Report from Dagstuhl Seminar 21302

Approximate Systems

Edited by

Eva Darulova[1], Babak Falsafi[2], Andreas Gerstlauer[3], and Phillip Stanley-Marbell[4]

1 **MPI-SWS – Kaiserslautern, DE,** eva@mpi-sws.org
2 **EPFL – Lausanne, CH,** babak.falsafi@epfl.ch
3 **University of Texas at Austin, US,** gerstl@ece.utexas.edu
4 **University of Cambridge, GB,** phillip.stanleymarbell@gmail.com

Abstract

This report summarizes the presentations and discussion sessions at the Dagstuhl Seminar 21302 "Approximate Systems" that took place during July 25 – 30, 2021. Due to COVID, the seminar was held in a hybrid fashion, with around 1/3 of the attendees on-site and the remaining ones online. The seminar discussed advances and open challenges in applying approximate computing techniques across the stack and across different application domains, and we hope that this report can provide a useful resource also for other researchers.

Seminar July 25–30, 2021 – http://www.dagstuhl.de/21302
2012 ACM Subject Classification Hardware → Analysis and design of emerging devices and systems; Computer systems organization → Architectures; Computer systems organization → Embedded and cyber-physical systems; Software and its engineering → Software notations and tools
Keywords and phrases approximate computing, energy-efficient computing, pareto optimization
Digital Object Identifier 10.4230/DagRep.11.6.147

1 Executive Summary

Eva Darulova
Babak Falsafi
Andreas Gerstlauer
Phillip Stanley-Marbell

Resource efficiency is becoming an increasingly important challenge, especially due to the pervasiveness of computing systems and the diminishing returns from performance improvements of process technology scaling. At the same time, many important applications have nondeterministic specifications or are robust to noise in their execution. They thus do not necessarily require fully reliable computing systems and their resource consumption can be reduced by introducing or exposing approximations.

While trading correctness for efficiency has been part of computing systems since the early days, it has seen renewed interest in the past decade. Different techniques have been since developed for applying and controlling approximations and the errors they introduce at different levels of the compute stack. Unfortunately, most of these techniques have been

applied in isolation, making simplified assumptions about the other levels. It is thus unclear how all the different techniques interact, combine and complement or negate each other to provide end-to-end application benefits.

The aim of this seminar was to bring together researchers from different domains working on approximate computing, algorithms, programming languages, compilers, architecture and circuits, in order to explore open challenges and opportunities and to define cross-area research directions and collaborations relating to an end-to-end application of approximate computing principles across the compute stack.

The seminar consisted of brief presentations by a subset of the participants that covered the entire computing stack from hardware to applications, and that focused on the current challenges. The talks were followed by discussions in breakout groups that first focused on the different application areas of high-performance computing, embedded systems and deep learning, followed by group discussions on particular fundamental and cross-cutting challenges that were identified during the first breakout session. This report includes the abstracts of the participant's presentations as well as summaries of the breakout group discussions.

2 Table of Contents

3 Overview of Talks

3.1 Approximate Computing to Fight Temperature Effects in NPUs

Hussam Amrouch (Universität Stuttgart, DE)

Main reference Hussam Amrouch, Georgios Zervakis, Sami Salamin, Hammam Kattan, Iraklis Anagnostopoulos, Jörg Henkel: "NPU Thermal Management", IEEE Trans. Comput. Aided Des. Integr. Circuits Syst., Vol. 39(11), pp. 3842–3855, 2020.
URL https://doi.org/10.1109/TCAD.2020.3012753

Neural processing units (NPUs) are becoming an integral part in all modern computing systems due to their substantial role in accelerating neural networks. In this talk, I will discuss the thermal challenges that NPUs bring, demonstrating how multiply-accumulate (MAC) arrays, which form the heart of any NPU, impose serious thermal bottlenecks to any on-chip systems due to their excessive power densities. Some of the questions that we will discuss are 1) the effectiveness of precision scaling and frequency scaling (FS) in temperature reductions for NPUs and 2) how advanced on-chip cooling using superlattice thin-film thermoelectric (TE) open doors for new tradeoffs between temperature, throughput, cooling cost, and inference accuracy in NPU chips.

3.2 On the Curse and the Beauty of Randomness for Guaranteeing Reliable Quality with Unreliable Silicon

Andreas Burg (EPFL – Lausanne, CH)

Joint work of Andreas Burg, Reza Ghanaatian
Main reference Reza Ghanaatian, Marco Widmer, Andreas Burg: "Design for Test with Unreliable Memories by Restoring the Beauty of Randomness", IEEE Design Test, pp. 1–1, 2021.
URL https://doi.org/10.1109/MDAT.2021.3081687

Process variations lead to reliability issues in advanced process nodes. Unfortunately, the associated reliability issues lead to a huge quality spread between manufactured ASICs even for the most fault tolerant applications. This quality spread has burdened "approximate computing" since today's production-test strategies fail to separate dies with sufficient quality from dies with insufficient quality. We analyse this issue, which is ignored in many publications that only report an average quality metric, and provide a surprising and counter-intuitive solution to the testability issue. The key idea is thereby to restore an environment in which frozen (e.g., stuck-at) faults in the hardware no longer have a deterministic effect on computation results. This measure leads to an ergodic fault model that restores the beauty of randomness in a sense that it enables meaningful stochastic quality metrics and allows for simple error mitigation strategies such as averaging which are invalid without randomization.

3.3 Self-Adaptive FPGA-Based Image Processing Using Approximate Arithmetics

Jürgen Teich (Universität Erlangen-Nürnberg, DE)

Joint work of Jutta Pirkl, Andreas Becher, Jorge Echavarria, Jürgen Teich, Stefan Wildermann
Main reference Jutta Pirkl, Andreas Becher, Jorge Echavarria, Jürgen Teich, Stefan Wildermann: "Self-Adaptive FPGA-Based Image Processing Filters Using Approximate Arithmetics", in Proc. of the 20th International Workshop on Software and Compilers for Embedded Systems, SCOPES 2017, Sankt Goar, Germany, June 12-13, 2017, pp. 89–92, ACM, 2017.
URL https://doi.org/10.1145/3078659.3078669

In this talk, we propose a concept of self-adaptive image processing that is able to autonomously adapt 2D-convolution filter operators of different accuracy degrees by means of partial reconfiguration on Field-Programmable-Gate-Arrays (FPGAs). Experimental evaluation shows that the dynamic system is able to better exploit a given error tolerance than any static approximation technique due to its responsiveness to changes in input data. Additionally, it provides a user control knob to select the desired output quality via the metric threshold at runtime.

3.4 Opportunities and Challenges for Approximation in DNA storage

Djordje Jevdjic (National University of Singapore, SG)

DNA has emerged as a chemical medium for both data storage and computation, offering a number of important and unique advantages and promising to close the widening gap between the demand and supply for data storage. However, due to the high error rates and the complex nature of errors in DNA significant amounts of redundant resources must be invested to allow for full recovery of binary data from DNA molecules. The stochastic nature of the chemical processes involved and the approximate nature of data recovery algorithms presents a number of opportunities for approximations across this unique stack. This talk will cover opportunities and challenges in building an error-efficient DNA-based data storage system.

3.5 Calyx: Your DSL-to-Hardware Compiler Construction Kit

Adrian Sampson (Cornell University – Ithaca, US)

Main reference Rachit Nigam, Samuel Thomas, Zhijing Li, Adrian Sampson: "A compiler infrastructure for accelerator generators", in Proc. of the ASPLOS '21: 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Virtual Event, USA, April 19-23, 2021, pp. 804–817, ACM, 2021.
URL https://doi.org/10.1145/3445814.3446712

Calyx is an open-source infrastructure for building DSL-to-hardware compilers. It's centered around a new representation for programs that blends structure (hardware-like components and their connections) and control (temporal ordering). The infrastructure enables optimization and lowering passes that translate high-level DSL semantics into RTL implementations.

3.6 System-aware Distributed Machine Learning

Gauri Joshi (Carnegie Mellon University – Pittsburgh, US)

Large-scale machine learning training, in particular, distributed stochastic gradient descent (SGD), needs to be robust to inherent system variabilities such as unpredictable computation and communication delays. These scalability hurdles are amplified in the emerging framework of federated learning where machine learning models are training on resource-limited edge devices. In this talk, I will discuss open problems in distributed and federated learning.

3.7 Approximate AI on the Edge

David Atienza Alonso (EPFL – Lausanne, CH)

Wearable devices are poised as the next frontier of innovation in the context of Internet-of-Things (IoT) that can benefit from approximate computing to be able to provide personalized healthcare at minimum energy, which can improve our lives and transform the medical industry. This new family of smart wearable medical devices provides a great opportunity for the integration of the nex-generation of artificial intelligence (AI) based technologies in combination with approximate computing in medical devices. However, major key challenges remain in achieving this potential due to the inherent resource-constrained nature of wearable systems, coupled with the uncertainty of the output of the final system when approximation is used at different levels of the system design. In this talk, the current approaches to deliver approximate computing in edge AI to create the next-generation of heterogeneous smart wearables architectures are discussed. The critical architectural enabler is the combination of multiple processors, with a coarse-grained reconfigurable AI accelerator and in-memory computing) as a scalable way to fully deliver the concept of personalized medicine at minimal power. Then, the key challenges to propose an iterative design and optimization flow to bring AI (particularly convolutional neural networks – CNNs) to resource-constrained embedded platforms through selectively applying approximation at different levels will be presented.

3.8 Numerical Encoding for DNN Training

Babak Falsafi (EPFL – Lausanne, CH)

Main reference Mario Drumond, Tao Lin, Martin Jaggi, Babak Falsafi: "Training DNNs with Hybrid Block Floating Point", in Proc. of the Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pp. 451–461, 2018.
URL https://proceedings.neurips.cc/paper/2018/hash/6a9aeddfc689c1d0e3b9ccc3ab651bc5-Abstract.html

The wide adoption of DNNs has given birth to unrelenting computing requirements, forcing datacenter operators to adopt domain-specific accelerators to train them. These accelerators typically employ densely packed full-precision floating-point arithmetic to maximize performance per area. Ongoing research efforts seek to further increase that performance density by

replacing floating-point with fixed- point arithmetic. However, a significant roadblock for these attempts has been fixed point's narrow dynamic range, which is insufficient for DNN training convergence. We identify block floating point (BFP) as a promising alternative representation since it exhibits wide dynamic range and enables the majority of DNN operations to be performed with fixed-point logic. Unfortunately, BFP alone introduces several limitations that preclude its direct applicability. In this work, we introduce HBFP, a hybrid BFP-FP approach, which performs all dot products in BFP and other operations in floating point. HBFP delivers the best of both worlds: the high accuracy of floating point at the superior hardware density of fixed point. For a wide variety of models, we show that HBFP matches floating point's accuracy while enabling hardware implementations that deliver up to 8.5× higher throughput.

3.9 Approximating Numerical Kernels and Beyond

Eva Darulova (MPI-SWS – Kaiserslautern, DE)

Joint work of Anastasia Volkova, Anastasiia Izycheva, Helmut Seidl, Heiko Becker, Magnus Myreen, Zachary Tatlock, Debasmita Lohar, Sylvie Putot, Eric Goubault

Computing resources are fundamentally limited and sometimes an exact solution may not even exist. Thus, when implementing real-world systems, approximations are inevitable, as are the errors they introduce. The magnitude of errors is problem-dependent but higher accuracy generally comes at a cost in terms of memory, energy or runtime, effectively creating an accuracy-efficiency tradeoff. To take advantage of this tradeoff, we need to ensure that the computed results are sufficiently accurate, otherwise we risk disastrously incorrect results or system failures. Unfortunately, the current way of programming with approximations is mostly manual, and consequently costly, error prone and often produces suboptimal results. I will show how we can already approximate straight-line numerical kernels fully automatically, while guaranteeing a user-provided error bound, and discuss our work towards supporting programs beyond kernels that feature conditional statements and loops. Finally, I will sketch what the outstanding challenges are.

3.10 Context-Aware Coding for Computer Memories

Lara Dolecek (University of California at Los Angeles, US)

Joint work of Lara Dolecek, Clayton Schoeny, Mark Gottscho, Puneet Gupta

Error-control coding (ECC) is routinely used to overcome errors in computer memories. In this talk, we demonstrate how intrinsic system knowledge can be used to offer error recovery beyond the baseline ECC guarantees. This system knowledge is used as context for error recovery, and comes in a variety of ways, including data type, instruction structure, and frequency of instructions, among others. We present a heuristic error recovery approach for a known ECC method and a new code design strategy that can take advantage of the underlying system properties. The proposed approach can have benefits in a variety of applications that have intrinsic structure or redundancy, and we envision this idea to be applicable beyond computer memories.

3.11 An Optimization Playground for Precision and Number Representation Tuning

Olivier Sentieys (University & INRIA – Rennes, FR)

Joint work of Van-Phu Ha, Tomofumi Yuki, Daniel Ménard, Olivier Sentieys
Main reference Van-Phu Ha, Olivier Sentieys: "Leveraging Bayesian Optimization to Speed Up Automatic Precision Tuning", in Proc. of the Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021, pp. 1542–1547, IEEE, 2021.
URL https://doi.org/10.23919/DATE51398.2021.9474209

Energy, delay, and area vary a lot between number representations (e.g., float, fixed-point) and word-length (i.e., bit-width of data and computation). Automatic precision tuning is an optimization process that determines the number of bits for each data, minimizing a cost/energy function, constrained by (application) quality degradation (e.g., noise power, SSIM, abs. error). This talk first presents some of the latest results in this field before to move to the problem of jointly exploring number representation during optimization. Results include the development of a custom float operator library and their use in applications such as the training process of deep neural networks with ultra-low precision. This talk concludes with related new problems that may be of interest to build approximate systems.

3.12 A Review and Characterization of Approximate Arithmetic Circuits for Approximate Computing

Jie Han (University of Alberta – Edmonton, CA)

Joint work of Honglan Jiang, Francisco J. H. Santiago, Hai Mo, Leibo Liu, Fabrizio Lombardi, Jie Han
Main reference Honglan Jiang, Francisco Javier Hernandez Santiago, Hai Mo, Leibo Liu, Jie Han: "Approximate Arithmetic Circuits: A Survey, Characterization, and Recent Applications", Proc. IEEE, Vol. 108(12), pp. 2108–2135, 2020.
URL https://doi.org/10.1109/JPROC.2020.3006451
Main reference Honglan Jiang, Cong Liu, Leibo Liu, Fabrizio Lombardi, Jie Han: "A Review, Classification, and Comparative Evaluation of Approximate Arithmetic Circuits", ACM J. Emerg. Technol. Comput. Syst., Vol. 13(4), pp. 60:1–60:34, 2017.
URL https://doi.org/10.1145/3094124

Approximate computing is emerging as a new paradigm for high-performance and energy-efficient design of circuits and systems. This talk aims to provide a brief review and characterization of recently proposed approximate arithmetic circuits under different design constraints. Specifically, approximate adders, multipliers and dividers are characterized via synthesis under optimizations for performance and area, respectively. The error and circuit characteristics are then generalized for different classes of designs. The applications of these circuits in image processing and deep neural networks indicate that such computations are more sensitive to errors in addition than those in multiplication, so a larger approximation can be tolerated in multipliers than in adders. The use of approximate arithmetic circuits can improve the quality of image processing and deep learning in addition to the benefits in performance and power consumption for these applications.

3.13 How do Approximations Impact Analysis, Compiling, and Testing

Sasa Misailovic (University of Illinois – Urbana-Champaign, US)

Tradeoffs between accuracy, performance and energy exist in many resource-intensive applications pervasive in machine learning and robotics. Manually optimizing these tradeoffs with flexible accuracy or precision requirements is extremely difficult. I will highlight our work on programming systems (including languages, compilers, and runtime systems) for accuracy aware optimization of programs.

I will discuss several challenges and lessons learned on how to 1) cope with randomness when testing systems, 2) conquer approximation in heterogeneous systems by novel compilers, and 3) support concurrent and distributed computations in program analysis. I will conclude with a discussion about how we can make future approximation-aware systems more usable.

3.14 An Adaptive Application Framework with Customizable Quality Metrics

Ulrich Kremer (Rutgers University – Piscataway, US)

Joint work of Ulrich Kremer, Liu Liu, Sibren Isaacman
Main reference L. Liu, S. Isaacman, and U.Kremer: An Adaptive Application Framework with Customizable Quality Metrics. ACM Transactions on Design Automation of Electronic Systems (TODAES), Special Issue on Approximate Systems, October 2021, to be pulished.
URL https://doi.org/10.1145/3477428

Many embedded environments require applications to produce outcomes under different, potentially changing, resource constraints. Relaxing application semantics through approximations enables trading off resource usage for outcome quality. Although quality is a highly subjective notion, previous work assumes given, fixed low-level quality metrics that often lack a strong correlation to a user's higher-level quality experience. Users may also change their minds with respect to their quality expectations depending on the resource budgets they are willing to dedicate to an execution. This motivates the need for an adaptive application framework where users provide execution budgets and a customized quality notion. The paper presents a novel adaptive program graph representation that enables user-level, customizable quality based on basic quality aspects defined by application developers. Developers also define application configuration spaces, with possible customization to eliminate undesirable configurations. At runtime, the graph enables the dynamic selection of the configuration with maximal customized quality within the user provided resource budget.

An adaptive application framework based on our novel graph representation has been implemented on Android and Linux platforms, and evaluated on eight benchmark programs, four with fully customizable quality. Using custom quality instead of the default quality, users may improve their subjective quality experience value by up to 3.59×, with 1.76× on average under different resource constraints. Developers are able to exploit their application structure knowledge to define configuration spaces that are on average 68.7compared to existing, structure oblivious approaches. The overhead of dynamic reconfiguration averages less than 1.84% of the overall application execution time.

3.15 How to Reduce Numerical Precision in Weather and Climate Simulations

Peter Dueben (ECMWF – Reading, GB)

This talk will give an overview on ongoing efforts to explore the reduction of numerical precision in weather and climate models. While the European Centre for Medium-Range Weather Forecasts (ECMWF) has recently switched from double to single precision in operational predictions, we also investigate the use of lower precision levels, such as half precision, for our models. The precision reduction is non-trivial as it is difficult to diagnose a precision level that is still "good enough" when simulating a chaotic system – such as atmosphere or ocean. On the other hand, we have good knowledge about forecast uncertainties which can be used to optimise precision within the simulations.

3.16 Some Mathematical Challenges in Inexact Computing

Laura Monroe (Los Alamos National Laboratory, US)

This talk is an overview of the relationship between mathematics and inexact systems, with an emphasis on errors and error-correction. In particular, we emphasize hardware/software codesign and the interplay between mathematics, the base physics of the system, and the algorithms and software that brings them together.

Challenges up and down the stack are discussed, and several examples are given, including software-defined error correction (Gottscho et al.); the interplay between Hamming and application-based distances, with applications ranging from computational fluid dynamics to basic integer calculations; and natural application resilience. The fault model is discussed, with its derivation from the physical device and modes of addressing device-dependent faults.

Finally, we propose development of a catalog of design patterns, inspired by those in the object-oriented design community [1] and the resilience community [2], as a tool describing solutions to problems commonly seen in inexact systems and software and representing best practices used by experienced practitioners in the field.

References

1 Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, Mass. : Addison-Wesley, 2016

2 Saurabh Hukerikar and Christian Engelmann. *Resilience Design Patterns: A Structured Approach to Resilience at Extreme Scale*. Journal of Supercomputing Frontiers and Innovations (JSFI), volume 4, number 3, pages 4-42, October 1, 2017

4 Working groups

4.1 Approximate Computing Challenges for HPC Applications

Eva Darulova (MPI-SWS – Kaiserslautern, DE)

High-performance computing, such as used for example in weather simulations, has traditionally not actively applied (additional) approximations beyond those necessary by the domain. Computations were done with 64 bit floating-point arithmetic and assumed to be correct enough; hardware is supposed to be correct, too. The picture is starting to change as hardware is not becoming automatically faster and more efficient if one just waits, and is also becoming more heterogeneous. Hence, in principle, approximate computing is of interest to the HPC community. One particular case of existing deliberate approximations are recent efforts to move computations from 64 bit floats to lower precisions. These efforts are motivated by available low-precision hardware for machine learning. The conversion is currently done manually and takes a very long time. Hence, tool support in form of debugging tools would be much appreciated. Static analysis tools or fully automated tools are likely to not be very meaningful, since an exact baseline is often not available, i.e. the approximated code needs to match the existing implementation's behavior. The community seems to be slowly moving away from Fortran to Python or DSLs, making it more feasible to develop debugging tools. In addition to finite-precision, we have identified further approximations at different levels of the stack that may be beneficial to HPC. One are techniques from federated machine learning that may be helpful for the often massively parallel HPCs applications (e.g. a weather simulation may discretize the earth and distribute the simulation for each part on 1000 nodes). Further, error correction techniques may be of interest as the probability of random bitflips increases, one hand hand due to the sheer size of the computations, and on the other hand due to the use of approximate hardware. Accelerators have the potential to provide speed-ups important to HPC, today mostly GPUs are used. FPGAs have been used to run only small models, as programming them is manual and painful. In summary, there seems to be potential to apply approximations in HPC across the stack. In order to develop the corresponding techniques and tools, representative benchmarks or example codes are needed.

4.2 Approximate Computing Challenges for Embedded Systems

Phillip Stanley-Marbell (University of Cambridge, GB)

The discussion in the embedded systems theme centered on the idea that in embedded computing systems, which by definition interface with the physical world, it is essential to know when the result of an approximation technique has led to an erroneous data value or erroneous control flow behavior. One way in which erroneous behavior could be detected is by checking for violation of some invariant property, either on a single value of machine state or across multiple items of program state (e.g., an invariant across entries in a matrix). A general concept discussed was the idea of exploiting the fact that the signals in embedded systems are usually physical signals which need to obey the laws of physics.

The discussions observed that erroneous behavior of interest will typically be input-dependent, since erroneous behaviors that are not input-dependent could in principle be found by static analysis techniques. Dynamic detection of erroneous behavior resulting from approximation is, at the moment, not a well-explored topic and could be fertile ground for future research.

Once erroneous behavior is detected, there is the natural question of how to make this detected state available to a system to act upon it. Erroneous behavior could be detected inside, e.g., a microprocessor or compute accelerator, in which case one natural way to notify the system of the detection is by raising an interrupt. In hardware systems in general, the detection could be used to set a hardware signal, while in software systems an exception could be raised (to be handled by an appropriate exception handler).

Finally, when erroneous behavior has been detected and the system notified, it will still be a challenge to develop new methods to adapt to the result of erroneous behavior resulting from approximation. This was again identified in the embedded theme as a fertile ground for future research.

4.3 Approximate Computing Challenges for Deep Learning

Babak Falsafi (EPFL – Lausanne, CH)

Machine learning has emerged as a killer application with wide applicability across a number of domains. There are two trends that are at inflection point in ML: (1) the imminent end of Moore's Law and the need for post-Moore platforms, and (2) the continued exponential growth in ML at 20% per year (as forecasted by a number of think-tanks including IDC) and the need for scaling platforms. There are a number of fundamental challenges in ML platform design. One is the lack explainability and ad hoc methods to improve algorithms. Another is the divergence in platforms between inference and training. A third fundamental challenge is search for models that would allow for iso-accuracy in prediction while reducing the required computational resources. Fortunately, ML inherently lends itself well to cross-layer optimization in platforms and co-design. One great area to explore is convergence of inference/training through common numerical encodings that would enable algorithm/hardware co-design. Explainability of optimal numerical encoding would require hand-in-hand collaboration of computer system designers and numerical analysts. Another area would be hardware mechanisms that would facilitate parameter and model search. A third area of research would be how the choice of ML application would impact algorithm/hardware co-design.

4.4 Design Patterns for Approximation Across the Stack

Damien Zufferey (MPI-SWS – Kaiserslautern, DE)

Specific approximation techniques, e.g., numerical precision, can be applied on their own. However, to get more benefit one can apply approximation consistently across the entire software and hardware stack. Optimization can span the computation, communication,

memory, storage, and time. Cyber-physical systems is an application domain that can benefit from approximations that touches all theses domains. For instance, reduced precision and perforation can reduce the energy needed by a control loop but the trade-off is affecting the system's stability. Sensors are an ideal place to integrate very fast and efficient analog processing. In the case of distributed control, reducing the amount and frequency of communication is what reducing numerical precision and control loop frequency to local controllers. Optimizations across the stack are challenging to evaluate and tune especially when the ground truth is not know, e.g., SLAM. To help programmers integrate approximations into their system, we propose to write a book gathering design patterns for approximate computing. The book will contain guidelines about when to use approximation, how to use it, what the caveats are. Beyond performances, the book will also discuss the impact of approximation on aspects such as robustness and security.

4.5 Intermediate Representations and Tool Flows for Approximate Computing

Andreas Gerstlauer (University of Texas at Austin, US)

With a large number of design knobs when applying approximations across the compute stack, approximation-aware design automation solutions and design tools will be indispensable in navigating associated design spaces. This will require combining approximations across multiple abstraction levels into integrated cross-layer tool flows. Many approximation techniques at higher levels of abstraction are inherently application-specific. Furthermore, tool flows that are generic to span across application areas have proven to be too complex and infeasible to develop. Instead, domain-specific tool flows have been successfully applied in many key application areas, such as TensorFlow- or PyTorch-based flows in machine learning. Such flows are built around domain-specific languages (DSLs) and domain-specific intermediate representation (IRs), which often take the form of more functional-oriented programming models. On top of such high-level domains-specific IRs, various source-level optimizations, including domain-specific approximations such as neural network pruning are then applied. Domain-specific approximations will also need to account for the inherent dependency of approximations on application-specific inputs and input distributions. At the same time, there are a range of implementation-dependent approximation techniques, such precision scaling or code transformations that are target-specific but general across domains. It is desirable to implement such target-aware optimizations in a common implementation back-end that can be shared across different domain-specific tool flows. We envision tool flows that combine various domain-specific front-end IRs feeding into a common back-end IR for compilation, synthesis and implementation on different software and hardware targets (Figure 1). Such back-end IRs will likely take a more target-specific imperative form. They will need to support back-end approximations using appropriate domain-specific quality models coming from the top as well as target-specific cost models coming from the bottom. Various compiler and high-level synthesis IRs exist, but they are predominantly based on sequential software models that are a poor fit for custom hardware targets and associated

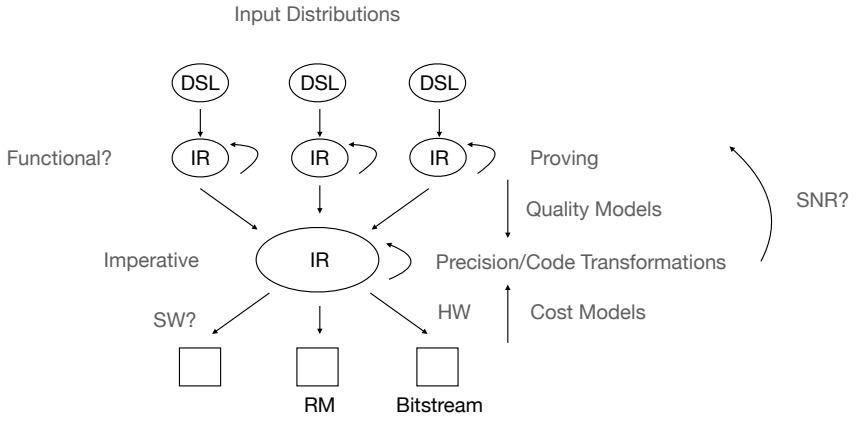

Figure 1 Approximation-aware cross-layer tool flows.

hardware approximations. Some efforts, such as the Calyx or HPVM projects at Cornell and UIUC are underway to develop new hardware- and approximation-aware IRs. However, complete cross-layer tool flows that combine domain-specific front-end with target-specific back-end approximations are still lacking and require further research.

4.6 Differentiation of Error Models

Andreas Burg (EPFL – Lausanne, CH)

The working group discussed the need for a careful and rigorous differentiation between different types of errors that are considered under the approximate computing paradigm. In fact, erroneous or approximate behaviour of a circuit or implementation can originate from different causes or origins at different stages of the life-cycle of a design (from the design stage, to the manufacturing, to the operation in the field). Such different origins require fundamentally different error models that must not be confused neither when assessing the modelling and impact of errors, nor or when considering suitable mitigation measures on circuit, architecture or algorithm levels, at design time or during operation. On the one hand, approximate computing paradigms often introduce errors intentionally at design time, for example at the algorithm level or through approximate arithmetic components. Such errors are deterministic in nature and perfectly known. Hence purely stochastic assessment of their impact and forward error correction for mitigation are not immediately applicable. However, the frequently used and technically inaccurate modelling as stochastic errors may still be useful to provide "compact" quality metrics (avrg, variance, ...) across large random data sets. Such deterministic errors (e.g., introduced at design time) may also be seen as related to source coding, while coding efficiency is not necessarily measured in a reduction in number of bits. Such a view may provide new insights, but need further consideration. On the other hand, errors or parametric variations introduced for example during manufacturing are not known at design time, but still deterministic after manufacturing. Hence, stochastic modelling must be done with care since every realization of the design is ultimately different in its (deterministic) behaviour (non-ergodic behaviour), which requires a yield analysis

and sophisticated test methodologies to identify dies with degraded quality. Finally, some sources of error such as single-event upsets and external noise (e.g., on the power supply) are both unknown and stochastic. Hence, assessment of their impact with deterministic models is not feasible, but in turn stochastic models for impact analysis and error mitigation measures such as error correction apply. Finally, joint source-channel coding ideas could address deterministic approximations and random errors jointly, but further elaboration is also here necessary.

In the second part of the discussion, the group discussed error models for DNA storage and potential options for error control coding. The main issue with DNA storage is capturing the effect of erasures and insertions which render conventional codes not immediately applicable. Work-arounds exist in the literature, but research is still in its infancy.

4.7 Challenges for Approximate Hardware

Georgios Zervakis (KIT – Karlsruhe, DE)

Approximate hardware design forms a very promising solution to boost the efficiency of Domain Specific Accelerators. For example, Samsung already uses approximate multipliers in some of its DSPs while the conventional today 8-bit fixed-point inference accelerators can be viewed as an approximation of the traditionally used single floating-point representations. Nevertheless, embracing approximate circuits for general purpose computing appears less promising or not mature yet. To design approximate circuits, algorithmic approximations seem to deliver better solutions. One of the main reasons is that they can be better supported by the EDA tools. The major deficiency identified in the design of approximate accelerators is to understand how errors, with respect to both inputs (sensing) or computation (approximate units), propagate throughout the application. In addition, errors are input dependent but existing error compensation/balance techniques are mainly based on statistics and cannot always guarantee better accuracy. Mixed approximation or reconfigurable approximation kernels are required along with approximation techniques that force error cancellation throughout the different approximated computations. Moreover, approximate circuit verification and large system simulation when using approximate accelerators still remain open issues. Significant research has focused on arithmetic units and small accelerators. However, the overall system performance or gains is unclear and a systematic methodology to translate the gains and error or the approximate circuit to system gains and quality is required. Finally, two propositions were made: i) examine approximate design for gain in other metrics such as security and fabrication cost and ii) use analog computations for near sensing application and combine approximation in the analog domain with approximation in the digital domain.

Participants

- Hussam Amrouch
Universität Stuttgart, DE
- David Atienza Alonso
EPFL – Lausanne, CH
- Eric Atkinson
MIT – Cambridge, US
- Andreas Burg
EPFL – Lausanne, CH
- Eva Darulova
MPI-SWS – Kaiserslautern, DE
- Lara Dolecek
University of California at Los Angeles, US
- Babak Falsafi
EPFL – Lausanne, CH
- Djordje Jevdjic
National University of Singapore, SG
- Debasmita Lohar
MPI-SWS – Saarbrücken, DE
- Jürgen Teich
Universität Erlangen-Nürnberg, DE
- Damien Zufferey
MPI-SWS – Kaiserslautern, DE

Remote Participants

- Sara Achour
Stanford University, US
- R.Iris Bahar
Brown University – Providence, US
- Swarnendu Biswas
Indian Institute of Technology Kanpur, IN
- Peter Dueben
ECMWF – Reading, GB
- Andreas Gerstlauer
University of Texas at Austin, US
- Ghayoor Gillani
University of Twente, NL
- Jie Han
University of Alberta – Edmonton, CA
- Anastasiia Izycheva
TU München, DE
- Vijay Janapa Reddi
Harvard University – Cambridge, US
- Gauri Joshi
Carnegie Mellon University – Pittsburgh, US
- Ulrich Kremer
Rutgers University – Piscataway, US
- Sasa Misailovic
University of Illinois – Urbana-Champaign, US
- Laura Monroe
Los Alamos National Laboratory, US
- Sri Parameswaran
UNSW – Sydney, AU
- Adrian Sampson
Cornell University – Ithaca, US
- Olivier Sentieys
University & INRIA – Rennes, FR
- Phillip Stanley-Marbell
University of Cambridge, GB
- Radha Venkatagiri
Oregon State University, US
- Norbert Wehn
TU Kaiserslautern, DE
- Georgios Zervakis
KIT – Karlsruher Institut für Technologie, DE