An Approximation-based Fault Detection Scheme for Image Processing Applications

Matteo Biasielli, Luca Cassano, Antonio Miele
Dip. Elettronica, Informazione e Bioingegneria – Politecnico di Milano – Italy
{*first_name.last_name*}*@polimi.it*

***Abstract*—Image processing applications expose an intrinsic resilience to faults. In this application field the classical Duplication with Comparison (DWC) scheme, where output images are discarded as soon as the two replicas' outputs differ for at least one pixel, may be over-conseravative. This paper introduces a novel lightweight fault detection scheme for image processing applications; i) it extends the DWC scheme by substituting one of the two exact replicas with a faster approximated one; and ii) it features a Neural Network-based checker designed to distinguish between *usable* and *unusable* images instead of faulty/unfaulty ones. The application of the hardening scheme on a case study has shown an execution time reduction from 27% to 34% w.r.t. the DWC, while guaranteeing a comparable fault detection capability.**

***Index Terms*—Approximate Computing, Convolutional Neural Networks, Fault Detection, Image Processing**

I. Introduction

Image processing applications are very often inherently resilient to a certain degree of errors, because i) they may deal with noisy inputs (e.g., images coming from sensors), or ii) the output may be a probabilistic estimate (e.g., in machine learning applications), or iii) the final user of the application's output may effectively carry out the task although the input image is corrupted [1]. Image processing is nowadays employed in a variety of systems (e.g., automotive and aerospace) where safety-/mission-critical applications and non-critical applications coexist. Classical fault detection/tolerance approaches either employ special-purpose radiation-hardened devices [2], which are generally slower than commercial ones, or rely on functionality replication and on a bit-wise comparison [3]. In all cases, the performance degradation introduced to satisfy reliability-related requirements is significant.

Indeed, non-safety-/mission-critical applications may not require a 100% bit-wise error tolerance. Let us consider as an example a satellite that takes aerial pictures of the Earth, executes a pipeline of filters on the captured images and sends then the output images to a ground station. As it has already been discussed by Biasielli *et al.* [4], if a fault affects the execution of the image processing filters, it would be highly beneficial for the satellite to be able to decide whether the corrupted output image is *unusable* by the application running on the ground station or it is still *usable* although being corrupted. In the latter case, re-execution of the application would be avoided, thus saving time.

We propose a lightweight fault detection scheme for image processing applications that combines image usability classification with approximate computing to reduce the time overhead introduced by duplication and to avoid unnecessary re-executions, thus, further saving time. We borrow from [4] the idea of classifying output images based on their *usability* by the final end-user application and the use of a Convolutional Neural Network (CNN) as a checker instead of the classical Two-Rail Checker (TRC). On the other hand, we extend the proposal in [4] by introducing approximate computing. Indeed, instead of having two exact replicas of the pipeline, our hardening scheme pairs the nominal pipeline producing output images at the minimum quality level required by the application designer with a redundant approximated counterpart. The approximated counterpart executes same functionality as the nominal pipeline but on downscaled images, as in [5] (where such technique is not used for fault detection but exclusively for power saving). The defined fault detection scheme allows for a significant time saving w.r.t. both the classical DWC and the proposal in [4] without loosing fault detection capability.

We applied the proposed approach to a case study application for the identification of buildings in aerial photos and we ran the hardened pipeline on a single-core commercial embedded microprocessor. The achieved time savings range from 27% to 34% w.r.t. the traditional DWC approach and from 22% to 34% w.r.t. the proposal in [4], with a false negative rate (images that are not usable by the end-user application that are wrongly not discarded) always lower than 1%, that is acceptably low and comparable with the one in [4].

II. Working scenario and related work

A. The considered working scenario

We consider a microprocessor-based system running an image processing application being a pipeline of several image filters. We focus on Single Event Upsets (SEUs) occurring in the registers of the microprocessor. As discussed in [6], an SEU affecting a non-hardened processor running a software may may lead to effects such as application crashes or hangs, software exceptions and application wrong results. This last case, called Silent Data Corruption (SDC), is the most problematic one since the operating system is not able to autonomously detect the failure. We assume a time-triggered scheduling of the pipeline's filters; therefore, we focus faults causing a wrong output (generally an intermediate image or a data matrix representing a heat map or a feature map) to be returned by a pipeline stage. Finally, we assume that at most a single fault per time may corrupt the application execution.

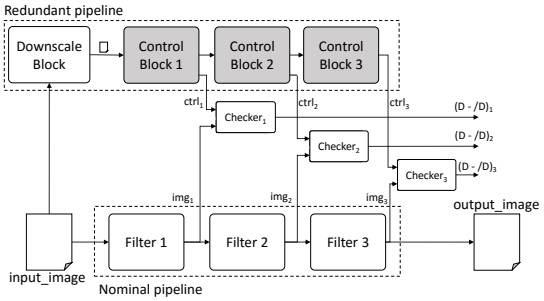

Figure 1. Overall scheme of the proposed fault detection approach.

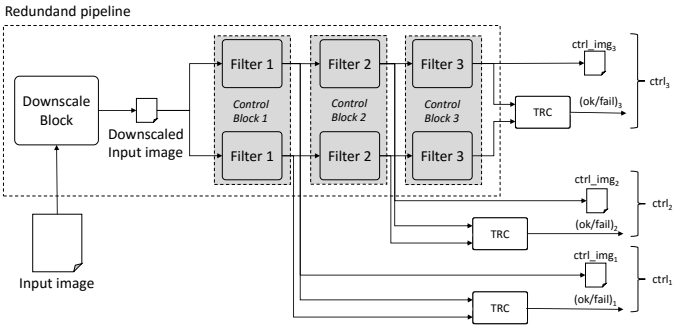

Figure 2. Internal structure of the control block.

B. Related work

The idea of producing inexact, acceptable results after the occurrence of a fault has been proposed in the past mainly for Register-Transfer Level (RTL) designs of Digital Signal Processing (DSP) circuits [7], [8]. The purpose of such a redefinition of the error detection paradigm has been used by approximate computing techniques to reduce the overhead introduced by redundancy-based hardening schemes. In [7] the error is detected only if the difference between two redundant results is larger than a given threshold. Reduced Precision Redundancy (RPR) is applied to Triple Modular Redundancy (TMR) in [8] in order to define replicas that elaborate only on a subset of the most significant bits processed by the nominal system. All these works measure the impact of approximation on the single processed value (e.g., the single pixel) and at that granularity they decide whether the results are *good* or not. As opposite, the fault detection scheme we are here proposing takes into account the usability of the entire output image.

Approximate computing has also been employed at logic the level to design fault-tolerant circuits with a reduced area occupation [9], [10]. Again, such approaches suffer from the limitation of working on a single logic value at the time.

The idea of analysing the usability of the overall result of an image processing application has been considered in recent publications to assess the robustness of machine learning applications for image processing [11], [12]. In particular, SDCs affecting pedestrian detection applications are analyzed and classified as critical/not critical based on the fact that the corrupted output is still usable or not. As previously discussed, the paradigm shift in fault detection from the classical correct/corrupted output to usable/unusable one has been proposed in [4]. However, no approximation was adopted to reduce the overhead introduced by the redundancy scheme.

III. The proposed fault detection scheme

The overall structure of the fault detection scheme is depicted in Figure 1, where we considered a generic example of image processing pipeline composed of three filters. The proposed scheme extends the classical DWC applied at the granularity of the single pipeline stage. The nominal pipeline is paired with a *redundant pipeline* composed of several *control blocks* (one for each stage of the nominal pipeline). The outputs of the control blocks are exclusively used to check the outputs of the nominal pipeline. Unlike DWC, where the two replicas perform exactly the same computation, in the proposed scheme, the redundant pipeline is an approximated version of the nominal pipeline. Approximation is obtained by downscaling $input\_image$, i.e., the input image to the nominal pipeline, by means of the *downscale block*, and then by applying the same functionality of the nominal pipeline. Execution time of the redundant pipeline is significantly shorter even if its output presents a lower quality.

The proposed scheme aims at deciding whether to discard (D) or not (/D) possibly faulty intermediate images based on the usability of the final pipeline's $output\_image$. To do so we substituted the classical TRC with a CNN-based checker. Each i^{th} stage of the pipeline will have its own checker. The main peculiarity of the new checker is that it receives in input two different images, the nominal output img_i produced by the i^{th} nominal filter and the smaller approximated one contained in the control information $ctrl_i$ produced by the i^{th} control block. Therefore, the checker has to be able to distinguish between the differences in the two images caused by the approximation and those additional ones that are caused by faults. In the latter case, the checker has also to be able to predict whether the fault will cause a disruptive effect in the final $output\_image$ or not. To recover from the occurrence of a fault, the proposed scheme adopts re-execution at the granularity of the single pipeline stage to compute the correct result.

A. The redundant pipeline architecture

Figure 2 depicts the internal organization of the redundant pipeline. As previously discussed, it is composed of a downscale block and one control block for each stage of the nominal pipeline. Each control block is composed of two replicas of the corresponding nominal filter, both taking in input a downscaled version of either $input\_image$ or the output data of the previous control block, dubbed $ctrl\_img_{i-1}$. The output of each filters pair is checked through a classical TRC. The response of the TRC, dubbed $(ok/fail)_i$, together with the output data of one of the two filter replicas (which is a downscaled version of the output of the corresponding nominal filter) represents the control information ($ctrl_i$) produced by the control block for filter i.

We introduced filter duplication in the control blocks to reduce the number of false positives, i.e., those cases where the output image would be usable but the checker triggers a re-

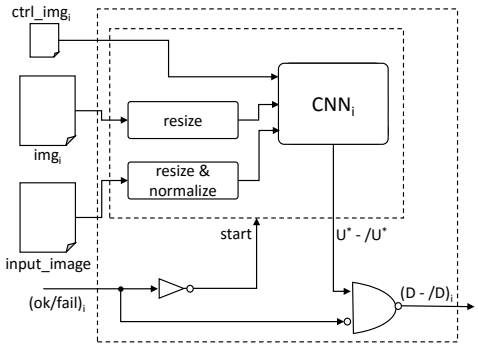

Figure 3. Conceptual representation of the ith checker.

Table I
FILTERS AND CHECKERS EXECUTION TIMES

Module	Execution time (ms)
Sharp.	97.00
R&C	184.00
Thresh.	32.00
Aggreg.	34.00
Approx. Sharp.	3.90
Approx. R&C	6.94
Approx. Thresh.	1.34
Approx. Aggreg.	1.45
Checker SHA	3.31
Checker AGGR	2.15
Checker THR	15.50
Checker RC	2.13
TRC	0.20

execution. Indeed, in case the filter in the control block is not duplicated and its outputs checked, a fault occurring during the execution of the control block may lead the checker to classify the output image of the nominal filter as unusable although it is even uncorrupted. Conversely, in case the filter in the control block is duplicated and checked, when a fault occurs in one of the two control replicas, the TRC will detect it and notify this $(ok/fail)_i$ message in the control information sent to the checker. As discussed in the following subsection, the checker will assume that the output of the nominal filter is uncorrupted in case the $(ok/fail)_i$ message is set to `fail`, given the considered single fault scenario.

Even if the architecture comprises three replicas (one nominal and two approximated), the proposed scheme does not allow to achieve an approximate TMR as done in [8], where the system provides an approximated result if the nominal pipeline fails. Indeed our redundant pipeline works on an extremely downscaled input image and its output could not be provided as output of the system, already being the quality of the nominal pipeline's output the minimum required by the application designer. Finally, it is worth mentioning that the proposed scheme with the duplicated filters in the redundant pipelines is affordable from the execution time point of view because of the introduced approximation.

B. The checker architecture

The checker represents the brain of the proposed fault detection scheme. Indeed, based on the intermediate data img_i produced by the filter and the corresponding control information $ctrl_i$, it is in charge of deciding whether to discard the intermediate result (`D`) or not (`/D`) by predicting the usability of the final nominal $output\_image$ w.r.t. the end-user application purposes. This is achieved by exploiting a CNN that is specifically trained for each filter in the considered application, as discussed in the next section and that will notify the predicted usability (`U*`) vs. unusability (`/U*`).

A representation of the checker for a given nominal filter i is shown in Figure 3. The checker takes the output of the nominal filter (img_i), the control information $ctrl_i$ produced by the associated control block (being $ctrl\_img_i$ and $(ok/fail)_i$) and the global input image of the nominal pipeline $input\_image$.

First, the checker analyzes the $(ok/fail)_i$ signal sent by the control block. As previously discussed, under the single fault assumption, if this signal states a *fail*, the output of the checker will be automatically set to `/D` since a fault occurred in the control block and the output of the nominal filter is assumed to be fault-free. Otherwise, the CNN is invoked. The CNN takes in input img_i, $ctrl\_img_i$ and $input\_image$.

It is worth mentioning that img_i and $input\_image$ are downscaled to the same size of $ctrl\_img_i$ to have matrices of the same size, thus composing a three-dimensional matrix. Moreover, if $ctrl\_img_i$ and img_i are feature maps or heat maps with values in a range different from the color intensity of the pixels of $input\_image$, this last one is normalized to be compliant with the former ones. This step is necessary to improve the performance of the CNN. The output of the CNN is the usable/unusable prediction that is used to decide whether to discard or not the output of the nominal filter.

Finally, the methodology proposed in [4] has been here adopted to design the checker architecture.

IV. EXPERIMENTAL EVALUATION

We experimentally evaluated the proposed fault detection scheme also comparing against the classical DWC and the enhanced DWC proposed in [4]. We considered the same experimental setup described in [4] which allowed us to have a fair comparison against this work. In particular, we considered the same application meant to identify buildings in aerial images and the same companion Oracle. The application pipeline is composed of a sequence of four filters namely: Sharpening, Reshape&Convolution, Thresholding and Aggregation. The Oracle first applies a low-pass filter on the output images to remove noise and then compares the current result with the golden counterpart: the current output is declared to be usable if the areas identified as buildings overlap for at least a given threshold. We employed the same dataset composed of 1,000 images downloaded from Microsoft Bing Maps.

The hardened version of the application has been implemented in C++ and the design methodology in Python, by using TensorFlow [13] for the CNN. LLFI [6] has been used as fault injector. The hardened pipeline has been executed on an ARM A15 core hosted on a Samsung Exynos device.

The execution times of the four filters measured on the target architecture are reported in rows 1-4 of Table I. We obtained the redundant pipeline with a $5\times$ image downscaling on each

Table II
STAGE-BY-STAGE CLASSIFICATION ACCURACY IN FAULTY CONDITIONS

	DWC		[4]				**Our**			
Step	D /U	D U	D /U	/D U	D U	/D /U	D /U	/D U	D U	/D /U
Sharp.	56.7	43.3	56.1	39.1	4.2	0.6	56.2	40.4	2.9	0.5
R&C	55.2	44.8	55.1	41.3	3.6	0.1	54.3	41.9	2.9	0.9
Tresh.	51.1	48.9	48.3	38.9	11.0	1.8	47.2	44.2	5.6	2.9
Aggreg.	40.5	59.5	40.3	54.7	4.8	0.2	39.6	54.6	4.9	0.9
AVG	50.9	49.1	50.0	43.5	5.9	0.7	49.3	45.3	4.1	1.3

Table III
COMPLETE PIPELINE CLASSIFICATION ACCURACY WITH FAULTS

	D /U	/D U	D U	/D /U	Avg time (ms)
DWC	50.5	0.0	49.5	0.0	733.60
[4]	50.2	41.0	8.5	0.3	624.33
Our	49.6	42.3	7.2	0.9	536.12

size; thus, from the 1080×720 input images we obtain 216×156 smaller ones. The execution times of the control blocks for the four stages of the considered pipeline are reported in rows 5-8 of Table I. The last five rows report the execution times of the four checkers and of the TRC, respectively.

As a first analysis we assessed the capability of each checker in identifying unusable images when faults were injected solely in the corresponding pipeline's stage. The results from these experiments are reported in Table II. When comparing the proposed approach with the classical DWC it is possible to observe a significant shift from `D U` to `/D U`, meaning that the checkers are able to identify those faulty images that will not lead to unusable global outputs. This demonstrates that the approach would actually prevent unnecessary re-executions. When compared with [4], our approach achieves comparable results although operating with an approximated replica.

The second experiment aimed at assessing the benefit of the proposed approach in a realistic scenario, where the entire application was considered for random fault injection. This experiment is very important since it allows us to analyse possible interactions among checkers, e.g., a faulty image could be classified as usable by the checker associated with the corrupted stage and it could then be discarded by a subsequent checker. This experiment also allowed us to measure the average execution time of the hardened pipeline and thus the execution time reduction achieved by the proposed approach w.r.t. the classical DWC and [4]. Results from this experiment are reported in Table III. As a first comment, we can again observe a significant shift from `D U` to `/D U` both w.r.t. DWC and [4]. Moreover, it has to be noticed that the amount of false positives (`D U`) is much lower than in [4], while the amount of false negatives (`/D /U`) is slightly higher but still below 1%. Finally, the average execution time reduction achieved with our approach in the presence of faults is significantly high: about 27% w.r.t. DWC and 22% w.r.t. the approach in [4].

As a final experiment, we analysed the behavior of our approach in a fault-free scenario. This allowed us to assess the capability of our solution in considering as "normal" the physiological difference between the output of the nominal pipeline and the redundant approximated one. Moreover, this experiment allowed us to measure the execution time of the hardened pipeline when no fault occurs. Results from this experiment are reported in Table IV, where DWC and [4] are aggregated since they have the same behavior in absence of faults. Unlike, DWC and [4], our approach wrongly discards about 5% of the unfaulty images. This "waste of time" is totally balanced by the time saving introduced by approximation. Indeed, the average execution time reduction is about 34% w.r.t. the considered baseline solutions. As a final consideration, it is worth mentioning that when discarding an image in absence of faults our approach may lead to an infinite loop. Such problem may easily be circumvented by avoiding checker's execution after a pipeline's stage re-execution (which is consistent with the considered single fault scenario).

Table IV
COMPLETE PIPELINE CLASSIFICATION ACCURACY WITHOUT FAULTS

	D /U	/D U	D U	/D /U	Avg time
DWC / [4]	0.0	100.0	0.0	0.0	652.80ms
Our	0.0	94.7	5.3	0.0	429.19ms

V. CONCLUSIONS

We presented a lightweight fault detection scheme for image processing applications that combines image usability prediction with approximate computing to avoid unnecessary re-executions thus reducing overhead introduced by duplication. The application of the methodology to a case study reduced execution time from 27% to 34% w.r.t. DWC and from 22% to 34% w.r.t. [4], with comparable fault detection capabilities.

REFERENCES

[1] S. Mittal, "A Survey of Techniques for Approximate Computing," *ACM Computing Surv.*, vol. 48, no. 4, pp. 62:1–62:33, 2016.

[2] J. Andersson, M. Hjorth, F. Johansson, and S. Habinc, "LEON Processor Devices for Space Missions: First 20 Years of LEON in Space," in *Int. Conf. Space Mission Challenges for Inform. Tech.*, 2017, pp. 136–141.

[3] M. Fayyaz and T. Vladimirova, "Fault-tolerant distributed approach to satellite on-board computer design," in *Aerospace Conf.*, 2014, pp. 1–12.

[4] M. Biasielli, C. Bolchini, L. Cassano, and A. Miele, "A Smart Fault Detection Scheme for Reliable Image Processing Applications," in *Proc. Design, Automa. and Test in Europe Conf. (DATE)*, 2019, pp. 698–703.

[5] A. Raha and V. Raghunathan, "Towards full-system energy-accuracy tradeoffs: A case study of an approximate smart camera system," in *Proc. Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[6] J. Wei, A. Thomas, G. Li, and K. Pattabiraman, "Quantifying the Accuracy of High-Level Fault Injection Techniques for Hardware Faults," in *Proc. Intl. Conf. Dep. Systems and Networks (DSN)*, 2014, pp. 375–382.

[7] B. Shim, S. R. Sridhara, and N. R. Shanbhag, "Reliable low-power digital signal processing via reduced precision redundancy," *IEEE Trans. on VLSI Systems*, vol. 12, no. 5, pp. 497–510, 2004.

[8] A. Ullah, P. Reviriego, S. Pontarelli, and J. A. Maestro, "Majority voting-based reduced precision redundancy adders," *IEEE Trans. Device and Materials Reliability*, vol. 18, no. 1, pp. 122–124, 2018.

[9] I. Albandes, A. Serrano-Cases, A. J. Sánchez-Clemente, M. Martins, A. Martínez-Álvarez, S. Cuenca-Asensi, and F. L. Kastensmidt, "Improving approximate-TMR using multi-objective optimization genetic algorithm," in *Proc. of Latin-Amer. Test Symp. (LATS)*, 2018, pp. 1–6.

[10] A. J. Sanchez-Clemente, L. Entrena, R. Hrbacek, and L. Sekanina, "Error Mitigation Using Approximate Logic Circuits: A Comparison of Probabilistic and Evolutionary Approaches," *IEEE Trans. on Reliability*, vol. 65, no. 4, pp. 1871–1883, 2016.

[11] F. Fernandes dos Santos, P. F. Pimenta, C. Lunardi, L. Draghetti, L. Carro, D. Kaeli, and P. Rech, "Analyzing and Increasing the Reliability of Convolutional Neural Networks on GPUs," *IEEE Transactions on Reliability*, vol. 68, no. 2, pp. 663–677, June 2019.

[12] F. Fernandes dos Santos, L. Carro, and P. Rech, "Kernel and layer vulnerability factor to evaluate object detection reliability in GPUs," *IET Computers Digital Techniques*, vol. 13, no. 3, pp. 178–186, 2019.

[13] M. Abadi *et al.*, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems," 2015. [Online]. Available: www.tensorflow.org/