Energy Introspector: A Parallel, Composable Framework for Integrated Power-Reliability-Thermal Modeling for Multicore Architectures

William J. Song, Saibal Mukhopadhyay, and Sudhakar Yalamanchili

School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332
wjhsong@gatech.edu, {saibal.mukhopadhyay, sudha.yalamanchili}@ece.gatech.edu

Sustaining processor performance growth is challenged by physical limitations due to increased power and heat dissipations. Power and thermal management techniques combined with inherent workload dynamics create the spatiotemporal variations of power, temperature, and degradation in processors. As industry moves to smaller feature sizes, the performance will become increasingly dominated by the physics. The challenge is in understanding how the physics is manifested at the microarchitecture level. This requires the modeling and simulation environment that can capture multiple, distinct physical phenomena and their concurrent impact on the microarchitecture.

There already exist various point tools for physical modeling popularly used in the architecture research community, e.g., power, reliability, etc [1]-[5]. Considerable effort has been invested in their development and availability. The principal challenge is to accurately incorporate the interactions between the models of multiple, distinct physical phenomena into the same domain of analysis. The architecture community continues to develop new models or update existing tools along with technology changes. Thus, a modeling and simulation framework must be compatible with various implementations of models and open to the integration of new models as they are developed. There are various engineering challenges to the construction of a composable, scalable, and standardized simulation framework. We introduce the integrated power, reliability, and thermal/cooling modeling framework for multicore architectures, named *Energy Introspector* (EI). We emphasize that the purpose of the EI framework is to provide an infrastructure and *standardized interface* to bridge different models/tools without inter-dependency in software integration while the interactions between the multiple physics (e.g., thermal, reliability) are correctly captured. The framework is easily extensible (e.g., add new models), has well defined interfaces to multicore simulation models [6], and itself can execute in parallel.

Fig. 1 illustrates architecture-level modeling with the EI. Each tool is typically specialized to model a single type of physical property, and different tools have i) different functionality, ii) are designed with different usage models in mind, and iii) span different time scales [1]-[5]. Supporting the features of all these tools in the same framework is challenging. In the EI framework, the models are categorized

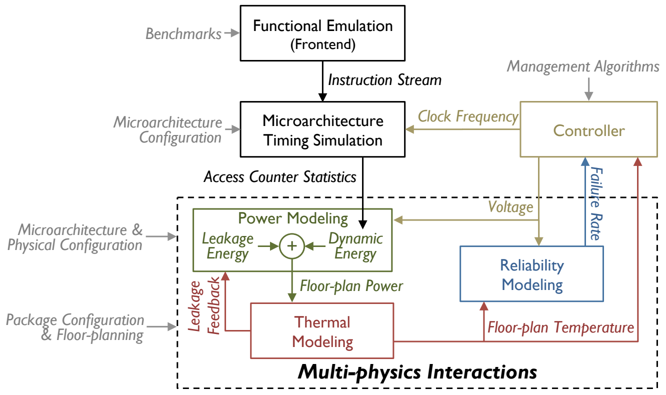
Benchmarks
Functional Emulation (Frontend)
Instruction Stream
Management Algorithms
Microarchitecture Configuration
Microarchitecture Timing Simulation
Clock Frequency
Controller
Access Counter Statistics
Voltage
Failure Rate
Microarchitecture & Physical Configuration
Power Modeling
Leakage Energy
Dynamic Energy
Reliability Modeling
Floor-plan Power
Leakage Feedback
Package Configuration & Floor-planning
Thermal Modeling
Floor-plan Temperature
Multi-physics Interactions

Fig. 1. Architecture-level modeling of interactions between physical phenomena and microarchitecture.

into classes called *libraries* including *energy*, *thermal*, and *reliability libraries*. The energy library includes the models used for power modeling (e.g., Cacti/McPAT [3], Orion/DSENT [2], etc.), and the thermal library includes the package-level temperature/cooling models such as 3D-ICE [4] and HotSpot [1]. The reliability library includes wear-out models [5] such as negative bias temperature instability (NBTI) and time dependent temperature breakdown (TDDB) that are used to calculate transient failure rates of processor components with current operating conditions (e.g., voltage and temperature stresses). For each model integrated into the EI framework, a *wrapper class* is created. The wrapper class is defined as the subclass of one of the library classes. It includes the header/source codes of the tools to be integrated and re-defines the use of the models according to the functions of corresponding library. Therefore, the models of the same library type can be used in an identical way although their implementations are different.

The challenge for a flexible and composable framework is that multiple libraries and their subclass models have to be consistently connected to configure a processor model. In addition, processor descriptions differ between architecture simulators and physical models. Architecture simulators are comprised of functional blocks (e.g., cores, caches, network), while the physical models use circuit or package-level representations of a processor design. Therefore, there has to be a method to establish a correspondence between the elements of processor descriptions and relevant library models. In the EI framework, a processor is represented as a hierarchy tree of *pseudo components*. Pseudo components are virtual units where the integrated library models estimate physical

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

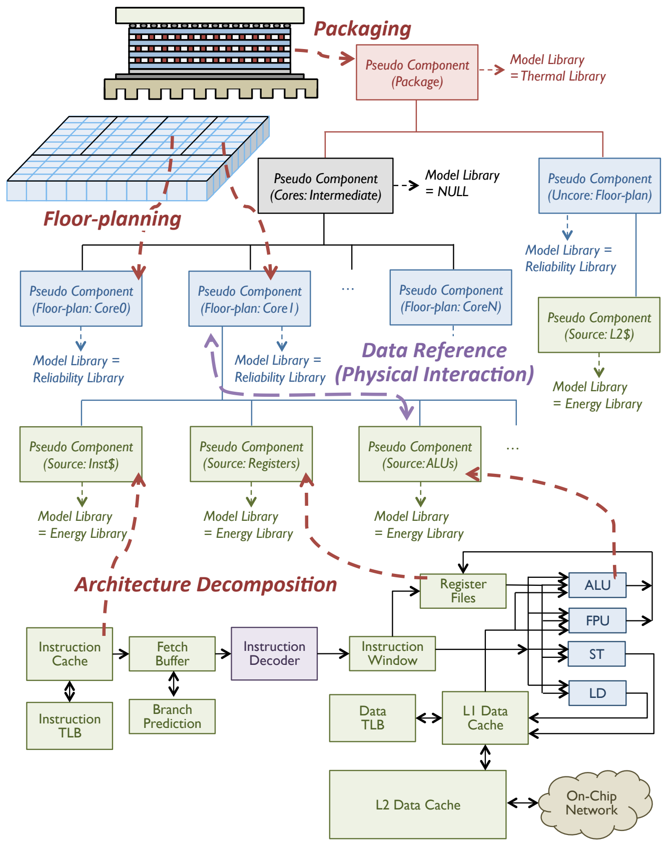
Packaging
Pseudo Component (Package)
Model Library = Thermal Library
Floor-planning
Pseudo Component (Cores: Intermediate)
Model Library = NULL
Pseudo Component (Uncore: Floor-plan)
Model Library = Reliability Library
Pseudo Component (Floor-plan: Core0)
Pseudo Component (Floor-plan: Core1)
Pseudo Component (Floor-plan: CoreN)
Model Library = Reliability Library
Model Library = Reliability Library
Data Reference (Physical Interaction)
Pseudo Component (Source: L2$)
Model Library = Energy Library
Pseudo Component (Source: Inst$)
Pseudo Component (Source: Registers)
Pseudo Component (Source: ALUs)
Model Library = Energy Library
Model Library = Energy Library
Model Library = Energy Library
Architecture Decomposition
Register Files
ALU
FPU
ST
LD
Instruction Cache
Fetch Buffer
Instruction Decoder
Instruction Window
Instruction TLB
Branch Prediction
Data TLB
L1 Data Cache
L2 Data Cache
On-Chip Network

Fig. 2. Energy Introspector framework comprised of pseudo component hierarchy. Physical interactions are emulated by cross-references of data between pseudo components.

phenomena. Different physical properties are characterized at different levels of processor abstraction. For instance, energy (or power) is characterized with architecture/circuit-level components with access (or switching) counts. Temperature is calculated at the package level, and reliability is characterized at the block level. Therefore, pseudo components represent different levels of processor abstraction. The pseudo component hierarchy can be configured for different processor designs. The tree can have as many levels in depth, and each component can have as many sub-components as necessary.

Each pseudo component includes the data queues to store the computed results and shared data (e.g., voltage, frequency). The data queues are used to handle cross-reference and synchronization between the models. For instance, power data calculated by the energy library are stored in the queue, and any libraries (e.g., thermal library) that depend on the power can query the queue to retrieve the data. Since the data are stored in separate structures than the integrated tools, variables and states of the models can be updated by referring to the data queues without erroneous accesses to the computed results of previous computing intervals. Data in the queues are associated with the time information for correct synchronization, and error detection is provided.

Combining the interaction chain of multiple physical phenomena in Fig. 1 and processor description method using the pseudo component hierarchy, the Energy Introspector framework is depicted in Fig. 2. The simulated microarchitecture is decomposed into source components where the linked energy library models estimate power dissipation, and the calculated power values are stored (at each computing interval) in the data queues of these pseudo components. Power data are synchronized along the pseudo component hierarchy by aggregating the power numbers from the leaves toward the root of the tree, and the data queues of pseudo components are updated. Since the data in the queues are tagged with the time information, violations can be detected. After power synchronization is completed, the pseudo components that are designated as the floor-plans of the package-level model have updated power numbers, and the thermal library can safely process the temperature computation. The thermal library model internally converts the floor-plan powers to grid-level power distribution, calculates thermal grid states, and translates the grid-level states to floor-plan temperatures. After the temperature calculation is performed, the EI interface updates the temperatures of floor-plan components. When synchronizing the temperature data, it is assumed that the temperature is uniform within each floor-plan component. If there are multiple sub-components belong to the same floor-plan component, they are updated with the same temperature value. When new data are inserted into the queues, the callback functions of libraries are called to update the dependent variables and states of subclass models. Reliability is characterized at the block level (e.g., floor-plans). It is expressed as a failure rate and dependent on operating conditions such as temperature and voltage stresses. The failure rates can be used to address the relative impacts of concurrent workload executions and microarchitectural operations on the lifetime reliability. This process is repeated at every computing interval. Computed results stored in the data queues of pseudo components are viewable to user simulators, and power/thermal controls such as voltage-frequency scaling can be simply applied by inserting new voltage/frequency values into the data queues, which triggers the callback functions of library models to update the states. The EI framework supports parallel simulations via the MPI interface. In summary, the EI framework has the following features and contributions.

- *Compatible integration* of physical models/tools
- *Coordinated interactions* between multiple, distinct physical models
- *Composable framework* to model and simulate different processor designs
- *User API* for user architecture simulators to simplify the use of models
- *Parallel simulation* using the MPI interface for scalability

References

[1] Huang et al., "HotSpot: A Compat Thermal Modeling Methodology for Early-Stage VLSI Design," *Trans. on VLSI*, Nov. 2006.
[2] Kahng et al., "Orion 2.0: A Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," *DATE*, Apr. 2009.
[3] Li et al., "McPAT: Integrated Power, Area, Timing Modeling Framework for Multicore Architectures," *MICRO*, Dec. 2009.
[4] Sridhar et al., "3D-ICE: Fast Compact Transient Thermal Modeling for 3D ICs with Inter-Tier Liquid Cooling," *ICCAD*, Nov. 2010.
[5] Srinivasan et al., "Lifetime Reliability: Toward An Architectural Solution," *IEEE Micro*, May 2005.
[6] Wang et al., "Manifold: A Parallel Simulation Framework for Multicore Systems," *ISPASS*, Mar. 2014.