Editorial

Oliver Keszocze*

Approximate Computing

https://doi.org/10.1515/itit-2022-0027
Received April 27, 2022; accepted April 27, 2022

Keywords: Approximate Computing, Circuit Design, Logic Synthesis, Error Analysis

ACM CCS: Hardware → Integrated Circuits → Logic Circuits, Hardware → Electronic Design Automation → Logic Synthesis

System designers are constantly faced with the task of designing faster systems with a higher throughput, yet, at the same time, lower energy consumption. Artificial Intelligence (AI) applications, most notably in the form of artificial neural networks, are becoming more and more popular and ubiquitous. These applications pose a particular challenge for system designers as they are known for their high demand in computational power resulting in high energy consumption. This greatly hinders the idea of Edge Computing, where computations are carried out on cheap, small and often battery powered devices "on the edge", i. e. in the field.

For many decades, *Moore's law* and *Dennard scaling* led to ever smaller and faster circuits. It seems, that both "laws" have been broken down. Shrinking transistors without negative side-effects is not possible anymore. Parts of modern integrated circuits even need to be turned off during operation in order to meet thermal design power constraints known as "dark silicon".

Fortunately, many practical applications can tolerate a certain degree of incorrectness in computations. This can be, for example, due to the limited perception of the human eye, the inherently probabilistic nature of the application or, noisy or redundant input data. Examples for this include image processing where a human observer will not be able notice small deviations in a smoothing operation. Neural networks, for example, are trained to correctly classify input data only to a certain accuracy and, hence, will never produce absolutely accurate results. Designers of GPS devices, know that there is no point in developing a device computing with a higher accuracy than the GPS signal (or the sensors used) can provide.

***Corresponding author: Oliver Keszocze,** Department of Computer Science, Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), 91058 Erlangen, Germany, e-mail: oliver.keszoecze@fau.de

These opportunities for incorrectness are exploited by the Approximate Computing design paradigm. The core idea behind this paradigm is to find the best trade-off between the degree correctness of computed results (within well-defined bounds) and (possibly significant) gains in non-functional aspects such as energy/power consumption, area, latency, or hardware cost. This allows to partially overcome the problems mentioned above.

Despite being a quite young, Approximate Computing is very active field of research reflected by a growing literature as well as dedicated workshops or tracks being organized. The research is not restricted to logic/electrical circuits but actually spans the whole stack from software to circuit, as outlined below:

Software/algorithm level: Techniques on this level can be quite simple, such as early termination of an iterative algorithm (e. g. the Babylonian method for the computation of the square root), loop perforation or removing neurons form an artificial neural network. But also more ingenious approximation techniques such as the infamous "fast inverse square root" algorithm, approximating $1/\sqrt{x}$ by heavily exploiting the different number representations used in the video game Quake 3, have been devised.

Architectural level: On the architectural level, approximations can be introduced by carefully selecting the system's components. Dedicated approximate accelerators and application-specific instruction processors having a dedicated approximation-aware instruction set architectures have been developed, usually aiming to increase the computational speed. Storing and transmitting data is one of the main sources of energy consumption in modern digital systems. Approaches to reduce the energy consumption range from reducing the precision of the stored values, e. g. using fewer bits, or developing dedicated approximate number formats to designing approximate memory units.

Circuit level: On this level, two main approaches can be distinguished: a) voltage scaling and b) deliberately introducing mathematical errors in arithmetic circuits, e. g. by cutting the carry chain of an adder circuit. The former approach is mainly used to save energy whilst the latter one allows to fine-tune the circuit's area, latency, and/or energy-consumption depending on the design goals. This second approach has been extensively investigated in the field of Approximate Computing in recent publications.

This special issue contains four articles dealing with various aspects of Approximate Computing on different levels of abstraction. As already stated above, approximate circuit design for arithmetic circuits has been a very active field of research yielding many different approximated circuit designs. Consequently, this special issue begins with the overview article "A Survey of Approximate Arithmetic Circuits and Blocks" by Ke Chen, Peipei Yin, Weiqiang Liu and Fabrizio Lombardi. The paper provides the reader with a thorough review of approximate circuits for the most common mathematical operations before looking into Fast Fourier transforms and Multiply-Accumulate (MAC) circuits. As MAC operations constitute the vast majority of all computations in artificial neural networks, optimizing them for energy-efficiency may by one key aspect when trying to bring AI to the edge.

The following article, "Design and Error Analysis of Accuracy-configurable Sequential Multipliers via Segmented Carry Chains" by Jorge Echavarria, Stefan Wildermann, Oliver Keszocze, Faramarz Khosravi, Andreas Becher and Jürgen Teich, proposes a novel multiplier circuit of arbitrary bit-width and further presents a closed form for the error compared to an exact multiplier. The design is evaluated on multiple FPGA and ASIC targets to estimate resource savings.

In "Unlocking Approximation for In-Memory Computing with Cartesian Genetic Programming and Computer Algebra for Arithmetic Circuits", Saman Froehlich and Rolf Drechlser bridge the gap between the circuit level and the architectural level. They use ReRam, a novel memory architecture known for its low power consumption, to perform in-memory computing. They can show that their combination of Cartesian Genetic Programming for the circuit generation and the use of Symbolic Computer Algebra to determine the errors outperforms the state-of-the-art circuit designs.

The last paper of this issue, "Approximating Stochastic Numbers To Reduce Latency" by Syoki Kawaminami, Yukino Watanabe and Shigeru Yamashita, brings ideas of Approximate Computing to the field of Stochastic Computing. While Stochastic Computing already yields circuits with comparatively low costs, they come with the drawback of requiring many cycles to compute the result. In this article, the authors present a method to approximate stochastic numbers in order to reduce the cycles needed.

Bionotes

Prof. Oliver Keszocze
Department of Computer Science,
Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU), 91058 Erlangen,
Germany
oliver.keszoecze@fau.de

Prof. Oliver Keszocze is a Professor of Computer Science at the chair for Hardware-Software-Co-Design at the Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany. His current research interests include several aspects of logic synthesis with a focus on approximate computing for both, ASIC and FPGA. He has been serving as a TPC member for several conferences, including DATE, ICCAD, and ASP-DAC and is a reviewer for IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems as well as for several other journals.