PyMTL: A Python-Based Framework for Hardware Modeling

Shunning Jiang and Shuang Chen

Computer Systems Laboratory
School of Electrical and Computer Engineering
Cornell University
Computer Architecture Development Methodologies

Applications

Functional-Level Modeling
  – Behavior

Functional-Level Modeling
  – Behavior
  – Cycle-Approximate
  – Analytical Area, Energy, Timing

Cycle-Level Modeling
  – Behavior
  – Cycle-Approximate
  – Analytical Area, Energy, Timing

Register-Transfer-Level Modeling
  – Behavior
  – Cycle-Accurate Timing
  – Gate-Level Area, Energy, Timing

Transistors

Algorithms

Compilers

Instruction Set Architecture

Microarchitecture

VLSI
**Computer Architecture Development Methodologies**

- **Computer Architecture Methodology Gap**
  FL, CL, RTL modeling use very different languages, patterns, tools, and methodologies

- **PyMTL Approach: Modeling Towards Layout**
  Unified Python-based framework for FL, CL, and RTL modeling

- **Functional-Level Modeling**
  - Algorithm/ISA Development
  - MATLAB/Python, C++ ISA Sim

- **Cycle-Level Modeling**
  - Design-Space Exploration
  - C++ Simulation Framework
  - SW-Focused Object-Oriented
  - gem5, SESC, McPAT

- **Register-Transfer-Level Modeling**
  - Prototyping & AET Validation
  - Verilog, VHDL Languages
  - HW-Focused Concurrent Structural
  - EDA Toolflow
Why Python?

- Python is well regarded as a highly productive language with lightweight, pseudocode-like syntax
- Python supports modern language features to enable rapid, agile development (dynamic typing, reflection, metaprogramming)
- Python has a large and active developer and support community
- Python includes extensive standard and third-party libraries
- Python enables embedded domain-specific languages
- Python facilitates engaging application-level researchers
- Python includes built-in support for integrating with C/C++
- Python performance is improving with advanced JIT compilation
What is PyMTL?

- A Python DSEL for concurrent-structural hardware modeling
- A Python API for analyzing models described in the PyMTL DSEL
- A Python tool for simulating PyMTL FL, CL, and RTL models
- A Python tool for translating PyMTL RTL models into Verilog
- A Python testing framework for model validation
What Does PyMTL Enable?

- Incremental refinement from algorithm to accelerator implementation
- Automated testing and integration of PyMTL-generated Verilog
What Does PyMTL Enable?

- Incremental refinement from algorithm to accelerator implementation
- Automated testing and integration of PyMTL-generated Verilog
- Multi-level co-simulation of FL, CL, and RTL models
What Does PyMTL Enable?

- Incremental refinement from algorithm to accelerator implementation
- Automated testing and integration of PyMTL-generated Verilog
- Multi-level co-simulation of FL, CL, and RTL models
- Construction of highly-parameterized RTL chip generators
The PyMTL Toolflow

**Specification**
- PyMTL Test Harness
- PyMTL FL Model
- PyMTL RTL Model
- SystemVerilog Model

**Tools**
- Elaboration
- Simulation Tool
- Translation Tool

**Output**
- Traces & VCD
- Verilog

PyMTL: A Python-Based Framework for Hardware Modeling
Using PyMTL RTL for ECE 4750

**Specification**
- PyMTL Test Harness
- PyMTL FL Model
- PyMTL RTL Model

**Tools**
- Elaboration
- Simulation Tool
- Translation Tool

**Output**
- Traces & VCD
- Verilog

**Tools**
- SystemVerilog Model
- PyMTL Wrapper
- Verilog Import Tool
Using Verilog RTL for ECE 4750

**Specification**
- PyMTL Test Harness
- PyMTL FL Model
- PyMTL RTL Model

**Tools**
- Elaboration
- Simulation Tool
- Translation Tool

**Output**
- Traces & VCD
- Verilog

PyMTL: A Python-Based Framework for Hardware Modeling