Energy Efficiency through Significance-Based Computing

Dimitrios S. Nikolopoulos and Hans Vandierendonck, *Queen's University of Belfast*

Nikolaos Bellas, Christos D. Antonopoulos, and Spyros Lalis, *University of Thessaly*

Georgios Karakonstantis and Andreas Burg, *École Polytechnique Fédérale de Lausanne*

Uwe Naumann, *RWTH Aachen University*

An extension of approximate computing, significance-based computing exploits applications' inherent error resiliency and offers a new structural paradigm that strategically relaxes full computational precision to provide significant energy savings with minimal performance degradation.

The IT industry's unprecedented growth in recent decades—fueled by aggressive shrinking of transistor size and substantial increases in energy-efficient, high-performance devices—has enabled promising new computing paradigms such as the Internet of Things. But several engineering issues threaten this trend. As hundreds of cores are integrated within a single chip, power consumption will increase to such a degree that soon more than half the transistors in each chip will have to be powered off to avoid burnout. And as process technology progresses to lower geometries, circuits become more prone to performance variations and so do not always meet the desired specifications, resulting in unexpected failures throughout the device's operation lifetime.

Consequently, to guarantee their products' fault-free operation, manufacturers introduce redundancy at various levels of design abstraction trying to detect and correct any single failure. Unfortunately, such measures lead to significant power overhead, further increasing on-chip power consumption. Even more problematic, simply scaling supply voltage to save power is often not a viable option under these conditions, because circuits become more prone to faults at low voltages.

Overcoming the energy scaling barrier is motivating designers to seek innovative breakthroughs to better deal with these power consumption factors. One promising new paradigm is *approximate* computing, which exploits the fault tolerance inherent in many application classes as a way to relax traditional requirements for 100 percent computational precision. For example, recent research has shown that utilizing applications' inherent error resiliency in domains such as signal processing, multimedia operations, high-performance computing, and data analytics can achieve marked energy savings.

However, most current approximate computing techniques are applied ad hoc and thus limited to specific algorithms and software

 Published by the IEEE Computer Society

blocks. What's needed, we believe, is greater algorithmic rigor coupled with a solid methodology that can better characterize and exploit application resiliency, targeting increased power efficiency for all future systems. Thus we propose *significance-based* computing as a more disciplined approach to the approximate computing model.

SIGNIFICANCE-BASED COMPUTING

Not every algorithm part or execution phase plays an equal role in determining the quality of service (QoS) for users in most application domains. Some control codes and computations contribute higher quality results than others, and so they are more critical for ensuring adequate system operation.

We call this characteristic "computational significance," and we consider it a property that all system layers should exploit efficiently and in a disciplined manner, particularly as designers tackle the so-called energy wall and increasing reliability issues arise in sub–28-nm geometries.

The goal of significance-based computing is to allow approximate computation on top of unreliable components—always within user-provided quality bounds—by educated control of error occurrence and propagation throughout the algorithm. The resulting significance-based systems selectively protect the execution of significant computations while allowing a controlled error amount in less significant parts of the algorithm, thus minimizing the energy required to support the system, yet still enabling it to produce useful results.

Mathematically, in a given computation, the significance of any intermediate result/variable can be correlated to
the degree to which its value actually affects the output, or, more generally, a quality metric of the output. Classical first-order and potential higher-order derivative information can serve as indicators for such sensitivity, and hence for determining significance under certain problem-domain-dependent constraints.

The goal of significance-based computing is to allow approximate computation on top of unreliable components—always within user-provided quality bounds.

EXAMPLE

We can illustrate significance-based computing's potential viability using a simple but important example from the high-performance computing domain.

Iterative solvers repeatedly refine the estimates used for solving a system of equations until they reach the required accuracy level. Soft errors occurring in the solvers' state can be allowed to remain without noticeable influence on the quality of results. Alternatively, these errors' impact can be mitigated by increasing the number of iterations invoked to reach convergence, which imposes a cost. In the worst case, soft errors may break convergence altogether.

Fortunately, most soft errors that materialize as bit flips in the solver's state don't induce this extreme worst-case scenario; thus, iterative solvers are suitable candidates for execution on inherently unreliable hardware. Iterative solvers also have built-in resilience, which can secure a satisfactory result—again, sometimes at the cost of additional iterations.

An iterative solver's convergence rate provides a metric of significance for the individual iterations (in other words, for the sensitivity to variables within each iteration). If, for example, a method exhibits a logarithmic convergence rate, then the algorithm's last few iterations are far more significant for convergence than the preceding ones, because errors in early iterations can be corrected by subsequent iterations. This, in turn, dictates an execution strategy whereby we can allow early iterations to be computed imprecisely.

An energy-efficient approach to imprecise iteration execution uses processor cores and memories that operate below their nominal voltage, thus risking timing errors but achieving dramatic power savings. The application and system software can statically or adaptively control the solution quality by assigning high significance to the algorithm's late iterations, or by executing additional iterations if necessary. On a finer-grained scale, we can consider the algorithm's sensitivity to errors in only designated parts of the binary representation of each iteration's variables.

The weighted Jacobi method can serve as a representative use case. Jacobi iteratively solves the system $Ax = b$ for a diagonally dominant matrix A. The algorithm iterates until the convergence condition $\|Ax - b\| \leq$ limit is satisfied, with convergence guaranteed if A is strictly diagonally dominant. We, therefore, use convergence as a quality metric.

Our analysis of the code suggests that, most commonly, the solver can be executed on unreliable hardware and tolerate the vast majority of soft errors that manifest as single bit flips. In this example, only errors in the exponent field of the iteration matrix A's double-precision functional programming (FP) elements would violate the convergence criterion; errors in the mantissa

(especially for early iterations) can be fixed by later iterations.

We experimented with replacing one reliable x86 core operating at nominal voltage with 16 equivalent—in terms of general microarchitecture and industry standard architecture—unreliable x86 cores operating at near-threshold voltage and commensurately low frequency. We were able to achieve energy savings up to 67 percent from significance-based execution, without performance degradation.

Though encouraging, such an exercise would not be viable from a chip-area perspective. Heterogeneous architectures that blend a few reliable cores with many cores that can operate in either reliable or unreliable regions would provide a more viable tradeoff in terms of high performance, low power, and resilience.

A SIGNIFICANCE-BASED HARDWARE AND SOFTWARE STACK

Currently, we are attempting to realize such heterogeneous architectures, along with a proper software stack that guides system operation, according to the significance-based computing paradigm principles being pursued under the project rubric Significance-Based Computing for Reliability and Power Optimization (SCoRPiO; www.scorpio-project.eu). Here we briefly present the main elements of our approach.

Application development support

Starting from the higher layers of abstraction, the programmer should be able to determine and express significance values for different code parts in an easy and intuitive way. At the same time, it is important to exploit the underlying hardware platform's parallel execution capability. To this end, we follow a directives-based approach, whereby parts of the program are explicitly marked as separate tasks with differing significance values, depending on how strongly the intermediate results these tasks compute affect the final computational output.

Operationally, the difference between significant and non-significant tasks is that the latter can execute unreliably, and thus may be allowed to fail or produce wrong results.

Operationally, the difference between significant and non-significant tasks is that the latter can execute unreliably, and thus may be allowed to fail or produce wrong results. To handle such side effects, the programmer can provide lightweight check functions that detect errors and control their propagation in the computation by taking appropriate action. Moreover, the programmer is able to provide metrics—or even code—that allow end-result quality assessment, in addition to specifying quality, performance, and energy constraints.

Given that programmer productivity is a primary concern, it's important to devise methods that either automatically determine significance for different intermediate variables or at least assist the programmer in this characterization process. As one example, SCoRPiO is investigating automatic differentiation (AD) in combination with interval analysis to compute the degree to which each task contributes to final program output quality. AD methods enable a fine-grained sensitivities analysis of numerical simulations, thus representing a preprocessing/profiling stage for an energy-aware compilation. Combinations of AD with interval arithmetic and/or stochastic methods, such as Monte Carlo, are also being considered.

Hardware architecture

At the base of the system, we've designed and developed a many-core platform where cores and their respective cache memories can operate in a conventional, fully reliable mode, as well as in a lower power, but unreliable mode. The main goal is to limit conventional redundancy-based schemes' power and performance overheads that try to protect every instruction equally against any fault.

We've also developed arithmetic units and memories that can exploit many applications' probabilistic nature and relax reliability constraints for some processed and stored data to provide power-saving benefits. Further, we expect power gains through targeted, opportunistic, and aggressive powering of some platform parts below nominal values.

In particular, each core's mode of operation is a function of its voltage and frequency settings. For example, "unreliable" mode may correspond to ultra-low energy configurations in which supply voltage is set below nominal values, causing occasional or even systematic circuit timing errors. This platform can serve as a co-processing accelerator, coupled with a fully reliable (that is, master) CPU hosting an off-the-shelf operating system.

Even when operating in unreliable mode, a core still has to guarantee minimum reliability to support crucial functions such as flow control, crash detection, or even soft failure recognition along with concomitant OS feedback. In addition, unreliable CPUs should offer a special set of reliable

instructions and hardened memory areas that the system software and/or application code can use to implement critical operations, such as updating loop counters or performing pointer arithmetic.

We analyze cores and memories under scaled voltages and variations, and extract and integrate accurate fault and power models in a simulator. The goal is to develop significance-based execution schemes at the software level.

System software

Aside from low-level drivers, all matters pertaining to actual application execution management on top of the many-core accelerator platform can be implemented as a runtime system that lives above the OS. A thin runtime skin is also required on each accelerator core to support controlled application code execution. The objective is typically to schedule an application's tasks on available cores so as to optimize execution in terms of completion time and/or energy consumption.

Significance-based computing introduces an additional aspect: although significant tasks must be scheduled for execution on reliable cores, less significant tasks can be executed on unreliable cores, thereby allowing the platform, in part, to operate at a lower voltage, trading output quality and/or performance for lower energy consumption.

Tradeoffs are typically application- and even input-specific. Thus, the runtime has to monitor application execution and gather information that can be used to make educated scheduling, memory management, and configuration decisions. Of particular importance here is handling errors resulting from unreliable execution. The runtime has to detect, assess the effect of, and even attempt to repair such errors—combining low-level hardware support with higher-level metadata—and also check functions, whether provided by the application, produced by the analysis tools, or obtained through a combination of both.

Overall, we expect the system we're developing to be able to aggressively reduce its power footprint by opportunistically powering hardware modules at below-nominal values. We see significance-based computing as laying a foundation not only for approaching the theoretical limits of energy reduction in CMOS technology, but also for accepting hardware faults in a controlled manner. In doing so—and in viewing eventual reliability issues not as a problem but rather as an opportunity to rethink computation conceptually in novel ways—we believe that we can allow the semiconductor industry to progress beyond 22-nm nodes and move safely into the post-Moore era.

Acknowledgments

This research has received funding from the European Commission's Seventh Framework Programme (FP7/2007-2013) under grant agreement FP7-323872 (Project "SCoRPiO").

Dimitrios S. Nikolopoulos *is a professor and chair of high-performance and distributed computing at the School of Electronics, Electrical Engineering, and Computer Science, Queen's University of Belfast. Contact him at d.nikolopoulos@qub.ac.uk.*

Hans Vandierendonck *is an assistant professor of high-performance and distributed computing at the School of Electronics, Electrical Engineering, and Computer Science, Queen's University of Belfast. Contact him at h.vandierendonck@qub.ac.uk.*

Nikolaos Bellas is *an associate professor in the Department of Electrical and Computer Engineering at the University of Thessaly and a research associate at CERTH. Contact him at nbellas@inf.uth.gr.*

Christos D. Antonopoulos *is an assistant professor in the Department of Electrical and Computer Engineering at the University of Thessaly and a research associate at CERTH. Contact him at cda@inf.uth.gr.*

Spyros Lalis *is an associate professor in the Department of Electrical and Computer Engineering at the University of Thessaly and a research associate at CERTH. Contact him at lalis@inf.uth.gr.*

Georgios Karakonstantis *is a research associate in the Telecommunications Circuits Lab at the École Polytechnique Fédérale de Lausanne. Contact him at georgios.karakonstantis@epfl.ch.*

Andreas Burg *is a professor and director of the Telecommunications Circuits Lab at the École Polytechnique Fédérale de Lausanne. Contact him at andreas.burg@epfl.ch.*

Uwe Naumann *is a professor of computer science at RWTH Aachen University, Germany, and member of the Numerical Algorithms Group, Oxford, UK. Contact him at naumann@stce.rwth-aachen.de.*

Editor: Kirk Cameron, Department of Computer Science, Virginia Tech; greenit@computer.org