Proceedings of the First International Workshop on Post Moore's Era Supercomputing

Editors

Jeffrey S. Vetter
Satoshi Matsuoka
Proceedings of the First International Workshop on Post Moore’s Era Supercomputing

Co-located with SC16
Salt Lake City
http://j.mp/pmes2016

14 Nov 2016

in cooperation with the IEEE Computer Society
# Contents

PMES Overview

1 Probabilistic Computing for HPC in the Post-Moore's Era. Laura Monroe, John Daly, Nathan Debardeleben, Sarah Michalak, Qiang Guan and Kevin Rudd. 3


3 Unum Arithmetic: Better Math with Clearer Tradeoffs. Nicholas Brummel, John Gustafson, Andrew Klofas, Carlos Maltzahn and Andrew Shewmaker. 9

4 Silicon Photonics in Post Moore's Law Era: Technological and Architectural Implications. Ke Wen, Sébastien Rumley, Payman Samadi, Christine P. Chen and Keren Bergman 12

5 Re-form: FPGA-powered true codesign flow for high-performance computing in the post-Moore era. Franck Cappello, Kazutomo Yoshii, Hal Finkel and Jason Cong. 15

6 Pipelined Intermediate Fabrics for Parallel Execution. Hiroka Ihara and Kenjiro Taura. 19

7 Heterogeneous Computing – A Path to Post-Moore Supercomputing: Architecture, Circuits, and Process. Wayne Burleson, Shomit Das, Yasuko Eckert and Nam Sung Kim. 22

8 Neuromorphic Computing: A Post-Moore’s Law Complementary Architecture. Catherine Schuman, J. Douglas Birdwell, Mark Dean, James Plank and Garrett Rose. 25

9 A Hafnium-Oxide Memristive Dynamic Adaptive Neural Network Array. Gangotree Chakma, Mark Dean, Karsten Beckmann, Harika Manem, Nathaniel Cady and Garrett Rose. 28

10 Modeling of Novel Transistors, Manufacturing Technologies, and Architectures to Preserve Digital Computing Performance Scaling. George Michelogiannakis, David Donofrio and John Shalf. 31

11 Finding Balance in the post-Moore's Law Era. Jeffrey Young and Richard Vuduc. 34


13 Quantum Computing and Moore's Law Are Becoming Entangled. Steven P. Reinhardt, Murray Thom and Mark W. Johnson. 40

14 Comparing Sparse Representations for Object Classification Generated on the Quantum D-Wave 2X with Conventional and Neuromorphic Solvers. Nga Nguyen, Brian Van Essen and Garrett Kenyon. 43
15 Experiences with Scheduling Problems on Adiabatic Quantum Computers. Tobias Stollenwerk and Achim Basermann.
Overview of the First International Workshop on Post Moore’s Era Supercomputing (PMES)

This interdisciplinary workshop is organized to explore the scientific issues, challenges, and opportunities for supercomputing beyond the scaling limits of Moore’s Law, with the ultimate goal of keeping supercomputing at the forefront of computing technologies beyond the physical and conceptual limits of current systems. Continuing progress of supercomputing beyond the scaling limits of Moore’s Law is likely to require a comprehensive re-thinking of technologies, ranging from innovative materials and devices, circuits, system architectures, programming systems, system software, and applications.

The workshop is designed to foster interdisciplinary dialog across the necessary spectrum of stakeholders: applications, algorithms, software, and hardware. Motivating workshop questions will include the following. "What technologies might prevail in the Post Moore’s Era?" "How can applications effectively prepare for these changes through co-design?" "What architectural abstractions should be in place to represent the traditional concepts like hierarchical parallelism, multi-tier data locality, and new concepts like variable precision, approximate solutions, and resource tradeoff directives?” “What programming models might insulate applications from these changes?”

Experts from academia, government, and industry in the fields of computational science, mathematics, engineering, and computer science will have the opportunity to participate in the workshop as a presenter, panelist, or audience member. Invited speakers will provide insights and challenges from their disciplinary perspectives, while peer-reviewed position papers on promising ideas will be presented to facilitate community interaction and diversity. Panel sessions will provide opportunities for interactions across disciplines and provocative questions from the audience.

Paper Review Process

We received 42 submissions this year. Due to time limits and space constraints of our workshop at SC16, we could only accept 15 position papers for presentation. Our acceptance rate of 36% was surprisingly tough for this initial PMES workshop. The papers went through a review process involving at least three (3) reviews for all papers, online discussion to rank papers, and a final, second-round discussion to conclude the set of papers that got accepted.

During the workshop, speakers of all accepted papers were given a 15 minute presentation slot, followed by a short panel consisting of multiple presenters.

As originally described, we distribute the position papers to attendees as a compiled technical report, so that authors retain the copyright to their papers.

Workshop Organizers

Workshop Co-Chairs

- Prof. Satoshi Matsuoka (Tokyo Institute of Technology)
- Dr. Jeffrey S. Vetter (Oak Ridge National Laboratory)

Program Committee

- Prof. Keren Bergman (Columbia)
- Prof. Tom Conte (Georgia Tech)
- Dr. Erik Debenedictis (Sandia National Laboratory)
- Prof. Franz Franchetti (Carnegie Mellon University)
- Prof. Koji Inoue (Kyushu University)
• Prof. Takeshi Iwashita (Hokkaido University)
• Prof. Kengo Nakajima (University of Tokyo)
• John Shalf (Lawrence Berkeley National Laboratory)
• Prof. Richard Vuduc (Georgia Tech)
• Prof. Gerhard Wellein (Erlangen)
Probabilistic Computing for HPC in the Post-Moore’s Era

Laura Monroe, John Daly, Nathan DeBardeleben, Sarah Michalak, Qiang Guan and Kevin Rudd

Abstract—Probabilistic computing is likely to be applicable to a wide range of post-Moore’s era architectures. The need for this form of computation is coming in the near future, because of reliability issues, and computing this way may bring many advantages, including power savings and increased resilience to faults. To date, however, the major examples have been special-purpose machines. We discuss in this position paper paths forward for more general use of this technology.

1 INTRODUCTION

Probabilistic computation is computing that relies upon some form of probabilism. This can be in hardware, where some element of the device incorporates a random or pseudo-random element that is used for computation. This may also be in software, where the algorithm depends on a random or pseudo-random input to calculate.

Hardware-based probabilistic computing may be purpose-built, as in probabilistic chips aimed at image processing or error-correction. Chips with many transient faults might also be considered probabilistic devices, in that the faults occur randomly, with some distribution, so one may treat the randomness introduced as a form of probabilism in itself. Purpose-built probabilistic devices have to this point been designed for a single specific need, aimed at tolerant applications like image processing.

Probabilistic computing has shown great promise in energy savings, for calculations that do not depend on exact and correct answers. Probabilistic computing also addresses the increase in faults anticipated because of decreased feature size and decreased voltage.

However, to be truly revolutionary in high-performance computing (HPC), some combination of general probabilistic and specialized probabilistic processors will have to be developed for use in the HPC domain. We present here a more integrated approach to probabilistic computation incorporating faulty general-purpose processors, and discuss some challenges in the use of probabilistic systems for HPC.

2 STATE OF THE ART

2.1 Historic Roots of Probabilistic Computing

Probabilistic approaches go back to the early days of computing, to von Neumann [16] and De Leeuw, Shannon et al [5].

2.2 Current Status: Reliability and Specialized Designs

We now find ourselves in a similar position to the early computer scientists. Although devices are still very reliable, their reliability is expected to decrease due to decreased feature size [10]. In addition, voltage threshold reduction techniques are on vendor roadmaps now for reasons of power savings [11]. Although the goal is to keep reliability constant, the possibility of power savings through reduced reliability addressed by probabilistic near- and sub-threshold techniques is attractive.

There have been several specialized probabilistic chips at the level of working prototypes developed in the past decade. These have been built for specific needs and have been aimed at performance and power savings. These include an image-processing chip developed by Chakrapani et al [2], a probabilistic error-correcting chip for flash memory developed at Lyric Labs [15], and the ensemble of devices in development for image-processing for the DARPA UPSIDE program [4].

2.3 Maturity Level

Probabilistic computing is attractive because of the large power savings seen on early and prototype devices. It also addresses the expected decrease in reliability, anticipated over the next decade. However, in order to be effectively employed for HPC it will be necessary to adapt traditional, deterministic application techniques to non-deterministic hardware. Our position is that such adaptation, while challenging, is not insurmountable.

In 2012 a series of inter-agency resilience workshops specifically considered the impact of hard and soft errors on applications of interest to scientific computing. The result of these workshops was perhaps surprising when participants
concluded that many applications already provide their own internal consistency checks against errors in computation. Even more surprising was the observation that for all real algorithms of interest to the scientific community no examples could be found of a class of algorithm for which internal consistency checks did not exist or could not be developed. [3]

Inability to verify the correctness of a computation could make probabilistic techniques a non-starter for most scientific applications. However, with that major hurdle circumvented, at least for the moment, researchers have begun to reexamine the potential for implementing HPC applications on non-deterministic hardware [12] [13] [7].

3 Path Forward for HPC Probabilistic Compute

We see three possible paths for deployment of probabilistic computing into the HPC ecosystem. We make the case for a middle path, incorporating the best from general-purpose approaches and from more specialized designs.

3.1 A General-Purpose Probabilistic Processor

One approach would be the development of a general-purpose probabilistic processor. This would need to provide power or performance improvements over a deterministic processor. Otherwise, why use a non-deterministic approach when deterministic works just as well?

Alternatively, this might include the “faulty chip” approach, with improved methods of possibly probabilistic error correction. This approach would be a general-purpose chip as in use today, except with less reliability.

3.2 Yet More Specialized Probabilistic Designs

Another approach is that of specialization and accelerators. The devices designed to date have been specialized chips. One immediate technical issue is that of mapping important HPC problems onto probabilistic chips. There is also a cost/benefit question. It is quite costly to design and deploy such a thing. Not only must there be a technical path forward for the given special need, but as for any specialized design, the purpose must be important enough to justify the cost.

3.3 A Proposed Path

A third alternative would be a middle path: a somewhat faulty general purpose main system probabilistically addressing error, but including specialized probabilistic hardware and software addressing some set of identified needs. Research and development of these two paths can take place in parallel, with integration taking place upon maturity of given technologies.

3.3.1 The General-Purpose System

We propose a general-purpose framework composed of less-reliable general-purpose processors. By relaxing reliability constraints on the design, there will be some energy conservation.

Because reliability will become more of an issue as the end of Moore’s Law approaches, the community will either need to correct for such faults consistently, or develop algorithms that tolerate error. Research is in progress examining the ways basic algorithms may perform probabilistically, and the results have been surprising. For instance, early results on multi-precision integer calculations with problems like greatest common divisor (GCD) indicate that discrete math may exhibit a surprising degree of intrinsic fault tolerance [12]. Even basic integer and floating point operations may have inherent mathematical fault tolerance [13] [7].

Reliability issues will emerge with smaller feature size, but voltage-reduction approaches could lead to even more of these issues. Certainly, it is preferable to maintain the high level of reliability we have now, but if some reduction can save more power, this is worth attempting.

3.3.2 The Specialized Accelerators

The design of the specialized accelerators depends on the identification of specific HPC applications that might benefit from a probabilistic approach. If these are closely enough related to an existing probabilistic design and application, that work could even be leveraged for use in the HPC environment.

One example might be data analysis after computation of one or more timesteps. This is related to the image analysis that has been successfully implemented on existing prototype probabilistic chips. Another example might be an approximate computing chip [6] [9], that is abstemious in energy usage while still producing results that are “good enough”.

4 Challenges

This is a rather radical way of computing, and it comes with challenges. Here we address the challenges of use, and not the challenges of technical development of probabilistic hardware.

- This is a different way of thinking about computing. Computing is often presented as a deterministic process that calculates a “correct” answer. In contrast to this, probabilistic computing will not always give the right answers, or even consistent answers across runs. How much does one care about the “right” answer, and why?
- Things need to work. It is not enough to provide a faulty system without some way to get past the faults. Where do faults manifest and how? Can they be quantified? Can a mathematical theory be developed to address them? Can the application tolerate some amount of fault?
- What does it mean “to work”? We need to have a rigorous understanding of good enough. Does this mean perfect correctness? Faults that are (mostly) correctable? Faults that are characterized and tolerable? Algorithms that are close enough? What does close mean (in mathematically rigorous terms)? This may be domain-specific.
- We need to understand fault models. How do hardware failures manifest, specifically and statistically? We need this so we can design algorithms for general faulty processors.
- We need to be able to program these systems. What are the programming models we will use? How might one hide the probabilism to make it easier to program? Is reproducibility important? How is debugging accomplished, in the absence of reproducibility?
- We need common applications or algorithms that profit from the power and performance advantages of probabilistic computing.

5 Conclusion

Probabilistic computing shows great promise for power savings. This is also a mode of computing that is likely to be needed in the future. We have discussed an integrated approach to probabilistic computing for HPC applications. Many research challenges remain, but we believe that they are tractable and that the advantages presented by probabilistic computing make them worthwhile to address.
REFERENCES


An Energy-Efficient Physical Platform for Solving Differential Equations

Jaeha Kung, Yun Long, Student Member, IEEE, and Saibal Mukhopadhyay, Senior Member, IEEE

Abstract—We advocate the design of an energy-efficient physical platform for solving differential equations. Our approach crosscuts dynamic system based computing model, digital and mixed-signal hardware architecture, and CMOS/post-CMOS devices to achieve this goal. The preliminary results show the promise of significant gain in performance and energy-efficiency.

1 SUPERCOMPUTING NEEDS
Solving coupled partial differential equations (PDE) is a basic operation in scientific computing and complex system analysis [2], [8], [12]. Various hardware approaches were presented to accelerate coupled PDE solution including analog ASICs [9], [11], FPGA-based coprocessor [1], [10], [19], and GPU [3], [7]. The CPU/GPU based approaches provide the advantage of run-time programmability while analog ASICs provide better efficiency but lack of programmability. The FPGA acceleration of numerical algorithms can improve efficiency over GPU; but can only be programmed off-line to represent a specific equation.

2 PROPOSED TECHNOLOGY
We are advocating a programmable physical (hardware) platform for solving coupled PDEs and ODEs. Our approach crosscuts across computing model, hardware architecture, and device physics to achieve this goal (Fig. 1). The computing model is based on dynamical system such as Cellular Nonlinear Network (CellNN) [4], [14], [15]. The hardware architecture, will be optimized to realize the CellNN model and can be designed as a digital, mixed-signal, or in-memory processing platform. The physical platform can be realized using CMOS and post-CMOS devices such as RRAM or memristor (Fig. 1). The programming includes direct mathematical transformations and/or training to map an equation to the CellNN model. Once mapped, the hardware will directly solve the equation for a given set of initial and boundary conditions.

Novelty: As of our approach physically realizes a computing model we expect to improve energy-efficiency over conventional CPU/GPU. On the other hand, as our approach is rooted on a generic computing model, not on designing an analog circuit for a specific equation, the platform is programmable. We expect the proposed platform to provide two to three orders of magnitude improvement in energy-efficiency over state-of-the-art software programmable platforms like a GPU.

3 TECHNICAL DETAILS AND RESULTS
The computational model of CellNN is based on a two dimensional array of cells where each cell follows an ODE based dynamics [4], [14], [15], [16]. Each cell in CellNN is connected to (local) neighboring cells resulting in a system of coupled ODEs. The weight of local connections, defined as the templates, defines the coupling and hence, the nature of the system of the equation [6], [13], [17], [18].

3.1 Computational Mapping
A multilayer CellNN with linear and nonlinear templates is used to map DEs. The goal of the mapping process is to determine the template values. The number of layers is a function of the number of variables involved and the order of derivatives for each variable in the system (Fig. 1). Each layer represents the space-discretization of each equation with coupling defined by finite difference method (linear templates) (Fig. 2) [5], [13]. The coupling between the CellNN layers represents the interactions of different variables (nonlinear template). There can be a multivariate function, a function with several state variables, involved in each equation. This nonlinear function is approximated by series expansion methods such as Taylor series to map to a nonlinear template (Fig. 2).

3.2 Hardware Architecture
In this position paper we show the examples of CMOS-based digital and RRAM-based mixed-signal architecture.

Digital Architecture: Fig. 1 shows the architecture of a digital CellNN accelerator. A network-on-chip is considered where each router node is connected to a memory channel and a group of processing elements (PE). In each PE, there is computing unit with a local buffers to store weights and partial data. To implement it as a generic multilayer CNN accelerator, PEs are grouped in program-time to cover specific layer and partial input data. Router nodes are also used when there is a need for inter-/intra-layer data transfer.

Mixed-signal Accelerator using RRAM: The basic dynamics of the CNN cell can be realized using analog CMOS circuits. The template weights can be realized using CMOS operational transconductance amplifiers. However, the complexity of the design can be reduced using analog RRAM as the synaptic weight as shown in Fig. 1. After applying voltage, current flowing through RRAM devices in the same row sums together and then generates output voltage. With external input and feedback, different state variables can be coupled together to perform computation.

3.3 Simulation Results
Our initial analysis considers programming the CellNN architecture on four DEs: (1) PDEs with linear templates: heat...
Fig. 1. The basic concept of the physical DE solver using CellNN based computing model.

Fig. 2. The illustration of the mapping process and templates for example case studies with only linear and linear/nonlinear templates.

with no real-time weight update, the RRAM platform shows better performance. However, in examples with nonlinear templates with real-time weight update (e.g. RD equation), the performance of the RRAM platform drops due to the relatively higher programming time of RRAM.

4 Future Work

This position paper advocates the use of physical solvers based on dynamic system, specifically, Cellular Nonlinear Network, to accelerate solution of coupled PDEs. The initial results are encouraging. However, significant research is necessary to develop a generic, automated mapping process; develop optimal digital and mixed-signal architecture considering the effect of memory bandwidth; and exploring various post-CMOS devices. A key challenge is to characterize the interaction of algorithmic approximation and physical noise from device/circuit; and perform cross-layer optimization to improve accuracy, performance, and energy-efficiency.
REFERENCES


Unum Arithmetic: Better Math with Clearer Tradeoffs
Nic Brummel, John L. Gustafson, Andrew Klofas, Carlos Maltzahn, Andrew Shewmaker

Abstract—To raise performance beyond Moore’s law scaling, Approximate Computing reduces arithmetic quality to increase operations per second or per joule. It works on only a few applications. The quality-speed tradeoff seems inescapable; however, Unum Arithmetic simultaneously raises arithmetic quality yet reduces the number of bits required. Unums extend IEEE floats (type 1) or provide custom number systems to maximize information per bit (type 2). Unums achieve Approximate Computing cost savings without sacrificing answer quality.

1 INTRODUCTION
Approximate Computing encompasses a range of ideas as intended to reduce numerical quality in return for higher speed [1]. Such techniques include aggressive clipping of trailing fraction bits, alternative math libraries that compute to lower accuracy with fewer clock cycles, hardware that does not propagate a carry bit when adding, and so on. All of these ideas presume that better answers and higher computing costs go hand-in-hand. In early 2015, the unum (universal number) data type was introduced as an extension of IEEE floats [2]. Unlike floats, it does not round, overflow to ±∞, or underflow to 0, but instead uses a single bit at the end of the fraction field (the ubit) to indicate if a number is an exact float or in the open interval between floats. The improved mathematical behavior allows unums to be safely used down to precisions as low as four bits, and they adjust in precision one bit at a time, up to thousands of bits. In some applications, unums provide tightly bounded answers using fewer bits than floats. In contrast with approximate computing, they offer a way to simultaneously improve answer quality while reducing the number of bits needed.

That form of unum is now referred to as type 1; the “unum type 2” completely breaks from IEEE floats to design a system for representing real numbers (and sets of real numbers) that maximizes information per bit with a Shannon-like maximum entropy approach [3]. A user (or a compiler) can customize the number system to fit the requirements of an application for accuracy and dynamic range, which allows unprecedented answer quality from a reduced number of bits.

2 TECHNOLOGY DESCRIPTION
Conventional floats have three bit fields: sign bit, exponent, and mantissa. We prefer the term “fraction” for mantissa. The IEEE Standard allows a choice between 16, 32, 64, 80, and 128 precisions, but for any precision, the Standard fixes the number of exponent and fraction bits, resulting in inefficiency for any particular application.

Type 1 unums allow adjustable exponent and fraction sizes at the bit level, all the way down to a single bit. Fig. 1 shows an example of the economization typical of a unum, in this case representing Avogadro’s number.

IEEE 754 Standard 64-Bit Float representing 6.022x10^{23}

<table>
<thead>
<tr>
<th>sign</th>
<th>exponent</th>
<th>fraction</th>
<th>ubit</th>
<th>size</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1100110</td>
<td>11111111100000</td>
<td>1111011</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 1. IEEE Format versus Unum Type 1 Format

Unums offer the same trade-off versus floats as variable-width versus fixed-width typefaces: Harder for the design engineer and more logic for the computer, but superior for everyone else in terms of usability, compactness, and overall cost. Many examples are shown in [2]; here is a new example designed to stress destructive cancellation, a common source of accuracy loss in application programs. It demonstrates that economy need not reduce accuracy or dynamic range: With a budget of 32 bits per number, compute

\[ x = \left( \frac{27}{10} - e \right)^{67/16} \]

starting from computer representations of the integers, \( \pi \), and \( e \). The correct value to seven places is 302.88274.

IEEE 32-bit binary floats, with a dynamic range of over 2.8x10^{83} and a nominal precision of 7.2 decimal digits, produce the answer 302.91244, correct to only three digits. In terms of Units in the Last Place (ULPs), the answer is off by 973 ULPs, even though the expression only requires 9 operations. The format provides no warning to the user regarding this loss of accuracy.

Approximate arithmetic, where (for example) we improve hardware speed by using only 29 bits instead of 32 and truncate numbers instead of rounding, produces the answer 303.04124, which is wrong by about 0.297. Now only the first two digits are correct.
Interval arithmetic values \([a, b]\), where \(a\) and \(b\) are IEEE 16-bit floats and thus fit in the 32-bit budget, offer the hope of a rigorous bound on the answer, a sort of automatic error analysis. Each endpoint has only 3.3 decimal digits of precision and a dynamic range of \(1.0 \times 10^{32}\), a considerable sacrifice of computing vocabulary for the benefit of a bound. It produces the very uninformative answer that \(x\) lies in the interval \([18.21875, 33056.]\), a bound far too loose to have practical value.

Unum arithmetic (Type 1) has adjustable ranges; with three bits for the exponent field length and five for the fraction field length, unums have a precision of up to 9.9 decimals and a dynamic range of \(2.4 \times 10^{96}\), quite a bit more than offered by 32-bit floats. The number of bits per unum varies from 12 to 49 as needed. If we cap the range at 36 bits maximum, which keeps the average size per unum below 32 bits, we get the result that \(x\) is provably inside \((302.75, 303)\). The average unum size of about 29 bits saves a modest amount of storage and bandwidth but providing a reasonably accurate answer that also conveys invaluable information of an absolute bound on the result. Notice that the Approximate Arithmetic example above gives an answer that lies outside this rigorous bound.

There are many other examples for which unums provide bounds that are tighter than the inaccuracy of floats, using fewer bits. Unums are not restricted to applications that tolerate approximate computing techniques.

3 SC Challenges Addressed

A unum environment will simultaneously increase programmer productivity and make hardware more cost-effective by addressing the following challenges:

1. Mostly-automatic numerical error analysis
2. Higher performance on bandwidth-limited codes
3. Power/heat-dissipation reduction
4. Better accuracy with less storage requirements
5. Numerical instability made visible to user
6. Elimination of roundoff errors that are indistinguishable from coding errors
7. Bitwise identical results across platforms
8. Parallelization of codes without numerical effects

4 Novelty

Unum arithmetic is less than three years old; some public descriptions were made prior to the publication of primary text on the subject in February 2015, but it is quite recent that a viable alternative to the 1980s-era float format has been found and tested extensively. A groundswell of activity has started to develop libraries and convert languages and applications to use the new format.

Type 2 unums are even more novel, having been developed only as of February 2016. MIT researchers recently announced a Julia language prototype for type 2 unums [5]. Julia appears to be the ideal language for unums.

Some quotes from Amazon.com reviews of [2] indicate the level of novelty of the idea:

"...a bold and brilliant proposal for a revolutionary number representation system, unum, for scientific (and potentially all other) computers. ...The book is a call to action for the next stage: implementation and testing that would lead to wide-scale adoption." — Gordon Bell

"...a sketch of what perhaps might be the future of computing." — Ulrich Kulisch, U. of Karlsruhe

"...an extraordinary reinvention of computer arithmetic and elementary numerical methods from the ground up... These changes are not just marginal technical improvements... they lead to what can only be described as a radical re-foundation of elementary numerical analysis with new methods that are free of rounding error, fully parallelizable, fully portable, easier for programmers to master, and often more economical of memory, bandwidth, and power than comparable floating point methods." — David Jefferson, LLNL

5 Maturity

Software for unums is maturing rapidly; multiple unum environments now exist in C, C++, Julia [4], and Python. The Python and Mathematica environments are particularly complete environments for type 1 unums. The authors are presently creating an open source C repository for type 2 unums. DARPA has funded unum development efforts and has shown a strong interest in funding application development based on unum arithmetic.

Native hardware is only in the planning stages, making it difficult to benchmark unums against floats. REX Computing has stated plans to support unums in a future version of its Neo chip. The LULESH proxy code at Lawrence Livermore National Laboratory has been ported to use unums instead of floats.

Outgoing IEEE President Tom Conte has urged the creation of an IEEE Standard for the unum data type; the inventor of the format believes this is premature.

6 Risks

Because unums are a superset of floats, they can already run any existing program that currently works with floats. The only downside of using them in that way is that they might offer no improvement but slightly increase instead of decrease the bits required.

When used as bit-efficient interval bounds, there is a risk of falling into some of the pitfalls of traditional interval arithmetic that result in bounds too loose to be useful. A variety of ways to avoid these pitfalls are described in [2], but much more experimentation is needed to know whether tight bounds can always be achieved without expertise.

7 Conclusion

As HPC attempts ever more ambitious tasks with ever less access to people with numerical analysis expertise, approximate computing may be a step in the wrong direction. A result provably bounded to five decimals may have far more value than a guess that displays 15 decimals, all of which might be wrong. The laudable goal of using fewer bits per number can instead be achieved through new formats that maximize information per bit, and that allow the computer to bear more of the numerical analysis burden.
REFERENCES


Silicon Photonics in Post Moore’s Law Era: Technological and Architectural Implications

Ke Wen*, Sébastien Rumley, Payman Samadi, Christine P. Chen, Keren Bergman

Department of Electrical Engineering, Columbia University, New York, *kw2501@columbia.edu

1 A RIPPLE EFFECT FROM CACHE REDUCTION

Moore’s law is ending as we enter the last years of shrinking transistors. Chip designers will thus have to use the available transistors more effectively. One may interpret a few already signs of this trend. As shown in Figure 1, area devoted to cache on a CPU chip is decreasing, both in terms of (a) MB per FLOPS and (b) normalized chip area -- cache size (MB) × features size (nm2) / die size (mm2). Especially, a sharp fall (Fig. 1a) is clear as the industry gets into the many-core era (around 2013). Interestingly, this cache cliff matches the time when Moore’s Law was said to be dead in the economic sense -- starting from 2013, the number of transistors bought per dollar has stayed stagnant [1]. The fact that chipmakers are willingly trading the cache area for more FLOPS, along with the rise of data-centric throughput computing [2], calls for significantly higher off-chip memory bandwidth. Fig. 1c shows this trend: the sharp increase of the off-chip memory bandwidth matches the cache cliff of Fig. 1a. This increase, however, is still not enough to balance the FLOPS increase as the bytes per flop ratio continues to drift away from the ideal point.

There is more to this grim description. The memory bandwidth increase is also rapidly stressing the pin count limit of the processor package. For example, KNL requires 3647 pins in the socket, plus 1024 pins in the interposer for each of the eight on-package memory stacks. The pin density of standard chip package, however, cannot scale indefinitely. The ever-increasing bandwidth demand thus requires a more efficient chip I/O technology for processors beyond the Moore’s Law.

2 SILICON PHOTONICS FOR CHIP I/O

Compatible with CMOS lithography fabrication, Silicon photonics (SiP) has become one of the leading solutions to the aforementioned chip I/O issue. An example of “extending the power of silicon to new arenas” [3], SiP leverages the transparency of silicon to light with 1.2~5 μm wavelength for high-speed transmission. Each SiP waveguide can support terabit/s bandwidth, orders of magnitude higher than what can be achieved with conventional electrical I/O. For example, while an 8-channel (4-layer) High Bandwidth Memory (HBM) cube requires a 1024-bit bus for 100 GB/s, a single SiP waveguide can provide the same bandwidth with 32 wavelengths each at 25 Gb/s.

Silicon photonic is compatible with silicon interposers used to carry processor and memory chips, forming a high-bandwidth chip-to-chip interconnect on package (Fig. 1d). Components such as waveguides, modulators, photodetectors and switches can be directly fabricated on the silicon interposer with low cost. The SiP switch, controllable by the processor, can provide flexible and transparent connection between any memory stack and any processor interface. A SiP interposer fabricated by PECST of Japan was reported to achieve bandwidth density of 6.6 Tb/s/cm² [4].

Another important aspect of SiP is extending high-bandwidth I/O off package, enabled by efficient coupling

![Fig. 1. (a) Cache size normalized by GFLOPS; (b) Normalized cache area; (c) off-chip bandwidth; (d) SiP interposer based architecture; (e) SiP-enabled high-capacity HBM-based node; (f) alleviating hotspot; (g) optimizing core-memory affinity; (h) Flexfly network.](image-url)
between waveguides and fibers. This is a much-needed capability, as the interposer area (about 700 mm²) will limit the capacity of on-package memory (OPM). The current solution is to pair the fast OPM with slow, off-package, DRAM. Such small-fast, large-slow exclusiveness may significantly complicate application programming and memory management. The distance-independent transmission of photonics can solve this problem, enabling a uniform, high-capacity HBM architecture as shown in Fig. 1e. With 1 Tb/s bandwidth per fiber, four fibers can supply the 256 GB/s bandwidth needed by a HBM2 cube. With 24 fibers per coupling assembly and four such assemblies, an interposer hosting processors can connect to a total of 24 HBM cubes, accounting for 192 GB memory capacity and 6 TB/s aggregate bandwidth. SiP technologies can thus enable a flat, easy-to-manage memory hierarchy.

3 ARCHITECTURAL IMPLICATIONS

3.1 Node Level: Optimizing Memory Locality

The benefit of silicon photonics is not limited to sheer bandwidth growth. As mentioned earlier, the reconfigurable SiP switch can provide connection between any memory cube and any processor interface. This functionality can help precisely deliver memory data to the consumer cores, without traversing the network on chip (NoC), effectively mitigating the NUMA problem faced by the many-core era [5] [6]. As shown in Fig. 1f, a reconfiguration of the SiP switch can reduce the NoC hop count from 10 (dashed yellow, as in native connection) to 1 (solid yellow). This hop decrease immediately translates into a few tens of nanoseconds less latency and a significant drop in energy dissipation. The routing of high-speed memory data out of the NoC plane may also save the NoC bandwidth for more core-to-core communication, a trend as “MPI everywhere” (assigning each core with a MPI process) emerges [7]. SiP waveguides with ultra-low loss of 1.2 dB/m has been demonstrated [8], meaning nearly distance-independent energy consumption for chip scale, as compared to 25 pJ per 64-bits per mm in case of moving data electrically on chip [2].

Another possibility is to use the SiP switch to alleviate the hotspot effect on the NoC when hotspot memory access happens (Fig. 1g). In this scenario, the SiP switch can TDM select the memory interface to inject data stream from the hotspot memory, thus distributing the traffic to different NoC sections [6].

3.2 System Level: Flexible Topology

SiP switching can be also utilized to form flexible system-level topology [9]. The need for flexible topology roots from the diverse spectrum of applications that run in a supercomputer. The clear difference in their communication characteristics, in terms of neighboring relationship, traffic volume, etc., makes it very difficult to find a “best-for-all” topology. SiP switching, in contrast, is capable of dynamically “rewiring” the connections among a set of electronic endpoints. These electronic endpoints can be either compute nodes or electrical routers. The benefit is directing bandwidth to where it is needed without over-provisioning it [10]. Recently, a reconfigurable Dragonfly architecture utilizing small-radix SiP switches has been demonstrated [11]. The architecture, called Flexfly, is capable of concentrating the fully-connected group-to-group links of Dragonfly into, for example, a thick ring-like topology (Fig. 1h). It is shown to help applications like GTC to achieve 1.8x speedup over conventional adaptive UGAL routing.

4 PHOTONIC-ELECTRONIC INTEGRATION

There are three methods for integrating SiP and electronic devices: front-end, back-end and hybrid integrations.

In front-end integration, electronic and photonic devices are formed on the same layer. The advantage is that nanophotonics can piggyback on the mask. However, the challenge remains to guide light with sufficient isolation, especially, separation between the waveguide and the silicon substrate. While CMOS-SOI uses a thin buried oxide (BOX) of 200 nm, photonics SOI requires a BOX of 1 μm. Approaches that utilize thicker-BOX have been proposed [12, 13]. This may, however, reduce the heat dissipation capability of electronics [14]. Methods that do not modify the standard CMOS have thus been proposed [14-17], which locally remove the underlying Si substrate to mitigate losses.

Back-end integration is another monolithic method [18, 19]. It allows deposition of sufficient isolation oxide on top of the existing CMOS-SOI. However, this method introduces additional back-end steps and thus extra cost. The back-end method may also face a stricter thermal budget in order to prevent damage to electronic CMOS. As a result, engineers have to look at using other materials for the photonic layer. Yet, to date, silicon nitride [18], amorphous silicon [20] and laser-annealed polysilicon [18] have been proven as feasible material.

The hybrid integration method forms photonic and electronic circuits on separate chips and bonds them through flip-chip bonding. As such, the photonic and electronic chips can be each optimized using different process flows. Hybrid integration is to date the majority choice of SiP research and development parties. Signaling speeds of 25 Gb/s [4] and 50 Gb/s [21] have recently been demonstrated using flip-chip bonding.

7 CONCLUSION

The end of Moore’s Law comes at a time when efficient allocation of transistor real estate has become imperative for computing. The resulting cache reduction and the rise of data-centric throughput computing calls for efficient off-chip, off-package data movement. Silicon photonics could potentially be one of the promising solutions the computing world is looking for to continue performance growth. Yet, industry-level electronic-photonic integration, and system co-design are yet to be realized, along with reducing manufacturing costs. New architectural implications of silicon photonics at both node level and system level also require further investigation into how to enable new dimensions of performance improvement.
ACKNOWLEDGMENTS

This work was supported by the U.S. Department of Energy Lawrence Berkeley National Laboratory under subcontract 7257488 and Sandia National Laboratories under contract PO 1319001.

REFERENCES


Re-form: FPGA-powered true codesign flow for high-performance computing in the post-Moore era

Franck Cappello, Kazutomo Yoshii, Hal Finkel, Jason Cong

Abstract—Multicore scaling will end soon because of practical power limits. Dark silicon is becoming a major issue even more than the end of Moore’s law. In the post-Moore era, the energy efficiency of computing will be a major concern. FPGAs could be a key to maximizing the energy efficiency. In this paper we address severe challenges in the adoption of FPGA in HPC and describe “Re-form,” an FPGA-powered codesign flow.

1 INTRODUCTION

The performance progress of microprocessors has been driven by Moore’s law, doubling the number of transistors every 18 to 24 months [2]. In the first three decades, every technology generation with doubled transistor density made the transistor switching 40% faster and improved energy efficiency 65%, an effect known as MOSFET scaling or Dennard’s law [9]. With 100 nm or smaller feature sizes, however, the static power or the leakage current became too large to ignore [17], requiring the frequency scaling to stop. After the end of Dennard’s law, multicore design became mainstream in order to exploit transistor density and support more parallelism. Unfortunately multicore scaling will also end soon primarily because of practical power limits [25]. In fact, dark silicon [12] and the utilization wall [25] are becoming a major concern.

The International Technology Roadmap for Semiconductors forecasts sizes of 7 nm in 2020 and 5 nm in 2023, so there is still another decade before reaching the limit of CMOS technology. However, moving toward a new manufacturing process requires significant investments (e.g., multi billion U.S. dollars), and further parametric variations and leakage current may override the performance benefits from advanced technology. Thus, the end of the transistor size scaling may come earlier.

To overcome this scaling issue, researchers are intensively investigating new structures, new materials, new switching technologies and new manufacturing technologies, including tunneling FETs [20], spintronics [3], carbon nanotubes [24], nanoscale vacuum tubes [16], Josephson junctions [15], and single-atom transistors [14]. Emerging 3D integration of CMOS [26] is one of the most promising solutions to extend the Moore’s law, and it will also improve energy efficiency; but it poses several technical challenges such as cost, design complexity, and dynamic thermal variability. Unlike the transition from bipolar to CMOS, however, no technology breakthrough that offers exponential growth is likely to become ready for deployment in the foreseeable post-Moore era.

Another direction is to develop specialized architectures such as Anton [10] and Anton II [23], which were developed by D. E. Shaw for molecular dynamics. A common limitation of these systems, however, is that only few applications will benefit from the special hardware. Clearly, specific architectures require higher nonrecurring engineering costs (NRE).

Modern FPGA platforms with thousands of hardened digital signal processing (DSP) or floating-point blocks are becoming attractive alternatives because of lower overall NRE compared with specialized architectures. Indeed the adoption rate in other information technology domains has clearly accelerated in the past two years, with a number of significant events. One was the public acknowledgment of the use of FPGAs in datacenters by some of the largest Internet service providers, such as Microsoft and Baidu, for a number of latency-sensitive applications, such as search [22] and speech recognition [21]. Another significant event was Intel’s acquisition of Altera, the second largest FPGA company worldwide. These developments indicate that FPGA-based customizable computing is going from advanced research projects into mainstream computing technologies. However, the use of FPGA in scientific computing has been limited for multiple reasons.

In this paper we present our gap analysis of the adoption of FPGA technology in high-performance computing, and we briefly describe “Re-form,” an FPGA-powered true codesign flow, which is at an early stage of development.

2 GAP ANALYSIS

As of this writing, multicore or GPU-based systems dominate in the Top 500 supercomputer list. Reconfigurable computing and FPGAs in particular have not been adopted broadly by the scientific computing community for five main reasons: capability limits, cost, compilation time, programmability, and performance. Concerning capabilities, FPGAs did not feature enough resources (logic cells and DSPs) to compete with CPUs and GPUs of the same generation on floating-point performance. The cost of high-end FPGAs compared with CPUs and GPUs was detrimental. The compilation of a complex C program with high-level synthesis could take tens of hours, drastically impacting productivity and making performance optimization and debugging difficult. Until recently no parallel programming model existed for programming FPGAs for scientific applications. Moreover, compared with CPUs with similar capabilities, reaching high performance on scientific applications with FPGAs required much more programmer time and wider skills (the programmer needs to know hardware description languages).

The two first reasons (capability limits and cost) are rapidly fading as high-end FPGAs SoCs are integrating significantly more resources and are becoming adopted in extreme-scale data centers. The third reason (compilation time) is related to the place and route step that is proprietary in the FPGA tool
chain and represents a significant issue. Programmability and performance are the primary factors still blocking adoption. These are the two major problems that we discuss below.

Programmability: Scientific programmers of large HPC applications cannot code applications or even kernels at the hardware level (with a hardware description language). Typical applications are written in C, C++ or Fortran, exploiting distributed-memory parallelism with MPI and node-level shared-memory parallelism with OpenMP. Moreover, a large portion of HPC codes are being enhanced, or will be enhanced, to use OpenMP4 parallelization and target-offloading directives. However, no production-quality compiler is capable of compiling OpenMP4 codes for FPGAs. FPGA high-level synthesis (HLS) production tools [7], [27] compile ANSI C and C++ (some tools accept other languages), and recent HLS tools can compile codes with OpenCL directives [8]. Other tools translate OpenACC- and OpenMP-like codes into OpenCL codes that HLS tools can compile. The most advanced production-quality OpenMP-like compiler for FPGA, Merlin from Falcon Computing Solutions [6], [13], offers only a limited subset of OpenMP-like constructs. The most advanced OpenMP-like academic compiler (OmpSsS [11] from Barcelona Supercomputing Center) does not provide the same level of performance optimizations compared with Merlin. Thus, a significant gap remains between what the programmers need (OpenMP) and what HLS compilers offer (OpenCL and OpenMP like).

Performance: In theory, programmers can reuse their existing OpenCL codes, developed for CPUs, and generate an FPGA design that runs the codes without requiring any hardware skills. In practice, however, this is not the case, as demonstrated by the paper on Gzip compression [1]. The authors used multiple optimizations at the C code level, including rewriting loop cores to optimize the hardware produced by the OpenCL compiler and the rest of the tool chain. This effort required not only deep knowledge of circuit design and FPGA hardware but also a profound understanding of how the OpenCL compiler and tool chain transforms a given code into an FPGA hardware configuration. Thus, an important gap remains between the optimizations that scientific programmers are used to and the optimizations that FPGA requires.

3 Approach

Figure 1 outlines “Re-form,” our true codesign flow. Our objective is to provide a compiler frontend that accepts OpenMP4-based codes, applies automatic source-level optimizations, and generates an intermediate representation (e.g., SPIR-V, OpenCL) for underlying HLS tools.

Programmability: We leverage many pre-existing FPGA software technologies and components. Because no OpenMP4 compiler yet exists for FPGA, we will first explore and develop a path for the compilation of OpenMP4 applications, reusing the open-source LLVM/Clang compiler [19] infrastructure and vendor OpenCL/C-based high-level-synthesis (HLS) tools. The parsing, semantic analysis, basic code generation, and runtime system for OpenMP4 have been implemented in the LLVM/Clang compiler framework by contributors to the LLVM project. The output of Clang consumed by the optimizer when processing source code with OpenMP4 directives can be logically divided into two pieces: (1) host (i.e., CPU-targeted) code, which calls the OpenMP runtime library to transfer data between the host and the accelerator and to run functions on the accelerator, and (2) target (i.e., accelerator-targeted) code, which is to run on the accelerator. The OpenMP4 directives provide information to the compiler both about the parallelism in the algorithms and about the data required to run those algorithms on an accelerator. That data is consolidated and transformed by Clang into calls to the OpenMP4 runtime library on the host side. This library is hardware agnostic and supports plug-ins for different kinds of accelerator hardware.

On a related node, longer FPGA compilation times negatively impacts productively and programmability. Addressing compile-time issues without significantly degrading performance will be challenging. There are some techniques with the potential to help: Creating regular grids of smaller individually-routed elements has been shown to significantly decrease overall place-and-route time [4]. Overlay architectures, running at nearly peak speed, have been demonstrated [5], [18], and targeting a family of such pre-place-and-routed architectures with more-traditional compiler technology may be able to restrict the compile-time problem to cases where tuned synthesis inputs are deemed worthwhile.

Performance: We will explore and develop optimizations at the software and hardware level to improve the performance/watt of the generated FPGA configurations following a cyclical codesign approach: we will design optimizations for software (OpenMP4 parallelization, OpenMP4 compiler, HLS) and hardware (computing and memory interface structures) for HPC motifs, in a coordinated way.

Especially pertaining to off-chip memory, we will employ a library approach to optimize expensive off-chip memory accesses and implement memory interfaces as predefined hardware libraries that can be selected via “extended pragmas,” focusing on three memory interface techniques: per-data structure memory hierarchy, irregular access optimization, and data reduction.

4 Conclusion

FPGAs are gaining the spotlight as a computing resource; modern FPGAs include thousands of hard DSPs or floating-point units. In the preparatory stages, we addressed the technology gaps in adopting FPGA technology for HPC. Our goal is to design and implement “Re-form,” an FPGA-powered true codesign flow that significantly improves the energy efficiency of the post-Moore era supercomputers.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy Office of Science, under contract DE-AC02-06CH11357.
REFERENCES


**GOVERNMENT LICENSE**

The submitted manuscript has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory (“Argonne”). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. The U.S. Government retains for itself, and others acting on its behalf, a paid-up nonexclusive, irrevocable worldwide license in said article to reproduce, prepare derivative works, distribute copies to the public, and perform publicly and display publicly, by or on behalf of the Government.
Pipelined Intermediate Fabrics for Parallel Execution

Hiroka Ihara, The Univ. of Tokyo and Kenjiro Taura, The Univ. of Tokyo

Abstract—It is known that intermediate fabrics for FPGA accelerators can improve the end-user productivity through both program deployment free of logic synthesis and high portability. In this paper, one possible ecosystem for intermediate fabric is discussed, where pipelined reconfigurable architecture is employed to enable scalable and parallel execution. Such ecosystem can improve the utilization rate of FPGA accelerators in the field of supercomputing.

1 INTRODUCTION

FPGA is likely to become one of the key players in the post-moore era, since they encompass room for further performance gain through near-hardware level of problem-specific optimization [5]. Extensive yet fruitful researches have been done seeking to increase their affinity with HPC programming, including new compilers for high-level languages and numerous useful algorithms devoted to the architecture [5]. However as an almost inevitable implication of this highly reconfigurable architecture, lengthy compilation process of FPGA programs, along with their portability being low at binary level, has long since been known to degrade the end-user productivity [6]. An intermediate fabric (IF), sometimes paraphrased as an “overlay”, which is a collection of coarse-grained function blocks connected together with programmable switches (as shown in Figure 1), can be synthesized and deployed in advance as a hardware abstraction layer to enable instant deployment of FPGA programs without complete logic synthesis, and to enable binary-level compatibility across various underlying FPGA products [1]. Coole and Stitt (2014) showed that they exhibit not only prominent compilation speed up to ten thousand times faster than the traditional high-level synthesis, but also performance close to, or sometimes even better than, that of traditionally compiled logic [2]. In this paper, we propose an extension of IF called PRA-IF, which can hopefully add scalability and parallelism to the architecture, both of which are important attributes of HPC-capable architectures. It is essentially an incorporation of Pipelined Reconfigurable Architecture first introduced by Goldstein et al. into IF, aimed at realtime time-multiplexing of logic configurations. Our goal is to enable a wider range of HPC programs to benefit from FPGA, eventually increasing the utilization rate of expensive systems employing the device.

2 INTERMEDIATE FABRICS

Simplified diagram of the IF architecture introduced by Coole et al. is shown in Figure 1. It is a coarse-grained network of function blocks that virtualizes FPGA, where the connection can be reprogrammed by feeding a configuration bitstream from the outside. The dedicated compiler maps a user program written in OpenCL to a configuration (netlist), while selecting the IF implementation best suited for the algorithm. The IF is then deployed onto the FPGA before it is configured according to the netlist, and when the computation is finished, another IF can supersede to handle different programs. This architecture could exploit the reconfigurability of FPGA by making the IF exchangeable, whereas it still had difficulty coping with a group of programs that vary greatly in terms of complexity, which resulted in low LUT utilization when the kernel happened to be smaller than the system had expected. It could be naturally inferred that the architecture will not be able to process multiple execution contexts in realtime, because an IF reserved the entire hardware during its timeslot. These limitations in both scalability and parallelism should be addressed when the purpose of introducing IF is better utilization rate of FPGA resource. In particular, the disability to execute multiple kernels in parallel would cause the utilization of CPUs, which is also crucial, to decrease through longer turnaround time.

3 PIPELINED RECONFIGURABLE ARCHITECTURE

A hardwired coprocessor implementation called PipeRench, introduced by Goldstein et al., employed a novel pipeline architecture (Pipelined Reconfigurable Architecture, PRA) to successfully address the scalability issue inherent in reconfigurable systems [4]. PipeRench coprocessor is a massive collection of pipeline stages, namely pairs of programmable ALU and a register file, connected together in a reconfigurable manner as shown in Figure 2. Their compiler basically converts a input program written in DIL (Dataflow Intermediate Language) into a logical pipeline, mapping it onto the physical processing elements. The novelty was that the logical pipeline stages split into steps are executed one by one, each of which reconfiguring some part of the physical pipeline for one step. In this way, the processor could virtually handle any size of computation kernels by taking multiple steps to complete. Applying PRA to FPGA-based accelerators in the form of IF is the starting...
point of our research, not only because the architecture is a solution to one side of the problem (scalability), but also it can be extended to support multiple execution context. The core idea is that given an appropriate memory model and a mapping algorithm, PRA might have the capability to time-multiplex on itself pipeline stages from several programs.

4 PRA-IF ARCHITECTURE

We are currently discussing how a combination of IF and PRA (PRA-IF) can contribute to better FPGA utilization. The overall workflow should appear to be a combination of what each architecture call for: the host processor compiles the user program into a pipelineable dataflow, which is then transferred to the IF as its configuration. At the same time, the IF implementation that best suits the target program is selected and deployed. The difference from how PRA was originally configured are that each pipeline stage in the original program is scheduled by the PRA autonomously in realtime, not by the compiler before execution, and that multiple processes might be trying to execute their own program on the same IF. In other words, some dynamic work sharing algorithm is running on the PRA itself, which arranges the assignment of multiple logical pipelines in realtime onto the physical resource available.

The PRA-IF architecture should also take advantage of FPGA-specific features that are presented in a hardwired die. Among them are the integrated SRAM blocks, used by PRA-IF to boost the data I/O performance. Each pipeline stage is equipped with a separate memory access port capable of randomly addressing into the SRAM blocks every cycle independently. The SRAM blocks, in turn, communicate with the external memory controller to transfer data into and from themselves in a more coarse-grained and periodical fashion. Timings and patterns of such data exchanges are explicitly dictated in the source program, while each random access into the SRAM blocks is implicitly managed as a load-store operation.

5 EXAMPLE OF PROGRAM EXECUTION

An example PRA-IF coprocessor hosting two independent computation kernels is shown in Figure 3. Each process on the host computer translates its user program into logical pipeline steps, and push them onto the job queue inside the PRA-IF coprocessor. In Figure 3, Program 1 submitted by Process 1 consists of two steps, while Program 2 consists of four steps, resulting in a total of six jobs. All the six jobs, once pushed onto the queue, are scheduled together inside the PRA-IF in realtime and mapped onto the physical pipeline. Each physical pipeline stage reconfigures itself according either to the job assigned by the scheduler, or to the execution state of the pipeline. Because the PRA-IF is implemented on the FPGA, the IF can be exchanged when the programs can benefit from specialized components such as floating-point units or FFT.

6 PROGRAMMING LANGUAGE AND RUNTIME SYSTEM

The pipeline configuration is simplified to be functionally identical to a VLIW program, where conditional branches are replaced with sequences of conditional instructions that inflicts penalty neither on the instruction bandwidth nor the execution latency. In order to generate configuration for PRA-IF however, compilers and runtime systems will have to face requirements that differ from what existing reconfigurable systems focus on, mainly due to the dynamic work sharing. Challenges for the compiler include how to control the interference between separate execution contexts, when static scheduling is impossible. This task is carried out by attaching a bounding box to the program, which specifies what resource should be locked at a specific moment. The runtime scheduler on the hardware side would orchestrate multiple execution contexts using mask bits indicating resources being occupied.

On the other hand, difficulty for the runtime system arises when evaluating the set of available IFs based on their aptitude to the target kernel, when the target kernel is a complex mixture of multiple contexts. The runtime system must ensure that the IF deployed on the FPGA can safely handle all of the programs running at every moment, while maximizing their individual performance. Consequently, every IF instance must at least have the same degree of capability, making sure that any possible configuration can be satisfied with its requirements. Configurations might be modified by the runtime so that it can adapt to the actual makeup of the pipeline hardware, e.g. units of memory access or available pipeline width.

7 CONCLUSION

Combination of intermediate fabrics and pipelined reconfigurable architecture can possibly help improve the utilization rate of FPGA accelerators in the field of supercomputing. The former technology was conceived to eliminate compilation overhead and portability issues, while the latter can provide scalability and realtime awareness. We are currently going through theoretical validation of the system, mainly focused on compiler infrastructure.

ACKNOWLEDGMENTS

This work was supported by Grant-in-Aid for Scientific Research (A).
REFERENCES


Heterogeneous Computing –
A Path to Post-Moore Supercomputing:
Architecture, Circuits, and Process

Wayne Burleson, Shomit Das, Yasuko Eckert, Nam Sung Kim

Abstract—As the trends driven by Moore’s Law come to an end, increased heterogeneity at all levels of computing is required to continue supercomputing progress. Supercomputing, perhaps more than other domains of computing such as client devices or Internet of Things, has relied on ever increasing resources and efficient management. This paper describes the opportunities and challenges of heterogeneous technologies at the architecture, circuit, and manufacturing process levels that provide a foundation for Post-Moore Supercomputing.

1 INTRODUCTION

Advances in supercomputing have relied on Moore’s Law almost since its inception, providing improved performance, power efficiency, and cost objectives due to the ever increasing density of CMOS integrated circuits. Past supercomputers primarily used homogeneous resources to manage complexity and manufacturing cost and to facilitate programming, reliability, and resource management. However, homogeneity brings limitations in efficiency. In contrast, heterogeneous technologies, architectures, circuits, and programming models allow workloads to be closely matched to the most appropriate resources [1]-[5]. Orchestrated use of heterogeneity throughout the software and hardware stacks becomes critical to continue and even accelerate the progress of supercomputing in the Post Moore’s Era. This paper reviews the advantages and disadvantages of heterogeneity and describes emerging techniques to manage the challenges at three levels: 1) Heterogeneous System Architectures; 2) Heterogeneous Circuit Designs and Tools; and 3) Monolithic 3D Integration. Examples from AMD Research and other groups are shown as early examples of these promising approaches. Heterogeneous software and programming models [6]-[9] are also an important technology in the Post Moore’s Era, however are beyond the scope of this paper.

2 HETEROGENEOUS SYSTEM ARCHITECTURES

Increases in the level of processor heterogeneity will become one of the key techniques to drive performance and energy-efficiency scaling in the absence of CMOS transistor scaling. This trend is already happening as exemplified by Google’s reliance on a custom ASIC called Tensor Processing Unit (TPU) to boost their machine-learning applications [10]. They claim an order of magnitude improvement in performance per Watt, equivalent to an improvement of three generations of Moore’s Law. Leveraging reduced computational precision, the TPU scales down the amount of hardware and power per unit operation for machine-learning algorithms. Going forward, we expect more such efforts to be realized, leading to a highly tuned specialized hardware repository. A single die and/or a tightly integrated package may consist of diverse compute units, such as CPU, GPU, FPGA, ASIC, DSP, neuromorphic, approximate computing, and quantum computing blocks, combined together with various memories and Networks on Chip (NoC). The exact combination of the compute units will depend on the characteristics of the target applications and design, verification, and manufacturing costs.

The increased reliance on heterogeneity is likely to cause shifts in traditional computer architecture. There will be greater focus on building accelerators or custom chips for specific applications rather than making general-purpose processors more energy efficient, due to higher payoffs. Further, efficient management of individually optimized processors will become more important as the number and diversity of the processors increase. Techniques such as tighter integration of processing units, job-scheduling policies, data placement, management, and distribution, and interconnect designs must be developed.

HSA [8] is a good first step towards enabling efficient heterogeneous systems. AMD’s Exascale project is already innovating many of the above aspects by integrating 3D and 2.5D die-stacking technologies, HSA, compute-optimized GPU, processing in/ near memory (PIM/ PNM), and 3D interconnect networks, just to name a few [1]. But opportunities and challenges still remain to push the heterogeneity to the next level of systems almost entirely composed of accelerators. One of the challenges of such systems is exposing the right level of hardware details to the software. Hardware-software co-design is likely required.

W.P. Burleson is with AMD Research, Boxboro, MA. E-mail: waynебurleson@amd.com
Shomit Das is with AMD Research, Austin, TX. E-mail: Shomit.Das@amd.com
Y. Eckert is with AMD Research, Bellevue, WA. E-mail: Yasuko.Eckert@amd.com
N.S. Kim is with University of Illinois, Urbana-Champaign, IL and AMD Research, Austin, TX. E-mail: nskim@illinois.edu
to facilitate easy programming for highly heterogeneous systems and code portability across different combinations of accelerators.

## 2.1 Longer Term

Supercomputing may see a finer granularity of heterogeneity in the future. Given that technological advances such as monolithic 3D integration described later allow tighter coupling of heterogeneous technology layers at the transistor level, it may become feasible to construct a single processor from multiple different technologies (heterogeneity within a processor), such as combinations of FPGA and ASIC for different parts of a processor, for enhanced energy efficiency.

Supercomputing may no longer need many general-purpose processors, such as CPUs and even GPGPUs, if we need to rely on highly specialized accelerators and custom chips to hit a desired performance/Watt goal.

## 3 Heterogeneous Circuit Designs and Tools

Heterogeneous circuit styles, including voltage and timing domains, are needed for efficiency, but increase the complexity of design and verification. Some techniques that we predict to gain further traction are described below.

Fine grained Dynamic Voltage and Frequency Scaling (DVFS) is already in use; however Adaptive Voltage Frequency Scaling (AVFS) using high bandwidth critical path accumulators will continue to advance [11]-[14]. These paths provide a statistical sampling of the full set of Fmax limiting paths, and help decide post-silicon timing margins along with improved dynamic variability tracking. This scheme will allow each accelerator or function block to have its own dedicated voltage-frequency domain. Such autonomous techniques require digitally-controlled on-chip fully integrated voltage regulators (IVR). The IVRs will be designed to respond to dynamic sensors, such as voltage droop sensing, replica paths, and core warning signals and provide fast, accurate voltage supply to the blocks [15]. Also, run-time voltage droop handling will be provided by adaptive clocking and clock stretching techniques.

The need for power savings will likely inspire even more circuits to be operated with voltage supplies near the threshold voltage, known as Near Threshold Computing (NTC) [16]-[18]. We believe that timing variability due to process, voltage, and temperature (PVT) variations, which will be exacerbated at low voltages, will be the primary limiter to adopt this technology, rather than performance degradation. Going forward, the circuit designer may more frequently utilize NTC in low-voltage FinFET devices and pipelines with several stages between flops. We expect cell-based parametric on-chip variation (POCV) methods [19][20] will become the de facto practice for process variation analysis and for margining timing.

Challenges in heterogeneous circuit design styles include: 1) efficient interfaces between voltage and timing domains, 2) validating mixed systems and their interfaces, 3) creating and maintaining diverse sets of tools and larger libraries of cells and IP (intellectual-property) blocks, 4) dynamic variation tolerant design mechanisms, and 5) education of designers in alternative circuit styles.

## 4 Monolithic 3D

Tight integration of heterogeneous process technologies is necessary to realize efficient heterogeneous architectures. Monolithic 3D integration is an emerging technology that integrates heterogeneous technology layers at the transistor scale, thereby enabling higher communication density than the current 3D- and 2.5D-stacking technologies that rely on coarse-grained through-silicon vias and silicon interposers [21]. Although monolithic 3D integration can enable many innovative circuit and architecture technologies that have been inconceivable in the current 3D and 2.5D technologies, some major processing obstacles have prohibited the demonstration of its full potential. Especially, the most critical challenges in monolithic 3D integration are that all the upper layers must be processed at low temperature (less than 400 °C).

Faced with such a challenge, new devices such as tunneling field-effect transistor (TFET) and ferroelectric field-effect transistors (FeFET) based on 2D materials have been proposed. The advantage of such devices based on 2D materials is that the processing temperature is low enough for the monolithic 3D integration. Furthermore, these devices exhibit some advantages over the traditional CMOS devices. For example, recent TFET based on 2D materials demonstrated superior energy-versus-delay characteristics to conventional MOSFET devices [22], and random access memory built with HfO2-based FeFETs (FRAM) [23] consumes lower power than SRAM, DRAM, magneto-resistive RAM (MRAM), and phase-change RAM (PRAM), with far more endurance than resistive RAM (RRAM).

The aforementioned benefits of the monolithic 3D integration technology enable exploration of innovative memory-centric circuit and processor organizations that directly expose and integrate FRAM-based large-capacity, energy-efficient, non-volatile memory with TFET-based processing logic, facilitating fine-grained in-memory processing and other architectures with reduced data-movement costs.

## 5 Conclusion

Heterogeneity at various levels in supercomputing can provide the efficiency needed to sustain performance trends even in a post-Moore era. However, heterogeneity introduces complexity that must be tackled with new design techniques and tools, opening numerous new research directions across all areas of computing disciplines.

### Acknowledgment

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies. Nam Sung Kim is supported partially by National Science Foundation (NSF) under grant number CNS-1115835.
REFERENCES


Neuromorphic Computing: A Post-Moore’s Law Complementary Architecture

Catherine D. Schuman, Member, IEEE, J. Douglas Birdwell, Fellow, IEEE, Mark E. Dean, Fellow, IEEE, James S. Plank, Member, IEEE, and Garrett S. Rose, Member, IEEE

Abstract—We describe our approach to post-Moore’s law computing with three neuromorphic computing models that share a RISC philosophy, featuring simple components combined with a flexible and programmable structure. We envision these to be leveraged as co-processors, or as data filters to provide in situ data analysis in supercomputing environments.

1 INTRODUCTION

The post-Moore’s law era is upon us; it is vital that the community examine complementary, non-von Neumann architectures. A confluence of events beyond the end of Moore’s law indicate that new architectures should be researched, implemented, and deployed. These events include the end of Dennard scaling, the prevalence of the von Neumann bottleneck, and the continued difficulties of standard parallel programming. We advocate neuromorphic computing systems as complementary non-von Neumann architectures to augment existing systems. The key properties of neuromorphic systems that we are exploiting are (1) collocated computation and memory, (2) two basic components: neurons and synapses, (3) simple communication between components, (4) inherent parallelism in computation, and (5) spiking or discrete-event systems with low power requirements. We advocate the use of neuromorphic computing as a complementary architecture because the collocated memory and simple communication avoid von Neumann bottlenecks and are not as affected by slower communication speeds; the basic components and the inherent parallelism of computation help overcome the issues associated with parallel programming and synchronization, and innovations in device design help to overcome the issues with the end of Moore’s law and Dennard scaling.

2 APPROACH

Our approach to neuromorphic computing includes software and hardware components, as well as simulators for testing. We are exploring implementations using off-the-shelf components, such as FPGAs and GPUs, as well as custom implementations in VLSI and emerging device technologies such as memristors. Our goal has been to produce a common programming technology, and architectures with shared characteristics. All of our systems share two characteristics: very simple neurons and synapses with two parameters each (one parameter is a notion of memory while the other affects temporal components) and programmable/structure (in the number and placement of neurons and synapses). Each architecture also includes a primitive notion of on-line learning, inspired by long-term potentiation and long-term depression in biological systems. Our three major architectures (Figure 1) are:

- NIDA: A 3D model that is implemented in simulation only, from which other implementations are derived [9].
- DANNA: An 2D model designed specifically for digital hardware, currently implemented in FPGAs and in early-stage development for VLSI [4].
- mrDANNA: An implementation of NIDA using memristors as synapses.

Fig. 1. Our neuromorphic models: NIDA (simulation only), DANNA (FPGA, future VLSI), and mrDANNA (leveraging memristors).

We are developing simulation software and customized training implementations for each architecture. The training is based on evolutionary optimization, which defines the model structure and parameters for each application. In addition, we are developing system software for each implementation. Our neuromorphic approach differs from other neuromorphic computing approaches by taking a RISC approach, choosing very simple component functionality (in the neuron and synapse), but allowing for flexibility in the way those components are combined (in the programmability of structure available in each...
of our neuromorphic architectures). We compare our architectures (NIDA/DANNA/mrDANNA) with five other major neuromorphic computing efforts in Table 1. In this table, we quantify programmable structure as whether the architecture allows for varying numbers and placements of neurons and synapses in the array and component complexity in terms of the number of parameters for neurons and synapses (listed in the table as number of neuron parameters / number of synapse parameters). Our programmability has an advantage for applications in supercomputing because it allows flexibility in parameter settings and in the way the components are connected and laid out in the system. Because of this flexibility, the user has a greater degree of freedom to exploit the properties of the neuromorphic architecture, potentially widening the scope of applicability of neuromorphic systems. The simplicity of our components helps alleviate the burden of actually programming the device to perform tasks. Our evolutionary optimization programming method can determine parameters and/or structure, allowing the user as much input in the training process as they desire.

2.1 Maturity of the Architecture

The most mature of all of our implementations is the FPGA implementation, with a working prototype. The mrDANNA implementation is in simulation, and the DANNA VLSI implementation is in early development. Supporting software for each system is in progress. DANNA simulation software is complete, and a version that uses GPUs is in development. Circuit-level simulations using SPICE have been completed for small mrDANNA networks, but a custom software simulation implementation is still in development.

2.2 Supercomputing Applications

We have identified two areas in supercomputing through which neuromorphic computers may be useful. One area is as a co-processor and the other is as an in situ data analysis processor. The co-processor may be used in a similar way as a GPU; in particular, certain applications or functionality within an application may be accelerated using the neuromorphic co-processor. Because neuromorphic systems in general and ours in particular can process spatiotemporal information, a neuromorphic system can also be used to pre-process data as it’s being generated by the supercomputer (for example, data from a large-scale science simulation). We have used our neuromorphic systems for simple classification [8], anomaly detection [7], and control [3] tasks. However, we do not expect to achieve state-of-the-art results on machine learning tasks such as image classification using our approach; deep learning techniques such as convolutional neural networks will likely still achieve better results. It may be possible to run a version of a convolutional neural network on our architectures, but we would still expect a performance drop as a result of the mapping process. This is why we advocate the use of neuromorphic systems as pre-processors for the data. We believe it will be possible to use a neuromorphic system as a filter, to identify interesting or anomalous results in the data generated, which can then be further processed by an off-chip, more intensive machine learning technique.

2.3 Risks

The investment in custom chips (DANNA VLSI and mrDANNA) will be significant; however, our focus on off-the-shelf commodities, such as FPGAs, minimizes the risk of investment, as the FPGA can be used in other ways (beyond being programmed as a DANNA). However, power efficiency and scalability is severely decreased by limiting to FPGAs. We recommend deployment in stages: small-scale deployment using FPGA implementations, in order to understand the characteristics and capabilities of the system on desired applications, followed by larger-scale deployment using either VHDL DANNA or mrDANNA as appropriate for the application. In addition to capital investment, significant time investment will be required to identify the proper applications and to develop programs or code for the neuromorphic devices. Especially for the co-processor example, the development of usability tools (such as compilers or compiler directives) to ease the transition between existing programming techniques and programming for neuromorphic computers will be a necessary step to alleviate some of the risk associated with adopting neuromorphic computers.

3 Conclusion

Neuromorphic computing is a promising complementary architecture for the post-Moore’s law era of computing. With its key properties of collocated computation and memory, RISC-like components (neurons and synapses), simple communication, and temporal processing ability, it has the potential to re-shape the way we think about what is possible through computing. Our neuromorphic architectures are implementations with these characteristics that are in various stages of development. We plan to produce functional prototypes of each of these architectures, that will be made available to the community for research purposes. We believe our architectures are particularly amenable for supercomputing applications because of their programmability.

<table>
<thead>
<tr>
<th>Programmable Structure</th>
<th>Component Complexity</th>
<th>On-Chip Learning</th>
<th>Materials/Devices</th>
</tr>
</thead>
<tbody>
<tr>
<td>DARWIN/10</td>
<td>Neurons and synapses</td>
<td>2 / 2</td>
<td>Yes</td>
</tr>
<tr>
<td>trueNORTH [3]</td>
<td>Neurons and synapses</td>
<td>2 / 2</td>
<td>No</td>
</tr>
<tr>
<td>BrainScales [2]</td>
<td>Fixed (Synapses on/off)</td>
<td>10 / 7</td>
<td>No</td>
</tr>
<tr>
<td>SpiNNaker [6]</td>
<td>Neurons and synapses</td>
<td>Variable (Multiple models)</td>
<td>Yes</td>
</tr>
<tr>
<td>Neurogrid [1]</td>
<td>Fixed (Synapses on/off)</td>
<td>79 / 8 + additional</td>
<td>No</td>
</tr>
</tbody>
</table>
REFERENCES


A Hafnium-Oxide Memristive Dynamic Adaptive Neural Network Array

Gangotree Chakma, Member, IEEE, Mark E. Dean, Fellow, IEEE, Garrett S. Rose, Member, IEEE, Karsten Beckmann, Harika Manem, Member, IEEE, and Nathaniel C. Cady, Member, IEEE

Abstract—Power and precise scaling are significant restraining elements to advancements in computing. This paper presents a power efficient memristive device technology in the design of a neuromorphic architectural model that promises to overcome many of the performance limitations of conventional Von Neumann systems. The resulting memristive Dynamic Adaptive Neural Network Array (mrDANNA) addresses contemporary application challenges while also enabling continued performance scaling. The mrDANNA system is a mixed-mode neuromorphic computing system built with reconfigurable structure, dynamic adaptation, low-power operation, and is well suited for processing spatio-temporal data. This work is specifically based on a HfO$_2$ memristor device, experimental results for which are presented in this paper. Proof-of-concept simulation results for a mrDANNA pattern recognition network are also presented showing high accuracy for recognizing basic shapes.

1 INTRODUCTION

In recent years, the semiconductor industry has begun to experience a significant slowdown in the performance improvements gained from technology scaling. While this is due in part to the impending end of Moore’s Law scaling, power consumption and architectural limitations have also become critical limiting factors for the level of performance achievable. Neuromorphic or neuro-inspired computer architectures are particularly worthwhile given the increasing number of big data problems requiring techniques and systems that can capture knowledge from an abundance of data. Here, we present a memristor-based Dynamic Adaptive Neural Network array (mrDANNA) with the potential to address contemporary application challenges while also enabling continued performance scaling.

The work leverages a hybrid CMOS-Memristor process developed and tested at the SUNY Polytechnic Institute, College of Nanoscale Science and Engineering (CNSE) for its design and eventual hardware realization. The process integrates metal-oxide memristors in the metal layers of the IBM 65nm 10LP process, leading to a seamless CMOS/Memristor integration process.

The specific neuromorphic architecture on which the mrDANNA is based is the Neuroscience-Inspired Dynamic Architecture (NIDA) [5], [6], [8] as an approach to applying neuromorphic principles to a wide variety of applications. The structure and simplicity of the NIDA architectural model has been leveraged in the development of a Dynamic Adaptive Neural Network Array (DANNA) [1], an efficient digital system constructed from a basic element that can be configured to represent either a neuron or a synapse. Unique characteristics of the NIDA/DANNA approach over other neuromorphic or neuroscience-inspired systems include: a simplified neuron model, a higher functionality synapse model, real-time dynamic adaptability, configurability for the overall neuromorphic structure (e.g., number of neurons, number of synapses and connections), and scalability for element performance and system capacity.

2 MEMRISTIVE DEVICES

The memristive devices considered for this work were designed and fabricated in-house at the SUNY Polytechnic Institute’s Center for Semiconductor Research (CSR). Devices were manufactured on a 300mm wafer platform and integrated with the IBM 65nm 10LP process technology. It should be noted that facilities in-house allow for an area-efficient and seamless flow of front-end CMOS and back-end memristive and metallization processes. The seamless integration of CMOS with memristive technology is a unique feature as compared to related efforts where memristive devices are integrated post-fabrication on an existing CMOS chip [4]. A custom, cost-effective build embeds a HfO$_2$ memristor devices between metal 1 (M1) and metal 2 (M2) metallization layers. Fig. 1 (right) illustrates the device cross-section, with Fig. 1 (left) showing the TEM cross-section of the fabricated device.

The I-V characteristics of a fabricated HfO$_2$ memristor device are shown in Fig. 2. An average low resistance state (LRS) and high resistance state (HRS) of 10K and 150K, respectively, were observed during pulsing measurements. The on/off ratio and LRS are likely to increase, which is critical for low power operation, by manipulating the thicknesses and stoichiometries in the memristor film stack as well as moving the transistor in the 1T1M configuration from a discrete to an on-chip component effectively reducing parasitic capacitance. The devices show an excellent readout stress insensitivity, with the resistive states being insensitive to trillions of nanosecond pulse readouts. In addition, a low positive temperature dependence (5.9e-4 1/C) results in little change to the circuit performance over a large range of temperatures. Recent results also indicate controllable analog/multi-level switching in the memristive devices, which is key for their application as synaptic weights in the mrDANNA system. The model used for the simulation...
Fig. 1. Illustration (right) and TEM image (left) of a memristor device embedded between M1 and M2. The Tungsten V1 serves as the inert bottom electrode, HfO$_x$ is the active metal-oxide layer, the Ti layer serves as an oxygen-getter for the HfO$_x$, the TiN is an inert top electrode and the copper V1 and M2 provide the contact to the device.

Fig. 2. IV measurements of the memristive device for >450K set and reset cycles, with $V_{set} = 0.7V$, setting it to an LRS state and $V_{reset} = -1V$ resetting it to an HRS state.

Work in this paper is based on a model first developed and presented by McDonald et al. in [2], [3]. While the original model could be used for unipolar, nonpolar and bipolar behavior, this work is restricted to bipolar behavior following the observed characteristics of the device.

3 Memristive Neural Circuits

Synapses in the mrDANNA circuit must represent either a positive or negative weight and include delay distance as prescribed by the NIDA/DANNA model [1], [7]. Unlike the digital DANNA, in mrDANNA the synaptic weights are represented using memristors where the memristance is proportional to the desired weight. For the neuron, we implement an integrate-and-fire circuit similar to that described by Wu [9]. Here the design allows the neurons to operate in two different phases, integration and firing. When the neuron operates in its integration phase, the op amp acts as an integrator such that charge accumulates resulting in the membrane potential. A comparator circuit compares the membrane potential with the threshold voltage and generates a driving voltage. A resulting pulse on driving voltage then drives a “firing flop” which generates the corresponding output spike of the neuron.

The synaptic buffer drives voltage inputs across the two memristors of a synapse, one buffered and one complemented to allow for negative weights. It also controls the Long Term Potentiation (LTP) and Long Term Depression (LTD) mechanisms used for online learning and adaptation. The feedback signal from the IAF neuron connected to the synaptic buffers regulate the LTP/LTD event. Based on a firing event from the post-synaptic IAF neuron, as sensed by the synaptic buffer, the weight will be updated (increase/decrease) accordingly. A high level view of mrDANNA is provided in Fig. 3.

4 Shape Recognition Example

To showcase the usefulness of this type of network with synapses and integrate and fire neurons, a circuit for recognizing four basic shapes, triangle, square, diamond and plus, has been constructed. Besides recognizing noiseless images, some imperfect noisy images of triangles, squares, diamonds and also plus signs have been considered to determine the accuracy level of recognition of this network. Zero, one, two and three noise bits were considered for simulation and results for percentage of accuracy are shown in Fig. 4.

Fig. 4. Accuracy of triangle data and other shapes.

The results show that the network recognizes most of the cases with noisy bits up to 3 bits among the 25 bits of the image. The network recognizes images with one noisy bit with a hundred percent accuracy for all noise bits, the accuracy level slightly goes down but the percentage of accuracy is higher than 80 percent at 3 noise bits, which makes the circuit worthy enough in recognizing a particular shape. Average power calculated for this application is approximately 1.6mW.

5 Conclusion

This work demonstrates the efficiency of recognizing different patterns and also presents the power consumption for different LRS and HRS levels. In future work, more complex networks for spatiotemporal data applications and large pattern recognition will be solved with this efficient computing architecture.
REFERENCES


Modeling of Novel Transistors, Manufacturing Technologies, and Architectures to Preserve Digital Computing Performance Scaling

George Michelogiannakis, Member, IEEE, Dave Donofrio, Member, IEEE, and John Shalf, Member, IEEE

Abstract—The approaching end of traditional MOSFET technology scaling creates the need to identify novel devices, manufacturing technologies, memories, and architectures to preserve digital computing performance scaling. To this end, we argue for the need to generate circuit-level models and integrate them into existing simulation and digital design infrastructure for rapid architectural design exploration. Using CHISEL, we can generate behavioral and circuit models of novel technologies.

1 INTRODUCTION

Recent years have brought us closer to the end of traditional MOSFET technology scaling. Traditional CMOS is now predicted to initially slow down and eventually cease scaling by the beginning or middle of the next decade [13], with industry forecasting that technologies beyond 2nm or 3nm may be infeasible or impractical [9], [10]. This realization does not mean the end of performance scaling for digital computing, but rather an invitation to preserve performance scaling by adopting novel CMOS devices, manufacturing technologies, memories, and architectures. The approaching end of lithographic scaling threatens decades of DOE investment in hardware and software, as well as threats to hinder advancement in numerous scientific and society challenges that depend and will continue to depend on digital computing [1].

To create a tangible strategy, each of novel technology should be evaluated in the architectural and eventually the system levels. This will allow answering such questions as the feasibility of the increased parallelism that tunnel FETs (TFETs) [7] require to improve performance, how to alleviate reliability challenges by novel devices using architectural techniques, how to 3D stack logic and memory layers to reduce data movement and heat density, what is the impact of TB-level non-volatile memory on top of processing logic to programmers and power management, as well as many other similar questions. Answering such questions requires architectural-level modelling and evaluation and helps make best use of emerging technologies as well as guide each technology’s future progress. Whats more, evaluating novel devices in the architectural level allows studying the potential of architectural techniques such as specialization (use of accelerators), adopting non-Von Neumann architectures, and the impact they have to the software layer such as programming models.

To enable this crucial design space exploration, existing architectural- and system-level simulators need to be updated with circuit-level models for the aforementioned novel technologies. This effort includes generating those circuit-level models from low-level models such as voltage–current curves for transistors, and must include important characteristics such as reliability in addition to performance and energy. In this position paper, we make the case for extending a hardware description language (HDL) such as CHISEL [5], to generate both software and hardware architectural models that include new technologies.

2 BACKGROUND AND RELATED WORK

Several novel devices (transistors) have recently been fabricated and demonstrated, and are promising candidates to replace MOSFETs. Carbon nanotube transistors (CNFETs) have demonstrated a 1000× improvement of the energy-delay product (EDP) for memory-bound applications, 10× EDP improvement for compute-bound applications, and 30× for mixed workloads [3]. In addition, TFETs and negative capacitance FETs operate at a lower voltage than equivalent MOSFET transistors [4], [11]. Each new device introduces different tradeoffs such as performance at low and high voltages, reliability, energy, and others, which need to be carefully evaluated in the architectural level to properly assess impact. At the same time, new manufacturing technologies such as 3D stacking [16] of multiple logic and memory layers are quickly becoming feasible, but their higher-level impact and potential are not readily apparent. These options, combined with new memory technologies each with a different set of tradeoffs such as magnetic RAM [8], resistive RAM [2], create a vast landscape of options to preserve digital computing performance scaling.

Numerous alternatives exist for architectural-level software simulation, such as Gem5 [6]. Recently, HDLs such as PyMTL [12], Bluespec [14], and CHISEL [5] were proposed that can generate both behavioral (software) and circuit-level (hardware) models from a single code base. Software models can execute autonomously, much like a software simulator, while hardware models have to go through synthesis and placement to produce silicon or be placed on an FPGA. These new HDLs provide powerful means for rapid design exploration of large-scale architectures, but currently lack support for novel technologies. Past work has taken the first step towards evaluating new technologies [15], but only evaluated new devices and only for a 32-bit adder instead of representative future architectures. Similar infrastructure is being developed for alternative computation models such as neuromorphic and quantum, but the focus of this paper is modelling new technologies for digital computing.
Fig. 1. Based on current–voltage curves of new devices such as the one shown for TFETs, we need to generate circuit-level models suitable for architectural simulation.

Fig. 2. Different combinations of memory and logic layers change distances and available bandwidth between layers.

3 Circuit-Level Models of Novel Technologies

Generating circuit-level architectural models of new devices requires deriving energy and delay per operation from low-level voltage and current curves (Figure 1), and a given set of assumptions such as operating temperature. Moreover, these models should include error rate, variability, and other aspects important to adopting new devices. The same is true for modelling new memory technologies. Different memories have different access times and energy per access. Those may depend on where data is located in the memory array as well as other active requests to the memory. In addition to different error rates, some memory technologies are also non-volatile. This is a critical property of new memories that also needs to be understood in the architectural and system levels. Such models enable us to better gauge the impact to the memory hierarchy and power management of having large amounts of non-volatile memory near or on top of future processors. This contradicts current programming assumptions that non-volatile memory is distant and expensive to access.

3D integration of multiple kinds of layers also affects distances and relevant performance–cost tradeoffs as shown in Figure 2. In that picture, the choice of the kind of logic (e.g., accelerators or general-purpose) and memory for each layer affects the latency, energy, and available bandwidth for any two blocks to communicate. In addition, some combinations of layers may not be feasible due to high heat density, and some combinations may be a poor choice if they stress the limited bandwidth available between layers. 3D integration should be a parameter in the architectural models of our infrastructure in a way that each memory or logic block can be quickly placed in different layers, and the energy cost and available bandwidth between layers can be readily calculated based on technology models.

Finally, specialized architectures such as accelerators, GPUs, or fixed-function blocks should also be easily modelled. This can be done either by implementing the specialized architecture in an HDL, or more easily by creating abstract blocks with configurable delays and energy costs to perform a certain action. The action can be specified in terms of result even in high-level functional language, without having to describe the detailed hardware to generate the result. Essentially this allows simulation of a circuit where different blocks are modelled in different levels of detail. This implies that different blocks must have different levels of models, and that the level of abstraction of each component must be selected carefully. This will enable a quick exploration of potential new specialized architectures before fully implementing them.

4 Integration to Existing Infrastructure

We consider that extending a HDL such as CHISEL is a promising avenue to realizing the aforementioned goals. CHISEL generated both hardware and software models from a single code base, both of which are necessary for a complete study. Currently, CHISEL uses a backbone where the code description is transformed into a graph, and then converted to the desired output. CHISEL’s backbone can be extended to allow a choice of which devices and other technologies to use, and the performance, energy, and reliability models for each. The same is true for specialized architectures which can be defined as black boxes with associated cost and performance models.

The performance and cost models can be incorporated into CHISEL’s software (simulation) models such that performance models affect timing and delays during the simulation, and cost models also affect the reported power consumption after all events during the simulation are recorded. For hardware models (e.g., Verilog) generated by CHISEL, performance and cost models will both affect timing and placement during synthesis. Both hardware and software models can be used to estimate heat density and system-level error rates, based on relevant circuit-level models.

As a next step, we can develop a system-on-chip using CHISEL that includes cores as well as a complete memory hierarchy (including caches). This serves as a testbed for experiments on programming models, compilers, and algorithms. This is a necessary step to capture implications that novel technologies have on programming models, stemming from reliability, an increased need for parallelism, non-volatile nearby memories, and other potential aspects on top of typical performance–cost tradeoffs.

5 Conclusion

To respond to the approaching end of MOSFET technology scaling and preserve performance scaling of digital computing, we need to create reliable circuit-level performance and cost models of emerging technologies to use in architectural and system studies. This will allow us to evaluate new technologies in the architectural scale which will better guide technology development as well as motivate changes in the architecture and software. In this position paper, we argue for the need to develop such models for novel devices, memories, 3D integration, and specialized architectures. In addition, we briefly describe how to integrate these models into existing infrastructure to perform the necessary architectural- and system-level evaluations and gauge the impact of new technologies to the architecture and software.

Acknowledgments

This work was supported by the Director, Office of Science, of the U.S. Department of Energy under Contract No. DE- AC02-05CH11231.
REFERENCES


Finding Balance in the post-Moore’s Law Era

Jeffrey Young, Member, IEEE, and Rich Vuduc, Member, IEEE

Abstract—Recent developments in 3D stacked memory technologies and the promise of fast optical interconnects have offered new possibilities for balancing communication and computation time in future supercomputers. However, the end of Moore’s Law means that future architectural improvements will become more limited in scope. We theoretically evaluate a proposal that balanced “post-Moore’s” systems will be most achievable via a combination of small, lightweight processors, high-bandwidth memories with optimized “horizontal” communication, and network-oriented algorithm codesign.

1 INTRODUCTION

As we move towards exascale computing and beyond, technology limitations of the post-Moore’s Law era mean that we will need to allocate a finite supply of transistors and energy by creating both balanced and scalable supercomputers. In previous work, we postulated that a lightweight, CPU-style system would achieve the best balance (ie, flop to byte ratio) but also that newer xPU stacked memories and network topologies might change this analysis [3], [4]. Emerging technologies, such as stacked memories like High Bandwidth Memory (HBM) and Hybrid Memory Cube (HMC), and first-generation optical networks, such as NVLink and Omni-Path, provide us with new architectural possibilities for designing future systems with a better balance of computation and fast data movement.

In part due to these architectural trends, we argue that scalable, balanced supercomputers will have to focus more on memory-centric architectures and emerging network designs, while algorithms will need to be optimized in tandem with horizontal or network-based movement. In addition, we speculate that as the slowdown in Moore’s Law reduces the benefit of increased cache sizes and the possibility of higher peak flop rates, maximizing local and network bandwidth becomes the most important factor for creating a balanced system.

A recent position paper suggests that although stacked memories will improve memory bandwidth, future applications still might not have enough memory-level parallelism (MLP) to take advantage of larger amounts of local bandwidth [10]. In addition, design-focused proposals argue that future systems using stacked memories with silicon interposers need to be designed more like a Network on Chip (NoC) in order to mitigate thermal concerns while also providing appropriate amounts of local bandwidth and storage [9]. Both of these memory-related papers point to two concerns that we believe are important for balanced system design in the post-Moore’s era: 1) balanced systems need to be aggressively networked with high-bandwidth memories to provide flexibility in maximizing MLP, and 2) algorithms need to be designed cooperatively with local and global networks in order to best balance computation and communication.

As part of our argument, we address trends in future systems by looking closely at the costs for an algorithm running on an abstract distributed memory system. Distributed algorithms face two communication costs: vertical communication

---

1ST INTERNATIONAL WORKSHOP ON POST-MOORE’S ERA SUPERCOMPUTING (PMES)
and the time for all network communication is approximately $T_{\text{net}} \approx Q_{\text{net}}/B_{\text{bisect}}$. After some algebra, the communication times relative to the compute time are

$$\frac{T_{\text{mem}}}{T_{\text{comp}}} = \frac{R_0}{B_{\text{mem}}} \times \frac{1}{g(Z)}$$

(4)

and

$$\frac{T_{\text{net}}}{T_{\text{comp}}} \approx \frac{R_0}{B_{\text{link}}} \times \frac{P^2 \cdot h_1(P, d)}{h_2(n)}.$$  

(5)

Equations (4) and (5) summarize the tradeoffs affecting communication costs. Regarding the memory hierarchy, decreasing the flop-to-byte ratio ($R_0/B_{\text{mem}}$) or increasing the fast memory size help reduce cost; however, recall $g(Z)$ tends to grow slowly with $Z$. Regarding the network, decreasing the (analogous) flop-to-byte ratio of $R_0/B_{\text{link}}$ also helps. And while network cost may grow with increasing $P$ via $h_1(P, d)$, that cost should be weighed against the potential asymptotic reductions that come from a distributed algorithm via $h_2(n)$.

One caveat is that if the network energy is substantially higher than local memory energy, then distributing more aggressively—to increase horizontal communication while reducing vertical communication—may actually yield less energy-efficient systems [3]. This tradeoff will likely be heavily influenced by the success or failure of optical interconnects, which could make horizontal communication more competitive in terms of network energy usage while also providing better system balance.

### 3 Finding Balance As Moore’s Law Concludes

Looking forward at long-term trends in memory-centric system and algorithm design, we have proposed two new post-Moore “iron laws,” equations (4) and (5). In the most extreme post-Moore’s scenario, technology scaling stops, and Equation (4) becomes either slow-growing or constant. This change occurs because $Z$ cannot increase since there are no transistors left to increase the size of fast memory and because $R_0$ plateaus due to the lack of transistor scaling. For this specific case, increasing $B_{\text{mem}}$ and modifying the algorithm-specific knob of $g()$ via locality-aware algorithm design become the primary methods to improve system balance, at least until $B_{\text{mem}}$ also plateaus, likely one to two generations after $R_0$. In the network case outlined in Equation (4), designing a balanced system will require coordinated design choices as to the dimensionality of the network, $P$ and $h_1()$, and algorithmic optimization of $h_2()$. In summary, system balance can be improved by limited scaling to $B_{\text{mem}}$ and $B_{\text{link}}$ even after technology scaling stops for processor technology, and further algorithmic optimization provides the best avenue for improvement when architectural modifications become too costly.

Previous approaches to system design have captured some of these design characteristics, specifically the BlueGene [2], [8] series of machines that focused on maximizing the ratio of compute nodes to $B_{\text{link}}$ through the use a 5-D torus ($d = 5$) and routing logic integrated onto a System on Chip. However, recent trends in supercomputer design have at least temporarily moved back towards a combination of lower-dimension, commodity interconnects like PCI Express, InfiniBand, and NVLink, which may end up reducing system balance with respect to powerful new GPU and Phi accelerators.

While these two equations cannot capture the full scope of system design in the post-Moore’s era, they do indicate that our research focus should be on optimizing local and global interconnects and that coordinated algorithmic changes are required in order to make the best use of limited resources in future supercomputers.
REFERENCES


The Orchestration Stack:
The Impossible Task of Designing Software for Unknown Future Post-CMOS Hardware

Marcus Völп†, Sascha Klüppelholz, Jeronimo Castrillon, Hermann Härting, Nils Asmussen, Uwe Aßmann, Franz Baader, Christel Baier, Gerhard Fettweis, Jochen Fröhlich, Andrés Goens, Sebastian Haas, Dirk Habich, Mattis Hasler, Immo Huismann, Tomas Karnagel, Sven Karol, Wolfgang Lehner, Linda Leuschner, Matthias Lieber, Siqi Ling, Steffen Märcker, Johannes Mey, Wolfgang Nagel, Benedikt Nöthen, Rafael Peñaloza†, Michael Raitza, Jörg Stiller, Annett Ungethüm, and Axel Voigt
Center for Advancing Electronics Dresden (cfaed) — Technische Universität Dresden, Germany
† now at SnT-CritiX — University of Luxembourg, ‡ now at KRDB Research Centre — Free University of Bozen-Bolzano

Abstract—Future systems based on post-CMOS technologies will be wildly heterogeneous, with properties largely unknown today. This paper presents our design of a new hardware/software stack to address the challenge of preparing software development for such systems. It combines well-understood technologies from different areas, e.g., network-on-chips, capability operating systems, flexible programming models and model checking. We describe our approach and provide details on key technologies.

1 INTRODUCTION

In 2012, the large-scale research project Center for Advancing Electronics Dresden (cfaed) was set up in Dresden to explore new materials and technologies for electronic information processing, which potentially help overcoming the limits of today’s CMOS-technology. The project consists of multiple sub-projects that focus on promising concrete technologies, including reconfigurable transistors based on silicon nanowires (SiNW) [43], [17], [12] and carbon nanotubes (CNT) [38], [36], [32], [29], organic electronics [31], chemical information processing (e.g., microchmemechanical labs-on-chip [42]) and self-assembling nano-structures built with DNA origami [14]).

The long-term - allegedly impossible - task of cfaed’s Orchestration sub-project is to unleash the full potential of future - yet unknown - computing platforms and to turn breakthroughs in emerging materials and technologies into application performance. We envision wildly heterogeneous computing systems with potentially large numbers of possibly unreliable processing elements and deep heterogeneous memory subsystems that are in part built from the above technologies.

Since these new technologies are not yet available and their characteristics unknown, we use heterogeneous CMOS systems as a starting point for our research. Our objective is to initially design CMOS-based systems such that they can be more easily used for novel technologies and architectures. Since heterogeneity already is an important concept for overcoming barriers limiting conventional CMOS-based architectures (e.g., power-density problems [39], [16]), we can start with an already large base of heterogeneity [9], [28], [10], [26], [33], [35], [37], [5], [34]. Sec. 2 describes the Orchestration Stack which addresses expected challenges for wildly heterogeneous systems and Sec. 3 presents initial implementations contributing to the stack.

2 THE ORCHESTRATION STACK

On the lowest layer of the Orchestration Stack (see Fig. 1) we assume to have a variety of heterogeneous components based on different technologies such as SiNWs, CNTs or classical CMOS hardware (possibly with upcoming channel materials [21]) that have different characteristics in terms of performance and costs. These components can be specialized processing elements, e.g., accelerators, heterogeneous memories [44] or interfaces to bridge to peripherals, e.g., wireless communication devices, or to novel computing fabrics, e.g., labs-on-a-chip. Components can also be partially reconfigurable circuits combining different processing elements or providing a platform for application-specific and even software-delivered circuits.

All we require is that the components have a well-defined interface that allows embedding them into a tile-based architecture and that enables to exchange data and commands using some kind of network (e.g., a network-on-chip (NoC) [18]). New materials providing this interface can hence be embedded into this architecture. In Sec. 3 we present such an architecture and illustrate first steps we took towards our technical vision of fast reconﬁgurable hardware built from SiNWs or CNTs. The operating system’s (OS) task is to isolate hardware components and establish communication channels to other tiles and remote memories. As is common practice, the OS consist of a kernel and several servers on top of the kernel [15], [13], [30]. However, since no assumptions can be made on the components, the classical user/kernel-mode separation of privilege cannot be expected to be available. Hence, a fairly different design is needed for the OS. Sec. 3 presents a first prototype. The main challenge in programming wildly heterogeneous, parallel systems is to master and hide the complexity from upper layers of the stack while preserving the opportunities provided by the underlying hardware. The keys for addressing this challenge are heterogeneous programming interfaces in terms of domain specific languages (DSLs), programming models and runtime systems. These interfaces help separating the concerns

Fig. 1. The Orchestration Stack

The research presented in this article is supported by the German research council (DFG) and the state Saxony through the cluster of excellence “Center for Advancing Electronics Dresden” (cfaed), cfaed.tu-dresden.de.
of core algorithmic problems and possible implementations from structural properties of the architecture and particular properties of the heterogeneous components. Along interfaces, compilers are needed to lower the abstractions and, for example, reason about parallel execution and data layouts. Compilers must identify application resource demands specific to the hardware while exploiting heterogeneous resources. This includes methods for deciding where, when and how to run which parts of the application and where to store data, which, in turn, requires models of the specific heterogeneous resources. Compilers should thus generate alternatives equipped with meta-information for possible mappings of algorithms to resources. Based on the generated meta-information, application runtimes and OS-level resource managers negotiate desired and available resources to find a global schedule of resources that meets all application requirements. Once granted by the OS, the runtime adjusts the application by switching to the respective compiler generated alternative.

To benefit best from the flexibility of heterogeneous computing platforms with dynamic resource (re)allocation mechanisms, application algorithms need to provide a high degree of flexibility. Within Orchestration we work on data base applications (DB), computational fluid dynamics (CFD) and computational biology and use them as drivers for our approach.

Last but not least, we integrate formal methods in our design process to quantitatively analyze low-level orchestration protocols for stochastically modeled classes of applications and systems [4]. The model-based formal quantitative analysis, carried out using probabilistic model checking (cf. [11], [8]), is particularly useful for the comparative evaluation of (existing and future) design alternatives, to compare the performance of heuristic orchestration policies with theoretical optimal solutions or to determine optimal system parameter settings.

3 Towards an Implementation of the Stack

This section provides some details of and pointers to components well fitting to the spirit of the Orchestration Stack. These examples include some of our own work and some others we found in the scientific literature.

SiNW Reconfigurable Circuits: As first steps to replace common CMOS circuits by new technology, we chose to start rebuilding CMOS-based reconfigurable circuits using Silicon NanoWire technology. SiNW technology promises simpler design and manufacturing processes, as it is doping-free, has homogeneous physical and electrical characteristics [17] and is inherently CMOS compatible. In SiNW transistors, polarity is individually controllable via a separate polarity gate. Thus, p-type and n-type transistors can be mixed on the die, which eases wiring constraints and allows for tighter placement.

Making use of this property allowed us to reduce the transistor count for a 6-function programmable logic cell over two inputs from 92 transistors for CMOS-based circuits to 26 [40]. We also improved basic gates like NAND, NOR, X(N)OR, majority/minority and MUX. Starting from this, we observe the effects of these improvements on a larger circuit, an 8-bit conditional carry adder. Using SiNW transistors, the speed can be improved by 25%, the area by 14% and the transistor count by almost 50%.

The Tomahawk Architecture: Tomahawk [2] is a CMOS-based multiprocessor system-on-a-chip (MPSOC) with processing elements and accelerators for digital signal processing and database querying [1], [41], [23]. Processing elements are equipped with local scratch-pad memories and connected via a NoC. The Tomahawk architecture allows for connecting arbitrary, untrusted hardware components, e.g., freely programmable FPGAs. To unify the control over these tiles, each tile is connected to the NoC via a Data Transfer Unit (DTU). The DTU provides controlled message passing and memory access to other networked components. It has two interfaces, one for the untrusted component to access outside memory and to send/receive messages, the other for higher privileged components to control the permissions of these accesses. The only requirement for the untrusted component to access memory and to send messages is the ability to access the DTU registers. Notably, it does not require complex architectural properties such as virtual memory for protection. In addition to the DTU-mechanisms, the Tomahawk provides a logically decoupled processing element called CoreManager, coordinating the processing elements and responsible for the allocation and configuration of processing elements and global memory and tile-to-tile data transfers (similar to [22]).

The M³ Operating System: Similar to other microkernel-based approaches, M³ systems [3] are split into privileged kernels and unprivileged servers and applications. However, unlike in traditional OS approaches, the kernel cannot rely on processor features like user/kernel mode and memory management units to shield itself from applications.

Instead, in M³ systems, one or more M³ kernels run on dedicated and privileged tiles, while servers and applications run on unprivileged tiles. The key for isolation is that only privileged tiles can configure DTUs to, e.g., create communication channels. With this design, arbitrary components can be integrated as tiles and controlled by the M³ kernel. The configuration of DTUs is controlled by means of capabilities modeled after the L4 capability [27] system. Capabilities are created and protected by the kernel and can be exchanged between servers and applications. OS functionality like file systems or network stacks are provided by servers on unprivileged tiles and can be accessed from applications via DTU messages. For example, a file system can be built as an untrusted FPGA.

Heterogeneous Programming Interfaces: As mentioned in Sec. 2, we achieve separation of concerns with interfaces at different levels. As an example at a lower level, we have built a dataflow-based language together with a retargetable compiler that generates optimized code for the Tomahawk architecture [7]. This includes a mapping of actors to heterogeneous processing elements, and of data transfers to the underlying message passing interface over the DTU. At higher levels, we have made significant progress in new algorithms for CFDs [19], [20], a DSL for computational biology [25] and a general skeleton framework [24]. We also see great potential on interfaces that allow describing memory access patterns, as successfully shown by Ben Nun et al [6] using template meta-programming for different GPU architectures.

4 Conclusion

It is much too early to draw conclusions on whether or not the Orchestration Stack will successfully enable the effective and efficient usage of novel post-CMOS technologies. However, our initial experiences in building parts of the stack - mostly restricted to CMOS - did not expose any obvious show stoppers.

References


Quantum Computing and Moore’s Law Are Becoming Entangled

Steven P. Reinhardt, Murray C. Thom, and Mark W. Johnson, (Member, IEEE)

Abstract—Quantum computing (QC) offers the potential of astounding performance growth by harnessing quantum effects for our calculations. Over the past 10 years the first quantum computing systems have been scaling in size exponentially alongside Moore’s Law in a branch of QC called quantum annealing (QA). The rapid growth in performance has been matched by a rapid growth in software inviting a new set of pioneers to apply the systems to industrial applications.

1 RAPID PROGRESS

Quantum computing (QC) harnesses the physics of quantum mechanics to accelerate computations beyond that of classical systems, circumventing the slowdown of Moore’s Law. QC relies on quantum bits, or qubits. A qubit may encode a one or zero just as a conventional bit does, but it may also exist in a quantum-mechanical superposition of one and zero at the same time. Quantum mechanics is usually relevant only in describing the behavior at the atomic or subatomic level, but with careful engineering, its counter-intuitive behaviors can be harnessed in manufactured devices. Quantum annealing (QA) is an approach to QC that bears little resemblance to classical computing, but rather seeks to minimize a programmed cost function by inducing a physical system, whose behavior embodies that cost function, to evolve to its lowest energy state.

Rose and colleagues [1] had the critical insight that a QA processor could be built within a few years and be a powerful learning vehicle for building larger arrays of qubits and couplers, which would be important no matter the eventual architecture, and would expand our knowledge of QC most rapidly, notably in development of algorithms and applications. Prototype D-Wave QA systems have been commercially available since 2011 [2]. Demonstrations in 2013 showed advantages over commercial software [3]. In 2015, tests demonstrated a computational advantage over specialized heuristic algorithms [4]. The growth in system performance has been propelled by a rapid growth in integration scale, with three successive generations of processors comprising 128 qubits in 2011, 512 in 2013, and 1152 in 2015. A strength of QC systems in an era of decreasing transistors on a die is the most recent such system that solves problems that can be posed as Ising models, or, equivalently, quadratic unconstrained binary optimization (QUBO) problems.

Despite the clear advances in performance for QC systems, one still commonly hears predictions that QC will be impractical for another 10 to 20 years. But here one must pay close attention to the different approaches to QC, which include most prominently QA and gate-model QC. Gate-model QC has the potential to enable a broader range of quantum algorithms, but its physical requirements are more demanding, limiting scaling of current gate-model QC prototypes to about 10 qubits. This is what leads gate-model QC advocates to estimate decades of development to reach industrial relevance [5], [6].

By contrast, QA-based QC is the only approach that has demonstrated an exponential increase of integration scale over time. This is largely due to the way the systems use quantum mechanics during computation. Rather than storing information in the delicate quantum states, they use the quantum states to make transitions and can recover from the effects of any noise interruptions. This intrinsic robustness to quantum-mechanical errors has enabled exponential scaling to more than 1000 qubits in current commercially available systems. This empirical progress is driving work on better theoretical models of the scaling performance of QA-based systems.

D-Wave expects improvements in many QA technology areas, including qubit count, QPU interconnectivity, device quality, increased quantum coherence, and problem-to-processor mapping. With greater numbers of more-coherent quantum devices in future QC systems, aspects of gate-model designs can be incorporated to enable a wider set of quantum algorithms.

Early application developers have applied QC systems to neural-net training in deep machine learning [7], to complex strategy optimization in finance [8], and the solution of constraint satisfaction problems [9]. Feedback from these developers provides vital guidance to D-Wave in its development of future systems, but only a few potential applications have been explored in depth. We believe that detailed exploration of a wider variety of potential applications will yield further feedback that will accelerate QC systems’ relevance to industry.

2 HARNESsing QUANTUM ANNEALING

Nishimori et al. [10] and Farhi et al. [11] first noted that QA could exploit quantum mechanical effects with likely computational benefit. The D-Wave 2X™ system, shown in Figure 1, is the most recent such system that solves problems that can be posed as Ising models, or, equivalently, quadratic unconstrained binary optimization (QUBO) problems. The approach
employs a system of coupled superconducting flux qubits, along with classical control circuitry, all fabricated on a superconducting integrated circuit (SCIC) chip [12]. The coupled qubits are designed to behave according to the spin Hamiltonian:

$$H = \sum_i h_i Z_i + \sum_{i,j} J_{i,j} Z_i Z_j$$

(1)

where $h_i$ can be interpreted as a local magnetic field on the $i$th spin, $J_{i,j}$ represents the interaction between spins $i$ and $j$, and $Z_i$ is a Pauli spin operator. QUBO problems can be mapped onto the physical system by appropriate selection of local fields $h_i$ and spin-spin interaction energies $J_{i,j}$. The quantumness of early D-Wave systems was controversial but eventually settled affirmatively, notably by Boixo et al. [13] and Lanting et al. [14].

QA was originally envisioned as an approach to solving hard combinatorial optimization problems [10] [11], and indeed this is a natural application of the algorithm [9]. A real implementation of QA is heuristic and probabilistic, whose effectiveness may be characterized by the probability of returning the best, or a useful, answer. Recently QA has been demonstrated to surpass, in some cases, the performance of tuned heuristic algorithms. In cases where heuristic algorithms are able to get better answers, a QA processor has been shown to take less time to get to an approximate best answer, or within a certain target level of the cost function [4]. When evaluating this time-to-target metric, it appears that the QA processor’s ability to rapidly generate good diverse answers gives it an advantage compared to classical heuristics.

Further, the solutions returned from a QA processor, whether they minimize the cost function or not, fall into a statistical distribution reflecting the physical implementation of the cost function in the QA processor. The realization that this is not simply the Boltzmann distribution of classical statistical physics, but best characterized as a quantum Boltzmann distribution, has opened new possibilities for application of QA, most notably the investigation for use as a quantum Boltzmann machine for training deep learning networks [15].

### 3 Recent Advances

With QC systems already working at significant scale, we can turn to delivering performance strongly differentiated from classical systems. With the technology being young, D-Wave has made strong recent progress on many levels.

The number of qubits continues to increase with each generation of processors, approximately doubling every 12 to 24 months. With the latest D-Wave 2X system using a 250-nm fabrication process, more than an order of magnitude behind the state of the art, there is considerable headroom to increase density before encountering linewidth stagnation.

Following King et al.’s exploration of problem degeneracy and the way it can mask QA performance [16], D-Wave found ways to mitigate degeneracy on certain problem types, increasing D-Wave 2X performance for the hardest problems by a factor of 5 compared with simulated quantum annealing (SQA).

Turning to software tools, the development of a solver that partitions a large QUBO problem into smaller sets of variables has increased the size of problems that can be solved on a D-Wave system by a factor of 10 [17].

With these advances and the delivery of numerous others in active development, D-Wave expects performance to continue its recent rapid growth.

### 4 Relevance for Industrial Applications

The rapid improvement of QC system performance over the last several years must continue for a few more years to deliver differentiated performance to a significant number of customers and users. Most industrial users will expect to use a QC system via high-level interfaces that are relevant to their expertise, e.g., machine learning, optimization, or constraint satisfaction, and thus software tools that map such high-level interfaces advantageously to the QC system will be needed. While this complicates the task of initially delivering QC system performance to a broad audience, as not only best methods for delivering system performance must be discovered but also best methods for delivering such performance to high-level interfaces, if done well it also ensures the sustainability of QC applications growth through multiple generations of perhaps dramatic system evolution. Even with potentially valuable improvements throughout the technology stack, rapid progress vitally depends on feedback from pioneering developers and users of early applications for our systems. With feedback informsed by knowledge of both the early applications and the current state of D-Wave systems, including tools, the system will co-evolve with the applications to achieve industrial relevance as soon as practical.

### 5 Conclusion

D-Wave is using the quantum annealing architecture to drive understanding of diverse technical topics required for any QC architecture, as a first step toward more general applicability. Rapid technological progress in quantum annealing is possible because it is relatively robust, and harnesses a physical system’s natural inclination to remain in its ground state. This unorthodox approach to problem solving appears to be applicable to pattern matching with neural nets and may accelerate progress in those applications in the near future. Guided by vital feedback from early application developers, D-Wave systems will continue to advance rapidly, programming tools and techniques will emerge in the community, and the first industrially relevant quantum computing applications will disrupt the niches in which they compete.
REFERENCES


Solving sparse representation for object classification using quantum D-Wave 2X machine

Nga T. T. Nguyen and Garrett T. Kenyon

Abstract—We use quantum annealing to generate sparse representations of Canny-filtered CIFAR-10 images using binary neurons and randomly-generated kernels that are consistent with the physical architecture of the D-Wave 2X (1152-qubit) computer. Each kernel is replicated with a small stride. The resulting basis is overcomplete and produces a non-convex cost function possessing multiple local minima. Using the lowest energy sparse representations generated by the D-Wave, classification accuracy across ~1,000 images was 44.73%.

1 SPARSE REPRESENTATION OF IMAGES

Finding sparse representations of images is an important problem in computer vision, with applications including denoising, upsampling, compression and object detection. Moreover, sparse coding explains many of the response properties of simple cells in the mammalian primary visual cortex [1]. Given an overcomplete basis, sparse coding algorithms seek to identify the minimal set of generators that most accurately reconstruct each input image. In neural terms, each neuron is a generator that adds its associated feature vector to the reconstructed image with an amplitude equal to its activation. For any particular input image, the optimal sparse representation is given by the vector of neural activations that minimizes both image reconstruction error and the number of neurons with non-zero activity.

When using binary neurons, analogous to applying an L0 sparseness penalty, sparse coding falls into an NP-hard complexity class of decision problems, i.e. solving this optimization problem is at least as hard as a (NP-complete) problem that can be solved by a nondeterministic-Turing machine in polynomial time. Sparse coding with an L0 norm sparseness penalty poses considerable difficulty due to the fact that the energy function $E(\mathbf{b}, \phi)$ is non-convex and contains multiple local minima [2]. Here, we investigate the application of Quantum Annealing to the solution of sparse coding problems. A D-Wave [3] quantum computer is designed to find optimal solutions to a (discrete) Ising quantum system over a large number of variables via quantum annealing.

2 MAPPING SPARSE CODING ONTO QUBO D-WAVE 2X PROBLEM

In the currently available D-Wave 2X chip [3], 1152 qubits are fabricated in 12x12 different unit cells where each unit cell has 8 qubits. This arrangement forms a Chimera structure with dimensions 12x12x8. Due to this Chimera geometry, all the qubits obey restrictive constraints on their bipartite couplings, which means that qubits cannot interact with one another freely. Specifically, each qubit in one of the 12x12 cells from one geometry, say horizontal (each set of 4 qubits in each cell aligned along the horizontal, e.g., see Fig. 2), only can interact with the 4 other qubits from the other geometry in that same cell, say vertical (each set of 4 qubits aligned along the vertical, Fig. 3), to form intra-cell couplings; and this same qubit can only interact with their nearest-neighbors in other unit cells to form inter-cell couplings. Such systems can be described by the following Hamiltonian:

$$H(h, Q, a) = \sum_i h_i a_i + \sum_{i,j} Q_{ij} a_i a_j$$

where $\{a_i\}$ define a set of 1152 qubits of corresponding weights $\{h_i\}$ and coupling strengths called couplers $\{Q_{ij}\}$. This D-Wave Hamiltonian, similar to an Ising model, corresponds to lateral inhibitory interactions between neurons which drives the activity to a local minimum of the sparse approximation cost function [2].

A sparse coding problem with binary neurons corresponds to finding the lowest energy solution to a cost function $E$ for a given input image $\mathbf{I}$:

$$E = \min_{\phi} \left\{ \frac{1}{2} \| \mathbf{I} - \phi \mathbf{a} \|^2 + \lambda \| \phi \|_0 \right\}$$

The first term serves to minimize the difference between the original image $\mathbf{I}$ and the reconstructed image, computed as a sum over features $\phi$ weighted by the binary activation coefficients $\phi$, $a = 0, 1$. The second term serves to maximize the sparsity of the representation by penalizing the total of non-zero activation coefficients (L0 norm).

To map a sparse inference problem onto the D-Wave, neural activations are represented as a vector of coefficients $\mathbf{a}$ such that the input to the set of neurons from the input image is encoded by the bias terms $h_i$, corresponding to convolutions of the underlying image with the associated features $\{\phi_{ki}\}$. On the D-Wave, all neurons are binary and either fire maximally (unit activity) or are silent (zero activity). Thus, neural activations are mapped onto the qubits with spin 1; while all the remaining inactive neurons that give zero contribution to the input im-
age correspond to the remaining qubits with spin 0. In the language of the D-Wave quantum machine the formulation of the sparse coding problem corresponds to the Quantum Unconstrained Binary Optimization (QUBO) form. To map our problem of image reconstruction onto a QUBO formalism, we construct $N_f$ hand-designed or randomly-generated features where $8 \leq N_f \leq 1152$ and these $N_f$ features obey the coupling constraints in the Chimera graph. The 8-hand-designed features visually illustrate this orthogonality. In case these $N_f$ features are randomly generated, we employ Gram-Schmidt algorithm to make them fulfill the Chimera coupling constraints. For example, consider the case $N_f = 8$ kernels as 8 basis feature vectors $\{\psi_i\}$ with $i \in \{1, \ldots, 8\}$ (as an example, see Figs. 2 and 3). These kernels obey the constrains for bipartite couplings in a unit cell of qubits, i.e. $\psi_i \psi_j = \delta_{i,j} |\text{div}(i-1,4) - \text{div}(j-1,4)| = 1$. We then replicate the $N_f$ ($=8$ in this case) $\{\psi_i\}_{12 \times 12}$ times so as to match the arrangement of the D-Wave qubits unit cells. Finally, we convolve all of $12 \times 12 \times 8$ cells $\equiv \phi$, as local “kernels”, with the input image by sliding these kernel throughout the image. Then we will use these new $\phi$ together with input $I$ as translation tools to map our optimization problem onto finding the ground state energy of a classical Ising system. We obtained transformation relations $\{h,Q\}$ as: $h = -\phi^T \tilde{I} + (\lambda + \frac{1}{2})$, $Q = \frac{1}{2} \phi^T \phi$. 

We used convolution [4], [5] in stead of matrix-vector product for the feature and coefficient operators since image statistics are approximately translationally invariant. The restriction to nearest-neighbor couplings between qubits is consistent with the local nature of interactions between neurons. All of the 1152 qubits will be used to model our CIFAR-10 [6] input but note that this is still a small number as compared to the number of neurons used in many sparse coding models. Our implementation on the D-Wave with $24 \times 24$ input patches corresponds to a nominal overcompleteness of $2 (= \frac{12 \times 12 \times 8}{24 \times 24 \times 1} = \frac{1152}{364})$ but this number will increase for smaller patch sizes. Specifically, the overcompleteness is 8 for $12 \times 12$ and 18 for $8 \times 8$ patch sizes, respectively. We use a relative sparseness penalty $\lambda = 0.025$ for our runs.

Due to the cooling process involved in initializing the D-Wave, not all of the 1152 qubits are successfully calibrated in any instantiation (52 qubits not functioning in these runs). All other qubits that are theoretically allowed to interact with these error qubits will share zero couplers. Furthermore, there are a few missing couplers between functioning qubits that should otherwise exist. In total, this instantiation provides 1100 active qubits and 3068 couplers. We used images drawn from the CIFAR-10 database, consisting of natural thumbnail images depicting one of ten distinct objects categories (4 vehicles and 6 animals) captured in 50,000 tiny $32 \times 32$ squares. Thumbnail images are appropriate given the limited number of qubits on the D-Wave 2X computer and classification performance on the CIFAR-10 dataset has been extensively benchmarked via a wide variety of image processing techniques. Edges are important features for object recognition because they play a key role in the detection of discontinuities in brightness. To better align our sparse coding problem to the limited capacity of the D-Wave 2X computer using the 8 hand-designed (middle) and 32 randomly-generated (bottom) kernels. The images shown belong to the four CIFAR-10 object categories, from left to right: AIRPLANE, AUTOMOBILE, SHIP, and TRUCK. Wave 2X, we apply a Canny filter that omits color information and interior structures that in our experiments would have required a larger machine to properly account for. Using Canny filtered gray-scale images significantly reduces the amount of information the sparse representation must encode. Sparsity was approximately 50%, giving the fraction of neurons being active in the D-Wave 2X sparse representation of each image on average.

3 Results and Discussion

We obtain sparse representations using binary neurons and features vectors consistent with the D-Wave architecture for a part of CIFAR-10 training images (Fig. 4). We train a linear classifier (linear SVM) to classify the sparse representations of the images after training on the sparse representations of ~1,000 training images (see Fig. 1). The classification accuracy for 10 CIFAR-10 classes on this 1K-image training set is 44.73%. Note that this result is obtained on 1) a relatively small portion (~2%) of the entire CIFAR-10 dataset, 2) Canny-filtered, grayscale, $24 \times 24$ center cropped images in contrast with the original RGB image inputs.

References


Experiences with Scheduling Problems on Adiabatic Quantum Computers

Tobias Stollenwerk and Achim Basermann, German Aerospace Center (DLR), Linder Höhe, Cologne, Germany
E-mail: {Tobias.Stollenwerk, Achim.Basermann}@dlr.de

Abstract—The realization of quantum computing models is nowadays a grand challenge regarding supercomputer architectures which may potentially exceed the scaling limits of Moore’s Law. The Canadian company D-Wave Systems Inc. developed a hardware for quantum annealing which is well suited for solving combinatorial optimization problems. We investigate the formulation of a scheduling problem for quantum annealing and report on our experience with the D-Wave system at NASA Ames.

1 INTRODUCTION
Planning and scheduling problems are usually hard combinatorial optimization problems. Therefore new hardware architectures with potential supremacy above classical approaches, like quantum devices, are worth studying.

The realization of quantum computing models is nowadays a grand challenge regarding supercomputer architectures which may potentially exceed the scaling limits of Moore’s Law [2]. The Canadian company D-Wave Systems Inc. developed the first commercially available quantum annealer. Quantum annealing is well suited for solving combinatorial optimization problems with quadratic objective function over binary variables without constraints (QUBOs). The execution of QUBO problems on a D-Wave machine requires a mapping of the problem to the interconnection topology of the hardware, the so-called Chimera graph.

2 SATELLITE SCHEDULING
Hard planning and scheduling problems as they appear in aerospace research can be mapped and solved on quantum optimizers [4]. In this paper, we will focus on a problem from the field of satellite mission planning based on [3]. The planning of satellite missions can be mapped to a state machine [5]. The goal is to achieve a certain mission objective while obeying several boundary conditions (charge, data storage, etc.).

2.1 Exemplary Formulation for quantum annealers
The D-Wave quantum annealer can be regarded as a heuristic solver of combinatorial optimization problems of the QUBO form

\[ Q(x) = \sum_i h_i x_i + \sum_{ij} J_{ij} x_i x_j , \]

where \( x_i \) are binary variables. A QUBO is comprised of linear and quadratic terms. It can be represented as an undirected graph in the following way: Each binary variable is represented as a node in the graph. The coefficients of the linear terms are assigned to each node and each non-vanishing coefficient of the quadratic terms is represented as a weighted edge between two nodes. However, the hardware does not allow for arbitrary connections between nodes since the D-Wave chips have a Chimera graph design [4]. This problem can be overcome by representing a logical qubit \( x_i \) by several physical qubits on the chip. All of these physical qubits are coupled together ferromagnetically in order to make sure that the physical qubits representing a logical qubit agree on a value. This procedure is called embedding. The result is another QUBO with reduced connectivity to fit onto the Chimera graph hardware. The more the graph representing the problem differs from the Chimera graph, the more physical qubits are needed. In [1] it is shown that in the case of a complete graph of size \( N \) the number of physical qubits is \( \mathcal{O}(N^2) \). An example of the embedding of a complete graph of size 10 is shown in figure 1.

2.2 QUBO formulation for satellite scheduling
In this section, we will derive a QUBO formulation for a simplified version of the model from [5]. The resulting model exhibits some similarities to the Mars lander mission planning done in [4], [6]. We assume the satellite can occupy three states: charging \( (c) \), downlink \( (d) \) and experiment \( (e) \). We discretize the time and assume time steps \( t \in \{0, 1, \ldots, T\} \). The variable \( x_{st} \) tells us if the satellite is in the state \( s \in \{c, d, e\} \) at time \( t \). With this, the time sequence of these variables represents the schedule we want to optimize. The optimization goal is
to record as much data as possible during the mission. There are two satellite variables which may change over time: The charge of the battery $C$ and the data stored on the memory $D$.

The rate with which these variables are changing depending on state $s$ are denoted by $c_s$ and $d_s$ respectively. For example the experiment state will increase the data $d_s > 0$ and decrease the charge $c_s < 0$. Both the battery and the memory define an upper and lower limit for the charge and the data, respectively.

Not every state can be occupied at each instance in time. For example the charging through solar panels is only possible in the sunlight, or the downlink is only possible in the vicinity of a ground station. Therefore for each state $s$ there is a subset of times $\tau_s \subseteq \{0,1,\ldots,T\}$ at which the satellite can occupy this state. To enforce this constraint, we remove all the variables $x_{st} \in \{x_{st}|t \in \tau_s\}$. For the sake of simplicity, we assume that each state has minimum duration of 1.

The QUBO $Q = \sum_i Q_i$ is comprised of the following contributions:

1) At each time step the satellite can only occupy a single state. This is enforced by

$$Q_1 = p_1 \sum_i \left( \sum_s x_{st} - 1 \right)^2 .$$

2) At all times, the charge must be in between the upper $c_{\max}$ and lower $c_{\min}$ limit of the battery.

$$c_{\min} < c_0 + \sum_s c_s \sum_{\tau < t} x_{st} < c_{\max} .$$

Here, $c_0$ is the charge at the mission start. In order to enforce this inequality, we need to introduce slack variables

$$y_t := c_0 + \sum_s c_s \sum_{\tau < t} x_{st} - c_{\min} \in \{0, c_{\max} - c_{\min}\} .$$

We need to represent these slack variables in terms of binary variables:

$$y_t = \sum_{\alpha} 2^{y_{t\alpha}} .$$

The contribution to the QUBO reads

$$Q_2 = p_2 \sum_t \left( c_0 + \sum_s c_s \sum_{\tau < t} x_{st} - c_{\min} - \sum_{\alpha} 2^{y_{t\alpha}} \right)^2 .$$

3) Analogously, we can obtain the contribution from the memory constraint as

$$Q_3 = p_3 \sum_t \left( d_0 + \sum_s d_s \sum_{\tau < t} x_{st} - d_{\min} - \sum_{\alpha} 2^{z_{t\alpha}} \right)^2 .$$

4) The contribution which enforces the maximal downlink reads

$$Q_4 = -p_4 \sum_s d_s \sum_{t=1}^T x_{st} .$$

The penalty weights $p_i$ need to be chosen in such a way that the hard constraints are fulfilled, i.e.,

$$\sum_{i=1}^3 Q_i = 0 ,$$

and the value of $Q_4$ is as small as possible.

### 3 Experiments on D-Wave’s 2X System

Due to the statistical nature of the machine, multiple runs are necessary to solve a QUBO with a certain success probability. Usually one solves the same the problem thousands of times before investigating the statistics. The success probability $p$ is then given by

$$p = \frac{N_{\text{successful}}}{N_{\text{total}}} ,$$

where $N_{\text{successful}}$ and $N_{\text{total}}$ are the number of successful and total runs, respectively. As it is done for example in [4], one typically uses the expected run time to obtain a 99% success probability

$$T = \frac{\ln(1 - 0.99)}{\ln(1 - p)} T_{\text{Anneal}}$$

as a measure of performance. Here $T_{\text{Anneal}}$ is the run time of a single run of the adiabatic quantum computer. Usually the value is set to values around $T_{\text{Anneal}} = 20\mu$sec.

### 4 Conclusion

We explored the applicability of quantum annealing to a selected space planning problem. For the satellite scheduling problem, we found an adequate QUBO formulation.

Our experiments on D-Wave’s 2X System indicate that this problem can be solved. As common for heuristic solvers, multiple runs and a statistical analysis of the results were necessary to guarantee a high solution quality. Due to the small problem sizes, runs on the D-Wave machine were extremely fast and global solutions to the combinatorial optimization problems were found with very high probability.

Absolute performance and scalability of quantum annealing hardware for general problems is hard to assess from experiments. One reason for this is the fact that only problems of moderate size could be executed on the available D-Wave hardware.

### Acknowledgments

The authors would like to thank NASA Ames for their support during performing the experiments on the D-Wave 2X system, for many valuable discussions and the opportunity to use the D-Wave 2X machine at NASA Ames.

### References


