Exploiting Approximate Computing for implementing Low Cost Fault Tolerance Mechanisms

Bosio, A.; O'Connor, I.; Rodrigues, G. S.; Lima, F. K.; Hamdioui, S.

DOI
10.1109/DTIS48698.2020.9081268

Publication date
2020

Document Version
Final published version

Published in
Proceedings - 2020 15th IEEE International Conference on Design and Technology of Integrated Systems in Nanoscale Era, DTIS 2020

Exploiting Approximate Computing for implementing Low Cost Fault Tolerance Mechanisms

A. Bosio[1], I. O'Connor[1], G. S. Rodrigues[2], F. K. Lima[2], S. Hamdioui[3]
[1]*INL - École Centrale de Lyon, France* – Email: alberto.bosio@ec-lyon.fr
[2]*Instituto de Informatica, PGMicro - Universidade Federal do Rio Grande do Sul, Brazil* – Email: gsrodrigues@inf.ufrgs.br
[3]*Computer Engineering Lab, Delft University of Technology, The Netherlands* – Email: S.Hamdioui@tudelft.nl

***Abstract*—Today's computer architectures and semiconductor technologies are facing major challenges making them incapable to deliver the required features (such as computer efficiency) for emerging applications. Alternative architectures are being under investigation in order to continue deliver sustainable benefits for the foreseeable future society at affordable cost. These architectures are not only changing the traditional computing paradigm (e.g., in terms of programming models, compilers, circuit design), but also setting up new challenges and opportunities concerning the test and reliability. This tutorial targets the challenges and opportunities of using approximate computing for achieving low cost fault tolerance mechanisms.**

***Index Terms*—Approximate computing, fault model, test, reliability**

I. Introduction

Energy and computer efficiency is undoubtedly one of the major driving forces of current computer industry, which is relevant not only for supercomputers, but also for small portable personal electronics and sensors. However, today's computing architectures (mainly based on the CMOS technology) are facing major challenges making them unable to meet the requirements. Such challenges are: power wall, memory wall and Instruction Level Parallelism wall [1]–[3]. For example, the memory wall is due to the increasing gap between processor and memory speeds, which limits the data transfer time and leads to significant energy consumption during the data transfer varying from 70% up to 90% of the overall energy spent by the computing system [4]. Moreover, even the dominating CMOS technology (which made manufacturing of computers feasible) is suffering, especially nodes below 20 nm. At this level the physical characteristics of such devices are leading to high static power consumption, reduced reliability; not to mention increased cost [5]. All of these have led to saturated computer performance and the slowdown of the traditional device scaling, making today's computing systems unable to deliver the required computing and energy efficiency. For example, artificial intelligence is ready to provide solutions in many domains; however, the resource and power demands of the underlying algorithms and implementations are way too high for the target applications. For instance, the amazing performance of AlphaGo [6] required 4 to 6 weeks of training executed on 2000 CPUs and 250 GPUs for a total of about 600kW of power consumption (while the human brain of a go player requires about 20W).

A new design and programming paradigm, Approximate Computing (AxC), has been proposed as a means to make computing systems more energy efficient, faster, and less complex. Intuitively, instead of performing exact computation and, consequently, requiring a high amount of resources, AxC aims at selectively violation of the specifications, trading accuracy for efficiency. The effectiveness of imprecise computation for both software and hardware components implementing inexact algorithms has been demonstrated in the literature, showing an inherent resiliency to errors [7]–[9].

This tutorial focuses on the safety critical systems and it further discusses the impact of Approximate Computing on the system reliability [10]. More in particular, it aims at showing that it is possible to use Approximate Computing to implement efficient fault tolerant architectures. The ultimate goal is to determine the trade-off between the degree of the approximation VS the reliability to finally increase the system lifetime.

Approximate computing has been proposed to achieve energy efficient computation at the cost of accuracy reduction [11]. Hardware designs can profit from approximation to generate circuits with smaller area, thus reducing energy consumption and delay. Software projects use approximation mainly to reduce memory footprint and execution time. Approximation also impacts the system fault tolerance due to its nature [12]. Approximate computing algorithms already handle small inaccuracies generated by the approximation. Thus, very small data corruption errors might not even be noticed by the system as a whole. Some approximation strategies are also inherently fault tolerant. Such is the case of successive approximation: an approximation method that consists of loop executions generating an ever-improving output. This approximation method can also work as a fault tolerance mechanism by itself, given that an error affecting one iteration of the loop can be corrected on the following ones [13]. A designer can use loop perforation to balance execution time and accuracy on successive approximation algorithms, which also impacts the fault tolerance of the system [13]. Another very common approximation method is data size reduction [9], which consists of representing data with less bits than usual. This method has little-to-none impact on software execution time but can highly reduce memory footprint.

Numerical and mathematical properties can also be used to provide valid functional approximation. Taylor series, for example, are used in mathematics to represent a function as a sum of previously calculated terms. The more terms are used, the more accurate the approximation. This type of method can be applied both to software and hardware designs, with different costs [14]. On hardware, the price to pay for more accuracy is either more hardware area or a higher delay: a designer might choose an implementation with pipelines to make it faster (and bigger) or a smaller, loop-execution circuit with a higher delay. On software, the price to pay for this type of approximation is always the execution time. Even on parallel systems, where multiple terms could be executed concurrently, this execution would take processing resources that could otherwise be used to improve the system's performance. Naturally, this approximation method also has a high impact on the system fault tolerance:

978-1-7281-5426-8/20/$31.00 ©2020 IEEE

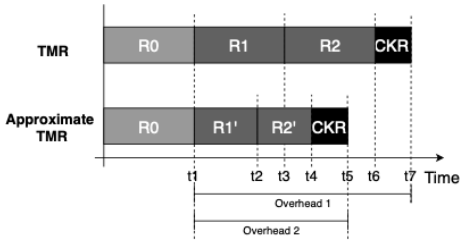

Fig. 1. Approximate TMR diagram.

using bigger hardware increases the probability of a fault, due to a higher number of critical bits. Algorithms with higher execution times are also known to have a higher susceptibility to errors [15], given that they are exposed to more faults per second (in a real use-case scenario of the system execution in a hazardous environment).

Approximate computing can also be used to reduce the costs of traditional fault tolerance methods. Triple modular redundancy (TMR) is one of the most studied fault tolerance and error masking methods in the literature [15]. In its more traditional form, it consists of triplicating a circuit or software code and implementing a checker to verify the consistency of the three execution outputs. If one of the outputs is different from the other two, it shall contain an error that can be masked by the method by accepting the output from the other redundancies as the correct one. Triplicating a whole portion of the system, however, has a high cost (at least 300% area overhead, or execution time for non-parallel software). Approximate computing can be used to provide approximate low-cost redundancies, thus reducing the fault tolerance method costs.

Approximate TMR (ATMR) consists of implementing a TMR with approximate redundancies. It can be applied to both hardware and software projects. Nevertheless, ATMR has to deal with the accuracy loss inherent to approximation. On a traditional TMR approach, the three output values can be compared and checked for errors by a simple bitwise operation. However, an ATMR method needs to handle a possible accuracy difference between the three redundancies. One way of dealing with approximation on ATMR is defining design spaces and assuring that, even in the absence of faults, at least two results will always have the same output [16]. This technique assures that a possible difference caused by the approximation will not turn into an error in the absence of faults. Another way of dealing with the approximation issue on the ATMR checker is with difference thresholds. In this case, the ATMR checker shall only consider an error if the difference between the redundancies outputs is higher than a given threshold. This threshold is defined by the system inaccuracy acceptance. Fig. 1 depicts an example of ATMR compared to the TMR. It can be noticed that ATMR will execute tasks R1' and R2' that are approximate version of the task R0. In this way the overall execution time (t5) will be lower than the TMR execution time (t7).

Some safety-critical systems, in special real-time systems, might not need error masking. Real-time systems deal with *data freshness* requirements, which define time intervals on which data is considered to be updated and valid. A navigation system, for example, might present an error in the data that comes from a radar scan, but because new data coming from a new scan will be generated soon the erroneous data will be overwritten (or even become useless) shortly. In those cases, error masking might be not only unnecessary but also impracticable due to the short data freshness time interval. It is, however, important for the user to know if the current data is to be trusted or not. In an avionics system, for instance, a pilot must know if the date he sees in a panel is trustworthy or not, and take safety measures if needed. Approximation can be used to provide cheap redundancy to mathematically predict if a certain data is inside a possible window of value, and warn the user in the case where the data is absurd [17].

ACKNOWLEDGMENT

This work has been partially founded by CNRS PICS07968 project.

REFERENCES

[1] B. Hoefflinger, "Chips 2020," *The Frontiers Collection*, 2012. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23096-7

[2] D. A. Patterson, "Future of computer architecture," in *Berkeley EECS Annual Research Symposium (BEARS), College of Engineering, UC Berkeley, US*, 2006.

[3] A. Bosio *et al.*, "Rebooting computing: The challenges for test and reliability," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct 2019, pp. 8138–8143.

[4] S. Hamdioui *et al.*, "Memristor based computation-in-memory architecture for data-intensive applications," in *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, March 2015, pp. 1718–1725.

[5] S. Hamdioui *et al.*, "Memristor for Computing: Myth or Reality?" in *Proc. Conf. Des. Autom. Test Eur.* European Design and Automation Association, 2017, pp. 722–731.

[6] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan 2016. [Online]. Available: http://dx.doi.org/10.1038/nature16961

[7] Q. Xu *et al.*, "Approximate computing: A survey," *IEEE Design Test*, vol. 33, no. 1, pp. 8–22, 2016.

[8] L. Anghel *et al.*, "Test and reliability in approximate computing," *Journal of Electronic Testing*, vol. 34, no. 4, pp. 375–387, Aug 2018. [Online]. Available: https://doi.org/10.1007/s10836-018-5734-9

[9] S. Rehman *et al.*, *Heterogeneous Approximate Multipliers: Architectures and Design Methodologies*. Springer International Publishing, 2019, pp. 45–66.

[10] G. Rodrigues *et al.*, "Approximate tmr based on successive approximation and loop perforation in microprocessors," *Microelectronics Reliability*, vol. 100-101, p. 113385, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0026271419304706

[11] J. Han *et al.*, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*, May 2013, pp. 1–6.

[12] G. S. Rodrigues *et al.*, "Evaluating the behavior of successive approximation algorithms under soft errors," in *2017 18th IEEE Latin American Test Symposium (LATS)*, March 2017, pp. 1–6.

[13] G. S. Rodrigues *et al.*, "Exploring the inherent fault tolerance of successive approximation algorithms under laser fault injection," in *2018 IEEE 19th Latin-American Test Symposium (LATS)*, March 2018, pp. 1–6.

[14] G. S. Rodrigues *et al.*, "Analyzing the use of taylor series approximation in hardware and embedded software for good cost-accuracy tradeoffs," in *Applied Reconfigurable Computing. Architectures, Tools, and Applications*, N. Voros *et al.*, Eds. Cham: Springer International Publishing, 2018, pp. 647–658.

[15] ——, "Performances vs reliability: how to exploit approximate computing for safety-critical applications," in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, July 2018, pp. 291–294.

[16] I. A. Gomes *et al.*, "Exploring the use of approximate tmr to mask transient faults in logic with low area overhead," *Microelectronics Reliability*, vol. 55, no. 9, pp. 2072 – 2076, 2015, proceedings of the 26th European Symposium on Reliability of Electron Devices, Failure Physics and Analysis. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0026271415300676

[17] G. S. Rodrigues *et al.*, "Arft: An approximative redundant technique for fault tolerance," in *2018 Conference on Design of Circuits and Integrated Systems (DCIS)*, Nov 2018, pp. 1–6.