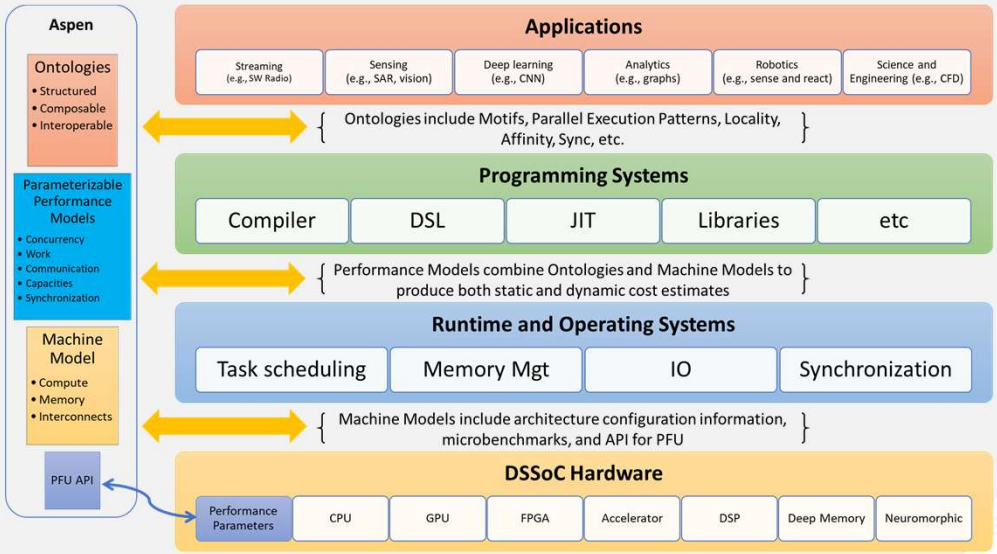
Enabling Co-Design for DSSoC Processors

Cosmic Castle

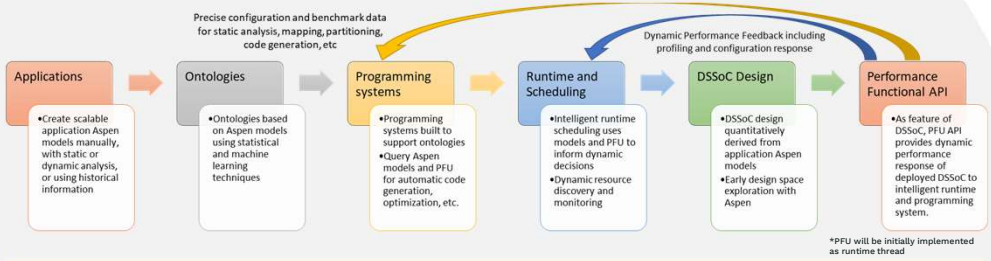
Jeffrey S. Vetter

Oak Ridge National Laboratory

Architectures Thrust: Domain-Specific System on Chip (DSSoC)



Co-Design for DSSoCs



Aspen Performance Models

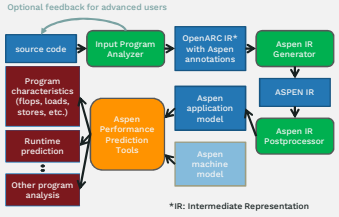
Aspen Modeling Framework: Aspen is a domain-specific language and toolkit for performance modeling:

- Application Models: represent basic resource descriptions and control flow of applications using expressions including user-defined parameters and optional traits to describe semantic information
- Abstract Machine Models: represent basic characteristics of systems, such as multiplicity and hierarchy of each hardware component, as well as how resources map to costs (e.g., time and power).

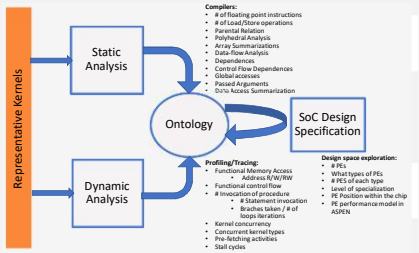
COMPASS: a framework for automated performance model generation and performance prediction, which generates a structured, parametrized Aspen performance model.

- Static analysis: analyzes an input program and generates a parametrized application performance model
- Aspen performance prediction tools: digest the generated Aspen models, synthesize symbolic equations representing application characteristics, and derive various performance predictions

Holistic Modeling for Design, Analysis, and Execution of DSSoCs: the above Aspen frameworks will provide an integrated methodology for design, analysis, and execution of DSSoCs across the architecture, programming and runtime system, and application ontologies.



Quantitative Ontologies



Static Analysis
- Extracts architectural-independent features
- Provides structural knowledge of the analyzed kernels
- Indicates similarity relationships between kernels
- Enables deciding which kind of PEs to be included in the DSSoC design

Dynamic Analysis
- Complements static analysis with pointer analysis and dynamic binding
- Determines how many kernels are executed concurrently and of which kind
- Collects dynamic information (data movement, pre-fetching and stall cycles)
- Enables power consumption measurements and estimation

SoC Design Specification
- Follows a data-driven design space exploration analysis
- Uses PE performance models obtained from Aspen
- Plans to employ robust and well-known numerical optimization techniques
- Refines ontology to add or remove PEs and to modify some of the design parameters

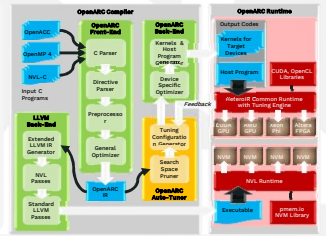
Programming Systems

Goal: Enable de-coupling of programmers from the underlying hardware with enough abstraction while still being able to utilize the underlying hardware optimally.

Proposed Work: The proposed software ecosystem will provide new levels of performance portability and tool-chain interoperability across diverse range of emerging technologies.

- Provide performance portable programming solutions for the heterogeneous DSSoC.
- Provide integrated performance analysis/debugging solutions for these programming systems.
- Provide a common compiler tool-chain infrastructure for the above solutions.

The proposed solutions are based on 1) the **LLVM** production compiler infrastructure, 2) **ARES HLIR**, a high-level compiler intermediate representation, 3) **OpenARC**, a research compiler for rapid prototyping of directive-based language extensions, and 4) a portable profiling and tracing toolkit for performance analysis of parallel programs.



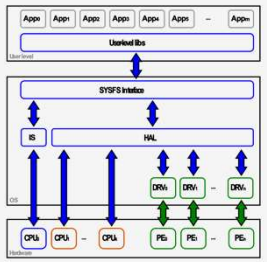
Runtime and Operating Systems

The goal of the **Intelligent Scheduler** (IS) is to find the optimal mapping between tasks and the available PEs. The IS receives inputs from:
- Ontology
- Application and kernel performance models
- Introspective information collected by the Performance Functional Unit (PFU) thread

IS algorithm: The IS makes informed decisions about which task to run on the available PEs. The process is divided into **Matching** (based on ontology and performance models) and **Scheduling** (runtime scheduling algorithm and dynamic resource allocation). Our solution will provide a scheduling framework into which scheduling algorithms can bel plugged in and out.

PE Hardware Abstraction Layer (HAL): The HAL provides a generic PE abstraction to applications and runtime software and hide the low-level architecture details of each PE.

Introspective Layer (IL): The IL collects accurate performance and power metrics from each PE while executing tasks (Performance Functional Unit thread).



Innovation and Impact

Innovative Claims

We propose a multi-level, vertically-integrated strategy to design DSSoCs, based on an extension to a cross-cutting Co-Design methodology (Aspen). Aspen allows:

- ❑ Architecture-independent application models
- ❑ Quantitative ontologies for specific domains
- ❑ Define PEs and their placement in the DSSoCs
- ❑ Performance models to inform configurable compiler infrastructure and intelligent runtime scheduling
- ❑ Fast design space exploration of DSSoCs
- ❑ Design and implement novel embedded Performance API for dynamic configuration and performance feedback

Impact

ORNL will provide fundamental methodologies for Co-Design that will enable architects and customers to design efficient and high performing DSSoCs without incurring unreasonable ASIC production or software cost.

- ❑ Open source software
- ❑ Distribute results and tools to ERI stakeholders
- ❑ Pushing technologies to open standards and vendors
- ❑ Organize cooperative workshops and 1-to-1 interactions

OAK RIDGE National Laboratory

Project URL: https://ft.ornl.gov/dssoc

PI: Jeffrey S. Vetter: vetter@computer.org

This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.
Distribution Statement A – Approved for Public Release, Distribution Unlimited

THE ELECTRONICS RESURGENCE INITIATIVE