Energy Introspector: Standard Physical Library Interface for Full-System Microarchitecture and Multi-Physics Simulations

William J. Song[†], Saibal Mukhopadhyay[†], Arun Rodrigues[‡], and Sudhakar Yalamanchili[†]
[†]School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332
[‡]Sandia National Laboratories, Albuquerque, NM 87185

I. Challenges and Motivation

Modeling and simulation of future microarchitectures and applications require more than performance measurement and estimation. Analysis must be "holistic" including power, energy, thermal, and reliability concerns since these physical constraints and their coupled interactions have become major performance determinants of processors. Various models of different physical phenomena have been developed and released in tools such as 1) power models in Cacti/McPAT [8], [18], Orion/DSENT [5], [17], DRAMsim [10], 2) thermal models of HotSpot [4] and 3D-ICE [14], 3) wear models including negative-bias temperature instability (NBTI), time-dependent dielectric breakdown (TDDB) [15], [16], and 4) other proprietary or custom models [2], [19]. However, they are not generally integrated with microarchitecture simulators. We note that as a general practice these models are independently employed, for example via the prevalent use of offline trace-driven simulations, in which a trace generated from a simulator or tool drives another modeling tool as independent simulations shown in Fig. 1-(a). Such a practice is oblivious to the interacting nature of multiple physical phenomena and fundamentally limits studies of advanced research problems lying at the intersection of performance, energy/power, thermal/cooling, and reliability. To address this, we have been developing the *Energy Introspector* (EI) [11], [12] and integrating it with different simulation frameworks including *Manifold* [20], *MacSim* [7], *Structural Simulation Toolkit* (SST) [9], and IBM POWER7+ Architecture proprietary simulators. It is our position that the Energy Introspector will enable the holistic analysis which future processors will require. The EI has proven its ability to work with multiple simulation infrastructures, and its unique capability demonstrates a path forward for community-wide adoption. In this paper, we report on the design and motivate the use of an integrated physical modeling library interface, central to building full-system application-microarchitecture-physics co-simulation environment.

II. Our Approach and Design

The primary goal of the Energy Introspector (EI) [11], [12] is to enable the holistic analysis across microarchitecture, application, power, energy, thermal, and reliability, as illustrated in Fig. 1-(b). The integration of the EI with microarchitecture simulators builds full-system simulation environment including all these component models. The EI provides a standard library interface for the integration of various distinct physical modeling tools. Each physical model is encapsulated into a standardized class called *library*; for example the HotSpot thermal model [4] is encapsulated into a library class. The EI transparently handles multi-physics interactions by transferring and synchronizing data between the libraries inside the framework, driven by simple user application programming interface (API) functions such as calculate_temperature(). Integration of physical models via standardized APIs in the EI enables the users to simply choose the most appropriate models to configure and

Acknowledgement: This research was supported by the Semiconductor Research Corporation under task #2084.001, IBM/SRC Graduate Fellowship, and Sandia National Laboratories.

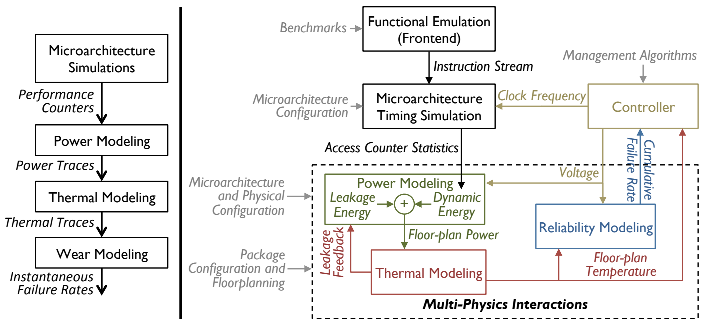

Fig. 1. (a) General practices of using physical models via trace-driven simulations. Each model forms an independent stage of simulations in an open loop manner [3]. (b) Full-system simulation with multi-physics interactions. All models are concurrently simulated within the same framework.

simulate different processor and package designs. In addition, any new or updated physical models can be added to the EI without creating cross-dependencies with existing models, while the physical interactions are captured via the standard library interface, as shown in Fig. 1-(b). This design significantly eases its integration into microarchitectural simulation infrastructures, making it particularly seamless with componentized simulation frameworks such as SST [9] and Manifold [20].

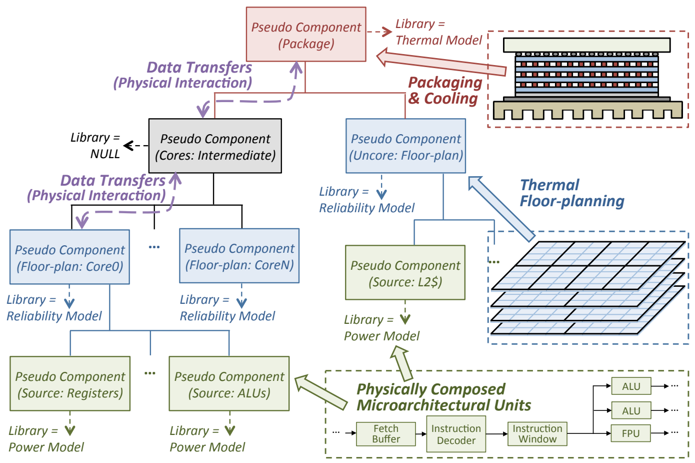
Pseudo Component (Package)
Library = Thermal Model
Data Transfers (Physical Interaction)
Packaging & Cooling
Library = NULL
Pseudo Component (Cores: Intermediate)
Pseudo Component (Uncore: Floor-plan)
Data Transfers (Physical Interaction)
Library = Reliability Model
Thermal Floor-planning
Pseudo Component (Floor-plan: Core0)
Pseudo Component (Floor-plan: CoreN)
Pseudo Component (Source: L2$)
Library = Reliability Model
Library = Reliability Model
Library = Power Model
Pseudo Component (Source: Registers)
Pseudo Component (Source: ALUs)
Physically Composed Microarchitectural Units
ALU
ALU
FPU
Fetch Buffer
Instruction Decoder
Instruction Window
Library = Power Model
Library = Power Model

Fig. 2. In the Energy Introspector, a processor is described as the hierarchy of pseudo components that represent different levels of processor abstraction. Pseudo components are associated with libraries that encapsulate individual physical models being simulated. The Energy Introspector manages data transfers and synchronizations between pseudo components, as illustrated in Fig. 1-(b).

The physical properties of a processor, and hence choice of physical models, are defined at different levels of processor abstraction. For instance, power is typically characterized at the level of functional components based on their circuit-level models (e.g., caches with SRAM models) in proportion to microarchitectural activities (e.g., performance counters), but temperature is calculated by a package-level model based on power distributions on source dies. Therefore, there has to be a method to construct this physically represented hierarchy of a processor and associate each component in the hierarchy with an appropriate library and its constituent physical model. These physically represented components in the EI are called *pseudo components*. A processor is defined as the hierarchy of these pseudo components (see Fig. 2), and the tree is composable to configure different processor or package designs. A pseudo component itself does not represent any functional or physical unit, but its role depends on which library is associated with it. Technically all pseudo components function in the same way, but they have different availability of data depending on where they are located in the hierarchy and which libraries are associated with them. The use of *pseudo component hierarchy* to represent physical constitution of a processor enables seamless interconnection between multiple libraries that encapsulate different individual models.

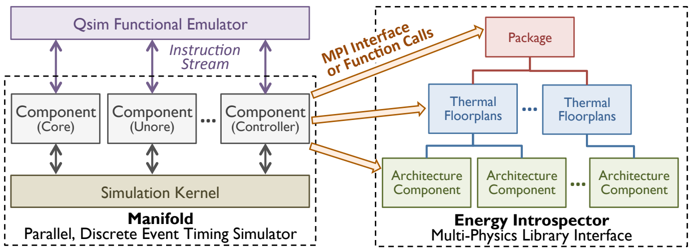

Fig. 3. Integration of *Energy Introspector* multi-physics library interface with *Manifold* microarchitecture simulator. The EI provides a set of API functions to be called by the microarchitecture simulator.

The Energy Introspector is driven by microarchitecture simulations. It provides a set of user API functions to be called by the simulators. Fig. 3 shows an example of full-system application/OS-microarchitecture-physics co-simulation environment based on the integration of Qsim [6], Manifold [20], and Energy Introspector [12]. The Qsim instance performs functional executions of application binaries on a Linux kernel, and the component timing models of Manifold fetch and simulate the execution of instructions generated from Qsim, as also shown in Fig. 1-(b). Each Manifold component collects activity statistics (e.g., access counters) of microarchitectural units being physically modeled in the EI for power calculations. At every defined sampling interval, the Manifold components call the EI functions (either via function calls or MPI interfaces) to calculate the power of microarchitectural units. On the reception of requests, the EI calculates the power of corresponding pseudo components based on provided activity statistics, and the pseudo components are updated and synchronized in the tree (see Fig. 2). Any pseudo components (e.g., floor-plan or package-level pseudo components that are not even associated with power models) can be probed to retrieve the power data since they are all internally synchronized within the EI framework. After power calculations are completed for the given sampling interval, temperature calculation is invoked. The EI ensures correct calculations of models by detecting errors when pseudo components have asynchronously time-stamped data. Re-calculations of models (e.g., temperature and leakage power feedback, dynamic voltage-frequency scaling, etc.) are internally managed when data are transferred and synchronized between pseudo components. Therefore, microarchitecture simulators are only required to call a set of simple computational and probing functions. This has enabled the construction of a simple, standardized API for multi-physics modeling [11].

III. Assessment

*1) **Challenges**:* We address three main challenges; i) application-driven multi-physics modeling and interactions (i.e., power/energy, thermal/cooling, and reliability) in heterogeneous multicore processors, ii) coupling the impacts of these physical phenomena to microarchitectural performance, and iii) facilitation of using various physical modeling tools with microarchitecture simulation frameworks.

*2) **Maturity**:* This standard library approach has been applied to several application/OS-microarchitecture-physics modeling problems. For example, Xiao et al. [21] explored how much leakage power saving can be achieved by employing microfluidic cooling in 3D ICs, and presented a method of optimizing cooling system physical geometry (i.e., the pin fin dimensions) that maximizes the energy efficiency and leakage saving (see Fig. 4-(a)). The unique aspect is that the EI provided environment to easily measure the tradeoffs between energy, performance, thermal, and coolant flow rate based on full-system simulations of application binaries.

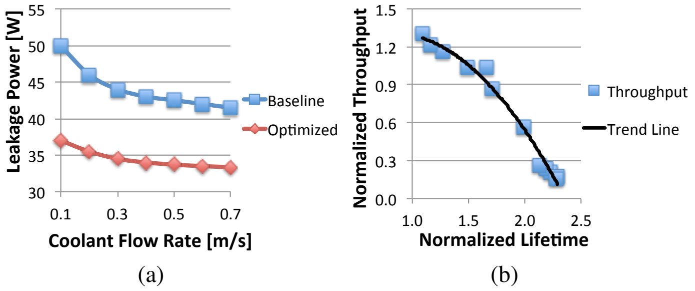

Fig. 4. Case Study (a): leakage reduction by optimizing pin fin geometry in 3D-ICs [21]. Case Study (b): characterization of throughput vs lifetime reliability tradeoff in multicore processor [13].

*3) **Uniqueness**:* Existing approaches have developed point solutions for specific combinations (e.g., microarchitecture-power simulations). However, as point solutions they are not easily extensible to include "other physical models" or configure "different microarchitectural designs". Nor have these approaches attempted to address the software engineering aspects of such modeling problems. We argue that there ought to be a *standard API* for physics modeling, and the EI provides the standard interface as one of the features.

*4) **Novelty**:* To the best of our knowledge, this novel framework uniquely enables research at the intersection of performance, energy/power, thermal/cooling, and reliability at full-system cycle-level multicore simulations. A few related works used simulation environment with power, thermal, and/or reliability models [1], [3], but they fundamentally differ in being trace-driven simulation and integrated with specific models rather than designed for interchangeable physical models (i.e., extensibility). In particular, the related efforts 1) lack modeling of concurrent interactions between multiple physical phenomena and their impact on microarchitecture performance and 2) do not have flexibility of alternative model configurations, but rather provide point solutions to specific models.

*5) **Applicability**:* We have strived to organize the EI to be scalable across multiple distinct types of physical phenomena. This means new libraries (corresponding to new phenomena) can be added, and new models (of existing library types) can be easily added.

*6) **Effort**:* The stable infrastructure and API specifications have been developed. This cross-layer implementation requires comprehensive understanding of underlying processor physics. The standard API of the EI minimizes the efforts to integrate multi-physics modeling with microarchitecture simulation infrastructures.

REFERENCES

[1] A. Bartolini, M. Cacciari, A. Tilli, L. Benini, and M. Gries, “A Virtual Platform Environment for Exploring Power, Thermal, and Reliability Management Control Strategies in High-Performance Multicores,” *GLSVLSI*, May 2010.

[2] M. Cho, K. Ahmed, W. Song, S. Yalamanchili, and S. Mukhopadhyay, “Post-Silicon Characterization and On-Line Prediction of Transient Thermal Field in Integrated Circuits Using Thermal System Identification,” *Trans on. CPMT*, Mar. 2012.

[3] A. Coskun, T. Rosing, K. Mihic, G. Micheli, and Y. Leblebici, “Analysis and Optimization of MPSoC Reliability,” *JOLPE*, Jan. 2006.

[4] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. Stan, “HotSpot: A Compact Thermal Modeling Methodology for Early-Stage VLSI Design,” *Trans. on VLSI*, Nov. 2006.

[5] A. Kahng, B. Li, L. Peh, K. Samadi, “Orion 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration,” *DATE*, Apr. 2009.

[6] C. Kersey, A. Rodrigues, and S. Yalamanchili, “A Universal Parallel Frontend for Execution Driven Microarchitecture Simulation,” *RAPIDO*, Jan. 2012.

[7] H. Kim, J. Lee, N. Lakshminarayana, J. Sim, J. Lim, and T. Pho, “MacSim: A CPU-GPU Heterogeneous Simulation Framework,” User Manual.

[8] S. Li, J. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, “McPAT: Integrated Power, Area, Timing Modeling Framework for Multicore Architectures,” *MICRO*, Dec. 2009.

[9] A. Rodrigues, E. Cooper-Balis, K. Bergman, K. Ferreira, D. Bunde, and K. Hemmert, “Improvements to The Structural Simulation Toolkit,” SIMUTOOLS, Mar. 2012.

[10] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, “DRAMsim2: A Cycle Accurate Memory System Simulator,” *Computer Architecture Letters*, Jan. 2011.

[11] W. Song, S. Mukhopadhyay, A. Rodrigues, and S. Yalamanchili, “Energy Introspector: A Microarchitecture Framework for Integrated Power, Thermal, Reliability Simulations,” User Manual.

[12] W. Song, S. Mukhopadhyay, and S. Yalamanchili, “Energy Introspector: A Parallel, Composable Framework for Integrated Power-Reliability-Thermal Modeling for Multicore Architectures,” *ISPASS* (Short Paper), Mar. 2014.

[13] W. Song, S. Mukhopadhyay, and S. Yalamanchili, “Lifetime Reliability and Accelerated Execution,” *SRC TECHCON*, Sept. 2013.

[14] A. Sridhar, A. Vincenzi, M. Ruggiero, T. Brunschwiler, and D. Atienza, “3D-ICE: Fast Compact Transient Thermal Modeling for 3D ICs with Inter-Tier Liquid Cooling,” *ICCAD*, Nov. 2010.

[15] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, “The Case for Lifetime Reliability-Aware Microprocessors,” *ISCA*, June 2004.

[16] J. Srinivasan, S. Adve, P. Bose, and J. Rivers, “Lifetime Reliability: Toward An Architectural Solution,” *IEEE Micro*, May 2005.

[17] C. Sun, C. Chen, G. Kurian, L. Wei, J. Miller, A. Agarwal, L. Peh, and V. Stojanovic, “DSENT - A Tool Connecting Emerging Photonics with Electronics for Opto-Electronic Networks-on-Chip Modeling,” *NOCS*, May 2012.

[18] S. Thoziyoor, J. Ahn, M. Monchiero, J. Brockman, and N. Jouppi, “A Comprehensive Memory Modeling Tool and Its Application to The Design and Analysis of Future Memory Hierarchies,” *ISCA*, June 2008.

[19] Z. Wan, Y. Kim, Y. Joshi, “Compact Modeling of 3D-Stacked Die Inter-Tier Microfluidic Cooling Under Non-Uniform Heat Flux,” *IMECE*, Nov. 2012.

[20] J. Wang, J. Beu, R. Bheda, T. Conte, Z. Dong, C. Kersey, M. Rasquinha, G. Riley, W. Song, H. Xiao, P. Xu, and S. Yalamanchili, “Manifold: A Parallel Simulation Framework for Multicore Systems,” *ISPASS*, Mar. 2014.

[21] H. Xiao, Z. Wan, S. Yalamanchili, and Y. Joshi, “Leakage Power Characterization and Minimization in 3D Stacked Multi-core Chips with Microfluidic Cooling,” *SemiTherm*, Mar. 2014.