Special Session: A Quantifiable Approach to Approximate Computing*

Chaofan Li
Department of ECE, Texas A&M University
College Station, TX, USA

Deepashree Sengupta
Department of ECE, University of Minnesota
Minneapolis, MN, USA

Farhana Sharmin Snigdha
Department of ECE, University of Minnesota
Minneapolis, MN, USA

Wenbin Xu
Department of ECE, Texas A&M University
College Station, TX, USA

Jiang Hu
Department of ECE, Texas A&M University
College Station, TX, USA

Sachin S. Sapatnekar
Department of ECE, University of Minnesota
Minneapolis, MN, USA

ABSTRACT

Approximate computing has applications in areas such as image processing, neural computation, distributed systems, and real-time systems, where the results may be acceptable in the presence of controlled levels of error. The promise of approximate computing is in its ability to render just enough performance to meet quality constraints. However, going from this theoretical promise to a practical implementation requires a clear comprehension of the system requirements and matching them to the design of approximations as the system is implemented. This involves the tasks of (a) identifying the design space of potential approximations, (b) modeling the injected error as a function of the level of approximation, and (c) optimizing the system over the design space to maximize a metric, typically the power savings, under constraints on the maximum allowable degradation. Often, the error may be introduced at a low level of design (e.g., at the level of a full adder) but its impact must be percolated up to system-level error metrics (e.g., PSNR in a compressed image), and a practical approach must devise a coherent and quantifiable way of translating between error/power tradeoffs at all levels of design.

CCS CONCEPTS

• **Computer systems organization → Embedded systems**; • **Hardware → High-level and register-transfer level synthesis**; **Logic synthesis**; **System-level fault tolerance**; **Methodologies for EDA**;

KEYWORDS

Approximate computing, error resilience, analysis, optimization.

*This work was supported in part by the NSF under awards CCF-1162267, CCF-1525925, and CCF-1525749.

CASES '17 Companion, October 15–20, 2017, Seoul, Republic of Korea

ACM ISBN 978-1-4503-5184-3/17/10…$15.00
https://doi.org/10.1145/3125501.3125511

ACM Reference Format:
Chaofan Li, Deepashree Sengupta, Farhana Sharmin Snigdha, Wenbin Xu, Jiang Hu, and Sachin S. Sapatnekar. 2017. Special Session: A Quantifiable Approach to Approximate Computing. In *Proceedings of CASES '17 Companion, Seoul, Republic of Korea, October 15–20, 2017,* 2 pages.
https://doi.org/10.1145/3125501.3125511

Approximate computing provides an effective approach to trading off energy/power for computational accuracy. This model is particularly relevant for a number of emerging applications where accuracy requirements are flexible, e.g., tasks related to image and video applications, recognition, and classification. While truncation can perform an operation at reduced precision through bit-width reduction, approximation introduces intentional dithering that can statistically provide better precision than truncation.

Approximations can be made at various levels of abstraction:
(1) *Approximate algorithms* could, for example, correspond to reducing the number of iterations in an iterative-improvement algorithm, such as an iterative linear equation solver, or a Newton-Raphson root finder. Note that while this may superficially seem similar to truncation, such operations use full bit-widths, so that the lower order bits include some statistical pseudo-randomness. For a neuromorphic application, errors may correspond to relaxing recognition accuracy for enhanced energy. The level of acceptable approximation depends greatly on the application.
(2) *Approximate data* could be used to simplify the storage burden for data-intensive applications. For instance, image data may use approximate storage, and can leverage the properties of a wide range of memories, e.g., solid-state memories [11] or spintronic memories [10]. Differential techniques for storing approximate data may employ schemes that protect the higher order bits more strongly than the lower-order bits [2].
(3) *Approximate hardware* could introduce intentional errors to improve key performance characteristics (area, delay, and power) of the underlying hardware. Examples include simplified designs for full adders [3], multi-bit adders [5, 8, 15], specialized functional units [14], and data flow graphs [6, 9, 13].

A great deal of prior work has focused on approximations that have been made at the atomic hardware level (e.g., at the levels of gates or adder topologies) and have been evaluated at the application level to determine the level of acceptable error/hardware tradeoffs. A typical design flow works in the opposite direction: an

application is mapped on to a high-level description of a circuit, which is then translated into a gate-level implementation laid out in silicon. At the atomic level, the sources of error and their distribution is most clearly understood, but this level of abstraction is far removed from the application level. For approximate computing to be effectively utilized in such a design flow, it is essential to build a quantitative approach to modeling errors at each level of abstraction and translating them to errors at other levels.

A primary complexity is in the fact that quality metrics at different levels of design may differ. Although metrics such as delay/throughput, area, and power dissipation are typically relevant and can be quantified at all levels, the error or quality metric may be measured differently at various stages of design. For a multimedia application, the user-experience metric is typically captured by the peak signal-to-noise ratio (PSNR) or structural similarity index metric (SSIM) for a set of benchmark applications, while for a neuromorphic application, it could be the percentage of true positives. At the level of electronic design, the application is translated to a high-level computational structure such as a directed acyclic graph (DAG), where the designer deals with data bits, and it is infeasible to work directly with benchmark data. At this stage, the error metric may be computed in terms of a bit error rate (BER), or a probability distribution of error, or its mean and variance. Moving further down in abstraction to the logic level, the error metrics are even further removed from application-level metrics and must necessarily operate in the domain of bit-level errors.

The computation of area, delay, and power metrics can mirror abstraction paradigms in conventional design flows, but the quantification of error and quality metrics is a new requirement. For hardware structures, bit-level error metrics [1, 4] such as the error rate, error significance, average error, and mean square error can be computed efficiently. These metrics provide limited information about the error, and the entire error distribution can be more helpful in evaluating the quality of an approximation. This has led to the development of methods that compute the entire probability mass function (PMF) [7, 12], providing detailed information about the entire distribution of error. The fundamental idea of the approach in [12] is to compute the error PMF for individual module in the circuit (e.g., a full adder), corresponding to a node in the circuit DAG, and then propagating the PMF from the primary inputs to the primary outputs using a topological traversal. The notion of determining output error distributions in a DAG based on node errors can be captured by the output sensitivity to a node error. Such approaches have been used for fast error computation in DAGs [6, 14] in the inner loop of optimizers for approximate computing.

So far, the greatest focus in terms of achieving performance benefits under quantifiable levels of error has been at the hardware level. Several open issues remain to be addressed in future:
(1) At the hardware level, methods that create the link between bit-level error prediction metrics and application-level metrics (e.g., PSNR/SSIM for images, or the true positive ratio for neuromorphic applications) are essential to enable true cross-layer optimization.
(2) It is well recognized that truncation and approximation are both important methods for optimizing performance metrics such as energy, power, and delay. An optimal combination of truncation and approximation is necessary to obtain the best performance.
(3) The relationship between approximation and redudancy is a matter of considerable interest. For example, in a sensor network with redundant data, the level of approximation depends not only on the desired quality of result, but also on the amount of sensor redundancy, which defines the level of error tolerance.
(4) Approximation decisions at the hardware level can be made dynamically on the basis of input data. Cross-level optimization requires methods that can rapidly determine the level of approximation required for satisfactory accuracy for a given application and a set of input data, and communicate between the hardware and software levels to achieve optimal results.
(5) The techniques used for error computation in structures such as DAGs provide a strong base for analyzing the impact of approximations at other levels of design. For instance, the sequence of computations can be represented using a directed graph, and the notions of quantifying error in DAGs can be extended to address approximations at the algorithmic and software levels.

Overcoming these issues can expand the applicability of approximate computing to embedded systems, and enable higher throughput at lower power and lower energy than is achievable today.

REFERENCES

[1] M. K. Ayub, O. Hasan, and M. Shafique. 2017. Statistical Error Analysis for Low Power Approximate Adders. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. 75:1–75:6.

[2] Q. Fan, D. J. Lilja, and S. S. Sapatnekar. 2017. Cost-Quality Trade-offs of Approximate Memory Repair Mechanisms for Image Data. In *Proceedings of the IEEE International Symposium on Quality Electronic Design*. 438–444.

[3] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy. 2013. Low-Power Digital Signal Processing Using Approximate Adders. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 32, 1 (2013), 124–137.

[4] J. Han and M. Orshansky. 2013. Approximate Computing: An Emerging Paradigm For Energy-Efficient Design. In *Proceedings of the IEEE European Test Symposium*.

[5] A. B. Kahng and S. Kang. 2012. Accuracy-Configurable Adder for Approximate Arithmetic Designs. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. 820–825.

[6] C. Li, W. Luo, S. S. Sapatnekar, and J. Hu. 2015. Joint Precision Optimization and High Level Synthesis for Approximate Computing. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. 104.1–104.6.

[7] S. Mazahir, O. Hasan, R. Hafiz, M. Shafique, and J. Henkel. 2017. Probabilistic Error Modeling for Approximate Adders. *IEEE Trans. Comput.* 66, 3 (2017), 515–530.

[8] J. Miao, K. He, A. Gerstlauer, and M. Orshansky. 2012. Modeling and Synthesis of Quality-Energy Optimal Approximate Adders. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 728–735.

[9] K. Nepal, Y. Li, R. I. Bahar, and S. Reda. 2014. ABACUS: A Technique for Automated Behavioral synthesis of Approximate Computing Circuits. In *Proceedings of the IEEE Design, Automation, and Test in Europe*.

[10] A. Ranjan, S. Venkataramani, X. Fong, K. Roy, and A. Raghunathan. 2015. Approximate Storage for Energy Efficient Spintronic Memories. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. 195:1–195:6.

[11] A. Sampson, J. Nelson, K. Strauss, and L. Ceze. 2013. Approximate Storage in Solid-state Memories. In *Proceedings of the IEEE/ACM International Symposium on Microarchitecture*. 25–36.

[12] D. Sengupta and S. S. Sapatnekar. 2015. FEMTO: Fast Error Analysis in Multipliers Through Topological Traversal. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. 294–299.

[13] D. Sengupta, F. S. Snigdha, J. Hu, and S. S. Sapatnekar. 2017. SABER: Selection of Approximate Bits for the Design of Error Tolerant Circuits. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. 72:1–72:6.

[14] F. S. Snigdha, D. Sengupta, J. Hu, and S. S. Sapatnekar. 2016. Optimal Design of JPEG Hardware Under the Approximate Computing Paradigm. In *Proceedings of the ACM/EDAC/IEEE Design Automation Conference*. 106:1–106:6.

[15] W. Xu, S. S. Sapatnekar, and J. Hu. 2017. A Simple Yet Efficient Accuracy Configurable Adder Design. In *Proceedings of the IEEE/ACM International Symposium on Low Power Electronics and Design*.