Pushing the Limits: How Fault Tolerance Extends the Scope of Approximate Computing

Wunderlich, Hans-Joachim; Braun, Claus; Schöll, Alexander

Proceedings of the 22nd IEEE International Symposium on On-Line Testing and Robust System Design (IOLTS'16) Sant Feliu de Guixols, Catalunya, Spain, 4-6 July 2016

doi: http://dx.doi.org/10.1109/IOLTS.2016.7604686

Abstract: Approximate computing in hardware and software promises significantly improved computational performance combined with very low power and energy consumption. This goal is achieved by both relaxing strict requirements on accuracy and precision, and by allowing a deviating behavior from exact Boolean specifications to a certain extent. Today, approximate computing is often limited to applications with a certain degree of inherent error tolerance, where perfect computational results are not always required. However, in order to fully utilize its benefits, the scope of applications has to be significantly extended to other compute-intensive domains including science and engineering. To meet the often rather strict quality and reliability requirements for computational results in these domains, the use of appropriate characterization and fault tolerance measures is highly required. In this paper, we evaluate some of the available techniques and how they may extend the scope of application for approximate computing.

Preprint

Pushing the Limits: How Fault Tolerance Extends the Scope of Approximate Computing

Hans-Joachim Wunderlich, Claus Braun, Alexander Schöll

Institute of Computer Architecture and Computer Engineering, University of Stuttgart
Pfaffenwaldring 47, D-70569, Germany, Email: {wu,braun,schoell}@informatik.uni-stuttgart.de

***Abstract*—Approximate computing in hardware and software promises significantly improved computational performance combined with very low power and energy consumption. This goal is achieved by both relaxing strict requirements on accuracy and precision, and by allowing a deviating behavior from exact Boolean specifications to a certain extent. Today, approximate computing is often limited to applications with a certain degree of inherent error tolerance, where perfect computational results are not always required. However, in order to fully utilize its benefits, the scope of applications has to be significantly extended to other compute-intensive domains including science and engineering. To meet the often rather strict quality and reliability requirements for computational results in these domains, the use of appropriate characterization and fault tolerance measures is highly required. In this paper, we evaluate some of the available techniques and how they may extend the scope of application for approximate computing.**

***Index Terms*—Approximate Computing, Variable Precision, Metrics, Characterization, Fault Tolerance**

I. Introduction

The approximate computing paradigm spans the whole system stack from hardware up to software and algorithms [1, 2]. By exploiting the error tolerance and robustness of certain applications, approximate computing trades of precision against power, energy, storage, bandwidth, and performance. Originally strict requirements of near-perfect precision are relaxed, and deviating behavior from exact (e.g. Boolean) specifications is allowed to some extent. The current scope of applications for approximate computing can be roughly classified into the following four categories: *Applications with imprecise inputs*, where subsequent computations do not have to be more precise then the input data allows. Such applications include, for instance, voice recognition, motion detection or processing of sensor data. *Applications with imprecise outputs*, where the required output quality is defined by human perception. Well-known examples are audio, image and video processing in multimedia applications. *Applications with ambiguous outputs* computed based on heuristics or stochastic algorithms. This category also covers modern machine learning applications, as well as big data mining and analytics. *Applications with convergent outputs* that are computed based on optimization techniques or iterative methods, where the output may vary depending on the computational precision.

To extend the range of applications, it is essential to distinguish and clarify the terms *accuracy* and *precision* for approximate computing. Accuracy corresponds to the probability that a computed result is within a specified distance to the exact value. The precision of the underlying computations corresponds to this distance and can be determined by different error metrics [1] which quantify the average deviation between approximated results and their corresponding precise counterparts. The *error significance* describes the approximation error severity. For arithmetic results, the error significance between precise results p and approximated results a can be quantified, among others, by the *absolute error*

$$\varepsilon_{abs} := |p - a|$$

and the *relative error*

$$\varepsilon_{rel} := \frac{|p - a|}{|p|}.$$

The *error rate* quantifies the relation between the number of input values and the number of approximate results a that are unequal to the precise result p:

$$error_{rate} := \frac{|\text{imprecise results}|}{|\text{input values}|}$$

Besides, *application-specific* error metrics were proposed, that are also able to evaluate the accuracy of application results [3].

Precision is a characteristic of the approximate hardware, or more general, the approximate arithmetic itself. Approximate arithmetic structures such as adders or multipliers that exhibit a large degree of approximation are typically associated with a lower precision, meaning that the results of their operations are allowed to deviate more from the exact computational result. Accuracy, in turn, is a requirement defined by the application, in particular by its underlying algorithms and the processed data. In Figure 1, an exact value is indicated by a red dot. Based on the accuracy requirements of the target application, a range of potential values with *tolerable error* is formed around the exact value, bounded by appropriate error bounds. The blue and the violet curve characterize two different approximate arithmetic structures with different precisions. The potential computational results within the blue curve (high precision) may meet the application's accuracy requirements. The computational results within the violet curve, however, are more likely to reside outside of the tolerable error range.

To cover also the requirements of scientific computing applications, the design space with the objective parameters area, energy, precision, performance and dependability has

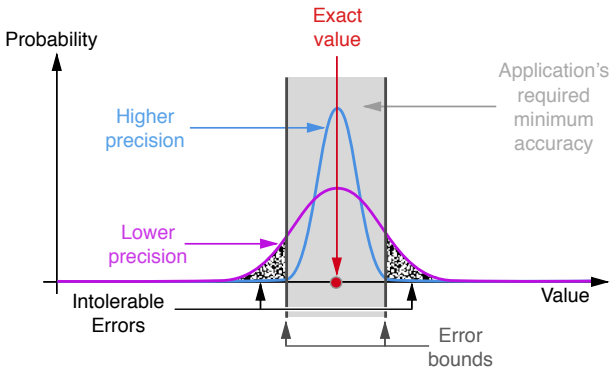

Fig. 1. Relation between accuracy and precision.

to extend its focus from precision to accuracy (Figure 2). Hence, the *required minimum accuracy* of a target application becomes a key design aspect, which has direct influence on the weighting of all the other cornerstones of the design space.

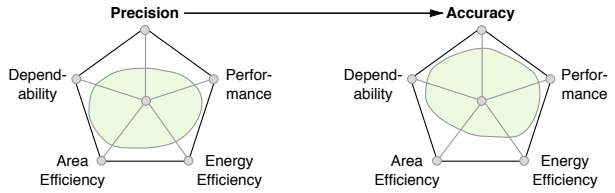

Fig. 2. Changing design space for approximate computing.

Domains such as scientific high-performance computing (HPC) and engineering often pose rather strict requirements on the accuracy of computational results in order to draw reliable and trustworthy conclusions. One major challenge in the new design space for approximate computing is therefore the development of architectures and algorithms which guarantee certain properties of the computational results and errors within known and controllable boundaries.

Detailed and efficient *characterization* is mandatory to determine appropriate error boundaries for applications in approximate computing either offline or online at runtime. Appropriate *fault tolerance* measures, in turn, can be used as an effective tool at software and algorithm level to control such error boundaries and to tackle critical errors outside of the allowed range to assure the quality of computational results.

II. Probabilistic Error Functions for Characterization and Fault Tolerance

The maximum tolerable error (MTE) ε_{max} is an essential criterion for every application in approximate computing since it decides on the degree of approximation that can be applied. *Allowed errors* are those errors, due to approximation or other causes, which reside within the boundaries set by the MTE. One of the main goals of characterization in approximate computing is therefore the efficient determination of these boundaries. Based on the characterization information, appropriate fault tolerance measures such as Algorithm-Based Fault Tolerance [4] can then be applied to monitor and control the error behavior and to tackle critical errors outside of the allowed error boundaries. Since the MTE does not only depend on the underlying algorithms but also significantly on the processed data, the characterization has to be *data-aware*.

Probabilistic error functions (PEF) provide information about the behavior and characteristics of errors based on a probabilistic model. PEFs have been introduced, for instance, for the analysis of rounding errors in floating-point and logarithmic arithmetic [5, 6]. For applications from science and engineering, probabilistic error functions form an attractive and *data-aware* option for the estimation of error boundaries in approximate computing. The basic idea of PEFs is to determine a confidence interval

$$I_{\varepsilon_{max}} := [E(\varepsilon_{max}) - \omega \cdot \sigma(\varepsilon_{max}), E(\varepsilon_{max}) + \omega \cdot \sigma(\varepsilon_{max})]$$

for the maximum tolerable error consisting of the error's expectation value $E(\varepsilon_{max})$ and its standard deviation $\sigma(\varepsilon_{max})$. In order to compute these values, an appropriate probability distribution is required. Here, the fact can be exploited that the mantissas m of normalized floating-point numbers start to follow a reciprocal distribution

$$r(m) = \frac{1}{m \cdot ln(2)}, \text{ with } m \in [\frac{1}{2}, 1),$$

after the application of basic arithmetic operations $\circ \in \{+, -, \cdot, /\}$ [7, 8] (see Figure 3). In [5], terms for the expectation value and variance of the error in mantissas after the application of the four basic arithmetic operations have been introduced as

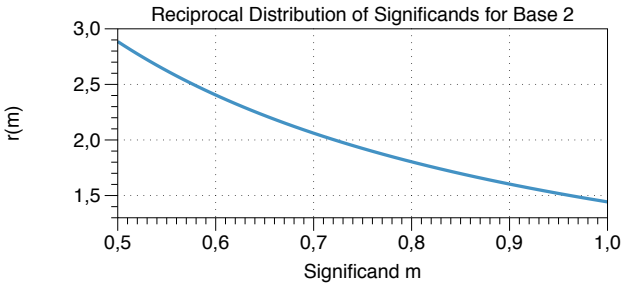

Fig. 3. Reciprocal distribution of floating-point significands.

$$E(\varepsilon_*) = \frac{1}{3} \cdot 2^{-2t} \quad \text{and} \quad Var(\varepsilon_*) = \frac{1}{12} \cdot 2^{-2t}$$

for multiplication and division, as well as

$$E(\varepsilon_+) = 0 \quad \text{and} \quad Var(\varepsilon_+) \leq \frac{1}{8} \cdot 2^{-2t}$$

for addition and subtraction. In these equations the term t denotes the number of bits used to represent the mantissas within a floating-point number. By using these terms as building blocks, the expectation value, variance and standard deviation of the MTE for more complex, composite arithmetic operations can be derived. If a floating-point number f is the result of a sequence of n independent operations, the

expectation value and variance of the MTE are given as

$$E(\varepsilon_{max}) = \sum_{i=1}^{n} E(\varepsilon) \text{ and } Var(\varepsilon_{max}) = \sum_{i=1}^{n} Var(\varepsilon),$$

where ε can be either ε_* or ε_+.

In [9], probabilistic error functions have been applied to estimate the maximum tolerable error in checksum elements of ABFT-protected matrix operations [4]. These elements are computed as inner products for which the expectation value and variance can be computed according to the basic terms given above. The probabilistic error bound has been estimated for each checksum element and has then been used as threshold during the result checking procedure, where checksums from before and after the matrix multiplication are compared. Differences of checksums that exceeded the threshold value were detected as error and corrected subsequently.

Based on the information on the maximum tolerable error provided by the PEF, the number of required precise bits can be determined for a computation. Figure 4 shows exemplarily a histogram with these numbers of required precise bits for a matrix-matrix multiplication. The input values for the matrix

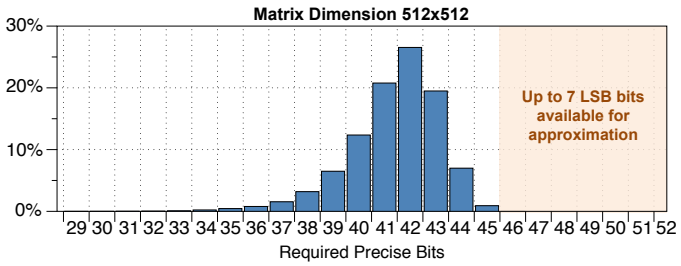

Fig. 4. Example of characterization for matrix-matrix multiplication.

multiplication in this case where uniformly distributed random numbers. The histogram shows that up to 45 precise bits in the mantissa are required, based on the characterization by the PEF. As a consequence for this matrix multiplication, result values with up to 7 approximate bits can be tolerated.

III. Extending Approximate Computing To Iterative Linear Solvers using Fault Tolerance

Large-scale applications from the science and engineering domain often rely on systems of linear equations and their solutions. One of the most common solvers is the Preconditioned Conjugate Gradient (PCG) method [10], which solves linear systems of the form $Ax = b$ iteratively. Compared to direct methods like e.g. the Gaussian-Elimination, PCG finds solutions typically faster.

Static approximation techniques that use single degrees of precision for arithmetic operations are often not suitable for iterative methods as the underlying error resilience may change over time [11]. With static approximation techniques being applied, the typically constant approximation error can exceed the changing error resilience at different points in time, leading to wrong solutions or additional iterations required for correct convergence. Relaxing the solution accuracy constraint (i.e. $Ax \approx b$) to avoid additional energy demand is not feasible as science and engineering applications typically rely on tight accuracy constraints. Instead, the *level of approximation* (i.e. underlying precision) has to be exchanged at runtime according to the changing error resilience. Different approximate hardware designs such as [12–14] provide such exchangeable approximation levels. However, the error resilience must be continuously evaluated during the iterative solving process to minimize energy demands while ensuring correct solver results.

The *residual* $\delta_k := ||b - Ax_k||_2$ determines the accuracy of an *intermediate result* x_k with respect to the solution x after iteration k. The accuracy of the intermediate result x_k improves with a decreasing residual δ_k. PCG iterations are performed until some acceptable result accuracy ϵ is satisfied (e.g. $\delta_k < \epsilon$) which allows to accept the intermediate result x_k as result. Reductions in energy demand can be achieved by minimizing the precision for the underlying arithmetic computations while maintaining the accuracy of single PCG iterations. The intermediate results x_k of PCG iterations are accurate, if the inherent PCG *convergence invariants* are satisfied for successive PCG iterations. By assuring the accuracy for all intermediate results, accurate PCG results with a minimum number of iterations are obtained. PCG represents the solution x as a linear combination of *search directions* $\{p_0, p_1, p_2, ..., p_N\}$ and $x = x_0 + \sum_{k \leq N} \alpha_k p_k$. In order to achieve correct convergence [10], these search directions must be *A*-orthogonal:

$$p_k A p_i \approx 0, k \neq i.$$

At the same time, the internally used *residual variable* δ'_k must represent the actual residual with

$$\delta'_k \approx \delta_k := ||b - Ax_k||_2.$$

The fault tolerance technique presented in previous works [15, 16] evaluates these inherent convergence invariants efficiently to detect errors with high coverage. The approximation technique sketched below can exploit this fault tolerance technique to enable the execution of the Preconditioned Conjugate Gradient (PCG) with correct solutions on approximate hardware. The underlying idea is to periodically estimate the underlying error resilience from inherent solver properties and to evaluate these estimations using this fault tolerance technique. The error resilience is determined by the maximum approximation error which is not violating the property of *A*-orthogonality for successive search directions p_k. While the orientation in p_k is increasingly affected by approximation errors for smaller $||p_k||_2$, it is typically less sensitive for large $||p_k||_2$. For this reason, the magnitude of $||p_k||_2$ is periodically estimated for subsequent iterations using the *tangential angle* φ in the intermediate result x_k with $\varphi := \tan^{-1}(\delta_k)$. This estimation is then translated to a level of approximation using the function $H(\varphi) : \{[0°, 90°[\rightarrow \mathbb{N}\}$ which can be, for instance, a step function. Afterwards, the fault tolerance technique periodically evaluates whether the chosen approximation level is suitable for solver convergence. In case of a violation,

the underlying approximation level is exchanged by increased precision.

This approximation approach has been evaluated using a software-based model for approximate floating-point multiplication. For the highest approximation level, 35 operand bits were approximated (i.e. multiplication using 17 precise bits) while the lowest approximation level provides complete precision. As benchmarks, 14 matrices from the Florida Sparse Matrix Collection [17] were used. All evaluated experiments converged to a correct solution. Experimental results show that the number of iterations is only increased on average by 9.5% (as shown in Figure 5) while the complete hardware utilization is reduced on average by 14.5% (as shown in Figure 6) compared to executing PCG on precise hardware. A reduction in hardware utilization can be observed for 11 out of 14 matrices. These reductions of hardware utilization can be explained by the efficiency of the underlying fault tolerance technique that induces only low runtime overhead to evaluate the inherent solver properties.

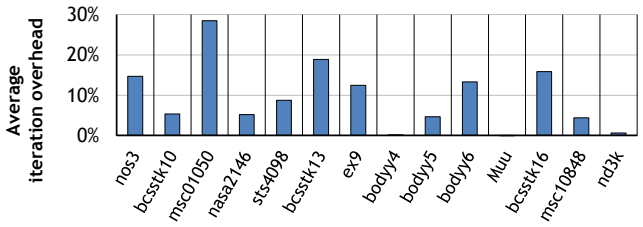

Fig. 5. Average iteration overhead compared to execution on precise hardware.

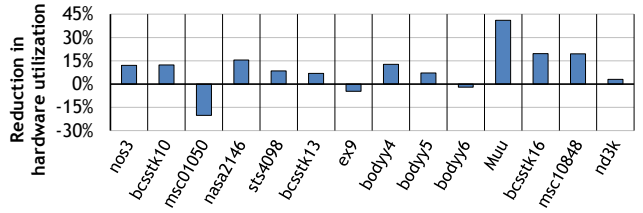

Fig. 6. Reduction in hardware utilization compared to execution on precise hardware.

IV. Conclusion

Approximate computing promises high computational performance combined with low resource requirements. In domains such as scientific computing and engineering, computational results typically have to fulfill certain accuracy and reliability requirements, which often can not directly be supported by approximate computing. However, combinations of efficient application characterization and effective fault tolerance measures can help to enable approximate computing in these domains. Probabilistic error functions are an attractive option for the estimation of boundaries for the maximum tolerable error, which can then be monitored and controlled by fault tolerance techniques such as ABFT. The application of PEFs with ABFT for linear algebra matrix operations has already been successfully demonstrated. Moreover, highly optimized fault tolerance techniques that exploit inherent algorithmic invariants can be used to enable the use of approximate computations, for instance, for iterative linear system solvers.

V. Acknowledgment

The authors would like to thank the German Research Foundation (DFG) for financial support of the project within the Cluster of Excellence in Simulation Technology (EXC 310/2) at the University of Stuttgart.

Bibliography

[1] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design", in *Proc. 18th IEEE European Test Symposium (ETS'13)*, May 2013, pp. 1–6.

[2] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate Computing and the Quest for Computing Efficiency", in *Proc. 52nd ACM/EDAC/IEEE Design Automation Conference (DAC'15)*, June 2015, pp. 1–6.

[3] B. Grigorian and G. Reinman, "Dynamically adaptive and reliable approximate computing using light-weight error analysis", in *Proc. NASA/ESA Conference on Adaptive Hardware and Systems (AHS'14)*, July 2014, pp. 248–255.

[4] K.-H. Huang and J. A. Abraham, "Algorithm-Based Fault Tolerance for Matrix Operations", *IEEE Trans. on Computers*, vol. 33, no. 6, pp. 518–528, Jun. 1984.

[5] J. L. Barlow and E. H. Bareiss, "On Roundoff Error Distributions in Floating Point and Logarithmic Arithmetic", *Computing*, vol. 34, no. 4, pp. 325–347, 1985.

[6] J. L. Barlow and E. H. Bareiss, "Probabilistic Error Analysis of Gaussian Elimination in Floating Point and Logarithmic Arithmetic", *Computing*, vol. 34, no. 4, pp. 349–364, 1985. [Online]. Available: http://dx.doi.org/10.1007/BF02251834

[7] R. W. Hamming, "On the Distribution of Numbers", *Bell System Technical Journal*, vol. 49, no. 8, pp. 1609–1625, 1970.

[8] R. S. Pinkham, "On the Distribution of First Significant Digits", *The Annals of Mathematical Statistics*, vol. 32, no. 4, pp. 1223–1230, 1961. [Online]. Available: http://www.jstor.org/stable/2237922

[9] C. Braun, S. Halder, and H.-J. Wunderlich, "A-ABFT: Autonomous Algorithm-Based Fault Tolerance for Matrix Multiplications on Graphics Processing Units", in *Proc. 44th IEEE/IFIP Intl. Conf. on Dependable Systems and Networks (DSN'14)*, Atlanta, GA, USA, Jun. 2014, pp. 443–454.

[10] Y. Saad, *Iterative Methods for Sparse Linear Systems*. Siam, 2003.

[11] Q. Zhang, F. Yuan, R. Ye, and Q. Xu, "ApproxIt: An Approximate Computing Framework for Iterative Methods", in *Proc. 51st ACM/EDAC/IEEE Design Automation Conference (DAC'14)*, 2014, pp. 1–6.

[12] A. B. Kahng and S. Kang, "Accuracy-configurable adder for approximate arithmetic designs", in *Proceedings of the 49th Annual Design Automation Conference*. ACM, 2012, pp. 820–825.

[13] H. Zhang, W. Zhang, and J. Lach, "A Low-Power Accuracy-Configurable Floating Point Multiplier", in *IEEE International Conference on Computer Design (ICCD)*, Seoul, South Korea, Oct. 2014, pp. 48–54.

[14] C. Liu, J. Han, and F. Lombardi, "A Low-Power, High-Performance Approximate Multiplier with Configurable Partial Error Recovery", in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE'14)*, Dresden, Germany, 2014, pp. 95:1–95:4.

[15] A. Schöll, C. Braun, M. A. Kochte, and H.-J. Wunderlich, "Efficient On-Line Fault-Tolerance for the Preconditioned Conjugate Gradient Method", in *Proc. IEEE Intl. On-Line Testing Symposium (IOLTS)*, Elia, Greece, Jul. 2015, pp. 95–100.

[16] A. Schöll, C. Braun, M. A. Kochte, and H.-J. Wunderlich, "Low-Overhead Fault-Tolerance for the Preconditioned Conjugate Gradient Solver", in *Proc. Intl. Symp. on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT'15)*, Amherst, MA, Oct. 2015, pp. 60–66.

[17] T. A. Davis and Y. Hu, "The University of Florida Sparse Matrix Collection", *ACM Trans. on Mathematical Software*, vol. 38, no. 1, pp. 1:1–1:25, Nov. 2011.