconds to complete the execution of the algorithm, instead of 240.4 seconds for the double-precision implementation. A fixed-point implementation

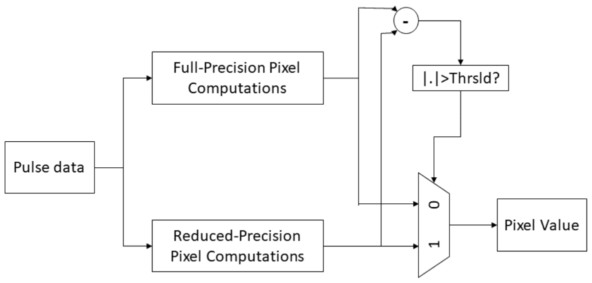
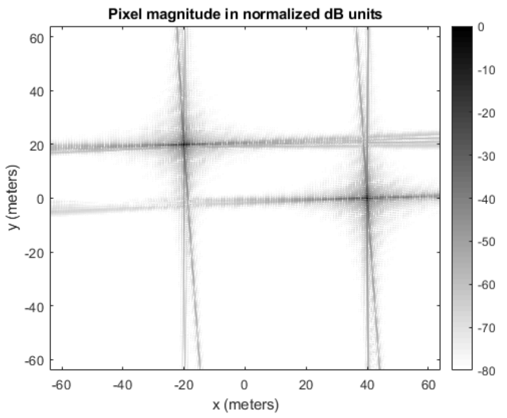
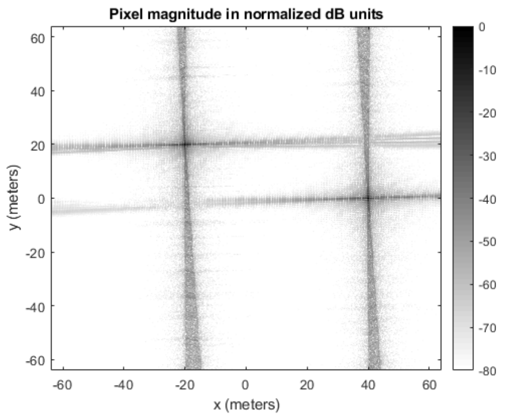

Pulse data
Full-Precision Pixel Computations
Reduced-Precision Pixel Computations
-
|.|>Thrsld?
0
1
Pixel Value

Figure 3 The architecture of the modified RPR fault tolerance mechanism.



(a) SAR image generated using the default gcc/glibc version of the BackProjection algorithm. SNR: 138.9dB.



(b) SAR image generated by removing double-precision variables from the original code and replacing them with single-precision variables. SNR: 15.7dB.

Figure 4 Comparison between the double-precision floating-point version of the algorithm and the single-precision floating-point version.

failed to produced valid images because of the large dynamic range required for the values causing overflow errors.

F. Trigonometric Functions Approximations

Since the trigonometric functions are the most computational demanding ones in the computation of a pixel, several approximations were investigated to reduce the time required by the sin/cos implementation in the default library (gcc/glibc 2.19). The default implementation uses the Chebyshev polynomials. The optimizations for the trigonometric functions tested are:

- COordinate Rotation DIgital Computer (CORDIC) algorithm [21];
- Taylor Series;
- Interpolation LUT;
- LibFixMath;
- Ganssle [22].

The results are presented in Table 3, which includes the execution times on a ARM Cortex-A9 CPU found on a Zynq device from Xilinx, and the impact on the image quality, compared to the default implementation. Figure 5 shows the correspondence between the processing time and the SNR of the produced image. This figure also shows the Pareto frontier, where the optimal point are located. The representations considered were: “sp” for floating-point single-precision, “dp” for floating-point double-precision and “fp” for fixed-point. The execution times were obtained from the execution on a Zynq SoC device.

The results on Table 3 show that all approximations execute faster than the GCC/GLIBC default implementation, as expected. However, most of these optimizations lead to a significant precision loss. From the selected approximation functions, CORDIC is the algorithm with the worst performance, with all its tested versions being slower than any other version of another algorithm.

The results obtained from the Taylor Series algorithm were outperformed by the Ganssle approximations, both in terms of SNR and execution time. The Interpolation LUT approximation (id=13) provided the fastest execution and outperformed some variations of the other algorithms. It is a good alternative in systems with very limited memory since the LUT table occupies 66 bytes only, however, if memory does not represent an issue, the LibFixMath library is a better alternative.

Besides the LUT approximations, LibFixMath provides two other approximations based on Taylor Series. These two variations are outperformed by the Ganssle approximations and even the Taylor Series implementation, with worse performance and less precision. The LibFixMath LUT variation is one of the best options for the Backprojection optimization.

The Ganssle approximations are a good alternative to replace the trigonometric functions in the Backprojection algorithm since most of their points lay on the Pareto frontier. The first variation, the one that uses 3 coefficients to

Table 3 Comparison of the results produced by different approximations for the trigonometric functions.

Algorithm	Approximation	Format	Time [s]	SNR [dB]	id
GCC/GLIBC	-	dp	477.4	138.9	1
CORDIC	5 iterations	fp	214.7	30.7	2
	10 iterations	fp	238.8	60.5	3
	15 iterations	fp	262.7	90.5	4
	20 iterations	fp	286.3	120.2	5
	25 iterations	fp	311.3	136.1	6
	30 iterations	fp	335.1	136.3	7
Taylor Series	3 terms	sp	176.3s	43.6	8
	4 terms	sp	186.0	71.8	9
	5 terms	sp	192.3	103.8	10
	6 terms	sp	201.5	133.6	11
	7 terms	sp	210.4	135.3	12
Interpolation LUT		fp	123.2	69.1	13
LibFixMath	Taylor I	fp	179.3	54.5	14
	Taylor II	fp	158.8	33.6	15
	LUT	fp	134.8	99.2	16
Ganssle	3 coeff.	sp	163.5	66.3	17
	4 coeff.	sp	167.3	105.2	18
	5 coeff.	sp	170.7	118.3	19
	7 coeff.	sp	176.5	134.8	20
	7 coeff.	dp	179.8	135.3	21

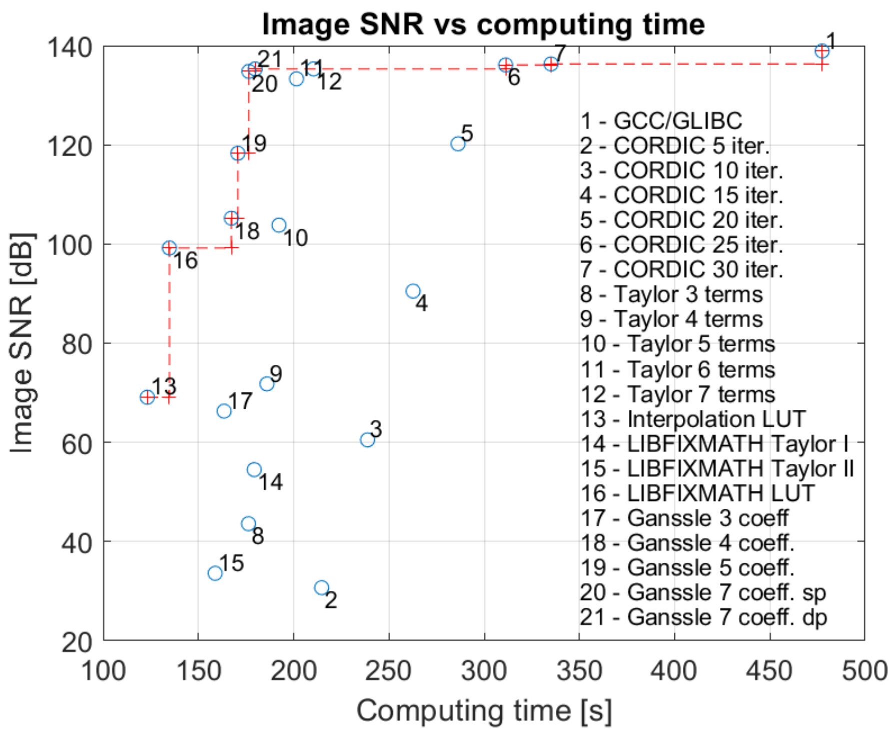


Figure 5 Pareto frontier for the execution times of the trigonometric functions against the produced SNR in the output image.

calculate the final result, is outperformed by both the LUT methods. Nevertheless, the other approximations provide higher precision without a significant increase in the execution time.

The 4-coefficient variation does not provide more precision than LibFixMath LUT function. Moreover, the execution time increased by more than 30 seconds, making the former a better alternative. The 5-coefficient variation provides more precision with an execution time increase of 35 seconds. The 7-coefficient function, implemented with single precision, provides a precision 4dB less than the GCC/GLIBC in the SNR, and an increase of 42 seconds.

In conclusion, the approximations more suitable for the BackProjection algorithm are the LibFixMath LUT (id=16) and the Ganssle approximations of 5 and 7 coefficients (id=19,21). These three functions are used in the implementation of the reduced-precision redundancy mechanism.

IV. Implementation Results

The system was implemented on a Zynq XC7Z020 device from Xilinx [23] supported by a Pynq-Z2 board from TUL. The Zynq device contains a dual-core ARM Cortex-A9 processor and 13k logic slices of reconfigurable logic. Besides the FPGA, the Pynq board has 512MB of DDR3 memory.

The input dataset was obtained from [12], which generates a synthetic image with 512×512 pixels and sequences of 512 pulses, with 13 points in the interpolation step, and an upsample factor of 8.

Table 4 Comparison between the execution times depending on the approximation.

Design	Baseline (single-core)	Baseline (dual-core)	LibFixMath	Ganssle 5-coeff.	Ganssle 7-coeff.
Exec. Time [s]	477.4	240.4	61.1	76.9	79.3

A. Precision Optimization Evaluation

Regarding the optimized architecture, the objective was to assess the quality of the generated images, in terms of SNR. From Table 4 it can be concluded that the architecture implemented using the LibFixMath is 1.58 times faster than the default GCC/GLIBC serial implementation of the algorithm. The 5-coefficient Ganssle and the 7-coefficient Ganssle algorithm were almost 1.50 faster than the baseline version.

When compared to the full precision computation of the BP algorithm, the final architecture using the LibFixMath LUT method, the 5-coefficient and 7-coefficient Ganssle algorithms corresponded to 25%, 32% and 33% of the processing time, respectively.

B. Solution Evaluation

To test the developed architecture, a set of tests were performed. The fault injection was implemented in software and at compile-time by introducing bit-flips according to a distribution which simulates the measurements, performed in the L2 space, and were reported in [24]. In this experiment, on average, there was one SEU per day. However, it should be noted that other locations in space will induce more bit-flips.

The evaluation included testing the generated image for each of the approximations implemented: LibFixMath, 5-coefficient and 7-coefficient Ganssle. Algorithm 1 was implemented using three precision reduction optimizations: the LibFixMath LUT and the 5 and 7-coefficient Ganssle trigonometric functions. The execution times of the complete architecture for each of these optimizations are presented in Table 4.

Regarding the Reduced-Precision Redundancy mechanism, the objective was to observe the final quality of the generated images, using the SNR, in the presence of faults. To test the this mechanism, the following tests were implemented. To inject faults, a fault injection function was called after every statement and a bit-flip could or not affect the last modified variable. Faults are injected at a random time, from a Gaussian distribution, within a random location, from a uniform distribution. The bit-flip per day rate depends on the test.

- **Test RPR With Aggressive Fault Injection** The average occurrences of bit-flips in space is 1 per day. To evaluate the mechanism on a more aggressive scenario, with worse conditions, this fault injection follows a normal distribution with a mean value of 40 and a standard deviation of 5. The results of this test are presented in Table 5.
- **Test RPR With 1440, 2880 and 8640 Bit-Flips per Day** Considering the average of bit-flips, a worse-case scenario was tested: an average of 1440 bit-flips per day, or one every 60, 30 and 10 seconds, respectively. The

Table 5 Results of RPR with Agressive Fault-Injection. Results for SNR in dB

Iteration	LibFixMath	Ganssle 5-coeff	Ganssle 7-coeff
#1	55.4	37.9	-62.3
#2	63.4	79.8	103.3
#3	-inf	82.1	94.7

Table 6 Results of RPR with 1440, 2880, and 8640 bit-flips per day.

		Implementation		
Bit-Flips	**Iteration**	**LibFixMath**	**Ganssle 5-coeff.**	**Ganssle 7-coeff.**
1440	#1	138.9dB	138.8dB	19.9dB
	#2	138.6dB	138.5dB	134.8dB
	#3	138.8dB	138.8dB	138.8dB
2880	#1	97.8dB	67.9dB	109.9dB
	#2	8.3dB	129.1dB	34.4dB
	#3	90.3dB	101.1dB	83.3dB
8640	#1	– inf	– inf	– inf
	#2	-0.4dB	– inf	NaN
	#3	– inf	NaN	– inf

bit-flip affects a random bit in a random variable. The results of this test are presented in Table 6.

Each of the RPR tests was executed three times for each of the optimizations implemented: LibFixMath, 5-coefficient and 7-coefficient Ganssle algorithms.

Compared to alternative implementation such as the TMR, which required almost 12 minutes to execute, for the same 512×512 image, the overhead in the computation of the 3 executions accounts for 8 minutes extra.

C. Power Consumption

Since the target device is appropriate for autonomous systems, it is important to evaluate the power consumed by the different implementations and draw conclusions. Table 7 presents the instant power consumed by the default implementation of the application and the different implementations for the computations of approximations using both CPU cores. The power was measured by reading the current supplied to the circuit with a PL303QMD power supply from TTI.

Given the repetition of a short set of instructions for a long period of time, the instant power was constant throughout the experiments. Energy is the total power required to complete each experiment, and is given by: $E = Power \times Time$. Comparing the baseline single-core implementation against its dual-core version it is possible to observe that the extra energy required for the second CPU core is around 150 mW but allowed to almost reduce by half the processing time. In addition, the reduced-precision fault tolerance mechanism, depending on the approximation algorithm considered, uses from 30% to 40% of energy relative to the dual-core implementation. Moreover, computing approximations uses simpler instructions, therefore consuming less instant power. The duration of the execution is what impacts the most

Table 7 Energy and power consumption depending on the optimization used in the modified optimization mechanism.

Design	**Baseline (single-core)**	**Baseline (dual-core)**	**LibFixMath**	**Ganssle 5-coeff.**	**Ganssle 7-coeff.**
Power [W]	1.695	1.841	1.834	1.829	1.826
Energy [J]	808.1	414.0	541.9	581.4	587.2
Extra Energy	95,1%	Ref	30,8%	40,4%	41,8%
Energy Savings	Ref	48,7%	32,9%	28,1%	27,3%

in terms of energy consumption. On average there was an increase in 30% of time and energy required by the fault tolerance mechanism, but since it takes advantage of the dual-core CPU, it accounts for about 30% energy savings compared to a single-core implementation without any fault tolerance mechanisms.

V. Results Discussion

The improvement of the proposed solution takes into account the benefit introduced by the resilience to faults, and the cost required, in terms of computing time and extra power required. The results for the executions with fault injection of 1440 bit-flips were close to the default GCC/GLIBC implementation SNR value of the image, except the first execution of the 7-coefficient Ganssle approximation.

The other experiments deviated from the default value by a maximum of 4.1dB, and an average of 0.65dB, when in the presence of errors. The low SNR value of the first iteration of the 7-coefficient Ganssle approximation is justified by the fault injection in random variables. Certain variables are more critical than others, for example, the final result of the approximation has a greater impact on the final image quality.

Most of the results for very aggressive error rates were not considered acceptable, since the SNR values are inferior to 100dB. In two experiments, the third of LibFixMath and the first of the 7-coefficient Ganssle approximation were either minus infinite or a negative value, which generate a blank image.

The overall SNR values obtained for 2880 bit-flips are inferior when compared to the results of 1440 bit-flips, which was expected since the rate of bit-flips doubled. The 5-coefficient Ganssle approximation provided the best results of this test: two out of three SNR values are considered acceptable and the other has a SNR almost half of the baseline value. The results obtained using the 7-coefficient Ganssle approximation generate one acceptable image. For this test, the optimization which provided the best results was the 5-coefficient Ganssle approximation.

The rate of 8640 bit-flips represents a fault injection of 10 bit-flips per second. At this rate the proposed mechanism was not successful at detecting and correcting faults. The values in the results table are `nan`, $-\infty$ or negative values, which generate a blank image. A SNR equal to `nan` happens when a bit-flip affects a floating-point variable and the resulting value is not considered a valid floating-point representation. Regarding the SNR of $-\infty$, the calculation of this metric involves a logarithm operation, which equals $-\infty$ in C when calculating the logarithm of 0. The mechanism became ineffective due to the elevated rate of bit-flips, leading to the conclusion the mechanism is only able to tolerate a certain rate of faults.

VI. Conclusions and Future Work

This work presented a novel fault-tolerance mechanism for the implementation of the Backprojection algorithm for on-board SAR imaging systems. The proposed fault tolerance mechanism avoids the use of costly traditional mechanisms, such as TMR or LockStep, while taking advantage of the dual-core processor on the Zynq device to improve performance while exhibiting graceful degradation. The main drawback of this mechanism, is the inability to detect or correct control errors.

This work focused on the study of the approximations for the computations to be protected using a reduced-precision fault tolerance mechanism. Depending on the optimization considered at the design stage, the overhead introduced by the novel fault tolerance mechanism is within 25% to 33% of the prone-to-failure version of the Backprojection Algorithm. Therefore, it is a good alternative for increasing reliability in intensive space applications.

In spite of the limitations of a software implementation a modified reduced-precision fault tolerance mechanism, the algorithm was tested under pessimistic conditions, different from the average use scenario.

Future work envisions the use of the reconfigurable fabric to speedup the execution of the BackProjection algorithm, while including the novel fault tolerance mechanism.

Funding Sources

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with references UID/CEC/50021/2013 and PTDC/EEI-HAC/31819/2017 (SARRROCA).

References

[1] Awange, J. L., and Kyalo Kiema, J. B., *Disaster Monitoring and Management*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013, pp. 437–482. doi:10.1007/978-3-642-34085-7_26, URL https://doi.org/10.1007/978-3-642-34085-7_26.

[2] Lentaris, G., Maragos, K., Stratakos, I., Papadopoulos, L., Papanikolaou, O., Soudris, D., Lourakis, M., Zabulis, X., Gonzalez-Arjona, D., and Furano, G., "High-Performance Embedded Computing in Space: Evaluation of Platforms for Vision-Based Navigation," *Journal of Aerospace Information Systems*, Vol. 15, No. 4, 2018, pp. 178–192. doi:10.2514/1.I010555, URL https://doi.org/10.2514/1.I010555.

[3] Desai, M., and Jenkins, W., "Convolution backprojection image reconstruction for spotlight mode synthetic aperture radar," *IEEE Transactions on Image Processing*, Vol. 1, No. 4, 1992, pp. 505–517. doi:10.1109/83.199920, URL http://ieeexplore.ieee.org/document/199920/.

[4] Wielage, M., Cholewa, F., Riggers, C., Pirsch, P., and Blume, H., "Parallelization strategies for fast factorized backprojection

SAR on embedded multi-core architectures," *2017 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS)*, IEEE, 2017, pp. 1–6. doi:10.1109/COMCAS.2017.8244770, URL `http://ieeexplore.ieee.org/document/8244770/`.

[5] Vu, V. T., Sjögren, T. K., and Pettersson, M. I., "SAR imaging in ground plane using Fast Backprojection for mono- and bistatic cases," *IEEE National Radar Conference - Proceedings*, 2012, pp. 0184–0189. doi:10.1109/RADAR.2012.6212134.

[6] Pritsker, D., "Efficient Global Back-Projection on an FPGA," *2015 IEEE Radar Conference (RadarCon)*, 2015, pp. 0204–0209. doi:10.1109/RADAR.2015.7130996.

[7] Gorham, L. A., and Moore, L. J., "SAR image formation toolbox for MATLAB," 2010, pp. 769906–769913. URL `https://doi.org/10.1117/12.855375`.

[8] Gocho, M., Oishi, N., and Ozaki, A., "Distributed Parallel Backprojection for Real-Time Stripmap SAR Imaging on GPU Clusters," *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, Vol. 2017-Septe, 2017, pp. 619–620. doi:10.1109/CLUSTER.2017.64.

[9] Park, J., Tang, P. T. P., Smelyanskiy, M., Kim, D., and Benson, T., "Efficient backprojection-based synthetic aperture radar computation with many-core processors," *High Performance Computing, Networking, Storage and Analysis (SC), 2012 International Conference for*, 2012, pp. 1–11. doi:10.1109/SC.2012.53.

[10] Cumming, I. G., and Wong, F. H., *Digital Processing of Synthetic Aperture Radar Data: Algorithms and Implementation*, Artech House, Boston, MA, 2005.

[11] ROCCA, F., CAFFORIO, C., and PRATI, C., "Synthetic Aperture Radar: A New Application For Wave Equation Techniques," *Geophysical Prospecting*, Vol. 37, No. 7, 1989, pp. 809–830. doi:10.1111/j.1365-2478.1989.tb02235.x, URL `https://doi.org/10.1111/j.1365-2478.1989.tb02235.x`.

[12] Barker, K., Benson, T., Campbell, D., Ediger, D., Gioiosa, R., Hoisie, A., Kerbyson, D., Manzano, J., Marquez, A., Song, L., Tallent, N., and Tumeo, A., *PERFECT (Power Efficiency Revolution For Embedded Computing Technologies) Benchmark Suite Manual*, Pacific Northwest National Laboratory and Georgia Tech Research Institute, December 2013. `http://hpc.pnnl.gov/projects/PERFECT/`.

[13] Fang, W.-C., Le, C., and Taft, S., "On-board fault-tolerant SAR processor for spaceborne imaging radar systems," *2005 IEEE International Symposium on Circuits and Systems*, 2005, pp. 420–423 Vol. 1. doi:10.1109/ISCAS.2005.1464614.

[14] Jacobs, A., Cieslewski, G., Reardon, C., and George, A., "Multiparadigm Computing for Space-Based Synthetic Aperture Radar." , 01 2008.

[15] Wang, S.-J., and Jha, N. K., "Algorithm-based fault tolerance for FFT networks," *IEEE Transactions on Computers*, Vol. 43, No. 7, 1994, pp. 849–854. doi:10.1109/12.293265.

[16] Wunderlich, H., Braun, C., and Schöll, A., "Fault tolerance of approximate compute algorithms," *2016 IEEE 34th VLSI Test Symposium (VTS)*, 2016, pp. 1–1. doi:10.1109/VTS.2016.7477307.

[17] Duarte, R. P., and Bouganis, C., "Zero-latency datapath error correction framework for over-clocking DSP applications on FPGAs," *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)*, 2014, pp. 1–7. doi:10.1109/ReConFig.2014.7032566.

[18] Duarte, R. P., Véstias, M., and Neto, H., "On the Computation of Approximation Coefficients in ROM-Based Redundancy for SEU Mitigation on FPGAs," *RFPL, 1st Reliable Field Programmable Logic Workshop, FPL 2017 - 27th International Conference on Field Programmable Logic and Applications,*, IEEE, 2017.

[19] Cruz, H., Duarte, R. P., and Neto, H., "Fault-Tolerant Architecture for On-board Dual-Core Synthetic-Aperture Radar Imaging," *Applied Reconfigurable Computing*, edited by C. Hochberger, B. Nelson, A. Koch, R. Woods, and P. Diniz, Springer International Publishing, Cham, 2019, pp. 3–16.

[20] Grayver, E., Chin, A., Hsu, J., Stanev, S., Kun, D., and Parower, A., "Software defined radio for small satellites," *2015 IEEE Aerospace Conference*, 2015, pp. 1–9. doi:10.1109/AERO.2015.7118901.

[21] Volder, J., "The CORDIC Computing Technique," *Papers Presented at the the March 3-5, 1959, Western Joint Computer Conference*, ACM, New York, NY, USA, 1959, pp. 257–261. doi:10.1145/1457838.1457886, URL `http://doi.acm.org/10.1145/1457838.1457886`.

[22] Ganssle, J., *The Firmware Handbook*, Academic Press, Inc., Orlando, FL, USA, 2004.

[23] Xilinx, "Zynq-7000 All Programmable SoC Technical Reference Manual," , 2017.

[24] García-Lario, P., Lorente, R., Merin, B., Sánchez-Portal, M., and Kidger, M., *Herschel Observers' Manual*, Herschel Space Observatory, version 5.0.3 ed., Mar. 2014. HERSCHEL-HSC-DOC-0876.