Experiences Using a Novel Python-Based Hardware Modeling Framework for Computer Architecture Test Chips

Christopher Torng, Moyang Wang, Bharath Sudheendra, Nagaraj Murali, Suren Jayasuriya, Shreesha Srinath, Taylor Pritchard, Robin Ying, and Christopher Batten

School of Electrical and Computer Engineering
Cornell University

1 Abstract

This poster will describe a taped-out 2×2 mm 1.3 M-transistor test chip in IBM 130 nm designed using our new Python-based hardware modeling framework. The goal of our tapeout was to demonstrate the ability of this framework to **enable Agile hardware design flows**.

Specifically, our approach has two pieces:

- **Unify** all simulation (behavioral, cycle-level timing, RTL, and gate-level) within a single-language development framework
- **Integrate** this framework with highly automated standard-cell design flows

For small teams working on small computer architecture test chips for research or as part of an Agile hardware design flow, such an approach can enable rapid design iteration from RTL to layout, shortening the time to tapeout despite limited manpower.

2 The PyMTL Hardware Modeling Framework

Computer architects have long traded off simulation time and accuracy by leveraging multiple modeling abstractions including **functional-level (FL)**, **cycle-level (CL)**, and **register-transfer-level (RTL)** modeling. However, each of these uses distinct modeling languages, modeling patterns, and modeling tools, creating a large impediment to vertically integrated, iterative refinement of a design from algorithm, to exploration, to implementation.

	FL	CL	RTL
Modeling Languages	Productivity Level (PLL)	Efficiency Level (ELL)	Hardware Description (HDL)
	MATLAB/Python	*C/C++*	*Verilog/VHDL*
Modeling Patterns	Functional: Data Structures, Algorithms	Object Oriented: Classes, Methods, Ticks and/or Events	Concurrent-Structural: Combinational Logic, Clocked Logic, Port Interfaces
Modeling Tools	Third-party Algorithm Packages and Toolboxes	Computer Architecture Simulation Frameworks	Simulator Generators, Synthesis Tools, Verification Tools

PyMTL is a unified and highly productive framework for FL, CL, and RTL modeling based on a common high-productivity language (`Python2.7`).

What is PyMTL?

PyMTL Hardware Modeling Framework [*]

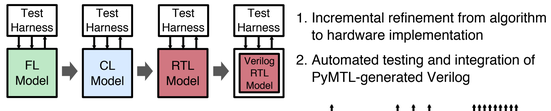
[*] D. Lockhart, G. Zibrat, and C. Batten. PyMTL: A Unified Framework for Vertically Integrated Computer Architecture Research. Int'l Symp. on Microarchitecture, Dec 2014.

Functional-Level Modeling (FL)
- Behavior

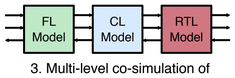
Cycle-Level Modeling (CL)
- Behavior
- Cycle-Approximate
- Analytical Area, Energy, Timing

Register-Transfer-Level Modeling (RTL)
- Behavior
- Cycle-Accurate Timing
- Gate-Level Area, Energy, Timing

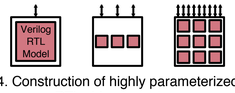
What Does PyMTL Enable?



1. Incremental refinement from algorithm to hardware implementation
2. Automated testing and integration of PyMTL-generated Verilog



3. Multi-level co-simulation of FL, CL, and RTL models



4. Construction of highly parameterized RTL chip generators

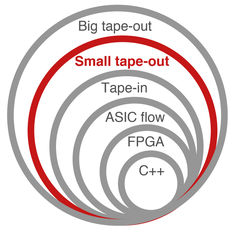
3 PyMTL for Computer Architecture Test Chips

Why Build Computer Architecture Test Chips?

There are many reasons to build computer architecture test chips in the contexts of both academia and industry.

Key Aspect in Agile Hardware Design

- Rapid design iteration
- Reduces cost of validation
- "Building the right thing"



* Adapted from Yunsup Lee IEEE Micro 2016

Benefits Research

- Builds research credibility
- Highly reliable power and energy estimates for new architecture techniques

Design Methodologies: Large Chips vs. Small Chips

Large-scale commercial chips and small computer architecture test chips have different design methodologies with different production targets.

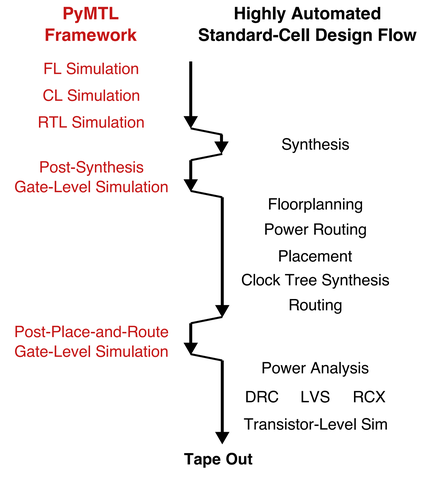
Large-Scale Commercial Chips
- High-volume and high-yield production
- Overcome design challenges with large teams

Computer Architecture Test Chips
- Low-volume and reasonable-yield production
- Overcome design challenges despite small teams and limited resources

Small teams that build computer architecture test chips with **highly productive development frameworks** can shorten the time to tapeout.

PyMTL in Agile Hardware Design

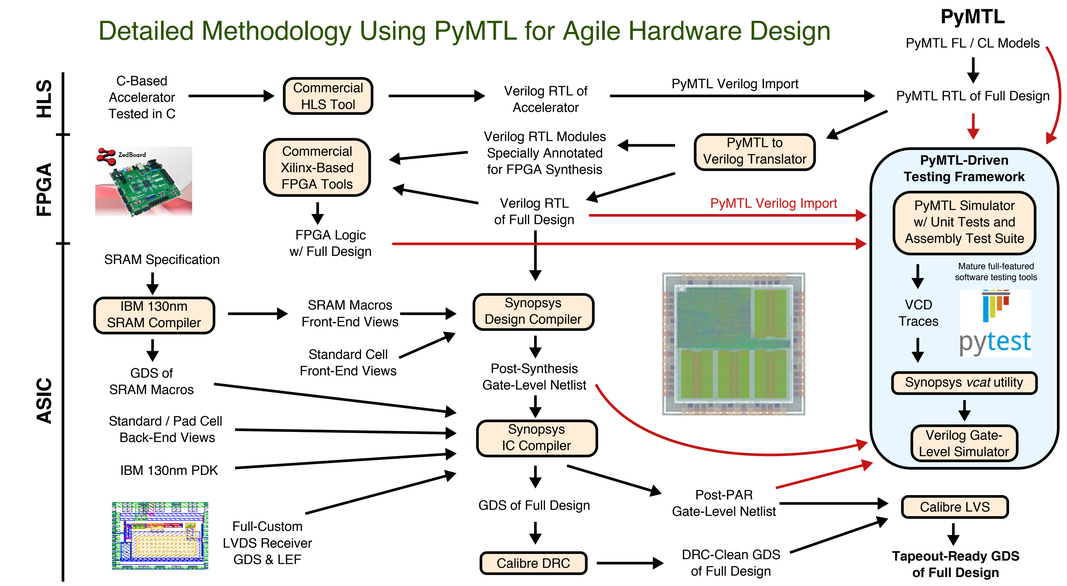
One such framework can be created by combining the PyMTL framework with a highly automated standard-cell design flow, potentially enabling small teams to push RTL changes to layout with a validated gate-level netlist within a day.



PyMTL provides the **unified design and testing environment** for not only FL, CL, and RTL models but also for post-synthesis and post-PAR gate-level models. Combining with a **highly automated standard-cell design flow** allows small teams to meet low-volume and reasonable-yield production targets while shortening the time to tapeout.

4 PyMTL-Integrated Tapeout Methodology

Detailed Methodology Using PyMTL for Agile Hardware Design



5 PyMTL in Practice: BRG Test Chip 1

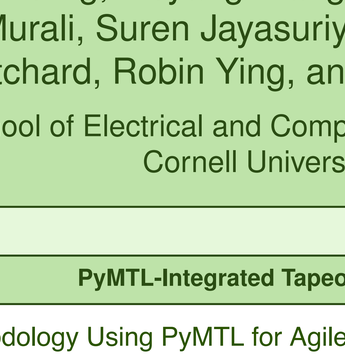
BRG Test Chip 1 (BRGTC1) was designed using the PyMTL hardware modeling framework.

Our methodology uses PyMTL for:

- Design (FL/CL/RTL)
- Composition with a commercial high-level synthesis (HLS) toolflow
- As a unified testing environment

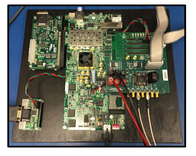
The testing environment drives:

- FL/CL/RTL simulation
- FPGA emulation
- Post-synthesis/PAR gate-level simulation



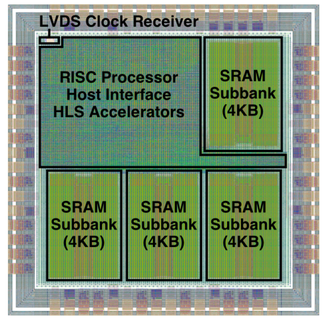
Testing Plans After Fabrication

- Xilinx ZC706 FPGA development board for FPGA prototyping
- Custom-designed FMC mezzanine card for ASIC test chips



The testing platform enables running test programs on BRGTC1 to compare the performance/energy of pure-software kernels versus the HLS-generated sorting accelerator.

Chip Plot



Taped Out: March 2016 **Expected Return**: Fall 2016

A 2×2 mm 1.3 M-transistor RISC processor in IBM 130 nm, 16 KB SRAM, and HLS-generated accelerators.
Static Timing Analysis Freq. @ 246 MHz

6 Acknowledgments

Chip fabrication was made possible by the MOSIS Educational Program. This work was supported in part by NSF CAREER Award #1149464, NSF XPS Award #1337240, NSF CRI Award #1512937, and equipment/tool/IP donations from Intel, Synopsys, Cadence, Mentor Graphics, Xilinx, and ARM. We acknowledge and thank Mark Buckler for EDA toolflow development and Ivan Bukreyev for advice on full-custom design.

Publication: Student Poster at the 28th Symposium on High Performance Chips (HotChips-28), Aug. 2016. **URL:** `http://www.csl.cornell.edu/~cbatten/pdfs/torng-brgtc1-hotchips2016.pdf` **Contact Author:** Christopher Torng, , `clt67@cornell.edu`