Toward Codesign in High Performance Computing Systems

[Invited Special Session paper]

Richard F. Barrett
Sandia National Laboratories
Albuquerque, NM, USA
rfbarre@sandia.gov

Sudip S. Dosanjh
Sandia National Laboratories
Albuquerque, NM, USA
ssdosan@sandia.gov

Michael A. Heroux
Sandia National Laboratories
Albuquerque, NM, USA
maherou@sandia.gov

X.S. Hu
University of Notre Dame
Notre Dame, IN, USA
shu@nd.edu

S. Parker
Nvidia, Inc.
Santa Clara, CA, USA
sparker@nvidia.com

J. Shalf
Lawrence Berkeley Nat'l Lab
Berkeley, CA, USA
JShalf@lbl.gov

ABSTRACT

Preparations for exascale computing have led to the realization that computing environments will be significantly different from those that provide petascale capabilities. This change is driven by energy constraints, which has compelled hardware architects to design systems that will require a significant re-thinking of how application algorithms are selected and implemented. The "codesign" principle may offer a common basis for application and system developers as well as architects to work synergistically towards achieving exascale computing. This paper aims to introduce to the embedded system design community the unique challenges and opportunities as well as exciting developments in exascale HPC system codesign. Given the success of adopting codesign practices in the embedded system design area, this effort should be mutually beneficial to both communities.

Keywords

hardware/software codesign, scientific applications, high performance computing, parallel architectures

1. INTRODUCTION

Computational requirements for energy research, national security and advanced science are predicted to require a thousand-fold increase in supercomputing performance during the next decade [16]. However, the transition to exascale high-performance computing (HPC) systems that are operable within affordable power budgets will not be possible based solely on existing computer industry roadmaps [4]. Thus we must not only support an acceleration of industry roadmaps to deliver power efficient architectures but we must also augment this with modifications to, or in some cases rewriting of, applications so that new approaches to hardware design may be utilized at a significantly increased scale [1]. The benefits of doing so are profound in that it impacts the entire computing industry, addressing cross-cutting issues such as energy efficiency, concurrency and programmability for users of single workstations, data centers and large supercomputers. For the users of exascale machines, there will be additional challenges including the scalability and reliability that are brought about by the extreme size of such systems.

It is well accepted that HPC system development practice requires fundamental changes. The "codesign" principle may offer a common basis for application and system developers as well as computer architects to work synergistically towards achieving exascale computing. Codesign has a long track record in the embedded system design community and plays a key role in delivering performance and energy efficiency for cost-conscious consumer devices. However, how to translate the codesign principle into concrete codesign practices still represents a frontier to be explored by the HPC community.

Considerable effort is being invested for developing systematic approaches for architecture and algorithm codesign in the HPC community. For example, a suite of application proxies, called ÒminiappsÓ are being developed to aid the exploration of the architectural design space. Simulation tool suites are being constructed to allow performance/power analysis for different architectures running different algorithm implementations. Recent efforts have identified various means to reconfigure code for extracting the performance potential of heterogeneous computing resources (e.g. GPU-accelerated multicore nodes) for important computations such as sparse matrix operations. Programming models are evolving to ease the burden on the code developer while at the same time focusing attention on key performance issues. At the same time, this work has also identified ways in which architectures could be modified to better support these tasks, including wider registers and faster memory subsystems. Reductions in energy consumption remain a challenging issue for hardware, system software, and algorithm and application developers. All these efforts are

IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2012, November 5-8, 2012, San Jose, California, USA

essential for achieving more cost-effective and efficient HPC system designs.

This paper aims to introduce to the Electronic Design Automation (EDA) community the exciting development toward exascale HPC system codesign. We begin by discussing HPC system codesign challenges and opportunities, and describing important design space considerations. We then elaborate the importance of application proxies and their essential features. Lastly, we share some industry experiences with applying codesign principles in this context. We believe that there are many new research problems to be addressed in this area and that the expertise of EDA researchers can be valuable to tackle some of these problems.

2. CODESIGN CHALLENGES AND OPPORTUNITIES

Exascale computing will require a radical redesign of HPC node architectures to fit within emerging constraints on power, the increasing cost of data movement, and stalled processor clock rates. Applications and algorithms will need to adapt to the evolution of node architectures. The codesign of applications, architectures and programming environments will enable navigation of the increasingly daunting constraint space for feasible exascale system designs to achieve more balanced, superior systems.

Conventional HPC system design involves a pipelined collaboration process that primarily engages in requirements gathering at the input of the design process and evaluation of the result after the product is delivered 4-6 years later. However, the rapid and disruptive changes anticipated in hardware design over this next decade necessitate a more agile development process, such as the hardware-software codesign processes developed for rapid product development in the embedded space. Design methodologies on which we have relied so far never had to consider power constraints or parallelism of the scale being contemplated for exascale systems. Furthermore, the programming model and software environment for future extreme-scale systems is anticipated to be substantially different from the current practice. The designers of HPC hardware and software components have an urgent need for a systematic design methodology that reflects future design concerns and constraints.

The codesign strategy is based on developing partnerships with computer vendors and application scientists and engaging them to participate in a highly collaborative and iterative design process well before a given system is available for commercial use. The process is built around identifying leading edge, high-impact scientific applications and providing concrete optimization targets rather than focusing on speeds and feeds (FLOPs and bandwidth) and percent of peak. Rather than asking "what kind of scientific applications can run on an exascale system" after it arrives, this application-driven design process instead asks "what kind of system should be built to meet the needs of the most important science problems." This leverages deep understanding of specific application requirements and broad-based computational science portfolio. Putting delivered scientific application performance into the driver's seat of the design process is essential to define a common optimization target for all of the design teams – from hardware to applications.

Target application programs often consist of a million source lines of code involving multiple programming languages, third party library dependencies, and other complexities. To address this complexity, we are considering a software model which includes application proxies, providing a tractable means for exploring key issues associated with design goals. Different types of proxies are required to implement different aspects of the codesign process. Table 1 shows several proxies and their definitions. Figure 1 depicts the interplay between the hierarchy of reduced applications in code analysis for codesign.

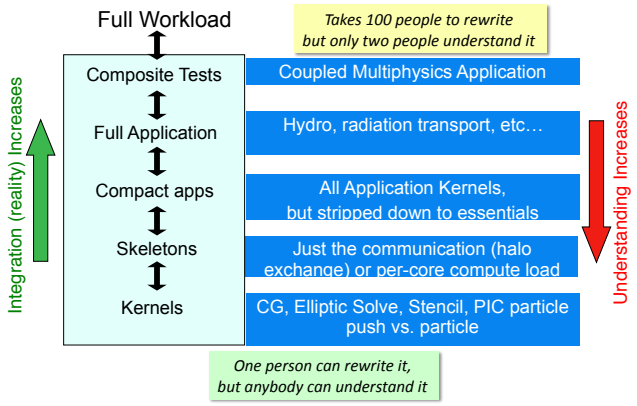

Figure 1: Depiction of the interplay between the hierarchy of reduced applications that we employ in the codesign process. Kernels enable rapid exploration of new languages and algorithms because of ease of rewriting while full applications are harder to rewrite, but ensure adherence to original application requirements.

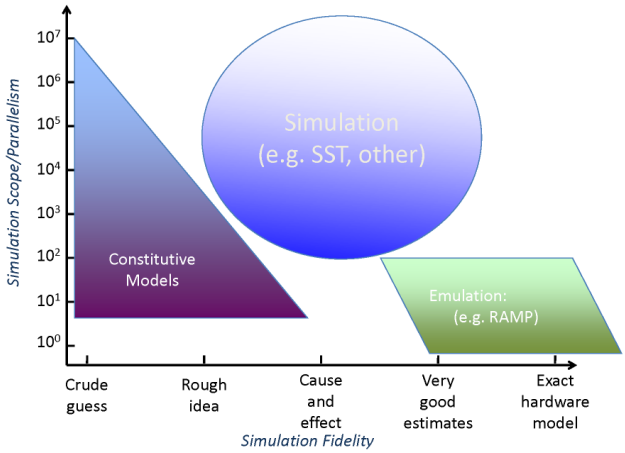

Figure 2: Multiple modeling approaches are required to cover both the scale and accuracy required to understand system design trade-offs. The two key axes for simulation and modeling techniques are fidelity of the model (horizontal axis) and the scale of system you can simulate.

Table 1: Application proxies and their definitions.

Surrogate	Description
Compact app.	Small app. having fewer features and simplified boundary conditions relative to full apps.
Mini-app.	Small, self-contained program that embodies essential performance characteristics of key applications.
Skeleton app.	Captures control flow and communication pattern of app. Can only be run in a simulator.
Proxy app.	General term for all the above "app" approaches.
Mini-driver	Small programs that act as drivers of performance-impacting library packages.
Kernel	Captures node-level aspects of an algorithm

Modeling, simulation, and compiler analysis all play synergistic roles in the codesign process to cover a broad space of design parameters. Figure 2 shows that a multi-resolution approach using multiple modeling methodologies should be employed to cover both the scale of exascale systems and the fidelity required to have confidence in our design choices. Tools such as cycle accurate hardware simulation (the parallelogram in Figure 2) offer extreme detail in their modeling capability, but limit the scale of system that can modeled to node size or small clusters. Constitutive models (the triangle in Figure 2) and other empirical modeling methods can cover much larger systems, but by definition only model effects included as parameters in the model. The majority of modeling is done by empirical models because they are faster to construct and evaluate, but the software-based, and cycle-accurate models are used to verify that the simpler model has included all important effects, and has not neglected anything any essential (but unanticipated) effects.

Architecture simulation (e.g. [11, 9, 6]) provides an avenue to experiment with hardware configurations, programming models, and algorithms on exascale class machines before full implementations of the hardware and software components of a system are available. Models of the hardware, the runtime system, and the application can be integrated into to the simulation to evaluate the trade-offs between design choices imposed on each aspect of the integrated system. Coarse-grained simulations (the circle in Figure 2) allow large-scale systems to be studied in a way that captures the complex interactions between various hardware components, including those interactions which only arise at the largest scales.

The ultimate goal of employing a codesign process is to explore rapidly the vast number of design options (to be discussed more in the next section) and thus dramatically accelerate the design cycle. Figure 3 illustrates the practice as of today and the what we hope to achieve. The success of codesign depends on (i) building integrated design teams that include both application experts working in concert with computer architects and algorithm designers, and (ii) developing sophisticated tools to accelerate the design cycle. The optimization target for the process is the delivered application performance for the combined hardware and software environment. Therefore, it is essential to clearly identify the application up-front to act as the common metric for success for all aspects of the system design during the iterative codesign process.

The codesign process involving a collaboration between an applications team, code analysis, and architectural simulation has been demonstrated for applications ranging from Climate Modeling (Green Flash) [5], Seismic Imaging (Green Wave) [10], and an automated co-tuning process that includes both auto-tuning of hardware and software in the same process [13].

3. DESIGN SPACE EXPLORATION

Given the complexity of constructing large HPC systems and the problems associated with modifying applications to run on them, a dialog needs to be established between computer companies and application developers where feedback is able to rapidly assess and optimize designs as they are created. A systematic way of supporting such a dialog is through a formal design space exploration process. In this process, we envisage assessment based on balancing performance benefit versus cost in terms of software complexity, portability, silicon area, etc. In order for trust to exist in this dialog, a number of approaches might be considered including execution on prototype or early design hardware, the construction of application models including simulators or analytic performance models and an attention to creating solutions that work across a broad range of applications and do not benefit a single problem.

Our methodology for exascale design space exploration [3] includes measurement on prototype hardware, experimentation in the form of re-factoring and re-implementation using a variety of programming models and algorithms and prediction using architectural simulators. To this end, we are investigating several architectural testbeds which are representative of industry trends including Intel's Many Integrated Core (MIC) processors, GPUs from NVIDIA, Fusion APUs from AMD and nodes from Convey and Tilera. Such studies are providing useful feedback to computer architects, application developers and algorithm researchers. Our experimentation is also wider than just hardware, including

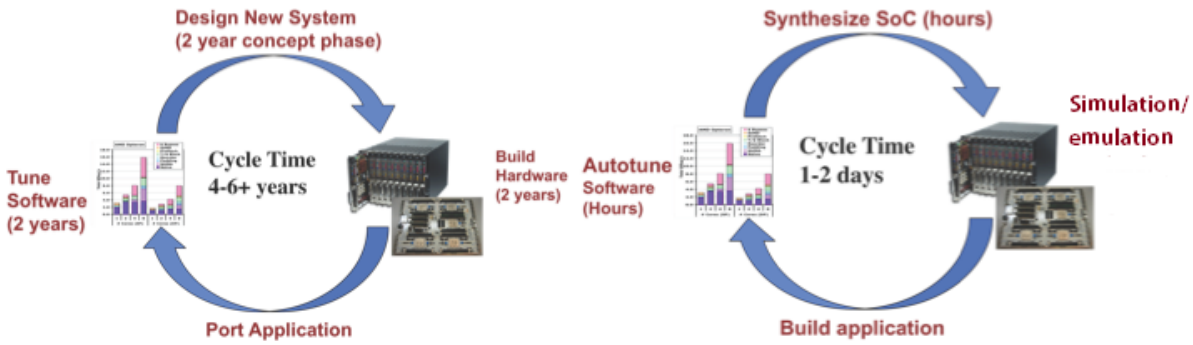

Figure 3: Conventional design cycles last 4-6 years. With rapid synthesis tools to generate prototype designs, FPGA-accelerated emulation and software auto-tuning, we hope to get the codesign cycle time down to 1-2 days.

evaluation of execution models such as ParalleX and low-level activities such as direct measurement of energy use in contexts such as the variation of network injection bandwidth.

A fundamental hurdle to design space exploration is the scale of modern scientific applications, which is a limiter for rapid prototype assessment. Applications are typically millions of lines of source and are written to use complex algorithms and data structures. Whilst our eventual goal is to run applications on such large systems, the effort required to port them is likely to be prohibitive if multiple platforms must be assessed in short time frames. It is in this context that the notion of a mini-application (or simply miniapp), i.e., application proxy, has been developed – a miniapp is a condensed implementation of one or multiple key performance issues that affect parent codes, written to be amenable to refactoring or change but representative enough to be useful in the scientific problem domain.

The ability to predict the performance, and more importantly the performance limitations, of hardware which does not currently exist or is significantly different from contemporary systems is a key facet of design exploration. Since many proposed exascale point designs are currently proprietary or encumbered with intellectual property, many of our early evaluations are being conducted using the notion of an Abstract Machine Model (or AMM) which defines the key architectural building blocks but no specific detail. We then are able to augment AMMs with details provided by performance models, architectural simulators and information obtained from our miniapps running on test-bed platforms to inform us of performance trade offs and available design decisions.

In order to convey confidence in our use of application proxies, we have developed a validation methodology [2] for demonstrating the applicability of miniapps to their parent codes enabling HPC vendors and researchers to have a high degree of confidence in results obtained from studies using miniapps. This methodology is guided by the principles developed for experimental validation [14], where the application is analogous to the physical observation and the miniapp is the experiment. A more detailed discussion on miniapps will be given in the next section. A similar effort is being applied to hardware/software co-simulation tools, and initial validation results of SST are encouraging.

A variety of advanced architecture testbeds are being used to study performance issues of key algorithms. These studies are helping guide algorithms research and are providing useful feedback to computer architects. For example, excellent performance was obtained for a very challenging finite-element miniapp on a Nvidia GPU. Register spilling was identified as a key issue that must, in the future, be overcome through architecture and system software enhancements, as well as through algorithms research. (See Section V for more detail on this example.)

Simulators like the SST, when combined with application proxies, allow fast and efficient design space exploration. This process can be used to make decisions such as which memory technology and processor core is best suited to a given problem. For example, one such experiment looked at the impact of different memory technologies (DDR2, DDR3, and GDDR5), and processor core issue widths (1, 2, 4, 8 wide issue) on the power, performance, and cost of a sparse linear system solver captured within the Mantevo HPCCG miniapp as well as Lulesh[1], a proxy for a challenging shock hydrodynamics code.

Network bandwidth degradation studies on supercomputers are helping define network requirements for future systems. These studies are also helping identify methods for improving application performance.

4. MINIAPPS: VEHICLES FOR CODESIGN

Applications programs for computational science and engineering (CSE) designed to execute in HPC environments are presently based on a distributed memory parallel computing model using the Message Passing Interface (MPI) for exchanging data between computational nodes. This general approach has been highly successful for application areas such as climate modeling, material science, computational mechanics and many other areas. For more than 15 years

[1] https://computation.llnl.gov/casc/ShockHydro/

we have enjoyed a stable and improving execution environment where, from generation to generation, the number of nodes and the performance of each core on a node increased regularly and sufficiently to keep pace with computational growth needs.

Presently this model has stagnated and even declined. Node counts are not expected to increase dramatically and single core (sequential) performance has in fact declined. The only true path to increased performance at this time is to exploit new trends in intra-node parallelism. However, most HPC applications have no existing infrastructure to exploit this parallelism, and no well-defined path to its realization.

To aid in the rapid exploration of this new design space, we have developed a growing collection of "mini-applications", or miniapps. These are small stand-alone programs that embody some key performance-impacting elements of the large-scale applications of interests, such that we can rapidly re-design and re-implement the miniapps and learn about the performance, programmability and system design tradeoffs in a rapid and agile cycle. Unlike a benchmark, the result of which is a metric to be ranked, the output of a miniapp is a richer set of information, which must be interpreted within some, often subjective, context. We distinguish this from a compact-application whose purpose is to replicate a complex domain-specific behavior being used in a parent application. Miniapps are designed specifically to capture some key performance issue in the full application but to present it in a simplified setting which is amenable to rapid modification and testing. This is also distinct from a skeleton application, which is typically designed to focus on inter-process communication often producing a "fake" computation. Miniapps create a meaningful context in which to explore the key performance issue, are developed and owned by application code teams, are limited to $O(1K)$ source lines of code, and are intended to be modified with the only constraint being the continued relevance to parent applications.

Miniapps distill from large-scale applications the most important design choices that must be made, and their refactored versions serve as examples of how the large-scale applications can be rewritten [2]. From the Mantevo project [7], for example, a finite-element miniapp, miniFE, tracked the sensitivity to memory bandwidth of Charon, an electronics device simulation application [12]; molecular dynamics miniapp illustrated the computation of a Lennard-Jones force calculation in LAMMPS [15]; and a finite difference miniapp illuminated an inter-process communication scaling problem, seen only at very large scale, for a multi-material Eulerian code [8]. Just as important, this methodology also illustrates areas in which these miniapps did not adequately represent the full application.

Furthermore, miniapps provide system designers with concrete examples of how applications are written, so that design choices can be informed in a very tangible way by application needs. Finally, miniapps provide a concrete forum for detailed and precise communication between developers of all components of the computing platform: applications, libraries, compilers, runtime, OS and hardware. Use of miniapps has substantially increased the effectiveness of the combined efforts of these teams.

5. INDUSTRY EXPERIENCE IN CODESIGN FOR HPC

In this section, we discuss as an example a recent success in codesign. The example involves the exploration of implementation options for the code MiniFE from the Mantevo suite. The codesign process for this example is limited to node-level design with the focus on the interplay between algorithm development and architecture exploitation. The specific application shows sensitivity to memory bandwidth but rather than just asking for bandwidth we sought out to understand why it is sensitive to bandwidth and to explore other options for improving performance.

We focused on key performance limiters in the matrix assembly phase of the algorithm on an NVIDIA GPU using the CUDA programming model. A straightforward CUDA port was created that uses a single thread per element which adds a local element contribution into a global matrix. The computation of the element operator involves a number of floating-point heavy operations including computing the matrix determinant and the Jacobian. The large number of floating point operations suggest that the performance should be FLOP limited but analysis using NVIDIA's compute profiler has shown that the performance is in fact bandwidth bound due to register spilling.

The cause of this register spilling was identified as the element operator which requires a large thread state - 704 bytes total for node-IDs, node coordinates, the diffusion matrix, and the source vector, plus additional for temporary data and pointers. The Fermi GPU architecture supports up to 63 32-byte registers per thread limiting the total register storage to 252 bytes. As a result of this limit, any additional state must be spilled to L1 and L2 caches but with 8192 simultaneous active threads, most of the spills are not contained by these caches. Consequently, registers are spilled to global memory causing the computation to become bandwidth bound.

As an alternative design, we tuned the kernel to reduce register usage, including algorithmic changes that exploit symmetry in the diffusion operator and reorder computations so that data is loaded immediately prior to being used. We have also applied several traditional optimization techniques including pointer restriction, inlining of functions, and unrolling of loops. Finally, we also position a portion of the state in shared memory and experimented with L1 cache sizes. Whilst these optimization greatly reduce register spilling, 512 bytes of state is still spilled per thread. To ensure fair comparison, all optimizations that were applicable to the original CPU code were back ported also improving the CPU performance.

The performance of the CUDA version of miniFE was compared to the MPI-parallel version of miniFE, with both versions running on a Tesla M2090 and a hex-core Intel Xeon 2.7GHz E5-2680. We tested for various problem sizes of N hexahedral elements. The speedup for each of the phases of the miniFE is reported in Figure 4. The data show that algorithm-architecture codesign provides different speedups for different phases of the program, and a comprehensive design space exploration is needed.

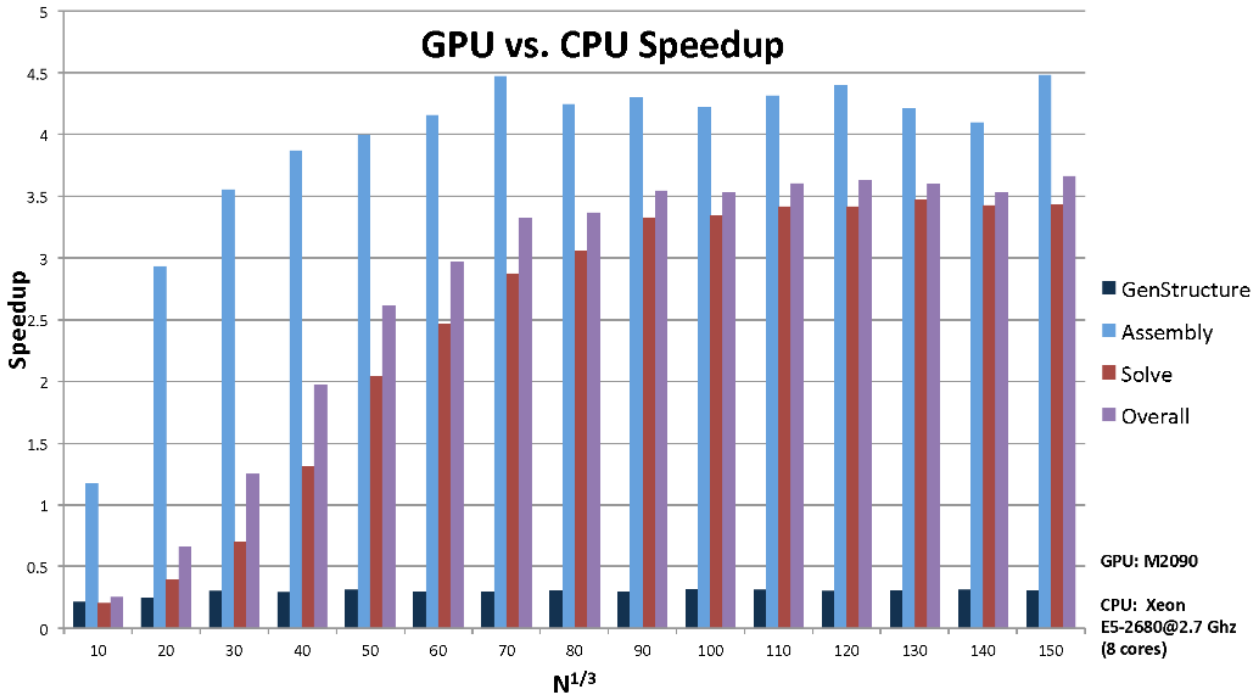

Figure 4: Speedup of CUDA version of miniFE over the MPI-parallel version of miniFE. Both implementations are run on a system consisting of a Tesla M2090 and a hex-core Intel Xeon 2.7GHz E5-2680.

Future generations of NVIDIA GPUs are expected to address some of the findings from the above example study, including an increased number of registers per thread and increases in the size of L1 and L2 memories. Improvements in the CUDA compiler may also lead to a reduction in the number of register spills or the impact that register spills will have on execution time. Nonetheless, the application improvements are still likely to be beneficial. We expect that these investigations will also feed forward into subsequent GPU design cycles, thus completing the codesign feedback loop.

6. SUMMARY

The accelerated push to exascale, driven by mission requirements spanning a breadth of areas, has compelled the development of a new means to develop HPC environments for application scientists and engineers. The codesign approach, inspired by the embedded system design community, is providing the HPC community with an opportunity to strengthen its collaborative interactions internally and externally.

In this paper we have described various challenges and opportunities throughout the HPC codesign space, and have discussed the development effort for enabling this codesign practice. We believe that there are many new research problems to be addressed in this area and that the expertise of EDA researchers can be valuable to tackle these problems.

Acknowledgements

The work described in this article was supported by the US Department of EnergyÕs Office of Advanced Scientific Computing Research. Lawrence Berkeley National Laboratory is supported by the DoE Office of Advanced Scientific Computing Research under contract DE-AC02-05CH11231.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

X. Sharon Hu would like to acknowledge the support from Sandia National Laboratories for this work.

7. REFERENCES

[1] J. Ang et al. High Performance Computing: From Grids and Clouds to Exascale, chapter Exascale Computing and the Role of Co-design. IOS PressInc, 2011.

[2] R.F. Barrett, P.S. Crozier, S.D. Hammond, M.A. Heroux, P.T. Lin, T.G. Trucano, and C. Vaughan. Assessing the Validity of the Role of Mini-Applications in Predicting Key Performance Characteristics of Scientific and Engineering Applications. Technical Report SAND2012-TBD, Sandia National Laboratories, 2012. In preparation.

[3] R.F. Barrett, D.W. Doerfler, S.S. Dosanjh, S.D. Hammond, K.S. Hemmert, M.A. Heroux, P.T. Lin, J.P. Lutjiens, K.T. Pedretti, A.F. Rodrigues, , and T.G. Trucano. Exascale Design Space Exploration and

Co-design. Under review, 2012.

[4] ITRS International Roadmap Committee. International Technology Roadmap for Semiconductors, 2011.

[5] David Donofrio, Leonid Oliker, John Shalf, Michael F. Wehner, Chris Rowen, Jens Krueger, Shoaib Kamil, and Marghoob Mohiyuddin. Energy-efficient computing for extreme-scale science. Computer, 42:62–71, 2009.

[6] S. D. Hammond, G. R. Mudalige, J. A. Smith, S. A. Jarvis, J. A. Herdman, and A. Vadgama. WARPP: a toolkit for simulating high-performance parallel scientific codes. In Proceedings of the 2nd International Conference on Simulation Tools and Techniques, Simutools '09, pages 19:1–19:10, ICST, Brussels, Belgium, Belgium, 2009. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[7] M.A. Heroux et al. Improving Performance via Mini-applications. Technical Report SAND2009-5574, Sandia National Laboratories, September 2009. https://software.sandia.gov/mantevo/.

[8] E.S. Hertel, Jr., R.L. Bell, M.G. Elrick, A.V. Farnsworth, G.I. Kerley, J.M. McGlaun, S.V. Petney, S.A. Silling, P.A. Taylor, and L. Yarrington. CTH: A Software Family for Multi-Dimensional Shock Physics Analysis. In Proceedings, 19th International Symposium on Shock Waves, 1993.

[9] Curtis L. Janssen, Helgi Adalsteinsson, Scott Cranford, Joseph P. Kenny, Ali Pinar, David A. Evensky, and Jackson Mayo. A simulator for large-scale parallel architectures. International Journal of Parallel and Distributed Systems, 1(2):57–73, 2010.

[10] Jens Krueger, David Donofrio, John Shalf, Marghoob Mohiyuddin, Samual Williams, Leonid Oliker, and Franz-Josef Pfreundt. Hardware/software codesign for energy-efficient seismic modeling. In Proceedings of SC2011, 2011.

[11] Sandia National Laboratiries. Structural Simulation Toolkit (SST). http://www.cs.sandia.gov/sst/.

[12] Paul T. Lin, John N. Shadid, Marzio Sala, Raymond S. Tuminaro, Gary L. Hennigan, and Robert J. Hoekstra. Performance of a Parallel Algebraic Multilevel Preconditioner for Stabilized Finite Element Semiconductor Device Modeling. Journal of Computational Physics, 228(17), 2009.

[13] Marghoob Mohiyuddin, Mark Murphy, Leonid Oliker, John Shalf, John Wawrzynek, and Samuel Williams. A design methodology for domain-optimized power-efficient supercomputing. In SC '09: Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, pages 1–12, New York, NY, USA, 2009. ACM.

[14] W.L. Oberkampf, T.G. Trucano, and C. Hirsch. Verification, Validation and Predictive Capability in Computational Engineering and Physics. In Hopkins University, pages 345–384, 2002.

[15] S. J. Plimpton, R. Pollock, and M. Stevens. Particle-mesh ewald and rrespa for parallel molecular dynamics simulations. In Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing, March 1997.

[16] R. Stevens, A. White, S. Dosanjh, et al. Scientific Grand Challenges: Architectures and Technology for Extreme-scale Computing Report. Technical report, 2011.