An Open-Source Python-Based Hardware Generation, Simulation, and Verification Framework

We present an overview of previously published features and work in progress for PyMTL, an open-source Python-based hardware generation, simulation, and verification framework that brings computing productivity benefits to hardware design. PyMTL provides a natural environment for multi-level modeling using method-based interfaces, features highly parametrized static elaboration and analysis/transform passes, supports fast simulation and property-based random testing in pure Python environment, and includes seamless SystemVerilog integration.

PyMTL’s Eight Features

- **Multi-level modeling**
  - Easier to implement using a Python-like syntax
  - Portable across platforms

- **Method-based interfaces**
  - PyMTL-based simulation
  - Python-based testing

- **Python/SystemVerilog integration**
  - Flow from PyMTL to SystemVerilog
  - Support for SystemVerilog simulations

- **Pure-Python simulation**
  - Fully integrated Python simulation
  - Easy to integrate with other tools

- **Property-based random testing**
  - Scalable and repeatable testing
  - Cost-effective testing

- **Highly Parametrized Static Elaboration**
  - Flexible parameterization
  - Customizable elaboration

PyMTL Use Cases

- **Course Lab Assignments**
- **Computer Architecture Research**

PyMTL has been used by over 400 students across two universities, including a graduate-level undergraduate computer architecture course at Cornell (ECE 4750), in a similar course at Boston University (EC 513), and in a graduate-level ASCI design course at Cornell University (ECE 5345). Students were required to use PyMTL in SystemVerilog, but some students chose to use Python instead. They were strongly encouraged to use PyMTL, but they were required to test their designs using PyMTL.

PyMTL has been used in courses at Cornell University and Boston University. It is an open-source Python-based framework that brings computing productivity benefits to hardware design. PyMTL provides a natural environment for multi-level modeling using method-based interfaces, features highly parametrized static elaboration and analysis/transform passes, supports fast simulation and property-based random testing in pure Python environment, and includes seamless SystemVerilog integration.

**PyMTL’s Use Cases**

### Computer Architecture Research

**Cosimulate PyMTL cycle-accurate accelerator model with gem5 C++/memory**

- **Shunning Jiang, Christopher Torng, Christopher Batten**
- **Ji Kim, Shreesha Srinath, Berkin Ilbeyi, Khalid Al-Hawaj, and Christopher Batten.**

**An Architectural Framework for Accelerating Dynamic Parallel Algorithms on Reconfigurable Hardware.**

- **Xcel CL**
- **Xcel FL**
- **Xcel RTL**
- **Mem CL**
- **Mem FL**
- **Mem RTL**
- **VLIW FL**
- **VLIW RTL**

**PyMTL on Microarchitecture (MICRO-51), Oct. 2018.**

**Create architecture and code generation on accelerator with PyMTL/RTL modeling:**

- **Shunsuke, Shunsuke, Christopher Batten, and Gyesik Bae.**

**An Architectural Framework for Accelerating Dynamic Parallel Algorithms on Reconfigurable Hardware.**

- **Xcel CL**
- **Xcel FL**
- **Xcel RTL**
- **Mem CL**
- **Mem FL**
- **Mem RTL**
- **VLIW FL**
- **VLIW RTL**

**PyMTL on Microarchitecture (MICRO-51), Oct. 2018.**

**Create architecture and code generation on accelerator with PyMTL/RTL modeling:**

- **Shunsuke, Shunsuke, Christopher Batten, and Gyesik Bae.**

**An Architectural Framework for Accelerating Dynamic Parallel Algorithms on Reconfigurable Hardware.**

- **Xcel CL**
- **Xcel FL**
- **Xcel RTL**
- **Mem CL**
- **Mem FL**
- **Mem RTL**
- **VLIW FL**
- **VLIW RTL**

**PyMTL on Microarchitecture (MICRO-51), Oct. 2018.**